

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет

вычислительной техники

Кафедра

«Вычислительная техника»

Направление подготовки: 09.03.01 «Информатика и вычислительная техника»

Профиль подготовки: «Программное обеспечение средств вычислительной
техники и автоматизированных систем»

БАКАЛАВРСКАЯ РАБОТА

на тему

**Бизнес-проект «ИНТЕЛЛЕКТУАЛЬНАЯ
ОБРАЗОВАТЕЛЬНАЯ ПЛАТФОРМА»**

Студент

(подпись, дата)

Шнайдер Кирилл Сергеевич
(ФИО полностью)

Руководитель

(подпись, дата)

Митрохин М.А.
(фамилия, инициалы)

Нормоконтролёр

(подпись, дата)

Бычков А.С.
(фамилия, инициалы)

Работа допущена к защите (протокол зас. кафедры от 22.05.2026 г. № 14)

Заведующий кафедрой

(подпись)

Митрохин М.А.
(фамилия, инициалы)

Работа защищена с отметкой отлично (протокол заседания ГЭК от

02 июня 2026 г. № 1)

Секретарь ГЭК

(подпись)

Гурин Е.И.
(фамилия, инициалы)

Пенза 2026

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

Факультет

вычислительной техники

Кафедра

«Вычислительная техника»

«Утверждаю»

Заведующий кафедрой

М.А. Митрохин

« 07 » 04 2026 г.

**ЗАДАНИЕ
НА БАКАЛАВРСКУЮ РАБОТУ**

1. Студент Шнайдер Кирилл Сергеевич гр. 22ВВП2 факультета ФВТ

2. Тема работы Бизнес-проект «Интеллектуальная образовательная платформа»

Тема утверждена приказом ПГУ № 347/о от "07" апреля 2026 г.

3. Руководитель бакалаврской работы Митрохин М.А.

4. Задание на работу (назначение разработки, исходные данные и т.п.)

1) Назначение: серверная часть интеллектуальной образовательной платформы

2) Среда выполнения: Windows 10, Linux, Mac OS.

Языковые средства разработки: GoLang, SQL, UML.

Инструментальные средства: JetBrains GoLand, Swagger, Docker, PostgreSQL, Draw.io, Kafka, Redis.

Технологии разработки: функциональная парадигма, микросервисная архитектура

3) Основные функции приложения:

а) регистрация и авторизация;

б) добавление, изменение, удаление курсов, лекций;

в) проверка речевых показателей;

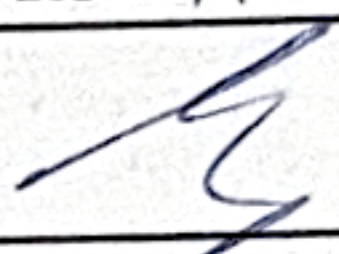
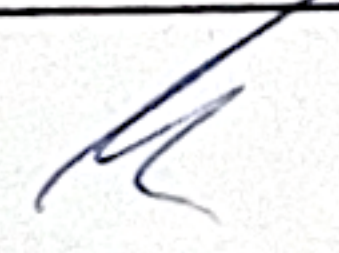
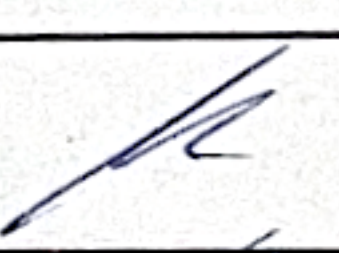
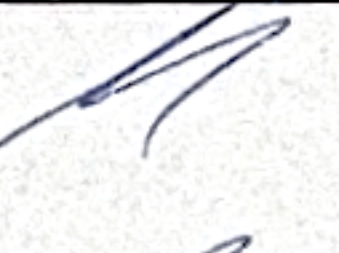
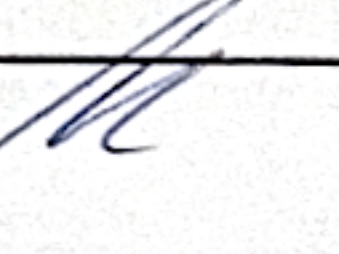
г) добавление, изменение пользовательской информации;

е) добавление, изменение, удаление лекций, привязка к курсу

5. Перечень подлежащих разработке вопросов

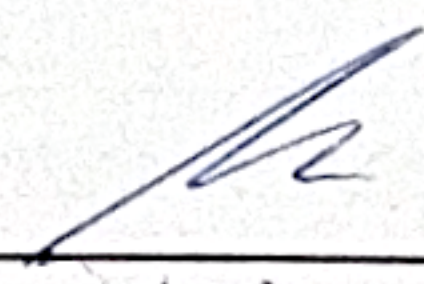
Требуется выполнить анализ следующих предметных областей: изучить существующие подходы к проектированию и реализации backend-сервисов для образовательных платформ, проанализировать методы управления данными курсов и связанным контентом, а также рассмотреть подходы к валидации и обработке ошибок в REST API. Необходимо исследовать современные практики модульного тестирования Go приложений и выбрать наиболее подходящие инструменты тестирования HTTP-обработчиков и сервисного слоя. Также необходимо изучить функциональные возможности используемого технологического стека языка Go и вопросы построения архитектуры приложения с разделением на слои handler, service, repository. Дополнительно требуется рассмотреть аспекты реализации CRUD-операций, проверки бизнес-правил, а также возможности расширения и интеграции backend-модуля с клиентскими приложениями и внешними сервисами.

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1.	Изучение предметной области	20%	07.04.26 – 13.04.26	
2.	Анализ требований	10%	13.04.26 – 17.04.26	
3.	Выбор технологий, применяемых при проектировании и разработке	10%	17.04.26 – 24.04.26	
4.	Проектирование архитектуры	20%	24.04.26 – 28.04.26	
5.	Разработка интеллектуальной образовательной платформы	30%	28.04.26 – 17.05.26	
6.	Оформление пояснительной записки	10%	17.05.26 – 21.05.26	

Дата выдачи задания "07" апреля 2026 г.

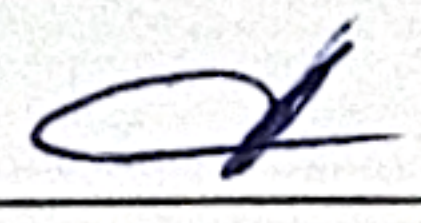
Руководитель бакалаврской работы


(подпись, дата)

Задание к исполнению принял студент


(подпись, дата)

Работу к защите допустить
Декан факультета

 Л.Р. Филова
(подпись, дата)

26.05.2026

Реферат

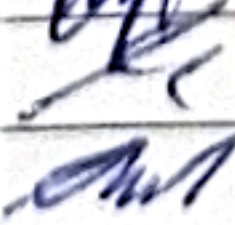
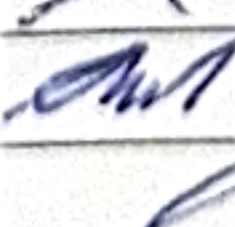
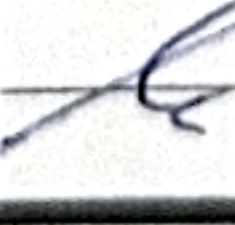
Бакалаврская работа содержит 102 листа, 24 рисунка, 9 таблиц, 18 источников и 4 приложения.

ИНТЕЛЛЕКТУАЛЬНАЯ ОБРАЗОВАТЕЛЬНАЯ ПЛАТФОРМА, WEB-ПРИЛОЖЕНИЕ, МИКРОСЕРВИСНАЯ АРХИТЕКТУРА, GOLANG, POSTGRESQL, APACHE KAFKA, UML

Цель работы заключается в разработке платформы, которая предоставляет пользователю возможность получать доступ к образовательным ресурсам разной направленности, в том числе к курсам с программным инструментом анализа речевых ошибок и выявления дефектов речи в записи голоса.

В работе рассматриваются принципы построения распределённых образовательных систем и типовые паттерны взаимодействия микросервисов. В процессе выполнения дипломного проектирования предложена микросервисная архитектура с выделением API Gateway как единой точки входа для клиентских приложений. Шлюз обеспечивает маршрутизацию запросов, единообразную аутентификацию и авторизацию, ограничение нагрузки и согласованный формат ответов. Специализированные сервисы отвечают за доменные области – управление пользователями и профилями, учебными курсами и материалами, обращение к сервису диагностики речи. Разработаны и описаны контракты взаимодействия между gateway и сервисами. Отличительной особенностью решения является сочетание централизованного управления трафиком на уровне шлюза с декомпозицией бизнес-логики по микросервисам, что повышает отказоустойчивость и масштабируемость серверной части интеллектуальной образовательной платформы.

ПГУ 1.09.03.01.50.001 ПЗ

					ПГУ 1.09.03.01.50.001 ПЗ			
Изм.	Лист	№ докум.	Подпи	Дата	Бизнес-проект «Интеллектуальная образовательная платформа» Пояснительная записка	Лит.	Лист	Листов
Разраб.		Шнайдер К.С.		18.05.2026				
Провер.		Митрохин М.А.		18.05.2026			2	102
Н. Контр.		Бычков А.С.		25.05.2026		ФВТ, каф. ВТ гр. 22ВВП2		
Утверд.		Митрохин М.А.		19.05.2026				

Оглавление

Перечень сокращений и обозначений	5
Введение	6
1 Интеллектуальная образовательная платформа	8
1.1 Постановка задачи	8
1.2 Анализ предметной области.....	9
1.2.1 Выбор среды разработки.....	16
1.2.2 Анализ требований.....	18
1.2.2.1 Анализ функциональных требований	18
1.2.2.2 Анализ нефункциональных требований	23
1.2.3 Постановка задачи на разработку приложения.....	24
1.2.4 Выбор архитектуры приложения.....	24
1.3 Проектирование и разработка	27
1.3.1 Логическая модель базы данных	29
1.3.2 Обобщенная структура сервиса.....	32
1.3.3 Разработка серверной части.....	33
1.3.4 Разработка физической модели базы данных	35
1.3.5 Разработка архитектурных слоев внутри микросервисов.....	38
1.3.5.1 Разработка слоя представления данных в микросервисах	38
1.3.5.2 Разработка слоя бизнес-логики в микросервисах.....	40
1.3.5.3 Разработка слоя обращения к базе данных в микросервисах	41
1.4 Тестирование серверной части образовательной платформы	44
1.4.1 Функциональное тестирование добавления курсов и их компонентов...	44
1.4.2 Модульное тестирование добавления курсов и их компонентов.....	46
2 Охрана труда.....	49
2.1 Характеристика потенциальных рисков при работе с ПК	49
2.2 Оборудование рабочего места.....	49
2.3 Освещённость рабочего места.....	50
2.4 Электробезопасность	51
2.5 Пожарная безопасность	51

2.6 Санитарно-гигиеническая характеристика производства	52
3 Стартап часть	53
3.1 Проблематика и рыночный потенциал.....	53
3.2 Целевая аудитория и каналы взаимодействия	55
3.3 План коммерциализации и экономика проекта	56
3.4 Обоснование достижимости результата	58
Заключение.....	62
Список использованных источников	63
Приложение А – UML – диаграммы.....	65
Приложение Б – Листинг программы	70
Приложение В – Документы, подтверждающие участие в мероприятиях с продуктом «ПРОПС»	93
Приложение Г – Презентация	97

Перечень сокращений и обозначений

Бизнес-логика – совокупность предметных правил и ограничений, задающих порядок работы системы в конкретной предметной области. Репозиторий – компонент слоя доступа к данным, инкапсулирующий операции чтения и записи в хранилище.

Контроллер (handler) – компонент слоя представления, обеспечивающий приём запроса, валидацию данных, вызов бизнес-логики и подготовку ответа.

Слой представления – верхний уровень приложения, через который система принимает запросы и возвращает ответы клиенту.

API (Application Programming Interface) – это формализованный интерфейс взаимодействия между программными компонентами: он задаёт правила, форматы и способы обмена данными.

HTTP – запрос – сообщение клиента серверу, предназначенное для получения, создания, изменения или удаления данных.

Soft skills – гибкие навыки.

REST (Representational State Transfer) – архитектурный стиль для построения распределённых веб-сервисов, где взаимодействие организовано через HTTP-ресурсы и стандартные методы запросов.

					ПГУ 1.09.03.01.50.001 ПЗ	Лист
						5
Изм.	Лист	№ докум.	Подпись	Дата		

Введение

В современном мире цифровые технологии играют ключевую роль в развитии образования. Онлайн-платформы позволяют получать знания вне зависимости от географического положения, времени и индивидуальных особенностей обучающихся. Особенно актуальным становится переход от традиционных форм обучения к интерактивным и персонализированным образовательным решениям.

С развитием технологий искусственного интеллекта появляются новые возможности для повышения эффективности обучения. Одним из перспективных направлений является внедрение интеллектуальных ассистентов, способных анализировать поведение пользователя, адаптировать учебный материал и предоставлять персонализированную обратную связь. Это особенно важно в областях, требующих развития практических навыков, таких как речь, коммуникация и ораторское мастерство.

Современные образовательные платформы, функционирующие в формате онлайн-школ, должны обеспечивать высокую доступность, масштабируемость и устойчивость к нагрузкам. Для этого используются клиент-серверные архитектуры и облачные технологии, позволяющие обрабатывать большое количество пользователей одновременно. Такие системы требуют как надежной программной реализации, так и эффективных механизмов управления и обработки данных.

Одной из ключевых проблем традиционного онлайн-обучения является недостаток качественной обратной связи. Пользователь часто не получает своевременной оценки своих навыков, особенно в таких субъективных областях, как техника речи. Ручная проверка требует значительных временных и человеческих ресурсов, что делает процесс масштабирования затруднительным.

Решением данной проблемы является разработка интеллектуальной образовательной платформы, включающей в себя ИИ-ассистента для анализа и улучшения речевых навыков. Такой ассистент способен оценивать

					ПГУ 1.09.03.01.50.001 ПЗ	Лист
						6
Изм.	Лист	№ докум.	Подпись	Дата		

произношение, интонацию и другие параметры, предоставляя пользователю рекомендации в режиме реального времени. Это позволяет существенно повысить качество обучения и вовлеченность пользователей. Преимущества использования подобной системы включают:

- снижение затрат, автоматизация процессов оценки и обратной связи позволяет сократить необходимость в большом количестве преподавателей и экспертов, снижая операционные расходы;

- повышение эффективности обучения, персонализированные рекомендации и адаптация учебных материалов способствуют более быстрому освоению навыков;

- доступность, пользователи могут обучаться в любое время и из любой точки мира, имея доступ к платформе через интернет;

- интерактивность, использование ИИ делает процесс обучения более живым и вовлекающим, имитируя взаимодействие с реальным преподавателем.

Целью данной работы является разработка интеллектуальной образовательной платформы в формате онлайн-школы с курсами имеющими интегрированные ИИ-ассистенты для развития техник речи. Клиентская часть системы обеспечивает взаимодействие пользователя с курсами, выполнение заданий и получение обратной связи. Серверная часть отвечает за обработку данных, хранение информации и работу алгоритмов искусственного интеллекта. Система должна обеспечивать одновременную работу большого числа пользователей.

1 Интеллектуальная образовательная платформа

1.1 Постановка задачи

Целью выпускной квалификационной работы является создание серверной части образовательной платформы. Разрабатываемое программное обеспечение должно, в первую очередь, обеспечивать прохождение обучающимися учебных программ, а также давать автоматизированную оценку речевых навыков с последующей корректировкой выявленных недостатков. Дополнительно система должна поддерживать комфортную и настраиваемую работу пользователя с учебным контентом: курсами, методическими материалами и контрольными заданиями.

В ходе проектирования и реализации предполагается решение следующих задач:

- внедрение механизмов регистрации и аутентификации пользователей;
- реализация учебного процесса на базе курсов;
- управление учебными курсами;
- поддержка параметрической настройки содержания и структуры курса;
- интеллектуальная проверка речи пользователя с использованием ИИ-ассистента;
- интеграция серверной части с системой хранения данных.

По завершении работ должна быть сформирована программная система микросервисного типа, взаимодействующая по протоколам REST API, с использованием брокера сообщений Apache Kafka.

Минимальные и рекомендованные требования в таблице 1.

Таблица 1 – Требования к программному обеспечению

Характеристика	Минимальные требования	Рекомендованные требования
Процессор	x64 с 1.5 ГГц	x64 с 3 ГГц
Оперативная память	2 Гб	8 Гб

Продолжение таблицы 1

Характеристика	Минимальные требования	Рекомендованные требования
Тип и объем жесткого диска	HDD с 60 Гб	SSD с 512 Гб
Пропуская способность сети	100 Мбит/сек	1 Гбит/сек
Операционная система	Ubuntu 18.04	Ubuntu 20.04
Обязательное ПО	Docker 3.7	Docker 3.9

1.2 Анализ предметной области

По результатам анализа рынка программных решений, близких по функциональным возможностям к разрабатываемой платформе, для сравнительного исследования отобраны платформы Coursera, Udemy, Skillbox и Stepik.

Общие сведения о выбранных аналогах образовательной платформы приведены в таблице 2.

Таблица 2 – Краткая описание общих сведений об аналогах

Название	Организация	Описание
Coursera	Coursera Inc	Образовательная онлайн-платформа, предоставляющая услуги дистанционного обучения в соответствии с современными международными образовательными стандартами. Обеспечивает доступ к курсам ведущих университетов и организаций, позволяет отслеживать прогресс обучения на всех этапах, реализует гибкие траектории освоения учебного материала и предоставляет возможность получения сертификации по результатам прохождения курсов.
Udemy	Udemy Inc	Образовательная платформа, предоставляющая широкий спектр онлайн-курсов по различным направлениям подготовки в соответствии с актуальными требованиями рынка труда. Обеспечивает пользователям возможность выбора образовательного контента, контроля процесса обучения, а также самостоятельного освоения материалов в индивидуальном темпе.

Продолжение таблицы 2

Название	Организация	Описание
Skillbox	ООО «Скилбокс»	Образовательная онлайн-платформа, ориентированная на реализацию программ профессионального обучения с акцентом на практическое применение знаний. Обеспечивает сопровождение обучающихся в процессе обучения, включает выполнение практических заданий с участием наставников и позволяет формировать профессиональные компетенции, востребованные в современных условиях.
Stepik	ООО «Цифровые образовательные решения»	Образовательная платформа, реализующая интерактивные методы обучения с применением автоматизированной проверки заданий. Обеспечивает контроль уровня усвоения материала, поддерживает элементы адаптивного обучения и ориентирована на формирование практических навыков, преимущественно в технических и естественно-научных дисциплинах.

У отобранных аналогов различаются состав функций и модель взаимодействия с пользователем. В ходе сравнения для каждого программного средства фиксируются положительные стороны и ограничения.

Coursera представляет собой международный образовательный ресурс с каталогом дистанционных курсов от университетов и коммерческих организаций. Платформа ориентирована на получение академических и прикладных профессиональных компетенций. Интерфейс сервиса показан на рисунке 1.

К преимуществам Coursera можно отнести:

- широкий выбор курсов от ведущих университетов;
- наличие сертификатов и программ профессиональной переподготовки;
- удобный интерфейс и гибкий график обучения.

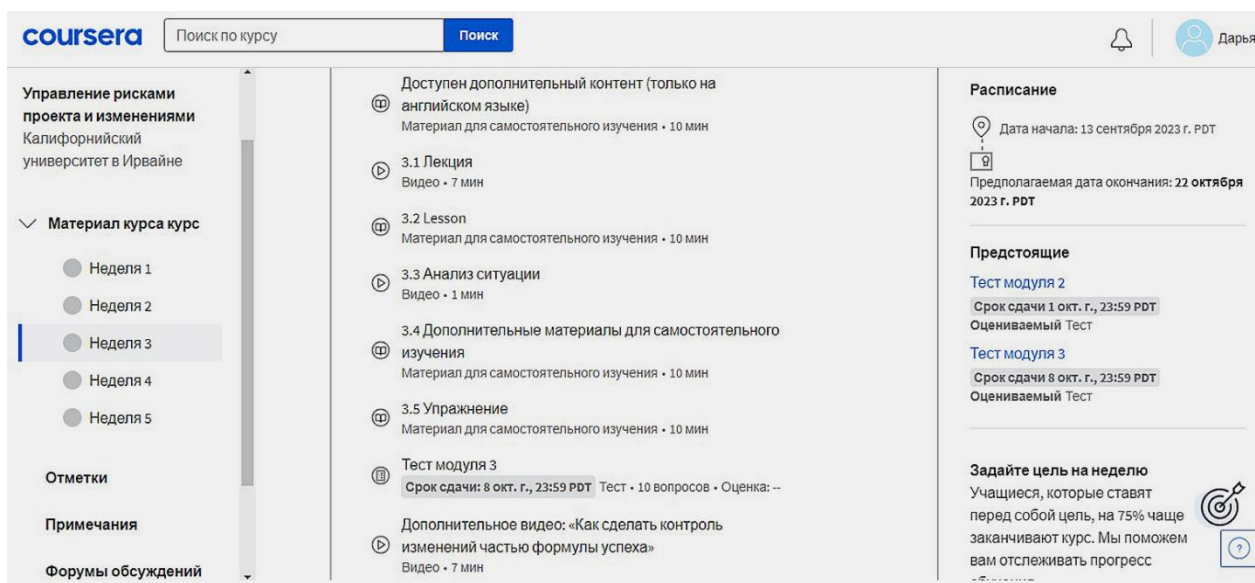


Рисунок 1 – Интерфейс образовательной платформы Coursera

К минусам можно отнести:

- ограниченные возможности практической отработки навыков;
- часть курсов доступна только по платной подписке;
- слабая персонализация обучения.

Udemy – онлайн-платформа для обучения, предоставляющая курсы по широкому спектру тем, созданные независимыми преподавателями.

К основным плюсам Udemy можно отнести:

- разнообразие учебных программ в разных предметных областях;
- умеренный уровень стоимости образовательных материалов;
- возможность обучения в удобном темпе.

К минусам можно отнести:

- неоднородное качество курсов;
- ограниченная практическая составляющая;
- отсутствие глубокой обратной связи.

Интерфейс платформы показан на рисунке 2.

