МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение высшего образования
«Тольяттинский государственный университет»

Институт цифровых технологий
(наименование института полностью)

Департамент бакалавриата
(наименование)

09.03.03 Прикладная информатика
(код и наименование направления подготовки / специальности)

Цифровая трансформация бизнеса
(направленность (профиль) / специализация)

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (БАКАЛАВРСКАЯ РАБОТА)

на тему Разработка проекта мобильного приложения для мониторинга индивидуального самочувствия с использованием датчиков и аналитики данных

Обучающийся Ю. А. Коньшина
(Инициалы Фамилия) (личная подпись)

Руководитель Доктор социол. наук, доцент Е.В. Желнина
(ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Тольятти 2025

Аннотация

С. 70, рис. 16, табл. 5, лит. 50 источников

Информационная система, мобильное приложение, мониторинг здоровья, аналитика данных, мфц, python, django, idef0, uml, rest api, база данных.

Разработан проект мобильного приложения для мониторинга индивидуального самочувствия, предназначенного для внедрения на базе Многофункционального центра (МФЦ) в рамках расширения цифровых услуг.

Проведен анализ деятельности МФЦ в сфере здравоохранения, выявлены проблемы существующего реактивного подхода к оказанию услуг. Построены функциональные модели бизнес-процесса «как есть» с использованием методологии IDEF0.

Сформулированы цель, задачи и требования к системе. Произведен выбор и обоснование трехуровневой архитектуры и технологического стека (Python, Django, Streamlit). Разработана логическая модель АИС с использованием нотаций UML и спроектирована реляционная модель базы данных.

Реализован программный прототип, включающий серверную часть с REST API, аналитический модуль для обработки данных и клиентский веб-интерфейс. Проведено комплексное модульное и интеграционное тестирование программного обеспечения для подтверждения его корректной работы и функциональности.

Работа представляет собой готовый к пилотному развертыванию прототип; рассматривается вопрос о его дальнейшем развитии и внедрении.

Оглавление

Введение

В современных условиях цифровой трансформации, охватывающей все сферы общественной жизни, особое значение приобретает модернизация системы здравоохранения. Одним из ключевых направлений развития, обозначенных в рамках национального проекта «Здравоохранение», является внедрение инновационных технологий для персонализированной медицины и превентивных мер, направленных на сохранение здоровья граждан. Возрастает роль проактивного подхода, при котором акцент смещается с лечения уже возникших заболеваний на их раннюю диагностику и профилактику. В этом контексте разработка мобильных приложений для мониторинга индивидуального самочувствия становится не просто технологической тенденцией, а насущной необходимостью, позволяющей повысить качество и доступность медицинских услуг.

Актуальность данной работы обусловлена наличием существенного разрыва между возможностями современных технологий и их интеграцией в государственную систему здравоохранения. Несмотря на существование множества коммерческих и государственных приложений, на рынке отсутствует комплексное решение, которое бы объединяло непрерывный сбор данных с персональных датчиков, интеллектуальный анализ этих данных для формирования персонализированных рекомендаций и бесшовную интеграцию с государственными информационными системами, такими как портал «Госуслуги» и медицинские информационные системы. Существующие продукты либо сфокусированы на узких задачах (например, фитнес-трекинг), либо не обеспечивают необходимого уровня аналитики и персонализации, что создает барьеры для их массового внедрения и использования в качестве эффективного инструмента профилактической медицины. Таким образом, разработка проекта такого интегрированного мобильного приложения является высокоактуальной задачей, отвечающей

стратегическим целям развития цифрового здравоохранения в Российской Федерации.

Целью настоящей выпускной квалификационной работы является разработка проекта мобильного приложения для мониторинга индивидуального самочувствия, которое использует данные с носимых датчиков и методы аналитики для формирования персонализированных рекомендаций.

Для достижения поставленной цели в работе были поставлены следующие задачи:

- провести анализ предметной области, включая исследование существующих коммерческих и государственных решений для мониторинга здоровья;
- выполнить функциональное моделирование бизнес-процессов «как есть» и разработать модель «как должно быть» с использованием нового приложения;
- спроектировать системную архитектуру приложения, разработана его информационная модель и логическая структура базы данных;
- разработать программный прототип приложения с использованием современных технологий, включающий серверную часть, пользовательский интерфейс и модуль анализа данных;
- провести тестирование ключевых функций программного прототипа на контрольном примере.

Объектом исследования выступают информационные процессы, связанные с мониторингом индивидуального самочувствия граждан, сбором, обработкой и анализом медицинских данных.

Предметом исследования являются методы и средства проектирования и разработки программного обеспечения для автоматизации процессов мониторинга здоровья, включая архитектурные решения, модели данных, алгоритмы анализа и технологии реализации пользовательского интерфейса.

Теоретико-методологической основой исследования послужил комплексный подход, сочетающий в себе несколько методологий. На этапе анализа и проектирования применялся системный и структурный подходы. Для моделирования бизнес-процессов использовалась методология функционального моделирования IDEF0, а для разработки логической модели системы и ее компонентов – унифицированный язык моделирования UML (Unified Modeling Language). Практическая реализация прототипа основывалась на принципах объектно-ориентированного программирования (ООП) с использованием языка программирования Python, веб-фреймворка Django для серверной части, системы управления базами данных SQLite и библиотеки Streamlit для создания интерактивного пользовательского интерфейса.

Практическая значимость полученных результатов заключается в создании проекта готового к внедрению цифрового инструмента, способного оказать существенное положительное влияние как на систему здравоохранения, так и на качество жизни граждан. Внедрение подобного приложения позволит снизить нагрузку на медицинские учреждения за счет ранней профилактики заболеваний, повысить информированность населения о состоянии своего здоровья и сформировать культуру ответственного отношения к нему. Экономическая эффективность проекта, рассчитанная в работе, подтверждает его высокую рентабельность инвестиций (637%) и короткий срок окупаемости (менее двух месяцев), что делает его привлекательным для реализации в рамках государственных программ.

Основные положения и результаты работы прошли апробацию в ходе разработки и тестирования программного прототипа. Функциональность системы была проверена на контрольном примере, который продемонстрировал корректную работу алгоритмов сбора данных, их анализа и формирования персонализированных рекомендаций на основе введенных показателей здоровья.

Данная работа выполнялась по инициативе автора. Структурно работа состоит из введения, трех глав, заключения, списка использованных источников и приложений. В первой главе, «Функциональное моделирование предметной области», проводится технико-экономический анализ сферы цифрового здравоохранения, исследуются существующие аналоги, моделируются бизнес-процессы в нотации IDEF0 и формулируется постановка задачи на разработку.

Во второй главе, «Логическое проектирование АИС», описывается выбор технологий, разрабатывается логическая модель системы с использованием диаграмм UML, проектируется информационное обеспечение и структура базы данных. В третьей главе, «Физическое проектирование АИС», осуществляется выбор архитектуры и технологий реализации, описывается разработка программных модулей на Python и Django, создание пользовательского интерфейса и проводится тестирование разработанного прототипа. В заключении подводятся итоги проделанной работы, формулируются основные выводы и намечаются перспективы дальнейшего развития проекта.

Глава 1 Функциональное моделирование предметной области

1.1 Технико-экономическая характеристика предметной области

В качестве предметной области для разработки проекта выступает деятельность Многофункционального центра предоставления государственных и муниципальных услуг (МФЦ) [1]. Данное учреждение создано с ключевой целью – предоставлять государственные и муниципальные услуги по принципу «одного окна». Основная миссия МФЦ заключается в оптимизации взаимодействия граждан с государственными структурами, минимизации их временных и ресурсных затрат при получении услуг, а также в общем повышении качества и доступности этих услуг.

Стратегическим направлением развития МФЦ является внедрение передовых цифровых технологий. В рамках этой стратегии МФЦ реализует пилотные проекты в социально значимых сферах, включая здравоохранение. Эти проекты нацелены на повышение доступности медицинских услуг и усиление мер по профилактике заболеваний среди населения [2]. Инициатива по разработке мобильного приложения для мониторинга индивидуального самочувствия является логичным продолжением этого курса и полностью согласуется с общей стратегией цифровизации государственных сервисов.

Организационная структура МФЦ представляет собой линейно-функциональную модель, которая дополнена элементами проектной организации для эффективной реализации инновационных задач. Структура включает руководство (директор и заместители), а также профильные подразделения: административный отдел, отдел приема и выдачи документов, отдел информационных технологий, правовой отдел, отдел контроля качества и, что особенно важно для данного проекта, отдел цифровой трансформации [3]. Схема организационной структуры представлена на рисунке 1.

