МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ

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

ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет	*Кафедра*
вычислительной техники	МОиПЭВМ

Направление подготовки 09.03.04 «Программная инженерия»

Направленность Программная инженерия

БАКАЛАВРСКАЯ РАБОТА

на тему

Программные средства для осознанного планирования и контроля здорового образа жизни

Студент _____ Чапаев Илья Алексеевич
(подпись, дата) (ФИО полностью)

Руководитель _____ Балашова И. Ю.
(подпись, дата) (фамилия, инициалы)

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

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

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

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

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

Пенза, 2024

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

Факультет	*Кафедра*
Вычислительной техники	МО и ПЭВМ

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

ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ
БАКАЛАВРА

1. Студент Чапаев Илья Алексеевич гр. 20ВП1 факультета ВТ
<Фамилия, имя, отчество полностью>

2.Тема работы: Программные средства для осознанного планирования и контроля здорового образа жизни

Тема утверждена приказом ПГУ №_____от «_____»_____20_____

2. Руководитель работы к.т.н., доцент кафедры «МОиПЭВМ» Балашова И.Ю.

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

1) Требуется разработать сервис, обеспечивающий планирование и контроль здорового образа жизни.
2) Среда выполнения приложения: Ubuntu 20.04, Web-браузер
3) Языковые средства разработки: C#, SQL, JavaScript, UML
4) Инструментальные средства разработки: Jet Brains Rider, Jet Brains DataGrip, Visual Paradigm 17.0
5) Технологии разработки: ООП, сервисно-ориентированный подход.
6) Основные функции приложения:
а) регистрация и авторизация;
б) учет питания с детализацией входящих нутриентов;
в) учет времени физической нагрузки;
г) учет времени сна;
д) учет формирования полезных привычек;
е) учет эмоционального состояния, посредством психологических тестов.

4. Перечень подлежащих разработке вопросов

1) Постановка задачи и анализ предметной области
2) Проектирование
3) Реализация
4) Контроль качества программного обеспечения
5) План разработки и оценка бюджета
6) Оформление пояснительной записки

5. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	Анализ предметной области и требований к системе	10%	08.04.2024 – 13.04.2024	
2	Проектирование программных средств	20%	15.04.2024 – 25.04.2024	
3	Реализация программных средств	30%	26.04.2024 – 12.05.2024	
4	Тестирование программных средств	20%	13.05.2024 – 22.05.2024	
5	Оформление пояснительной записки	20%	23.05.2024 – 03.06.2024	

Дата выдачи задания «21» декабря 2023г.

Руководитель бакалаврской работы _____
(подпись, дата)

Задание к исполнению принял студент _____
(подпись, дата)

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

Реферат

Пояснительная записка содержит 111 листов, 50 рисунков, 17 таблиц, 26 использованных источников, 8 приложений.

ПЛАНИРОВАНИЕ, КОНТРОЛЬ, ЗДОРОВЫЙ ОБРАЗ ЖИЗНИ, REST, MVC, C#, UML, СЕРВЕР БАЗЫ ДАННЫХ, POSTGRESQL, DOCKER.

Объект исследования – средства автоматизации планирования и контроля здорового образа жизни.

Цель выпускной квалификационной работы – разработать сервис для учета ведения здорового образа жизни.

Технологии разработки – разработка сервера приложений проводилась на языке C# с использованием фреймворка ASP.NET, для создания сервера БД использовался PostgreSQL, клиентская часть разрабатывалась на языке JavaScript с использованием фреймворка React.js. В качестве среды разработки использовались интегрированные среды разработки JetBrains Rider, WebStorm и менеджер баз данных DataGrip.

Среда исполнения – Linux.

Результат работы – в ходе выполнения выпускной квалификационной работы были созданы клиентская и серверная части сервиса планирования и контроля здорового образа жизни. Все требования к разработке приложения соблюдены.

Содержание

Введение 7
1 Анализ программных средств для осознанного планирования и контроля здорового образа жизни 9
1.1 Анализ предметной области и постановка задачи 9
1.2 Анализ программного обеспечения аналогичного назначения 10
1.2.1 Приложение «MyFitnessPal» 10
1.2.2 Приложение «Sleep cycle» 11
1.2.3 Приложение «Lifesum» 12
1.2.4 Приложение «Welltory» 13
1.2.5 Приложение «Samsung Health» 14
1.2.6 Сравнение программных средств для осознанного планирования и контроля здорового образа жизни 15
1.3 Анализ функциональных требований 17
1.4 Анализ нефункциональных требований 20
2 Проектирование программного средства для осознанного планирования и контроля здорового образа жизни 22
2.1 Архитектура приложения 22
2.2 Проектирование структур данных 24
2.2.1 Концептуальная модель базы данных 24
2.2.2 Логическая модель базы данных 26
2.3 Проектирование пользовательского интерфейса 29
2.4 Проектирование программных средств 35
3 Реализация программного средства для осознанного планирования и контроля здорового образа жизни 37
3.1 Реализация базы данных 37
3.2 Реализация программных компонентов 39
4 Контроль качества программного обеспечения 46
4.1 Выбор режима тестирования 46
4.2 Тестирование серверной части веб-приложения 47

4.3 Тестирование клиентской части веб-приложения 48
4.4 Метрики устойчивости и абстрактности компонент 49
4.5 Метрики кода 51
4.6 Планирование разработки и оценка бюджета 51
Заключение 53
Список использованных источников 54
Приложение А (справочное) Глоссарий 57
Приложение Б (рекомендуемое) Атрибуты сущностей базы данных 60
Приложение В рекомендуемое) Программные компоненты серверной части .. 67
Приложение Г (рекомендуемое) Графическая часть интерфейса программных средств 69
Приложение Д (рекомендуемое) Графическая часть оценки стоимости проекта 72
Приложение Е (рекомендуемое) Результаты тестирования клиентской части приложения 75
Приложение Ж (рекомендуемое) Листинг серверной серверной части программных средств 78
Приложение И (рекомендуемое) Листинг клиентской части программных средств 96

Введение

В современном мире, с его ускоренным темпом жизни, поддержание здорового образа жизни (ЗОЖ) становится все более актуальной задачей. Люди все больше стремятся к осознанному планированию и контролю своего здорового образа жизни, что включает учет сна, питания, физической активности, психологического состояния и полезных привычек. Эффективное управление этими аспектами требует современных программных средств, которые не только фиксируют и анализируют данные, но и предоставляют рекомендации для улучшения образа жизни [1].

До появления специализированных программ поддержание ЗОЖ требовало значительных усилий и времени. Людям приходилось вести ручные записи и использовать разрозненные инструменты для отслеживания различных аспектов здорового образа жизни, что было неудобно и неэффективно. Например, учет питания мог включать ведение дневников, ручной расчет калорий или использование отдельных приложений для отслеживания макронутриентов. Физическая активность фиксировалась отдельно, что не давало целостного представления о прогрессе [2].

С развитием технологий ситуация значительно изменилась. Теперь доступны интегрированные системы для осознанного планирования и контроля ЗОЖ, которые учитывают сон, питание, физическую активность, психологическое состояние и полезные привычки. Такие системы позволяют пользователям составлять меню, отслеживать потребление нутриентов, планировать физическую активность и проводить тесты на психологическое состояние в одном месте. Это значительно упрощает процесс управления здоровым образом жизни и делает его более удобным и эффективным.

Сегодня многие приложения и системы предлагают функции учета и анализа различных аспектов ЗОЖ, предоставляя пользователям рекомендации на основе собранных данных. Это позволяет не только отслеживать свои показатели, но и получать персонализированные советы, которые помогают улучшить образ жизни.

Однако существующие решения имеют свои ограничения. Например, многие из них не обеспечивают комплексного подхода, объединяющего все аспекты ЗОЖ в одном приложении. Некоторые приложения не поддерживают полноценный учет и анализ эмоционального состояния или привычек пользователя, что снижает их эффективность [3].

Таким образом, существует необходимость в создании нового приложения для ЗОЖ, которое объединяет все ключевые функции в одном месте и предоставляет пользователям более целостное и персонализированное решение. Разрабатываемое в рамках данной дипломной работы приложение направлено на удовлетворение этой потребности, предлагая комплексный подход к осознанному планированию и контролю здорового образа жизни.

Целью выпускной квалификационной работы является разработка программных средств для осознанного планирования и контроля здорового образа жизни. Для достижения поставленной цели необходимо решить следующие задачи:

– провести сравнительный анализ существующих программных продуктов для поддержки ЗОЖ;

– провести анализ требований к разрабатываемому программному продукту;

– осуществить проектирование программных средств и структур данных разрабатываемого программного продукта;

– провести планирование бюджета на разработку программных средств.

Кроме того, в ходе выполнения выпускной квалификационной работы также требуется оформить пояснительную записку в соответствии с требованиями.

1 Анализ программных средств для осознанного планирования и контроля здорового образа жизни

1.1 Анализ предметной области и постановка задачи

Здоровый образ жизни (ЗОЖ) включает в себя несколько ключевых аспектов: сбалансированное питание, достаточный сон, регулярную физическую активность, формирование полезных привычек и поддержание эмоционального благополучия.

Для сбалансированного питания важно поддерживать в норме количество потребляемых нутриентов, включая белки, жиры и углеводы. Достаточный сон требует мониторинга продолжительности и качества сна для поддержания оптимального физического и психического состояния. Регулярная физическая активность предполагает планирование и выполнение физических упражнений, направленных на улучшение общей физической формы и здорового образа жизни. Формирование полезных привычек включает отслеживание регулярности выполнения действий, которые улучшат качество жизни. Поддержание эмоционального благополучия требует регулярной оценки психологического состояния и применения стратегий для управления стрессом и эмоциональными нагрузками [2].

Для осознанного управления этими аспектами требуется комплексный подход, который может быть реализован с помощью специализированных программных средств. Для определения требований к разрабатываемому приложению необходимо провести анализ предметной области и выявить наиболее вероятные варианты использования.

Исходя из всего описанного выше, целью является создание программного продукта на основе архитектуры, состоящей из серверного приложения, базы данных и клиентского веб-приложения, который охватывает:

– планирование рациона питания и учет потребленных продуктов, с детализацией нутриентов;

– учет времени и качества сна;

– учет времени физической активности;

– формирование и контроль полезных привычек;

– анализ эмоционального состояния на основе данных психологических тестов;

– аналитику учитываемых компонентов ЗОЖ посредством графиков и диаграмм.

Стоит также отметить, что после окончания основного процесса разработки необходимо осуществлять поддержку функционала приложения и, при необходимости, внедрять новые возможности для расширения функционала приложения.

1.2 Анализ программного обеспечения аналогичного назначения

1.2.1 Приложение «MyFitnessPal»

Приложение «MyFitnessPal» является одним из самых популярных инструментов для отслеживания питания и физической активности. Интерфейс приложения представлен на рисунке 1.

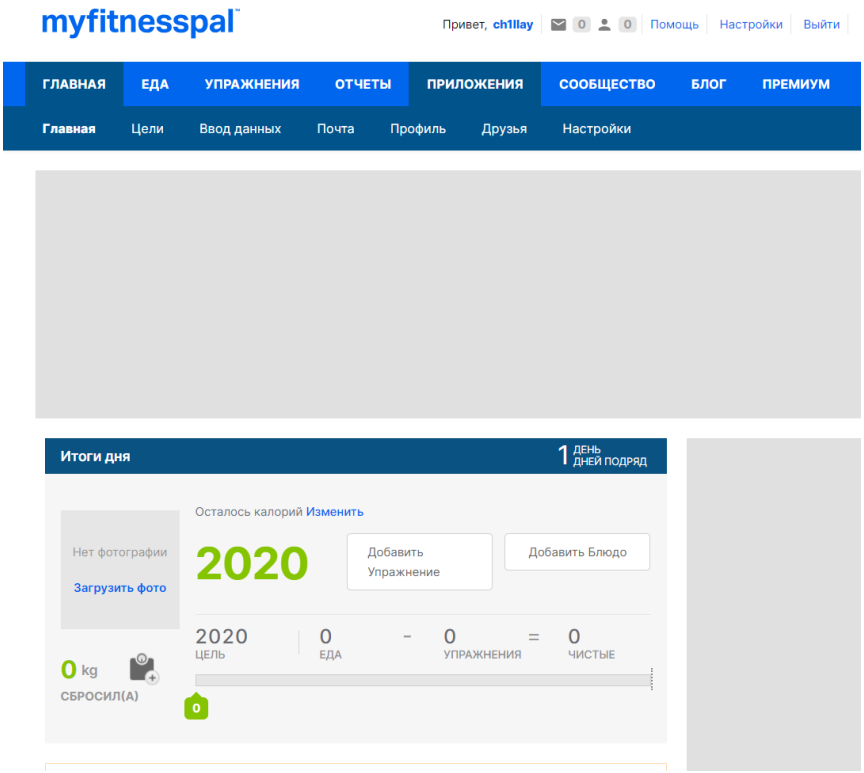


Рисунок 1 – Интерфейс приложения «MyFitnessPal»

Среди его достоинств можно выделить огромную базу данных продуктов питания (более 14 миллионов), детальное отслеживание калорийности и макронутриентов, инструменты для учета различных видов упражнений, возможность постановки целей и социальную поддержку через взаимодействие с другими пользователями. Приложение доступно как на смартфонах, так и на веб-сайте. Однако, приложение не обеспечивает полноценного отслеживания сна, оценки эмоционального состояния и формирования полезных привычек. Кроме того, многие функции доступны только по подписке, что ограничивает возможности бесплатной версии [4].

1.2.2 Приложение «Sleep cycle»

Приложение «Sleep Cycle» специализируется на мониторинге сна и его качества. Интерфейс приложения представлен на рисунке 2.

Рисунок 2 – Интерфейс приложения «Sleep cycle»

Приложение отслеживает фазы сна, анализирует храп, предоставляет графики и статистику по качеству сна, и имеет интеллектуальный будильник. Однако, Sleep Cycle не отслеживает питание, физическую активность, эмоциональное состояние и не помогает в формировании полезных привычек [5].

1.2.3 Приложение «Lifesum»

Приложение «Lifesum» акцентирует внимание на здоровом питании и предоставляет пользователям рецепты и планы питания. Интерфейс приложения представлен на рисунке 3.

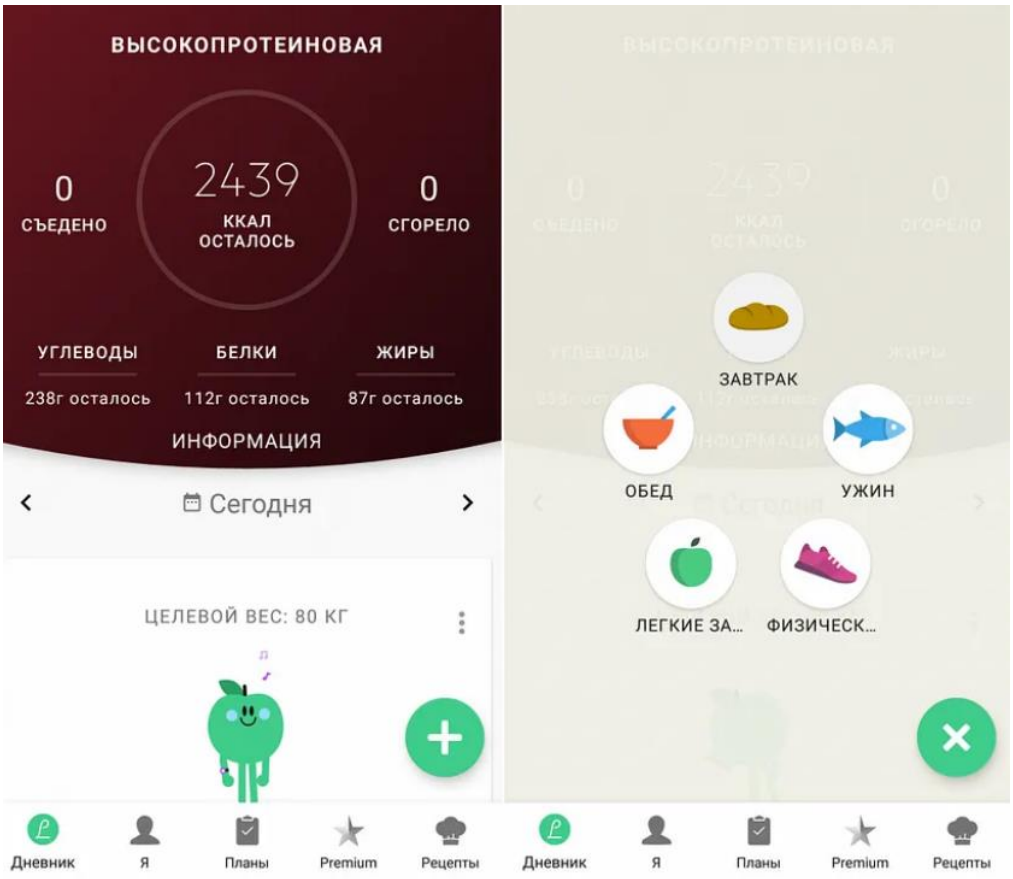


Рисунок 3 – Интерфейс приложения «Lifesum»

Преимущества включают акцент на персонализированные рекомендации и анализ пищевых привычек. Однако, приложение не обеспечивает отслеживания сна, физической активности и оценки эмоционального состояния. Много полезных функций доступны только по подписке [6].

1.2.4 Приложение «Welltory»

Приложение «Welltory» предлагает персонализированный подход к здоровью и фитнесу, используя искусственный интеллект для анализа данных. Интерфейс приложения представлен на рисунке 4.

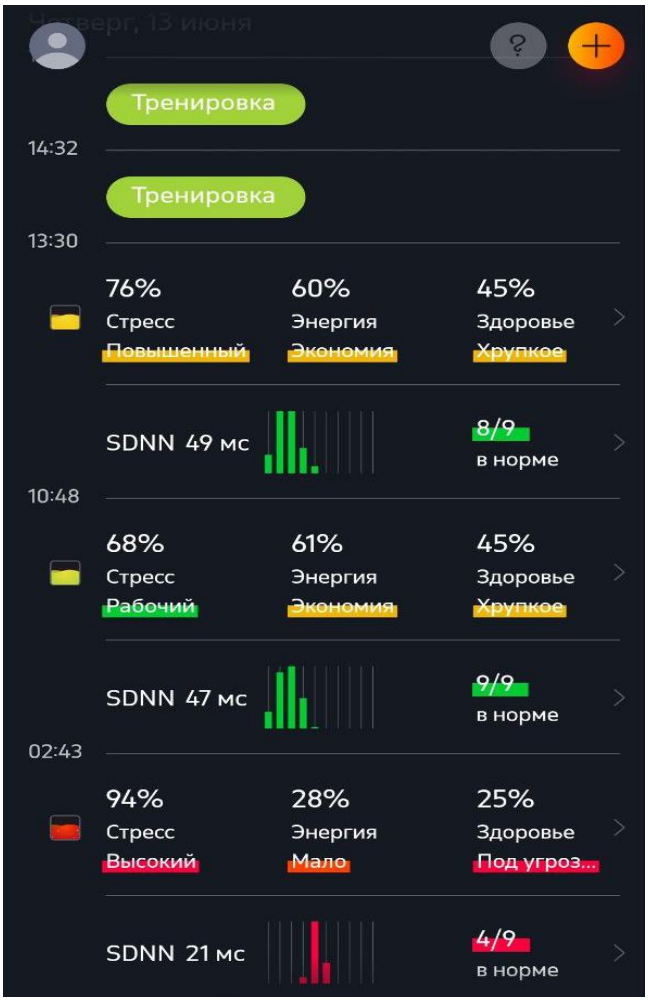


Рисунок 4 – Интерфейс приложения «Welltory»

Среди его преимуществ можно выделить персонализированный подход, мотивацию и поддержку на пути к цели, а также сочетание фитнеса, питания и ментального здорового образа жизни. Приложение доступно на смартфонах. Однако, приложение является относительно новым и имеет меньшее количество пользователей по сравнению с более известными приложениями. Некоторые функции доступны только по подписке, и интерфейс может показаться сложным для начинающих [7].

1.2.5 Приложение «Samsung Health»

«Samsung Health» – это бесплатное приложение для мониторинга физической активности от компании Samsung. Интерфейс приложения изображен на рисунке 5.

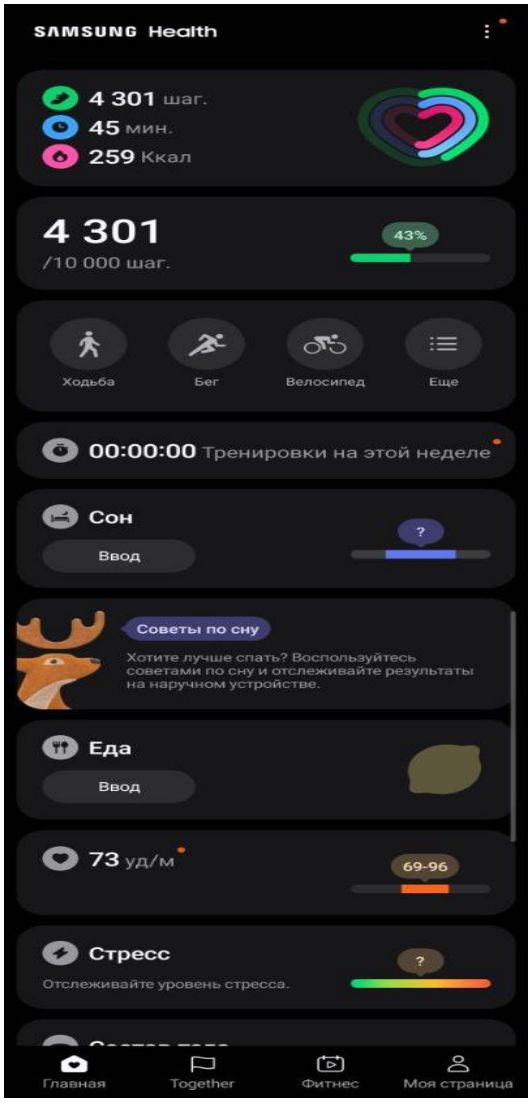


Рисунок 5 – Интерфейс приложения «Samsung Health»

Среди преимуществ можно выделить персонализированный подход, интеграцию с устройствами Samsung и некоторыми сторонними гаджетами, возможность отслеживания шагов, мониторинга калорий и различных видов активности, таких как бег, велосипедный спорт и плавание. Приложение также поддерживает контроль питания и помогает поддерживать ментальное здоровье через соответствующие функции.

К недостаткам можно отнести ограниченную совместимость с устройствами других производителей, что может ограничить его использование для ряда пользователей [8].

1.2.6 Сравнение программных средств для осознанного планирования и контроля здорового образа жизни

Проведем сравнительный анализ рассмотренных выше программных средств. Для оценки приложений будет использоваться трехбалльная шкала оценки, в которой 0 означает «полное отсутствие реализации», 1 – «частичную реализацию», а 2 – «полную реализацию» рассматриваемой функции. Результаты сравнения программных средств для осознанного планирования и контроля здорового образа жизни представлены в таблице 1.

Таблица 1 – Сравнение программных средств аналогичного назначения

Критерий \ Приложение	MyFitnessPal	Sleep cycle	Lifesum	Welltory	Samsung Health
Учет питания	2	0	2	1	2
Отслеживание физической активности	2	0	0	1	2
Учет продолжительности сна	1	2	0	0	2
Оценка эмоционального состояния	0	0	0	2	2
Формирование полезных привычек	1	1	0	2	0
Интеграция с другими сервисами	2	2	0	1	1
Возможность зарегистрироваться и авторизоваться	2	2	2	2	2
ИТОГО	10	7	4	9	11

Подводя итог, можно сделать вывод, что ни одно из рассмотренных приложений не может в полной мере выполнить поставленную задачу.

Лидером среди рассмотренных приложений является «Samsung Health». Оно охватывает широкий спектр функций, включая отслеживание питания,

физической активности, сна и эмоционального состояния, а также интеграцию с устройствами Samsung и некоторыми сторонними гаджетами. Приложение предоставляет пользователям возможность комплексно следить за своим здоровьем и формированием полезных привычек. Однако ограниченная совместимость с устройствами других производителей может затруднить его использование для некоторых пользователей.

На втором месте находится «MyFitnessPal». Это приложение отлично отслеживает питание и физическую активность, предлагая огромную базу данных продуктов и детальное отслеживание калорийности и макронутриентов. Однако, не поддерживает функции отслеживания сна, оценки эмоционального состояния и формирования полезных привычек. Многие функции доступны только по подписке, что ограничивает возможности бесплатной версии.

Приложение «Welltory» занимает третье место благодаря персонализированному подходу, включающему функции отслеживания физической активности, сна, эмоционального состояния и формирования полезных привычек. Однако, оно не реализует функцию отслеживания питания, и многие его возможности доступны только по подписке, что может ограничить доступ для некоторых пользователей.

«Sleep Cycle» качественно отслеживает сон, анализируя фазы, храп и качество сна, и предоставляет графики и статистику. Это приложение является отличным инструментом для улучшения качества сна, но не поддерживает отслеживание питания, физической активности, эмоционального состояния и формирования привычек. Для комплексного подхода к здоровому образу жизни требуется дополнение другими приложениями.

«Lifesum» в наименьшей степени реализует поставленную задачу. Приложение акцентирует внимание на здоровом питании, предоставляя пользователям рецепты и планы питания, а также персонализированные рекомендации. Однако оно не поддерживает отслеживание сна, физической активности, эмоционального состояния и формирования привычек. Много полезных функций доступны только по подписке.

В результате принято решение разработать собственное программное средство, которое будет включать в себя лучшие решения от рассмотренных приложений и будет лишено наиболее существенных недостатков.

1.3 Анализ функциональных требований

После проведения исследования по существующим приложениям для осознанного планирования и контроля здорового образа жизни, можно выделить функциональные требования:

– обеспечение регистрации;
– обеспечение авторизации;
– внесение настроек физической активности;
– запись данных о физической активности;
– внесение настроек сна;
– запись данных о сне;
– запись данных о съеденном блюде и просмотр учета питания;
– добавление и просмотр учета привычек;
– прохождение психологического теста и просмотр учета эмоционального состояния;
– просмотр аналитики питанию;
– просмотр аналитики сну;
– просмотр аналитики физической активности;
– просмотр аналитики привычек;
– просмотр аналитики эмоционального состояния.

В соответствии с выявленными требованиями была разработана диаграмма вариантов использования UML, представленная на рисунке 6.

Стоит отметить, что на диаграмме вариантов использования используется два типа отношений: отношения ассоциации и отношения включения [9]. Первый тип служит для обозначения связи между актером и отдельным вариантом использования, отображенным на диаграмме.