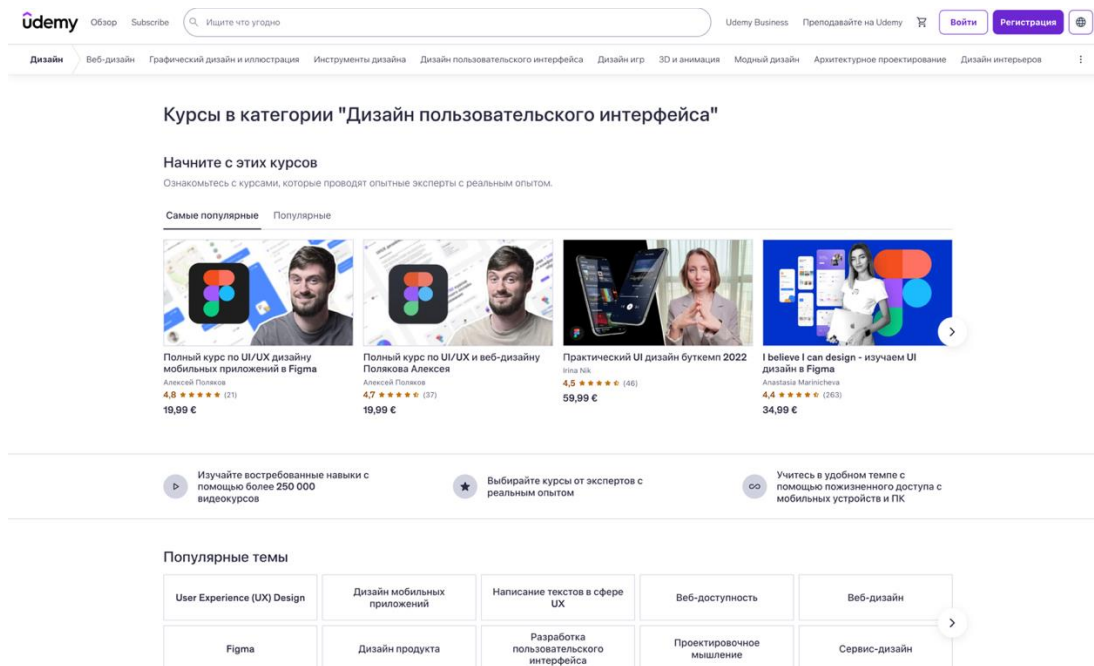


Рисунок 2 – Интерфейс образовательной платформы Udeemy

Skillbox – российская образовательная платформа, ориентированная на получение практических навыков и освоение цифровых профессий. Обучение часто сопровождается поддержкой наставников. Интерфейс платформы показан на рисунке 3.

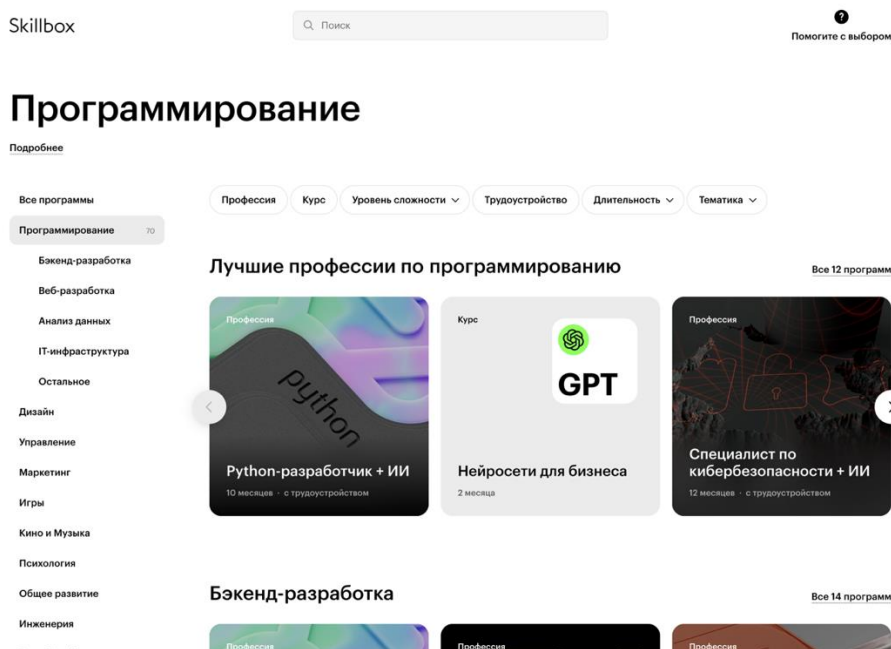


Рисунок 3 – Интерфейс образовательной платформы Skillbox

К преимуществам Skillbox можно отнести:

- наличие практических заданий с проверкой наставниками;
- ориентированность на востребованные профессии;
- поддержка студентов в процессе обучения.

К минусам можно отнести:

- высокая стоимость обучения;
- длительность образовательных программ;
- ограниченная автоматизация проверки заданий.

Stepik – образовательная платформа, ориентированная на интерактивное обучение, особенно в технических и программных дисциплинах. Интерфейс платформы показан на рисунке 4.

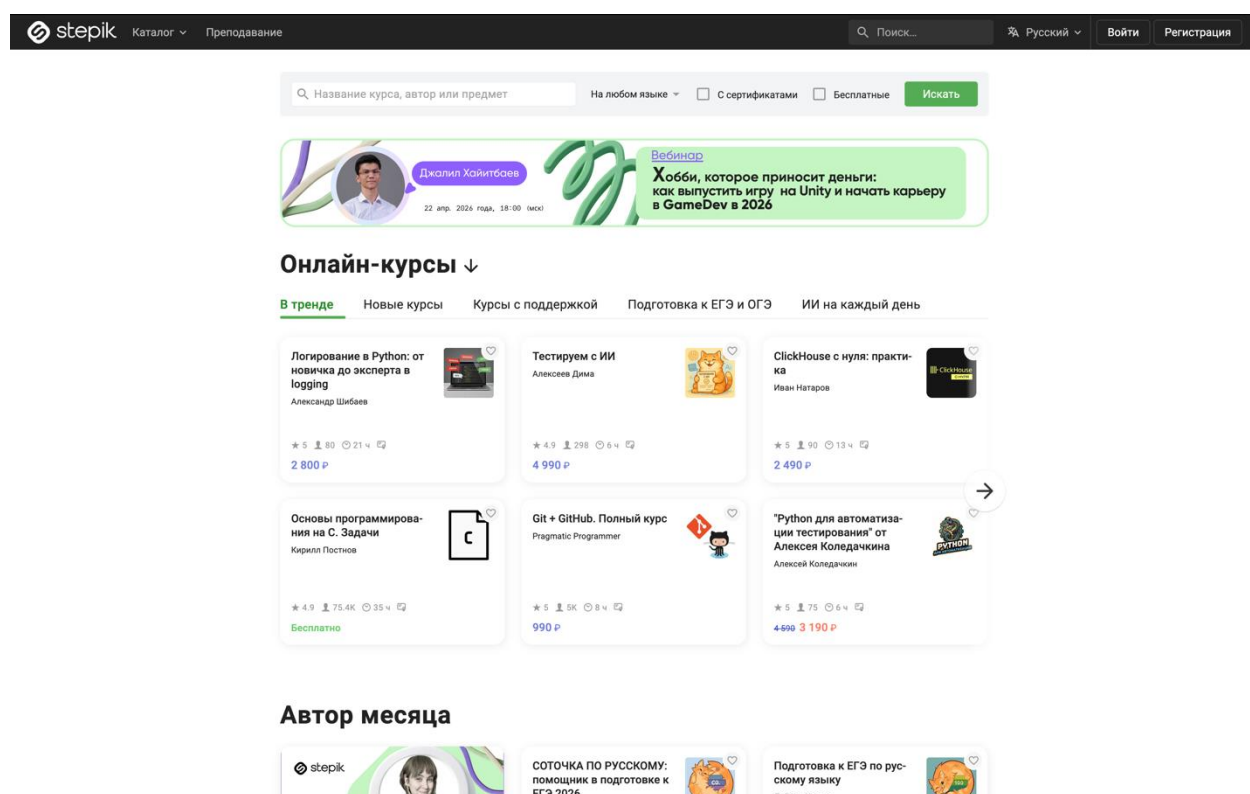


Рисунок 4 – Интерфейс платформы Stepik

К преимуществам Stepik можно отнести:

- интерактивные задания с автоматической проверкой;
- поддержка адаптивного обучения;

– удобство для изучения технических дисциплин.

К минусам можно отнести:

– ограниченная работа с речевыми навыками;

– узкая направленность большинства курсов;

– отсутствие комплексного анализа пользовательских навыков.

Для решения задачи сравнительного анализа образовательных платформ в данной работе используется Метод анализа иерархий (МАИ), разработанный американским математиком Томасом Саати.

Данный выбор обусловлен спецификой объекта исследования. Оценка образовательных платформ является сложной многокритериальной задачей. В процессе одновременно присутствуют как количественные параметры, так и качественные, трудноизмеримые характеристики

Для проведения сравнительного анализа аналогов интеллектуальной образовательной платформы была сформирована система критериев, позволяющая всесторонне оценить существующие решения. Выбор критериев обусловлен необходимостью учета ключевых характеристик программных продуктов, влияющих на эффективность их использования, а также на степень удовлетворения потребностей пользователей.

Критерии отбора сформированы с учетом следующих факторов:

– значимость для конечного пользователя;

– влияние на функциональные возможности приложения;

– распространенность использования в аналогичных исследованиях;

– возможность однозначной интерпретации результатов оценки.

Дополнительно обоснование выбора критериев подтверждается результатами проведенного эмпирического исследования с участием потенциальных пользователей. В ходе исследования были выявлены наиболее значимые характеристики приложения с точки зрения удобства использования, функциональности и пользовательского опыта. Полученные данные позволили уточнить и приоритизировать критерии оценки, что повысило их релевантность

					ПГУ 1.09.03.01.50.001 ПЗ	Лист
						1
Изм.	Лист	№ докум.	Подпись	Дата		4

и практическую значимость.

Для количественной оценки степени соответствия каждому критерию используется трёхбалльная шкала:

- 0 баллов – полное несоответствие критерию;
- 1 балл – частичное соответствие критерию;
- 2 балла – полное соответствие критерию.

Выбор трёхуровневой шкалы оценивания обусловлен необходимостью обеспечения баланса между точностью оценки и простотой её применения. Данная шкала позволяет снизить уровень субъективности при проведении анализа, обеспечивает наглядность представления результатов и упрощает процедуру сравнения различных аналогов.

Результаты сравнительного анализа приведены в таблице 3.

Таблица 3 – Сравнительная характеристика отобранных аналогов

Критерии	Coursera	Udemy	Skillbox	Stepik
Интерактивное обучение	1	1	1	2
Практическая направленность	1	1	2	1
Автоматизированная проверка знаний	1	0	0	2
Персонализация обучения	1	1	1	1
Итого	4	3	4	6

На основании проведённого сравнения можно заключить, что отечественные и зарубежные образовательные сервисы демонстрируют существенные сильные стороны: разнообразие учебных программ, зрелую организацию учебного процесса и применение практико-ориентированных заданий. Однако полное соответствие сформулированным требованиям, в особенности по направлению контроля и коррекции речи, для рассмотренных аналогов не выявлено. Это обосновывает необходимость создания

образовательной платформы со следующими функциональными возможностями:

- реализацию интерактивных курсов с акцентом на практическое применение знаний;
- анализ речевых навыков пользователя с использованием технологий искусственного интеллекта;
- предоставление персонализированной обратной связи в режиме реального времени;
- адаптацию учебного процесса под индивидуальные особенности пользователя;
- обеспечение высокой эффективности и удобства взаимодействия.

1.2.1 Выбор среды разработки

Для разработки серверных компонентов на языке Go, для проектирования, отладки и сопровождения баз данных требовались специализированные инструменты с поддержкой рефакторинга, интеграции с системами контроля версий и удобной работы с SQL.

Visual Studio Code – кроссплатформенный редактор с расширениями для Go и баз данных. К достоинствам относятся: бесплатность, большое сообщество, гибкость за счёт расширений. В свою очередь к недостаткам можно отнести сильно зависящую от набора плагинов функциональность для Go и SQL; также рефакторинг и анализ проекта уступают специализированным IDE; настройка занимает время.

GoLand – интегрированная среда разработки JetBrains для языка Go. Достоинствами является встроенный анализатор кода, навигация по проекту и символам, рефакторинги, отладчик, тесты, профилирование и интеграция с Docker, Kubernetes, единообразный пользовательский интерфейс с другими продуктами JetBrains. Недостатками является коммерческая лицензия и повышенные требования к ресурсам по сравнению с лёгкими редакторами.

DBeaver – универсальный клиент для работы с различными СУБД.

					ПГУ 1.09.03.01.50.001 ПЗ	Лист
						1
Изм.	Лист	№ докум.	Подпись	Дата		6

Достоинствами является широкая поддержка СУБД, открытая модель распространения Community-версии, визуализация схем и данных. В бесплатной версии функционал значительно ограничен, что является главным недостатком.

DataGrip – среда JetBrains для работы с базами данных и SQL. Достоинства: интеллектуальное автодополнение и проверка SQL, удобная работа с несколькими подключениями и диалектами, визуальное редактирование схем, сравнение данных и миграций, согласованность интерфейса с GoLand. Недостатками также является коммерческая лицензия, для задач, где достаточно простых запросов, функционал может избыточен.

В таблице 2 представлена сводная информация о достоинствах и недостатках сред имитационного моделирования транспортных потоков.

Таблица 2 – Сводная таблица достоинств и недостатков сред моделирования

Программное обеспечение	Достоинство	Недостаток
Visual Studio Code	Бесплатность, гибкость за счёт расширений.	Зависимость от плагинов, слабый встроенный рефакторинг и анализ для Go и SQL.
GoLand	Рефакторинг, отладка, тесты и анализ кода Go «из коробки».	Платная лицензия и выше требования к ресурсам.
DBeaver	Много СУБД, есть бесплатная редакция.	Расширенные функции платные, слабая связка с IDE
DataGrip	Продвинутая работа с SQL и схемами, единый стиль с GoLand	Платная лицензия, избыточность для тривиальных сценариев.

Таким образом, для разработки на Go выбрана среда GoLand как специализированная IDE с полным циклом поддержки языка и инструментов качества кода [10]. Для проектирования запросов, сопровождения схем БД и анализа данных выбран DataGrip как профессиональный SQL-клиент той же экосистемы, что снижает порог переключения между задачами backend и БД и обеспечивает согласованные сочетания клавиш, поиск и работу с проектами. Совокупность GoLand и DataGrip соответствует требованиям проекта по удобству разработки, отладки и сопровождения данных.

1.2.2 Анализ требований

1.2.2.1 Анализ функциональных требований

Диаграмма вариантов использования описывает отношения между прецедентами и действующими лицами, вовлечёнными в эксплуатацию системы. Указанные диаграммы не используются для детализации архитектуры и не отображают внутреннее устройство программного комплекса [1]. Их назначение зафиксировать на концептуальном уровне, какие взаимодействия пользователей с системой требуются для выполнения прикладных задач.

При формировании функциональных требований к интеллектуальной образовательной платформе разработана диаграмма вариантов использования со следующими прецедентами:

- аутентификация;
- завершение сеанса;
- администрирование курсов;
- администрирование учётных записей пользователей;
- администрирование заданий;
- администрирование тестов;
- контроль речи пользователя;
- формирование и получение отчёта;
- доставка уведомлений на e-mail.

Построенная диаграмма представлена на рисунке 5.

Спецификация прецедента «Войти в систему» приведена на рисунке 6.

На основании выполненного анализа функциональных требований определён базовый набор реализуемых возможностей, достаточный для поддержки заявленных бизнес-сценариев образовательной платформы.

Спецификация прецедента «Войти в систему» приведена на рисунке 6.

Спецификация прецедента «Просмотр каталога курсов» приведена на рисунке 7.

					ПГУ 1.09.03.01.50.001 ПЗ	Лист
						1
Изм.	Лист	№ докум.	Подпись	Дата		8

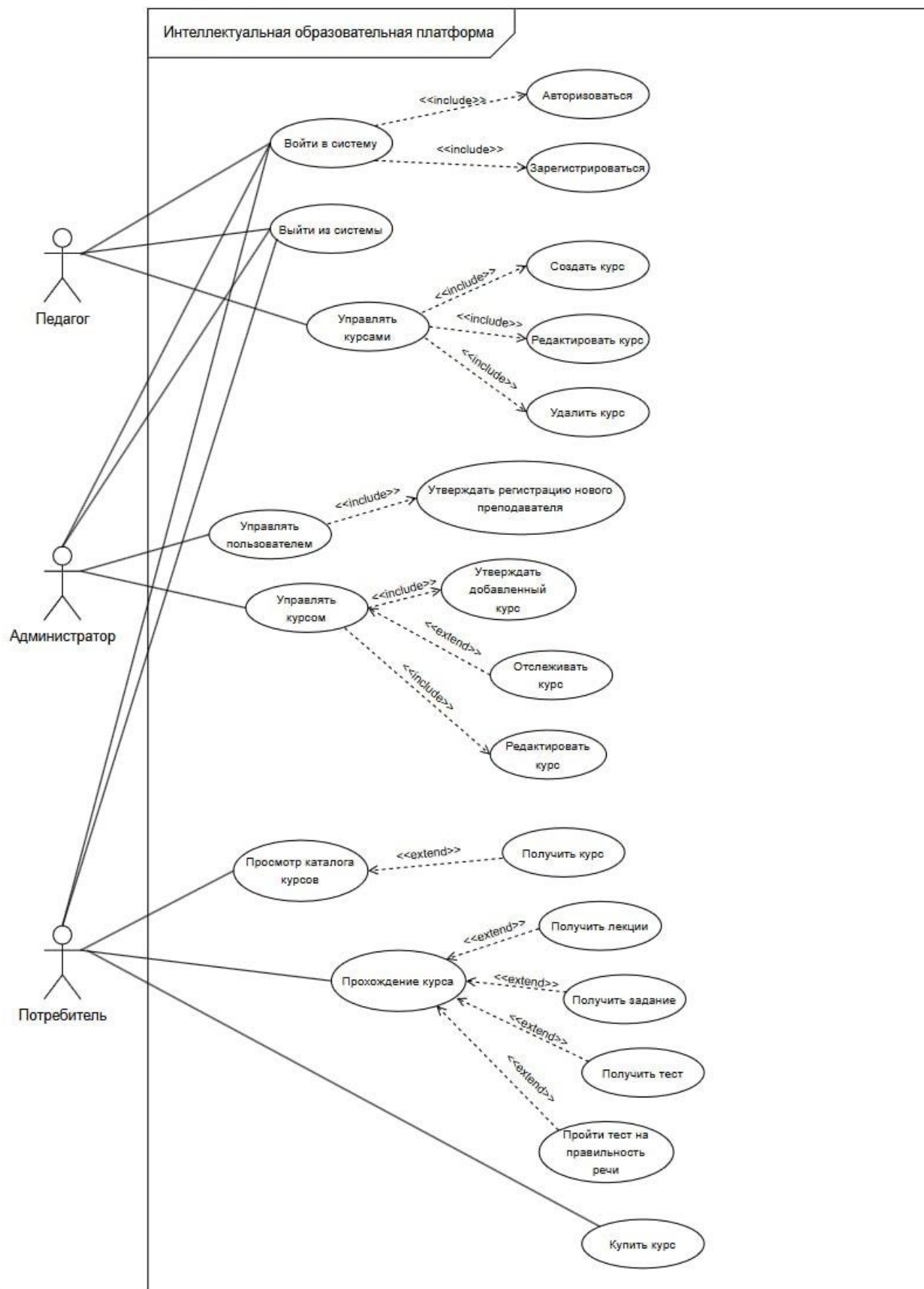


Рисунок 5 – Диаграмма вариантов использования образовательной платформы

Прецедент: Войти в систему
ID: 1
Главные актеры: Пользователь любой группы
Второстепенные актеры: Нет
Предусловия: 1. Пользователь не был авторизован/зарегистрирован
Основной поток: 1. Пользователь переходит на страницу входа в приложение 2. Пользователь вводит данные для входа в существующий аккаунт 3. Пользователь получает доступ к функционалу приложения
Постусловия: 1. Пользователь вошел в приложение, ему доступен функционал
Альтернативные потоки: 2.а. Пользователь вводит данные для регистрации (логин, пароль) 2.б. Пользователь переходит к форме заполнения дополнительной информации о себе

Рисунок 6 – Спецификация варианта использования «Войти в систему»

Прецедент: Просмотр каталога курсов
ID: 2
Главные актеры: Пользователь класса потребитель
Второстепенные актеры: Нет
Предусловия: 1. Потребитель авторизован в системе
Основной поток: 1. Потребитель переходит в каталог курсов 2. Система запрашивает список доступных курсов 3. Система отображает каталог курсов с названиями, описаниями и стоимостью 4. Потребитель просматривает и фильтрует курсы по категориям или названию 5. Потребитель выбирает интересующий курс для детального просмотра
Постусловия: 1. Потребитель ознакомлен с доступными курсами
Альтернативные потоки: 3.а. Доступных курсов нет — система отображает сообщение об отсутствии курсов

Рисунок 7 – Спецификация варианта использования «Просмотр каталога курсов»

Спецификация прецедента «Получение курса» приведена на рисунке 8.

Спецификация прецедента «Пройти курс» приведена на рисунке 9.

Спецификация прецедента «Пройти тест на правильность речи» приведена на рисунке 10.

Прецедент: Получение курса
ID: 3
Главные актеры: Пользователь класса потребитель
Второстепенные актеры: Нет
Предусловия: 1. Потребитель авторизован в системе 2. Потребитель выбрал курс из каталога
Основной поток: 1. Потребитель открывает страницу выбранного курса 2. Система запрашивает детальную информацию о курсе 3. Система отображает описание курса, список лекций, заданий и тестов 4. Если курс бесплатен — потребитель нажимает «Записаться» 5. Если курс платный — потребитель переходит к оплате 6. Система записывает потребителя на курс
Постусловия: 1. Потребитель записан на курс 2. Курс доступен потребителю для прохождения
Альтернативные потоки: 4.a. Потребитель уже записан на курс — система отображает соответствующее сообщение

Рисунок 8 – Спецификация варианта использования «Получение курса»

Прецедент: Пройти курс
ID: 4
Главные актеры: Пользователь класса потребитель
Второстепенные актеры: Нет
Предусловия: 1. Потребитель авторизован в системе 2. Потребитель записан на курс
Основной поток: 1. Потребитель открывает страницу записанного курса 2. Система отображает прогресс прохождения и список материалов 3. Потребитель выбирает лекцию из списка 4. Система запрашивает содержимое лекции и отображает его (текст, видео, изображения) 5. Потребитель изучает материал и отмечает лекцию как просмотренную 6. Потребитель выбирает задание, доступное после прохождения лекции 7. Система запрашивает содержимое задания через системный шлюз и отображает его условие 8. Потребитель выполняет задание; система сохраняет его и передаёт на проверку 9. Потребитель выбирает доступный тест 10. Система запрашивает вопросы теста и отображает их с вариантами ответов 11. Потребитель отвечает на вопросы и отправляет тест 12. Система проверяет ответы, рассчитывает и сохраняем 13. После прохождения всех материалов курс считается завершённым
Постусловия: 1. Прогресс потребителя сохранён 2. При полном прохождении — курс отмечен как завершённый
Альтернативные потоки: 4.a. Лекция недоступна — система отображает сообщение об ошибке 6.a. Лекция не пройдена — задание недоступно, доступ заблокирован 8.a. Потребитель отправил незаполненный ответ — система отображает ошибку 9.a. Задание не выполнено — тест недоступен 12.a. Потребитель не набрал минимальный порог — система предлагает пройти тест повторно

Рисунок 9 – Спецификация варианта использования «Пройти курс»

Прецедент: Пройти тест на правильность речи
ID: 5
Главные актеры: Пользователь класса потребитель
Второстепенные актеры: Нет
Предусловия: 1. Потребитель авторизован 2. Потребитель записан на курс 3. Текущий раздел курса предусматривает проверку речи
Основной поток: 1. Потребитель открывает задание на проверку речи 2. Система отображает текст или задание для устного воспроизведения 3. Потребитель записывает голосовой ответ 4. Система передаёт запись на анализ через системный шлюз 5. Система возвращает результат анализа корректности речи 6. Система отображает результат потребителю
Постусловия: 1. Результат проверки речи сохранён 2. Прогресс потребителя обновлён
Альтернативные потоки: 3.а. Потребитель не предоставил запись — система отображает ошибку 5.а. Анализ недоступен — система сообщает о временной недоступности функции

Рисунок 10 – Спецификация варианта использования «Пройти тест на правильность речи»

Спецификация прецедента «Купить курс» приведена на рисунке 11.

