МИНОБРНАУКИ РОССИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет Компьютерных наук

Кафедра программирования и информационных технологий

Разработка медицинского ассистента с интегрированным чат-ботом

Курсовая работа

Направление: 09.03.04 Программная инженерия

Зав. Кафедрой _____ д. ф.-м. н, доцент С.Д. Махортов

Руководитель _____ ст. преподаватель В.С. Тарасов

Руководитель практики _____ ассистент Е.Д. Проскуряков

Обучающийся _____ Е.С. Труфанов, 3 курс, д/о

Обучающийся _____ Д.С. Ушаков, 3 курс, д/о

Обучающийся _____ Д.А. Сакун, 3 курс, д/о

Обучающийся _____ Е.В. Вологжин, 3 курс, д/о

Обучающийся _____ Н.С. Масалкин, 3 курс, д/о

Воронеж 2025

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Здоровье – это основное условие и залог счастливой и полноценной жизни каждого человека. Это та основа, без которой нельзя добиться запланированных целей, преодолеть разные трудности и решать основные жизненные задачи. Когда люди не следят за своим здоровьем, с возрастом это может привести к неприятным и печальным последствиям: развитию хронических заболеваний, потере трудоспособности, преждевременному старению; кто-то может стать жертвой малоподвижности, а другие страдают бессонницей. Люди, которые ответственно относятся к своему здоровью, реже сталкиваются с необходимостью экстренной госпитализации, тратят меньше средств на медицинские услуги и имеют более высокое качество жизни.

Мы выяснили, что важно следить за здоровьем, чтобы сделать свою жизнь долгой и активной без ограничений. Однако современный ритм жизни часто вынуждает нас отодвигать на второй план то, что должно было стать важным приоритетом. Мы откладываем визит к врачу, ссылаясь на нехватку времени, не понимая, что с организмом происходит, и заглушаем симптомы обезболивающими, игнорируя их причину.

Такая проблема усугубляется несовершенством системы здравоохранения. В крупных городах нужно потратить много времени, чтобы попасть к врачу – отстоять длинные очереди, чтобы записаться, а потом найти время посетить врача. По результату приема, врачи дают разные медицинские бумажные выписки, которые невозможно оперативно получить при необходимости – они теряются и хранятся в разных местах.

Усугубляет ситуацию то, что люди спрашивают о своих симптомах у друзей, родственниках, ищут признаки заболеваний в интернете, в результате путаются и не понимают, к какому врачу все-таки записаться и что хотя бы примерно с ними происходит, насколько симптомы серьезны.

Особенно тяжело приходится людям с хроническими заболеваниями, которым необходимо постоянно следить за своим состоянием и регулярно

принимать лекарства. Если человек принимает несколько препаратов одновременно, запомнить их названия становится настоящим испытанием.

Исходя из вышеперечисленных фактов, мы подумали, что было бы неплохо, если бы существовал медицинский чат-ассистент, благодаря которому пользователи смогут по своим симптомам определить вероятные заболевания. Также мы хотим, чтобы каждый пользователь мог поставить напоминания о приеме лекарств и настроить необходимое расписание. Мы сделаем систему, в которой пациент сможет хранить важную информацию о своем здоровье в электронной медицинской карте, а не в множестве бумажных выписок.

Актуальность обусловлена необходимостью в удобном цифровом инструменте для объединения анализа симптомов, умных напоминаний о приеме лекарств и медкарты.

Целью курсовой работы является разработка медицинского цифрового ассистента с интегрированным чат-ботом для автоматизированной диагностики симптомов, управления приемом лекарств и централизированного хранения медицинских данных пользователей.

Задачи:

— изучить предметную область, провести обзор аналогов;

— разработать медицинский ассистент с интегрированным чат-ботом.

1 Постановка задачи

1.1 Цели создания приложения

Целями создания системы являются:

— удовлетворить пациентов точностью ИИ-диагностики после года использования должна составлять не менее 6 из 10 баллов (где 1 – полностью не удовлетворен, 10 – полностью удовлетворен), что повысит доверие к сервису и снизит количество обращений в медучреждения;

— в течение первого года работы системы (2025-2026 гг.) привлечь не менее 100 активных пользователей, что позволит сформировать базу данных для улучшения алгоритмов;

— в течение первого года работы системы (2025-2026 гг) обеспечить соблюдение режима приема лекарств не менее чем у 60% пользователей, благодаря персонализированным напоминаниям.

1.2 Задачи приложения

Приложение позволяет решать следующие задачи:

— получать информацию о возможных заболеваниях на основе указанных симптомов;

— устанавливать расписание приема лекарственных препаратов;

— формировать и редактировать интерактивную медицинскую карту;

— осуществлять редактирование данных своего аккаунта после авторизации или регистрации в системе.

2 Требования к приложению

2.1 Требования к приложению в целом

Структура автоматизированной системы должна обеспечивать устойчивую и эффективную работу всех ее компонентов. Важно предусмотреть возможность интеграции новых функций, а также обеспечить гибкость и масштабируемость для будущих модернизаций.

2.2 Требования к функциям, выполняемым приложением

Функциональные требования описывают конкретные возможности и действия, которые система должна выполнять для удовлетворения потребностей пользователей и бизнеса. Они определяют, как продукт будет реагировать на определенные входные данные, условия эксплуатации и пользовательские сценарии. Формулируются в виде четких и измеримых утверждений.

Разрабатываемое приложение должно соответствовать следующим требованиям:

Система должна предоставлять возможность для неавторизованного пользователя:

— регистрации нового пользователя;

— авторизации существующего пользователя;

— настройки расписания приема лекарств.

Система должна предоставлять возможность для авторизованного пользователя:

— доступа к персональной медицинской карте;

— просмотра истории сообщений в чате-ассистенте;

— использования экстренной SOS-кнопки;

— управления расписанием приема лекарств;

— восстановления пароля.

2.2.1 Вход в приложение

Система должна предоставлять пользователю возможность:

— зарегистрировать профиль;

— войти по почте и паролю;

— восстановить пароль;

— заполнить данные медицинской карты во время регистрации.

2.2.2 Медицинская карта

Для авторизованного пользователя система должна предоставлять доступ к личной медицинской карты:

— просмотр медицинской карты;

— редактирование и удаление медицинской карты.

При создании и редактировании медицинской карты система должна позволять учитывать следующие сведения:

— ФИО;

— рост;

— вес(кг);

— группа крови;

— аллергии;

— заболевания.

Как результат, эта информация должна сохраниться в системе и быть доступной для авторизованного пользователя.

2.2.3 Расписание приема лекарств

Каждому пользователю система должна предоставлять доступ к расписанию приема лекарственных препаратов:

— просмотр расписания;

— редактирование и удаление расписания.

При создании и редактировании расписания лекарств система должна позволять учитывать следующие сведения:

— название лекарственного препарата;

— день приема;

— время приема;

— настройка уведомлений.

Как результат, эта информация должна сохраниться в системе и быть доступной для каждого пользователя приложения.

2.2.4 Медицинский чат-ассистент

Для авторизованного пользователя система должна предоставлять доступ к медицинскому ассистенту в виде чат-бота, в котором реализован следующий функционал:

— просмотр предыдущих сообщений;

— определение возможного заболевания, по введенным пользователем симптомам.

При получении симптома от пользователя система должна предоставлять список возможных заболеваний.

Выбранное заболевание должно включать в себя следующие пункты:

— описание болезни;

— виды заболевания;

— причины возникновения заболевания.

Как результат, эта информация должна сохраниться в системе и быть доступной для всех зарегистрированных пользователей.

2.2.5 Экстренная кнопка (SOS)

Система должна предоставлять авторизованному пользователю возможность воспользоваться виджетом, открывающим медицинскую карту.

Как результат, эту информацию можно использовать при обращении в скорую медицинскую помощь.

2.2.6 Настройки приложения

Система должна предоставлять авторизованному пользователю возможность воспользоваться настройками, включающие в себя:

— изменение пароля;

— выход из аккаунта;

— включение/выключение push-уведомлений.

При изменении пароля система должна учитывать:

— старый пароль;

— новый пароль;

— подтверждение нового пароля.

2.3 Требования к оформлению и верстке страниц

Экраны мобильного приложения должны быть оформлены в едином стиле с использованием ограниченного набора шрифтов.

Необходимо корректное и одинаковое отображение экранов мобильного приложения на устройствах с операционной системой Android 11 и выше.

Рисунок 1 демонстрирует прототип цветового оформления дизайна мобильного приложения:

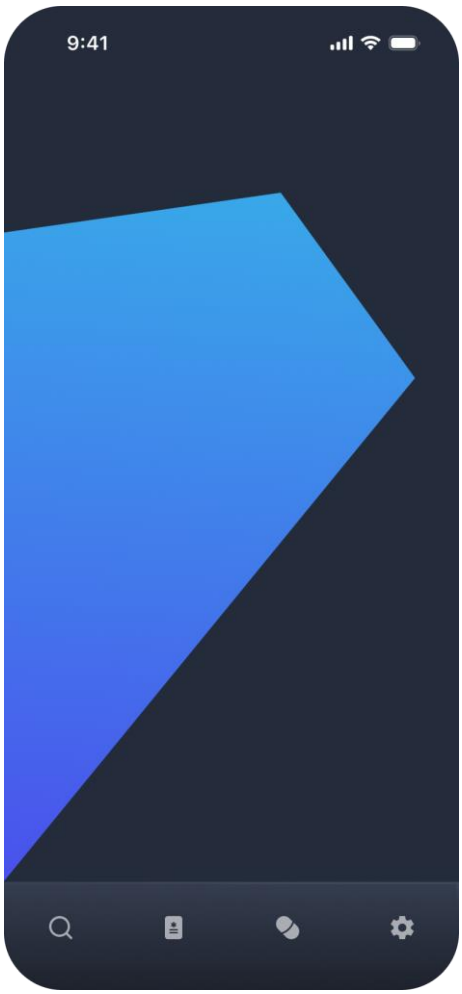

Рисунок 1 — Цветовое оформление дизайна мобильного приложения

Рисунок 2 демонстрирует прототипы основных типов экранов мобильного приложения:

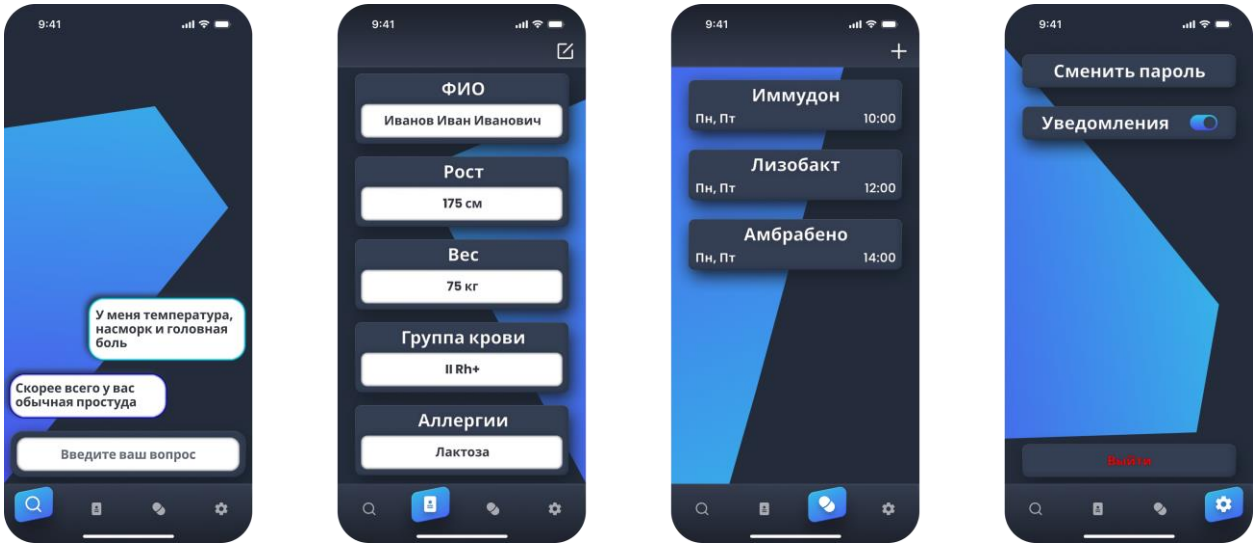

Рисунок 2 — Прототипы экранов с чатом, медицинской картой, расписанием приёма лекарственных препаратов и настройками

2.4 Требования по безопасности

— обмен данных между клиентом и сервером должен осуществлять по протоколу HTTPS;

— пароли пользователей должны хранится в базе данных в хешированном виде;

— для хеширования должен использоваться BCrypt Password Encoder и JWT Token.

3 Анализ предметной области

3.1 Глоссарий

Back-end — часть программного обеспечения, отвечающая за обработку данных и взаимодействие с сервером.

Dart — язык программирования, созданный компанией Google.

Feature-модуль — это модуль, предназначенный для организации приложения по конкретным функциям или областям.

Flutter — комплект средств разработки и фреймворк с открытым исходным кодом для создания мобильных и веб-приложений.

Front-end — часть программного обеспечения, отвечающая за визуальное представление данных и взаимодействие с пользователем.

JWT (Json Web Token) — ключ аутентификации пользователя.

MVVM (Model-View-ViewModel) — это архитектурный шаблон, используемый в разработке программного обеспечения для разделения пользовательского интерфейса (UI) от бизнес-логики и данных.

MVP — продукт, обладающий минимальными, но достаточными функциями для удовлетворения первых потребителей.

REST API — архитектурный стиль веб-служб, использующий протокол HTTP для передачи данных между клиентом и сервером.

RESTful-сервис – это веб-сервис, построенный на архитектуре REST.

Spring — фреймворк для разработки приложений на языке Java.

Yandex Cloud — публичная облачная платформа от транснациональной интернет-компании «Яндекс»

Автоматизированная система (АС) — программа или набор программ, предназначенных для выполнения задач без прямого участия человека.

Авторизация — процесс предоставления пользователю определенных прав доступа и привилегий в системе.

Аккаунт — персональная учетная запись пользователя, позволяющая ему получить доступ к системе или сервису.

Аннотация – специальная форма синтаксических метаданных, которая может быть добавлена в исходный код.

Аутентификация — процесс проверки подлинности пользователя по его учетным данным.

База данных (БД) — организованная совокупность данных, хранимая и обрабатываемая с использованием компьютерных систем.

Искусственный интеллект (ИИ) — комплекс технологий, имитирующих когнитивные функции человека, включая самообучение и принятие решений.

Контроллер – класс, предназначенный для непосредственной обработки запросов от клиента и возвращения результатов.

Программное обеспечение (ПО) — совокупность программных инструкций и данных для функционирования компьютера.

Сервер — компьютер или программа, предоставляющая ресурсы и услуги другим устройствам.

Система управления базами данных (СУБД) — программное обеспечение для создания, хранения и управления базами данных.

Темизация — централизованное управление визуальным стилем приложения (цвета, шрифты, темы оформления).

Фреймворк — набор библиотек и инструментов, упрощающих и ускоряющих разработку программного обеспечения.

3.2 Целевая аудитория

PocketHealth — мобильное приложение для людей, которые хотят следить за своим здоровьем, но не имеют медицинского образования. Наши основные пользователи — это жители России, которые сталкиваются с проблемами при поиске медицинской информации и организации приема лекарств.

Основные категории пользователей:

— молодые специалисты и офисные работники — заботятся о здоровье, но не всегда находят время на визит к врачу;

— люди с хроническими заболеваниями — нуждаются в напоминаниях о приеме лекарств и мониторинге состояния;

— жители крупных городов — высокая нагрузка на поликлиники заставляет искать альтернативные способы оценки симптомов;

— активные пользователи смартфонов — уже используют мобильные сервисы для фитнеса и здоровья и готовы к новым технологиям.

Приложение разработано с целью решить проблемы пользователей: правильно интерпретировать симптомы, быстро предоставить доступ к медицинской информации и создать возможность настройки индивидуальных напоминаний о приеме лекарств.

3.3 Обзор аналогов

В рамках исследования рынка были проанализированы следующие цифровые решения: Your.MD и Medisafe.

Эти приложения, в отличие от приложения PocketHealth, отличаются отсутствием возможности заполнения собственной медицинской карты. Приложение Your.MD предлагает функцию цифрового помощника для первичной диагностики по симптомам, настройку расписания для приема препаратов. Medisafe дает доступ пользователю для настройки напоминаний о приеме лекарств. Рассмотрим каждое из этих предложений.

3.3.1 Your.MD

Your.MD — это приложение, которое помогает пользователям управлять своим здоровьем с помощью искусственного интеллекта. Система включает диагностику симптомов, напоминания о приеме лекарств, онлайн-запись к врачу и персональные рекомендации. Пользователь может быстро получить анализ своего состояния, советы по лечению и профилактике, а также удобный доступ к медицинским услугам.

Рассмотрим достоинства приложения:

— наличие медицинского чат-ассистента, способного распознавать

симптомы и выдавать предположения по диагнозу;

— наличие напоминаний о приеме лекарств;

— наличие базовых функций настроек, таких как включение/выключение уведомлений и смена пароля.

Но у этого приложения есть и недостатки:

— отсутствие возможности заполнения медицинской карты;

— отсутствие быстрого получения своих медицинских данных.

3.3.2 Medisafe

Medisafe — это приложение, которое помогает пользователям не забывать о приеме лекарств. Оно отправляет напоминания, отслеживает дозировки и позволяет делиться расписанием с близкими или врачами. Приложение также анализирует совместимость препаратов и предоставляет персонализированные советы по их приему.

Из достоинств приложения можно выделить удобный интерфейс и возможность системы информировать пользователя, когда и какое лекарство принимать. На рисунке 3 представлены экраны приложения для добавления препарата. Каждый шаг – это отдельный экран, что помогает не перегружать интерфейс. При добавлении препарата система предоставляет список возможных названий лекарств, что делает приложение более удобным.

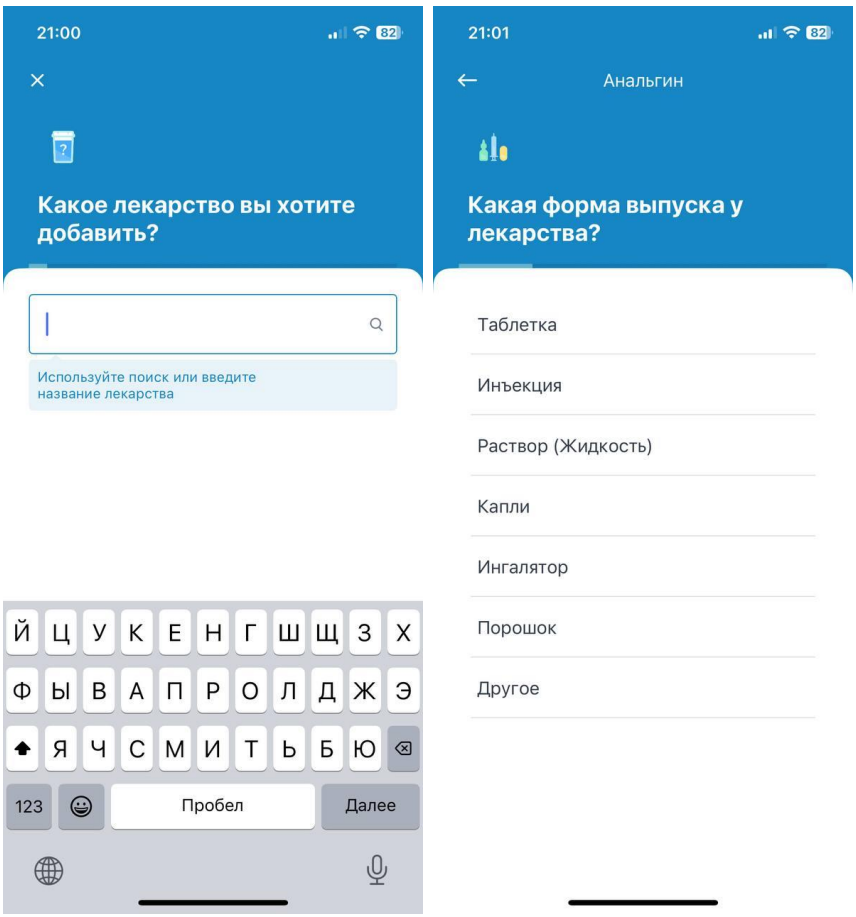

Рисунок 3 – Экраны добавления препарата в приложении Medisafe

В этом приложении можно выделить и недостатки. В системе недостаточно возможностей. Недостатками являются:

— отсутствие чат-ассистента, который мог бы определить предполагаемые заболевания по симптомам;

— отсутствие возможности заполнить медкарту;

— отсутствие возможности быстро получить важные данные.