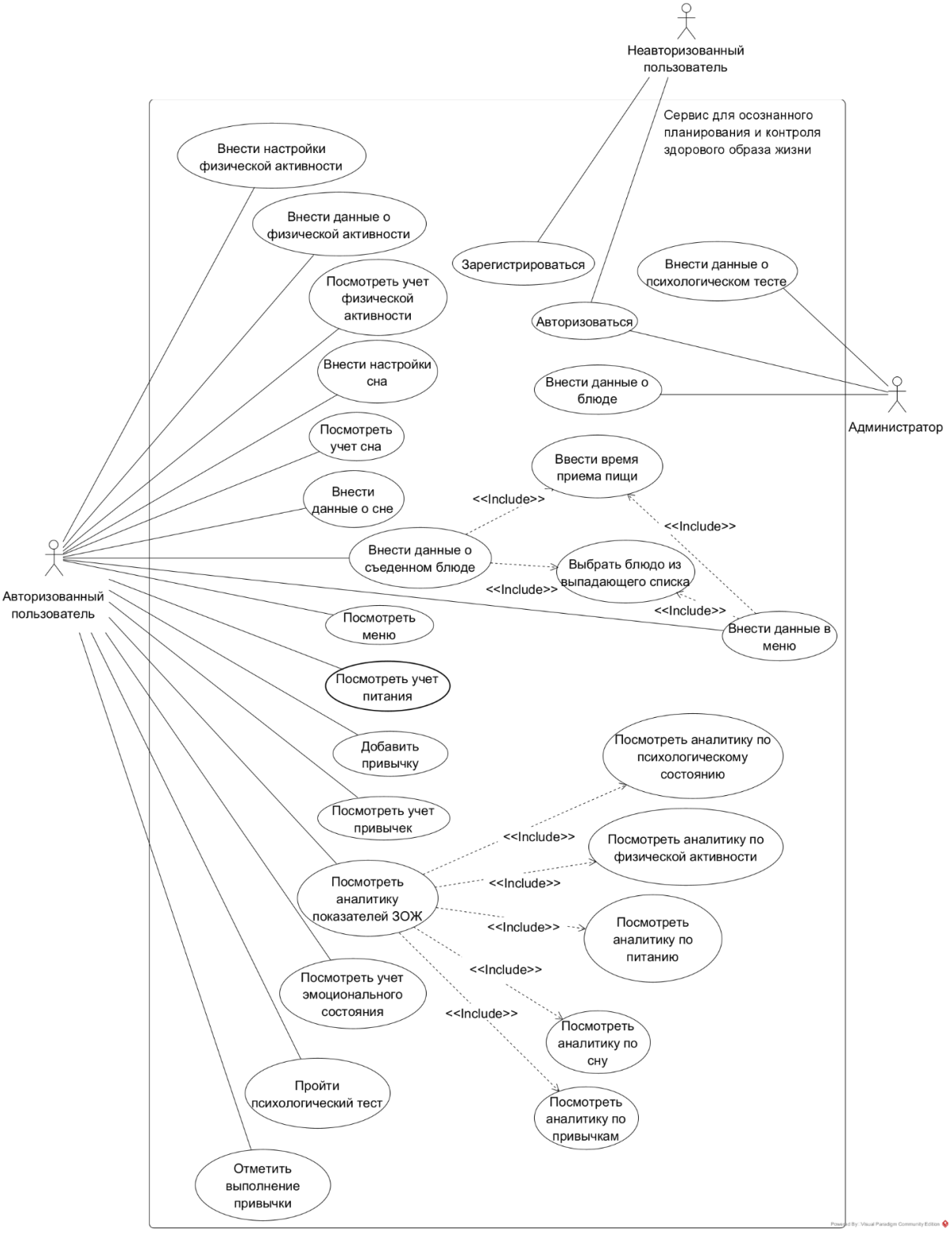


Рисунок 6 – Диаграмма вариантов использования

Сценарий варианта использования «Посмотреть учет питания» представлен на рисунке 7.

Прецендент: Посмотреть учет питания
Id:1
Краткое описание: Пользователь просматривает учет съеденных продуктов с детализацией по нутриентам
Главные актеры: Авторизованный пользователь
Второстепенные актеры: Нет
Предусловие: 1. Пользователь открыл приложение 2. Пользователь авторизовался в системе
Основной поток: 1. Пользователь открыл вкладку «Питание» 2. Пользователь выбрал период отображения рациона
Постусловия: 1. Система отображает на странице учет съеденных продуктов с детализацией по нутриентам
Альтернативные потоки: Нет

Рисунок 7 – Сценарий варианта использования «Посмотреть учет питания»

На рисунке 8 представлена спецификация прецедента «Посмотреть учет привычек».

Прецендент: Посмотреть учет привычек
Id:2
Краткое описание: Пользователь просматривает регулярность выполнения привычек
Главные актеры: Авторизованный пользователь
Второстепенные актеры: Нет
Предусловие: 1. Пользователь открыл приложение 2. Пользователь авторизовался в системе
Основной поток: 1. Пользователь открыл вкладку привычек 2. Пользователь выбрал привычку
Постусловия: 1. Система отображает на странице учет регулярности выполнения привычки
Альтернативные потоки: Нет

Рисунок 8 – Сценарий варианта использования «Посмотреть учет привычек»

Сценарий варианта использования «Пройти психологический тест» представлен на рисунке 9.

Прецендент: Пройти психологический тест
Id:3
Краткое описание: Пользователь проходит психологический тест
Главные актеры: Авторизованный пользователь
Второстепенные актеры: Нет
Предусловие: 1. Пользователь открыл приложение 2. Пользователь авторизовался в системе
Основной поток: 1. Пользователь открыл вкладку с психологическими тестами 2. Пользователь выбрал психологический тест 3. Пользователь нажал на кнопку «Пройти тест» 4. Пользователь отвечает на вопросы теста 5. Пользователь просматривает результат оценки теста
Постусловия: 1. Система отображает на странице список психологических тестов 2. Система отображает на странице вопросы психологического теста с вариантами ответов 3. Система отображает на странице результаты прохождения теста
Альтернативные потоки: Нет

Рисунок 9 – Сценарий варианта использования «Пройти психологический тест»

В результате анализа функциональных требований были выделены варианты использования программы. Для основных вариантов использования были представлены их спецификации, которые позволяют отобразить последовательность действий пользователя и системы.

1.4 Анализ нефункциональных требований

Разработка базы данных будет проводиться с использованием PostgreSQL. База данных будет размешена на удаленном сервере. Клиентская часть приложения будет написана на языке JavaScript с использованием библиотеки React. Серверная часть приложения будет написана на языке C# с использованием фреймворка ASP .NET [10].

React представляет собой библиотеку для разработки Web-приложений по принципу SPA (Single Page Application) – при изменении контекста новая страница не создается, а лишь перерисовывается под новые компоненты. ASP .NET представляет собой платформу с открытым исходным кодом. Она предназначена для построения серверных приложений для различных операционных систем с поддержкой среды выполнения .NET. Платформа ASP .NET представляет собой уровень абстракции, обеспечивающий возможность управления управление отправкой запроса посредством паттерна MVC [11].

Клиентское приложение должно быть отлажено и успешно работать в браузере Google Chrome версии 124.0 и выше. Серверное приложение должно быть отлажено, развернуто на удаленном сервере.

2 Проектирование программного средства для осознанного планирования и контроля здорового образа жизни

2.1 Архитектура приложения

При разработке веб-приложения использовалась многоуровневая архитектура с использованием микросервисного подхода. Схема архитектуры представлена на рисунке 10.

.

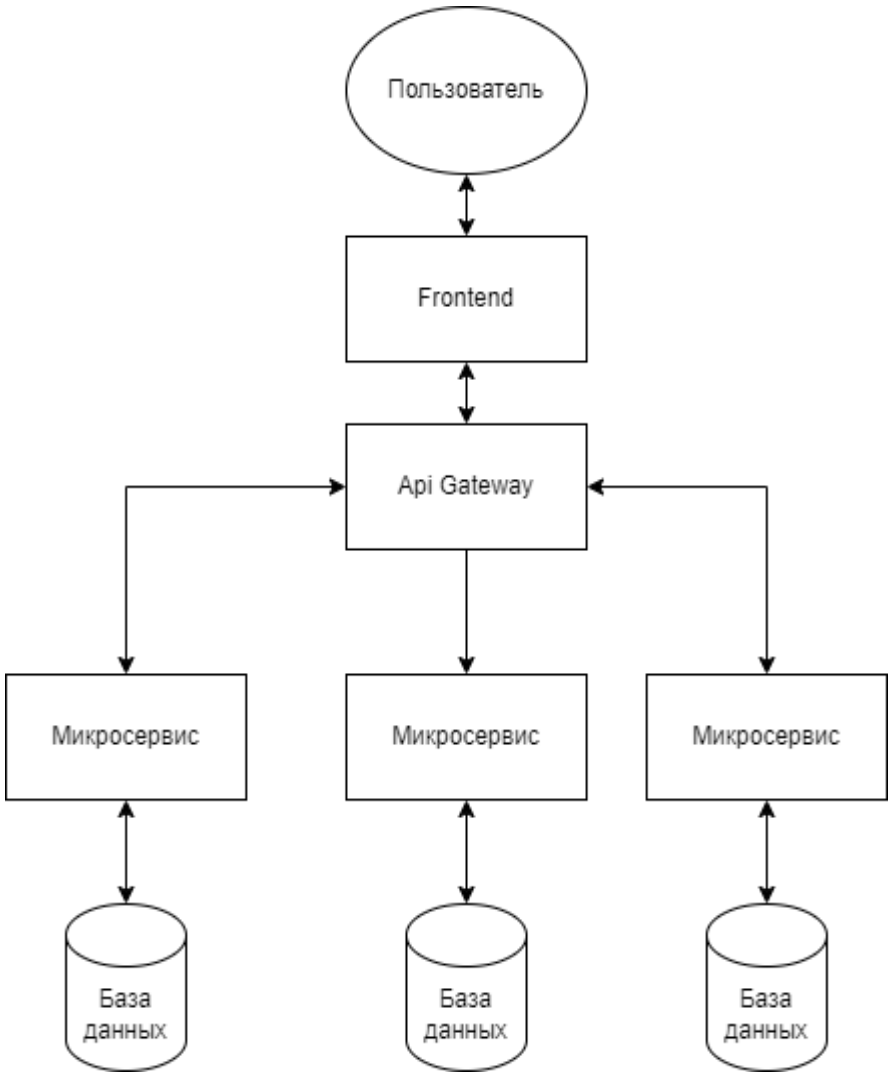


Рисунок 10 – Крупноблочная схема архитектуры

В этой архитектуре выделяются следующие основные слои: слой представления, слой промежуточного уровня, слой бизнес-логики и слой данных.

Слой представления отвечает за взаимодействие с пользователем. Он охватывает все аспекты пользовательского интерфейса, принимая запросы от пользователей и передавая их на следующий уровень – API Gateway. Этот слой также отвечает за отображение данных и обеспечение удобного взаимодействия пользователя с приложением.

API Gateway служит промежуточным слоем между слоем представления и микросервисами. Он принимает запросы от клиентской части, маршрутизирует их к соответствующим микросервисам и возвращает ответы обратно на слой представления. API Gateway обеспечивает централизованное управление трафиком, балансировку нагрузки и контроль доступа. Это позволяет упростить взаимодействие клиентской части с множеством микросервисов, скрывая сложность внутренней системы и предоставляя единую точку входа для всех запросов [12].

Слой бизнес-логики состоит из множества микросервисов, каждый из которых выполняет специфические задачи и работает независимо от других. Микросервисы реализуют основную бизнес-логику приложения, обрабатывают команды, поступающие от слоя представления, и взаимодействуют с базами данных. Этот слой отвечает за выполнение вычислений на основе вводимых и хранимых данных, обработку команд от слоя представления и передачу информации в слой данных.

Слой данных представлен базами данных, которые взаимодействуют с микросервисами. Каждая база данных обеспечивает хранение и управление данными для своего микросервиса. Использование отдельных баз данных позволяет избежать проблем, связанных с единой точкой отказа, и улучшает производительность и масштабируемость системы. Это также позволяет использовать наиболее подходящие технологии хранения данных для каждой конкретной задачи.

Взаимодействие между слоями представления и бизнес-логики осуществляется с помощью REST API по протоколу HTTP.

REST – это архитектурный стиль, использующий стандартные HTTP-методы (GET, POST, PUT, DELETE) для взаимодействия между компонентами системы. REST API предоставляет простой и понятный способ взаимодействия между клиентом и сервером, позволяя обмениваться данными в формате JSON или XML.

API Gateway – это сервер, который является единой точкой входа в систему и принимает все входящие запросы от клиентов, маршрутизируя их к соответствующим микросервисам. API Gateway управляет аутентификацией, авторизацией, балансировкой нагрузки, мониторингом и другими задачами.

Многоуровневая архитектура с использованием микросервисного подхода обладает рядом преимуществ. Масштабируемость позволяет масштабировать каждый микросервис независимо, что эффективно управляет ресурсами и нагрузкой. Гибкость обеспечивает возможность разрабатывать, развертывать и обновлять микросервисы независимо друг от друга. Изоляция данных упрощает управление данными и уменьшает риск влияния одной службы на другую. Надежность и отказоустойчивость позволяют системе продолжать функционировать даже в случае сбоя одного из микросервисов, минимизируя влияние на общую систему [13].

2.2 Проектирование структур данных

2.2.1 Концептуальная модель базы данных

На основе анализа предметной области и результатов моделирования бизнес-процессов была составлена концептуальная модель базы данных в виде диаграммы «сущность-связь», представленной на рисунке 11.

Концептуальная модель представляет собой первоначальное структурирование информации в предметной области без учета деталей возможной реализации базы данных.

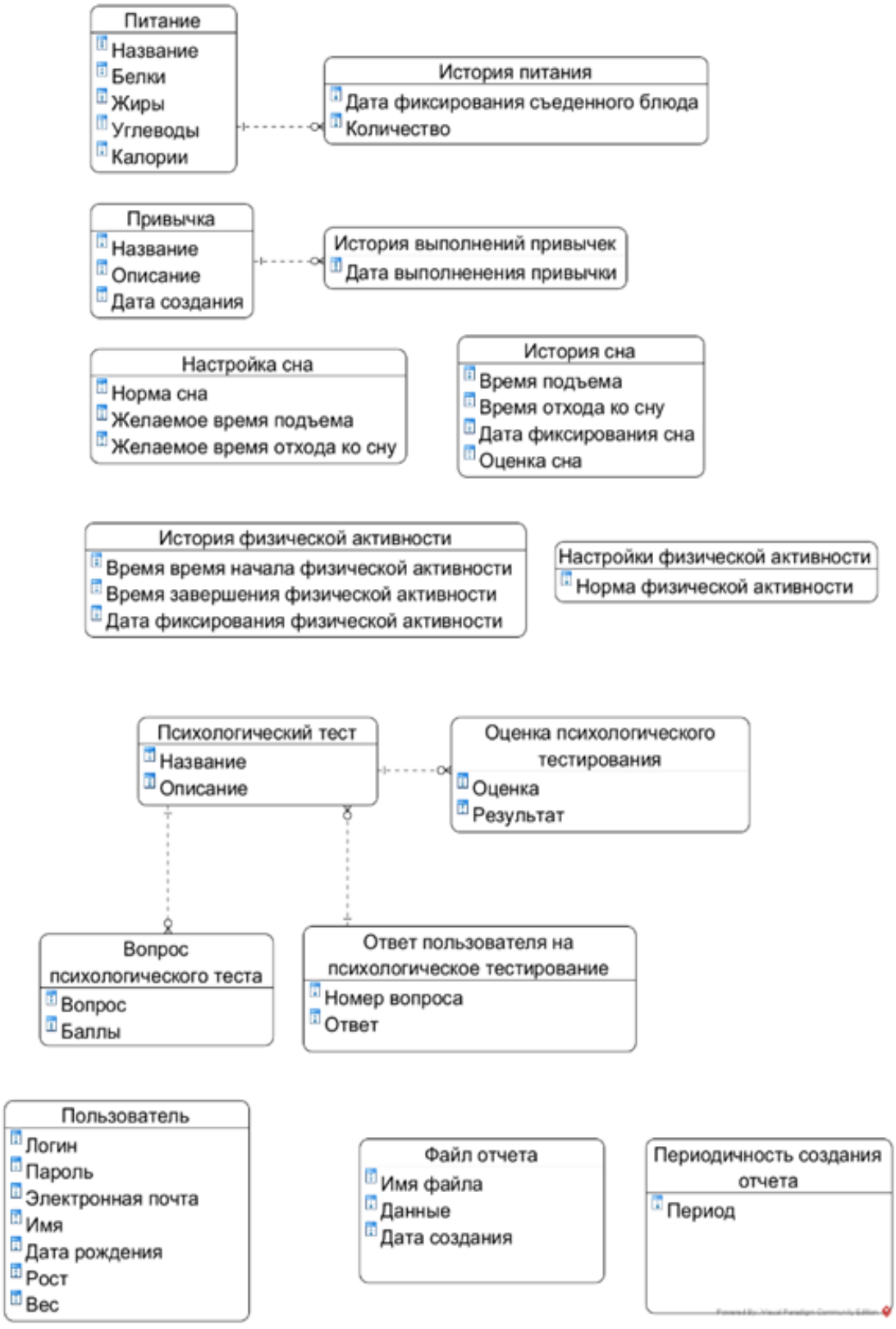


Рисунок 11 – Графическое представление концептуальной модели базы данных

На основе концептуальной модели базы данных было составлено описание сущностей, приведенное в таблице 2.

Таблица 2 – Сущности БД

Название сущности	Идентификатор сущности	Назначение сущности
Пользователь	User	Содержит сведения о пользователях приложения
Питание	FoodProduct	Содержит сведения о продуктах питания и их нутриентах
История питания	FoodProductHistory	Содержит сведения об учете продуктов питания пользователя
Привычка	Habit	Содержит сведения о привычках пользователя
История выполнения привычек	HabitHistory	Содержит данные о регулярности выполнения привычек
Настройки сна	SleepSettings	Содержит сведения о настройках сна пользователя
История сна	SleepHistory	Содержит записи о сне пользователя
Настройка физической активности	PhysicalActivitySettings	Содержит сведения о настройках физической активности
История физической активности	PhysicalActivityHistory	Содержит записи о физической активности пользователя
Психологический тест	PsychologicalTest	Содержит сведения о психологических тестах
Вопрос психологического теста	PsychologicalTestQuestion	Содержит сведения о вопросах психологического теста
Оценка психологического тестирования	PsychologicalTestEvolution	Содержит сведения об оценке психологического тестирования
Ответ пользователя на психологическое тестирование	UserPsychologicalTestResult	Содержит записи об ответах пользователя на психологический тест

Концептуальная модель базы данных построена в среде Visual Paradigm 17.0 в нотации IDEF1X.

2.2.2 Логическая модель базы данных

Логическая модель – это расширение концептуальной модели данных. Она включает в себя все сущности, атрибуты, ключи и взаимосвязи, которые представляют бизнес-информацию и определяют бизнес-правила [14].

На основании концептуальной модели была сконструирована логическая модель баз данных, разделенная по предметным областям. Логическая модель для базы данных «Авторизация» представлена на рисунке 12.

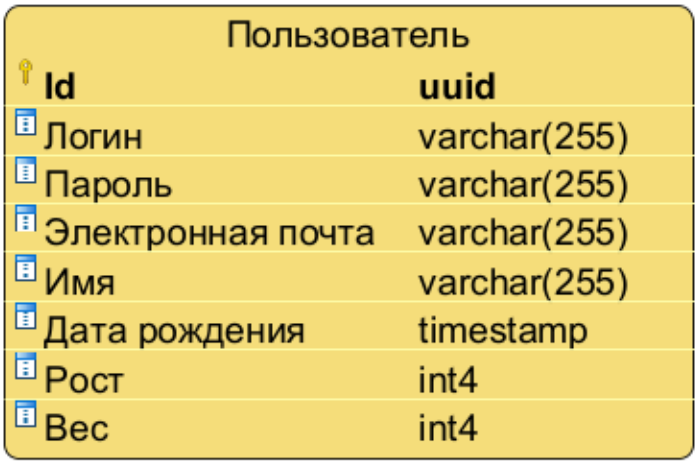


Рисунок 12 – Графическое представление логической модели базы данных «Авторизация»

Логическая модель для базы данных «Отчетность» представлена на рисунке 13.

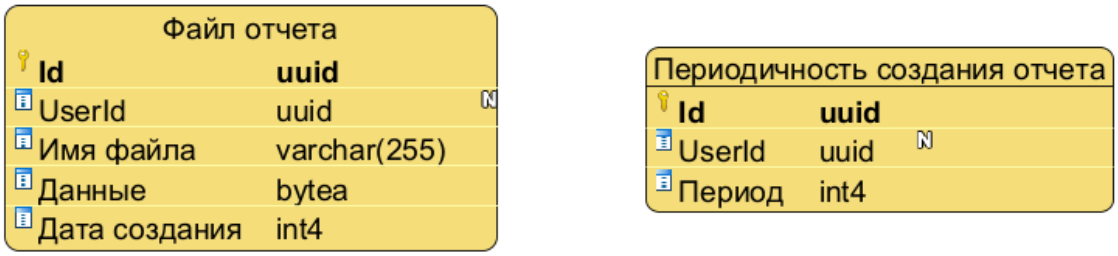


Рисунок 13 – Графическое представление логической модели базы данных «Отчетность»

Логическая модель для базы данных «Здоровье» представлена на рисунке 14.

На логической модели базы данных «Здоровье» присутствуют связи «многие-ко-многим», «один-ко-многим», которые описывают возможное взаимодействие записей, которые в будущем будут храниться в базе данных. Так, например, один психологический тест может иметь множество оценок.

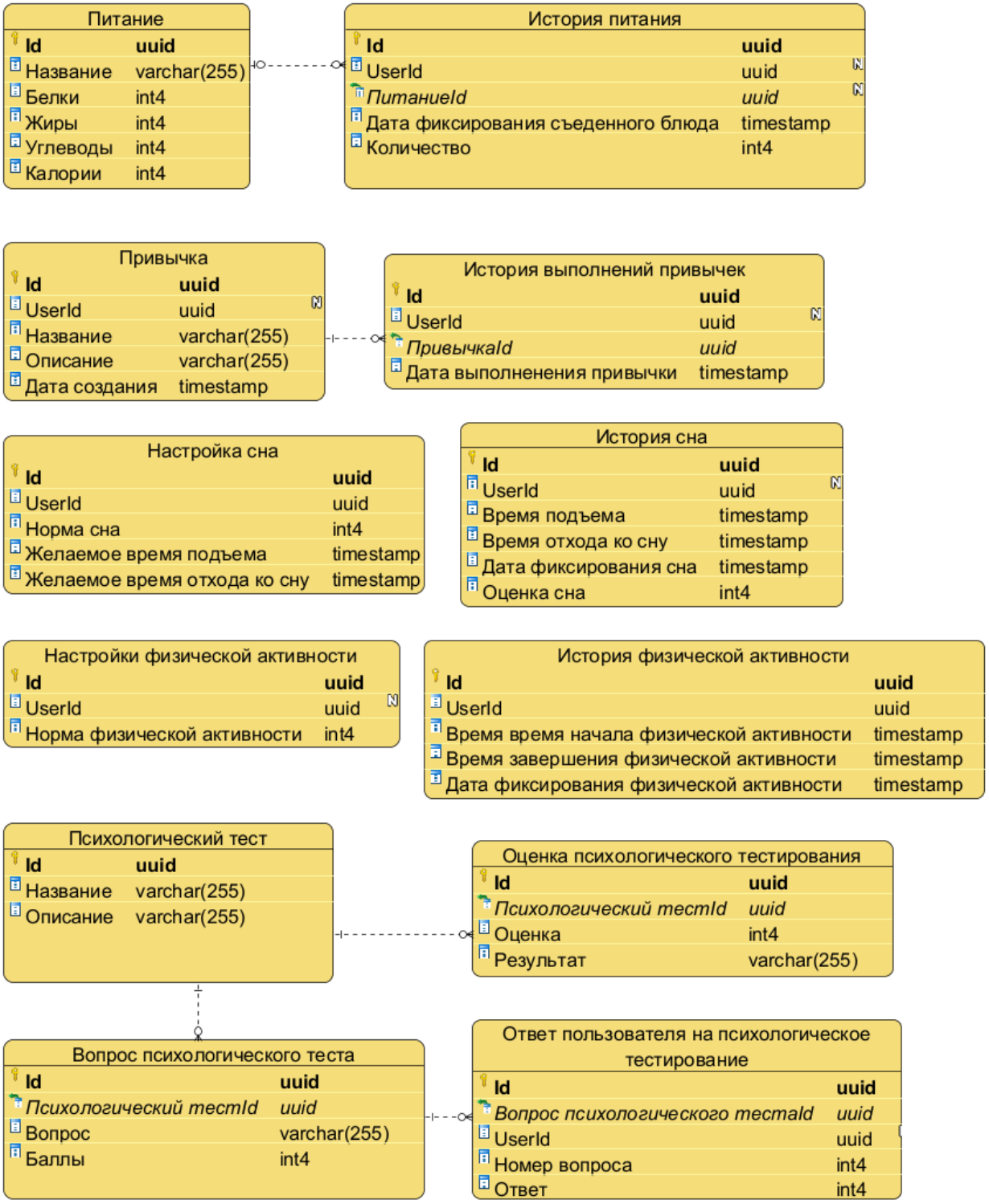


Рисунок 14 – Графическое представление логической модели базы данных «Здоровье»

Логическая модель построена в среде Visual Paradigm 17.0 в нотации IDEF1X.

2.3 Проектирование пользовательского интерфейса

На основе диаграммы вариантов использования для клиентского приложения был спроектирован графический интерфейс. Пользовательский интерфейс должен обеспечивать простое и эффективное выполнение основных функций пользователей.

Пользовательское приложение представляет собой SPA-приложение (Single page application). SPA – это реализация веб-приложения, которое загружает только один веб-документ, а затем обновляет содержимое тела этого единственного документа через API-интерфейсы JavaScript. Такой подход обеспечивает более быструю работу приложения, так как перезагрузка страниц не требуется, и пользователь получает более плавный и интерактивный опыт. Основные преимущества SPA включают в себя уменьшение времени загрузки страниц, повышение производительности и улучшение взаимодействия с пользователем [15]. На рисунке 15 представлена страница регистрации.

