МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет	*Кафедра*
вычислительной техники	МОиПЭВМ

Направление подготовки 09.03.04 «Программная инженерия»

Направленность Программная инженерия

БАКАЛАВРСКАЯ РАБОТА

на тему

БИЗНЕС-ПРОЕКТ «REM»: КЛИЕНТ-СЕРВЕРНОЕ ПРИЛОЖЕНИЕ ДЛЯ ИЗУЧЕНИЯ ИНОСТРАННОЙ ЛЕКСИКИ, ИСПОЛЬЗУЮЩЕЕ КОМБИНАЦИЮ РАЗЛИЧНЫХ СПОСОБОВ ПРЕДСТАВЛЕНИЯ ИЗУЧАЕМОГО МАТЕРИАЛА

Студент _____ Дьячков Даниил Александрович
(подпись, дата) (ФИО полностью)

Руководитель _____ Такташкин Д.В.
(подпись, дата) (фамилия, инициалы)

Нормоконтролёр _____ Такташкин Д.В.
(подпись, дата) (фамилия, инициалы)

Работа допущена к защите (протокол заседания кафедры от __._____№_____)

Заведующий кафедрой _____ Козлов А.Ю.
(подпись) (фамилия, инициалы)

Работа защищена с отметкой ____(протокол заседания ГЭК от ___№_____)

Секретарь ГЭК _____ Попова Н.А.
(подпись) (фамилия, инициалы)

Пенза, 2024

Экспертное заключение
о согласовании темы ВКР в форме стартапа

Экспертная комиссия в составе: декана ФВТ д.т.н., профессора Л.Р. Фионовой, доцента кафедры ВТ, к.э.н., И.В. Акифьева, главного разработчика ООО «Ай-теко ВЦ», к.т.н. М.П. Синева; рассмотрев стартап-проект «Rem» Дьячкова Даниила Александровича подтверждает его актуальность, практическую значимость и возможность коммерциализации.

Решение экспертной комиссии: выпускная квалификационная работа на тему «Бизнес-проект «Rem»: клиент-серверное приложение для изучения иностранной лексики, использующее комбинацию различных способов представления изучаемого материала» Дьячкова Даниила Александровича может выполняться в форме стартапа (бизнес-проекта).

Председатель ЭК

д.т.н., профессор
декан ФВТ Л.Р. Фионова

Члены ЭК

к.э.н., доцент кафедры ВТ И.В. Акифьев

к.т.н., главный разработчик
ООО «Ай-теко ВЦ» М.П. Синев

Федеральное государственное бюджетное образовательное учреждение высшего образования
ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет	*Кафедра*
Вычислительной техники	МО и ПЭВМ

«Утверждаю»
Заведующий кафедрой
_____А.Ю. Козлов
«25» декабря 2023 г

ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ В ФОРМЕ СТАРТАПА

1. Студент Дьячков Даниил Александрович гр. 20ВП1 факультета ВТ
2. Тема проекта: Бизнес-проект «Rem»: клиент-серверное приложение для изучения иностранной лексики, использующее комбинацию различных способов представления изучаемого материала

Тема утверждена приказом ПГУ № _____ от «___»_____20__ г.

3. Руководитель работы: Такташкин Д.В.
4. Техническое задание на квалификационную работу в форме стартапа (назначение, технические требования)

Назначение: информационная система решает задачу автоматизации заучивания иностранных слов

Информационная система состоит из серверной и клиентской частей, взаимодействующих друг с другом через интернет-соединение

Основные функции информационной системы:

а) Представление обучающего материала в различных формах

б) Интерактивное взаимодействие с пользователем в процессе обучения

Структуры данных: деревья, множества, очереди

Технологии разработки: ООП, ФП

Языки программирования: Dart, Python

Среда исполнения: Docker-образ, Android OS

5. Перечень вопросов, подлежащих разработке

1) *Анализ предметной области*
2) *Анализ требований к информационной системе*
3) *Проектирование*
4) *План разработки и оценки бюджета*
5) *Разработка*
6) *Контроль качества программного кода*

6. Календарный график по выполнению работы

Наименование этапов разработки	Объем работы	Сроки выполнения	Подпись руководителя, консультанта
1)Бизнес-план	5%	08.04.2024 – 10.04.2024	
2)Анализ предметной области и требований к системе	5%	11.04.2024 – 13.04.2024	
3)Проектирование информационной системы «Rem»	15%	15.04.2024 – 25.04.2024	
4)Разработка информационной системы «Rem»	60%	26.04.2024 – 12.05.2024	
5)Контроль качества программного кода	5%	13.05.2024 – 22.05.2024	
6)Оформление пояснительной записки	10%	23.05.2024 – 03.06.2024	

Дата выдачи задания «21» декабря 2023 г.
Руководитель бакалаврской работы _____ Такташкин Д.В.
(подпись, дата)

Задание к исполнению принял студент _____ Дьячков Д.А.
(подпись, дата)

Работу к защите допустить
Декан ФВТ _____ Фионова Л.Р.
(подпись, дата)

Реферат

Пояснительная записка содержит 298 листов, 27 рисунков, 14 таблиц, один листинг, 34 использованных источника, 11 приложений.

БАЗА ДАННЫХ, ИНФОРМАЦИОННАЯ СИСТЕМА, СУБД, ПРЕДМЕТНАЯ ОБЛАСТЬ, PYTHON, POSTGRESQL, МИКРОСЕРВИСЫ, ПАТТЕРНЫ РАЗРАБОТКИ, АРХИТЕКТУРА ИНФОРМАЦИОННОЙ СИСТЕМЫ, PYTON, DART, FLUTTER, КРИВАЯ ЭББИНГАУЗА, ИНОСТРАННЫЕ ЯЗЫКИ.

В данной ВКРС объектом разработки выступает информационная система, предназначенная для изучения иностранной лексики и использующее комбинацию различных способов представления изучаемой информации.

Целью ВКРС является повышение эффективности образовательного процесса изучения иностранных языков посредством разработки информационной системы.

В результате ВКРС был разработан продукт на стадии MVP, техническая новизна которого заключается в разработке и применении подхода гибких архитектурных квантов в микросервисной архитектуре, основанного на достижении компромисса между дублированием и переиспользованием кода.

Разработанное мобильное приложение требует 100 мб долговременной памяти и версию ОС Android не ниже восьмой.

Разработанный продукт может использоваться в сфере дополнительного образования языковыми школами для осуществления дистанционного взаимодействия с обучающимися.

Содержание

Введение ... 8
1 Бизнес-проект «Rem» ... 10
1.1 Анализ рынка ... 10
1.2 Описание бизнес-идеи ... 12
1.3 Постановка цели и задач бизнес-проекта ... 13
1.4 Описание продукта ... 13
1.5 Финансовая поддержка ... 15
1.6 Маркетинговый план ... 15
1.7 Финансовый план ... 17
2 Цифровая платформа для изучения иностранной лексики «Rem» ... 21
2.1 Анализ требований к проекту «Rem» ... 21
2.1.1 Анализ цифровых подходов к изучению иностранных языков ... 21
2.1.2 Обзор аналогов информационной системы «Rem» ... 22
2.1.3 Анализ требований к информационной системе «Rem» ... 26
2.1.3.1 Анализ функциональных требований ... 26
2.1.3.2 Анализ нефункциональных требований ... 28
2.1.4 Постановка целей и задач ... 32
2.2 Проектирование информационной системы «Rem» ... 33
2.2.1 Проектирование архитектуры ... 33
2.2.1.1 Проектирование архитектуры сервера ... 33
2.2.1.2 Проектирование архитектуры приложения ... 44
2.2.2 Проектирование базы данных ... 47
2.2.2.1 Проектирование базы данных сервера ... 48
2.2.2.2 Проектирование базы данных приложения ... 49
2.2.3 Проектирование интерфейса ... 50
2.2.4 Оценка бюджета на разработку проекта «Rem» ... 51
2.3 Реализация информационной системы «Rem» ... 56

2.3.1 Реализация архитектуры информационной системы «Rem» 56
2.3.1.1 Реализация клиента 56
2.3.1.2 Реализация сервера 58
2.3.2 Реализация базы данных информационной системы «Rem» 64
2.3.2.1 Реализация серверной базы данных 64
2.3.2.2 Реализация базы данных мобильного приложения 65
2.3.3 Реализация пользовательского интерфейса проекта «Rem» 65
2.3.4 Тестирование информационной системы «Rem» 67
2.3.4.1 Функциональное тестирование серверной части 67
2.3.4.2 Модульное тестирование серверной части 73
2.3.4.3 Метрики кода информационной системы «Rem» 75
Заключение 78
Список использованных источников 80
Приложение А (рекомендуемое) Диплом грантополучателя 84
Приложение Б (рекомендуемое) Сертификат финалиста преакселератора ПГУ 86
Приложение В (рекомендуемое) Письма поддержки 88
Приложение Г (рекомендуемое) Спецификации вариантов использования 94
Приложение Д (рекомендуемое) Список задач диаграммы Ганта 98
Приложение Е (рекомендуемое) Проектирование пользовательского интерфейса 100
Приложение Ж (рекомендуемое) Реализация пользовательского интерфейса 108
Приложение И (рекомендуемое) Результаты функционального тестирования 118
Приложение К (рекомендуемое) Код модульных тестов 151
Приложение Л (рекомендуемое) Код клиентской части 232

Введение

На сегодняшний день языковой сегмент рынка цифровых образовательных технологий активно развивается как в России [1], так и во всём мире.

Перед компаниями, ведущими свою деятельность в сфере обучения иностранным языкам встают проблемы не только экономического, но и научного характера: каким образом сделать изучение иностранного языка более эффективным?

Ответить на этот вопрос должно помочь знание устройства механизмов памяти, которые использует наш мозг для изучения новой информации.

Одним из первых значимых исследований в этой области, остающимся актуальным и сегодня, был эксперимент Г. Эббингауза по запоминанию обессмысленных слогов, проведённый во второй половине 19 века. Г. Эббингауз исследовал механизмы безассоциативной памяти.

Исследования Г. Эббингауза очень важны, однако их результаты проблематично использовать в чистом виде для изучения иностранных языков, ведь человеческая речь, в отличии от материала, изучаемого Г. Эббингаузом, имеет смысловую нагрузку, которую необходимо учитывать при обучении человека иностранному языку.

Со временем человечество открыло множество методик изучения иностранных языков (таблица 1).

Однако большинство наиболее эффективных методик полагаются в основном на ассоциативную память.

Такие методики ставят в абсолют смысловую нагрузку и в большинстве своём не учитывают специфику механизмов работы человеческой памяти.

Однако у подавляющего большинства приведённых ниже методов (смотри таблицу 1) присутствует ещё один недостаток – низкая способность к цифровизации.

Таблица 1 – Методы обучения иностранным языкам

Прямой подход	Сознательный подход	Деятельностный подход	
Прямые методы (беспереводные)	Сознательные методы (переводные)	Комбинированные методы (смешанные)	Интенсивные методы
натуральный	грамматико- переводный	чтения	суггестопедический
прямой	сознательно-сопоставительный	коммуникативный	метод Г. Лозанова
аудио-визуальный	текстуально- переводный	тандем-метод	эмоционально-
аудио-лингвальный		сознательно- практический	смысловой
армейский			активизации
устный			гипнопедия
«метод гувернантки»			ритмопедия
			гувернантки»
			«экспресс-метод»
			психотерапевтический

Разработчики цифровых решений в области изучения иностранных языков, принимая во внимание этот недостаток, используют наиболее доступную форму методов изучения иностранных языков – геймификацию.

1 Бизнес-проект «Rem»

1.1 Анализ рынка

Согласно данным аналитической компании Smart Ranking, рост рынка цифрового образования (EdTech) на конец 2023 года составил 32%. Объем рынка на конец 2023 года оценивается в 35 миллиарда рублей, при этом выручка крупнейших 150 игроков составила 123 миллиарда рублей. Более половины выручки пришлось на 10 крупнейших игроков рынка [1]. Рост мирового рынка EdTech показан на рисунке 1.

Рисунок 1 – Динамика роста мирового рынка EdTech

Доля суммарной выручки участников рейтинга языкового сегмента рынка EdTech составила шесть процентов, а рост за 2023 год составил 14%. Бесспорным лидером языкового сегмента является Skyeng [2], выручка которого составила в процентном соотношении 90% от всей выручки языкового сегмента [1]. Выручка крупнейших игроков языкового сегмента рынка представлена на рисунке 2.

Также стоит отметить, что участники рынка EdTech выявили повышение спроса на расширение функционала, который делает системы обучения полностью автономными [3].

Помимо восьми языковых школ, вошедших в рейтинг агентства Smart Ranking, существует множество мелких компаний (в том числе самозанятых и

индивидуальных предпринимателей, занимающихся репетиторством), не вошедших в аналитический отчет, которые также присутствуют на рынке дополнительного образования в языковом сегменте. Несмотря на то, что доля каждой из таких компаний по отдельности мала, совокупная величина может составить существенную часть капитализации рынка.

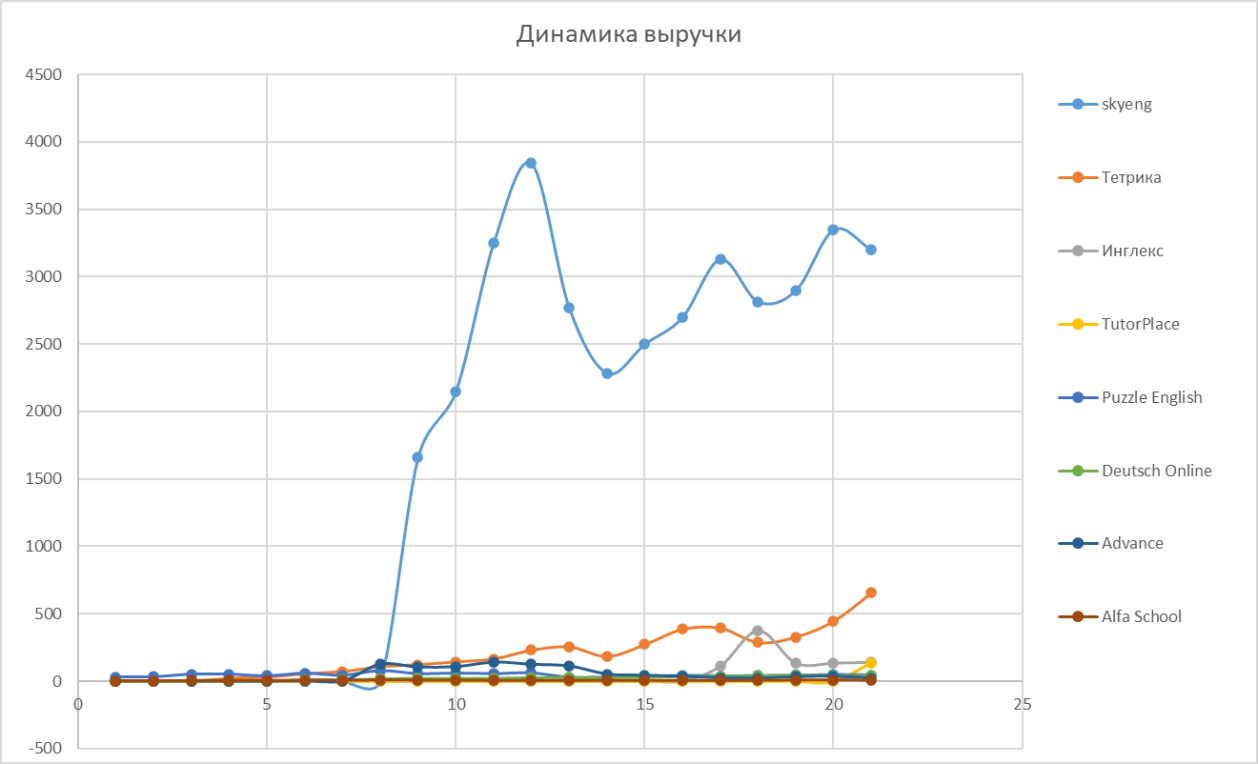


Рисунок 2 – Выручка крупнейших игроков языкового сегмента рынка

По оценке аналитической компании Smart Ranking, одними из решающих факторов роста российского рынка EdTech являются:

– уход с рынка зарубежных онлайн-площадок;

– рост заинтересованности государства в развитие российского онлайн-образования, появление программ госфинансирования;

– рост доверия потребителей к онлайн-образованию, начавшийся в период пандемии COVID-19.

В связи с этим, у локальных языковых школ появилась возможность увеличить капитализацию и охват целевой аудитории, однако для достижения этого необходимы качественные цифровые инструменты взаимодействия с обучающимися.

Небольшие языковые школы можно условно разделить на несколько групп по степени цифровизации бизнес-процессов и технологичности взаимодействия с обучающимися:

а) информационные технологии используются только для обозначения места и времени офлайн-занятий или не используются вовсе;

б) информационные технологии представляют из себя сторонние ресурсы, не предоставляющие возможности персонализации;

в) информационные технологии используются для проведения онлайн занятий в формате видео-звонков или видеоконференций;

г) информационные технологии интегрированы в процесс выполнения или учета заданий и специализированы для цифрового образования;

д) информационная система позволяет администрировать нагрузку преподавателей и обучающихся, интегрирована в систему менеджмента сотрудников компании;

е) информационная система интегрирована в совершение денежных операций.

1.2 Описание бизнес-идеи

Небольшие языковые школы зачастую не имеют возможности для цифровизации своих бизнес-процессов до уровня, на котором информационные технологии специализированы для дистанционного обучения иностранному языку и выше, потому что для этого требуется разработка собственной информационной системы.

Для небольших языковых школ существует альтернативное решение: использование Learning Management System (LMS – системы управления электронным обучением) [4]. Однако минусом LMS является отсутствие узкоспециализированных возможностей, специфичных для сферы изучения языков.

Цифровизация изучения иностранных языков требует применения подходов, отличных от тех, что применяются в онлайн образовании в других сферах.

Разработка новых цифровых решений достаточного качества несет в себе существенные затраты и не по карману большинству языковых школ. Готовые решения, такие как LMS, недостаточно функциональны для качественного взаимодействия языковых школ с обучающимися онлайн.

Для решения проблемы таких небольших компаний, занимающих языковой сегмент рынка образовательных технологий (потенциальных клиентов) предлагается:

– разработать собственную информационную систему для организации процесса дистанционного обучения иностранным языкам;

– продавать целевой аудитории (ЦА) продукта разработанную информационную систему как коробочное решение.

1.3 Постановка цели и задач бизнес-проекта

Целью данного бизнес-проекта является разработка бизнес-плана и получение финансирования на первый этап реализациии.

Для достижения поставленной цели бизнес-проекта необходимо решить следующие задачи:

– провести анализ рынка и предполагаемой экономической ниши;

– описать бизнес-идею и разрабатываемый продукт;

– составить маркетинговый и финансовый планы;

– получить финансирование на реализацию первого этапа проекта.

1.4 Описание продукта

В рамках данной выпускной квалификационной работы в форме стартапа (ВКРС) [5] предлагается разработать стартап-проект «Rem» – информационную систему для изучения иностранных языков.

Результатом ВКРС будет информационная система на стадии минимально-жизнеспособного продукта (MVP).

Ценностным предложением для целевой аудитории проекта является помощь языковым онлайн-школам и репетиторам в повышении эффективности дистанционного обучения за счет автоматизации:

– сбора статистических данных по результатам освоения материала;

– проверки домашних заданий;

– автоматического составления подборок слов для изучения и т.д.

Что ведет:

– к повышению лояльности клиентов;

– к снижению нагрузки на преподавателя;

– к увеличению прибыли.

Стартап-проект «Rem» будет распространяться по модели software as a service (SaaS, программное обеспечение как услуга). Разработанная информационная система будет встраиваться в технологические и бизнес-процессы языковых школ.

Уровень технологической готовности (TRL, technology readiness level) продукта равен четырем (рисунок 3) (получен лабораторный образец, подготовлен лабораторный стенд, проведены испытания базовых функций связи с другими элементами системы).

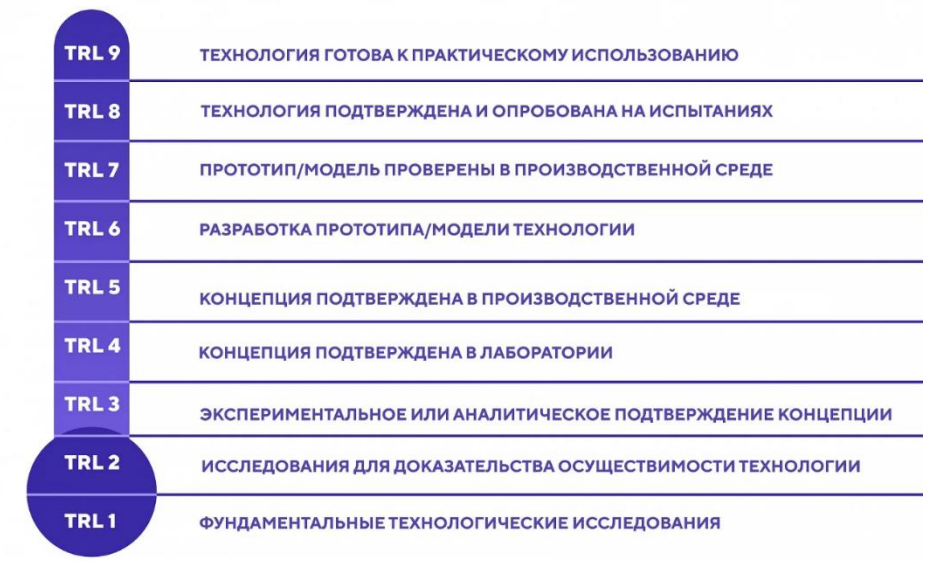


Рисунок 3 – Уровни TRL

Выделенная проблема основной целевой аудитории, которую решит разрабатываемый проект заключается в неэффективном взаимодействии языковых школ с клиентами (обучающимися), которое приводит к снижению результативности преподавателей, что в свою очередь приводит к потере денег ЦА.

1.5 Финансовая поддержка

Реализованный в рамках ВКРС проект выиграл грант Фонда содействия инновациям «УМНИК» в размере 500000 рублей на реализацию клиент-серверного приложения для изучения иностранной лексики, использующего комбинацию различных способов представления изучаемого материала.

1.6 Маркетинговый план

Коммерциализацию предлагаемого стартап-проекта предлагается производить по двум схемам экономического взаимодействия:

– b2b [6] (основная) – продажа готового полнофункционального решения для бизнеса (языковые школы и репетиторы), получение прибыли с интеграции системы и последующей абонентской платы;

– b2c [7] (второстепенная) – продажа готовых решений с ограниченным функционалом по системе абонентской платы для людей, самостоятельно изучающих иностранные языки.

Целевая аудитория разрабатываемого стартап-проекта:

– языковые школы и репетиторы (основная);

– люди, самостоятельно изучающие иностранные языки.

Каналами продвижения будущего продукта являются:

– прямые продажи и партнерства (сотрудничество с языковыми школами и образовательными организациями может быть эффективным способом продвижения будущего продукта, поскольку у небольших языковых школ имеется проблема, которую разрабатываемый проект решит путём интеграции в обучающие процессы языковых школ и позволит увеличить

качество обучения и эффективность взаимодействия между наставниками и обучающимися);

– маркетинг через контент (создание и распространение полезного контента, связанного с обучением иностранным языкам, может помочь привлечь внимание к будущему продукту, которое может включать: блоги, инфографику, видеоуроки и вебинары, которые демонстрируют преимущества системы);

– email-маркетинг (рассылка электронных писем потенциальным клиентам с информацией о продукте и его преимуществах может быть эффективным способом продвижения, email-маркетинг также можно использовать для отправки сообщений об обновлений продукта пользователям);

– социальные сети, которые могут быть мощным инструментом для продвижения продукта, что может быть достигнуто путем использовать социальных сетей для общения с целевой аудиторией, распространения контента и демонстрации преимуществ продукта, проведения конкурсов и раздач, что может помочь увеличить как вовлеченность аудитории, так и осведомленность о продукте;

– SEO и контекстная реклама, позволяющая оптимизировать веб-сайт (домашняя страница проекта, реализация которой выходи за рамки выполнения ВКРС) для поисковых систем и повысить эффективность контекстной рекламы, помогающую привлечь больше потенциальных клиентов (SEO-оптимизация может включать в себя добавление ключевых слов, связанных с обучением иностранным языкам и технологиями обучения, а также создание рекламных объявлений, которые появляются в результатах поиска, выполняемого потенциальными клиентами в браузере).

1.7 Финансовый план

Стоимость абонентской платы для языковых школ (в случае распространения продукта по модели b2b) будет зависеть от числа обучающихся языковой школы, использующих приложение.

Стоимость месячной подписки:

– 1 – 30 человек – 2000 рублей в месяц;

– 30 – 100 человек – 6000 рублей в месяц;

– 100 – 500 человек – 12500 рублей в месяц;

– 500 – 2000 человек – 28000 рублей в месяц;

– 2000 – 5000 человек – 58000 рублей в месяц;

– больше 5000 человек – по договоренности.

Стоимость годовой подписки:

– 1 – 30 человек – 20000 рублей в год;

– 30 – 100 человек – 60000 рублей в год;

– 100 – 500 человек – 120000 рублей в год;

– 500 – 2000 человек – 288000 рублей в год;

– 2000- – 5000 человек – 650000 рублей в год;

– больше 5000 человек – по договоренности.

При увеличении количества обучающихся в течение времени действия подписки клиенту предлагается обновить тарифный план в соответствии с его обновленными потребностями.

Стоимость подписки при распространении проекта по модели b2c будет следующей:

– ежемесячная – 250 рублей в месяц;

– ежегодная – 2500 рублей в год.

Стоимость абонентской платы предполагается из расчета среднего уровня затрат на обслуживание вычислительных мощностей, необходимых для предоставления услуг, в размере 10 рублей в месяц на 1 обучающегося.

План развития стартап-проекта представлен в таблице 2.

Таблица 2 – Календарный план

Этап	Описание работ
Первый этап реализации (с 08.2023 по 02.2025)	Август 2023 – сентябрь 2023: разработка мобильного приложения версии 0.1-alfa, в котором отсутствует связь с сервером. Главным функционалом мобильного приложения на этом этапе является алгоритм изучения терминов. Октябрь 2023 – декабрь 2023: разработка серверной части информационной системы, способной сохранять данные пользователя, предоставляющей доступ к добавлению и изменению данных, осуществляющей авторизацию пользователей и контроль доступа к операциям над данными. Январь 2024 – март 2024: разработка второй версии мобильного приложения (0.2-alfa), интегрированного с сервером. Апрель 2024 – июнь 2024: выполнение ВКРС в соответствии с завяленным графиком: анализ предметной области и требований к системе; проектирование информационной системы «Rem»; разработка информационной системы «Rem» (переписывание уже реализованного функционала в соответствии с проведённым проектированием); контроль качества программного кода; оформление пояснительной записки; защита ВКРС. Август 2024 – сентябрь 2024: оформление научно-технического отчёта в рамках гранта «УМНИК» Фонда содействия инновациям, проведение необходимых исследований в области внимания. Октябрь 2024 ноябрь 2024: проведение исследований совместно с партнером проекта кафедрой ПГУ «Русский язык как иностранный», сбор статистических данных. Декабрь 2024 – январь 2025: проектирование архитектуры нейронной сети для эффективного подбора порядка механик обучения. Февраль 2025 оформление заключительного отчёта по грантовой программе.
Второй этап реализации (с 03.2025 по 08.2026)	Март 2025 май 2025: доработка информационной системы до стадии продукта, готового к выходу на рынок. Июнь 2025 – август 2025: оформление юридического лица, поиск клиентов и партнёров (до первого сентября планируется найти пять языковых школ, готовых интегрировать разработанный продукт в свою инфраструктуру).
Третий этап реализации (с 09.2025 по 08.2026)	Сентябрь: старт продаж. Сентябрь 2025 –декабрь 2025: техническая поддержка продукта, исправление ошибок в коде, выявленных в результате эксплуатации, добавление функционала, запрошенного пользователями. Январь 2026 – март 2025: поиск новых клиентов, интеграция нейронной сети в информационную систему. Апрель 2026: написание заявки на участие в грантовом конкурсе «Старт-1». Май 2026 – июнь 2026: дообучение нейронной сети на данных, полученных в ходе эксплуатации информационной системы. Июль 2026 август 2026: тестирование каналов продвижения продукта, поиск новых клиентов, инвесторов и источников финансирования.
Четвертый этап реализации (с 09.2026 по 08.2027)	Сентябрь 2026 апрель 2027: активный поиск клиентов, расширение функционала, дообучение нейронной сети. Май 2027 получение средств по грантовому конкурсу «Старт-1» Июнь 2027 август 2027: активный поиск клиентов, расширение функционала, дообучение нейронной сети.
Пятый этап реализации (с 09.2027 по 08.2028)	Май 2028 получение средств по грантовому конкурсу «Старт-2». Сентябрь 2027 август 2028: активный поиск клиентов, расширение функционала, дообучение нейронной сети.

Приведем экономическую модель, составленную в ходе разработки финансового плана (таблица 3).

Таблица 3 – Финансовая модель

			Первый этап	Второй этап	Третий этап	Четвертый этап	Пятый этап
Количество клиентов b2b			0	0	5	15	50
Количество клиентов b2c			0	0	0	100	500
Количество пользователей информационной системы			0	0	250	1100	5000
Стоимость обслуживания одного пользователя в месяц в рублях					10	10	30
Длительность этапа (в месяцах)			19	8	12	12	12
Выручка (тыс. руб.)	в месяц		0	0	33,75	169	661
	за весь период		0	0	405	2028	7932
Расходы (тыс. руб.)	переменные за месяц		0	0	2,5	11	150
	постоянные	Закупка оборудования	300	1500	0	4000	4000
		Продвижение	0	0	0	100	200
		Заработная плата в месяц	0	0	20	70	100
		Административные	30	0	50	50	50
	за весь период		330	1500	320	5122	7250
Операционная прибыль за период без учёта закупки оборудования (тыс. руб.)			-30	0	85	906	4682
Маржа за весь период без учёта закупки оборудования (%)					21	44,7	59
Операционная прибыль за период (тыс. руб.)			-330	-1500	85	−3094	682
Маржа за весь период (%)					21	−152	8,5
Грант (тыс. руб.)			500	2000	0	6000	4000

Как можно видеть из финансовой модели (смотри таблицу 3), графой, требующей наибольшее количество денежных ресурсов, является закупка оборудования. Деньги из этой графы будут потрачены на покупку серверного оборудования:

– на первом этапе сервер в своей минимальной конфигурации, необходимый для проведения закрытого бета-тестирования информационной системы;

– на втором этапе – серверное оборудование, способное обеспечить одновременную работу 30000 пользователей;

– на четвертом этапе – серверное оборудование для обеспечения доступа к функциональности нейросетевого алгоритма обучения для одновременного обучения 10000 пользователей;

– на пятом этапе – серверное оборудование для обеспечения одновременного обучения дополнительных 10000 пользователей.

На пятом этапе стоимость обслуживания одного пользователя в месяц вырастет, поскольку к началу пятого этапа планируется внедрить массовое использование нейросетевого алгоритма обучения, на работу которого будет тратиться больше электричества и серверного времени.

Предполагается, что основное большинство клиентов будут языковые школы с количеством обучающихся от 30 до 100 человек. В процессе развития стартапа ожидается, что от одного до 10 процентов составят клиентов составят языковые школы с количеством учеников до 30 человек, так же ожидается, что от одного до пяти процентов составят языковые школы, с количеством от 100 до 500 человек, и около одного процента составят языковые школы, количество обучающихся которых превышает 500 человек.

Согласно финансовой модели, предлагаемый к разработке проект преодолеет порог рентабельности на пятом этапе реализации (в первой половине 2028 года), что подтверждает экономическую целесообразность разработки проекта.

2 Цифровая платформа для изучения иностранной лексики «Rem»

2.1 Анализ требований к проекту «Rem»

2.1.1 Анализ цифровых подходов к изучению иностранных языков

На сегодняшний день существует два основных подхода к обучению, с использованием которых разрабатываются информационные системы: геймификация и интервальное повторение.

Подход, основанный на геймификации, направлен на часть памяти, которая работает с осмысленными образами, а также на ассоциативную память.

Подход, основанный на интервальном повторении, использует в своей основе кривую забывания Эббингауза [8]. Стоит отметить, что в экспериментах Г. Эббингауза [9] использовались обессмысленные последовательности слогов и исследовал способности человека запоминать обессмысленную информацию. В результате проведенного исследования не удалось найти экспериментально подтвержденных данных на счет метода интервального повторения. Так же следует уделить внимание метрике экономии времени, которую использовал Г. Эббингауз в своем эксперименте. Данная метрика оценивает, сколько было сэкономлено времени при повторном обучении по отношению к первому. Подобная постановка метрики сама по себе не предполагает возможность изучить материал полностью и без ошибок (в этом случае, метрика будет просто равна нулю) [10]. Все вышеприведенные факты свидетельствуют о том, что эффективность использования кривой Эббингауза для изучения иностранных языков не является доказанной. Тем не менее, на практике данная кривая используется довольно часто в информационных системах.

Существует так же третий подход, менее известный и редко используемый. Данный подход основан на разноплановых ассоциациях (визуальных, аудиальных, лингвистических) с иностранным словом напрямую, то есть без предоставления пользователю перевода слова.

Представителем информационной системы, использующей данный подход, является семейство приложения Aword [11].

Использование обозначенных выше подходов характерно для информационных систем, специализирующихся на изучении иностранных языков, большая часть из которых является мобильными приложениями, распространяемых по модели b2c. Продается, как правило, подписка на приложение, действующая определенное время.

Примечательно, что не было найдено ни одной информационной системы (ровно, как и метода обучения, не использующего цифровые решения), использующей комбинированный подход, включающий в себя как механизмы геймификации, так и метод интервального повторения.

2.1.2 Обзор аналогов информационной системы «Rem»

В рамках выполнения ВКР был произведен анализ рынка, в результате которого были выявлены аналоги, которых можно условно разделить на три основные категории:

– приложения для индивидуального изучения иностранных языков;

– информационные системы крупных языковых школ;

– LMS.

Аналоги первого типа, как правило, не имеют интеграции с бизнес-процессами языковых школ, и предназначены в основном для самостоятельного изучения иностранных языков.

Первый тип аналогов можно разделить на две крупные группы по способу добавления контента.

Аналоги из первой группы имеют закрытую систему добавления контента, что означает, что пользователь не имеет возможности добавить свои собственные слова для изучения. По мнению автора ВКР, такое ограничение связано со сложностью способов представления информации, применяемых в приложениях первого типа. Следует отметить, что данное ограничение

позволяет гарантировать качественность предоставляемого пользователю материала.

Рассмотрим приложение Duolingo [12] – классический представитель первой группы аналогов. Весь контент приложения доступен бесплатно. Курсы состоят из модулей, каждый из которых разделен на уровни. Уровни разделены на отдельные уроки. Каждый урок состоит из нескольких небольших упражнений, которые помогают практиковать навыки чтения, письма, произношения и аудирования. В качестве недостатков можно выделить следующие:

– изучаемые уроки и слова заданы разработчиками Duolingo и у пользователя нет возможности внести собственные термины в систему для изучения;

– разнообразие механик геймификации усложняет процесс добавления новых уровней и уроков, что априори не позволит дать возможность пользователям самостоятельно изучать через приложение собственные темы.

К приложениям первой группы аналогов, помимо Duolingo, можно отнести так же lingualeo [13], Aword [11], «Курсы английского| игра слов» [14] и многие другие.

Аналоги из второй группы предоставляют пользователям возможность свободно добавлять изучаемые слова, что неизбежно приводит к понижению среднего уровня контента по сравнению с контентом, полностью добавленным профессионалами.

Классическим представителем второй группы аналогов является приложение Quizlet [15], которое позиционируется как универсальное приложение для изучения всего, не только иностранных языков. Примерами других областей знания, в изучении которых может применяться Quizlet служат: физика (изучение формул, терминов и определений), биология, химия, математика и т.д. Механика работы приложения строится на концепции интервального повторения.

В приложении термины делятся на модули. Каждый модуль можно изучить, посмотреть карточки модуля и пройти тестирование.

Главным преимуществом Quizlet, по сравнению с Duolingo, является свободное создания модулей пользователями.

В качестве недостатков Quizlet можно назвать:

– отсутствие проверки качества пользовательских модулей;

– отсутствие игровых механик, позволяющих улучшить качество запоминания.

Помимо Quizlet, аналогами второй группы можно считать приложения словари, к примеру U-dictionary [16] и подобные.

Аналоги второго типа – это информационные системы крупных языковых школ. Главным отличием от разрабатываемого продукта является способ распространения: ИС языковых школ используются для получения выгоды внутри определенной организации, а на разрабатываемый продукт планируется продавать лицензии на использование непосредственно языковым школам без собственной ИС, то есть разрабатываемый продукт будет не способом получения прибыли, а товаром.

Предлагаемый способ коммерциализации не может быть применен к аналогам второго типа, так как их внутренняя инфраструктура неразрывно связана с бизнес-процессами организации.

Наиболее близким аналогом второго типа по функционалу является информационная система Skyeng [new_2], которая была разработана частной школой английского языка для внутреннего использования. В этой ИС, помимо специфичных механик языковой школы, реализованы:

– изучение новых слов;

– практика в режиме Self-Study при помощи просмотра видео и упражнений с аудиокнигами.

Другими аналогами второго типа являются ИС следующих языковых школ: Тетрика [17], Инглекс [18], TutorPlace [19], Puzzle English [20], Deutsch Online [21], Advance [22], Alfa School [23].

Аналоги третьего типа представляют из себя LMS (Learning Management System, системы управления обучением). Это программное обеспечение, позволяющее создавать онлайн-курсы, обеспечивать управление и администрирование этих курсов, предоставлять доступ к обучающим материалам, тестам и заданиям.

Ниже представлен краткий анализ наиболее крупных LMS.

а) iSpring Learn [24]:

1) тарифы:

– тариф «Start» – от 143 рублей за пользователя в месяц при обучении до 25 человек;

– тариф «Business» – от 138 рублей за пользователя в месяц при обучении да 300 человек;

2) недостатки:

– нет email-рассылок;

– нет системы управления взаимоотношениями с клиентами для статистики и учета;

б) TeachBase [25]:

1) тарифы:

– тариф «Demo» – бесплатно предоставляет доступ к системе на 2 недели для 10 пользователей;

– тариф «Оптимальный» – 163 рубля за пользователя для 50 сотрудников;

2) недостатки:

– недостаточное количество практических занятий;

– невозможность формирования групповых отчетов о результатах выполнения отдельного задания;

Помимо индивидуальных недостатков, LMS обладают, общим для всех, не позволяющим осуществлять использование LMS изучения иностранных языков: отсутствие специализированных способов представления изучаемого материала, необходимых для более качественного изучения иностранных

языков. Этот недостаток вызван объективными причинами: LMS имеют очень широкую область применения, что неизбежно ведет к повышению абстрактности способов представления информации.

2.1.3 Анализ требований к информационной системе «Rem»

2.1.3.1 Анализ функциональных требований

Выявим общие функциональные требования, предъявляемые к разрабатываемой информационной системе:

– система должна предоставлять пользователю доступ к изучаемым словам (терминам);

– процесс изучения терминов пользователем должен фиксироваться и сохраняться, что необходимо для корректной работы алгоритма изучения терминов и повышения дружелюбности пользовательского интерфейса;

– система должна реализовывать функционал для добавления пользователем своих собственных терминов;

– система должна позволять пользователям изучать чужие (созданные другим пользователем) термины;

– система не должна позволять пользователям изменять термины, которые были созданы другими пользователями;

– система должна обеспечивать возможность объединять изучаемые термины в тематические блоки (модули), что необходимо для повышения дружелюбности пользовательского интерфейса;

– система должна обеспечивать возможность объединения тематических модулей в группы (папки), что необходимо для повышения дружелюбности пользовательского интерфейса;

– система должна обеспечивать возможность создания вложенных папок (папки могут содержать не только тематические модули, но и вложенные папки), что необходимо для повышения дружелюбности пользовательского интерфейса;

– система должна обеспечивать возможность добавления дополнительной информации об изучаемых терминах, что необходимо для корректной работы алгоритма.

На основе выявленных функциональных требований была разработана диаграмма вариантов использования мобильного приложения, представленная на рисунке 4.

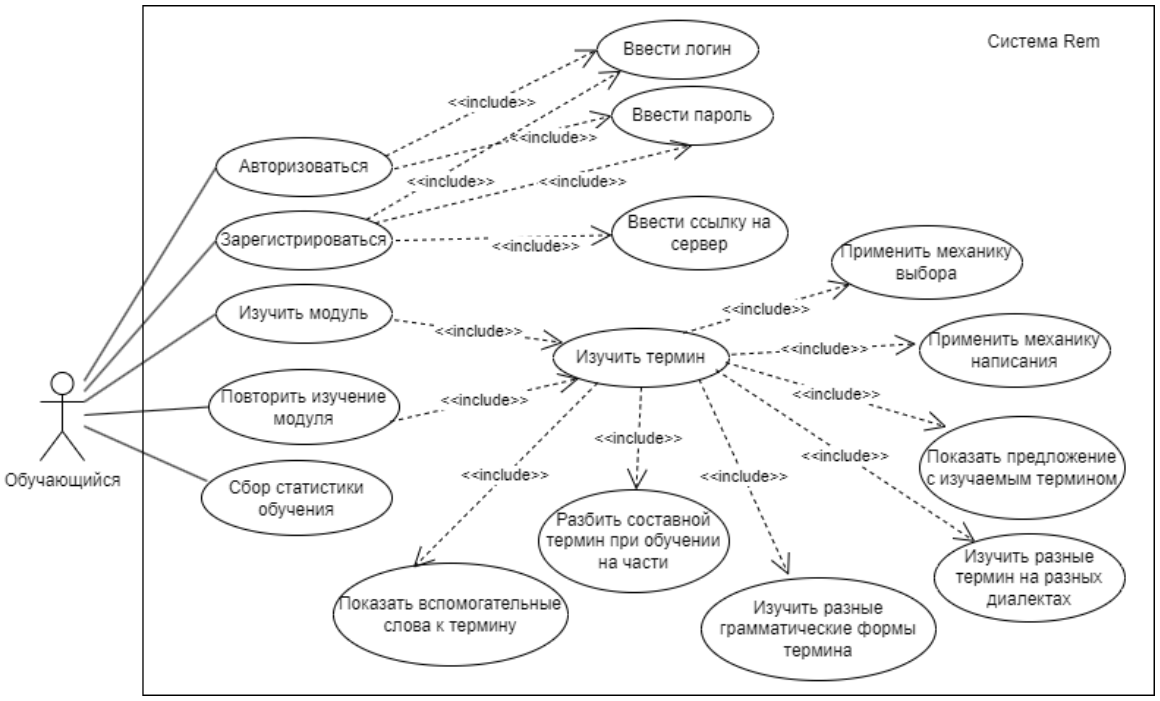


Рисунок 4 – Диаграмма вариантов использования мобильного приложения

На основе выявленных функциональных требований была разработана диаграмма вариантов использования сервера (рисунок 5), на которой помимо функционала прецедентов, связанных с изучением терминов, так же присутствуют прецеденты добавления данных. Дополнительный функционал сервера необходим для применения разработанной информационной системы по модели SaaS (b2b). При этой модели использования заниматься созданием контента будут представители языковой школы через специальный WEB-интерфейс, разработка которого выходит за рамки данной ВКР.

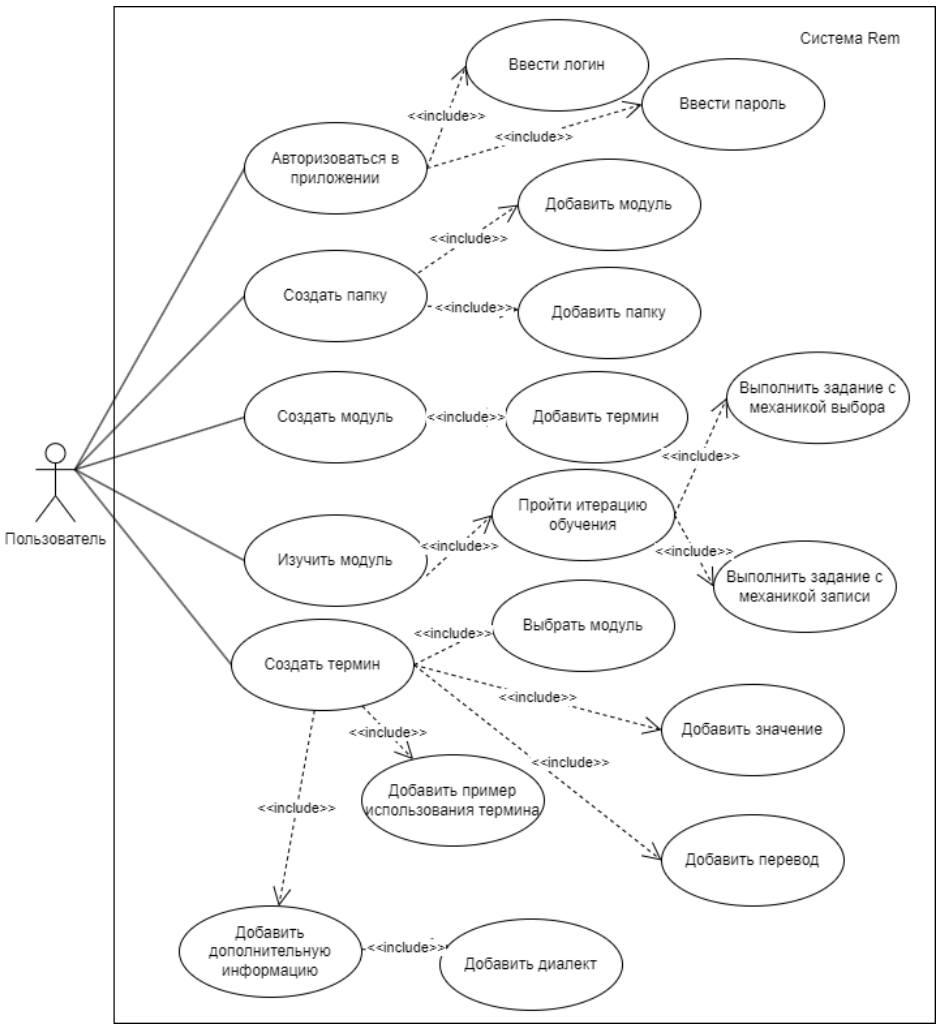


Рисунок 5 – Диаграмма вариантов испольоования сервера

Спецификации наиболее значимых с точки зрения работоспособности системы вариантов использования приведены в приложении Г.

2.1.3.2 Анализ нефункциональных требований

Разрабатываемая информационная система ориентирована на взаимодействие с пользователем, следовательно, должна обладать удобным пользовательским интерфейсом.

Пользователь будет осуществлять взаимодействие с системой через мобильное приложение на клиентском устройстве. Пользователю необходимо иметь доступ к своим данным с разных клиентских устройств, в связи с этим разрабатываемая информационная система должна реализовывать функциональность аккаунтов пользователей. Пользователь может войти в один аккаунт на нескольких устройствах.

Для реализации необходимой функциональности, информационная система должна включать в свой состав серверную часть, обеспечивающую хранение, добавление, модификацию данных, доступ к данным. Клиентское приложение должно реализовывать функционал хранения данных пользователя, полученных с сервера, что необходимо для использования приложения без доступа к интернету. Мобильное приложение должно обновлять локальную базу данных с материалом, применяемым для обучения пользователя, и синхронизировать локальное состояние базы данных с другими устройствами пользователя.

Приложение должно отправлять локальные изменения базы данных на сервер для синхронизации прогресса обучения пользователя.

Сервер должен быть написан на языке программирования Python и использовать базу данных PostgreSQL, клиент должен быть написан на языке программирования Dart.

Сервер должен выдерживать одновременную работу 30000 пользователей. Во время активной работы клиентское устройство пользователя должно совершать в среднем один запроса в три минуты (среднее время длины одной итерации обучения).

Сервер должен быть рассчитан на хранение данных 100000 пользователей. Максимальное расчетное среднее количество записей на одного пользователя 10000 записей в базе данных, среднее количество ячеек в которых равно 10, если допустить, что каждые 1000 записей будут весить 1 мегабайт (данные получены в ходе эксперимента, проведенного в рамках анализа нефункциональных требований к информационной системе «Rem»), и применить метод Ферми, то получим, что база данных на 1000000000 записей будет занимать примерно один терабайт дискового пространства, при этом погрешность вычислений, произведенных по методу Ферми, весьма велика.

Для резервного копирования данных выделим в полтора раза больше дискового пространства, чем требуется базе данных для 1000000000 записей,

с расчетом на то, что резервные копии будут занимать меньше места, чем база данных, поскольку не будут содержать таблиц индексов.

Из результатов эксперимента, проведенного в рамках анализа нефункциональных требований, известно, что в среднем 100 запросов пользователей производят один килобайт файла журналов. Для расчёта количества дискового пространства, необходимого для журналирования, возьмём наихудший сценарий, при котором максимальное количество клиентов одновременно пользуются системой. В этом случае за одну секунду к серверу в среднем будет совершаться 167 запросов. Тогда в сутки будет произведено 140 гигабайт файлов журнала.

Общую массу файлов журналирования будем хранить трое суток, журналы с отчетом об ошибках (предположительно до пяти процентов от общего объема дискового пространства остальных файлов журналов) будем хранить 14 суток. Тогда для хранения файлов журнала потребуется 500 гигабайт дискового пространства при максимальной нагрузке на систему.

Средняя длительность запроса к серверу не должна превышать одной секунды, из которой в среднем от 60% до 90 % времени должно приходиться на ожидание ответа от базы данных (содержащей до 1000000000 записей) в асинхронном режиме.

По результатам эксперимента, проведенного в рамках анализа нефункциональных требований, один запрос к конечной точке, выполняющей сериализацию данных в формат json из python-объекта, в среднем занимает 80 миллисекунд при развертывании сервера в однопоточном режиме на устройстве с тактовой частотой процессора 4,5 ГГц.

Воспользуемся методом Ферми и на основе этих данных получим, что в секунду одноядерный сервер с тактовой частотой в один гигагерц способен обработать в среднем 2,7 запроса. Если учитывать, что клиентское устройство во время активной работы в среднем будет совершать один запрос в три минуты, то такой сервер сможет поддерживать одновременную работу 500 пользователей. Для увеличения количества поддерживаемых

одновременно активных пользователей до целевого, предлагается использовать сервер с 12 ядрами и тактовой частотой пять ГГц.

Для расчета количества ядер, необходимых для работы базы данных, воспользуемся результатами эксперимента, проведенного в рамках анализа нефункциональных требований, и применим метод Ферми. Если на один запрос к базе данных с 1000000000 записями будет требоваться 0,1 секунды процессорного времени (результаты взяты из данных открытых тестов в базе данных с 2300000000 записями), то получим, что для обеспечения одновременной работы требуемого количества пользователей базе данных потребуется сервер с 12 ядрами и тактовой частотой пять ГГц.

В результате эксперимента было установлено, что для поддержания 5 соединений с сервером базе данных требуется 30 мегабайт оперативной памяти. Для поддержания по 10 соединений (это число было вычислено путем применения метода Ферми к уже посчитанным данным о количестве запросов в единицу времени и ориентировочному проценту времени, потраченному на взаимодействие с базой данных) для каждого из 12 процессов потребуется примерно 3,5 гигабайт оперативной памяти. Выделим также четыре гигабайта на поддержание работы движка базы данных с 1000000000 записями.

По результатам проведения эксперимента было выявлено, что интерпретатор python с развернутым сервером занимает 60 мегабайт оперативной памяти, для обработки 1000 запросов потребовалось 150 мегабайт оперативной памяти. Применим к этим данным метод Ферми, и получим, что для функционирования 12 ядерного сервера потребуется примерно 12 гигабайт оперативной памяти.

Выделим также 4,5 гигабайта на работу основной операционной системы.

Требования к аппаратному и программному обеспечению устройства, на котором будет развертываться серверная часть информационной системы:

– операционная система windows (10 и выше), Debian (10 и выше), ubuntu (22.04 и выше);

– установлен docker engine и docker-compose;

– установлена система контроля версий git;

– имеется доступ к сети интернет со скоростью, не меньшей 50 мегабит в секунду;

– сервер должен иметь как минимум 24 ядерный процессор с тактовой частотой не менее пяти ГГц;

– объем дискового пространства не должен быть меньше трех терабайт;

– количество оперативной памяти не меньше 24 гигабайт.

Требования к аппаратному и программному обеспечению устройства, на котором будет развертываться мобильное приложение:

– операционная система Android восемь и выше;

– доступ к сети интернет (скорость выше одного мегабита в секунду);

– свободный объем дискового пространства больше 200 мегабайт;

– объем свободной (не занятой операционной системой) оперативной памяти более 300 мегабайт.

2.1.4 Постановка целей и задач

Целью данной выпускной квалификационной работы (ВКР) является разработка информационной системы «Rem» для изучения иностранных языков, использующей комбинацию различных способов представления изучаемого материала.

Для достижения поставленной цели необходимо решить следующие задачи:

– проанализировать предметную область изучения иностранных языков, выявить конкурентов и сформулировать требования;

– спроектировать информационную систему «Rem» (серверную и клиентскую части), базу данных и пользовательский интерфейс;

– произвести оценку бюджета на разработку информационной системы «Rem»;

– реализовать серверную и клиентскую части информационной системы «Rem», базу данных и пользовательский интерфейс клиентской части;

– произвести тестирование и анализ качества разработанного программного кода.

2.2 Проектирование информационной системы «Rem»

2.2.1 Проектирование архитектуры

2.2.1.1 Проектирование архитектуры сервера

Серверная часть приложения должна обеспечивать доступ к данным; хранение, добавление изменение и персонализацию данных; авторизацию и аутентификацию пользователей системы.

Согласно определению, данному в книге «Фундаментальный подход к программной архитектуре» М. Ричардса и Н. Форда, «архитектура ПО – состоит из структуры системы в сочетании со свойствами архитектуры, которые должны поддерживаться системой, архитектурными решениями и принципами проектирования» [26 стр. 27].

Поскольку разрабатываемая информационная система предназначена для коммерческого использования и распространения по модели SaaS, особой важностью обладают следующие архитектурные свойства:

– масштабируемость;

– адаптируемость;

– развертываемость;

– эволюционность;

– тестируемость.

Сравним наиболее распространенные архитектуры программного обеспечения (ПО) по степени поддержки выявленных архитектурных свойств от 1 до 5 (1 – архитектура почти не поддерживает свойство, 5 – архитектура хорошо поддерживает свойство). Результаты сравнения представлены в

таблице 4, данные для сопоставления взяты из книги М. Ричардса и Н. Форда «Фундаментальный подход к программной архитектуре».

Таблица 4 – Сравнение архитектурных свойств

Архитектура	Развер-тываем-ость	Адаптиру-еьость	Масштабиру-емость	Эволюцион-ность	Тестиру-емость	Все го	Просто-та реали-зации
Монолитные архитектуры							
многоуров-невая	1	1	1	1	2	6	5
конвейерная	2	1	1	3	3	10	5
микроядер-ная	3	1	1	3	3	11	4
Распределенные							
сервисная	4	2	3	3	4	16	3
событийная	3	3	5	5	2	18	1
микросер-висная	4	5	5	5	4	23	1

Под сервисной архитектурой понимается «архитектура на основе сервисов» [26].

Из приведенной сравнительной таблицы видно, что для разрабатываемой информационной системы наиболее подходящей является микросервисная архитектура. Однако сложность реализации данной архитектуры крайне высока, а время на разработку – ограничено. В силу этих доводов было принято решение о выборе сервисной архитектуры, так как (в отличии от событийно-ориентированной) информационную систему, использующую данную архитектуру, можно сравнительно легко разбить на микросервисы, тем самым изменив архитектуру системы, что в свою очередь позволит быстро увеличить способность системы к масштабируемости.

Поскольку в дальнейшем планируется изменение архитектуры системы на микросервисную, разработка системы будет включать в себя шаги, необязательные для реализации сервисной архитектуры в чистом виде, но в будущем позволяющие упростить разбиение переход к микросервисной архитектуре.

Выделим независимые друг от друга части (модули первого уровня) для запланированной ИС (таблица 5).

Таблица 5 – Модули первого порядка

№	Название	Роль
1	модуль авторизации	аутентификация пользователей системы
2	модуль пользователей	CRUD операции над сущностью пользователей
3	модуль папок	CRUD операции над сущностью папок
4	модуль модулей	CRUD операции над сущностью модулей
5	модуль терминов	CRUD операции над сущностью терминов
6	модуль доп. информации о термине	CRUD операции над сущностью, содержащей дополнительную информацию о термине
7	модуль предложений	CRUD операции над сущностью предложений
8	модуль временных меток	CRUD операции над сущностью временных меток

Следующим шагом выделим части, являющиеся общими для нескольких модулей первого уровня (модули второго уровня) (таблица 6).

Таблица 6 – Модули второго уровня

№	Название	Роль	№ зависящих частей
9	движок БД	предоставляет доступ к БД	2, 3, 4, 5, 6, 7, 8
10	модуль утилит	реализует общие функции, необходимые для работы серверной части в целом	1, 2, 3, 4, 5, 6, 7, 8, 9
11	модуль конфигурации	реализует работу с конфигами	1, 2, 3, 4, 5, 6, 7, 8, 9
12	движок миграций БД	реализует общие механизмы миграции БД	2, 3, 4, 5, 6, 7, 8, 9, 10, 11

В разрабатываемой системе возникает потребность в использовании одинакового функционала в концептуально независимых частях (это является серьезной проблемой, которую необходимо решить на этапе проектирования, ведь в будущем планируется изменить архитектуру на микросервисную, в результате чего каждый из модулей первого уровня станет отдельным

архитектурным квантом, и не будет иметь возможности для использования функционала, расположенного в другом архитектурном кванте).

Существует лишь один классический способ, который позволил бы интегрировать необходимый функционал модулей второго уровня в каждый модуль первого уровня (без переиспользования кода (переиспользование кода невозможно по причине несовместимости с выбранной архитектурой)) – дублирование кода.

Однако этот метод имеет массу недостатков. Самый значимый из них: «При дублировании возникает проблема с поддержкой актуальной версии каждого из дублей. В больших системах с тысячами микросервисов это проблема встает особенно остро, так как с увеличением количества мест, нуждающихся в исправлениях (актуализации версии дублированного кода), возрастает вероятность того, что разработчик просто забудет актуализировать версию дублированного кода в нескольких местах» [27].

В виду приведенных выше соображений было принято решение о разработке в рамках проектирования данной информационной системы оригинального метода достижения компромиссности между переиспользованием и дублированием кода в рамках микросервисной архитектуры [27], применение которого позволило бы сократить негативное влияние как переиспользования кода, так и дублирования.

На основе модулей первого и второго уровня выделим независимые git репозитории для реализации разработанного метода достижения компромиссности (таблица 7). Поскольку было принято архитектурное решение о проектировании системы, архитектурно наиболее близкой к микросервисной, в дальнейшем на этапе проектирования будем называть сервисы системы микросервисами. Такое изменение терминологии позволит избежать путаницы, ведь выделенные модули по своей сути ближе именно к микросервисам, нежели к сервисам.

На основе выделенных git submodules построим диаграмму компонентов для git-репозиториев.

Таблица 7 – Выделение git submodules

№ п/п	Название компонента	Назначение	БД	.gitignore, лицензия, readme	Поддержка python	Автоматически сгенерированный http клиент для взаимодействия с другими сервисами	Менеджер конфигов	Утилитарный менеджер	Зависит ОТ
1	микросервис авторизации	авторизация пользователей	-	+	+	+	+	+	10, 12, 13, 14, 15,
2	микросервис пользователей	хранение, добавление модификация и получение данных пользователей	+	+	+	+	+	+	10, 12, 13, 14, 15, 16, 17
3	микросервис папок	хранение, добавление модификация и получение данных о файловой структуре	+	+	+	+	+	+	10, 12, 13, 14, 15, 16, 17
4	микросервис модулей	хранение, добавление модификация и получение данных о модулях	+	+	+	+	+	+	10, 12, 13, 14, 15, 16, 17
5	микросервис терминов	хранение, добавление модификация и получение данных о терминах	+	+	+	+	+	+	10, 12, 13, 14, 15, 16, 17
6	микросервис дополнительной информации о терминах	хранение, добавление модификация и получение дополнительной информации о терминах	+	+	+	+	+	+	10, 12, 13, 14, 15, 16, 17
7	микросервис временных меток	хранение, добавление модификация и получение данных о временных метках	+	+	+	+	+	+	10, 12, 13, 14, 15, 16, 17
8	микросервис предложений	хранение, добавление модификация и получение данных о предложениях	+	+	+	+	+	+	10, 12, 13, 14, 15, 16, 17
10	базовый git-репозиторий	.gitignore, лицензия, readme	-	реализует	-	-	-	-	
12	базовый python репозиторий	реализация специфичных для python файлов, модификация .gitignore и readme	-	+	реализует	-	-	-	10

Продолжение таблицы 7

№ п/п	Название компонента	Назначение	БД	.gitignore, лицензия, readme	Поддержка python	Автоматически сгенерированный http клиент для взаимодействия с другими сервисами	Менеджер конфигов	Утилитарный менеджер	Зависит ОТ
13	репозиторий python-конфигов	работа с файлами конфигурации, переменными окружения	-	+	+	-	реализует	-	10, 12
14	репозиторий python-утилит	реализация общих функций, необходимых всем python-микросервисам, таких как взаимодействие с БД	-	+	+	-	+	реализует	10, 12, 13
15	шаблон python-микросервиса	задается базовая файловая структура микросервиса	-	+	+	реализует	+		10, 12, 13,14
16	Движок миграции БД	контролирует структуру БД в процессе разработки	реализует	+	+	-	+	+	10, 12, 13, 14
17	шаблон python-микросервиса с БД	добавляется движок миграции данных в шаблон микросервиса	+	+	+	+	+	+	10, 12, 13, 14, 15, 16

Для приведенных на рисунке 6 репозиториев отношение расширения на практике может быть реализовано при помощи клонирования базового репозитория. В случае необходимости обновления базовых компонентов расширяющего репозитория необходимо выполнить слияние базового репозитория с исходным (расширяющим) репозиторием (git merge upstream master). Таким образом будет достигаться имплементация базового функционала расширяющими репозиториями.

Отношение композиции может быть реализовано через git submodules (вложенные git репозитории). Данное решение оправданно, так как отношение композиции является частным случаем отношения агрегации. Однако, в силу особенности механизмов, реализующих данное отношение на практике, классы-части, которые соединяются отношением композиции с классом-целым, не могут существовать обособленно [28].

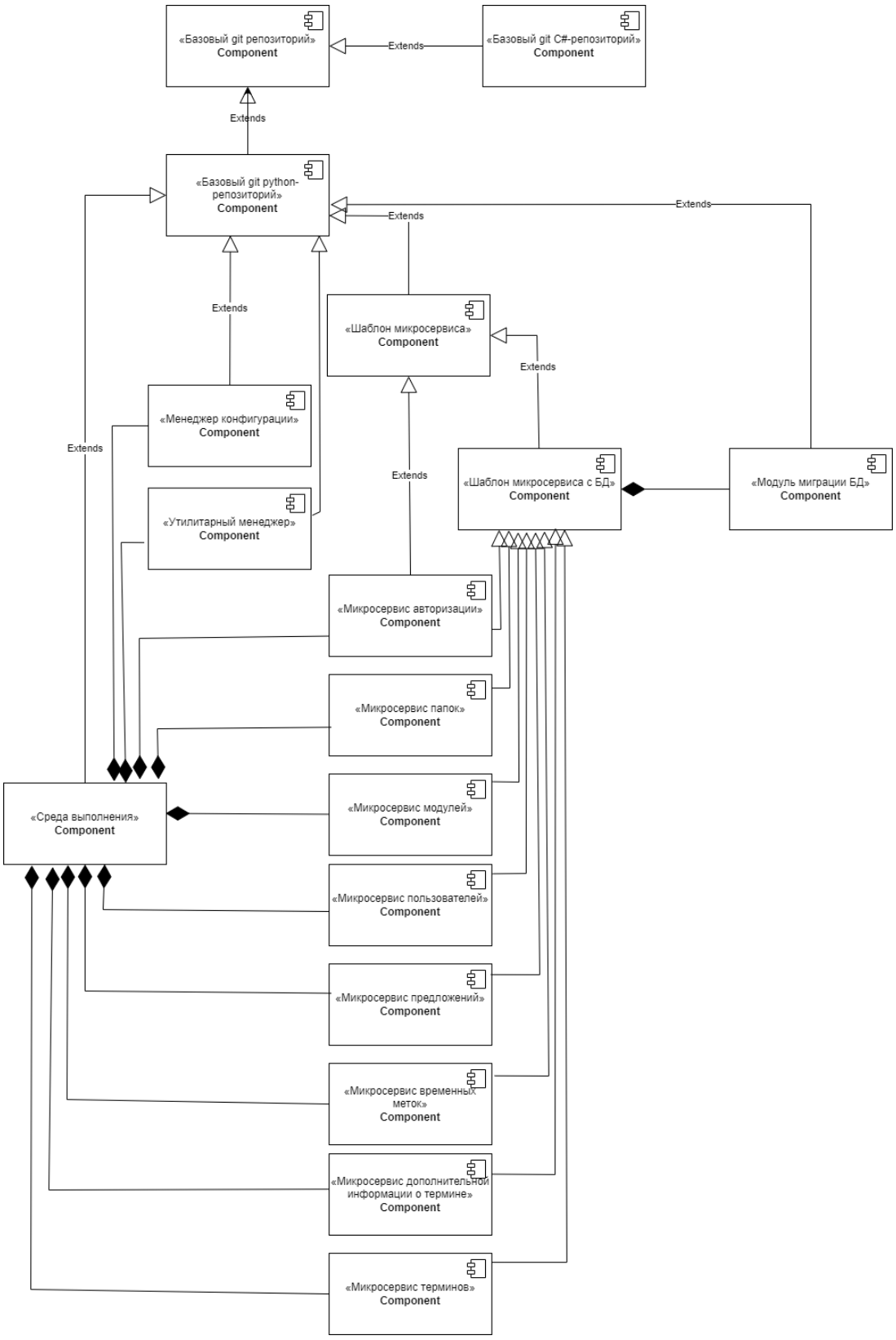


Рисунок 6 – Диаграмма компонентов для git-репозиториев

Следует отметить, что модуль миграции БД зависит от данных микросервиса (таблиц базы данных конкретного микросервиса). При этом

основной функционал микросервиса во время исполнения не зависит от функционала модуля миграций БД. Исходя из этих соображений было принято архитектурное решение о включении отдельного экземпляра модуля миграции БД (в качестве вложенного git репозитория) в каждый из микросервисов, функциональность которого включает взаимодействие с БД напрямую.

Наличие отношений композиции делает невозможным реализацию микросервисов как отдельных архитектурно-независимых квантов, что обозначает отсутствие возможности запуска микросервиса вне состава репозитория контейнера (выполняющего роль окружения для запуска и агрегатора для всех контейнеров, а также поставщика утилитарного менеджера и менеджера конфигов).

При этом, несмотря на невозможность запуска репозиториев вне состава репозитория контейнера, репозиторий контейнера может быть запущен вовсе без микросервисов.

Возможности git submodules позволяют, помимо ведения версионирования, реализовать гибкость в разбиении системы на архитектурные кванты путем запуска нескольких инстансов, содержащих различные вложенные репозитории микросервисов. (При клонировании репозитория контейнера предоставляется возможность указать специальный флаг, определяющий вложенные репозитории, которые будут загружены из системы контроля версий (и войдут в состав архитектурного кванта), а также репозитории, которые не будут загружены. Загруженные репозитории микросервисов будут работать в рамках одного архитектурного кванта).

Микросервисы, объединенные в один архитектурный квант, будут расположены в одном проекте, в одном родительском каталоге, выполняться одним python-интерпретатором в рамках одного потока (из-за ограничений GIL python не поддерживает многопоточность напрямую (несмотря на это есть способы организовать как многопоточную, так и многопроцессорную обработку) и взаимодействовать друг с другом (если такое взаимодействие

необходимо для исправной работы микросервисов) через прямой вызов функций конечных точек, а не по сетевым протоколам).

Таблицы каждого микросервиса будут расположены в отдельной схеме БД, которая (в свою очередь) будет расположена в одной БД с остальными схемами микросервисов, расположенных в том же архитектурном кванте. Система миграции каждого микросервиса отвечает только за создание только той схемы, в которой расположены данные родительского микросервиса. Между таблицами, расположенными в разных схемах (соответственно, и находящихся в зоне ответственности разных микросервисов) не существует связей, они полностью изолированы друг от друга, что позволяет разворачивать каждый микросервис в отдельном инстансе, тем самым достигается гибкость в развертывании системы.

В рамках одного архитектурного кванта согласованность данных реализуется при помощи передачи ORM объекта транзакции БД из инициализирующей функции конечной точки в вызываемую функцию (это возможно, поскольку в рамках одного архитектурного кванта микросервисы взаимодействуют через прямой вызов функции). Это позволяет производить сложные операции над данными, задействующие несколько микросервисов, в рамках одной физической транзакции БД, за счет чего обеспечивается согласованность данных.

Взаимодействие между микросервисами, не расположенными в рамках одного архитектурного кванта, осуществляется через сетевые протоколы (преимущественно http, RESTful API).

Данные микросервисов, расположенных в различных архитектурных квантах, хранятся в разных базах данных. Согласованность данных осуществляется путем внедрения архитектурного паттерна SAGA, однако его полноценная реализация выходит за рамки выпускной квалификационной работы. Именно отсутствие полноценной реализации паттерна SAGA не позволяет называть архитектуру разрабатываемой системы микросервисной.

На основе приведенного выше описания построим детализированную диаграмму компонентов (рисунок 7).

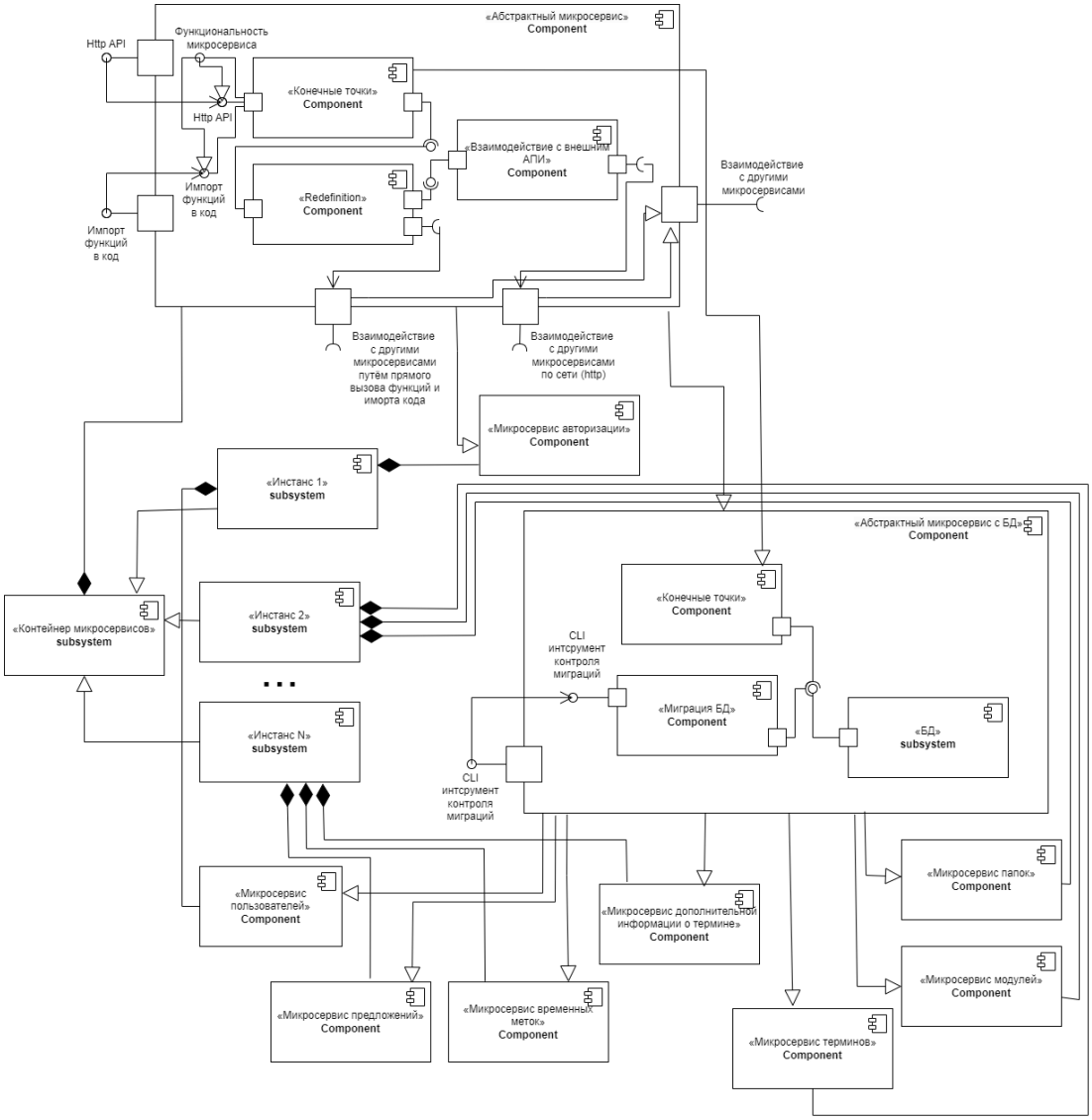


Рисунок 7 – Детализированная диаграмма компонентов

Все микросервисы расширяют функциональность абстрактного микросервиса, который содержит компонент конечных точек (модуль, в котором реализовано непосредственно API микросервиса и вся бизнес-логика).

Для работы компоненту конечных точек может потребоваться функциональность, реализованная в других микросервисах. Для организации доступа к этой функциональности (как для микросервисов, расположенных в рамках одного архитектурного кванта, так и для микросервисов из других архитектурных квантов) был разработан компонент «redefinition».

В этом компоненте объявлены все функции, используемые данным микросервисом, но физически расположенные в других микросервисах. При старте сервера данный компонент проверяет возможность импорта объявленных функций (это возможно только в том случае, если запрашиваемая функция расположена в микросервисе, находящемся в том же архитектурном кванте, что и данный микросервис). Если это возможно, то определение объявленной функции импортируется из кода стороннего микросервиса. В противном случае определение объявленной функции импортируется из компонента «Взаимодействие с внешним АПИ».

Этот компонент представляет из себя автоматически сгенерированный python-код http-клиента для взаимодействия с микросервисами, расположенными в другом архитектурном кванте. Данный http-клиент генерируется на основе OpenAPI Swagger всех микросервисов, необходимых для корректной работы данного микросервиса, (Swagger предоставляется каждым микросервисом отдельно, однако есть и общий Swagger всего архитектурного кванта, полученный путем объединения Swagger всех микросервисов текущего архитектурного кванта).

Функциональность «абстрактного микросервера» расширяют микросрвис авторизации и «абстрактный микросервис с БД» (для работы микросервиса авторизации не требуется БД, так как все данные о пользователях, необходимые для работы, расположены в микросервисе пользователей).

В «Абстрактном микросервисе с БД» расширяется функциональность компонента «Конечные точки», а именно, добавляется еще один порт, через который компонент получает данные из требуемого интерфейса «взаимодействия с БД». Добавляется также подсистема самой БД и компонент миграции БД, который тоже имеет требуемый интерфейс «взаимодействия с БД».

Компонент миграции БД предоставляет console like interface (CLI) для контроля за версионированием схемы БД данного микросервиса. Физически данный компонент представлен как вложенный git репозиторий.

Все микросервисы кроме «авторизации» расширяют функционал «абстрактного микросервера с БД».

Композиция «абстрактного микросервиса» с подсистемой контейнера микросервисов подчеркивает то, что ни один из микросервисов не может существовать самостоятельно, каждым микросервисом обязательно должна владеть подсистема «контейнер микросервисов».

Каждый экземпляр подсистемы «контейнер микросервисов» представляет из себя отдельный архитектурный квант.

При этом, необязательно, чтобы подсистема «контейнер микросервисов» была только одна и содержала все микросервисы. Это показано путем расширения функционала подсистемы «Контейнер микросервисов» тремя инстансами (но в общем случае их может быть сколько угодно). Каждый микросервис обязан быть включен в один из инстансов.

2.2.1.2 Проектирование архитектуры приложения

Клиентская часть информационной системы представляет из себя мобильное приложение, которое должно предоставлять конечному пользователю возможность изучения иностранных слов. Данные о процессе изучения должны передаваться на сервер. В случае изменения данных на сервере приложение должно загружать обновленные данные и использовать их в процессе обучения в случае надобности.

Выделим свойства архитектуры, наиболее значимые для проектируемого мобильного приложения. Поскольку основная роль мобильного приложения – предоставлять доступ к функционалу системы (изучению иностранных языков) для конечных пользователей, приложение должно быть удобным для пользователя. В связи с этим, наиболее значимыми свойствами архитектуры являются «Надежность» и «Тестируемость».

Функционал разрабатываемой информационной системы в будущем будет расширяться (планируется значительное расширение), в связи с этим были выделены архитектурные свойства системы, связанные с простотой модернизации мобильного приложения: «Эволюционность» и «Модульность». В таблице 8 (которая была составлена на основе данных, приведенный в книге «Фундаментальный подход к программной архитектуре» [26]) приведен сравнительный анализ наиболее популярных архитектур для мобильных приложений по выделенным свойствам архитектуры.

Таблица 8 – Выбор архитектуры мобильного приложения

№	Эволюционность	Модульность	Надежность	Тестируемость	Всего	Простота
Монрлитные						
Многоуровневая	1	1	3	2	7	5
Конвеерная	3	3	3	3	12	5
Микроядерная	3	3	3	3	12	4
Распределенные						
Сервисная	3	4	4	4	15	3
Событийная	5	4	3	2	14	1
Микросервисная	5	5	4	4	18	1

По результатам (смотри таблицу 8) наибольшее количество баллов (каждое свойство оценивалось от 1 до 5, итог вычислялся путем суммирования оценок каждого свойства архитектуры) набрала микросервисная архитектура. Однако, стоит учитывать, что для мобильных приложений данная архитектура не типична (как и событийная в классическом ее понимании), поскольку является распределенной. Анализ текущего состояния отрасли мобильных приложений показал, что микросервисная архитектура для мобильных приложений, хоть и используется, но крайне редко в масштабных и сложных проектах.

Так же следует учитывать сложность реализации микросервисной и событийной архитектур. Архитектура на основе сервисов (сервисная),

несмотря на свой распределенный характер, применяется в мобильных приложениях довольно часто. В качестве сервисов используются изолированные потоки или процессы.

В результате анализа требований к мобильному приложению были выделены два сервиса:

– сервис взаимодействия с пользователем – выполняет взаимодействие с пользователем, отвечает за отрисовку пользовательского интерфейса, за алгоритм процесса обучения, за обработку событий, поступивших от пользователя;

– сервис синхронизации – отвечает за синхронизацию данных мобильного приложения и сервера, выполняет обновления локальной базы данных, загружает на сервер локальные изменения данных пользователя, полученные в процессе изучения иностранных слов.

Для работы мобильного приложения без доступа к интернету и для снижения нагрузки на сервер, в мобильном приложении необходимо сохранять данные пользователя. Для этого будет использоваться локальная база данных, к которой будут иметь доступ все сервисы.

Укрупненная диаграмма компонентов, построенная на основе технического задания и архитектуры приведена на рисунке 8.