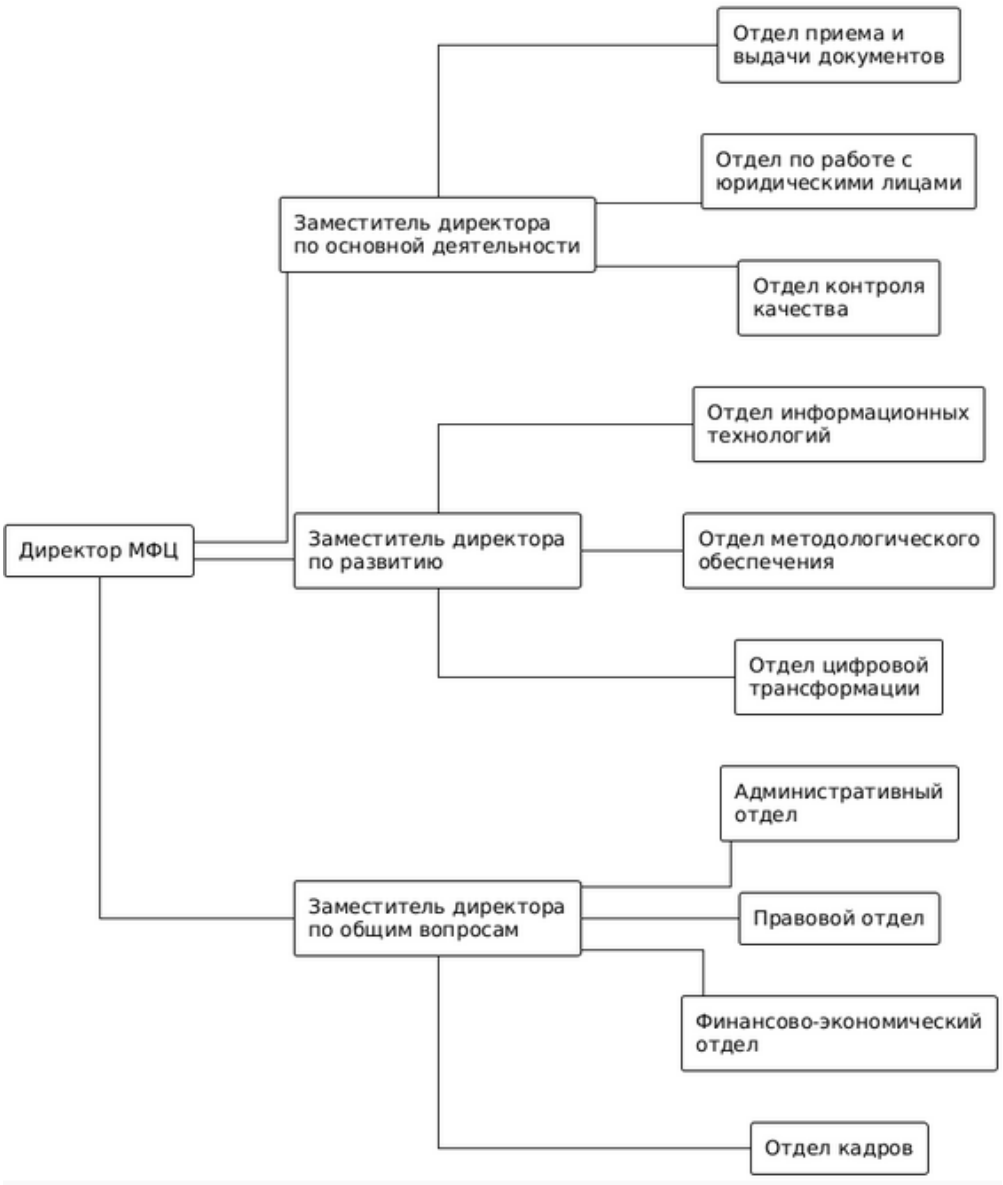

Рисунок 1 – Организационная структура МФЦ

Проект по разработке мобильного приложения для мониторинга самочувствия реализуется на базе отдела цифровой трансформации при активном участии специалистов из отдела информационных технологий. Для выполнения поставленных задач сформирована междисциплинарная проектная группа [4], в которую вошли представители ключевых подразделений МФЦ.

Для обоснования необходимости автоматизации и разработки нового решения необходимо проанализировать технико-экономические показатели

текущего бизнес-процесса МФЦ в части предоставления услуг, связанных со здравоохранением («модель как есть»). Главными объектами управления здесь являются процессы приема и обработки заявлений граждан на получение медицинских услуг. Субъектами управления выступают специалисты и руководство МФЦ.

Основные технико-экономические показатели существующего бизнес-процесса и его проблемные зоны сведены в таблицу 1.

Таблица 1 – Технико-экономические показатели бизнес-процесса «как есть»

Показатель	Характеристика текущего состояния	Ограничения и количественно-стоимостные оценки
Цель и результаты деятельности	Обработка заявлений граждан для получения административных медицинских услуг (запись к врачу, получение справок).	Деятельность носит реактивный характер. Функции профилактики и мониторинга здоровья полностью отсутствуют.
Продукция и услуги	Консультирование, прием и выдача документов, передача данных в медицинские учреждения.	Услуги ограничиваются административными процедурами, не создавая дополнительной ценности для здоровья граждан.
Основные этапы и процессы	Личное посещение МФЦ, подача бумажных документов, межведомственная обработка, повторное посещение для получения результата.	Процесс требует значительных временных затрат от гражданина. Срок ожидания результата составляет от 5 до 10 дней.
Используемые ресурсы	Высокие трудозатраты специалистов на ручную обработку документов. Бумажный документооборот.	Отсутствуют инструменты для персонализированного подхода. Высокие операционные расходы на обработку одного обращения.
Качество и оперативность	Низкая оперативность из-за длительного цикла обработки. Достоверность информации может снижаться из-за человеческого фактора.	Возможности оперативного реагирования на изменения в самочувствии граждан полностью отсутствуют.

Приведенные в таблице показатели наглядно демонстрируют низкую эффективность существующей модели. Отсутствие систематического мониторинга здоровья, необходимость личного присутствия и длительные сроки обработки создают барьеры для граждан и не позволяют реализовать превентивный подход к здравоохранению. Таким образом, существует острая

и экономически обоснованная необходимость в разработке автоматизированного решения, которое позволит трансформировать текущий процесс, устранить выявленные недостатки и создать принципиально новую цифровую услугу.

1.2 Концептуальное моделирование предметной области

1.2.1 Выбор технологии концептуального моделирования предметной области

Концептуальное моделирование является критически важным этапом проектирования, позволяющим формализовать и структурировать знания о предметной области. Этот процесс включает в себя анализ существующих бизнес-процессов и формирование четкого видения будущей, усовершенствованной системы. Результаты данного этапа служат фундаментом для всех последующих проектных решений.

Для всестороннего анализа и описания предметной области был выбран комбинированный подход, который интегрирует две ведущие методологии моделирования: IDEF0 [5] и UML [6]. Такой выбор обусловлен тем, что каждая из этих технологий позволяет сфокусироваться на разных аспектах исследуемой системы, эффективно дополняя друг друга.

Методология IDEF0 (Integration Definition forFunctionModeling) была выбрана для проведения функционального моделирования и анализа существующего бизнес-процесса, то есть для построения модели «как есть». Главным преимуществом IDEF0 является ее способность наглядно представлять систему в виде иерархии взаимосвязанных функциональных блоков [7]. Нотация четко показывает входы, выходы, управляющие воздействия и механизмы для каждого процесса, что делает ее идеальным инструментом для структурного анализа и выявления узких мест в текущей деятельности МФЦ.

В свою очередь, унифицированный язык моделирования UML (Unified Modeling Language) будет применяться на последующих этапах для объектно-ориентированного анализа и проектирования новой системы. В отличие от IDEF0, которая фокусируется на функциях, UML позволяет описать систему с точки зрения объектов, их атрибутов и взаимодействия [8]. В частности, диаграммы вариантов использования будут использованы для определения функциональных требований к приложению. Совместное использование этих технологий обеспечивает комплексный подход, от высокоуровневого анализа до детального проектирования.

Первоочередной задачей концептуального моделирования является детальное описание и анализ текущего порядка предоставления МФЦ услуг, связанных со сферой здравоохранения. Этот процесс, определенный как модель «как есть» (AS-IS) [9], служит отправной точкой для выявления существующих проблем, неэффективных операций и скрытых затрат. Глубокое понимание текущей ситуации позволяет не просто автоматизировать существующие шаги, а провести их реинжиниринг, создав качественно новую, более эффективную модель взаимодействия гражданина с системой здравоохранения через цифровую платформу.

1.2.2 Моделирование бизнес-процессов предметной области для постановки задачи автоматизированного варианта решения

На сегодняшний день процесс получения медицинских услуг через МФЦ носит исключительно административный и очный характер. Он не включает в себя элементы мониторинга здоровья или проактивной профилактики. Весь процесс сводится к последовательности ручных операций по обработке бумажных документов [10]. Гражданин инициирует процесс, лично посещая отделение МФЦ с пакетом документов. Специалист принимает заявление, проверяет его и запускает внутреннюю процедуру обработки, которая включает передачу данных в соответствующее медицинское учреждение. По завершении обработки, которая может

занимать до 10 дней, гражданин должен повторно посетить МФЦ для получения итогового документа.

Для формализованного описания этого процесса была построена контекстная диаграмма IDEF0 (уровень А-0) [11], представленная на рисунке 2. Она отражает процесс в целом, его границы и взаимодействие с внешней средой.