3.3.3 Вывод по обзору аналогов

В ходе исследования рынка приложений медицинских чат-ассистентов было выявлено 2 прямых конкурента. Таблица 1 содержит результаты проведённого конкурентного исследования. Также мы добавили в последний столбец наше разработанное приложение.

Таблица 1 — Результаты конкурентного исследования

Характеристика	Your.MD	Medisafe	PocketHealth
Наличие искусственного интеллекта	+	-	+
Диагностика симптомов	+	-	+
Напоминания о приеме препаратов	+	+	+
Возможность заполнения собственной медицинской карты	-	-	+

4 Моделирование системы

4.1 Диаграмма классов

Диаграмма классов – это структурная диаграмма языка моделирования UML. Она демонстрирует общую структуру иерархии классов системы, их атрибутов, методов и взаимосвязи между ними.

На рисунке 4 представлена диаграмма классов. Давайте немного рассмотрим структуру:

— User – основной класс, представляет пользователя системы;

— Medical Card – класс, содержащий медицинскую информацию;

— Medication Schedule – класс, который описывает расписание приема лекарств;

— Chat History – класс для хранения сообщений пользователя;

— Knowledge Base – класс, содержащий описание заболеваний, их симптомы и причины.

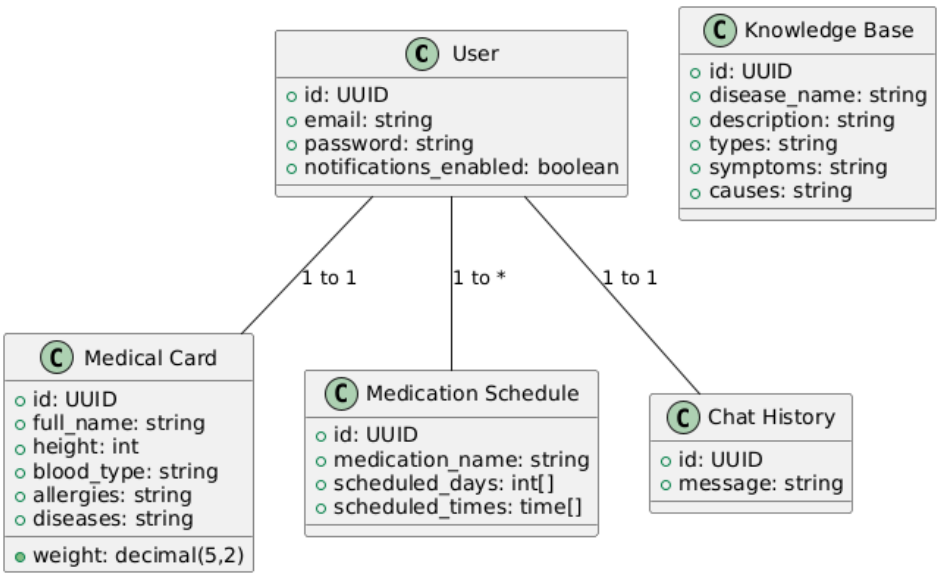

Рисунок 4 — Диаграмма классов

4.2 Диаграмма прецендентов

Диаграмма прецендентов необходима для отражения отношения между акторами и прецендентами.

В приложении А представлены сценарии взаимодействия пользователя с медицинским приложением. Актером выступает Пользователь, инициирующий все действия в системе.

4.3 Диаграмма активности

Для наглядного описания логики работы приложения мы разработали диаграмму активности, ее фрагменты отражают ключевые процессы. Такими являются регистрация, авторизация, работа с чат-ассистентом и управление расписанием. Рассмотрим самые основные из них.

На рисунке 5 представлен фрагмент диаграммы, описывающий процесс взаимодействия пользователя с чат-ботом для анализа симптомов. Пользователь вводит симптомы в чат, после чего система производит их анализ и предполагает возможные заболевания.

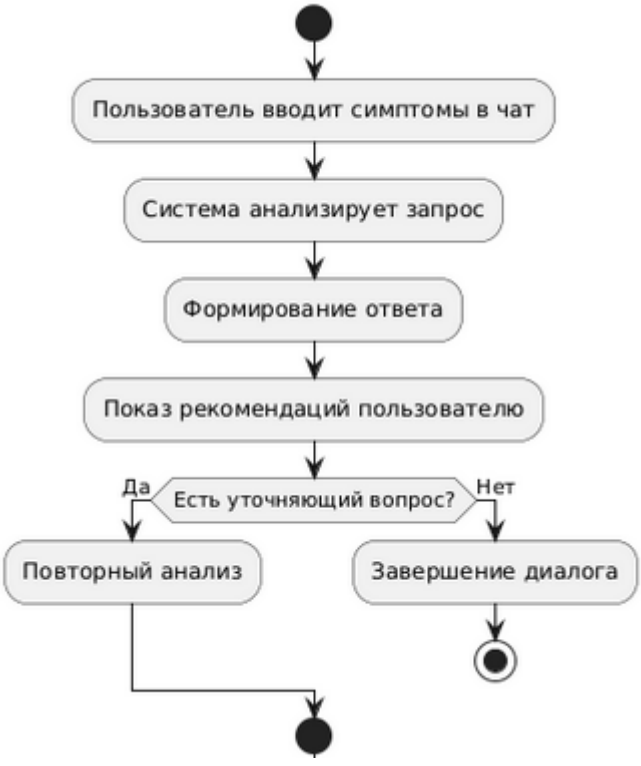

Рисунок 5 – Диаграмма активности (фрагмент с чат-ботом)

Рассмотрим поведение пользователя в разделе «Расписание» мобильного приложения. На рисунке 6 представлен фрагмент диаграммы,

отражающий возможные действия пользователя: ему предоставляется возможность добавлять расписание, редактировать его и удалять.

Рисунок 6 – Диаграмма активности (фрагмент с расписанием)

4.4 ER-диаграмма

ER-диаграмма показывает, какие данные есть в нашей системе и как они связаны. В приложении Б представлена ER-диаграмма базы данных медицинского приложения, которая включает в себя 6 сущностей: users, medical_cards, medication_schedule, chat_history, knowledge_base. Центральной сущностью выступает users, к которой присоединены остальные таблицы с различными типами связей. Таблица medical_cards отражает медицинские данные пользователя и связана с ним отношением один к одному. Есть таблица

medication_schedule, которая реализует график приёма лекарств и связана с пользователем отношением один ко многим. Также есть таблица chat_history хранит переписку пользователя с чат-ботом и также имеет отношение один к одному. Таблица knowledge_base служит справочником по заболеваниям и симптомам и используется для анализа введённых пользователем данных.

4.5 Диаграмма последовательности

Диаграмма последовательности может иллюстрировать взаимодействие компонентов системы при типичных пользовательских сценариях. Диаграмму можно разделить на следующие фрагменты: регистрация, авторизация, восстановление пароля, проверка кода, установка нового пароля, диалог с ассистентом, создание расписания и его удаление.

Рассмотрим фрагмент, связанный с удалением расписания, представленный в приложении В. На диаграмме показан процесс взаимодействия между пользователем, клиентской частью, сервером и базой данных при добавлении и удалении расписания приёма лекарств.

Пользователь заполняет форму с параметрами (название препарата, дни приема и время), данные отправляются на сервер и сохраняются в базе данных.

В случае удаления записи, пользователю необходимо выделить нужный элемент, запрос на удаление отправляется на сервер. После подтверждения удаления, данные удалятся из базы, а интерфейс обновится.

5 Реализация

5.1 Средства реализации

Приложение имеет архитектуру, соответствующую модели Клиент-Серверного взаимодействия на основе REST API.

Для реализации серверной части использовались следующие средства:

— язык программирования Java версии 17;

— фреймворк Spring Boot версии 3.4.3;

— СУБД PostgreSQL версии 17.

Для реализации клиентской части мобильного приложения и сервисного веб-приложения использовались следующие средства:

— язык программирования Dart версии 3.7.0;

— Flutter SDK версии 3.29.0.

Инструменты для введения документации:

— YouTrack — это средство для управления проектами и задачами, которое обеспечивает удобный интерфейс, гибкую настройку и возможность отслеживать прогресс работы над проектом;

— Miro — это инструмент для совместной работы и визуализации идей, который позволяет создавать диаграммы, макеты, схемы и другие элементы проекта;

— PlantUML — это инструмент с открытым исходным кодом, позволяющий пользователям создавать диаграммы на обычном текстовом языке;

— Figma — инструмент для дизайна интерфейсов, который обеспечивает возможность создания прототипов, макетов и дизайнов веб-приложений.

Также мы использовали дополнительные инструменты:

— Git — распределенная система управления версиями, которая обеспечивает контроль изменений в программном коде, возможность ветвления и слияния программного кода;

— GitHub — это платформа для хостинга проектов на базе Git, которая обеспечивает возможность хранения программного кода, управления задачами, рецензирования программного кода и совместной работы над проектами;

— платформа Docker — открытая платформа для разработки, доставки и эксплуатации приложений. Позволяет создавать среды разработки и развертывать приложения с минимальными затратами на конфигурацию и совместимость.

5.2 Архитектура клиентской части

Для реализации клиентской части приложения мы использовали фреймворк Flutter, который обеспечивает кроссплатформенную разработку мобильных приложений с использованием языка программирования Dart. Архитектура фронтенда построена на принципах модульности, разделения ответственности. Проект разделен на модули, каждый из которых включает соответствующие экраны и компоненты. Это упрощает масштабирование и поддержку.

На рисунке 7 отображена основная организация файлов и папок, которая отражает модульную структуру приложения.