Прецедент: Купить курс
ID: 6
Главные актеры: Пользователь класса потребитель
Второстепенные актеры: Нет
Предусловия: 1. Потребитель авторизован 2. Потребитель выбрал платный курс из каталога 3. Курс является платным
Основной поток: 1. Потребитель нажимает на кнопку «Купить» 2. Система отображает страницу оплаты с итоговой суммой 3. Потребитель вводит платёжные данные 4. Система передаёт данные оплаты в систему 5. Система обрабатывает платёж и подтверждает его 6. Система записывает потребителя на курс 7. Система отображает подтверждение оплаты и предоставляет доступ к курсу
Постусловия: 1. Платёж проведён 2. Потребитель записан на курс и имеет к нему полный доступ
Альтернативные потоки: 5.а. Платёж отклонён — система отображает сообщение об ошибке оплаты и предлагает попробовать снова или выбрать другой способ оплаты

Рисунок 11 – Спецификация варианта использования «Купить курс»

Спецификация прецедента «Управлять курсом» приведена на рисунке 12.

Прецедент: Управлять курсами
ID: 7
Главные актеры: Пользователь класса преподаватель
Второстепенные актеры: Нет
Предусловия: 1. Педагог авторизован в системе 2. Педагог имеет подтвержденную учетную запись преподавателя
Основной поток: 1. Педагог переходит в раздел управления курсами 2. Система отображает список курсов, принадлежащих педагогу 3. Педагог выбирает одно из действий: создать, редактировать или удалить курс
Постусловия: 1. Выполнено выбранное действие над курсом
Альтернативные потоки: 3.а. У педагога нет созданных курсов — система отображает пустой список с предложением создать первый курс

Рисунок 12 – Спецификация варианта использования «Управлять курсом»

1.2.2.2 Анализ нефункциональных требований

В ходе формирования нефункциональных требований к интеллектуальной образовательной платформе необходимо определить архитектурную модель, удовлетворяющую эксплуатационным характеристикам. Установлено, что система должна быть гибкой, масштабируемой, отказоустойчивой и производительной.

Указанный набор свойств наиболее полно реализуется при применении микросервисного подхода. Данный подход основан на разделении приложения на независимые сервисы с автономным развёртыванием.

Далее приведена аргументация выбора архитектуры по каждому из критериев.

Гибкость – изменения локализуются в отдельном сервисе и не требуют переработки всей кодовой базы.

Масштабируемость и производительность – горизонтальное масштабирование выполняется выборочно для «узких мест», без пропорционального увеличения ресурсов всей инфраструктуры.

Отказоустойчивость – изоляция сервисов ограничивает распространение сбоев и поддерживает доступность остальной функциональности.

1.2.3 Постановка задачи на разработку приложения

Выпускная квалификационная работа направлена на проектирование и реализацию серверного программного обеспечения интеллектуальной образовательной платформы. Ключевые возможности сервиса включают дистанционное прохождение курсов, контроль речевых навыков и их коррекцию средствами интеллектуального ассистента, а также работу с учебным контентом и проверку знаний через клиентское приложение.

К числу основных задач разработки относятся:

- аутентификация и учёт пользователей;
- ведение учебного процесса на базе курсов;
- администрирование каталога;
- параметрическая настройка структуры и содержания курса;
- интеллектуальная обработка речи обучающегося;
- построение API для клиентской части;
- обеспечение доступа серверных модулей к СУБД.

Итогом является микросервисный программный комплекс с взаимодействием по REST с использованием брокера сообщений Apache Kafka, с возможностью сохранения данных с СУБД.

1.2.4 Выбор архитектуры приложения

На основании сформулированных функциональных и нефункциональных требований, в работе обоснован выбор микросервисной архитектуры. Данная модель сочетает приемлемую трудоёмкость внедрения с возможностью независимого подбора технологического стека для каждого сервиса. Это повышает адаптивность решения при развитии отдельных подсистем. Указанные характеристики соответствуют задачам создания распределённой прикладной системы с централизованным управлением и контролируемым сопровождением.

					ПГУ 1.09.03.01.50.001 ПЗ	Лист
						2
Изм.	Лист	№ докум.	Подпись	Дата		4

Для фиксации состава программных модулей и их взаимодействия построена диаграмма компонентов серверной части платформы. На диаграмме отображаются физические элементы развёртывания и связи между ними: микросервисы, реализуемые с использованием различных языков и фреймворков, а также компоненты персистентного хранения [2]. Диаграмма применяется для обобщённого описания архитектуры, анализа зависимостей и оформления проектной документации на этапах проектирования и передачи результатов разработки.

Состав серверной части, отражённый на диаграмме компонентов, декомпозирован в набор автономных микросервисов. Каждый модуль закреплён за собственным фрагментом предметной области и выполняет ограниченный круг задач:

- шлюз – централизованный вход в backend, стыковка с клиентским приложением;
- модуль идентификации – учётная запись, проверка подлинности и разграничение доступа;
- модуль профилей – сопровождение сведений о пользователях и их атрибутов;
- модуль контента – размещение учебных материалов и выдача их потребителям сервисов;
- модуль персонализации – накопление и применение пользовательских настроек при конструировании курсов;
- платёжный модуль – оформление покупок и проведение расчётных операций;
- модуль речевой аналитики – обработка аудио ввода нейросетевыми средствами и фиксация речевых дефектов;
- модуль оповещений – подготовка и рассылка e-mail-сообщений на контакт, заданный при регистрации.

Подобная предметная декомпозиция снижает связанность компонентов, облегчает их независимое развитие и точечное наращивание вычислительных

мощностей без перестройки всего программного комплекса.

Межсервисное взаимодействие реализовано в синхронном и асинхронном режимах. REST обеспечивает прямой обмен данными между компонентами, Kafka – обработку фоновых событий без блокировки клиентских запросов. Кэш Redis используется для временного хранения востребованных данных и разгрузки основной базы, что улучшает производительность платформы [17].

Диаграмма компонентов представлена на рисунке 12.

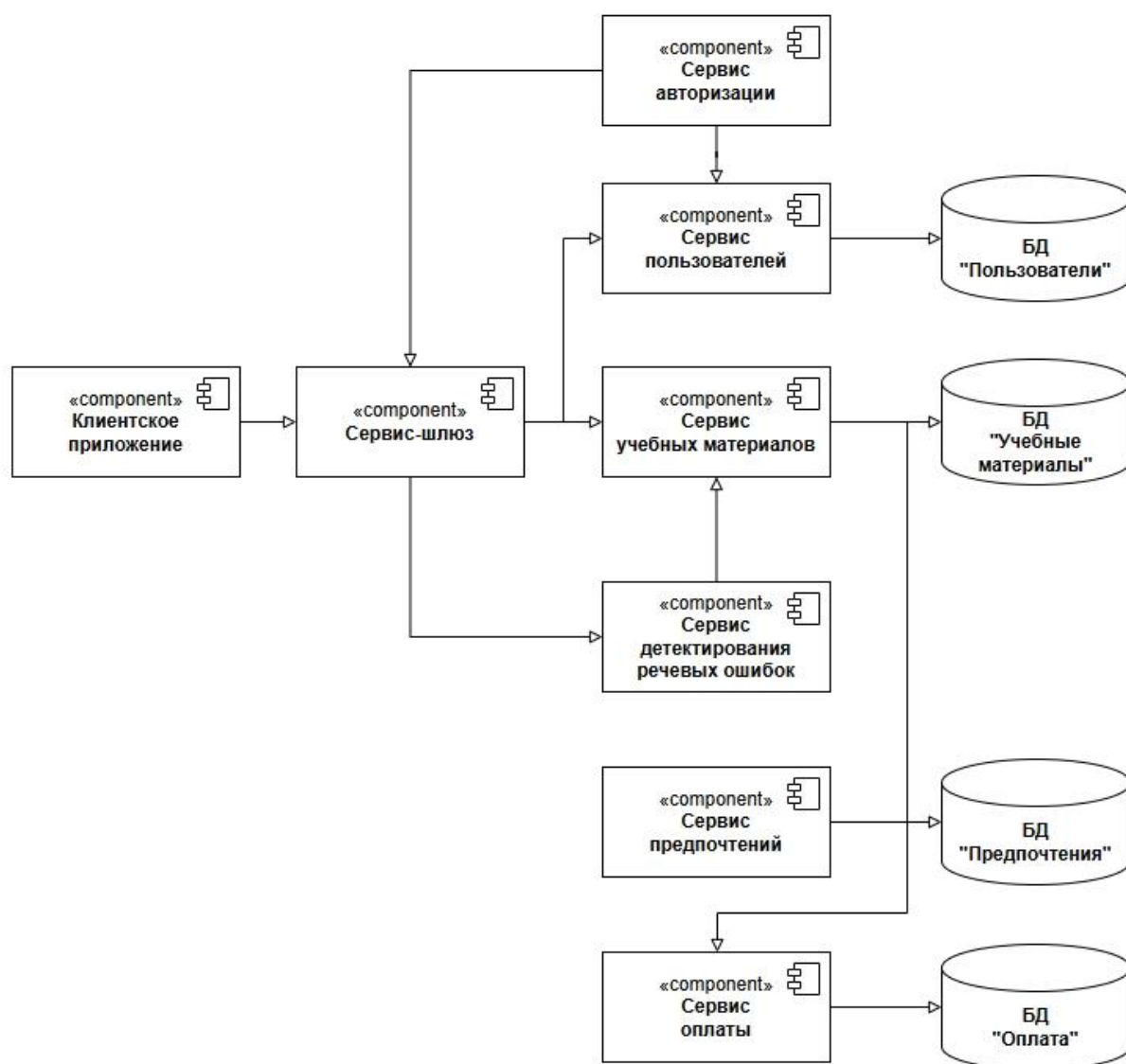


Рисунок 12 – Диаграмма компонентов

1.3 Проектирование и разработка

На концептуальном уровне описываются объекты предметной области и отношения между ними. Такой подход используется для формирования целостного представления о данных, которые должны поддерживаться серверным приложением, для согласования состава сущностей с функциональными требованиями платформы [3, 9].

Для серверной части разрабатываемой образовательной платформы по указанным требованиям построена концептуальная модель в нотации «сущность–связь». Соответствующая диаграмма представлена на рисунке 13.

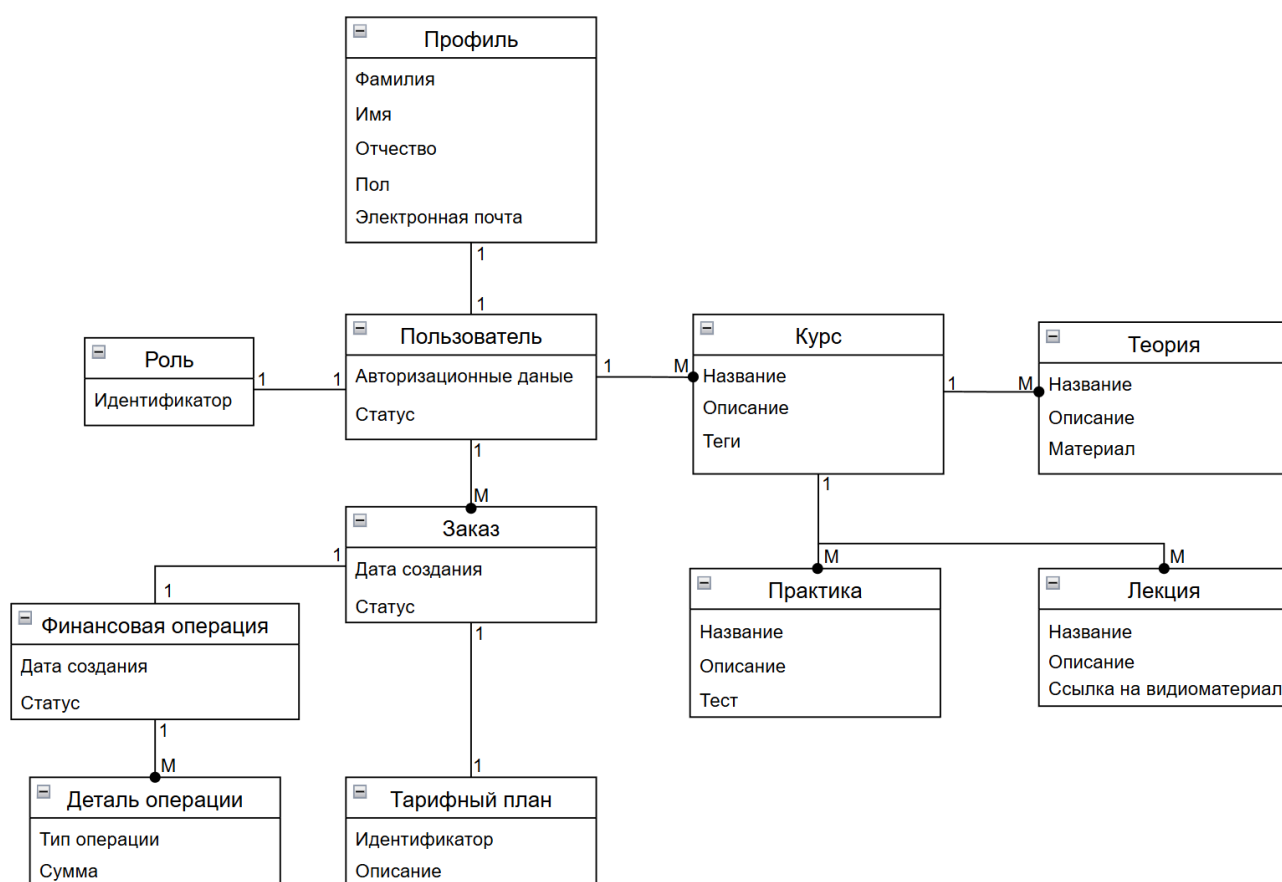


Рисунок 13 – Концептуальная модель БД

На концептуальной ER-диаграмме зафиксированы все объекты предметной области и отношения, обеспечивающие их взаимодействие. Описание сущностей представлено в таблице 5.

Таблица 5 – Описание сущностей

Название	Идентификатор	Описание
Роль	role	Фиксация ролей пользователей и правил доступа в системе
Пользователь	user	Учёт зарегистрированных пользователей платформы
Профиль	profile	Хранение персональных атрибутов пользователя
Курс	course	Описание учебных курсов
Теория	theory	Текстовые теоретические материалы в составе курса
Лекция	lecture	Информация о видео лекциях
Задание	task	Данные заданий для проверки знаний
Заказ	order	Регистрация оформленных на платформе заказов
Содержимое заказа	order_item	Перечень позиций, входящих в состав заказа
Финансовая операция	operation	Обобщённые сведения о платёжных операциях
Деталь операции	operation_detail	Подробные параметры выполненных операций
Тарифный план	tariff_plan	Условия и параметры тарифа, назначенного пользователю

В таблице 6 приведено описание связей между выделенными сущностями.

Таблица 6 – Описание связей между сущностями

Название связи	Описание	Мощность
role-user	Связывает пользователя с назначенной ему ролью	1:1
user-profile	Связывает учётную запись пользователя с его профилем	1:1
user-order	Отражает оформление пользователем одного или нескольких заказов	1:M

Продолжение таблицы 6

Название связи	Описание	Мощность
course-theory	Связывает курс с входящими в него теоретическими материалами	1:M
course-lecture	Связывает курс с входящими в него лекциями	1:M
order-operation	Отражает оформление пользователем одного или нескольких заказов	1:1
operation-operation_detail	Связывает операцию с набором её детализирующих записей	1:M
order-operation	Отражает наличие у заказа одной или нескольких операций	1:M
order-tariff_plan	Связывает заказ с выбранным тарифным планом	1:1
user-course	Отражает участие пользователя в одном или нескольких курсах	1:M

Концептуальная модель базы данных построена в нотации IDEF1X [4] с использованием графического редактора draw.io.

1.3.1 Логическая модель базы данных

На логическом уровне фиксируются структуры таблиц, состав полей, первичные и внешние ключи, а также ограничения целостности, вытекающие из бизнес-правил образовательной платформы. Логическая модель является промежуточным этапом между концептуальным описанием и физической реализацией СУБД [6].

Для серверного приложения предусмотрено разделение данных по доменам с выделением самостоятельных баз: пользователи, профили, учебные курсы, заказы. Подход обеспечивает независимое сопровождение подсистем, разгрузку отдельных хранилищ, точечное масштабирование и независимое резервное копирование.

Диаграмма логической модели базы данных «Пользователи» представлена на рисунке 14.

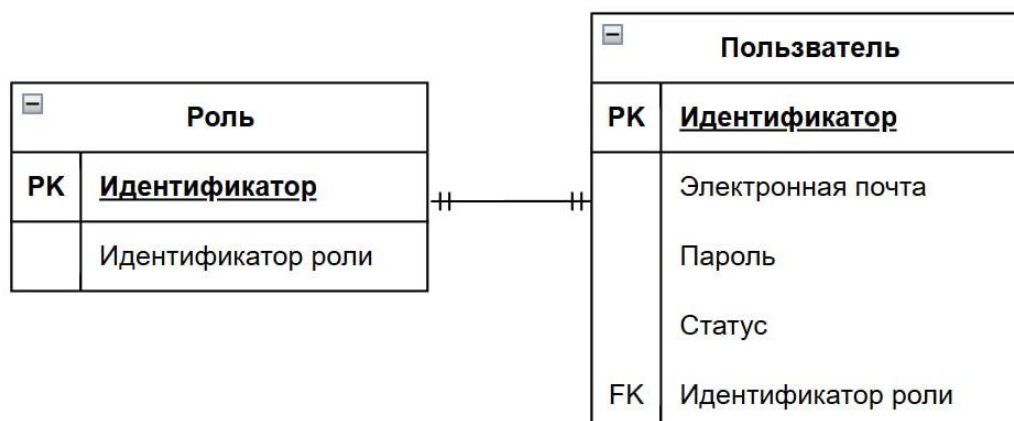


Рисунок 14 – Логическая модель БД «Пользователи»

Диаграмма логической модели базы данных «Профили» представлена на рисунке 15.



Рисунок 15 – Логическая модель БД «Профили»

Диаграмма логической модели базы данных «Курсы» представлена на рисунке 16.

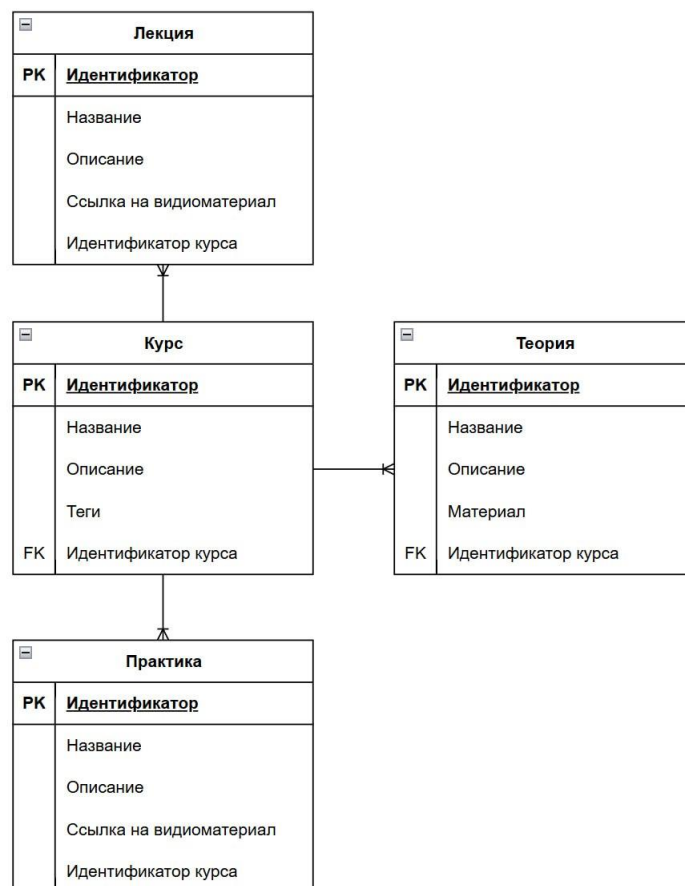


Рисунок 16 – Логическая модель БД «Курсы»

Диаграмма логической модели базы данных «Заказы» представлена на рисунке 17.

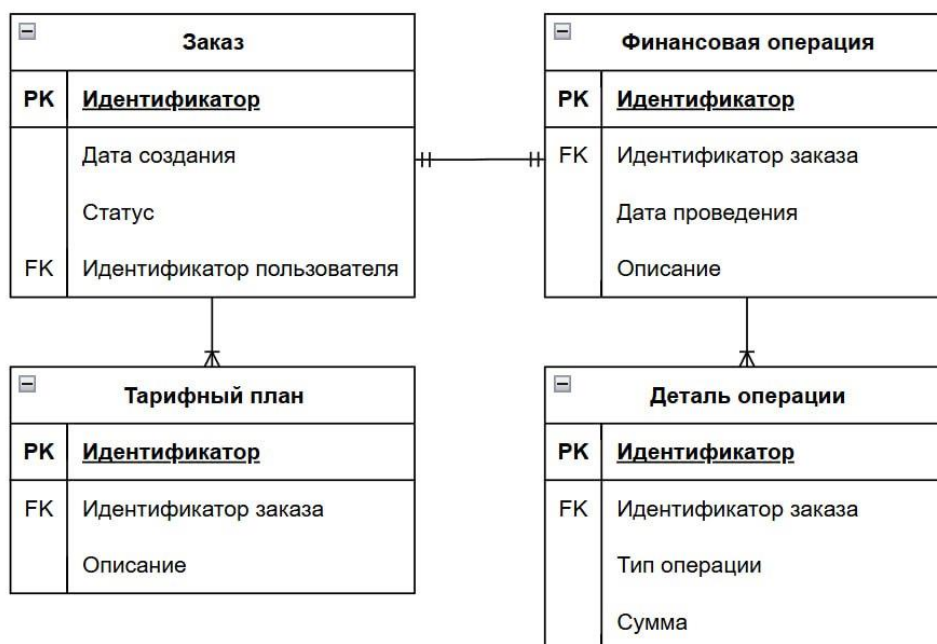


Рисунок 17 – Логическая модель БД «Курсы»

Полученные логические модели служат основой для проектирования физического уровня хранения и развёртывания баз данных в программной инфраструктуре серверного приложения.