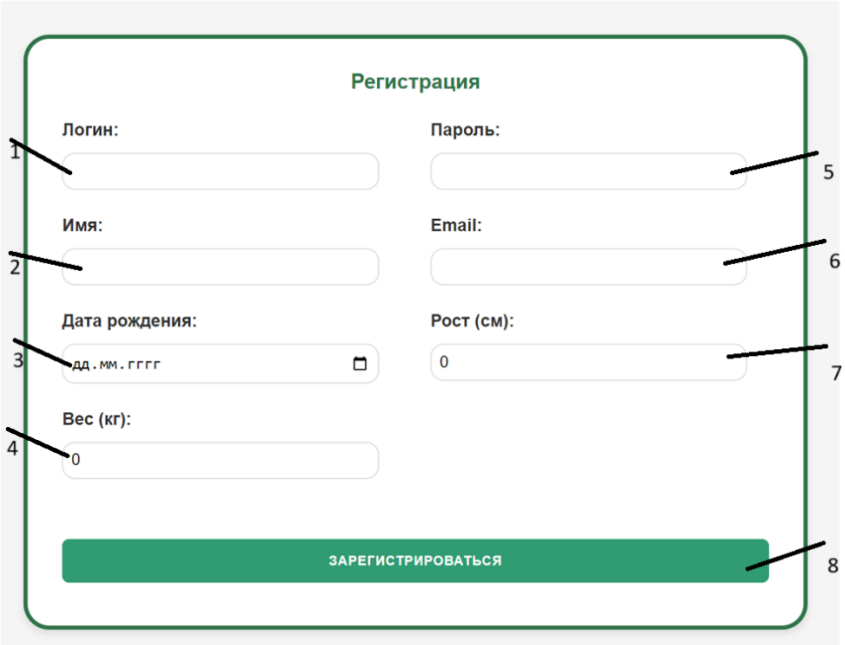


Рисунок 15 – Страница регистрации

Основные элементы интерфейса страницы регистрации, представленной на рисунке 15:

– 1 – поле для ввода логина;

– 2 – поле для ввода имени;
– 3 – поле для ввода даты рождения;
– 4 – поле для ввода веса (кг);
– 5 – поле для ввода пароля;
– 6 – поле для ввода адреса электронной почты;
– 7 – поле для ввода роста (см);
– 8 – кнопка для регистрации.

На рисунке 16 представлена страница авторизации.

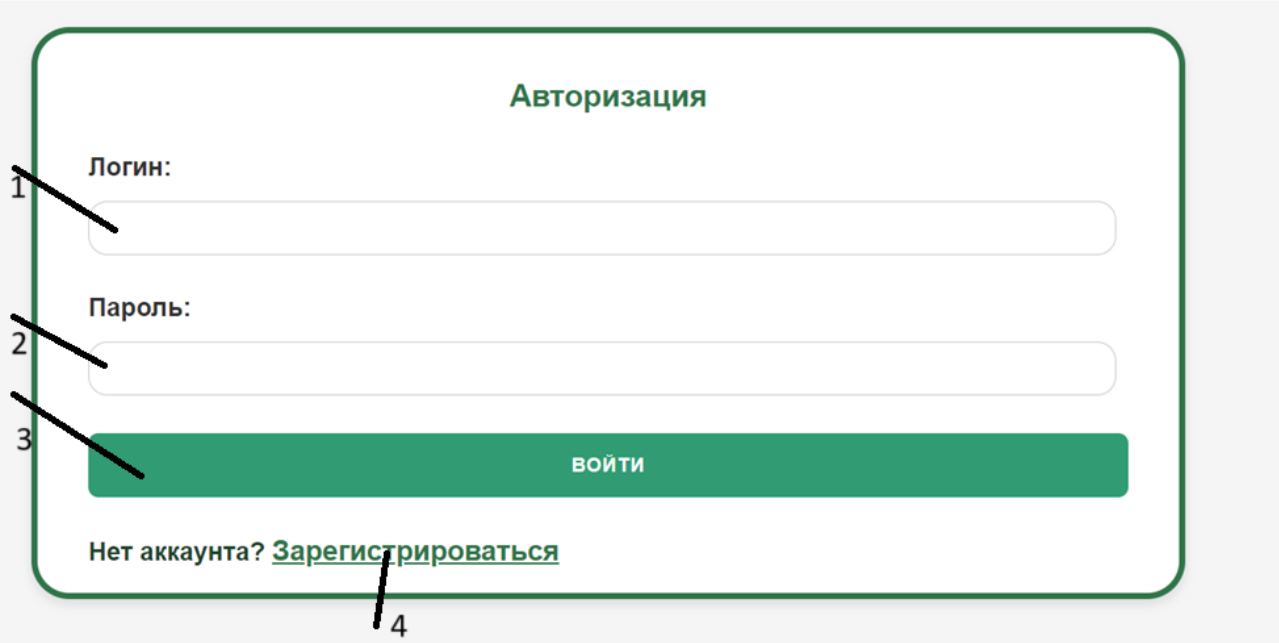


Рисунок 16 – Страница авторизации

Основные элементы интерфейса страницы авторизации, представленной на рисунке 16:

– 1 – поле для ввода логина;
– 2 – поле для ввода пароля;
– 3 – кнопка для входа;
– 4 – ссылка для перехода к регистрации.

На рисунке 17 представлена страница раздела «Питание».

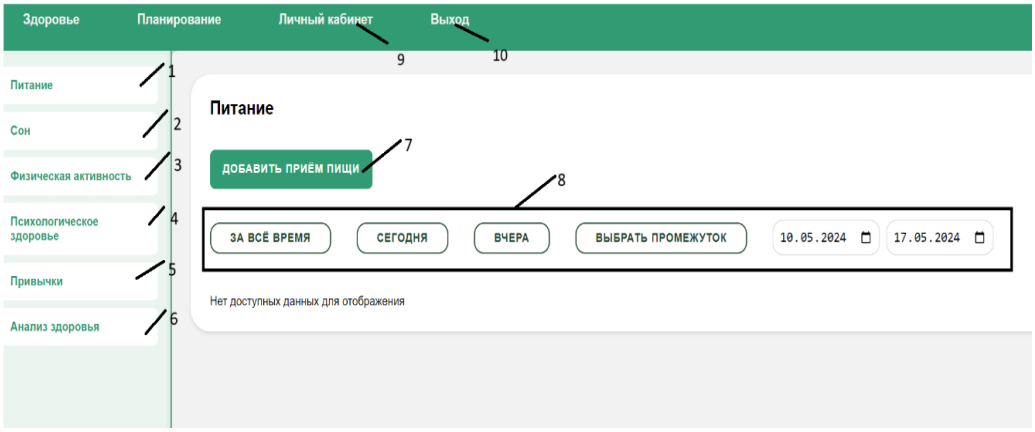


Рисунок 17 – Страница раздела «Питание»

Основные элементы страницы питания, представленной на рисунке 17:

– 1 – раздел «питание»;
– 2 – раздел «сон»;
– 3 – раздел «физическая активность»;
– 4 – раздел «психологическое здоровье»;
– 5 – раздел «привычки»;
– 6 – раздел «анализ здорового образа жизни»;
– 7 – кнопка «добавить прием пищи»;
– 8 – кнопки выбора временного интервала;
– 8 – раздел «личный кабинет»;
– 10 – раздел «выход».

На рисунке 18 представлен интерфейс модульного окна «Добавить прием пищи».

Рисунок 18 – Модальное окно «Добавить прием пищи»

Основные элементы интерфейса окна добавления приему пищи, представленного на рисунке 18:

– 1 – кнопка закрытия формы;
– 2 – поле выбора типа приема пищи;
– 3 – поле для ввода названия блюда;
– 4 – поле для ввода количества граммов;
– 5 – кнопка для добавления информации.

На рисунке 19 представлена страница раздела «Сон».

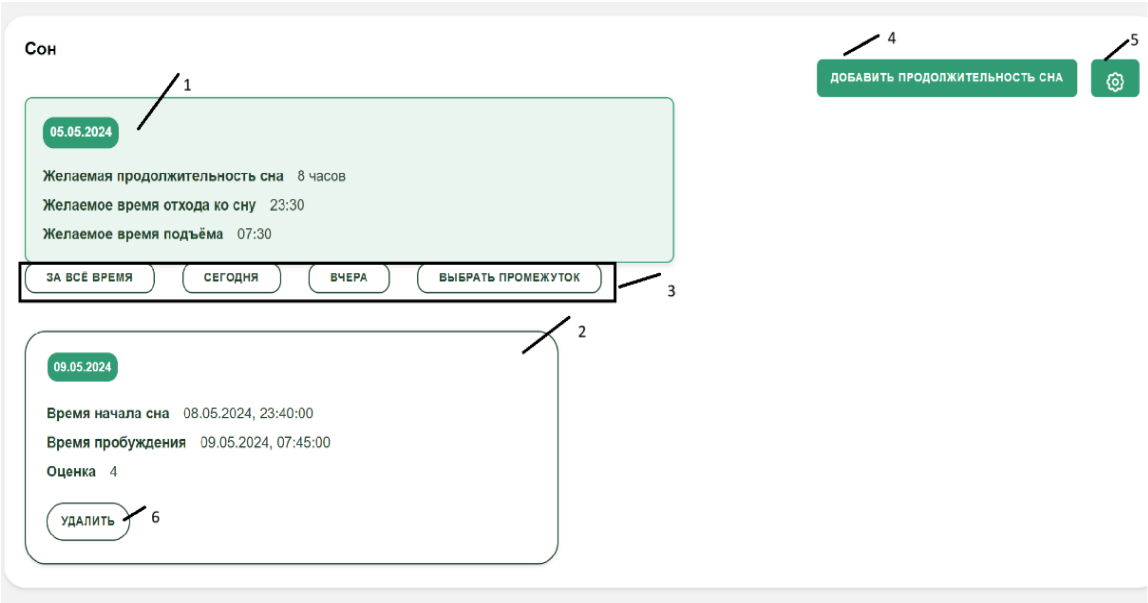


Рисунок 19 – Страница раздела «Сон»

Основные элементы интерфейса окна, представленного на рисунке 19:

– 1 – дата записи о сне;
– 2 – запись о времени начала и окончания сна;
– 3 – кнопки выбора временного интервала для отображения данных;
– 4 – оценка сна;
– 5 – кнопка для добавления желаемой продолжительности сна;
– 6 – кнопка для удаления записи.

На рисунке 20 представлена страница раздела «Физическая активность».

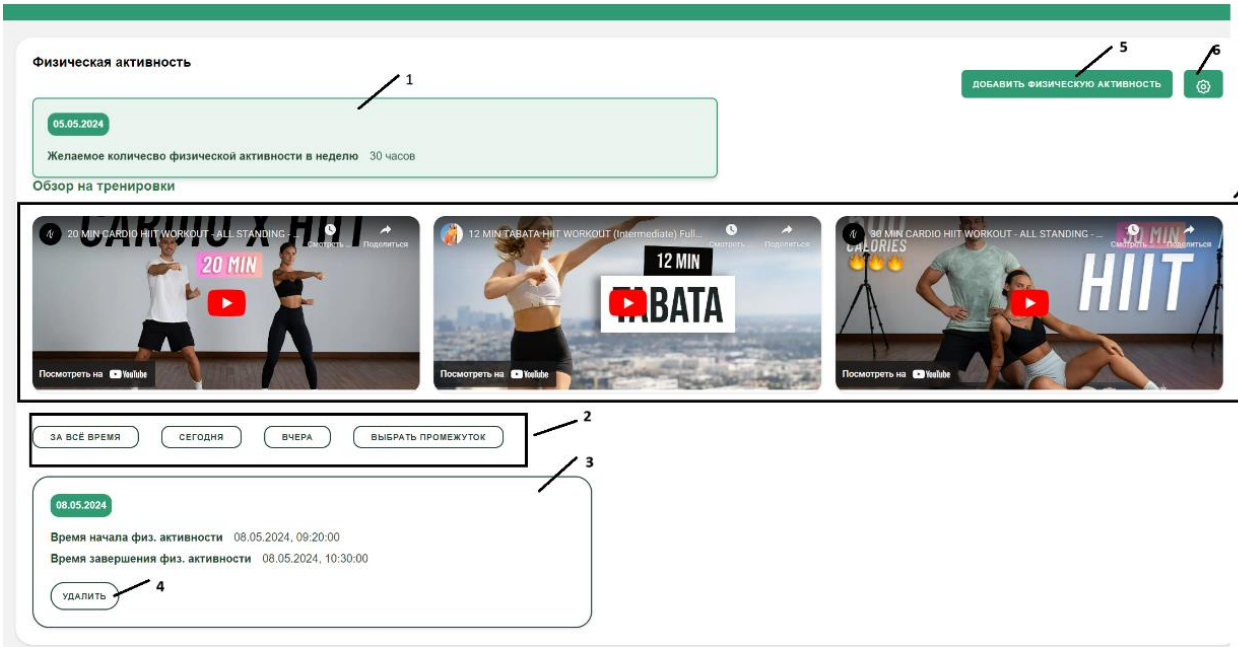


Рисунок 20 – Страница раздела «Физическая активность»

Основные элементы интерфейса окна, представленного на рисунке 20:

– 1 – заголовок «физическая активность»;

– 2 – кнопки для выбора периода: «за все время», «сегодня», «вчера», «выбрать промежуток»;

– 3 – информация о физической активности: дата, время начала и завершения;

– 4 – кнопка «удалить»;

– 5 – кнопка «добавить физическую активность».

На рисунке 21 представлена страница раздела «Психологическое состояние». В данном разделе пользователь может пройти тестирование для оценки своего психологического состояния, в результате которого ему будет отображена результирующая оценка. Кроме того, по результатам данной оценки выводится сообщение с рекомендациями, которые пользователь может выполнить для улучшение качества своего психологического состояния. Одни и те же тесты можно проходить несколько раз, что позволяет отслеживать динамику психологического состояния пользователя.

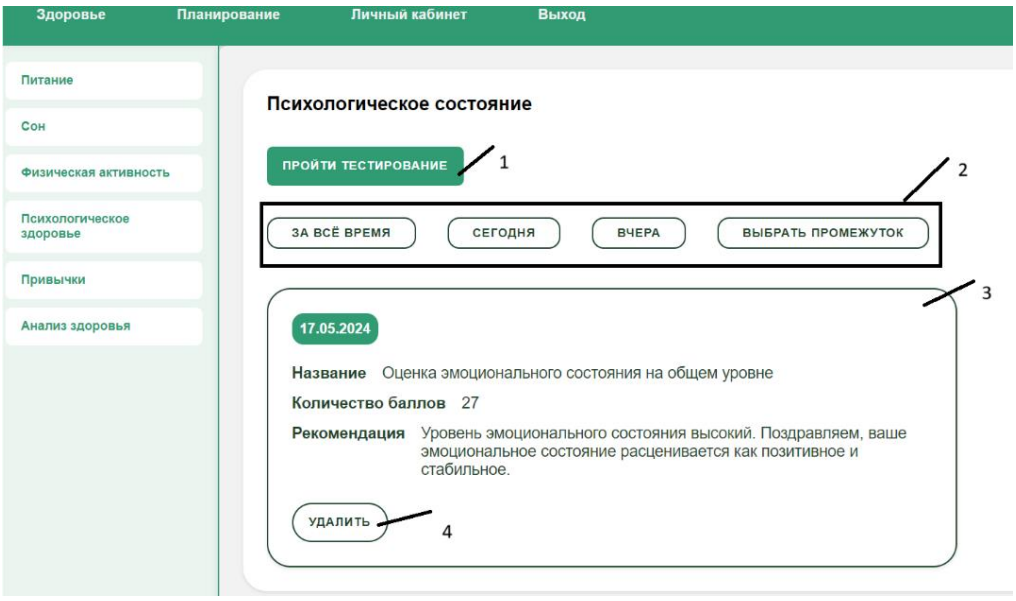


Рисунок 21 – Страница раздела «Психологическое состояние»

Основные элементы интерфейса окна, представленного на рисунке 21:

– 1 – кнопка «пройти тестирование»;

– 2 – кнопки для выбора периода: «за все время», «сегодня», «вчера», «выбрать промежуток»;

– 3 – информация о прохождении психологического теста;

– 4 – кнопка «удалить».

На рисунке 22 представлен интерфейс модального окна «Привычки».

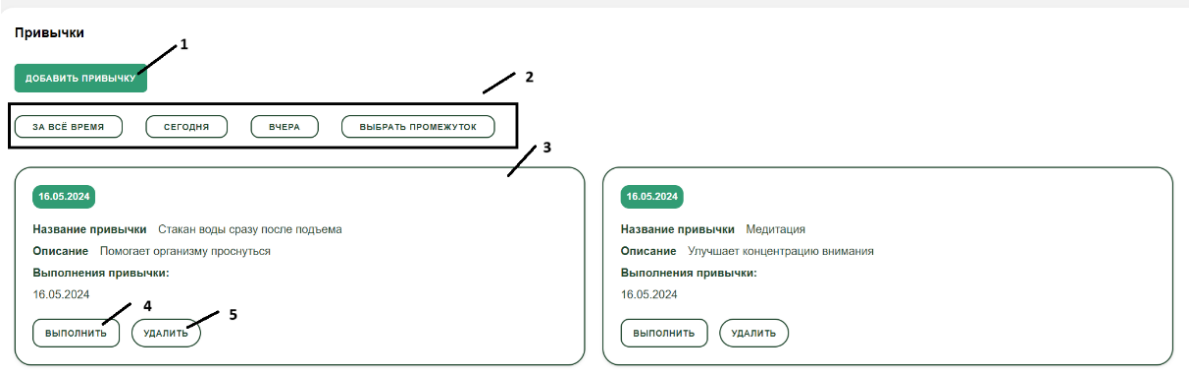


Рисунок 22 – Модальное окно «Привычки»

Основные элементы окна, представленного на рисунке 22:

– 1 – кнопка «добавить привычку»;

– 2 – кнопки для выбора периода времени;

– 3 – информация о привычке;

– 4 – кнопка для отслеживания выполнения;

– 5 – кнопка для удаления привычки.

Интерфейс раздела аналитики представлен в приложения Б.

Таким образом, спроектированный пользовательский интерфейс удовлетворяет функциональным требованиям приложения.

2.4 Проектирование программных средств

На основе логической модели базы данных была спроектирована диаграмма классов мобильного приложения, представленная на рисунке 23.

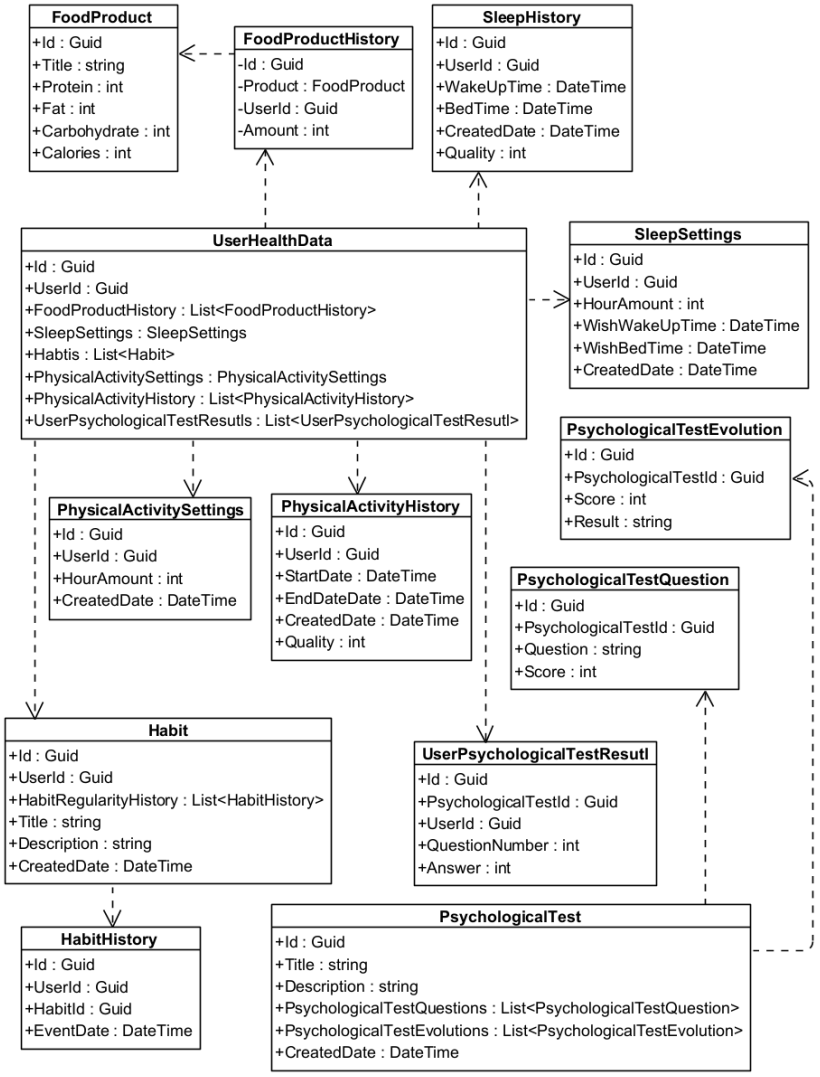


Рисунок 23 – Диаграмма классов моделей данных приложения

Краткое описание функционального назначения представленных на диаграмме классов приведено в таблице 3.

Таблица 3 – Компоненты диаграммы классов

Компонент	Назначение
FoodProduct	Класс, описывающий модель данных продукт питания
FoodProductHistory	Класс, описывающий модель данных употребленных пользователем продуктов питания
Habit	Класс, описывающий модель данных привычки
HabitHistory	Класс, описывающий модель данных выполняемых пользователем привычек
SleepSettings	Класс, описывающий модель данных настроек сна
SleepHistory	Класс, описывающий модель данных истории сна
PhisicalActivitySettings	Класс, описывающий модель данных настроек физической активности
PhisicalActivityHistory	Класс, описывающий модель данных истории физической активности
PsycologicalTest	Класс, описывающий модель данных психологического теста
PsycologicalTestQuestion	Класс, описывающий модель вопроса психологического теста
PsycologicalTestEvolution	Класс, описывающий модель данных оценки психологического теста
UserPsycologicalTestResult	Класс, описывающий модель данных результатов прохождения психологического теста
UserHealthData	Класс, описывающий модель данных компонентах здорового образа жизни пользователя

В ходе разработки приложения будет реализовано 14 классов, отвечающих за работу с данными предметной области.

3 Реализация программного средства для осознанного планирования и контроля здорового образа жизни

3.1 Реализация базы данных

На основе построенной ранее логической модели базы данных при помощи системы Visual Paradigm 17.0 была построена физическая модель [16]. Физическая модель базы данных «Отчетность» представлена на рисунке 24.

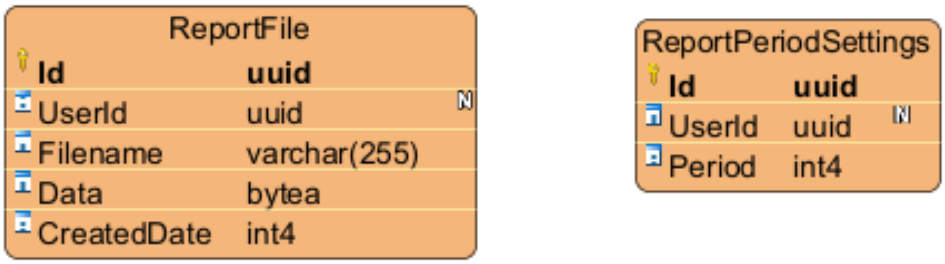


Рисунок 24 – Графическое представление физической модели базы данных «Отчетность»

Графическое представление физической модели базы данных «Авторизация» представлено на рисунке 25.

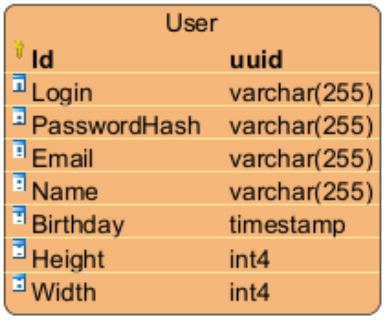


Рисунок 25 – Графическое представление физической модели базы данных «Авторизация»

Графическое представление физической модели базы данных «Здоровье» представлено на рисунке 26.

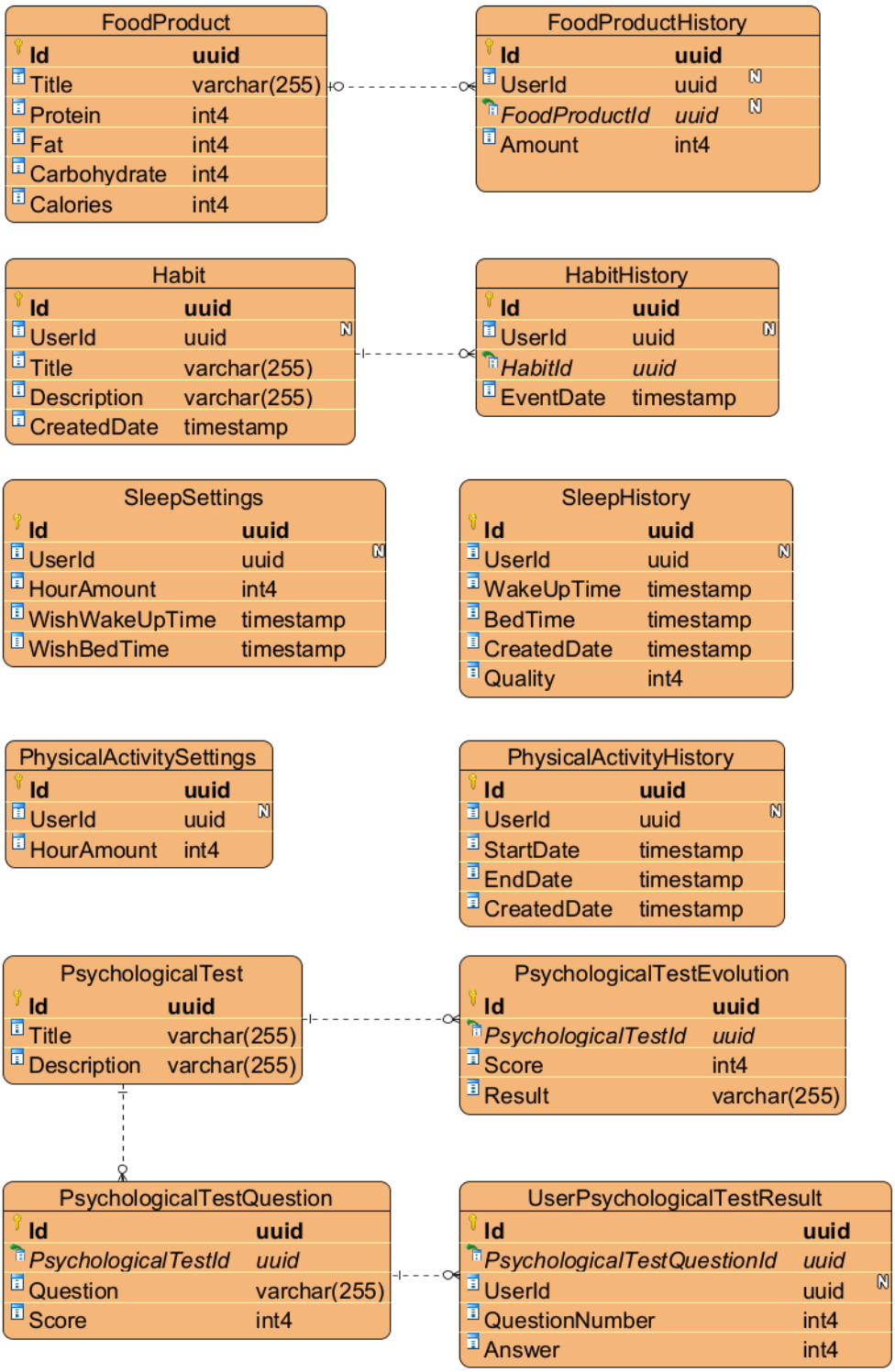


Рисунок 26 – Графическое представление физической модели базы данных «Здоровье»

Наборы атрибутов для каждой сущности предметной области, их характеристики и связи приведены в приложении Б.