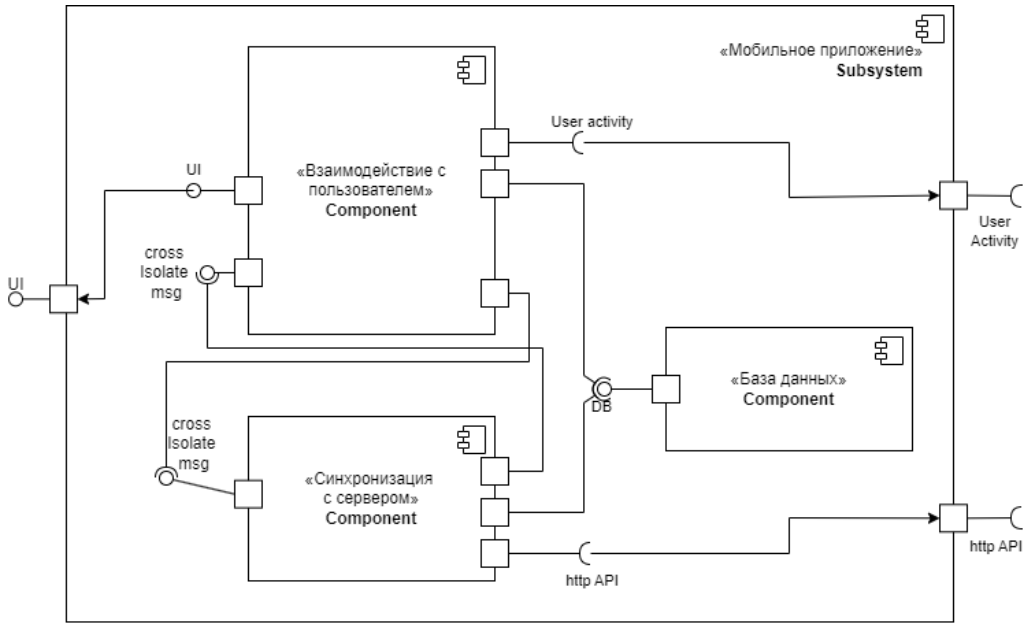


Рисунок 8 – Диаграмма компонентов для мобильного приложения

На диаграмме компонентов сервис «Взаимодействие с пользователем» и сервис «Синхронизация с сервером» взаимодействуют друг с другом посредством межпроцессорной коммуникации. Косвенно взаимодействие между этими сервисам может быть реализовано через изменение базы данных. Поскольку прямое взаимодействие сервисов друг с другом увеличивает связанность кода, что в свою очередь негативно сказывается на модульности [26], данный способ коммуникации менее предпочтителен. В силу этих обстоятельств был выработан принцип проектирования [26], заключающийся в рекомендации использования косвенного взаимодействия между сервисами.

2.2.2 Проектирование базы данных

Для проектируемой информационной системы была разработана концептуальная модель предметной области (рисунок 9).

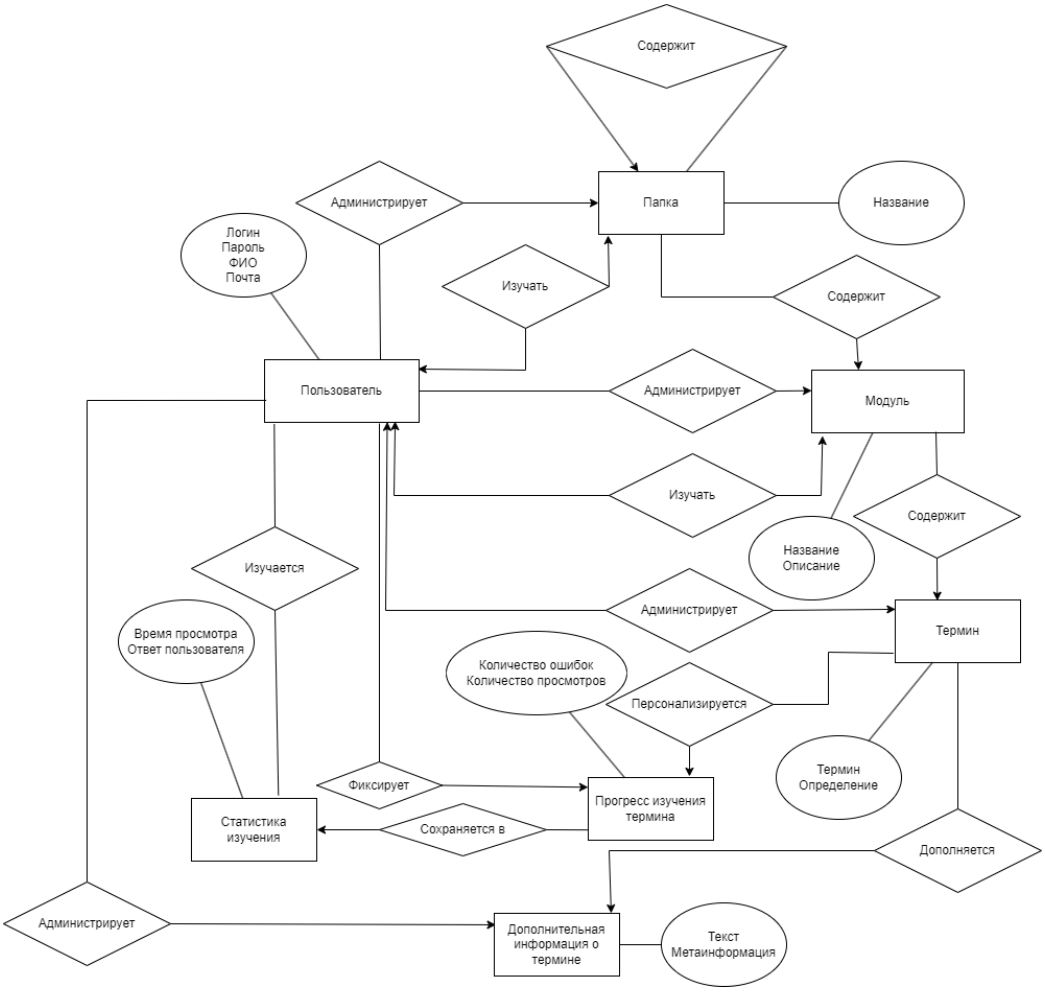


Рисунок 9 – Концептуальная модель предметной области

Из концептуальной модели видно, что сущность пользователя связана с сущностями «Папка», «Модуль», «Термин», «Дополнительная информация о термине» отношением «Администрирует» (один ко многим). Доступ на редактирование добавление и удаление данных этих сущностей имеет только пользователь, являющийся создателем этих экземпляров сущностей.

Также сущность «Пользователь» связана отношением «Изучать» (многие ко многим) с сущностями «Папка», «Модуль» и «Термин» (отношение «Изучать» реализуется через промежуточную сущность «Процесс изучения термина», которая отвечает за сохранение прогресса обучения пользователя). Данный тип отношения позволяет пользователю только просматривать данные сущностей.

Сущность «Процесс изучения термина» связана отношением «многие к одному» с сущностью «Статистика изучения». Роль данной сущности состоит в сохранении истории процесса обучения.

Сущность «Папка» содержит рекурсивное отношение «Содержит», которое обеспечивает возможность хранения вложенных папок.

На дальнейших этапах проектирования сущность «Дополнительная информация о термине» будет разбита по типам дополнительной информации (предложения и остальная дополнительная информация), что позволит сделать алгоритм обучения более удобным с точки зрения реализации.

2.2.2.1 Проектирование базы данных сервера

В процессе адаптирования концептуальной модели предметной области для реализации функциональности серверной части информационной системы была разработана логическая модель базы данных, представленная на рисунке 10.

Для обеспечения персонализации были добавлены сущности «Персональная информация о папке» и «Персональная информация о модуле», реализующие отношение «Изучается» (многие-ко-многим).

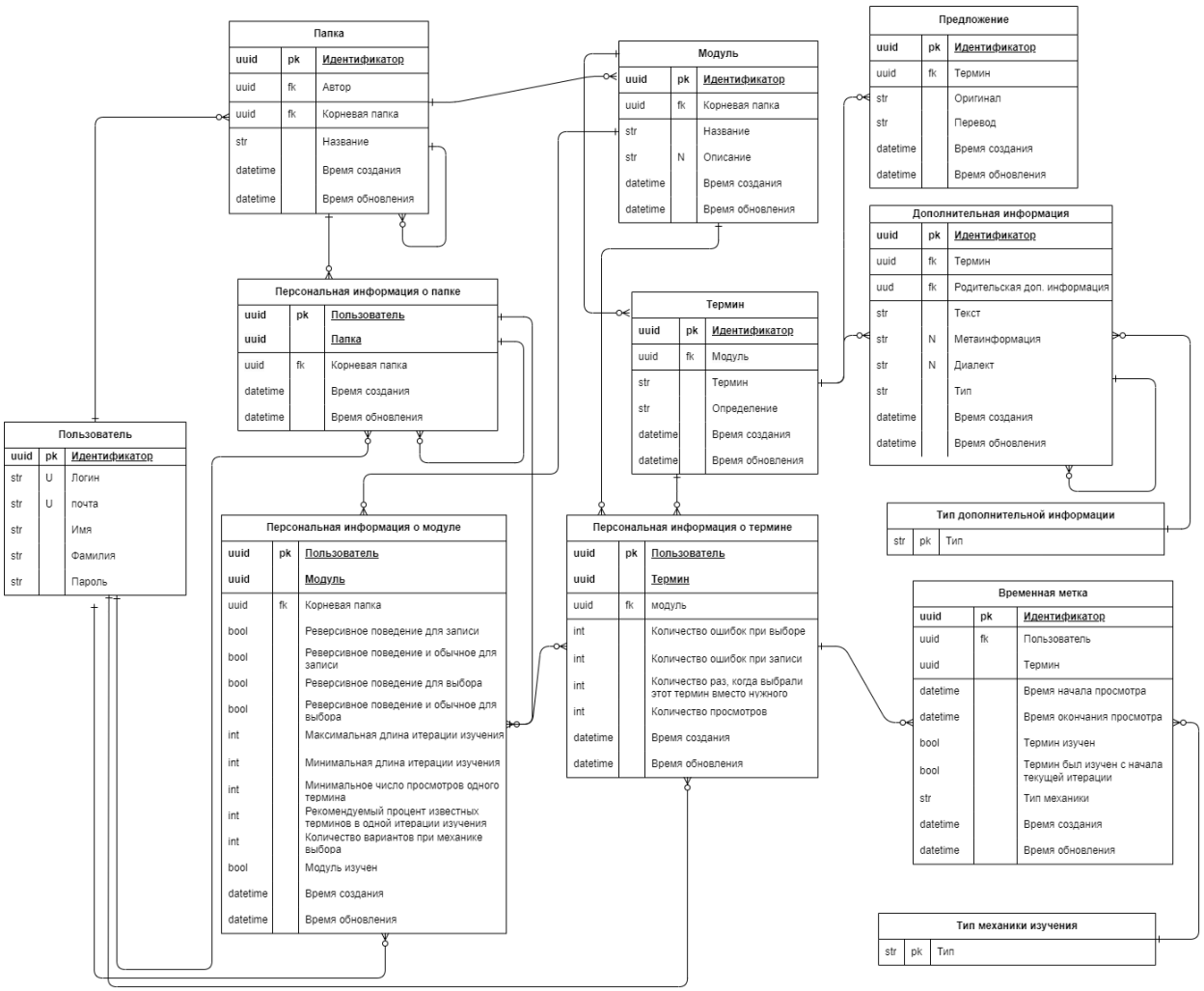


Рисунок 10 – Логическая модель базы данных сервера

Так же были добавлены перечисляемые типы данных для обозначения типа механики обучения и типа дополнительной информации.

2.2.2.2 Проектирование базы данных приложения

В процессе адаптирования концептуальной модели предметной области для реализации функциональности клиентской части информационной системы была разработана логическая модель базы данных, представленная на рисунке 11.

Мобильное приложение клиента, в отличии от серверной части, не предполагает большого наличия пользователей, поэтому было принято архитектурное решение не разделять сущности на персональную часть и общую часть с целью упрощения разработки и ведения таблиц базы данных.

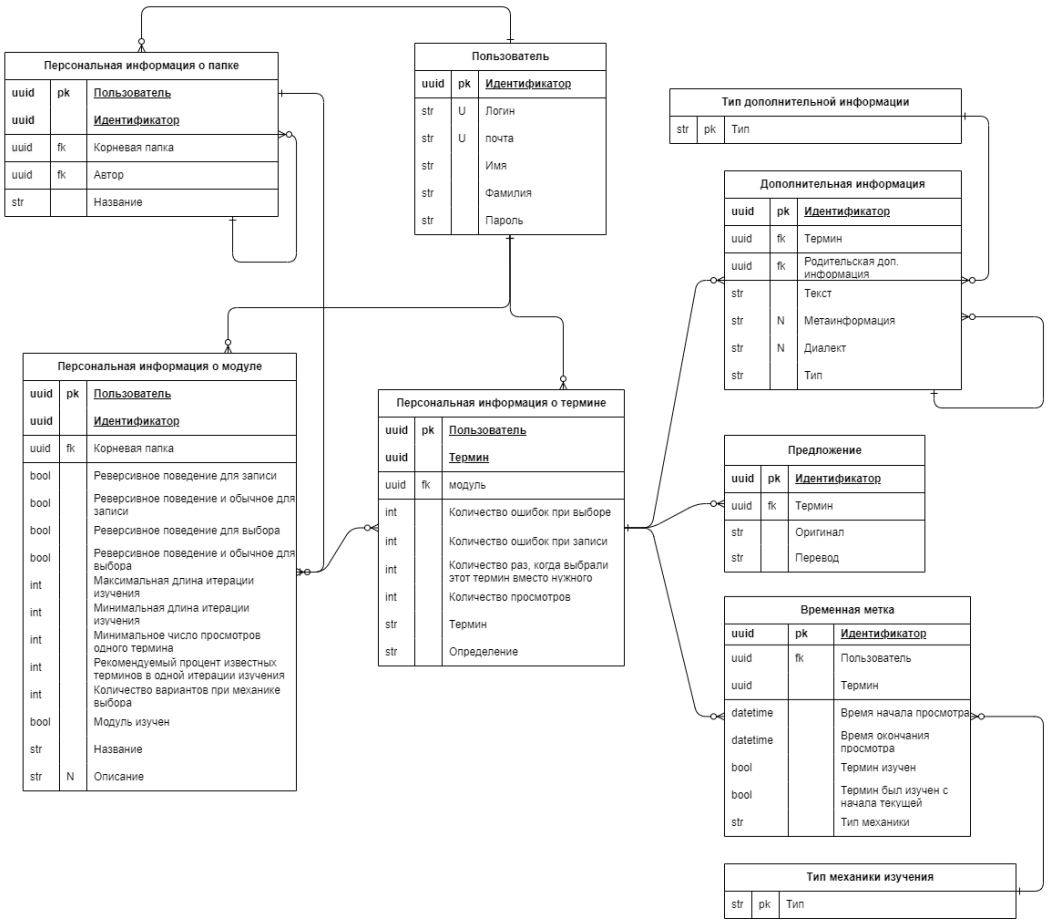


Рисунок 11 – Логическая модель БД клиента

Общая информация о сущностях была включена в таблицы с персональной информацией о процессе обучения.

2.2.3 Проектирование интерфейса

Проектирование интерфейса мобильного приложения осуществлялось в специализированном WEB-редакторе для конструирования flutter-интерфейсов Fluttervіz. Скриншот визуального интерфейса представлен на рисунке 12.

Fluttervіz является бесплатным Web-приложением, находящемся на этапе бетта-тестирования и представляет из себя визуальный конструктор пользовательского интерфейса. Данный редактор имеет ряд встроенных виджетов для организации структуры экрана мобильного приложения.

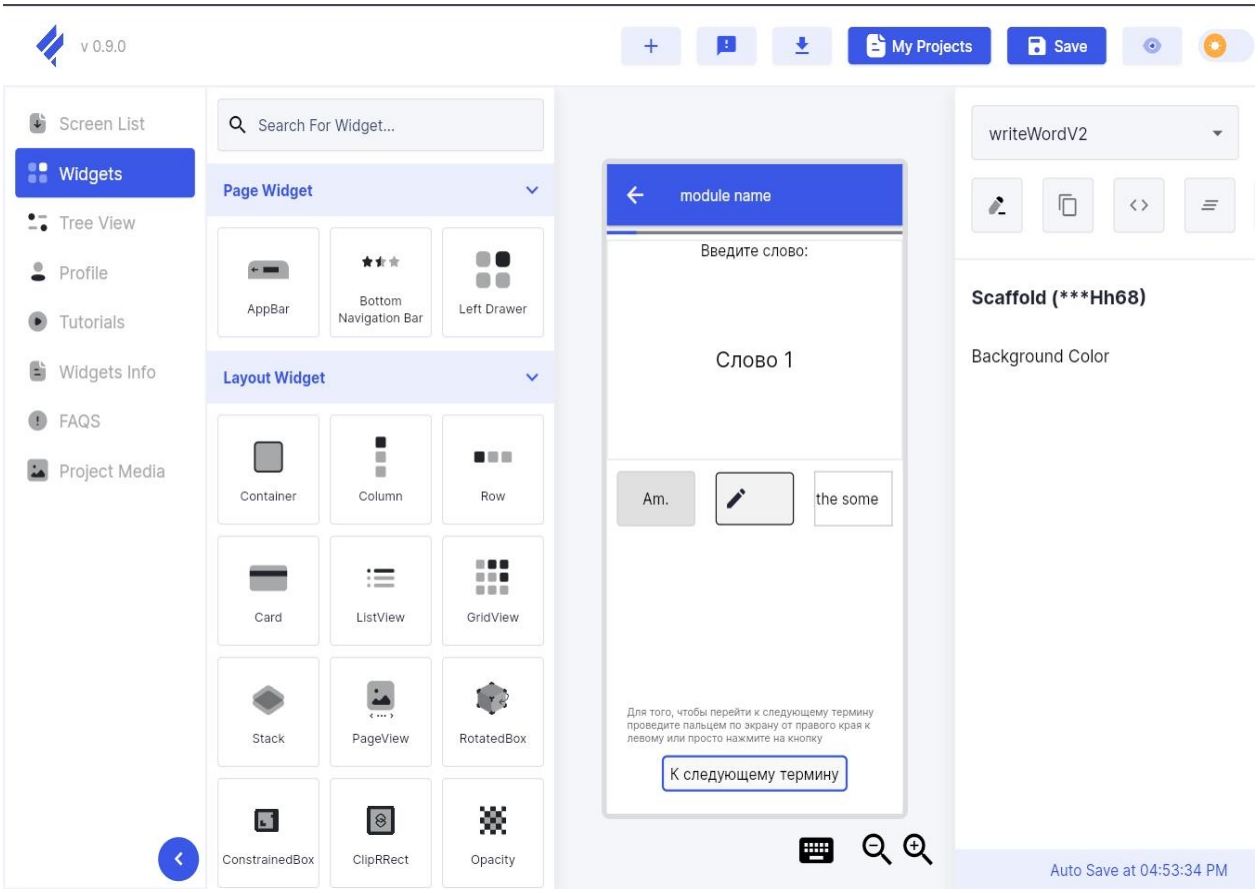


Рисунок 12 – Интерфейс редактора Fluttervіz

На основе сконструированного интерфейса Fluttervіz строит первичный код пользовательского интерфейса на языке Dart.

Скриншоты экранов пользовательского интерфейса в онлайн-редакторе Fluttervіz представлены в приложении Е.

2.2.4 Оценка бюджета на разработку проекта «Rem»

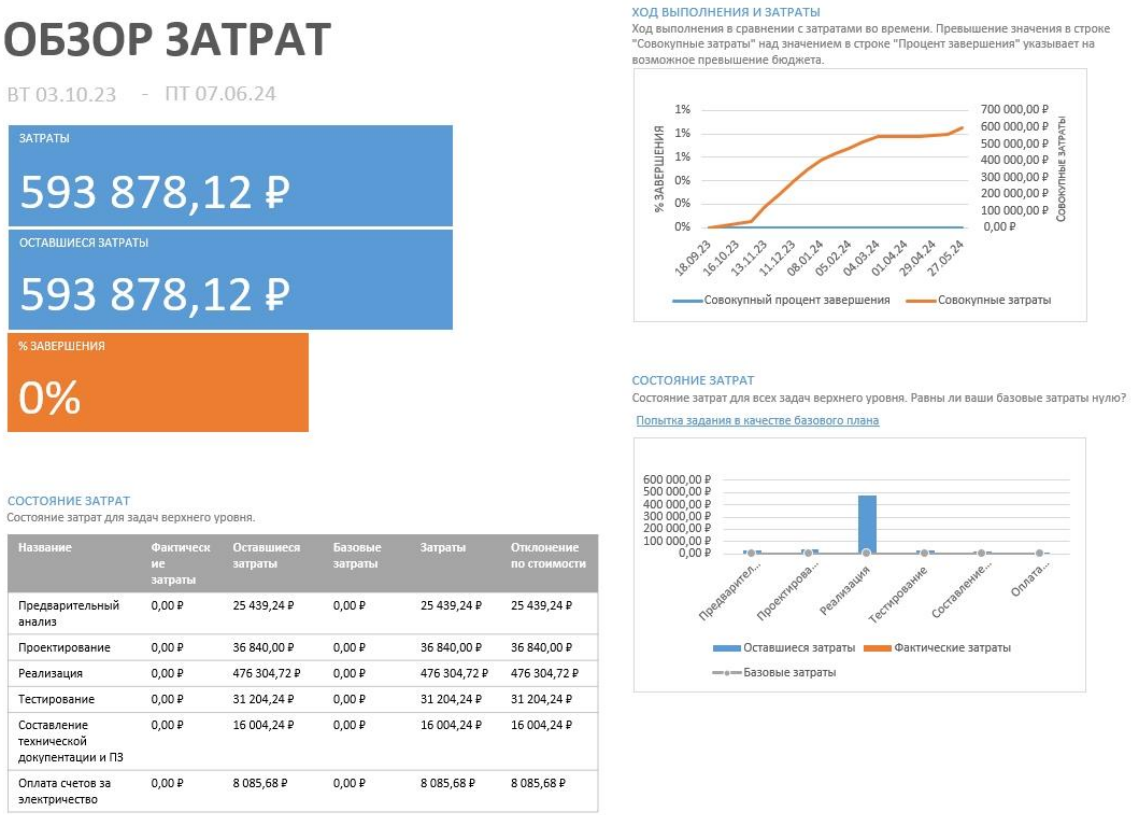
В ходе планирования разработки информационной системы была разработана диаграмма Ганта, представленная а приложении Д.

Было выявлено, что для реализации проекта потребуются следующие специалисты: аналитик, backend-разработчик, modile-разработчик, дизайнер и тестировщик (таблица 9).

Таблица 9 – Ставки специалистов

Название ресурса	Ставка (р/ч)
Аналитик	200
Backend-разработчик	312
Mobile-разработчик	375
Дизайнер	173
тестировщик	156

Обзор затрат на задачи представлен на рисунке 13.



Название	Фактические затраты	Оставшиеся затраты	Базовые затраты	Затраты	Отклонение по стоимости
Предварительный анализ	0,00 ₽	25 439,24 ₽	0,00 ₽	25 439,24 ₽	25 439,24 ₽
Проектирование	0,00 ₽	36 840,00 ₽	0,00 ₽	36 840,00 ₽	36 840,00 ₽
Реализация	0,00 ₽	476 304,72 ₽	0,00 ₽	476 304,72 ₽	476 304,72 ₽
Тестирование	0,00 ₽	31 204,24 ₽	0,00 ₽	31 204,24 ₽	31 204,24 ₽
Составление технической докупентации и ПЗ	0,00 ₽	16 004,24 ₽	0,00 ₽	16 004,24 ₽	16 004,24 ₽
Оплата счетов за электричество	0,00 ₽	8 085,68 ₽	0,00 ₽	8 085,68 ₽	8 085,68 ₽

Рисунок 13 – Обзор затрат на задачи

Стоимость разработки включает заработную плату исполнителям. Поскольку в первой части ВКРС исполнителем является автор ВКРС и не получает за разработку денежного вознаграждения, в силу этих причин затраты на выплату заработной платы не учитываются в разделе финансовой части.

Диаграмма Ганта представлена на рисунке 14.

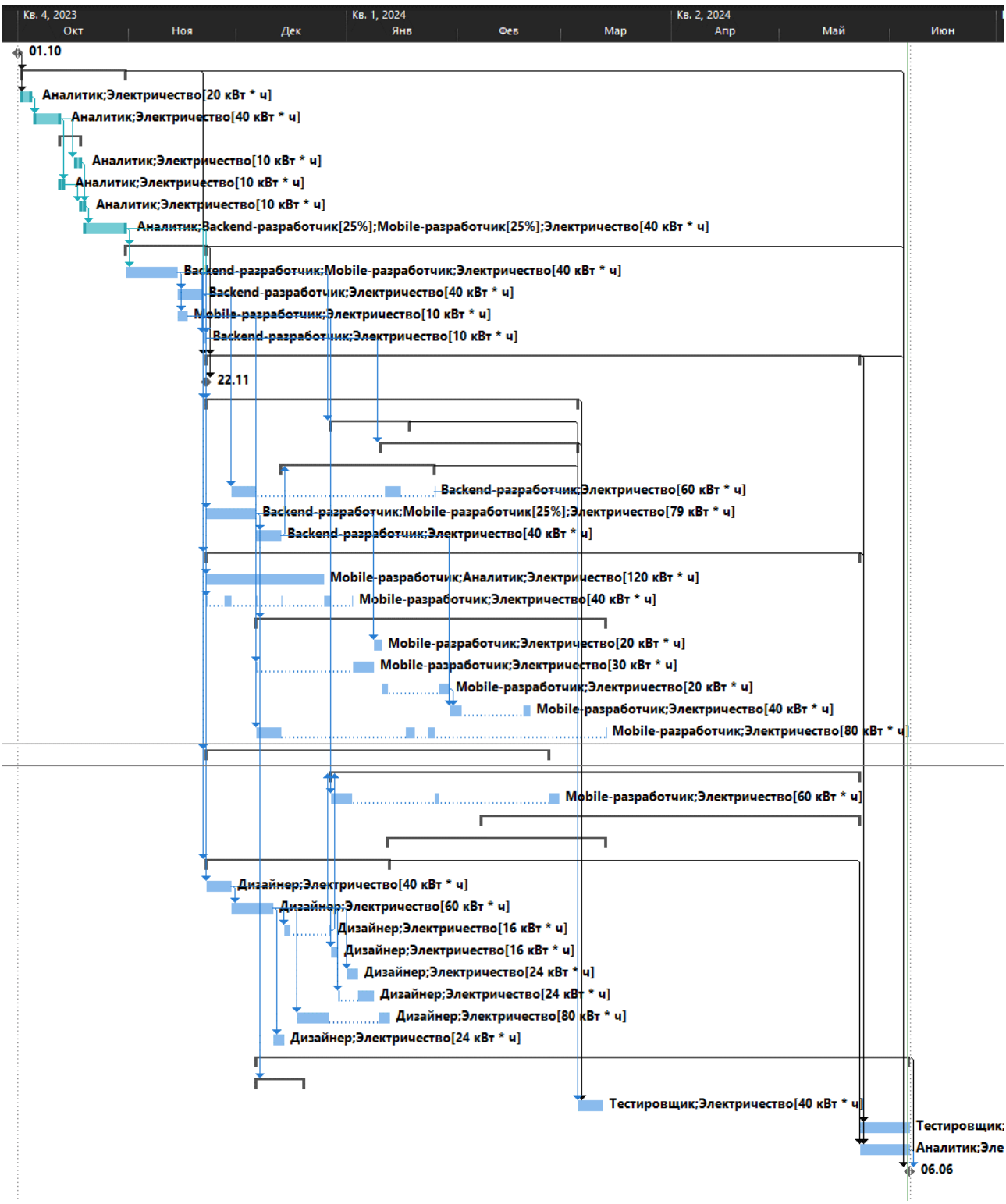


Рисунок 14 – Диаграмма Ганта

Обзор трудозатрат представлен на рисунке 15

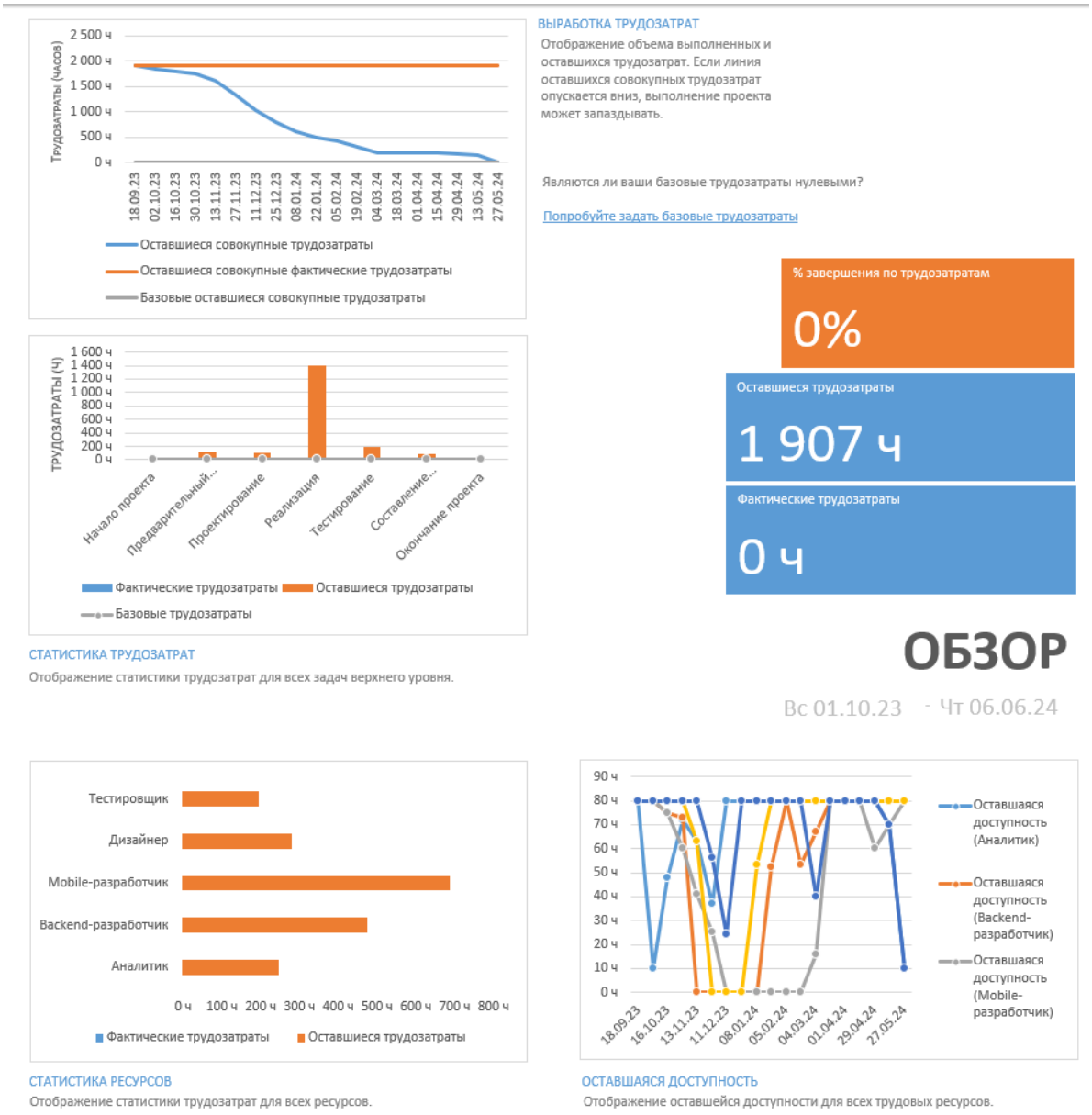


Рисунок 15 – Обзор трудозатрат

График выработки представлен на рисунке 16.

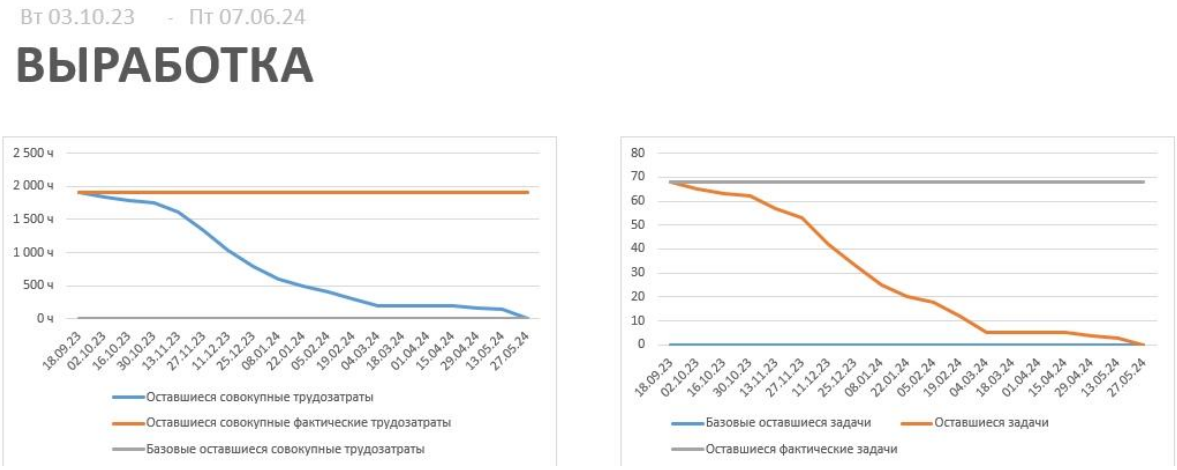


Рисунок 16 – Графики выработки

Обзор затрат ресурсов представлен на рисунке 17.

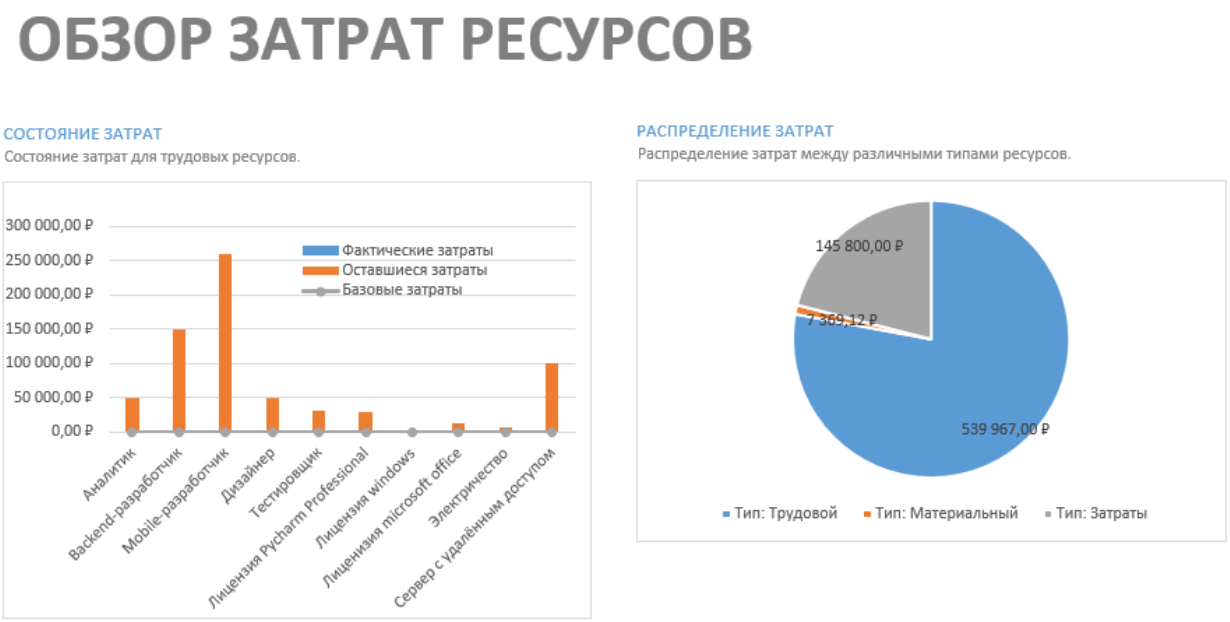


Рисунок 17 – Обзор затрат ресурсов

В таблице 10 приведены затраты на используемые ресурсы.

Таблица 10 – Затраты на ресурсы

Наименование ресурса	Затраты
Аналитик	50 000,00 ₽
Backend-разработчик	149 760,00 ₽
Mobile-разработчик	259 875,00 ₽
Дизайнер	49 132,00 ₽
Тестировщик	31 200,00 ₽
Лицензия Pycharm Professional	30 000,00 ₽
Лицензия windows	2 400,00 ₽
Лицензия microsoft office	13 400,00 ₽
Электричество	7 369,12 ₽
Сервер с удаленным доступом	100 000,00 ₽

Планируемая стоимость разработки составила 693 тысячи рублей, планируемая суммарная трудоемкость – 1907 часов.

2.3 Реализация информационной системы «Rem»

2.3.1 Реализация архитектуры информационной системы «Rem»

В рамках выполнения ВКР было произведено кодирование информационной системы.

2.3.1.1 Реализация клиента

Диаграмма развертывания для клиента представлена на рисунке 18.

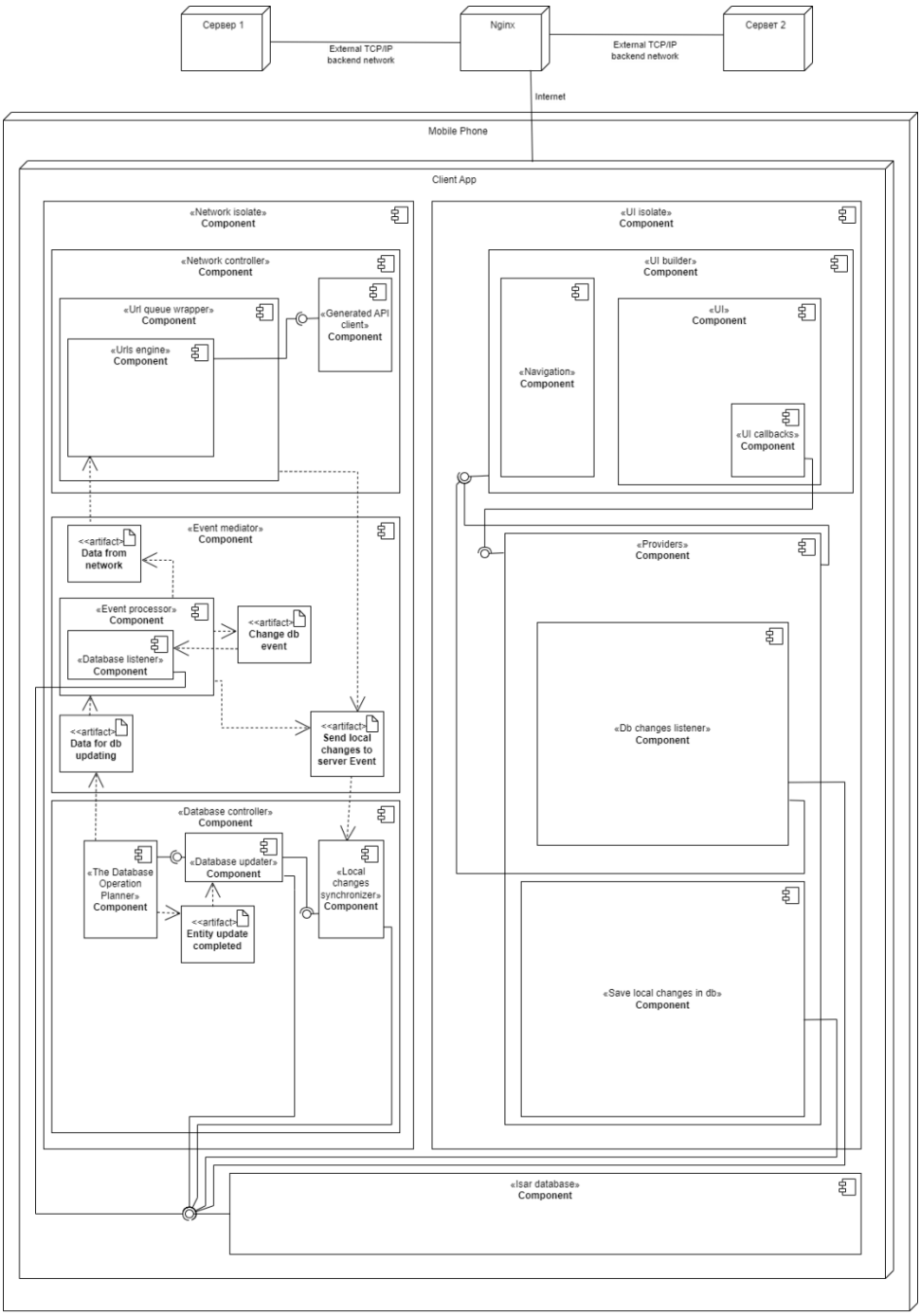


Рисунок 18 – Диаграмма развертывания мобильного приложения

Развертывание мобильного приложения осуществляется путем разархивирования «.apk» файла на устройство клиента.

Для реализации мобильного приложения был использован язык Dart. Основным доводом в пользу использования этого языка является кроссплатформенность.

Помимо мобильного приложения под управлением операционной системы Android разработанный код успешно компилируется в исполняемые файлы для операционных систем Windows и Unix-подобных. Помимо этого, Dart предоставляет возможность на основе написанного кода мобильного приложения собрать Web-приложение, которое будет открываться в браузере и не привязано к конкретной операционной системе. Так же возможно собрать исполняемый файл, который будет запускаться на iPhone и компьютерах с операционной системой Mac OS.

В качестве фреймворка, отвечающего за пользовательский интерфейс был использован Flutter – де-факто стандарт индустрии разработки мобильных приложений на Dart.

Для реализации базы данных была использована библиотека Isar, взаимодействие с сервером осуществлялось с использованием библиотеки Dio.

Для каждой конечной точки сервера был автоматически сгенерирован код отправки запроса и приема возвращаемых данных средством openapi-generator, которое на основе предоставленной Swagger-документации генерирует функции http-клиента.

Разработка мобильного приложения производилась в среде разработки Android Studio.

Наиболее значимые фрагменты разработанного кода приведены в приложении М. Полный код клиента размещен на GitHub в закрытой организации [29].

2.3.1.2 Реализация сервера

Для реализации серверной части информационной системы был использован язык программирования Python 3.11, основным достоинством которого является высокая скорость прототипирования. При разработке невысоко-нагруженного сервера низкая скорость работы языка не несет в себе серьезных проблем, поскольку основная часть времени выполнения одной бизнес-операции (пользовательского запроса) приходится на обращение к внешним (относительно интерпретатора python) сервисам (к клиенту, к базе данных), которое осуществляется асинхронно.

В качестве фреймворка для реализации сервера был использован Fast-API, принципами которого является скорость работы сервера и удобство использования. Еще одним преимуществом данного фреймворка является наличие модуля генерации автоматической интерактивной документации Swagger (в формате OpenAPI 3.1.0), которая производится на основе реализованных в программе конечных точек API.

Движок Fast-api создает ASGI приложение, которое может быть запущено на серверном движке uvicorn. Использование менеджера процессов gunicorn (запускающего uvicorn инстансы) позволяет обойти ограничения Python GIL, запрещающее запуск более одного потока Python и позволяющее использовать все процессорные ядра сервера.

В качестве ORM была использована SQLAlchemy 2, предоставляющая возможность выполнять асинхронные операции взаимодействия с базой данных. Преимуществом использования этой библиотеки является наличие интеграции с системой автоматических миграций структуры БД alembic, которая осуществляет по запросу пользователя (через CLI интерфейс) миграции на основе моделей данных, реализованных в коде.

Alembic отслеживает изменения в ORM-классах SQLAlchemy, путем сопоставления схемы данных с текущей структурой БД и автоматически генерирующая миграции структуры базы данных. В alembic предусмотрена так же возможность отката миграций базы данных, реализующаяся за счет

того, что при составлении файла миграции alembic генерирует инструкции для возврата базы данных в исходное состояние.

При реализации микросервисной архитектуры наиболее сложным и трудозатратным является реализация обеспечения согласованности данных между различными архитектурными квантами. Классическим вариантом обеспечения согласованности данных является применение паттерна SAGA.

Поскольку на этапе проектирования было принято архитектурное решение о создании информационной системы, наиболее приближенной по архитектуре к микросервисам, был реализован механизм обеспечения согласованности данных, для реализации которого был выбран стиль инъекции зависимостей (DI).

В качестве языковых механизмов для реализации системы DI были выбраны аннотации Python и класс Depends библиотеки Fast-API. Пример использования разработанного механизма приведен в листинге 1.

Листинг 1 – Пример использования DI при реализации паттерна SAGA

```
@module_app.post("/personalize/create")
async def create_personalize_module(
        new_personal_module: Annotated[
            CreateOnlyPersonalizePartModuleDTO,
            Depends(
                CheckDbEntitiesExistDep2[
                    CreateOnlyPersonalizePartModuleDTO,
                    lambda i, **k: {"folder_id": i.root_folder_id,
"current_user": k["current_user"]},
                    lambda i, **k: i.root_folder_id is None or
k["from_folder_microservice"] is True,
                    {"current_user": Annotated[UserDTO,
Depends(get_current_user)],
                     "from_folder_microservice":
from_folder_microservice_annotation}
                ](FolderAPIConfig, check_personalize_folder, )
            )
        ]
):
      pass
```

Функция create_personalize_module реализует добавление персональной информации к модулю для конкретного пользователя. За механизм DI отвечает класс Fast-API Depends, на вход которому передается класс CheckDbEntitiesExistDep2, реализующий функциональность в рамках

паттерна SAGA и отвечающий за проверку существования родительских сущностей в базах данных, расположенных в других архитектурных квантах.

В качестве дженериков классу CheckDbEntitiesExistDep2 передаются:

– модель данных, передаваемых клиентом для осуществления запрошенного функционала, на основе которой внутренние механизмы класса CheckDbEntitiesExistDep2 получат переданную пользователем информацию из конкретного запроса через внутренние механизмы Fast-API;

– функция получения обязательных аргументов из пользовательского запроса, необходимых для получения информации от микросервиса папок о существовании родительской папки (или отсутствии доступа к родительской папке);

– функция, определяющая условие, при котором обращение к микросервису папок не будет произведено (для случаев, когда пользователь не указал родительскую папку, а добавил модуль в корень каталога);

– функция, возвращающая дополнительные параметры из текущего запроса пользователя (к примеру, токен авторизации пользователя).

Для создания объекта, вызывающегося при работе механизма DI, классу CheckDbEntitiesExistDep2 передаются в качестве аргументов два параметра: класс-конфигуратор API взаимодействия с целевым микросервисом и функция-обертка для вызова нужной конечной точки, созданной путем применения подхода meta-программирования инструментом openapi-generator, использующим Swagger-документацию для автоматической генерации кода http-python-клиента.

С целью продемонстрировать возможности разработанной информационной системы к масштабированию и гибкому выделению архитектурных квантов серверная часть информационной системы для развертывания была разбита на два архитектурных кванта: архитектурный квант авторизации и архитектурный квант обучения.

Диаграмма развертывания архитектурного кванта авторизации представлена на рисунке 19.

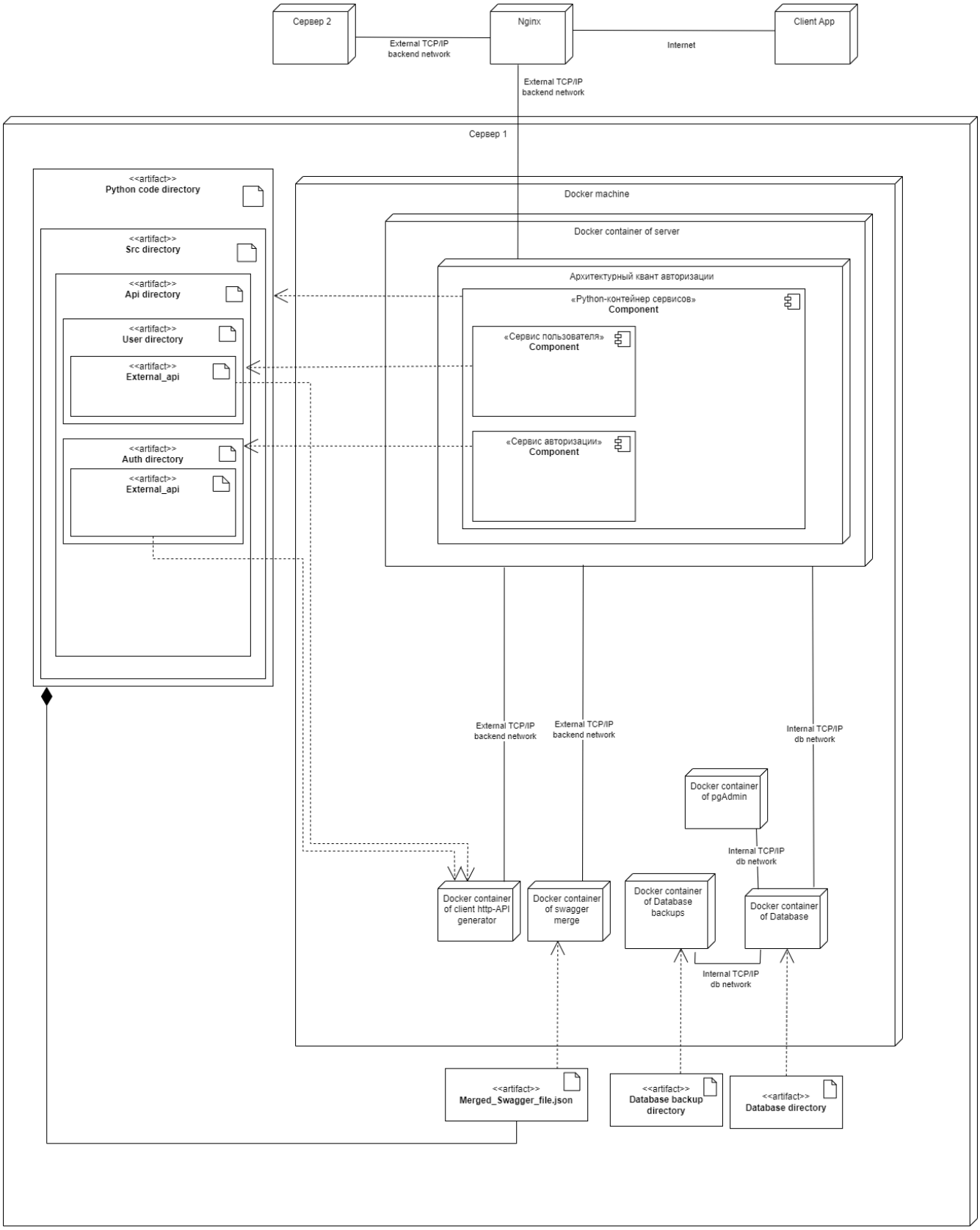


Рисунок 19 – Диаграмма развертывания архитектурного кванта авторизации

Диаграмма развертывания архитектурного кванта обучения представлена на рисунке 20.

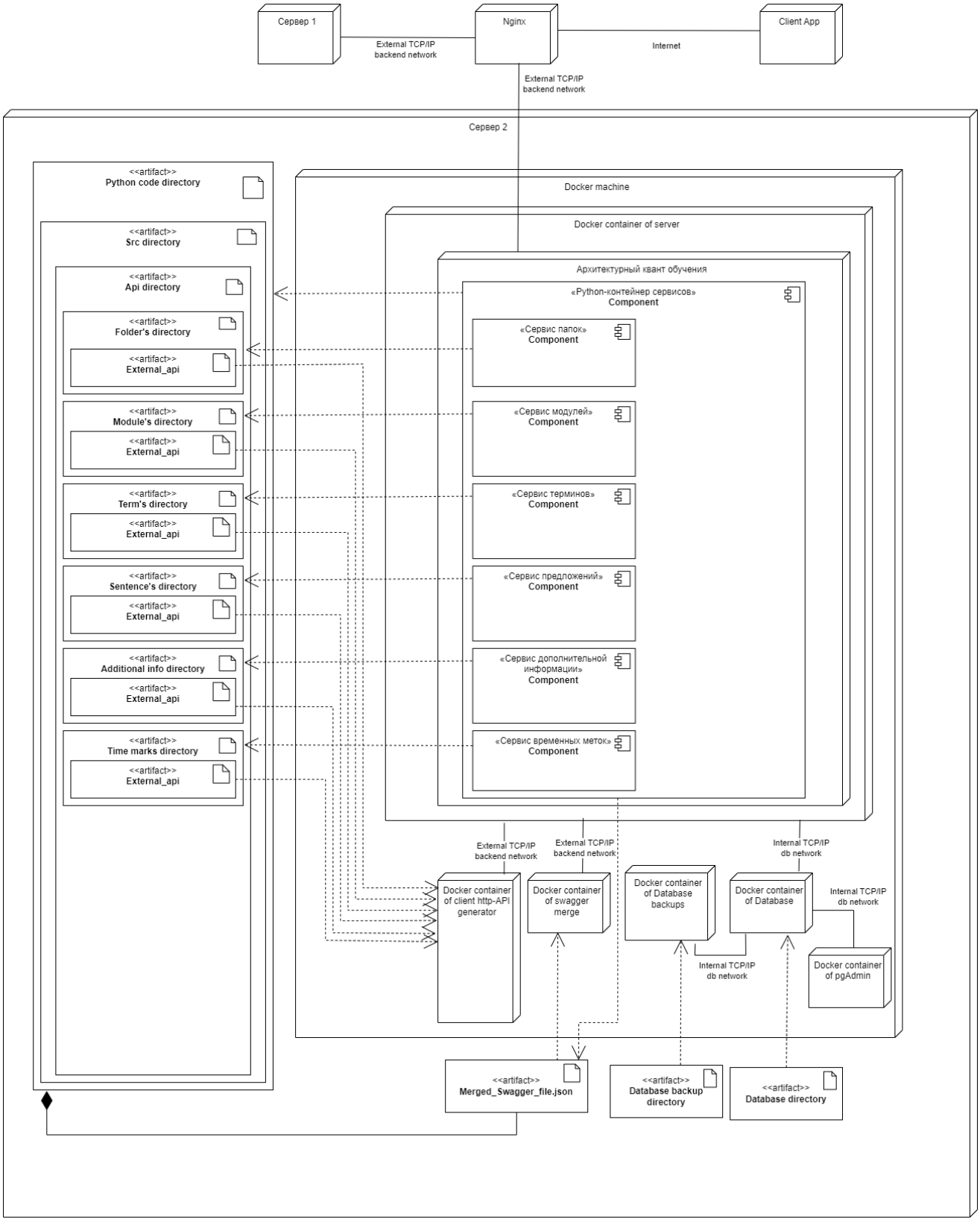


Рисунок 20 – Диаграмма развертывания архитектурного кванта обучения

С целью обеспечения кроссплатформенности и увеличения скорости развертки инфраструктуры информационной системы была использована технология контейнеризации docker, для интеграции которой были созданы docker-образы для:

– python-сервера (для каждого архитектурного кванта компоновка docker образа python-сервера может отличаться, ведь каждый архитектурный квант может содержать разный набор сервисов);

– docker-образ базы данных PostgreSQL;

– docker-образ системы администрирования баз данных pgAdmin;

– docker-образ системы резервного копирования базы данных.

Docker-образ сервера взаимодействует по внутренней сети с контейнером базы данных PostgreSQL. Взаимодействие по внутренней сети предотвращает получение несанкционированного доступа к базе данных и позволяет обезопасить данные сервера.

Осуществление резервного копирования данных происходит раз в сутки и выполняется специальным контейнером, взаимодействующим с базой данных по внутренней сети.

В случае развертывания инстанса на новой машине, помимо основных контейнеров, перечисленных выше, запускаются:

– контейнер, генерирующий для каждого из сервисов python-код API-клиента для взаимодействия с сервисами, расположенными в других архитектурных квантах;

– контейнер, объединяющий Swagger-документацию всех сервисов архитектурного кванта в единый Swagger файл.

Наиболее значимые фрагменты разработанного кода приведены в приложении Л. Полный код сервера размещен на GitHub в закрытой организации [30].

2.3.2 Реализация базы данных информационной системы «Rem»

2.3.2.1 Реализация серверной базы данных

В процессе разработки серверной базы данных была спроектирована физическая модель базы данных (рисунок 21).

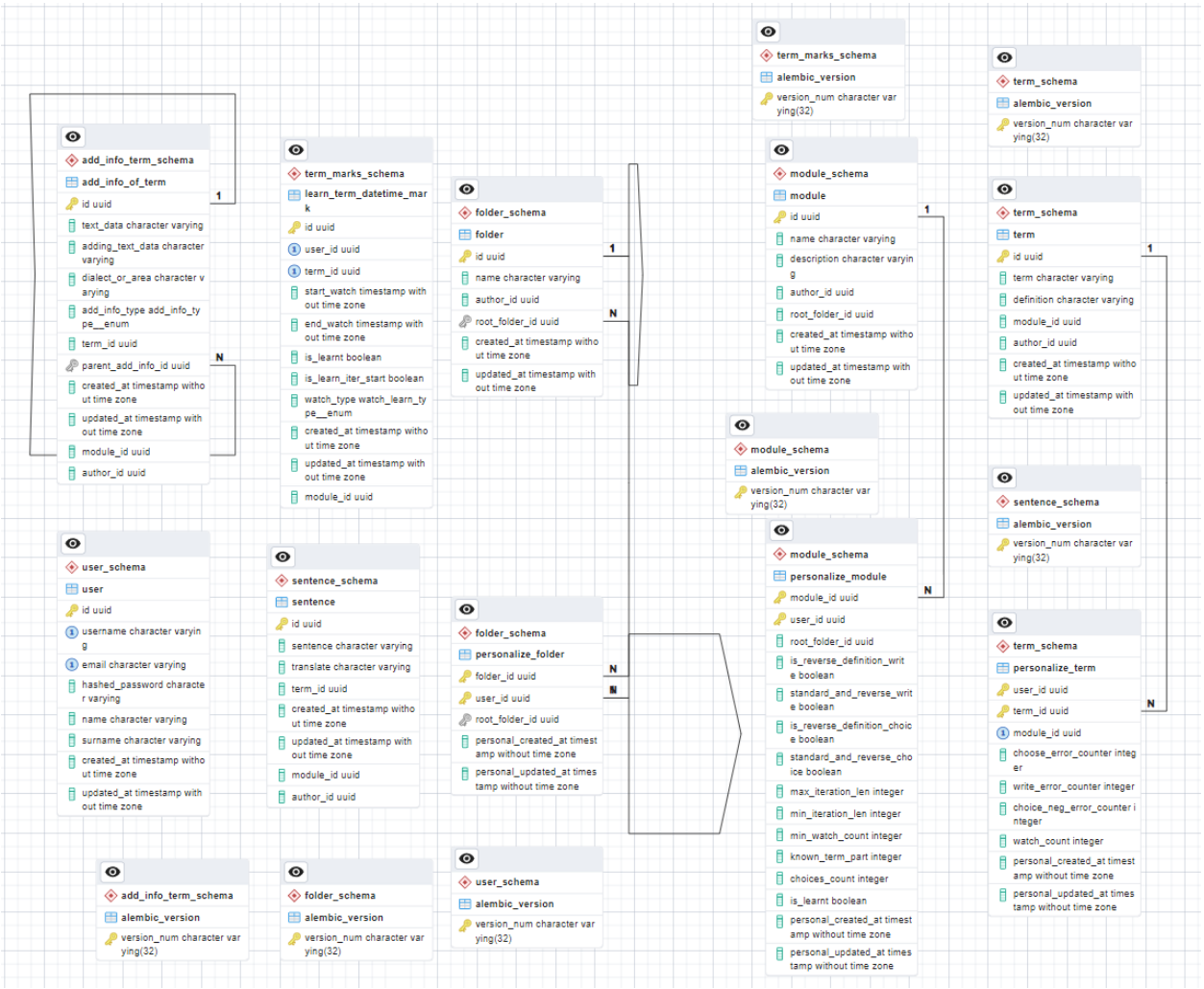


Рисунок 21 – Физическая модель БД, сгенерированная средствами pgAdmin

Поскольку было принято архитектурное решение разрабатывать информационную систему наиболее приближенную к микросервисной архитектуре, логическая модель базы данных была разбита по схемам базы данных в соответствии с делением на сервисы. Поскольку каждая их схем базы данных может быть расположена на отдельном физическом сервере, между таблицами, расположенными в разных схемах, не должно быть связей.

Целостность данных будет обеспечиваться на уровне взаимодействия сервисов, а не на уровне базы данных.

2.3.2.2 Реализация базы данных мобильного приложения

В процессе реализации мобильного приложения была спроектирована физическая модель базы данных (рисунок 22).

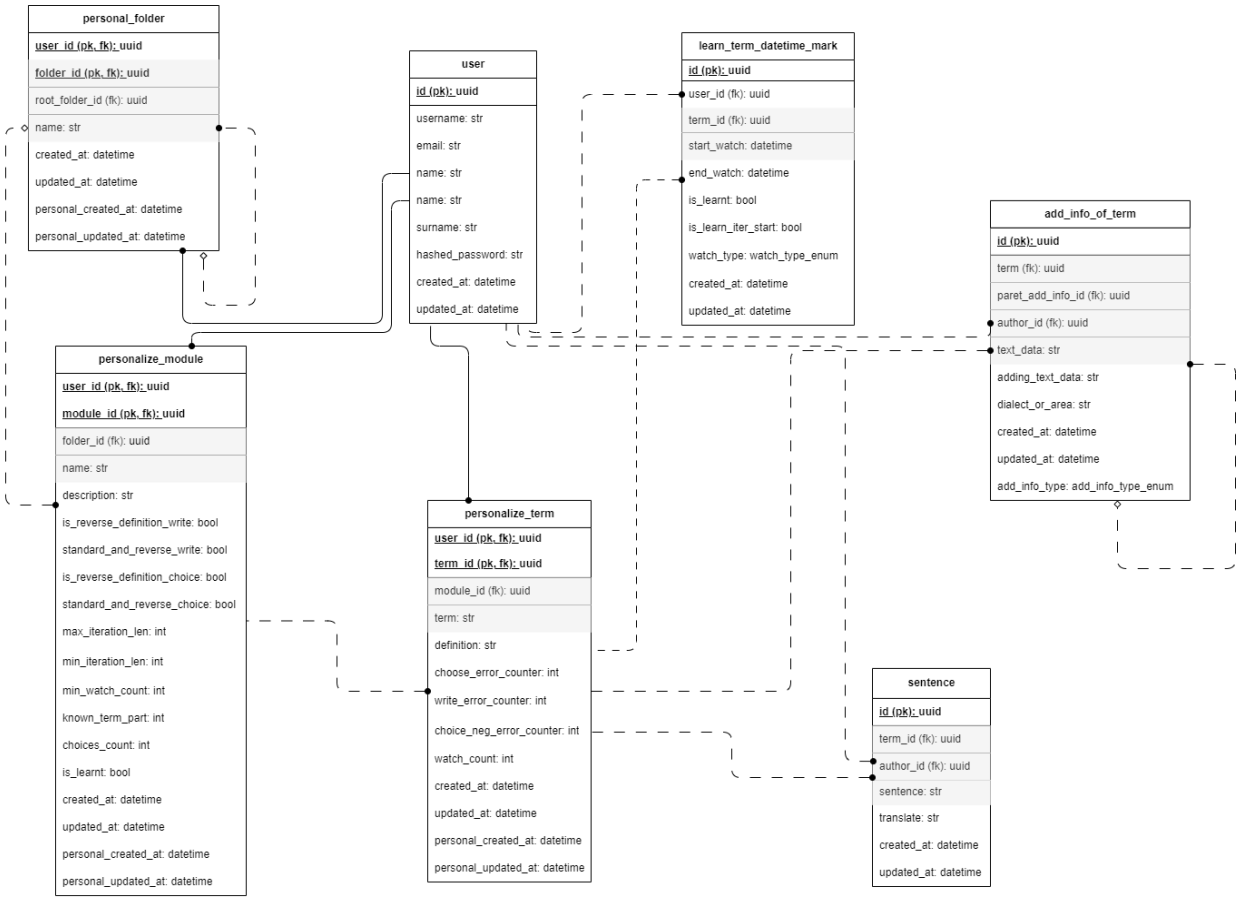


Рисунок 22 – Физическая модель БД клиента

Сущности «Тип механики изучения» и «Тип дополнительной информации» были представлены в качестве перечислений.

2.3.3 Реализация пользовательского интерфейса проекта «Rem»

В ходе разработки мобильного приложения был реализован пользовательский интерфейс для клиентской части информационной системы.

На рисунке 23 приведен скриншот экрана модуля, на котором отображаются: название модуля, прогресс изучения (синим цветом на дугообразной шкале достижений отмечен процент просмотренных терминов,

желтым – процент терминов, для которых механика выбора была успешно пройдена нужное количество раз, зеленым – процент терминов, полностью освоенных пользователем).

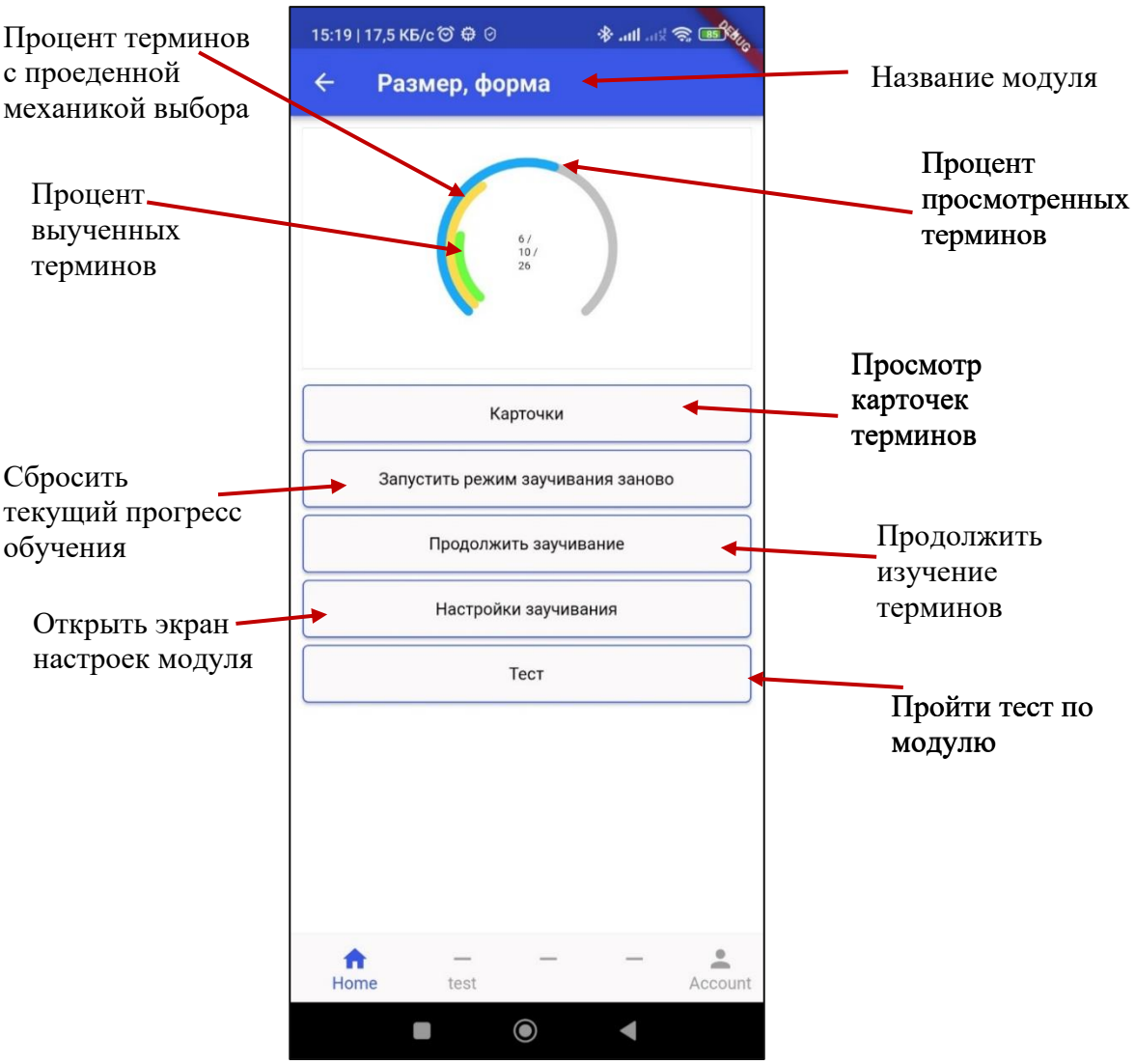


Рисунок 23 – Главный экран лексического модуля

Ниже расположено меню навигации по лексическому модулю, в нем присутствуют:

– кнопка показа карточек терминов;

– кнопка сброса текущего прогресса обучения;

– кнопка продолжения режима изучения;

– кнопка перехода в режим настроек;

– и кнопка тестирования по пройденному материалу.

Другие экраны пользовательского интерфейса представлены в приложении Ж.

2.3.4 Тестирование информационной системы «Rem»

2.3.4.1 Функциональное тестирование серверной части

Для проведения функционального тестирования воспользуемся классами эквивалентности. «Выделяя классы эквивалентности, мы заявляем, что если на одном элементе класса не находится ошибка, то маловероятно, что она найдется на любом другом представителе этого класса» [31 стр. 52].

Классы эквивалентности являются эвристикой, помогающей систематизировать написание тест кейсов [32].

Выделим классы эквивалентности сущностей информационной системы: папки, модули, термины, предложения, дополнительная информация, временные метки. Каждый из этих классов эквивалентен всем остальным, поскольку информационная система предоставляет доступ к одинаковым операциям (создание, обновление и получение данных) и является частью первой группы эквивалентности (ГЭ1).

Выделим группы эквивалентных классов из ГЭ1 на основе операций, выполняемых над данными:

– ГЭ2 – группа эквивалентных классов, содержащих операции создания сущностей;

– ГЭ3 – группа эквивалентных классов, содержащих операции обновления сущностей;

– ГЭ4 – группа эквивалентных классов, содержащих операции получения сущностей.

Для каждой группы эквивалентности выделим ограничения на входные данные:

а) ГЭ1:

1) пользователь должен быть авторизован;

2) поля идентификаторов должны быть валидными UUID;

3) данные, передаваемые в теле запроса, должны иметь формат json;

4) строки, передаваемые в теле запроса, должны иметь двойные кавычки в начале и в конце строки;

5) параметры, имеющие целочисленный тип, должны быть валидными целыми числами;

6) параметры, имеющие вещественный тип, должны быть валидными вещественными числами, представленными в виде десятичной дроби (в качестве символа-разделителя между целой и дробной частью должна использоваться точка);

7) параметры, имеющие логический тип, должны принимать одно из двух значений: false или true;

8) параметры, имеющие тип даты и времени должны содержать строку в формате ISO 8601 [33] без часового пояса (время должно передаваться по Гринвичу);

9) необязательные параметры могут быть не переданы, переданы со значением null или переданы со значением соответствующего им типа;

б) ГЭ2:

1) передаваемый идентификатор новой сущности должен быть: null, не передан или не совпадать ни с одним из уже существующих идентификаторов той же сущности;

2) передаваемые идентификаторы родительских сущностей должны соответствовать сущностям, уже существующим в базе данных;

3) идентификатор пользователя, выполняющего операцию, должен совпадать с идентификатором, указанном в поле автора, для каждой родительской сущности.

Результаты функционального тестирования второй группы эквивалентности (ТЭ2) представлены в таблице 11.

Таблица 11 – Результаты функционального тестирования ГЭ2

№ теста	Название теста	Входные данные	Ожидаемый результат	Полученный результат	Вывод
Класс эквивалентности /term/create					

1	создание термина без указания идентификатора	{
		"term": "string",
		"definition": "string",
		"module_id": "3fa85f64-5717-4562-b3fc-2c963f66afa6"
		}
			термин будет создан	Представлен на листинге И.1	тест пройден

2	создание термина с указанием null в качестве идентификатора	{
		"term": "string",
		"definition": "string",
		"id": null,
		"module_id": "3fa85f64-5717-4562-b3fc-2c963f66afa6"
		}
			термин будет создан	представлен на листинге И.1	тест пройден

3	создание термина с указанием несуществующего в бд идентификатора	{
		"term": "string",
		"definition": "string",
		"id": "3fa85f64-5717-4562-b3fc-2c963f66afa6",
		"module_id": "3fa85f64-5717-4562-b3fc-2c963f66afa6"
		}
			термин будет создан	представлен на листинге И.1	тест пройден

4	создание термина без указания родительского идентификатора	{
		"term": "string",
		"definition": "string"
		}
			программа сообщит об отсутствии обязательного поля, термин не будет создан	представлен на листинге И.2	тест пройден

5	создание термина без указания одного из обязательных полей	{
		"term": "string",
		"definition": "string",
		"module_id": "3fa85f64-5717-4562-b3fc-2c963f66afa6"
		}
			программа сообщит об отсутствии обязательного поля, термин не будет создан	представлен на листинге И.3	тест пройден

6	создание термина с указанием несуществующего в бд идентификатора модуля	{
		"term": "string",
		"definition": "string",
		"module_id": "3fa85f64-5717-4562-b3fc-2c963f66afa6"
		}
			программа сообщит об ошибке, термин не будет создан	представлен на листинге И.4	тест пройден

7	создание термина с указанием идентификатора модуля, идентификатор автора которого не совпадает с идентификатором пользователя	{
		"term": "string",
		"definition": "string",
		"module_id": "e22ef047-9165-4a56-a897-c2fc6f8ece3c"
		}
			программа сообщит об ошибке, термин не будет создан	представлен на листинге И.4	тест пройден

8	создание термина неавторизированным пользователем	{
		"term": "string",
		"definition": "string",
		"module_id": "3fa85f64-5717-4562-b3fc-2c963f66afa6"
		}
			программа сообщит об ошибке доступа к указанной операции	представлен на листинге И.5	тест пройден

а) ГЭЗ:

1) передаваемый идентификатор сущности должен соответствовать одной из сущностей, уже существующих в системе;

2) идентификатор пользователя должен быть равен идентификатору, записанному в поле автора сущности с переданным идентификатором.

Результаты функционального тестирования ГЭЗ (таблица 12).

Таблица 12 – Результаты функционального тестирования ГЭЗ

№ теста	Название теста	Входные данные	Ожидаемый результат	Полученный результат	Вердикт
Класс эквивалентности /term/update/{term_id}					

9	обновление термина	term_id: 0bd1f733-87ab-4bdf-aa62-87be0ee7f83c
		{
		"term": "updated_term",
		"definition": "string",
		"updated_at": "2024-05-31T14:15:02.991Z"
		}
			программа обновит термин	представлен на листинге И.6	тест пройден

10	обновление термина с временем обновления большим, чем текущее	term_id: 0bd1f733-87ab-4bdf-aa62-87be0ee7f83c
		{
		"term": "updated_term",
		"definition": "string",
		"updated_at": "2025-05-31T14:15:02.991Z"
		}
			программа сообщит о некорректном времени обновления	представлен на листинге И.7	тест пройден

11	обновление термина с полем обновления меньшим, чем установлено при последнем обновлении	term_id: 0bd1f733-87ab-4bdf-aa62-87be0ee7f83c
		{
		"term": "string",
		"definition": "string",
		"updated_at": "2023-06-01T07:34:13.442Z"
		}
			программа не произведет обновления	представлен на листинге И.8	тест пройден

12	обновление термина с несуществующим идентификатором	term_id: 0bd1f733-87ab-4bdf-aa62-87be0ee7f83a
		{
		"term": "string",
		"definition": "string",
		"updated_at": "2024-06-01T07:34:13.442Z"
		}
			программа сообщит о том, что термин не был найден	представлен на листинге И.9	тест пройден

13	обновление термина с идентификатором автора, не совпадающим с идентификатором текущего пользователя	term_id: 75ced4be-38d2-42b1-a03b-9347d0f556b5
		{
		"term": "string",
		"definition": "string",
		"updated_at": "2024-06-01T07:34:13.442Z"
		}
			программа сообщит об ошибке доступа	представлен на листинге И.10	тест пройден

а) ГЭ4:

1) список идентификаторов сущностей, из которых необходимо получить дочерние сущности должен содержать хотя бы один элемент;

2) поле limit должно содержать значение от 1 до 100 включительно;

3) поле offset должно содержать неотрицательное значение;

4) поле updated_at должно содержать либо null либо значение, меньшее текущего времени.

Результаты функционального тестирования ГЭ5 приведены в таблице 13.

Таблица 13 – Результаты функционального тестирования

№ теста	Название теста	Входные данные	Ожидаемый результат	Полученный результат	Вердикт
Класс эквивалентности /term/all					
14	получение всех терминов из одного модуля	author_only: false limit: 3 offset: 0 module_ids: [06d677a2-f66e-40c9-8d89-75425cddf211]	программа выведет три (или меньше) модуля	представлен на листинге И.11	тест пройден
15	получение всех терминов из одного модуля	author_only: false limit: 10 offset: 0 module_ids: [06d677a2-f66e-40c9-8d89-75425cddf211]	программа выведет 10 (или меньше) модулей	представлен на листинге И.12	тест пройден
16	получение всех терминов из одного модуля	author_only: false limit: 100 offset: 0 module_ids: [06d677a2-f66e-40c9-8d89-75425cddf211]	программа выведет 10 (или меньше) модулей	представлен на листинге И.13	тест пройден
17	получение всех терминов из одного модуля	author_only: false limit: 3 offset: 3 module_ids: [06d677a2-f66e-40c9-8d89-75425cddf211]	программа выведет вторую тройку модулей	представлен на листинге И.14	тест пройден
17	получение всех терминов из одного модуля	author_only: false limit: 6 offset: 1 module_ids: [06d677a2-f66e-40c9-8d89-75425cddf211]	программа выведет шесть или меньше модулей	представлен на листинге И.15	тест пройден
18	получение всех терминов из одного модуля	author_only: false limit: 10 offset: 90 module_ids: [06d677a2-f66e-40c9-8d89-75425cddf211]	программа выведет десятую десятку модулей, или меньше	представлен на листинге И.16	тест пройден

Продолжение таблицы 13

№ теста	Название теста	Входные данные	Ожидаемый результат	Полученный результат	Вердикт
19	получение всех терминов из одного модуля	author_only: false limit: 10 offset: 30 module_ids: [06d677a2-f66e-40c9-8d89-75425cddf211]	программа выведет четвертую десятку модулей, или меньше	представлен на листинге И.17	тест пройден
20	получение всех терминов из одного модуля	author_only: false limit: 10 offset: 30 module_ids: [06d677a2-f66e-40c9-8d89-75425cddf211, 06d677a2-f66e-40c9-8d89-75425cddf211]	программа выведет четвертую десятку модулей, или меньше. Результат будет аналогичен тесту 19, количество терминов не увеличится	представлен на листинге И.18	тест пройден
21	получение всех терминов из одного модуля	author_only: false limit: 10 offset: 30 module_ids: [06d677a2-f66e-40c9-8d89-75425cddf211, 06d677a2-f66e-40c9-8d89-75425cddf211, 06d677a2-f66e-40c9-8d89-75425cddf211, 06d677a2-f66e-40c9-8d89-75425cddf211, 06d677a2-f66e-40c9-8d89-75425cddf211,]	программа выведет четвертую десятку модулей, или меньше. Результат будет аналогичен тесту 19, количество терминов не увеличится	представлен на листинге И.19	тест пройден
22	получение всех терминов из двух модулей	author_only: false limit: 10 offset: 30 module_ids: [06d677a2-f66e-40c9-8d89-75425cddf211, 097ba77c-e36e-4ba8-8cd0-decb46eb7872]	программа выведет четвертую десятку модулей, или меньше	представлен на листинге И.20	тест пройден
23	получение всех терминов из двух модулей	author_only: false limit: 100 offset: 0 module_ids: [06d677a2-f66e-40c9-8d89-75425cddf211, 097ba77c-e36e-4ba8-8cd0-decb46eb7872]	программа выведет первые сто модулей или меньше	представлен на листинге И.21	тест пройден
24	получение терминов из двух модулей, один из которых не существует	author_only: false limit: 100 offset: 0 module_ids: [06d677a2-f66e-40c9-8d89-75425cddf211, 097ba77c-e36e-4ba8-8cd0-decb46eb7871]	программа выведет четвертую десятку терминов первого модуля	представлен на листинге И.22	тест пройден

Окончание таблицы 13

№ теста	Название теста	Входные данные	Ожидаемый результат	Полученный результат	Вердикт
25	получение терминов из двух модулей, оба из которых не существуют	author_only: false limit: 100 offset: 0 module_ids: [06d677a2-f66e-40c9-8d89-75425cddf212, 097ba77c-e36e-4ba8-8cd0-decb46eb7871]	программа выведет пустой список	представлен на листинге И.23	тест пройден
26	получение терминов из модуля, автором которого не является пользователь, но при этом пользователь добавил модуль для изучения	author_only: false limit: 100 offset: 0 module_ids: [f6c16a78-adf8-4dba-a12d-ac3d08d8ff05]	программа выдаст список терминов	представлен на листинге И.24	тест пройден
27	получение терминов из модуля, автором которого не является пользователь, и пользователь не добавлял этот модуль для изучения	author_only: false limit: 100 offset: 0 module_ids: [288384971-8ead-4866-9a57-7eaaff64865e]	программа выдаст список терминов	представлен на листинге И.25	тест пройден
28	получение терминов без указания модуля	author_only: false limit: 100 offset: 0	программа выдаст ошибку валидации	представлен на листинге И.26	тест пройден

В результате проведенного функционального тестирования недостатков в разработанном программном обеспечении выявлено не было.