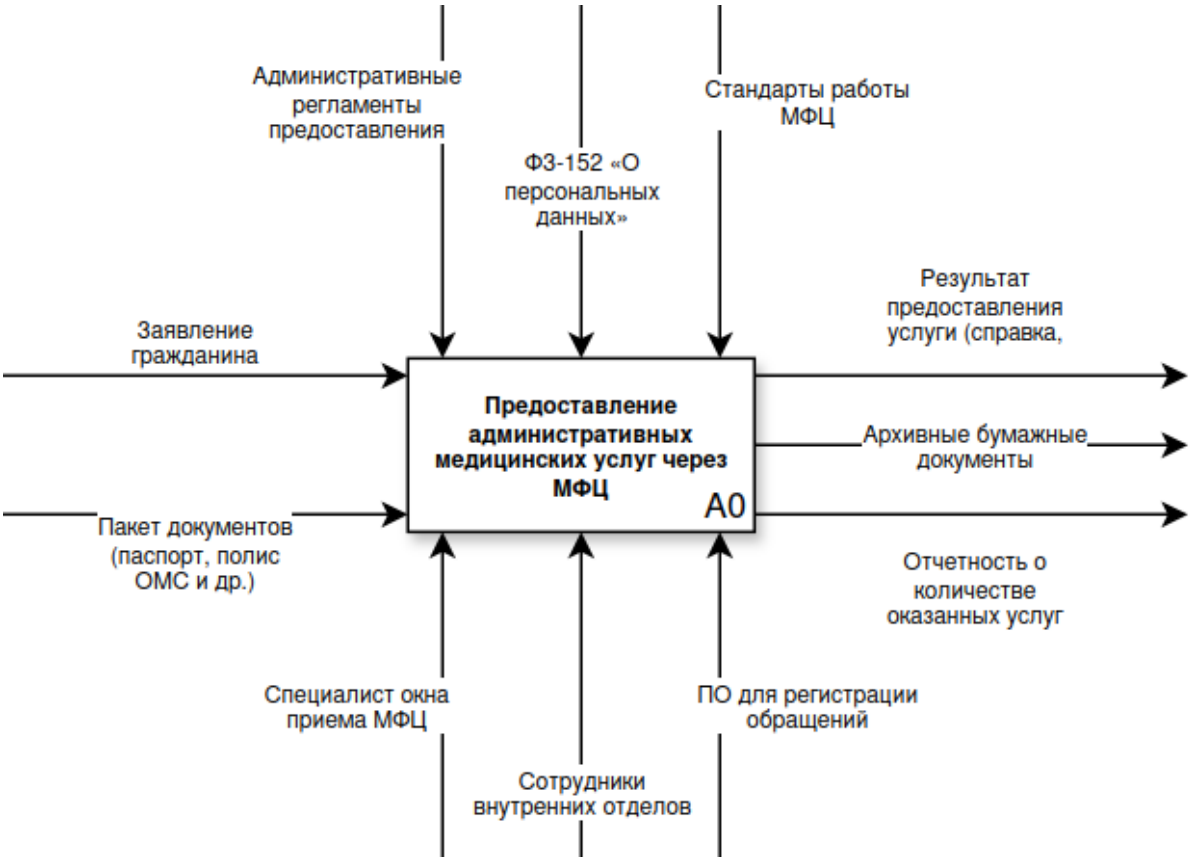

Рисунок 2 – Контекстная диаграмма (А-0) процесса «Предоставление административных медицинских услуг через МФЦ» (модель «как есть»)

Теперь стоит перейти к анализу модели, представленном в следующем разделе.

1.2.3 Разработка и анализ модели бизнес-процесса «как есть»

Для более детального анализа была выполнена декомпозиция контекстной диаграммы на три основных подпроцесса, которые отражают

ключевые этапы жизненного цикла обращения [12]. Диаграмма декомпозиции (уровень A0) представлена на рисунке 3.

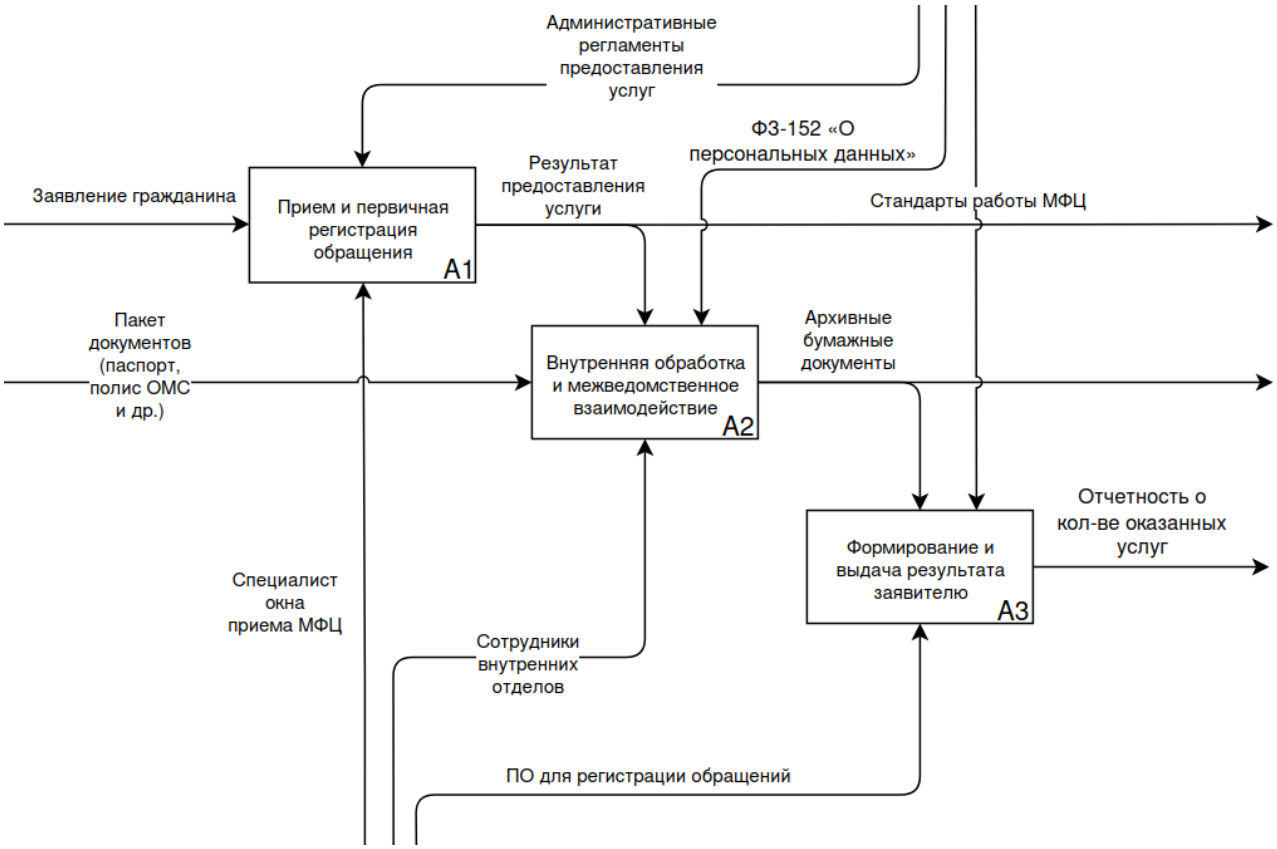

Рисунок 3 – Диаграмма декомпозиции (A0) процесса «Предоставление административных медицинских услуг через МФЦ»

Анализ построенной модели «как есть» выявляет ряд системных недостатков. Во-первых, это высокая трудоемкость и низкая производительность, обусловленные зависимостью от ручных операций. Во-вторых, низкая оперативность, так как длительный цикл обработки и необходимость двух личных визитов делают услугу медленной. В-третьих, полное отсутствие проактивности, поскольку система работает исключительно в реактивном режиме. Наконец, несовершенство сбора и хранения информации на бумажных носителях повышает риски и затрудняет анализ.

1.2.4 Обоснование необходимости автоматизированного варианта решения и формирование требований к новой технологии

Проведенный анализ убедительно доказывает, что существующая модель предоставления услуг устарела и не отвечает современным вызовам. Ее недостатки приводят не только к операционной неэффективности, но и упускают огромный потенциал в области превентивной медицины. Необходимость внедрения автоматизированного решения продиктована потребностью в переходе от реактивной модели к проактивной, ориентированной на вовлечение граждан в процесс управления собственным здоровьем.

На основе выявленных недостатков были сформулированы ключевые требования к новой технологии. Система должна обеспечивать автоматизированный сбор данных о показателях здоровья с персональных датчиков. Новая технология должна включать аналитический модуль, способный в реальном времени анализировать данные и выявлять риски. Вместо стандартных процедур, система должна формировать персонализированные рекомендации. Приложение должно стать единой точкой доступа к сервису, доступной круглосуточно. Наконец, решение должно быть интегрировано с государственными системами, такими как ЕСИА [13], и обеспечивать высокий уровень безопасности персональных данных. Эти требования ложатся в основу постановки задачи на разработку и проектирование модели «как должно быть».

1.3 Анализ существующих разработок на предмет соответствия сформулированным требованиям

1.3.1 Определение критериев анализа

Для принятия обоснованного решения о необходимости разработки нового программного продукта был проведен анализ рынка существующих информационных систем, предназначенных для мониторинга здоровья.

Анализ сфокусирован на сопоставлении функциональных и технологических возможностей аналогов с ключевыми требованиями, сформулированными в предыдущем разделе.

На основе требований, выработанных в пункте 1.2.4, были определены следующие критерии для сравнительного анализа существующих программных решений.

Первым критерием является функциональная полнота мониторинга. Он оценивает способность приложения собирать и обрабатывать данные с широкого спектра медицинских датчиков (пульсометры, тонометры, глюкометры), а не только фитнес-показатели.Вторым критерием выступает качество аналитики и персонализации. Здесь оценивается наличие в системе интеллектуальных алгоритмов для анализа собранных данных, выявления индивидуальных паттернов, рисков и автоматического формирования персонализированных, а не общих, рекомендаций.

Третий, и ключевой для данного проекта, критерий – интеграция с государственными информационными системами (ГИС) РФ [14]. Этот параметр определяет возможность бесшовной аутентификации пользователя через ЕСИА («Госуслуги») и потенциальную совместимость с медицинскими информационными системами (МИС) для обмена данными.Четвертый критерий – технологическая доступность и кроссплатформенность, который отражает доступность приложения для широкого круга пользователей на основных мобильных операционных системах (iOS, Android).

Пятый критерий – безопасность и соответствие законодательству РФ. Он оценивает, обеспечивает ли решение надлежащий уровень защиты персональных медицинских данных согласно требованиям Федерального закона № 152-ФЗ «О персональных данных» [15].

Данный набор критериев позволяет провести комплексную и объективную оценку существующих разработок.

1.3.2 Сравнительная характеристика существующих разработок

Рынок приложений для мониторинга здоровья можно условно разделить на две большие категории: коммерческие продукты от крупных технологических компаний и государственные разработки.