Создание таблиц баз данных реализовано посредством системы миграций с помощью библиотеки FluentMigrator платформы .NET. Такой подход обеспечивает надежное развертывание и консистентность данных на всех возможных окружениях, которые могут применяться при развертывании приложения.

3.2 Реализация программных компонентов

Приложение разрабатывалось с применением паттерна проектирования MVC (Model-View-Controller). MVC – это шаблон программирования, который позволяет разделить логику приложения на три части:

– «Model» (модель): получает данные от контроллера, выполняет необходимые операции и передаёт их в вид;

– «View» (вид или представление): получает данные от модели и выводит их для пользователя;

– «Controller» (контроллер): обрабатывает действия пользователя, проверяет полученные данные и передаёт их модели.

Этот паттерн разработки нужен для того, чтобы разделить логические части приложения и создавать их отдельно друг от друга, то есть писать независимые блоки кода, которые можно, как угодно, менять, не затрагивая другие [17].

При разработке были выделены следующие программные модули:

– «Авторизация» – микросервис, реализующий логику аутентификации и авторизации пользователя с использованием технологии JSON Web Token (JWT);

– «Здоровье» – микросервис, реализующий основную бизнес-логику приложения;

– «Отчетность» – микросервис, реализующий логику формирования отчетов [18].

Для данных модулей были разработаны диаграммы классов. Диаграмма классов для микросервиса «Авторизация» представлена на рисунке 27.

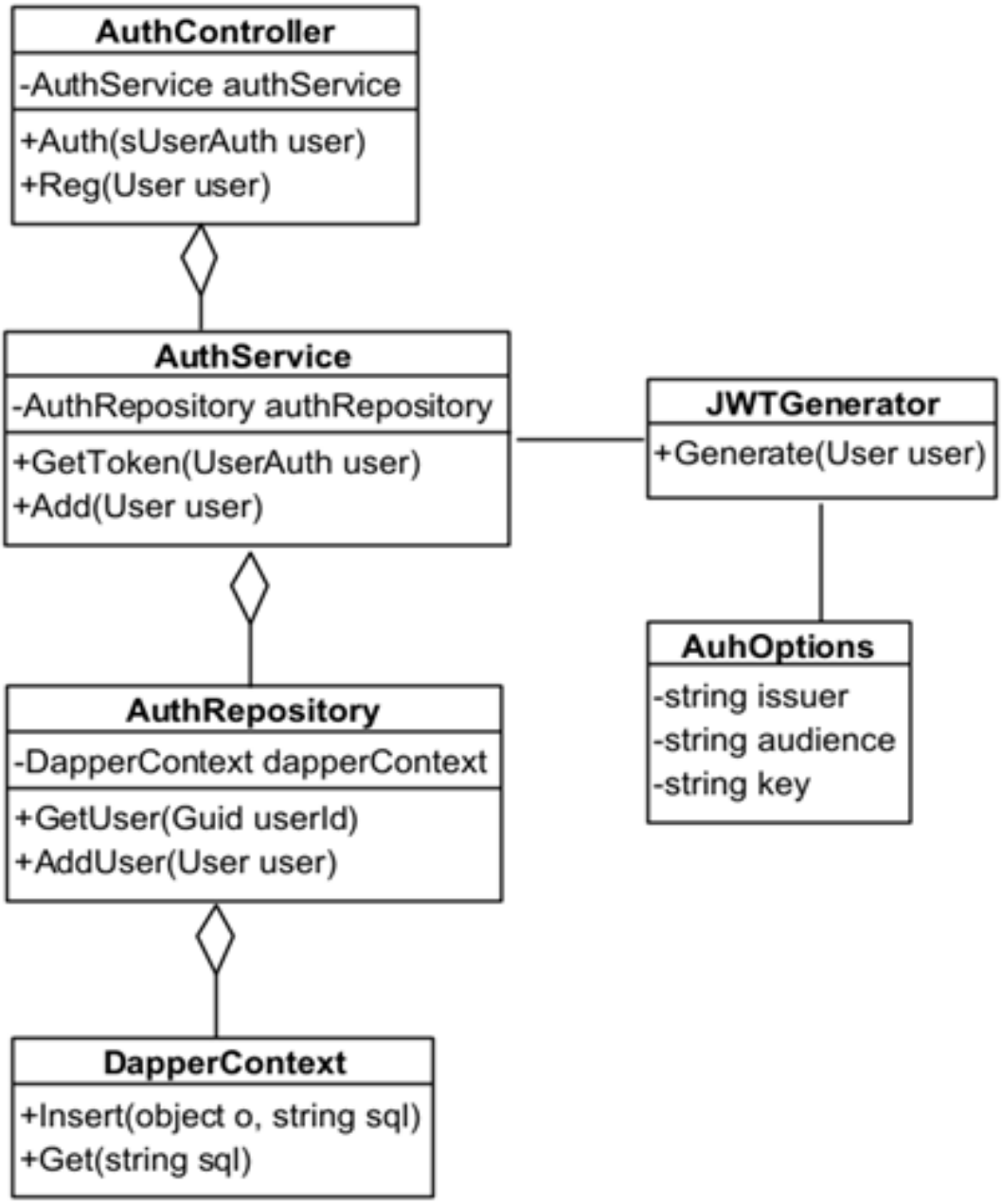


Рисунок 27 – Диаграмма классов микросервиса «Авторизация»

Диаграмма классов микросервиса «Здоровье» представлена на рисунке В.1 в приложении В. Диаграмма классов микросервиса «Отчетность» представлена на рисунке 28.

На диаграмме классов микросервиса «Здоровье» следует обратить внимание на атрибуты и методы, реализованные для каждого класса.

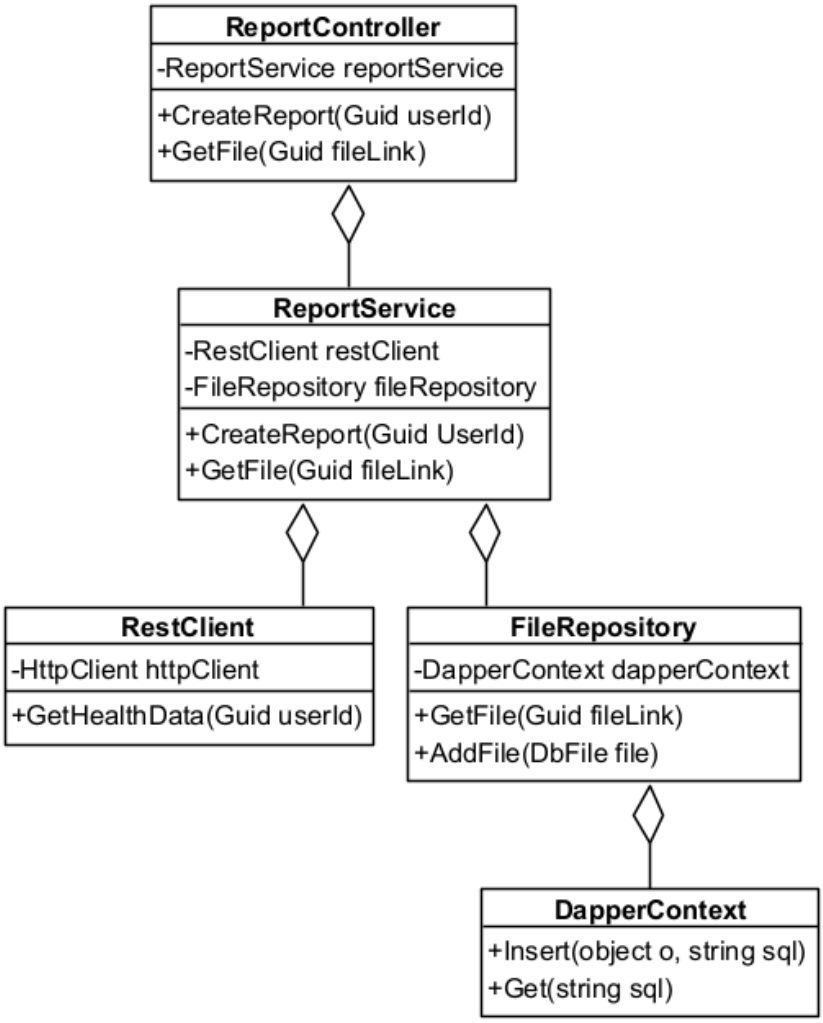


Рисунок 28 – Диаграмма классов микросервиса «Отчетность»

Серверное приложение предоставляет REST API для взаимодействия. Предоставляемые методы описаны в сгенерированной документации OpenApi Swagger на рисунке Г.2 в приложении Г.

Серверная часть приложения разрабатывалась с использованием интегрированной среды разработки JetBrains Rider на языке C# и фреймворка ASP.NET. Для взаимодействия с базой данных использовались SQL-запросы, обрабатываемые с помощью библиотеки ORM Dapper [19]. Код серверной части приложения содержится в приложении Ж.

На рисунке 29 представлена диаграмма компонентов серверной части приложения микросервиса «Здоровье».

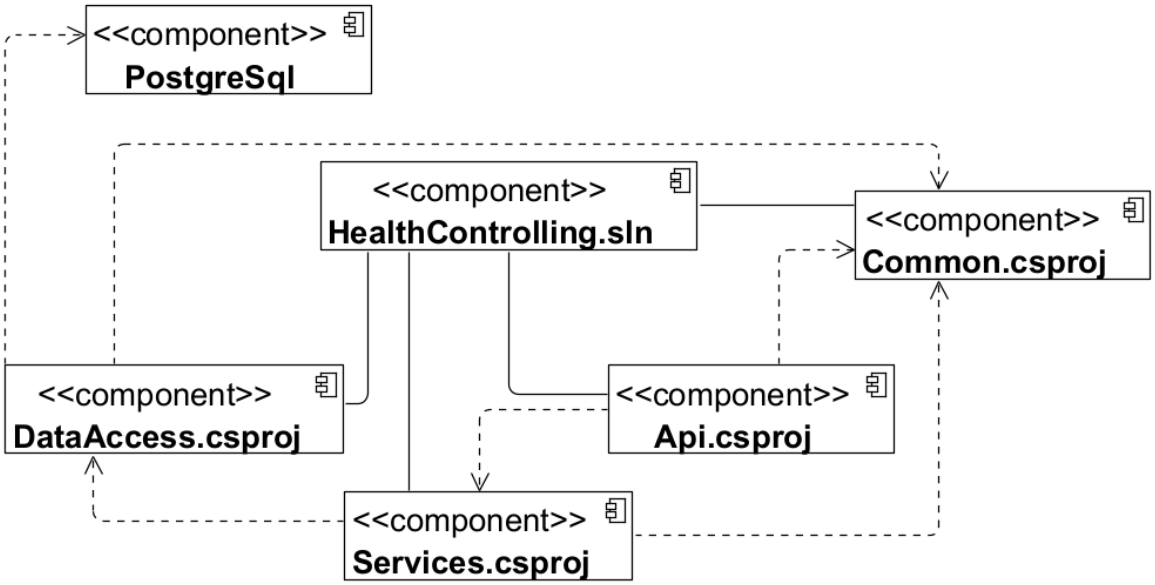


Рисунок 29 – Диаграмма компонентов микросервиса «Здоровье»

Компоненты микросервиса «Авторизация» представлены на рисунке 30.

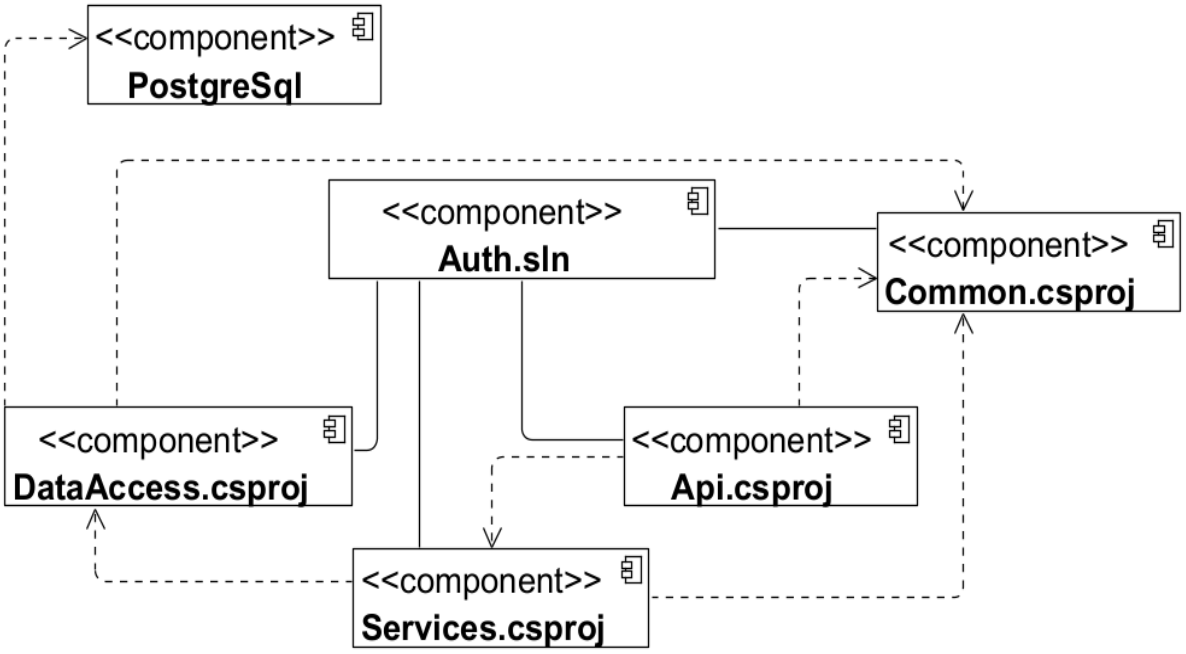


Рисунок 30 – Диаграмма компонентов микросервиса «Авторизация»

Для микросервиса «Отчетность» также была разработана диаграмма компонентов (рисунок 31).

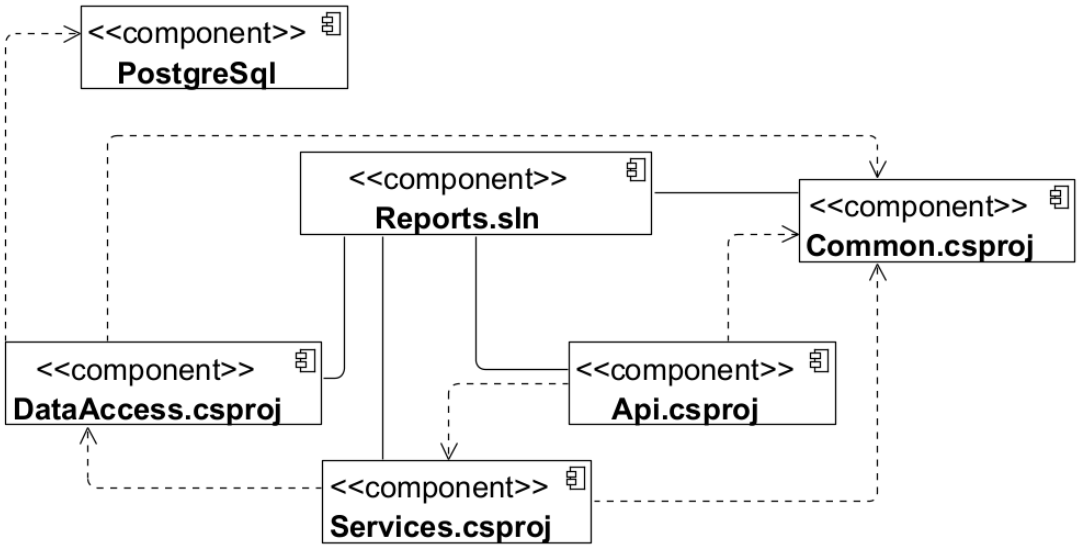


Рисунок 31 – Диаграмма компонентов микросервиса «Отчетность»

Рассмотрим каждый компонент:

– «HealthControlling.sln» – файл решения, собирающий проектные файлы расширения .csproj микросервиса «Здоровье»;

– «Auth.sln» – файл решения, собирающий проектные файлы расширения .csproj микросервиса «Авторизация»;

– «Reports.sln» – файл решения, собирающий проектные файлы расширения .csproj микросервиса «Авторизация»;

– «Api.csproj» – компонент отвечающие за принятие и обработку запросов;

– «Services.csproj» – компонент, отвечающий бизнес-логику приложения;

– «DataAccess.csproj» – компонент, отвечающий за взаимодействие с базой данных;

– «PostgreSql» – компонент, представляющий базу данных.

Описанные программные модули разворачивались с использованием технологии контейнеризации Docker. На рисунке 32 представлена диаграмма развертывания приложения, которая изображает основные компоненты системы [20].

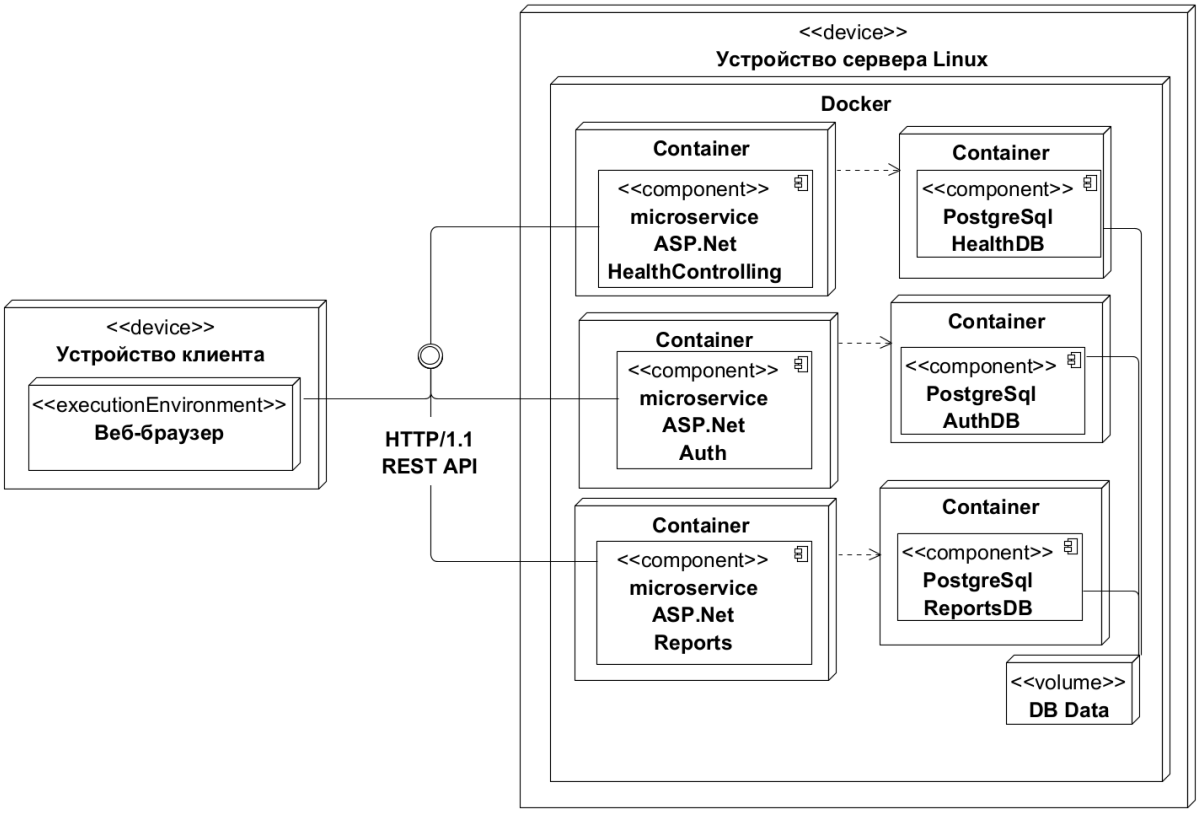


Рисунок 32 – Диаграмма развертывания

Из диаграммы видно, что программные модули обернуты в docker-контейнеры, что позволяет в дальнейшем легко разворачивать приложение на другие сервера с платформами, поддерживаемыми docker. Docker позволяет хранить приложение со всем его окружением и зависимостями в контейнере и запускать его из изолированной среды контейнера. Порядок развертки задан посредством технологии Docker-Compose. Каждый микросервис запускается в своем контейнере и имеет зависимость от соответствующей ему базы данных PostgreSql, каждая из которых имеет возможность сохранять информацию хранимых объектов на диск устройства [21]. Взаимодействие с контейнерами микросервисов доступно по протоколу HTTP/1.1 с использованием технологии REST API.

Клиентское приложение разрабатывалось на языке JavaScript, с применением фреймворка «React.js». Запросы к серверу отправлялись посредством библиотеки «Axios.js».

Клиентское приложение запускается на устройствах пользователей в браузере. При открытии пользователем веб-приложения браузер получает все необходимые для запуска файлы клиентского приложения. Листинг клиентского приложения представлен в приложении И.

4 Контроль качества программного обеспечения

4.1 Выбор режима тестирования

Существует достаточно много видов тестирования программного продукта. Были рассмотрены следующие виды тестирования [22]:

– функциональное тестирование;
– модульное тестирование;
– интеграционное тестирование.

В функциональном тестировании основное внимание уделяется бизнес-требованиям к приложению. Они проверяют только результат некоторого действия и не проверяют промежуточные состояния системы при выполнении этого действия [23].

Юнит тестирование или модульное тестирование работает на очень низком уровне, близко к исходному коду приложения. Они заключаются в тестировании отдельных методов и функций классов, компонентов или модулей, используемых в программном обеспечении (ПО). Модульные тесты, как правило, не требуют больших расходов на автоматизацию и могут выполняться сервером непрерывной интеграции очень быстро [24].

Интеграционное тестирование проверяет, хорошо ли работают вместе различные модули и сервисы, используемые приложением. Например, можно протестировать взаимодействие с базой данных или убедиться, что микросервисы работают вместе так, как задумано. Этот вид тестирования является более затратным, поскольку для проведения тестов требуется запуск различных компонентов приложения.

Было принято решение осуществить проверку серверной части программы с помощью модульного тестирования, а клиентской части – с помощью функционального тестирования. Такой выбор обусловлен тем, что к программе были предъявлены в основном функциональные требования, выполнение которых нужно проверить.

4.2 Тестирование серверной части веб-приложения

Модульное тестирование позволяет проверить в автоматическом режиме корректность работы отдельных модулей программы.

Идея состоит в том, чтобы писать тесты для каждой нетривиальной функции или метода. Это позволяет достаточно быстро проверить, не привело ли очередное изменение кода к регрессии, то есть к появлению ошибок в уже оттестированных местах программы, а также облегчает обнаружение и устранение таких ошибок.

Также при модульном тестировании необходимо знать метрику покрытия кода. Она показывает процент исходного кода программы, который был выполнен в процессе тестирования. Для валидации русского и иностранного паспорта при покупке были написаны unit-тесты. Для написания использовался nuget-пакет «NUnit». «NUnit» – это открытая среда юнит-тестирования приложений для .NET. План тестирования приведен в таблице 4.

Таблица 4 – План модульного тестирования

Тест	Описание	Ожидаемый результат	Актуальный результат
TestAddHabitHistory	Проверка корректности добавления выполненной привычки	Пройден	Пройден
TestAddPhysicalActivityHistory	Проверка корректности добавления физической активности	Пройден	Пройден
TestAddSleepHistory	Проверка корректности добавления периода сна	Пройден	Пройден
TestGetUserHealthData	Проверка корректности получения данных по аспектам здорового образа жизни пользователя	Пройден	Пройден

Результаты модульного тестирования приведены на рисунке 33, из них видно, что все тесты успешно пройдены.

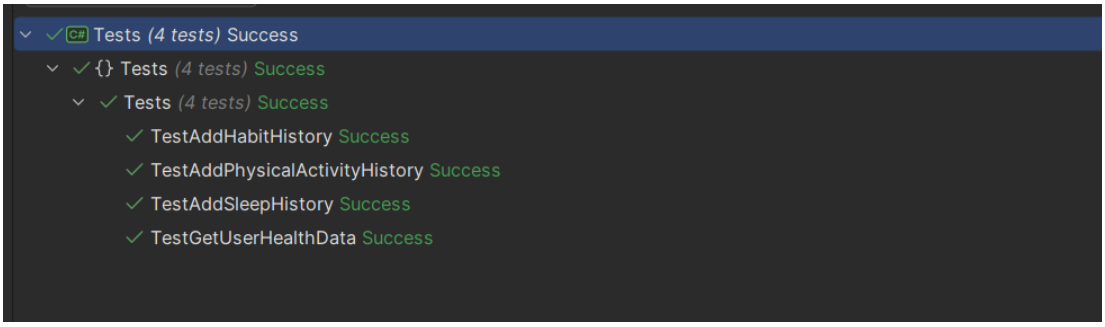


Рисунок 33 – Результаты модульного тестирования

В ходе тестирования были выявлены и исправлены ошибки в коде серверной части веб-приложения.

4.3 Тестирование клиентской части веб-приложения

Для основных вариантов использования были выполнены функциональные тесты. План тестирования приведен в таблице 5. Результаты тестирования представлены в приложении Е.

Таблица 5 – План модульного тестирования

№ п/п	Назначение теста	Описание	Ожидаемый результат	Результаты теста
1	Тест предназначен для проверки варианта использования «Просмотреть учет привычек»	Выбор раздела «Привычки», получение данных об учете выполненных привычек	Система отображает данные о выполненных привычках пользователя	Пройден (рисунок Е.1)
2	Тест предназначен для проверки варианта использования «Просмотреть учет сна»	Выбор раздела «Сон», получение данных об учете периодов сна	Система отображает данные о периодах пользователя	Пройден (рисунок Е.2)
3	Тест предназначен для проверки варианта использования «Внести данные о сне»	Выбор раздела «Сон», добавление данных периода сна	Пройден	Пройден (рисунок Е.3)
4	Тест предназначен для проверки варианта использования «Внести данные о настройках сна»	Выбор раздела «Сон», добавление данных настроек сна	Пройден	Пройден (рисунок Е.4)

В ходе тестирования были выявлены и исправлены ошибки в коде программы.