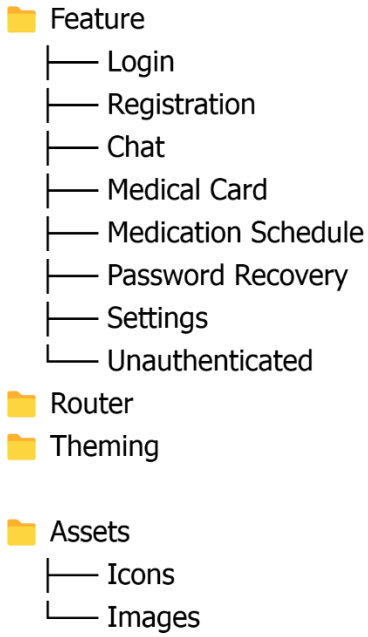

Рисунок 7 — Структура клиентской части приложения

Такое представление демонстрирует соблюдение принципа разделения ответственности и поддержку чистой архитектуры. Каждому элементу пользовательского интерфейса и логики соответствует строго определенное место в файловой системе проекта.

5.2.1 Модульная организация кода

Клиентская часть приложения организована по модульному принципу, где каждая функциональная область представляется отдельным логическим модулем. Такая структура реализована в директории Feature и содержит поддиректории для каждого крупного раздела приложения, таких как, например: авторизация, регистрация, чат, медицинская карта, расписание приема лекарств, восстановление пароля, настройки и экраны для неавторизованных пользователей.

Внутри каждого модуля расположена папка View, в ней размещаются все экраны, которые относятся к данному разделу. Также в каждом из них лежит файл view.dart, предоставляющий доступ к экранам. Благодаря такой структуре мы централизируем подключение представлений и упрощаем

маршрутизацию. В некоторых модулях можно увидеть дополнительные директории Widgets – там располагаются переиспользуемые пользовательские элементы интерфейса. Такими могут быть кастомные поля ввода, кнопки или компоненты карточек.

Рассмотрим один из модулей, структура которого представлена на рисунке 8.

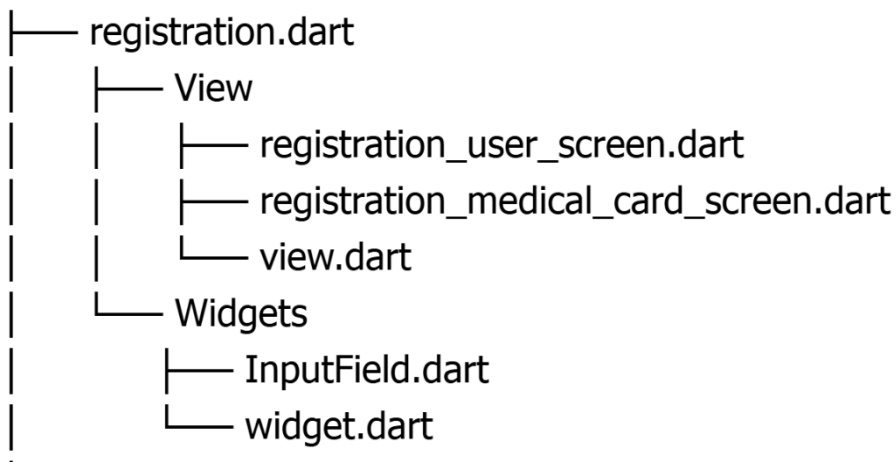

Рисунок 8 — Структура модуля регистрации

Примером такой модульной организации служит модуль registration, отвечающий за процесс регистрации нового пользователя.

Модуль включает в себя файл registration.dart, который можно назвать точкой входа модуля. В такой файле собирается вся логика, связанная с этой частью приложения, например: настройка маршрутов, подключение экранов и импорт вспомогательных элементов. Это делает код удобнее для управления и изменений.

В модуле есть подкаталог View, который содержит представления, связанные с процессом регистрации. Например, registration_user_screen.dart отвечает за экран, где пользователь вводит основные данные. Также есть файл для заполнения медицинских данных.

Чтобы не импортировать каждый экран по отдельности в других частях программы, используется файл view.dart. Он работает как центральная точка подключения: вместо того чтобы прописывать множество отдельных

импортов, достаточно обратиться к нему, и он сам предоставит доступ ко всем нужным экранам модуля. Это делает код чище и удобнее для работы.

В модуле есть подкаталог Widgets, который содержит повторно используемые пользовательские элементы интерфейса. InputField.dart – поле для ввода текста, которое можно использовать в разных частях приложения. Файл widget.dart используется для сбора разных пользовательских виджетов и вспомогательных элементов в одном месте.

Мы рассмотрели пример модульной организации для реализации регистрации. Такая организация приближает архитектуру приложения к шаблону MVVM (Model-View-ViewModel), где представления (View) изолированы от пользовательских компонентов (Widgets), а бизнес-логика может при необходимости выноситься в отдельные слои.

5.2.2 Навигация и темизация

В проекте реализована четкая структура, которая упрощает разработку и поддержку приложения. Центральным элементом навигации является модуль Router, где в файле CustomPageRoute.dart настроены все переходы между экранами с плавными анимациями. Такой подход дает несколько преимуществ: во-первых, все маршруты собраны в одном месте, что упрощает их изменение и добавление новых. Во-вторых, анимации переходов между экранами выглядят профессионально и согласованно по всему приложению.

Визуальная часть приложения управляется через модуль Theming, который определен в файле theme.dart. Здесь собраны все стили, цвета, шрифты и темы, что позволяет легко менять оформление всего приложения из одного места. Например, если нужно изменить основной цвет интерфейса, достаточно внести правку в этом файле, и изменения автоматически применятся ко всем экранам. Это особенно важно для больших проектов, где нужно поддерживать единый стиль.

Все графические ресурсы (изображения, иконки) аккуратно организованы в тематических папках. Например, иконки для меню лежат в

одной папке, а иллюстрации для экранов регистрации — в другой. Такая система позволяет быстро находить нужные файлы и предотвращает беспорядок в ресурсах проекта.

5.3 Архитектура серверной части

Серверная часть приложения реализовывалась с использованием фреймворка Spring Boot, который предоставляет мощные возможности для построения RESTful-сервисов, работы с базой данных и организации модульной архитектуры. Мы стремились достичь чистоты архитектуры, читаемости кода и возможности масштабирования.

На рисунке 9 представлена логическая структура серверной части, в которой каждый модуль имеет свою зону ответственности. Архитектура строится на разделении по слоям: от конфигурационного уровня и маршрутизации до работы с данными и безопасностью.

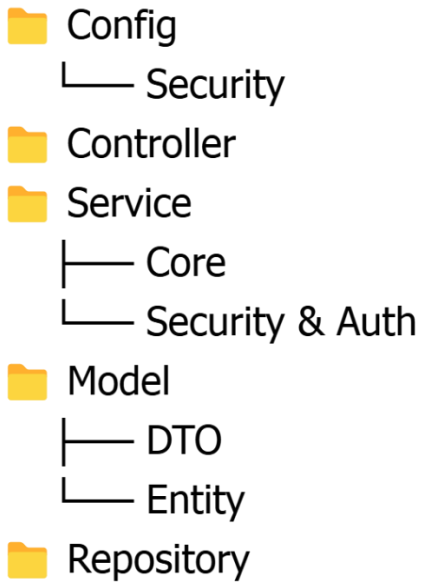

Рисунок 9 — Общая архитектура серверной части

Такой подход позволяет минимизировать связность компонентов между собой, а также повышает удобство поддержки проекта. Давайте рассмотрим каждый из слоев.

5.3.1 Контроллеры и маршрутизация

На внешнем уровне архитектуры находятся контроллеры – это классы, отвечающие за обработку HTTP-запросов от клиента. Контроллеры являются входной точкой приложения с точки зрения пользователя или внешней системы. В архитектуре REST API каждый контроллер представляет определённую функциональную область и обрабатывает запросы, относящиеся к конкретному ресурсу. Такой подход обеспечивает чёткое разделение ответственности и облегчает поддержку системы.

На рисунке 10 представлена структура папки controller, содержащая пять основных контроллеров. Например, один из контроллеров занимается вопросами аутентификации, включая регистрацию, вход в систему и восстановление доступа. Другой — обрабатывает обмен сообщениями с медицинским чат-ботом. Также предусмотрены контроллеры, отвечающие за работу с медицинскими картами, управление расписанием приёма лекарств и настройками профиля. Благодаря такой структуре, каждый контроллер обслуживает конкретный ресурс, обеспечивая логическую изоляцию и упрощая масштабирование приложения.

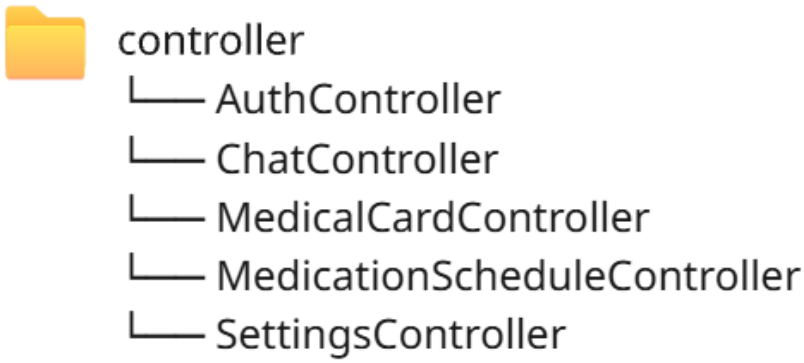

Рисунок 10 — Структура контроллеров проекта

Каждый метод контроллера аннотирован с помощью @GetMapping, @PostMapping, @PutMapping или @DeleteMapping, это позволяет определить соответствие между HTTP-методом и функцией обработчика. Данные

аннотации необходимы для того, чтобы автоматически связать URL и действия, сформировать поддерживаемую маршрутизацию.

Контроллер получает данные от клиента, вызывает соответствующий метод в слое сервиса и возвращает результат в виде JSON-ответа. Таким образом, контроллер не содержит бизнес-логики, а только делигирует ее выполнению сервисному слою.

5.3.2 Сервисный слой

Сервисный уровень архитектуры представляет собой промежуточный слой между контроллерами и хранилищами данных. Именно здесь сосредоточена основная бизнес-логика приложения. Каждый сервис отвечает за выполнение определённых задач в рамках бизнес-процессов, таких как обработка пользовательских действий, валидация данных, взаимодействие с базой данных и внешними API.