2.3.4.2 Модульное тестирование серверной части

В ходе проверки качества кода было проведено модульное тестирование. Для написания тестовых сценариев была использована библиотека pytest. Результат тестирования приведен на рисунке 24.

При автоматическом тестировании количество запросов составило:

– два для микросервиса авторизации;

– 8948 для микросервиса папок;

– 16599 для микросервиса модулей;

– 97976 для микросервиса терминов;

– 143 для микросервиса пользователей.

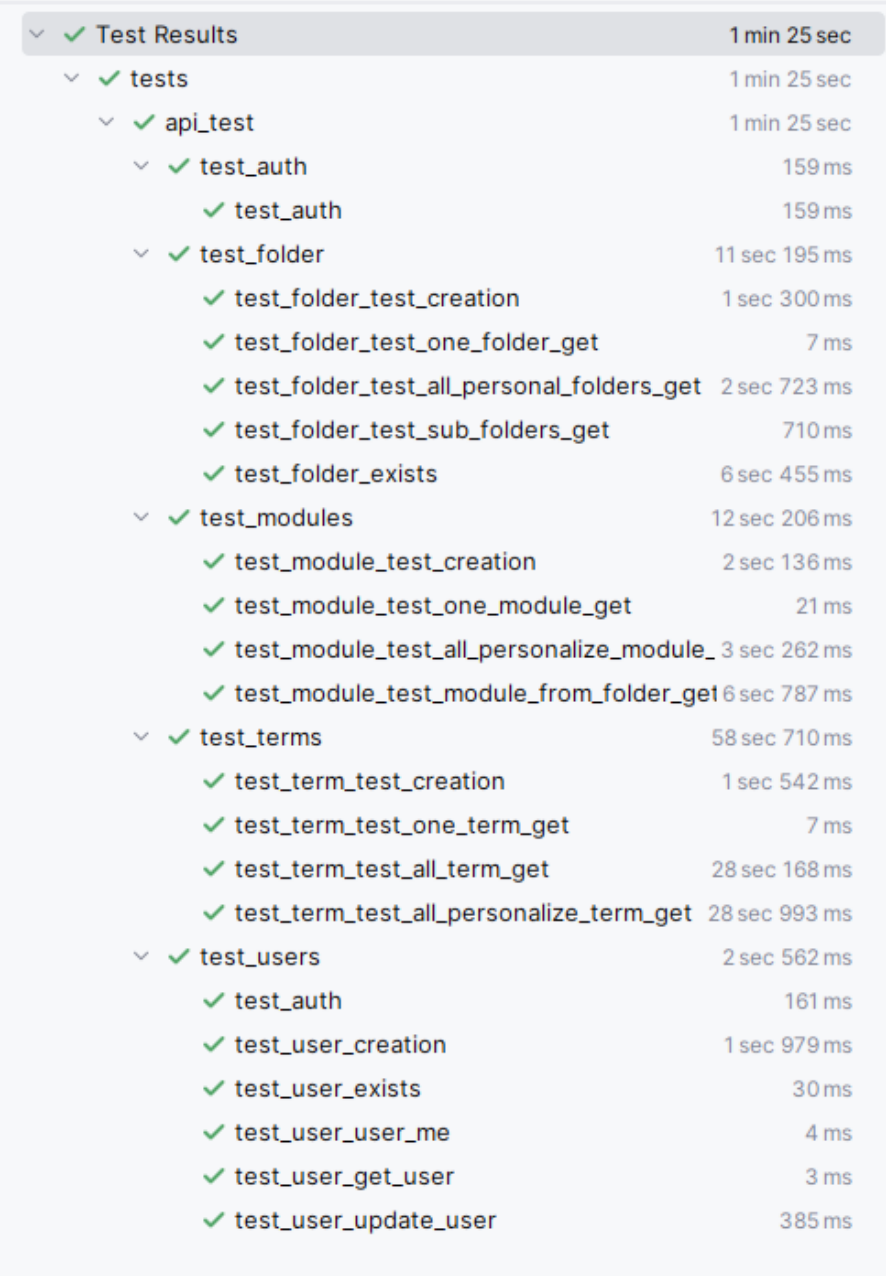


Рисунок 24 – Результаты модульного тестирования

Вывод одного из тестов представлен на рисунке 25.

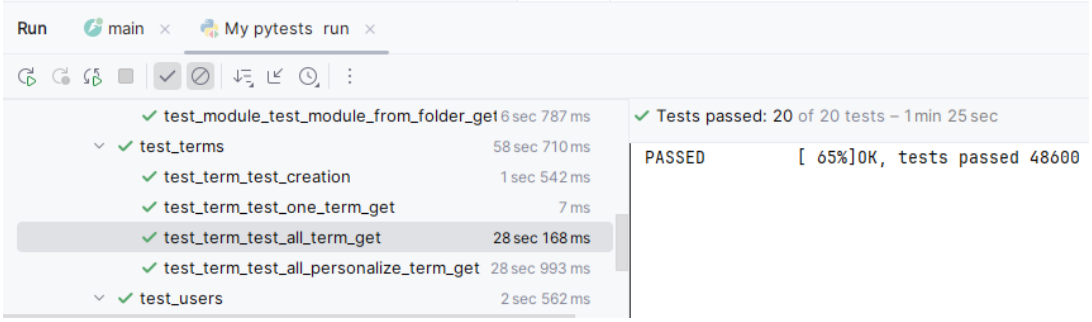


Рисунок 25 – Модульный тест конечной точки получения терминов

Код модульных тестов представлен в приложении К.

Все проведенные модульные тесты были успешно пройдены, было написано 20 модульных тестов, покрывающих наиболее значимый функционал сервера. В процессе модульного тестирования совершено 123666 запросов к серверу. Работоспособность кода проверена и подтверждена.

2.3.4.3 Метрики кода информационной системы «Rem»

С целью проверки качества кода была выделена функция с наиболее значимым функционалом среди функций конечных точек, наиболее близких к медианной длине функций всей программы. Код функции представлен на рисунке 26.

Рассчитаем комплексный показатель качества для функции get_all_module. Данные для расчета представлены в таблице 14.

```
DaniinXorchenabo *
@module_app.get(
    path: "/personalize/from_folder/{folder_id}",
    response_model=Ans[list[PersonalizeModuleDTO], BError | None],
    status_code=200,
    responses={**base_resp_models( *status_codes: 401, 409, 422, 456, 500)}
)
async def get_all_module(
        folder_id: Annotated[
            UUID | Literal["null"] | None, Path(description="Установите null для получения корневых модулей")],
        db_limits: Annotated[Callable[[Select], Select], Depends(limits_dep(
            PersonalizeModuleORM,
            updated_at_condition=two_ent_upd_condition(PersonalizeModuleORM, ModuleORM)
        ))],
        db_session: Annotated[AsyncSession, Depends(get_db_session)],
        current_user: Annotated[UserDTO, Depends(get_current_user)],
        author_only: bool = False,
):
    folder_id = (None if folder_id is None or str(folder_id).lower() == 'null' else folder_id)
    query = (
        select(PersonalizeModuleORM)
        .join(PersonalizeModuleORM.module_entity)
        .add_columns(ModuleORM)
        .where(
            and_((PersonalizeModuleORM.user_id == current_user.id),
             (PersonalizeModuleORM.root_folder_id == folder_id))
        )
    )

    if author_only:
        query = query.where(ModuleORM.author_id == current_user.id)
    res = await db_session.execute(
        db_limits(query)
    )

    data = list(res)
    return [PersonalizeModuleDTO.model_validate(i[0].module_entity.__dict__ | i[0].__dict__)
            for i in data]
```

Рисунок 26 – Код функции получения всех модулей из папки

Таблица 14 – Расчет словаря и длины

Операторы			Операнды		
Оператор	№	Количество	Операнд	№	Количество
=	1	5	folder_id	1	5
if	2	2	current_user	2	2
Is	3	1	author_only	3	1
Or	4	1	PersonalizeModuleORM	4	4
Else	5	1	ModuleORM	5	2
()	6	2	query	6	4
await	7	1	db_session	7	1
,	8	1	res	8	2
:	9	1	data	9	2
[]	10	1	i		3
For … in …	11	1			
|	12	1			
return	13	1			
==	14	4			
Всего	n1 = 14	N1 = 23	Всего	n2 = 9	N2 = 26

Используя эти данные (смотри таблицу 14) рассчитаем метрику вычислительной сложности Halstead Volume (HV), равную (N1 + N2) * log2(n1 + n2) = 221.7.

Затем составим управляющий граф для функции get_all_module, представленный на рисунке 27.

На основании построенного управляющего графа вычислим метрику Cyclomatic Complexity (цикломатическая сложность, CC), показывающую структурную сложность кода, равную четырём (два оператора условия, один цикл, плюс один по формуле 4.10 [34]).

Количество строк кода (LoC) – 18.

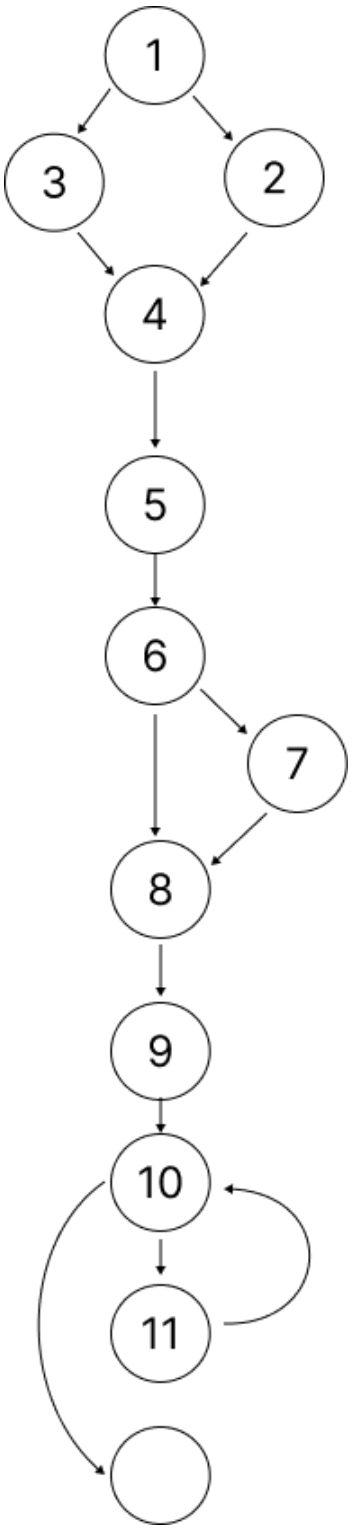


Рисунок 27 – Управляющий граф функции получения модулей из папки

Комплексный показатель качества [34] равен 64, что свидетельствует о высоком качестве кода.

Заключение

В результате выполнения бизнес-проекта был:

– проведен анализ рынка и предполагаемой экономической ниши;

– описаны бизнес-идея и разрабатываемый продукт;

– составлены финансовый и маркетинговый планы;

– выигран грант «УМНИК» в размере 500000 рублей от Фонда содействия инновациям (приложение А).

Рассмотрим основные результаты, полученные в ходе выполнения ВКР.

1. Проведен анализ предметной области. В результате чего были выявлены основные подходы к изучению иностранного языка с использованием цифровых средств.

2. Изучены аналоги. В результате были выделены три основные группы, на которые можно разделить выявленные аналоги.

3. Проанализированы требования. Была выявлена необходимая функциональность разработанной системы, на основе которой поставлены цели и задачи на ВКР.

4. Осуществлено проектирование информационной системы. Были выбраны: сервисная архитектура для мобильного приложения и сервисная архитектура для сервера, спроектированы логические модели баз данных и свёрстан макет пользовательского интерфейса.

5. Произведена оценка бюджета. Было рассчитано, что на промышленную разработку запланированной части информационной системы будет потрачено 693000 рублей и потребуется 1907 человеко-часов.

6. Написана информационная система. Был написан программный код для мобильного приложения и сервера, в так же создана базы данных.

7. Произведены тестирование и оценка метрик написанной информационной системы. Качество написанного кода подтверждено.

На реализацию бизнес-проекта, в камках которого осуществлялось выполнение ВКР, был выигран грант «УМНИК» в размере 500 тысяч рублей от Фонда содействия инновациям (приложение А).

Было подано две заявки на регистрацию интеллектуальной собственности на программы для ЭВМ (программа клиента и программа сервера), которые на данный момент находятся на стадии рассмотрения (заявка на программу ЭВМ 2024662046 и заявка на программу ЭВМ 2024662670).

В процессе реализации ВКР была опубликована одна [27]. Одна статья находится на публикации.

В процессе реализации проекта был пройден пре-акселератор ПГУ, проект вошел в 10-ку лучших и был признан финалистом пре-акселерационной программы Пензенского государственного университета (приложение Б).

Готовность в содействии и помощи проекту выразили следующие организации:

– ООО «Открытые решения»;

– частная школа английского языка «Лингва»;

– международная школа программирования и математики «Алгоритмика»;

– кафедра «МОиПЭВМ» Пензенского Государственного университета;

– бизнес-инкубатор ПГУ;

– кафедра «Русского языка как иностранного».

Письма поддержки представлены в приложении В.

Список использованных источников

1. Коломенцева Д.А. Большой годовой отчет edtech-рынка, прогнозы на 2024 год // аналитическое агентство Smart Ranking. [Электронный ресурс]. URL: https://smartranking.ru/ru/researchshop/Edtech/edtech2023/ (дата обращения: 08.06.2024). – Режим доступа: платный.

2. Официальный сайт компании skyeng // ОАНО ДПО «СКАЕНГ». [Электронный ресурс]. URL: https://skyeng.ru/ (дата обращения: 08.06.2024).

3. Коломенцева Д.А. Второй квартал и первое полугодие 2023 на edtech-рынке // аналитическое агентство Smart Ranking. [Электронный ресурс]. URL: https://smartranking.ru/ru/researchshop/Edtech/edtech-obzor/ (дата обращения: 08.06.2024). – Режим доступа: платный.

4. Системы управления обучением // Wikimedia Foundation and contributors. [Электронный ресурс]. URL: https://ru.wikipedia.org/wiki/Система_управления_обучением (дата обращения: 08.06.2024).

5. Пензенский Государственный Университет. Положение. Положение от 28.12.2021 № 212 -20 о порядке выполнения обучающимися по образовательным программам высшего образования выпускной квалификационной работы в форме стартапа (бизнес-проекта). [Текст]: офиц. текст. 2021. – 11 с.

6. Бизнес для бизнеса // Wikimedia Foundation and contributors. [Электронный ресурс]. URL: https://ru.wikipedia.org/wiki/B2B (дата обращения: 08.06.2024).

7. Бизнес для потребителя // Wikimedia Foundation and contributors. [Электронный ресурс]. URL: https://ru.wikipedia.org/wiki/B2C (дата обращения: 08.06.2024).

8. Кривая забывания // Wikimedia Foundation and contributors. [Электронный ресурс]. URL: https://ru.wikipedia.org/wiki/Кривая_забывания (дата обращения: 08.06.2024).

9. Ebbinghaus, H. Über das Gedächtnis / H. Ebbinghaus. – Leipzig : Dunker, 1885. – 188 с.

10. Ebbinghaus, H. «Urmanuskript «Ueber das Gedächtniss» / H. Ebbinghaus – Leipzig : Dunker, 1880. – 87 с.

11. kbmobile. Aword учите английский язык // Google Inc. [Электронный ресурс]. URL: https://play.google.com/store/apps/details?id=com.kbmobile.englishwordsguide (дата обращения: 08.06.2024).

12. Официальный сайт компании Duolingo // Duolingo Inc. [Электронный ресурс]. URL: https://ru.duolingo.com/ (дата обращения: 08.06.2024).

13. Официальный сайт компании Lingualeo // Lingualeo Inc. [Электронный ресурс]. URL: https://lingualeo.com/ru (дата обращения: 08.06.2024).

14. Knudge.me. Курсы английского. Игра слов // Google Inc. [Электронный ресурс]. URL: https://play.google.com/store/apps/details?id=com.knudge.me (дата обращения: 08.06.2024).

15. Официальный сайт компании Quizlet // Quizlet, Inc. [Электронный ресурс]. URL: https://quizlet.com (дата обращения: 08.06.2024).

16. Официальный сайт мобильного приложения u-dictionary // NetEase, Inc. [Электронный ресурс]. URL: https://www.u-dictionary.com/#/English (дата обращения: 08.06.2024).

17. Официальный сайт онлайн-школы // ООО «ПРЕПРЕП.РУ». [Электронный ресурс]. URL: https://tetrika-school.ru/ (дата обращения: 08.06.2024).

18. Официальный сайт онлайн-школы // ООО «ИНГЛЕКС». [Электронный ресурс]. URL: https://englex.ru (дата обращения: 08.06.2024).

19. Официальный сайт онлайн-школы // ООО «ДИДЖИТАЛ КОНТЕНТ». [Электронный ресурс]. URL: https://tutorplace.ru/ (дата обращения: 08.06.2024).

20. Официальный сайт онлайн-школы // ООО «Паззл Инглиш Технолоджиз». [Электронный ресурс]. URL: https://puzzle-english.com/ (дата обращения: 08.06.2024).

21. Официальный сайт онлайн-школы // ООО «ЛернПроект». [Электронный ресурс]. URL: https://deutschonline.ru/ (дата обращения: 08.06.2024).

22. Официальный сайт онлайн-школы // ООО «АДВАНС». [Электронный ресурс]. URL: https://advance-club.ru/ (дата обращения: 08.06.2024).

23. Официальный сайт онлайн-школы // Н.Р. Багдасарян. [Электронный ресурс]. URL: https://alfaschool.ru/ (дата обращения: 08.06.2024).

24. Официальный сайт LMS-системы // ООО «Ричмедиа». [Электронный ресурс]. URL: https://www.ispring.ru/ispring-learn (дата обращения: 08.06.2024).

25. Официальный сайт LMS-системы // ООО «Интернет Школа». [Электронный ресурс]. URL: https://www.teachbase.ru/ (дата обращения: 08.06.2024).

26. Ричардс, М. Фундаментальный подход к программной архитектуре: паттерны, свойства, проверенные методы / М. Ричардс, Н. Форд – СПб. : Питер, 2023. – 448 с.

27. Дьячков, Д.А. Достижение компромиссности между переиспользованием и дублированием кода в микросервисной архитектуре /. Д.А. Дьячков, А.С. Кудашов, В.А. Агапова // Современные тенденции развития фундаментальных и прикладных наук : материалы VII Всероссийской научно-практической конференции – Брянск : Брянский государственный инженерно-технологический университет – 2024. С. 266 – 269.

28. Горбаченко, Г.И. Проектирование информационных систем с CA ERwin Modeling Suite 7.3 / Г.И. Горбаченко, Г.Ф. Убиенных, Г.В. Бобрышева – Пенза : ПГУ, 2012. – 145 с.

29. Дьячков Д.А. Исходный код мобильного приложения бизнес-проекта «Rem» // Microsoft Corporation. [Электронный ресурс]. URL: https://github.com/The-remember-app/remember_android_app (дата обращения: 08.06.2024) – Режим доступа: по заявке.

30. Дьячков Д.А. Исходный код серверной части бизнес-проекта «Rem» // Microsoft Corporation. [Электронный ресурс]. URL: https://github.com/The-remember-app (дата обращения: 08.06.2024) – Режим доступа: по заявке.

31. Myers, G.J.The Art of Software Testing / G.J. Myers – Hoboken, New Jersey : John Wiley & Sons, 2004. – 234 с.

32. Бах. Д. Переосмысление классов эквивалентности, часть 1 // Software-Testing.RU / Н.С. Баранцева. [Электронный ресурс]. URL: https://www.software-testing.ru/library/testing/test-analysis/2467-equivalence-classes (дата обращения: 08.06.2024).

33. ГОСТ Р 7.0.64-2018 (ИСО 8601:2004). Представление дат и времени. Общие требования. – Введ. 01.01.2019. – М. : Стандартинформ, 2018. – 32 с.

34. Гурьянов, Л.В. Выпускная квалификационная работа: стандарты и программные технологии. Учебно-методическое пособие для бакалавров по направлениям 09.03.04 «Программная инженерия», 09.03.02 «Информационные системы и технологии» / Л.В. Гурьянов, В.Н. Князев, И.Ю. Балашова, Е.А. Дзюба, Н.А. Попова, Д.В. Такташкин, С.В. Самуйлов – Пенза : ПГУ, 2020. – 184с.

ПРИЛОЖЕНИЕ А
(РЕКОМЕНДУЕМОЕ)
ДИПЛОМ ГРАНТОПОЛУЧАТЕЛЯ

Рисунок А.1 –

Рисунок А.2 –

ПРИЛОЖЕНИЕ Б
(РЕКОМЕНДУЕМОЕ)
СЕРТИФИКАТ ФИНАЛИСТА ПРЕАКСЕЛЕРАТОРА ПГУ

Акселерационная программа

Пензенского государственного университета

СЕРТИФИКАТ

финалиста

Настоящий сертификат удостоверяет,
что команда стартап-проекта **«Разработка технологии организации процесса обучения иностранным языкам на основе нейросетевого анализа»** в составе

Дьячков Даниил Александрович, Левин Андрей Алексеевич, Байрамова Айтач Мубариз-кызы

участники

успешно выступила на Демо-дне и завершила обучение по Акселерационной программе ПГУ, реализованной в рамках федерального проекта «Платформа университетского технологического предпринимательства» государственной программы «Научно-технологическое развитие Российской Федерации» (сентябрь – декабрь 2023 г.)

Проректор по научной работе и инновационной деятельности Пензенского государственного университета **С.М. Васин**

г. Пенза, 2023 г.

ПРИЛОЖЕНИЕ В
(РЕКОМЕНДУЕМОЕ)
ПИСЬМА ПОДДЕРЖКИ

ООО ОЦ «ЛИНГВА»
+ 7 (8412) 20-06-36
lingua_center@mail.ru
Россия, 440062 г. Пенза, ул. Хорошая д. 90
ОГРН 1185835008281
ИНН 5836687032
Исх. № 1 от «20» октября 2023 г.

Федеральное государственное бюджетное учреждение «Фонд содействия развитию малых форм предприятий в научно-технической сфере» (Фонд содействия инновациям)

Генеральному директору
Полякову Сергею Геннадьевичу

Уважаемый Сергей Геннадьевич!

Настоящим письмом компания ООО ОЦ «ЛИНГВА» в лице Генерального директора Дятловой Екатерины Александровны выражает заинтересованность в реализации и продвижении проекта Дьячкова Даниила «Разработка информационной системы организации процесса дистанционного обучения иностранному языку с применением нейросетевых технологий» в рамках программы «УМНИК».

Проект представляет собой многомодульную информационную систему, которая предназначена для повышения эффективности процесса изучения иностранных языков, оптимизации взаимодействия между обучаемым и ментором, и решения следующих задач образовательного процесса:

- автоматизация проверки знаний;
- формирование индивидуальных образовательных траекторий;
- применение удобного и интересного для учеников формата выполнения домашних заданий;
- сбор и систематизация наглядной статистика обучения для менторов.

В ходе реализации планируется разработка трёх нейросетевых программных модулей для определения степени текущего внимания, степени освоение материала и составления оптимального порядка изучаемого материала в рамках индивидуальных образовательных траекторий. Кроме того, планируется разработка административного web-приложения и мобильного приложения для изучения иностранных языков.

Компания поддерживает участие Проекта в грантовом конкурсе программы «УМНИК» и рекомендует проект к рассмотрению конкурсной комиссией.

Компания готова оказать консультативную и информационную поддержку Проекту, а также экспертную поддержку в лице своих сотрудников. Кроме того, обеспечить освещение проекта на своих информационных ресурсах: сайт Компании – https://centerlingua.ru/, группа в социальной сети ВКонтакте – https://vk.com/centerlinguaru.

Генеральный директор Е.А. Дятлова

Рисунок В.1 – Письмо поддержки от ООО ОЦ «Лингва»

МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Пензенский государственный университет»
(ФГБОУ ВО «ПГУ»)
ул. Красная, д. 40, г. Пенза, Россия, 440026
Тел/факс: (841-2) 56-51-22, e-mail: cnit@pnzgu.ru, http://www.pnzgu.ru
ОКПО 02069042, ОГРН 1025801440620, ИНН/КПП 5837003736/583701001

_____ № _____
На № _____ от _____

Федеральное государственное бюджетное учреждение «Фонд содействия развитию малых форм предприятий в научно-технической сфере» (Фонд содействия инновациям)

Генеральному директору
Полякову Сергею Геннадьевичу

Уважаемый Сергей Геннадьевич!

Настоящим письмом кафедра «Математическое обеспечение и применение ЭВМ» Пензенского государственного университета в лице Заведующего кафедрой Козлова Андрея Юрьевича выражает заинтересованность в реализации и продвижении проекта Дьячкова Даниила «Разработка информационной системы организации процесса дистанционного обучения иностранному языку с применением нейросетевых технологий» в рамках программы «УМНИК».

Проект представляет собой многомодульную информационную систему, которая предназначена для повышения эффективности процесса изучения иностранных языков, оптимизации взаимодействия между обучаемым и ментором, и решения следующих задач образовательного процесса:

- автоматизация проверки знаний;
- формирование индивидуальных образовательных траекторий;
- применение удобного и интересного для учеников формата выполнения домашних заданий;
- сбор и систематизация наглядной статистики обучения для менторов.

В ходе реализации планируется разработка трёх нейросетевых программных модулей для определения степени текущего внимания, степени освоения материала и составления оптимального порядка изучаемого материала в рамках индивидуальных образовательных траекторий. Кроме того, планируется разработка административного web-приложения и мобильного приложения для изучения иностранных языков.

Компания поддерживает участие Проекта в грантовом конкурсе программы «УМНИК» и рекомендует проект к рассмотрению конкурсной комиссией.

Компания готова оказать консультативную и информационную поддержку Проекту, а также экспертную поддержку в лице своих сотрудников. Кроме того, обеспечить освещение проекта на своих информационных ресурсах: сайт Компании – https://dep_mopevm.pnzgu.ru, группа в социальной сети ВКонтакте – https://vk.com/mopevm.

Заведующий кафедрой А.Ю.Козлов

Рисунок В.2 – Письмо поддержки от кафедры МОиПЭВМ

МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Пензенский государственный университет»
(ФГБОУ ВО «ПГУ»)
ул. Красная, д. 40, г. Пенза, Россия, 440026
Тел/факс: (841-2) 56-51-22, e-mail: cnit@pnzgu.ru, http://www.pnzgu.ru
ОКПО 02069042, ОГРН 1025801440620, ИНН/КПП 5837003736/583701001

23.10.2023 № 29-650

Генеральному директору
Фонда содействия инновациям
С.Г. Полякову

Уважаемый Сергей Геннадьевич!

Кафедра «Русский язык как иностранный» Института международного сотрудничества ФГБОУ ВО «Пензенский государственный университет» настоящим письмом выражает заинтересованность в реализации и продвижении проекта Дьячкова Даниила «Разработка информационной системы организации процесса дистанционного обучения иностранному языку с применением нейросетевых технологий» в рамках программы «УМНИК».

Данный проект представляет собой многомодульную информационную систему и направлен на решение актуальной проблемы повышения эффективности процесса изучения иностранных языков, оптимизации взаимодействия между обучаемым и ментором. Задачи, поставленные в проекте (автоматизация проверки знаний; формирование индивидуальных образовательных траекторий; применение удобного и интересного для учеников формата выполнения домашних заданий; сбор и систематизация наглядной статистики обучения для менторов), имеют важное значение для повышения эффективности образовательного процесса.

Согласно предусмотренному плану проекта в ходе его реализации планируется разработка трёх нейросетевых программных модулей для определения степени текущего внимания, степени освоения материала и составления оптимального порядка изучаемого материала в рамках индивидуальных образовательных траекторий. Кроме того, планируется разработка административного web-приложения и мобильного приложения для изучения иностранных языков. Считаем, данные разработки будут иметь практическую ценность и могут найти применение в процессе преподавания иностранных языков, в том числе русского языка как иностранного.

Кафедра «Русский язык как иностранный» поддерживает участие Проекта в грантовом конкурсе программы «УМНИК» и рекомендует проект к рассмотрению конкурсной комиссией.

Кафедра готова оказать консультативную и информационную поддержку Проекту, а также экспертную поддержку в лице своих сотрудников.

Заведующий кафедрой
«Русский язык как иностранный»,
к. филол. н. О. В. Барабаш

Рисунок В.3 – Письмо от кафедры «Русский язык как иностранный»

Алгоритмика

Международная школа программирования и математики

Федеральное государственное бюджетное учреждение «Фонд содействия развитию малых форм предприятий в научно-технической сфере» (Фонд содействия инновациям)

Генеральному директору
Полякову Сергею Геннадьевичу

Уважаемый Сергей Геннадьевич!

Настоящим письмом Индивидуальный предприниматель Тришина Дарья Дмитриевна, франчайзи-партнер Школы программирования «Алгоритмика» в городе Пензе и в городе Кузнецк, выражает заинтересованность в реализации и продвижении проекта Дьячкова Даниила Александровича «Разработка информационной системы организации процесса дистанционного обучения иностранному языку с применением нейросетевых технологий» в рамках программы «УМНИК».

Проект представляет собой многомодульную информационную систему, которая предназначена для повышения эффективности процесса изучения иностранных языков, оптимизации взаимодействия между обучаемым и ментором, и решения следующих задач образовательного процесса:

- автоматизация проверки знаний;
- формирование индивидуальных образовательных траекторий;
- применение удобного и интересного для учеников формата выполнения домашних заданий;
- сбор и систематизация наглядной статистика обучения для менторов.

В ходе реализации планируется разработка трёх нейросетевых программных модулей для определения степени текущего внимания, степени освоения материала и составления оптимального порядка изучаемого материала в рамках индивидуальных образовательных траекторий. Кроме того, планируется разработка административного web-приложения и мобильного приложения для изучения иностранных языков.

Компания поддерживает участие Проекта в грантовом конкурсе программы «УМНИК» и рекомендует проект к рассмотрению конкурсной комиссией.

Компания готова оказать консультативную и информационную поддержку Проекту, а также экспертную поддержку в лице своих сотрудников. Кроме того, обеспечить освещение проекта на своих информационных ресурсах: сайт Компании – https://penza.algoritmika.org, группа в социальной сети ВКонтакте – https://vk.com/algoritmika_penza.

Индивидуальный предприниматель Тришина Дарья Дмитриевна

Тришина Дарья Дмитриевна

Рисунок В.4 – Письмо поддержки от школы «Алгоритмика»

Открытые Решения_
разработка программного обеспечения

> 8 800 250 9669
> info@osinit.com
www.osinit.com

> ООО «Открытые решения»
ИНН 5835081619

Исх. № 1202 от «23» октября 2023 г.

Федеральное государственное бюджетное учреждение «Фонд содействия развитию малых форм предприятий в научно-технической сфере» (Фонд содействия инновациям)

Генеральному директору
Полякову Сергею Геннадьевичу

Уважаемый Сергей Геннадьевич!

Настоящим письмом компания ООО «Открытые Решения» в лице Исполнительного директора Бочкарева Игоря Вячеславовича выражает заинтересованность в реализации и продвижении проекта Дьячкова Даниила «Разработка информационной системы организации процесса дистанционного обучения иностранному языку с применением нейросетевых технологий» в рамках программы «УМНИК».

Проект представляет собой многомодульную информационную систему, которая предназначена для повышения эффективности процесса изучения иностранных языков, оптимизации взаимодействия между обучаемым и ментором, и решения следующих задач образовательного процесса:

- автоматизация проверки знаний;
- формирование индивидуальных образовательных траекторий;
- применение удобного и интересного для учеников формата выполнения домашних заданий;
- сбор и систематизация наглядной статистика обучения для менторов.

В ходе реализации планируется разработка трёх нейросетевых программных модулей для определения степени текущего внимания, степени освоение материала и составления оптимального порядка изучаемого материала в рамках индивидуальных образовательных траекторий. Кроме того, планируется разработка административного web-приложения и мобильного приложения для изучения иностранных языков.

Компания поддерживает участие Проекта в грантовом конкурсе программы «УМНИК» и рекомендует проект к рассмотрению конкурсной комиссией.

Компания готова оказать консультативную и информационную поддержку Проекту, а также экспертную поддержку в лице своих сотрудников. Кроме того, обеспечить освещение проекта на своих информационных ресурсах: сайт платформы подготовки IT-специалистов – software-academy.ru, группы в социальных сетях – https://vk.com/osinitcom; t.me/open_solutions.

Исполнительный директор И.В. Бочкарев

Рисунок В.5 – Письмо поддержки от ООО «Открытые решения»

ПРИЛОЖЕНИЕ Г
(РЕКОМЕНДУЕМОЕ)
СПЕЦИФИКАЦИИ ВАРИАНТОВ ИСПОЛЬЗОВАНИЯ

Наименование: авторизоваться
ID: 1
Краткое описание: ученик входит в свой аккаунт, чтобы получить доступ к своим (добавленным) терминам и получить возможность их изучать
Действующие лица: ученик, мобильное приложение, сервер
Предусловие: ученик не авторизирован
Основной поток: 1) Пользователь вводит логин и пароль 2) Мобильное приложение отправляет данные на сервер 3) Сервер проверяет наличие пользователя в БД 4) Сервер сопоставляет введенный пароль с хэшем пароля из БД 5) Сервер возвращает генерирует токен доступа 6) Сервер отправляет токен доступа в мобильное приложение 7) Мобильное приложение запоминает токен доступа
Постусловие: ученик вошел в аккаунт

Рисунок Г.1 – Спецификация прецедента «Авторизоваться»

Наименование: применить механику выбора
ID: 2
Краткое описание: пользователь изучает модуль по определенной тематике и выполняет задания. Механика выбора предполагает, что пользователю показан один термин, для него представлены варианты ответов (определений), из которых необходимо выбрать один верный
Действующие лица: пользователь, система
Предусловие: пользователь выбрал модуль для изучения
Основной поток: 1) Пользователь выбирает вариант ответа 2) Система проверяет корректность пары термин-определение
Постусловие: пользователь выполнил задание

Рисунок Г.2 – Спецификация прецедента «применить механику выбора»

Наименование: добавить пример использования термина
ID: 3
Краткое описание: пользователь добавляет предложение или словосочетание для изучения термина в контексте
Действующие лица: пользователь, система
Предусловие: пользователь добавил термин
Основной поток: 1) Пользователь добавляет предложение или словосочетание 2) Пример добавляется в базу данных
Постусловие: пользователь добавил пример использования термина

Рисунок Г.3 – Спецификация прецедента «добавить пример»

Наименование: Добавить модуль
ID: 4
Краткое описание: пользователь добавляет модуль для изучения
Действующие лица: пользователь, система
Предусловие: пользователь создал папку и модуль
Основной поток: 1) Пользователь добавляет модуль в папку 2) Для модуля в базе данных добавляется родительская папки
Постусловие: пользователь добавил модуль

Рисунок Г.4 – Спецификация прецедента «Добавить модуль»

Наименование: изучить модуль
ID: 5
Краткое описание: ученик изучает все термины, объединенные в словарный модуль
Действующие лица: ученик, мобильное приложение, сервер
Предусловие: ученик выбрал словарный модуль для изучения
Основной поток: 1) Ученик наживает кнопку «Выучить модуль заново» 2) Мобильное приложение записывает в каждый термин модуля количество повторений каждой механики, необходимое для того, чтобы считать термин выученным. 3) Мобильное приложение отправляет данные из п 2 на сервер 4) Сервер сохраняет данные из п 2 в бд 5) Мобильное приложение выбирает определенное количество терминов для первой итерации обучения 6) Пользователь выполняет по одной предложенной системой механике изучения для каждого термина в рамках итерации 7) Мобильное приложение обновляет данные по количеству необходимых механик каждого термина из итерации в локальной БД 8) Мобильное приложение отправляет данные из п7 на сервер 9) Сервер сохраняет данные из п7 в БД 10) Пользователь наживает кнопку продолжить обучение 11) Выполняются пункты 5 – 10 до того, как все термины модуля не будут изучены
Постусловие: пользователь выучил модуль

Рисунок Г.5 – Спецификация прецедента «Изучить модуль»

Наименование: изучить термин
ID: 6
Краткое описание: ученик изучает один из терминов модуля
Действующие лица: ученик, мобильное приложение, сервер
Предусловие: было инициировано изучение модуля
Основной поток: 1) Пользователь выполнил механику изучения предыдущего термина 2) Пользователь выполняет механику изучения «Выбор» 3) Мобильное приложение увеличивает счетчик пройденных механик выбора для данного термина 4) Мобильное приложение выдает пользователю механику изучения следующего термина 5) П1 – п5 повторяются до тех пор, пока счетчик проеденных механик выбора не сравняется с целевым количеством механик выбора, установленных при старте изучения модуля 6) Пользователь выполнил механику изучения предыдущего термина 7) Мобильное приложение выдает пользователю механику изучения «письмо» 8) Мобильное приложение увеличивает счетчик пройденных механик письма для данного термина 9) Мобильное приложение выдает пользователю механику изучения следующего термина 10) П7 – п11 повторяются до тех пор, пока счетчик проеденных механик письма не сравняется с целевым количеством механик письма, установленных при старте изучения модуля
Постусловие: пользователь выучил термин

Рисунок Г.6 – Спецификация прецедента «Изучить термин»

Наименование: применить механику выбора
ID: 7
Краткое описание: ученик изучает один из терминов модуля
Действующие лица: ученик, мобильное приложение, сервер
Предусловие: было инициировано изучение модуля
Основной поток: 1) Пользователь выполнил механику изучения предыдущего термина 2) Мобильное приложение выдает пользователю механику изучения «выбор» 3) Пользователь выполняет механику изучения «Выбор» 4) Мобильное приложение увеличивает счетчик пройденных механик выбора для данного термина 5) Мобильное приложение выдает пользователю механику изучения следующего термина
Постусловие: счетчик проеденных механик выбора равен с целевым количеством механик выбора

Рисунок Г.7 – Спецификация прецедента «применить механику выбора»

Наименование: применить механику письма
ID: 8
Краткое описание: ученик изучает один из терминов модуля
Действующие лица: ученик, мобильное приложение, сервер
Предусловие: было инициировано изучение модуля и счетчик проеденных механик выбора равен с целевым количеством механик выбора
Основной поток: 1) Пользователь выполнил механику изучения предыдущего термина 2) Мобильное приложение выдает пользователю механику изучения «письмо» 3) Пользователь выполняет механику изучения «письмо» 4) Мобильное приложение увеличивает счетчик пройденных механик письма для данного термина 5) Мобильное приложение выдает пользователю механику изучения следующего термина 6) П1 – п5 повторяются до тех пор, пока счетчик проеденных механик письма не сравняется с целевым количеством механик письма, установленных при старте изучения модуля
Постусловие: пользователь выучил термин

Рисунок Г.8 – Спецификация прецедента «применить механику письма»

Прецедент	Изучить модуль
ID	1
Краткое описание	Набор действий, в результате которых пользователь запоминает термины, составляющие модуль
Главные актеры	Пользователь
Второстепенные актеры	-
Предусловия	Пользователь авторизован в системе и выбрал модуль для изучения
Основной поток	1. Пользователь внутри модуля нажимает на кнопку «Изучить модуль» 2. Система показывает пользователю первую механику изучения первого термина 3. Пользователь выполняет механику обучения 4. Если пользователь выполнил механику обучения правильно, тогда 4.1. Положительный результат для термина и механики передается в алгоритм изучения 5. Иначе 5.1. Отрицательный результат для термина и механики передается в алгоритм изучения 6. Алгоритм повторяется до тех пор, пока пользователь не выучит все термины
Постусловия	-
Альтернативные потоки	-

Рисунок Г.9 – Сценарий прецедента «Изучить модуль»

ПРИЛОЖЕНИЕ Д
(РЕКОМЕНДУЕМОЕ)
СПИСОК ЗАДАЧ ДИАГРАММЫ ГАНТА

Название задачи	Длительнс	Начало	Окончани	Предшестви	Названия ресурсов
Начало проекта	0 дней	Вс 01.10.23	Вс 01.10.23		
Предварительный анализ	**21 дней**	**Пн 02.10.23**	**Пн 30.10.23**	**1**	**Электричество[1 кВт * ч]**
Анализ цифровых подходов к изучению иностранных языков	20 ч	Пн 02.10.23	Ср 04.10.23	1	Аналитик;Электричество[20 кВт
Обзор аналогов	40 ч	Чт 05.10.23	Чт 12.10.23	3	Аналитик;Электричество[40 кВт
Анализ требований	**3,75 дней**	**Чт 12.10.23**	**Ср 18.10.23**		
Анализ функциональных требований	10 ч	Вт 17.10.23	Ср 18.10.23	4	Аналитик;Электричество[10 кВт
Анализ нефункциональных требований	10 ч	Чт 12.10.23	Пт 13.10.23	4	Аналитик;Электричество[10 кВт
Постановка целей и задач на разработку	10 ч	Ср 18.10.23	Чт 19.10.23	6;7	Аналитик;Электричество[10 кВт
Формирование ТЗ	60 ч	Чт 19.10.23	Пн 30.10.23	8	Аналитик;Backend-разработчик
Проектирование	**16,88 дней**	**Вт 31.10.23**	**Ср 22.11.23**	**9**	
Выбор архитектуры ИС в целом	40 ч	Вт 31.10.23	Вт 14.11.23	9	Backend-разработчик;Mobile-ра
Выбор архитектуры сервера	40 ч	Вт 14.11.23	Вт 21.11.23	11	Backend-разработчик;Электрич
Выбор архитектуры клиента	20 ч	Вт 14.11.23	Пт 17.11.23	11	Mobile-разработчик;Электриче
Предварительное проектирование БД	10 ч	Вт 21.11.23	Ср 22.11.23	11;12	Backend-разработчик;Электрич
Реализация	**130,88 дней**	**Ср 22.11.23**	**Чт 23.05.24**	**2;10**	
Закупка	0 дней	Ср 22.11.23	Ср 22.11.23	10	Лицензия Pycharm Professional[
Реализация сервера	**73,75 дней**	**Ср 22.11.23**	**Вт 05.03.24**	**11;12**	
CI / CD	**16,13 дней**	**Ср 27.12.23**	**Чт 18.01.24**	**11**	
БД	**39,38 дней**	**Ср 10.01.24**	**Вт 05.03.24**	**14**	
Разработка сервисов	**30,63 дней**	**Ср 13.12.23**	**Чт 25.01.24**	**35**	
Движок перегрузки импорта	60 ч	Ср 29.11.23	Чт 25.01.24	12	Backend-разработчик;Электрич
Проектирование Swagger-интерфейса	79 ч	Ср 22.11.23	Ср 06.12.23	9;11;12	Backend-разработчик;Mobile-ра
Реализация http-сервера по swagger интерфейсу, возвращающему заглушки данных	40 ч	Ср 06.12.23	Ср 13.12.23	34	Backend-разработчик;Электрич
Реализация клиента	**130,88 дней**	**Ср 22.11.23**	**Чт 23.05.24**	**11;13**	
Реализация алгоритма обучения	120 ч	Ср 22.11.23	Пн 25.12.23	13	Mobile-разработчик;Аналитик;З
Реализация взаимодействия между изолятами	40 ч	Ср 22.11.23	Вт 02.01.24	13	Mobile-разработчик;Электриче
Реализация сервиса синхронизации	**70 дней**	**Ср 06.12.23**	**Ср 13.03.24**	**34**	
Генерация http-клиента по swagger-макету	20 ч	Пн 08.01.24	Ср 10.01.24	34	Mobile-разработчик;Электриче
Разработка класса взаимодействия с сетью	30 ч	Ср 06.12.23	Пн 08.01.24	13	Mobile-разработчик;Электриче
разработка расширений для обновления БД	20 ч	Ср 10.01.24	Пн 29.01.24	47	Mobile-разработчик;Электриче
Реализация отправки локальных изменений на сервер	40 ч	Пн 29.01.24	Ср 21.02.24	35;42	Mobile-разработчик;Электриче
Реализация движка событий	80 ч	Ср 06.12.23	Ср 13.03.24	13	Mobile-разработчик;Электриче
Локальная БД	**67,75 дней**	**Ср 22.11.23**	**Пн 26.02.24**	**13**	
Реализация пользовательского интерфейса	**106,38 дней**	**Ср 27.12.23**	**Чт 23.05.24**	**65;66;67**	
Реализация движка навигации	60 ч	Ср 27.12.23	Чт 29.02.24	13	Mobile-разработчик;Электриче
Встраивание в UI BO	**76,25 дней**	**Ср 07.02.24**	**Чт 23.05.24**		
Реализация дизайн-макета	**43,38 дней**	**Пт 12.01.24**	**Ср 13.03.24**		
Разработка пользовательского интерфейса	**37,13 дней**	**Ср 22.11.23**	**Пт 12.01.24**	**9;13**	
Разработка концепции дизайна	40 ч	Ср 22.11.23	Ср 29.11.23	13	Дизайнер;Электричество[40 кВт
Разработка концепций экранов	60 ч	Ср 29.11.23	Пн 11.12.23	65	Дизайнер;Электричество[60 кВт
Разработка схемы переключения экранов	16 ч	Чт 14.12.23	Ср 27.12.23	66	Дизайнер;Электричество[16 кВт
Разработка экранов авторизации и регистрации	16 ч	Ср 27.12.23	Пт 29.12.23	66	Дизайнер;Электричество[16 кВт
Разработка экранов навигации по директориям	24 ч	Пн 01.01.24	Ср 03.01.24	66	Дизайнер;Электричество[24 кВт
Разработка экранов модулей	24 ч	Пт 29.12.23	Пн 08.01.24	66	Дизайнер;Электричество[24 кВт
Разработка экранов обучения	80 ч	Пн 18.12.23	Пт 12.01.24	66	Дизайнер;Электричество[80 кВт
Разработка экранов настроек	24 ч	Пн 11.12.23	Чт 14.12.23	66	Дизайнер;Электричество[24 кВт
Тестирование	**131 дней**	**Ср 06.12.23**	**Чт 06.06.24**		
Написание unit-тестов	**9,25 дней**	**Ср 06.12.23**	**Вт 19.12.23**	**34**	
Ручное тестирование сложных сценариев для сервера	40 ч	Вт 05.03.24	Вт 12.03.24	17;21;18;24;3	Тестировщик;Электричество[40
Ручное тестирование мобильного приложения	80 ч	Чт 23.05.24	Чт 06.06.24	36	Тестировщик;Электричество[80
Составление технической докупентации и ПЗ	80 ч	Чт 23.05.24	Чт 06.06.24	15;36;64	Аналитик;Электричество[1 кВт
Окончание проекта	0 дней	Чт 06.06.24	Чт 06.06.24	2;10;15;73;85	