4.4 Метрики устойчивости и абстрактности компонент

Метрика устойчивости для компоненты отражает количественную меру работы, которую необходимо произвести при изменении других зависимых компонент [25].

Для вычисления позиционной устойчивости кода используется следующая формула:

$$I = \frac{Ce}{Ca + Ce},$$

где Ce – количество классов вне данного компонента, зависящих от классов внутри компонента;

Ca – количество классов внутри данного компонента, зависящих от классов вне компонента.

Величина меры неустойчивости изменяется в диапазоне от [0, 1]. В первом случае компонент максимально устойчив, а во втором – максимально неустойчив.

Для компонента HabitService была рассчитана величина метрики неустойчивости:

$$I = \frac{3}{3 + 3} = 0.5.$$

Мерой абстрактности компонента служит метрика A. Ее значение определяется простым отношением количества интерфейсов и абстрактных классов к общему числу классов в компоненте [22]. Расчет метрики A:

$$A = \frac{Na}{Nc},$$

где Na – количество абстрактных классов в компоненте;

Nc – количество классов в нем же.

Значение метрики A изменяется в диапазоне от 0 до 1. Первый случай означает полное отсутствие абстрактных классов в компоненте, а второй означает, что компонент не содержит ничего, кроме абстрактных классов.

Для компонента HabitService была рассчитана метрика абстрактности A:

$$A = \frac{2}{7} = 0.285.$$

Вычислив меру устойчивости компоненты и ее абстрактность можно отобразить их на графике. На вертикальной оси необходимо отложить меру абстрактности A, а на горизонтальной оси – меру неустойчивости I.

Компоненты который располагаются в верхнем левом углу или в нижнем правым являются максимально качественным. На практике часто случается, так что полученная точка лежит на диагонали между точкой (1,0) и точкой (0,1). Такая диагональ называется главная последовательность (рисунок 34).

Для вычисления количественной оценки используется формула вычисления расстояние до главной последовательности:

$$D = \left|\frac{A + I - 1}{\sqrt{2}}\right|.$$

Чем ближе точка располагается к главной последовательности, тем лучше. На рисунке 34 представлены зоны исключения.

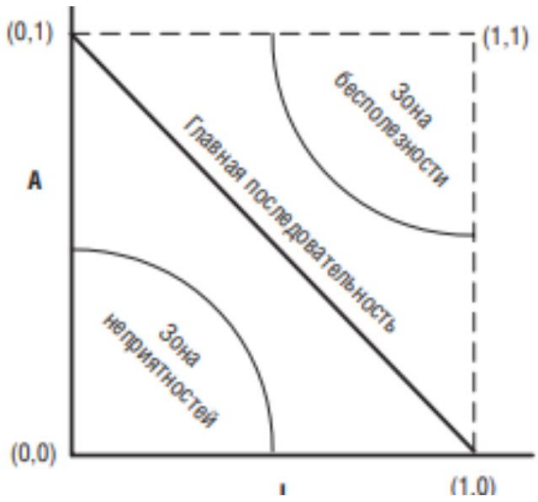


Рисунок 34 – Зоны исключения

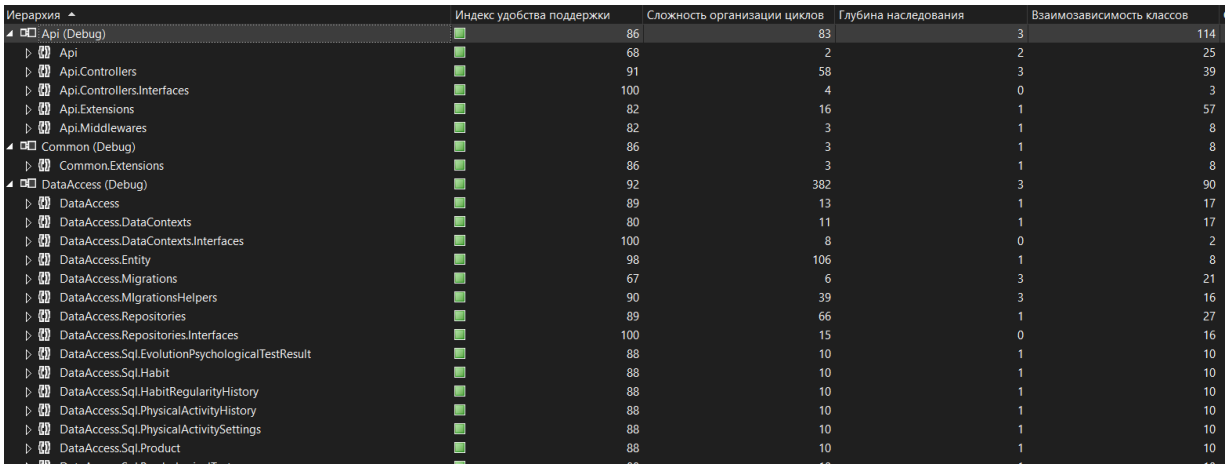
Кроме того, было вычислено расстояние до главной последовательности по следующей формуле:

$$D = \left|\frac{0.285 + 0.5 - 1}{\sqrt{2}}\right| = 0.15.$$

Расстояние отношений метрик первой компоненты до главной последовательности достаточно близко к нулю, а значит, ее отношение абстрактности к устойчивости является правильным.

4.5 Метрики кода

Был проведен анализ качества кода реализованной программы с помощью Visual Studio. Метрики кода приведены на рисунке 35.



Иерархия	Индекс удобства поддержки	Сложность организации циклов	Глубина наследования	Взаимозависимость классов
Api (Debug)	86	83	3	114
Api	68	2	2	25
Api.Controllers	91	58	3	39
Api.Controllers.Interfaces	100	4	0	3
Api.Extensions	82	16	1	57
Api.Middlewares	82	3	1	8
Common (Debug)	86	3	1	8
Common.Extensions	86	3	1	8
DataAccess (Debug)	92	382	3	90
DataAccess	89	13	1	17
DataAccess.DataContexts	80	11	1	17
DataAccess.DataContexts.Interfaces	100	8	0	2
DataAccess.Entity	98	106	1	8
DataAccess.Migrations	67	6	3	21
DataAccess.MIgrationsHelpers	90	39	3	16
DataAccess.Repositories	89	66	1	27
DataAccess.Repositories.Interfaces	100	15	0	16
DataAccess.Sql.EvolutionPsychologicalTestResult	88	10	1	10
DataAccess.Sql.Habit	88	10	1	10
DataAccess.Sql.HabitRegularityHistory	88	10	1	10
DataAccess.Sql.PhysicalActivityHistory	88	10	1	10
DataAccess.Sql.PhysicalActivitySettings	88	10	1	10
DataAccess.Sql.Product	88	10	1	10

Рисунок 35 – Метрики кода

Индекс удобства поддержки показывает, насколько удобно и легко поддерживать данный код. Рассчитывается по формуле:

$$MI = MAX\left(0, (171 - 5.2 * \ln(HV) - 0.23 * CC - 16.2 * \ln(LoC)) * \frac{100}{171}\right),$$

где HV – Halstead Volume, вычислительная сложность;

CC – Cyclomatic Complexity, структурная сложность кода;

LoC – количество строк исходного кода [25].

Нормой является значение от 20 до 100. В данном случае значения метрики для каждого модуля близки к 100, что означает, что код удобен в поддержке.

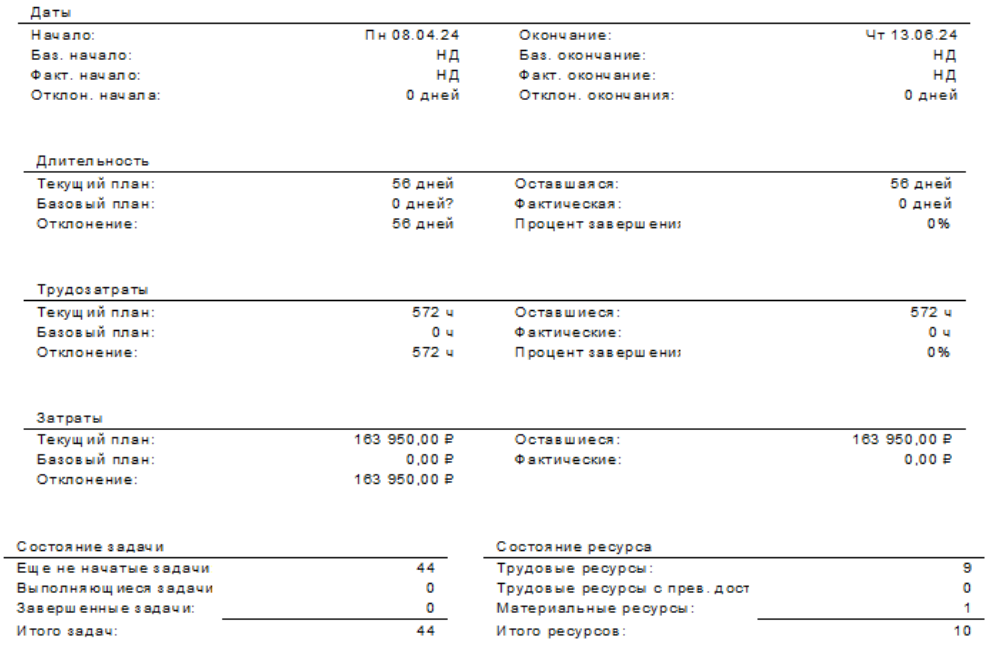
4.6 Планирование разработки и оценка бюджета

Планирование разработки и оценка бюджета была выполнена при помощи программы для управления проектами Microsoft Project [26]. Проект был начат 8 апреля 2024 года.

При планировании разработки было выделено 44 задачи. Список задач и их описание представлено на рисунке Д.1 в приложении Д.

Для реализации проекта необходимо выделить основные ресурсы, их типы и стоимость. На рисунке Д.2 в приложении Д представлена диаграмма Ганта, построенная в среде Microsoft Project на основе введенных данных.

На рисунке 36 представлен обзорный отчет проекта.



Даты			
Начало:	Пн 08.04.24	Окончание:	Чт 13.06.24
Баз. начало:	НД	Баз. окончание:	НД
Факт. начало:	НД	Факт. окончание:	НД
Отклон. начала:	0 дней	Отклон. окончания:	0 дней

Длительность			
Текущий план:	56 дней	Оставшаяся:	56 дней
Базовый план:	0 дней?	Фактическая:	0 дней
Отклонение:	56 дней	Процент завершени	0%

Трудозатраты			
Текущий план:	572 ч	Оставшиеся:	572 ч
Базовый план:	0 ч	Фактические:	0 ч
Отклонение:	572 ч	Процент завершени	0%

Затраты			
Текущий план:	163 950,00 ₽	Оставшиеся:	163 950,00 ₽
Базовый план:	0,00 ₽	Фактические:	0,00 ₽
Отклонение:	163 950,00 ₽		

Состояние задачи	
Еще не начатые задачи	44
Выполняющиеся задачи	0
Завершенные задачи:	0
Итого задач:	44

Состояние ресурса	
Трудовые ресурсы:	9
Трудовые ресурсы с прев. дост	0
Материальные ресурсы:	1
Итого ресурсов:	10

Рисунок 36 – Обзорный отчет проекта

Из отчета, представленного на рисунке 36, видно, что предполагаемые затраты на разработку составляют 163959 рублей, а общая длительность проекта – 56 дней.

Заключение

В ходе выполнения выпускной квалификационной работы разработано веб-приложение для осознанного планирования и контроля здорового образа жизни.

Выполнен анализ предметной области приложений для ведения здорового образа жизни. Изучены информационные источники по данной теме, выделены наиболее популярные в настоящее время программные средства для планирования и контроля ЗОЖ. Проведен сравнительный анализ существующих решений, выявлены их достоинства и недостатки. На основе данного анализа сформулированы требования к разрабатываемому веб-приложению. Создана диаграмма вариантов использования, описывающая основные функции веб-приложения для осознанного планирования и контроля здорового образа жизни.

Проведено планирование и расчет бюджета разрабатываемого программного средства при помощи системы управления проектами Microsoft Project 2007.

На этапе проектирования веб-приложения определена его архитектура, разработана модель базы данных и спроектирован интерфейс приложения. Также были построены концептуальная и логическая модели базы данных и диаграмма классов UML для моделей предметной области приложения.

На этапе реализации веб-приложения была построена физическая модель базы данных, на основе которой база была создана и заполнена тестовыми данными. Для реализованных программных модулей разработаны диаграммы классов, компонентов и развертывания.

Произведено модульное и функциональное тестирование разработанного программного обеспечения. Все тесты пройдены успешно, ошибок не выявлено.

Список использованных источников

1. Чапаев И. А. Осознанное планирование жизни с использованием специализированных программных средств / И.А. Чапаев, А.Н. Дюжева, И.Ю. Балашова // Новые информационные технологии и системы: материалы XX международной науч.-техн. конф. – Пенза: Изд-во Пензенского государственного университета, 2023. – С. 218 – 221.

2. Долой лень, пора тренироваться! ЗОЖ с мобильными приложениями // Хабр [Электронный ресурс]. URL: https://habr.com/ru/articles/368081/ (дата обращения: 25.12.2023)

3. Чапаев И. А. Повышение продуктивности деятельности студентов вузов с использованием программных средств для осознанного планирования жизни / И.А. Чапаев, А.Н. Дюжева, И.Ю. Балашова // Инновационное образовательное пространство педагогического вуза. Проблемы и перспективы: материалы всероссийской науч.-пр. конф. – Саранск: Изд-во Мордовского государственного университета, 2023. – С. 179 – 183.

4. Бесплатный счетчик калорий и дневник питания и упражнений MyFitnessPal // MyFitnessPal [Электронный ресурс]. URL: https://www.myfitnesspal.com/ru/premium/ (дата обращения: 25.12.2023)

5. SleepCycle // SleepCycle [Электронный ресурс]. URL: https://www.sleepcycle.com/ (дата обращения: 25.12.2023)

6. Здоровое питание — это просто // LifeSum [Электронный ресурс]. URL: https://lifesum.com/ru/ (дата обращения: 25.12.2023)

7. Welltory // Welltory [Электронный ресурс]. URL: https://welltory.com/ (дата обращения: 25.12.2023)

8. SamsungHealth // SamsungHealth [Электронный ресурс]. URL: https://www.samsung.com/ru/apps/samsung-health/ (дата обращения: 25.12.2023)

9. Использование диаграммы вариантов использования UML при проектировании программного обеспечения // Хабр [Электронный ресурс]. URL: https://habr.com/ru/articles/566218/ (дата обращения: 23.01.2024)

10. Введение в ASP.NET CORE // ITVDN [Электронный ресурс]. URL: https://itvdn.com/ru/blog/article/aspnet_core#:~:text=ASP.NET%20Core%20–%20это%20новый,на%20полной%20платформе%20.NET%20Framework (дата обращения: 09.02.2024)

11. Архитектура клиент-сервер // ITELON [Электронный ресурс]. URL: https://itelon.ru/blog/arkhitektura-klient-server/ (дата обращения: 25.12.2023)

12. Клиент-серверная архитектура в картинках // Хабр [Электронный ресурс]. URL: https://habr.com/ru/articles/495698/ (дата обращения: 25.12.2023)

13. API от А до Я (теория и практика) // Хабр [Электронный ресурс]. URL: https://habr.com/ru/articles/768752/ (дата обращения: 10.01.2024)

14. Моделирование данных в СУБД: от концептуального проектирования к реализации // AppMaster[Электронный ресурс]. URL: https://appmaster.io/ru/blog/modelirovanie-dannykh-v-rdbms (дата обращения: 01.02.2024)

15. Page Application – что это и как с ним работать: создаем SPA на JS за час // Skillbox Media [Электронный ресурс]. URL: https://skillbox.ru/media/code/kak_sozdat_spa_na_js_i_php_za_chas/(дата обращения: 23.01.2024)

16. Физические модели баз данных // НОУ ИНТУИТ [Электронный ресурс]. URL: https://intuit.ru/studies/courses/1001/297/lecture/7415 (дата обращения: 21.02.2024)

17. ASP.NET Core MVC // metanit.com [Электронный ресурс]. URL: https://metanit.com/sharp/aspnet5/3.1.php (дата обращения: 08.03.2024)

18. Как создать простое Rest API на .NET Core // Хабр [Электронный ресурс]. URL: https://habr.com/ru/articles/531106/ (дата обращения: 06.03.2024)

19. Платформа .NET и C# от А до Я // csharp [Электронный ресурс]. URL: https://csharp.webdelphi.ru/ (дата обращения: 09.02.2024)

20. Простое руководство к диаграммам развертывания UML // creately [Электронный ресурс]. URL: https://creately.com/blog/ru/uncategorized-ru/учебное-пособие-по-диаграмме-развёрт/ (дата обращения: 15.02.2024)

21. Стоунз, Р. Основы PostgreSQL / Р. Стоунз, Н. Меттью. – М.: Символ-Плюс, 2002. – 640 с.

22. Тестирование и 7 основных этапов его проведения // Neiros [Электронный ресурс]. URL: https://neiros.ru/blog/code/testirovanie-i-7-osnovnykh-etapov-ego-provedeniya/ (дата обращения: 20.03.2024)

23. Функциональное тестирование: что это, этапы, виды и инструменты использования // Tproger [Электронный ресурс]. URL: https://tproger.ru/articles/funkcionalnoe-testirovanie-chto-eto-etapy-vidy-i-instrumenty-ispolzovaniya (дата обращения: 20.03.2024)

24. Модульное тестирование: что это? Типы, инструменты // logrocon [Электронный ресурс]. URL: https://logrocon.ru/news/unit_testing (дата обращения: 20.03.2024)

25. Программный код и его метрики // Хабр [Электронный ресурс]. URL: https://habr.com/ru/companies/intel/articles/106082/ (дата обращения: 27.03.2024)

26. Зубрицкий, А. А. Эффективное управление проектами с использованием Microsoft Project 2016 / А. А. Зубрицкий. – М.: Наука, 2016. – 123 с.

Приложение А

(справочное)

Глоссарий

Здоровый образ жизни (ЗОЖ) – это такой образ жизни человека, который направлен на укрепление здоровья и профилактику болезней

UML (Unified Modeling Language) – язык графического описания для объектного моделирования в области разработки программного обеспечения, для моделирования бизнес-процессов, системного проектирования и отображения организационных структур.

JavaScript – мультипарадигменный язык программирования. Поддерживает объектно-ориентированный, императивный и функциональный стили.

React – JavaScript-библиотека с открытым исходным кодом для разработки пользовательских интерфейсов.

Asp Net – это веб-платформа с открытым исходным кодом, созданная Microsoft для создания современных веб-приложений и служб с помощью .NET.

SPA (Single Page Application) – это веб-приложения, которые загружают одну HTML-страницу и динамически обновляют ее при взаимодействии с пользователем.

MVC – это способ организации кода, который предполагает выделение блоков, отвечающих за решение разных задач. Один блок отвечает за данные приложения, другой отвечает за внешний вид, а третий контролирует работу приложения.

Микросервисный подход – это подход к созданию приложения в виде набора независимо развертываемых сервисов, которые являются децентрализованными и разрабатываются независимо друг от друга. Эти сервисы слабо связаны, независимо развертываются и легко обслуживаются.

API Gateway – обратный прокси-сервер, который перенаправляет запросы от клиентов к службам

REST API – это архитектурный стиль, который определяет правила обмена данными между клиентом и сервером.

Протокол HTTP – это протокол, позволяющий получать различные ресурсы, например HTML-документы.

API-интерфейсы – это совокупность инструментов и функций в виде интерфейса для создания новых приложений, благодаря которому одна программа будет взаимодействовать с другой.

JetBrains Rider – кроссплатформенная интегрированная среда разработки программного обеспечения для платформы .NET, разрабатываемая компанией JetBrains.

ORM Dapper – это продукт для объектно-реляционного сопоставления (ORM) на платформе Microsoft .NET.

Docker – это проект с открытым исходным кодом для автоматизации развертывания приложений в виде переносимых автономных контейнеров, выполняемых в облаке или локальной среде.

NUnit – открытая среда юнит-тестирования приложений для .NET.

Приложение Б

(рекомендуемое)

Атрибуты сущностей базы данных

Таблица Б.1 – Спецификации строк таблицы «PsychologicalTest

№ п/п	Наименование	Описание	Тип данных	Размер	Роль	Обязательный	Ограничения значений
1	ID	Уникальный идентификатор психологического теста	uuid	7	PK	Да	«00000000-0000-0000-00000000-zzzzzzzz-zzzz-zzzz-zzzz-zzzzzzzz»
2	Title	Название теста	varchar	255		Да	01.01.1950 - 01.01.2024
3	Description	Содержание теста	varchar	Не ограничен		Да	а-Я a-Z 0–9
4	Period	Период выполнения теста	varchar	255		Да	01.01.1950 - 01.01.2024

Таблица Б.2 – Спецификации строк таблицы «PsyhologicalTestQuestion»

№ п/п	Наименование	Описание	Тип данных	Размер	Роль	Обязательный	Ограничения значений
1	ID	Уникальный идентификатор вопроса психологического тестирования	uuid	7	PK	Да	«00000000-0000-0000-00000000-zzzzzzzz-zzzz-zzzz-zzzz-zzzzzzzz»
2	PsychologicalTestID	Идентификатор психологического теста, к которому привязан вопрос	uuid	7	FK	Да	«00000000-0000-0000-00000000-zzzzzzzz-zzzz-zzzz-zzzz-zzzzzzzz»
3	Title	Вопрос психологического теста	varchar	255		Да	01.01.1950 – 01.01.2024
4	MaxValue	Баллы за вопрос	Int4	4		Да	1 - 9999999

Таблица Б.3 – Спецификации строк таблицы «EvalutionTestResult»

№ п/п	Наименование	Описание	Тип данных	Размер	Роль	Обязательный	Ограничения значений
1	ID	Уникальный идентификатор оценки психологического тестирования	uuid	7	PK	Да	«00000000-0000-0000-00000000-zzzzzzzz-zzzz-zzzz-zzzz-zzzzzzzz»
2	PsychologicalTestID	Идентификатор психологического теста, к которому привязана оценка	uuid	7	FK	Да	«00000000-0000-0000-00000000-zzzzzzzz-zzzz-zzzz-zzzz-zzzzzzzz»
3	ToValue	Начало выполнения какой-либо задачи	Int4	4		Да	1 - 9999999
4	TotalResultValue	Завершение выполнения какой-либо задачи	varchar	255		да	01.01.1950 - 01.01.2024

Таблица Б.4 – Спецификации строк таблицы «UserPsychologicalTestResult»

№ п/п	Наименование	Описание	Тип данных	Размер	Роль	Обязательный	Ограничения значений
1	ID	Уникальный идентификатор ответов пользователя на психологическое тестирование	uuid	7	PK	да	«00000000-0000-0000-00000000-zzzzzzzz-zzzz-zzzz-zzzz-zzzzzzzz»
2	PsychologicalTestID	Идентификатор психологического теста, к которому привязаны ответы пользователя	uuid	7	FK	да	«00000000-0000-0000-00000000-zzzzzzzz-zzzz-zzzz-zzzz-zzzzzzzz»
3	UserID	Идентификатор пользователя, к которому привязаны ответы пользователя на психологический тест	uuid	7	FK	да	«00000000-0000-0000-00000000-zzzzzzzz-zzzz-zzzz-zzzz-zzzzzzzz»
4	QuestionNumber	Номер вопроса	Int4	4		да	1 - 9999999
5	AnswerValue	Ответ пользователя на вопрос	Int4	4		да	1 - 9999999

Таблица Б.5 – Спецификации строк таблицы «SleepSettings»

№ п/п	Наименование	Описание	Тип данных	Размер	Роль	Обязательный	Ограничения значений
1	ID	Уникальный идентификатор настройки сна	uuid	7	PK	да	«00000000-0000-0000-00000000-zzzzzzzz-zzzz-zzzz-zzzz-zzzzzzzz»
2	UserID	Идентификатор пользователя, к которому привязаны настройки сна	uuid	7	FK	да	«00000000-0000-0000-00000000-zzzzzzzz-zzzz-zzzz-zzzz-zzzzzzzz»
3	HourAmount	Количество часов, которые пользователь хотел бы уделять на сон	Float4	4		да	1 - 999999999
4	WishedWakeUpTime	Желательное пользователем время подъёма	timestamp	8		нет	01.01.1950 - 01.01.2024
5	WisheedTime	Желательное пользователем время отхода ко сну	timestamp	8		нет	01.01.1950 - 01.01.2024

Таблица Б.6 – Спецификации строк таблицы «SleepHistory»

№ п/п	Наименование	Описание	Тип данных	Размер	Роль	Обязательный	Ограничения значений
1	ID	Уникальный идентификатор истории сна	uuid	7	FK	Да	«00000000-0000-0000-00000000-zzzzzzzz-zzzz-zzzz-zzzz-zzzzzzzz»
2	UserID	Идентификатор пользователя, к которому привязана история сна	uuid	7	FK	Да	«00000000-0000-0000-00000000-zzzzzzzz-zzzz-zzzz-zzzz-zzzzzzzz»
3	CreatedDate	Дата фиксирования результатов сна	timestamp	8		Да	01.01.1950-01.01.2040
4	WishedWakeUpTime	Время подъёма пользователя	timestamp	8		Да	01.01.1950-01.01.2040
5	WisheedTime	Время отхода ко сну пользователя	timestamp	8		Да	01.01.1950-01.01.2040
6	SleepAssessment	Оценка пользователем качества сна	Int4	4		Нет	1 - 9999999

Таблица Б.7 – Спецификации строк таблицы «Habbit»

№ п/п	Наименование	Описание	Тип данных	Размер	Роль	Обязательный	Ограничения значений
1	ID	Уникальный идентификатор привычки	uuid	7	PK	Да	«00000000-0000-0000-00000000-zzzzzzzz-zzzz-zzzz-zzzz-zzzzzzzz»
2	UserID	Идентификатор пользователя, к которому привязана привычка	uuid	7	FK	Да	«00000000-0000-0000-00000000-zzzzzzzz-zzzz-zzzz-zzzz-zzzzzzzz»
3	Title	Название привычки	varchar	255		Да	01.01.1950 - 01.01.2024
4	Description	Описание привычки	varchar	не ограничена		Да	1 - 9999999

Таблица Б.8 – Спецификации строк таблицы «HabbitRegularityHistory»

№ п/п	Наименование	Описание	Тип данных	Размер	Роль	Обязательный	Ограничения значений
1	ID	Уникальный идентификатор контроля привычки	uuid	7	PK	Да	«00000000-0000-0000-00000000-zzzzzzzz-zzzz-zzzz-zzzz-zzzzzzzz»
2	HabbitID	Идентификатор привычки, к которому привязан контроль	uuid	7	FK	Да	«00000000-0000-0000-00000000-zzzzzzzz-zzzz-zzzz-zzzz-zzzzzzzz»
3	EventDate	Дата начала существования плана	timestamp	2		Да	