На рисунке 11 представлена структура сервисного уровня проекта. Для каждой функциональной области предусмотрен отдельный класс сервиса.

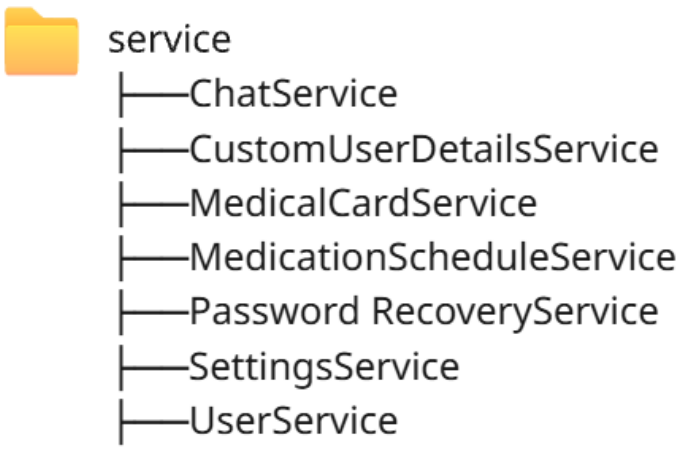

Рисунок 11 — Структура сервисного слоя проекта

Сервисы можно разделить на два логических блока – Core-сервисы и Security&Auth-сервисы.

Core-сервисы отвечают за основную предметную логику приложения. В них реализуются функции, связанные с регистрацией и управлением пользователями, обработкой данных медицинской карты, взаимодействием с

чат-ботом, а также управление настройками. Например, такими сервисами являются MedicalCardService, ChatService, SettingsService.

Security&Auth-сервисы реализуют логику, связанную с безопасностью и доступом пользователей. Это включает, например, аутентификацию и генерацию JWT-токенов, логику восстановления пароля.

5.3.3 Модели и репозитории

Рассмотрим уровень доступа к данным, который в нашем проекте представлен тремя ключевыми категориями: моделями сущностей (entity), объектами передачи данных (DTO) и репозиториями. Рисунок 12 демонстрирует частичную структуру каталогов проекта, описанных в этом разделе.

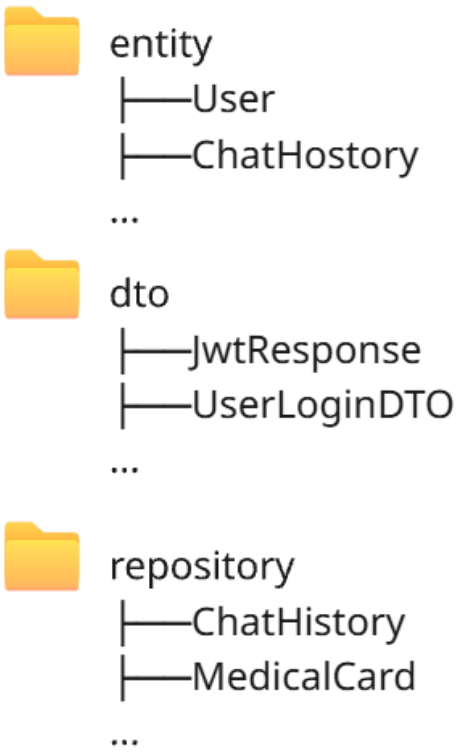

Рисунок 12 — Частичная структура модельных классов, DTO, репозиториев

Entity-классы в проекте представляют собой таблицы из базы данных в виде обычных Java-объектов. Благодаря специальным аннотациям из JPA каждый класс соответствует отдельной таблице в базе данных. Например, класс User содержит поля: id, email, password, notificationsEnabled, confirmPassword.

DTO (Data Transfer Objects) — промежуточный формат данных, который мы используем для передачи между слоями (контроллерами, сервисами и т.д.). DTO-интерфейсы инкапсулируют только те поля, которые нужны на конкретном этапе обработки — например, UserLoginDTO содержит только email и password, в то время как JwtResponse — только токен. Такое разделение позволяет защитить чувствительные данные и проводить валидацию входных данных на уровне отдельных DTO-классов.

Репозитории обеспечивают доступ к данным на уровне базы данных. Каждый репозиторий наследует JpaRepository, предоставляя тем самым широкий набор CRUD-операций без необходимости реализовывать их вручную.

5.3.4 Чат-ассистент на основе ИИ

Для реализации чат-ассистента необходимо было подключить языковую модель. Чтобы ответы были более точными и адаптированными под наши задачи, мы провели дообучение модели YandexGPT 5 Lite с использованием датасетов с данными заболеваний, их причин и симптомов. Настройка производилось в облачной платформе Yandex Cloud, где модель была размещена и масштабирована для доступа через API. Для взаимодействия приложения с языковой моделью на сервере реализован класс ChatService.

5.3.5 Конфигурация и безопасность

Отдельно мы выделили конфигурационный уровень, который размещен в папке config. Он содержит классы для управления глобальными настройками проекта, такими как безопасность, фильтрация запросов, параметры подключения.

Важным элементом является система безопасности, реализованная с помощью технологий JWT. В папке мы выделили следующие классы:

— JwtTokenProvider — для генерации и валидации токенов;

— JwtTokenFilter — для перехвата и проверки входящих запросов;

— SecurityConfig — для настройки безопасности.

Благодаря использованию JWT возможно обеспечить безопасную аутентификацию и авторизацию пользователей без необходимости хранения сессий.

6 Реализация интерфейса

6.1 Экраны входа

Экраны входа представлены на рисунке 13. При запуске приложения система предоставляет возможность авторизоваться. Для входа в систему требуется ввести логин (электронную почту) и пароль. Мы предусмотрели удобство пользователя и реализовали функцию отображения скрытого пароля. Для того, чтобы отобразить буквы пароля нужно нажать на иконку «глаз». Ниже размещены кнопки с дополнительными действиями – система предоставляет возможность восстановить пароль, перейти к регистрации и продолжить работу без аккаунта. Основная кнопка «Войти» оформлена в виде активного элемента с заливкой, что подчеркивает ее приоритетное значение.

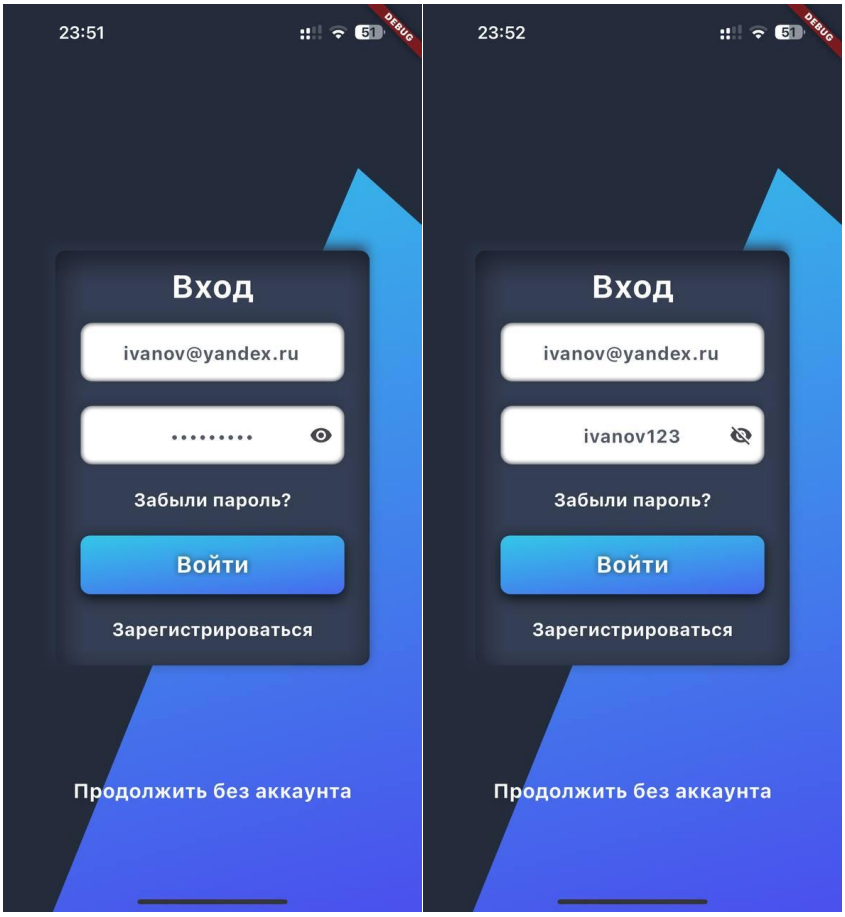

Рисунок 13 — Экраны входа

6.2 Экраны регистрации

Пользователю, желающему зарегистрироваться в приложении, на экране входа необходимо нажать на кнопку «Зарегистрироваться». Система позволяет

создать аккаунт в приложении. На рисунке 14 представлен экран регистрации пользователя.

Для регистрации в форме представлены три поля: для ввода адреса электронной почты, пароля и его подтверждения. Двойной ввод позволяет избавиться от ошибок при вводе пароля. Также реализована возможность скрывать или отображать введённый пароль, что улучшает удобство использования и безопасность. Кнопка «Продолжить» выделена синим цветом, чтобы акцентировать внимание пользователя на следующем шаге процесса регистрации. Под ней расположена ссылка "Войти", позволяющая вернуться к экрану авторизации, если у пользователя уже есть аккаунт.