Крупные зарубежные коммерческие решения, такие как Apple Health, Google Fit и Samsung Health [16], обладают развитой экосистемой, хорошей интеграцией с носимыми устройствами и высоким качеством пользовательского интерфейса. Однако их основной недостаток – полная оторванность от российской государственной системы здравоохранения. Они не поддерживают аутентификацию через ЕСИА и не могут быть интегрированы с отечественными медицинскими информационными системами. Кроме того, их аналитические возможности часто ограничены фитнес-целями и не предоставляют глубокого медицинского анализа. Отечественные коммерческие аналоги, например «Здоровье.ру», хотя и адаптированы к российскому рынку, как правило, обладают ограниченной функциональностью и не предлагают сложной аналитики данных.

Государственные разработки, такие как «ЕМИАС.ИНФО» и «Госуслуги Здоровье», имеют важное преимущество – они изначально интегрированы в государственную ИТ-инфраструктуру [17]. Однако их функциональность носит преимущественно административный характер: запись к врачу, просмотр медицинских документов, получение справок. Функции для непрерывного мониторинга самочувствия с использованием датчиков и формирования персонализированных рекомендаций на основе аналитики в них практически отсутствуют.

Результаты сравнительного анализа сведены в таблицу 2.

Анализ таблицы показывает, что на рынке отсутствует единое решение, удовлетворяющее всем сформулированным требованиям. Коммерческие продукты сильны в мониторинге, но не имеют государственной интеграции. Государственные сервисы интегрированы, но лишены функций мониторинга и аналитики. Этот разрыв и определяет нишу для разрабатываемого приложения.Покупка и доработка существующего решения представляется

нецелесообразной. Адаптация зарубежных продуктов невозможна из-за их закрытой архитектуры и несоответствия законодательству РФ [18]. Модификация существующих государственных порталов потребовала бы их полного архитектурного перепроектирования, что сопоставимо с разработкой нового продукта.

Таблица 2 – Сравнительный анализ аналогов по установленным критериям (Обозначения: «+» – полное соответствие; «±» – частичное соответствие; «–» – несоответствие.)

Критерий / Аналог	Apple Health / Google Fit	ЕМИАС.ИНФО / Госуслуги Здоровье	Проектируемая система
Функциональная полнота мониторинга	+	–	+
Качество аналитики и персонализации	±	–	+
Интеграция с ГИС РФ (ЕСИА, МИС)	–	+	+
Технологическая доступность	±	+	+
Безопасность (согласно ФЗ-152)	–	+	+

Как следствие, наиболее корректным и экономически оправданным путем является создание нового программного обеспечения с нуля. Выбор технологического стека Python + Django + SQLite для этой задачи обоснован тем, что он обеспечивает необходимую гибкость, скорость разработки и широкие возможности для реализации сложной бизнес-логики. Python обладает мощными библиотеками для анализа данных, что критически важно для аналитического модуля, а фреймворк Django позволяет быстро создать надежную и масштабируемую серверную часть, готовую к интеграции с государственными информационными системами.

1.4 Постановка задачи на разработку проекта создания/внедрения АИС

Постановка задачи является основополагающим документом, который формализует цели, назначение и требования к разрабатываемой автоматизированной информационной системе (АИС). Она базируется на результатах анализа предметной области, выявленных недостатках существующей модели и сформулированных потребностях в новом технологическом решении.

Основной целью разработки является создание программного продукта, который устранит ключевые недостатки текущей системы взаимодействия граждан со сферой здравоохранения, переведя ее из реактивного формата в проактивный. Это достигается через решение двух взаимосвязанных задач: во-первых, создание принципиально новой цифровой услуги, которая повысит информированность граждан о состоянии их здоровья и будет способствовать раннему выявлению рисков заболеваний; во-вторых, кардинальное улучшение качества обработки информации за счет перехода от медленного бумажного документооборота к полностью автоматизированному цифровому процессу. Назначением разрабатываемой АИС является автоматизация полного цикла процессов по мониторингу индивидуального самочувствия, начиная от ввода и контроля данных о показателях здоровья до выполнения аналитических расчетов и выдачи персонализированных рекомендаций в удобной для пользователя форме [19].

Функционально система должна работать в непрерывном диалоговом режиме. Источниками оперативной информации служат как данные, вводимые пользователем вручную через экранные формы «Добавить измерение» и «Добавить симптом», так и автоматизированные потоки данных с персональных носимых датчиков. Эта информация в режиме реального времени поступает в базу данных и обрабатывается аналитическим модулем на сервере, который запускает алгоритмы для оценки пульса,

артериального давления, расчета ИМТ и других показателей. Результаты анализа незамедлительно представляются пользователю на главном экране – «Дашборде» – в виде сводных карточек, интерактивных графиков и списка персонализированных рекомендаций. Весь информационный обмен поддерживается реляционной базой данных под управлением СУБД SQLite, которая хранит как условно-постоянную информацию (медицинские профили пользователей), так и оперативные данные (измерения, симптомы) [20], при этом целостность и конфиденциальность данных обеспечиваются современными мерами безопасности, включая токен-аутентификацию.

С формальной точки зрения, задачу, решаемую системой, можно описать как задачу оптимизации, где целевой функцией является максимизация полезности и своевременности информации о здоровье для пользователя. Система должна стремиться минимизировать временной лаг между фиксацией потенциально значимого отклонения в показателях и получением пользователем соответствующего уведомления. Данная оптимизация должна производиться при соблюдении строгих ограничений, накладываемых законодательством о защите персональных данных (ФЗ-152) и требованиями к производительности и отказоустойчивости системы.

Для реализации этих требований архитектура АИС должна быть построена по трехзвенной модели «клиент-сервер», обеспечивающей необходимое разделение логики, представления и хранения данных, а также создающей основу для будущего масштабирования. Выбор технологического стека основан на принципах использования бесплатно распространяемого ПО и гибкости. Уровень бизнес-логики реализуется на языке Python с использованием фреймворка Django и Django REST Framework [21][22], что обусловлено его надежностью и богатыми возможностями для анализа данных. Уровень данных на этапе прототипирования использует СУБД SQLite, при этом применение Django ORM гарантирует легкий переход на более производительные базы данных, такие как PostgreSQL, по мере роста проекта [23]. Такой подход обеспечивает модульность, экономическую

эффективность и высокий потенциал для дальнейшей интеграции системы в более крупные экосистемы цифрового здравоохранения.

1.5 Разработка модели бизнес-процесса «как должно быть»

Результатом проведенного анализа является концептуальная модель усовершенствованного бизнес-процесса «как должно быть» (TO-BE). Эта модель кардинально трансформирует существующий порядок взаимодействия, смещая фокус с оказания реактивных административных услуг на предоставление проактивного цифрового сервиса по мониторингу здоровья. Разрабатываемая система устраняет выявленные недостатки, такие как низкая оперативность, высокая трудоемкость и отсутствие превентивных функций, создавая принципиально новый, высокоэффективный технологический процесс.

Сущность нового процесса заключается в непрерывном, автоматизированном сборе и интеллектуальном анализе данных о состоянии здоровья пользователя с целью формирования персонализированных рекомендаций и раннего оповещения о потенциальных рисках. В отличие от существующей модели, где периодичность взаимодействия определялась возникновением у гражданина конкретной потребности, новый процесс является перманентным и работает в режиме реального времени [24]. Это обеспечивает высочайшую оперативность и достоверность информации, так как данные поступают напрямую с датчиков или от пользователя, минимизируя влияние человеческого фактора и задержки, свойственные бумажному документообороту.

Информационно-технологическая схема усовершенствованного процесса представлена в виде диаграммы декомпозиции IDEF0 на Рисунке 4. Модель описывает три ключевых, последовательно выполняемых этапа: сбор и регистрация данных, их аналитическая обработка и, наконец, визуализация результатов для пользователя.

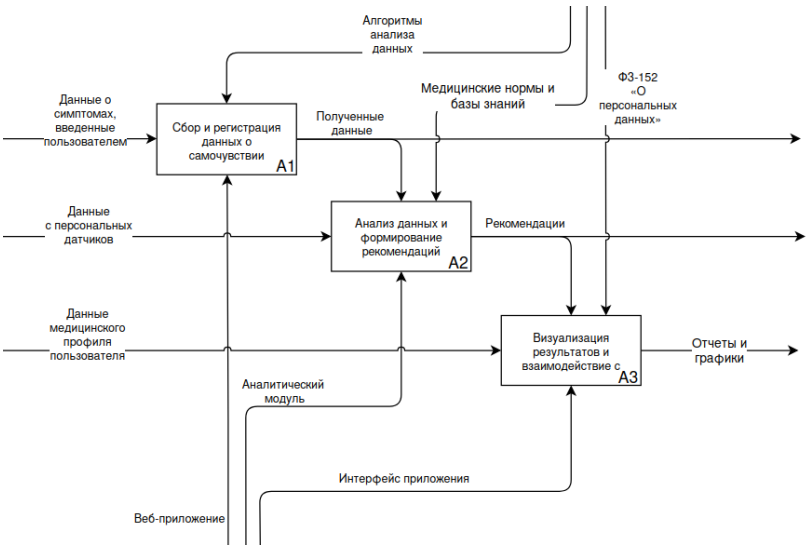

Рисунок 4 – Диаграмма декомпозиции (A0) процесса «Проактивный мониторинг индивидуального самочувствия» (модель «как должно быть»)

Входной информацией для нового процесса служат уже не бумажные заявления, а цифровые потоки данных. К ним относятся файлы и сообщения от персональных датчиков, передаваемые по защищенным протоколам, а также структурированные данные, которые пользователь вводит через экранные формы приложения, описывая свое самочувствие. Эта информация поступает в базу данных системы для дальнейшей обработки. Выходной информацией являются не статические документы, а динамически генерируемые отчеты, справки и рекомендации. Они представляются пользователю в виде интерактивных графиков на дашборде приложения, текстовых сообщений с советами и push-уведомлений в случае выявления критических отклонений от нормы.