Таблица Б.9 – Спецификации строк таблицы «PhysicalActivitySettings»

№ п/п	Наименование	Описание	Тип данных	Размер	Роль	Обязательный	Ограничения значений
1	2	3	4	5	6	7	8
1	ID	Уникальный идентификатор настройки физической активности	uuid	7	PK	Да	«00000000-0000-0000-00000000-zzzzzzzz-zzzz-zzzz-zzzz-zzzzzzzz»

Продолжение таблицы Б.9

1	2	3	4	5	6	7	8
2	UserID	Идентификатор пользователя, к которому привязаны настройки физической активности	uuid	7	FK	Да	«00000000-0000-0000-00000000-zzzzzzzz-zzzz-zzzz-zzzz-zzzzzzzz»
3	HourAmount	Часовая норма на неделю, которую пользователь должен выделять на физическую активность	Int4	4		Да	1 - 9999999

Таблица Б.10 – Спецификации строк таблицы «PhysicalActivityHistory»

№ п/п	Наименование	Описание	Тип данных	Размер	Роль	Обязательный	Ограничения значений
1	ID	Уникальный идентификатор истории физической активности	uuid	7	PK	Да	«00000000-0000-0000-00000000-zzzzzzzz-zzzz-zzzz-zzzz-zzzzzzzz»
2	UserID	Идентификатор пользователя, к которому привязана история физической активности	uuid	7	FK	Да	«00000000-0000-0000-00000000-zzzzzzzz-zzzz-zzzz-zzzz-zzzzzzzz»
3	StartDate	Время начала физической активности	timestamp	8		Да	
4	EndDate	Время завершения физической активности	timestamp	8		Да	

Таблица Б.11 – Спецификации строк таблицы «User»

№ п/п	Наименование	Описание	Тип данных	Размер	Роль	Обязательный	Ограничения значений
1	2	3	4	5	6	7	8
1	ID	Уникальный идентификатор пользователя	uuid	7	PK	Да	«00000000-0000-0000-00000000-zzzzzzzz-zzzz-zzzz-zzzz-zzzzzzzz»

Продолжение таблицы Б.11

1	2	3	4	5	6	7	8
2	Login	Логин пользователя	varchar	255	FK	Да	01.01.1950 - 01.01.2024
3	PasswordHash	Пароль пользователя	varchar	255		Да	01.01.1950 - 01.01.2024
4	Email	Электронная почта пользователя	varchar	255		Да	01.01.1950 - 01.01.2024
5	Name	Имя пользователя	varchar	255		Да	01.01.1950 - 01.01.2024
6	Birthday	Дата дня рождения пользователя	timestamp	8		Да	01.01.1950 - 01.01.2024
7	Height	Рост пользователя	Int4	4		Да	1 - 9999999
8	Weight	Вес пользователя	Int4	4		Да	1 - 9999999

Таблица Б.12 – Спецификации строк таблицы «ReportFile»

№ п/п	Наименование	Описание	Тип данных	Размер	Роль	Обязательный	Ограничения значений
1	ID	Уникальный идентификатор информации о файле	uuid	7	PK	Да	«00000000-0000-0000-00000000-zzzzzzzz-zzzz-zzzz-zzzz-zzzzzzzz»
2	UserID	Идентификатор пользователя, к которому привязана информация о файле	uuid	7	FK	Да	«00000000-0000-0000-00000000-zzzzzzzz-zzzz-zzzz-zzzz-zzzzzzzz»
3	FileName	Название файла	varchar	255		Нет	а-Я a-Z 0-9
4	CreatedDate	Дата создания файла	timestamp	8		Нет	01.01.1950 - 01.01.2024

Приложение В

(рекомендуемое)

Графическая часть интерфейса программных средств

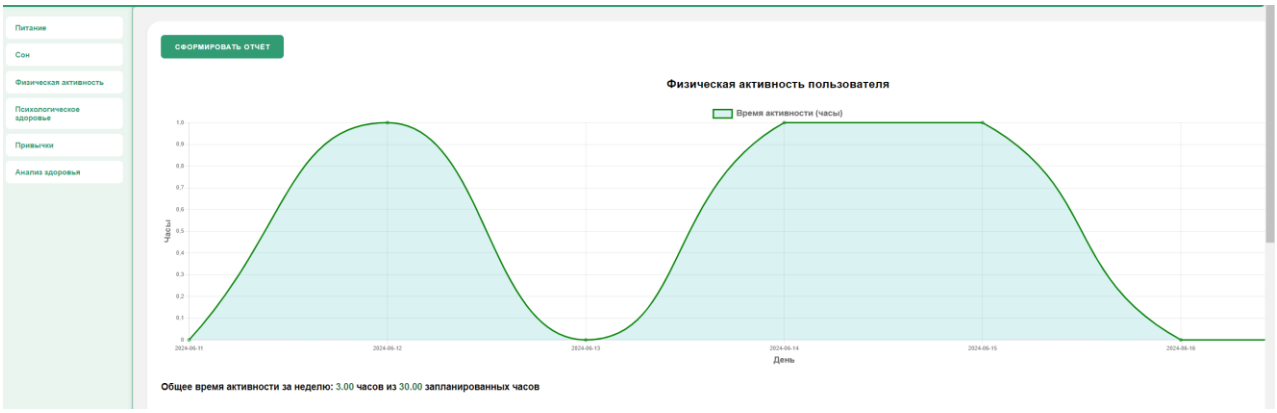


Рисунок В.1 – Аналитика физической активности

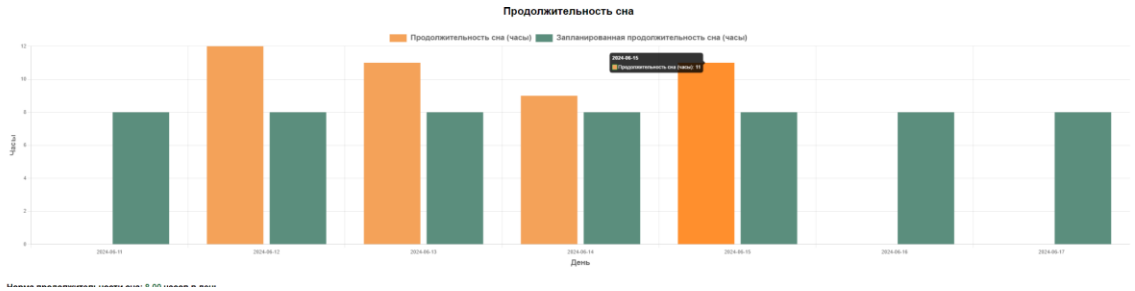


Рисунок В.2 – Аналитика продолжительности сна

Анализ эмоционального состояния

Дата	Название теста	Балл	Рекомендация
2024-06-17	Оценка эмоционального состояния на общем уровне	27	Уровень эмоционального состояния высокий. Поздравляем, ваше эмоциональное состояние расценивается как позитивное и стабильное.

Рисунок В.3 – Аналитика эмоционального состояния

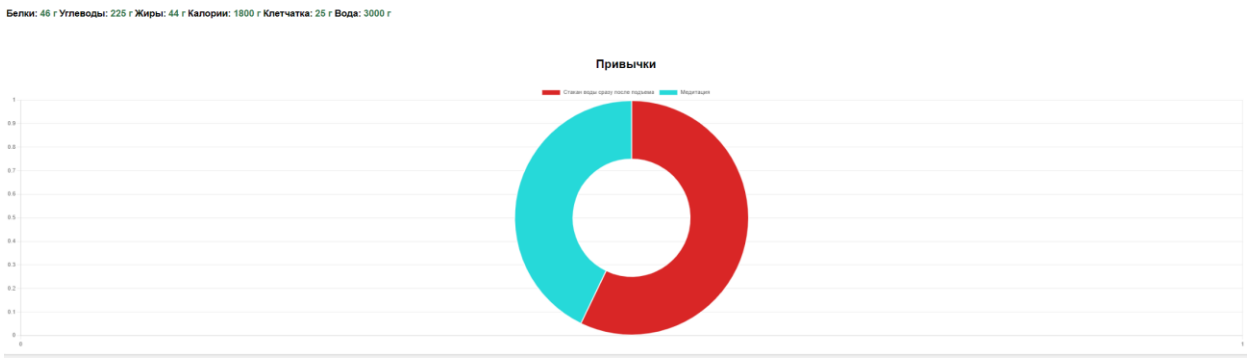


Рисунок В.4 – Аналитика выполнения привычек

Приложение Г

(рекомендуемое)

Программные компоненты серверной части

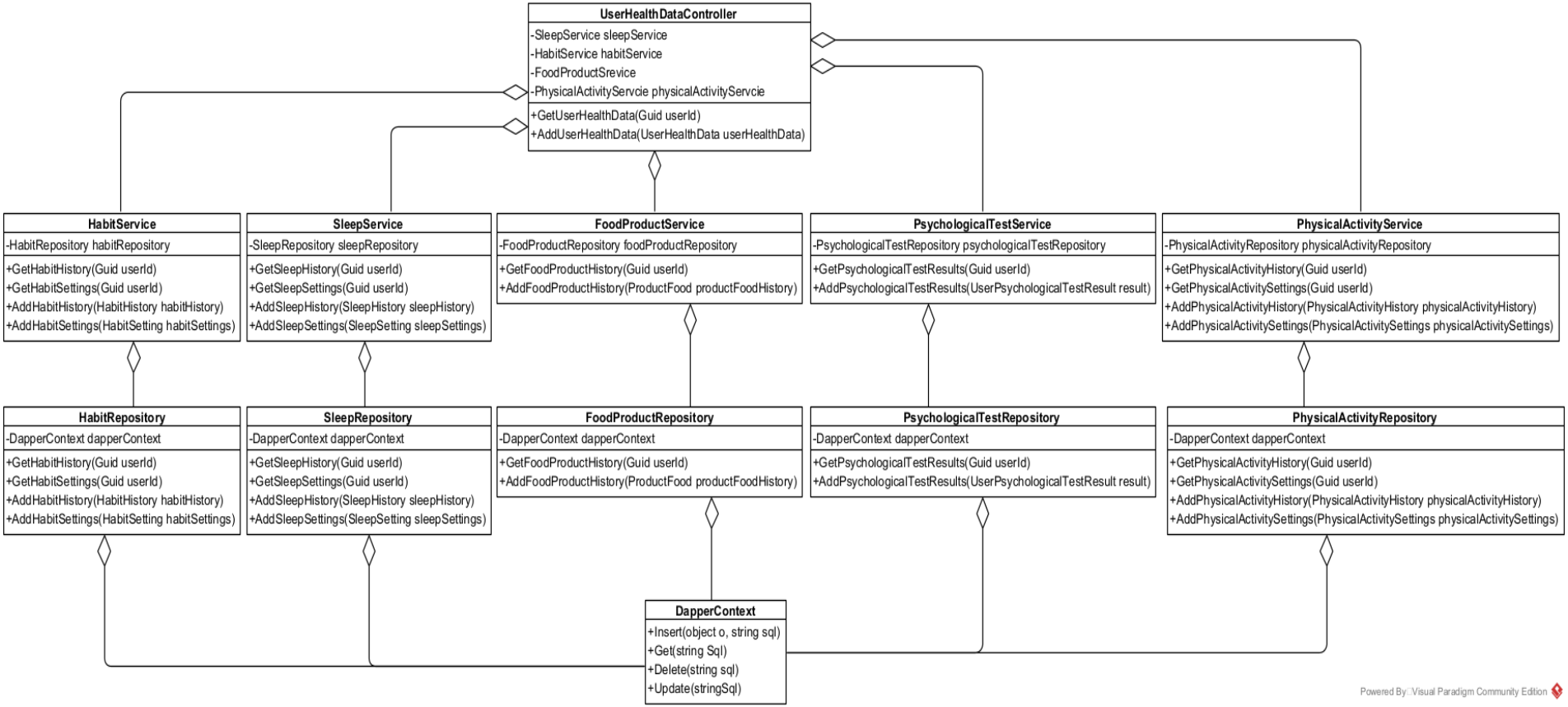


Рисунок Г.1 – Диаграмма классов микросервиса «Здоровье»

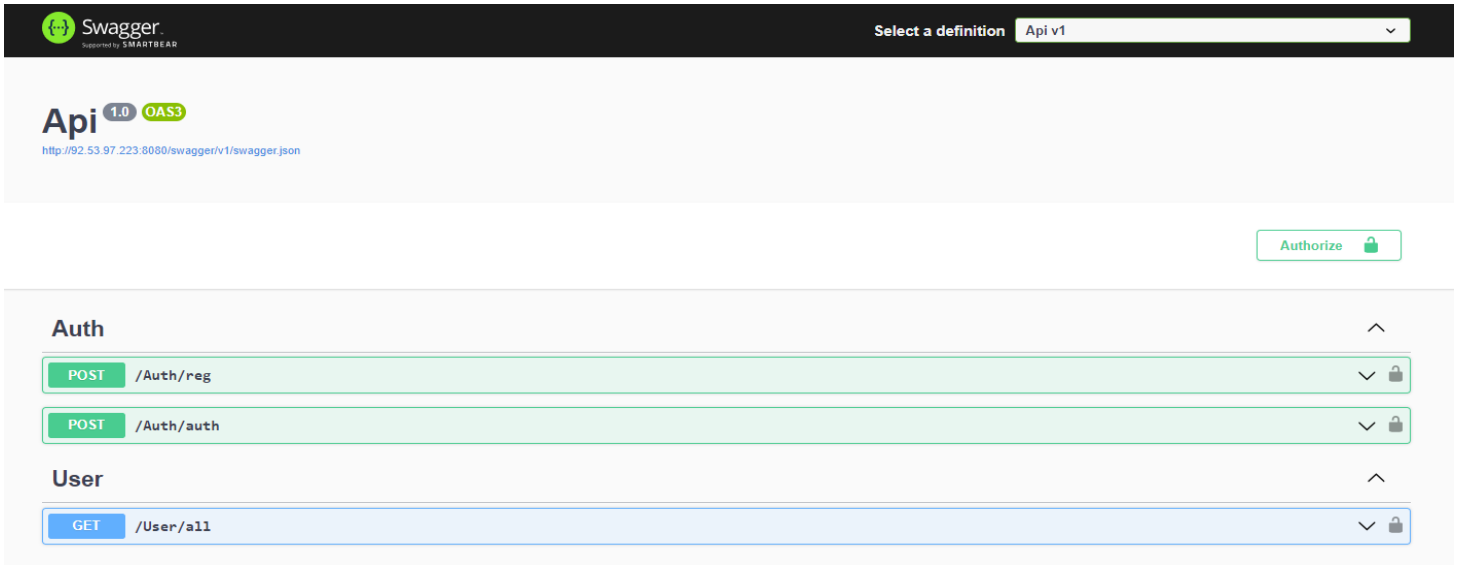


Рисунок Г.2 – Представление запросов API в Swagger для микросервиса «Авторизация»

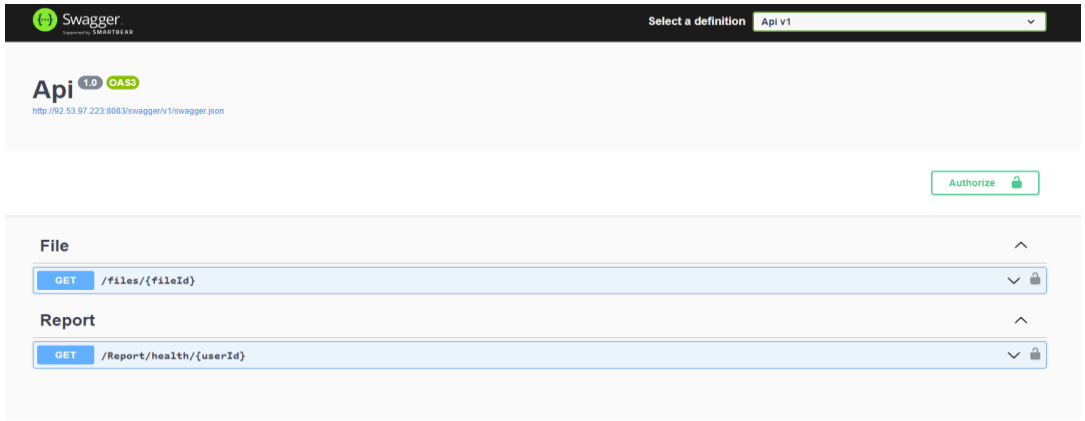


Рисунок Г.3 – Представление запросов API в Swagger для микросервиса «Отчетность

Приложение Д

(рекомендуемое)

Графическая часть оценки стоимости проекта

1		Начало проекта	0 дней	Пн 08.04.24	
2		**Анализ предметной области и постановка задачи на разработку серверного приложения**	**7,5 дней**	**Пн 08.04.24**	**Компьютеры[0]**
3		**Анализ программных средств для осознанного планирования жизни**	**1 день**	**Пн 08.04.24**	
4		Анализ предметной области	0,33 дней	Пн 08.04.24	Аналитик
5		Анализ свойств объектов управления и информационных объектов предметной области	0,34 дней	Пн 08.04.24	Аналитик
6		Анализ процессов управления и ведения данных в предметной области	0,33 дней	Пн 08.04.24	Аналитик
7		**Сравнительный анализ аналогичных разработок**	**1 день**	**Вт 09.04.24**	
8		Цели и задачи сравнительного анализа	0,25 дней	Вт 09.04.24	Аналитик
9		Выбор приложений-аналогов для анализа	0,25 дней	Вт 09.04.24	Аналитик
10		Обзор приложений-аналогов	0,25 дней	Вт 09.04.24	Аналитик
11		Результаты сравнительного анализа	0,25 дней	Вт 09.04.24	Аналитик
12		**Постановка задачи на разработку программных средств**	**4 дней**	**Ср 10.04.24**	
13		Цели разработки и назначение приложения	1 день	Ср 10.04.24	Программист 1[50%];Программист 2[50%]
14		Перечень функций, реализуемых в приложении	1 день	Чт 11.04.24	Программист 1
15		Требования к организации серверного и клиентской части приложения	1 день	Пт 12.04.24	Программист 1[50%];Программист 2[50%]
16		Постановка требований к интерфейсу программных средств	1 день	Сб 13.04.24	Программист 2
17		**Планирование и оценка процесса разработки приложения**	**1,5 дней**	**Вс 14.04.24**	
18		Выбор нотации и программных средств планирования и оценки процесса разработки	0,25 дней	Вс 14.04.24	Программист 1[50%];Программист 2[50%]
19		Планирование процесса разработки приложения	1 день	Вс 14.04.24	Аналитик
20		Оценка временных и финансовых затрат на разработку	0,25 дней	Пн 15.04.24	Аналитик
21		**Проектирование программных средств**	**6,5 дней**	**Пн 15.04.24**	**Интернет[200,00 ₽]**
22		Моделирование функциональных возможностей приложения	1,25 дней	Пн 15.04.24	Программист 2
23		Разработка единой концептуальной модели данных	2 дней	Вт 16.04.24	Разработчик базы данных
24		Обощенная архитектура программных средств для осознанного планирования жизни	2 дней	Чт 18.04.24	Программист 1
25		Требования к аппаратно-программному обеспечению клиентской части приложения и серверной	1 день	Пн 22.04.24	Программист 1[50%];Программист 2[50%]
26		Обоснование выбора технологий и средств разработки приложения	0,25 дней	Вт 23.04.24	Программист 1[50%];Программист 2[50%]
27		**Разработка программных средств для осознонного планирования жизни**	**28 дней**	**Ср 24.04.24**	**Интернет[350,00 ₽]**
28		Разработка архитектуры приложения	5 дней	Ср 24.04.24	Программист 1
29		Разработка физической модели базы данных	5 дней	Ср 01.05.24	Разработчик базы данных
30		**Разработка серверной части приложения**	**12 дней**	**Ср 08.05.24**	
31		Разработка API методов	3 дней	Ср 08.05.24	Программист 2
32		Разработка программного обеспечения микросервиса "Авторизация"	4 дней	Сб 11.05.24	Программист 1
33		Разработка программного обеспечения микросервиса "Здоровье"	8 дней	Сб 11.05.24	Программист 2
34		Разработка программного обеспечения микросервиса "Отчетность"	5 дней	Чт 16.05.24	Программист 1
35		Разработка пользовательского интерфейса	5 дней	Ср 24.04.24	Дизайнер;Программист 2[50%]
36		Разработка клиентской части приложения	7 дней	Вт 21.05.24	Программист 2
37		**Контроль качества программного обеспечения**	**8 дней**	**Ср 29.05.24**	**Интернет[200,00 ₽]**
38		Планирование процесса контроля качества программного обеспечения	2 дней	Ср 29.05.24	Тестировщик
39		Тестирование программных средств	3 дней	Пт 31.05.24	Тестировщик
40		Оценка качества программного кода	3 дней	Вт 04.06.24	Тестировщик
41		Сдача проекта	1 день	Пт 07.06.24	
42		Оформление и сбор документации	4 дней	Сб 08.06.24	Программист 2
43		Ввод в эксплуатацию	1 день	Чт 13.06.24	Программист 2
44		Конец проекта	0 дней	Чт 13.06.24	

Рисунок Д.1 – Список задач проекта

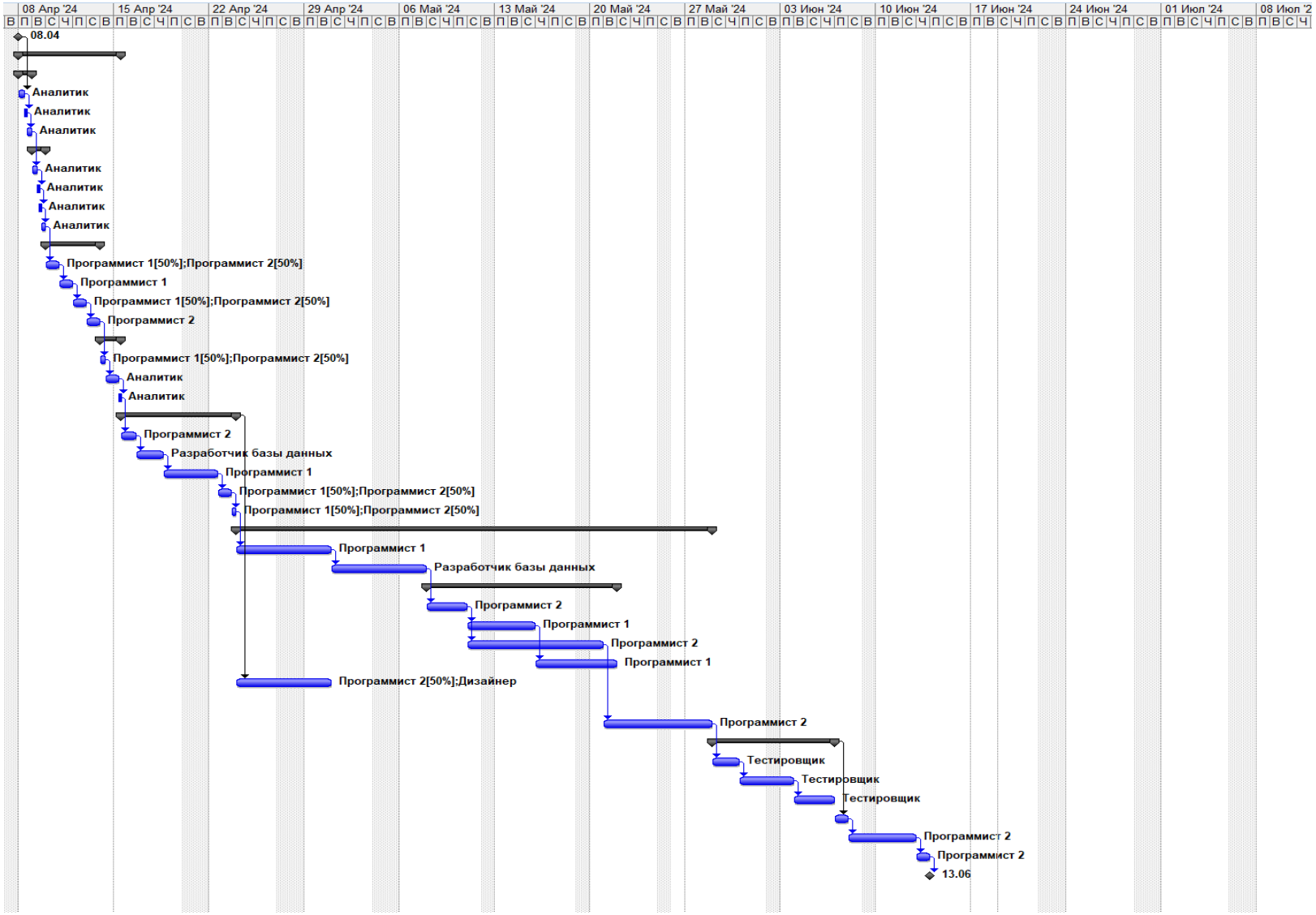


Рисунок Д.2 – Диаграмма Ганта

Приложение Е

(рекомендуемое)

Результаты тестирования клиентской части приложения

Рисунок Е.1 – Представление запросов API в Swagger для микросервиса «Здоровье»