ПРИЛОЖЕНИЕ Е
(РЕКОМЕНДУЕМОЕ)
ПРОЕКТИРОВАНИЕ ПОЛЬЗОВАТЕЛЬСКОГО ИНТЕРФЕЙСА

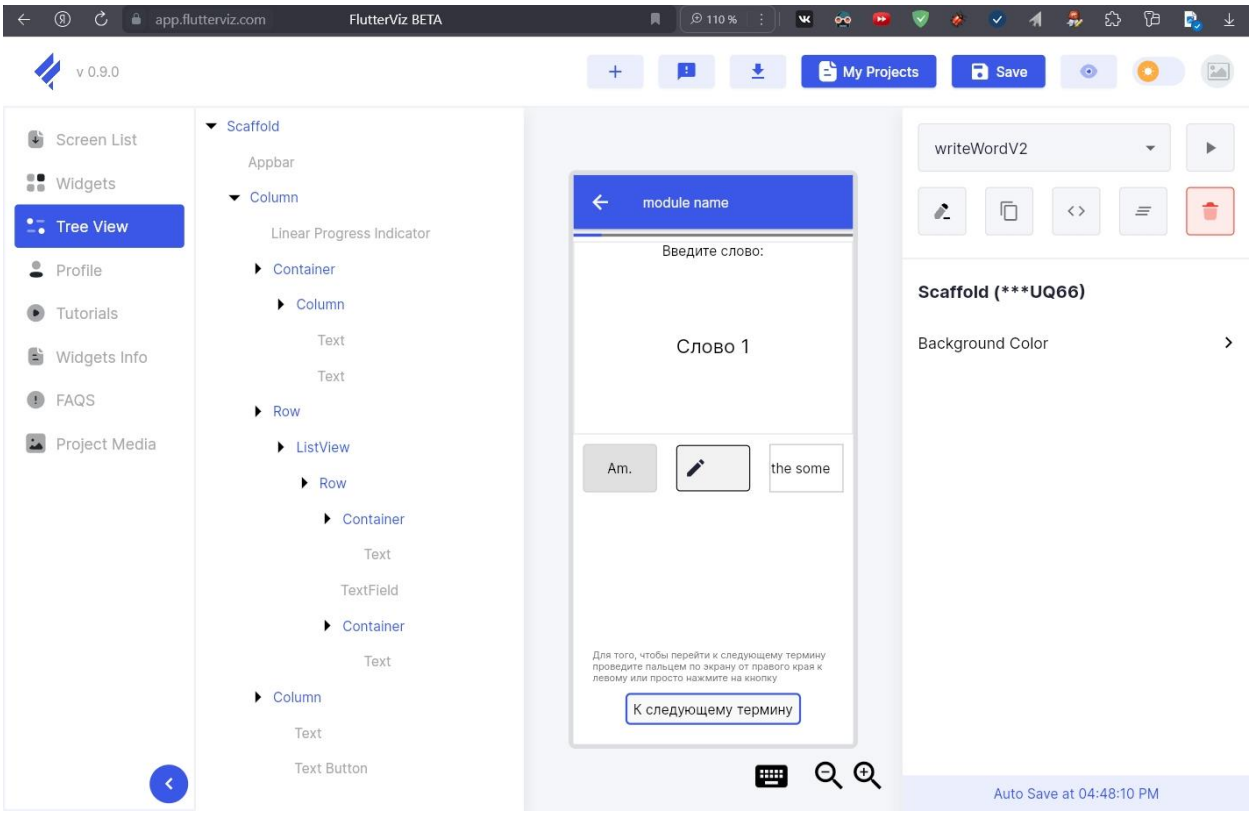


Рисунок Е.1 – Экран ввод составного слова

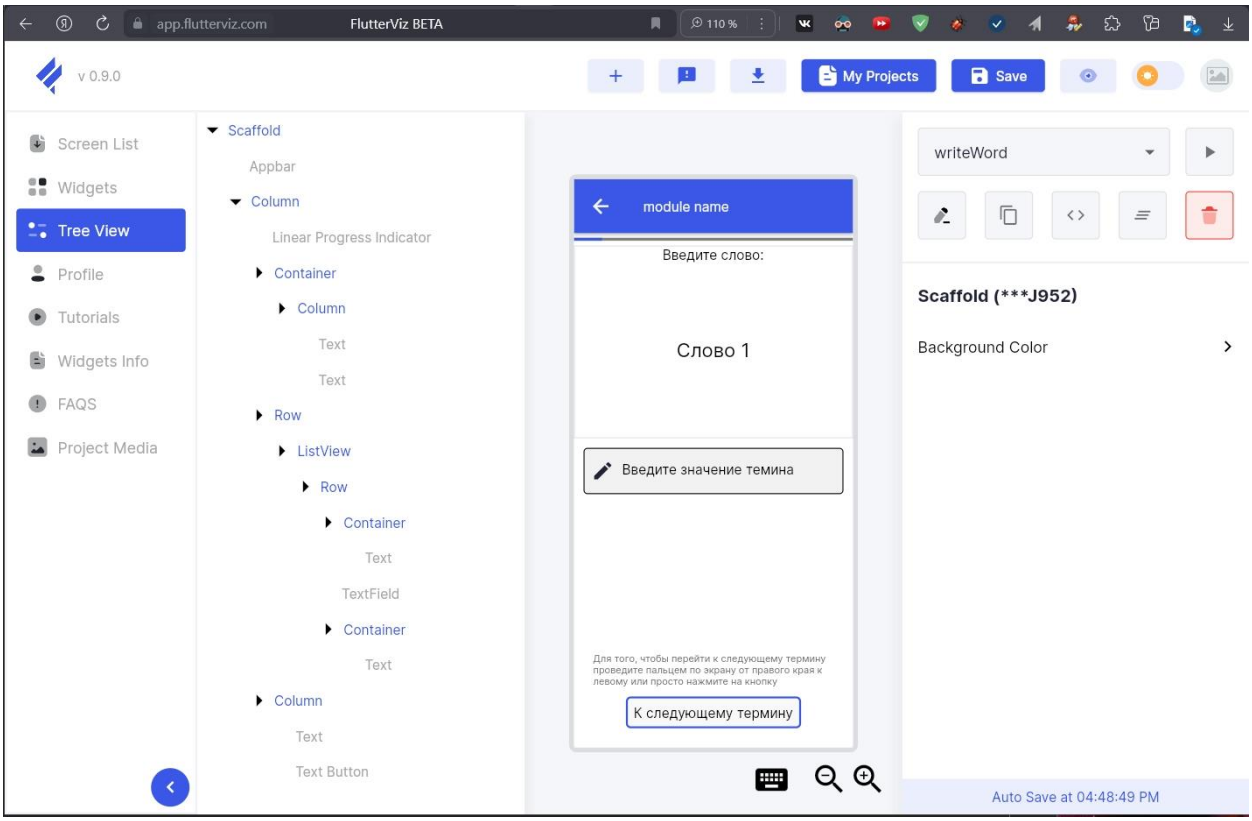


Рисунок Е.2 – Экран ввода простого слова

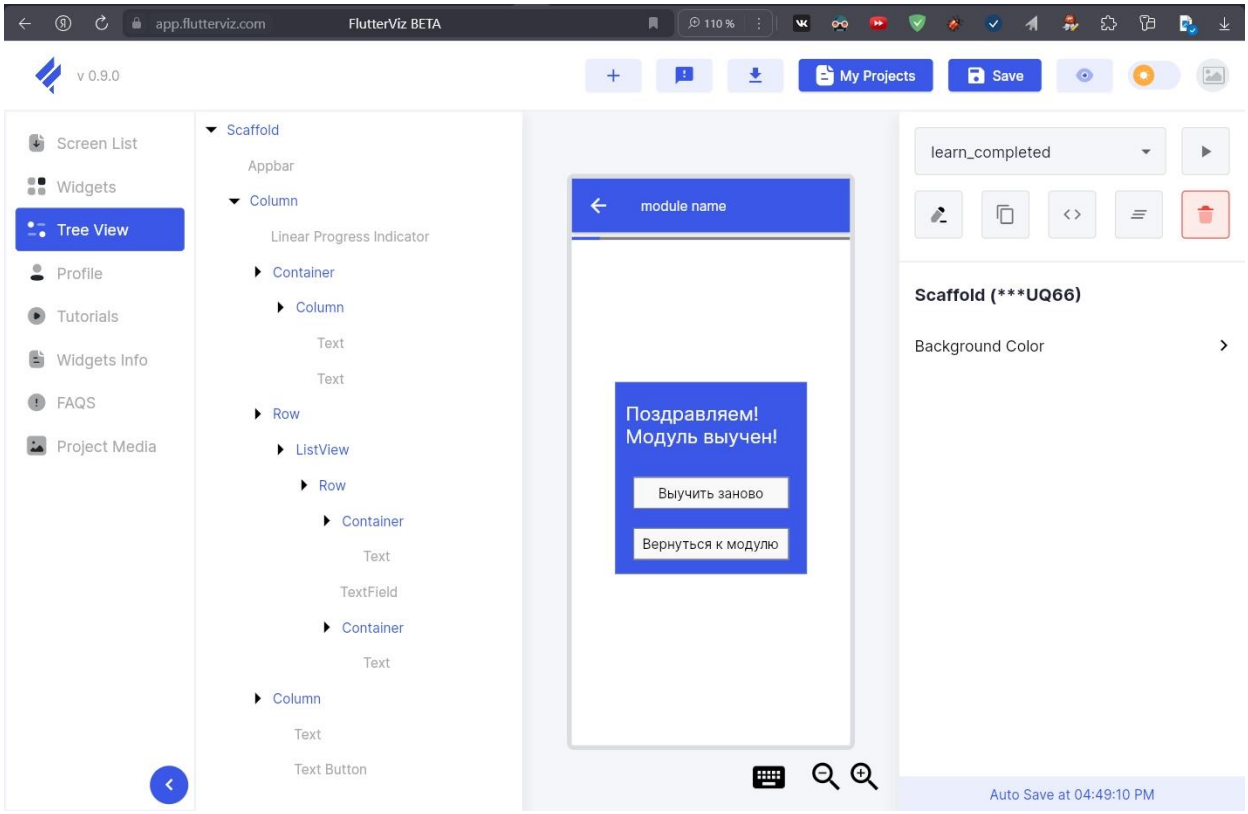


Рисунок Е.3 – Экран окончания изучения термина

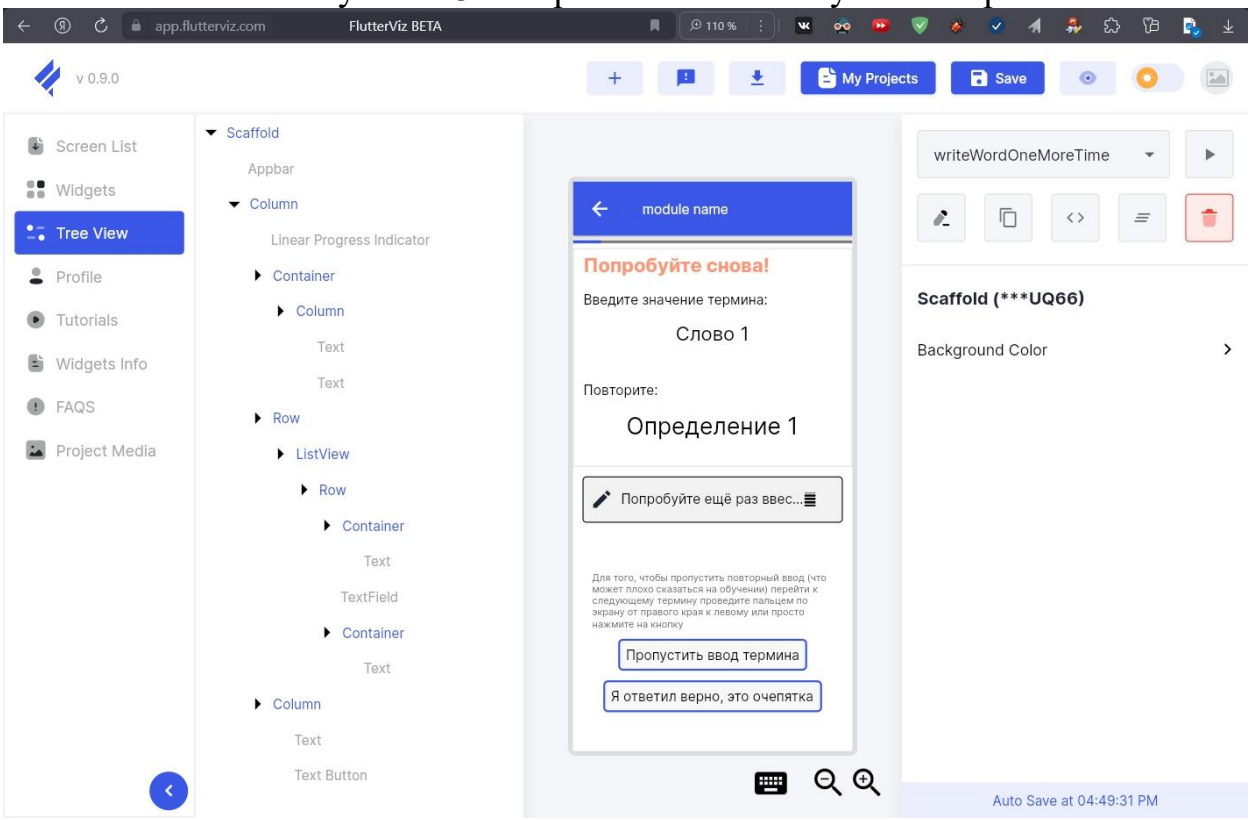


Рисунок Е.4 – Экран механики устранения пьсименной ошибки

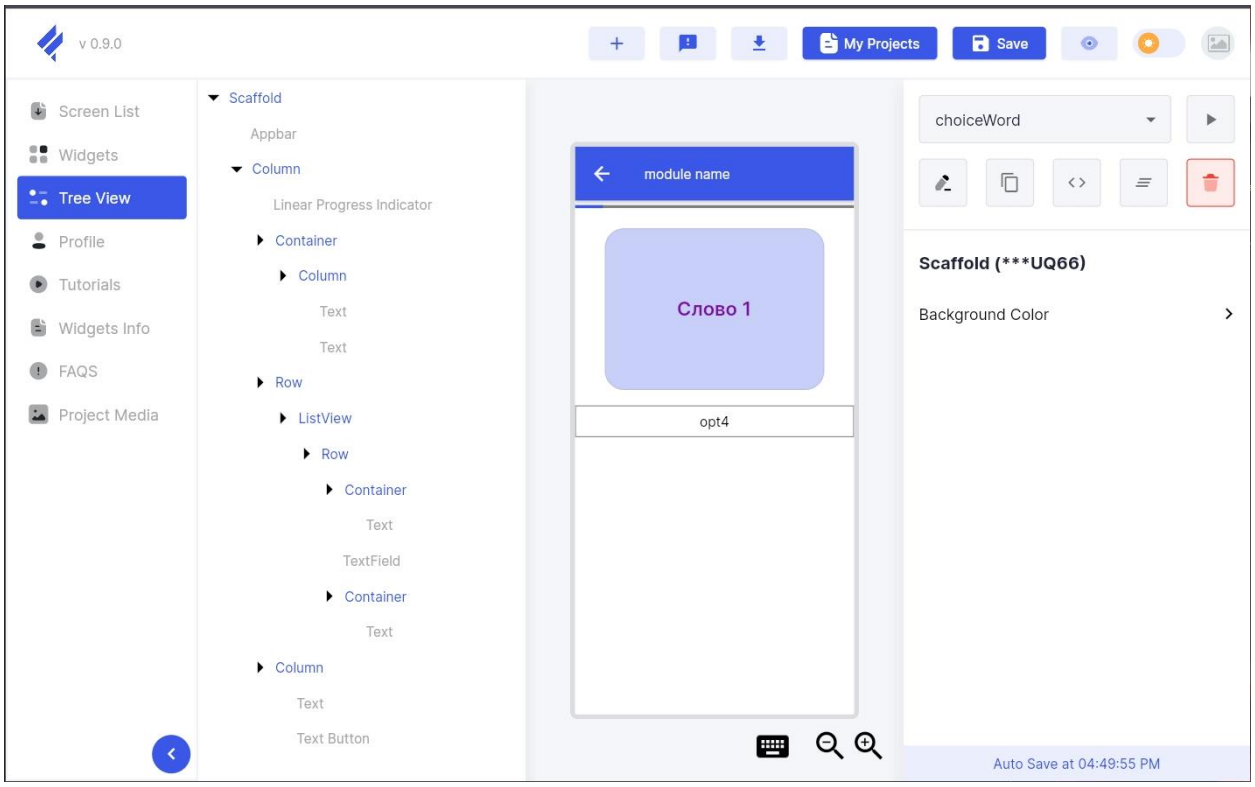


Рисунок Е.5 – Экран механики выбора

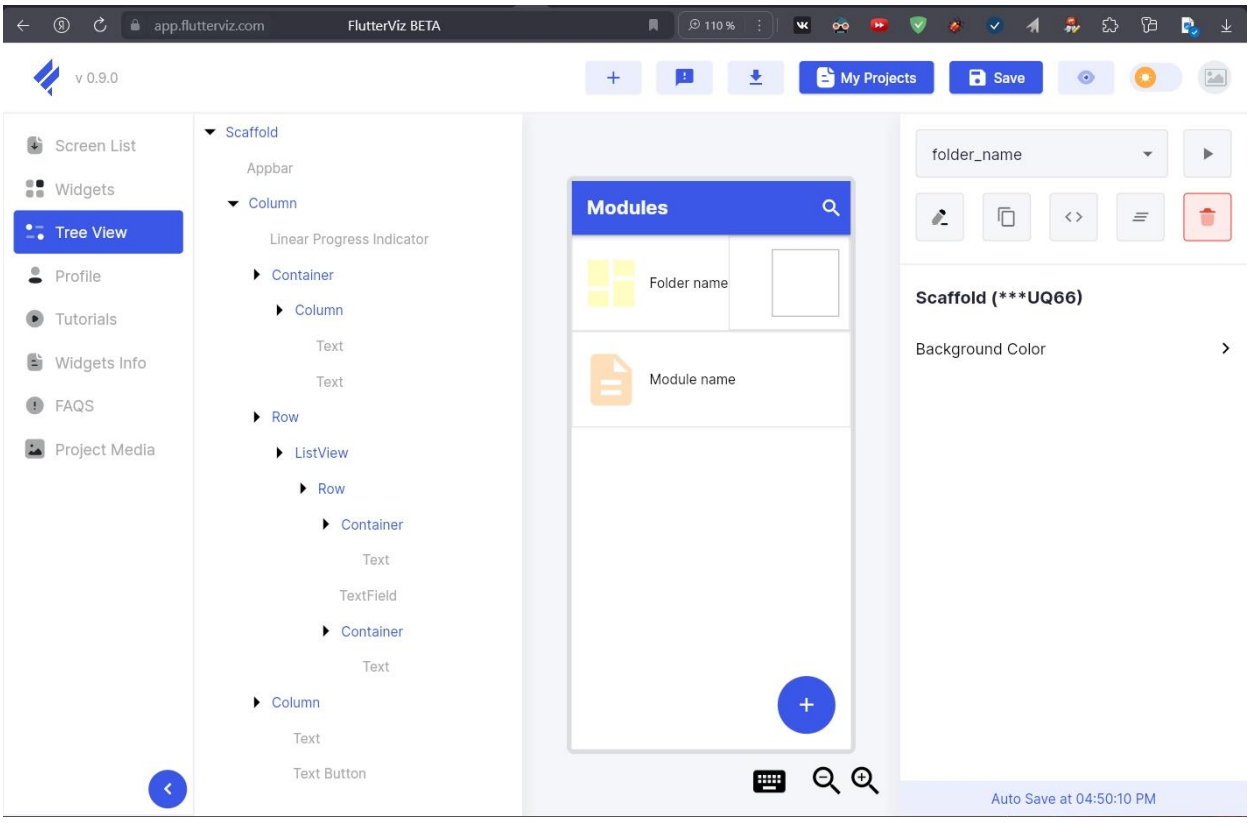


Рисунок Е.6 – Экран выбора аваки или модуля

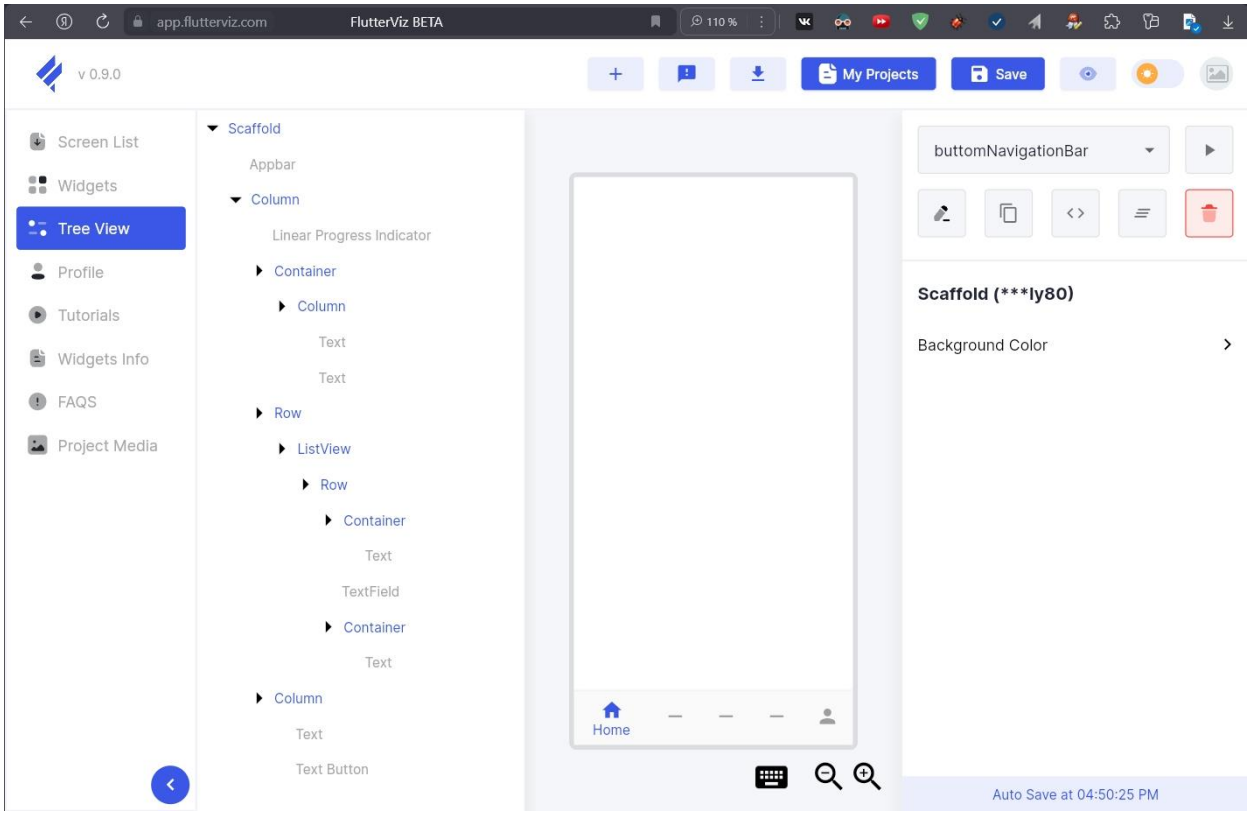


Рисунок Е.7 – Нижнее меню навигации

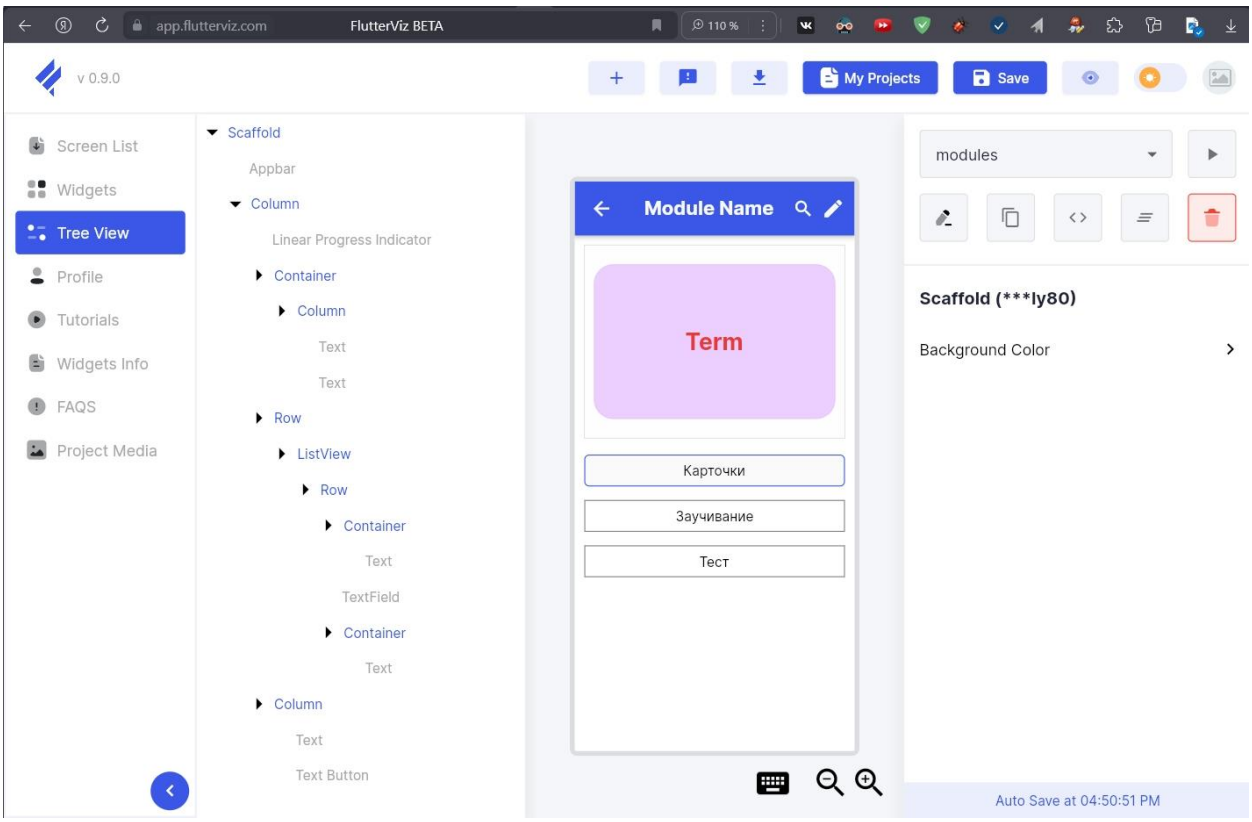


Рисунок Е.8 – Домашняя страница модуля

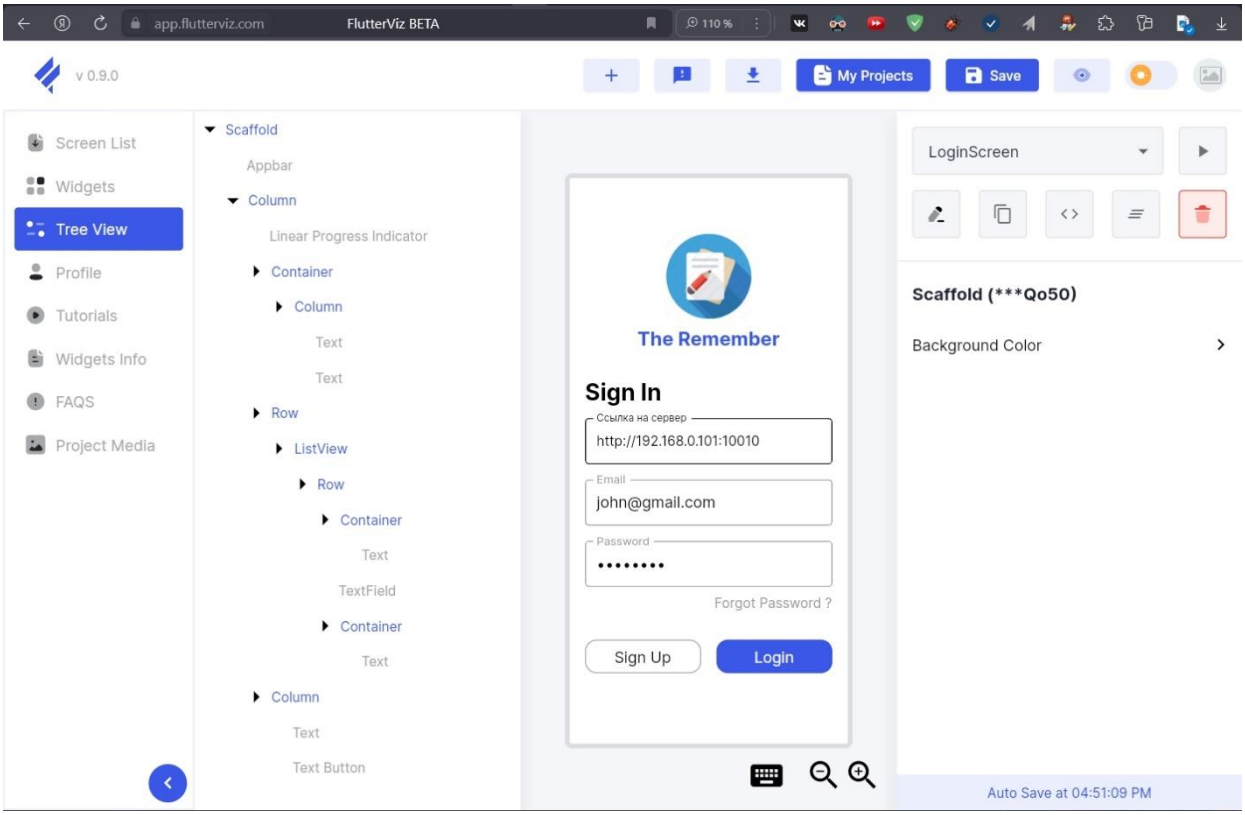


Рисунок Е.9 – Экран авторизации пользователя

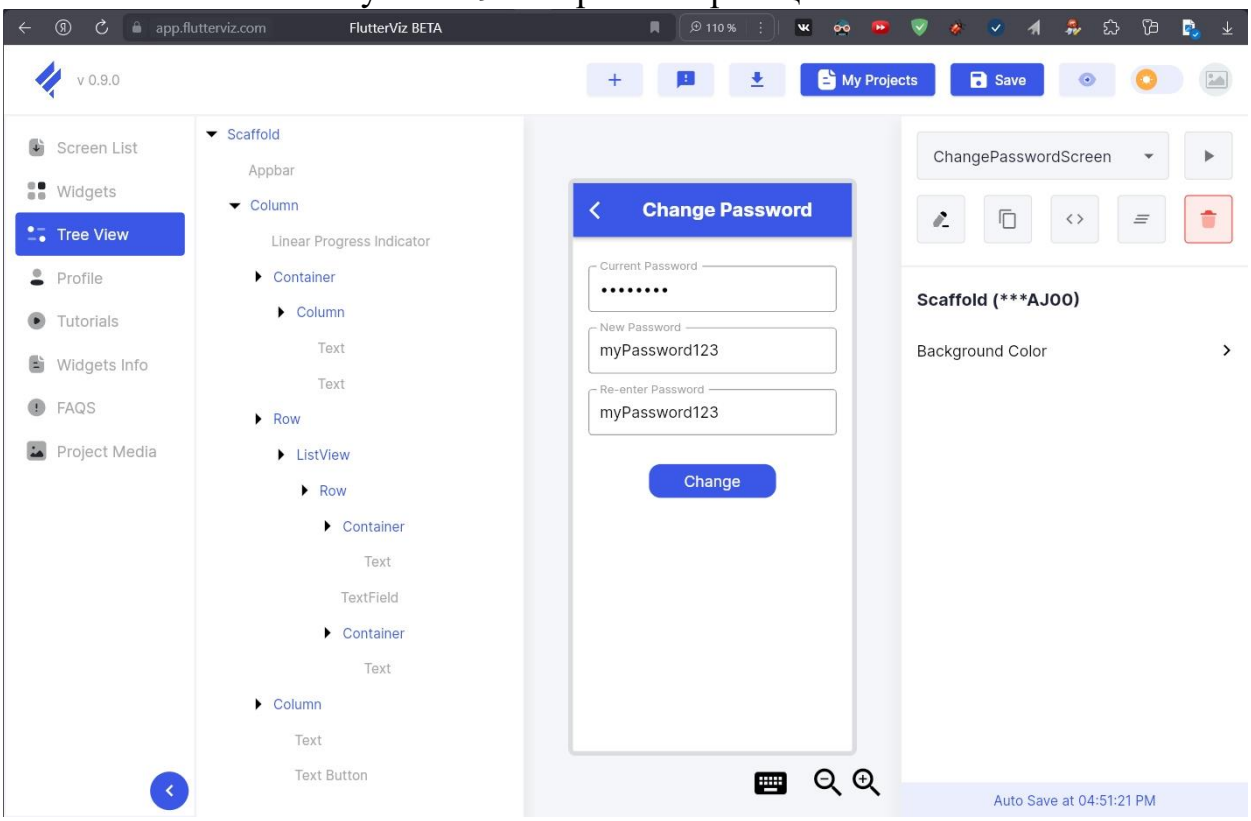


Рисунок Е.10 – Экран смены пароля

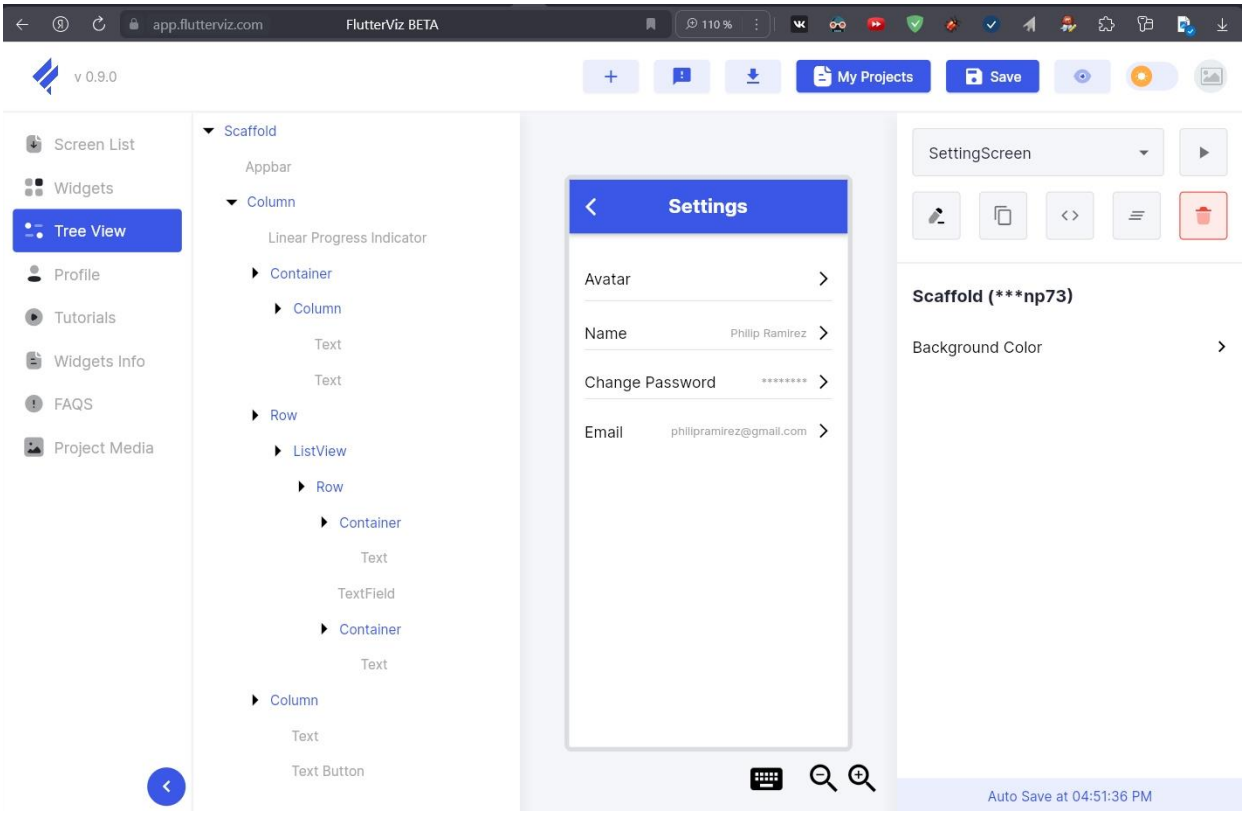


Рисунок Е.11 – Экран настроек пользователя

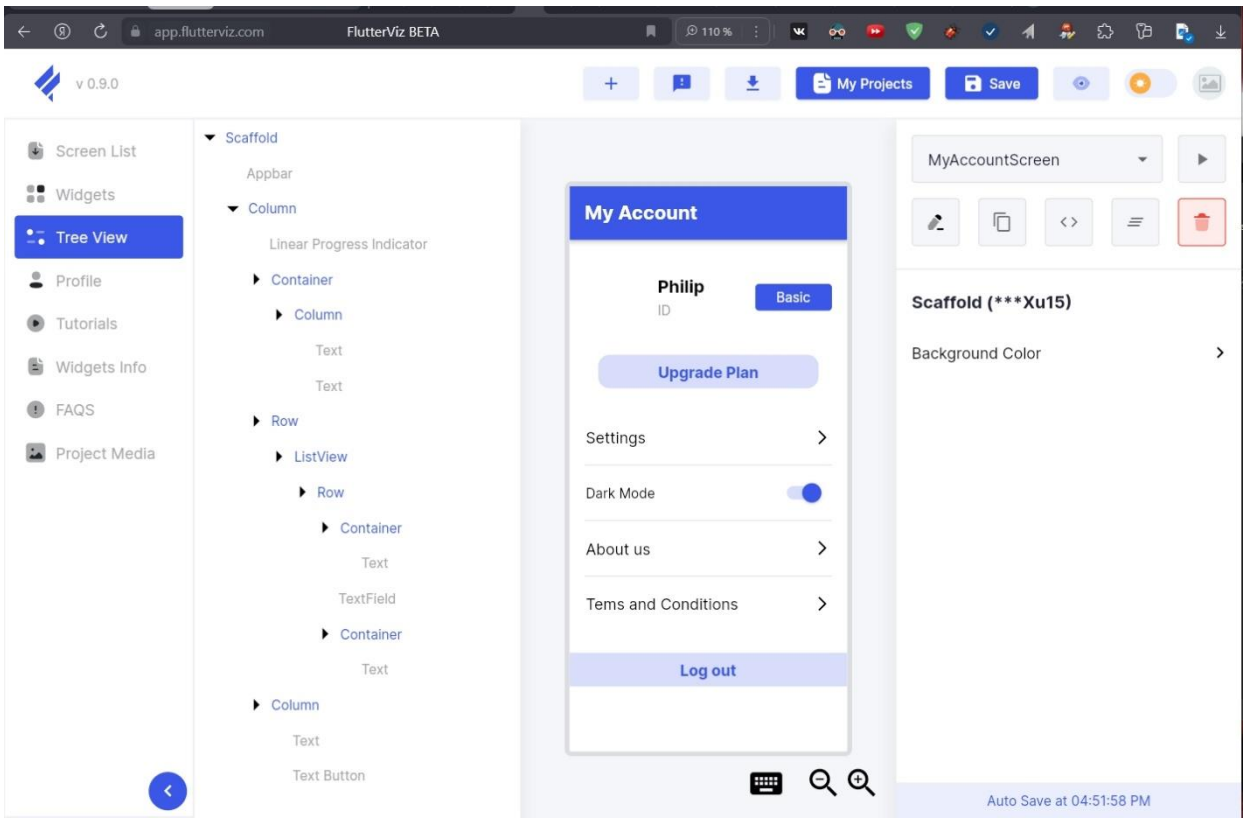


Рисунок Е.12 – Экран профиля пользователя

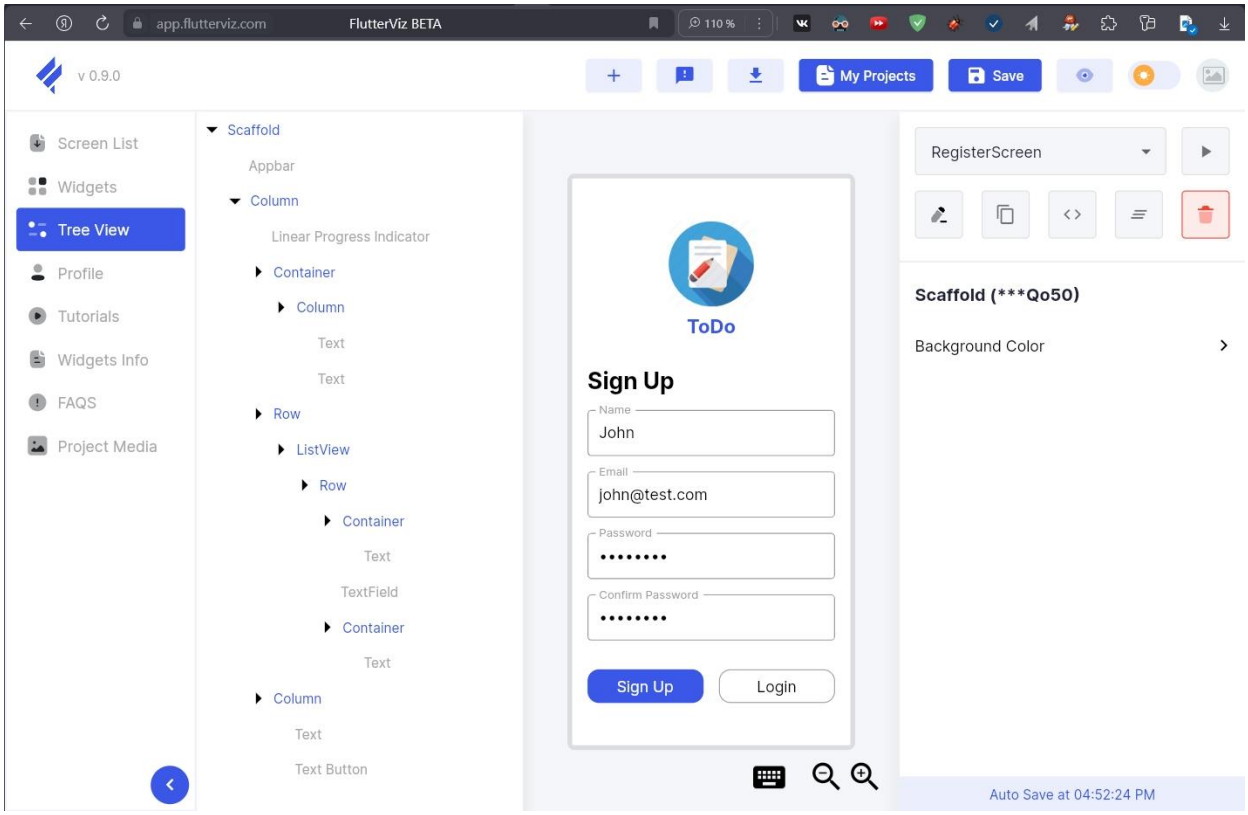


Рисунок Е.13 – Экран регистрации

ПРИЛОЖЕНИЕ Ж
(РЕКОМЕНДУЕМОЕ)
РЕАЛИЗАЦИЯ ПОЛЬЗОВАТЕЛЬСКОГО ИНТЕРФЕЙСА

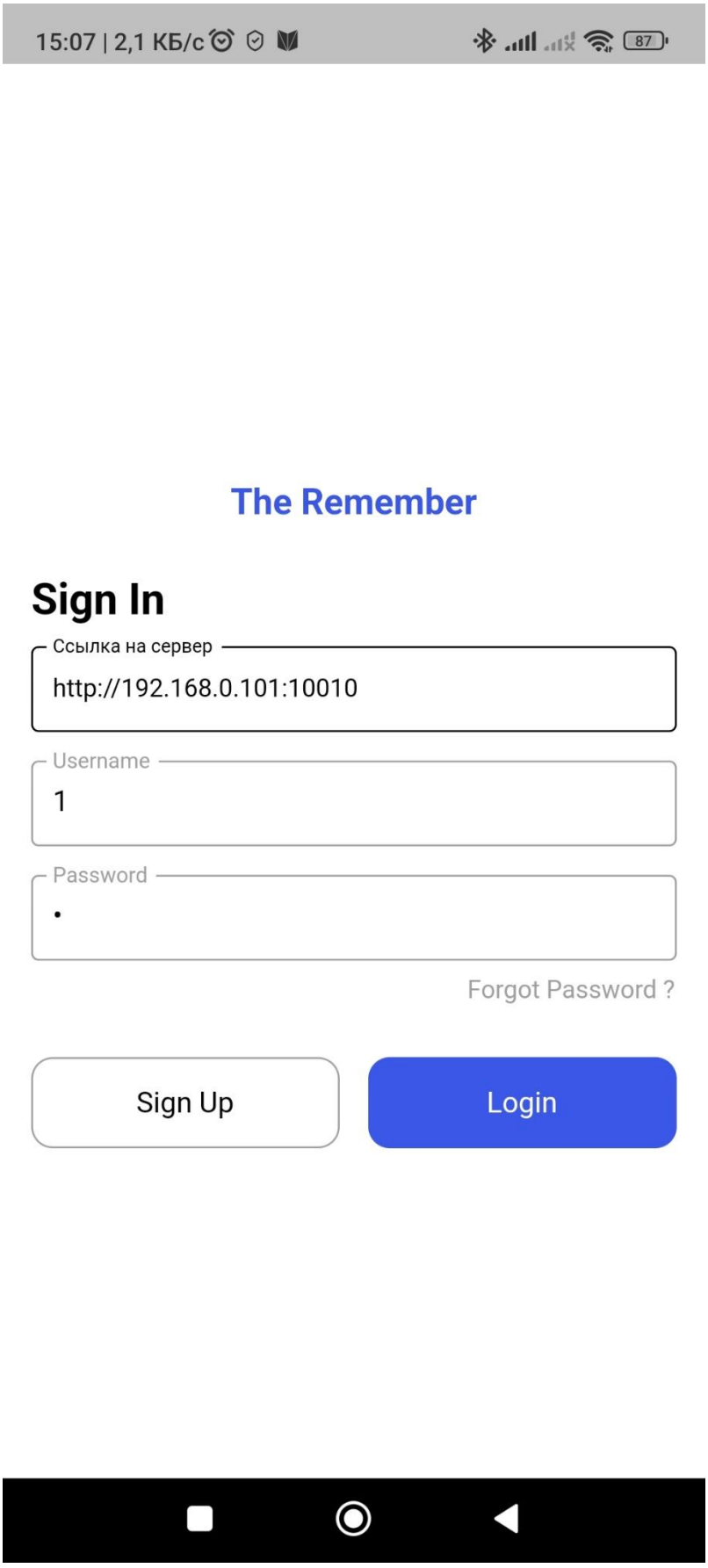


Рисунок Ж.1 – Экран авторизации

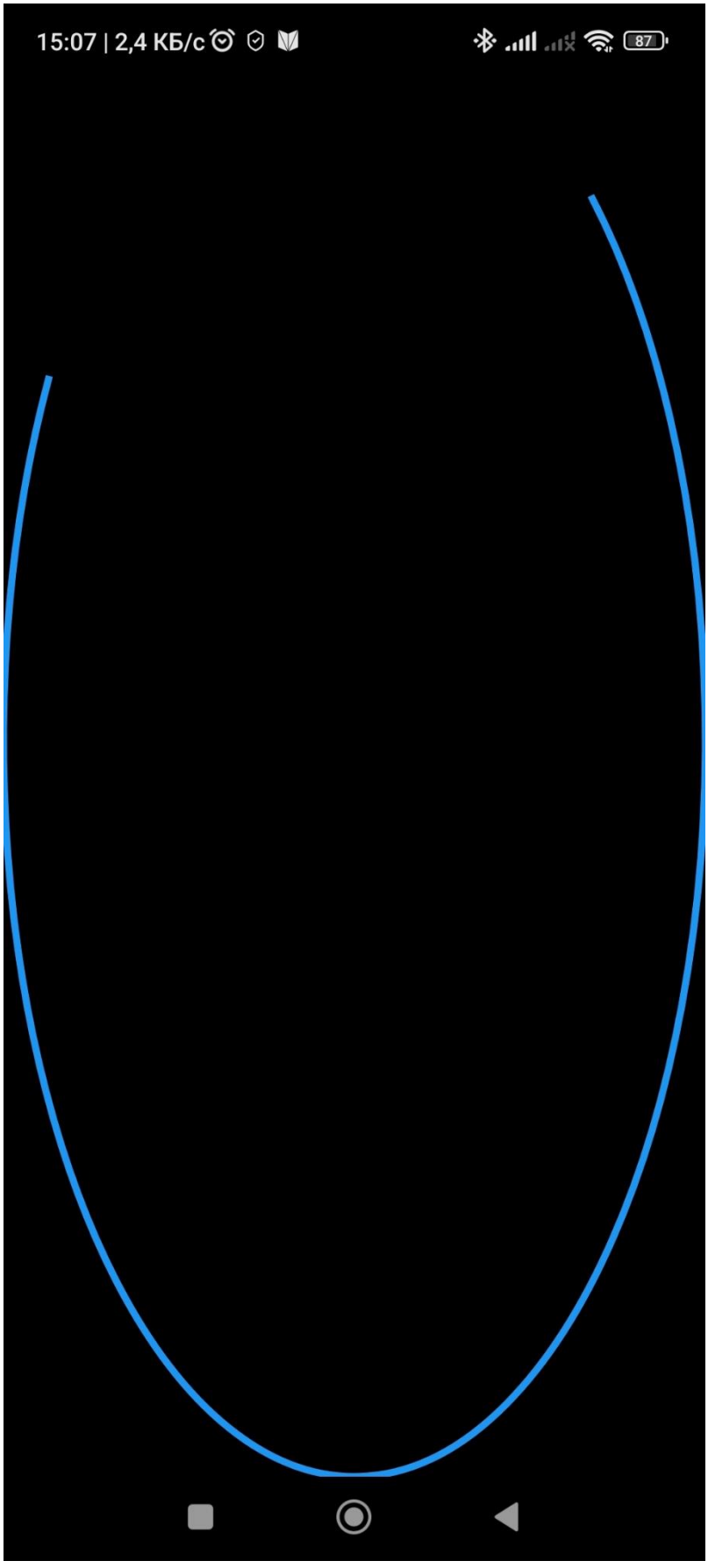


Рисунок Ж.2 – Экран ожидания ответа сервера

Рисунок Ж.3 – Экран корневой папки

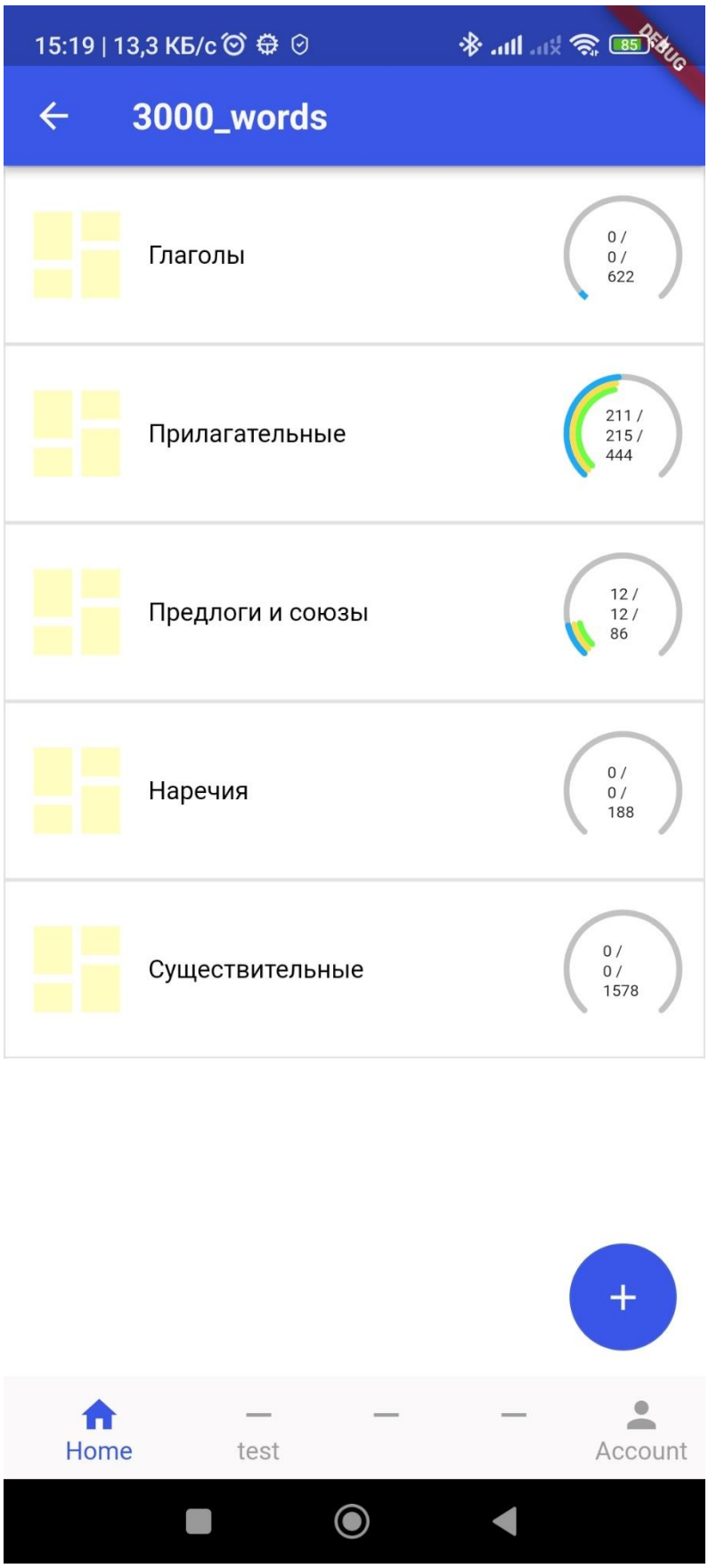


Рисунок Ж.4 – Экран вложенной папки

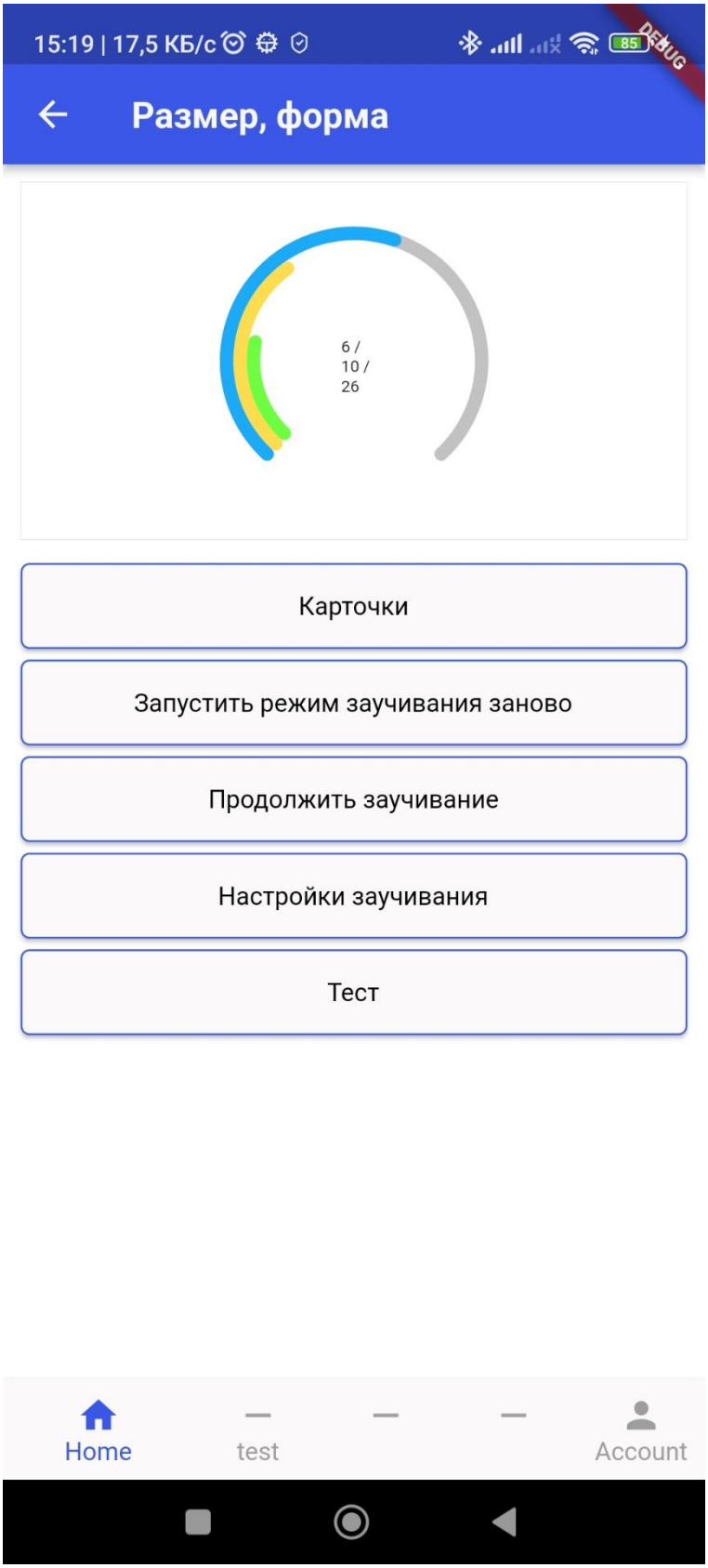


Рисунок Ж.5 – Экран модуля

Рисунок Ж.6 – Экран механики выбора

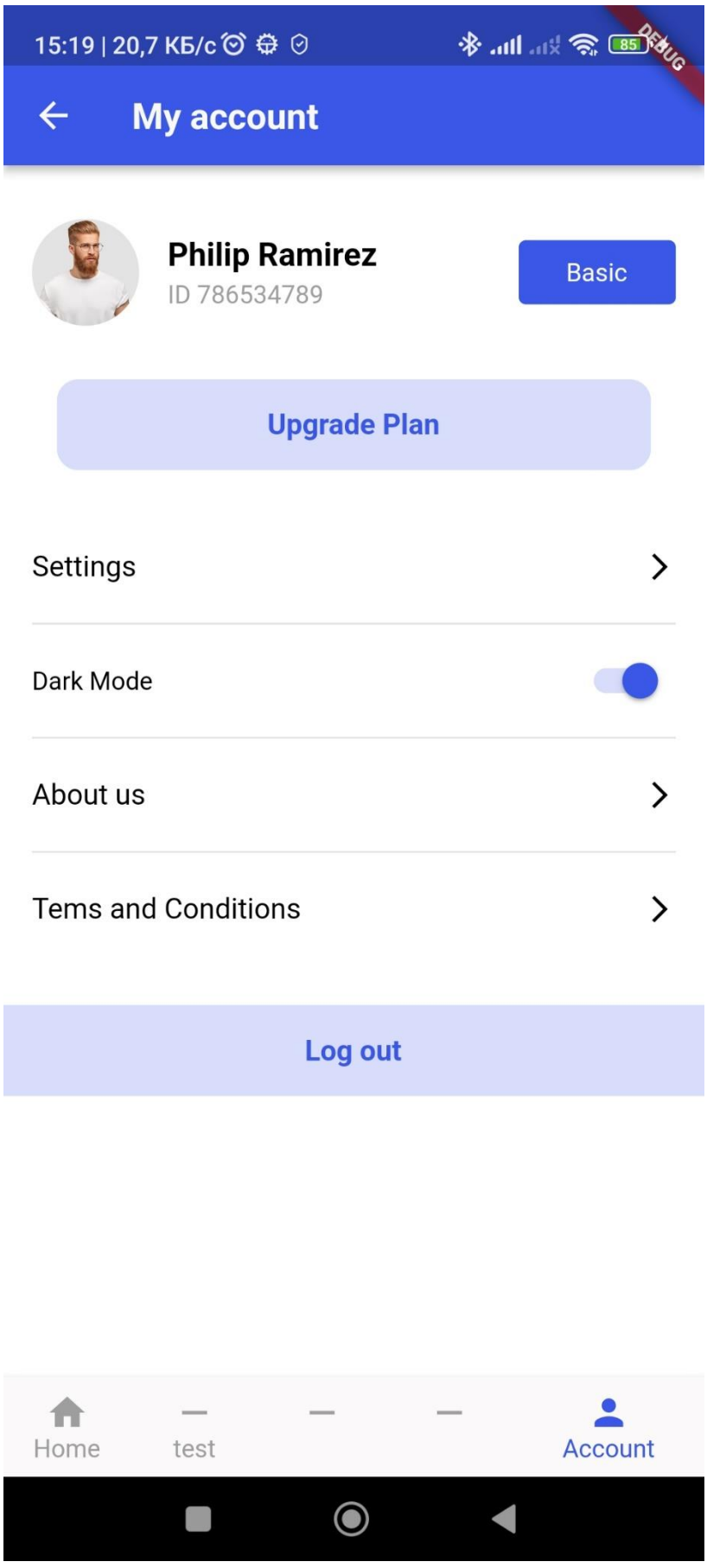


Рисунок Ж.7 – Экран профиля пользователя

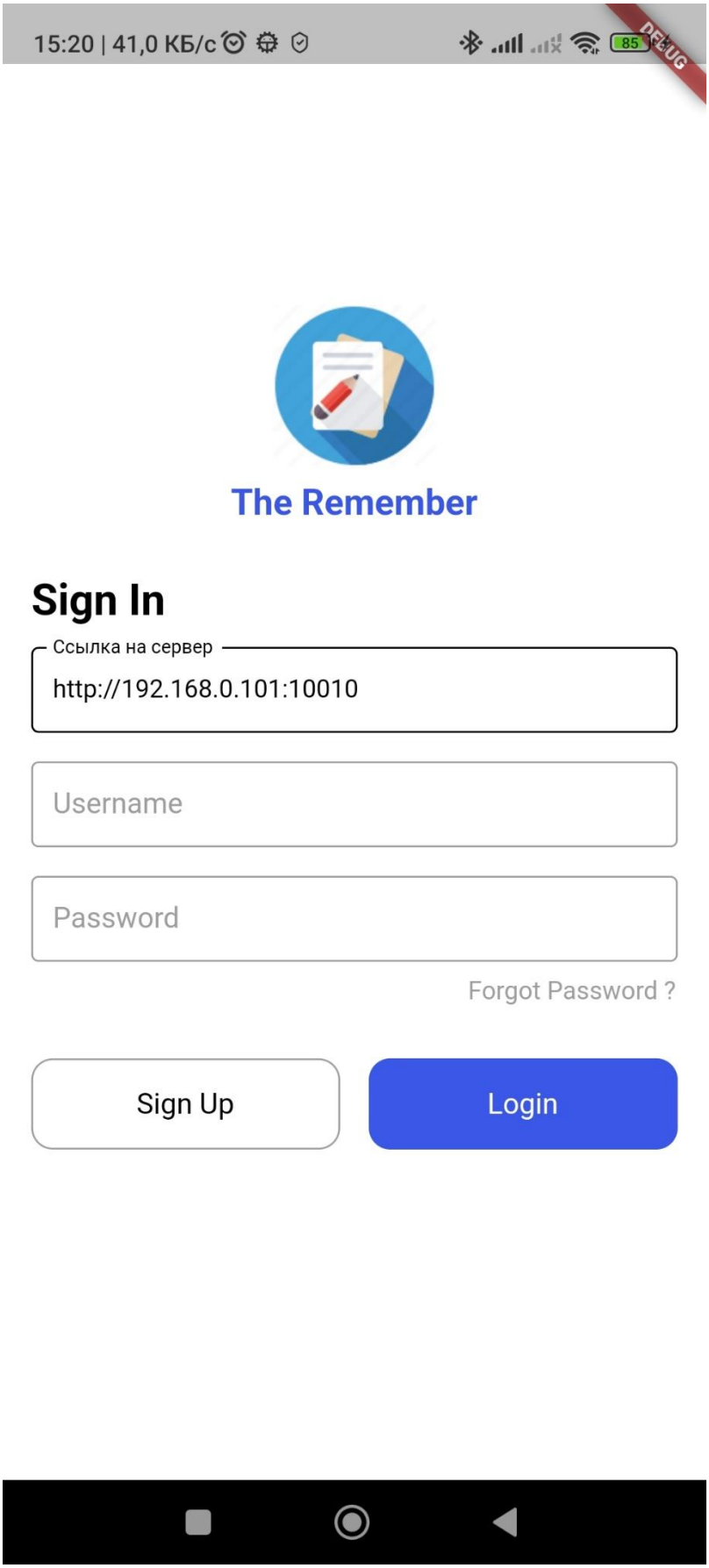


Рисунок Ж.8 – Экран регистрации пользователя

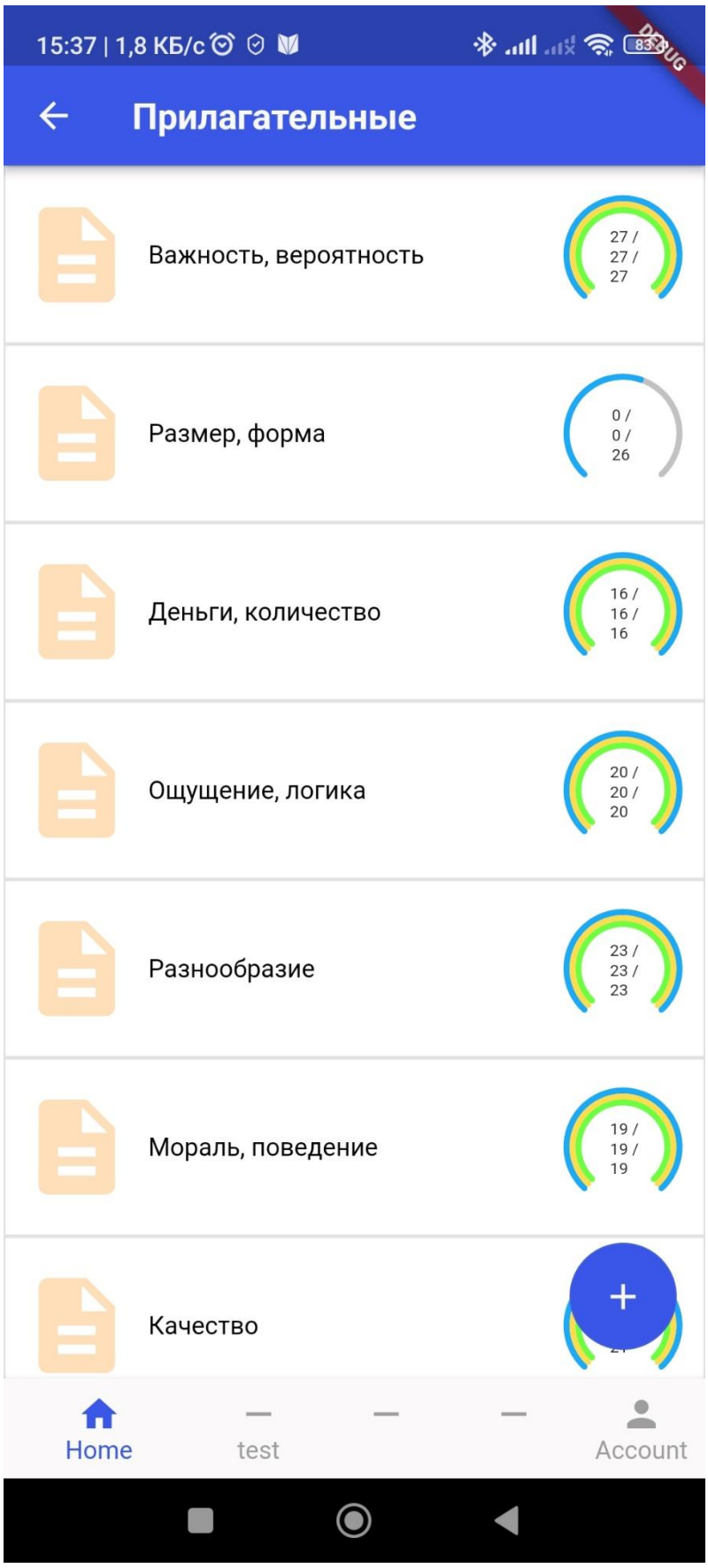


Рисунок Ж.9 – Экран папки, содержащей модули

ПРИЛОЖЕНИЕ И
(РЕКОМЕНДУЕМОЕ)
РЕЗУЛЬТАТЫ ФУНКЦИОНАЛЬНОГО ТЕСТИРОВАНИЯ

Листинг И.1 – Успешное создание термина


```
{
  "data": {
    "term": "string",
    "definition": "string",
    "id": "3fa85f64-5717-4562-b3fc-2c963f66afa6",
    "module_id": "3fa85f64-5717-4562-b3fc-2c963f66afa6",
    "created_at": "2024-05-31T13:32:57.214Z",
    "updated_at": "2024-05-31T13:32:57.214Z"
  },
  "errors": [
    "string"
  ],
  "info": {
    "api_version": "0.0.1",
    "deprecated": false
  }
}
```


Листинг И.2 – Создание термина без идентификатора модуля


```
{
  "data": null,
  "errors": [
    {
      "error_type": "validation",
      "message": null,
      "description": null,
      "status_code": 422,
      "detail": [
        {
          "loc": [
            "body",
            "module_id"
          ],
          "msg": "Field required",
          "type": "missing"
        }
      ],
      "body": {
        "term": "string",
        "definition": "string"
      }
    }
  ],
  "info": {
    "api_version": "0.0.1",
    "deprecated": false
  }
}
```


Листинг И.3 – Создание термина без идентификатора модуля


```
{
  "data": null,
  "errors": [
    {
      "error_type": "validation",
      "message": null
      "description": null,
      "status_code": 422,
```

```
      "detail": [
        {
          "loc": [
            "body",
            "definition"
          ],
          "msg": "Field required",
          "type": "missing"
        }
      ],
      "body": {
        "term": "string",
        "module_id": "3fa85f64-5717-4562-b3fc-2c963f66afa6"
      }
    }
  ],
  "info": {
    "api_version": "0.0.1",
    "deprecated": false
  }
}
```


Листинг И.4 – Создание термина с указанием несуществующего в бд модуля


```
{
  "data": null,
  "errors": [
    {
      "detail": "Conflict"
    }
  ],
  "info": {
    "api_version": "0.0.1",
    "deprecated": false
  }
}
```


Листинг И.5 – Создание термина неавторизированным пользователем


```
{
  "data": null,
  "errors": [
    {
      "error_type": "not_authenticated",
      "message": null
      "description": null,
      "status_code": 401,
      "detail": "Not authenticated"
    }
  ],
  "info": {
    "api_version": "0.0.1",
    "deprecated": false
  }
}
```


Листинг И.6 – Успешное обновление термина


```
{
  "data": {
    "term": "updated_term",
    "definition": "string",
```

```
    "id": "0bd1f733-87ab-4bdf-aa62-87be0ee7f83c",
    "module_id": "513384f3-8439-4ef7-912c-40534be9736a",
    "created_at": "2023-11-20T19:43:44.071330",
    "updated_at": "2024-05-31T14:15:44.197675"
  },
  "errors": null,
  "info": {
    "api_version": "0.0.1",
    "deprecated": false
  }
}
```


Листинг И.7 – Обновление термина с полем времени большим, чем текцщее


```
{
  "data": null,
  "errors": [
    {
      "error_type": "validation",
      "message": "not_set",
      "description": null,
      "status_code": 422,
      "detail": [
        {
          "loc": [
            "updated_at"
          ],
          "msg": "datetime is a future",
          "type": "uncorrected"
        }
      ],
      "body": {
        "term": "updated_term",
        "definition": "string",
        "updated_at": "2025-05-31T14:15:02.991000"
      }
    }
  ],
  "info": {
    "api_version": "0.0.1",
    "deprecated": false
  }
}
```


Листинг И.8 – Обновление с полем времени меньшим, чем установленное


```
{
  "data": null,
  "errors": [
    {
      "error_type": "data",
      "message": "time is less than current",
      "description": null,
      "status_code": 422,
      "detail": [
        "updated_at"
      ],
      "body": {
        "term": "string",
        "definition": "string",
        "updated_at": "2023-06-01T07:34:13.442Z"
       }
    }
```

```
  ],
  "info": {
    "api_version": "0.0.1",
    "deprecated": false
  }
}
```


Листинг И.9 – Обновление термина с несуществующим идентификатором


```
{
  "data": null,
  "errors": [
    {
      "error_type": "not_found",
      "message": "not_set",
      "description": null,
      "status_code": 404,
      "detail": {}
    }
  ],
  "info": {
    "api_version": "0.0.1",
    "deprecated": false
  }
}
```


Листинг И.10 – Обновление термина не автором термина


```
{
  "data": null,
  "errors": [
    {
      "error_type": "forbidden",
      "message": "not_set",
      "description": null,
      "status_code": 403,
      "detail": {}
    }
  ],
  "info": {
    "api_version": "0.0.1",
    "deprecated": false
  }
}
```


Листинг И.11 – Получение терминов из одного модуля с лимитом 3


```
{
  "data": [
    {
      "term": "earn",
      "definition": "зарабатывать",
      "id": "dca43768-3afd-47d9-8b73-3f315b45d084",
      "module_id": "06d677a2-f66e-40c9-8d89-75425cddf211",
      "created_at": "2023-08-17T12:52:32.312388",
      "updated_at": "2023-08-17T12:52:32.312388"
    },
    {
      "term": "make money (made, made)",
      "definition": "зарабатывать",
      "id": "64277a6e-127c-4d6a-9fea-13501388b530",
      "module_id": "06d677a2-f66e-40c9-8d89-75425cddf211",
```

```
      "created_at": "2023-08-17T12:52:32.312388",
      "updated_at": "2023-08-17T12:52:32.312388"
    },
    {
      "term": "cost (cost, cost)",
      "definition": "стоить",
      "id": "0e927d12-4fda-4f8a-ad14-cfdb7fd388cd",
      "module_id": "06d677a2-f66e-40c9-8d89-75425cddf211",
      "created_at": "2023-08-17T12:52:32.312388",
      "updated_at": "2023-08-17T12:52:32.312388"
    }
  ],
  "errors": null,
  "info": {
    "api_version": "0.0.1",
    "deprecated": false
  }
}
```


Листинг И.12 – Получение терминов из одного модуля с лимитом 10


```
{
  "data": [
    {
      "term": "earn",
      "definition": "зарабатывать",
      "id": "dca43768-3afd-47d9-8b73-3f315b45d084",
      "module_id": "06d677a2-f66e-40c9-8d89-75425cddf211",
      "created_at": "2023-08-17T12:52:32.312388",
      "updated_at": "2023-08-17T12:52:32.312388"
    },
    {
      "term": "make money (made, made)",
      "definition": "зарабатывать",
      "id": "64277a6e-127c-4d6a-9fea-13501388b530",
      "module_id": "06d677a2-f66e-40c9-8d89-75425cddf211",
      "created_at": "2023-08-17T12:52:32.312388",
      "updated_at": "2023-08-17T12:52:32.312388"
    },
    {
      "term": "promote",
      "definition": "продвигать, рекламировать, повышать по службе",
      "id": "2de2897b-f3e7-4f8c-a434-d67fc41529b8",
      "module_id": "06d677a2-f66e-40c9-8d89-75425cddf211",
      "created_at": "2023-08-17T12:52:32.312388",
      "updated_at": "2023-08-17T12:52:32.312388"
    },
    {
      "term": "charge",
      "definition": "брать плату",
      "id": "499b0474-1b85-4471-8a07-869bd874e2a7",
      "module_id": "06d677a2-f66e-40c9-8d89-75425cddf211",
      "created_at": "2023-08-17T12:52:32.312388",
      "updated_at": "2023-08-17T12:52:32.312388"
    },
    {
      "term": "estimate",
      "definition": "оценивать",
      "id": "c3fdd4b5-5e6b-443b-9062-b7ca23a2eb7f",
      "module_id": "06d677a2-f66e-40c9-8d89-75425cddf211",
      "created_at": "2023-08-17T12:52:32.312388",
      "updated_at": "2023-08-17T12:52:32.312388"
    },
```

```
    {
      "term": "account",
      "definition": "считать, вести учет",
      "id": "eda7249d-4094-4e27-bacb-cc09ad7fb8ff",
      "module_id": "06d677a2-f66e-40c9-8d89-75425cddf211",
      "created_at": "2023-08-17T12:52:32.312388",
      "updated_at": "2023-08-17T12:52:32.312388"
    },
    {
      "term": "invest",
      "definition": "инвестировать",
      "id": "a470e759-c9a4-4119-9533-6f048995c6b1",
      "module_id": "06d677a2-f66e-40c9-8d89-75425cddf211",
      "created_at": "2023-08-17T12:52:32.312388",
      "updated_at": "2023-08-17T12:52:32.312388"
    },
    {
      "term": "assess",
      "definition": "оценивать",
      "id": "72fb96c3-bb8a-4d58-ad4f-e080ae25e095",
      "module_id": "06d677a2-f66e-40c9-8d89-75425cddf211",
      "created_at": "2023-08-17T12:52:32.312388",
      "updated_at": "2023-08-17T12:52:32.312388"
    },
    {
      "term": "compete",
      "definition": "конкурировать",
      "id": "723b922f-dc8c-4cec-8be0-98e6e73fc452",
      "module_id": "06d677a2-f66e-40c9-8d89-75425cddf211",
      "created_at": "2023-08-17T12:52:32.312388",
      "updated_at": "2023-08-17T12:52:32.312388"
    },
    {
      "term": "cost (cost, cost)",
      "definition": "стоить",
      "id": "0e927d12-4fda-4f8a-ad14-cfdb7fd388cd",
      "module_id": "06d677a2-f66e-40c9-8d89-75425cddf211",
      "created_at": "2023-08-17T12:52:32.312388",
      "updated_at": "2023-08-17T12:52:32.312388"
    }
  ],
  "errors": null,
  "info": {
    "api_version": "0.0.1",
    "deprecated": false
  }
}
```


Листинг И.13 – Получение терминов из одного модуля с лимитом 100


```
{
  "data": [
    {
      "term": "cost (cost, cost)",
      "definition": "стоить",
      "id": "0e927d12-4fda-4f8a-ad14-cfdb7fd388cd",
      "module_id": "06d677a2-f66e-40c9-8d89-75425cddf211",
      "created_at": "2023-08-17T12:52:32.312388",
      "updated_at": "2023-08-17T12:52:32.312388"
    },
    {
      "term": "earn",
      "definition": "зарабатывать",
```

```
  "id": "dca43768-3afd-47d9-8b73-3f315b45d084",
  "module_id": "06d677a2-f66e-40c9-8d89-75425cddf211",
  "created_at": "2023-08-17T12:52:32.312388",
  "updated_at": "2023-08-17T12:52:32.312388"
},
{
  "term": "make money (made, made)",
  "definition": "зарабатывать",
  "id": "64277a6e-127c-4d6a-9fea-13501388b530",
  "module_id": "06d677a2-f66e-40c9-8d89-75425cddf211",
  "created_at": "2023-08-17T12:52:32.312388",
  "updated_at": "2023-08-17T12:52:32.312388"
},
{
  "term": "promote",
  "definition": "продвигать, рекламировать, повышать по службе",
  "id": "2de2897b-f3e7-4f8c-a434-d67fc41529b8",
  "module_id": "06d677a2-f66e-40c9-8d89-75425cddf211",
  "created_at": "2023-08-17T12:52:32.312388",
  "updated_at": "2023-08-17T12:52:32.312388"
},
{
  "term": "charge",
  "definition": "брать плату",
  "id": "499b0474-1b85-4471-8a07-869bd874e2a7",
  "module_id": "06d677a2-f66e-40c9-8d89-75425cddf211",
  "created_at": "2023-08-17T12:52:32.312388",
  "updated_at": "2023-08-17T12:52:32.312388"
},
{
  "term": "estimate",
  "definition": "оценивать",
  "id": "c3fdd4b5-5e6b-443b-9062-b7ca23a2eb7f",
  "module_id": "06d677a2-f66e-40c9-8d89-75425cddf211",
  "created_at": "2023-08-17T12:52:32.312388",
  "updated_at": "2023-08-17T12:52:32.312388"
},
{
  "term": "account",
  "definition": "считать, вести учет",
  "id": "eda7249d-4094-4e27-bacb-cc09ad7fb8ff",
  "module_id": "06d677a2-f66e-40c9-8d89-75425cddf211",
  "created_at": "2023-08-17T12:52:32.312388",
  "updated_at": "2023-08-17T12:52:32.312388"
},
{
  "term": "invest",
  "definition": "инвестировать",
  "id": "a470e759-c9a4-4119-9533-6f048995c6b1",
  "module_id": "06d677a2-f66e-40c9-8d89-75425cddf211",
  "created_at": "2023-08-17T12:52:32.312388",
  "updated_at": "2023-08-17T12:52:32.312388"
},
{
  "term": "assess",
  "definition": "оценивать",
  "id": "72fb96c3-bb8a-4d58-ad4f-e080ae25e095",
  "module_id": "06d677a2-f66e-40c9-8d89-75425cddf211",
  "created_at": "2023-08-17T12:52:32.312388",
  "updated_at": "2023-08-17T12:52:32.312388"
},
{
  "term": "compete",
  "definition": "конкурировать",
```

```
  "id": "723b922f-dc8c-4cec-8be0-98e6e73fc452",
  "module_id": "06d677a2-f66e-40c9-8d89-75425cddf211",
  "created_at": "2023-08-17T12:52:32.312388",
  "updated_at": "2023-08-17T12:52:32.312388"
},
{
  "term": "purchase",
  "definition": "покупать",
  "id": "f784ae8d-b6c9-4682-8346-1e4d891d72f6",
  "module_id": "06d677a2-f66e-40c9-8d89-75425cddf211",
  "created_at": "2023-08-17T12:52:32.312388",
  "updated_at": "2023-08-17T12:52:32.312388"
},
{
  "term": "evaluate",
  "definition": "оценивать",
  "id": "c2bcd52f-caa4-4b47-a81b-0f0179cf9fa0",
  "module_id": "06d677a2-f66e-40c9-8d89-75425cddf211",
  "created_at": "2023-08-17T12:52:32.312388",
  "updated_at": "2023-08-17T12:52:32.312388"
},
{
  "term": "trade",
  "definition": "торговать",
  "id": "044676da-a33a-42ae-9d98-6ce853cf7846",
  "module_id": "06d677a2-f66e-40c9-8d89-75425cddf211",
  "created_at": "2023-08-17T12:52:32.312388",
  "updated_at": "2023-08-17T12:52:32.312388"
},
{
  "term": "market",
  "definition": "продвигать на рынке (рекламировать)",
  "id": "95f3ac6a-dd35-49d2-8826-8009a44e360d",
  "module_id": "06d677a2-f66e-40c9-8d89-75425cddf211",
  "created_at": "2023-08-17T12:52:32.312388",
  "updated_at": "2023-08-17T12:52:32.312388"
},
{
  "term": "withdraw",
  "definition": "снимать (деньги со счета в банке)",
  "id": "28ad02e5-647d-4dc7-b263-7dd7c1735e65",
  "module_id": "06d677a2-f66e-40c9-8d89-75425cddf211",
  "created_at": "2023-08-17T12:52:32.312388",
  "updated_at": "2023-08-17T12:52:32.312388"
},
{
  "term": "convert",
  "definition": "конвертировать",
  "id": "e0e298a5-4ad6-4c6d-bc56-dab9413b6daa",
  "module_id": "06d677a2-f66e-40c9-8d89-75425cddf211",
  "created_at": "2023-08-17T12:52:32.312388",
  "updated_at": "2023-08-17T12:52:32.312388"
},
{
  "term": "spend (spent, spent)",
  "definition": "тратить (напр. деньги), проводить время",
  "id": "fdf9bf83-51d3-4df8-9ee2-10ac33a1e4bd",
  "module_id": "06d677a2-f66e-40c9-8d89-75425cddf211",
  "created_at": "2023-08-17T12:52:32.312388",
  "updated_at": "2023-08-17T12:52:32.312388"
},
{
  "term": "buy (bought, bought)",
  "definition": "покупать",
```

```
  "id": "78bf43c6-e5b4-43ea-87e1-84fae74365eb",
  "module_id": "06d677a2-f66e-40c9-8d89-75425cddf211",
  "created_at": "2023-08-17T12:52:32.312388",
  "updated_at": "2023-08-17T12:52:32.312388"
},
{
  "term": "sell (sold, sold)",
  "definition": "продавать",
  "id": "723f2f7e-bb02-4263-bcae-3b6d40e14ec4",
  "module_id": "06d677a2-f66e-40c9-8d89-75425cddf211",
  "created_at": "2023-08-17T12:52:32.312388",
  "updated_at": "2023-08-17T12:52:32.312388"
},
{
  "term": "deal (dealt, dealt)",
  "definition": "вести дела",
  "id": "a23e657b-2880-4990-be2f-0c22d6bef186",
  "module_id": "06d677a2-f66e-40c9-8d89-75425cddf211",
  "created_at": "2023-08-17T12:52:32.312388",
  "updated_at": "2023-08-17T12:52:32.312388"
},
{
  "term": "produce",
  "definition": "производить",
  "id": "7388a982-85df-42f2-9363-b1d0f6549738",
  "module_id": "06d677a2-f66e-40c9-8d89-75425cddf211",
  "created_at": "2023-08-17T12:52:32.312388",
  "updated_at": "2023-08-17T12:52:32.312388"
},
{
  "term": "save",
  "definition": "делать сбережения, экономить",
  "id": "349589f8-8a48-4358-929d-02ce0f9b0435",
  "module_id": "06d677a2-f66e-40c9-8d89-75425cddf211",
  "created_at": "2023-08-17T12:52:32.312388",
  "updated_at": "2023-08-17T12:52:32.312388"
},
{
  "term": "rule",
  "definition": "править (властвовать)",
  "id": "175a3b99-3f9c-4bf5-8dcf-4c45f41f81ba",
  "module_id": "06d677a2-f66e-40c9-8d89-75425cddf211",
  "created_at": "2023-08-17T12:52:32.312388",
  "updated_at": "2023-08-17T12:52:32.312388"
},
{
  "term": "elect",
  "definition": "выбирать (на выборах)",
  "id": "fecb963a-17fa-4740-a954-db43ab2fc938",
  "module_id": "06d677a2-f66e-40c9-8d89-75425cddf211",
  "created_at": "2023-08-17T12:52:32.312388",
  "updated_at": "2023-08-17T12:52:32.312388"
},
{
  "term": "vote",
  "definition": "голосовать",
  "id": "ac0aac4d-d725-411a-9622-966a6d701c58",
  "module_id": "06d677a2-f66e-40c9-8d89-75425cddf211",
  "created_at": "2023-08-17T12:52:32.312388",
  "updated_at": "2023-08-17T12:52:32.312388"
},
{
  "term": "contribute",
  "definition": "вносить вклад во что-то",
```

```
      "id": "9f65debb-8d8a-48e0-8f1d-eabad602e89e",
      "module_id": "06d677a2-f66e-40c9-8d89-75425cddf211",
      "created_at": "2023-08-17T12:52:32.312388",
      "updated_at": "2023-08-17T12:52:32.312388"
    },
    {
      "term": "settle",
      "definition": "расселяться (создавать поселение)",
      "id": "748e3ee3-9f88-458f-a28e-8d2cf4079a74",
      "module_id": "06d677a2-f66e-40c9-8d89-75425cddf211",
      "created_at": "2023-08-17T12:52:32.312388",
      "updated_at": "2023-08-17T12:52:32.312388"
    },
    {
      "term": "impose",
      "definition": "налагать (напр. обязательство)",
      "id": "5a19be98-4a18-4bcc-9f75-62f61511538a",
      "module_id": "06d677a2-f66e-40c9-8d89-75425cddf211",
      "created_at": "2023-08-17T12:52:32.312388",
      "updated_at": "2023-08-17T12:52:32.312388"
    },
    {
      "term": "transfer",
      "definition": "передавать, переводить (напр. деньги)",
      "id": "d5bbcb02-e75c-4d08-b0d1-6a9e72a8cc99",
      "module_id": "06d677a2-f66e-40c9-8d89-75425cddf211",
      "created_at": "2023-08-17T12:52:32.312388",
      "updated_at": "2023-08-17T12:52:32.312388"
    },
    {
      "term": "distribute",
      "definition": "распространять",
      "id": "4ab64ff0-b7b1-4b66-9667-92ec5329eb8e",
      "module_id": "06d677a2-f66e-40c9-8d89-75425cddf211",
      "created_at": "2023-08-17T12:52:32.312388",
      "updated_at": "2023-08-17T12:52:32.312388"
    },
    {
      "term": "deposit",
      "definition": "вносить деньги (на счет в банке)",
      "id": "ef0449e2-595d-496b-ad84-cb377cf0c1de",
      "module_id": "06d677a2-f66e-40c9-8d89-75425cddf211",
      "created_at": "2023-08-17T12:52:32.312388",
      "updated_at": "2023-08-17T12:52:32.312388"
    },
    {
      "term": "afford",
      "definition": "повзволять себе",
      "id": "7c5b87f8-77f3-41cd-95a7-06282311335c",
      "module_id": "06d677a2-f66e-40c9-8d89-75425cddf211",
      "created_at": "2023-08-17T12:52:32.312388",
      "updated_at": "2023-08-17T12:52:32.312388"
    }
  ],
  "errors": null,
  "info": {
    "api_version": "0.0.1",
    "deprecated": false
  }
}
```


Листинг И.14 – Получение терминов из одного модуля с лимитом и оффсетом

```
{
  "data": [
    {
      "term": "charge",
      "definition": "брать плату",
      "id": "499b0474-1b85-4471-8a07-869bd874e2a7",
      "module_id": "06d677a2-f66e-40c9-8d89-75425cddf211",
      "created_at": "2023-08-17T12:52:32.312388",
      "updated_at": "2023-08-17T12:52:32.312388"
    },
    {
      "term": "estimate",
      "definition": "оценивать",
      "id": "c3fdd4b5-5e6b-443b-9062-b7ca23a2eb7f",
      "module_id": "06d677a2-f66e-40c9-8d89-75425cddf211",
      "created_at": "2023-08-17T12:52:32.312388",
      "updated_at": "2023-08-17T12:52:32.312388"
    },
    {
      "term": "cost (cost, cost)",
      "definition": "стоить",
      "id": "0e927d12-4fda-4f8a-ad14-cfdb7fd388cd",
      "module_id": "06d677a2-f66e-40c9-8d89-75425cddf211",
      "created_at": "2023-08-17T12:52:32.312388",
      "updated_at": "2023-08-17T12:52:32.312388"
    }
  ],
  "errors": null,
  "info": {
    "api_version": "0.0.1",
    "deprecated": false
  }
}
```


Листинг И.15 – Получение терминов из одного модуля с лимитом и оффсетом


```
{
  "data": [
    {
      "term": "make money (made, made)",
      "definition": "зарабатывать",
      "id": "64277a6e-127c-4d6a-9fea-13501388b530",
      "module_id": "06d677a2-f66e-40c9-8d89-75425cddf211",
      "created_at": "2023-08-17T12:52:32.312388",
      "updated_at": "2023-08-17T12:52:32.312388"
    },
    {
      "term": "promote",
      "definition": "продвигать, рекламировать, повышать по службе",
      "id": "2de2897b-f3e7-4f8c-a434-d67fc41529b8",
      "module_id": "06d677a2-f66e-40c9-8d89-75425cddf211",
      "created_at": "2023-08-17T12:52:32.312388",
      "updated_at": "2023-08-17T12:52:32.312388"
    },
    {
      "term": "charge",
      "definition": "брать плату",
      "id": "499b0474-1b85-4471-8a07-869bd874e2a7",
      "module_id": "06d677a2-f66e-40c9-8d89-75425cddf211",
      "created_at": "2023-08-17T12:52:32.312388",
      "updated_at": "2023-08-17T12:52:32.312388"
    },
```

```
    {
      "term": "estimate",
      "definition": "оценивать",
      "id": "c3fdd4b5-5e6b-443b-9062-b7ca23a2eb7f",
      "module_id": "06d677a2-f66e-40c9-8d89-75425cddf211",
      "created_at": "2023-08-17T12:52:32.312388",
      "updated_at": "2023-08-17T12:52:32.312388"
    },
    {
      "term": "account",
      "definition": "считать, вести учет",
      "id": "eda7249d-4094-4e27-bacb-cc09ad7fb8ff",
      "module_id": "06d677a2-f66e-40c9-8d89-75425cddf211",
      "created_at": "2023-08-17T12:52:32.312388",
      "updated_at": "2023-08-17T12:52:32.312388"
    },
    {
      "term": "cost (cost, cost)",
      "definition": "стоить",
      "id": "0e927d12-4fda-4f8a-ad14-cfdb7fd388cd",
      "module_id": "06d677a2-f66e-40c9-8d89-75425cddf211",
      "created_at": "2023-08-17T12:52:32.312388",
      "updated_at": "2023-08-17T12:52:32.312388"
    }
  ],
  "errors": null,
  "info": {
    "api_version": "0.0.1",
    "deprecated": false
  }
}
```


Листинг И.16 – Получение терминов из одного модуля с лимитом и оффсетом


```
{
  "data": [],
  "errors": null,
  "info": {
    "api_version": "0.0.1",
    "deprecated": false
  }
}
```


Листинг И.17 – Получение терминов из одного модуля с лимитом и оффсетом


```
{
  "data": [
    {
      "term": "deposit",
      "definition": "вносить деньги (на счет в банке)",
      "id": "ef0449e2-595d-496b-ad84-cb377cf0c1de",
      "module_id": "06d677a2-f66e-40c9-8d89-75425cddf211",
      "created_at": "2023-08-17T12:52:32.312388",
      "updated_at": "2023-08-17T12:52:32.312388"
    },
    {
      "term": "afford",
      "definition": "повзволять себе",
      "id": "7c5b87f8-77f3-41cd-95a7-06282311335c",
      "module_id": "06d677a2-f66e-40c9-8d89-75425cddf211",
```

```
      "created_at": "2023-08-17T12:52:32.312388",
      "updated_at": "2023-08-17T12:52:32.312388"
    }
  ],
  "errors": null,
  "info": {
    "api_version": "0.0.1",
    "deprecated": false
  }
}
```


Листинг И.18 – Получение терминов из одного модуля, переданного дважды


```
{
  "data": [
    {
      "term": "deposit",
      "definition": "вносить деньги (на счет в банке)",
      "id": "ef0449e2-595d-496b-ad84-cb377cf0c1de",
      "module_id": "06d677a2-f66e-40c9-8d89-75425cddf211",
      "created_at": "2023-08-17T12:52:32.312388",
      "updated_at": "2023-08-17T12:52:32.312388"
    },
    {
      "term": "afford",
      "definition": "повзволять себе",
      "id": "7c5b87f8-77f3-41cd-95a7-06282311335c",
      "module_id": "06d677a2-f66e-40c9-8d89-75425cddf211",
      "created_at": "2023-08-17T12:52:32.312388",
      "updated_at": "2023-08-17T12:52:32.312388"
    }
  ],
  "errors": null,
  "info": {
    "api_version": "0.0.1",
    "deprecated": false
  }
}
```


Листинг И.19 – Получение терминов из одного модуля, переданного пять раз


```
{
  "data": [
    {
      "term": "deposit",
      "definition": "вносить деньги (на счет в банке)",
      "id": "ef0449e2-595d-496b-ad84-cb377cf0c1de",
      "module_id": "06d677a2-f66e-40c9-8d89-75425cddf211",
      "created_at": "2023-08-17T12:52:32.312388",
      "updated_at": "2023-08-17T12:52:32.312388"
    },
    {
      "term": "afford",
      "definition": "повзволять себе",
      "id": "7c5b87f8-77f3-41cd-95a7-06282311335c",
      "module_id": "06d677a2-f66e-40c9-8d89-75425cddf211",
      "created_at": "2023-08-17T12:52:32.312388",
      "updated_at": "2023-08-17T12:52:32.312388"
    }
  ],
  "errors": null,
  "info": {
    "api_version": "0.0.1",
```

```
    "deprecated": false
  }
}
```


Листинг И.20 – Получение терминов из двух модулей с лимитом и оффсетом


```
{
  "data": [
    {
      "term": "deposit",
      "definition": "вносить деньги (на счет в банке)",
      "id": "ef0449e2-595d-496b-ad84-cb377cf0c1de",
      "module_id": "06d677a2-f66e-40c9-8d89-75425cddf211",
      "created_at": "2023-08-17T12:52:32.312388",
      "updated_at": "2023-08-17T12:52:32.312388"
    },
    {
      "term": "afford",
      "definition": "повзволять себе",
      "id": "7c5b87f8-77f3-41cd-95a7-06282311335c",
      "module_id": "06d677a2-f66e-40c9-8d89-75425cddf211",
      "created_at": "2023-08-17T12:52:32.312388",
      "updated_at": "2023-08-17T12:52:32.312388"
    },
    {
      "term": "force",
      "definition": "сила",
      "id": "a16319f2-c6f1-48b2-9d4b-3115d5ff5985",
      "module_id": "097ba77c-e36e-4ba8-8cd0-decb46eb7872",
      "created_at": "2023-08-17T12:52:32.312388",
      "updated_at": "2023-08-17T12:52:32.312388"
    },
    {
      "term": "power",
      "definition": "энергия",
      "id": "5927852b-50f6-4468-bf80-e827ce5a6cc8",
      "module_id": "097ba77c-e36e-4ba8-8cd0-decb46eb7872",
      "created_at": "2023-08-17T12:52:32.312388",
      "updated_at": "2023-08-17T12:52:32.312388"
    },
    {
      "term": "strength",
      "definition": "сила",
      "id": "ecce45cd-89dc-4697-99af-d97a95767042",
      "module_id": "097ba77c-e36e-4ba8-8cd0-decb46eb7872",
      "created_at": "2023-08-17T12:52:32.312388",
      "updated_at": "2023-08-17T12:52:32.312388"
    },
    {
      "term": "speed",
      "definition": "скорость",
      "id": "6655a7a3-f7c5-4870-a9db-a9f7b8b2232f",
      "module_id": "097ba77c-e36e-4ba8-8cd0-decb46eb7872",
      "created_at": "2023-08-17T12:52:32.312388",
      "updated_at": "2023-08-17T12:52:32.312388"
    },
    {
      "term": "size",
      "definition": "размер",
      "id": "0e3b15da-3f36-4780-b95e-84b11c4c6bb4",
      "module_id": "097ba77c-e36e-4ba8-8cd0-decb46eb7872",
      "created_at": "2023-08-17T12:52:32.312388",
      "updated_at": "2023-08-17T12:52:32.312388"
```

```
    },
    {
      "term": "length",
      "definition": "длина",
      "id": "3cd8fcf1-b38a-49b4-b238-c0eb0de3058d",
      "module_id": "097ba77c-e36e-4ba8-8cd0-decb46eb7872",
      "created_at": "2023-08-17T12:52:32.312388",
      "updated_at": "2023-08-17T12:52:32.312388"
    },
    {
      "term": "shape (form)",
      "definition": "форма",
      "id": "4ddf2029-8d0d-42e5-9e56-44adb7808935",
      "module_id": "097ba77c-e36e-4ba8-8cd0-decb46eb7872",
      "created_at": "2023-08-17T12:52:32.312388",
      "updated_at": "2023-08-17T12:52:32.312388"
    },
    {
      "term": "temperature",
      "definition": "температура",
      "id": "8926c189-8ccc-4d0b-bd8c-cde1043e948e",
      "module_id": "097ba77c-e36e-4ba8-8cd0-decb46eb7872",
      "created_at": "2023-08-17T12:52:32.312388",
      "updated_at": "2023-08-17T12:52:32.312388"
    }
  ],
  "errors": null,
  "info": {
    "api_version": "0.0.1",
    "deprecated": false
  }
}
```


Листинг И.21 – Получение терминов из двух модулей с лимитом и оффсетом


```
{
  "data": [
    {
      "term": "cost (cost, cost)",
      "definition": "стоить",
      "id": "0e927d12-4fda-4f8a-ad14-cfdb7fd388cd",
      "module_id": "06d677a2-f66e-40c9-8d89-75425cddf211",
      "created_at": "2023-08-17T12:52:32.312388",
      "updated_at": "2023-08-17T12:52:32.312388"
    },
    {
      "term": "earn",
      "definition": "зарабатывать",
      "id": "dca43768-3afd-47d9-8b73-3f315b45d084",
      "module_id": "06d677a2-f66e-40c9-8d89-75425cddf211",
      "created_at": "2023-08-17T12:52:32.312388",
      "updated_at": "2023-08-17T12:52:32.312388"
    },
    {
      "term": "make money (made, made)",
      "definition": "зарабатывать",
      "id": "64277a6e-127c-4d6a-9fea-13501388b530",
      "module_id": "06d677a2-f66e-40c9-8d89-75425cddf211",
      "created_at": "2023-08-17T12:52:32.312388",
      "updated_at": "2023-08-17T12:52:32.312388"
    },
    {
      "term": "promote",
```

```
  "definition": "продвигать, рекламировать, повышать по службе",
  "id": "2de2897b-f3e7-4f8c-a434-d67fc41529b8",
  "module_id": "06d677a2-f66e-40c9-8d89-75425cddf211",
  "created_at": "2023-08-17T12:52:32.312388",
  "updated_at": "2023-08-17T12:52:32.312388"
},
{
  "term": "charge",
  "definition": "брать плату",
  "id": "499b0474-1b85-4471-8a07-869bd874e2a7",
  "module_id": "06d677a2-f66e-40c9-8d89-75425cddf211",
  "created_at": "2023-08-17T12:52:32.312388",
  "updated_at": "2023-08-17T12:52:32.312388"
},
{
  "term": "estimate",
  "definition": "оценивать",
  "id": "c3fdd4b5-5e6b-443b-9062-b7ca23a2eb7f",
  "module_id": "06d677a2-f66e-40c9-8d89-75425cddf211",
  "created_at": "2023-08-17T12:52:32.312388",
  "updated_at": "2023-08-17T12:52:32.312388"
},
{
  "term": "account",
  "definition": "считать, вести учет",
  "id": "eda7249d-4094-4e27-bacb-cc09ad7fb8ff",
  "module_id": "06d677a2-f66e-40c9-8d89-75425cddf211",
  "created_at": "2023-08-17T12:52:32.312388",
  "updated_at": "2023-08-17T12:52:32.312388"
},
{
  "term": "invest",
  "definition": "инвестировать",
  "id": "a470e759-c9a4-4119-9533-6f048995c6b1",
  "module_id": "06d677a2-f66e-40c9-8d89-75425cddf211",
  "created_at": "2023-08-17T12:52:32.312388",
  "updated_at": "2023-08-17T12:52:32.312388"
},
{
  "term": "assess",
  "definition": "оценивать",
  "id": "72fb96c3-bb8a-4d58-ad4f-e080ae25e095",
  "module_id": "06d677a2-f66e-40c9-8d89-75425cddf211",
  "created_at": "2023-08-17T12:52:32.312388",
  "updated_at": "2023-08-17T12:52:32.312388"
},
{
  "term": "compete",
  "definition": "конкурировать",
  "id": "723b922f-dc8c-4cec-8be0-98e6e73fc452",
  "module_id": "06d677a2-f66e-40c9-8d89-75425cddf211",
  "created_at": "2023-08-17T12:52:32.312388",
  "updated_at": "2023-08-17T12:52:32.312388"
},
{
  "term": "purchase",
  "definition": "покупать",
  "id": "f784ae8d-b6c9-4682-8346-1e4d891d72f6",
  "module_id": "06d677a2-f66e-40c9-8d89-75425cddf211",
  "created_at": "2023-08-17T12:52:32.312388",
  "updated_at": "2023-08-17T12:52:32.312388"
},
{
  "term": "evaluate",
```

```
  "definition": "оценивать",
  "id": "c2bcd52f-caa4-4b47-a81b-0f0179cf9fa0",
  "module_id": "06d677a2-f66e-40c9-8d89-75425cddf211",
  "created_at": "2023-08-17T12:52:32.312388",
  "updated_at": "2023-08-17T12:52:32.312388"
},
{
  "term": "trade",
  "definition": "торговать",
  "id": "044676da-a33a-42ae-9d98-6ce853cf7846",
  "module_id": "06d677a2-f66e-40c9-8d89-75425cddf211",
  "created_at": "2023-08-17T12:52:32.312388",
  "updated_at": "2023-08-17T12:52:32.312388"
},
{
  "term": "market",
  "definition": "продвигать на рынке (рекламировать)",
  "id": "95f3ac6a-dd35-49d2-8826-8009a44e360d",
  "module_id": "06d677a2-f66e-40c9-8d89-75425cddf211",
  "created_at": "2023-08-17T12:52:32.312388",
  "updated_at": "2023-08-17T12:52:32.312388"
},
{
  "term": "withdraw",
  "definition": "снимать (деньги со счета в банке)",
  "id": "28ad02e5-647d-4dc7-b263-7dd7c1735e65",
  "module_id": "06d677a2-f66e-40c9-8d89-75425cddf211",
  "created_at": "2023-08-17T12:52:32.312388",
  "updated_at": "2023-08-17T12:52:32.312388"
},
{
  "term": "convert",
  "definition": "конвертировать",
  "id": "e0e298a5-4ad6-4c6d-bc56-dab9413b6daa",
  "module_id": "06d677a2-f66e-40c9-8d89-75425cddf211",
  "created_at": "2023-08-17T12:52:32.312388",
  "updated_at": "2023-08-17T12:52:32.312388"
},
{
  "term": "spend (spent, spent)",
  "definition": "тратить (напр. деньги), проводить время",
  "id": "fdf9bf83-51d3-4df8-9ee2-10ac33a1e4bd",
  "module_id": "06d677a2-f66e-40c9-8d89-75425cddf211",
  "created_at": "2023-08-17T12:52:32.312388",
  "updated_at": "2023-08-17T12:52:32.312388"
},
{
  "term": "buy (bought, bought)",
  "definition": "покупать",
  "id": "78bf43c6-e5b4-43ea-87e1-84fae74365eb",
  "module_id": "06d677a2-f66e-40c9-8d89-75425cddf211",
  "created_at": "2023-08-17T12:52:32.312388",
  "updated_at": "2023-08-17T12:52:32.312388"
},
{
  "term": "sell (sold, sold)",
  "definition": "продавать",
  "id": "723f2f7e-bb02-4263-bcae-3b6d40e14ec4",
  "module_id": "06d677a2-f66e-40c9-8d89-75425cddf211",
  "created_at": "2023-08-17T12:52:32.312388",
  "updated_at": "2023-08-17T12:52:32.312388"
},
{
  "term": "deal (dealt, dealt)",
```

```
  "definition": "вести дела",
  "id": "a23e657b-2880-4990-be2f-0c22d6bef186",
  "module_id": "06d677a2-f66e-40c9-8d89-75425cddf211",
  "created_at": "2023-08-17T12:52:32.312388",
  "updated_at": "2023-08-17T12:52:32.312388"
},
{
  "term": "produce",
  "definition": "производить",
  "id": "7388a982-85df-42f2-9363-b1d0f6549738",
  "module_id": "06d677a2-f66e-40c9-8d89-75425cddf211",
  "created_at": "2023-08-17T12:52:32.312388",
  "updated_at": "2023-08-17T12:52:32.312388"
},
{
  "term": "save",
  "definition": "делать сбережения, экономить",
  "id": "349589f8-8a48-4358-929d-02ce0f9b0435",
  "module_id": "06d677a2-f66e-40c9-8d89-75425cddf211",
  "created_at": "2023-08-17T12:52:32.312388",
  "updated_at": "2023-08-17T12:52:32.312388"
},
{
  "term": "rule",
  "definition": "править (властвовать)",
  "id": "175a3b99-3f9c-4bf5-8dcf-4c45f41f81ba",
  "module_id": "06d677a2-f66e-40c9-8d89-75425cddf211",
  "created_at": "2023-08-17T12:52:32.312388",
  "updated_at": "2023-08-17T12:52:32.312388"
},
{
  "term": "elect",
  "definition": "выбирать (на выборах)",
  "id": "fecb963a-17fa-4740-a954-db43ab2fc938",
  "module_id": "06d677a2-f66e-40c9-8d89-75425cddf211",
  "created_at": "2023-08-17T12:52:32.312388",
  "updated_at": "2023-08-17T12:52:32.312388"
},
{
  "term": "vote",
  "definition": "голосовать",
  "id": "ac0aac4d-d725-411a-9622-966a6d701c58",
  "module_id": "06d677a2-f66e-40c9-8d89-75425cddf211",
  "created_at": "2023-08-17T12:52:32.312388",
  "updated_at": "2023-08-17T12:52:32.312388"
},
{
  "term": "contribute",
  "definition": "вносить вклад во что-то",
  "id": "9f65debb-8d8a-48e0-8f1d-eabad602e89e",
  "module_id": "06d677a2-f66e-40c9-8d89-75425cddf211",
  "created_at": "2023-08-17T12:52:32.312388",
  "updated_at": "2023-08-17T12:52:32.312388"
},
{
  "term": "settle",
  "definition": "расселяться (создавать поселение)",
  "id": "748e3ee3-9f88-458f-a28e-8d2cf4079a74",
  "module_id": "06d677a2-f66e-40c9-8d89-75425cddf211",
  "created_at": "2023-08-17T12:52:32.312388",
  "updated_at": "2023-08-17T12:52:32.312388"
},
{
  "term": "impose",
```

```
  "definition": "налагать (напр. обязательство)",
  "id": "5a19be98-4a18-4bcc-9f75-62f61511538a",
  "module_id": "06d677a2-f66e-40c9-8d89-75425cddf211",
  "created_at": "2023-08-17T12:52:32.312388",
  "updated_at": "2023-08-17T12:52:32.312388"
},
{
  "term": "transfer",
  "definition": "передавать, переводить (напр. деньги)",
  "id": "d5bbcb02-e75c-4d08-b0d1-6a9e72a8cc99",
  "module_id": "06d677a2-f66e-40c9-8d89-75425cddf211",
  "created_at": "2023-08-17T12:52:32.312388",
  "updated_at": "2023-08-17T12:52:32.312388"
},
{
  "term": "distribute",
  "definition": "распространять",
  "id": "4ab64ff0-b7b1-4b66-9667-92ec5329eb8e",
  "module_id": "06d677a2-f66e-40c9-8d89-75425cddf211",
  "created_at": "2023-08-17T12:52:32.312388",
  "updated_at": "2023-08-17T12:52:32.312388"
},
{
  "term": "deposit",
  "definition": "вносить деньги (на счет в банке)",
  "id": "ef0449e2-595d-496b-ad84-cb377cf0c1de",
  "module_id": "06d677a2-f66e-40c9-8d89-75425cddf211",
  "created_at": "2023-08-17T12:52:32.312388",
  "updated_at": "2023-08-17T12:52:32.312388"
},
{
  "term": "afford",
  "definition": "повзволять себе",
  "id": "7c5b87f8-77f3-41cd-95a7-06282311335c",
  "module_id": "06d677a2-f66e-40c9-8d89-75425cddf211",
  "created_at": "2023-08-17T12:52:32.312388",
  "updated_at": "2023-08-17T12:52:32.312388"
},
{
  "term": "force",
  "definition": "сила",
  "id": "a16319f2-c6f1-48b2-9d4b-3115d5ff5985",
  "module_id": "097ba77c-e36e-4ba8-8cd0-decb46eb7872",
  "created_at": "2023-08-17T12:52:32.312388",
  "updated_at": "2023-08-17T12:52:32.312388"
},
{
  "term": "power",
  "definition": "энергия",
  "id": "5927852b-50f6-4468-bf80-e827ce5a6cc8",
  "module_id": "097ba77c-e36e-4ba8-8cd0-decb46eb7872",
  "created_at": "2023-08-17T12:52:32.312388",
  "updated_at": "2023-08-17T12:52:32.312388"
},
{
  "term": "strength",
  "definition": "сила",
  "id": "ecce45cd-89dc-4697-99af-d97a95767042",
  "module_id": "097ba77c-e36e-4ba8-8cd0-decb46eb7872",
  "created_at": "2023-08-17T12:52:32.312388",
  "updated_at": "2023-08-17T12:52:32.312388"
},
{
  "term": "speed",
```

```
  "definition": "скорость",
  "id": "6655a7a3-f7c5-4870-a9db-a9f7b8b2232f",
  "module_id": "097ba77c-e36e-4ba8-8cd0-decb46eb7872",
  "created_at": "2023-08-17T12:52:32.312388",
  "updated_at": "2023-08-17T12:52:32.312388"
},
{
  "term": "size",
  "definition": "размер",
  "id": "0e3b15da-3f36-4780-b95e-84b11c4c6bb4",
  "module_id": "097ba77c-e36e-4ba8-8cd0-decb46eb7872",
  "created_at": "2023-08-17T12:52:32.312388",
  "updated_at": "2023-08-17T12:52:32.312388"
},
{
  "term": "length",
  "definition": "длина",
  "id": "3cd8fcf1-b38a-49b4-b238-c0eb0de3058d",
  "module_id": "097ba77c-e36e-4ba8-8cd0-decb46eb7872",
  "created_at": "2023-08-17T12:52:32.312388",
  "updated_at": "2023-08-17T12:52:32.312388"
},
{
  "term": "shape (form)",
  "definition": "форма",
  "id": "4ddf2029-8d0d-42e5-9e56-44adb7808935",
  "module_id": "097ba77c-e36e-4ba8-8cd0-decb46eb7872",
  "created_at": "2023-08-17T12:52:32.312388",
  "updated_at": "2023-08-17T12:52:32.312388"
},
{
  "term": "temperature",
  "definition": "температура",
  "id": "8926c189-8ccc-4d0b-bd8c-cde1043e948e",
  "module_id": "097ba77c-e36e-4ba8-8cd0-decb46eb7872",
  "created_at": "2023-08-17T12:52:32.312388",
  "updated_at": "2023-08-17T12:52:32.312388"
},
{
  "term": "capacity",
  "definition": "вместимость",
  "id": "7e40f1b9-c1a1-4b62-83ec-8ad23602aa03",
  "module_id": "097ba77c-e36e-4ba8-8cd0-decb46eb7872",
  "created_at": "2023-08-17T12:52:32.312388",
  "updated_at": "2023-08-17T12:52:32.312388"
},
{
  "term": "volume",
  "definition": "объем",
  "id": "a681963f-7c5a-406c-b976-8f8e76661501",
  "module_id": "097ba77c-e36e-4ba8-8cd0-decb46eb7872",
  "created_at": "2023-08-17T12:52:32.312388",
  "updated_at": "2023-08-17T12:52:32.312388"
},
{
  "term": "mass",
  "definition": "масса",
  "id": "91da349d-ad9b-43ea-b332-9356e082abaf",
  "module_id": "097ba77c-e36e-4ba8-8cd0-decb46eb7872",
  "created_at": "2023-08-17T12:52:32.312388",
  "updated_at": "2023-08-17T12:52:32.312388"
},
{
  "term": "weight",
```

```
  "definition": "вес",
  "id": "49b51c70-dcc4-49ac-97cc-5cd31e0203f4",
  "module_id": "097ba77c-e36e-4ba8-8cd0-decb46eb7872",
  "created_at": "2023-08-17T12:52:32.312388",
  "updated_at": "2023-08-17T12:52:32.312388"
},
{
  "term": "smell",
  "definition": "запах",
  "id": "8fa89d23-3140-417d-80ed-934b4ed3caf2",
  "module_id": "097ba77c-e36e-4ba8-8cd0-decb46eb7872",
  "created_at": "2023-08-17T12:52:32.312388",
  "updated_at": "2023-08-17T12:52:32.312388"
},
{
  "term": "pace",
  "definition": "темп",
  "id": "a40924d6-05b8-45d7-a6f7-e5e11b76ad15",
  "module_id": "097ba77c-e36e-4ba8-8cd0-decb46eb7872",
  "created_at": "2023-08-17T12:52:32.312388",
  "updated_at": "2023-08-17T12:52:32.312388"
},
{
  "term": "contrast",
  "definition": "контраст",
  "id": "5f4a54be-dd95-4a17-ab3e-319efe9f4eca",
  "module_id": "097ba77c-e36e-4ba8-8cd0-decb46eb7872",
  "created_at": "2023-08-17T12:52:32.312388",
  "updated_at": "2023-08-17T12:52:32.312388"
},
{
  "term": "absence",
  "definition": "отсутствие",
  "id": "200e7de3-90c7-40ae-9551-22190521f388",
  "module_id": "097ba77c-e36e-4ba8-8cd0-decb46eb7872",
  "created_at": "2023-08-17T12:52:32.312388",
  "updated_at": "2023-08-17T12:52:32.312388"
},
{
  "term": "presence",
  "definition": "присутствие",
  "id": "5f41012b-80d9-4549-a469-c262e0c9acc9",
  "module_id": "097ba77c-e36e-4ba8-8cd0-decb46eb7872",
  "created_at": "2023-08-17T12:52:32.312388",
  "updated_at": "2023-08-17T12:52:32.312388"
},
{
  "term": "state",
  "definition": "состояние (напр. вещества)",
  "id": "416042d8-8a2b-4821-8295-472ea07a6988",
  "module_id": "097ba77c-e36e-4ba8-8cd0-decb46eb7872",
  "created_at": "2023-08-17T12:52:32.312388",
  "updated_at": "2023-08-17T12:52:32.312388"
},
{
  "term": "noise",
  "definition": "шум",
  "id": "053383d3-86d7-411f-b115-f7c5b3c45044",
  "module_id": "097ba77c-e36e-4ba8-8cd0-decb46eb7872",
  "created_at": "2023-08-17T12:52:32.312388",
  "updated_at": "2023-08-17T12:52:32.312388"
},
{
  "term": "resistance",
```

```
  "definition": "сопротивление",
  "id": "3bcb5d1d-88b2-4dc5-803a-1aa6b4b833b5",
  "module_id": "097ba77c-e36e-4ba8-8cd0-decb46eb7872",
  "created_at": "2023-08-17T12:52:32.312388",
  "updated_at": "2023-08-17T12:52:32.312388"
},
{
  "term": "tension",
  "definition": "напряжение, натяжение",
  "id": "2231d93d-e6b2-43f3-8849-ff905195bb14",
  "module_id": "097ba77c-e36e-4ba8-8cd0-decb46eb7872",
  "created_at": "2023-08-17T12:52:32.312388",
  "updated_at": "2023-08-17T12:52:32.312388"
},
{
  "term": "kilogram",
  "definition": "килограмм",
  "id": "4958d43a-2c47-4b7b-a45a-8b0ffe0b84c6",
  "module_id": "097ba77c-e36e-4ba8-8cd0-decb46eb7872",
  "created_at": "2023-08-17T12:52:32.312388",
  "updated_at": "2023-08-17T12:52:32.312388"
},
{
  "term": "meter",
  "definition": "метр",
  "id": "cd281ee5-6bc3-4696-bd0e-91b63fa30afd",
  "module_id": "097ba77c-e36e-4ba8-8cd0-decb46eb7872",
  "created_at": "2023-08-17T12:52:32.312388",
  "updated_at": "2023-08-17T12:52:32.312388"
},
{
  "term": "inch",
  "definition": "дюйм",
  "id": "8176405b-84bf-484f-addb-87e1351ca63b",
  "module_id": "097ba77c-e36e-4ba8-8cd0-decb46eb7872",
  "created_at": "2023-08-17T12:52:32.312388",
  "updated_at": "2023-08-17T12:52:32.312388"
},
{
  "term": "mile",
  "definition": "миля",
  "id": "dbe85ca9-364d-4dba-8a0d-1bfe78905379",
  "module_id": "097ba77c-e36e-4ba8-8cd0-decb46eb7872",
  "created_at": "2023-08-17T12:52:32.312388",
  "updated_at": "2023-08-17T12:52:32.312388"
},
{
  "term": "pound",
  "definition": "фунт",
  "id": "3ace2eb8-6e3e-4ff8-9df9-c3866a606727",
  "module_id": "097ba77c-e36e-4ba8-8cd0-decb46eb7872",
  "created_at": "2023-08-17T12:52:32.312388",
  "updated_at": "2023-08-17T12:52:32.312388"
},
{
  "term": "foot",
  "definition": "фут",
  "id": "88ea30ce-8f58-44fd-8404-f5e1f98fc7ad",
  "module_id": "097ba77c-e36e-4ba8-8cd0-decb46eb7872",
  "created_at": "2023-08-17T12:52:32.312388",
  "updated_at": "2023-08-17T12:52:32.312388"
},
{
  "term": "absence",
```

```
      "definition": "отсутствие",
      "id": "a20c7c59-1eb4-4cce-b165-8030ec0537e2",
      "module_id": "097ba77c-e36e-4ba8-8cd0-decb46eb7872",
      "created_at": "2023-08-17T12:52:32.312388",
      "updated_at": "2023-08-17T12:52:32.312388"
    },
    {
      "term": "state",
      "definition": "состояние (напр. вещества)",
      "id": "cab059ba-5c77-4529-a7e9-854b650e15b9",
      "module_id": "097ba77c-e36e-4ba8-8cd0-decb46eb7872",
      "created_at": "2023-08-17T12:52:32.312388",
      "updated_at": "2023-08-17T12:52:32.312388"
    },
    {
      "term": "colour (br.), color (am.)",
      "definition": "цвет",
      "id": "6cecfb17-f124-470c-83f6-447829cc0e28",
      "module_id": "097ba77c-e36e-4ba8-8cd0-decb46eb7872",
      "created_at": "2023-08-17T12:52:32.312388",
      "updated_at": "2023-08-17T12:52:32.312388"
    }
  ],
  "errors": null,
  "info": {
    "api_version": "0.0.1",
    "deprecated": false
  }
}
```