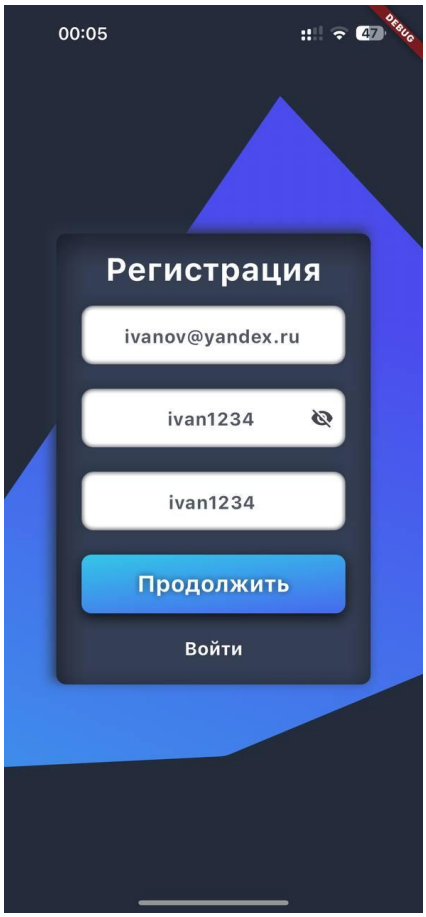

Рисунок 14 — Экран регистрации

При нажатии на кнопку «Продолжить» пользователь переходит на экран, показанный на рисунке 15. Приложение позволяет пользователю заполнить основную информацию для медицинской карты. Необходимо заполнить поля ввода для регистрации. Пользователь должен ввести ФИО, рост, вес, группу

крови, аллергии и заболевания. Мы предусмотрели возможность вернуться назад, если пользователь захотел изменить раннее введенные данные – для этого необходимо нажать на кнопку внизу «Вернуться назад». Кнопка регистрации выделена синим фоном.

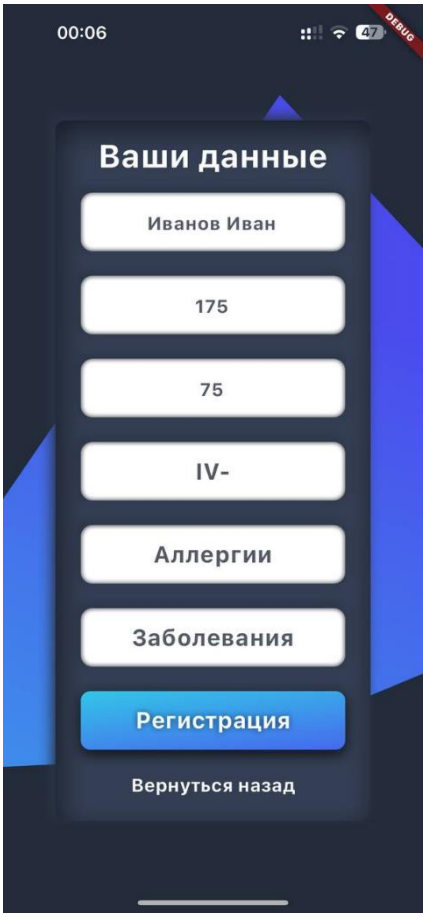

Рисунок 15 — Экран регистрации – заполнение основных данных

6.3 Экраны восстановления пароля

Система реализует функциональность восстановления пароля через email. Пользователь, которому необходимо восстановить пароль, должен на экране входа нажать на кнопку «Забыли пароль?». После нажатия пользователь переходит на экран, показанный на рисунке 16.

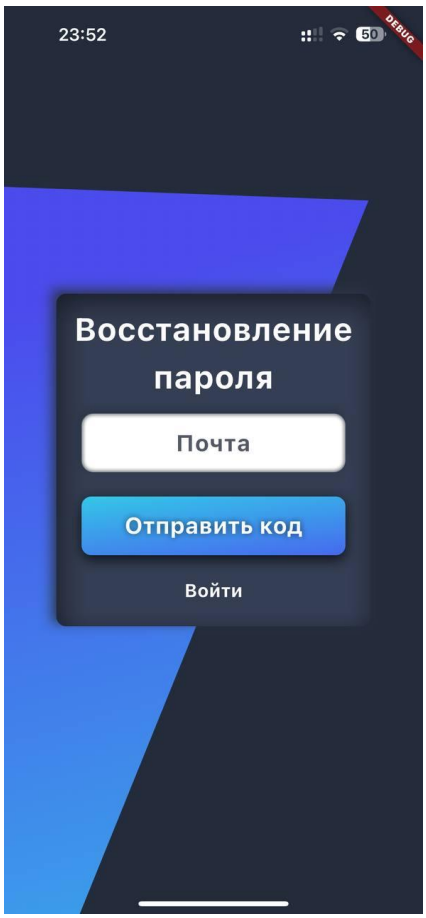

Рисунок 16 – Экран восстановления пароля – ввод почты

Экран восстановления пароля предоставляет пользователю возможность восстановить доступ к своей учётной записи в случае утери пароля. На интерфейсе есть только одно поле для ввода адреса электронной почты, привязанной к аккаунту, на которую будет отправлен код подтверждения. Это обеспечивает быструю процедуру восстановления.

Кнопка «Отправить код» выделяется синим цветом, но, если пользователь вспомнил пароль, в системе реализован функционал возвращения на экран формы, для этого необходимо нажать на кнопку «Войти», которая находится внизу.

После нажатия на кнопку «Отправить код», приложение позволяет пользователю ввести код. Экран для ввода кода показан на рисунке 17.

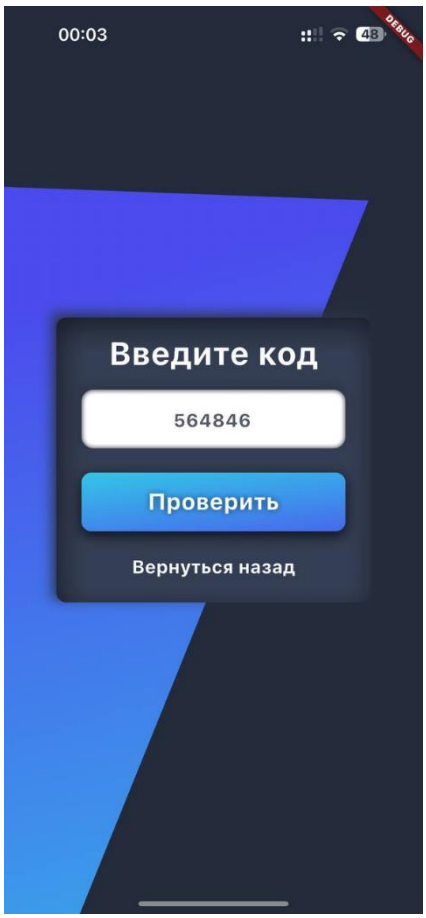

Рисунок 17 – Экран ввода кода для восстановления пароля

В случае неправильного ввода почты, пользователь может вернуться назад. Когда пользователь введет пароль, ему будет необходимо нажать на кнопку «Проверить». Система предоставит возможность ввести новый пароль два раза на экране, показанном на рисунке 18. Нажав на кнопку «Сменить пароль», пользователь перейдет на экран входа, где сможет ввести логин и новый пароль своего аккаунта и войти в приложение.

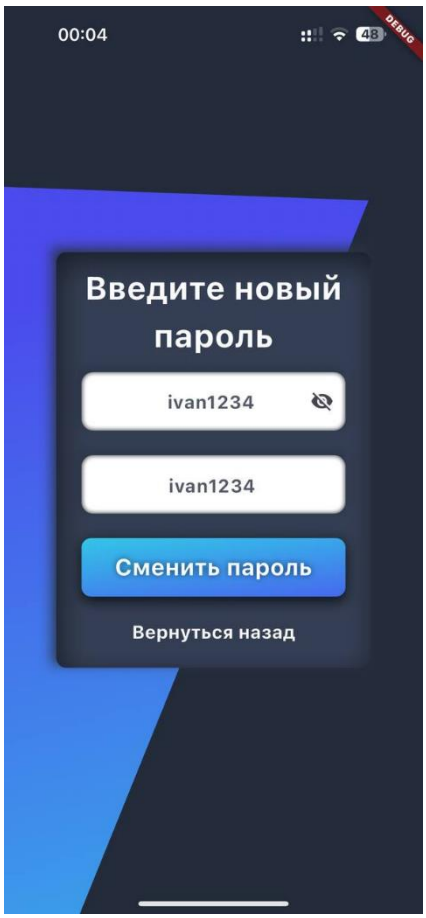

Рисунок 18 – Экран ввода нового пароля

6.4 Экран чата-ассистента

Авторизованный пользователь имеет возможность пользоваться медицинским чат-ассистентом. Он может написать симптомы своей болезни, после чего система сгенерирует предполагаемый диагноз и пояснение. На рисунке 19 представлен экран с примером использования чат-ассистента.

Интерфейс выполнен в виде привычного мессенджера. Сообщения пользователя и ассистента визуально различаются по цвету и стилю оформления: пользовательские — в голубых рамках, ответы системы — фиолетовых. Внизу размещено поле для ввода текста, а также иконки панели навигации, позволяющие быстро переключаться между разделами приложения.

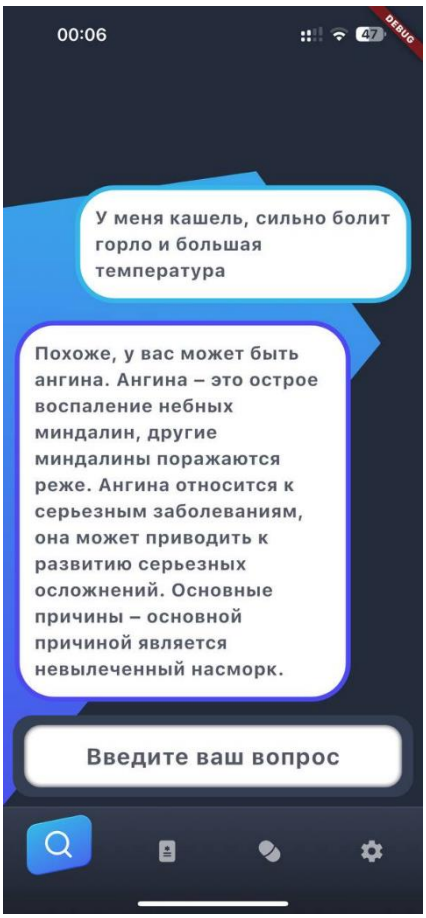

Рисунок 19 – Экран чат-ассистента

Такой подход обеспечивает доступность медицинской информации в интерактивной форме и может быть полезен для предварительной самодиагностики или ориентации в симптомах, особенно в условиях ограниченного доступа к врачу.

6.5 Экраны расписания

Пользователь может перейти в другой раздел приложения, быстро нажав внизу на нужную иконку панели навигации, так он может перейти на экран с расписанием лекарств, показанный на рисунке 20.