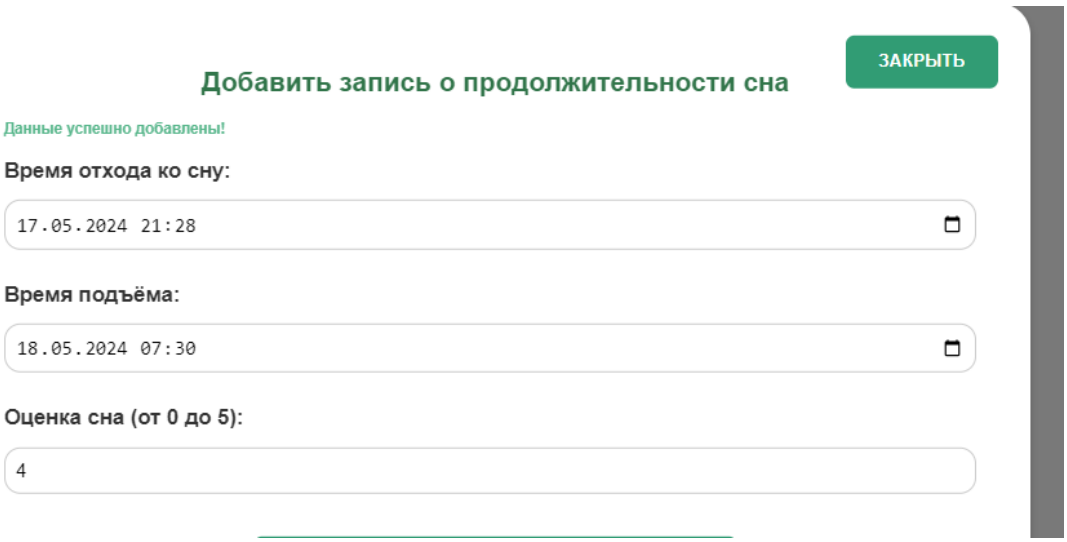


Рисунок Г.2 – Тест добавления записи продолжительности сна

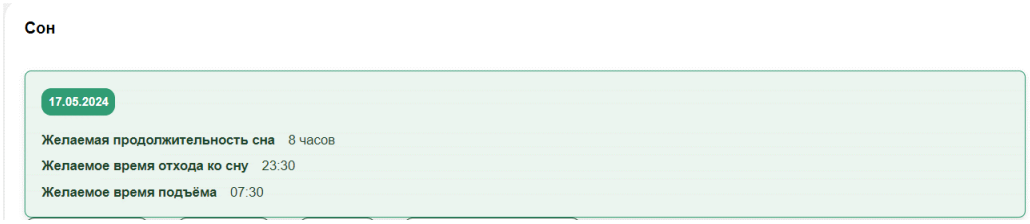


Рисунок Е.3 – Тест добавления настроек сна

Рисунок Е.4 – Тест добавления записи истории сна

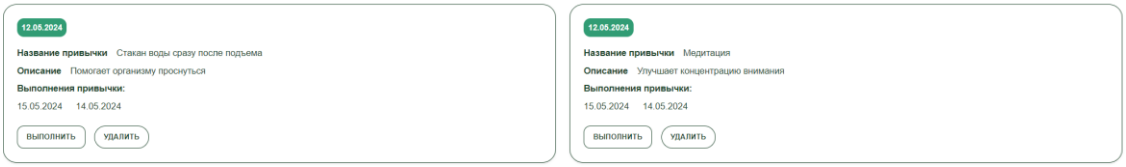


Рисунок Е.5 – Тест добавления записи выполнения привычек

Приложение Ж

(рекомендуемое)

Листинг серверной части программных средств

Листинг Ж.1 – Файл UserService.cs

```
using AutoMapper;
using Common.Extensions;
using DataAccess.Entity;
using DataAccess.Repositories.Interfaces;
using Services.Interfaces;
using Services.Models;

namespace Services;

public class UserService : IUserService
{
    private readonly IMapper _mapper;
    private readonly IUserRepository _userRepository;

    public UserService(IUserRepository userRepository, IMapper mapper)
    {
        _userRepository = userRepository;
        _mapper = mapper;
    }

    public async Task<User?> GetByLogin(string login)
    {
        return (await
_userRepository.GetByLogin(login)).Map<User>(_mapper);
    }

    public async Task<List<User>> GetAll()
    {
        return (await _userRepository.GetAll()).MapToList<User>(_mapper);
    }

    public async Task<User> Add(User user)
    {
        user.Id = Guid.NewGuid();
        user.PasswordHash = user.Password;

        return (await
_userRepository.Add(user.Map<DbUser>(_mapper))).Map<User>(_mapper);
    }
}
```

Листинг Ж.2 – Файл UserRepository.cs

```
using DataAccess.DataContexts.Interfaces;
using DataAccess.Entity;
using DataAccess.Repositories.Interfaces;
using DataAccess.Sql.User;

namespace DataAccess.Repositories;

public class UserRepository : IUserRepository
```

```
{
    private readonly IDataContext _dataContext;

    public UserRepository(IDataContext dataContext)
    {
        _dataContext = dataContext;
    }

    public async Task<IEnumerable<DbUser>> GetAll()
    {
        return await
_dataContext.EnumerableOrEmptyAsync<DbUser>(Users.GetAll, new { });
    }

    public async Task<DbUser> Add(DbUser model)
    {
        return await _dataContext.InsertAsync<DbUser>(Users.Insert,
model);
    }

    public Task<IEnumerable<DbUser>> GetByIds(IEnumerable<int> ids)
    {
        throw new NotImplementedException();
    }

    public Task<DbUser> GetById(int id)
    {
        throw new NotImplementedException();
    }

    public async Task<DbUser?> GetByLogin(string login)
    {
        return await
_dataContext.FirstOrDefaultAsync<DbUser>(Users.GetByLogin, new {login =
login});
    }
}
```

Листинг Ж.3 – Файл FileService.cs

```
using DataAccess.Entity;
using DataAccess.Repositories.Interfaces;
using Services.Interfaces;

namespace Services;

public class FileService(IFileRepository fileRepository) : IFileService
{
    public async Task<Guid> Save(string filename, byte[] content)
    {
        return (await fileRepository.Add(new DbFile
        {
            Id = Guid.NewGuid(),
            Filename = filename,
            Content = content,
```

```
            CreatedDate = DateTime.Now
        })).Id;
    }

    public async Task<DbFile?> Get(Guid id)
    {
        return await fileRepository.Get(id);
    }
}
```

Листинг Ж.4 – Файл ReportController.cs

```
using Microsoft.AspNetCore.Mvc;
using Services.Interfaces;

namespace Api.Controllers;

[ApiController]
[Route("[controller]")]
public class ReportController(IReportService reportService) : Controller
{
    [HttpGet("planning/{userId}")]
    public async Task<IActionResult> GetPlanningReport(Guid userId)
    {
        return Ok(await reportService.CreatePlanning(userId));
    }

    [HttpGet("health/{userId}")]
    public async Task<IActionResult> GetHealthReport(Guid userId)
    {
        return Ok(await reportService.CreateHealth(userId));
    }
}
```

Листинг Ж.5 – Файл FileModel.cs

```
namespace Services.Models;

public class FileModel
{
    public Guid Id { get; set; }
    public string Filename { get; set; }
    public string ContentType { get; set; }
    public byte[] Content { get; set; }
    public DateTime CreatedDate { get; set; }
}
```

Листинг Ж.6 – Файл User.cs

```
namespace Services.Models;

public class User
{
```

```
    public Guid Id { get; set; }
    public string Login { get; set; }
    public string Password { get; set; }
    public string PasswordHash { get; set; }
    public string Email { get; set; }
    public string Name { get; set; }
    public DateTime Birthday { get; set; }
    public int Height { get; set; }
    public int Weight { get; set; }
}
```

Листинг Ж.6 – Файл EvolutionPsychologicalTestResult.cs

```
namespace Services.Models.Health;

public class EvolutionPsychologicalTestResult
{
    public Guid Id { get; set; }
    public Guid PsychologicalTestId { get; set; }
    public int ToValue { get; set; }
    public string TotalResultValue { get; set; }
}
```

Листинг Ж.7 – Файл PhysicalActivityHistory.cs

```
namespace Services.Models.Health;

public class PhysicalActivityHistory
{
    public Guid Id { get; set; }
    public Guid UserId { get; set; }
    public DateTime CreatedDate { get; set; }
    public DateTime StartTime { get; set; }
    public DateTime EndTime { get; set; }
}
```

Листинг Ж.8 – Файл PsychologicalTestQuestion.cs

```
namespace Services.Models.Health;

public class PsychologicalTestQuestion
{
    public Guid Id { get; set; }
    public Guid PsychologicalTestId { get; set; }
    public string Title { get; set; }
    public int? MaxValue { get; set; } = 0;
}
```

Листинг Ж.9 – Файл UserHealthData.cs

```
namespace Services.Models.Health;

public class UserHealthData
```

```
{
    public Guid UserId { get; set; }
    public List<Habit> Habits { get; set; }
    public SleepSettings SleepSettings { get; set; }
    public List<SleepHistory> SleepHistories { get; set; }
    public PhysicalActivitySettings PhysicalActivitySettings { get; set;
}
    public List<PhysicalActivityHistory> PhysicalActivityHistories { get;
set; }
    public List<UserPsychologicalTestResult> UserPsychologicalTestResults
{ get; set; }
    public List<UserPsychologicalTestResultFull>
UserPsychologicalTestFullResults { get; set; }
}
```

Листинг Ж.10 – Файл PhysicalActivityHistoryService.cs

```
using AutoMapper;
using Common.Extensions;
using DataAccess.Entity;
using DataAccess.Repositories.Interfaces;
using Services.Interfaces;
using Services.Models;

namespace Services;

public class PhysicalActivityHistoryService(IMapper mapper,
IPhysicalActivityHistoryRepository physicalActivityHistoryRepository) :
IPhysicalActivityHistoryService
{
    public async Task<PhysicalActivityHistory>
Add(PhysicalActivityHistory model)
    {
        return (await
physicalActivityHistoryRepository.Add(model.Map<DbPhysicalActivityHistory
>(mapper))).Map<PhysicalActivityHistory>(mapper);
    }

    public async Task<PhysicalActivityHistory>
Update(PhysicalActivityHistory model)
    {
        return (await
physicalActivityHistoryRepository.Update(model.Map<DbPhysicalActivityHist
ory>(mapper))).Map<PhysicalActivityHistory>(mapper);
    }

    public async Task<bool> Delete(Guid id)
    {
        return await physicalActivityHistoryRepository.Delete(id);
    }

    public async Task<List<PhysicalActivityHistory>>
GetByIds(IEnumerable<Guid> ids)
    {
```

```
        return (await
physicalActivityHistoryRepository.GetByIds(ids)).MapToList<PhysicalActivi
tyHistory>(mapper);
    }

    public async Task<PhysicalActivityHistory> GetById(Guid id)
    {
        return (await physicalActivityHistoryRepository.GetByIds(new
List<Guid> {id})).Map<PhysicalActivityHistory>(mapper);
    }

    public async Task<List<PhysicalActivityHistory>> GetByUser(Guid
userId)
    {
        return (await physicalActivityHistoryRepository.GetByUser(new
List<Guid> {userId})).MapToList<PhysicalActivityHistory>(mapper);
    }

    public async Task<List<PhysicalActivityHistory>>
GetByUsers(List<Guid> userIds)
    {
        return (await
physicalActivityHistoryRepository.GetByUser(userIds)).MapToList<PhysicalA
ctivityHistory>(mapper);
    }
}
```

Листинг Ж.11 – Файл PsychologicalTestQuestionService.cs

```
using AutoMapper;
using Common.Extensions;
using DataAccess.Entity;
using DataAccess.Repositories.Interfaces;
using Services.Interfaces;
using Services.Models;

namespace Services;

public class PsychologicalTestQuestionService(IMapper mapper,
IPsychologicalTestQuestionRepository
userPsychologicalTestResultServiceRepository) :
IPsychologicalTestQuestionService
{
    public async Task<PsychologicalTestQuestion>
Add(PsychologicalTestQuestion model)
    {
        return (await
userPsychologicalTestResultServiceRepository.Add(model.Map<DbPsychologica
lTestQuestion>(mapper))).Map<PsychologicalTestQuestion>(mapper);
    }

    public async Task<PsychologicalTestQuestion>
Update(PsychologicalTestQuestion model)
    {
```

```
        return (await
userPsychologicalTestResultServiceRepository.Update(model.Map<DbPsycholog
icalTestQuestion>(mapper))).Map<PsychologicalTestQuestion>(mapper);
    }

    public async Task<bool> Delete(Guid id)
    {
        return await
userPsychologicalTestResultServiceRepository.Delete(id);
    }

    public async Task<List<PsychologicalTestQuestion>>
GetByIds(IEnumerable<Guid> ids)
    {
        return (await
userPsychologicalTestResultServiceRepository.GetByIds(ids)).MapToList<Psy
chologicalTestQuestion>(mapper);
    }

    public async Task<PsychologicalTestQuestion> GetById(Guid id)
    {
        return (await
userPsychologicalTestResultServiceRepository.GetByIds(new List<Guid>
{id})).Map<PsychologicalTestQuestion>(mapper);
    }

    public async Task<List<PsychologicalTestQuestion>>
GetByPsychologicalTests(List<Guid> psychologicalTestIds)
    {
        return (await
userPsychologicalTestResultServiceRepository.GetByPsychologicalTests(psyc
hologicalTestIds)).MapToList<PsychologicalTestQuestion>(mapper);
    }

    public async Task<List<PsychologicalTestQuestion>>
GetByPsychologicalTest(Guid psychologicalTestId)
    {
        return await GetByPsychologicalTests(new List<Guid>
{psychologicalTestId});
    }
}
```

Листинг Ж.12 – Файл SleepSettingsService.cs

```
using AutoMapper;
using Common.Extensions;
using DataAccess.Entity;
using DataAccess.Repositories.Interfaces;
using Services.Interfaces;
using Services.Models;

namespace Services;

public class SleepSettingsService(IMapper mapper,
ISleepSettingsRepository sleepSettingsRepository) : ISleepSettingsService
```

```
{
    public async Task<SleepSettings> Add(SleepSettings model)
    {
        return (await
sleepSettingsRepository.Add(model.Map<DbSleepSettings>(mapper))).Map<Slee
pSettings>(mapper);
    }

    public async Task<SleepSettings> Update(SleepSettings model)
    {
        return (await
sleepSettingsRepository.Update(model.Map<DbSleepSettings>(mapper))).Map<S
leepSettings>(mapper);
    }

    public async Task<bool> Delete(Guid id)
    {
        return await sleepSettingsRepository.Delete(id);
    }

    public async Task<List<SleepSettings>> GetByIds(IEnumerable<Guid>
ids)
    {
        return (await
sleepSettingsRepository.GetByIds(ids)).MapToList<SleepSettings>(mapper);
    }

    public async Task<SleepSettings> GetById(Guid id)
    {
        return (await sleepSettingsRepository.GetByIds(new List<Guid>
{id})).Map<SleepSettings>(mapper);
    }

    public async Task<List<SleepSettings>> GetByUser(Guid userId)
    {
        return await GetByUsers(new List<Guid> {userId});
    }

    public async Task<List<SleepSettings>> GetByUsers(List<Guid> userIds)
    {
        return (await
sleepSettingsRepository.GetByUser(userIds)).MapToList<SleepSettings>(mapp
er);
    }
}
```

Листинг Ж.13 – Файл EvolutionPsychologicalTestResultController.cs

```
using Api.Controllers.Interfaces;
using AutoMapper;
using Microsoft.AspNetCore.Mvc;
using Services.Interfaces;
using Services.Models;

namespace Api.Controllers;
```

```
[ApiController]
[Route("evolution-psychological-test-result")]
public class EvolutionPsychologicalTestResultController(IMapper mapper,
IEvolutionPsychologicalTestResultService
evolutionPsychologicalTestResultService)
    : Controller, ICrudController<EvolutionPsychologicalTestResult, Guid>
{
    private readonly IMapper _mapper = mapper;

    [HttpPost]
    public async Task<ActionResult<EvolutionPsychologicalTestResult>>
Add(EvolutionPsychologicalTestResult evolutionPsychologicalTestResult)
    {
        return Ok(await
evolutionPsychologicalTestResultService.Add(evolutionPsychologicalTestRes
ult));
    }

    [HttpGet("by-id/{id}")]
    public async Task<ActionResult<EvolutionPsychologicalTestResult>>
GetById(Guid id)
    {
        return await evolutionPsychologicalTestResultService.GetById(id);
    }

    [HttpPut]
    public async Task<ActionResult<EvolutionPsychologicalTestResult>>
Update(EvolutionPsychologicalTestResult model)
    {
        return Ok(await
evolutionPsychologicalTestResultService.Add(model));
    }

    [HttpDelete("{id}")]
    public async Task<ActionResult<EvolutionPsychologicalTestResult>>
Delete(Guid id)
    {
        return Ok(await
evolutionPsychologicalTestResultService.Delete(id));
    }
}
```

Листинг Ж.14 – Файл PhysicalActivityHistoryController.cs

```
using Api.Controllers.Interfaces;
using AutoMapper;
using Microsoft.AspNetCore.Mvc;
using Services.Interfaces;
using Services.Models;

namespace Api.Controllers;

[ApiController]
[Route("physical-activity-history")]
public class PhysicalActivityHistoryController(IMapper mapper,
IPhysicalActivityHistoryService physicalActivityHistoryService)
```

```
    : Controller, ICrudController<PhysicalActivityHistory, Guid>
{
    private readonly IMapper _mapper = mapper;

    [HttpPost]
    public async Task<ActionResult<PhysicalActivityHistory>>
Add(PhysicalActivityHistory physicalActivityHistory)
    {
        return Ok(await
physicalActivityHistoryService.Add(physicalActivityHistory));
    }

    [HttpGet("by-id/{id}")]
    public async Task<ActionResult<PhysicalActivityHistory>> GetById(Guid
id)
    {
        return await physicalActivityHistoryService.GetById(id);
    }

    [HttpPut]
    public async Task<ActionResult<PhysicalActivityHistory>>
Update(PhysicalActivityHistory model)
    {
        return Ok(await physicalActivityHistoryService.Add(model));
    }

    [HttpDelete("{id}")]
    public async Task<ActionResult<PhysicalActivityHistory>> Delete(Guid
id)
    {
        return Ok(await physicalActivityHistoryService.Delete(id));
    }
}
```

Листинг Ж.15 – Файл PsychologicalTestController.cs

```
using AutoMapper;
using Microsoft.AspNetCore.Mvc;
using Services.Interfaces;
using Services.Models;

namespace Api.Controllers;

[ApiController]
[Route("psychological-test")]
public class PsychologicalTestController(IMapper mapper,
IPsychologicalTestService psychologicalTestService) : Controller
{
    [HttpGet("all")]
    public async Task<ActionResult<PsychologicalTest>> GetAll()
    {
        return Ok(await psychologicalTestService.GetAll());
    }

    [HttpPost]
```

```
    public async Task<ActionResult<PsychologicalTest>>
Add(PsychologicalTest physicalActivitySettings)
    {
        return Ok(await
psychologicalTestService.Add(physicalActivitySettings));
    }

    [HttpGet("by-id/{id}")]
    public async Task<ActionResult<PsychologicalTest>> GetById(Guid id)
    {
        return await psychologicalTestService.GetById(id);
    }

    [HttpPut]
    public async Task<ActionResult<PsychologicalTest>>
Update(PsychologicalTest model)
    {
        return Ok(await psychologicalTestService.Add(model));
    }

    [HttpDelete("{id}")]
    public async Task<ActionResult<PsychologicalTest>> Delete(Guid id)
    {
        return Ok(await psychologicalTestService.Delete(id));
    }
}
```

Листинг Ж.16 – Файл SleepSettingsController.cs

```
using Api.Controllers.Interfaces;
using AutoMapper;
using Microsoft.AspNetCore.Mvc;
using Services.Interfaces;
using Services.Models;

namespace Api.Controllers;

[ApiController]
[Route("sleep-settings")]
public class SleepSettingsController(IMapper mapper,
ISleepSettingsService sleepSettingsService) : Controller,
ICrudController<SleepSettings, Guid>
{
    private readonly IMapper _mapper = mapper;

    [HttpPost]
    public async Task<ActionResult<SleepSettings>> Add(SleepSettings
sleepSettings)
    {
        return Ok(await sleepSettingsService.Add(sleepSettings));
    }

    [HttpGet("by-id/{id}")]
    public async Task<ActionResult<SleepSettings>> GetById(Guid id)
    {
```

```
        return await sleepSettingsService.GetById(id);
    }

    [HttpPut]
    public async Task<ActionResult<SleepSettings>> Update(SleepSettings
model)
    {
        return Ok(await sleepSettingsService.Add(model));
    }

    [HttpDelete("{id}")]
    public async Task<ActionResult<SleepSettings>> Delete(Guid id)
    {
        return Ok(await sleepSettingsService.Delete(id));
    }
}
```

Листинг Ж.17 – Файл EvolutionPsychologicalTestResultRepository.cs

```
using DataAccess.DataContexts.Interfaces;
using DataAccess.Entity;
using DataAccess.Repositories.Interfaces;
using DataAccess.Sql.EvolutionPsychologicalTestResult;

namespace DataAccess.Repositories;

public class EvolutionPsychologicalTestResultRepository(IDataContext
dataContext) : IEvolutionPsychologicalTestResultRepository
{
    public async Task<DbEvolutionPsychologicalTestResult>
Add(DbEvolutionPsychologicalTestResult model)
    {
        model.Id = Guid.NewGuid();

        return await
dataContext.InsertAsync<DbEvolutionPsychologicalTestResult>(EvolutionPsyc
hologicalTestResult.Insert, model);
    }

    public async Task<DbEvolutionPsychologicalTestResult>
Update(DbEvolutionPsychologicalTestResult model)
    {
        return await
dataContext.UpdateAsync<DbEvolutionPsychologicalTestResult>(EvolutionPsyc
hologicalTestResult.Update, model);
    }

    public async Task<bool> Delete(Guid id)
    {
        return await
dataContext.DeleteAsync(EvolutionPsychologicalTestResult.Delete, new
{id});
    }
```

```
    public async Task<IEnumerable<DbEvolutionPsychologicalTestResult>>
GetByIds(IEnumerable<Guid> ids)
    {
        return await
dataContext.EnumerableOrEmptyAsync<DbEvolutionPsychologicalTestResult>(Ev
olutionPsychologicalTestResult.GetByIds, new {ids});
    }

    public async Task<IEnumerable<DbEvolutionPsychologicalTestResult>>
GetByPsychologicalTests(List<Guid> psychologicalTestIds)
    {
        return await
dataContext.EnumerableOrEmptyAsync<DbEvolutionPsychologicalTestResult>(Ev
olutionPsychologicalTestResult.GetAllByPsychologicalTests, new
{psychologicalTestIds});
    }
}
```

Листинг Ж.18 – Файл PhysicalActivityHistoryRepository.cs

```
using DataAccess.DataContexts.Interfaces;
using DataAccess.Entity;
using DataAccess.Repositories.Interfaces;
using DataAccess.Sql.PhysicalActivityHistory;

namespace DataAccess.Repositories;

public class PhysicalActivityHistoryRepository(IDataContext dataContext)
: IPhysicalActivityHistoryRepository
{
    public async Task<DbPhysicalActivityHistory>
Add(DbPhysicalActivityHistory model)
    {
        model.Id = Guid.NewGuid();

        return await
dataContext.InsertAsync<DbPhysicalActivityHistory>(PhysicalActivityHistor
y.Insert, model);
    }

    public async Task<DbPhysicalActivityHistory>
Update(DbPhysicalActivityHistory model)
    {
        return await
dataContext.UpdateAsync<DbPhysicalActivityHistory>(PhysicalActivityHistor
y.Update, model);
    }

    public async Task<bool> Delete(Guid id)
    {
        return await
dataContext.DeleteAsync(PhysicalActivityHistory.Delete, new {id});
    }
```

```
    public async Task<IEnumerable<DbPhysicalActivityHistory>>
GetByIds(IEnumerable<Guid> ids)
    {
        return await
dataContext.EnumerableOrEmptyAsync<DbPhysicalActivityHistory>(PhysicalAct
ivityHistory.GetByIds, new {ids});
    }

    public async Task<IEnumerable<DbPhysicalActivityHistory>>
GetByUser(List<Guid> userIds)
    {
        return await
dataContext.EnumerableOrEmptyAsync<DbPhysicalActivityHistory>(PhysicalAct
ivityHistory.GetAllByUser, new {userIds});
    }
}
```

Листинг Ж.19 – Файл PsychologicalTestQuestionRepository.cs

```
using DataAccess.DataContexts.Interfaces;
using DataAccess.Entity;
using DataAccess.Repositories.Interfaces;
using DataAccess.Sql.PsychologicalTestQuestion;

namespace DataAccess.Repositories;

public class PsychologicalTestQuestionRepository(IDataContext
dataContext) : IPsychologicalTestQuestionRepository
{
    public async Task<DbPsychologicalTestQuestion>
Add(DbPsychologicalTestQuestion model)
    {
        model.Id = Guid.NewGuid();

        return await
dataContext.InsertAsync<DbPsychologicalTestQuestion>(PsychologicalTestQue
stion.Insert, model);
    }

    public async Task<DbPsychologicalTestQuestion>
Update(DbPsychologicalTestQuestion model)
    {
        return await
dataContext.UpdateAsync<DbPsychologicalTestQuestion>(PsychologicalTestQue
stion.Update, model);
    }

    public async Task<bool> Delete(Guid id)
    {
        return await
dataContext.DeleteAsync(PsychologicalTestQuestion.Delete, new {id});
    }

    public async Task<IEnumerable<DbPsychologicalTestQuestion>>
GetByIds(IEnumerable<Guid> ids)
```

```
    {
        return await
dataContext.EnumerableOrEmptyAsync<DbPsychologicalTestQuestion>(Psycholog
icalTestQuestion.GetByIds, new {ids});
    }

    public async Task<IEnumerable<DbPsychologicalTestQuestion>>
GetByPsychologicalTests(List<Guid> psychologicalTestIds)
    {
        return await
dataContext.EnumerableOrEmptyAsync<DbPsychologicalTestQuestion>(Psycholog
icalTestQuestion.GetByPsychologicalTest, new {psychologicalTestIds});
    }
}
```

Листинг Ж.20 – Файл SleepSettingsRepository.cs

```
using DataAccess.DataContexts.Interfaces;
using DataAccess.Entity;
using DataAccess.Repositories.Interfaces;
using DataAccess.Sql.SleepSettings;

namespace DataAccess.Repositories;

public class SleepSettingsRepository(IDataContext dataContext) :
ISleepSettingsRepository
{
    public async Task<DbSleepSettings> Add(DbSleepSettings model)
    {
        model.Id = Guid.NewGuid();

        return await
dataContext.InsertAsync<DbSleepSettings>(SleepSettings.Insert, model);
    }

    public async Task<DbSleepSettings> Update(DbSleepSettings model)
    {
        return await
dataContext.UpdateAsync<DbSleepSettings>(SleepSettings.Update, model);
    }

    public async Task<bool> Delete(Guid id)
    {
        return await dataContext.DeleteAsync(SleepSettings.Delete, new
{id});
    }

    public async Task<IEnumerable<DbSleepSettings>>
GetByIds(IEnumerable<Guid> ids)
    {
        return await
dataContext.EnumerableOrEmptyAsync<DbSleepSettings>(SleepSettings.GetById
s, new {ids});
    }
```

```
    public async Task<IEnumerable<DbSleepSettings>> GetByUser(List<Guid>
userIds)
    {
        return await
dataContext.EnumerableOrEmptyAsync<DbSleepSettings>(SleepSettings.GetAllB
yUser, new {userIds});
    }
}
```

Листинг Ж.21 – Файл EvolutionPsychologicalTestResult.cs

```
namespace Services.Models;

public class EvolutionPsychologicalTestResult
{
    public Guid Id { get; set; }
    public Guid PsychologicalTestId { get; set; }
    public int ToValue { get; set; }
    public string TotalResultValue { get; set; }
}
```

Листинг Ж.22 – Файл PhysicalActivityHistory.cs

```
namespace Services.Models;

public class PhysicalActivityHistory
{
    public Guid Id { get; set; }
    public Guid UserId { get; set; }
    public DateTime CreatedDate { get; set; }
    public DateTime StartTime { get; set; }
    public DateTime EndTime { get; set; }
}
```