1.3.2 Обобщенная структура сервиса

В общем виде разрабатываемая интеллектуальная платформа представляет собой клиент-серверное приложение [5]. Клиентская часть – программный компонент, обеспечивающий взаимодействие пользователя с функциями системы. На практике клиентом, как правило, выступает веб-браузер, установленный на персональном компьютере пользователя. Серверная часть – вычислительная программная среда исполнения, на котором размещается и функционирует разрабатываемое серверное приложение.

К достоинствам клиент–серверной архитектуры относятся:

- централизованное размещение прикладного ПО и данных на сервере, что исключает дублирование программных модулей и сведений на рабочих местах пользователей;
- сниженные требования к аппаратному обеспечению клиента – достаточно веб-браузера;
- повышение информационной безопасности за счёт хранения данных в СУБД и отсутствия прямого доступа пользователей к базе.

К недостаткам указанного подхода можно отнести:

- зависимость доступности сервиса от работоспособности сервера приложений и сервера баз данных: выход из строя любого из узлов приводит к недоступности системы;
- вероятность перегрузки серверной инфраструктуры, в том числе при DDoS-воздействии, что может вызвать падение производительности и увеличение времени отклика [7].

Архитектура приложения представлена на диаграмме развёртывания на рисунке 18.

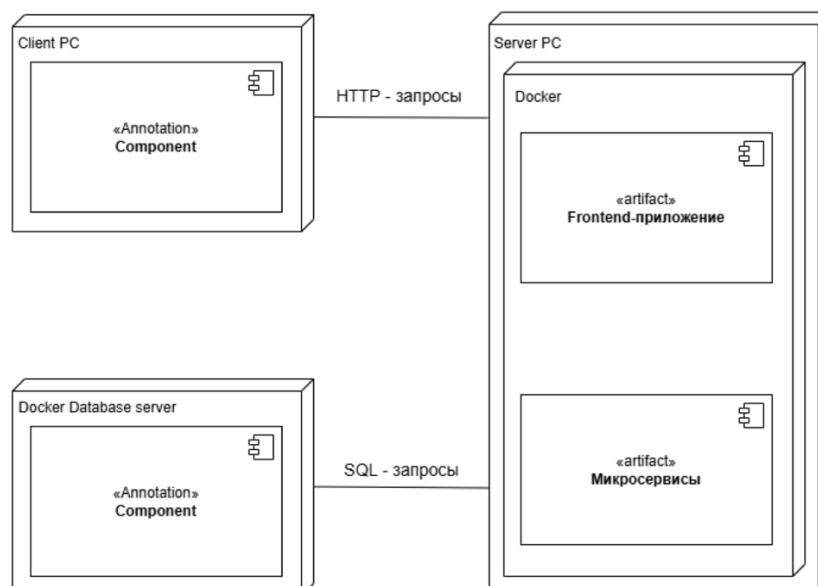


Рисунок 18 – Диаграмма развертывания

Обмен между уровнями организован следующим образом. Пользователь через веб-браузер инициирует обращение к серверу приложений по протоколу HTTP. После чего сервер выполняет прикладную обработку запроса. При обращении к постоянному хранилищу сервер приложений взаимодействует с СУБД посредством SQL-запросов.

Серверная часть и базы данных развернуты в контейнерах Docker [16]. Сборка образов описывается в Dockerfile, а запуск набора контейнеров – сценариями docker-compose. Это обеспечивает воспроизводимое развёртывание на выделенных узлах без ручной настройки окружения на каждом сервере.

1.3.3 Разработка серверной части

Серверное приложение платформы спроектировано как набор слабосвязанных микросервисов – программных модулей, которые можно развёртывать и обновлять независимо. Логика и данные каждого модуля относятся к отдельному домену. Обмен выполняется по сетевым API, а персистентность, как правило, не выносится за пределы сервиса [8].

В составе backend реализовано 7 автономных сервисов с HTTP-интерфейсами. Сервисы оперируют собственными наборами данных, сопровождаются выделенными БД и формализованными контрактами доступа.

Клиентские обращения принимает API Gateway – единый вход в систему. Шлюз маршрутизирует запросы к нужным микросервисам, опираясь на согласованные контракты. Еще одной ролью шлюза является инкапсуляция доменных моделей.

Каждый микросервис спроектирован по многослойной схеме: функциональность разделена на уровни с чётко зафиксированной ответственностью. Подобное разделение облегчает сопровождение и развитие кода, повышает модульность компонентов и способствует повторному применению общих элементов реализации [13]. Обобщённая диаграмма слоёв представлена на рисунке 19.

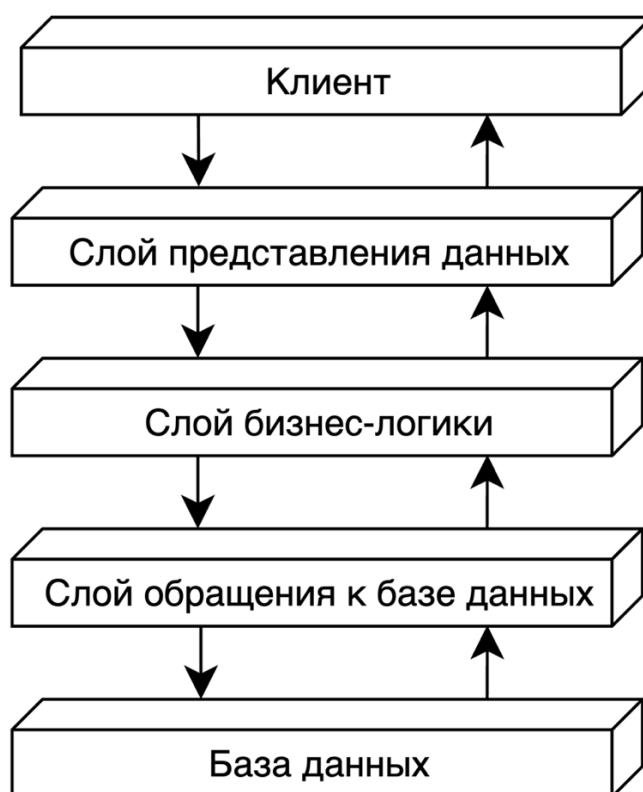


Рисунок 19 – Типовая архитектура микросервисов

Обработка запроса в микросервисе организована по уровням ответственности: контроллеры принимают и проверяют входные данные. Бизнес-правила и сценарии выполняются в слое доменной логики. Сервисный уровень координирует операции и при необходимости инициирует доступ к хранилищу через слой доступа к базе данных. Взаимодействие с PostgreSQL описывается моделями, согласованными со схемой таблиц. Вычислительно сложные агрегации могут дополняться функциями и триггерами СУБД.

Синхронный контур клиент–сервер реализован в стиле REST [15]. Фоновые и неблокирующие процедуры вынесены в Kafka [12]. Для ускорения повторных обращений к образовательной платформе применяется кэш Redis, разгружающий PostgreSQL и сокращающий время отклика.

1.3.4 Разработка физической модели базы данных

Физическая модель данных – завершающий этап проектирования БД, формируемый на основе логической модели. На физическом уровне логическая схема преобразуется в структуры, пригодные для развёртывания в СУБД. Определяются таблицы и атрибуты, ключи, ограничения целостности, а также эксплуатационные параметры – индексы, размещение данных, приёмы повышения быстродействия и денормализации [14]. Построение выполняется под выбранную для серверной части платформы СУБД [11].

Физическая модель базы данных «Пользователи» представлена на рисунке 19.

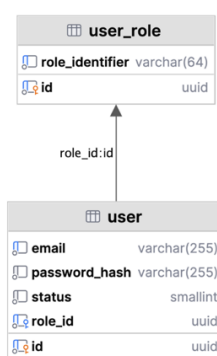


Рисунок 19 – Физическая модель БД «Пользователи»

Физическая модель базы данных «Профили» представлена на рисунке 20.

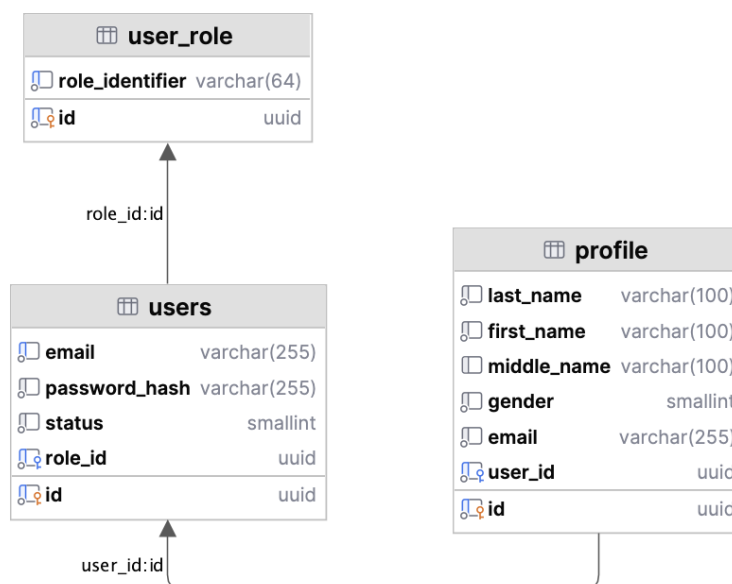


Рисунок 20 – Физическая модель БД «Профили»

Физическая модель базы данных «Курсы» представлена на рисунке 21.

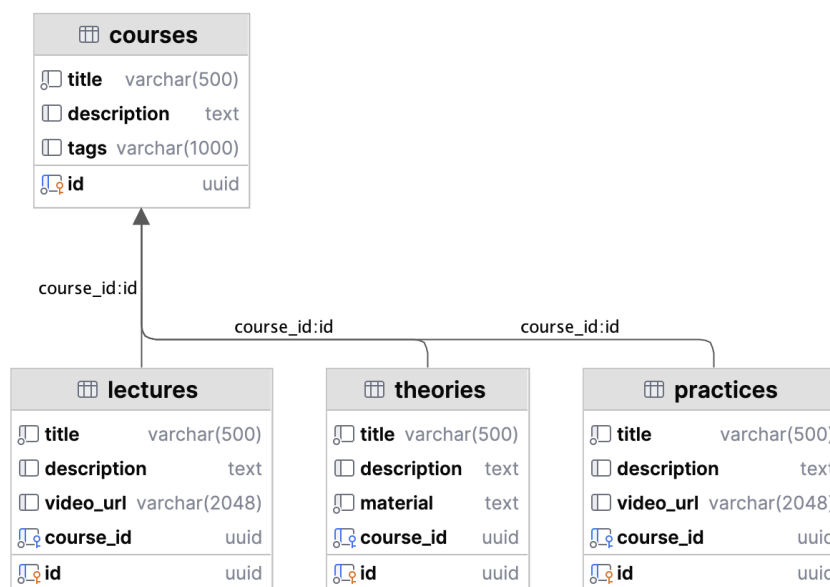


Рисунок 21 – Физическая модель БД «Курсы»

На рисунке 22 представлена физическая модель базы данных «Заказы».

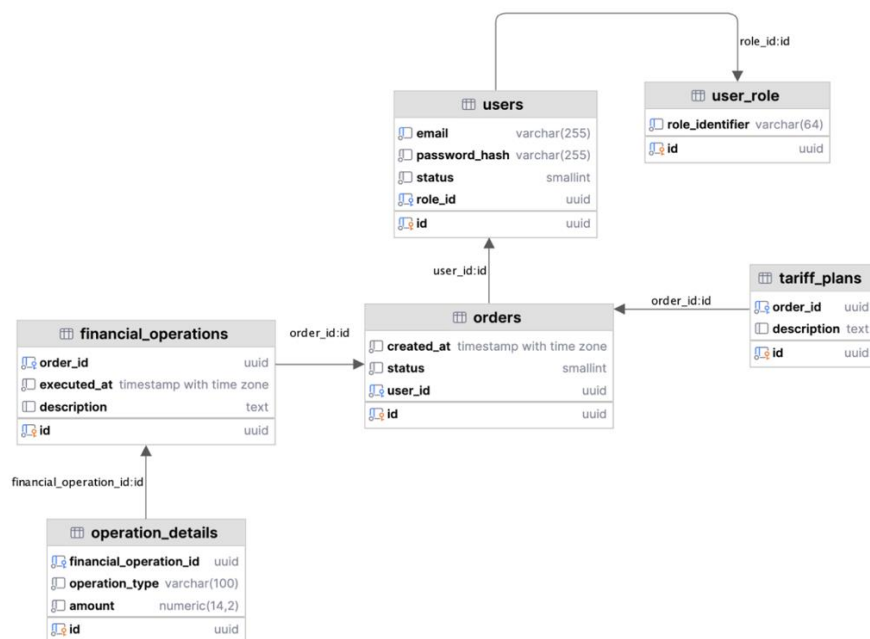


Рисунок 22 – Физическая модель БД «Заказы»

В таблице 7 приведено описание всех баз данных интеллектуальной образовательной платформы.

Таблица 7 – Описание сущностей баз данных

Наименование	Описание	Категория
user_role	Справочник ролей: код роли для назначения прав доступа	Кодовая
users	Учётные записи пользователей	Стержневая
profile	Персональные данные пользователя	Характеристическая
courses	Учебные курсы: название, описание, теги	Стержневая
lectures	Лекции курса	Характеристическая
theories	Теоретические материалы курса	Характеристическая
practices	Практические занятия курса	Характеристическая
orders	Заказы пользователя	Стержневая
financial_operations	Финансовая операция по заказу	Стержневая
tariff_plans	Тарифные условия в разрезе заказа	Характеристическая
operation_details	Строки детализации финансовой операции	Характеристическая

1.3.5 Разработка архитектурных слоев внутри микросервисов

Ключевое требование микросервисного подхода – изоляция обязанностей модулей. Слойная организация кода способствует наращиванию нагрузки, модульному тестированию и сопровождению без перестройки всего приложения. Структуру и связи уровней целесообразно иллюстрировать на course-service. Он демонстрирует принятые в системе паттерны, общие для остальных сервисов. Микросервисной архитектуре важно явно разграничивать ответственность компонентов. Для обеспечения масштабируемости сервиса, удобства в тестировании и сопровождении, каждый микросервис строится слоями.

1.3.5.1 Разработка слоя представления данных в микросервисах

На уровне представления описываются внешние REST-контракты микросервиса – перечень доступных клиенту возможностей. Реализация оформляется обработчиками, которые при запуске приложения связываются с маршрутизатором, принимают HTTP-запросы GET, POST, PUT, DELETE, проверяют корректность запроса и передают управление бизнес-логике. Настройка роутера показана в листинге 1.

Листинг 1 – Инициализация роутера и подключение маршрутов

```
teacherGroup := apiGroup.Group("/teacher")
teacherGroup.Use(middleware.ValidateTeacherRole())
{
    teacherGroup.POST("/courses/create",
handler.CreateCourse(logger, courseService))
    teacherGroup.DELETE("/courses/delete/:courseId",
handler.DeleteCourse(logger))
    teacherGroup.POST("/courses/update/:courseId",
handler.UpdateCourse(logger))
    teacherGroup.POST("/courses/lecture/create",
handler.CreateLecture(logger, courseService))
    teacherGroup.DELETE("/courses/lecture/delete/:lectureId",
handler.DeleteLecture(logger))
    teacherGroup.POST("/courses/lecture/update/:lectureId",
handler.UpdateLecture(logger))
    teacherGroup.POST("/courses/theory/create",
handler.CreateTheory(logger, courseService))
    teacherGroup.DELETE("/courses/theory/delete/:theoryId",
handler.DeleteTheory(logger))
}
```

					ПГУ 1.09.03.01.50.001 ПЗ	Лист
						3
Изм.	Лист	№ докум.	Подпись	Дата		8


```

    teacherGroup.POST("/courses/theory/update/:theoryId",
handler.UpdateTheory(logger))
}
courseGroup := apiGroup.Group("/courses")
courseGroup.Use(middleware.ValidateAuth())
{
    courseGroup.GET("/get/:profileId", handler.GetCourse(logger,
courseService))
    courseGroup.POST("/update/:courseId",
handler.UpdateCourse(logger))
    courseGroup.POST("/push", handler.PushCourse(logger,
courseService))
}

```

На листинге 2 показана реализация обработчика входящих запросов.

Листинг 2 – Реализации функции обработки запроса

```

func GetCourse(logger plogger.Logger, courseService
*service.CourseService) gin.HandlerFunc {
    return func(c *gin.Context) {
        var req getCourseUri
        if err := c.ShouldBindUri(&req); err != nil {
            phhttp.ErrorResponse(c, http.StatusBadRequest, "bad body")
            return
        }
        course, err := courseService.Get(c, uuid.MustParse(req.ProfileId))
        switch {
        case err == nil:
        case errors.Is(err, service.ErrNotFound):
            logger.Error(err.Error())
            phhttp.ErrorResponse(c, http.StatusNotFound,
service.ErrNotFound.Error())
            return
        default:
            logger.Error(err.Error())
            phhttp.ErrorResponse(c, http.StatusInternalServerError,
"internal server error")
            return
        }

        response := newGetCourseResponse(course)
        phhttp.OkResponseWithResult(c, response)
    }
}

```

Уровень API выполняет роль интерфейса взаимодействия с клиентом. Обработывает HTTP-обращения, валидирует параметры запроса и делегирует выполнение бизнес-правилам. Подобное разграничение функций повышает

модульность сервиса и облегчает его развитие и наращивание производительности.

1.3.5.2 Разработка слоя бизнес-логики в микросервисах

На уровне бизнес-логики реализуются предметные правила и прикладные сценарии микросервиса независимо от транспортного протокола и способа доступа к БД. Реализация оформляется классами/модулями Service. Имена методов отражают вид операции: Get, Add, Update, Delete. Фрагмент инициализации сервиса и перечень процедур показаны в листинге 3.

Листинг 3 – Инициализация сервиса и описание функций

```
type CourseService interface {
    Push(ctx context.Context, profileId uuid.UUID, mistake string)
    (bool, error)
    Get(ctx context.Context, profileId uuid.UUID) (entity.Course,
    error)
    Create(ctx context.Context, course entity.Course) error
    CreateLecture(ctx context.Context, courseId uuid.UUID, lecture
    entity.Lecture) error
    CreateTheory(ctx context.Context, courseId uuid.UUID, theory
    entity.Theory) error
    DeleteCourseFromProfileById(ctx context.Context, profileId
    uuid.UUID, courseId uuid.UUID) error
}

type Impl struct {
    courseRepo courseRepo
}

func NewCourseService(cRepo courseRepo) Course {
    return &impl{
        courseRepo: cRepo,
    }
}
```

Бизнес-правила оперируют доменными сущностями, а доступ к персистентности выполняется через репозитории. При сохранении доменные объекты преобразуются в форматы таблиц с помощью мапперов. Для межсервисных вызовов здесь же могут использоваться HTTP и Kafka-клиенты. Иллюстрация метода бизнес-логики – в листинге 4.

					ПГУ 1.09.03.01.50.001 ПЗ	Лист
						4
Изм.	Лист	№ докум.	Подпись	Дата		0

Листинг 4 – Пример функции слоя бизнес-логики

```
func (cs *Impl) Create(ctx context.Context, course entity.Course)
error {

    isCourseExists, err := cs.isCourseExists(ctx, course.Id)
    if err != nil {
        return err
    }

    if isCourseExists {
        err = ErrAlreadyExists
        return err
    }
err = cs.courseRepo.CreateCourse(ctx, course)
    switch {
    case err == nil:
    default:
        return err
    }

    return nil
}
```

Листинг демонстрирует сценарий добавления учебного курса. Сначала через слой взаимодействия с базой данных выполняется проверка существования записи - isCourseExists. Если дубликат не обнаружен и ошибок нет, инициируется CreateCourse с передачей модели course без дополнительного доменного маппинга. Диаграмма классов слоя бизнес-логики сервиса работы с курсами представлена на рисунке 23.

На уровне бизнес-логики сосредоточены правила обработки информации и интеграционные обращения к иным модулям. Это повышает сопровождаемость и облегчает наращивание производительности сервиса.

Слой бизнес-логики обеспечивает как реализацию предметных операций, так и координацию с остальными частями системы.

1.3.5.3 Разработка слоя обращения к базе данных в микросервисах

Слой взаимодействия с базой данных обеспечивает подключение к базе и выполнение CRUD-операций через SQL-запросы. Компоненты этого слоя реализуются в виде репозитория, обычно с суффиксом Repository и отвечают за чтение, запись, обновление и удаление данных.

					ПГУ 1.09.03.01.50.001 ПЗ	Лист
						4
Изм.	Лист	№ докум.	Подпись	Дата		1

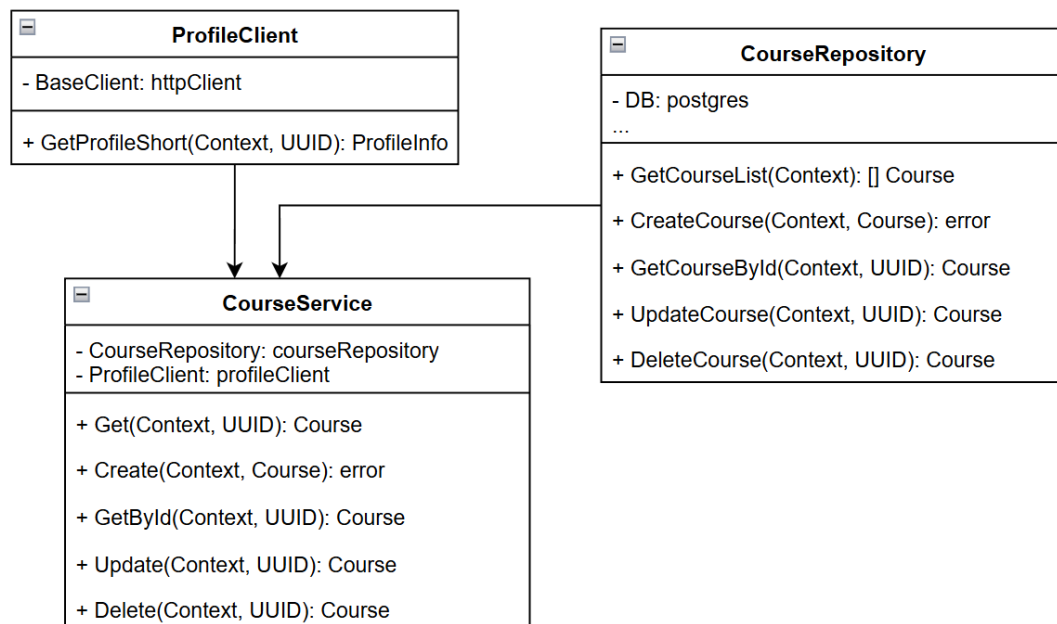


Рисунок 23 – Диаграмма классов сервиса

Сложная бизнес-логика в репозиториях не размещается: их задача – изолировать работу с БД и предоставить сервисному слою понятный интерфейс доступа к данным. Пример инициализации репозитория и описание функций представлен на листинге 5.