Модель решения задачи представляет собой совокупность программных алгоритмов, которые преобразуют исходные данные в выходные результаты. Например, получив массив измерений артериального давления за определенный период, система применяет логические правила, основанные на медицинских классификаторах, для определения категории (норма, прегипертония, гипертония 1 степени и т.д.) и формирует соответствующий текст рекомендации. Аналогично, данные о росте и весе

пользователя преобразуются в Индекс Массы Тела (ИМТ) [25] с последующей классификацией и выдачей совета. Порядок работы пользователя с выходной информацией является интерактивным. Пользователь не просто пассивно получает результат, а может взаимодействовать с ним: изучать графики, масштабируя их по временным отрезкам, отмечать рекомендации как прочитанные и, в случае получения тревожного уведомления, использовать предложенные системой действия, например, функцию записи на прием к врачу.

Выводы по главе 1

В рамках первой главы был проведен комплексный анализ предметной области, который подтвердил низкую оперативность и высокую трудоемкость существующего бизнес-процесса предоставления услуг в сфере здравоохранения на базе МФЦ. Исследование рынка существующих программных продуктов выявило отсутствие комплексного решения, которое бы совмещало функции проактивного мониторинга здоровья с необходимой интеграцией в российскую государственную информационную среду. На основании этих данных была доказана целесообразность разработки новой автоматизированной системы. В результате была сформулирована детальная постановка задачи и разработана концептуальная модель усовершенствованного бизнес-процесса «как должно быть». Реализация предложенного автоматизированного решения в виде мобильного приложения позволит кардинально трансформировать текущую деятельность, обеспечив переход от реактивного оказания административных услуг к современному цифровому сервису по управлению индивидуальным здоровьем граждан.

Глава 2 Логическое проектирование АИС

2.1 Выбор технологии логического моделирования АИС

Для решения задач логического проектирования в настоящей работе была выбрана методология объектно-ориентированного анализа и проектирования, а в качестве основного инструмента – унифицированный язык моделирования UML (Unified Modeling Language) [26]. Данный выбор обусловлен тем, что UML является общепринятым промышленным стандартом для визуализации, специфицирования, конструирования и документирования программных систем. Его использование позволяет создать ясную и однозначную модель, понятную всем участникам проекта, от аналитиков до разработчиков.

Ключевым преимуществом UML является его объектно-ориентированная парадигма, которая идеально согласуется с выбранным для реализации технологическим стеком. Поскольку программный прототип будет разрабатываться на объектно-ориентированном языке Python с использованием фреймворка Django, логическая модель, построенная в терминах классов, объектов и их отношений, сможет быть напрямую и с минимальными искажениями транслирована в программный код [27]. Это обеспечивает целостность и последовательность на всех этапах разработки.

Кроме того, логическая модель в нотации UML выполняет важнейшую функцию связующего звена между общей архитектурой приложения и структурой его базы данных. Процесс разработки проблемно-ориентированной модели данных фактически сводится к построению отображения между объектной моделью системы, представленной в виде диаграммы классов UML, и реляционной моделью данных, которая станет основой для физического проектирования базы данных [28]. Таким образом, UML обеспечивает методологическую основу для плавного и контролируемого перехода от концепции к реализации.

2.2 Логическая модель АИС и ее описание

На основе концептуальной модели и сформулированных требований была разработана логическая модель проектируемой автоматизированной информационной системы. Эта модель, созданная с использованием нотаций унифицированного языка моделирования UML, состоит из нескольких диаграмм, которые комплексно описывают как функциональные, так и структурные аспекты будущего программного продукта.

Для отражения функционального аспекта системы была разработана диаграмма вариантов использования (Use Case Diagram) [29], показанная на Рисунке 5. Она определяет границы системы, основных действующих лиц (акторов) и цели, которые эти акторы могут достигать при взаимодействии с приложением. В системе выделено два ключевых актора: «Обычный пользователь», который является конечным потребителем сервиса, и «Администратор», обладающий расширенными правами для управления системой. Диаграмма наглядно демонстрирует, что Администратор наследует все возможности Обычного пользователя, но дополнительно получает доступ к функциям управления данными всех пользователей.

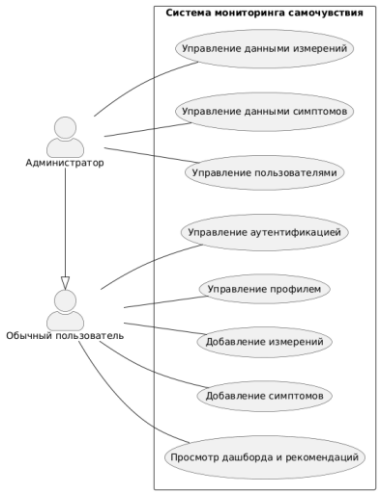

Рисунок 5 – Диаграмма вариантов использования (Use Case)

Как видно из диаграммы, Обычный пользователь решает задачи, связанные с личным мониторингом здоровья: он проходит аутентификацию, управляет своим медицинским профилем, вносит данные об измерениях и симптомах, а также просматривает итоговую аналитику на главном экране. Администратор, в свою очередь, выполняет служебные функции по управлению учетными записями пользователей и их данными, что необходимо для поддержания целостности системы и оказания поддержки [30].

Если диаграмма вариантов использования описывает что система делает, то диаграмма классов раскрывает из чего она состоит. Диаграмма классов (Рисунок 6) представляет собой статическую структуру системы, описывая ключевые сущности предметной области, их атрибуты и взаимосвязи. Она является основой для проектирования базы данных и разработки программных модулей [31]. Информационная модель системы основана на пяти ключевых классах: User, UserProfile, Measurement, Symptomи Recommendation. Исходный код рисунка 5 и некоторых последующих дан в приложении А.

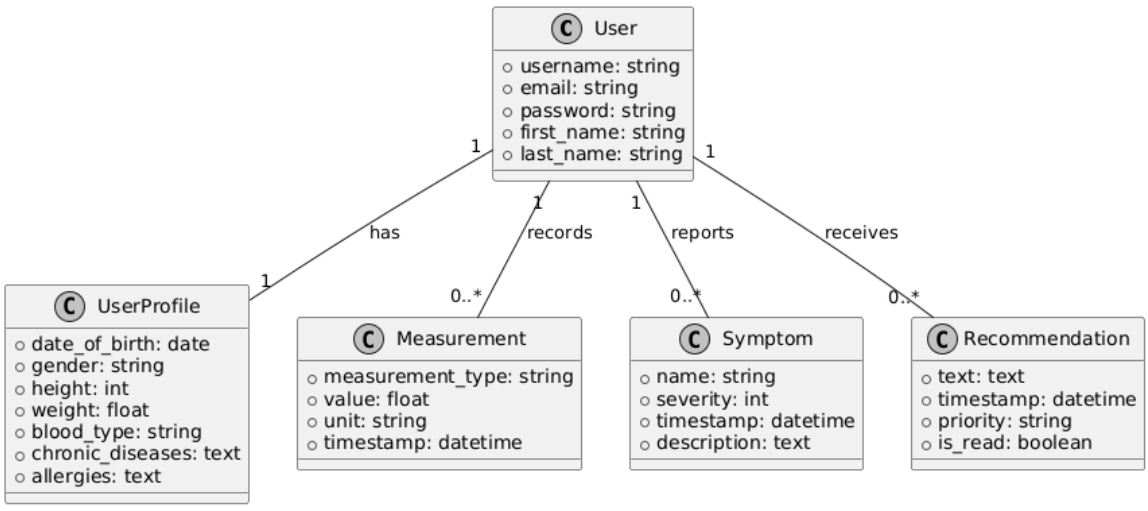

Рисунок 6 – Диаграмма классов (Class Diagram)

Центральной сущностью является класс User, который представляет стандартную модель пользователя системы. С ним по принципу «один к одному» связан класс UserProfile, который расширяет базовую информацию о пользователе специфическими медицинскими данными, такими как дата рождения, рост, вес и хронические заболевания. Класс User также имеет ассоциации типа «один ко многим» с классами Measurement, Symptom и Recommendation. Это означает, что один пользователь может иметь множество записей об измерениях, множество зарегистрированных симптомов и множество полученных от системы рекомендаций. Такая структура позволяет гибко и эффективно хранить и обрабатывать весь объем накапливаемых данных.

Представленные диаграммы вариантов использования и классов являются достаточными для определения основного функционала и структуры АИС. Помимо этого, логическая модель служит основой для спецификации пользовательского интерфейса. Основные экраны приложения – «Дашборд», «Добавить измерение», «Добавить симптом» и «Профиль» – являются логическими представлениями, которые предоставляют пользователю доступ к выполнению вариантов использования, описанных выше, и взаимодействию с объектами, определенными в диаграмме классов.

2.3 Информационное обеспечение АИС

2.3.1 Используемые классификаторы и системы кодирования

Информационное обеспечение представляет собой совокупность всей информации, используемой в системе, включая методы ее классификации, кодирования, а также организацию хранения и обработки. Для проектируемой АИС оно является ядром, определяющим как логику работы аналитических алгоритмов, так и содержание пользовательского интерфейса [32].

Для обеспечения унификации, точности и автоматизированной обработки данных в системе применяется ряд внутренних, локальных классификаторов. Они используются для кодирования однотипных признаков, что позволяет стандартизировать ввод информации, упростить логику программного кода и обеспечить корректность аналитических расчетов. Все классификаторы разработаны специально для данной системы и централизованно ведутся на уровне серверной части приложения. Структура основных используемых кодовых обозначений приведена в Таблице 3.