Листинг Ж.23 – Файл PsychologicalTest.cs

```
namespace Services.Models;

public class PsychologicalTest
{
    public Guid Id { get; set; }
    public string Title { get; set; }
    public string Description { get; set; }
    public string Period { get; set; }
    public List<PsychologicalTestQuestion>? Questions { get; set; }
    public List<EvolutionPsychologicalTestResult>? EvolutionTestResults {
get; set; }
}
```

Листинг Ж.24 – Файл SleepSettings.cs

```
namespace Services.Models;

public class SleepSettings
{
    public Guid Id { get; set; }
    public Guid UserId { get; set; }
    public float HourAmount { get; set; }
    public DateTime CreatedDate { get; set; }
    public TimeSpan WishedBedTime { get; set; }
    public TimeSpan WishedWakeUpTime { get; set; }
}
```

Листинг Ж.25 – Файл UserPsychologicalTestResult.cs

```
namespace Services.Models;

public class UserPsychologicalTestResult
{
    public Guid Id { get; set; }
    public Guid PsychologicalTestId { get; set; }
    public Guid UserId { get; set; }
    public DateTime CreatedDate { get; set; }
    public int SleepAssessment { get; set; }
    public int QuestionNumber { get; set; }
    public int AnswerValue { get; set; }
}
```

Приложение И

(рекомендуемое)

Листинг клиентской части программных средств

Листинг И.1 – Файл HealthPage.jsx

```
import React from 'react';
import Nutrition from './components/health/Nutrition';
import Sleep from './components/health/Sleep';
import Habbit from './components/health/Habbit';
import Sport from './components/health/Sport';
import Psychology from './components/health/Psychology';
import Analize_H from './components/health/Analize_H';
import { Routes, Route, Link } from 'react-router-dom';
import React, { useEffect } from 'react';
import axios from 'axios';
import healthDataUpdate from './components/health/healthDataUpdate';

function Health() {

    const userDataString = localStorage.getItem('user');
    const userData = userDataString ? JSON.parse(userDataString) : null;
    const user = userData ? userData.user : null;

    healthDataUpdate(user.id);

    return (
        <div className="content_col">
            <div className="leftnav">
                <Link to="nutrition">Питание</Link>
                <Link to="sleep">Сон</Link>
                <Link to="sport">Физическая активность</Link>
                <Link to="psychosocial">Психологическое здоровье</Link>
                <Link to="habbits">Привычки</Link>
                <Link to="analize_health">Анализ здорового образа
жизни</Link>
            </div>
            <div className="content_razdel">
                <div className="content_razdel_style">
                    <Routes>
                        <Route path="nutrition" element={<Nutrition />}
/>
                        <Route path="sleep" element={<Sleep />} />
                        <Route path="sport" element={<Sport />} />
                        <Route path="psychosocial" element={<Psychology
/>} />
                        <Route path="habbits" element={<Habbit />} />
                        <Route path="analize_health" element={<Analize_H
/>} />
                    </Routes>
                </div>
            </div>
        </div>
    );
}
export default Health;
```

Листинг И.2 – Файл MainPage.jsx

```
import React from 'react';
import './css/main.css';
```

```
import Health from './HealthPage';
import Planning from './PlanningPage';
import Profile from './ProfilePage';
import Logout from './LogoutPage';
import { Routes, Route, Link } from 'react-router-dom';


const MainPage = () => {
    return (
        <div>
            <div className="topnav">
                <Link to="health">Здоровье</Link>
                <Link to="planning">Планирование</Link>
                <Link to="profile">Личный кабинет</Link>
                <Link to="logout">Выход</Link>
            </div>
            <div className="content">
                <Routes>
                    <Route path="health/*" element={<Health />} />
                    <Route path="planning" element={<Planning />} />
                    <Route path="profile" element={<Profile />} />
                    <Route path="logout" element={<Logout />} />
                </Routes>
            </div>
        </div>
    );
}

export default MainPage;
```

Листинг И.3 – Файл RegPage.jsx

```
import React, { useState } from 'react';
import axios from 'axios';

function RegPage() {
    const [login, setLogin] = useState('');
    const [password, setPassword] = useState('');
    const [name, setName] = useState('');
    const [email, setEmail] = useState('');
    const [birthday, setBirthday] = useState('');
    const [errorMessage, setErrorMessage] = useState('');
    const [height, setHeight] = useState(0);
    const [weight, setWeight] = useState(0);

    const handleSubmit = async (event) => {
        event.preventDefault();
        // Преобразовываем строковые значения в числа
        const numericHeight = Number(height);
        const numericWeight = Number(weight);

        // Проверка, что поля не пустые
        if (!login || !password || !name || !email || !birthday ||
!numericHeight || !numericWeight) {
            setErrorMessage("Все поля должны быть заполнены!");
```

```
            return;
        }

        const userData = {
            Login: login,
            Password: password,
            Name: name,
            Email: email,
            Birthday: birthday,
            Height: numericHeight,
            Weight: numericWeight
        };

        try {

            const withHash = false;
            // Используйте обратные кавычки для создания шаблонной строки
            const url = `/api1/auth/reg/?withHash=${withHash}`;
            console.log("URL being used:", url);

            axios.post(url, userData, {
                headers: {
                    'Content-Type': 'application/json'
                }
            }).then(response => {
                setErrorMessage('Вы успешно зарегистрированы!');
                navigate('/auth');
            }).catch(error => {
                console.error('Ошибка при отправке запроса:', error);
                if (error.response && error.response.data) {
                    // Может содержать более конкретную информацию об
ошибке
                    console.error('Детали ошибки:', error.response.data);
                    setErrorMessage(error.response.data.message ||
'Ошибка при отправке запроса');
                } else {
                    setErrorMessage('Ошибка при отправке запроса');
                }
            });
        } catch (error) {
            console.error('Ошибка при отправке запроса:', error);
            setErrorMessage('Ошибка при отправке запроса');
        }
    };



    return (
        <div className="reg-page">
            <h1>Регистрация</h1>
            {errorMessage && <p className="error">{errorMessage}</p>}
            <form onSubmit={handleSubmit}>
                <div>
                    <label htmlFor="login">Логин:</label>
                    <input
                        type="text"
                        id="login"
                        value={login}
```

```
            onChange={(e) => setLogin(e.target.value)}
        />
    </div>
    <div>
        <label htmlFor="password">Пароль:</label>
        <input
            type="password"
            id="password"
            value={password}
            onChange={(e) => setPassword(e.target.value)}
        />
    </div>
    <div>
        <label htmlFor="name">Имя:</label>
        <input
            type="text"
            id="name"
            value={name}
            onChange={(e) => setName(e.target.value)}
        />
    </div>
    <div>
        <label htmlFor="email">Email:</label>
        <input
            type="email"
            id="email"
            value={email}
            onChange={(e) => setEmail(e.target.value)}
        />
    </div>
    <div>
        <label htmlFor="birthday">Дата рождения:</label>
        <input
            type="date"
            id="birthday"
            value={birthday}
            onChange={(e) => setBirthday(e.target.value)}
        />
    </div>
    <div>
        <label htmlFor="height">Рост (см):</label>
        <input
            type="number"
            id="height"
            value={height}
            onChange={(e) => setHeight(e.target.value)}
        />
    </div>
    <div>
        <label htmlFor="weight">Вес (кг):</label>
        <input
            type="number"
            id="weight"
            value={weight}
            onChange={(e) => setWeight(e.target.value)}
        />
    </div>
    <button type="submit">Зарегистрироваться</button>
```

```
            </form>
        </div>
    );
}

export default RegPage;
```

Листинг И.4 – Файл Habbit.jsx

```
import React, { useState, useEffect } from 'react';
import axios from 'axios';
import Modal from '../Modal';
import healthDataUpdate from './healthDataUpdate';
import HabbitDataComponent from './HabbitDataComponent';

function Habbit() {
    const userDataString = localStorage.getItem('user');
    const userData = userDataString ? JSON.parse(userDataString) : null;
    const user = userData ? userData.user : null;

    const [isOpen, setIsOpen] = useState(false);
    const [isOpenTwo, setIsOpenTwo] = useState(false);

    const [title, setTitle] = useState('');
    const [description, setDescription] = useState('');
    const [errorMessage, setErrorMessage] = useState('');

    const [habbitDataUpdated, setHabbitDataUpdated] = useState(false);

    const [healthData, setHealthData] =
useState(JSON.parse(localStorage.getItem('userHealthData')));

    const closeModal = () => {
        setIsOpen(false);
    };

    const closeModalTwo = () => {
        setIsOpenTwo(false);
    };

    useEffect(() => {
        if (isOpen || isOpenTwo) {
            setHabbitDataUpdated(false); // Сбрасываем флаг при открытии
модального окна
        } else {
            setHabbitDataUpdated(true); // Выставляем обновленное
значение при закрытии модального окна
        }
    }, [isOpen, isOpenTwo]);

    const openModal = () => {
        setIsOpen(true);
    };

    const openModalTwo = () => {
        setIsOpenTwo(true);
```

```
    };


    const handleSubmit = async (event) => {
        event.preventDefault();

        if (!title || !description) {
            setErrorMessage("Все поля должны быть заполнены!");
            return;
        }

        const userHabbitData = {
            UserId: user.id,
            CreatedDate: new Date(),
            Title: title,
            Description: description
        };

        try {
            const url = `/api3/habit`;

            axios.post(url, userHabbitData, {
                headers: {
                    'Content-Type': 'application/json'
                }
            }).then(response => {
                setErrorMessage('Данные успешно добавлены!');
                healthDataUpdate(user.id);
            }).catch(error => {
                console.error('Ошибка при отправке запроса:', error);
                if (error.response && error.response.data) {
                    console.error('Детали ошибки:', error.response.data);
                    setErrorMessage(error.response.data.message ||
'Ошибка при отправке запроса');
                } else {
                    setErrorMessage('Ошибка при отправке запроса');
                }
            });
        } catch (error) {
            console.error('Ошибка при отправке запроса:', error);
            setErrorMessage('Ошибка при отправке запроса');
        }
    };


    return (
        <div>
            <h2>Привычки</h2>
            <div className='sleep-row-btn habbit-row-btn'>
                <button onClick={openModal}>Добавить привычку</button>
                {isOpen && (
                    <Modal onClose={closeModal}>
                        <h2>Добавить привычку</h2>
                        {errorMessage && <p
classNam="error">{errorMessage}</p>}
                        <form onSubmit={handleSubmit}>
                            <div>
```

```
                                <label htmlFor="title">Название
привычки:</label>
                                <input
                                    type="text"
                                    id="title"
                                    value={title}
                                    onChange={(e) =>
setTitle(e.target.value)}
                                />
                            </div>
                            <div>
                                <label htmlFor="wakeUpTime">Описание
привычки:</label>
                                <input
                                    type="text"
                                    id="description"
                                    value={description}
                                    onChange={(e) =>
setDescription(e.target.value)}
                                />
                            </div>
                            <button type="submit">Добавить</button>
                        </form>
                    </Modal>
                )}
            </div>
            <div className='history'>
                {habbitDataUpdated && <HabbitDataComponent />}
            </div>
        </div>
    );
};

export default Habbit;
```

Листинг И.5 – Файл Nutrition.jsx

```
import React, { useState, useEffect } from 'react';
import Modal from '../Modal';
import { v4 as uuidv4 } from 'uuid';

function formatDate(dateString) {
    const options = { year: 'numeric', month: 'numeric', day: 'numeric'
};
    return new Date(dateString).toLocaleDateString('ru-RU', options);
}

function roundToTwo(num) {
    return Math.round(num * 100) / 100;
}

function Nutrition() {
    const [isOpen, setIsOpen] = useState(false);
    const [mealType, setMealType] = useState('');
    const [dishName, setDishName] = useState('');
    const [dishKol, setDishKol] = useState([]);
```

```
    const [belki, setBelki] = useState('');
    const [uglevody, setUglevody] = useState('');
    const [kalorii, setKalorii] = useState([]);
    const [kletchatka, setKletchatka] = useState('');
    const [zhiry, setZhiry] = useState('');
    const [water, setWater] = useState([]);
    const [meals, setMeals] = useState([]);
    const [filter, setFilter] = useState([]);
    const [dateRange, setDateRange] = useState({ startDate: new Date(),
endDate: new Date() });
    const [filteredHistories, setFilteredHistories] = useState([]);
    const [mealsHistories, setMealsHistories] = useState([]);

    const mealTypes = ['Завтрак', 'Обед', 'Ужин', 'Перекус'];
    const dishes = ['Куриное филе', 'Овсяная каша на воде', 'Яблоки
свежие', 'Болгарский перец', 'Творог (5% жирности)',
        'Лосось запеченный', 'Омлет', 'Картофельное пюре', 'Шпинат'];

    const filterHistories = () => {
        const today = new Date();
        today.setHours(0, 0, 0, 0);
        const yesterday = new Date(today);
        yesterday.setDate(yesterday.getDate() - 1);

        switch (filter) {
            case 'today':
                return mealsHistories.filter(history => new
Date(history.createdDate).setHours(0, 0, 0, 0) === today.getTime());
            case 'yesterday':
                return mealsHistories.filter(history => new
Date(history.createdDate).setHours(0, 0, 0, 0) === yesterday.getTime());
            case 'range':
                return mealsHistories.filter(history => {
                    const historyDate = new Date(history.createdDate);
                    return historyDate >= dateRange.startDate &&
historyDate <= dateRange.endDate;
                });
            case 'all':
            default:
                return mealsHistories;
        }
    };

    useEffect(() => {
        const updateMealsData = () => {
            const storedData = localStorage.getItem('userMeals');
            if (storedData) {
                const parsedData = JSON.parse(storedData);
                if (Array.isArray(parsedData) && parsedData.length > 0) {
                    var userMeals =
JSON.parse(localStorage.getItem('userMeals')) || [];
                    const histories = userMeals || []; // добавлено
"|| []" для обеспечения пустого массива по умолчанию
                    setMealsHistories(histories);
                } else {
                    console.log('Parsed data is not an array or is
empty');
                }
```

```
            } else {
                console.log('No stored data found in localStorage');
            }
        };

        const handleStorageChange = (event) => {
            if (event.key === 'userMeals') {
                updateMealsData();
            }
        };

        updateMealsData();

        setFilteredHistories(filterHistories());

        window.addEventListener('storage', handleStorageChange);

        return () => {
            window.addEventListener('storage', handleStorageChange);
        };
    }, [filter, dateRange, mealsHistories]);

    const onDelete = (id) => {
        let userMeals = JSON.parse(localStorage.getItem('userMeals'))
|| []; // Получаем данные из localStorage

        userMeals = userMeals.filter(item => item.id !== id);

        // Обновляем данные в localStorage
        localStorage.setItem('userMeals', JSON.stringify(userMeals));

        setMealsHistories(userMeals);
        setFilteredHistories(userMeals);
    };

    const handleSubmit = (event) => {
        event.preventDefault();
        let belki, uglevody, kalorii, kletchatka, zhiry, water;
        if (dishName === 'Куриное филе') {
            belki = roundToTwo(0.31 * dishKol); uglevody = roundToTwo(0 *
dishKol); kalorii = roundToTwo(1.65 * dishKol); kletchatka = roundToTwo(0
* dishKol); zhiry = roundToTwo(0.036 * dishKol); water = roundToTwo(0.64
* dishKol);
        } else if (dishName === 'Овсяная каша на воде') {
            belki = roundToTwo(0.032 * dishKol); uglevody =
roundToTwo(0.142 * dishKol); kalorii = roundToTwo(0.88 * dishKol);
kletchatka = roundToTwo(0.021 * dishKol); zhiry = roundToTwo(0.017 *
dishKol); water = roundToTwo(0.8 * dishKol);
        } else if (dishName === 'Яблоки свежие') {
            belki = roundToTwo(0.03 * dishKol); uglevody =
roundToTwo(0.14 * dishKol); kalorii = roundToTwo(0.52 * dishKol);
kletchatka = roundToTwo(0.024 * dishKol); zhiry = roundToTwo(0.002 *
dishKol); water = roundToTwo(0.85 * dishKol);
        } else if (dishName === 'Болгарский перец') {
            belki = roundToTwo(0.01 * dishKol); uglevody =
roundToTwo(0.047 * dishKol); kalorii = roundToTwo(0.2 * dishKol);
kletchatka = roundToTwo(0.017 * dishKol); zhiry = roundToTwo(0.002 *
dishKol); water = roundToTwo(0.93 * dishKol);
```

```
        } else if (dishName === 'Творог (5% жирности)') {
            belki = roundToTwo(0.17 * dishKol); uglevody =
roundToTwo(0.033 * dishKol); kalorii = roundToTwo(1.21 * dishKol);
kletchatka = roundToTwo(0 * dishKol); zhiry = roundToTwo(0.05 * dishKol);
water = roundToTwo(0.73 * dishKol);
        } else if (dishName === 'Лосось запеченный') {
            belki = roundToTwo(0.22 * dishKol); uglevody = roundToTwo(0 *
dishKol); kalorii = roundToTwo(2.06 * dishKol); kletchatka = roundToTwo(0
* dishKol); zhiry = roundToTwo(0.13 * dishKol); water = roundToTwo(0.64 *
dishKol);
        } else if (dishName === 'Омлет') {
            belki = roundToTwo(0.11 * dishKol); uglevody =
roundToTwo(0.016 * dishKol); kalorii = roundToTwo(1.54 * dishKol);
kletchatka = roundToTwo(0 * dishKol); zhiry = roundToTwo(0.12 * dishKol);
water = roundToTwo(0.74 * dishKol);
        } else if (dishName === 'Картофельное пюре') {
            belki = roundToTwo(0.017 * dishKol); uglevody =
roundToTwo(0.11 * dishKol); kalorii = roundToTwo(0.83 * dishKol);
kletchatka = roundToTwo(0.018 * dishKol); zhiry = roundToTwo(0.038 *
dishKol); water = roundToTwo(0.82 * dishKol);
        } else if (dishName === 'Шпинат') {
            belki = roundToTwo(0.03 * dishKol); uglevody =
roundToTwo(0.038 * dishKol); kalorii = roundToTwo(0.23 * dishKol);
kletchatka = roundToTwo(0.024 * dishKol); zhiry = roundToTwo(0.004 *
dishKol); water = roundToTwo(0.91 * dishKol);
        }

        setBelki(belki);
        setUglevody(uglevody);
        setKalorii(kalorii);
        setKletchatka(kletchatka);
        setZhiry(zhiry);
        setWater(water);

        const userMealsData = {
            id: uuidv4(),
            type: mealType,
            dish: dishName,
            dishKol: dishKol,
            belki: belki,
            uglevody: uglevody,
            kalorii: kalorii,
            kletchatka: kletchatka,
            zhiry: zhiry,
            water: water,
            createdDate: new Date(),
        };

        var savedMeals = JSON.parse(localStorage.getItem('userMeals'))
|| [];
        savedMeals.push(userMealsData);
        localStorage.setItem('userMeals', JSON.stringify(savedMeals));
        closeModal();
    };

    const openModal = () => {
        setIsOpen(true);
    };
```

```
    const closeModal = () => {
        setIsOpen(false);
    };

    return (
        <div>
            <h2>Питание</h2>
            <div className='sleep-row-btn habbit-row-btn'>
                <button onClick={openModal}>Добавить приём пищи</button>
                {isOpen && (
                    <Modal onClose={closeModal}>
                        <h2>Добавить приём пищи</h2>
                        <form onSubmit={handleSubmit}>
                            <div>
                                <label htmlFor="mealType">Тип приёма
пищи:</label>
                                <select value={mealType} onChange={e =>
setMealType(e.target.value)}>
                                    {mealTypes.map(type => <option
key={type} value={type}>{type}</option>)}
                                </select>
                            </div>
                            <div>
                                <label htmlFor="dishName">Название
блюда:</label>
                                <select value={dishName} onChange={e =>
setDishName(e.target.value)}>
                                    {dishes.map(dish => <option
key={dish} value={dish}>{dish}</option>)}
                                </select>
                            </div>
                            <div>
                                <label htmlFor="dishKol">Количество,
граммы:</label>
                                <input
                                    type="number"
                                    id="dishKol"
                                    value={dishKol}
                                    onChange={(e) =>
setDishKol(e.target.value)}
                                />
                            </div>
                            <button type="submit">Добавить</button>
                        </form>
                    </Modal>
                )}
            </div>
            <div className='history'>
                <button className="button_filter" onClick={() =>
setFilter('all')}>За всё время</button>
                <button className="button_filter" onClick={() =>
setFilter('today')}>Сегодня</button>
                <button className="button_filter" onClick={() =>
setFilter('yesterday')}>Вчера</button>
                <button className="button_filter" onClick={() =>
setFilter('range')}>
                    Выбрать промежуток
```

```
                </button>
                {filter === 'range' && (
                    <>
                        <input
                            type="date"
                            value={dateRange.startDate &&
!isNaN(dateRange.startDate.getTime()) ?
dateRange.startDate.toISOString().split('T')[0] : ''}
                            onChange={e => setDateRange({ ...dateRange,
startDate: new Date(e.target.value) })}
                        />
                        <input
                            type="date"
                            value={dateRange.endDate &&
!isNaN(dateRange.endDate.getTime()) ?
dateRange.endDate.toISOString().split('T')[0] : ''}
                            onChange={e => setDateRange({ ...dateRange,
endDate: new Date(e.target.value) })}
                        />
                    </>
                )}
                <div className='info_block'>
                    {filteredHistories && filteredHistories.length > 0 ?
(
                        filteredHistories.map((history, index) => (
                            <div className='info_box' key={index}>
                                <div className='info_box_data'>
                                    {history.type}<p>
</p>{formatDate(history.createdDate)}
                                </div>
                                <div className='info_box_content'>
                                    <div className='setting_item'>
                                        Название блюда
                                        <p>{history.dish}</p>
                                    </div>
                                    <div className='setting_item'>
                                        Пищевая ценность в
{history.dishKol} грамм
                                    </div>
                                    <div className='setting_item
setting_item_nutri'>
                                        <p>Белки: {history.belki}</p>
                                        <p>Углеводы:
{history.uglevody}</p>
                                        <p>Калории: {history.kalorii}</p>
                                        <p>Жиры: {history.zhiry}</p>
                                        <p>Клетчатка:
{history.kletchatka}</p>
                                        <p>Вода: {history.water}</p>
                                    </div>
                                </div>
                                <button className='info_box_btn'
onClick={() => onDelete(history.id)}>Удалить</button>
                            </div>
                        ))
                    ) : (
                        <p>Нет доступных данных для отображения</p>
                    )}
```

```
                </div>
            </div>
        </div>
    );
}

export default Nutrition;
```

Листинг И.6 – Файл SleepDataComponent.jsx

```
import React, { useState, useEffect } from 'react';
import axios from 'axios';
import healthDataUpdate from './healthDataUpdate';

function formatDateTime(dateString) {
    const options = { year: 'numeric', month: 'numeric', day: 'numeric',
hour: '2-digit', minute: '2-digit', second: '2-digit' };
    return new Date(dateString).toLocaleDateString('ru-RU', options);
}

function formatDate(dateString) {
    const options = { year: 'numeric', month: 'numeric', day: 'numeric'
};
    return new Date(dateString).toLocaleDateString('ru-RU', options);
}

function formatTime(timeString) {
    if (typeof timeString === 'string') {
        return timeString.slice(0, 5);
    }
    return 'Время не задано';
}

function SleepDataComponent() {
    const userDataString = localStorage.getItem('user');
    const userData = userDataString ? JSON.parse(userDataString) : null;
    const user = userData ? userData.user : null;

    const [sleepHistories, setSleepHistories] = useState([]);
    const [sleepSettings, setSleepSettings] = useState([]);
    const [filter, setFilter] = useState([]);
    const [dateRange, setDateRange] = useState({ startDate: new Date(),
endDate: new Date() });
    const [filteredHistories, setFilteredHistories] = useState([]);

    const filterHistories = () => {
        const today = new Date();
        today.setHours(0, 0, 0, 0);
        const yesterday = new Date(today);
        yesterday.setDate(yesterday.getDate() - 1);

        switch (filter) {
            case 'today':
                return sleepHistories.filter(history =>
```

```
                    new Date(history.createdDate).setHours(0, 0, 0, 0)
=== today.getTime());
            case 'yesterday':
                return sleepHistories.filter(history =>
                    new Date(history.createdDate).setHours(0, 0, 0, 0)
=== yesterday.getTime());
            case 'range':
                return sleepHistories.filter(history => {
                    const historyDate = new Date(history.createdDate);
                    return historyDate >= dateRange.startDate &&
historyDate <= dateRange.endDate;
                });
            case 'all':
            default:
                return sleepHistories;
        }
    };

    useEffect(() => {
        const updateSleepData = () => {
            const storedData = localStorage.getItem('userHealthData');
            if (storedData) {
                const parsedData = JSON.parse(storedData);
                if (Array.isArray(parsedData) && parsedData.length > 0) {
                    const userHealthData = parsedData[0];
                    const histories = userHealthData.sleepHistories
|| [];
                    const settings = userHealthData.sleepSettings || {};
                    setSleepHistories(histories);
                    setSleepSettings(settings);
                } else {
                    console.log ('Parsed data is not an array or is
empty');
                }
            } else {
                console.log('No stored data found in localStorage');
            }
        };

        const handleStorageChange = (event) => {
            if (event.key === 'userHealthData') {
                updateSleepData();
            }
        };

        updateSleepData();

        setFilteredHistories(filterHistories());

        window.addEventListener('storage', handleStorageChange);

        return () => {
            window.addEventListener('storage', handleStorageChange);
        };
    }, [filter, dateRange, sleepHistories]);

    const handleDelete = (sleepHistoryId) => {
```

```
        const updatedHistories = sleepHistories.filter(history =>
history.id !== sleepHistoryId);
        setSleepHistories(updatedHistories);
        healthDataUpdate(user.id);
    };

    const onDelete = async (sleepHistoryId) => {
        try {
            const response = await axios.delete(`/api3/sleep-
history/${sleepHistoryId}`);
            if (response.data) {
                handleDelete(sleepHistoryId);
            } else {
                console.error('Failed to delete sleep history');
            }
        } catch (error) {
            console.error('An error occurred while trying to delete sleep
history:', error);
        }
    };

    return (
        <div>
            <div className='settings_block'>
                {
                    sleepSettings ? (
                        <div className='setting'>
                            <div
className='setting_data'>{formatDate(sleepSettings.createdDate)}</div>
                            <div className='setting_item'>
                                Желаемая продолжительность сна
                                <p>{sleepSettings.hourAmount} часов</p>
                            </div>
                            <div className='setting_item'>
                                Желаемое время отхода ко сну

<p>{formatTime(sleepSettings.wishedBedTime)}</p>
                            </div>
                            <div className='setting_item'>
                                Желаемое время подъёма

<p>{formatTime(sleepSettings.wishedWakeUpTime)}</p>
                            </div>
                        </div>
                    ) : <p>Желаемые настройки сна не введены
пользователем</p>
                }
            </div>
```