Пример контракта репозитория представлен на листинге 5.

Листинг 5 – Инициализация репозитория и описание функций

```

type courseRepo interface {
    CreateCourse(ctx context.Context, course entity.Course) error
    GetCourseById(ctx context.Context, courseId uuid.UUID)
(entity.Course, error)
    CreateLecture(ctx context.Context, lecture entity.Lecture) error
    GetLectureByTag(ctx context.Context, tag string) (entity.Lecture,
error)
    CreateTheory(ctx context.Context, theory entity.Theory) error
    GetTheoryByTag(ctx context.Context, tag string) (entity.Theory,
error)
    GetPracticeByTag(ctx context.Context, tag string)
(entity.Practice, error)
    GetPracticeTasks(ctx context.Context, practiceId uuid.UUID)
([]entity.Task, error)
    AddTag(ctx context.Context, courseId uuid.UUID, tags string) error
    GetCoursesTags(ctx context.Context, profileId uuid.UUID)
(entity.PushTags, error)
    GetCourseIdByTag(ctx context.Context, tag string) (courseId
uuid.UUID, err error)
}
  
```

```

    SetCourse(ctx context.Context, profileId uuid.UUID, courseId
uuid.UUID) error
    GetCourseTagsByProfileId(ctx context.Context, profileId uuid.UUID,
mistake string) (bool, error)
    DeleteCourseFromProfileById(ctx context.Context, profileId
uuid.UUID) error
    ClearMistakeFromProfile(ctx context.Context, profileId uuid.UUID,
mistake string) error
    GetCourseTag(ctx context.Context, courseId uuid.UUID) (string,
error)
}

```

Пример функции слоя работы с базой данных представлен на листинге 6.

Листинг 6 – Пример функции слоя работы с базой данных

```

func (cr *CourseRepository) CreateCourse(ctx context.Context,
course entity.Course) error {
    args := []interface{}{
        course.Id,
        course.Name,
        course.Description,
    }

    _, err := cr.db.ExecContext(ctx, query.CreateCourse, args...)
    if err != nil {
        return err
    }

    return nil
}

```

В приведенном примере рассматривается функция создания курса и записи ее в базу данных. Для взаимодействия с базой данных используется функция `ExecContext`, в которую передается SQL-скрипт `CreateCourse`. При выполнении может возникать ошибка, которая обрабатывается ниже.

Рассмотрим взаимодействие всех компонентов и слоев между собой на примере создания курса. Процесс создания курса начинается с того, что клиент отправляет запрос на создание курса. Сценарий добавления курса запускается клиентским обращением к API. На уровне представления `CreateCourseHandler` проверяет запрос и делегирует выполнение `CourseService`. Сервисный уровень инициирует запись через `CourseRepository`, который выполняет `INSERT` в таблицу курсов и передаёт обратно идентификатор новой записи. После этого `CourseService` инициирует обращение к внешнему микросервису через

ClientServiceClient для получения связанных пользовательских данных, необходимых для дальнейшей обработки результата. ClientServiceClient отправляет HTTP-запрос в целевой сервис и возвращает полученные данные обратно в CourseService. Когда все шаги завершены, CourseService формирует итоговый результат и передаёт его обратно в CreateCourseHandler. Далее слой представления отправляет ответ клиенту, и в клиентском приложении отображается результат создания курса.

1.4 Тестирование серверной части образовательной платформы

1.4.1 Функциональное тестирование добавления курсов и их компонентов

После завершения разработки образовательной платформы было проведено функциональное тестирование модуля взаимодействия с курсами.

Данный этап является критически важным для подтверждения качества системы, так как позволяет проверить соответствие реализованного функционала заданным требованиям и спецификациям. В процессе тестирования контролировались корректность обработки входных данных, устойчивость API к ошибочным сценариям и корректность бизнес-логики при работе с курсами, лекциями и теоретическими материалами.

Для проведения функционального тестирования был выбран модуль Courses, включая связанные операции по Course, Lecture, Theory и очистке курса у профиля.

Функционально модуль обеспечивает:

- создание курсов;
- создание лекций и теорий с привязкой к курсу;
- назначение курса пользователю по тегу ошибки;
- получение курса пользователя;
- удаление привязки курса к профилю.

Для данного модуля были выделены следующие требования:

					ПГУ 1.09.03.01.50.001 ПЗ	Лист
						4
Изм.	Лист	№ докум.	Подпись	Дата		4

- валидация входа: ошибка при неверной структуре JSON или некорректном UUID;
- уникальность: ошибка при дублировании сущности;
- целостность связей: ошибка, если зависимая сущность не найдена;
- авторизация: ошибка при отсутствии токена или недостаточных правах;
- штатные операции: успешное создание, чтение и удаление при валидных данных.

Также были разработаны тест-кейсы, они представлены в таблице 8.

Таблица 8 – Тест-кейсы

Эндпоинт	Сценарий	Ожидаемый результат	Результат
POST /api/teacher/courses/create	Создать курс с валидным телом	200 ОК, курс создан в БД	Пройден
POST /api/teacher/courses/create	Повторно создать курс с тем же id	409, сообщение «курс уже существует»	Пройден
POST /api/teacher/courses/create	Невалидный JSON	400, сообщение «bad body»	Пройден
POST /api/teacher/courses/lecture/create	Создать лекцию для существующего курса	200 ОК, лекция и связь с курсом созданы	Пройден
POST /api/teacher/courses/lecture/create	course_id не UUID	400, сообщение «invalid course_id»	Пройден
POST /api/teacher/courses/lecture/create	Курс не существует	404, сообщение «курс не найден»	Пройден
POST /api/teacher/courses/lecture/create	Повтор лекции (дубликат id)	409, сообщение «лекция уже существует»	Пройден
POST /api/teacher/courses/theory/create	Создать теорию для существующего курса	200 ОК, теория и связь с курсом созданы	Пройден
POST /api/teacher/courses/theory/create	id теории не UUID	400, сообщение «invalid id»	Пройден
POST /api/teacher/courses/theory/create	Курс не существует	404, сообщение «курс не найден»	Пройден

Продолжение таблицы 8

Эндпоинт	Сценарий	Ожидаемый результат	Результат
Все endpoints	Проверка доступа без токена/неверная роль	403, сообщение «доступ запрещен»	Пройден

Все этапы модульного тестирования прошли успешно. Программа корректно обрабатывает во всех возможных тестируемых ситуациях.

1.4.2 Модульное тестирование добавления курсов и их компонентов

Для подтверждения корректности функционирования разработанного программного средства выполнено модульное тестирование компонентов, реализующих операции создания и редактирования курсов, а также создание связанных сущностей: лекций, теоретических материалов и практических заданий.

Испытания проводились для программных модулей пакета handler, реализованных на языке Go.

При проведении тестирования использованы стандартные средства языка Go: пакет testing, а также httptest для имитации HTTP-запросов и проверки ответов обработчиков. Исходные тексты тестов приведены в приложении Б. Результаты испытаний представлены в таблице 9.

Таблица 9 – Исходные тесты.

Входные данные	Ожидаемый результат	Фактический результат	Статус
Запрос на создание курса с корректными значениями полей name, description	Успешное создание курса, возврат идентификатора созданной записи	Курс создан, идентификатор возвращен	Пройден
Запрос на редактирование курса с некорректным значением courseId	Формирование ошибки валидации идентификатора	Получена ошибка валидации	Пройден

Продолжение таблицы 9

Входные данные	Ожидаемый результат	Фактический результат	Статус
Запрос на редактирование курса с некорректным значением courseId	Формирование ошибки валидации идентификатора	Получена ошибка валидации	Пройден
Запрос на создание лекции с корректными данными и существующим курсом	Успешное создание лекции и ее привязка к курсу	Лекция создана и привязана к курсу	Пройден
Запрос на создание курса с именем, уже присутствующим в системе	Формирование ошибки конфликта	Получена ошибка конфликта	Пройден
Запрос на редактирование курса с корректным идентификатором courseId и корректным телом запроса	Успешное обновление параметров курса	Параметры курса обновлены	Пройден
Запрос на редактирование курса, отсутствующего в базе данных	Формирование ошибки «объект не найден»	Получена ошибка «объект не найден»	Пройден
Запрос на создание практического материала при моделировании внутренней ошибки сервиса	Формирование ошибки внутреннего уровня	Получена ошибка внутреннего уровня	Пройден

По результатам модульного тестирования установлено, что реализованные программные модули корректно функционируют в штатных и граничных режимах работы, а фактические результаты соответствуют ожидаемым. Таким образом, проверенные компоненты удовлетворяют предъявляемым функциональным требованиям. В листинге 7 представлены модульные тесты микросервиса работы с курсами.

Листинг 7 – Модульные тесты микросервиса работы с курсами

```
type courseServiceMock struct {
```



```

pushFn                                func(ctx context.Context, profileId
uuid.UUID, mistake string) (bool, error)
getFn                                func(ctx context.Context, profileId
uuid.UUID) (entity.Course, error)
createFn                              func(ctx context.Context, course
entity.Course) (uuid.UUID, error)
updateFn                              func(ctx context.Context, courseId
uuid.UUID, course entity.Course) error
deleteFn                              func(ctx context.Context, courseId
uuid.UUID) error
listFn                                func(ctx context.Context)
([]entity.Course, error)
createLectureFn                       func(ctx context.Context, courseId
uuid.UUID, lecture entity.Lecture) error
createTheoryFn                       func(ctx context.Context, courseId
uuid.UUID, theory entity.Theory) error
createPracticeFn                     func(ctx context.Context, courseId
uuid.UUID, practice entity.Practice) error
deleteCourseFromProfileFn            func(ctx context.Context, profileId
uuid.UUID, courseId uuid.UUID) error

```

					ПГУ 1.09.03.01.50.001 ПЗ	Лист
						4
Изм.	Лист	№ докум.	Подпись	Дата		8

2 Охрана труда

В данном разделе рассматриваются требования по обеспечению безопасности жизнедеятельности и охраны труда пользователей ПЭВМ, участвующих в разработке и эксплуатации программного обеспечения, а также способы минимизации вредных факторов на здоровье.

2.1 Характеристика потенциальных рисков при работе с ПК

В соответствии с ГОСТ 12.0.003-2015 «Система стандартов безопасности труда. Электронно-вычислительная техника. Общие требования безопасности», при эксплуатации ПК выделяются следующие основные риски:

- риски поражения электрическим током из-за: неисправной проводки, отсутствия заземления, попадания жидкости на технику, короткого замыкания при перегрузке сети;
- риски заболеваний опорно-двигательной системы из-за: неправильного положения тела; неправильной высоты стула и стола;
- риски заболеваний зрения из-за: использования ПК без перерывов для глаз; неправильной яркости экрана и расположения монитора.

2.2 Оборудование рабочего места

Основные требования, которые должны соблюдаться при пользовании рабочем местом:

- расстояние от монитора до глаз должно быть не менее 500 – 700 мм. Монитор должен располагаться ниже уровня глаз под углом 15-20°;
- высота стола должна быть 680 – 800 мм (в зависимости от роста пользователя);
- кресло должно иметь регулируемую высоту сиденья и спинку, а также глубину сиденья 400 - 450 мм.

На рисунке 24 показана правильная организация рабочего пространства.

					ПГУ 1.09.03.01.50.001 ПЗ	Лист
						4
Изм.	Лист	№ докум.	Подпись	Дата		9

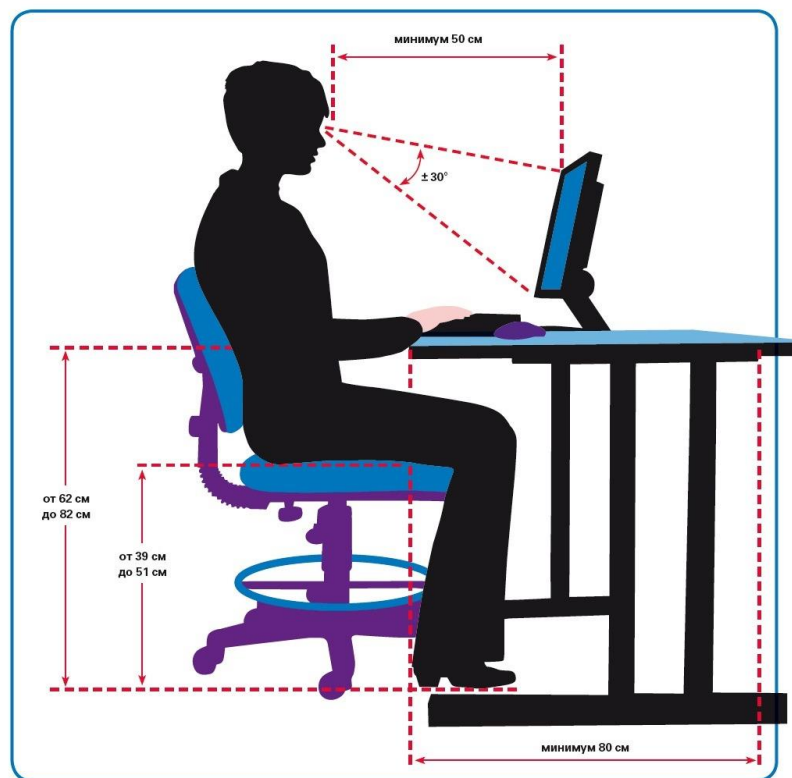


Рисунок 24 – Правильная организация рабочего места

Рабочее место с монитором должно гарантировать пользователю удобную работу с ПК в положении сидя. При нахождении за ПК должны выполняться следующие условия: прямая спина с опорой на спинку кресла; расслабленные плечи; кисти и запястья в прямом положении; ноги должны устойчиво стоять на полу.

2.3 Освещённость рабочего места

Для обеспечения здоровья органов зрения на рабочем месте должны соблюдаться стандарты освещения. Согласно санитарным правилам, общая освещённость рабочего места должна составлять не менее 400 люкс. Источники освещения должны обеспечивать равномерный рассеянный свет без бликов на мониторе. Цветовая температура источников освещения должна составлять 4000 – 5000K, что соответствует белому цвету. Недопустимо применение мерцающих источников света и расположение в поле зрения пользователя ярких источников света.

2.4 Электробезопасность

Электробезопасность при работе с ПК регламентируется ГОСТ 12.1.019-2017 «Электробезопасность». Исходя из стандарта, всё электрооборудование должно иметь защитное заземление. ГОСТ предусматривает использование только исправного оборудования с целой изоляцией и компонентами.

Кабели должны быть проложены в местах, где они не будут подвергаться механическим воздействиям и повреждениям. Стандарт требует, чтобы все электрические компоненты были защищены от случайного прикосновения, а корпуса электрооборудования были заземлены.

Пользователи оборудования должны быть обучены правилам электробезопасности и оказании первой помощи при поражении электрическим током.

2.5 Пожарная безопасность

Основные риски пожарной безопасности связаны с возможным перегревом оборудования, короткими замыканиями и возгоранием компонентов. Для предотвращения опасных ситуаций необходимо поддерживать исправное состояние электрооборудования с средствами защиты. Особое внимание необходимо обратить на правильную организацию рабочего пространства - компьютеры должны располагаться на расстоянии не менее 0.5 метра от отопительных приборов и горючих материалов. В помещениях с компьютерами необходимо наличие средств пожаротушения.

Для повышения противопожарной безопасности необходимо соблюдать следующие правила: проходы и запасные выходы не должны быть перекрыты посторонними предметами; в здании должна быть установлена автоматическая пожарная сигнализация и планы эвакуации;

					ПГУ 1.09.03.01.50.001 ПЗ	Лист
						5
Изм.	Лист	№ докум.	Подпись	Дата		1

2.6 Санитарно-гигиеническая характеристика производства

Санитарно-гигиеническая характеристика производства регламентируется в соответствии с СанПиН 1.2.3685-21 [18] и другими нормативными документами.

Основными критериями являются: влажность и температура воздуха в помещении, количество света, скорость движения воздуха.

Рабочие столы должны располагаться не более, чем в 6 метрах от оконных проемов. Окна и рабочие места должны регулярно очищаться от пыли и загрязнений. Температурный режим в помещении должен поддерживаться на уровне 22 – 25 °С при влажности воздуха около 60 %. Необходимо наличие вентиляции или системы кондиционирования. Уровень шума не должен превышать 65 дБ.

В помещении необходимо обеспечить: доступ к питьевой воде; доступ к огнетушителям (1 единицу на 50 м³ помещения); аптечки первой помощи.

Все сотрудники должны проходить инструктажи по технике безопасности не реже 1 раза в год.

					ПГУ 1.09.03.01.50.001 ПЗ	Лист
						5
Изм.	Лист	№ докум.	Подпись	Дата		2

3 Стартап часть

3.1 Проблематика и рыночный потенциал

Эффективная коммуникация является одним из ключевых факторов личного и профессионального успеха. Однако для значительной части людей публичные выступления и деловое общение остаются серьёзным барьером. По данным Университета ИТМО, до 95% людей испытывают дискомфорт или страх перед аудиторией, что делает глоссофобию одной из самых распространённых фобий [18].

Одной из причин этого страха является неуверенность в качестве собственной речи: плохая дикция, неправильные ударения, слова-паразиты, монотонность и недостаточная чёткость артикуляции. Исследования показывают, что даже небольшие речевые дефекты снижают понятность речи и могут влиять на восприятие говорящего как менее уверенного и компетентного.

Проблема особенно актуальна в условиях современного рынка труда, где всё большее значение приобретают soft skills. По данным Гарвардского и Стэнфордского университетов, а также Фонда Карнеги, гибкие навыки могут определять до 85% профессионального успеха [19]. Центральное место среди них занимает коммуникация: умение вести переговоры, выступать публично, презентовать идеи и уверенно взаимодействовать с коллегами, клиентами и руководством.

При этом системное обучение риторике, технике речи и публичным выступлениям редко входит в стандартные образовательные программы. Для большинства желающих доступны только специализированные курсы, стоимость которых начинается от 15 000 рублей и может достигать 200 000–300 000 рублей. Это делает качественное развитие речевых навыков недоступным для широкой аудитории, особенно для школьников, студентов и молодых специалистов.

Таким образом, существует высокий спрос на доступные,

					ПГУ 1.09.03.01.50.001 ПЗ	Лист
						5
Изм.	Лист	№ докум.	Подпись	Дата		3

персонализированные и эффективные инструменты для развития речевых компетенций.

Целевой рынок проекта включает два основных сегмента: B2C и B2B.

B2C-сегмент составляют индивидуальные пользователи, которым важно улучшить дикцию, повысить уверенность и развить навыки публичной и деловой коммуникации. К ним относятся студенты, молодые специалисты, фрилансеры, преподаватели, соискатели работы, абитуриенты творческих и гуманитарных направлений. Для них особенно важны доступная стоимость, мобильность, возможность заниматься в удобное время и получать персонализированную обратную связь.

B2B-сегмент включает компании и организации, заинтересованные в развитии коммуникативных навыков сотрудников. Потенциальными клиентами являются бизнес-компании, контакт-центры, телекоммуникационные организации, образовательные платформы и корпоративные университеты. Для них продукт ценен как инструмент повышения качества клиентского сервиса, эффективности переговоров, презентаций и внутренней коммуникации.

Основными географическими рынками являются Россия и Беларусь, а также русскоязычные пользователи и компании из стран ближнего зарубежья, где владение качественной русской речью является важным конкурентным преимуществом.

Рыночный потенциал проекта подтверждается ростом онлайн-образования. По данным Grand View Research, мировой рынок электронного обучения в 2024 году оценивался в 373,47 млрд долларов США и может достичь 1,07 трлн долларов к 2032 году при CAGR 13,4%. Особенно быстро развиваются мобильное обучение, корпоративное образование и обучение soft skills.

В России рынок электронного обучения также демонстрирует устойчивый рост: в 2024 году его объём оценивается в 85–90 млрд рублей, а сегмент B2C-обучения взрослых – примерно в 30 – 35 млрд рублей. Это подтверждает наличие значительного спроса на цифровые образовательные решения, в том числе в области развития речевых и коммуникативных навыков.

					ПГУ 1.09.03.01.50.001 ПЗ	Лист
						5
Изм.	Лист	№ докум.	Подпись	Дата		4

3.2 Целевая аудитория и каналы взаимодействия

Целевая аудитория проекта представлена двумя основными сегментами: индивидуальными пользователями (B2C) и корпоративными клиентами (B2B).

B2C-сегмент включает физических лиц, заинтересованных в развитии речевых и коммуникативных навыков, повышении уверенности при публичных выступлениях, деловом и повседневном общении. К данной группе относятся студенты и школьники старших классов, молодые специалисты, фрилансеры, соискатели работы, преподаватели, лекторы, тренеры, а также абитуриенты направлений, где высокий уровень устной речи является значимым критерием отбора. Для указанной аудитории ключевыми факторами являются доступная стоимость обучения, возможность самостоятельной работы, гибкий график занятий, мобильность и получение персонализированной обратной связи.

B2B-сегмент представлен организациями, заинтересованными в повышении качества коммуникации своих сотрудников. К нему относятся компании малого, среднего и крупного бизнеса, телекоммуникационные организации, контакт-центры, образовательные платформы и корпоративные университеты. Для данных клиентов развитие речевых компетенций сотрудников связано с повышением эффективности переговоров, улучшением качества клиентского сервиса, развитием презентационных навыков, укреплением профессионального имиджа компании и снижением затрат на традиционные формы корпоративного обучения.

Таким образом, целевая аудитория проекта охватывает как индивидуальных пользователей, стремящихся повысить личную и профессиональную уверенность через развитие речи, так и корпоративных клиентов, рассматривающих речевые и коммуникативные навыки как инструмент повышения эффективности бизнеса. Основными географическими рынками проекта являются Российская Федерация и Республика Беларусь, а также русскоязычные пользователи и организации стран ближнего зарубежья.

					ПГУ 1.09.03.01.50.001 ПЗ	Лист
						5
Изм.	Лист	№ докум.	Подпись	Дата		5

3.3 План коммерциализации и экономика проекта

Основной целью коммерциализации образовательной платформы «ПРОПС» является формирование устойчивой бизнес-модели, обеспечивающей финансовую стабильность проекта, его дальнейшее развитие и масштабирование. Проект ориентирован на B2C- и B2B-сегменты и направлен на удовлетворение спроса на доступные инструменты развития речевых и коммуникативных навыков.

Основной моделью монетизации является подписочная модель с элементами Freemium. Пользователям предоставляется бесплатный пробный доступ к базовому функционалу платформы, после чего они могут перейти на один из платных тарифов. Предполагаются два основных тарифа: «Студенческий» стоимостью 500 руб. в месяц и «Стандартный» стоимостью 1700 руб. в месяц. В перспективе возможно введение премиум-тарифа, включающего индивидуальные консультации, мастер-классы, сертификаты и расширенную аналитику.