На данном экране отображается список назначенных лекарств с указанием дня недели и времени приема. Благодаря такому интерфейсу пользователь может отслеживать прием лекарств, система предоставляет необходимый функционал. В приведённом примере пользователь принимает «Кардиомагнил» по понедельникам и пятницам в 10:00 и 17:00, а препарат «Имудон» — по вторникам и воскресеньям в 12:00 и 21:00.

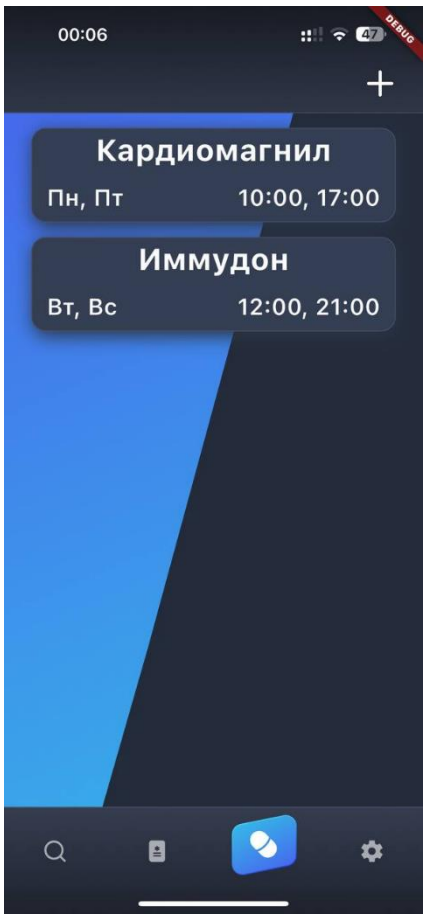

Рисунок 20 – Экран расписания

На экране расписания каждый препарат выделен в отдельной карточке, где дни недели и время структурированы и читаемы. В правом верхнем углу расположен значок «плюс», позволяющий добавить новое расписание. Экраны для добавления нового расписания представлены на рисунке 21.

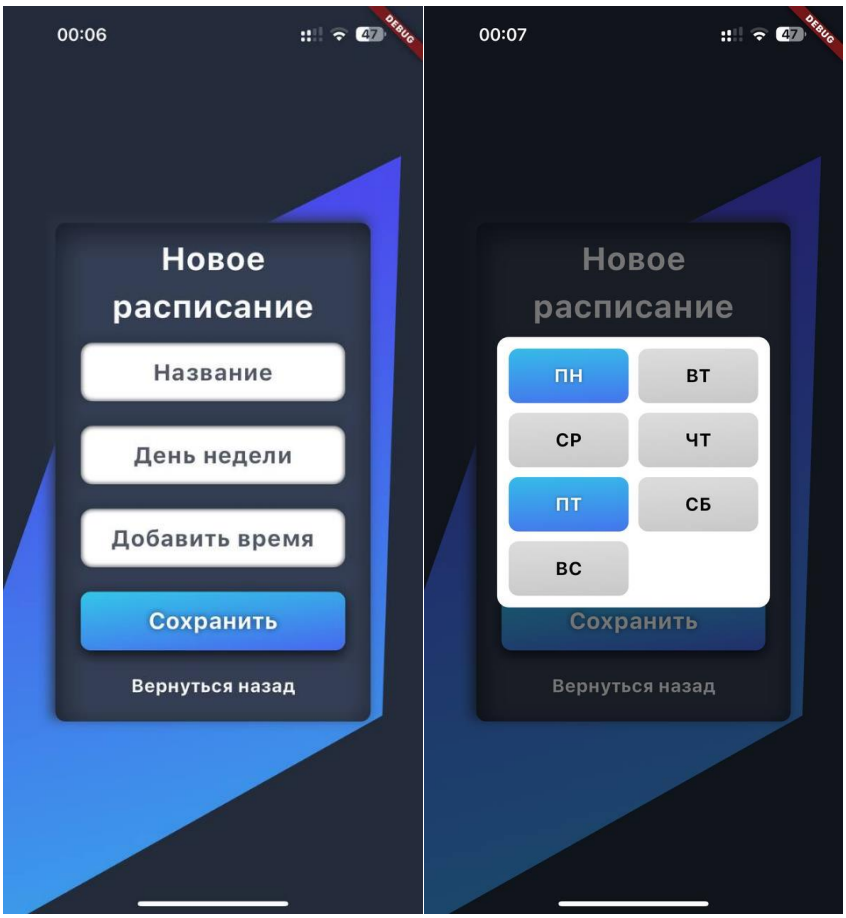

Рисунок 20 – Экраны добавления расписания

На первом экране отображаются три поля: для ввода названия препарата, выбора дня недели и добавления времени приёма. После ввода данных пользователь может сохранить расписание, нажав кнопку «Сохранить». Также предусмотрена возможность вернуться на предыдущий экран.

Второй экран представляет собой форму выбора дня недели. Каждый день представлен в виде интерактивной кнопки, позволяющей выбирать сразу несколько дней. Выбранные дни визуально выделяются синим цветом, это обеспечивает удобство взаимодействия.

Также приложение позволяет удалять расписание, если оно больше неактуально. Экран с такой функциональностью показан на рисунке 22.

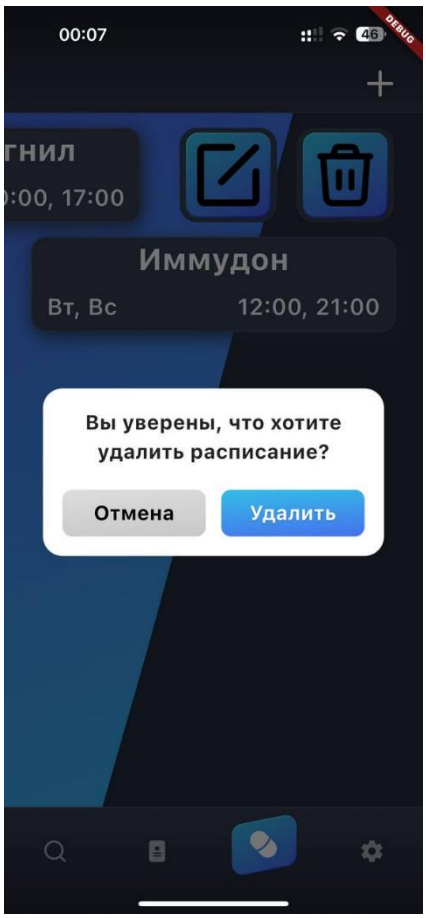

Рисунок 22 – Экран удаления расписания

При нажатии на иконку «Корзина», система предоставляет возможность удалить расписание или отменить выбранное действие. Кнопка «Удалить» в приоритете для пользователя, который нажал на «Корзину», поэтому визуально выделена синим цветом.

6.6 Экраны настроек

На рисунке 23 представлен экран настроек в профиле авторизованного пользователя, интерфейс модуля настроек мобильного приложения. Пользователь может управлять двумя основными параметрами: сменой пароля и уведомлениями, а также выйти из аккаунта.

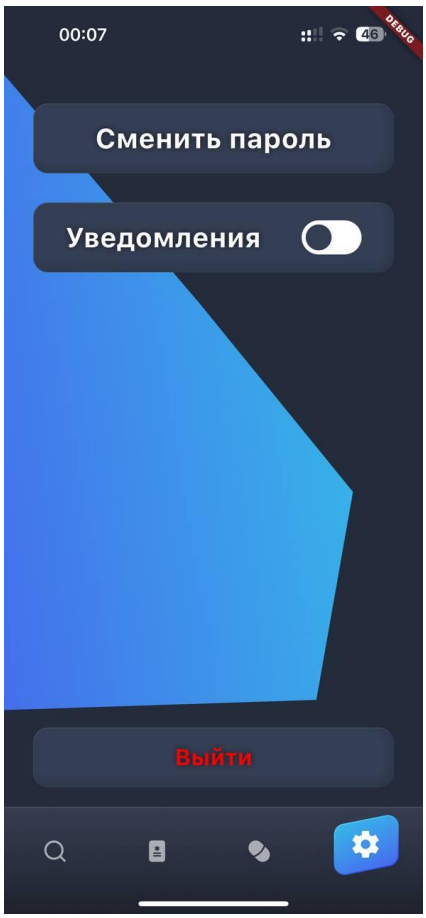

Рисунок 23 – Экран настроек

Верхняя часть экрана содержит кнопку «Сменить пароль», при нажатии на которую пользователь перенаправляется на форму изменения учетных данных. Ниже расположена настройка уведомлений, выполненная в виде переключателя.

6.7 Экран медицинской карты

Каждому авторизованному пользователю доступна медицинская карта. Экран медицинской карты показан на рисунке 24.

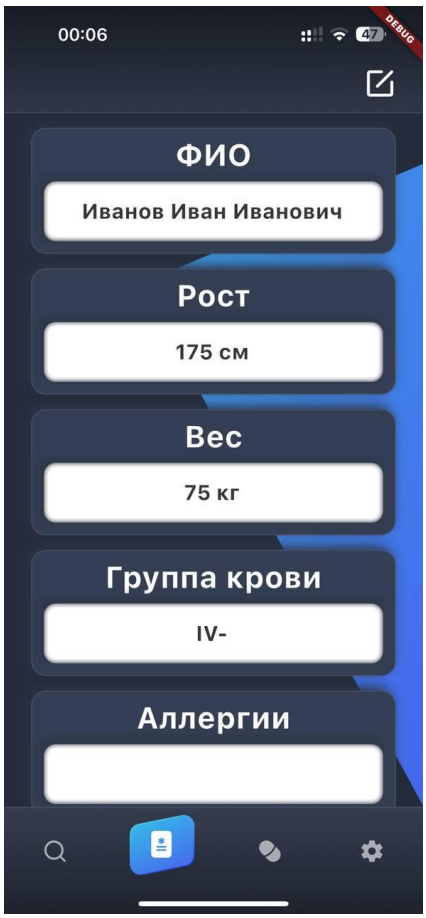

Рисунок 24 – Экран медицинской карты

Экран медицинской карты предоставляет пользователю быстрый доступ к основным медицинским данным, необходимым для ведения персонального профиля в приложении. На экране отображаются следующие поля:

— ФИО – полное имя пользователя;

— Рост – указание роста в сантиметрах;

— Вес – указание массы тела в килограммах;

— Группа крови – информация о группе и резус-факторе;

— Аллергии – наличие аллергических реакций (в данном случае поле пустое).