Листинг И.22 – Получение из модулей, один из которых не существует


```
{
  "data": [
    {
      "term": "deposit",
      "definition": "вносить деньги (на счет в банке)",
      "id": "ef0449e2-595d-496b-ad84-cb377cf0c1de",
      "module_id": "06d677a2-f66e-40c9-8d89-75425cddf211",
      "created_at": "2023-08-17T12:52:32.312388",
      "updated_at": "2023-08-17T12:52:32.312388"
    },
    {
      "term": "afford",
      "definition": "повзволять себе",
      "id": "7c5b87f8-77f3-41cd-95a7-06282311335c",
      "module_id": "06d677a2-f66e-40c9-8d89-75425cddf211",
      "created_at": "2023-08-17T12:52:32.312388",
      "updated_at": "2023-08-17T12:52:32.312388"
    }
  ],
  "errors": null,
  "info": {
    "api_version": "0.0.1",
    "deprecated": false
  }
}
```


Листинг И.23 – Получение терминов из двух несуществующих модулей


```
{
  "data": [],
  "errors": null,
```

```
  "info": {
    "api_version": "0.0.1",
    "deprecated": false
  }
}
```


Листинг И.24 – Получение терминов из добавленного пользователем модуля


```
{
  "data": [
    {
      "term": "with (coffe with sugar)",
      "definition": "с (кофе с сахаром)",
      "id": "22f97dbb-fa86-4873-a819-34791c8379f0",
      "module_id": "f6c16a78-adf8-4dba-a12d-ac3d08d8ff05",
      "created_at": "2023-08-17T12:52:32.312388",
      "updated_at": "2023-08-17T12:52:32.312388"
    },
    {
      "term": "about (tell me about yourself)",
      "definition": "о (расскажите мне о себе)",
      "id": "90cb28cc-ad25-4e1c-b007-c8e68b3e5ff1",
      "module_id": "f6c16a78-adf8-4dba-a12d-ac3d08d8ff05",
      "created_at": "2023-08-17T12:52:32.312388",
      "updated_at": "2023-08-17T12:52:32.312388"
    },
    {
      "term": "against (against war)",
      "definition": "против (против войны)",
      "id": "26633fac-d9f9-402e-9aab-54aa29345828",
      "module_id": "f6c16a78-adf8-4dba-a12d-ac3d08d8ff05",
      "created_at": "2023-08-17T12:52:32.312388",
      "updated_at": "2023-08-17T12:52:32.312388"
    },
    {
      "term": "without (without you)",
      "definition": "без (без тебя)",
      "id": "7d9ea70f-d107-4e56-84c9-9f47b72c353c",
      "module_id": "f6c16a78-adf8-4dba-a12d-ac3d08d8ff05",
      "created_at": "2023-08-17T12:52:32.312388",
      "updated_at": "2023-08-17T12:52:32.312388"
    },
    {
      "term": "including (two cars including one broken)",
      "definition": "включая (две машины, включая одну сломанную)",
      "id": "80a9b430-537b-4260-8b12-f32d75ac4928",
      "module_id": "f6c16a78-adf8-4dba-a12d-ac3d08d8ff05",
      "created_at": "2023-08-17T12:52:32.312388",
      "updated_at": "2023-08-17T12:52:32.312388"
    },
    {
      "term": "according to (according to the rules)",
      "definition": "в соответствии с (в соответствии с правилами)",
      "id": "c07741f5-0ef0-40d6-be43-4dfafa8de902",
      "module_id": "f6c16a78-adf8-4dba-a12d-ac3d08d8ff05",
      "created_at": "2023-08-17T12:52:32.312388",
      "updated_at": "2023-08-17T12:52:32.312388"
    },
    {
      "term": "because of (because of the car accident)",
      "definition": "из-за (из-за автомобильной аварии)",
      "id": "30c65c92-0321-436c-81c1-dde3dc49337e",
      "module_id": "f6c16a78-adf8-4dba-a12d-ac3d08d8ff05",
```

```
      "created_at": "2023-08-17T12:52:32.312388",
      "updated_at": "2023-08-17T12:52:32.312388"
    },
    {
      "term": "despite (despite the bad weather)",
      "definition": "несмотря на (несмотря на плохую погоду)",
      "id": "b4bda8aa-0bc2-4ee4-be36-b79ce534e41c",
      "module_id": "f6c16a78-adf8-4dba-a12d-ac3d08d8ff05",
      "created_at": "2023-08-17T12:52:32.312388",
      "updated_at": "2023-08-17T12:52:32.312388"
    },
    {
      "term": "instead of (new part instead of broken one)",
      "definition": "вместо (новая запчасть вместо сломанной)",
      "id": "ba260916-b8b8-49ed-9dcf-b9d4ae7c7930",
      "module_id": "f6c16a78-adf8-4dba-a12d-ac3d08d8ff05",
      "created_at": "2023-08-17T12:52:32.312388",
      "updated_at": "2023-08-17T12:52:32.312388"
    },
    {
      "term": "in spite of (in spite of the rain)",
      "definition": "несмотря на (несмотря на дождь)",
      "id": "620d1b79-dddc-47f9-a978-999beff401dd",
      "module_id": "f6c16a78-adf8-4dba-a12d-ac3d08d8ff05",
      "created_at": "2023-08-17T12:52:32.312388",
      "updated_at": "2023-08-17T12:52:32.312388"
    },
    {
      "term": "regardless of (regardless of what you say)",
      "definition": "несмотря на, независимо от (несмотря на то, что ты
скажешь)",
      "id": "dc767185-5de0-423f-9980-6ad1b4a97091",
      "module_id": "f6c16a78-adf8-4dba-a12d-ac3d08d8ff05",
      "created_at": "2023-08-17T12:52:32.312388",
      "updated_at": "2023-08-17T12:52:32.312388"
    },
    {
      "term": "by means of (by means of tools)",
      "definition": "с помощью (с помощью инструментов)",
      "id": "4ef259cf-720e-4d59-a3ae-30a248d9da60",
      "module_id": "f6c16a78-adf8-4dba-a12d-ac3d08d8ff05",
      "created_at": "2023-08-17T12:52:32.312388",
      "updated_at": "2023-08-17T12:52:32.312388"
    },
    {
      "term": "on behalf of (on behalf of my boss)",
      "definition": "от имени (от имени моего босса)",
      "id": "65ff1700-65dd-4471-b81c-5ae25acd6b2b",
      "module_id": "f6c16a78-adf8-4dba-a12d-ac3d08d8ff05",
      "created_at": "2023-08-17T12:52:32.312388",
      "updated_at": "2023-08-17T12:52:32.312388"
    },
    {
      "term": "for (this gift is for you)",
      "definition": "для (этот подарок для тебя)",
      "id": "ca8afa1a-1fc3-4200-8d42-e96235d8cbaf",
      "module_id": "f6c16a78-adf8-4dba-a12d-ac3d08d8ff05",
      "created_at": "2023-08-17T12:52:32.312388",
      "updated_at": "2023-08-17T12:52:32.312388"
    },
    {
      "term": "compared to (this is nothing compared to the real troubles)",
      "definition": "в сравнении с (это ничто в сравнении с настоящими
трудностями)",
```

```
      "id": "0fe8db4a-ba54-4e4e-b99c-8a78bcb0c739",
      "module_id": "f6c16a78-adf8-4dba-a12d-ac3d08d8ff05",
      "created_at": "2023-08-17T12:52:32.312388",
      "updated_at": "2023-08-17T12:52:32.312388"
    },
    {
      "term": "thanks to (thanks to the doctor, i`m feeling fine)",
      "definition": "благодаря (благодаря доктору, я чувствую себя отлично)",
      "id": "f69d4341-4911-42db-8566-01dc33e647b4",
      "module_id": "f6c16a78-adf8-4dba-a12d-ac3d08d8ff05",
      "created_at": "2023-08-17T12:52:32.312388",
      "updated_at": "2023-08-17T12:52:32.312388"
    },
    {
      "term": "rather… than (they`d rather buy fish than meat)",
      "definition": "скорее, чем (они купят скорее рыбу, чем мясо)",
      "id": "33900a09-f883-4f04-b8de-7fb7d0738d41",
      "module_id": "f6c16a78-adf8-4dba-a12d-ac3d08d8ff05",
      "created_at": "2023-08-17T12:52:32.312388",
      "updated_at": "2023-08-17T12:52:32.312388"
    },
    {
      "term": "rather than (they`d buy fish rather than meat)",
      "definition": "скорее, чем\\a не (они купят рыбу, а не мясо)",
      "id": "e788be2d-aa4a-4371-ad13-63bcbea65ddb",
      "module_id": "f6c16a78-adf8-4dba-a12d-ac3d08d8ff05",
      "created_at": "2023-08-17T12:52:32.312388",
      "updated_at": "2023-08-17T12:52:32.312388"
    }
  ],
  "errors": null,
  "info": {
    "api_version": "0.0.1",
    "deprecated": false
  }
}
```


Листинг И.25 – Получение из модуля, который пользователь не добавлял


```
{
  "data": [
    {
      "term": "top",
      "definition": "верхний (главный)",
      "id": "bb45c9af-5d82-447c-a975-78c61dd9e96e",
      "module_id": "68384971-8ead-4866-9a57-7eaaff64865e",
      "created_at": "2023-08-17T12:52:32.312388",
      "updated_at": "2023-08-17T12:52:32.312388"
    },
    {
      "term": "right",
      "definition": "правильный (правый)",
      "id": "04bd6740-4f5a-46c8-b222-9272082730f9",
      "module_id": "68384971-8ead-4866-9a57-7eaaff64865e",
      "created_at": "2023-08-17T12:52:32.312388",
      "updated_at": "2023-08-17T12:52:32.312388"
    },
    {
      "term": "local",
      "definition": "местный",
      "id": "548ed1f9-0d2f-46fc-8dab-fe6656a771b2",
      "module_id": "68384971-8ead-4866-9a57-7eaaff64865e",
      "created_at": "2023-08-17T12:52:32.312388",
```

```
  "updated_at": "2023-08-17T12:52:32.312388"
},
{
  "term": "available",
  "definition": "доступный",
  "id": "065594c6-cd8e-435f-83fb-164726e37141",
  "module_id": "68384971-8ead-4866-9a57-7eaaff64865e",
  "created_at": "2023-08-17T12:52:32.312388",
  "updated_at": "2023-08-17T12:52:32.312388"
},
{
  "term": "central",
  "definition": "центральный",
  "id": "560d1aca-8c07-4b64-bc3b-7d7ff008c162",
  "module_id": "68384971-8ead-4866-9a57-7eaaff64865e",
  "created_at": "2023-08-17T12:52:32.312388",
  "updated_at": "2023-08-17T12:52:32.312388"
},
{
  "term": "left",
  "definition": "левый (оставшийся)",
  "id": "e7a236c6-599e-4132-a5ef-a723d5622f67",
  "module_id": "68384971-8ead-4866-9a57-7eaaff64865e",
  "created_at": "2023-08-17T12:52:32.312388",
  "updated_at": "2023-08-17T12:52:32.312388"
},
{
  "term": "close",
  "definition": "близкий (закрытый)",
  "id": "78cba102-b4ac-4daf-b276-06a18621a28f",
  "module_id": "68384971-8ead-4866-9a57-7eaaff64865e",
  "created_at": "2023-08-17T12:52:32.312388",
  "updated_at": "2023-08-17T12:52:32.312388"
},
{
  "term": "far",
  "definition": "далекий",
  "id": "4ca4ebd6-bdd0-4225-a53a-223b3e0cf1ab",
  "module_id": "68384971-8ead-4866-9a57-7eaaff64865e",
  "created_at": "2023-08-17T12:52:32.312388",
  "updated_at": "2023-08-17T12:52:32.312388"
},
{
  "term": "deep",
  "definition": "глубокий",
  "id": "6487bca7-8c2f-4e8a-889d-6c9a0b6f26d9",
  "module_id": "68384971-8ead-4866-9a57-7eaaff64865e",
  "created_at": "2023-08-17T12:52:32.312388",
  "updated_at": "2023-08-17T12:52:32.312388"
},
{
  "term": "middle",
  "definition": "средний",
  "id": "915f46ca-2e9b-4527-a374-6e6d48a6d4e4",
  "module_id": "68384971-8ead-4866-9a57-7eaaff64865e",
  "created_at": "2023-08-17T12:52:32.312388",
  "updated_at": "2023-08-17T12:52:32.312388"
},
{
  "term": "western",
  "definition": "западный",
  "id": "06d6b118-f1fb-4eae-b401-ea5f6d0e066a",
  "module_id": "68384971-8ead-4866-9a57-7eaaff64865e",
  "created_at": "2023-08-17T12:52:32.312388",
```

```
  "updated_at": "2023-08-17T12:52:32.312388"
},
{
  "term": "southern",
  "definition": "южный",
  "id": "8479b1a8-1ec0-4463-a238-7fe094eafe51",
  "module_id": "68384971-8ead-4866-9a57-7eaaff64865e",
  "created_at": "2023-08-17T12:52:32.312388",
  "updated_at": "2023-08-17T12:52:32.312388"
},
{
  "term": "front",
  "definition": "передний",
  "id": "cd61e870-60d5-4ad4-9d58-92588604fde7",
  "module_id": "68384971-8ead-4866-9a57-7eaaff64865e",
  "created_at": "2023-08-17T12:52:32.312388",
  "updated_at": "2023-08-17T12:52:32.312388"
},
{
  "term": "eastern",
  "definition": "восточный",
  "id": "84450e67-cda8-4178-b1f6-a4496df36c5d",
  "module_id": "68384971-8ead-4866-9a57-7eaaff64865e",
  "created_at": "2023-08-17T12:52:32.312388",
  "updated_at": "2023-08-17T12:52:32.312388"
},
{
  "term": "northern",
  "definition": "северный",
  "id": "e4a19bd7-9acd-400f-a58a-7488913686d6",
  "module_id": "68384971-8ead-4866-9a57-7eaaff64865e",
  "created_at": "2023-08-17T12:52:32.312388",
  "updated_at": "2023-08-17T12:52:32.312388"
},
{
  "term": "internal",
  "definition": "внутренний",
  "id": "26cbbbc4-c2ef-4090-8e63-98d2be53e395",
  "module_id": "68384971-8ead-4866-9a57-7eaaff64865e",
  "created_at": "2023-08-17T12:52:32.312388",
  "updated_at": "2023-08-17T12:52:32.312388"
},
{
  "term": "upper",
  "definition": "верхний",
  "id": "a4a73d75-808f-45e8-b8e3-a7873e6c0fb3",
  "module_id": "68384971-8ead-4866-9a57-7eaaff64865e",
  "created_at": "2023-08-17T12:52:32.312388",
  "updated_at": "2023-08-17T12:52:32.312388"
},
{
  "term": "vast",
  "definition": "широкий (обширный)",
  "id": "5dd49d3f-9839-4a00-8502-6c5720a55e48",
  "module_id": "68384971-8ead-4866-9a57-7eaaff64865e",
  "created_at": "2023-08-17T12:52:32.312388",
  "updated_at": "2023-08-17T12:52:32.312388"
},
{
  "term": "limited",
  "definition": "ограниченный",
  "id": "0515736a-7209-454e-9142-bf005ac9a75e",
  "module_id": "68384971-8ead-4866-9a57-7eaaff64865e",
  "created_at": "2023-08-17T12:52:32.312388",
```

```
  "updated_at": "2023-08-17T12:52:32.312388"
},
{
  "term": "outside",
  "definition": "внешний",
  "id": "49e5122a-c9e1-4476-913e-02da33076282",
  "module_id": "68384971-8ead-4866-9a57-7eaaff64865e",
  "created_at": "2023-08-17T12:52:32.312388",
  "updated_at": "2023-08-17T12:52:32.312388"
},
{
  "term": "nearby",
  "definition": "находящийся поблизости",
  "id": "1a105d0b-c8db-487e-bfe7-f0d3c2eceaad",
  "module_id": "68384971-8ead-4866-9a57-7eaaff64865e",
  "created_at": "2023-08-17T12:52:32.312388",
  "updated_at": "2023-08-17T12:52:32.312388"
},
{
  "term": "inner",
  "definition": "внутренний",
  "id": "3e7140d0-ac33-4f9e-9849-48b2f626c9a2",
  "module_id": "68384971-8ead-4866-9a57-7eaaff64865e",
  "created_at": "2023-08-17T12:52:32.312388",
  "updated_at": "2023-08-17T12:52:32.312388"
},
{
  "term": "biological",
  "definition": "биологический",
  "id": "dd06ca9d-736d-412d-ae38-ec686dbf0805",
  "module_id": "68384971-8ead-4866-9a57-7eaaff64865e",
  "created_at": "2023-08-17T12:52:32.312388",
  "updated_at": "2023-08-17T12:52:32.312388"
},
{
  "term": "distant",
  "definition": "удаленный",
  "id": "a907c943-759b-4d0b-9eea-e155783c04e9",
  "module_id": "68384971-8ead-4866-9a57-7eaaff64865e",
  "created_at": "2023-08-17T12:52:32.312388",
  "updated_at": "2023-08-17T12:52:32.312388"
},
{
  "term": "bottom",
  "definition": "нижний",
  "id": "21ccc33c-9e31-4207-8895-b4e0c13dbe9e",
  "module_id": "68384971-8ead-4866-9a57-7eaaff64865e",
  "created_at": "2023-08-17T12:52:32.312388",
  "updated_at": "2023-08-17T12:52:32.312388"
},
{
  "term": "remote",
  "definition": "удаленный (дистанционный)",
  "id": "1df90a87-feb1-4aba-9374-fe432880fa58",
  "module_id": "68384971-8ead-4866-9a57-7eaaff64865e",
  "created_at": "2023-08-17T12:52:32.312388",
  "updated_at": "2023-08-17T12:52:32.312388"
},
{
  "term": "external",
  "definition": "внешний",
  "id": "0924aa88-6217-47ca-a735-3d285b05214f",
  "module_id": "68384971-8ead-4866-9a57-7eaaff64865e",
  "created_at": "2023-08-17T12:52:32.312388",
```

```
  "updated_at": "2023-08-17T12:52:32.312388"
},
{
  "term": "lost",
  "definition": "потерянный",
  "id": "9353ca6a-2566-45ed-9649-649dd8c1b8ba",
  "module_id": "68384971-8ead-4866-9a57-7eaaff64865e",
  "created_at": "2023-08-17T12:52:32.312388",
  "updated_at": "2023-08-17T12:52:32.312388"
},
{
  "term": "extensive",
  "definition": "обширный",
  "id": "0a680205-b7df-4ea8-bd14-eb59fd397a01",
  "module_id": "68384971-8ead-4866-9a57-7eaaff64865e",
  "created_at": "2023-08-17T12:52:32.312388",
  "updated_at": "2023-08-17T12:52:32.312388"
},
{
  "term": "natural",
  "definition": "природный (естественный)",
  "id": "d00d2d2e-511e-496c-b254-b6018a604a7f",
  "module_id": "68384971-8ead-4866-9a57-7eaaff64865e",
  "created_at": "2023-08-17T12:52:32.312388",
  "updated_at": "2023-08-17T12:52:32.312388"
},
{
  "term": "environmental",
  "definition": "относящийся к окружающей среде",
  "id": "9b947caa-90b7-43bd-a824-90eb62d95f8f",
  "module_id": "68384971-8ead-4866-9a57-7eaaff64865e",
  "created_at": "2023-08-17T12:52:32.312388",
  "updated_at": "2023-08-17T12:52:32.312388"
},
{
  "term": "fresh",
  "definition": "свежий",
  "id": "0489281a-e86d-4919-8fa6-f8506a42db65",
  "module_id": "68384971-8ead-4866-9a57-7eaaff64865e",
  "created_at": "2023-08-17T12:52:32.312388",
  "updated_at": "2023-08-17T12:52:32.312388"
},
{
  "term": "global",
  "definition": "глобальный",
  "id": "f2e19e5a-7d68-41cf-b8aa-dc938e59a03f",
  "module_id": "68384971-8ead-4866-9a57-7eaaff64865e",
  "created_at": "2023-08-17T12:52:32.312388",
  "updated_at": "2023-08-17T12:52:32.312388"
},
{
  "term": "wild",
  "definition": "дикий",
  "id": "32f0fb87-c140-4a09-b626-6a6ed7a23eed",
  "module_id": "68384971-8ead-4866-9a57-7eaaff64865e",
  "created_at": "2023-08-17T12:52:32.312388",
  "updated_at": "2023-08-17T12:52:32.312388"
},
{
  "term": "growing",
  "definition": "растущий",
  "id": "9df5cc2b-9a8a-4e92-9965-b2a5829eaee0",
  "module_id": "68384971-8ead-4866-9a57-7eaaff64865e",
  "created_at": "2023-08-17T12:52:32.312388",
```

```
      "updated_at": "2023-08-17T12:52:32.312388"
    },
    {
      "term": "living",
      "definition": "живущий",
      "id": "3cee89ae-3b88-485a-b550-edee8af46b69",
      "module_id": "68384971-8ead-4866-9a57-7eaaff64865e",
      "created_at": "2023-08-17T12:52:32.312388",
      "updated_at": "2023-08-17T12:52:32.312388"
    },
    {
      "term": "racial",
      "definition": "расовый",
      "id": "b2b621e6-0559-4de6-a1e6-dd8acedfaf66",
      "module_id": "68384971-8ead-4866-9a57-7eaaff64865e",
      "created_at": "2023-08-17T12:52:32.312388",
      "updated_at": "2023-08-17T12:52:32.312388"
    },
    {
      "term": "solar",
      "definition": "солнечный (напр. солнечная система)",
      "id": "116afce7-cab0-4f49-ae53-c5f104c10b60",
      "module_id": "68384971-8ead-4866-9a57-7eaaff64865e",
      "created_at": "2023-08-17T12:52:32.312388",
      "updated_at": "2023-08-17T12:52:32.312388"
    },
    {
      "term": "existing",
      "definition": "существующий",
      "id": "e1077e9e-3cf6-47ea-a877-a8b4d92bf397",
      "module_id": "68384971-8ead-4866-9a57-7eaaff64865e",
      "created_at": "2023-08-17T12:52:32.312388",
      "updated_at": "2023-08-17T12:52:32.312388"
    }
  ],
  "errors": null,
  "info": {
    "api_version": "0.0.1",
    "deprecated": false
  }
}
```


Листинг И.26 – Получение терминов без указания модулей


```
{
    "data": null,
    "errors": [
        {
            "error_type": "validation",
            "message": "not_set",
            "description": null,
            "status_code": 422,
            "detail": [
                {
                    "loc": [
                        "query",
                        "module_ids"
                    ],
                    "msg": "Input should be a valid list",
                    "type": "list_type"
                }
            ],
            "body": null
```

```
            }
        ],
        "info": {
            "api_version": "0.0.1",
            "deprecated": false
        }
    }
```

ПРИЛОЖЕНИЕ К
(РЕКОМЕНДУЕМОЕ)
КОД МОДУЛЬНЫХ ТЕСТОВ

Листинг К.1 – Код движка модульных тестов

```
# test_app.py
import asyncio
from contextlib import asynccontextmanager
from functools import lru_cache
from os.path import join, split

import httpx
import pytest
import pytest_asyncio
import sqlalchemy
from asgi_lifespan import LifespanManager
from fastapi import FastAPI
from orjson import orjson
from pytest_asyncio import is_async_test
from starlette.applications import Starlette
from starlette.responses import PlainTextResponse
from starlette.routing import Route
import logging

from the_remember.src.config.config import CONFIG
from the_remember.src.utils.exeptions.handlers import add_exception_handlers
from the_remember.tests.configs import BASE_URL
from the_remember.src.utils.init_app import add_middlewares, add_once_events,
add_swagger


@pytest.fixture(scope="session")
def event_loop():
    try:
        loop = asyncio.get_running_loop()
    except RuntimeError:
        loop = asyncio.new_event_loop()
    yield loop
    loop.close()


def init_routers(app: FastAPI):
    from the_remember.src.api import apps as api_apps

    # print(api_apps)
    # filename = join(split(split(split(split(__file__)[0])[0])[0])[0],
"merge-openapi", "openapi-merge.json")
    openapi_merge = {"inputs": [], "output": "./output.swagger.json"}

    def _get(tested: str, target_sub: str):
        index_ = 0
        for index, [a, b] in enumerate(zip(tested.strip("/").split("/"),
target_sub.strip("/").split('/'))):
            if (a != b):
                return 0
            index_ = index
        return (index_ + 1 if len(tested) < len(target_sub) else 0)

    for sub_apps_url, config in api_apps.items():
        _app = app
        app_prefix = ""
        if sub_apps_url.count("/") > 1:
            # print(sub_apps_url, sub_apps_url.count("/"))
            target_api = max(api_apps, key=lambda i: _get(i, sub_apps_url))
            if _get(target_api, sub_apps_url) != 0:
                _app = api_apps[target_api]["app"]
```

```
                app_prefix = target_api
            # else
        _app.mount(
            config.get('prefix', "").removeprefix(app_prefix),
            config["app"],
            name=config.get('tags', [None])[0]
        )
        openapi_merge["inputs"].append(
            {
                "inputURL":
f"http://backend:{CONFIG.BACKEND_PORT}{config.get('prefix',
'not_found')}/openapi.json",
                "pathModification": {
                    "prepend": config.get('prefix', 'not_found')
                }
            })

    return app, [app_["app"] for _, app_ in api_apps.items()]


def init_app(app: FastAPI):
    app, other_apps = init_routers(app)
    for app_ in [app, *other_apps]:
        add_middlewares(app_)
        add_exception_handlers(app_)

    app = add_once_events(app)
    app = add_swagger(app)
    return app


@lru_cache(maxsize=None)
def create_app():
    @asynccontextmanager
    async def lifespan(app):
        print("Starting up")
        yield
        print("Shutting down")

    app = FastAPI(lifespan=lifespan)
    app = init_app(app)
    return app


@pytest_asyncio.fixture(scope="session")
async def app():
    app = create_app()
    async with LifespanManager(app) as manager:
        print("We're in!")
        yield manager.app


@pytest_asyncio.fixture(scope="session")
async def client(app):
    logging.basicConfig()
    logging.getLogger('sqlalchemy').setLevel(logging.ERROR)

    print("---*****", BASE_URL)

    async with httpx.AsyncClient(app=app, base_url=BASE_URL) as client:
        print("Client is ready")
        yield client
```

```
@pytest_asyncio.fixture(scope="session")
async def users_auth(client):
    user = {
        "username": "maintainer",
        "email": "maintainer",
        "name": "maintainer",
        "surname": "maintainer",
        "password": "maintainer"
    }
    try:
        new_user = await client.post("/user/create", json=user)
        res = new_user.json()
        res = user
    except sqlalchemy.exc.IntegrityError as err:
        pass

    response = await client.post(
        "/auth/token",
        data={"username": user['username'], "password": user['password']},
        headers={"Content-Type": "application/x-www-form-urlencoded"}
    )
    assert response.status_code == 200
    user_auth_headers = {"Authorization": f"Bearer
{response.json()['access_token']}"}
    response = await client.get("/user/me", headers=user_auth_headers)
    res = response.json()['data']

    yield [
        (user_auth_headers, user | res)
    ]
```

Листинг К.2 – Код дополнительных инструментов для движка модульных тестов

```
import itertools
import uuid
from datetime import datetime
from typing import Callable, Iterable, Any

from httpx import Response


def dict_cutter(inp: dict, pass_condition=lambda i: False) -> dict:
    is_exists = itertools.product(
        *[[(True, i), (False, i)] for i in inp.keys()
          if pass_condition(i) is False],
    )
    for i in is_exists:
        # print(i)
        yield {excl_flag[1]: inp[excl_flag[1]] for excl_flag in i if
excl_flag[0]}


def optionals_as_null(dict_gen: Iterable[dict], options: set[str]) -> dict:
    for i in dict_gen:
        op_fields = options & set(i.keys())
        op_fields_iter = itertools.product(
            *[[(field, None), (field, i[field])] for field in op_fields],
        )
        for op in op_fields_iter:
            yield i | dict(op)
```

```
def func_calling(dict_gen: Iterable[dict]):
    for i in dict_gen:
        yield {k: (v() if isinstance(v, Callable) else v) for (k, v) in
i.items()}


def good_response_validate(response: Response, good=True):
    resp_data = response.json()
    # print("err ", resp_data)
    assert resp_data['errors'] is None or not bool(resp_data['errors'])
    assert 'api_version' in resp_data['info'] and 'deprecated' in
resp_data['info']
    assert resp_data['data'] is not None


def string_to_datetime(s: str) -> datetime:
    return datetime.strptime(s, '%Y-%m-%dT%H:%M:%S.%f')


class AnyOf(object):
    def __init__(self, values: list):
        self.values = values


def safe_convert_to_uuid(string):
    try:
        return uuid.UUID(str(string))
    except (ValueError, AttributeError):
        return None

def any_of_replacer(dict_gen: Iterable[dict]):
    for i in dict_gen:
        any_of_fields = {k: v for (k, v) in i.items() if isinstance(v,
AnyOf)}
        one_of_any = itertools.product(
            *[
                [(field, val) for val in value.values]
                for (field, value) in any_of_fields.items()
            ],
        )
        for modif_fields in one_of_any:
            yield i | dict(modif_fields)


def decorate_tests_count_wraps(func: Callable):
    hashed_data: set[frozenset[Any | tuple[str, Any]]] = set()
    async def decorator(*args, **kwargs):
        nonlocal hashed_data
        r = await func(*args, **kwargs)
        (res, test_count) = r
        as_hash = frozenset([*args, *[tuple(*i) for i in kwargs.items()]])

        if as_hash in hashed_data:
            test_count = 0
        else:
            hashed_data.add(as_hash)
        return res, test_count
    return decorator
```

Листинг К.3 – Код движка модульных тестов авторизации

```
from typing import Any
```

```
import pytest


@pytest.mark.asyncio(scope="session")
async def test_auth(client, users_auth: list[(Any, dict)]):
    tests_count = 0
    client = client
    print("Testing auth")
    print(users_auth)
    response = await client.post(
        "/auth/token",
        data={"username": users_auth[0][1]['username'], "password":
users_auth[0][1]['password']},
        headers={"Content-Type": "application/x-www-form-urlencoded"},
    )
    assert response.status_code == 200
    assert response.json().get("token_type") == "bearer" and
isinstance(response.json().get("access_token"), str)
    tests_count += 1
    user_auth_headers = {"Authorization": f"Bearer
{response.json()['access_token']}"}

    # print(response.json())
    # assert response.text == "Hello, world!"
    print(f"OK, tests passed {tests_count}")
    return user_auth_headers, tests_count
```

Листинг К.4 – Код движка модульных микросервиса папок

```
import asyncio
import json
import uuid
from datetime import datetime
from random import random, randint
from typing import Any
from functools import lru_cache

import httpx
import pytest
from async_lru import alru_cache
from sqlalchemy import select

# from the_remember.src.utils.get_data_from_auth_token import
get_db_session_cm
from the_remember.tests.api_test.test_users import create_test_user
from the_remember.tests.urils import dict_cutter, optionals_as_null,
func_calling, good_response_validate, \
    string_to_datetime, AnyOf, any_of_replacer, safe_convert_to_uuid,
decorate_tests_count_wraps


@decorate_tests_count_wraps
@alru_cache(maxsize=None)
async def create_test_folder_graph(client, test_name="unknown_folder_test"):
    """
    return:
        (
        folders: dict[str, Any],
        users_auth: dict[user_id:str, auth_header_dict:dict[str, str]]
        tests_count: int
        )
    """
    tests_count = 0
```

```
    users = list(func_calling([{
        "id": lambda: str(uuid.uuid4()),
        "username": lambda: f"test_{test_name}_{i}_{str(randint(0,
1000000000))}",
        "email": lambda: f"test_{test_name}_{i}_{str(randint(0,
1000000000))}@example.com",
        "name": f"test_{test_name}_{i}",
        "surname": f"test_{test_name}_{i}",
        "password": f"test_{test_name}_{i}"
    } for i in range(3)]))

    users: Any = await asyncio.gather(
        *[asyncio.create_task(create_test_user(client, i, no_validate=False))
for i in users])
    users, _test_counts = zip(*users)
    tests_count += sum(_test_counts)
    users: list[tuple[dict[str, Any], dict[str, str]]]

    folders = list(func_calling([{
        "user": users[0],
        "id": lambda: str(uuid.uuid4()),
        "name": "test",
        "root_folder_id": None
    }] * 3))
    sub_folders = list(func_calling([{
        "user": users[0],
        "id": lambda: str(uuid.uuid4()),
        "name": "test",
        "root_folder_id": folders[min(i // 3, len(folders))]["id"]
    } for i in range(8)]))
    sub_sub_folders = list(func_calling([{
        "user": users[0],
        "id": lambda: str(uuid.uuid4()),
        "name": "test",
        "root_folder_id": folders[min(i // 8, len(sub_folders))]["id"]
    } for i in range(12)]))

    folders = await asyncio.gather(*[
        create_test_folder(
            client,
            {k: v for k, v in i.items() if 'user' != k}, i['user'][1],
            no_validate=False) for i in folders])
    folders += await asyncio.gather(*[create_test_folder(
        client,
        {k: v for k, v in i.items() if 'user' != k}, i['user'][1],
        no_validate=False) for i in sub_folders])
    folders += await asyncio.gather(*[create_test_folder(
        client,
        {k: v for k, v in i.items() if 'user' != k}, i['user'][1],
        no_validate=False) for i in sub_sub_folders])
    folders: Any
    folders, _test_count = zip(*folders)
    tests_count += sum(_test_counts)

    folders: dict[str, dict[str, Any]] = {i["data"]["id"]: i["data"] for i in
folders}

    return (folders, {i[0]["id"]: i[1] for i in users}), tests_count


async def create_test_folder(client, folder: dict, auth_headers,
no_validate=True):
    tests_count = 0
```

```
    response = await client.post("/folder/create", json=folder,
headers=auth_headers)
    new_folder = response.json()
    if no_validate is False:
        tests_count += 1
        assert response.status_code in [201, 456]
        if response.status_code == 201:
            good_response_validate(response)
            assert all([v == new_folder['data'][k]
                        for k, v in folder.items()
                        if k in new_folder['data']])
        elif response.status_code == 456:
            assert any([i.get('message') == 'duplicate_unique_field' for i in
new_folder.get('errors', [dict()])])
            response = await client.get(f"/folder/{folder['id']}",
headers=auth_headers)
            assert response.status_code == 200
            good_response_validate(response)
            tests_count += 1
            new_folder = response.json()
        else:
            raise NotImplementedError(f"Unexpected status code:
{response.status_code}")
    return new_folder, tests_count


def get_sub_folders(folder_id: str, folders: dict):
    return {k: v for k, v in folders.items() if v.get("root_folder_id") ==
folder_id}


@pytest.mark.asyncio(scope="session")
async def test_folder_test_creation(client):
    tests_count = 0

    ((folders, auth_to_users), _tests) = await
create_test_folder_graph(client)
    tests_count += _tests

    new_folder = {
        "id": lambda: str(uuid.uuid4()),
        "name": "test",
        "root_folder_id": list(folders)[0]
    }
    _target_folder_id = new_folder['root_folder_id']

    user_auth_headers =
auth_to_users[folders[_target_folder_id]['author_id']]
    optional = {"id", "root_folder_id"}
    required = set(new_folder) - optional
    unique = {"id"}

    # validate input data testing
    for folder_data in
func_calling(optionals_as_null(dict_cutter(new_folder), optional)):
        tests_count += 1
        new_user = await client.post("/folder/create", json=folder_data,
headers=user_auth_headers)
        res = new_user.json()
        if set(folder_data) & required == required:
            good_response_validate(new_user)
            assert new_user.status_code == 201
        else:
            assert new_user.status_code == 422
```

```
    print(f"OK, tests passed {tests_count}")


@pytest.mark.asyncio(scope="session")
async def test_folder_test_one_folder_get(client):
    tests_count = 0

    ((folders, auth_to_users), _tests) = await
create_test_folder_graph(client)
    tests_count += _tests

    new_folder = {
        "folder_id": AnyOf([list(folders)[0], uuid.uuid4(), 'null'])
    }
    _target_folder_id = new_folder['folder_id'].values[0]

    user_auth_headers =
auth_to_users[folders[_target_folder_id]['author_id']]
    optional = set()
    required = set(new_folder) - optional

    # validate input data testing
    for folder_data in
func_calling(any_of_replacer(optionals_as_null(dict_cutter(new_folder),
optional))):
        tests_count += 1
        response = await client.get(f"/folder/{folder_data.get('folder_id',
'null')}",
                                    headers=user_auth_headers)
        exist_folder = response.json()
        if safe_convert_to_uuid(folder_data.get('folder_id')):
            if folder_data.get('folder_id') in folders:
                good_response_validate(response)
                assert response.status_code == 200
            else:
                assert response.status_code == 404
        else:
            assert response.status_code == 422

    print(f"OK, tests passed {tests_count}")


@pytest.mark.asyncio(scope="session")
async def test_folder_test_all_personal_folders_get(client):
    tests_count = 0

    ((folders, auth_to_users), _tests) = await
create_test_folder_graph(client)
    tests_count += _tests

    new_folder = {
        "limit": AnyOf([100, 1, 0, -1, -20, 101, 200]),
        "offset": AnyOf([100, 1, 0, -1, -20, 101, 200]),
        "updated_after": datetime(2020, 1, 1).strftime('%Y-%m-
%dT%H:%M:%S.%f'),
        "author_only": AnyOf([False, True]),
        "auth": AnyOf(list(auth_to_users.values())),
    }
    # _target_folder_id = new_folder['folder_id'].values[0]

    user_auth_headers = list(auth_to_users.values())
    optional = {"limit", "offset", "updated_after", "author_only"}
    required = set(new_folder) - optional
```

```
    query_params = {"limit", "offset", "updated_after", "author_only"}

    # validate input data testing
    for folder_data in
func_calling(any_of_replacer(optionals_as_null(dict_cutter(new_folder),
optional))):
        tests_count += 1
        # print(folder_data)
        response = await client.get(
            f"/folder/personalize/all",
            params={k: v for k, v in folder_data.items()},
            headers=folder_data.get('auth'))

        exist_folder = response.json()
        if folder_data.get('auth') is None:
            assert response.status_code == 401

        elif any(folder_data.get(i, "no_set_param") is None for i in
query_params):
            assert response.status_code == 422

        elif ((folder_data.get('limit') is None) or 0 <
(folder_data.get('limit', 1)) <= 100) \
                and ((folder_data.get('offset') is None) or 0 <=
(folder_data.get('offset', 1))) \
                :
            # if folder_data.get('folder_id') in folders:
            # print(folder_data)
            good_response_validate(response)
            assert response.status_code == 200
            # else:
            #     assert response.status_code == 404
        else:
            assert response.status_code == 422

    print(f"OK, tests passed {tests_count}")


@pytest.mark.asyncio(scope="session")
async def test_folder_test_sub_folders_get(client):
    tests_count = 0

    ((folders, auth_to_users), _tests) = await
create_test_folder_graph(client)
    tests_count += _tests

    new_folder = {
        "folder_id": AnyOf([list(folders)[0], str(uuid.uuid4()), 'null']),
        "limit": AnyOf([100, 1, 0, -1, -20, 101, 200]),
        "offset": AnyOf([100, 1, 0, -1, -20, 101, 200]),
        "updated_after": datetime(2020, 1, 1).strftime('%Y-%m-
%dT%H:%M:%S.%f')
    }
    _target_folder_id = new_folder['folder_id'].values[0]

    user_auth_headers =
auth_to_users[folders[_target_folder_id]['author_id']]
    optional = {"limit", "offset", "updated_after"}
    required = set(new_folder) - optional
    query_params = {"limit", "offset", "updated_after"}

    # validate input data testing
```

```
    for folder_data in
func_calling(any_of_replacer(optionals_as_null(dict_cutter(new_folder),
optional))):
        tests_count += 1
        response = await client.get(
            f"/folder/{folder_data.get('folder_id', 'null')}/sub_folders",
            params=folder_data,
            headers=user_auth_headers)
        exist_folder = response.json()

        if any(folder_data.get(i, "no_set_param") is None for i in
query_params):
            assert response.status_code == 422

        elif safe_convert_to_uuid(folder_data.get('folder_id')) \
                and ((folder_data.get('limit') is None) or 0 <
(folder_data.get('limit', 1)) <= 100) \
                and ((folder_data.get('offset') is None) or 0 <=
(folder_data.get('offset', 1))) \
                :
            # if folder_data.get('folder_id') in folders:
            # print(folder_data)
            good_response_validate(response)
            assert response.status_code == 200
            # else:
            #     assert response.status_code == 404
        else:
            assert response.status_code == 422

    print(f"OK, tests passed {tests_count}")


@pytest.mark.asyncio(scope="session")
async def test_folder_exists(client):
    tests_count = 0
    ((folders, auth_to_users), _tests) = await
create_test_folder_graph(client)
    tests_count += _tests

    error_ids = [
        (str(uuid.uuid1())),
        (str(uuid.uuid3(uuid.uuid4(), name='Jon'))),
        (str(uuid.uuid4())),
        (str(uuid.uuid5(uuid.uuid4(), name='Jon'))),
    ]

    folders |= {i: None for i in error_ids}

    params = {
        "folder_id": AnyOf(list(folders)),
        "root_folder_id": AnyOf(list(folders)),
        "author_id": AnyOf(list(auth_to_users)),
    }
    optional = {"root_folder_id", "author_id"}
    required = set(params) - optional

    for test_data in
func_calling(any_of_replacer(optionals_as_null(dict_cutter(params),
optional))):
        tests_count += 1
        test_data['folder_id'] = test_data.get('folder_id', 'null')
        response = await client.get(
            f"/folder/exist/{test_data['folder_id']}", params={k: v for k, v
in test_data.items() if k != 'folder_id'},
```

```
            headers=auth_to_users.get(test_data.get('author_id')) or
list(auth_to_users.values())[0]
        )
        resp_data = response.text
        # print("resp_data", resp_data)
        test_data = dict(**test_data)
        test_data['id'] = test_data.pop('folder_id')
        if any(i is None or i == 'null' for i in test_data.values()):
            assert response.status_code == 422
        elif (folders.get(test_data['id'])
              and all(v == folders[test_data['id']][k]
                      for k, v in test_data.items())):
            assert response.status_code == 200
            assert resp_data == 'true'
        else:
            assert response.status_code == 200
            assert resp_data == 'false'
    print(f"OK, tests passed {tests_count}")
```

Листинг К.5 – Код движка модульных тестов микросервиса модулей

```
import asyncio
import uuid
from datetime import datetime
from random import random, choice
from typing import Any

import pytest
from async_lru import alru_cache

from the_remember.tests.api_test.test_folder import create_test_folder_graph
from the_remember.tests.urils import func_calling, good_response_validate,
optionals_as_null, dict_cutter, AnyOf, \
    any_of_replacer, safe_convert_to_uuid, decorate_tests_count_wraps


@decorate_tests_count_wraps
@alru_cache(maxsize=None)
async def create_test_modules_graph(client, test_name="unknown_folder_test"):
    """
    return:
        (
        folders: dict[str, Any],
        users_auth: dict[user_id:str, auth_header_dict:dict[str, str]]
        tests_count: int
        )
    """
    tests_count = 0
    ((folders, auth_to_users), _tests) = await
create_test_folder_graph(client)

    modules = list(func_calling([{
        "user": list(auth_to_users.values())[0],
        "id": lambda: str(uuid.uuid4()),
        "name": "test",
        "description": "some description",
        "root_folder_id": ([{"id": None}] + list(folders.values()))[min(i //
3, len(folders))]["id"]
    } for i in range(20)]))

    modules = await asyncio.gather(*[
        create_test_module(
            client,
```

```
            {k: v for k, v in i.items() if 'user' != k}, i['user'],
            no_validate=False) for i in modules])

    modules: Any
    modules, _test_count = zip(*modules)
    tests_count += sum(_test_count)

    modules: dict[str, dict[str, Any]] = {i["data"]["id"]: i["data"] for i in
modules}

    return (modules, folders, auth_to_users), tests_count


async def create_test_module(client, module: dict, auth_headers,
no_validate=True):
    tests_count = 0
    response = await client.post("/module/create", json=module,
headers=auth_headers)
    new_folder = response.json()
    if no_validate is False:
        tests_count += 1
        assert response.status_code in [201, 456]
        if response.status_code == 201:
            good_response_validate(response)
            assert all([v == new_folder['data'][k]
                        for k, v in module.items()
                        if k in new_folder['data']])
        elif response.status_code == 456:
            assert any([i.get('message') == 'duplicate_unique_field' for i in
new_folder.get('errors', [dict()])])
            response = await client.get(f"/module/{module['id']}",
headers=auth_headers)
            assert response.status_code == 200
            good_response_validate(response)
            tests_count += 1
            new_folder = response.json()
        else:
            raise NotImplementedError(f"Unexpected status code:
{response.status_code}")
    return new_folder, tests_count


@pytest.mark.asyncio(scope="session")
async def test_module_test_creation(client):
    tests_count = 0

    ((exists_modules, folders, auth_to_users), _tests) = await
create_test_modules_graph(client)
    tests_count += _tests

    new_module = {
        "id": AnyOf([lambda: str(uuid.uuid4()), lambda:
str(choice(list(exists_modules)))]),
        "name": "test",
        "description": "some description",
        "root_folder_id": AnyOf([str(choice(list(folders))) for _ in
range(5)] + [str(uuid.uuid4())]),
        "auth": AnyOf([v | {"x-id": k} for k, v in auth_to_users.items()]),

    }
    # _target_folder_id = new_folder['root_folder_id']

    # user_auth_headers =
auth_to_users[folders[_target_folder_id]['author_id']]
```

```
    optional = {"id", "root_folder_id", "description"}
    required = set(new_module) - optional
    unique = {"id"}
    body = {"id", "root_folder_id", "description", "name"}

    # validate input data testing
    for module_data in
func_calling(any_of_replacer(optionals_as_null(dict_cutter(new_module),
optional))):
        tests_count += 1
        response = await client.post(
            "/module/create",
            json={k: v for k, v in module_data.items() if k in body},
            headers=module_data.get("auth", dict())
        )
        created_module = response.json()
        if module_data.get("auth") is None:
            assert response.status_code == 401
        elif set(module_data) & required != required:
            assert response.status_code == 422
        elif ((r_f_id := module_data.get("root_folder_id")) is not None
              and (r_f := folders.get(r_f_id)) is not None
              and (curr_author := module_data.get("auth", dict()).get("x-
id")) is not None
              and r_f['author_id'] != curr_author):
            # Корневая папка была создана другим человеком
            assert response.status_code == 409
        elif ((r_f_id := module_data.get("root_folder_id")) is not None
              and (r_f := folders.get(r_f_id)) is None):
            # Корневой папки не существует
            assert response.status_code == 409
        elif module_data.get("id") in exists_modules:
            assert response.status_code == 456

        elif set(module_data) & required == required:
            good_response_validate(response)
            assert response.status_code == 201
        else:
            assert response.status_code == 422

    print(f"OK, tests passed {tests_count}")


@pytest.mark.asyncio(scope="session")
async def test_module_test_one_module_get(client):
    tests_count = 0

    ((exists_modules, folders, auth_to_users), _tests) = await
create_test_modules_graph(client)
    tests_count += _tests

    checking_module = {
        "module_id": AnyOf([list(exists_modules)[0], uuid.uuid4(), 'null']),
        "auth": AnyOf([v | {"x-id": k} for k, v in auth_to_users.items()]),
    }

    optional = set("auth")
    required = set(checking_module) - optional
    path_params = {"module_id"}

    # validate input data testing
    for module_data in
func_calling(any_of_replacer(optionals_as_null(dict_cutter(checking_module),
optional))):
```

```
        tests_count += 1
        response = await client.get(f"/module/{module_data.get('module_id',
'null')}",
                                    headers=module_data.get("auth", dict()))
        exist_module = response.json()
        if module_data.get("auth") is None:
            assert response.status_code == 401
        elif safe_convert_to_uuid(module_data.get('module_id')):
            if module_data.get('module_id') in exists_modules:
                good_response_validate(response)
                assert response.status_code == 200
            else:
                assert response.status_code == 404
        else:
            assert response.status_code == 422

    print(f"OK, tests passed {tests_count}")


@pytest.mark.asyncio(scope="session")
async def test_module_test_all_personalize_module_get(client):
    tests_count = 0

    ((exists_modules, folders, auth_to_users), _tests) = await
create_test_modules_graph(client)
    tests_count += _tests

    get_modules_params = {
        "limit": AnyOf([100, 1, 0, -1, -20, 101, 200]),
        "offset": AnyOf([100, 1, 0, -1, -20, 101, 200]),
        "author_only": AnyOf([False, True]),
        "updated_after": datetime(2020, 1, 1).strftime('%Y-%m-
%dT%H:%M:%S.%f'),
        "auth": AnyOf([v | {"x-id": k} for k, v in auth_to_users.items()]),
    }

    optional = {"limit", "offset", "updated_after", "author_only", "auth"}
    required = set(get_modules_params) - optional
    query_params = {"limit", "offset", "updated_after", "author_only"}

    # validate input data testing
    for req_params in
func_calling(any_of_replacer(optionals_as_null(dict_cutter(get_modules_params
), optional))):
        tests_count += 1
        response = await client.get(
            f"/module/personalize/all",
            params={k: v for k, v in req_params.items() if k in
query_params},
            headers=req_params.get("auth", dict()))
        exist_modules = response.json()

        if req_params.get("auth") is None:
            assert response.status_code == 401
        elif any(req_params.get(i, "no_set_param") is None for i in
query_params):
            assert response.status_code == 422

        elif ((req_params.get('limit') is None) or 0 <
(req_params.get('limit', 1)) <= 100) \
                and ((req_params.get('offset') is None) or 0 <=
(req_params.get('offset', 1))) \
                :
            # if folder_data.get('folder_id') in folders:
```

```
            # print(folder_data)
            good_response_validate(response)
            assert response.status_code == 200
            # else:
            #     assert response.status_code == 404
        else:
            assert response.status_code == 422

    print(f"OK, tests passed {tests_count}")


@pytest.mark.asyncio(scope="session")
async def test_module_test_module_from_folder_get(client):
    tests_count = 0

    ((exists_modules, folders, auth_to_users), _tests) = await
create_test_modules_graph(client)
    tests_count += _tests

    get_modules_params = {
        "folder_id": AnyOf(
            [str(choice(list(folders))) for _ in range(3)]
            + [str(choice(list({i['root_folder_id']
                                for i in exists_modules.values()
                                if i.get('root_folder_id') is not None})))
               for _ in range(3)]
            + [str(uuid.uuid4()), 'null']),
        "limit": AnyOf([100, 1, 0, -1, -20, 101, 200]),
        "offset": AnyOf([100, 1, 0, -1, -20, 101, 200]),
        # "author_only": AnyOf([False, True]),
        "updated_after": datetime(2020, 1, 1).strftime('%Y-%m-
%dT%H:%M:%S.%f'),
        "auth": AnyOf([v | {"x-id": k} for k, v in auth_to_users.items()]),
    }

    optional = {"limit", "offset", "updated_after", "auth"}
    required = set(get_modules_params) - optional
    query_params = {"limit", "offset", "updated_after"}

    # validate input data testing
    for req_params in
func_calling(any_of_replacer(optionals_as_null(dict_cutter(get_modules_params
), optional))):
        tests_count += 1
        # print(req_params)
        response = await client.get(
            f"/module/from_folder/{req_params.get('folder_id', 'null')}",
            params={k: v for k, v in req_params.items() if k in
query_params},
            headers=req_params.get("auth", dict()))
        exist_modules = response.json()

        if req_params.get("auth") is None:
            assert response.status_code == 401
        elif any(req_params.get(i, "no_set_param") is None for i in
query_params):
            assert response.status_code == 422

        elif safe_convert_to_uuid(req_params.get('folder_id')) \
                and ((req_params.get('limit') is None) or 0 <
(req_params.get('limit', 1)) <= 100) \
                and ((req_params.get('offset') is None) or 0 <=
(req_params.get('offset', 1))) \
                :
```

```
            # if folder_data.get('folder_id') in folders:
            # print(folder_data)
            good_response_validate(response)
            assert response.status_code == 200
            # else:
            #     assert response.status_code == 404
        else:
            assert response.status_code == 422

    print(f"OK, tests passed {tests_count}")
```

Листинг К.6 – Код движка модульных тестов микросервиса терминов

```
import asyncio
import uuid
from datetime import datetime
from random import random, choice
from typing import Any, Sequence

import pytest
from async_lru import alru_cache

from the_remember.tests.api_test.test_folder import create_test_folder_graph
from the_remember.tests.api_test.test_modules import
create_test_modules_graph
from the_remember.tests.urils import func_calling, good_response_validate,
optionals_as_null, dict_cutter, AnyOf, \
    any_of_replacer, safe_convert_to_uuid, decorate_tests_count_wraps


@decorate_tests_count_wraps
@alru_cache(maxsize=None)
async def create_test_terms_graph(client, test_name="unknown_folder_test"):
    """
    return:
        (
        folders: dict[str, Any],
        users_auth: dict[user_id:str, auth_header_dict:dict[str, str]]
        tests_count: int
        )
    """
    tests_count = 0
    ((modules, folders, auth_to_users), _tests) = await
create_test_modules_graph(client)

    terms = list(func_calling([{
        "user": list(auth_to_users.values())[0],
        "id": lambda: str(uuid.uuid4()),
        "term": "test",
        "definition": "тест",
        "module_id": (list(modules.values()))[min(i // 10,
len(modules))]["id"]
    } for i in range(200)]))

    terms = await asyncio.gather(*[
        create_test_term(
            client,
            {k: v for k, v in i.items() if 'user' != k}, i['user'],
            no_validate=False) for i in terms])

    terms: Any
    terms, _test_count = zip(*terms)
    tests_count += sum(_test_count)
```

```
    terms: dict[str, dict[str, Any]] = {i["data"]["id"]: i["data"] for i in
terms}

    return (terms, modules, folders, auth_to_users), tests_count


async def create_test_term(client, module: dict, auth_headers,
no_validate=True):
    tests_count = 0
    response = await client.post("/term/create", json=module,
headers=auth_headers)
    new_term = response.json()
    if no_validate is False:
        tests_count += 1
        assert response.status_code in [201, 456]
        if response.status_code == 201:
            good_response_validate(response)
            assert all([v == new_term['data'][k]
                        for k, v in module.items()
                        if k in new_term['data']])
        elif response.status_code == 456:
            assert any([i.get('message') == 'duplicate_unique_field' for i in
new_term.get('errors', [dict()])])
            response = await client.get(f"/module/{module['id']}",
headers=auth_headers)
            assert response.status_code == 200
            good_response_validate(response)
            tests_count += 1
            new_term = response.json()
        else:
            raise NotImplementedError(f"Unexpected status code:
{response.status_code}")
    return new_term, tests_count


@pytest.mark.asyncio(scope="session")
async def test_term_test_creation(client):
    tests_count = 0

    ((exist_terms, modules, folders, auth_to_users), _tests) = await
create_test_terms_graph(client)
    tests_count += _tests

    new_module = {
        "id": AnyOf([lambda: str(uuid.uuid4()), lambda:
str(choice(list(exist_terms)))]),
        "term": "test",
        "definition": "тест",
        "module_id": AnyOf([str(choice(list(modules))) for _ in range(5)] +
[str(uuid.uuid4())]),
        "auth": AnyOf([v | {"x-id": k} for k, v in auth_to_users.items()]),

    }
    # _target_folder_id = new_folder['root_folder_id']

    # user_auth_headers =
auth_to_users[folders[_target_folder_id]['author_id']]
    optional = {"id", "auth"}
    required = set(new_module) - optional
    unique = {"id"}
    body = {"id", "module_id", "term", "definition"}

    # validate input data testing
```

```
    for term_data in
func_calling(any_of_replacer(optionals_as_null(dict_cutter(new_module),
optional))):
        tests_count += 1
        response = await client.post(
            "/term/create",
            json={k: v for k, v in term_data.items() if k in body},
            headers=term_data.get("auth", dict())
        )
        created_module = response.json()
        if term_data.get("auth") is None:
            assert response.status_code == 401
        elif set(term_data) & required != required:
            assert response.status_code == 422
        elif ((r_m_id := term_data.get("module_id")) is not None
              and (r_m := modules.get(r_m_id)) is not None
              and (curr_author := term_data.get("auth", dict()).get("x-id"))
is not None
              and r_m['author_id'] != curr_author):
            # Корневой модуль была создана другим человеком
            assert response.status_code == 409
        elif ((r_m_id := term_data.get("module_id")) is not None
              and (r_m := modules.get(r_m_id)) is None):
            # Корневого модуля не существует
            assert response.status_code == 409
        # elif term_data.get("m") in modules:
        #     assert response.status_code == 456
        elif term_data.get("id") in exist_terms:
            assert response.status_code == 456
        elif set(term_data) & required == required:
            good_response_validate(response)
            assert response.status_code == 201
        else:
            assert response.status_code == 422

    print(f"OK, tests passed {tests_count}")


@pytest.mark.asyncio(scope="session")
async def test_term_test_one_term_get(client):
    tests_count = 0

    ((exist_terms, modules, folders, auth_to_users), _tests) = await
create_test_terms_graph(client)
    tests_count += _tests

    checking_module = {
        "term_id": AnyOf([list(exist_terms)[0], uuid.uuid4(), 'null']),
        "auth": AnyOf([v | {"x-id": k} for k, v in auth_to_users.items()]),
    }

    optional = set("auth")
    required = set(checking_module) - optional
    path_params = {"term_id"}

    # validate input data testing
    for term_data in
func_calling(any_of_replacer(optionals_as_null(dict_cutter(checking_module),
optional))):
        tests_count += 1
        response = await client.get(f"/term/{term_data.get('module_id',
'null')}",
                                    headers=term_data.get("auth", dict()))
        exist_term = response.json()
```

```
        if term_data.get("auth") is None:
            assert response.status_code == 401
        elif safe_convert_to_uuid(term_data.get('module_id')):
            if term_data.get('module_id') in exist_terms:
                good_response_validate(response)
                assert response.status_code == 200
            else:
                assert response.status_code == 404
        else:
            assert response.status_code == 422

    print(f"OK, tests passed {tests_count}")


@pytest.mark.asyncio(scope="session")
async def test_term_test_all_term_get(client):
    tests_count = 0

    ((exist_terms, modules, folders, auth_to_users), _tests) = await
create_test_terms_graph(client)
    tests_count += _tests

    get_modules_params = {
        "module_ids": AnyOf([
            [],
            [str(uuid.uuid4())],
            [str(uuid.uuid4()), str(uuid.uuid4())],
            [list(modules)[0]],
            [choice(list(modules)) ],
            [choice(list(modules)), choice(list(modules))],
            [choice(list(modules)) for _ in range(5)],
            [str(uuid.uuid4()), str(uuid.uuid4())],
            [(_m := choice(list(modules))), _m],
        ]),
        "limit": AnyOf([100, 1, 0, -1, -20, 101, 200]),
        "offset": AnyOf([100, 1, 0, -1, -20, 101, 200]),
        "author_only": AnyOf([False, True]),
        "updated_after": datetime(2020, 1, 1).strftime('%Y-%m-
%dT%H:%M:%S.%f'),
        "auth": AnyOf([v | {"x-id": k} for k, v in auth_to_users.items()]),
    }

    optional = {"limit", "offset", "updated_after", "author_only", "auth"}
    required = set(get_modules_params) - optional
    query_params = {"limit", "offset", "updated_after", "author_only",
"module_ids"}

    # validate input data testing
    for req_params in
func_calling(any_of_replacer(optionals_as_null(dict_cutter(get_modules_params
), optional))):
        tests_count += 1
        response = await client.get(
            f"/term/all",
            params={k: v for k, v in req_params.items() if k in
query_params},
            headers=req_params.get("auth", dict()))
        got_terms = response.json()

        if req_params.get("auth") is None:
            assert response.status_code == 401
        elif any(
                (req_params.get(i, "no_set_param") is None
                if isinstance(i, Sequence)
```

```
                else req_params.get(i, "no_set_param")is None)
                for i in query_params):
            assert response.status_code == 422

        elif ((req_params.get('limit') is None) or 0 <
(req_params.get('limit', 1)) <= 100) \
                and ((req_params.get('offset') is None) or 0 <=
(req_params.get('offset', 1))) \
                and (req_params.get('module_ids') is not None and
bool(req_params.get('module_ids'))):
            # if folder_data.get('folder_id') in folders:
            # print(folder_data)
            good_response_validate(response)
            assert response.status_code == 200
            # else:
            #     assert response.status_code == 404
        else:
            assert response.status_code == 422

    print(f"OK, tests passed {tests_count}")


@pytest.mark.asyncio(scope="session")
async def test_term_test_all_personalize_term_get(client):
    tests_count = 0

    ((exist_terms, modules, folders, auth_to_users), _tests) = await
create_test_terms_graph(client)
    tests_count += _tests

    get_modules_params = {
        "module_ids": AnyOf([
            [],
            [str(uuid.uuid4())],
            [str(uuid.uuid4()), str(uuid.uuid4())],
            [list(modules)[0]],
            [choice(list(modules)) ],
            [choice(list(modules)), choice(list(modules))],
            [choice(list(modules)) for _ in range(5)],
            [str(uuid.uuid4()), str(uuid.uuid4())],
            [(_m := choice(list(modules))), _m],
        ]),
        "limit": AnyOf([100, 1, 0, -1, -20, 101, 200]),
        "offset": AnyOf([100, 1, 0, -1, -20, 101, 200]),
        "author_only": AnyOf([False, True]),
        "updated_after": datetime(2020, 1, 1).strftime('%Y-%m-
%dT%H:%M:%S.%f'),
        "auth": AnyOf([v | {"x-id": k} for k, v in auth_to_users.items()]),
    }

    optional = {"limit", "offset", "updated_after", "author_only", "auth"}
    required = set(get_modules_params) - optional
    query_params = {"limit", "offset", "updated_after", "author_only",
"module_ids"}

    # validate input data testing
    for req_params in
func_calling(any_of_replacer(optionals_as_null(dict_cutter(get_modules_params
), optional))):
        tests_count += 1
        response = await client.get(
            f"/term/personalize/all",
            params={k: v for k, v in req_params.items() if k in
query_params},
```

```
            headers=req_params.get("auth", dict()))
        got_terms = response.json()

        if req_params.get("auth") is None:
            assert response.status_code == 401
        elif any(
                (req_params.get(i, "no_set_param") is None
                if isinstance(i, Sequence)
                else req_params.get(i, "no_set_param")is None)
                for i in query_params):
            assert response.status_code == 422

        elif ((req_params.get('limit') is None) or 0 <
(req_params.get('limit', 1)) <= 100) \
                and ((req_params.get('offset') is None) or 0 <=
(req_params.get('offset', 1))) \
                and (req_params.get('module_ids') is not None and
bool(req_params.get('module_ids'))):
            # if folder_data.get('folder_id') in folders:
            # print(folder_data)
            good_response_validate(response)
            assert response.status_code == 200
            # else:
            #     assert response.status_code == 404
        else:
            assert response.status_code == 422

    print(f"OK, tests passed {tests_count}")
```

Листинг К.7 – Код движка модульных тестов микросервиса пользователей

```
import json
import uuid
from datetime import datetime
from random import random, randint
from typing import Any

import httpx
import pytest
from async_lru import alru_cache
from sqlalchemy import select

from the_remember.src.utils.get_data_from_auth_token import get_db_session_cm
from the_remember.tests.api_test.test_auth import test_auth
from the_remember.tests.urils import dict_cutter, optionals_as_null,
func_calling, good_response_validate, \
    string_to_datetime, decorate_tests_count_wraps


# @decorate_tests_count_wraps
# @alru_cache(maxsize=None)
async def create_test_user(client, user: dict, no_validate=True):
    tests_count = 0
    create_response = await client.post("/user/create", json=user)
    new_user = create_response.json()
    if no_validate is False:
        tests_count += 1
        assert create_response.status_code in [201, 456]
        if create_response.status_code == 201:
            good_response_validate(create_response)
            assert all([v == new_user['data'][k]
                        for k, v in user.items()
                        if k in new_user['data']])
```

```
    (auth_headers, _tests) = await test_auth(client, [(None, user)])

    if create_response.status_code == 456:
        if no_validate is False:
            assert any([
                i.get('message') == 'duplicate_unique_field'
                for i in new_user.get('errors', [dict()])])

        response = await client.get(f"/user/{user['id']}",
headers=auth_headers)
        assert response.status_code == 200
        good_response_validate(response)
        tests_count += 1
        new_user = response.json()
    new_user['data']: dict[str, Any]
    return (new_user['data'], auth_headers), tests_count


@pytest.mark.asyncio(scope="session")
async def test_user_creation(client, users_auth):
    created_entities = []
    tests_count = 0
    user = {
        "id": lambda: str(uuid.uuid4()),
        "username": lambda: "test_user_" + str(randint(0, 1000000000)),
        "email": lambda: "test_user_" + str(randint(0, 1000000000)) +
"@example.com",
        "name": "test_user",
        "surname": "test_user",
        "password": "test_user"
    }
    optional = {"id"}
    required = set(user) - optional
    unique = {"id", "username", "email"}

    # validate input data testing
    for user_data in func_calling(optionals_as_null(dict_cutter(user),
optional)):
        tests_count += 1
        new_user = await client.post("/user/create", json=user_data)
        res = new_user.json()
        if set(user_data) & required == required:
            created_entities.append(res)
            good_response_validate(new_user)
            assert new_user.status_code == 201
        else:
            assert new_user.status_code == 422

    test_unic_user = list(func_calling([user]))[0]
    new_user_resp = await client.post("/user/create", json=test_unic_user)
    new_user = new_user_resp.json()
    created_entities.append(new_user)
    assert new_user_resp.status_code == 201
    good_response_validate(new_user_resp)
    tests_count += 1

    # Validate unique fields
    for real_unic_fields in func_calling(dict_cutter(user,
pass_condition=lambda key: key not in unique)):
        tests_count += 1
        user_data = test_unic_user | real_unic_fields
        new_user = await client.post("/user/create", json=user_data)
        res = new_user.json()
```

```
        # if (set(real_unic_fields) & unique) - optional == unique -
optional:
        if (set(real_unic_fields) - optional == unique - optional
                and all(
                    user_data.get(unic_optional_key) is None or
unic_optional_key in real_unic_fields
                    for unic_optional_key in optional & unique)
        ):
            created_entities.append(res)
            assert new_user.status_code == 201
            good_response_validate(new_user)
        else:
            assert new_user.status_code == 456

    print(f"OK, tests passed {tests_count}")
    return created_entities


@pytest.mark.asyncio(scope="session")
async def test_user_exists(client, users_auth):
    tests_count = 0
    async with get_db_session_cm() as db_session:
        from the_remember.src.api.users import UserORM
        users = await db_session.execute(
            select(UserORM.id).limit(10)
        )
    user_uuids = [(str(i[0]), True) for i in list(users)] + [
        (str(uuid.uuid1()), False),
        (str(uuid.uuid3(uuid.uuid4(), name='Jon')), False),
        (str(uuid.uuid4()), False),
        (str(uuid.uuid5(uuid.uuid4(), name='Jon')), False),
    ]
    print(users)
    for (user_uuid, is_correct) in user_uuids:
        tests_count += 1
        response = await client.get("/user/exist", params={"user_id":
user_uuid})
        resp_data = response.text
        print("resp_data", resp_data)
        if (is_correct):
            assert response.status_code == 200
            assert resp_data == 'true'
        else:
            assert response.status_code == 200
            assert resp_data == 'false'
    print(f"OK, tests passed {tests_count}")


@pytest.mark.asyncio(scope="session")
async def test_user_user_me(client, users_auth: list[tuple[dict, dict]]):
    tests_count = 0
    for user_auth in users_auth[:10]:
        response = await client.get("/user/me", headers=user_auth[0])
        resp_data = response.json()
        tests_count += 1
        # print(response.json())
        assert response.status_code == 200
        good_response_validate(response)
        assert resp_data['data'].get('email')

    response = await client.get("/user/me")
    resp_data = response.json()
    tests_count += 1
    assert response.status_code == 401
```

```
    print(f"OK, tests passed {tests_count}")


@pytest.mark.asyncio(scope="session")
async def test_user_get_user(client, users_auth: list[tuple[dict, dict]]):
    tests_count = 0
    for (user_auth, _user) in users_auth[:10]:
        for (_, user) in users_auth[:10]:
            print(user)
            response = await client.get(f"/user/{user['id']}",
headers=user_auth)
            resp_data = response.json()
            tests_count += 1
            assert response.status_code == 200
            good_response_validate(response)
            assert 'email' not in resp_data['data']

        response = await client.get(f"/user/{_user['id']}")
        resp_data = response.json()
        tests_count += 1
        assert response.status_code == 401
    print(f"OK, tests passed {tests_count}")


@pytest.mark.asyncio(scope="session")
async def test_user_update_user(client, users_auth: list[tuple[dict, dict]]):
    tests_count = 0

    update_user = {
        "id": str(uuid.uuid4()),  # 'ec4ad7f9-4fe7-4ea7-a287-22650ebd6847' or
        "username": "test_update_user_1345672" + str(uuid.uuid4()),
        "email": "test_update_user_1345672" + str(uuid.uuid4()) +
"@example.com",
        "name": "test_update_user",
        "surname": "test_update_user",
        "password": "test_update_user"
    }

    response = await client.post("/user/create", json=update_user)
    resp_data = response.json()
    assert response.status_code == 201
    good_response_validate(response)
    assert all([v == update_user[k]
                for k, v in resp_data['data'].items() if k in update_user])
    created_at = resp_data['data']['created_at']
    updated_at = string_to_datetime(resp_data['data']['updated_at'])
    response = await client.post(
        "/auth/token",
        data={"username": update_user['username'], "password":
update_user['password']},
        headers={"Content-Type": "application/x-www-form-urlencoded"}
    )
    assert response.status_code == 200
    user_auth_headers = {"Authorization": f"Bearer
{response.json()['access_token']}"}

    update_params = {
        "name": lambda: f"updated_name_{datetime.utcnow().strftime('%Y-%m-
%dT%H:%M:%S.%f')}",
        "surname": lambda: f"updated_surname_{datetime.utcnow().strftime('%Y-
%m-%dT%H:%M:%S.%f')}",
        "updated_at": lambda: datetime.utcnow().strftime('%Y-%m-
%dT%H:%M:%S.%f'),
    }
```

```
    optional = {"name", "surname"}
    required = set(update_params) - optional
    unique = {}

    # validate input data testing
    for (user_auth, _user) in [(user_auth_headers, update_user |
resp_data['data'])]:
        last_user = None
        for user_data in
func_calling(optionals_as_null(dict_cutter(update_params), optional)):
            print(user_data)

            tests_count += 1
            response = await client.put("/user/me/update", json=user_data,
headers=user_auth)
            updated_user = response.json()
            last_user = user_data
            print(updated_user)
            if updated_user.get("data") and set(updated_user.get("data")) &
required == required:

                good_response_validate(response)
                new_updated_at =
string_to_datetime(updated_user['data']['updated_at'])
                assert response.status_code == 200
                assert updated_user['data']['created_at'] == created_at
                assert new_updated_at > updated_at  # Обновление 100%
произошло
                # Дата обновления была выставлена сервером самостоятельно, а
не скопирована из запроса
                assert new_updated_at >
string_to_datetime(user_data['updated_at'])
                # сервер действительно обновил данные
                assert all([v is None or (v == (updated_user['data'][k]))
                            for k, v in user_data.items()
                            if k != 'updated_at'])

                updated_at = new_updated_at
            else:
                assert response.status_code == 422
        #
        response = await client.put("/user/me/update", json=last_user)
        resp_data = response.json()
        tests_count += 1
        assert response.status_code == 401
    print(f"OK, tests passed {tests_count}")
```