Дополнительным направлением коммерциализации является работа с корпоративными клиентами. Для B2B-сегмента предполагается продажа лицензий на использование платформы на определённый срок. Стоимость будет зависеть от количества сотрудников, объёма образовательного контента и дополнительных технических требований. Также возможна продажа специализированных курсов, проведение вебинаров и мастер-классов, а также выпуск подарочных сертификатов.

Маркетинговая стратегия проекта предполагает использование цифровых каналов продвижения: SEO и ASO-оптимизацию, контекстную и таргетированную рекламу, продвижение в социальных сетях, контент-маркетинг и сотрудничество с образовательными блогерами, вузами, онлайн-школами и бизнес-сообществами. Дополнительно планируется PR-продвижение через профильные СМИ, участие в конференциях, проведение мастер-классов в образовательных организациях и запуск программы лояльности для

					ПГУ 1.09.03.01.50.001 ПЗ	Лист
						5
Изм.	Лист	№ докум.	Подпись	Дата		6

пользователей.

Коммерциализация проекта будет осуществляться поэтапно. На первом этапе планируется запуск MVP с базовым функционалом ИИ-анализа речи и ключевыми обучающими модулями, привлечение первых пользователей и сбор обратной связи. На втором этапе предполагается расширение функционала, добавление новых упражнений и запуск пилотных проектов с корпоративными клиентами. На третьем этапе планируется укрепление позиций на российском рынке, внедрение дополнительных платных услуг и рассмотрение возможностей выхода на международные рынки.

Финансовая модель проекта основана на прогнозе роста числа платных пользователей: 1000 пользователей в первый год, 3000 – во второй год и 6000 – в третий год. При соотношении тарифов «Студенческий» и «Стандартный» 40% к 60% средний ежемесячный доход с пользователя составит 1220 руб.

Прогнозируемая выручка проекта составляет:

- 1-й год: 14 640 000 руб.;
- 2-й год: 43 920 000 руб.;
- 3-й год: 87 840 000 руб.

С учётом операционных расходов прибыль до налогообложения может составить 1 640 000 руб. в первый год, 31 420 000 руб. во второй год и 73 490 000 руб. в третий год. Таким образом, проект способен достичь точки безубыточности уже в первый год работы.

При первоначальных инвестициях в разработку около 5 млн руб. платформа демонстрирует быструю окупаемость и высокую экономическую привлекательность.

Риски проекта оцениваются как умеренные, поскольку онлайн-формат и автоматизированная система анализа речи позволяют снизить зависимость от большого штата специалистов и сократить постоянные расходы. По мере роста узнаваемости платформы и подтверждения её эффективности возможно увеличение стоимости услуг на 20–30% без существенного снижения спроса.

Таким образом, план коммерциализации платформы «ПРОПС»

					ПГУ 1.09.03.01.50.001 ПЗ	Лист
						5
Изм.	Лист	№ докум.	Подпись	Дата		7

демонстрирует потенциал устойчивого развития, масштабирования и получения прибыли при сохранении доступности продукта для целевой аудитории.

3.4 Обоснование достижимости результата

В данном разделе обосновывается практическая и теоретическая достижимость результатов, заявленных в проекте «ПРОПС». Аргументация опирается на три ключевых компонента: достаточность и качество обучающей выборки, адекватность математического аппарата и архитектурных решений, а также техническая реализуемость ключевых функций системы.

Достаточность и качество обучающей выборки.

– Объем данных. Для обучения и валидации описанной нейронной архитектуры использовалась обучающая база общей длительностью $4,38 \cdot 10^6$ секунд аудиозаписей русскоязычной речи с разметкой дефектов произношения. Такой объём обеспечивает статистическую репрезентативность по большинству типичных вариаций речи и позволяет надежно аппроксимировать распределения акустических признаков для классов дефектов А, В, С, F.

– Представительность. Набор включает записи носителей разных возрастных и демографических групп, разнообразные акустические условия записи и множество комбинаций совместно встречающихся дефектов. Это уменьшает риск смещения и повышает обобщающую способность модели.

– Влияние объёма на оценку редких событий. Большой объём данных гарантирует достаточное количество наблюдений даже для относительно редких сочетаний дефектов, что позволяет получать нетривиальные оценки условных вероятностей вида $P(X|Y)$.

Адекватность выбранного математического аппарата.

Эффективность CNN в спектральном анализе: артикуляционные дефекты проявляются как специфические аномалии в частотно-временном представлении сигнала. CNN, используя операцию свертки, способна автоматически выявлять иерархические спектральные паттерны, которые человек-эксперт или

традиционные алгоритмы могут пропустить.

Обработка совместных дефектов: применение все в качестве функции потерь позволяет модели предсказывать наличие каждого из четырех дефектов $X \in M$ независимо. Это прямо соответствует статистической модели, где дефекты могут встречаться совместно, и обеспечивает высокую точность даже при сложных комбинациях нарушений, подтвержденных анализом условных вероятностей $P(X|Y)$.

Временная локализация: поскольку *CNN* обрабатывает Лог-Мел пектрограмму покадрово, модель естественным образом решает задачу временной локализации, указывая точные интервалы, где дефект присутствует.

Достижимость целей проекта «ПРОПС» подтверждается технической возможностью реализации его ключевых функций:

- автоматическая диагностика дикции и артикуляции;
- обнаружение дефектов А, В, С, F является задачей акустической фонетики, которая успешно решается с помощью глубокого обучения. Эти дефекты имеют четкие акустические корреляты:
 - ротацизм (В): аномалии в третьей и четвертой формантах, отсутствие вибрации;
 - сигматизм: искажение спектральной плотности шума, смещение центра тяжести шума;
 - CNN, обученная на большом объеме данных, может достичь точности, сопоставимой с экспертной оценкой, в задаче покадровой классификации;
 - анализ комплексных параметров речи.

Помимо дикции, система анализирует лексику, грамматику, орфоэпию и темп. Достижимость этих функций основана на интеграции:

					ПГУ 1.09.03.01.50.001 ПЗ	Лист
						5
Изм.	Лист	№ докум.	Подпись	Дата		9

– орфоэпия (ударение): достижима путем сопоставления предсказанного ударения (полученного из акустических признаков и частоты основного тона) с нормативным ударением в словаре;

– лексика и грамматика: требуют интеграции с моделями обработки естественного языка, которые могут быть реализованы на основе трансформеров или рекуррентных нейронных сетей, работающих с текстовой транскрипцией, полученной после распознавания речи.

Таким образом, ядро проекта – высокоточная диагностика дикционных дефектов – полностью обосновано с математической и методологической точек зрения, а интеграция дополнительных модулей позволяет достичь заявленной функциональности по замещению функций преподавателя риторики.

Практическая обоснованность временных и ресурсных ограничений.

Формат занятий: 15–20 минут в день. Экспериментальные данные и существующие исследования по моторному обучению речи указывают на эффективность частых коротких тренировочных сессий для закрепления артикуляционных навыков. Система адаптирует нагрузку и содержание упражнений под темп прогресса, что увеличивает вероятность достижения целей в короткие сроки.

В расках работы над проектом были опубликованы следующие научные труды:

– Алимуратов А.К. Разработка способа нейросетевого детектирования речевых ошибок / А.К. Алимуратов, А.О. Березина, А.В. Медведев, К.С. Шнайдер, А.А. Мамонова // сборник трудов XXX международного симпозиума «Надежность и качество» / под общей редакцией Н.К. Юркова. - Пенза: Изд-во ПГУ, 2025. - Т. 1, - С. 320 – 323;

– А.К. Аллимуратов, А.О.Березина, А.Ю. Тычков, Б.А. Порезанов, А.А. Мамонова СЕГМЕНТАЦИЯ РЕЧЕВЫХ СИГНАЛОВ НА ТОНАЛЬНЫЕ И НЕТОНАЛЬНЫЕ УЧАСТКИ // Измерение. Мониторинг. Управление. Контроль. 2025. №4;

– Березина А.О., Медведев А.В., Шнайдер К.С., Митрохин М.А.

					ПГУ 1.09.03.01.50.001 ПЗ	Лист
						6
Изм.	Лист	№ докум.	Подпись	Дата		0

Интеграция нейросетевых технологий в современное образование для развития «гибких» навыков // Компьютерные инструменты в образовании. 2025. No1. С.48–60. doi: 10.32603/2071-2340-2025-1-48-60;

– Свидетельство о государственной регистрации программы для ЭВМ № 2024682565 // Программа нейросетевого анализа произношения правильной и неправильной постановки ударения;

– Свидетельство о государственной регистрации программы для ЭВМ № 2024682070 // Программа нейросетевого анализа произношения согласных звуков.

Команда проекта «ПРОПС» прошла акселерационную программу «Медитехнологии», организованную Агентством инноваций Москвы, подтверждение участия представлено в Приложении В. Впоследствии в рамках фестиваля «Ночь инноваций» в Москве была проведена апробация продукта «ПРОПС». По итогам тестирования были получены следующие результаты: скорость анализа речевых сигналов не превышала 7 секунд. Удобство использования подтвердили 98% участников, а все 100% опрошенных выразили заинтересованность в прохождении обучения на платформе, адаптированного под их индивидуальные речевые дефекты.

При первоначальных инвестициях в разработку около 5 млн руб. платформа демонстрирует быструю окупаемость и высокую экономическую привлекательность.

Риски проекта оцениваются как умеренные, поскольку онлайн-формат и автоматизированная система анализа речи позволяют снизить зависимость от большого штата специалистов и сократить постоянные расходы. По мере роста узнаваемости платформы и подтверждения её эффективности возможно увеличение стоимости услуг на 20 –30% без существенного снижения спроса.

Таким образом, план коммерциализации платформы «ПРОПС» демонстрирует потенциал устойчивого развития, масштабирования и получения прибыли при сохранении доступности продукта для целевой аудитории.

					ПГУ 1.09.03.01.50.001 ПЗ	Лист
						6
Изм.	Лист	№ докум.	Подпись	Дата		1

Заключение

В ходе выполнения работы изучены принципы построения клиент–серверных приложений с микросервисной архитектурой. Изучены подходы к проектированию баз данных, технологии взаимодействия компонентов посредством REST API, Apache Kafka и контейнеризации Docker. Получены навыки проектирования и реализации серверной части распределённой системы с многослойной организацией кода.

В процессе работы спроектирована и реализована серверная часть интеллектуальной образовательной платформы, включающая набор микросервисов, API Gateway и систему хранения данных с разделением по предметным областям. Для типового сервиса выполнено детальное описание архитектурных слоёв и сценария создания курса. Для модулей сформулированы и проверены функциональные требования, в том числе к валидации входных данных, уникальности сущностей, целостности связей и разграничению доступа.

В результате показано, что применение микросервисной архитектуры в сочетании с REST и асинхронным обменом сообщениями позволяет гибко развивать платформу, масштабировать отдельные компоненты и обеспечивать выполнение основных бизнес-сценариев. Использование кэширования и выделенных баз данных по доменам снижает нагрузку на СУБД и упрощает сопровождение системы.

Выполнен сравнительный анализ существующих образовательных платформ, что позволило обосновать актуальность разработки и выделить отличительную функцию проектируемого решения – поддержку проверки и коррекции речи с применением ИИ-ассистента.

В дальнейшем целесообразно расширить функциональность сервиса речевой аналитики, провести нагрузочное тестирование микросервисов, а также доработать мониторинг и отказоустойчивость.

					ПГУ 1.09.03.01.50.001 ПЗ	Лист
						6
Изм.	Лист	№ докум.	Подпись	Дата		2

Список использованных источников

1. UML – диаграмма вариантов использования // <https://habr.com> [Электронный ресурс]. URL: <https://habr.com/ru/articles/47940/> (дата обращения: 05.05.2026).
2. UML: обзор основных типов диаграмм, диаграмма компонентов. Часть 2 // <https://habr.com> [Электронный ресурс]. URL: <https://habr.com/ru/articles/756552/> (дата обращения: 05.05.2026).
3. Просто о сложном. Концептуальная модель // <https://mlkv.ru> [Электронный ресурс]. URL: <https://mlkv.ru/sample-page/материалы-по-базам-данных/концептуальная-модель/> (дата обращения: 05.05.2026).
4. Верников Г. Основы методологии IDEF1X // Проект «Корпоративный менеджмент» [Электронный ресурс]. URL: <https://www.cfin.ru/vernikov/idef/idef1x.shtml/> (дата обращения: 05.05.2026).
5. Клиент-серверная архитектура // <https://ru.hexlet.io> [Электронный ресурс]. URL: https://ru.hexlet.io/courses/internet-fundamentals/lessons/client-server/theory_unit (дата обращения: 05.05.2026).
6. Моделирование данных: обзор // <https://habr.com> [Электронный ресурс]. URL: <https://habr.com/ru/articles/556790/> (дата обращения: 10.05.2026).
7. Преимущества и недостатки клиент-серверной модели // <https://sky.pro> [Электронный ресурс]. URL: <https://sky.pro/wiki/gamedev/preimushestva-i-nedostatki-klient-servernoj-modeli-v-igrah/> (дата обращения: 10.05.2026).
8. Ньюмен С. От монолита к микросервисам. Эволюционные шаблоны для трансформации монолитной системы. – СПб.: БХВ-Петербург, 2022. – С. 21.
9. Моделирование данных: что это и зачем нужно // <https://practicum.yandex.ru> [Электронный ресурс]. URL: <https://practicum.yandex.ru/blog/chto-takoe-modelirovanie-dannyh-i-kak-ego-ispolzovat/> (дата обращения: 10.05.2026).
10. Донован А.А., Керниган Б.У. Язык программирования Go. – М.: Издательство «Вильямс», 2016. 432 с.

11. PostgreSQL: The World's Most Advanced Open Source Relational Database // <https://www.postgresql.org> [Электронный ресурс]. URL: <https://www.postgresql.org/docs/> (дата обращения: 10.05.2026).

12. Apache Kafka Documentation // <https://kafka.apache.org> [Электронный ресурс]. URL: <https://kafka.apache.org/documentation/> (дата обращения: 14.05.2026).

13. Многоуровневая архитектура программного обеспечения // <https://habr.com> [Электронный ресурс]. URL: <https://habr.com/ru/articles/313216/> (дата обращения: 14.05.2026).

14. Физическая модель данных // <https://practicum.yandex.ru> [Электронный ресурс]. URL: <https://practicum.yandex.ru/blog/fizicheskaya-model-dannyh/> (дата обращения: 14.05.2026).

15. What is REST // <https://restfulapi.net> [Электронный ресурс]. URL: <https://restfulapi.net/> (дата обращения: 05.05.2026).

16. Docker Overview // <https://docs.docker.com> [Электронный ресурс]. URL: <https://docs.docker.com/get-started/docker-overview/> (дата обращения: 16.05.2026).

17. Redis Documentation // <https://redis.io> [Электронный ресурс]. URL: <https://redis.io/docs/> (дата обращения: 16.05.2026).

18. СанПиН 1.2.3685-21 «Гигиенические нормативы и требования к обеспечению безопасности и (или) безвредности для человека факторов среды обитания» // Гарант [Электронный ресурс]. URL: <https://base.garant.ru/400392920/> (дата обращения: 16.05.2026).

					ПГУ 1.09.03.01.50.001 ПЗ	Лист
						6
Изм.	Лист	№ докум.	Подпись	Дата		4

Приложение А

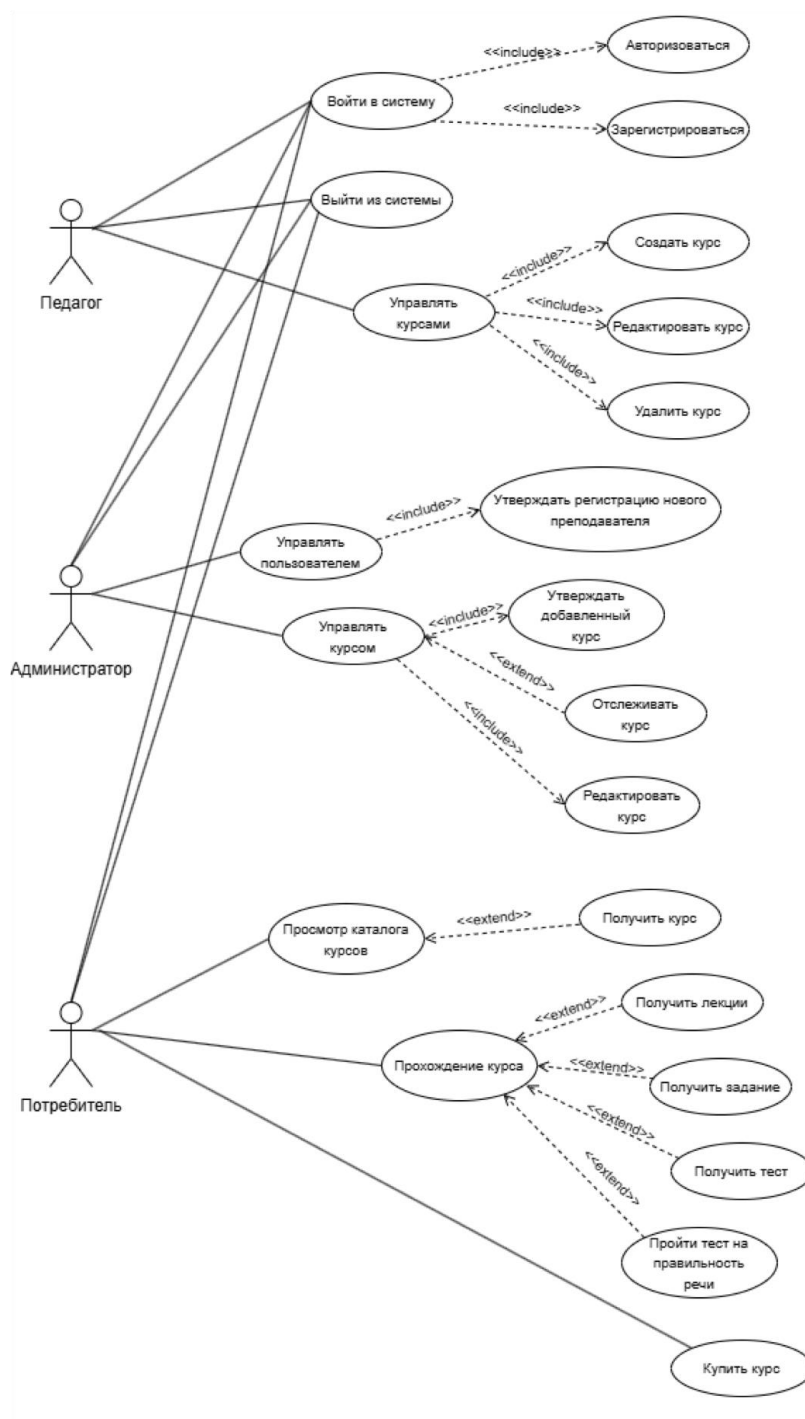
(обязательное)

Бизнес-проект Интеллектуальная образовательная платформа

UML – диаграммы

					ПГУ 1.09.03.01.50.001 ПЗ	Лист
						6
Изм.	Лист	№ докум.	Подпись	Дата		5

Приложение А.1 – Диаграмма вариантов использования



ПГУ 1.090301.50.002 Д13					
Имя	Фамилия	Имя отчества	Подпись	Дата	
Составитель	Сидорова А.С.				
Проверил	Михайлов М.А.				
Т. номер					
Л. номер	Сидорова А.С.				
Т.ж.	Михайлов М.А.				

Вариант-проект Индивидуальная оценка компетенции

УМ. Оценочные материалы

Лист 1

Лист 1

ФВТ, стр. 228В72

Изм.	Лист	№ докум.	Подпись	Дата

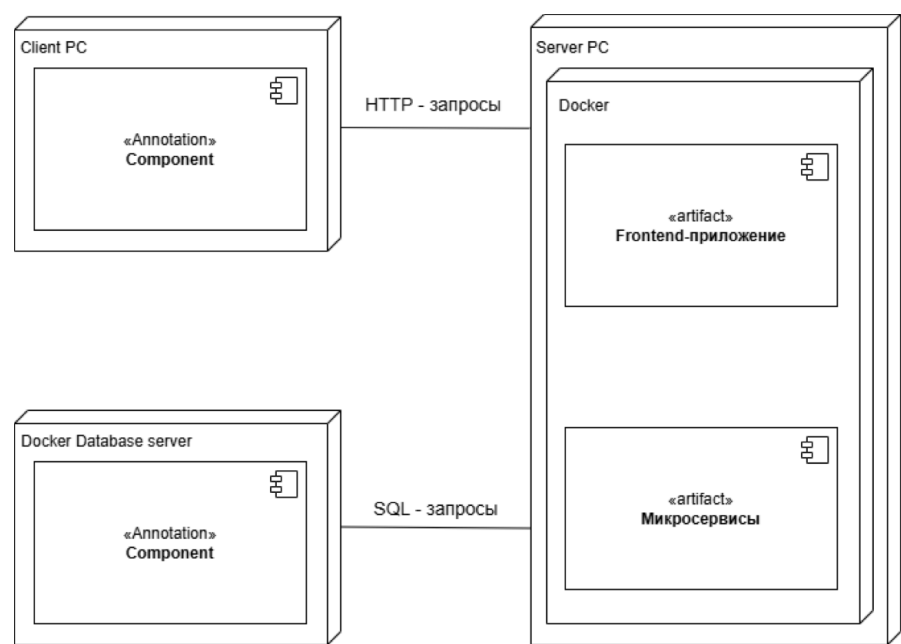
ПГУ 1.09.03.01.50.001 ПЗ

Лист

6

6

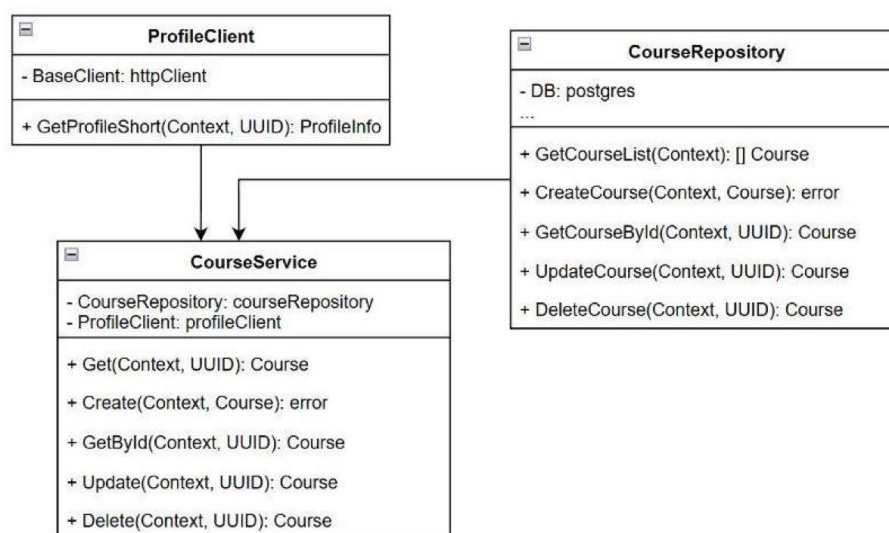
Приложение А.2 – Диаграмма развёртывания