Таблица 3 – Структура систем кодирования и классификаторов

Наименование кодируемого множества	Значность кода	Система кодирования	Система классификации	Вид классификатора
Пол пользователя	1	Порядковая (символьная)	Отсутствует	Локальный
Тип измерения показателя здоровья	Переменная	Мнемоническая	Иерархическая (одноуровневая)	Локальный
Приоритет рекомендации	Переменная	Мнемоническая	Порядковая	Локальный

Классификатор «Пол пользователя» используетодносимвольный порядковый код ('M' - мужской, 'F' - женский, 'O' - другой) для унификации хранения гендерной информации в профиле пользователя. Классификатор “Тип измерения показателя здоровья” применяет мнемоническую систему кодирования, где строковый идентификатор интуитивно понятен и отражает суть измеряемого параметра (например, 'pulse', 'pressure_sys', 'temperature'). Это упрощает разработку и дальнейшее расширение системы новыми типами измерений. Наконец, классификатор “Приоритет рекомендации” использует мнемонические коды ('low', 'medium', 'high') для ранжирования важности сгенерированных системой советов, что позволяет корректно отображать их в интерфейсе пользователя.

2.3.2 Характеристика нормативно-справочной и входной оперативной информации

Информационная база системы состоит из двух основных типов данных: нормативно-справочной (условно-постоянной) и входной оперативной.

Нормативно-справочная информация (НСИ) представлена данными, которые изменяются редко и служат контекстом для анализа оперативных данных. В проектируемой системе НСИ полностью соответствует сущности «Профиль пользователя» (UserProfile). Эта информация вводится и актуализируется пользователем по мере необходимости через экранную форму «Профиль». Данная форма содержит поля для ввода даты рождения, пола, роста, веса, группы крови, а также текстовые поля для описания хронических заболеваний и аллергий. Введенные данные сохраняются в таблице api_userprofile базы данных. Структура этой таблицы включает поля для каждого из перечисленных атрибутов и внешний ключ для связи с основной таблицей пользователей auth_user [33]. Частота актуализации этой информации низкая и инициируется исключительно пользователем.

Входная оперативная информация – это динамические данные, которые отражают текущее самочувствие пользователя и поступают в систему с высокой периодичностью. Источниками этой информации являются два основных цифровых «документа»: экранная форма «Добавить измерение» и экранная форма «Добавить симптом». Через форму «Добавить измерение» пользователь вводит значения показателей здоровья. Форма содержит выпадающий список для выбора типа измерения (на основе классификатора), поле для ввода числового значения и элемент для выбора даты и времени [34]. Эта информация формирует запись в таблице api_measurement.Через форму «Добавить симптом» пользователь регистрирует субъективные ощущения. Она включает поля для ввода названия симптома, его степени тяжести по шкале от 1 до 10 и текстового описания. Данные сохраняются в таблице api_symptom.

Структура таблиц для хранения оперативной информации (api_measurement и api_symptom) включает уникальный идентификатор записи (первичный ключ), внешний ключ для связи с пользователем, поля для хранения самих данных и временную метку (timestamp). Длительность хранения этой информации не ограничена, так как она формирует исторический архив, необходимый для анализа динамики показателей.

2.3.3 Характеристика выходной информации

Выходная информация в проектируемой АИС носит исключительно цифровой, интерактивный характер и предназначена для оперативного информирования конечного пользователя. Система не формирует печатных ведомостей или регламентированных отчетов. Вместо этого результаты работы аналитических алгоритмов представляются пользователю в реальном времени через основной экран приложения – «Дашборд».

Главным видом выходной информации являются персонализированные рекомендации. Они представляют собой текстовые сообщения, которые система генерирует автоматически на основе анализа входной оперативной и нормативно-справочной информации. Например, при фиксации значений артериального давления, превышающих норму, система формирует рекомендацию о необходимости контроля и консультации с врачом. Эти данные представляют собой результат работы системы и сохраняются в отдельной таблице api_recommendation [35]. Структура этой таблицы включает уникальный идентификатор, текст рекомендации, ее приоритет, временную метку создания и флаг, указывающий, была ли она прочитана пользователем.

Другим важным видом выходной информации является визуализация данных. На дашборде отображаются интерактивные графики, построенные на основе исторических данных из таблицы api_measurement. Эти графики (например, «График динамики артериального давления») позволяют пользователю наглядно оценить изменения своего состояния во времени, выявить тренды и паттерны. Они служат инструментом для оперативного

самоконтроля и не предназначены для официальной отчетности [36]. Очевидно, вся выходная информация направлена на решение главной задачи системы – предоставление пользователю удобного и понятного инструмента для управления собственным здоровьем.

2.4 Проектирование базы данных АИС

2.4.1 Выбор технологии проектирования БД АИС

Проектирование базы данных является завершающим этапом логического моделирования, в ходе которого абстрактные сущности и их отношения, определенные ранее, преобразуются в конкретную, структурированную модель данных. Качество проектирования базы данных напрямую влияет на производительность, надежность и масштабируемость всей информационной системы.

Для проектирования реляционной базы данных проектируемой АИС была выбрана методология IDEF1X. Этот выбор обусловлен тем, что IDEF1X [37] является строгим и формализованным стандартом, специально разработанным для создания информационных моделей, которые точно отражают структуру данных предметной области. Использование этой методологии позволяет обеспечить высокий уровень нормализации данных, минимизировать их избыточность и гарантировать целостность за счет четкого определения сущностей, атрибутов и связей между ними [38]. Процесс проектирования в данном случае упрощается, поскольку он представляет собой процедуру трансформации объектной модели, разработанной на предыдущем этапе в виде диаграммы классов UML, в логическую модель данных в нотации, близкой к IDEF1X.

2.4.2 Разработка концептуальной модели данных АИС

Концептуальная модель данных является высокоуровневым представлением, которое идентифицирует ключевые сущности предметной области без углубления в детали атрибутов или типов данных. На основе

анализа требований и логической модели системы были выделены следующие основные концептуальные сущности, подлежащие хранению в базе данных: Пользователь, его Медицинский профиль, Измерение показателей здоровья, зафиксированный Симптом и сгенерированная системой Рекомендация. Эти сущности полностью охватывают информационные потребности системы, связанные с мониторингом самочувствия.

2.4.3 Обоснование вида логической модели

В качестве вида логической модели была выбрана реляционная модель данных. Это решение является наиболее обоснованным и целесообразным по нескольким причинам. Во-первых, весь технологический стек проекта (Django, SQLite) изначально ориентирован на работу с реляционными базами данных. Django ORM (Object-RelationalMapping) [39] эффективно абстрагирует взаимодействие с реляционными СУБД, позволяя работать с таблицами как с объектами Python. Во-вторых, реляционная модель обеспечивает строгую целостность данных за счет использования механизмов первичных и внешних ключей, что критически важно для хранения медицинской информации. Структура данных предметной области, с ее четкими связями «один к одному» и «один ко многим», идеально отображается на реляционную структуру таблиц, что обеспечивает эффективность хранения и выполнения запросов.

2.4.4 Разработка логической модели данных АИС

На основе диаграммы классов и выбранной реляционной модели была разработана логическая модель данных, представленная в виде ER-диаграммы (Entity-RelationshipDiagram) на Рисунке 7. Диаграмма включает как таблицы, спроектированные специально для данной АИС (с префиксом api_) [40], так и стандартные таблицы фреймворка Django, необходимые для обеспечения базовой функциональности, такой как аутентификация пользователей.

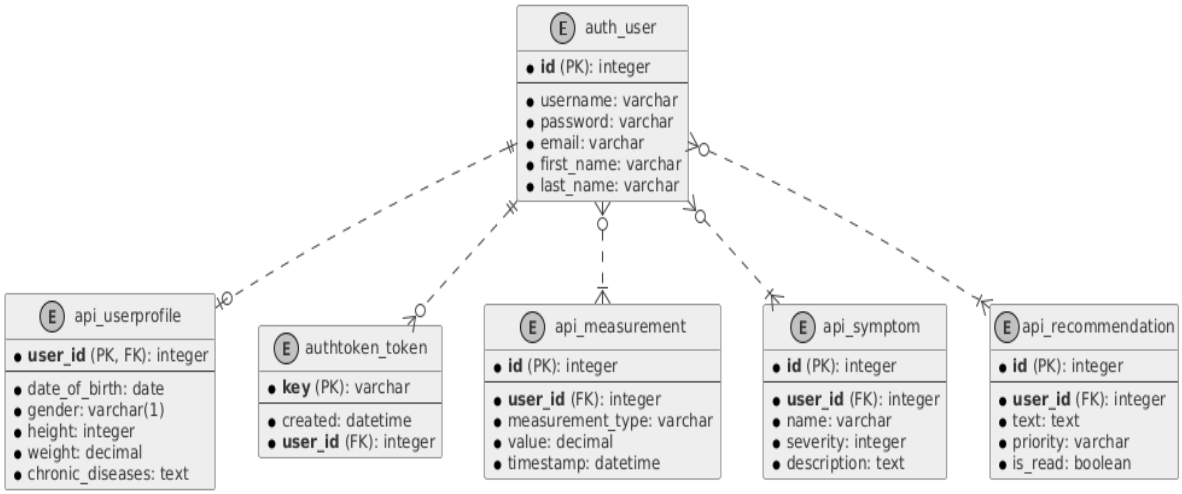

Рисунок 7 – ER-диаграмма логической модели базы данных

Данная логическая модель отражает все информационные сущности и их взаимосвязи. Центральной сущностью является таблица auth_user, стандартный компонент Django, который хранит основные учетные данные пользователей, включая логин и хешированный пароль. Эта таблица является родительской для всех остальных сущностей, связанных с пользователем. Таблица api_userprofile расширяет auth_user, храня условно-постоянную медицинскую информацию. Связь «один к одному» между ними, реализованная через первичный-внешний ключ user_id, гарантирует, что у каждого пользователя может быть только один медицинский профиль.

Сущности api_measurement, api_symptom и api_recommendation предназначены для хранения оперативной, накапливаемой информации. Они связаны с auth_user отношением «один ко многим» через внешний ключ user_id. Это корректно отражает реальность, в которой один пользователь может иметь множество записей об измерениях, симптомах и полученных рекомендациях за весь период использования приложения. Такая структура обеспечивает эффективное хранение временных рядов данных. Наконец, таблица authtoken_token является служебной и используется для реализации механизма токен-аутентификации. Она хранит уникальные ключи доступа и

связана с auth_user отношением «один к одному», обеспечивая безопасное взаимодействие между клиентской и серверной частями АИС.