ПРИЛОЖЕНИЕ Л
(РЕКОМЕНДУЕМОЕ)
КОД СЕРВЕРНОЙ ЧАСТИ

Листинг Л.1 – Файл swagger.py

```
import datetime
import inspect
import typing
from typing import Type, Annotated, Callable

from fastapi import Form, Query, FastAPI
from fastapi.openapi.utils import get_openapi
from pydantic import BaseModel
from pydantic.fields import FieldInfo
from sqlalchemy import Select, ColumnElement
from sqlalchemy.orm import ColumnProperty, Mapped

from the_remember.src.utils.db import AbstractDbEntity


def as_form(cls: Type[BaseModel]):
    new_parameters = []

    for field_name, model_field in cls.model_fields.items():
        model_field: FieldInfo
        # model_field.o

        new_parameters.append(
            inspect.Parameter(
                model_field.alias or field_name,
                inspect.Parameter.POSITIONAL_ONLY,
                default=Form(...) if model_field.is_required() else
Form(model_field.default),
                annotation=model_field.annotation,
            )
        )

    async def as_form_func(**data):
        # print("!!!", data)
        return cls(**data)

    sig = inspect.signature(as_form_func)
    sig = sig.replace(parameters=new_parameters)
    as_form_func.__signature__ = sig  # type: ignore
    setattr(cls, 'as_form', as_form_func)
    return cls


def limits_dep(
        db_model: typing.Type[AbstractDbEntity],
        default_limit: int = 100,
        default_offset: int = 0,
        max_limit: int = 100,
        min_limit: int = 1,
        change_created_at: Mapped[datetime.datetime] | None = None,
        change_updated_at: Mapped[datetime.datetime] | None = None,
        updated_at_condition: Callable[[datetime.datetime],
ColumnElement[bool]] = None,
):
    def dependency(
            limit: Annotated[int, Query(
                ge=min_limit,
                le=max_limit,
```

```
            )] = default_limit,
            offset: Annotated[int, Query(
                ge=0,
            )] = default_offset,
            updated_after: Annotated[datetime.datetime | None, Query(
                title='Время последнего обновления сущности',
                description="Будут возвращены лишь те сущности, "
                            "обновление которых было произведено позже
указанного времени. "
                            "Если параметр не указан, то фильтрация по
времени производиться не будет."

            )] = None
    ) -> Callable[[Select, ColumnProperty], Select]:
        def add_limit_and_offset(query: Select) -> Select:
            nonlocal updated_after
            if updated_after is not None:
                updated_after = updated_after.replace(tzinfo=None)
                if updated_at_condition is None:
                    if change_updated_at is None:
                        query = query.where(db_model.get_updated_at_field() >
updated_after)
                    else:
                        query = query.where(change_updated_at >
updated_after)
                else:
                    query = query.where(updated_at_condition(updated_after))
            if change_created_at is None:
                query = query.order_by(db_model.get_created_at_field())
            else:
                query = query.order_by(change_created_at)
            query = (
                query
                .limit(limit)
                .offset(offset)
            )
            return query

        return add_limit_and_offset

    return dependency


def two_ent_upd_condition(
        ent1: typing.Type[AbstractDbEntity],
        ent2: typing.Type[AbstractDbEntity]
) -> Callable[[datetime.datetime], ColumnElement[bool]]:
    def get_condition(updated_after: datetime.datetime):
        return (
                (ent1.get_updated_at_field() > updated_after)
                | (ent2.get_updated_at_field() > updated_after)
        )

    return get_condition


def fix_swagger_version(app: FastAPI):
    def wrapped():
        if app.openapi_schema:
            return app.openapi_schema

        openapi_schema = get_openapi(
            title="Custom title",
            version='3.1.0',  # or CONFIG.SWAGGER_VERSION,
```

```
            summary="This is a very custom OpenAPI schema",
            description="Here's a longer description of the custom
**OpenAPI** schema",
            routes=app.routes,
        )
        # openapi_schema["info"]["x-logo"] = {
        #     "url": "https://fastapi.tiangolo.com/img/logo-margin/logo-
teal.png"
        # }
        app.openapi_schema = openapi_schema
        return app.openapi_schema

    return wrapped
```

Листинг Л.2 – redefinition.py

```
import functools
import sys
from inspect import getmembers, signature
from pprint import pprint
from typing import Callable, Type
from uuid import UUID

from pydantic import BaseModel
from sqlalchemy.ext.asyncio import AsyncSession

from the_remember.src.config.external_api_config import UserAPIConfig,
FolderAPIConfig, TermAPIConfig, ModuleAPIConfig
from the_remember.src.utils.get_data_from_auth_token import
get_db_session_cm, TempUser, get_current_user, \
    get_current_user_no_coro


def redefinition(import_path: str, import_obj: str):
    try:
        target_func = getattr(__import__(import_path, fromlist=[import_obj]),
import_obj)

        def w_(_: Callable):
            @functools.wraps(target_func)
            def dec(*args, **kwargs):
                res = target_func(*args, **kwargs)
                return res

            return dec

        return w_
    except (ModuleNotFoundError,) as e:
        def w_(func: Callable):
            @functools.wraps(func)
            def dec(*args, **kwargs):
                res = func(*args, **kwargs)
                return res

            return dec

        return w_


def redefinition_client_api(
        import_path: str,
        import_obj: str,
```

```
        import_path_http_client: str,
        import_obj_http_client: str,
        # api_config_class: Type[BaseModel]
):
    try:
        target_func = getattr(
            __import__(import_path, fromlist=[import_obj]),
            import_obj
        )

        def w_(_: Callable):
            return target_func

        return w_
    except (ModuleNotFoundError,) as e:
        try:
            target_func = getattr(
                __import__(import_path_http_client,
fromlist=[import_obj_http_client]),
                import_obj_http_client
            )

            def w_(_: Callable):
                return target_func

            return w_
        except (ModuleNotFoundError,) as e2:
            try:
                raise e2 from e
            except Exception as e3:
                raise ModuleNotFoundError("Добавьте external_openapi для http
клиента "
                                          "при помощи client-api-
generator.docker-compose.yml") from e3


def redefinition_client_api_2(
        api_config_class: Type[BaseModel],
        service_name: str = None,
        api_func_name: str = None,
        external_file_name: str = None,
        external_func_name: str = None,
):
    try:
        def w_(target_func: Callable):

            res_func_is_final = True

            if (
                    service_name is None
                    or api_func_name is None
                    or external_file_name is None
                    or external_func_name is None
            ):
                res_func = target_func()
            else:
                curr_microservice: str =
target_func.__module__.removeprefix("the_remember.src.api.").split(".")[0]
                try:
                    res_func = getattr(

__import__(f'the_remember.src.api.{service_name}.src.view',
fromlist=[api_func_name]),
                        api_func_name
```

```
                    )
                    # from the_remember.src.api.folders.src.view import
exist_folder
                except AttributeError as exc:
                    print(f"Не удалось импортировать {api_func_name} из "
                          f"{f'the_remember.src.api.{service_name}.src.view'}
по причине {exc} "
                          f"Вызвовшая функция: {target_func.__code__}")
                    if "partially initialized module" not in str(exc):
                        raise ModuleNotFoundError() from exc

                    @functools.cache
                    def res_func_gen(*args, **kwargs):
                        res_func_ = getattr(

__import__(f'the_remember.src.api.{service_name}.src.view',
fromlist=[api_func_name]),
                                api_func_name
                            )
                        return res_func_

                    res_func = res_func_gen
                    res_func_is_final = False

                except (ModuleNotFoundError, ImportError) as exc:
                    print(f"Не удалось импортировать {api_func_name} из "
                          f"{f'the_remember.src.api.{service_name}.src.view'}
по причине {exc}")
                    try:
                        res_func = getattr(

__import__(f'the_remember.src.api.{curr_microservice}'
                                           f'.external_openapi.{service_name}'
                                           f'.services.{external_file_name}',
                                           fromlist=[external_func_name]),
                                external_func_name
                            )
                    except ModuleNotFoundError as exc:
                        print(f"Не удалось импортировать {external_func_name}
из "

f"{f'the_remember.src.api.{curr_microservice}.external_openapi.{service_name}
.services.{external_file_name}'} по причине {exc}")

                        @functools.cache
                        def res_func_gen(*args, **kwargs):
                            res_func_ = getattr(

__import__(f'the_remember.src.api.{service_name}.src.view',
fromlist=[api_func_name]),
                                    api_func_name
                                )
                            return res_func_

                        res_func = res_func_gen
                        res_func_is_final = False

            transform_rules = {
                lambda i: isinstance(i, UUID): lambda i: str(i)
            }

            def transform_rules_func(i):
                return functools.reduce(
                    lambda acc, curr: (curr[1](i) if curr[0](i) else i),
```

```
                    transform_rules.items(),
                    i
                )

            @functools.wraps(res_func)
            async def dec(*args, **kwargs):
                nonlocal res_func
                assert bool(args) is False, "keyword arguments only!"
                old_kwargs = kwargs
                if res_func_is_final is False:
                    res_func = res_func_gen()
                kwargs = {k: transform_rules_func(v) for k, v in
kwargs.items() if
                          k in signature(res_func).parameters}
                if (
                        'current_user' in signature(res_func).parameters
                        and 'current_user' not in kwargs
                        and (
                        bool(curr_token := ((old_kwargs.get('token', '') or
'').strip()))
                        or (
                                'api_config_override' in old_kwargs
                                and bool(curr_token :=
((old_kwargs['api_config_override'].access_token or '').strip()))
                        ))
                ):
                    kwargs |= {"current_user":
get_current_user_no_coro(curr_token)}
                res = await res_func(**kwargs)
                if isinstance(res, BaseModel) and hasattr(res, "data"):
                    res = res.data
                return res

            return dec

        return w_
    except (ModuleNotFoundError,) as e:
        raise ModuleNotFoundError("Добавьте external_openapi для http клиента
"
                                  "при помощи client-api-generator.docker-
compose.yml") \
            from e


def redefinition_client_api_3(
        *,
        service_name: str,
        api_func_name: str,
):
    service_config = {
        "users": {
            "external_file_name": 'async_Usersentities_service',
            "apis": {
                "exist_user": 'exist_user_exist_get',
                "check_user": 'check_user_check_user_get',
            },
            "api_config_class": UserAPIConfig,
        },
        "folders": {
            "external_file_name": 'async_Foldersentities_service',
            "apis": {
                "exist_folder": 'exist_folder_exist_get',
                "exist_personalize_folder":
"exist_personalize_folder_personalize_exist_get"
```

```
            },
            "api_config_class": FolderAPIConfig,
        },
        "modules": {
            "external_file_name": 'async_Moduleentities_service',
            "apis": {
                "exist_module": 'exist_module_exist_get',
                "exist_personalize_module":
"exist_personalize_module_personalize_exist_get",
                "create_personalize_module":
"create_personalize_module_personalize_create_post",
                "get_module_ids_from_folder":
"get_module_ids_from_folder_from_folder__folder_id__ids_only_get"
            },
            "api_config_class": ModuleAPIConfig,
        },
        "terms": {
            "external_file_name": 'async_Termentities_service',
            "apis": {
                "exist_term": 'exist_term_exist_get',
                "exist_personalize_term":
"exist_personalize_term_personalize_exist_get",
                "get_term_ids_from_module":
"get_term_ids_from_module_from_module__module_id__ids_only_get",
                "create_personalize_term":
"create_personalize_term_personalize_create_post"
            },
            "api_config_class": TermAPIConfig,
        }
    }
    try:
        return redefinition_client_api_2(
            service_config[service_name]['api_config_class'],
            service_name=service_name,
            api_func_name=api_func_name,

external_file_name=service_config[service_name]['external_file_name'],

external_func_name=service_config[service_name]['apis'][api_func_name],
        )
    except KeyError as exc:
        print("Запустите openapi generator!!! "
              "Продолжаем работать с пустышкой "
              "дабы openapi generator смог сгенерировать external api\n",
              exc,

              file=sys.stderr)

        def wap(func):
            async def dec(*args, **kwargs):
                raise NotImplementedError()

            return dec

        return wap
```

Листинг Л.3 – INIT_APP.py

```
import os
from os.path import split, join, isfile
from datetime import datetime
import time
```

```
from pprint import pprint

from fastapi import FastAPI
from fastapi.middleware.cors import CORSMiddleware
from fastapi.openapi.utils import get_openapi
import orjson

from the_remember.src.config.config import CONFIG
from the_remember.src.api import apps as api_apps
from the_remember.src.utils.exeptions.handlers import add_exception_handlers

start_time = time.time()


def add_swagger(app: FastAPI):
    def custom_openapi():
        global start_time
        if app.openapi_schema:
            return app.openapi_schema

        filename = join(split(split(split(split(__file__)[0])[0])[0])[0],
"merge-openapi", "output.swagger.json")
        mtime = isfile(filename) and os.path.getmtime(filename)
        if mtime is False or (CONFIG.FROM_DOCKER is True and mtime <
start_time):
            openapi_schema = None
        else:
            with open(filename, "r") as f:
                data = f.read()
            openapi_schema = orjson.loads(data)
            openapi_schema["openapi"] = CONFIG.SWAGGER_VERSION
            openapi_schema["servers"] = []
        app.openapi_schema = openapi_schema
        return app.openapi_schema

    app.openapi = custom_openapi

    return app


def init_app(app: FastAPI):
    app, other_apps = init_routers(app)
    for app_ in [app, *other_apps]:
        add_middlewares(app_)
        add_exception_handlers(app_)

    app = add_once_events(app)
    app = add_swagger(app)
    return app


def init_routers(app: FastAPI):
    filename = join(split(split(split(split(__file__)[0])[0])[0])[0], "merge-
openapi", "openapi-merge.json")
    openapi_merge = {"inputs": [], "output": "./output.swagger.json"}

    def _get(tested: str, target_sub: str):
        index_ = 0
        for index, [a, b] in enumerate(zip(tested.strip("/").split("/"),
target_sub.strip("/").split('/'))):
            if (a != b):
                return 0
            index_ = index
        return (index_ + 1 if len(tested) < len(target_sub) else 0)
```

```
    for sub_apps_url, config in api_apps.items():
        _app = app
        app_prefix = ""
        if sub_apps_url.count("/") > 1:
            # print(sub_apps_url, sub_apps_url.count("/"))
            target_api = max(api_apps, key=lambda i: _get(i, sub_apps_url))
            if _get(target_api, sub_apps_url) != 0:
                _app = api_apps[target_api]["app"]
                app_prefix = target_api
            # else
        _app.mount(
            config.get('prefix', "").removeprefix(app_prefix),
            config["app"],
            name=config.get('tags', [None])[0]
        )
        openapi_merge["inputs"].append(
            {
                "inputURL":
f"http://backend:{CONFIG.BACKEND_PORT}{config.get('prefix',
'not_found')}/openapi.json",
                "pathModification": {
                    "prepend": config.get('prefix', 'not_found')
                }
            })

    res = orjson.dumps(openapi_merge).decode("utf-8")
    with open(filename, "w") as f:
        print(res, file=f)

    return app, [app_["app"] for _, app_ in api_apps.items()]


def add_middlewares(app: FastAPI):
    origins = [
        "*",
    ]

    app.add_middleware(
        CORSMiddleware,
        allow_origins=origins,
        allow_credentials=True,
        allow_methods=["*"],
        allow_headers=["*"],
    )
    return app


def add_once_events(app: FastAPI):
    async def db_dispose():
        await CONFIG.engine.dispose()

    app.add_event_handler('shutdown', db_dispose)

    return app
```

Листинг Л.4 – get_data_from_auth_token.py

```
import asyncio
import types
from contextlib import asynccontextmanager
from inspect import signature, Parameter
from typing import Annotated, Awaitable, Any, Type, TypeVar, Callable, Unpack
```

```
from uuid import UUID

from fastapi import Depends, HTTPException
from jose import jwt, JWTError
from sqlalchemy.ext.asyncio import AsyncSession, async_sessionmaker
from starlette import status

from the_remember.src.api.auth.src.dto import TokenData
from the_remember.src.config.config import CONFIG
from pydantic import BaseModel

from the_remember.src.config.external_api_config import UserAPIConfig,
APIConfig, FolderAPIConfig


class TempUser(BaseModel):
    id: UUID
    username: str


async def get_db_session():
    async_session = async_sessionmaker(CONFIG.engine, expire_on_commit=False)
    async with async_session() as session:
        if True:
            try:
                yield session
            finally:
                pass


get_db_session_cm = asynccontextmanager(get_db_session)


async def get_current_user(
        token: Annotated[str, Depends(CONFIG.oauth2_scheme)],
        db_session: Annotated[AsyncSession, Depends(get_db_session)] = None
) -> TempUser:
    user = get_current_user_no_coro(token, db_session)
    return user


class CheckDbEntitiesDep(object):
    def __init__(self, config: Type[APIConfig], *test_methods):
        self.config = config
        self.test_methods = list(test_methods)

    def __call__(self, *arg, **kwargs):
        return all([i(*arg, config=self.config, **kwargs) is True for i in
self.test_methods])


class CheckDbEntitiesExistDep(CheckDbEntitiesDep):

    @staticmethod
    async def check(*coros: Awaitable[bool]):
        res = await asyncio.gather(*coros)
        print(res)
        if all(res):
            return True
        raise HTTPException(status_code=409)


LocDataType = TypeVar('LocDataType', bound=BaseModel)
```

```
class CheckDbEntitiesExistDep2(CheckDbEntitiesExistDep):

    async def __call__(self, result: Any, *, db_session, token, **kwargs):
        print(kwargs)
        api_conf = self.config(access_token=token)
        res = await self.check(
            *[
                asyncio.create_task(
                    i(db_session=db_session, api_config_override=api_conf,
**kwargs)
                )
                for i in self.test_methods
            ]
        )
        return result

    # __call__ = modif_call

    def __new__(cls, *args, **kwargs):
        inst = super().__new__(cls)
        return inst

    def __class_getitem__(
            cls,
            params: tuple[
                Type[Any],
                Callable[[BaseModel, Unpack[dict[str, Any]]], dict[str,
Any]],
                Callable[[BaseModel, Unpack[dict[str, Any]]], bool],
                dict[str, Annotated],
            ],
            *args,
            **kwargs
    ):
        item, get_args_from_item, skip, other_params = params

        async def __call_wrap__(
                self,
                return_value: Annotated[item, ...],
                db_session: Annotated[AsyncSession, Depends(get_db_session)],
                token: Annotated[str, Depends(CONFIG.oauth2_scheme)],
                **kwargs

        ) -> item:
            print(kwargs)
            kwargs_ = get_args_from_item(return_value, **kwargs)
            if skip(return_value, **kwargs):
                return True
            return await cls.__call__(self, return_value,
db_session=db_session, token=token, **kwargs_)

        ann = signature(__call_wrap__)
        ann._parameters = {k: v for k, v in ann.parameters.items() if k !=
'kwargs'} | {
            param_name: Parameter(
                name=param_name,
                kind=Parameter.POSITIONAL_OR_KEYWORD,
                annotation=annotation
            )
            for param_name, annotation in other_params.items()
        }
        __call_wrap__.__signature__ = ann
```

```
        dict(signature(__call_wrap__).parameters)

        signature(__call_wrap__)
        instance = cls
        module = None
        class_body = {
            "__module__": __name__ if module is None else module,
            "__call__": __call_wrap__,
        }
        instance2 = types.new_class(
            f"{cls.__name__}_{item.__class__}",
            (instance,),
            dict(),
            lambda ns: ns.update(class_body)
        )

        return instance2


class CheckDbUserDep(CheckDbEntitiesExistDep):
    def __init__(self, *test_methods):
        super().__init__(UserAPIConfig, *test_methods)

    async def __call__(self,
                       current_user: Annotated[TempUser,
Depends(get_current_user)],
                       db_session: Annotated[AsyncSession,
Depends(get_db_session)],
                       token: Annotated[str, Depends(CONFIG.oauth2_scheme)],

                       ) -> bool:
        api_conf = self.config(access_token=token)

        res = await self.check(*[
            asyncio.create_task(
                i(user_id=current_user.id, db_session=db_session,
api_config_override=api_conf)
            )
            for i in self.test_methods
        ])

        return res


class CheckDbFolderDep(CheckDbEntitiesExistDep):
    def __init__(self, *test_methods):
        super().__init__(FolderAPIConfig, *test_methods)

    async def __call__(self,
                       folder_id: UUID,
                       db_session: Annotated[AsyncSession,
Depends(get_db_session)],
                       token: Annotated[str, Depends(CONFIG.oauth2_scheme)],

                       ) -> bool:
        api_conf = self.config(access_token=token)

        res = await self.check(*[
            asyncio.create_task(
                i(current_user.id, db_session, api_config_override=api_conf)
            )
            for i in self.test_methods
        ])
```

```
        return res


async def get_db_write_session(
        db_session: Annotated[AsyncSession, Depends(get_db_session)]
):
    async with db_session.begin():
        yield db_session
```

Листинг Л.5 – get_from_html.py

```
import re
import urllib.request
from datetime import datetime
from time import sleep
from urllib.error import URLError
from uuid import UUID

from sqlalchemy.ext.asyncio import async_sessionmaker

from the_remember.src.api.folders.src.dto import CreateFolderAsTreeDTO
from the_remember.src.api.folders.src.view import create_folder_as_tree
from the_remember.src.api.modules.src.dto import CreateModuleAsTreeDTO
from the_remember.src.api.sentences.src.dto import CreateSentenceAsTreeDTO
from the_remember.src.api.terms.dtos.dto import CreateTermAsTreeDTO
from the_remember.src.api.users.src.dto import UserDTO
from the_remember.src.config.config import CONFIG
from the_remember.src.utils._help_file import r_d, bad_terms, add_words


async def get_3000_words():
    r_ =
r"<ol[^<]*(?:<li>[^<]*(?:<a(?:(?<=<a)([^>]*)>[^<]*(?=<\/a>))<\/a>[^@]*?)[^<]*
<\/li>[^<]*)*?(?=<\/ol>)<\/ol>"
    main_folder = CreateFolderAsTreeDTO(
        name='3000_words',
        sub_folders=[],
    )

    fp = urllib.request.urlopen("https://langformula.ru/voc3000/")
    mybytes = fp.read()
    mystr = mybytes.decode("utf8")
    fp.close()
    # print(mystr)
    for i in re.findall(

r"(?:<\/span>[^<]*<\/div>[^#]*?(?=<ol>))<ol>[^#]*?(?=<\/ol>)<\/ol>",
            mystr
    ):

        folder_title = re.search(r'<\/span>([^<]*)<\/div>', i).groups()[0]
        cur_folder = CreateFolderAsTreeDTO(
            name=' '.join(folder_title.split()),
            sub_folders=[],
            sub_modules=[]
        )
        main_folder.sub_folders += [cur_folder]
        print(folder_title)
        for html_link_tag in re.findall(

r"<li>[^<]*(?:<a(?:(?<=<a)([^>]*)>([^<]*)(?=<\/a>))<\/a>[^@]*?)[^<]*<\/li>[^<
]*", i):
```

```
            link = re.search(r'''href=(?:"|'|`)([^"'`]*)(?:"|'|`)''',
html_link_tag[0]).groups()[0]
            print(link, html_link_tag[1])
            module_name = html_link_tag[1]
            curr_module = CreateModuleAsTreeDTO(
                name=' '.join(module_name.split()),
                description=f'autogenerated from {link}',
                sub_terms=get_words_from_words_page(link)
            )
            cur_folder.sub_modules += [curr_module]
    print(main_folder.model_dump_json())
    return main_folder


def get_words_from_words_page(url: str):
    _r =
r'(?:<\/i>[^<]*?<strong[^>]*>(?:(?:<a[^>]*>([^<]*)<\/a>([^<]*)(?=</strong>)</
strong>([^—<&#8212--]*)(?:—|&#8212;|&#8212|-|-))|(?:([^<]*)</strong>([^—
<&#8212--]*)(?:—|&#8212;|&#8212|-|-)))([^<]*)<\/li>)'
    fp = None
    while fp is None:
        try:
            fp = urllib.request.urlopen(url)
        except URLError as e:
            print(e)
            sleep(1)
    mybytes = fp.read()
    mystr = mybytes.decode("utf8")
    fp.close()
    table_datas = re.findall(r'<table([^@]*?)<\/table>', mystr)
    print(table_datas)
    terms_entities: dict[str, CreateTermAsTreeDTO] = dict()
    for table_data in table_datas:
        for word in re.finditer(

r'<tr[^@]*?(?:<td[^>]*>([^<]*)<\/td>[^<]*)(?:<td[^>]*>([^<]*)<\/td>[^<]*)(?:<
td[^>]*>([^<]*)<\/td>[^<]*)<\/tr>',
                table_data):
            term, transcribe, definition = (word.groups())
            terms_entities[' '.join(term.strip().split()).lower()] =
CreateTermAsTreeDTO(
                term=' '.join(term.strip().split()).lower(),
                definition=' '.join(definition.strip().split()).lower(),
                transcription=transcribe,
            )
            print(term, transcribe, definition)
    try:
        for sentences_html_piece in  re.findall(
            r'((?:(?:<div[^>]*?class=\"su-
list\"[^>]*[^<]*)(?:\s*<div>\s*)?(?:<ul[^@]*?(?=<\/ul>)<\/ul>))[^<]*<blockquo
te[^@]*?(?=<\/blockquote>)<\/blockquote>[^@]*?</div>)',
            mystr
        ):
            print("sentences_html_piece", sentences_html_piece)

            print(sentences_html_piece)
            for sent_html in re.findall(

r'(?:<ul[^@]*?(?=<\/ul>)<\/ul>)[^<]*<blockquote[^@]*?<\/blockquote>',
sentences_html_piece):
                print('-----')
                print(sent_html)
                try:
                    gr = re.search(
```

```
r"(?:<\/i>[^<]*?(?:<a[^>]*>[^<]*)?<strong[^>]*>(?:(?:<a[^>]*>([^<]*)<\/a>([^<
]*)(?=</strong>)</strong>(?:[^<]*</a>)?([^—<&#8212--]*)(?:—|&#8212;|&#8212|-
|-))|(?:([^<]*)</strong>(?:[^<]*</a>)?(?:[^<]*<strong>)?([^—<&#8212--
]*)(?:(?:</strong>\s*)?(?:—|&#8212;|&#8212|-|-
))))([^<]*)<\/li>)[^@]*?(?=<p)(<p[^<]*<\/p>[^@]*?)(?=<\/blockquote>)",
                        sent_html
                    ).groups()
                    main_term2, sub_terms2, _, main_term1, sub_terms1,
definition, sentences_in_term = gr
                    main_term = main_term1 or main_term2
                    sub_terms = sub_terms1 or sub_terms2

                except AttributeError as e:
                    print(e)
                    is_ok = False
                    is_continue = False

                    while is_ok is False:
                        if sent_html in r_d:
                            main_term, sub_terms, definition,
sentences_in_term = r_d[sent_html]
                            break
                        if is_continue:
                            break

                        print('Введите main_term')
                        main_term = input()
                        print('Введите sub_terms')
                        sub_terms = input()
                        print('Введите definition')
                        definition = input()
                        print('Введите sentences_in_term')
                        sentences_in_term = input()

                        is_ok = input("Введите 1 для подтверждения") == '1'

                    if is_continue:
                        continue

                print(main_term, sub_terms, definition, sentences_in_term)

                term = ((' '.join(
                    (main_term.strip() + (
                        " " + sub_terms.strip() if sub_terms and
bool(sub_terms.strip()) else "")).split())
                         .lower())
                        .replace("( ", "(")
                        .replace(" )", ")"))
                if term in terms_entities:
                    terms_entities[term].sub_sentences = []
                elif term in bad_terms:
                    new_term = bad_terms[term]
                    terms_entities[new_term].sub_sentences = []
                    terms_entities[term] = terms_entities[new_term]
                else:
                    add_words(term, terms_entities)

                    print(terms_entities)
                    print(f'term {term} not in in terms_entities')
                    old_term = term

                    while term not in terms_entities:
                        print(f'term {term} not in in terms_entities')
```

```
                        term = input()

                    terms_entities[term].sub_sentences = []
                    terms_entities[old_term] = terms_entities[term]
                for sentence_ent in re.findall(
                        r'(?=<p)(?:<p[^>]*>([^&#8212\-\-\-]*).([^<]*))<\/p>',
                        sentences_in_term
                ):
                    print(sentence_ent)
                    sentence, translate = sentence_ent
                    print(sentence, translate)
                    terms_entities[term].sub_sentences += [
                        CreateSentenceAsTreeDTO(
                            sentence=' '.join(sentence.split()),
                            translate=' '.join(translate.split())
                        )
                    ]
    except AttributeError as e:
        print(e)
        # sentences_html_piece = input("Введите sentences_html_piece")
    return list(terms_entities.values())
```

Листинг Л.6 – saga/controller.py

```
import asyncio
import random
from abc import ABC, abstractmethod
from typing import Awaitable, Any


class SagaController(ABC):
    objects_and_works: dict[Any, Any]

    steps: list

    def __init__(self):
        self.fix_saga_transaction_in_db_coro: Awaitable | None = None
        SagaController.objects_and_works[self] = self
        self.run = None
        self.cancel = False

    async def run(self, *args, **kwargs):
        self.run = asyncio.current_task()
        self.fix_saga_transaction_in_db_coro = asyncio.create_task(
            self._fix_saga_transaction_in_db()
        )
        await self.fix_saga_transaction_in_db_coro

        try:

            for step in self.__class__.steps:
                if self.cancel:
                    break
                (args, kwargs) = await asyncio.create_task(step(*args,
**kwargs))

        except Exception as e:
            # TODO: сделать отображение ошибок куда-нибудь
            while True:
                try:
                    await self._saga_rollback()
                    break
```

```
                except Exception as e:
                    # TODO: сделать отображение ошибок куда-нибудь
                    await asyncio.sleep(random.randint(1, 10))

        await self._finish_saga_tr()

        SagaController.objects_and_works.pop(self)

    async def _fix_saga_transaction_in_db(self):
        """
        Положить транзакцию в БД
        """
        await asyncio.sleep(1)
        raise NotImplementedError()

    @abstractmethod
    async def _saga_transaction(self):
        raise NotImplementedError()

    @abstractmethod
    async def _saga_rollback(self):
        raise NotImplementedError()

    async def _finish_saga_tr(self):
        """
        Отправить в кафку сообщение об успешном завершении транзакции,
ужалить транзакцию из БД
        """
        # TODO: Implement
        await asyncio.sleep(1)

    async def kafka_send_rollback_flag(self):
        """
        Отправить в кафку сообщение об ошибке в транзакции.
        При получении этого сообщения все запущенные работы
        в рамках этой транзакции завершаются
        """
        # TODO: Implement
        await asyncio.sleep(1)
```

Листинг Л.7 – Файл answers/post_db.py

```
import typing
from abc import ABC
from datetime import datetime
from typing import Any

from pydantic import BaseModel
from sqlalchemy import inspect
# from pydantic.main import Model
from sqlalchemy.orm import InstrumentedAttribute, Relationship,
ColumnProperty

from the_remember.src.utils.db import AbstractDbEntity

from the_remember.src.api import api_collector

Model = typing.TypeVar('Model', bound='BaseModel')

orm_to_dto_mask: dict[
    typing.Type[AbstractDbEntity],
    dict[typing.Type[ColumnProperty] | typing.Type[Relationship], bool]] = {
    i: {k: True for k in inspect(i).column_attrs}
```

```
       | {k: False for k in inspect(i).relationships}
    for i in api_collector.__db_models__
}


class OrmBaseModel(BaseModel, ABC, extra='ignore', from_attributes=True):

    @classmethod
    def model_validate(cls: type[Model],
                       obj: Any,
                       *,
                       strict: bool | None = None,
                       from_attributes: bool | None = None,
                       context: dict[str, Any] | None = None,
                       ) -> Model:

        if bool(orm_to_dto_mask) is False:
            orm_to_dto_mask.update({
                i: {k: True for k in inspect(i).column_attrs}
                   | {k: False for k in inspect(i).relationships}
                for i in api_collector.__db_models__
            })
        if isinstance(obj, AbstractDbEntity):
            obj: AbstractDbEntity
            return cls(**{k.key: getattr(obj, k.key) for k, v in
orm_to_dto_mask[obj.__class__].items() if v})

        return super().model_validate(obj, strict=strict,
from_attributes=from_attributes, context=context)

    def model_dump(self, *args, **kwargs):
        res = super().model_dump(*args, **kwargs)
        res = {k: (v.replace(tzinfo=None) if isinstance(v, datetime) else v)
for k,v in res.items()}
        return res
```

Листинг Л.8 – Файл api/folders/view.py

```
import asyncio
import datetime
from typing import Annotated, Any, Literal, Callable
from uuid import UUID

import sqlalchemy
from fastapi import APIRouter, FastAPI
from fastapi._compat import ErrorWrapper
from fastapi.encoders import jsonable_encoder
from fastapi.exceptions import RequestValidationError, HTTPException
from fastapi.params import Depends, Query, Body, Path
# from pydantic.v1.error_wrappers import ErrorWrapper
from sqlalchemy import insert, select, delete, update, bindparam, join,
Select
from sqlalchemy.ext.asyncio import AsyncSession
from sqlalchemy.dialects.postgresql import insert as pg_insert
from sqlalchemy.orm import ColumnProperty

from the_remember.src.api.folders._redefinitions import get_current_user,
UserDTO, check_user, \
    kafka_create_personalize_folder_producer, create_personal_module,
get_module_ids_from_folder, \
    CreateOnlyPersonalizePartModuleDTO
```

```
# from the_remember.src.api.auth.logics import get_db_session,
get_current_user, get_db_write_session
from the_remember.src.api.folders.src.db_model import FolderORM,
PersonalizeFolderORM
from the_remember.src.api.folders.src.dto import CreateFolderDTO, FolderDTO,
FolderWithNestedEntitiesDTO, \
    FolderWithRootEntityDTO, PersonalizeFolderDTO,
OnlyPersonalizePartFolderDTO, \
    UpdateOnlyPersonalizePartFolderDTO, DeleteOnlyPersonalizePartFolderDTO,
CreateOnlyPersonalizePartFolderDTO, \
    UpdateFolderDTO
from the_remember.src.config.config import CONFIG
from the_remember.src.config.external_api_config import UserAPIConfig,
ModuleAPIConfig, FolderAPIConfig
from the_remember.src.utils.answers.base import Ans, BError, AnsInfo,
ErrOnly, AuthError, NotFoundError
from the_remember.src.utils.answers.enums import ErrorMessages
from the_remember.src.utils.exeptions.handlers import base_resp_models
# from the_remember.src.api.folders.logic import recourse_tree_to_db_models
from the_remember.src.utils.get_data_from_auth_token import
get_db_write_session, get_db_session, CheckDbEntitiesDep, \
    CheckDbUserDep
from the_remember.src.utils.saga.controller import SagaController
from the_remember.src.utils.swagger import limits_dep, two_ent_upd_condition,
fix_swagger_version

# from the_remember.src.api.users.dto import UserDTO

openapi_tags = [
    {
        "name": 'Folders entities',
        "description": "",
        "externalDocs": {
            "description": "current microservice docs",
            "url": f"{FolderAPIConfig.base_path}/docs",
        },
    },
]
folder_app_ = FastAPI(
    openapi_tags=openapi_tags,
    servers=[
        {"url": CONFIG.BACKEND_MICROSERVICE_URL_AUTH, "description": "Staging
environment"},
    ],
)
folder_app = APIRouter(tags=list(map(lambda i: i['name'], openapi_tags)))


# api_collector.apps.append(folder_app)


@folder_app.post(
    "/create",
    response_model=Ans[FolderDTO, BError | None],
    status_code=201,
    responses={**base_resp_models(401, 403, 404, 409, 422, 424, 456, 500)}
)
async def create_folder(
        new_folder: CreateFolderDTO,
        db_session: Annotated[AsyncSession, Depends(get_db_write_session)],
        current_user: Annotated[UserDTO, Depends(get_current_user)],
        checked: Annotated[bool, Depends(CheckDbUserDep(check_user))],
):
    # ErrorWrapper
```

```
    inserted_data: dict[str, Any] = jsonable_encoder(
        jsonable_encoder(
            new_folder, exclude_none=True
        ) | {"author_id": current_user.id})

    sub_query = select(*[bindparam(k).label(k) for k, v in
inserted_data.items()])

    if new_folder.root_folder_id is not None:
        # sub_query = sub_query.subquery()
        # j = join(
        #     FolderORM, sub_query,
        #     (FolderORM.id == bindparam('root_folder_id'))
        #     & (FolderORM.author_id == bindparam('author_id'))
        # )
        #
        # sub_query = select(sub_query).select_from(j)

        sub_query = (
            select(*[bindparam(k).label(k) for k, v in
inserted_data.items()])
            .where(
                (FolderORM.id == bindparam('root_folder_id'))
                & (FolderORM.author_id == bindparam('author_id'))
            )
        )

    sub_query = sub_query.params({k: v for k, v in inserted_data.items()})
    query = (
        insert(FolderORM)
        .from_select([*inserted_data.keys()], sub_query)
        .returning(FolderORM)
    )

    res = await db_session.execute(query)
    data_ = list(res)
    if bool(data_) is False:
        raise HTTPException(404, detail={
            "message": ErrorMessages.root_folder_not_found,
            "description": "Некорректный root_folder (вы не можете создать
дочернюю папку, "
                           "не являясь владельцем родительской, "
                           "или указанной корневой папки просто не
существует)"
        })
    return FolderDTO.model_validate(data_[0][0])


@folder_app.get(
    "/personalize/all",
    response_model=Ans[list[PersonalizeFolderDTO], BError | None],
    status_code=200,
    responses={**base_resp_models(401, 409, 422, 456, 500)}
)
async def get_all_folders(
        db_limits: Annotated[Callable[[Select], Select], Depends(limits_dep(
            PersonalizeFolderORM,
            updated_at_condition=two_ent_upd_condition(PersonalizeFolderORM,
FolderORM)
        ))],
        db_session: Annotated[AsyncSession, Depends(get_db_session)],
        current_user: Annotated[UserDTO, Depends(get_current_user)],
        # checked: Annotated[bool, Depends(CheckDbUserDep(check_user))],
        author_only: bool = False,
```

```
):
    query = (select(PersonalizeFolderORM)
             .join(PersonalizeFolderORM.folder_entity)
             .add_columns(FolderORM)
             .where(PersonalizeFolderORM.user_id == current_user.id))
    if author_only:
        query = query.where(FolderORM.author_id == current_user.id)
    res = await db_session.execute(
        db_limits(query)
    )

    data = list(res)
    return [PersonalizeFolderDTO.model_validate(i[0].folder_entity.__dict__ |
i[0].__dict__)
            for i in data]

@folder_app.put(
    "/update/{folder_id}",
    response_model=Ans[FolderDTO, BError | None],
    status_code=200,
    responses={**base_resp_models(401, 404, 409, 422, 456, 500)}
)
async def update_folder_entity(
        folder_id: UUID,
        update_folder: UpdateFolderDTO,
        db_session: Annotated[AsyncSession, Depends(get_db_write_session)],
        current_user: Annotated[UserDTO, Depends(get_current_user)],
        checked: Annotated[bool, Depends(CheckDbUserDep(check_user))],
):
    query = (
        update(FolderORM).
        where(
            (FolderORM.id == folder_id)
            & (FolderORM.author_id == current_user.id)
        )
        .returning(FolderORM)
        .values(**update_folder.model_dump())
    )

    res = list(await db_session.execute(query))
    data = list(res)
    if bool(data) is False:
        raise HTTPException(404, detail=dict())
    return FolderDTO.model_validate(data[0][0])


@folder_app.get(
    "/{folder_id}",
    response_model=Ans[FolderDTO, BError | None],
    status_code=200,
    responses={**base_resp_models(401, 404, 409, 422, 456, 500)}
)
async def get_one_folder(
        folder_id: UUID,
        db_session: Annotated[AsyncSession, Depends(get_db_session)],
        current_user: Annotated[UserDTO, Depends(get_current_user)],
):
    res = await db_session.execute(
        select(FolderORM)
        .where(
            # (FolderORM.author_id == current_user.id)
            # &
            (FolderORM.id == folder_id)
        )
```

```
    )

    data = list(res)
    if bool(data) is False:
        raise HTTPException(404, detail=dict())
    res2 = data[0][0]
    return FolderDTO.model_validate(res2)


@folder_app.get(
    "/{folder_id}/sub_folders",
    response_model=Ans[list[FolderDTO], BError | None],
    status_code=200,
    responses={**base_resp_models(401, 404, 409, 422, 456, 500)}
)
async def get_sub_folders(
        folder_id: UUID,
        db_limits: Annotated[Callable[[Select], Select],
Depends(limits_dep(FolderORM))],
        db_session: Annotated[AsyncSession, Depends(get_db_session)],
        current_user: Annotated[UserDTO, Depends(get_current_user)],
        # checked: Annotated[bool, Depends(CheckDbUserDep(check_user))],
):
    res = await db_session.execute(db_limits(
        select(FolderORM)
        .where(
            # (FolderORM.author_id == current_user.id)
            # &
            (FolderORM.root_folder_id == folder_id)
        )
    )
    )

    data = list(res)
    return [FolderDTO.model_validate(i[0]) for i in data]


@folder_app.get(
    "/personalize/{folder_id}",
    response_model=Ans[PersonalizeFolderDTO, BError | None],
    status_code=200,
    responses={**base_resp_models(401, 404, 409, 422, 456, 500)}
)
async def get_personalize_folder(
        folder_id: UUID,
        db_session: Annotated[AsyncSession, Depends(get_db_session)],
        current_user: Annotated[UserDTO, Depends(get_current_user)],
        author_only: bool = False,
        # checked: Annotated[bool, Depends(CheckDbUserDep(check_user))],
):
    query = (
        select(PersonalizeFolderORM)
        .join(PersonalizeFolderORM.folder_entity)
        .add_columns(FolderORM)
        .where(
            (PersonalizeFolderORM.user_id == current_user.id)
            & (FolderORM.id == folder_id)
        )
    )

    if author_only:
        query = query.where(FolderORM.author_id == current_user.id)

    res = await db_session.execute(
```

```
        query
    )

    data = list(res)
    if bool(data) is False:
        raise HTTPException(404, detail=dict())
    res2 = data[0][0]
    return PersonalizeFolderDTO.model_validate(res2.folder_entity.__dict__ |
res2.__dict__)


@folder_app.get(
    "/personalize/{folder_id}/sub_folders",
    response_model=Ans[list[PersonalizeFolderDTO], BError | None],
    status_code=200,
    responses={**base_resp_models(401, 404, 409, 422, 456, 500)}
)
async def get_personalize_sub_folders(
        folder_id: Annotated[
            UUID | Literal["null"] | None, Path(description="Установите null
для получения корневых папок")],
        db_limits: Annotated[Callable[[Select], Select], Depends(limits_dep(
            PersonalizeFolderORM,
            updated_at_condition=two_ent_upd_condition(PersonalizeFolderORM,
FolderORM)
        ))],
        db_session: Annotated[AsyncSession, Depends(get_db_session)],
        current_user: Annotated[UserDTO, Depends(get_current_user)],
        author_only: bool = False,
        # checked: Annotated[bool, Depends(CheckDbUserDep(check_user))],
):
    folder_id = (None if folder_id is None or str(folder_id).lower() ==
'null' else folder_id)
    query = (
        select(PersonalizeFolderORM)
        .join(PersonalizeFolderORM.folder_entity)
        .add_columns(FolderORM)
        .where(
            (PersonalizeFolderORM.user_id == current_user.id)
            & (FolderORM.root_folder_id == folder_id)
        )
    )

    if author_only:
        query = query.where(FolderORM.author_id == current_user.id)

    res = await db_session.execute(
        db_limits(query)
    )

    data = list(res)
    return [PersonalizeFolderDTO.model_validate(i[0].folder_entity.__dict__ |
i[0].__dict__)
            for i in data]


@folder_app.post(
    "/personalize/create",
    response_model=Ans[list[OnlyPersonalizePartFolderDTO], BError | None],
    status_code=201,
    responses={**base_resp_models(401, 403, 404, 409, 422, 424, 456, 500)}
)
async def create_personalize_folder(
        create_personal_folder: CreateOnlyPersonalizePartFolderDTO,
```

```
        db_session: Annotated[AsyncSession, Depends(get_db_write_session)],
        current_user: Annotated[UserDTO, Depends(get_current_user)],
        checked: Annotated[bool, Depends(CheckDbUserDep(check_user))],
        token: Annotated[str, Depends(CONFIG.oauth2_scheme)],
):

    target_folders_tree = sqlalchemy.text(  # language=PostgreSQL
        """
        ( WITH RECURSIVE folder_tree (id, root_folder_id) AS (
            SELECT :folder_id_ as id, :root_folder_id_ ::uuid as
root_folder_id
            UNION ALL
            SELECT f.id   as id, f.root_folder_id as root_folder_id
            FROM folder_tree as rec
            join folder_schema.folder f ON f.root_folder_id = rec.id
)
select * from folder_tree ) AS _rf
        """
    )

    inserted_data: dict[str, Any] = jsonable_encoder(
        jsonable_encoder(
            create_personal_folder,
            exclude={'folder_id', 'root_folder_id'}
        ) | {"user_id": current_user.id})
    res = await db_session.execute(

        insert(PersonalizeFolderORM)
        .from_select(
            ['folder_id', 'root_folder_id', *inserted_data.keys()],
            select(
                sqlalchemy.text("_rf.id"),
                sqlalchemy.text("_rf.root_folder_id"),
                *[
                    (
                        bindparam(k).label(k)
                    ) for k, v in inserted_data.items()
                ]
            ).params({
                k: v for k, v in inserted_data.items()
            }).select_from(target_folders_tree.bindparams(
                **{
                    "folder_id_": create_personal_folder.folder_id,
                    "root_folder_id_": create_personal_folder.root_folder_id,
                }))
        )
        .returning(PersonalizeFolderORM)
        # ,
        # []
    )
    data = list(res)
    if bool(data) is False:
        raise HTTPException(404, detail=dict())

    folder_to_modules = {pers_folder[0].folder_id: (await
get_module_ids_from_folder(
        folder_id=str(pers_folder[0].folder_id),
        db_session=db_session,
        api_config_override=ModuleAPIConfig(access_token=token),
        token=token,
        current_user=current_user,

    )) for pers_folder in data}
```

```
    res = [await create_personal_module(
        from_folder_microservice=True,
        db_session=db_session,
        current_user=current_user,
        checked=True,

new_personal_module=CreateOnlyPersonalizePartModuleDTO(module_id=module_id,
root_folder_id=folder_id),
        data=CreateOnlyPersonalizePartModuleDTO(module_id=str(module_id),
root_folder_id=str(folder_id)),
        api_config_override=ModuleAPIConfig(access_token=token),
        token=token
    )
           for folder_id, module_ids in folder_to_modules.items()
           for module_id in module_ids
           ]

    return [OnlyPersonalizePartFolderDTO.model_validate(i[0]) for i in data]


@folder_app.put(
    "/personalize/update/{folder_id}",
    response_model=Ans[OnlyPersonalizePartFolderDTO, BError | None],
    status_code=200,
    responses={**base_resp_models(401, 404, 409, 422, 456, 500)}
)
async def update_personalize_folder(
        folder_id: UUID,
        update_folder: UpdateOnlyPersonalizePartFolderDTO,
        db_session: Annotated[AsyncSession, Depends(get_db_write_session)],
        current_user: Annotated[UserDTO, Depends(get_current_user)],
        checked: Annotated[bool, Depends(CheckDbUserDep(check_user))],
):
    query = (
        update(PersonalizeFolderORM).
        where(
            (PersonalizeFolderORM.folder_id == folder_id)
            & (PersonalizeFolderORM.user_id == current_user.id)
        )
        .returning(PersonalizeFolderORM)
        .values(**update_folder.model_dump())
    )

    res = list(await db_session.execute(query))
    data = list(res)
    if bool(data) is False:
        raise HTTPException(404, detail=dict())
    return OnlyPersonalizePartFolderDTO.model_validate(data[0][0])


@folder_app.get(
    "/exist/{folder_id}",
    response_model=bool,
    status_code=200,
    responses={**base_resp_models(401, 404, 409, 422, 456, 500)}
)
async def exist_folder(
        # checked: Annotated[bool, Depends(CheckDbUserDep(check_user))],
        folder_id: Annotated[UUID, Path()],
        db_session: Annotated[AsyncSession, Depends(get_db_session)],
        root_folder_id: Annotated[UUID | None, Query()] = None,
        author_id: Annotated[UUID | None, Query()] = None,

):
```

```
    q = (
        select(FolderORM)
        .where(
            (FolderORM.id == folder_id)

        )
    )
    if (root_folder_id is not None):
        q = (q
             .where(FolderORM.root_folder_id == root_folder_id))
    if (author_id is not None):
        q = (q
             .where(FolderORM.author_id == author_id))
    res = await db_session.execute(
        select(q.exists())
    )
    data = list(res)
    if bool(data) is False:
        raise HTTPException(404, detail=dict())

    return data[0][0]


@folder_app.get(
    "/personalize/exist/{folder_id}",
    response_model=bool,
    status_code=200,
    responses={**base_resp_models(401, 404, 409, 422, 456, 500)}
)
async def exist_personalize_folder(
        folder_id: UUID,
        current_user: Annotated[UserDTO, Depends(get_current_user)],
        db_session: Annotated[AsyncSession, Depends(get_db_session)],
        root_folder_id: UUID | None = None,
):

    q = (
        select(PersonalizeFolderORM)
        .where(
            (PersonalizeFolderORM.folder_id == folder_id)
            & (PersonalizeFolderORM.user_id == current_user.id)
        )
    )
    if (root_folder_id is not None):
        q = (q.where(PersonalizeFolderORM.root_folder_id == root_folder_id))

    res = await db_session.execute(
        select(q.exists())
    )
    data = list(res)
    if bool(data) is False:
        raise HTTPException(404, detail=dict())

    return data[0][0]


folder_app_.include_router(folder_app)
folder_app = folder_app_
folder_app.openapi = fix_swagger_version(folder_app)
```

Листинг Л.9 – Файл api/modules/view.py

```
import asyncio
import datetime
from typing import Annotated, Literal, Callable
from uuid import UUID

from fastapi import APIRouter, FastAPI, HTTPException
from fastapi.params import Depends, Path, Query
from sqlalchemy import insert, select, delete, update, Select
from sqlalchemy.ext.asyncio import AsyncSession
from sqlalchemy.dialects.postgresql import insert as pg_insert
from sqlalchemy.orm import ColumnProperty

from the_remember.src.api.modules._redefinitions import UserDTO,
get_current_user, check_user, check_folder, \
    check_personalize_folder, get_term_ids_from_folder,
create_personalize_term, CreateOnlyPersonalizePartTermDTO
# from the_remember.src.api.auth.logics import get_db_session,
get_current_user, get_db_write_session
from the_remember.src.api.modules.src.db_model import ModuleORM,
PersonalizeModuleORM
from the_remember.src.api.modules.src.dto import ModuleDTO, CreateModuleDTO,
PersonalizeModuleDTO, \
    OnlyPersonalizePartModuleDTO, UpdateOnlyPersonalizePartModuleDTO,
DeleteOnlyPersonalizePartModuleDTO, \
    CreateOnlyPersonalizePartModuleDTO, UpdateModuleDTO
from the_remember.src.config.config import CONFIG
from the_remember.src.config.external_api_config import FolderAPIConfig,
TermAPIConfig, ModuleAPIConfig
from the_remember.src.utils.answers.base import Ans, BError
from the_remember.src.utils.exeptions.handlers import base_resp_models
# from the_remember.src.api.users.dto import UserDTO
from the_remember.src.utils.get_data_from_auth_token import
get_db_write_session, get_db_session, CheckDbUserDep, \
    CheckDbEntitiesExistDep2
from the_remember.src.utils.swagger import limits_dep, two_ent_upd_condition,
fix_swagger_version

openapi_tags = [
    {
        "name": 'Module entities',
        "description": "",
        "externalDocs": {
            "description": "current microservice docs",
            "url": f"{ModuleAPIConfig.base_path}/docs",
        },
    },
]
module_app_ = FastAPI(
    openapi_tags=openapi_tags,
    servers=[
        {"url": CONFIG.BACKEND_MICROSERVICE_URL_AUTH, "description": "Staging
environment"},
    ],
)
module_app = APIRouter(tags=list(map(lambda i: i['name'], openapi_tags)))


from_folder_microservice_annotation = Annotated[bool, Query(
    default_factory=lambda: False, deprecated=True,
    description="Костыль, при переписывании взаимодействия с использованием
Саги будет удален")]


@module_app.post(
```

```
    "/create",
    response_model=Ans[ModuleDTO, BError | None],
    status_code=201,
    responses={**base_resp_models(401, 403, 404, 409, 422, 424, 456, 500)}
)
async def create_module(
        db_session: Annotated[AsyncSession, Depends(get_db_write_session)],
        current_user: Annotated[UserDTO, Depends(get_current_user)],
        checked: Annotated[bool, Depends(CheckDbUserDep(check_user))],
        new_module: Annotated[
            CreateModuleDTO,
            Depends(
                CheckDbEntitiesExistDep2[
                    CreateModuleDTO,
                    lambda i, **k: {"folder_id": str(i.root_folder_id),
"author_id": str(k["current_user"].id)},
                    lambda i, **k: i.root_folder_id is None,
                    {"current_user": Annotated[UserDTO,
Depends(get_current_user)]}
                ](FolderAPIConfig, check_folder, )
            )
        ]
):
    res = await db_session.execute(
        insert(ModuleORM)
        .returning(ModuleORM),
        [
            new_module.model_dump(exclude_none=True) | {'author_id':
current_user.id}
        ]
    )
    data = list(res)
    if bool(data) is False:
        raise HTTPException(404, detail=dict())

    return ModuleDTO.model_validate(data[0][0])


@module_app.get(
    "/{module_id}",
    response_model=Ans[ModuleDTO, BError | None],
    status_code=200,
    responses={**base_resp_models(401, 404, 409, 422, 456, 500)},
)
async def get_one_module(
        module_id: UUID,
        db_session: Annotated[AsyncSession, Depends(get_db_session)],
        current_user: Annotated[UserDTO, Depends(get_current_user)]
):
    res = await db_session.execute(
        select(ModuleORM)
        .where(
            (PersonalizeModuleORM.module_id == module_id)
        )

    )

    data = list(res)
    if bool(data) is False:
        raise HTTPException(404, detail=dict())
    return
PersonalizeModuleDTO.model_validate(data[0][0].module_entity.__dict__ |
data[0][0].__dict__)
```

```
@module_app.get(
    "/from_folder/{folder_id}",
    response_model=Ans[list[UUID], BError | None],
    status_code=200,
    responses={**base_resp_models(401, 404, 409, 422, 456, 500)},
)
async def get_modules_from_folder(
        folder_id: UUID,
        db_limits: Annotated[Callable[[Select], Select],
Depends(limits_dep(ModuleORM))],
        db_session: Annotated[AsyncSession, Depends(get_db_session)],
        current_user: Annotated[UserDTO, Depends(get_current_user)]
) -> list[UUID]:

    res = await db_session.execute(db_limits(
        select(ModuleORM)
        .where(ModuleORM.root_folder_id == folder_id)
    )
    )
    data = list(res)

    if data and bool(data):
        return [i[0] for i in data]
    return []


@module_app.get(
    "/from_folder/{folder_id}/ids_only",
    response_model=Ans[list[UUID], BError | None],
    status_code=200,
    responses={**base_resp_models(401, 404, 409, 422, 456, 500)},
)
async def get_module_ids_from_folder(
        folder_id: UUID,
        db_session: Annotated[AsyncSession, Depends(get_db_session)],
        current_user: Annotated[UserDTO, Depends(get_current_user)]
) -> list[UUID]:

    res = await db_session.execute(
        select(ModuleORM.id)
        .where(ModuleORM.root_folder_id == folder_id)
    )
    data = list(res)

    if data and bool(data):
        return [i[0] for i in data]
    return []


@module_app.put(
    "/update/{module_id}",
    response_model=Ans[ModuleDTO, BError | None],
    status_code=200,
    responses={**base_resp_models(401, 403, 404, 409, 422, 424, 456, 500)},

)
async def update_module_entity(
        module_id: UUID,
        update_module: UpdateModuleDTO,
        db_session: Annotated[AsyncSession, Depends(get_db_write_session)],
        current_user: Annotated[UserDTO, Depends(get_current_user)],
        # checked: Annotated[bool, Depends(CheckDbUserDep(check_user))],
):
```

```
    query = (
        update(ModuleORM).
        where(
            (ModuleORM.id == module_id)
            & (ModuleORM.author_id == current_user.id)
        )
        .returning(ModuleORM)
        .values(**update_module.model_dump()))

    data = list(await db_session.execute(query))
    if bool(data) is False:
        raise HTTPException(404, detail=dict())
    return ModuleDTO.model_validate(data[0][0])



@module_app.get(
    "/personalize/all",
    response_model=Ans[list[PersonalizeModuleDTO], BError | None],
    status_code=200,
    responses={**base_resp_models(401, 409, 422, 456, 500)}
)
async def get_all_module(
        db_limits: Annotated[Callable[[Select], Select], Depends(limits_dep(
            PersonalizeModuleORM,
            updated_at_condition=two_ent_upd_condition(PersonalizeModuleORM,
ModuleORM)
        ))],
        db_session: Annotated[AsyncSession, Depends(get_db_session)],
        current_user: Annotated[UserDTO, Depends(get_current_user)],
        author_only: bool = False,
):
    query = (
        select(PersonalizeModuleORM)
        .join(PersonalizeModuleORM.module_entity)
        .add_columns(ModuleORM)
        .where(PersonalizeModuleORM.user_id == current_user.id)
    )
    if author_only:
        query = query.where(ModuleORM.author_id == current_user.id)
    res = await db_session.execute(
        db_limits(query)
    )

    data = list(res)
    return [PersonalizeModuleDTO.model_validate(i[0].module_entity.__dict__ |
i[0].__dict__)
            for i in data]


@module_app.get(
    "/personalize/{module_id}",
    response_model=Ans[PersonalizeModuleDTO, BError | None],
    status_code=200,
    responses={**base_resp_models(401, 404, 409, 422, 456, 500)},
)
async def get_one_module(
        module_id: UUID,
        db_session: Annotated[AsyncSession, Depends(get_db_session)],
        current_user: Annotated[UserDTO, Depends(get_current_user)]
):
    res = await db_session.execute(
```

```
        select(PersonalizeModuleORM)
        .join(PersonalizeModuleORM.module_entity)
        .add_columns(ModuleORM)
        .where((PersonalizeModuleORM.user_id == current_user.id)
               & (PersonalizeModuleORM.module_id == module_id))

    )

    data = list(res)
    if bool(data) is False:
        raise HTTPException(404, detail=dict())
    return
PersonalizeModuleDTO.model_validate(data[0][0].module_entity.__dict__ |
data[0][0].__dict__)


@module_app.get(
    "/personalize/from_folder/{folder_id}",
    response_model=Ans[list[PersonalizeModuleDTO], BError | None],
    status_code=200,
    responses={**base_resp_models(401, 409, 422, 456, 500)}
)
async def get_all_module(
        folder_id: Annotated[
            UUID | Literal["null"] | None, Path(description="Установите null
для получения корневых модулей")],
        db_limits: Annotated[Callable[[Select], Select], Depends(limits_dep(
            PersonalizeModuleORM,
            updated_at_condition=two_ent_upd_condition(PersonalizeModuleORM,
ModuleORM)
        ))],
        db_session: Annotated[AsyncSession, Depends(get_db_session)],
        current_user: Annotated[UserDTO, Depends(get_current_user)],
        author_only: bool = False,
):
    folder_id = (None if folder_id is None or str(folder_id).lower() ==
'null' else folder_id)
    query = (
        select(PersonalizeModuleORM)
        .join(PersonalizeModuleORM.module_entity)
        .add_columns(ModuleORM)
        .where(
            (PersonalizeModuleORM.user_id == current_user.id)
            & (PersonalizeModuleORM.root_folder_id == folder_id)
        )
    )

    if author_only:
        query = query.where(ModuleORM.author_id == current_user.id)
    res = await db_session.execute(
        db_limits(query)
    )

    data = list(res)
    return [PersonalizeModuleDTO.model_validate(i[0].module_entity.__dict__ |
i[0].__dict__)
            for i in data]


@module_app.post(
    "/personalize/create",
    response_model=Ans[OnlyPersonalizePartModuleDTO, BError | None],
    status_code=201,
    responses={**base_resp_models(401, 403, 404, 409, 422, 424, 456, 500)},
```

```
)
async def create_personalize_module(
        # create_module: CreateOnlyPersonalizePartModuleDTO,
        db_session: Annotated[AsyncSession, Depends(get_db_write_session)],
        current_user: Annotated[UserDTO, Depends(get_current_user)],
        from_folder_microservice: from_folder_microservice_annotation,
        checked: Annotated[bool, Depends(CheckDbUserDep(check_user))],
        token: Annotated[str, Depends(CONFIG.oauth2_scheme)],
        new_personal_module: Annotated[
            CreateOnlyPersonalizePartModuleDTO,
            Depends(
                CheckDbEntitiesExistDep2[
                    CreateOnlyPersonalizePartModuleDTO,
                    lambda i, **k: {"folder_id": i.root_folder_id,
"current_user": k["current_user"]},
                    lambda i, **k: i.root_folder_id is None or
k["from_folder_microservice"] is True,
                    {"current_user": Annotated[UserDTO,
Depends(get_current_user)],
                     "from_folder_microservice":
from_folder_microservice_annotation}
                ](FolderAPIConfig, check_personalize_folder, )
            )
        ],
        # new_personal_module: CreateOnlyPersonalizePartModuleDTO
):

    res = await db_session.execute(
        insert(PersonalizeModuleORM)
        .returning(PersonalizeModuleORM),
        [new_personal_module.model_dump() | {'user_id': current_user.id}]
    )
    data = list(res)
    if bool(data) is False:
        raise HTTPException(404, detail=dict())

    terms = {
        i[0].module_id: await asyncio.create_task(get_term_ids_from_folder(
            db_session=db_session,
            module_id=i[0].module_id,
            api_config_override=TermAPIConfig(access_token=token),
            current_user=current_user,
            token=token,
        ))
        for i in data
    }

    terms_res = [
        await create_personalize_term(
            from_module_microservice=True,
            db_session=db_session,
            current_user=current_user,

create_personal_term=CreateOnlyPersonalizePartTermDTO(term_id=term_id),
            data=CreateOnlyPersonalizePartTermDTO(term_id=str(term_id)),
            api_config_override=TermAPIConfig(access_token=token),
            token=token,
        )
        for module_id, term_ids in terms.items()
        for term_id in term_ids
    ]

    return OnlyPersonalizePartModuleDTO.model_validate(data[0][0])
```

```
@module_app.put(
    "/personalize/update/{module_id}",
    response_model=Ans[OnlyPersonalizePartModuleDTO, BError | None],
    status_code=200,
    responses={**base_resp_models(401, 403, 404, 409, 422, 424, 456, 500)},

)
async def update_personalize_module(
        module_id: UUID,
        update_personal_module: UpdateOnlyPersonalizePartModuleDTO,
        db_session: Annotated[AsyncSession, Depends(get_db_write_session)],
        current_user: Annotated[UserDTO, Depends(get_current_user)],
        # checked: Annotated[bool, Depends(CheckDbUserDep(check_user))],
):

    query = (
        update(PersonalizeModuleORM).
        where(
            (PersonalizeModuleORM.module_id == module_id)
            & (PersonalizeModuleORM.user_id == current_user.id)
        )
        .returning(PersonalizeModuleORM)
        .values(**update_personal_module.model_dump()))

    data = list(await db_session.execute(query))
    if bool(data) is False:
        raise HTTPException(404, detail=dict())
    return OnlyPersonalizePartModuleDTO.model_validate(data[0][0])


@module_app.get(
    "/exist/{module_id}",
    response_model=bool,
    status_code=200,
    responses={**base_resp_models(401, 404, 409, 422, 456, 500)}
)
async def exist_module(
        module_id: Annotated[UUID, Path()],
        db_session: Annotated[AsyncSession, Depends(get_db_session)],
        # current_user: Annotated[UserDTO, Depends(get_current_user)],
        root_folder_id: UUID | None = None,
        author_id: UUID | None = None,

):
    q = (
        select(ModuleORM)
        .where((ModuleORM.id == module_id))
    )
    if (root_folder_id is not None):
        q = (q.where(ModuleORM.root_folder_id == root_folder_id))
    if (author_id is not None):
        q = (q.where(ModuleORM.author_id == author_id))
    res = await db_session.execute(
        select(q.exists())
    )
    data = list(res)

    if bool(data) is False:
        raise HTTPException(404, detail=dict())

    return data[0][0]
```

```
@module_app.get(
    "/personalize/exist",
    response_model=bool,
    status_code=200,
    responses={**base_resp_models(401, 404, 409, 422, 456, 500)}
)
async def exist_personalize_module(
        module_id: UUID,
        db_session: Annotated[AsyncSession, Depends(get_db_session)],
        current_user: Annotated[UserDTO, Depends(get_current_user)],
        root_folder_id: UUID | None = None,

):

    q = (
        select(select(PersonalizeModuleORM)
        .where(
            (PersonalizeModuleORM.module_id == module_id)
            & (PersonalizeModuleORM.user_id == current_user.id)
        ))
    )
    if (module_id is not None):
        q = (q
             .where(PersonalizeModuleORM.root_folder_id == root_folder_id))

    res = await db_session.execute(
        select(q.exists())
    )
    data = list(res)
    if bool(data) is False:
        raise HTTPException(404, detail=dict())
    return data[0][0]


module_app_.include_router(module_app)
module_app = module_app_
module_app.openapi = fix_swagger_version(module_app)
```

Листинг Л.10 – Файл api/sentences/view.py

```
from typing import Annotated, Callable
from uuid import UUID

from fastapi import APIRouter, FastAPI, HTTPException
from fastapi.params import Depends, Path, Query
import sqlalchemy as sa
from sqlalchemy import insert, select, any_, Select, update
from sqlalchemy.ext.asyncio import AsyncSession
from sqlalchemy.orm import ColumnProperty

from the_remember.src.api.sentences.src.db_model import SentenceORM
from the_remember.src.api.sentences.src.dto import SentenceDTO,
CreateSentenceDTO, UpdateSentenceDTO

from the_remember.src.api.users.src.dto import UserDTO
from the_remember.src.config.config import CONFIG
from the_remember.src.config.external_api_config import TermAPIConfig,
SentenceAPIConfig
from the_remember.src.utils.answers.base import Ans, BError
from the_remember.src.utils.exeptions.handlers import base_resp_models
```

```
from the_remember.src.utils.get_data_from_auth_token import
get_db_write_session, get_db_session, CheckDbUserDep, \
    CheckDbEntitiesExistDep2
from the_remember.src.api.sentences._redefinitions import get_current_user,
check_user, check_term, check_module
from the_remember.src.utils.swagger import limits_dep, fix_swagger_version

openapi_tags = [
    {
        "name": 'Sentence entities',
        "description": "",
        "externalDocs": {
            "description": "current microservice docs",
            "url": f"{SentenceAPIConfig.base_path}/docs",
        },
    },
]
sentence_app_ = FastAPI(
    openapi_tags=openapi_tags,
    servers=[
        {"url": CONFIG.BACKEND_MICROSERVICE_URL_AUTH, "description": "Staging
environment"},
    ],
)
sentence_app = APIRouter(tags=list(map(lambda i: i['name'], openapi_tags)))


@sentence_app.post(
    "/create/{module_id}/{term_id}",
    response_model=Ans[SentenceDTO, BError | None],
    status_code=201,
    responses={**base_resp_models(401, 403, 404, 409, 422, 424, 456, 500)},
)
async def create_sentence(
        # new_sentence: CreateSentenceDTO,
        module_id: Annotated[UUID, Path()],
        term_id: Annotated[UUID, Path()],
        db_session: Annotated[AsyncSession, Depends(get_db_write_session)],
        current_user: Annotated[UserDTO, Depends(get_current_user)],
        checked: Annotated[bool, Depends(CheckDbUserDep(check_user))],
        new_sentence: Annotated[
            CreateSentenceDTO,
            Depends(
                CheckDbEntitiesExistDep2[
                    CreateSentenceDTO,
                    lambda i, **k: {
                        "term_id": k["term_id"],
                        "author_id": k["current_user"].id,
                        "module_id": k["module_id"]
                    },
                    lambda i, **k: False,
                    {"current_user": Annotated[UserDTO,
Depends(get_current_user)],
                     "module_id": Annotated[UUID, Path()],
                     "term_id": Annotated[UUID, Path()]}
                ](TermAPIConfig, check_term, check_module)
            )
        ]

):
    res = await db_session.execute(
        insert(SentenceORM)
        .values(
            module_id=module_id,
```

```
            term_id=term_id,
            author_id=current_user.id
        )
        .returning(SentenceORM),
        [new_sentence.model_dump(exclude_none=True)]
    )

    data = list(res)
    if bool(data) is False:
        raise HTTPException(404, detail=dict())
    return SentenceDTO.model_validate(data[0][0])


@sentence_app.get(
    "/all",
    response_model=Ans[list[SentenceDTO], BError | None],
    status_code=200,
    responses={**base_resp_models(401, 404, 409, 422, 456, 500)},
)
async def get_all_sentences(
        db_session: Annotated[AsyncSession, Depends(get_db_session)],
        current_user: Annotated[UserDTO, Depends(get_current_user)],
        module_ids: Annotated[
            frozenset[UUID], Query(description="Список модулей, из которых
будут получены предложения")],
        db_limits: Annotated[Callable[[Select], Select],
Depends(limits_dep(SentenceORM))],
        author_only: bool = False,
):
    query = (
        select(SentenceORM)
        .where(SentenceORM.module_id == any_(module_ids))
    )
    if author_only:
        query = query.where(SentenceORM.author_id == current_user.id)

    res = await db_session.execute(
        db_limits(query)
    )

    data = list(res)
    return [SentenceDTO.model_validate(i[0]) for i in data]


@sentence_app.get(
    "/from_module/{module_id}",
    response_model=Ans[list[SentenceDTO], BError | None],
    status_code=200,
    responses={**base_resp_models(401, 404, 409, 422, 456, 500)},
)
async def get_all_sentences_from_one_module(
        db_session: Annotated[AsyncSession, Depends(get_db_session)],
        current_user: Annotated[UserDTO, Depends(get_current_user)],
        module_id: UUID,
        db_limits: Annotated[Callable[[Select], Select],
Depends(limits_dep(SentenceORM))],
        author_only: bool = False,
):
    query = (
        select(SentenceORM)
        .where(SentenceORM.module_id == module_id)
    )
    if author_only:
        query = query.where(SentenceORM.author_id == current_user.id)
```

```
    res = await db_session.execute(
        db_limits(query)
    )

    data = list(res)
    return [SentenceDTO.model_validate(i[0]) for i in data]


@sentence_app.get(
    "/from_term/{term_id}",
    response_model=Ans[list[SentenceDTO], BError | None],
    status_code=200,
    responses={**base_resp_models(401, 404, 409, 422, 456, 500)},
)
async def get_all_sentences_from_one_term(
        db_session: Annotated[AsyncSession, Depends(get_db_session)],
        current_user: Annotated[UserDTO, Depends(get_current_user)],
        term_id: UUID,
        db_limits: Annotated[Callable[[Select], Select],
Depends(limits_dep(SentenceORM))],
        author_only: bool = False,
):
    query = (
        select(SentenceORM)
        .where(SentenceORM.term_id == term_id)
    )
    if author_only:
        query = query.where(SentenceORM.author_id == current_user.id)

    res = await db_session.execute(
        db_limits(query)
    )

    data = list(res)
    return [SentenceDTO.model_validate(i[0]) for i in data]


@sentence_app.put(
    "/update/{sentence_id}",
    response_model=Ans[SentenceDTO, BError | None],
    status_code=200,
    responses={**base_resp_models(401, 403, 404, 409, 422, 424, 456, 500)},
)
async def update_sentence_entity(
        sentence_id: UUID,
        update_sentence: UpdateSentenceDTO,
        db_session: Annotated[AsyncSession, Depends(get_db_write_session)],
        current_user: Annotated[UserDTO, Depends(get_current_user)],
        # checked: Annotated[bool, Depends(CheckDbUserDep(check_user))],
):

    query = (
        update(SentenceORM).
        where(
            (SentenceORM.id == sentence_id)
            & (SentenceORM.author_id == current_user.id)
        )
        .returning(SentenceORM)
        .values(**update_sentence.model_dump()))

    data = list(await db_session.execute(query))
    if bool(data) is False:
        raise HTTPException(404, detail=dict())
```

```
    return SentenceDTO.model_validate(data[0][0])



@sentence_app.get(
    "/{sentence_id}",
    response_model=Ans[SentenceDTO, BError | None],
    status_code=200,
    responses={**base_resp_models(401, 404, 409, 422, 456, 500)}
)
async def get_one_sentence(
        sentence_id: UUID,
        db_session: Annotated[AsyncSession, Depends(get_db_session)],
        current_user: Annotated[UserDTO, Depends(get_current_user)]
):
    # TODO: сделать проверку на доступ
    res = await db_session.execute(
        select(SentenceORM)
        .where((SentenceORM.id == sentence_id))
    )

    data = list(res)
    if bool(data) is False:
        raise HTTPException(404, detail=dict())
    return SentenceDTO.model_validate(data[0][0])


@sentence_app.get(
    "/exist",
    response_model=bool,
    status_code=200,
    responses={**base_resp_models(401, 404, 409, 422, 456, 500)}
)
async def exist_sentence(
        sentence_id: UUID,
        db_session: Annotated[AsyncSession, Depends(get_db_session)],
        # current_user: Annotated[UserDTO, Depends(get_current_user)],
        term_id: UUID | None = None,

):
    q = (
        select(SentenceORM)
        .where((SentenceORM.id == sentence_id))
    )
    if (term_id is not None):
        q = (q.where(SentenceORM.term_id == term_id))

    res = await db_session.execute(select(q.exists()))
    data = list(res)
    if bool(data) is False:
        raise HTTPException(404, detail=dict())
    return data[0][0]


sentence_app_.include_router(sentence_app)
sentence_app = sentence_app_
sentence_app.openapi = fix_swagger_version(sentence_app)
```

Листинг Л.11 – Файл api/term_info/view.py

```
from typing import Annotated, Callable
```

```
from uuid import UUID

from fastapi import APIRouter, FastAPI, HTTPException
from fastapi.params import Depends, Path, Query
import sqlalchemy as sa
from sqlalchemy import insert, select, any_, Select, update
from sqlalchemy.ext.asyncio import AsyncSession
from sqlalchemy.orm import ColumnProperty

from the_remember.src.api.term_info._redefinitions import UserDTO,
get_current_user, check_user, check_term, \
    check_module
from the_remember.src.api.term_info.src.db_model import AdditionalTermInfoORM
from the_remember.src.api.term_info.src.dto.base import
CreateAdditionalTermInfoDTO, AdditionalTermInfoDTO, \
    UpdateAdditionalTermInfoDTO
from the_remember.src.config.config import CONFIG
from the_remember.src.config.external_api_config import TermInfoAPIConfig,
TermAPIConfig
from the_remember.src.utils.answers.base import Ans, BError
from the_remember.src.utils.exeptions.handlers import base_resp_models
from the_remember.src.utils.get_data_from_auth_token import
get_db_write_session, get_db_session, CheckDbUserDep, \
    CheckDbEntitiesExistDep2
from the_remember.src.utils.swagger import limits_dep, fix_swagger_version

openapi_tags = [
    {
        "name": 'Additional info of terms',
        "description": "",
        "externalDocs": {
            "description": "current microservice docs",
            "url": f"{TermInfoAPIConfig.base_path}/docs",
        },
    },
]
add_term_info_app_ = FastAPI(
    openapi_tags=openapi_tags,
    servers=[
        {"url": CONFIG.BACKEND_MICROSERVICE_URL_AUTH, "description": "Staging
environment"},
    ],
)
add_term_info_app = APIRouter(tags=list(map(lambda i: i['name'],
openapi_tags)))


@add_term_info_app.post(
    "/create/{module_id}/{term_id}",
    response_model=Ans[AdditionalTermInfoDTO, BError | None],
    status_code=201,
    responses={**base_resp_models(401, 403, 404, 409, 422, 424, 456, 500)},
)
async def create_term(
        # new_sentence: CreateSentenceDTO,
        module_id: Annotated[UUID, Path()],
        term_id: Annotated[UUID, Path()],
        db_session: Annotated[AsyncSession, Depends(get_db_write_session)],
        current_user: Annotated[UserDTO, Depends(get_current_user)],
        checked: Annotated[bool, Depends(CheckDbUserDep(check_user))],
        new_sentence: Annotated[
            CreateAdditionalTermInfoDTO,
            Depends(
                CheckDbEntitiesExistDep2[
```

```
                    CreateAdditionalTermInfoDTO,
                    lambda i, **k: {
                        "term_id": k["term_id"],
                        "author_id": k["current_user"].id,
                        "module_id": k["module_id"]
                    },
                    lambda i, **k: False,
                    {"current_user": Annotated[UserDTO,
Depends(get_current_user)],
                     "module_id": Annotated[UUID, Path()],
                     "term_id": Annotated[UUID, Path()]}
                ](TermAPIConfig, check_term, check_module)
            )
        ]

):
    res = await db_session.execute(
        insert(AdditionalTermInfoORM)
        .values(
            module_id=module_id,
            term_id=term_id,
            author_id=current_user.id
        )
        .returning(AdditionalTermInfoORM),
        [new_sentence.model_dump(exclude_none=True)]
    )

    data = list(res)
    if bool(data) is False:
        raise HTTPException(404, detail=dict())
    return AdditionalTermInfoDTO.model_validate(data[0][0])


@add_term_info_app.get(
    "/all",
    response_model=Ans[list[AdditionalTermInfoDTO], BError | None],
    status_code=200,
    responses={**base_resp_models(401, 409, 422, 456, 500)},
)
async def get_all_add_term_info(
        db_session: Annotated[AsyncSession, Depends(get_db_session)],
        current_user: Annotated[UserDTO, Depends(get_current_user)],
        module_ids: Annotated[frozenset[UUID], Query(description="Список
модулей, из которых будут получены термины")],
        db_limits: Annotated[Callable[[Select], Select],
Depends(limits_dep(AdditionalTermInfoORM))],
        author_only: bool = False,
):
    query = (
        select(AdditionalTermInfoORM)
        .where(
            (AdditionalTermInfoORM.module_id == any_(module_ids))
        )
    )
    if author_only:
        query = query.where(AdditionalTermInfoORM.author_id ==
current_user.id)
    res = await db_session.execute(
        db_limits(query)
    )

    data = list(res)
    return [AdditionalTermInfoDTO.model_validate(i[0]) for i in data]
```

```
@add_term_info_app.get(
    "/from_module/{module_id}",
    response_model=Ans[list[AdditionalTermInfoDTO], BError | None],
    status_code=200,
    responses={**base_resp_models(401, 404, 409, 422, 456, 500)},
)
async def get_all_add_term_info_from_module(
        db_session: Annotated[AsyncSession, Depends(get_db_session)],
        current_user: Annotated[UserDTO, Depends(get_current_user)],
        module_id: UUID,
        db_limits: Annotated[Callable[[Select], Select],
Depends(limits_dep(AdditionalTermInfoORM))],
        author_only: bool = False,
):
    query = (
        select(AdditionalTermInfoORM)
        .where(
            (AdditionalTermInfoORM.module_id == module_id)
        )
    )
    if author_only:
        query = query.where(AdditionalTermInfoORM.author_id ==
current_user.id)
    res = await db_session.execute(
        db_limits(query)
    )

    data = list(res)
    return [AdditionalTermInfoDTO.model_validate(i[0]) for i in data]


@add_term_info_app.get(
    "/from_term/{term_id}",
    response_model=Ans[list[AdditionalTermInfoDTO], BError | None],
    status_code=200,
    responses={**base_resp_models(401, 404, 409, 422, 456, 500)},
)
async def get_all_add_term_info_from_module(
        db_session: Annotated[AsyncSession, Depends(get_db_session)],
        current_user: Annotated[UserDTO, Depends(get_current_user)],
        term_id: UUID,
        db_limits: Annotated[Callable[[Select], Select],
Depends(limits_dep(AdditionalTermInfoORM))],
        author_only: bool = False,
):
    query = (
        select(AdditionalTermInfoORM)
        .where(
            (AdditionalTermInfoORM.term_id == term_id)
        )
    )
    if author_only:
        query = query.where(AdditionalTermInfoORM.author_id ==
current_user.id)
    res = await db_session.execute(
        db_limits(query)
    )

    data = list(res)
    return [AdditionalTermInfoDTO.model_validate(i[0]) for i in data]
```

```
@add_term_info_app.put(
    "/update/{add_info_id}",
    response_model=Ans[AdditionalTermInfoDTO, BError | None],
    status_code=200,
    responses={**base_resp_models(401, 403, 404, 409, 422, 424, 456, 500)},

)
async def update_additional_info_entity(
        add_info_id: UUID,
        update_info: UpdateAdditionalTermInfoDTO,
        db_session: Annotated[AsyncSession, Depends(get_db_write_session)],
        current_user: Annotated[UserDTO, Depends(get_current_user)],
        # checked: Annotated[bool, Depends(CheckDbUserDep(check_user))],
):

    query = (
        update(AdditionalTermInfoORM).
        where(
            (AdditionalTermInfoORM.id == add_info_id)
            & (AdditionalTermInfoORM.author_id == current_user.id)
        )
        .returning(AdditionalTermInfoORM)
        .values(**update_info.model_dump()))

    data = list(await db_session.execute(query))
    if bool(data) is False:
        raise HTTPException(404, detail=dict())
    return AdditionalTermInfoDTO.model_validate(data[0][0])




@add_term_info_app.get(
    "/exist",
    response_model=bool,
    status_code=200,
    responses={**base_resp_models(401, 404, 409, 422, 456, 500)}
)
async def exist_term_info(
        add_term_id: UUID,
        db_session: Annotated[AsyncSession, Depends(get_db_session)],
        # current_user: Annotated[UserDTO, Depends(get_current_user)],
        term_id: UUID | None = None,

):
    q = (
        select(AdditionalTermInfoDTO)
        .where((AdditionalTermInfoORM.id == add_term_id))
    )
    if (term_id is not None):
        q = (q.where(AdditionalTermInfoORM.term_id == term_id))

    res = await db_session.execute(select(q.exists()))
    data = list(res)
    if bool(data) is False:
        raise HTTPException(404, detail=dict())
    return data[0][0]


add_term_info_app_.include_router(add_term_info_app)
add_term_info_app = add_term_info_app_
add_term_info_app.openapi = fix_swagger_version(add_term_info_app)
```