						ПГУ 1.090301.50.002 Д19			
Созд	Изм	Исх	Исх	Исх	Исх	Исх	Исх	Исх	Исх
Исх	Исх	Исх	Исх	Исх	Исх	Исх	Исх	Исх	Исх
Исх	Исх	Исх	Исх	Исх	Исх	Исх	Исх	Исх	Исх
Исх	Исх	Исх	Исх	Исх	Исх	Исх	Исх	Исх	Исх
Исх	Исх	Исх	Исх	Исх	Исх	Исх	Исх	Исх	Исх
Исх	Исх	Исх	Исх	Исх	Исх	Исх	Исх	Исх	Исх
Исх	Исх	Исх	Исх	Исх	Исх	Исх	Исх	Исх	Исх
Исх	Исх	Исх	Исх	Исх	Исх	Исх	Исх	Исх	Исх
Исх	Исх	Исх	Исх	Исх	Исх	Исх	Исх	Исх	Исх

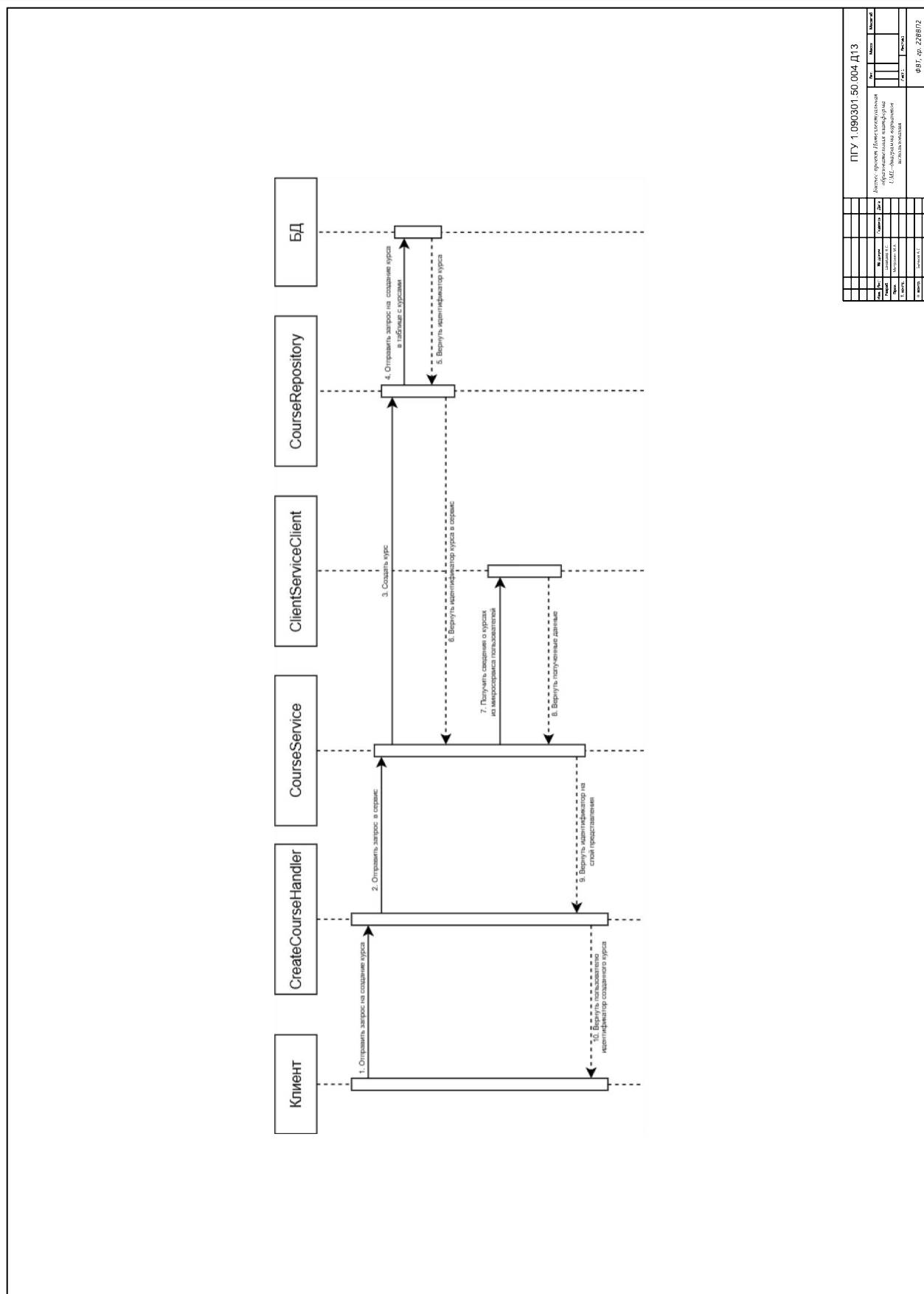
Изм.	Лист	№ докум.	Подпись	Дата

Приложение А.3 - Диаграмма классов



						ПГУ 1.090301.50.002 Д11				
						Бизнес-процесс: Исполнительные операции классифицируемые СМЛ: выданные классом				
Фин. Инст.	№ документа	Подпись	Дата							
Решение	Исполнитель К.С.									
Протокол	Материалы М.А.									
1. Инициация										
2. Инициация	Сервисная А.С.									
							Ф8Т, зр. 228ВП2			

Приложение А.4 - Диаграмма последовательности



Приложение Б
(обязательное)

Бизнес-проект Интеллектуальная образовательная платформа

Листинг программы

					ПГУ 1.09.03.01.50.001 ПЗ	Лист
						7
Изм.	Лист	№ докум.	Подпись	Дата		0

Файл «main.go»

```

package main
import (
    "context"
    "course-service/config"
    "course-service/internal/app"
    "course-service/pkg/pctx"
    "course-service/pkg/plogger"
    "os"
    "os/signal"
    "syscall"
    "time"
)
func main() {
    settings := config.Read()

    var opts []plogger.Option
    if settings.Env != config.EnvProd {
        opts = append(opts, plogger.Development())
    }
    logger := plogger.NewLogger("course-service", opts...)

    mainCtx, stop := signal.NotifyContext(context.Background(),
os.Interrupt, syscall.SIGTERM)
    defer stop()

    application, err := app.NewApp(logger, settings,
pctx.DefaultContextProvider(mainCtx))
    if err != nil {
        logger.Error(err.Error())
        return
    }

    application.Run()
    logger.Debug("course-service started")

    select {
    case <-mainCtx.Done():
    case <-application.Notify():
    }

    shutdownCtx, cancel := context.WithTimeout(context.Background(),
5*time.Second)
    defer cancel()

    application.Stop(shutdownCtx)
    logger.Debug("course-service stopped")
}

```

					ПГУ 1.09.03.01.50.001 ПЗ	Лист
						7
Изм.	Лист	№ докум.	Подпись	Дата		1

Файл «config/config.go»

```
package config

type Config struct {
    Env      Env      `json:"- "`
    Port     int      `json:"port"`
    Databases Databases `json:"databases"`
}

type Databases struct {
    Postgres Database `json:"postgres"`
}

type Database struct {
    ConnectionString string `json:"connectionString"`
    MigrationPath    string `json:"migrationPath"`
}
```

Файл «config/reader.go»

```
package config

import (
    "fmt"
    "github.com/spf13/viper"
    "os"
)

func Read() Config {
    env, configName := readEnv()

    viper.SetConfigName(configName)
    viper.SetConfigType("json")
    viper.AddConfigPath("./config")

    if err := viper.ReadInConfig(); err != nil {
        panic(fmt.Errorf("fatal error config file: %w", err))
    }

    var c Config
    c.Env = env
    if err := viper.Unmarshal(&c); err != nil {
        panic(fmt.Errorf("unable to decode into struct, %v", err))
    }

    return c
}

const _serviceEnvVarName = "ENV"

type Env string
```

					ПГУ 1.09.03.01.50.001 ПЗ	Лист
						7
Изм.	Лист	№ докум.	Подпись	Дата		2


```

const (
    EnvProd  = Env("prod")
    EnvTest  = Env("test")
    EnvLocal = Env("local")
    EnvEmpty = Env("")
)

func readEnv() (Env, string) {
    env := Env(os.Getenv(_serviceEnvVarName))

    var configName string
    switch env {
    case EnvProd:
        configName = "prod"
    case EnvTest:
        configName = "test"
    default:
        env = EnvLocal
        configName = "local"
    }

    return env, configName
}

```

Файл «config/local.json»

```

{
    "port": 8081,
    "databases": {
        "postgres": {
            "connectionString":
"postgres://postgres:postgres@localhost:5433/course_service?sslmode=disable",
            "migrationPath": "migration"
        }
    }
}

```

Файл «config/prod.json»

```

{
    "port": 8081,
    "databases": {
        "postgres": {
            "connectionString": "${POSTGRES_DSN}",
            "migrationPath": "migration"
        }
    }
}

```

					ПГУ 1.09.03.01.50.001 ПЗ	Лист
						7
Изм.	Лист	№ докум.	Подпись	Дата		3

Файл «internal/app/app.go»

```

package app

import (
    "context"
    "course-service/config"
    "course-service/internal/api"
    "course-service/internal/repository"
    "course-service/internal/service"
    "course-service/pkg/pctx"
    "course-service/pkg/plogger"
    "course-service/pkg/ppostgres"
    "errors"
    "fmt"
    "net/http"
)

type App struct {
    Server      *api.Server
    logger      plogger.Logger
    config      config.Config
    contextProvider pctx.DefaultContextProviderFunc
    notify      chan struct{}

    pgDB *ppostgres.Postgres

    courseRepo      *repository.CourseRepository
    courseService *service.CourseService
}

func NewApp(l plogger.Logger, cfg config.Config, contextProvider
pctx.DefaultContextProviderFunc) (*App, error) {
    application := &App{
        logger:      l,
        config:      cfg,
        contextProvider: contextProvider,
        notify:      make(chan struct{}, 1),
    }

    if err := application.initDatabases(); err != nil {
        return application, err
    }
    if err := application.initServices(cfg); err != nil {
        return application, err
    }

    return application, nil
}

func (a *App) initDatabases() error {
    var err error

```

					ПГУ 1.09.03.01.50.001 ПЗ	Лист
						7
Изм.	Лист	№ докум.	Подпись	Дата		4

```

if a.pgDB, err = ppostgres.NewWithMigration(
    a.contextProvider(),
    a.config.Databases.Postgres.ConnectionString,
    a.config.Databases.Postgres.MigrationPath,
); err != nil {
    return fmt.Errorf("database: %w", err)
}

a.courseRepo = repository.NewCourseRepository(a.pgDB.DB)
return nil
}

func (a *App) initServices(cfg config.Config) error {
    a.courseService = service.NewCourseService(a.courseRepo)
    a.Server = api.NewServer(cfg.Port, a.logger, a.contextProvider,
a.courseService)
    return nil
}

func (a *App) Run() {
    a.Server.Start()
    go func() {
        select {
        case err := <-a.Server.Notify():
            if !errors.Is(err, http.ErrServerClosed) {
                a.logger.Errorf(err.Error())
            }
        }
        a.notify <- struct{}{}
    }()
}

func (a *App) Notify() <-chan struct{} {
    return a.notify
}

func (a *App) Stop(ctx context.Context) {
    if a.Server != nil {
        if err := a.Server.Shutdown(ctx); err != nil {
            a.logger.Errorf("server: %v", err)
        }
    }
    if a.pgDB != nil {
        if err := a.pgDB.Close(); err != nil {
            a.logger.Errorf("pg database: %v", err)
        }
    }
}

```

Файл «internal/api/server.go»

package api

					ПГУ 1.09.03.01.50.001 ПЗ	Лист
						7
Изм.	Лист	№ докум.	Подпись	Дата		5


```

import (
    "context"
    "course-service/internal/api/handler"
    "course-service/internal/service"
    "course-service/pkg/pctx"
    "course-service/pkg/plogger"
    "fmt"
    "net"
    "net/http"
    "time"

    "github.com/gin-contrib/cors"
    "github.com/gin-gonic/gin"
)

const defaultShutdownTimeout = 3 * time.Second

type Server struct {
    Server          *http.Server
    notify          chan error
    shutdownTimeout time.Duration
}

func NewServer(
    port int,
    logger plogger.Logger,
    contextProvider pctx.DefaultContextProviderFunc,
    courseService *service.CourseService,
) *Server {
    r := gin.New()

    corsConfig := cors.DefaultConfig()
    corsConfig.AllowAllOrigins = true
    corsConfig.AddAllowHeaders("Authorization")
    r.Use(cors.New(corsConfig))

    apiGroup := r.Group("/api")
    {
        apiGroup.GET("/ping", handler.Ping())

        courses := apiGroup.Group("/courses")
        {
            courses.POST("/push", handler.PushCourse(logger,
courseService))
            courses.GET("/get/:profileId", handler.GetCourse(logger,
courseService))
            courses.GET("/list", handler.ListCourses(logger,
courseService))
        }

        teacherCourses := apiGroup.Group("/teacher/courses")

```

					ПГУ 1.09.03.01.50.001 ПЗ	Лист
						7
Изм.	Лист	№ докум.	Подпись	Дата		6

```

    {
        teacherCourses.POST("/create", handler.CreateCourse(logger,
courseService))
        teacherCourses.POST("/update/:courseId",
handler.UpdateCourse(logger, courseService))
        teacherCourses.DELETE("/delete/:courseId",
handler.DeleteCourse(logger, courseService))
        teacherCourses.POST("/:courseId/lecture/create",
handler.CreateLecture(logger, courseService))
        teacherCourses.POST("/:courseId/theory/create",
handler.CreateTheory(logger, courseService))
        teacherCourses.POST("/:courseId/practice/create",
handler.CreatePractice(logger, courseService))
    }

    apiGroup.DELETE("/admin/profiles/clear/:profileId/:courseId",
handler.ClearCourse(logger, courseService))
}

return &Server{
    Server: &http.Server{
        Addr:      fmt.Sprintf(":%d", port),
        Handler: r,
        BaseContext: func(listener net.Listener) context.Context {
            return contextProvider()
        },
    },
    notify:      make(chan error, 1),
    shutdownTimeout: defaultShutdownTimeout,
}
}

func (s *Server) Start() {
    go func() {
        s.notify <- s.Server.ListenAndServe()
        close(s.notify)
    }()
}

func (s *Server) Notify() <-chan error {
    return s.notify
}

func (s *Server) Shutdown(ctx context.Context) error {
    shutdownCtx, cancel := context.WithTimeout(ctx, s.shutdownTimeout)
    defer cancel()
    return s.Server.Shutdown(shutdownCtx)
}

```

Файл «internal/api/handler/ping.go»

package handler

					ПГУ 1.09.03.01.50.001 ПЗ	Лист
						7
Изм.	Лист	№ докум.	Подпись	Дата		7

```

import (
    "course-service/pkg/phttp"

    "github.com/gin-gonic/gin"
)

func Ping() gin.HandlerFunc {
    return func(c *gin.Context) {
        phttp.OkResponseWithResult(c, "pong")
    }
}

```

Файл «internal/api/handler/course.go»

```
package handler
```

```

import (
    "course-service/internal/entity"
    "course-service/internal/service"
    "course-service/pkg/phttp"
    "course-service/pkg/plogger"
    "errors"
    "net/http"

    "github.com/gin-gonic/gin"
    "github.com/google/uuid"
)

type courseRequest struct {
    ID          uuid.UUID `json:"id" `
    Name        string    `json:"name" binding:"required"`
    Description string    `json:"description" `
    Tag         string    `json:"tag" binding:"required"`
}

type lectureRequest struct {
    ID          uuid.UUID `json:"id" `
    Name        string    `json:"name" binding:"required"`
    Description string    `json:"description" `
    Tag         string    `json:"tag" binding:"required"`
    Link        string    `json:"link" `
}

type theoryRequest struct {
    ID          uuid.UUID `json:"id" `
    Name        string    `json:"name" binding:"required"`
    Description string    `json:"description" `
    Tag         string    `json:"tag" binding:"required"`
    Content     string    `json:"content" `
}

```

					ПГУ 1.09.03.01.50.001 ПЗ	Лист
						7
Изм.	Лист	№ докум.	Подпись	Дата		8


```

type practiceRequest struct {
    ID          uuid.UUID `json:"id" `
    Name        string    `json:"name" binding:"required"`
    Description string    `json:"description" `
    Tag         string    `json:"tag" binding:"required"`
}

type pushCourseRequest struct {
    ProfileID uuid.UUID `json:"profile_id" binding:"required"`
    Mistake   string    `json:"mistake_tag" binding:"required"`
}

func CreateCourse(logger plogger.Logger, courses
*service.CourseService) gin.HandlerFunc {
    return func(c *gin.Context) {
        var req courseRequest
        if err := c.ShouldBindJSON(&req); err != nil {
            phttp.ErrorResponse(c, http.StatusBadRequest, "bad body")
            return
        }

        id, err := courses.Create(c, entity.Course{
            ID:          req.ID,
            Name:        req.Name,
            Description: req.Description,
            Tag:         req.Tag,
        })
        writeNoContentOnError(c, logger, err)
        if err == nil {
            phttp.OkResponseWithResult(c, id)
        }
    }
}

func UpdateCourse(logger plogger.Logger, courses
*service.CourseService) gin.HandlerFunc {
    return func(c *gin.Context) {
        var req courseRequest
        if err := c.ShouldBindJSON(&req); err != nil {
            phttp.ErrorResponse(c, http.StatusBadRequest, "bad body")
            return
        }

        err := courses.Update(c, uuid.MustParse(c.Param("courseId")),
entity.Course{
            Name:        req.Name,
            Description: req.Description,
            Tag:         req.Tag,
        })
        writeNoContentOnError(c, logger, err)
    }
}

```

					ПГУ 1.09.03.01.50.001 ПЗ	Лист
						7
Изм.	Лист	№ докум.	Подпись	Дата		9

```

func DeleteCourse(logger plogger.Logger, courses
*service.CourseService) gin.HandlerFunc {
    return func(c *gin.Context) {
        writeNoContentOnError(c, logger, courses.Delete(c,
uuid.MustParse(c.Param("courseId"))))
    }
}

func ListCourses(logger plogger.Logger, courses
*service.CourseService) gin.HandlerFunc {
    return func(c *gin.Context) {
        result, err := courses.List(c)
        if err != nil {
            writeNoContentOnError(c, logger, err)
            return
        }
        phhttp.OkResponseWithResult(c, result)
    }
}

func PushCourse(logger plogger.Logger, courses
*service.CourseService) gin.HandlerFunc {
    return func(c *gin.Context) {
        var req pushCourseRequest
        if err := c.ShouldBindJSON(&req); err != nil {
            phhttp.ErrorResponse(c, http.StatusBadRequest, "bad body")
            return
        }
        writeNoContentOnError(c, logger, courses.Push(c, req.ProfileID,
req.Mistake))
    }
}

func GetCourse(logger plogger.Logger, courses
*service.CourseService) gin.HandlerFunc {
    return func(c *gin.Context) {
        course, err := courses.Get(c,
uuid.MustParse(c.Param("profileId")))
        if err != nil {
            writeNoContentOnError(c, logger, err)
            return
        }
        phhttp.OkResponseWithResult(c, course)
    }
}

func ClearCourse(logger plogger.Logger, courses
*service.CourseService) gin.HandlerFunc {
    return func(c *gin.Context) {
        profileID := uuid.MustParse(c.Param("profileId"))
        courseID := uuid.MustParse(c.Param("courseId"))
    }
}

```

					ПГУ 1.09.03.01.50.001 ПЗ	Лист
						8
Изм.	Лист	№ докум.	Подпись	Дата		0

```

        writeNoContentOrError(c, logger, courses.ClearCourse(c,
profileID, courseID))
    }
}

func CreateLecture(logger plogger.Logger, courses
*service.CourseService) gin.HandlerFunc {
    return func(c *gin.Context) {
        var req lectureRequest
        if err := c.ShouldBindJSON(&req); err != nil {
            phttp.ErrorResponse(c, http.StatusBadRequest, "bad body")
            return
        }
        err := courses.CreateLecture(c,
uuid.MustParse(c.Param("courseId")), entity.Lecture{
            ID:          req.ID,
            Name:         req.Name,
            Description: req.Description,
            Tag:          req.Tag,
            Link:         req.Link,
        })
        writeNoContentOrError(c, logger, err)
    }
}

func CreateTheory(logger plogger.Logger, courses
*service.CourseService) gin.HandlerFunc {
    return func(c *gin.Context) {
        var req theoryRequest
        if err := c.ShouldBindJSON(&req); err != nil {
            phttp.ErrorResponse(c, http.StatusBadRequest, "bad body")
            return
        }
        err := courses.CreateTheory(c,
uuid.MustParse(c.Param("courseId")), entity.Theory{
            ID:          req.ID,
            Name:         req.Name,
            Description: req.Description,
            Tag:          req.Tag,
            Content:      req.Content,
        })
        writeNoContentOrError(c, logger, err)
    }
}

func CreatePractice(logger plogger.Logger, courses
*service.CourseService) gin.HandlerFunc {
    return func(c *gin.Context) {
        var req practiceRequest
        if err := c.ShouldBindJSON(&req); err != nil {
            phttp.ErrorResponse(c, http.StatusBadRequest, "bad body")
            return
        }
    }
}

```



```

    }
    err := courses.CreatePractice(c,
uuid.MustParse(c.Param("courseId")), entity.Practice{
    ID:          req.ID,
    Name:        req.Name,
    Description: req.Description,
    Tag:         req.Tag,
    })
    writeNoContentOnError(c, logger, err)
}
}

func writeNoContentOnError(c *gin.Context, logger plogger.Logger,
err error) {
    switch {
    case err == nil:
        phttp.OkResponse(c)
    case errors.Is(err, service.ErrNotFound):
        phttp.ErrorResponse(c, http.StatusNotFound,
service.ErrNotFound.Error())
    case errors.Is(err, service.ErrAlreadyExists):
        phttp.ErrorResponse(c, http.StatusConflict,
service.ErrAlreadyExists.Error())
    default:
        logger.Error(err.Error())
        phttp.ErrorResponse(c, http.StatusInternalServerError, "internal
server error")
    }
}

```

Файл «internal/entity/course.go»

```

package entity

import "github.com/google/uuid"

type Course struct {
    ID          uuid.UUID `json:"id" db:"id"`
    Name        string    `json:"name" db:"course_name"`
    Description string    `json:"description" db:"description"`
    Tag         string    `json:"tag" db:"tag"`
    Lectures    []Lecture `json:"lectures,omitempty"`
    Theories    []Theory  `json:"theories,omitempty"`
    Practices   []Practice `json:"practices,omitempty"`
}

type Lecture struct {
    ID          uuid.UUID `json:"id" db:"id"`
    CourseID    uuid.UUID `json:"course_id" db:"course_id"`
    Name        string    `json:"name" db:"lecture_name"`
    Description string    `json:"description" db:"description"`
}