2.5 Требования к аппаратно-программному обеспечению АИС

Разработанная автоматизированная информационная система, будучи веб-приложением с трехзвенной архитектурой, предъявляет определенные требования к аппаратно-программной среде, в которой она будет развернута для эксплуатации. Оценка существующей ИТ-инфраструктуры МФЦ, на базе которого реализуется проект, показывает, что она ориентирована преимущественно на обслуживание внутренних административных задач и очного приема граждан, и может потребовать модернизации для обеспечения надежной работы публичного, высоконагруженного сервиса.

Для серверной части приложения, включающей веб-сервер, сервер приложений (Django) и сервер базы данных, необходима выделенная серверная среда [41]. На начальном этапе пилотного внедрения может быть достаточно одного физического или виртуального сервера со следующими минимальными характеристиками: 4-ядерный процессор, 8 ГБ оперативной памяти и твердотельный накопитель (SSD) объемом не менее 100 ГБ для размещения операционной системы, программного обеспечения и базы данных. В качестве серверной операционной системы рекомендуется использовать дистрибутивы Linux (например, Ubuntu Server или CentOS) ввиду их стабильности, безопасности и отсутствия лицензионных отчислений. Программное обеспечение должно включать веб-сервер (Nginx или Apache), сервер приложений (Gunicorn или uWSGI), интерпретатор Python версии 3.9 или выше, а также СУБД (PostgreSQL рекомендуется для промышленной эксплуатации вместо SQLite) [42][43].

Для клиентской части, с которой взаимодействует конечный пользователь, требования минимальны и сводятся к наличию современного мобильного устройства (смартфона или планшета) под управлением

актуальных версий операционных систем iOS или Android. Доступ к приложению осуществляется через стандартный веб-браузер, что снимает необходимость в установке нативного приложения на начальном этапе [44]. Однако для полноценного пользовательского опыта и интеграции с датчиками в будущем потребуется разработка мобильных клиентов.

С учетом потенциального роста числа пользователей и объема накапливаемых данных, существующую ИТ-инфраструктуру МФЦ необходимо будет масштабировать. Наиболее эффективным способом улучшения производительности является переход на облачный хостинг. Размещение серверной части в облаке (например, у российских провайдеров Yandex Cloud или VK Cloud) [45] позволит динамически управлять ресурсами: увеличивать вычислительную мощность и объем хранилища по мере роста нагрузки. Кроме того, облачные платформы предоставляют готовые решения для горизонтального масштабирования, балансировки нагрузки и резервного копирования, что значительно повышает отказоустойчивость и надежность системы. Такой подход является более гибким и экономически целесообразным по сравнению с закупкой и поддержкой собственного физического серверного оборудования.

Выводы по главе 2

В ходе выполнения второй главы был осуществлен переход от концептуального видения к формализованному логическому проекту будущей автоматизированной информационной системы. Была обоснована и выбрана методология объектно-ориентированного анализа и проектирования на базе унифицированного языка моделирования UML как наиболее соответствующая задачам и технологическому стеку проекта. Разработанная логическая модель, включающая диаграммы вариантов использования и классов, позволила четко определить функциональные границы системы, ее структурные компоненты и взаимосвязи между ними.

На основе логической модели было спроектировано информационное обеспечение АИС, в рамках которого были определены и описаны используемые классификаторы, а также охарактеризованы потоки нормативно-справочной, входной оперативной и выходной информации. Завершающим этапом главы стала разработка детальной логической модели реляционной базы данных, представленной в виде ER-диаграммы. Эта модель формализует структуру хранения данных и служит непосредственной основой для физической реализации базы данных на следующем этапе. Таким образом, в результате проделанной работы были созданы все необходимые проектные артефакты, которые в совокупности представляют собой исчерпывающую и непротиворечивую спецификацию для дальнейшей разработки и реализации системы.

Глава 3 Физическое проектирование АИС

3.1 Выбор архитектуры АИС

Выбор архитектуры является одним из наиболее ответственных решений на этапе физического проектирования, так как он определяет общую структуру приложения, способ взаимодействия его компонентов и закладывает основу для будущего развития. Для проектируемой системы, которая должна обеспечивать интерактивное взаимодействие с множеством пользователей через веб-интерфейс, были рассмотрены различные архитектурные подходы, включая файл-серверную, двухзвенную и трехзвенную модели «клиент-сервер».

Анализ показал, что файл-серверная и двухзвенная архитектуры не удовлетворяют требованиям проекта. Файл-серверная модель непригодна из-за отсутствия централизованной логики и проблем с безопасностью. Двухзвенная модель, где вся бизнес-логика находится на клиенте или на сервере БД, также не является оптимальной, поскольку она ограничивает гибкость, усложняет масштабирование и смешивает различные уровни ответственности в приложении [46].

В результате сравнительного анализа для реализации АИС была выбрана трехзвенная архитектура «клиент-сервер». Эта модель является современным промышленным стандартом для разработки веб-приложений и идеально соответствует задачам проекта. Она предполагает четкое разделение системы на три логических и физических уровня:

- уровень представления (клиент): Отвечает за пользовательский интерфейс, визуализацию данных и взаимодействие с пользователем [47]. В данном проекте этот уровень реализован в виде веб-приложения на базе фреймворка Streamlit;
- уровень бизнес-логики (сервер приложений): Содержит всю основную логику системы. Здесь выполняются обработка запросов от клиента,

аутентификация, авторизация, аналитические расчеты и формирование ответов [48]. Этот уровень реализован на Python с использованием веб-фреймворка Django;

- уровень данных (сервер баз данных): Отвечает за хранение, извлечение и управление данными [49]. Этот уровень представлен реляционной базой данных под управлением СУБД SQLite (на этапе прототипирования).

Основное преимущество трехзвенной архитектуры заключается в слабой связанности ее компонентов. Уровни взаимодействуют друг с другом через стандартизированные интерфейсы (в данном случае – через REST API [50]), что позволяет разрабатывать, модифицировать и масштабировать каждый из них независимо от остальных. Это значительно повышает гибкость и упрощает дальнейшее развитие системы.

Для визуализации структуры программного обеспечения была разработана UML-диаграмма пакетов (Рисунок 8), которая отражает основные модули системы и зависимости между ними.

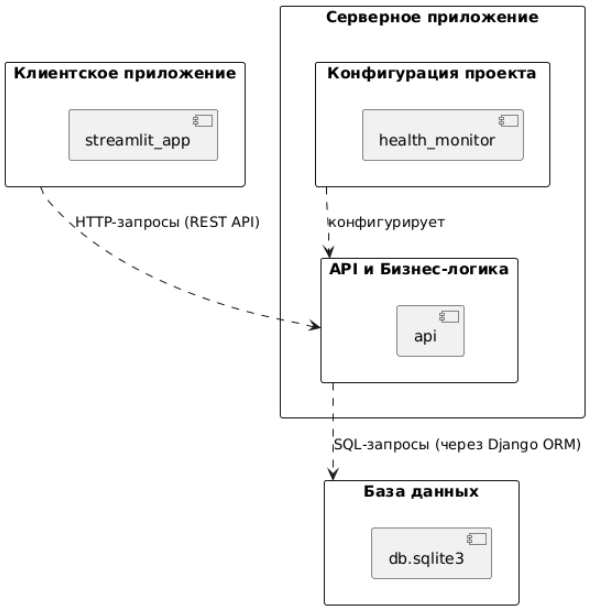

Рисунок 8 – UML-диаграмма пакетов АИС

На этой диаграмме видно, что клиентский пакет streamlit_app зависит только от пакета api, обращаясь к нему через четко определенный интерфейс. В свою очередь, пакет api, содержащий основную бизнес-логику, зависит от базы данных db.sqlite3, но это взаимодействие инкапсулировано внутри серверного приложения.

Для более детального представления развертывания системы была построена UML-диаграмма размещения компонентов (Рисунок 9). Она показывает, как программные артефакты распределяются по физическим или виртуальным узлам.