Листинг Л.12 – Файл api/term_marks/view.py

```
from typing import Annotated, Callable
from uuid import UUID

from fastapi import APIRouter, FastAPI, HTTPException
from fastapi.params import Depends, Path, Query
from sqlalchemy import select, any_, Select, update
from sqlalchemy.ext.asyncio import AsyncSession
from sqlalchemy.dialects.postgresql import insert
from sqlalchemy.orm import ColumnProperty

from the_remember.src.api.term_marks._redefinitions import UserDTO,
get_current_user, check_user, check_personalize_term
# from the_remember.src.api.auth.logics import get_current_user,
get_db_write_session
from the_remember.src.api.term_marks.src.db_model import
LearnTermDatetimeMarkORM
from the_remember.src.api.term_marks.src.dto import CreateLearnMarkDTO,
UpdateLearnMarksDTO
from the_remember.src.api.term_marks.src.dto import LearnMarkDTO
from the_remember.src.config.config import CONFIG
from the_remember.src.config.external_api_config import TermAPIConfig,
TermMarksAPIConfig
from the_remember.src.utils.answers.base import Ans, BError
from the_remember.src.utils.exeptions.handlers import base_resp_models
from the_remember.src.utils.get_data_from_auth_token import
get_db_write_session, get_db_session, CheckDbUserDep, \
    CheckDbEntitiesExistDep2
from the_remember.src.utils.swagger import limits_dep, fix_swagger_version


openapi_tags = [
    {
        "name": 'Term marks entities',
        "description": "",
        "externalDocs": {
            "description": "current microservice docs",
            "url": f"{TermMarksAPIConfig.base_path}/docs",
        },
    },
]
term_marks_app_ = FastAPI(
    openapi_tags=openapi_tags,
    servers=[
        {"url": CONFIG.BACKEND_MICROSERVICE_URL_AUTH, "description": "Staging
environment"},
    ],
)
term_marks_app = APIRouter(tags=list(map(lambda i: i['name'], openapi_tags)))

m_id_annn = Annotated[UUID, Path()]


@term_marks_app.post(
    "/create/{module_id}",
    response_model=Ans[LearnMarkDTO, BError | None],
    status_code=201,
    responses={**base_resp_models(401, 403, 404, 409, 422, 424, 456, 500)},
)
async def create_mark(
        module_id: m_id_annn,
        db_session: Annotated[AsyncSession, Depends(get_db_write_session)],
```

```
        current_user: Annotated[UserDTO, Depends(get_current_user)],
        checked: Annotated[bool, Depends(CheckDbUserDep(check_user))],
        new_marks: Annotated[
            CreateLearnMarkDTO,
            Depends(
                CheckDbEntitiesExistDep2[
                    CreateLearnMarkDTO,
                    lambda i, **k: {"term_id": i.term_id, "module_id":
k["module_id"]},
                    lambda i, **k: False,
                    {"current_user": Annotated[UserDTO,
Depends(get_current_user)],
                     "module_id": m_id_annn}
                ](TermAPIConfig, check_personalize_term, )
            )
        ]
):
    res = await db_session.execute(
        insert(LearnTermDatetimeMarkORM)
        .values(user_id=current_user.id, module_id=module_id)
        .returning(LearnTermDatetimeMarkORM), [
            new_marks.model_dump(exclude_none=True)
        ])
    data = list(res)
    if bool(data) is False:
        raise HTTPException(404, detail=dict())
    return LearnMarkDTO.model_validate(data[0][0])


@term_marks_app.get(
    "/all",
    response_model=Ans[list[LearnMarkDTO], BError | None],
    status_code=200,
    responses={**base_resp_models(401, 404, 409, 422, 456, 500)}
)
async def exist_term_marks(
        module_ids: Annotated[
            frozenset[UUID], Query(description="Список модулей, из которых
будут получены предложения")],
        db_session: Annotated[AsyncSession, Depends(get_db_session)],
        current_user: Annotated[UserDTO, Depends(get_current_user)],
        db_limits: Annotated[Callable[[Select], Select],
Depends(limits_dep(LearnTermDatetimeMarkORM))],
):
    query = (
        select(LearnTermDatetimeMarkORM)
        .where(
            (LearnTermDatetimeMarkORM.module_id == any_(module_ids))
            & (LearnTermDatetimeMarkORM.user_id == current_user.id)
        )
    )

    res = await db_session.execute(
        db_limits(query)
    )
    data = list(res)
    if bool(data) is False:
        raise HTTPException(404, detail=dict())
    return [LearnMarkDTO.model_validate(i[0]) for i in data]



@term_marks_app.put(
    "/update/{term_mark_id}",
```

```
    response_model=Ans[LearnMarkDTO, BError | None],
    status_code=200,
    responses={**base_resp_models(401, 403, 404, 409, 422, 424, 456, 500)},

)
async def update_learn_mark_entity(
        term_mark_id: UUID,
        update_mark: UpdateLearnMarksDTO,
        db_session: Annotated[AsyncSession, Depends(get_db_write_session)],
        current_user: Annotated[UserDTO, Depends(get_current_user)],
        # checked: Annotated[bool, Depends(CheckDbUserDep(check_user))],
):

    query = (
        update(LearnTermDatetimeMarkORM).
        where(
            (LearnTermDatetimeMarkORM.id == term_mark_id)
            & (LearnTermDatetimeMarkORM.user_id == current_user.id)
        )
        .returning(LearnTermDatetimeMarkORM)
        .values(**update_mark.model_dump()))

    data = list(await db_session.execute(query))
    if bool(data) is False:
        raise HTTPException(404, detail=dict())
    return LearnMarkDTO.model_validate(data[0][0])



@term_marks_app.get(
    "/from_module/{module_id}",
    response_model=Ans[LearnMarkDTO, BError | None],
    status_code=200,
    responses={**base_resp_models(401, 404, 409, 422, 456, 500)}
)
async def exist_term_marks(
        module_id: UUID,
        db_session: Annotated[AsyncSession, Depends(get_db_session)],
        current_user: Annotated[UserDTO, Depends(get_current_user)],
        db_limits: Annotated[Callable[[Select], Select],
Depends(limits_dep(LearnTermDatetimeMarkORM))],
):
    query = (
        select(LearnTermDatetimeMarkORM)
        .where(
            (LearnTermDatetimeMarkORM.module_id == module_id)
            & (LearnTermDatetimeMarkORM.user_id == current_user.id)
        )
    )

    res = await db_session.execute(
        db_limits(query)
    )
    data = list(res)
    if bool(data) is False:
        raise HTTPException(404, detail=dict())
    return [LearnMarkDTO.model_validate(i[0]) for i in data]


@term_marks_app.get(
    "/from_term/{term_id}",
    response_model=Ans[LearnMarkDTO, BError | None],
    status_code=200,
```

```
    responses={**base_resp_models(401, 404, 409, 422, 456, 500)}
)
async def exist_term_marks(
        term_id: UUID,
        db_session: Annotated[AsyncSession, Depends(get_db_session)],
        current_user: Annotated[UserDTO, Depends(get_current_user)],
        db_limits: Annotated[Callable[[Select], Select],
Depends(limits_dep(LearnTermDatetimeMarkORM))],
):
    query = (
        select(LearnTermDatetimeMarkORM)
        .where(
            (LearnTermDatetimeMarkORM.term_id == term_id)
            & (LearnTermDatetimeMarkORM.user_id == current_user.id)
        )
    )

    res = await db_session.execute(
        db_limits(query)
    )

    data = list(res)
    if bool(data) is False:
        raise HTTPException(404, detail=dict())
    return [LearnMarkDTO.model_validate(i[0]) for i in data]


@term_marks_app.get(
    "/exist",
    response_model=bool,
    status_code=200,
    responses={**base_resp_models(401, 404, 409, 422, 456, 500)}
)
async def exist_term_marks(
        term_marks_id: UUID,
        db_session: Annotated[AsyncSession, Depends(get_db_session)],
        current_user: Annotated[UserDTO, Depends(get_current_user)],
        term_id: UUID | None = None,
):
    q = (
        select(LearnTermDatetimeMarkORM)
        .where(
            (LearnTermDatetimeMarkORM.id == term_marks_id)
            & (LearnTermDatetimeMarkORM.user_id == current_user.id)
        )
    )
    if (term_id is not None):
        q = (q.where(LearnTermDatetimeMarkORM.term_id == term_id))

    res = await db_session.execute(select(q.exists()))
    data = list(res)
    if bool(data) is False:
        raise HTTPException(404, detail=dict())
    return data[0][0]


term_marks_app_.include_router(term_marks_app)
term_marks_app = term_marks_app_
term_marks_app.openapi = fix_swagger_version(term_marks_app)
```

Листинг Л.13 – Файл api/term/view.py

```
import datetime
from pprint import pprint
from typing import Annotated, Callable
import uuid

from fastapi import APIRouter, FastAPI, HTTPException
from fastapi.params import Depends, Query
import sqlalchemy as sa
from sqlalchemy import insert, select, delete, bindparam, update, any_,
Select
from sqlalchemy.dialects.postgresql import insert as pg_insert
from sqlalchemy.ext.asyncio import AsyncSession
from sqlalchemy.orm import ColumnProperty

from the_remember.src.api.terms._redefinitions import UserDTO,
get_current_user, check_user, check_module, \
    check_personal_module
from the_remember.src.api.terms.src.db_model import TermORM,
PersonalizeTermORM
from the_remember.src.api.terms.src.dto import TermDTO, CreateTermDTO,
PersonalizeTermDTO, \
    PersonalizeTermWithAddInfoDTO, \
    AdditionalTermInfoDTO, OnlyPersonalizePartTermDTO,
UpdateOnlyPersonalizePartTermDTO, \
    DeleteOnlyPersonalizePartTermDTO, CreateOnlyPersonalizePartTermDTO,
UpdateTermDTO
from the_remember.src.config.config import CONFIG
from the_remember.src.config.external_api_config import ModuleAPIConfig,
TermAPIConfig
from the_remember.src.utils.answers.base import Ans, BError
from the_remember.src.utils.exeptions.handlers import base_resp_models
from the_remember.src.utils.get_data_from_auth_token import
get_db_write_session, get_db_session, CheckDbUserDep, \
    CheckDbEntitiesExistDep2
from the_remember.src.utils.swagger import limits_dep, two_ent_upd_condition,
fix_swagger_version

openapi_tags = [
    {
        "name": 'Term entities',
        "description": "",
        "externalDocs": {
            "description": "current microservice docs",
            "url": f"{TermAPIConfig.base_path}/docs",
        },
    },
]
term_app_ = FastAPI(
    openapi_tags=openapi_tags,
    servers=[
        {"url": CONFIG.BACKEND_MICROSERVICE_URL_AUTH, "description": "Staging
environment"},
    ],

)
term_app = APIRouter(tags=list(map(lambda i: i['name'], openapi_tags)))


@term_app.post(
    "/create",
    response_model=Ans[TermDTO, BError | None],
    status_code=201,
    responses={**base_resp_models(401, 403, 404, 409, 422, 424, 456, 500)},
)
```

```
async def create_term(
        # new_term: CreateTermDTO,
        db_session: Annotated[AsyncSession, Depends(get_db_write_session)],
        current_user: Annotated[UserDTO, Depends(get_current_user)],
        checked: Annotated[bool, Depends(CheckDbUserDep(check_user))],
        new_term: Annotated[
            CreateTermDTO,
            Depends(
                CheckDbEntitiesExistDep2[
                    CreateTermDTO,
                    lambda i, **k: {"module_id": i.module_id, "author_id":
k["current_user"].id},
                    lambda i, **k: False,
                    {"current_user": Annotated[UserDTO,
Depends(get_current_user)]}
                ](ModuleAPIConfig, check_module, )
            )
        ]

):
    res = await db_session.execute(
        insert(TermORM)
        .returning(TermORM), [
            new_term.model_dump(exclude_none=True) | {'author_id':
current_user.id}
        ]
    )

    data = list(res)
    if bool(data) is False:
        raise HTTPException(404, detail=dict())
    return TermDTO.model_validate(data[0][0])


@term_app.get(
    "/all",
    response_model=Ans[list[TermDTO], BError | None],
    status_code=200,
    responses={**base_resp_models(401, 404, 409, 422, 456, 500)},
)
async def get_terms_from_module_list(
        db_session: Annotated[AsyncSession, Depends(get_db_session)],
        current_user: Annotated[UserDTO, Depends(get_current_user)],
        module_ids: Annotated[list[uuid.UUID], Query(description="Список
модулей, из которых будут получены термины")],
        db_limits: Annotated[Callable[[Select], Select],
Depends(limits_dep(TermORM))],
        author_only: bool = False,
) -> list[TermDTO]:
    query = (
        select(TermORM)
        .where(
            (TermORM.module_id == any_(module_ids))
        )
    )
    if author_only:
        query = query.where(TermORM.author_id == current_user.id)
    res = await db_session.execute(
        db_limits(query)
    )

    data = list(res)
    if data and bool(data):
        return [TermDTO.model_validate(i[0]) for i in data]
```

```
    return []


@term_app.get(
    "/all/search",
    response_model=Ans[list[TermDTO], BError | None],
    status_code=200,
    responses={**base_resp_models(401, 404, 409, 422, 456, 500)},
)
async def searching_in_terms(
        db_session: Annotated[AsyncSession, Depends(get_db_session)],
        current_user: Annotated[UserDTO, Depends(get_current_user)],
        module_ids: Annotated[frozenset[uuid.UUID], Query(description="Список
модулей, из которых будут получены термины",

default_factory=frozenset, )],
        db_limits: Annotated[Callable[[Select], Select],
Depends(limits_dep(TermORM))],
        pattern: str,
        author_only: bool = False,
        personalize_only: bool = False,
        accurate: bool = False,
) -> list[TermDTO]:
    if accurate is False:
        pattern = f'%{pattern}%'
    query = (
        select(TermORM)
        .where(
            (TermORM.term.ilike(pattern))
            | (TermORM.definition.ilike(pattern))
        )
    )
    if module_ids is not None and bool(module_ids):
        query = query.where(
            (TermORM.module_id == any_(module_ids))
        )
    if author_only:
        query = query.where(TermORM.author_id == current_user.id)
    elif personalize_only:
        query = query.where(PersonalizeTermORM.user_id == current_user.id)
    res = await db_session.execute(
        db_limits(query)
    )

    data = list(res)
    if data and bool(data):
        return [TermDTO.model_validate(i[0]) for i in data]
    return []


@term_app.get(
    "/{term_id}",
    response_model=Ans[PersonalizeTermDTO, BError | None],
    status_code=200,
    responses={**base_resp_models(401, 404, 409, 422, 456, 500)},
)
async def get_one_term(
        term_id: uuid.UUID,
        db_session: Annotated[AsyncSession, Depends(get_db_session)],
        current_user: Annotated[UserDTO, Depends(get_current_user)]
):
    res = await db_session.execute(
        select(TermORM)
        .where((TermORM.id == term_id))
```

```
    )

    data = list(res)
    if bool(data) is False:
        raise HTTPException(404, detail=dict())
    return PersonalizeTermDTO.model_validate(data[0][0].term_entity.__dict__
| data[0][0].__dict__)



@term_app.get(
    "/from_module/{module_id}",
    response_model=Ans[list[TermDTO], BError | None],
    status_code=200,
    responses={**base_resp_models(401, 404, 409, 422, 456, 500)},
)
async def get_terms_from_module(
        module_id: uuid.UUID,
        db_session: Annotated[AsyncSession, Depends(get_db_session)],
        current_user: Annotated[UserDTO, Depends(get_current_user)],
        db_limits: Annotated[Callable[[Select], Select],
Depends(limits_dep(TermORM))],
        author_only: bool = False,
) -> list[TermDTO]:
    query = (
        select(TermORM)
        .where(TermORM.module_id == module_id)
    )

    if author_only:
        query = query.where(TermORM.author_id == current_user.id)

    res = await db_session.execute(
        db_limits(query)
    )

    data = list(res)
    if data and bool(data):
        return [TermDTO.model_validate(i[0]) for i in data]
    return []


@term_app.get(
    "/from_module/{module_id}/ids_only",
    response_model=Ans[list[uuid.UUID], BError | None],
    status_code=200,
    responses={**base_resp_models(401, 404, 409, 422, 456, 500)},
)
async def get_term_ids_from_module(
        module_id: uuid.UUID,
        db_session: Annotated[AsyncSession, Depends(get_db_session)],
        current_user: Annotated[UserDTO, Depends(get_current_user)]
) -> list[uuid.UUID]:
    res = await db_session.execute(
        select(TermORM.id)
        .where(TermORM.module_id == module_id)
    )

    data = list(res)
    if data and bool(data):
        return [i[0] for i in data]
    return []
```

```
@term_app.put(
    "/update/{term_id}",
    response_model=Ans[TermDTO, BError | None],
    status_code=200,
    responses={**base_resp_models(401, 403, 404, 409, 422, 424, 456, 500)},
)
async def update_term_entity(
        term_id: uuid.UUID,
        update_term: UpdateTermDTO,
        db_session: Annotated[AsyncSession, Depends(get_db_write_session)],
        current_user: Annotated[UserDTO, Depends(get_current_user)]
):
    query = (
        update(TermORM).
        where(
            (TermORM.id == term_id)
            & (TermORM.author_id == current_user.id)
        )
        .returning(TermORM)
        .values(**update_term.model_dump()))

    data = list(await db_session.execute(query))
    if bool(data) is False:
        raise HTTPException(404, detail=dict())
    return TermDTO.model_validate(data[0][0])

@term_app.get(
    "/personalize/all",
    response_model=Ans[list[PersonalizeTermDTO], BError | None],
    status_code=200,
    responses={**base_resp_models(401, 409, 422, 456, 500)}
)
async def get_all_personalize_term(
        db_session: Annotated[AsyncSession, Depends(get_db_session)],
        current_user: Annotated[UserDTO, Depends(get_current_user)],
        module_ids: Annotated[frozenset[uuid.UUID], Query(description="Список
модулей, из которых будут получены термины")],
        db_limits: Annotated[Callable[[Select], Select], Depends(limits_dep(
            PersonalizeTermORM,
            updated_at_condition=two_ent_upd_condition(PersonalizeTermORM,
TermORM)
        ))],
        author_only: bool = False,
):
    query = (
        select(PersonalizeTermORM)
        .join(PersonalizeTermORM.term_entity)
        .add_columns(TermORM)
        .where(
            (PersonalizeTermORM.user_id == current_user.id)
            & (PersonalizeTermORM.module_id == any_(module_ids))
        )
    )
    if author_only:
        query = query.where(TermORM.author_id == current_user.id)
    res = await db_session.execute(
        db_limits(query)
    )
    data = list(res)
    return [PersonalizeTermDTO.model_validate(i[0].term_entity.__dict__ |
i[0].__dict__) for i in data]
```

```
@term_app.get(
    "/personalize/from_module/{module_id}",
    response_model=Ans[list[PersonalizeTermDTO], BError | None],
    status_code=200,
    responses={**base_resp_models(401, 409, 422, 456, 500)}
)
async def get_all_personalize_term_from_module(
        module_id: uuid.UUID,
        db_session: Annotated[AsyncSession, Depends(get_db_session)],
        current_user: Annotated[UserDTO, Depends(get_current_user)],
        db_limits: Annotated[Callable[[Select], Select], Depends(limits_dep(
            PersonalizeTermORM,
            updated_at_condition=two_ent_upd_condition(PersonalizeTermORM,
TermORM)
        ))],
        author_only: bool = False,
):
    query = (
        select(PersonalizeTermORM)
        .join(PersonalizeTermORM.term_entity)
        .add_columns(TermORM)
        .where(
            (PersonalizeTermORM.user_id == current_user.id)
            & (PersonalizeTermORM.module_id == module_id)
        )
    )
    if author_only:
        query = query.where(TermORM.author_id == current_user.id)
    res = await db_session.execute(
        db_limits(query)
    )

    data = list(res)
    return [PersonalizeTermDTO.model_validate(i[0].term_entity.__dict__ |
i[0].__dict__) for i in data]



from_module_microservice_annotation = Annotated[bool, Query(
    default_factory=lambda: False, deprecated=True,
    description="Костыль, при переписывании взаимодействия с использованием
Саги будет удален")]


@term_app.post(
    "/personalize/create",
    response_model=Ans[OnlyPersonalizePartTermDTO, BError | None],
    status_code=201,
    summary="SYSTEM ONLY",
    description="этот роут будет удален с внедрением Саги",
    deprecated=True,
    responses={**base_resp_models(401, 403, 404, 409, 422, 424, 456, 500)},
)
async def create_personalize_term(
        # create_personal_term: CreateOnlyPersonalizePartTermDTO,
        db_session: Annotated[AsyncSession, Depends(get_db_write_session)],
        current_user: Annotated[UserDTO, Depends(get_current_user)],
        from_module_microservice: from_module_microservice_annotation,
        create_personal_term: Annotated[
            CreateOnlyPersonalizePartTermDTO,
            Depends(
                CheckDbEntitiesExistDep2[
                    CreateOnlyPersonalizePartTermDTO,
```

```
                    lambda i, **k: {"module_id": i.module_id, "current_user":
k["current_user"]},
                    lambda i, **k: k["from_folder_microservice"] is True,
                    {"current_user": Annotated[UserDTO,
Depends(get_current_user)],
                     "from_module_microservice":
from_module_microservice_annotation}
                ](ModuleAPIConfig, check_personal_module, )
            )
        ],
):
    scalar_module_id = (
        select(TermORM.module_id)
        .where(TermORM.id == bindparam("term_id_"))
        .scalar_subquery()
    )

    res = await db_session.execute(
        insert(PersonalizeTermORM)
        .values(module_id=scalar_module_id)
        .returning(PersonalizeTermORM),
        [
            create_personal_term.model_dump() | {
                'user_id': current_user.id,
                "term_id_": create_personal_term.term_id
            }
        ]
    )
    data = list(res)
    if bool(data) is False:
        raise HTTPException(404, detail=dict())
    return OnlyPersonalizePartTermDTO.model_validate(data[0][0])


@term_app.put(
    "/personalize/update/{term_id}",
    response_model=Ans[OnlyPersonalizePartTermDTO, BError | None],
    status_code=200,
    responses={**base_resp_models(401, 403, 404, 409, 422, 424, 456, 500)},
)
async def update_personalize_term(
        term_id: uuid.UUID,
        update_personal_term: UpdateOnlyPersonalizePartTermDTO,
        db_session: Annotated[AsyncSession, Depends(get_db_write_session)],
        current_user: Annotated[UserDTO, Depends(get_current_user)]
):
    query = (
        update(PersonalizeTermORM).
        where(
            (PersonalizeTermORM.term_id == term_id)
            & (PersonalizeTermORM.user_id == current_user.id)
        )
        .returning(PersonalizeTermORM)
        .values(**update_personal_term.model_dump()))

    data = list(await db_session.execute(query))
    if bool(data) is False:
        raise HTTPException(404, detail=dict())
    return OnlyPersonalizePartTermDTO.model_validate(data[0][0])


@term_app.get(
    "/personalize/{term_id}",
    response_model=Ans[PersonalizeTermDTO, BError | None],
```

```
    status_code=200,
    responses={**base_resp_models(401, 409, 422, 456, 500)}
)
async def get_all_personalize_term_from(
        term_id: uuid.UUID,
        db_session: Annotated[AsyncSession, Depends(get_db_session)],
        current_user: Annotated[UserDTO, Depends(get_current_user)],
        author_only: bool = False,
):
    query = (
        select(PersonalizeTermORM)
        .join(PersonalizeTermORM.term_entity)
        .add_columns(TermORM)
        .where(
            (PersonalizeTermORM.user_id == current_user.id)
            & (PersonalizeTermORM.term_id == term_id)
        )
    )
    if author_only:
        query = query.where(TermORM.author_id == current_user.id)
    res = await db_session.execute(query)

    data = list(res)
    if bool(data) is False:
        raise HTTPException(404, detail=dict())
    return PersonalizeTermDTO.model_validate(data[0][0].term_entity.__dict__
| data[0][0].__dict__)


@term_app.get(
    "/exist",
    response_model=bool,
    status_code=200,
    responses={**base_resp_models(401, 404, 409, 422, 456, 500)}
)
async def exist_term(
        term_id: uuid.UUID,
        db_session: Annotated[AsyncSession, Depends(get_db_session)],
        # current_user: Annotated[UserDTO, Depends(get_current_user)],
        module_id: uuid.UUID | None = None,
        author_id: uuid.UUID | None = None,
):
    q = (
        select(TermORM)
        .where((TermORM.id == term_id))
    )
    if (module_id is not None):
        q = (q.where(TermORM.module_id == module_id))
    if (author_id is not None):
        q = (q.where(TermORM.author_id == author_id))

    res = await db_session.execute(
        select(q.exists())
    )
    data = list(res)
    if bool(data) is False:
        raise HTTPException(404, detail=dict())

    return data[0][0]
```

ПРИЛОЖЕНИЕ М
(РЕКОМЕНДУЕМОЕ)
КОД КЛИЕНТСКОЙ ЧАСТИ

Листинг М.1 – Главный файл программы

```
import 'dart:math';

import 'package:flutter/material.dart';
import 'package:the_remember/src/ui/ui_states/navigation_processor.dart';

void main() async {

  runApp( MyApp() );
}

Random random = new Random();


class MyApp extends StatelessWidget {
  // функция build отвечает за построение иерархии виджетов
  @override
  Widget build(BuildContext context) {
    // виджет MaterialApp - главный виджет приложения, который
    // позволяет настроить тему и использовать
    // Material Design для разработки.

    return
      // Consumer<UserApiProfile>(
      //   builder: (context, userApi, child)=>

      MaterialApp(
        // заголовок приложения
        // обычно виден, когда мы сворачиваем приложение
        title: 'Json Placeholder App',
        // настройка темы, мы еще вернемся к этому
        theme: ThemeData(
          primarySwatch: Colors.blue,
        ),
        // указываем исходную страницу, которую мы создадим позже
        home: NavigationProcessor(),
        onUnknownRoute: (settings) {
          // return MaterialPageRoute(builder: (context) {
          //   return StartModule();
          // });
        },
        onGenerateRoute: (settings) {

        });
      // );
  }
}
```

Листинг М.2 – Код сервиса взаимодействия с сервером

```
import 'dart:isolate';

import 'package:isar/isar.dart';
import
'package:the_remember/db_layer/network/entities_update/update_controller.dart
';
import 'package:the_remember/db_layer/network/get_urls.dart';
import
'package:the_remember/network_processor/url_controller/auth_extension.dart';
import 'package:the_remember/network_processor/url_controller/main.dart';
```

```
import '../db_layer/db_utils.dart';
import '../src/domain_layer/providers/isolates/network.dart';
import '../src/repositoris/db_data_source/http_utils.dart';
import '../src/repositoris/db_data_source/user.dart';
import '../src/urils/db/dbMixins.dart';
import '../src/urils/db/engine.dart';
import '../src/urils/isolate/base_msg.dart';
import '../src/urils/isolate/msgs/login_user.dart';
import '../src/urils/isolate/isolate_ listner.dart';
import '../src/urils/streams.dart';
import 'network_errors.dart';

Future<void> runNetworkIsolate(SendPort callerSendPort) async {
  ReceivePort newIsolateReceivePort = ReceivePort();
  callerSendPort.send(CrossIsolatesMessage<bool>(
    sender: newIsolateReceivePort.sendPort,
    message: true,
    msgType: CrossIsolatesMessageType.sendPort,
  ));

  newIsolateReceivePort.listen(IsolateMessageListener.instance.mainListener);

  var (dbPathStream, killerFlag) = IsolateMessageListener.instance
      .getMessageStream(CrossIsolatesMessageType.dbPath);
  CrossIsolatesMessage dbPath =
      await dbPathStream.first as CrossIsolatesMessage;
  killerFlag[0] = true;
  IzarManager.instance.dbExternalPath = dbPath.message as String;

  var networkProcessor = NetworkProcessor();
  networkProcessor.run(callerSendPort, newIsolateReceivePort);
}

enum EventsFutureType {
  authGen,
  timer,
  entUpdateGen,
}

class NetworkProcessor with OpenAndClose {
  static final NetworkProcessor instance = NetworkProcessor._internal();

  late final Stream<CrossIsolatesMessage<LoginUserIsolateMsg>>
userPassListener;
  late final SendPort callerSendPort;
  late final ReceivePort receivePort;
  late final DbUtils dbController;
  late final UrlsController urlController;

  UserDbDS? _activeUser;

  UserDbDS get activeUser {
    if (_activeUser == null) {
      throw UserNotActiveException();
    }
    return _activeUser!;
  }

  set activeUser(UserDbDS? value) {
    _activeUser = value;
  }

  NetworkProcessor._internal() {
```

```
    userPassListener = IsolateMessageListener.instance
        .getMessageStream<LoginUserIsolateMsg>(
            CrossIsolatesMessageType.userAndPassword)
        .$1;
    dbController = DbUtils();
  }

  factory NetworkProcessor() {
    return instance;
  }

  Future<void> run(
      SendPort callerSendPort_, ReceivePort newIsolateReceivePort) async {
    callerSendPort = callerSendPort_;
    receivePort = newIsolateReceivePort;
    var isar = await openConn();
    // await findActiveUserInDb(isar);
    userActiveControlLoopRun(isar);
    await multiUsersLoop(isar);
    await closeConn();
  }

  Future<void> userActiveControlLoopRun(Isar isar) async {
    var query = isar.collection<UserDbDS>().filter().activeEqualTo(true);
    var activeUserStream = query.build().watch(fireImmediately: true);
    var activeUserStreamController = activeUserStream.listen((activeUsers) {
      if (activeUsers.length == 1) {
        activeUser = activeUsers[0];
      } else {
        activeUser = null;
      }
    });
  }

  Future<void> multiUsersLoop(Isar isar) async {
    var (urlsGen, locUrlController) =
        await dbController.getConnectionUrls(isar);
    urlController = locUrlController;
    while (true) {
      try {
        var authedGen =
            getAuthHeaders(activeUser.username, activeUser.password,
urlsGen);
        var getNextAuthFunc = nextValFromStream(authedGen);
        timerEvent() => Future.delayed(
            const Duration(seconds: 5), () => (null,
EventsFutureType.timer));
        Future<(UnaryUrlController?, EventsFutureType)>
            getNextAuthFunc2() async {
          return (await getNextAuthFunc(), EventsFutureType.authGen);
        }

        Future<(dynamic, EventsFutureType)> eventsQueryGen2() async {
          return (
            await urlController.eventsQueryGen(),
            EventsFutureType.entUpdateGen
          );
        }

        /// Сборщик и обработчик событий,
        /// пришедших из другого изолята и из сети
        Map<
            EventsFutureType,
            (
```

```
            Future<(dynamic, EventsFutureType)>?,
            Future<(dynamic, EventsFutureType)> Function()
          )> eventsMap = {
        EventsFutureType.authGen: (null, getNextAuthFunc2),
        EventsFutureType.timer: (null, timerEvent),
        EventsFutureType.entUpdateGen: (null, eventsQueryGen2)
      };

      /// Цикл обработки событий
      while (true) {
        var currEvent = await Future.any([
          for (var item in eventsMap.entries)
            if (item.value.$1 != null) item.value.$1!
        ]);
        eventsMap[currEvent.$2] =
            (eventsMap[currEvent.$2]!.$2(), eventsMap[currEvent.$2]!.$2);

        // TODO: Совершаем работу в зависимости от типа события
        if (currEvent.$2 == EventsFutureType.authGen) {
          // TODO: Запускаем очередь событий для конкретной ссылки
          urlController.workloadRegistrar(
              currEvent.$1 as UnaryUrlController, 0);
          eventsMap[EventsFutureType.entUpdateGen] = (
            eventsMap[EventsFutureType.entUpdateGen]!.$2(),
            eventsMap[EventsFutureType.entUpdateGen]!.$2
          );
        } else  if (currEvent.$2 == EventsFutureType.entUpdateGen) {
          var (futureData, urlController) = currEvent.$1 as (dynamic,
UnaryUrlController);
          dbController.registerDbUpdateOperation(
            null,
              urlController,
              currEvent.$2 as
          );

        }
      }

      // await updateDbEntitiesFromServer(isar, urlsGen);
      // TODO: в конце переходим в фон и ждем, наверно
    } catch (UserNotActiveException) {
      var msg = await userPassListener.firstWhere((element) => true);

      late List<HttpUtilsDbDS> currConnUrls;
      if (msg.message.serverUrl != null) {
        var currConnUrl = await isar.httpUtilsDbDSs
            .filter()
            .httpUrlEqualTo(msg.message.serverUrl!)
            .findFirst();
        if (currConnUrl != null) {
          currConnUrls = [currConnUrl];
        } else {
          currConnUrl = HttpUtilsDbDS()..httpUrl = msg.message.serverUrl!;
          isar.writeTxnSync(
              () => {isar.httpUtilsDbDSs.putSync(currConnUrl!)});
          currConnUrls = [currConnUrl];
        }
        urlController = UrlsController(currConnUrls);
        urlsGen = urlController.activeApis;
      }

      // TODO: обработать ошибку, если поток пуст
      var urlObj = await getAuthHeaders(
              msg.message.username, msg.message.password, urlsGen)
```

```
            .first;

        msg.sender.send(CrossIsolatesMessage<LoginUserAnsIsolateMsg>(
          sender: receivePort.sendPort,
          // TODO: если без ошибок
          message: LoginUserAnsIsolateMsg(200, true),
          msgType: CrossIsolatesMessageType.loginUserAnsIsolateMsg,
        ));
      }
    }
  }

  Stream<UnaryUrlController> getAuthHeaders(String username, String password,
      Stream<UnaryUrlController> Function() urlsGen) async* {
    List<Map<String, String>> res = [];

    await for (var urlC in urlsGen()) {
      try {
        var authHeaders = await urlC.auth(username, password);
        res.add(authHeaders);
        yield urlC;
      } catch (AuthErrorUserPasswordNotCorrect) {
        // TODO: если  ни от одной ссылки  не вернулось ответа 200,
        //  то возвращаем сообщение в главный поток о том,
        //  что неверный username или password
      } catch (AuthErrorUserPasswordNotCorrect) {
        // FIXME: отлавливать ошибк connection timeout.
        //  поидее тут таких быть не должно,
        //  т.к. мы итерируемся только по активным ссылкам,
        //  но вдруг попадется
      }

      if (res.isEmpty) {
        // TODO: возвращаем сообщение в главный поток о том,
        //  что неверный username или password
        throw UserNotActiveException();
      }
    }
  }

  Future<void> updateDbEntitiesFromServer(
      Isar isar, Stream<UnaryUrlController> Function() urlsGen) async {
    // folderUpdate(isar, urlsGen);
    // modules_update();
    // terms_update();
    // sentenses_update();
    // add_term_info_update();
    // term_marks_update();
    throw UnimplementedError();
  }
}
```

Листинг М.3 – network_main.dart

```
import 'dart:convert';

import 'package:built_value/json_object.dart';
import 'package:built_value/serializer.dart';
import 'package:dio/dio.dart';
import 'package:isar/isar.dart';
import 'package:pretty_dio_logger/pretty_dio_logger.dart';
import 'package:the_remember/api_package/lib/api_package.dart';
```

```
import 'package:the_remember/src/repositoris/db_data_source/module.dart';
import 'package:the_remember/src/repositoris/db_data_source/sentence.dart';
import 'package:the_remember/src/repositoris/db_data_source/term.dart';
import 'package:the_remember/src/repositoris/db_data_source/user.dart';

import '../src/domain_layer/providers/isolates/network.dart';
import '../src/domain_layer/providers/user_api_provider.dart';
import '../src/repositoris/db_data_source/folder.dart';
import '../src/repositoris/db_data_source/http_utils.dart';
import '../src/repositoris/db_data_source/term_adding_info.dart';
import '../src/urils/db/dbMixins.dart';
import '../src/urils/db/engine.dart';
import '../src/urils/isolate/base_msg.dart';
import '../src/urils/isolate/msgs/login_user.dart';

// Future<FolderDTO> loadPet() async {
//   final Dio dio = Dio(BaseOptions(baseUrl: 'http://192.168.0.105:10010'));
//   final ApiPackage authApi = ApiPackage(dio: dio,
serializers:standardSerializers);
//   final DefaultApi mainApi = authApi.getDefaultApi();
//   JsonObject("string");
//   final authData = mainApi.loginForAccessTokenAuthTokenPost(username:
JsonObject("string"), password: JsonObject("string"));
//
//   const petId = 9;
//   final Response<Pet> response = await authApi.
//       // .getPetById(petId, headers: <String, String>{'Authorization':
'Bearer special-key'});
//   return response.data;
// }

Future<void> networkProcessor(UserApiProfile? userApi) async {
  return;
  // if (userApi?.baseApi == null && userApi?.user != null) {
  //   var conn = (await OpenAndClose3.openConnStatic(
  //       [CollectionSchema<HttpUtilsDbDS>, CollectionSchema<UserDbDS>]));
  //
  //   var serverUrlsEntities = (await conn[ConnType.server_urls]!
  //       .collection<HttpUtilsDbDS>()
  //       .where()
  //       .findAll());
  //   // var user = (await conn[ConnType.user]!
  //   //     .collection<UserDbDS>()
  //   //     .getByUuid(userApi!.user!.uuid));
  //   var serverUrls = [for (var i in serverUrlsEntities) i.httpUrl];
  //
  //   await OpenAndClose3.closeConnStatic(conn);
  //   await loginUser(userApi!.user!.username, userApi.user!.password,
  //       serverUrls: serverUrls, userApi: userApi);
  //   return;
  // } else if (userApi?.baseApi == null && userApi?.user == null) {
  //   // var conn = (await OpenAndClose3.openConnStatic(
  //   //     [CollectionSchema<HttpUtilsDbDS>,
CollectionSchema<UserDbDS>]));
  //   //
  //   // var serverUrlsEntities = (await conn[ConnType.server_urls]!
  //   //     .collection<HttpUtilsDbDS>()
  //   //     .where()
  //   //     .findAll());
  //   // // var user = (await conn[ConnType.user]!
  //   // //     .collection<UserDbDS>()
  //   // //     .getByUuid(userApi!.user!.uuid));
  //   // var serverUrls = [for (var i in serverUrlsEntities) i.httpUrl];
  //   //
```

```
  //   // await OpenAndClose3.closeConnStatic(conn);
  //   // await loginUser(userApi!.user!.username, userApi.user!.password,
  //   //     serverUrls: serverUrls, userApi: userApi);
  //   return;
  // }
  // print("networkProcessor(userApi) started!!!!!!!!!!!!");
  // var baseApi = userApi!.baseApi!;
  // var authHeaders = userApi.authHeaders;
  // // final Dio dio = Dio(BaseOptions(baseUrl: serverUrl));
  // // final ApiPackage authApi =
  // //     ApiPackage(dio: dio, serializers: standardSerializers);
  // // final AuthApi realAuthApi = authApi.getAuthApi();
  // // final authData = await realAuthApi.loginForAccessTokenAuthTokenPost(
  // //     username: JsonObject("1"), password: JsonObject("1"));
  // // authApi.setBearerAuth(
  // //     "Authorization", "Bearer
${authData.data!.accessToken!.asString}");
  //
  // final FoldersEntitiesApi folderApi = baseApi.getFoldersEntitiesApi();
  // final ModuleEntitiesApi moduleApi = baseApi.getModuleEntitiesApi();
  // final TermEntitiesApi termApi = baseApi.getTermEntitiesApi();
  // final SentenceEntitiesApi sentenceApi =
baseApi.getSentenceEntitiesApi();
  // // JsonObject("string");
  // late Map<ConnType, Isar> conn;
  // late Map<String, FolderDbDS> foldersFromDb;
  // late Map<String, ModuleDbDS> modulesFromDb;
  // late Map<String, TermEntityDbDS> termsFromDb;
  // late Map<String, TermAddingInfoDbDS> addInfoTermsFromDb;
  // late Map<String, SentenceDbDS> sentenceFromDb;
  //
  // var coro = (() async {
  //   // await Future.delayed(Duration(seconds: 5));
  //   conn = (await OpenAndClose3.openConnStatic([
  //     CollectionSchema<FolderDbDS>,
  //     CollectionSchema<ModuleDbDS>,
  //     CollectionSchema<TermEntityDbDS>,
  //     CollectionSchema<TermAddingInfoDbDS>,
  //   ]));
  //
  //   foldersFromDb = {
  //     for (var ent in await conn[ConnType.term]!
  //         .collection<FolderDbDS>()
  //         .filter()
  //         .userUuidEqualTo(userApi!.user!.uuid)
  //         .findAll())
  //       ent.uuid: ent
  //   };
  //   modulesFromDb = {
  //     for (var ent in await conn[ConnType.term]!
  //         .collection<ModuleDbDS>()
  //         .filter()
  //         .userUuidEqualTo(userApi.user!.uuid)
  //         .findAll())
  //       ent.uuid: ent
  //   };
  //   termsFromDb = {
  //     for (var ent in await conn[ConnType.term]!
  //         .collection<TermEntityDbDS>()
  //         .filter()
  //         .userUuidEqualTo(userApi.user!.uuid)
  //         .findAll())
  //       ent.uuid: ent
  //   };
```

```
  //   addInfoTermsFromDb = {
  //     for (var ent in await conn[ConnType.term]!
  //         .collection<TermAddingInfoDbDS>()
  //         .filter()
  //         .userUuidEqualTo(userApi.user!.uuid)
  //         .findAll())
  //       ent.uuid: ent
  //   };
  //   sentenceFromDb = {
  //     for (var ent in await conn[ConnType.term]!
  //         .collection<SentenceDbDS>()
  //         .filter()
  //         .userUuidEqualTo(userApi.user!.uuid)
  //         .findAll())
  //       ent.uuid: ent
  //   };
  // })();
  // return;
  // // TODO: Переписать всю работу с сетью
  // // final foldersCoro = folderApi.getAllFoldersFolderAllGet(headers:
authHeaders);
  // // final modulesCoro = moduleApi.getAllModuleModuleAllGet(headers:
authHeaders);
  // // final termsCoro = termApi.getAllTermTermAllGet(headers: authHeaders);
  // // final addInfoCoro =
  // //     termApi.getAllAddTermInfoTermAddInfoAllGet(headers: authHeaders);
  // // final sentenceCoro =
  // //     sentenceApi.getAllTermSentenceAllGet(headers: authHeaders);
  // //
  // // final folders = await foldersCoro;
  // //
  // // // const _type = FullType(FolderDTO);
  // // var test = standardSerializers.deserialize(
  // //   folders.data!.asList[0],
  // //   specifiedType: const FullType(FolderDTO),
  // // ) as FolderDTO;
  // // // standardSerializers.serialize(, specifiedType: _type);
  // //
  // // // FolderDTO.serializer.
  // //
  // // // var test = FolderDTO.serializer.deserialize(standardSerializers,
folders.data!.asList[0]);
  // // var res = await coro;
  // // var foldersToDb = {
  // //   for (var ent in [
  // //     for (var folder in folders.data!.asList)
  // //       FolderDbDS.fromJson(standardSerializers.deserialize(folder,
  // //               specifiedType: const FullType(PersonalizeFolderDTO))
  // //           as PersonalizeFolderDTO)
  // //   ])
  // //     ent.uuid: ent
  // // };
  // // for (var ent in foldersToDb.values) {
  // //   if (ent.rootFolderUuid != null) {
  // //     ent.rootFolder.value = foldersToDb[ent.rootFolderUuid]!;
  // //   }
  // // }
  // //
  // // ModuleDbDS moduleTransform(ModuleDbDS networkModule) {
  // //   networkModule.rootFolder.value =
foldersToDb[networkModule.rootFolderUuid];
  // //   if (modulesFromDb[networkModule.uuid] != null) {
  // //     var dbModule = modulesFromDb[networkModule.uuid]!;
  // //     if (networkModule.personalUpdatedAt
```

```
  // //         .isBefore(dbModule.personalUpdatedAt)) {
  // //       networkModule
  // //         // FIXME:
  // //         ..isReverseDefinitionWrite =
dbModule.isReverseDefinitionWrite
  // //         ..standardAndReverseWrite = dbModule.standardAndReverseWrite
  // //         ..isReverseDefinitionChoice =
dbModule.isReverseDefinitionChoice
  // //         ..standardAndReverseChoice =
dbModule.standardAndReverseChoice
  // //         ..maxIterationLen= dbModule.maxIterationLen
  // //         ..minIterationLen= dbModule.minIterationLen
  // //         ..minWatchCount= dbModule.minWatchCount
  // //         ..knownTermPart= dbModule.knownTermPart
  // //         ..choicesCount= dbModule.choicesCount
  // //         ..isLearnt= dbModule.isLearnt
  // //         ..personalUpdatedAt = dbModule.personalUpdatedAt;
  // //     }
  // //   }
  // //   return networkModule;
  // // }
  // //
  // // final modules = await modulesCoro;
  // // var modulesToDb = {
  // //   for (var ent in [
  // //     for (var module in modules.data!.asList)
  // //       ModuleDbDS.fromJson(standardSerializers.deserialize(module,
  // //               specifiedType: const FullType(PersonalizeModuleDTO))
  // //           as PersonalizeModuleDTO)
  // //   ])
  // //     ent.uuid: moduleTransform(ent)
  // // };
  // //
  // // final terms = await termsCoro;
  // // var termsToDb = {
  // //   for (var ent in [
  // //     for (var term in terms.data!.asList)
  // //       TermEntityDbDS.fromJson(standardSerializers.deserialize(term,
  // //               specifiedType: const FullType(PersonalizeTermDTO))
  // //           as PersonalizeTermDTO)
  // //   ])
  // //     ent.uuid: ent
  // // };
  // //
  // // List<TermEntityDbDS> serverUpdatesTerm = [];
  // //
  // // for (var networkTerm in termsToDb.values) {
  // //   networkTerm.module.value = modulesToDb[networkTerm.moduleUuid]!;
  // //   if (termsFromDb[networkTerm.uuid] != null) {
  // //     var dbTerm = termsFromDb[networkTerm.uuid]!;
  // //     if
(networkTerm.personalUpdatedAt.isBefore(dbTerm.personalUpdatedAt)) {
  // //       networkTerm
  // //         ..chooseErrorCounter = dbTerm.chooseErrorCounter
  // //         ..writeErrorCounter = dbTerm.writeErrorCounter
  // //         ..choiceNegErrorCounter = dbTerm.choiceNegErrorCounter
  // //         ..watchCount=dbTerm.watchCount
  // //         ..personalUpdatedAt = dbTerm.personalUpdatedAt;
  // //     } else if (networkTerm.personalUpdatedAt
  // //         .isAfter(dbTerm.personalUpdatedAt)) {
  // //       serverUpdatesTerm.add(dbTerm);
  // //     }
  // //   }
  // // }
```

```
  // // var serverTermUpdateCoro =
  // //     updatePersonalizedTerms(serverUpdatesTerm, userApi);
  // //
  // // final addInfoTerms = await addInfoCoro;
  // // var addInfoTermsToDb = {
  // //   for (var ent in [
  // //     for (var ait in addInfoTerms.data!.asList)
  // //       TermAddingInfoDbDS.fromJson(
  // //           standardSerializers.deserialize(ait,
  // //               specifiedType: const FullType(AdditionalTermInfoDTO)
  // //               // ..["add_info_type"]=[ait["add_info_type"]],
  // //               //     specifiedType: const
FullType(AdditionalTermInfoDTO)
  // //               ) as AdditionalTermInfoDTO,
  // //           userApi.user!.uuid)
  // //   ])
  // //     ent.uuid: ent
  // // };
  // //
  // // for (var networkAIT in addInfoTermsToDb.values) {
  // //   networkAIT.termEntity.value = termsToDb[networkAIT.termUuid]!;
  // //   if (addInfoTermsFromDb[networkAIT.uuid] != null) {
  // //     var dbAIT = addInfoTermsFromDb[networkAIT.uuid]!;
  // //     if (networkAIT.updatedAt.isBefore(dbAIT.updatedAt)) {
  // //       networkAIT = dbAIT;
  // //     }
  // //   }
  // // }
  // //
  // // final sentences = await sentenceCoro;
  // // var sentencesToDb = {
  // //   for (var ent in [
  // //     for (var ait in sentences.data!.asList)
  // //       SentenceDbDS.fromJson(
  // //           standardSerializers.deserialize(ait,
  // //               specifiedType: const FullType(SentenceDTO)
  // //             // ..["add_info_type"]=[ait["add_info_type"]],
  // //             //     specifiedType: const
FullType(AdditionalTermInfoDTO)
  // //           ) as SentenceDTO,
  // //           userApi.user!.uuid)
  // //   ])
  // //     ent.uuid: ent
  // // };
  // //
  // // for (var networkSentences in sentencesToDb.values) {
  // //   networkSentences.termEntity.value =
termsToDb[networkSentences.termUuid]!;
  // //   if (sentenceFromDb[networkSentences.uuid] != null) {
  // //     var dbSentence = sentenceFromDb[networkSentences.uuid]!;
  // //     if (networkSentences.updatedAt.isBefore(dbSentence.updatedAt)) {
  // //       networkSentences = dbSentence;
  // //     }
  // //   }
  // // }
  // //
  // //
  // // conn[ConnType.term]!.writeTxnSync(() {
  // //   conn[ConnType.term]!
  // //       .collection<FolderDbDS>()
  // //       .putAllSync(foldersToDb.values.toList());
  // //   conn[ConnType.term]!
  // //       .collection<ModuleDbDS>()
  // //       .putAllSync(modulesToDb.values.toList());
```

```
  // //   conn[ConnType.term]!
  // //       .collection<TermEntityDbDS>()
  // //       .putAllSync(termsToDb.values.toList());
  // //   conn[ConnType.term]!
  // //       .collection<TermAddingInfoDbDS>()
  // //       .putAllSync(addInfoTermsToDb.values.toList());
  // //   conn[ConnType.term]!
  // //       .collection<SentenceDbDS>()
  // //       .putAllSync(sentencesToDb.values.toList());
  // // });
  // // var res1 = conn[ConnType.term]!
  // //     .collection<FolderDbDS>()
  // //     .filter()
  // //     .rootFolderIsNull()
  // //     .findAllSync();
  // // // var res2 = conn[ConnType.term]!
  // // //     .collection<TermAddingInfoDbDS>()
  // // //     .where()
  // // //     .findAllSync();
  // // print(res1);
  // //
  // // await OpenAndClose3.closeConnStatic(conn);
  // // await serverTermUpdateCoro;
  // //
  // // // final folders = await mainApi.getAllFoldersFolderAllGet(headers:
authHeaders);
  // //
  // // // final modules = await mainApi.getAllModuleModuleAllGet(headers:
authHeaders);
  // // //
  // // // final terms = await mainApi.getAllTermTermAllGet(headers:
authHeaders);
}

Future<void> getUserFromServer(
  baseApiContainer,
  authHeaders,
  String username,
  String password,
  UserApiProfile? userApi,
) async {
  var conn = (await
OpenAndClose3.openConnStatic([CollectionSchema<UserDbDS>]));

  var user =
      (await
conn[ConnType.user]!.collection<UserDbDS>().getByUsername(username)
      // .filter()
      // .usernameEqualTo(username)
      // .findAll()
      // .where()
      // .findAll()
      );
  if (user == null) {
    var userApi = baseApiContainer.getUsersEntitiesApi();

    var userFromServerData =
        await userApi.readUsersMeUserMeGet(headers: authHeaders);
    user =
        UserDbDS.fromJson(userFromServerData.data as UserDTO, password,
true);
  } else if (user == userApi?.user) {
    userApi?.userChange();
  }
```

```
  userApi?.user = user;
  conn[ConnType.user]!.writeTxn(() async {
    (await conn[ConnType.user]!
        .collection<UserDbDS>()
        .put(user!..active = true));
  });

  await OpenAndClose3.closeConnStatic(conn);
}

Future getAuthHeaders(
    AuthApi realAuthApi,
    String url,
    ApiPackage baseApiContainer,
    String username,
    String password,
    UserApiProfile? userApi,
    Function(String?)? onErrorCallback) async {
  Response<Token>? authData = null;
  try {
    authData = await realAuthApi.loginForAccessTokenTokenPost(
        username: JsonObject(username), password: JsonObject(password));
  } catch (e) {
    if (onErrorCallback != null) {
      onErrorCallback(
          "На сервере ${url} пользователя с таким логином и паролем не
найдено");
    }
  } finally {
    if (authData?.statusCode == 200) {
      baseApiContainer.setBearerAuth(
          "Authorization", "Bearer
${authData?.data!.accessToken!.asString}");
      var authHeaders = {
        "Authorization": "Bearer ${authData?.data!.accessToken!.asString}"
      };

      userApi?.baseApi = baseApiContainer;
      userApi?.authHeaders = authHeaders;
      if (userApi?.user == null) {
        await getUserFromServer(
            baseApiContainer, authHeaders, username, password, userApi);
      } else {
        await networkProcessor(userApi);
      }
    } else {
      if (onErrorCallback != null) {
        onErrorCallback(
            "На сервере ${url} пользователя с таким логином и паролем не
найдено");
      }
    }
  }
}

Future testServerUrl(
    String url,
    List<bool> goodUrlFounded,
    String username,
    String password,
    UserApiProfile? userApi,
    Function(String?)? onErrorCallback) async {
  final Dio dio = Dio(BaseOptions(baseUrl: url));
  final ApiPackage baseApiContainer =
```

```
      ApiPackage(dio: dio, serializers: standardSerializers);
  final AuthApi realAuthApi = baseApiContainer.getAuthApi();

  // baseApiContainer.dio.interceptors.add(PrettyDioLogger());

  var healthyCheck = await realAuthApi.healthcheckGetHealthcheckGet();
  if (goodUrlFounded[0]) {
    return;
  }
  if (healthyCheck.statusCode == 200) {
    goodUrlFounded[0] = true;
  }
  await getAuthHeaders(realAuthApi, url, baseApiContainer, username,
password,
      userApi, onErrorCallback);
}



  // List<bool> goodUrlFounded = [false];
  //
  // if (userApi?.baseApi == null) {
  //   if (serverUrls == null) {
  //     var conn = (await OpenAndClose3.openConnStatic(
  //         [CollectionSchema<HttpUtilsDbDS>]));
  //
  //     var serverUrlsEntities = (await conn[ConnType.server_urls]!
  //         .collection<HttpUtilsDbDS>()
  //         .where()
  //         .findAll());
  //     serverUrls = [for (var i in serverUrlsEntities) i.httpUrl];
  //
  //     await OpenAndClose3.closeConnStatic(conn);
  //   } else {
  //     var conn = (await OpenAndClose3.openConnStatic(
  //         [CollectionSchema<HttpUtilsDbDS>]));
  //
  //     var serverUrlsEntities = [
  //       for (var i in serverUrls) HttpUtilsDbDS()..httpUrl = i
  //     ];
  //     conn[ConnType.server_urls]!.writeTxnSync(() =>
conn[ConnType.server_urls]!
  //         .collection<HttpUtilsDbDS>()
  //         .putAllSync(serverUrlsEntities));
  //
  //     // serverUrls = [for (var i in serverUrlsEntities) i.httpUrl];
  //
  //     await OpenAndClose3.closeConnStatic(conn);
  //   }
  //
  //   serverUrls.forEach((url) async => await testServerUrl(
  //       url, goodUrlFounded, username, password, userApi,
onErrorCallback));
  // } else if (userApi?.authHeaders == null ||
  //     (userApi?.authHeaders != null && userApi!.authHeaders.isEmpty)) {
  //   await getAuthHeaders(await userApi!.baseApi!.getAuthApi(),
"<unnamed>",
  //       userApi.baseApi!, username, password, userApi, onErrorCallback);
  // } else {
  //   await getUserFromServer(
  //       userApi?.baseApi, userApi?.authHeaders, username, password,
userApi);
  // }
```

```
//   return;
// }

Future<void> updatePersonalizedTerms(
    List<TermEntityDbDS> terms, UserApiProfile userApi) async {
  if (userApi.baseApi != null) {
    final TermEntitiesApi termApi = userApi.baseApi!.getTermEntitiesApi();
    // standardSerializers
    // standardSerializers.serialize(
    //   folders.data!.asList[0],
    //
    // )
    // String json = jsonEncode(list)
    // PersonalizeTermDTO.serializer.serialize(standardSerializers,
toDTOModel());
    // PersonalizeTermDTO.serializer.
    var data1 = [
      for (var i in terms) i.toJsonAsMap()
      // i.toDTO<UpdateOnlyPersonalizePartTermDTO>()
      // JsonObject(standardSerializers.serialize(
      //      i.toDTO<UpdateOnlyPersonalizePartTermDTO>()
      // ))
    ];
    var data2 = jsonEncode(data1);
    var data3 = JsonObject(data2);
    var res = await myFunc(
      userApi.baseApi!.dio,
      standardSerializers,
      body: data1,
      headers: userApi.authHeaders,
    );
    print(res);
  }
}

Future<Response<JsonObject>> myFunc(
  Dio _dio,
  Serializers _serializers, {
  List<Map<String, dynamic>>? body,
  CancelToken? cancelToken,
  Map<String, dynamic>? headers,
  Map<String, dynamic>? extra,
  ValidateStatus? validateStatus,
  ProgressCallback? onSendProgress,
  ProgressCallback? onReceiveProgress,
}) async {
  final _path = r'/term/personalize/create_or_update';
  final _options = Options(
    method: r'PUT',
    headers: <String, dynamic>{
      ...?headers,
    },
    extra: <String, dynamic>{
      'secure': <Map<String, String>>[
        {
          'type': 'oauth2',
          'name': 'OAuth2PasswordBearer',
        },
      ],
      ...?extra,
    },
    contentType: 'application/json',
    validateStatus: validateStatus,
  );
```

```
  dynamic _bodyData;

  try {
    _bodyData = body;
  } catch (error, stackTrace) {
    throw DioException(
      requestOptions: _options.compose(
        _dio.options,
        _path,
      ),
      type: DioExceptionType.unknown,
      error: error,
      stackTrace: stackTrace,
    );
  }

  final _response = await _dio.request<Object>(
    _path,
    data: _bodyData,
    options: _options,
    cancelToken: cancelToken,
    onSendProgress: onSendProgress,
    onReceiveProgress: onReceiveProgress,
  );

  JsonObject? _responseData;

  try {
    final rawResponse = _response.data;
    _responseData = rawResponse == null
        ? null
        : _serializers.deserialize(
            rawResponse,
            specifiedType: const FullType(JsonObject),
          ) as JsonObject;
  } catch (error, stackTrace) {
    throw DioException(
      requestOptions: _response.requestOptions,
      response: _response,
      type: DioExceptionType.unknown,
      error: error,
      stackTrace: stackTrace,
    );
  }

  return Response<JsonObject>(
    data: _responseData,
    headers: _response.headers,
    isRedirect: _response.isRedirect,
    requestOptions: _response.requestOptions,
    redirects: _response.redirects,
    statusCode: _response.statusCode,
    statusMessage: _response.statusMessage,
    extra: _response.extra,
  );
}


class UserNotActiveException implements Exception {
  UserNotActiveException();
}

class ServerUrlIsUnavailable implements Exception {
```

```
  ServerUrlIsUnavailable();
}

class AuthErrorUserPasswordNotCorrect implements Exception {
  final String url;
  final String username;
  final String password;
  AuthErrorUserPasswordNotCorrect(this.url, this.username, this.password);
}
```

Листинг М.4 – ModuleUnaryUrlControllerExt

```
import 'package:built_value/json_object.dart';
import 'package:dio/dio.dart';

import '../../api_package/lib/api_package.dart';
import '../network_errors.dart';
import 'main.dart';

extension ModuleUnaryUrlControllerExt on UnaryUrlController {

  Stream<AnsListPersonalizeModuleDTOUnionBErrorNoneType> getModules() async*
{
    await for (var respFuture in
pagination<AnsListPersonalizeModuleDTOUnionBErrorNoneType>(
        moduleApi.getAllModulePersonalizeAllGet
    , (p0, limit) => p0?.data?.data != null && p0!.data!.data!.asList.length
>= limit)){
      var resp = await respFuture;
      if (resp == null){
        throw UnimplementedError();
      }

      // var castedData = resp.data!.data as List<PersonalizeFolderDTO>;

      yield  resp.data!;
    }

    throw ServerUrlIsUnavailable();
  }
```

Листинг М.5 – url_conlrollers.dart

```
}
import 'dart:async';
import 'package:built_value/json_object.dart';
import 'package:dio/dio.dart';
import 'package:observable/observable.dart';
import
'package:the_remember/network_processor/url_controller/auth_extension.dart';
import
'package:the_remember/network_processor/url_controller/folder_extension.dart'
;
import
'package:the_remember/network_processor/url_controller/module_extension.dart'
;

import '../../api_package/lib/api_package.dart';
import '../../src/repositoris/db_data_source/http_utils.dart';
import '../../src/urils/streams.dart';
import '../network_errors.dart';
import '../new_network_processor.dart';
```

```
class UrlsController {
  final List<HttpUtilsDbDS> _urlObjects;
  late final List<String> _startedUrls;

  List<Stream Function() Function(UnaryUrlController)> segOfCommands = [
    (i) => i.getFolders,
    (i) => i.getModules,
  ];

  Map<UnaryUrlController, Function?> existsOneUrlGenerators = {};
  Map<UnaryUrlController, Future<(dynamic, UnaryUrlController, int)>?>
      currNetworkJobOfUrlControllers = {};

  /// Список, в котором содержатся активные ссылки
  final ObservableList<UnaryUrlController> _activeUrls =
      ObservableList<UnaryUrlController>();

  /// Список, в котором находятся ссылки,
  /// которые на данный момент заняты общением с сервером
  /// и не могут выполнять других операций
  late List<UnaryUrlController> _processingUrls = [];

  UrlsController(this._urlObjects) {
    _processingUrls = [
      for (var urlObject in _urlObjects) UnaryUrlController(urlObject)
    ];
    _startedUrls = [for (var urlObject in _urlObjects) urlObject.httpUrl];
  }

  Stream<UnaryUrlController> activeApis() async* {
    /// Возвращаем активные ссылки
    for (var i in _activeUrls) {
      yield i;
    }

    /// Для остальных ссылок проверяем, являются ли они активными или нет
    List<
        (
          UnaryUrlController,
          Future<(UnaryUrlController, Response<JsonObject>?)>
        )> healthCheckFutures = [
      for (var i in _processingUrls)
        if (i.healthCheckFuture != null)
          (i, i.healthCheckFuture!.then((futureRes) => (i, futureRes)))
    ];

    // late UnaryUrlController? currUrlController;

    // healthCheckFutures = [
    //   for (var i in healthCheckFutures)
    //     (
    //       i.$1,
    //       i.$2.whenComplete(() {
    //         currUrlController = healthCheckFutures
    //             .where((element) => element.$1 == i.$1)
    //             .firstOrNull
    //             ?.$1;
    //         healthCheckFutures = healthCheckFutures
    //             .where((element) => element.$1 != i.$1)
    //             .toList();
    //       })
    //     )
    // ];
```

```
    while (healthCheckFutures.isNotEmpty) {
      var (currUrlController, answer) =
          await Future.any([for (var i in healthCheckFutures) i.$2]);

      healthCheckFutures = healthCheckFutures
          .where((element) => element.$1 != currUrlController)
          .toList();

      if (UnaryUrlController.isCorrectAnswer(answer) &&
          currUrlController != null) {
        _activeUrls.add(currUrlController!);
        _processingUrls.remove(currUrlController);
        yield currUrlController!;
      }
    }
  }

  (UnaryUrlController, Future?, int) workloadRegistrar(
      UnaryUrlController urlController, int commandIndex) {
    if (existsOneUrlGenerators[urlController] == null) {
      if (segOfCommands.length < commandIndex) {
        var newGen = segOfCommands[commandIndex](urlController)();
        existsOneUrlGenerators[urlController] = nextValFromStream(newGen);
      } else {
        // TODO: Что делать, если последовательность закончилась???
        return (urlController, null, -1);
      }
    }
    Future getDataPartFuture = existsOneUrlGenerators[urlController]!();
    currNetworkJobOfUrlControllers[urlController] =
        getDataPartFuture.then((i) => (i, urlController, commandIndex));

    return (urlController, getDataPartFuture, commandIndex);
  }
  /// Return  Future<>
  Future<(dynamic, UnaryUrlController)> eventsQueryGen() async {
    var (firstFutureRes, urlController, commandIndex) = await Future.any([
      for (var future in currNetworkJobOfUrlControllers.values)
        if (future != null) future
    ]);
    workloadRegistrar(urlController, commandIndex);
    return (firstFutureRes, urlController);
  }
}

class UnaryUrlController {
  final HttpUtilsDbDS urlObj;
  late final String url;
  late Future<Response<JsonObject>?>? healthCheckFuture;
  late Dio dio;
  late ApiPackage apiContainersContainer;
  late AuthApi authApi;
  late FoldersEntitiesApi folderApi;
  late ModuleEntitiesApi moduleApi;
  late TermEntitiesApi termEntitiesApi;
  late SentenceEntitiesApi sentenceApi;
  late AdditionalInfoOfTermsApi addInfoApi;
  late TermMarksEntitiesApi termMarksApi;
  late Future<Map<String, String>>? authFuture;

  Future? get currentTask => _taskQueue.isEmpty ? null : _taskQueue[0];
  final List<Future> _taskQueue = [];
```

```
void addTaskToQueue(Future task) {
  _taskQueue.add(task);
  task.whenComplete(() {
    _taskQueue.remove(task);
  });
}

// Future? get currentTask {
//   if (healthCheckFuture != null && healthCheckFuture!)
//   return null;
// }

UnaryUrlController(this.urlObj) {
  url = urlObj.httpUrl;
  dio = Dio(BaseOptions(baseUrl: urlObj.httpUrl));
  apiContainersContainer =
      ApiPackage(dio: dio, serializers: standardSerializers);
  authApi = apiContainersContainer.getAuthApi();
  folderApi = apiContainersContainer.getFoldersEntitiesApi();
  moduleApi = apiContainersContainer.getModuleEntitiesApi();
  termEntitiesApi = apiContainersContainer.getTermEntitiesApi();
  sentenceApi = apiContainersContainer.getSentenceEntitiesApi();
  addInfoApi = apiContainersContainer.getAdditionalInfoOfTermsApi();
  termMarksApi = apiContainersContainer.getTermMarksEntitiesApi();
  healthCheckFuture =
      webProtocolDecorator(authApi.healthcheckGetHealthcheckGet());
}

Future<T?> webProtocolDecorator<T extends Response>(Future<T> networkQuery,
    {int targetStatusCode = 200}) async {
  // currentTask = networkQuery;
  var networkQueryNew = networkQuery.then((value) {
    if (value.statusCode == targetStatusCode) {
      return Future<T?>.value(value);
    }
    throw 'Bad';
  }).catchError((error, stackTrace) {
    return null;
  }, test: (error) {
    if (error is UserNotActiveException) {
      return false;
    } else if (error is DioException) {
      return true;
    }
    return (error is int && error >= 400);
  });
  addTaskToQueue(networkQueryNew);
  return networkQueryNew;
}

/// Получение данных с учетом пагинации
Stream<Future<Response<T>?>> pagination<T>(
    Future<Response<T>> Function() paginateFunc,
    bool Function(Response<T>?, int) isContinue,
    {int limit = 100}) async* {
  var offset = 0;
  Response<T>? respData;
  var authHeaders = await authFuture!;
  do {
    var rawRes = Function.apply(paginateFunc, [], {
      #limit: limit,
      #offset: offset,
      #headers: authHeaders,
      #authorOnly: false,
```

```
        #updatedAfter: urlObj.lastSync
      });
      while (_taskQueue.isNotEmpty) {
        await _taskQueue[0];
      }
      var res =
          webProtocolDecorator<Response<T>>(rawRes as Future<Response<T>>);

      yield res;
      var respData = await res;
    } while (isContinue(respData, limit));
  }

  /// Проверка, что ответ положительный
  static bool isCorrectAnswer(Response<JsonObject>? answer) {
    return answer?.data != null && (answer!.data as bool) == true;
  }
}

// typedef VarArgsCallback = Future<T> Function<T extends Response>(
//     List<dynamic> args, Map<String, dynamic> kwargs);
//
// class VarArgsFunction {
//   final VarArgsCallback callback;
//   static var _offset = 'Symbol("'.length;
//
//   VarArgsFunction(this.callback);
//
//   void call() => callback([], {});
//
//   @override
//   dynamic noSuchMethod(Invocation inv) {
//     return callback(
//       inv.positionalArguments,
//       inv.namedArguments.map(
//         (_k, v) {
//           var k = _k.toString();
//           return MapEntry(k.substring(_offset, k.length - 2), v);
//         },
//       ),
//     );
//   }
// }
```

Листинг М.6 – FolderUnaryUrlControllerExt

```
import 'package:built_value/json_object.dart';
import 'package:dio/dio.dart';

import '../../api_package/lib/api_package.dart';
import '../network_errors.dart';
import 'main.dart';

extension FolderUnaryUrlControllerExt on UnaryUrlController {

  Stream<AnsListPersonalizeFolderDTOUnionBErrorNoneType> getFolders() async*
{
    await for (var respFuture in
pagination<AnsListPersonalizeFolderDTOUnionBErrorNoneType>(
        folderApi.getAllFoldersPersonalizeAllGet
    , (p0, limit) => p0?.data?.data != null && p0!.data!.data!.asList.length
>= limit)){
```

```
      var resp = await respFuture;
      if (resp == null){
        throw UnimplementedError();
      }

      // var castedData = resp.data!.data as List<PersonalizeFolderDTO>;

      yield  resp.data!;
    }

    throw ServerUrlIsUnavailable();
  }

}
```

Листинг М.7 – AuthUnaryUrlControllerExt

```
import 'package:built_value/json_object.dart';
import 'package:dio/dio.dart';

import '../network_errors.dart';
import 'main.dart';

extension AuthUnaryUrlControllerExt on UnaryUrlController {

  Future<Map<String, String>> auth(String username, String password) async {
    var ans = await healthCheckFuture!;
    if (UnaryUrlController.isCorrectAnswer(ans)){
      authApi ??= apiContainersContainer.getAuthApi();
      authFuture = getAuthHeaders(username, password);
      return authFuture!;
    }
    throw ServerUrlIsUnavailable();
  }

  Future<Map<String, String>> getAuthHeaders(String username, String
password) async {

    var authData = await
webProtocolDecorator(authApi.loginForAccessTokenTokenPost(
        username: JsonObject(username), password: JsonObject(password)
    ));
    if (authData?.statusCode == 200) {
      var authHeaders = {
        "Authorization": "Bearer ${authData?.data!.accessToken!.asString}"
      };
      return authHeaders;
    }
    else if  (authData?.statusCode == 401) {
      throw AuthErrorUserPasswordNotCorrect(url, username, password);
    }
    throw ServerUrlIsUnavailable();
  }
}
```

Листинг М.8 – DbUtils

```
import 'package:isar/isar.dart';
```

```
import
'package:the_remember/db_layer/network/entities_update/update_controller.dart
';

import '../network_processor/network_errors.dart';
import '../src/repositoris/db_data_source/user.dart';
import '../src/urils/db/dbMixins.dart';

class DbUtils with OpenAndClose, NetworkUpdateProcessor{
  UserDbDS? _activeUser;

  UserDbDS get activeUser {
    if (_activeUser == null) {
      throw UserNotActiveException();
    }
    return _activeUser!;
  }
  set activeUser(UserDbDS? value) {
    _activeUser = value;
  }

  Future<void> findActiveUserInDb(Isar isar) async {
    // await Future.delayed(Duration(seconds: 1000));
    // IsarCollectionImpl<UserDbDS> userQu = isar.collection<UserDbDS>();
    // userQu.wh
    // userQu.
    List<UserDbDS> activeUsers = (await isar
        .collection<UserDbDS>()
    // .where()
        .filter()
        .activeEqualTo(true)
        .findAll());
    activeUsers = activeUsers.where((element) => element.active).toList();
    if (activeUsers.length == 1 && activeUsers[0].active) {
      activeUser = activeUsers[0];
    } else {
      activeUser = null;
    }
  }


}
```

Листинг М.9 – GetConnUrlsProcessorExt

```
import 'package:isar/isar.dart';

import '../../network_processor/url_controller/main.dart';
import '../../src/repositoris/db_data_source/http_utils.dart';
import '../db_utils.dart';

extension GetConnUrlsProcessorExt on DbUtils {

  Future<(Stream<UnaryUrlController> Function(), UrlsController)>
getConnectionUrls(
      Isar isar) async {
    var serverUrlsEntities =
    (await isar.collection<HttpUtilsDbDS>().where().findAll());
    // var user = (await conn[ConnType.user]!
    //     .collection<UserDbDS>()
```

```
    //     .getByUuid(userApi!.user!.uuid));
    var urlsController = UrlsController(serverUrlsEntities);

    return (urlsController.activeApis, urlsController);
    // var futures = [for (var i in serverUrls) webProtocolDecorator() ]
    // serverUrls.forEach((url) async => await testServerUrl(
    //     url, goodUrlFounded, username, password, userApi,
onErrorCallback));
  }
}
```

Листинг М.10 – NetworkUpdateProcessor

```
import
'package:the_remember/db_layer/network/entities_update/folder_update.dart';

import '../../../network_processor/url_controller/main.dart';
import '../../../src/urils/db/abstract_entity.dart';
import '../../db_utils.dart';

enum DbEntEnum { folder, module, term, sentence, addInfo, learnMark }

mixin NetworkUpdateProcessor {
  final Map<DbEntEnum, List<(Future, UnaryUrlController)>> updateSeqGraph = {
    DbEntEnum.folder: [],
    DbEntEnum.module: [],
    DbEntEnum.term: [],
    DbEntEnum.sentence: [],
    DbEntEnum.addInfo: [],
    DbEntEnum.learnMark: [],
  };

  final Map<DbEntEnum, List<DbEntEnum>> updateDepGraph = {
    DbEntEnum.folder: [],
    DbEntEnum.module: [DbEntEnum.folder],
    DbEntEnum.term: [DbEntEnum.module],
    DbEntEnum.sentence: [DbEntEnum.term],
    DbEntEnum.addInfo: [DbEntEnum.term],
    DbEntEnum.learnMark: [DbEntEnum.term],
  };
}

extension NetworkUpdateProcessorExt on DbUtils {
  Future<void> registerDbUpdateOperation(DbEntEnum entType,
      UnaryUrlController currUrl, Stream dataFromNetwork) async {
    await Future.wait([
      for (var depKey in updateDepGraph[entType]!)
        for (var future in updateSeqGraph[depKey]!)
          if (future.$2 == currUrl) future.$1
    ]);

    if (entType == DbEntEnum.folder) {
      updateSeqGraph[entType]!.add((folderUpdate(), currUrl));
      // ;
    }


  }
}
```

Листинг М.11 – FolderNetworkProcessorExt

```
import 'package:built_value/json_object.dart';
```

```
import 'package:built_value/serializer.dart';
import 'package:dio/dio.dart';
import 'package:isar/isar.dart';
import
'package:the_remember/network_processor/url_controller/folder_extension.dart'
;

import '../../../api_package/lib/api_package.dart';
import '../../../network_processor/network_errors.dart';
import '../../../network_processor/url_controller/main.dart';
import '../../../src/repositoris/db_data_source/folder.dart';
import '../../db_utils.dart';

extension FolderNetworkProcessorExt on DbUtils {
  Future<Map<String, String>> folderUpdate(
      AnsListPersonalizeFolderDTOUnionBErrorNoneType foldersPack) async {
    Map<String, FolderDbDS> entFromWeb = {};
    Isar isar = await openConn();
    // Получение обновлений с серверов
    // и слияние этих изменений с тем, что уже есть в БД
    // await for (var conn in urlsGen()) {
    //   print(conn.url);
    //   await for (var foldersPack in conn.getFolders()) {
    var newEnt = {
      for (var ent in [
        for (var folder in foldersPack.data!.asList)
          FolderDbDS.fromJson(standardSerializers.deserialize(folder,
                  specifiedType: const FullType(PersonalizeFolderDTO))
              as PersonalizeFolderDTO)
      ])
        ent.uuid: FolderDbDS.entityUpdateOrCreate(ent, entFromWeb)
    };

    var foldersFromDb = {
      for (var ent in await isar
          .collection<FolderDbDS>()
          .filter()
          .userUuidEqualTo(activeUser.uuid)
          .anyOf([
        for (var i in newEnt.keys)
          if (!entFromWeb.containsKey(i)) i
      ], (q, element) => q.uuidEqualTo(element)).findAll())
        ent.uuid: FolderDbDS.entityUpdateOrCreate(ent, newEnt)
    };
    newEnt.addAll(foldersFromDb);
    entFromWeb.addAll(newEnt);
    // }
    // }

    // Установка rootFolderUuid
    Map<String, List<FolderDbDS>> rootFoldersIds = {};
    for (var ent in entFromWeb.values) {
      if (ent.rootFolderUuid != null && ent.rootFolder.value == null) {
        if (entFromWeb[ent.rootFolderUuid] != null) {
          ent.rootFolder.value = entFromWeb[ent.rootFolderUuid]!;
        } else {
          rootFoldersIds[ent.rootFolderUuid!] ??= [];
          rootFoldersIds[ent.rootFolderUuid!]!.add(ent);
        }
      }
    }
    var rootFoldersFromDb = await isar
        .collection<FolderDbDS>()
        .filter()
```

```
        .userUuidEqualTo(activeUser.uuid)
        .anyOf(rootFoldersIds.keys, (q, element) => q.uuidEqualTo(element))
        .findAll();
    for (var rootFolder in rootFoldersFromDb) {
      for (var folder in rootFoldersIds[rootFolder.uuid] ?? []) {
        folder.rootFolder.value = rootFolder;
      }
    }

    // Обновление сущностей в БД
    // TODO: uncomment
    // isar.writeTxnSync(() {
    //
isar.collection<FolderDbDS>().putAllSync(entFromWeb.values.toList());
    // });
    print(entFromWeb);

    // Отправка изменений на сервер

    // var newData = conn.
    await closeConn();
    throw ServerUrlIsUnavailable();
  }
}
```

Листинг М.12 – utils.dart

```
import 'dart:async';
import 'dart:async';


Future<T>? Function() nextValFromStream<T>(Stream<T> s) {
  int _nextValFromStreamPointer = 0;

  Future<T>? _nextValFromStream() {
    Completer<T> c = new Completer<T>();
    try {
      s.elementAt(_nextValFromStreamPointer++).then(c.complete);
      return c.future;
    } on RangeError{
      return null;
    }
  }

  return _nextValFromStream;
}
```