```

					ПГУ 1.09.03.01.50.001 ПЗ	Лист
						8
Изм.	Лист	№ докум.	Подпись	Дата		2

```

Tag          string    `json:"tag" db:"tag"`
Link         string    `json:"link" db:"link"`
}

type Theory struct {
ID           uuid.UUID `json:"id" db:"id"`
CourseID     uuid.UUID `json:"course_id" db:"course_id"`
Name         string    `json:"name" db:"theory_name"`
Description  string    `json:"description" db:"description"`
Tag          string    `json:"tag" db:"tag"`
Content      string    `json:"content" db:"content"`
}

type Practice struct {
ID           uuid.UUID `json:"id" db:"id"`
CourseID     uuid.UUID `json:"course_id" db:"course_id"`
Name         string    `json:"name" db:"practice_name"`
Description  string    `json:"description" db:"description"`
Tag          string    `json:"tag" db:"tag"`
Tasks        []Task    `json:"tasks,omitempty"`
}

type Task struct {
ID           uuid.UUID `json:"id" db:"id"`
PracticeID   uuid.UUID `json:"practice_id" db:"practice_id"`
Name         string    `json:"name" db:"task_name"`
Body         string    `json:"body" db:"body"`
}

```

Файл «internal/service/errors.go»

```

package service

import "errors"

var (
ErrNotFound      = errors.New("not found")
ErrAlreadyExists = errors.New("already exists")
)

```

Файл «internal/service/course.go»

```

package service

import (
"context"
"course-service/internal/entity"
"course-service/internal/repository"
"errors"
)

```

					ПГУ 1.09.03.01.50.001 ПЗ	Лист
						8
Изм.	Лист	№ докум.	Подпись	Дата		3

```

    "github.com/google/uuid"
)

type CourseService struct {
    repo *repository.CourseRepository
}

func NewCourseService(repo *repository.CourseRepository)
*CourseService {
    return &CourseService{repo: repo}
}

func (s *CourseService) Create(ctx context.Context, course
entity.Course) (uuid.UUID, error) {
    if course.ID == uuid.Nil {
        course.ID = uuid.New()
    }
    if err := s.repo.CreateCourse(ctx, course); err != nil {
        if errors.Is(err, repository.ErrAlreadyExists) {
            return uuid.Nil, ErrAlreadyExists
        }
        return uuid.Nil, err
    }
    return course.ID, nil
}

func (s *CourseService) Update(ctx context.Context, id uuid.UUID,
course entity.Course) error {
    course.ID = id
    if err := s.repo.UpdateCourse(ctx, course); err != nil {
        return mapRepoError(err)
    }
    return nil
}

func (s *CourseService) Delete(ctx context.Context, id uuid.UUID)
error {
    return mapRepoError(s.repo.DeleteCourse(ctx, id))
}

func (s *CourseService) List(ctx context.Context) ([]entity.Course,
error) {
    return s.repo.ListCourses(ctx)
}

func (s *CourseService) Push(ctx context.Context, profileID
uuid.UUID, mistake string) error {
    return mapRepoError(s.repo.AssignCourseByTag(ctx, profileID,
mistake))
}

```

					ПГУ 1.09.03.01.50.001 ПЗ	Лист
						8
Изм.	Лист	№ докум.	Подпись	Дата		4


```

func (s *CourseService) ClearCourse(ctx context.Context, profileID,
courseID uuid.UUID) error {
    return mapRepoError(s.repo.ClearProfileCourse(ctx, profileID,
courseID))
}

func (s *CourseService) Get(ctx context.Context, profileID
uuid.UUID) (entity.Course, error) {
    course, err := s.repo.GetCourseByProfile(ctx, profileID)
    if err != nil {
        return entity.Course{}, mapRepoError(err)
    }
    return course, nil
}

func (s *CourseService) CreateLecture(ctx context.Context, courseID
uuid.UUID, lecture entity.Lecture) error {
    if lecture.ID == uuid.Nil {
        lecture.ID = uuid.New()
    }
    lecture.CourseID = courseID
    return mapRepoError(s.repo.CreateLecture(ctx, lecture))
}

func (s *CourseService) CreateTheory(ctx context.Context, courseID
uuid.UUID, theory entity.Theory) error {
    if theory.ID == uuid.Nil {
        theory.ID = uuid.New()
    }
    theory.CourseID = courseID
    return mapRepoError(s.repo.CreateTheory(ctx, theory))
}

func (s *CourseService) CreatePractice(ctx context.Context,
courseID uuid.UUID, practice entity.Practice) error {
    if practice.ID == uuid.Nil {
        practice.ID = uuid.New()
    }
    practice.CourseID = courseID
    return mapRepoError(s.repo.CreatePractice(ctx, practice))
}

func mapRepoError(err error) error {
    switch {
    case err == nil:
        return nil
    case errors.Is(err, repository.ErrNotFound):
        return ErrNotFound
    case errors.Is(err, repository.ErrAlreadyExists):
        return ErrAlreadyExists
    default:
        return err
    }
}

```

					ПГУ 1.09.03.01.50.001 ПЗ	Лист
						8
Изм.	Лист	№ докум.	Подпись	Дата		5

```
}  
}
```

Файл «internal/repository/course.go»

```
package repository  
  
import (  
    "context"  
    "course-service/internal/entity"  
    "database/sql"  
    "errors"  
  
    "github.com/google/uuid"  
    "github.com/jmoiron/sqlx"  
    "github.com/lib/pq"  
)  
  
var (  
    ErrNotFound          = errors.New("not found")  
    ErrAlreadyExists     = errors.New("already exists")  
)  
  
type CourseRepository struct {  
    db *sqlx.DB  
}  
  
func NewCourseRepository(db *sqlx.DB) *CourseRepository {  
    return &CourseRepository{db: db}  
}  
  
func (r *CourseRepository) CreateCourse(ctx context.Context, course  
entity.Course) error {  
    _, err := r.db.ExecContext(  
        ctx,  
        `insert into courses (id, course_name, description, tag) values  
($1,$2,$3,$4)`,  
        course.ID, course.Name, course.Description, course.Tag,  
    )  
    return mapPostgresError(err)  
}  
  
func (r *CourseRepository) UpdateCourse(ctx context.Context, course  
entity.Course) error {  
    res, err := r.db.ExecContext(  
        ctx,  
        `update courses set course_name = $1, description = $2, tag = $3  
where id = $4`,  
        course.Name, course.Description, course.Tag, course.ID,  
    )  
    if err != nil {  
        return mapPostgresError(err)  
    }
```

					ПГУ 1.09.03.01.50.001 ПЗ	Лист
						8
Изм.	Лист	№ докум.	Подпись	Дата		6

```

    }
    return rowsAffectedToError(res)
}

func (r *CourseRepository) DeleteCourse(ctx context.Context, id
uuid.UUID) error {
    res, err := r.db.ExecContext(ctx, `delete from courses where id =
$1`, id)
    if err != nil {
        return err
    }
    return rowsAffectedToError(res)
}

func (r *CourseRepository) ListCourses(ctx context.Context)
([]entity.Course, error) {
    var courses []entity.Course
    err := r.db.SelectContext(ctx, &courses, `select id, course_name,
description, tag from courses`)
    return courses, err
}

func (r *CourseRepository) CreateLecture(ctx context.Context,
lecture entity.Lecture) error {
    _, err := r.db.ExecContext(
        ctx,
        `insert into lectures (id, course_id, lecture_name, description,
tag, link) values ($1,$2,$3,$4,$5,$6)`,
        lecture.ID, lecture.CourseID, lecture.Name, lecture.Description,
lecture.Tag, lecture.Link,
    )
    return mapPostgresError(err)
}

func (r *CourseRepository) CreateTheory(ctx context.Context, theory
entity.Theory) error {
    _, err := r.db.ExecContext(
        ctx,
        `insert into theories (id, course_id, theory_name, description,
tag, content) values ($1,$2,$3,$4,$5,$6)`,
        theory.ID, theory.CourseID, theory.Name, theory.Description,
theory.Tag, theory.Content,
    )
    return mapPostgresError(err)
}

func (r *CourseRepository) CreatePractice(ctx context.Context,
practice entity.Practice) error {
    _, err := r.db.ExecContext(
        ctx,
        `insert into practices (id, course_id, practice_name,
description, tag) values ($1,$2,$3,$4,$5)`,

```

					ПГУ 1.09.03.01.50.001 ПЗ	Лист
						8
Изм.	Лист	№ докум.	Подпись	Дата		7


```

        practice.ID, practice.CourseID, practice.Name,
        practice.Description, practice.Tag,
    )
    return mapPostgresError(err)
}

func (r *CourseRepository) AssignCourseByTag(ctx context.Context,
profileID uuid.UUID, tag string) error {
    var courseID uuid.UUID
    err := r.db.GetContext(ctx, &courseID, `select id from courses
where tag = $1`, tag)
    if err != nil {
        if errors.Is(err, sql.ErrNoRows) {
            return ErrNotFound
        }
        return err
    }

    _, err = r.db.ExecContext(
        ctx,
        `insert into profile_courses (profile_id, course_id) values
($1,$2)
on conflict (profile_id) do update set course_id =
excluded.course_id`,
        profileID, courseID,
    )
    if err != nil {
        return err
    }

    return r.SetMistake(ctx, profileID, tag)
}

func (r *CourseRepository) ClearProfileCourse(ctx context.Context,
profileID, courseID uuid.UUID) error {
    res, err := r.db.ExecContext(
        ctx,
        `delete from profile_courses where profile_id = $1 and course_id
= $2`,
        profileID, courseID,
    )
    if err != nil {
        return err
    }
    return rowsAffectedToError(res)
}

func (r *CourseRepository) GetCourseByProfile(ctx context.Context,
profileID uuid.UUID) (entity.Course, error) {
    var course entity.Course
    err := r.db.GetContext(
        ctx,

```

					ПГУ 1.09.03.01.50.001 ПЗ	Лист
						8
Изм.	Лист	№ докум.	Подпись	Дата		8

```

        &course,
        `select c.id, c.course_name, c.description, c.tag
        from profile_courses pc
        join courses c on c.id = pc.course_id
        where pc.profile_id = $1`,
        profileID,
    )
    if err != nil {
        if errors.Is(err, sql.ErrNoRows) {
            return entity.Course{}, ErrNotFound
        }
        return entity.Course{}, err
    }

    if err = r.db.SelectContext(ctx, &course.Lectures, `select id,
course_id, lecture_name, description, tag, link from lectures where
course_id = $1`, course.ID); err != nil {
        return entity.Course{}, err
    }
    if err = r.db.SelectContext(ctx, &course.Theories, `select id,
course_id, theory_name, description, tag, content from theories
where course_id = $1`, course.ID); err != nil {
        return entity.Course{}, err
    }
    if err = r.db.SelectContext(ctx, &course.Practices, `select id,
course_id, practice_name, description, tag from practices where
course_id = $1`, course.ID); err != nil {
        return entity.Course{}, err
    }

    return course, nil
}

func (r *CourseRepository) SetMistake(ctx context.Context,
profileID uuid.UUID, mistake string) error {
    _, err := r.db.ExecContext(
        ctx,
        `insert into profile_course_tags (profile_id, sigm, cart, lamb)
        values ($1, $2, $3, $4)
        on conflict (profile_id) do update set
            sigm = profile_course_tags.sigm or excluded.sigm,
            cart = profile_course_tags.cart or excluded.cart,
            lamb = profile_course_tags.lamb or excluded.lamb`,
        profileID,
        mistake == "Сигматизм",
        mistake == "Картавость",
        mistake == "Ламбдацизм",
    )
    return err
}

func rowsAffectedToError(res sql.Result) error {

```

					ПГУ 1.09.03.01.50.001 ПЗ	Лист
						8
Изм.	Лист	№ докум.	Подпись	Дата		9

```

affected, _ := res.RowsAffected()
if affected == 0 {
    return ErrNotFound
}
return nil
}

func mapPostgresError(err error) error {
    var pqErr *pq.Error
    if errors.As(err, &pqErr) && pqErr.Code == "23505" {
        return ErrAlreadyExists
    }
    return err
}

```

Файл «migration/0001_init.sql»

```

-- +goose Up
create extension if not exists "uuid-osspl";

create table if not exists "courses"
(
    id uuid primary key default uuid_generate_v4(),
    course_name varchar(255) not null,
    description text,
    tag varchar(255) not null unique
);

create table if not exists "profile_courses"
(
    profile_id uuid primary key,
    course_id uuid not null references "courses" ("id") on delete
cascade
);

create table if not exists "profile_course_tags"
(
    profile_id uuid primary key,
    sigm boolean not null default false,
    cart boolean not null default false,
    lamb boolean not null default false
);

create table if not exists "lectures"
(
    id uuid primary key default uuid_generate_v4(),
    course_id uuid not null references "courses" ("id") on delete
cascade,
    lecture_name varchar(255) not null,
    description text,
    tag varchar(255) not null,
    link varchar(255)
);

```

					ПГУ 1.09.03.01.50.001 ПЗ	Лист
						9
Изм.	Лист	№ докум.	Подпись	Дата		0


```

);

create table if not exists "theories"
(
    id uuid primary key default uuid_generate_v4(),
    course_id uuid not null references "courses" ("id") on delete
cascade,
    theory_name varchar(255) not null,
    description text,
    tag varchar(255) not null,
    content text
);

create table if not exists "practices"
(
    id uuid primary key default uuid_generate_v4(),
    course_id uuid not null references "courses" ("id") on delete
cascade,
    practice_name varchar(255) not null,
    description text,
    tag varchar(255) not null
);

create table if not exists "tasks"
(
    id uuid primary key default uuid_generate_v4(),
    practice_id uuid not null references "practices" ("id") on
delete cascade,
    task_name varchar(255) not null,
    body text
);

-- +goose Down
drop table if exists "tasks";
drop table if exists "practices";
drop table if exists "theories";
drop table if exists "lectures";
drop table if exists "profile_course_tags";
drop table if exists "profile_courses";
drop table if exists "courses";
drop extension if exists "uuid-ossf";

```

Файл «Dockerfile»

```

FROM golang:1.23-alpine AS builder
WORKDIR /app
COPY go.mod go.sum ./
RUN go mod download
COPY . .
RUN go build -o course-service ./main.go

FROM alpine:3.20

```

					ПГУ 1.09.03.01.50.001 ПЗ	Лист
						9
Изм.	Лист	№ докум.	Подпись	Дата		1

```

WORKDIR /app
COPY --from=builder /app/course-service .
COPY --from=builder /app/config ./config
COPY --from=builder /app/migration ./migration
ENV CONFIG_FILE=/app/config/prod.json
CMD ["./course-service"]

```

Файл «docker-compose.yml»

```

services:
  course-postgres:
    image: postgres:16
    environment:
      POSTGRES_DB: course_service
      POSTGRES_USER: postgres
      POSTGRES_PASSWORD: postgres
    ports:
      - "5433:5432"

  course-service:
    build: .
    depends_on:
      - course-postgres
    environment:
      CONFIG_FILE: /app/config/local.json
    ports:
      - "8081:8081"

```

					ПГУ 1.09.03.01.50.001 ПЗ	Лист
						9
Изм.	Лист	№ докум.	Подпись	Дата		2

Приложение В

(не обязательное)

Документы, подтверждающие участие в мероприятиях с продуктом
«ПРОПС»

					ПГУ 1.09.03.01.50.001 ПЗ	Лист
						9
Изм.	Лист	№ докум.	Подпись	Дата		3

БЛАГОДАРСТВЕННОЕ ПИСЬМО

ВРУЧАЕТСЯ КОМПАНИИ

ООО «ПРОПС»

ЗА УЧАСТИЕ В ЭКСПОЗОНЕ ФЕСТИВАЛЯ
«НОЧЬ ИННОВАЦИЙ»

4 ОКТЯБРЯ
МОСКВА 2024

					ПГУ 1.09.03.01.50.001 ПЗ	Лист
						9
Изм.	Лист	№ докум.	Подпись	Дата		4



ТЕХНОЛОГИЧЕСКИЕ
КОНКУРСЫ

СЕРТИФИКАТ

команде проекта

ПРОПС

за участие
в Технологическом конкурсе
«Медиа оборудование»



ДЕПАРТАМЕНТ
ПРЕДПРИНИМАТЕЛЬСТВА
И ИННОВАЦИОННОГО РАЗВИТИЯ
ГОРОДА МОСКВЫ



АГЕНТСТВО
ИННОВАЦИЙ
МОСКВЫ

					ПГУ 1.09.03.01.50.001 ПЗ	Лист
						9
Изм.	Лист	№ докум.	Подпись	Дата		5

Приложение Г
(обязательное)

Бизнес-проект Интеллектуальная образовательная платформа

Презентация

					ПГУ 1.09.03.01.50.001 ПЗ	Лист
						9
Изм.	Лист	№ докум.	Подпись	Дата		6

Бизнес-проект “Интеллектуальная образовательная платформа”



Докладчик:
студент группы 22ВВП2
Шнайдер Кирилл

Цель

Разработка платформы, предоставляющей пользователю возможность получать доступ к образовательным ресурсам разной направленности, в том числе к курсам с программным инструментом анализа речевых ошибок и выявления дефектов речи в записи голоса

Задачи

- Внедрение механизмов регистрации и аутентификации пользователей
- Реализация учебного процесса на базе курсов
- Управление учебными курсами
- Интеллектуальная проверка речи пользователя с использованием ИИ-ассистента
- Интеграция серверной части с системой хранения данных

					ПГУ 1.09.03.01.50.001 ПЗ	Лист
						9
Изм.	Лист	№ докум.	Подпись	Дата		7

Аналоги

Программное обеспечение	Разработчик	Достоинства	Недостатки
Coursera	Coursera Inc	Широкий выбор курсов от ведущих университетов; Наличие сертификатов и программ профессиональной переподготовки;	Ограниченные возможности практической отработки навыков; Слабая персонализация обучения;
Udemy	Udemy Inc	Разнообразие учебных программ в разных предметных областях; Умеренный уровень стоимости образовательных материалов;	Неоднородное качество курсов; ограниченная практическая составляющая; Отсутствие глубокой обратной связи;
Skillbox	ООО «Скилбокс»	Наличие практических заданий с проверкой наставниками; Ориентированность на востребованные профессии;	Высокая стоимость обучения; Длительность образовательных программ;
Stepik	ООО «Цифровые образовательные решения»	Интерактивные задания с автоматической проверкой; Поддержка адаптивного обучения;	Ограниченная работа с речевыми навыками; Узкая направленность большинства курсов;

Стек технологий

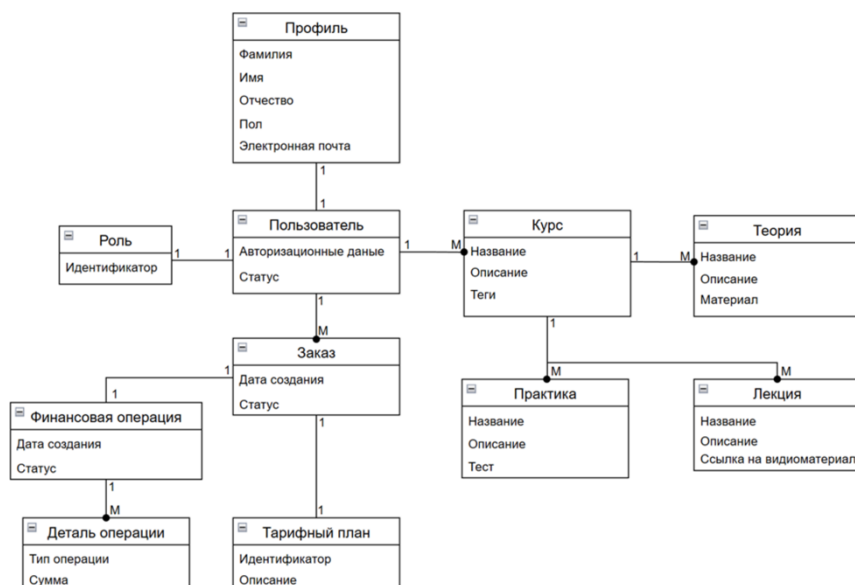


Swagger™
Supported by SMARTBEAR

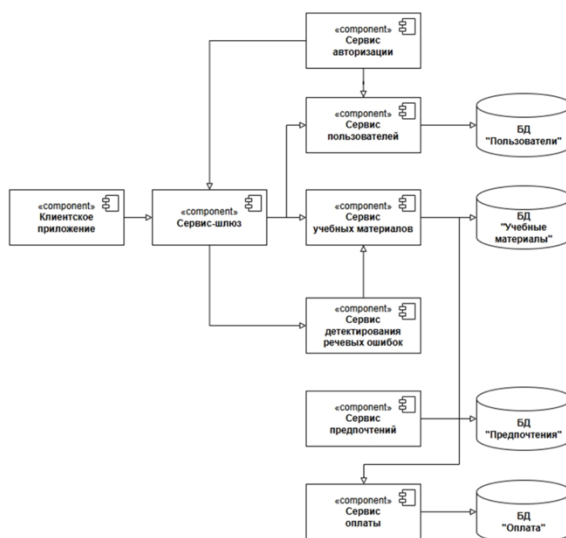


PyTorch

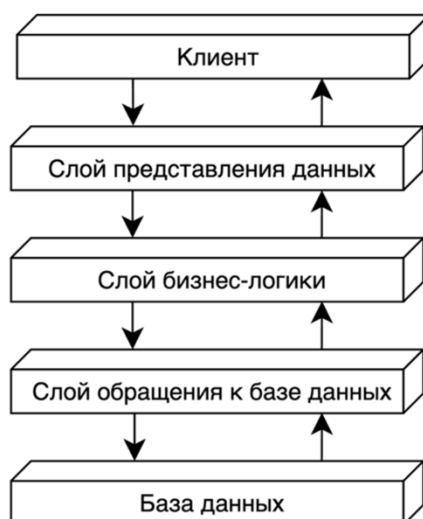
Логическая модель базы данных



Структура платформы



Архитектура микросервисов



Целевая аудитория и каналы взаимодействия

Аудитория

B2C-сегмент составляют индивидуальные пользователи, которым важно улучшить дикцию, повысить уверенность и развить навыки публичной и деловой коммуникации.

B2B-сегмент включает компании и организации, заинтересованные в развитии коммуникативных навыков сотрудников.

Каналы взаимодействия

- Контекстная и таргетированная реклама
- Контент-маркетинг и соцсети
- SEO - продвижение
- Видеомаркетинг

Финансовые показатели

Прогноз роста платных
пользователей:

- 1-й год: 1000 человек;
- 2-й год: 3000 человек;
- 3-й год: 6000 человек.

Прогнозируемая выручка
проекта составляет:

- 1-й год: 14 640 000 руб.;
- 2-й год: 43 920 000 руб.;
- 3-й год: 87 840 000 руб.

**Средний ежемесячный доход с
пользователя составит 1220 руб.**



Спасибо за внимание!

					ПГУ 1.09.03.01.50.001 ПЗ	Лист
						1
Изм.	Лист	№ докум.	Подпись	Дата		0