Каждое поле размещено в отдельной визуальной карточке с чётким разделением и читаемым шрифтом. Верхняя панель содержит иконку редактирования, что говорит о возможности изменения введённых данных.

Навигационное меню внизу позволяет быстро переключаться между другими разделами приложения.

6.8 Виджет SOS

Каждый авторизованный пользователь может создать виджет SOS на главном экране смартфона. При нажатии на него система предоставляет данные медицинской карты. Такой виджет показан на рисунке 25.

Рисунок 25 – Виджет SOS

ЗАКЛЮЧЕНИЕ

В ходе данной работы были выполнены все поставленные цели. Мы изучили предметную область и рассмотрели существующие решения поставленной проблемы.

В результате мы реализовали мобильное приложение медицинского ассистента «PocketHealth», основная функциональность которого включает:

— управление учетными записями;

— работа с медицинской картой;

— управление расписанием приема лекарств;

— взаимодействие с чат-ассистентом;

— использование экстренной кнопки SOS;

— настройки приложения.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Документация Flutter [Электронный ресурс]. – Режим доступа: https://docs.flutter.dev/ – Заглавие с экрана. – (Дата обращения 05.05.2025).

2. Документация Yandex Foundation Models [Электронный ресурс] – Режим доступа: https://yandex.cloud/ru/docs/foundation-models/ - Заглавие с экрана. – (Дата обращения 12.05.2025)

ПРИЛОЖЕНИЕ А
Диаграмма прецендентов

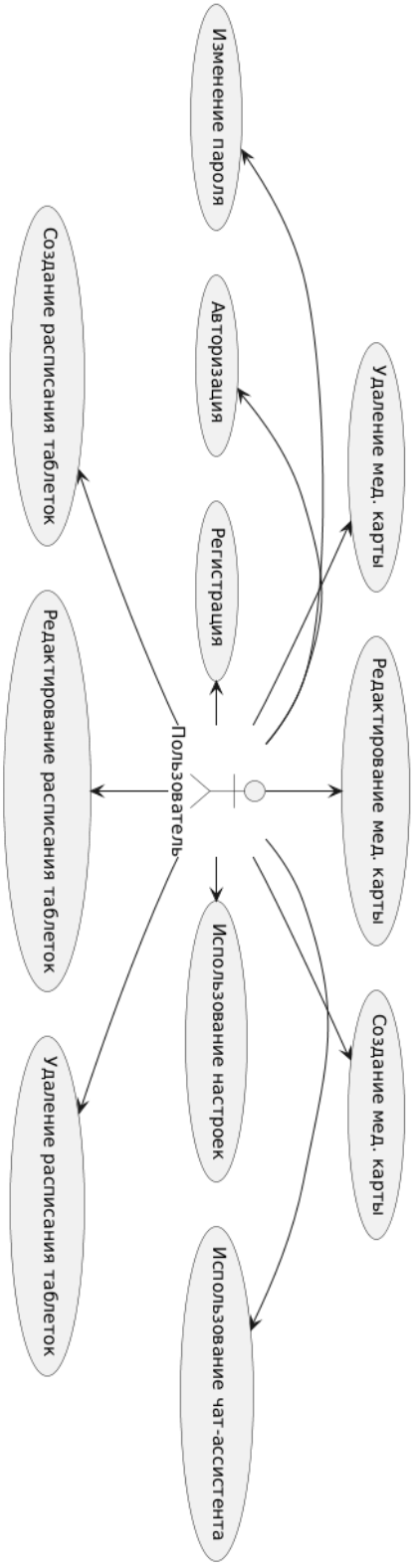

Рисунок А.1 — Диаграмма прецедентов

ПРИЛОЖЕНИЕ Б
ER-диаграмма

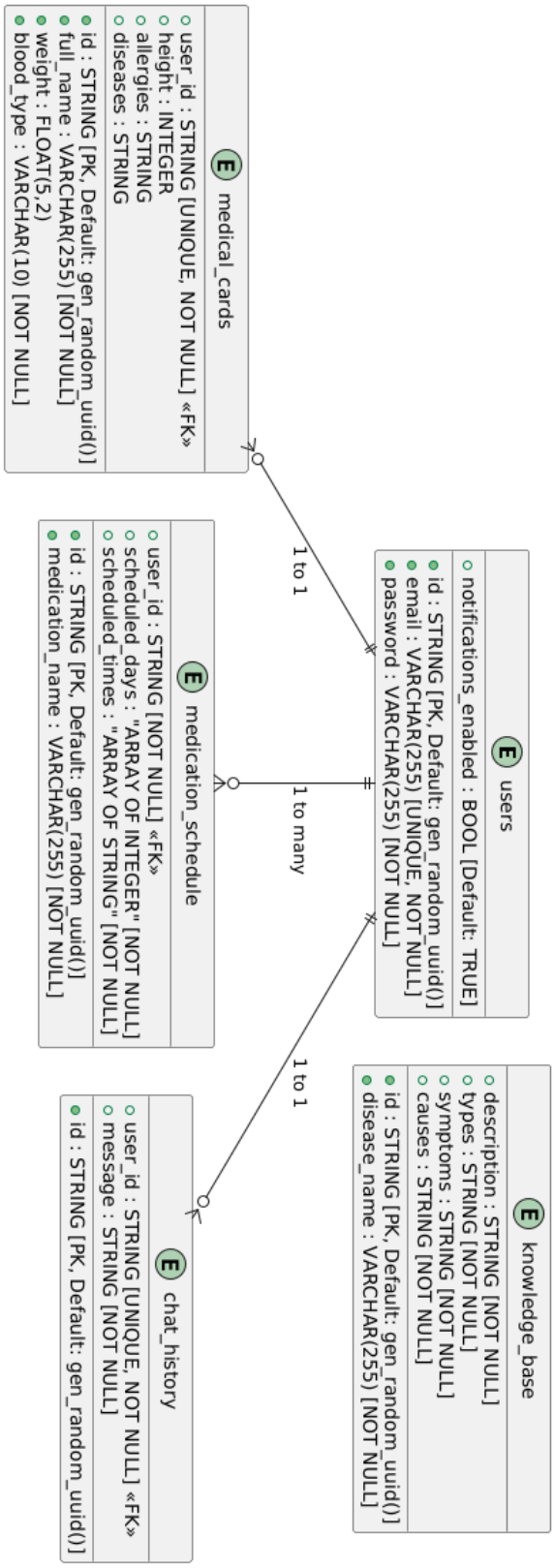

Рисунок Б.1 — ER-диаграмма

ПРИЛОЖЕНИЕ В
Диаграмма последовательности

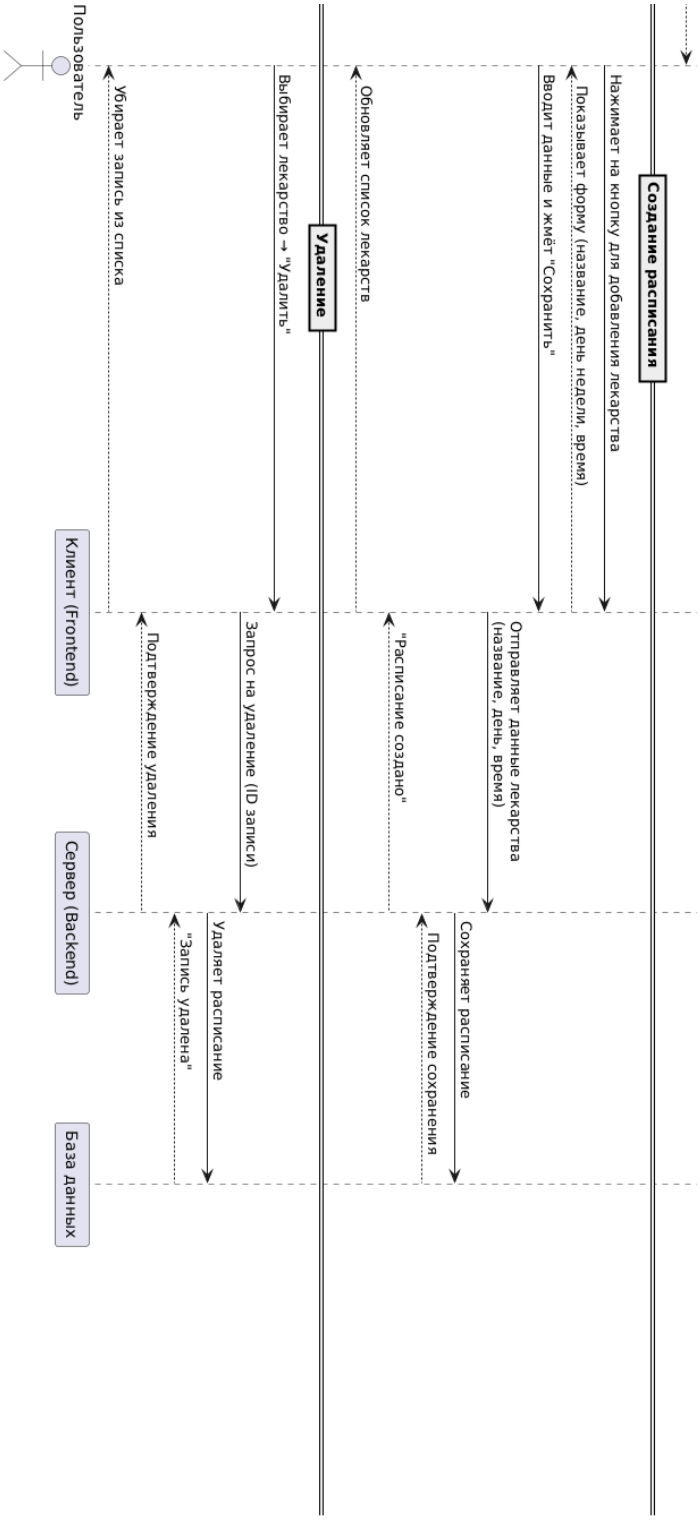

Рисунок В.1 — Диаграмма последовательности