Листинг М.13 – ModChangeNotifier

```
import 'package:flutter/cupertino.dart';

import '../db/dbMixins.dart';

abstract class ModChangeNotifier extends ChangeNotifier with OpenAndClose
{

  bool _isDispose = false;
  Map<Type, ModChangeNotifier?> subProvidersList = {};
  Map<Type, ModChangeNotifier?> parentProvidersList = {};

  ModChangeNotifier(){
    init(isRealInit: true);
```

```
  }

  @mustCallSuper
  void init({bool isRealInit = false}){
    for(var i in parentProvidersList.entries){
      i.value?.subProvidersList[this.runtimeType] = this;
    }
    if (!isRealInit) {
      notifyListeners();
    }
  }

  @override
  void dispose() {
    _isDispose = true;
    for(var i in parentProvidersList.entries){
      i.value?.subProvidersList[this.runtimeType] = null;
    }
    super.dispose();
    // Type ddd = this.runtimeType;
  }

  @override
  void notifyListeners(){
    if (!_isDispose) {
      super.notifyListeners();
      for(var i in subProvidersList.entries){
        i.value?.init();
      }
    }
  }

}
```

Листинг М.14 – IsolateMessageListener

```
import 'package:collection/collection.dart';

import '../../domain_layer/providers/isolates/network.dart';
import 'base_msg.dart';

class IsolateMessageListener{
  static final IsolateMessageListener instance =
IsolateMessageListener._internal();

  IsolateMessageListener._internal();

  factory IsolateMessageListener() {
    return instance;
  }

  final Map<CrossIsolatesMessageType, List<QueueList<CrossIsolatesMessage>>>
queries =
  {for (var i in CrossIsolatesMessageType.values) i: []};

  Future<void> mainListener(dynamic message)async{
    CrossIsolatesMessage incomingMessage = message as CrossIsolatesMessage;
    for (QueueList q in queries[incomingMessage.msgType] ?? <QueueList>[]){
      q.addLast(message);
    }
  }

  (Stream<CrossIsolatesMessage<T>>, List<bool>) getMessageStream<T>(
```

```
      CrossIsolatesMessageType targetMsType) {
    List<bool> killerFlag = [false];

    Stream<CrossIsolatesMessage<T>> _getMessageStream(
        CrossIsolatesMessageType targetMsType) async* {
      QueueList<CrossIsolatesMessage> createdNewList = QueueList<
          CrossIsolatesMessage>();
      queries[targetMsType] ??= [];
      queries[targetMsType]!.add(createdNewList);

      while (!killerFlag[0]) {
        for (var i in createdNewList) {
          yield (i as CrossIsolatesMessage<T>);
        }
        await Future.delayed(const Duration(seconds: 1));
      }
      queries[targetMsType]?.remove(createdNewList);
    }

    return (_getMessageStream(targetMsType), killerFlag);

  }
```

Листинг М.15 – IzarManager

```
import 'dart:async';

import 'package:flutter/services.dart';
import 'package:isar/isar.dart';
import 'package:path_provider/path_provider.dart';
import 'package:the_remember/src/repositoris/db_data_source/sentence.dart';
import
'package:the_remember/src/repositoris/db_data_source/term_adding_info.dart';
import 'package:the_remember/src/repositoris/db_data_source/user.dart';
import 'package:the_remember/src/urils/db/abstract_entity.dart';

import '../../repositoris/db_data_source/folder.dart';
import '../../repositoris/db_data_source/http_utils.dart';
import '../../repositoris/db_data_source/module.dart';
import '../../repositoris/db_data_source/term.dart';

List<CollectionSchema<AbstractEntity>> allSchemes = [
  TermEntityDbDSSchema,
  ModuleDbDSSchema,
  FolderDbDSSchema,
  UserDbDSSchema,
  HttpUtilsDbDSSchema,
  TermAddingInfoDbDSSchema,
  SentenceDbDSSchema
];

enum ConnType { module, folder, term, user, server_urls, term_info }

class IzarManager {
  static final IzarManager instance = IzarManager._internal();
  final Map<String, int> _izarCounter = <String, int>{};
  Future<Isar>? currOpenOfConnection;
  late Future<String> getDbPath;
  late String dbExternalPath = "";


  factory IzarManager() {
    return instance;
```

```
}

IzarManager._internal();

Future<Isar> _openIsar<T>(
    String name, List<CollectionSchema<T>> schemes) async {
  final izar = Isar.getInstance(name);

  if (izar != null && izar.isOpen) {
    final count = _izarCounter[name] ?? 1;
    _izarCounter[name] = count + 1;
    return izar;
  }

  if (_izarCounter[name] == null || _izarCounter[name] == 0) {
    _izarCounter[name] = 1;
  } else {
    _izarCounter[name] = (_izarCounter[name] ?? 0) + 1;
  }
  // ServicesBinding ;
  if (ServicesBinding.rootIsolateToken == null){
    getDbPath = (() async => dbExternalPath)();
  } else {
    getDbPath =
        getApplicationDocumentsDirectory().then((value) => value.path);
  }
  final dir = await getDbPath;
  return Isar.open(
    schemes,
    directory: dir,
    name: name,
    inspector: true,
  );
}

Future<void> closeIsar<T>(Isar izar) async {
  if (!izar.isOpen) {
    _izarCounter.remove(izar.name);
    return;
  }

  var count = _izarCounter[izar.name] ?? 1;
  count -= 1;
  _izarCounter[izar.name] = count;
  if (count > 0) return;

  await izar.close();
}

Future<Isar> openActivityDB(ConnType connName) async {
  // Future<Isar> res;
  // if (currOpenOfConnection != null) {
  //   res = currOpenOfConnection!;
  // } else {
  //   currOpenOfConnection = _openIsar('term', allSchemes);
  //   res = currOpenOfConnection!;
  //   currOpenOfConnection!.then((value) {
  //     this.currOpenOfConnection = null;
  //     return value;
  //   });
  // }
  //
  // return res;
  return openActivityDBv2([]);
```

```
  }

  Future<Isar> openActivityDBv2(
      List<CollectionSchema<AbstractEntity>> connSchemes) async {
    Future<Isar> res;
    if (currOpenOfConnection != null) {
      res = currOpenOfConnection!;
    } else {
      currOpenOfConnection = _openIsar('term', allSchemes);
      res = currOpenOfConnection!;
      currOpenOfConnection!.then((value) {
        this.currOpenOfConnection = null;
        return value;
      });
    }

    return res;
  }
}

}
```

Листинг М.16 – OpenAndCloseConnection

```
import 'package:isar/isar.dart';
import 'package:the_remember/src/repositoris/db_data_source/user.dart';

import '../../repositoris/db_data_source/folder.dart';
import '../../repositoris/db_data_source/http_utils.dart';
import '../../repositoris/db_data_source/module.dart';
import '../../repositoris/db_data_source/term.dart';
import '../../repositoris/db_data_source/term_adding_info.dart';
import 'abstract_entity.dart';
import 'engine.dart';




abstract class OpenAndCloseI{
  Future<void> openConn();
  Future<void> closeConn();

}

abstract class GetDataFromDbI{
  void completionData();
}


abstract mixin class OpenAndClose3<
T1 extends CollectionSchema<AbstractEntity>,
T2 extends CollectionSchema<AbstractEntity>,
T3 extends CollectionSchema<AbstractEntity>
>  {
  late Map<ConnType, Isar> conn = Map();
  int counter = 0;
  var classes = [T1, T2, T3];

  Future<void> openConn() async {
    // FIXME: Если кто-то сторонний откроет
    //  второй коннекшн одного и того же типа
    //  во время работы первого
    //  (стандартного коннекшена, который открыл сам класс),
```

```
    //  то не понятно что делать


    for (var kv in (await openConnStatic(classes)).entries){
      if (conn.containsKey(kv.key)){
        throw "Повторное откртие коннекшена в одном и том же классе";
      }
      conn[kv.key] = kv.value;
    }

  }

  Future<void> closeConn() async {
    await closeConnStatic(conn);
    conn.clear();
    // for(var kv in conn.entries){
    //   conn.remove(kv.key);
    // }


  }

  static Future<Map<ConnType, Isar>> openConnStatic(List<Type> classes) async
{
    Map<ConnType, Isar> locConn = Map();


    var  ddd  = classes.map((cl) async {

      var currConnType = switch (cl) {
        const (CollectionSchema<FolderDbDS>) => ConnType.folder,
        const (CollectionSchema<ModuleDbDS>) => ConnType.module,
        const (CollectionSchema<TermEntityDbDS>) => ConnType.term,
        const (CollectionSchema<UserDbDS>) => ConnType.user,
        const (CollectionSchema<HttpUtilsDbDS>) => ConnType.server_urls,
        const (CollectionSchema<TermAddingInfoDbDS>) => ConnType.term_info,
        const (CollectionSchema<AbstractEntity>) => null,
        _ => throw 'this type is not in in switch construction!',
      };
      if (currConnType != null) {
        locConn[currConnType] = await
IzarManager.instance.openActivityDB(currConnType);
        return  locConn[currConnType];
      }
      return null;

    });
    for(var v in ddd){
      await v;
    }
    return locConn;
  }

  static Future<void> closeConnStatic(Map<ConnType, Isar> conn) async {
    conn.entries.forEach((kv) async {
      final (key, unary_conn) = (kv.key, kv.value);
      await IzarManager.instance.closeIsar(unary_conn);
    });

  }


}
```

```
abstract mixin class OpenAndClose  {
  Isar? _conn = null;
  int _connCount = 0;
  Future<Isar>? _getConnCoro = null;
  Future<void>? _closeConnCoro = null;


  List<CollectionSchema<AbstractEntity>> get classes => [];

  Future<Isar> openConn() async {
    if (_closeConnCoro != null){
      await _closeConnCoro;
    }
    _connCount += 1;

    if (_getConnCoro != null){
      _conn = await _getConnCoro;
    } else if (_conn  == null) {
      _getConnCoro = IzarManager.instance.openActivityDBv2(this.classes);
      _conn = await _getConnCoro;
      _getConnCoro = null;
    }
    return _conn!;
  }

  Future<void> closeConn() async {
    if (_getConnCoro != null){
      await _getConnCoro;
    }
    _connCount -= 1;
    if (_connCount < 1) {
      _closeConnCoro = IzarManager.instance.closeIsar(_conn!);
      _conn = null;
      await _closeConnCoro;
      _closeConnCoro = null;
    }
  }
```

Листинг М.17 – words_BO.dart

```
import 'dart:math';

import 'package:flutter/cupertino.dart';
import 'package:isar/isar.dart';
import 'package:quiver/iterables.dart';
import 'package:the_remember/src/repositoris/db_data_source/module.dart';

import '../../../main.dart';
import '../../../network_processor/network_main.dart';
import '../../repositoris/db_data_source/term.dart';
import '../../repositoris/db_data_source/term_adding_info.dart';
import
'../../ui/ui_templates/pages/main/folders/module/learn/write_field_template.d
art';
import '../../urils/db/dbMixins.dart';
import '../../urils/db/engine.dart';

// import '../providers/folders/module/terms_in_module.dart';
import
'../providers/main/folders/module/learning/write/write_word_navigation.dart';
import '../providers/main/folders/module/terms_in_module.dart';
```

```
import '../providers/user_api_provider.dart';
// import 'package:quiver';

Future<void> startLearning(
    TermsInModuleProvider termPr, UserApiProfile userPr) async {
  await termPr.initLists;
  var currUpdateTime = DateTime.now();
  var min_ = 2;
  var max_ = termPr.termsList![0].module.value!.minWatchCount;
  for (var w in termPr.termsList!) {
    var chooseCount = min_ + random.nextInt(max_ - min_);
    w.chooseErrorCounter = chooseCount;
    w.writeErrorCounter = max_ - chooseCount;
    w.choiceNegErrorCounter = 0;
    w.watchCount = 0;
    w.personalUpdatedAt = currUpdateTime;
  }
  Map<int, TermEntityDbDS> currTermMap = {
    for (var i in termPr.termsList!) i.isarId: i
  };
  var networkFuture = updatePersonalizedTerms(termPr.termsList!, userPr);

  var conn =
      (await
OpenAndClose3.openConnStatic([CollectionSchema<TermEntityDbDS>]))
          .values
          .toList()[0];
  var wordsInCurrModule =
      await
conn.collection<TermEntityDbDS>().getAll(currTermMap.keys.toList());
  for (var w in wordsInCurrModule) {
    var updateTerm = currTermMap[w!.isarId]!;
    w
      ..chooseErrorCounter = updateTerm.chooseErrorCounter
      ..writeErrorCounter = updateTerm.writeErrorCounter
      ..choiceNegErrorCounter = updateTerm.choiceNegErrorCounter
      ..personalUpdatedAt = updateTerm.personalUpdatedAt;
  }
  // var wordsInCurrModule = await conn
  //     .collection<TermEntityDbDS>()
  //     .filter()
  // .
  //     .module((q) => q.isarIdEqualTo(termPr.u))
  //     .findAll();

  await conn.writeTxn(() async {
    // var wordsInCurrModule = await conn
    //     .collection<TermEntityDbDS>()
    //     .filter()
    //     .module((q) => q.isarIdEqualTo(moduleId))
    //     .findAll();

    await conn.collection<TermEntityDbDS>().putAll([
      for (var i in wordsInCurrModule)
        if (i != null) i
    ]);
  });
  // var wordsInCurrModule1 = await conn
  //     .collection<TermEntityDbDS>()
  //     .filter()
  //     .module((q) => q.isarIdEqualTo(moduleId))
  //     .findAll();
  await OpenAndClose3.closeConnStatic({ConnType.term: conn});
  // .putByIndex(indexName, object);
```

```
  // .where().

  // .
  //     .put(object)
  await networkFuture;
}

Future<List<TermEntityDbDS>> getAllTermsFromModule(int moduleId) async {
  var conn =
      (await
OpenAndClose3.openConnStatic([CollectionSchema<TermEntityDbDS>]))
          .values
          .toList()[0];
  var wordsInCurrModule = await conn
      .collection<TermEntityDbDS>()
      .filter()
      .module((q) => q.isarIdEqualTo(moduleId))
      .findAll();
  await OpenAndClose3.closeConnStatic({ConnType.term: conn});
  return wordsInCurrModule;
}

Future<void> learnTransactionCompleted(
  List<TermEntityDbDS> learnedData,
  UserApiProfile userApi, {
  bool learnFinished = false,
}) async {
  var updateFuture = null;

  // for(var i in learnedData){
  //   i
  //    ;
  // }
  var currUpdateTime = DateTime.now();
  Map<int, TermEntityDbDS> currTermMap = {
    for (var i in learnedData) i.isarId: i..personalUpdatedAt =
currUpdateTime
  };

  var currentModule = learnedData.isEmpty? null:
learnedData[0]?.module.value;

  if (learnFinished) {
    currentModule!.isLearnt = true;
    updateFuture = Future.wait([
      updatePersonalizedTerms(currTermMap.values.toList(), userApi)
  ] + (learnFinished? [(() async { return null;})()]: []));
  } else {
    updateFuture = updatePersonalizedTerms(currTermMap.values.toList(),
userApi);
  }

  // var networkFuture = updatePersonalizedTerms(termPr.termsList!, userPr);

  var conn =
      (await
OpenAndClose3.openConnStatic([CollectionSchema<TermEntityDbDS>]))
          .values
          .toList()[0];

  var wordsInCurrModule =
      await
conn.collection<TermEntityDbDS>().getAll(currTermMap.keys.toList());
  print(wordsInCurrModule);
```

```
  print(wordsInCurrModule.length);
  for (var w in wordsInCurrModule) {
    var updateTerm = currTermMap[w!.isarId]!;
    print("watchCount ${w.watchCount} ${updateTerm.watchCount}");
    print("chooseErrorCounter ${w.chooseErrorCounter}
${updateTerm.chooseErrorCounter}");
    print("writeErrorCounter ${w.writeErrorCounter}
${updateTerm.writeErrorCounter}");
    print("choiceNegErrorCounter ${w.choiceNegErrorCounter}
${updateTerm.choiceNegErrorCounter}");
    print("personalUpdatedAt ${w.personalUpdatedAt}
${updateTerm.personalUpdatedAt}");
    w
      ..chooseErrorCounter = updateTerm.chooseErrorCounter
      ..writeErrorCounter = updateTerm.writeErrorCounter
      ..choiceNegErrorCounter = updateTerm.choiceNegErrorCounter
      ..watchCount = updateTerm.watchCount
      ..personalUpdatedAt = updateTerm.personalUpdatedAt;
  }

  ModuleDbDS? currDbModule = null;
  if (learnFinished) {
    currDbModule = await conn.collection<ModuleDbDS>().getByComplexIndex(
        [currentModule!.uuid, currentModule.userUuid]);
    currDbModule!.isLearnt = true;
  }

  await conn.writeTxn(() async {
    // var wordsInCurrModule = await conn
    //     .collection<TermEntityDbDS>()
    //     .filter()
    //     .module((q) => q.isarIdEqualTo(moduleId))
    //     .findAll();

    await conn.collection<TermEntityDbDS>().putAll([
      for (var i in wordsInCurrModule)
        if (i != null) i
    ]);
    if (currDbModule != null && learnFinished) {
      await conn.collection<ModuleDbDS>().putByComplexIndex(currDbModule);
    }
  });

  var wordsInCurrModuleCheck =
  await conn.collection<TermEntityDbDS>().getAll(currTermMap.keys.toList());
  for (var w in wordsInCurrModuleCheck) {
    var updateTerm = currTermMap[w!.isarId]!;
    print("watchCount ${w.watchCount} ${updateTerm.watchCount}");
    print("chooseErrorCounter ${w.chooseErrorCounter} ${updateTerm
        .chooseErrorCounter}");
    print("writeErrorCounter ${w.writeErrorCounter} ${updateTerm
        .writeErrorCounter}");
    print("choiceNegErrorCounter ${w.choiceNegErrorCounter} ${updateTerm
        .choiceNegErrorCounter}");
    print("personalUpdatedAt ${w.personalUpdatedAt} ${updateTerm
        .personalUpdatedAt}");
  }

  // await conn.writeTxn(() async {
  //   await conn.collection<TermEntityDbDS>().putAll(learnedData);
  // });

  await OpenAndClose3.closeConnStatic({ConnType.term: conn});
  if (updateFuture != null) {
```

```
    await updateFuture!;
  }
}

void choiceWordChanging(
  TermEntityDbDS askedWord,
  TermEntityDbDS choiceWord,
  List<TermEntityDbDS> choiceList,
  TermsInModuleProvider termsPr,
) {
  askedWord.watchCount += 1;
  if (askedWord == choiceWord) {
    askedWord.chooseErrorCounter -= 1;

    termsPr.changedInLearningIterationTermsList!.add(askedWord);
  } else {
    askedWord.chooseErrorCounter += 1;
    if (choiceWord.watchCount > 0) {
      choiceWord.chooseErrorCounter += 1;
      choiceWord.watchCount += 1;
    }
    // choiceWord.choiceNegErrorCounter += 1;
    termsPr.changedInLearningIterationTermsList!
        .addAll([askedWord, choiceWord]);
  }

}

void writeWordChanging(
  TermEntityDbDS askedWord,
  String? inputWord,
  List<TermEntityDbDS> allModuleTerms,
  TermsInModuleProvider termsPr,
) {
  if (inputWord == null ||
      askedWord.maybeReverseDefinitionWrite.toLowerCase() ==
          inputWord!.toLowerCase()) {
    askedWord.writeErrorCounter -= 1;
  } else {
    askedWord.writeErrorCounter += 1;
    if (askedWord.writeErrorCounter % 3 == 0) {
      askedWord.chooseErrorCounter += 1;
    }
  }
  askedWord.watchCount += 1;
  termsPr.changedInLearningIterationTermsList!.add(askedWord);
}

List<TermEntityDbDS> getChoiceDefinitions(
  TermEntityDbDS targetWordEntity,
  bool reverseTerm,
  List<TermEntityDbDS> termsList,
) {
  var definitionDataPre = termsList;
  definitionDataPre.shuffle();

  var definitionData = [
    for (var ww in definitionDataPre)
      if (ww.isarId != targetWordEntity.isarId &&
          (targetWordEntity.maybeReverseDefinitionChoice != ww.term &&
              targetWordEntity.maybeReverseDefinitionChoice !=
ww.definition))
        ww
  ];
```

```
  definitionData.sort((term1, term2) {
    return term2.watchCount.compareTo(term1.watchCount);
  });
  definitionData = definitionData.sublist(
      0, targetWordEntity.module.value!.choicesCount - 1);
  definitionData.add(targetWordEntity);
  definitionData.shuffle();
  return definitionData;
}

List<TermEntityDbDS> getOneLearnIterationList(
  List<TermEntityDbDS> termsList,
) {
  var currTermList = [
    for (var w in termsList)
      if ((w.writeErrorCounter > 0 || w.chooseErrorCounter > 0)) w
  ];
  if (currTermList.length == 0) {
    return currTermList;
  }

  int targetIterationLen = currTermList[0].module.value!.maxIterationLen;
  int knowTargetIterationLen = (targetIterationLen.toDouble() *
          currTermList[0].module.value!.knownTermPart.toDouble() /
          100.0)
      .round();
  int unknownTargetIterationLen = targetIterationLen -
knowTargetIterationLen;

  var currLearntTermList = [
    for (var w in termsList)
      if ((w.writeErrorCounter <= 0 && w.chooseErrorCounter <= 0)) w
  ];

  int realUnknownIterationLen = targetIterationLen -
      min([currLearntTermList.length, knowTargetIterationLen])!;
  knowTargetIterationLen = targetIterationLen - realUnknownIterationLen;

  currTermList.sort((term1, term2) {
    if (term1.watchCount == 0) {
      return -1;
    } else if (term2.watchCount == 0) {
      return 1;
    }
    if (term1.chooseErrorCounter == term2.chooseErrorCounter) {
      return term2.writeErrorCounter.compareTo(term1.writeErrorCounter);
    }
    return term2.chooseErrorCounter.compareTo(term1.chooseErrorCounter);
  });
  if (currTermList.length >= realUnknownIterationLen) {
    currTermList = currTermList.sublist(0, realUnknownIterationLen);
  } else {
    // knowTargetIterationLen += ((realUnknownIterationLen -
currTermList.length)).round();
    knowTargetIterationLen = (currTermList.length.toDouble() *
            currTermList[0].module.value!.knownTermPart.toDouble() /
            100.0)
        .round();
    knowTargetIterationLen = max([
      knowTargetIterationLen,
      currTermList[0].module.value!.minIterationLen - currTermList.length
    ])!;
    targetIterationLen = currTermList.length;
  }
```

```
  if (currLearntTermList.length >= knowTargetIterationLen) {
    currLearntTermList = currLearntTermList.sublist(0,
knowTargetIterationLen);
  }

  for (var i in currLearntTermList) {
    var v = random.nextDouble() < 0.8;
    i
      ..chooseErrorCounter = v ? 1 : 0
      ..writeErrorCounter = v ? 0 : 1;
  }

  currTermList += currLearntTermList;
  currTermList.shuffle();

  return currTermList;
}

List<Widget> getOneWriteFieldInLearnProcess(
  TermAddingInfoDbDS? addTermInfo,
  TermEntityDbDS currentTerm,
  int fieldsCount,
  int currentFieldIndex,
  WriteWordNavigationProvider wwNavPr,
  List<GetTermSourceOrFormName> sourceOrFormNameWidgetList,
) {
  List<Widget> res = [];
  List<WriteWordRes> userInputsContainer = [];

  if (addTermInfo != null) {
    var helpPhrasesWithoutWordFirst = currentTerm.addInfoEntities
        .where((element) =>
                (element.addInfoType == AddInfoType.help_phrase_without_word
&&
                    element.addingTextData == 'at_first')
            // && (fieldsCount == 1 ||
            // element.parentAddInfoUuid == addTermInfo!.uuid)
            )
        .map((e) => GetTextPart("(" + (e.textData ?? "???") + ")"))
        .toList();
    res += helpPhrasesWithoutWordFirst;

    var helpPhrasesWithWord = currentTerm.addInfoEntities
        .where((element) =>
            (element.addInfoType == AddInfoType.help_phrase_with_word)
            // && (fieldsCount == 1 ||
            // element.parentAddInfoUuid == addTermInfo!.uuid)
            &&
            ((("...".allMatches(element.addingTextData!).length == 1) &&
                    (currentTerm.addInfoEntities.every((element) =>
                        element.addInfoType != AddInfoType.composite_word)))
||
                (("...".allMatches(element.addingTextData!).length ==
                    currentTerm.addInfoEntities
                        .where((element) =>
                            element.addInfoType ==
AddInfoType.composite_word)
                        .length))))
        .toList();
    helpPhrasesWithWord.shuffle(random);
    if (helpPhrasesWithWord.isNotEmpty) {
      var str =
          (' ' + helpPhrasesWithWord[0].addingTextData! + ' ').split("...");
      if (str[0].trim().isNotEmpty) {
```

```
        res += [GetTextPart(str[0])];
        str.removeAt(0);
        str += [''];
      } else {
        str.removeAt(0);
        str += [''];
      }
      // if (currentTerm.addInfoEntities.any((element) => element.addInfoType
== AddInfoType.composite_word))

      List<TermAddingInfoDbDS> termsList;
      if (currentTerm.addInfoEntities.any(
          (element) => element.addInfoType == AddInfoType.composite_word)) {
        termsList = currentTerm.addInfoEntities
            .where(
                (element) => element.addInfoType ==
AddInfoType.composite_word)
            .toList();
      } else {
        termsList = currentTerm.addInfoEntities
            .where((element) => ([
                  AddInfoType.usual_term,
                  AddInfoType.other_form,
                  AddInfoType.abbreviation,
                ].contains(element.addInfoType) &&
                element == addTermInfo))
            .toList();
      }

      for (var (index, data) in zip([str, termsList]).indexed) {
        var s = data[0] as String;
        var source = data[1] as LearnWriteEntity;
        res += [
          GetInputFieldPart(s, index, source, wwNavPr, userInputsContainer,
              fieldsCount, currentFieldIndex)
        ];
        if (s.trim().isNotEmpty) {
          res += [GetTextPart(s)];
        }
      }
    } else {
      res += [
        GetInputFieldPart(addTermInfo.textData!, 0, addTermInfo, wwNavPr,
            userInputsContainer, fieldsCount, currentFieldIndex)
      ];
    }

    var helpPhrasesWithoutWordEnd = currentTerm.addInfoEntities
        .where((element) =>
                (element.addInfoType == AddInfoType.help_phrase_without_word
&&
                    (element.addingTextData == 'at_end' ||
                        element.addingTextData == 'any'))
            // &&   (fieldsCount == 1 ||
            // element.parentAddInfoUuid == addTermInfo!.uuid)
            )
        .map((e) => GetTextPart("(" + (e.textData ?? "???") + ")"))
        .toList();

    res += helpPhrasesWithoutWordEnd;
  } else {
    res += [
      GetInputFieldPart(currentTerm.maybeReverseDefinitionWrite, 0,
currentTerm,
```

```
          wwNavPr, userInputsContainer, fieldsCount, currentFieldIndex)
    ];
  }

  var sourceOrFormName = sourceOrFormNameWidgetList.isEmpty
      ? ""
      : sourceOrFormNameWidgetList[0].text;
  var helpString = [for (var i in res) (i is GetTextPart) ? i.text : null];
  var stringKey =
      (sourceOrFormName ?? "") + [for (var i in helpString) i ??
'@@@'].join();
  var targetStrings = [
    for (var i in res)
      if (i is GetInputFieldPart) i.targetString
  ];
  var sourceEntity = [
    for (var i in res)
      if (i is GetInputFieldPart) i.currentLearnWriteEntity
  ];
  for (var (index, r)
      in res.where((element) => element is GetInputFieldPart).indexed) {
    if (r is GetInputFieldPart) {
      (r as GetInputFieldPart).realInit(
          stringKey, targetStrings, currentTerm, sourceEntity,
          oldUserInputsContainer: (wwNavPr.results[stringKey] != null &&
wwNavPr.results[stringKey]!.isNotEmpty)? wwNavPr.results[stringKey]?.last ??
null : null);
    }
  }
  wwNavPr.results[stringKey] = ( wwNavPr.results[stringKey] ?? []);
  // for (var i in )
  wwNavPr.results[stringKey]!.add(null);

  return res;
}

List<Widget> getWriteFieldsListInLearnProcess(
  TermEntityDbDS currentTerm,
  List<TermEntityDbDS> termsList,
  WriteWordNavigationProvider wwNavPr,
) {

  for (var kv in wwNavPr.results.entries){
    for (var i in kv.value.where((element) => (element != null))){
      i!.clear();
    }
    kv.value.clear();
  }

  List<OneVariantTermField> widgetList = [];

  var similarTerms = termsList
      .where((element) =>
          // currentTerm.maybeReverseTermWriteSet.difference(
          //     currentTerm. isTermReverseWrite()?   element.definitionSet :
element.termSet
          // ).isEmpty
          currentTerm.maybeReverseTermWriteSet ==
          (currentTerm.isTermReverseWrite()
              ? element.definitionSet
              : element.termSet))
      .toList();
  for (var i in [currentTerm] + similarTerms) {
    // i.addInfoEntities.
```

```
var fields = i.addInfoEntities
    .toList()
    .where((element) => (element.addInfoType == AddInfoType.usual_term ||
        element.addInfoType == AddInfoType.other_form ||
        element.addInfoType == AddInfoType.abbreviation ||
        (element.addInfoType == AddInfoType.composite_word &&
            element.addingTextData == "1")))
    .toList();

if (fields.isNotEmpty) {
  fields.sort((one, two) {
    if (one.addInfoType == AddInfoType.usual_term &&
        two.addInfoType != AddInfoType.usual_term) {
      return 1;
    } else if (one.addInfoType == AddInfoType.usual_term &&
        two.addInfoType == AddInfoType.usual_term) {
      return 0;
    } else {
      if (one.addInfoType == AddInfoType.abbreviation &&
          two.addInfoType != AddInfoType.abbreviation) {
        return 1;
      } else if (one.addInfoType == AddInfoType.abbreviation &&
          two.addInfoType == AddInfoType.abbreviation) {
        return 0;
      } else {
        if (one.addInfoType == AddInfoType.other_form &&
            two.addInfoType != AddInfoType.other_form) {
          return 1;
        } else if (one.addInfoType == AddInfoType.other_form &&
            two.addInfoType == AddInfoType.other_form) {
          if (one.addingTextData == 'infinitive') {
            return -1;
          } else if (one.addingTextData == 'v2') {
            if (two.addingTextData == 'infinitive') {
              return 1;
            } else if (two.addingTextData == 'v2') {
              return 0;
            } else {
              return -1;
            }
          } else if (one.addingTextData == 'v3') {
            if (two.addingTextData == 'infinitive') {
              return 1;
            } else if (two.addingTextData == 'v2') {
              return 1;
            } else if (two.addingTextData == 'v3') {
              return 0;
            } else {
              return -1;
            }
          } else {
            return 1;
          }

          return 0;
        } else {
          if (one.addInfoType == AddInfoType.composite_word &&
              two.addInfoType != AddInfoType.composite_word) {
            return 1;
          } else if (one.addInfoType == AddInfoType.composite_word &&
              two.addInfoType == AddInfoType.composite_word) {
            return 0;
          } else {
            return -1;
```

```
              }
            }
          }
        }
      });

      if (wwNavPr.writtenWord != null) {
        fields = fields.where((element) {
          if (
          (wwNavPr.errorCountMap[element as LearnWriteEntity] ?? 0) > 0
          || (wwNavPr.learnWriteEntityToStrKey[element] != null &&
              wwNavPr.learnWriteEntityToStrKey[element]!.any(
                      (strKey) =>
              wwNavPr.strKeyToSourceEntity[strKey] != null
                  &&
                  wwNavPr.strKeyToSourceEntity[strKey]!.any(
                      (List<LearnWriteEntity> element1) =>
element1.contains(element)
                  )
                  && wwNavPr.strKeyToSourceEntity[strKey]!.length > 1
              )
          )) {
            return true;
          }
          wwNavPr.disabledFields.add(element);
          return false;
        }).toList();
      }

      for (var kv in wwNavPr.strKeyToSourceEntity.entries){
        for (var i in kv.value){
          i.clear();
        }
        kv.value.clear();
      }

      widgetList.addAll(fields
          .map<OneVariantTermField>(
              (e) => OneVariantTermField(e, i, currentTerm))
          .toList() as List<OneVariantTermField>);
    } else {
      if (
      wwNavPr.writtenWord == null ||
          (
              wwNavPr.writtenWord != null &&
              (
                  wwNavPr.errorCountMap[currentTerm as LearnWriteEntity] ??
0) >   0
          )
      || (wwNavPr.learnWriteEntityToStrKey[currentTerm] != null &&
          wwNavPr.learnWriteEntityToStrKey[currentTerm]!.any((strKey) =>
          wwNavPr.strKeyToSourceEntity[strKey] != null &&
              wwNavPr.strKeyToSourceEntity[strKey]!.any(
                      (List<LearnWriteEntity> element1) =>
element1.contains(currentTerm)
              )
          )
      ) ){
        widgetList.add(OneVariantTermField(null, i, currentTerm));
      } else {
        wwNavPr.disabledFields.add(currentTerm);
      }
    }
  }
```

```
  for (var (index, i) in widgetList.indexed) {
    i
      ..fieldsCount = widgetList.length
      ..currentFieldIndex = index;
  }

  return widgetList as List<Widget>;
}

bool testWrittenWord(
  String? sourceOrFormName,
  List<String?> helpString,
  TermEntityDbDS currentTerm,
  List<TermAddingInfoDbDS>? addTermInfo,
  WriteWordNavigationProvider wwNavPr,
) {
  var correctStringKey =
      (sourceOrFormName ?? "") + [for (var i in helpString) i ??
'@@@'].join();
  wwNavPr.checkCorrectingWordProcessor[correctStringKey] =
      (wwNavPr.checkCorrectingWordProcessor[correctStringKey] ?? []) + [];

  return true;
}

void userInputRegister(
  String userInput,
  String strKey,
  WriteWordNavigationProvider wwNavPr,
) {}

List<GetTermSourceOrFormName> sourceOrFormNameProcessor(
    TermAddingInfoDbDS? addTermInfo, TermEntityDbDS currentTerm) {
  bool sourceOrFormName;
  if (addTermInfo != null) {
    sourceOrFormName = (
        (addTermInfo!.addInfoType == AddInfoType.usual_term
            ||  addTermInfo!.addInfoType == AddInfoType.other_form)
            &&
            (addTermInfo!.addingTextData != null
                &&  addTermInfo!.addingTextData!.isNotEmpty)
    )
        || addTermInfo!.addInfoType == AddInfoType.abbreviation
    || addTermInfo.dialectOrArea != null
    ;
  } else {
    sourceOrFormName = false;
  }

  List<GetTermSourceOrFormName> sourceOrFormNameWidgetList =
(sourceOrFormName
      ? <GetTermSourceOrFormName>[
          GetTermSourceOrFormName(addTermInfo!, currentTerm)
        ]
      : <GetTermSourceOrFormName>[]);
  return sourceOrFormNameWidgetList;
}

import 'package:isar/isar.dart';
```

Листинг М.18 – UnaryModuleStateDbMixin

```
import '../../../repositoris/db_data_source/folder.dart';
import '../../../repositoris/db_data_source/module.dart';
import '../../../repositoris/db_data_source/term.dart';
import '../../../ui/pages/modules/unary_module.dart';
import '../../../ui/ui_templates/abstract_ui.dart';
import '../../../urils/db/abstract_entity.dart';
import '../../../urils/db/dbMixins.dart';
import '../../../urils/db/engine.dart';


abstract class UnaryModuleStateI {
  late ModuleDbDS currentModuleEntity;
// UnaryFolderI(this.folderId);
}

mixin UnaryModuleStateDbMixin
    implements
        OpenAndClose3<
            CollectionSchema<TermEntityDbDS>,
            CollectionSchema<ModuleDbDS>,
            CollectionSchema<AbstractEntity>
        >,
        UnaryModuleStateI,
        AbstractUIStatefulWidget<UnaryModule>,
        GetDataFromDbI {
  // late ModuleDbDS currentModuleEntity;
  late List<TermEntityDbDS> currentTerms = [];
  late FolderDbDS? rootFolder = null;

  @override
  void completionData() async {
    await openConn();
    currentModuleEntity = currentModuleEntity;
    currentTerms = (await conn[ConnType.term]!
        .collection<TermEntityDbDS>()
    // .verifyLink(linkName, sourceIds, targetIds)
        .filter()
        .module((q) => q.isarIdEqualTo(currentModuleEntity.isarId))

        .findAll());
    if (currentModuleEntity.rootFolderUuid != null){
      rootFolder = (await conn[ConnType.term]!
          .collection<FolderDbDS>()
          // .getByUuid(currentModuleEntity.rootFolderUuid!)
          .getByComplexIndex([currentModuleEntity.rootFolderUuid!,
currentModuleEntity.userUuid])
      );
    }
    await closeConn();

    this.setState(() => null);
    // closeConn();
  }

  @override
  void dispose() async {
    // TODO: implement dispose


    // super.dispose();
  }
```

```
}
```

Листинг М.19 – UnaryFolderDbMixin

```
import 'package:flutter/cupertino.dart';
import 'package:isar/isar.dart';

import '../../../repositoris/db_data_source/folder.dart';
import '../../../repositoris/db_data_source/module.dart';
import '../../../repositoris/db_data_source/term.dart';
import '../../../ui/pages/modules/unary_folder.dart';
import '../../../urils/db/dbMixins.dart';
import '../../../urils/db/engine.dart';


abstract class UnaryFolderI{
   late int folderId;
   // UnaryFolderI(this.folderId);
}




mixin UnaryFolderDbMixin
implements
    OpenAndClose3<CollectionSchema<FolderDbDS>,
        CollectionSchema<ModuleDbDS>,
        CollectionSchema<TermEntityDbDS>>,
    State<UnaryFolder>,
    UnaryFolderI,
    GetDataFromDbI
{
  late List<FolderDbDS> subFolders = [];
  late List<ModuleDbDS> currModules = [];
  late FolderDbDS? folderEntity = null;
  late FolderDbDS? rootFolderEntity = null;

  @override
  void completionData() async {
    await openConn();
    currModules = (await conn[ConnType.term]!
        .collection<ModuleDbDS>()
        .filter()
        // .not().root_folderIsNull()
        .rootFolder((q) => q.isarIdEqualTo(folderId))
        .findAll());
    subFolders = await conn[ConnType.term]!
        .collection<FolderDbDS>()
        .filter()
        .rootFolder((q) => q.isarIdEqualTo(folderId))
        .findAll();
    folderEntity = await conn[ConnType.term]!
        .collection<FolderDbDS>().get(folderId);
    if (folderEntity!.rootFolderUuid != null){
      rootFolderEntity = (await conn[ConnType.term]!
          .collection<FolderDbDS>()
      .getByComplexIndex([folderEntity!.rootFolderUuid!,
folderEntity!.userUuid])
          // .getByUuid(folderEntity!.rootFolderUuid!)
      );
    }
    await closeConn();
    this.setState(() => null);
```

```
  }
  @override
  void dispose() async {
    // TODO: implement dispose


    // super.dispose();
  }

}
```

Листинг М.20 – UserApiProfile

```
import 'package:isar/isar.dart';
import 'package:the_remember/src/urils/db/abstract_entity.dart';

import '../../../api_package/lib/api_package.dart';
import '../../../network_processor/network_main.dart';
import '../../repositoris/db_data_source/user.dart';
import '../../urils/isolate/base_msg.dart';
import '../../urils/isolate/msgs/login_user.dart';
import '../../urils/profilers/abstract.dart';
import 'isolates/network.dart';

class UserApiProfile extends ModChangeNotifier {
  final NetworkIsolateProfile networkPr;
  late UserDbDS? _user;

  late bool _firstUserInit;

  late ApiPackage? _baseApi;

  late Map<String, String> _authHeaders;

  late Future _awaitUser;

  Future<UserDbDS?> Function() get awaitUser => () async {
        await _awaitUser;
        return _user;
      };

  UserApiProfile({required this.networkPr}) : super() {
    // print("UserApiProfile");
    // init();
  }

  @override
  void init({bool isRealInit = false}) {
    parentProvidersList[networkPr.runtimeType] = networkPr;
    _user = null;
    _firstUserInit = true;
    _baseApi = null;
    _authHeaders = {};
    getUser();
    super.init(isRealInit: isRealInit);
  }

  Map<String, String> get authHeaders => _authHeaders;

  set authHeaders(Map<String, String> value) {
    _authHeaders = value;
  }
```

```
ApiPackage? get baseApi => _baseApi;

set baseApi(ApiPackage? value) {
  _baseApi = value;
}

UserDbDS? get user {
  return _user;
}

set user(UserDbDS? user) {
  if (this._user != user || _firstUserInit) {
    _firstUserInit = false;
    this._user = user;
    _awaitUser = (() async => this._user)();
    if (user != null) {
      networkProcessor(this);
    }

    notifyListeners();
  }
}

Future<void> userChange() async {
  var changedUser = _user;
  notifyListeners();
  if (changedUser != null) {
    var conn = await openConn();
    await conn.writeTxn(() async {
      await conn.collection<UserDbDS>().put(changedUser);
    });
    await closeConn();
  }
}

Future<UserDbDS?> getUser() async {
  _awaitUser = _getUser();
  return await _awaitUser;
}

Future<UserDbDS?> _getUser() async {
  if (_user != null && _user!.active) {
    return _user;
  } else {
    // await Future.delayed(Duration(seconds: 5));
    var conn = await openConn();
    var activeUsers = (await conn
        .collection<UserDbDS>()
        .filter()
        .activeEqualTo(true)
        .findAll());
    //2
    if (activeUsers.length > 1 || (_user != null && !(_user!.active))) {
      for (var u in activeUsers) {
        u.active = false;
      }
      if (_user != null) {
        activeUsers += [_user!, _user!];
      }
      (await conn.writeTxn(() async {
        (await conn.collection<UserDbDS>().putAll(activeUsers));
      }));
    }
```

```
      var coro = closeConn();

      UserDbDS? currentUser;
      if (activeUsers.isEmpty || activeUsers.length > 1) {
        currentUser = null;
      } else {
        currentUser = activeUsers[0];
      }
      await coro;
      _user = currentUser;
      notifyListeners();
      return _user;
    }
  }

  Future<void> loginUser(
      String username,
      String password,
      String? serverUrl,
      {Function(String?)? onErrorCallback}
      ) async {
    var res = await networkPr.sendToNetworkIsolateWithAnswer<
          LoginUserIsolateMsg, LoginUserAnsIsolateMsg>(
        LoginUserIsolateMsg(username, password, serverUrl),
        CrossIsolatesMessageType.userAndPassword,
        CrossIsolatesMessageType.loginUserAnsIsolateMsg
    );

    if (res.message.data) {
      var isar = await openConn();

      var query = isar
          .collection<UserDbDS>()
          .filter()
          .usernameEqualTo(username);

      var userStream = query.build().watch(fireImmediately: true);
      user = (await userStream.firstWhere((element) =>
element.isNotEmpty))[0];

      // await for (var usersList in userStream){
      //   for (var currUser in usersList){
      //     this.user = currUser;
      //     break;
      //   }
      //   if (this.user != null){
      //     break;
      //   }
      // }

      await closeConn();
    } else {
      if (onErrorCallback != null) {
        onErrorCallback(
            "На сервере пользователя с таким логином и паролем не найдено.
Код ${res.message.statusCode}");
      }
    }
  }

  @override
  List<CollectionSchema<AbstractEntity>> get classes => [];
```

```
  @override
  void dispose() {
    super.dispose();
  }
}
```

Листинг М.21 – BottomNavigationProvider

```
import 'dart:core';

import 'package:flutter/cupertino.dart';

import '../../../../urils/profilers/abstract.dart';

class BottomNavigationProvider extends ModChangeNotifier {
  late int _activePageNumber ;
  late bool _firstUserInit ;

  BottomNavigationProvider() : super() {
    // init();
  }

  @override
  void init({bool isRealInit = false}) {
     _activePageNumber = 0;
     _firstUserInit = true;
    super.init(isRealInit: isRealInit);
  }

  set activePageNumber(int val) {
    if (this._activePageNumber != val || _firstUserInit) {
      _firstUserInit = false;
      this._activePageNumber = val;

      notifyListeners();
    }
  }

  int get activePageNumber => _activePageNumber;

  Widget Function(BuildContext)? _bottomNavWidget = null;

  Widget Function(BuildContext)? get bottomNavWidget => _bottomNavWidget;

  set bottomNavWidget(Widget Function(BuildContext)? value) {
    _bottomNavWidget = value;
  }
}
```

Листинг М.22 – AppBarNavigationProvider

```
import 'dart:core';

import 'package:flutter/cupertino.dart';

import '../../../../ui/ui_templates/abstract_ui.dart';
import '../../../../ui/ui_templates/navigation/app_bar_navigation_enum.dart';
import '../../../../urils/profilers/abstract.dart';

class AppBarNavigationProvider extends ModChangeNotifier {
  late  Map<AppBarNavigationEnum,
          PreferredSizeWidget Function(BuildContext, DFMapper)>
```

```
      _getAppBarWidget ;

  AppBarNavigationProvider(): super() {

  }

  @override
  void init({bool isRealInit = false}) {
    _getAppBarWidget = Map();
    super.init(isRealInit: isRealInit);
  }

  Map<AppBarNavigationEnum,
          PreferredSizeWidget Function(BuildContext, DFMapper)>
      get getAppBarWidget => _getAppBarWidget;

  set getAppBarWidget(
      Map<AppBarNavigationEnum,
              PreferredSizeWidget Function(BuildContext, DFMapper)>
          value) {
    _getAppBarWidget = value;
  }
```

Листинг М.23 – SubFolderAndModuleProvider

```
import 'package:isar/isar.dart';
import 'package:the_remember/src/repositoris/db_data_source/folder.dart';
import 'package:the_remember/src/repositoris/db_data_source/module.dart';
import 'package:the_remember/src/repositoris/db_data_source/term.dart';

import '../../../../repositoris/db_data_source/sentence.dart';
import '../../../../repositoris/db_data_source/term_adding_info.dart';
import '../../../../repositoris/db_data_source/user.dart';
import '../../../../ui/ui_templates/abstract_ui.dart';
import '../../../../urils/db/abstract_entity.dart';
import '../../../../urils/profilers/abstract.dart';
import 'folder_module.dart';


class SubFolderAndModuleProvider extends ModChangeNotifier {
  late List<FolderDbDS>? _subFoldersList ;
  late List<ModuleDbDS>? _subModulesList ;
  final FolderAndModuleProvider fmPr;
  final DFMapper dfMapper;
  late Future _initLists;

  @override
  List<CollectionSchema<AbstractEntity>> get classes => [
        UserDbDSSchema,
        FolderDbDSSchema,
        ModuleDbDSSchema,
        TermEntityDbDSSchema,
        TermAddingInfoDbDSSchema,
        SentenceDbDSSchema,
      ];

  SubFolderAndModuleProvider(this.dfMapper, {required this.fmPr}) : super() {
    // init(isRealInit: true);
  }

  @override
  void init({bool isRealInit = false}) {
    // if (_subFoldersList != null && fmPr.currentModule == null){
```

```
  _subFoldersList = null;
  _subModulesList = null;
  // if (isRealInit) {
  //   _initLists = _initListsFromDB();
  // } else {
  //   _initLists = (() async {
  //     await _initLists;
  //     _initListsFromDB();
  //   })();
  // }
  _initLists = _initListsFromDB();
  parentProvidersList[fmPr.runtimeType] = fmPr;
  super.init(isRealInit: isRealInit);

}

List<FolderDbDS>? get subFoldersList => _subFoldersList;

List<ModuleDbDS>? get subModulesList => _subModulesList;

Future get initLists => _initLists;

Future<void> _initListsFromDB() async {
  if (fmPr.currentModule == null) {
    var isar = await openConn();
    if (fmPr.currentFolder == null) {
      _subFoldersList = await isar
          .collection<FolderDbDS>()
          .filter()
          .rootFolderIsNull()
          .userUuidEqualTo(fmPr.userPr.user!.uuid)
          .findAll();
      _subModulesList = await isar
          .collection<ModuleDbDS>()
          .filter()
          .rootFolderIsNull()
          .userUuidEqualTo(fmPr.userPr.user!.uuid)
          .findAll();
    } else {
      _subFoldersList = await isar
          .collection<FolderDbDS>()
          .filter()
          .rootFolder((q) => q.isarIdEqualTo(fmPr.currentFolder!.isarId))
          .userUuidEqualTo(fmPr.userPr.user!.uuid)
          .findAll();
      _subModulesList = await isar
          .collection<ModuleDbDS>()
          .filter()
          .rootFolder((q) => q.isarIdEqualTo(fmPr.currentFolder!.isarId))
          .userUuidEqualTo(fmPr.userPr.user!.uuid)
          .findAll();
    }
    notifyListeners();
    await closeConn();
  } else {
    _subFoldersList = [];
    _subModulesList = [];
    notifyListeners();
  }
}

void changed() {
  notifyListeners();
}
```

```
  // @override
  // void dispose() {
  //   fmPr.subFolderPr = null;
  //   super.dispose();
  // }

  // @override
  // void notifyListeners() {
  //   super.notifyListeners();
  // }
}
```

Листинг М.24 – FolderProgressBarProvider

```
import 'package:isar/isar.dart';
import
'package:the_remember/src/domain_layer/providers/main/folders/sub_folder_modu
les.dart';
import 'package:the_remember/src/repositoris/db_data_source/folder.dart';
import 'package:the_remember/src/repositoris/db_data_source/module.dart';
import 'package:the_remember/src/repositoris/db_data_source/term.dart';

import '../../../../urils/db/abstract_entity.dart';
import '../../../../urils/profilers/abstract.dart';

class FolderStats {
  final int write;
  final int choice;
  final int watch;
  final int all;

  FolderStats(this.write, this.choice, this.watch, this.all);
}

class FolderProgressBarProvider extends ModChangeNotifier {
  // late List<FolderDbDS>? _subFoldersList ;
  // late List<ModuleDbDS>? _subModulesList ;
  final SubFolderAndModuleProvider subFmPr;

  // final DFMapper dfMapper;
  late Future _dbTransaction;
  late Map<FolderOrModule, List<ModuleDbDS>> _getPersonalStats = Map();
  late Map<ModuleDbDS, FolderStats> _getModuleState;

  FolderProgressBarProvider(this.subFmPr) : super() {}

  @override
  void init({bool isRealInit = false}) {
    // if (_subFoldersList != null && fmPr.currentModule == null){
    // _subFoldersList = null;
    // _subModulesList = null;
    if (parentProvidersList[subFmPr.runtimeType] == null ||
parentProvidersList[subFmPr.runtimeType] != subFmPr ||
_getPersonalStats.isNotEmpty) {
      _getPersonalStats = Map();
      _dbTransaction = _getStats();
      parentProvidersList[subFmPr.runtimeType] = subFmPr;
    }
    super.init(isRealInit: isRealInit);
  }

  // List<FolderDbDS>? get subFoldersList => _subFoldersList;
```

```
  //
  // List<ModuleDbDS>? get subModulesList => _subModulesList;
  //
  // Future get initLists => _initLists;

  void _addToTree(
      List<FolderOrModule> data, Map<String, List<FolderOrModule>> tree) {
    for (var i in data) {
      tree[i.rootFolderUuid!] =(tree[i.rootFolderUuid] ?? <FolderOrModule>[])
+  <FolderOrModule>[i as FolderOrModule];
    }
  }

  Future<FolderStats> getFolderOrModuleStats(FolderOrModule targetObj) async
{
    var res = await _dbTransaction;
    // _getPersonalStats.containsKey(targetObj)
    var statesList = [
      for (var i in _getPersonalStats[targetObj]!) _getModuleState[i]
    ];
    return FolderStats(
        statesList.map<int>((i) => i!.write).reduce((a, b) => a + b),
        statesList.map<int>((i) => i!.choice).reduce((a, b) => a + b),
        statesList.map<int>((i) => i!.watch).reduce((a, b) => a + b),
        statesList.map<int>((i) => i!.all).reduce((a, b) => a + b));
  }

  Future<void> _getStats() async {
    await subFmPr.initLists;
    var isar = await openConn();

    var subFolderList = [for (var i in subFmPr.subFoldersList!) i];
    List<FolderDbDS> allFolderList = [for (var i in subFmPr.subFoldersList!)
i];
    Map<String, List<FolderOrModule>> subFolderTree = {
      for (var i in subFolderList) i.uuid: <FolderOrModule>[]
    };
    List<FolderDbDS> nextSubFolderList = [];

    late Future<List<ModuleDbDS>> modules = (() async => <ModuleDbDS>[])();
    List<ModuleDbDS> allModules = [for (var i in subFmPr.subModulesList!) i];

    while (subFolderList.isNotEmpty) {
      var subFolderUuidList = [for (var i in subFolderList) i.uuid];
      var currModulesList = await modules;
      var folders = isar
          .collection<FolderDbDS>()
          // .where((e) => subFolderUuidList.contains(e.rootFolderUuid))
          .filter()
          .anyOf(subFolderUuidList,
              (q, element) => q.rootFolderUuidEqualTo(element))
          // .anyOf([for(var i in subFolderList) i.uuid],
FilterRepeatModifier())
          // .userUuidEqualTo(fmPr.userPr.user!.uuid)
          .findAll();
      allModules += currModulesList;
      _addToTree(currModulesList, subFolderTree);
      nextSubFolderList = await folders;
      allFolderList += nextSubFolderList;

      modules = isar
          .collection<ModuleDbDS>()
          // .where((e) => subFolderUuidList.contains(e.rootFolderUuid))
          .filter()
```

```
          .anyOf(subFolderUuidList,
              (q, element) => q.rootFolderUuidEqualTo(element))
          // .anyOf([for(var i in subFolderList) i.uuid],
FilterRepeatModifier())
          // .userUuidEqualTo(fmPr.userPr.user!.uuid)
          .findAll();
      _addToTree(nextSubFolderList, subFolderTree);
      subFolderList = nextSubFolderList;
    }
    var currModulesList = await modules;
    allModules += currModulesList;
    _addToTree(currModulesList, subFolderTree);
    var moduleStats = {
      for (var i in allModules) i: await i.getLearnProcess(isar)
    };
     await closeConn();

    var notComputedFoldersCount = subFolderTree.length;
    Map<String, List<ModuleDbDS>> computedFolders = {};
    while (notComputedFoldersCount != 0) {
      for (var i in subFolderTree.entries) {
        if (!computedFolders.containsKey(i.key)) {
          if (i.value.isEmpty) {
            computedFolders[i.key] = [];
            notComputedFoldersCount -= 1;
          } else if (i.value.every((element) =>
              (element is ModuleDbDS) ||
              (computedFolders.containsKey(element.uuid)))) {
            computedFolders[i.key] = [];
            notComputedFoldersCount -= 1;
            for (var ii in i.value) {
              if (ii is ModuleDbDS) {
                computedFolders[i.key]!.add(ii as ModuleDbDS);
              } else {
                computedFolders[i.key]!.addAll(computedFolders[ii.uuid]!);
              }
            }
          }
        }
      }
    }
    // await coro;
    _getPersonalStats = {
      for (var i in allFolderList) i: computedFolders[i.uuid]!
    };
    _getPersonalStats.addAll({
      for (var i in allModules) i: [i]
    });
    _getModuleState = {
      for(var i in
      {for (var i in allModules) i: moduleStats[i]! }.entries)
        i.key: FolderStats(i.value.$1, i.value.$2, i.value.$3, i.value.$4)
    };

    // var terms = await isar
    //     .collection<TermEntityDbDS>()
    //     .filter()
    //     .anyOf([for (var i in allModules) i.uuid ], (q, element) =>
q.moduleUuidEqualTo(element))
    //     .userUuidEqualTo(subFmPr.fmPr.userPr.user!.uuid)
    //     .findAll();

    // Map<ModuleDbDS, List<TermEntityDbDS>> terms
```

```
    notifyListeners();

  }

  void changed() {
    notifyListeners();
  }

// @override
// void dispose() {
//   fmPr.subFolderPr = null;
//   super.dispose();
// }

// @override
// void notifyListeners() {
//   super.notifyListeners();
// }
}
```

Листинг М.25 – FolderAndModuleProvider

```
import
'package:the_remember/src/domain_layer/providers/user_api_provider.dart';
import 'package:the_remember/src/repositoris/db_data_source/folder.dart';
import 'package:the_remember/src/repositoris/db_data_source/module.dart';

import '../../../../urils/db/abstract_entity.dart';
import '../../../../urils/profilers/abstract.dart';


class FolderAndModuleProvider extends ModChangeNotifier {
  // SubFolderAndModuleProvider? subFolderPr = null;
  late FolderDbDS? _currentFolder ;
  late List<FolderDbDS?> _rootFolder ;

  late ModuleDbDS? _currentModule ;
  late bool _firstFolderInit ;
  final UserApiProfile userPr;

  FolderAndModuleProvider({required this.userPr}) : super() {
    // init();
  }

  @override
  void init({bool isRealInit = false}) {
    _currentFolder = null;
    _rootFolder = [];

    _currentModule = null;
    _firstFolderInit = true;
    super.init(isRealInit: isRealInit);
  }

  FolderDbDS? get rootFolder => _rootFolder.last;

  AbstractEntity? get activeEntity => _currentModule ?? _currentFolder;

  set activeEntity(AbstractEntity? value) {
    if (value is FolderDbDS) {
      var newFolder = value as FolderDbDS;
      if (value == _currentFolder && _currentModule == null) {
      } else {
```

```
      _rootFolderSet(newFolder);
      _currentFolder = newFolder;
      _currentModule = null;
      notifyListeners();
    }
  } else if (value is ModuleDbDS) {
    if (value == _currentModule) {
    } else {
      _currentModule = value;
      notifyListeners();
    }
  } else if (value == null) {
    if (_currentFolder == null && _currentModule == null) {
    } else {
      _rootFolderSet(null);
      _currentFolder = null;
      _currentModule = null;
      notifyListeners();
    }
  } else {
    throw UnimplementedError();
  }
}

ModuleDbDS? get currentModule => _currentModule;

set currentModule(ModuleDbDS? value) {
  if (_currentModule != value || _firstFolderInit) {
    _firstFolderInit = false;
    _currentModule = value;
    notifyListeners();
  }
}

FolderDbDS? get currentFolder => _currentFolder;

set currentFolder(FolderDbDS? folder) {
  if ((this._currentFolder != folder || this._currentModule != null) ||
      _firstFolderInit) {
    _firstFolderInit = false;
    _rootFolderSet(folder);
    this._currentFolder = folder;
    this._currentModule = null;
    notifyListeners();
  }
}

void _rootFolderSet(FolderDbDS? newFolder) {
  if (newFolder == _currentFolder && _currentModule == null) {
  } else {
    if (newFolder == null) {
      if (_currentFolder != null) {
        _rootFolder.removeAt(_rootFolder.length - 1);
      }
    } else if (newFolder != _currentFolder) {
      if (_currentFolder == null) {
        _rootFolder.add(null);
      } else if (newFolder.rootFolderUuid == _currentFolder!.uuid) {
        _rootFolder.add(_currentFolder);
      } else if (_currentFolder!.rootFolderUuid == newFolder.uuid) {
        _rootFolder.removeAt(_rootFolder.length - 1);
      } else {
        throw UnimplementedError();
      }
```

```
      }
    }
  }

// @override
// void dispose() {
//   super.dispose();
// }

// @override
// void notifyListeners() {
//   super.notifyListeners();
//   subFolderPr?.init();
//   // subFolderPr = null;
// }
}
```

Листинг М.26 – TermsInModuleProvider

```
import 'package:isar/isar.dart';
import 'package:the_remember/src/repositoris/db_data_source/module.dart';
import 'package:the_remember/src/repositoris/db_data_source/term.dart';

import '../../../../../repositoris/db_data_source/sentence.dart';
import '../../../../../repositoris/db_data_source/term_adding_info.dart';
import '../../../../../urils/db/abstract_entity.dart';
import '../../../../../urils/profilers/abstract.dart';
import '../folder_module.dart';

class TermsInModuleProvider extends ModChangeNotifier {
  late List<TermEntityDbDS>? _termsList;

  late List<TermEntityDbDS>? _learningIterationTermsList;

  late List<TermEntityDbDS>? _changedInLearningIterationTermsList;

  final FolderAndModuleProvider fmPr;
  late Future _initLists;

  @override
  List<CollectionSchema<AbstractEntity>> get classes =>
      [TermEntityDbDSSchema, TermAddingInfoDbDSSchema, SentenceDbDSSchema];

  TermsInModuleProvider({required this.fmPr}) : super() {
    // init();
  }

  @override
  void init({bool isRealInit = false}) {
    _initLists = _initListsFromDB();
    _termsList = null;
    _learningIterationTermsList = null;
    _changedInLearningIterationTermsList = null;
    super.init(isRealInit: isRealInit);
  }

  List<TermEntityDbDS>? get termsList => _termsList;

  Future get initLists => _initLists;

  Future<void> _initListsFromDB() async {
    var isar = await openConn();
```

```
    // var module = await isar
    //     .collection<ModuleDbDS>()
    //     .filter()
    //     .uuidEqualTo(fmPr.currentModule!.uuid)
    //     .userUuidEqualTo(fmPr.userPr.user!.uuid)
    //     .findAll();
    // await module[0].words.load();
    // _termsList = module[0].words.toList();
    _termsList = await isar
        .collection<TermEntityDbDS>()
        .filter()
        .module((q) => q.uuidEqualTo(fmPr.currentModule!.uuid))
        .moduleUuidEqualTo(fmPr.currentModule!.uuid)
        .userUuidEqualTo(fmPr.userPr.user!.uuid)
        .findAll();
    var uuidTermList = {for (var i in _termsList!) i.uuid: i};
    var addTermsInfoEntities = await isar
        .collection<TermAddingInfoDbDS>()
        .filter()
        .anyOf(
            uuidTermList.keys, (q, String termUuid) =>
q.termUuidEqualTo(termUuid))
        .userUuidEqualTo(fmPr.userPr.user!.uuid)
        .findAll();
    var sentenceEntities = await isar
        .collection<SentenceDbDS>()
        .filter()
        .anyOf(
        uuidTermList.keys, (q, String termUuid) =>
q.termUuidEqualTo(termUuid))
        .userUuidEqualTo(fmPr.userPr.user!.uuid)
        .findAll();

    Map<String, List<TermAddingInfoDbDS>> addTermsInfoEntitiesMap = {};
    Map<String, List<SentenceDbDS>> sentenceEntitiesMap = {};
    for (var i in addTermsInfoEntities){
      addTermsInfoEntitiesMap[i.termUuid] =
(addTermsInfoEntitiesMap[i.termUuid] ?? [])
        ..add(i);
      i.termEntity.value = uuidTermList[i.termUuid]!;
    }
    for (var i in sentenceEntities){
      sentenceEntitiesMap[i.termUuid] = (sentenceEntitiesMap[i.termUuid] ??
[])
        ..add(i);
      i.termEntity.value = uuidTermList[i.termUuid]!;
    }

   for(var element in  _termsList!) {
      element.module.value = fmPr.currentModule;
      // element.addInfoEntities = (addTermsInfoEntitiesMap[element.uuid]!);
      element.addInfoEntities.addAll(addTermsInfoEntitiesMap[element.uuid]!);
      element.sentenceEntities.addAll(sentenceEntitiesMap[element.uuid] ??
[]);
      await element.addInfoEntities.load();
      await element.sentenceEntities.load();
    }

    // await _termsList![0].module.load();
    // _termsList![0].module.

    notifyListeners();
    await closeConn();
  }
```

```
  List<TermEntityDbDS>? get changedInLearningIterationTermsList =>
      _changedInLearningIterationTermsList;

  set changedInLearningIterationTermsList(List<TermEntityDbDS>? value) {
    _changedInLearningIterationTermsList = value;
  }

  List<TermEntityDbDS>? get learningIterationTermsList =>
      _learningIterationTermsList;

  set learningIterationTermsList(List<TermEntityDbDS>? value) {
    _learningIterationTermsList = value;
  }
}
```

Листинг М.27 – ModuleButtonNavigationProvider

```
import 'dart:core';

import
'package:the_remember/src/domain_layer/providers/main/folders/module/terms_in
_module.dart';
import
'package:the_remember/src/domain_layer/providers/user_api_provider.dart';

import '../../../../../urils/profilers/abstract.dart';
import '../../../../functions/words_BO.dart';

enum ModuleButtonNavigationEnum {
  startLearn,
  continueLearn,
  moduleSettings,
  mainModuleScreen
}

class ModuleButtonNavigationProvider extends ModChangeNotifier {
  final TermsInModuleProvider termPr;
  final UserApiProfile userPr;
  late ModuleButtonNavigationEnum _buttonType ;
  late bool _firstUserInit ;

  ModuleButtonNavigationProvider(this.termPr, this.userPr): super()  {
    // init();
  }

  @override
  void init({bool isRealInit = false}) {
    _buttonType =
        ModuleButtonNavigationEnum.mainModuleScreen;
    _firstUserInit = true;
    super.init(isRealInit: isRealInit);
  }

  set buttonType(ModuleButtonNavigationEnum val) {
    if (this._buttonType != val || _firstUserInit) {
      _firstUserInit = false;
      this._buttonType = val;
      if (val == ModuleButtonNavigationEnum.startLearn) {
        startLearning(termPr, userPr);
        buttonType = ModuleButtonNavigationEnum.continueLearn;
      }
```

```
      notifyListeners();
    }
  }

  void setButtonTypeWithoutSetState(ModuleButtonNavigationEnum val) {
    if (this._buttonType != val || _firstUserInit) {
      _firstUserInit = false;
      this._buttonType = val;
    }
  }

  ModuleButtonNavigationEnum get buttonType => _buttonType;
}
```

Листинг М.28 – LearnScreensNavigationProvider

```
import 'dart:core';

import 'package:isar/isar.dart';
// import
'package:the_remember/src/domain_layer/providers/folders/module/terms_in_modu
le.dart';
import
'package:the_remember/src/domain_layer/providers/user_api_provider.dart';
import 'package:the_remember/src/urils/db/abstract_entity.dart';

import '../../../../../../urils/profilers/abstract.dart';
import '../../../../../functions/words_BO.dart';
import '../module_buttoons_navigation.dart';
import '../terms_in_module.dart';

class LearnScreensNavigationProvider extends ModChangeNotifier {
  late int _activePageNumber = 0;
  late bool _firstUserInit = true;
  final UserApiProfile _userPr;
  final TermsInModuleProvider _termsPr;
  final ModuleButtonNavigationProvider _moduleNavPr;
  late Map<int, bool> buttonPressed;

  LearnScreensNavigationProvider(this._userPr, this._termsPr,
this._moduleNavPr): super()  {
    // init();
  }

  @override
  void init({bool isRealInit = false}) {
    _activePageNumber = 0;
    _firstUserInit = true;

    super.init(isRealInit: isRealInit);
  }

  set activePageNumber(int val) {
    if (this._activePageNumber != val || _firstUserInit) {
      _firstUserInit = false;
      this._activePageNumber = val;
      if (iterationFinished) {
        // learnTransactionCompleted();
        learnTransactionCompleted(
            _termsPr.changedInLearningIterationTermsList ?? [],
            _userPr
        );
```

```
        _moduleNavPr.buttonType =
ModuleButtonNavigationEnum.mainModuleScreen;
      } else {
        // else if (learnFinished) {
        //   learnNavPr.learnFinished
        //   _moduleNavPr.buttonType = ModuleButtonNavigationEnum.startLearn;
        // }

        notifyListeners();
      }
    }
  }

  int get activePageNumber => _activePageNumber;

  bool get iterationFinished =>
      _termsPr.learningIterationTermsList!.length <= _activePageNumber;

  bool get learnFinished => _termsPr.learningIterationTermsList!.isEmpty;

  bool get learnNotStarted => _termsPr.learningIterationTermsList == null;

  @override
  List<CollectionSchema<AbstractEntity>> get classes => [];

  @override
  void dispose(){
    super.dispose();


  }
}
```

Листинг М.29 – WriteWordNavigationProvider

```
import 'dart:core';

// import
'package:the_remember/src/domain_layer/providers/folders/module/terms_in_modu
le.dart';
import 'package:flutter/cupertino.dart';
import 'package:quiver/collection.dart';
import 'package:the_remember/src/repositoris/db_data_source/term.dart';

import '../../../../../../../ui/ui_templates/abstract_ui.dart';
import '../../../../../../../urils/profilers/abstract.dart';
import '../../../../../../../urils/sets.dart';
import '../../terms_in_module.dart';
import '../learning_navigation.dart';

class WriteWordRes {
  String userInput;
  final int indexInString;

  // final int fieldIndex;
  late String keyString;
  late Future<void> Function(BuildContext, AbstractUIStatefulWidget)
      errorCallback;
  late Future<void> Function(BuildContext, AbstractUIStatefulWidget)
      successCallback;

  WriteWordRes(
    this.userInput,
```

```
    this.indexInString,
    this.keyString,
    Future<void> Function(BuildContext, AbstractUIStatefulWidget)?
        errorCallback,
    Future<void> Function(BuildContext, AbstractUIStatefulWidget)?
        successCallback,
  ) {
    if (errorCallback != null) {
      this.errorCallback = errorCallback;
    }
    if (successCallback != null) {
      this.successCallback = successCallback;
    }
  }
}

// class ItemHashList extends List<String>{
//
// }

class WriteWordNavigationProvider extends ModChangeNotifier {
  late String? _writtenWord;

  TermEntityDbDS get _targetWord =>
      termsPr.learningIterationTermsList![learnNavPr.activePageNumber];
  final LearnScreensNavigationProvider learnNavPr;
  final TermsInModuleProvider termsPr;
  late Map<String, List<List<String>>> checkCorrectingWordProcessor;
  late Map<String, List<TermEntityDbDS>> strTermsToEntities;
  late Map<String, List<List<LearnWriteEntity>>> strKeyToSourceEntity;
  // late Map<TermEntityDbDS, Map<LearnWriteEntity, WriteWordRes>>
resultsAsMap;
  late Map<String, List<List<WriteWordRes>?>> results;
  late Map<LearnWriteEntity, int> errorCountMap;
  late List<LearnWriteEntity> disabledFields;
  late Map<LearnWriteEntity, List<String>> learnWriteEntityToStrKey;

  WriteWordNavigationProvider(this.termsPr, this.learnNavPr) : super() {}

  @override
  void init({bool isRealInit = false}) {
    parentProvidersList[learnNavPr.runtimeType] = learnNavPr;
    _writtenWord = null;
    checkCorrectingWordProcessor = Map();
    strTermsToEntities = Map();
    strKeyToSourceEntity = Map();
    // resultsAsMap = Map();
    results = Map();
    errorCountMap = Map();
    disabledFields = [];
    learnWriteEntityToStrKey = {};

    super.init(isRealInit: isRealInit);
  }

  String? get writtenWord => _writtenWord;

  set writtenWord(String? value) {
    _writtenWord = value;
    // writeWordChanging(_targetWord, _writtenWord, termsPr.termsList!);
    notifyListeners();
  }

  TermEntityDbDS get targetWord => _targetWord;
```

```
  void addWriteWordProcessor(
      String strKey,
      List<String> targetStrings,
      TermEntityDbDS currentTerm,
      List<LearnWriteEntity> sourceEntity,
      List<WriteWordRes> resultsContainer) {
    checkCorrectingWordProcessor[strKey] =
        (checkCorrectingWordProcessor[strKey] ?? []);
    checkCorrectingWordProcessor[strKey]!.add(targetStrings);
    strTermsToEntities[strKey] = (strTermsToEntities[strKey] ?? []);
    strTermsToEntities[strKey]!.add(currentTerm);
    strKeyToSourceEntity[strKey] = (strKeyToSourceEntity[strKey] ?? []);
    strKeyToSourceEntity[strKey]!.add(sourceEntity);
    for (var lvEntity in sourceEntity) {
      learnWriteEntityToStrKey[lvEntity] =
          (learnWriteEntityToStrKey[lvEntity] ?? []);
      learnWriteEntityToStrKey[lvEntity]!.add(strKey);

    }

    results[strKey] = (results[strKey] ?? []);
    if (results[strKey]!.isNotEmpty && results[strKey]!.last == null){
      results[strKey]![ results[strKey]!.length - 1] = resultsContainer;
    } else {
      results[strKey]!.add(resultsContainer);
    }
    // results[strKey]!.toList()
    print("");
  }

  Future<void> checkUserInput(
      BuildContext? context,
      AbstractUIStatefulWidget? setStateWidget,
      {bool isAnotherWrite = false,
      bool passedReInput = false,
      bool answerIsPreventiveCorrect = false}) async {
    int errorCount = 0;

    Map<TermEntityDbDS, int> termErrorCounter = {};

    for (var kv in strKeyToSourceEntity.entries.where((element) =>
        !(element.value.any((element1) =>
            element1.any((element2) => disabledFields.contains(element2))))))
{
      var strKey = kv.key;
      var currentTerms = strTermsToEntities[strKey]!;

      for (var i in currentTerms) {
        termErrorCounter[i] = (termErrorCounter[i] ?? 0);
        errorCountMap[i] = (errorCountMap[i] ?? 0);
      }
      for (var i in kv.value){
        for (var ii in i){
          errorCountMap[ii] = (errorCountMap[ii] ?? 0);
        }
      }

      var targetWords = [
        for (var i in kv.value) [for (var ii in i) ii.targetWrite]
      ];
      var userInputs = [
        for (var i in results[strKey]!.where((element) => (element != null)))
          [for (var ii in i!) ii.userInput]
```

```
      ];
      var _targetWordsStrings = [
        for (var (index, i) in targetWords.indexed) (i.join("####"), -1)
      ];
      var _userInputsStrings = [
        for (var (index, i) in userInputs.indexed) (i.join("####"), -1)
      ];
      var targetWordsStrings = {
        for (var (index, i) in _targetWordsStrings.indexed) i
      };
      var userInputsStrings = {
        for (var (index, i) in _userInputsStrings.indexed) i
      };

      Map<(String, int), int> targetWordsStringsCounter = {};
      Map<(String, int), int>  userInputsCounter = {};
      for (var i in _targetWordsStrings) {
        targetWordsStringsCounter[i] = (targetWordsStringsCounter[i] ?? 0) +
1;
      }
      for (var i in _userInputsStrings) {
        userInputsCounter[i] = (userInputsCounter[i] ?? 0) + 1;
      }




      Set<(String, int)> correctTermsStr;
      Set<(String, int)> errorTermsStr;
      // setsEqual({1}, {1});
      if (setsEqual(targetWordsStrings.intersection(userInputsStrings),
          targetWordsStrings)) {
        correctTermsStr = targetWordsStrings;
        errorTermsStr = {};
      } else {
        correctTermsStr = targetWordsStrings.intersection(userInputsStrings);
        errorTermsStr = mapCounterDifference(
            userInputsCounter,
            targetWordsStringsCounter
        ).keys.toSet();
        // errored
        // var correctTerms = [
        //   for (var (i, index) in correctTermsStr) targetWords[index]
        // ];
        // var errorTerms = [
        //   for (var (i, index) in errorTermsStr) userInputs[index]
        // ];
      }

      var targetWordsStringsOnly = {for (var i in correctTermsStr) i.$1};
      var targetWordsAsStrings = [ for (var i in
strKeyToSourceEntity[strKey]!) for (var ii in i) ii];
      for (var learnWriteEntity in targetWordsAsStrings.where(
              (element) =>
targetWordsStringsOnly.contains((element.targetWrite))
      )){
        var currTerm = learnWriteEntity.termEntityInterface;
        // if (!isAnotherWrite) {
        //   currTerm.writeErrorCounter += 1;
        // }
        if (!answerIsPreventiveCorrect) {
          termErrorCounter[currTerm] = (termErrorCounter[currTerm] ?? 0);
        }
        errorCountMap[learnWriteEntity] =
```

```
        (errorCountMap[learnWriteEntity] ?? 0);
      }


      // for (var index = 0; index < correctTermsStr.length; index++) {
      //   var currTerm = strTermsToEntities[strKey]![0].termEntityInterface;
      //   // if (!isAnotherWrite) {
      //   //   currTerm.writeErrorCounter += 1;
      //   // }
      //   if (!answerIsPreventiveCorrect) {
      //     termErrorCounter[currTerm] = (termErrorCounter[currTerm] ?? 0);
      //   }
      //   errorCountMap[strTermsToEntities[strKey]![0]] =
      //   (errorCountMap[strTermsToEntities[strKey]![0]] ?? 0);
      // }


      errorCount += errorTermsStr.length;

      // var targetWordsErrorsOnly = {for (var i in targetWordsStrings)
i.$1};
      for (var learnWriteEntity in targetWordsAsStrings.where(
              (element) =>
!targetWordsStringsOnly.contains((element.targetWrite))
      )){
      // for (var (errorStrField, index) in errorTermsStr) {
        var currTerm = learnWriteEntity.termEntityInterface;
        // if (!isAnotherWrite) {
        //   currTerm.writeErrorCounter += 1;
        // }
        if (!answerIsPreventiveCorrect) {
          termErrorCounter[currTerm] = (termErrorCounter[currTerm] ?? 0) + 1;
        }
        errorCountMap[learnWriteEntity] =
            (errorCountMap[learnWriteEntity] ?? 0) + 1;

        // for (var i in strKeyToSourceEntity[strKey]![0]) {
          if (currTerm != learnWriteEntity) {
            errorCountMap[learnWriteEntity] =
(errorCountMap[learnWriteEntity] ?? 0) + 1;
          }
        // }
      }



      await Future.wait([
            for (var (i, index) in correctTermsStr)
              for (var ii
                  in results[strKey]!.where((element) => (element != null)))
                for (var iii in ii!)
                  iii.successCallback(context!, setStateWidget!)
          ] +
          [
            for (var (i, index) in errorTermsStr)
              for (var ii
                  in results[strKey]!.where((element) => (element != null)))
                for (var iii in ii!)
                  iii.errorCallback(context!, setStateWidget!)
          ]);

      // var correct = [
      //   for (var i in targetWordsStrings.intersection( userInputsStrings))
      //
```

```
      // ];
    }

    if (errorCount == 0 || passedReInput || answerIsPreventiveCorrect) {
      for (var kv in termErrorCounter.entries) {
        if (kv.value == 0 || answerIsPreventiveCorrect) {
          // TODO: баллы начислять только после окна с исправлением ошибок
          kv.key.writeErrorCounter -= 1;
        } else {
          kv.key.writeErrorCounter += 1;
          if (kv.key.writeErrorCounter % 3 == 2 &&
              kv.key.writeErrorCounter > 2) {
            kv.key.chooseErrorCounter += 1;
          }
        }
        kv.key.watchCount += 1;
        termsPr.changedInLearningIterationTermsList!.add(kv.key);
      }
    }

    if (errorCount == 0 || passedReInput || answerIsPreventiveCorrect) {
      learnNavPr.activePageNumber += 1;
    } else {
      if (!isAnotherWrite) {
        writtenWord = "";
      }
    }
  }

  void isUserCompletedInput() {}
}
```

Листинг М.30 – ChoiceLearnButtonsProvider

```
import 'dart:core';

// import
'package:the_remember/src/domain_layer/providers/folders/module/terms_in_modu
le.dart';
import 'package:the_remember/src/repositoris/db_data_source/term.dart';

import '../../../../../../../urils/profilers/abstract.dart';
import '../../terms_in_module.dart';
import '../learning_navigation.dart';

class ChoiceLearnButtonsProvider extends ModChangeNotifier {
  final TermsInModuleProvider termsPr;
  final LearnScreensNavigationProvider learnNavPr;
  late TermEntityDbDS? _buttonClicked;
  TermEntityDbDS get  _correctButton =>
termsPr.learningIterationTermsList![learnNavPr.activePageNumber];

  TermEntityDbDS get correctButton => _correctButton;

  ChoiceLearnButtonsProvider(this.termsPr, this.learnNavPr): super()  {
    // init();
  }

  @override
  void init({bool isRealInit = false}) {
    _buttonClicked = null;
```

```
    super.init(isRealInit: isRealInit);
  }

  set buttonClicked(TermEntityDbDS? val) {
    if (this._buttonClicked != val) {
      this._buttonClicked = val;

      notifyListeners();
    }
  }

  TermEntityDbDS? get buttonClicked => _buttonClicked;
}
```