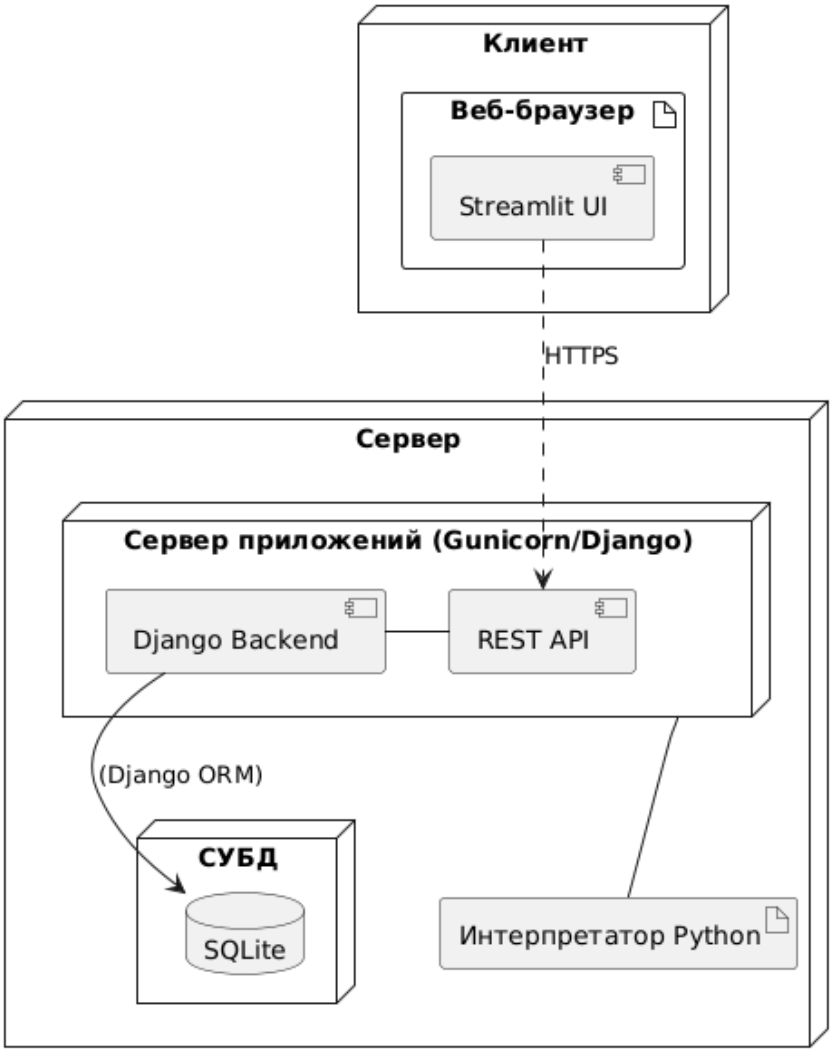

Рисунок 9 – UML-диаграмма размещения компонентов АИС

Диаграмма размещения наглядно демонстрирует, что серверная часть, включающая интерпретатор Python, сервер приложений Django и СУБД SQLite, разворачивается на едином серверном узле. Клиентская часть, представленная интерфейсом Streamlit, исполняется в веб-браузере на устройстве пользователя. Взаимодействие между клиентом и сервером осуществляется по сетевому протоколу HTTPS, что обеспечивает шифрование передаваемых данных. Выбранная архитектура является надежной, безопасной и масштабируемой основой для создания и развития проектируемой информационной системы.

3.2 Выбор технологии разработки программного обеспечения АИС

Выбор технологического стека для разработки программного обеспечения является стратегическим решением, которое оказывает прямое влияние на скорость реализации проекта, его итоговую стоимость, а также на производительность и возможности дальнейшего развития системы. При выборе рассматривались два основных подхода: использование готовых универсальных платформ или RAD-систем, и разработка с нуля с использованием бесплатно распространяемых средств разработки с открытым исходным кодом. Учитывая уникальность требований проекта, сочетающего медицинский мониторинг с интеграцией в государственную ИТ-инфраструктуру, было принято решение в пользу второго варианта, так как он обеспечивает максимальную гибкость и полный контроль над функциональностью.

Для принятия обоснованного решения был проведен сравнительный анализ нескольких популярных технологических стеков, которые потенциально подходят для создания веб-приложений с интенсивной обработкой данных. В качестве основных критериев оценки выступали: скорость разработки и прототипирования, наличие и зрелость библиотек для

анализа данных, масштабируемость, а также размер сообщества и доступность документации.

В качестве альтернатив рассматривались такие промышленные стандарты, как Java со фреймворком Spring и JavaScript-стек на базе Node.js (например, с фреймворком Express) и клиентской библиотекой React. Java и Spring являются мощным и надежным решением для создания высоконагруженных корпоративных систем, однако характеризуются более высоким порогом вхождения и большей многословностью кода, что могло бы замедлить процесс прототипирования. Стек на базе Node.js и React, в свою очередь, является лидером в создании современных, быстрых и интерактивных пользовательских интерфейсов, но его экосистема для сложного научного и математического анализа данных все еще уступает более зрелым альтернативам.

В результате сравнительного анализа был сделан выбор в пользу стека, основанного на языке программирования Python и веб-фреймворке Django. Ключевым фактором, определившим этот выбор, стали непревзойденные возможности Python в области анализа данных. Наличие таких фундаментальных библиотек, как Pandas и NumPy, делает Python отраслевым стандартом для любых задач, связанных с обработкой, анализом и манипуляцией данными. Это позволило реализовать аналитический модуль системы максимально эффективно и с использованием проверенных инструментов. Фреймворк Django был выбран как серверная основа благодаря его философии «батарейки в комплекте» (batteries-included). Он предоставляет из коробки мощную ORM для работы с базой данных, надежную систему аутентификации, готовую административную панель и множество других компонентов, что значительно ускоряет разработку и позволяет сосредоточиться на уникальной бизнес-логике приложения, а не на создании базовой инфраструктуры. Для быстрой разработки прототипа пользовательского интерфейса был выбран фреймворк Streamlit, который идеально подходит для создания интерактивных веб-приложений,

ориентированных на демонстрацию данных, и не требует глубоких знаний в области фронтенд-разработки.

Результаты сравнительного анализа основных рассматриваемых технологий сведены в Таблицу 4.

Таблица 4 – Сравнительный анализ технологий разработки

Критерий	Python + Django/Streamlit	Java + Spring	Node.js + React
Скорость разработки и прототипирования	Очень высокая	Средняя	Высокая
Возможности анализа данных	Отличные	Удовлетворительные	Удовлетворительные
Экосистема и зрелость	Обширная, зрелая	Очень обширная, зрелая	Обширная, быстроразвивающаяся
Масштабируемость	Высокая	Очень высокая	Высокая
Стоимость владения (лицензии)	Низкая (полностью Open Source)	Низкая (Open Source)	Низкая (Open Source)

Как видно из таблицы, выбранный технологический стек является наиболее сбалансированным решением именно для задач данного проекта. Он обеспечивает максимальную скорость прототипирования и предоставляет лучшие в своем классе инструменты для реализации ключевой функции системы – аналитики данных, при этом сохраняя высокий потенциал для дальнейшего масштабирования и промышленной эксплуатации.

3.3 Выбор СУБД АИС

Выбор системы управления базами данных (СУБД) является фундаментальным решением на этапе физического проектирования, так как СУБД отвечает за надежное хранение, целостность и быстрый доступ ко всей информации, накапливаемой в системе. При выборе СУБД для данного проекта основными критериями выступали полная совместимость с выбранным стеком разработки (Python и Django), отсутствие лицензионных

отчислений, простота развертывания на этапе прототипирования, а также наличие четкого пути для дальнейшего масштабирования системы при переходе к промышленной эксплуатации.

В качестве основных кандидатов были рассмотрены три ведущие бесплатно распространяемые реляционные СУБД, имеющие отличную поддержку со стороны фреймворка Django: PostgreSQL, MySQL и SQLite. PostgreSQL является мощной, объектно-реляционной СУБД, которая часто считается стандартом для сложных и высоконагруженных Django-проектов благодаря своей надежности и расширяемости. MySQL (и ее форкMariaDB) – это еще одна чрезвычайно популярная и проверенная временем СУБД, широко используемая в веб-разработке. SQLite представляет собой встраиваемую, бессерверную СУБД, которая хранит всю базу данных в одном файле и не требует отдельного процесса для работы. Для принятия взвешенного решения был проведен их сравнительный анализ по ключевым для проекта характеристикам, результаты которого представлены в Таблице 5.

Таблица 5 – Сравнительный анализ систем управления базами данных

Критерий	SQLite	PostgreSQL	MySQL / MariaDB
Простота установки и настройки	Очень высокая	Средняя	Средняя
Требования к ресурсам	Очень низкие	Высокие	Высокие
Производительность (при высокой нагрузке)	Низкая	Очень высокая	Высокая
Масштабируемость	Низкая	Очень высокая	Высокая
Совместимость с Django	Отличная (встроенная)	Отличная	Отличная

Анализ показывает, что для высоконагруженных промышленных систем PostgreSQL и MySQL являются очевидными фаворитами благодаря своей производительности и масштабируемости. Однако для текущего этапа проекта, который заключается в разработке и тестировании функционального

прототипа, эти преимущества не являются критичными. Напротив, их недостатки, такие как необходимость установки и настройки отдельного серверного процесса и более высокие требования к системным ресурсам, могут неоправданно замедлить и усложнить процесс разработки и развертывания.

Как следствие, SQLite представляет собой наиболее рациональный и прагматичный выбор. Ее главное преимущество – исключительная простота. Поскольку SQLite является встраиваемой библиотекой и не требует отдельного сервера, она идеально подходит для быстрой разработки и прототипирования. Вся база данных содержится в одном файле, что позволяет легко переносить проект, делиться им с другими разработчиками и быстро настраивать рабочее окружение. Для задач прототипа, где нагрузка на систему невелика, а количество одновременных пользователей ограничено, производительности SQLite более чем достаточно. Стоит также отметить, что выбор SQLite на начальном этапе не создает технологического тупика. Благодаря использованию в проекте Django ORM, которая абстрагирует работу с базой данных, переход на более мощную СУБД, такую как PostgreSQL, при необходимости масштабирования системы в будущем, является стандартной и хорошо документированной процедурой, которая потребует лишь минимальных изменений в конфигурационных файлах проекта. Таким образом, для этапа физического проектирования и реализации прототипа АИС, SQLite предлагает оптимальное сочетание простоты, скорости разработки и полной совместимости с выбранным технологическим стеком.

3.4 Разработка физической модели данных АИС

Физическая модель данных является финальным этапом проектирования базы данных, на котором логическая модель, разработанная ранее, транслируется в конкретную структуру, предназначенную для

реализации средствами выбранной системы управления базами данных. Этот этап определяет точные имена таблиц и столбцов, типы данных, индексы, первичные и внешние ключи, а также другие атрибуты, которые напрямую влияют на производительность и физическое хранение информации. В отличие от логической модели, которая является абстрактной, физическая модель полностью зависит от особенностей и синтаксиса целевой СУБД, в данном случае – SQLite.

Разработка физической модели в рамках выбранного технологического стека (Python и Django) имеет свою специфику. Она не выполняется путем написания прямых SQL-запросов CREATE TABLE. Вместо этого, модель данных описывается в виде классов на языке Python в файле models.py с использованием Django ORM (Object-RelationalMapping). Затем Django автоматически генерирует и выполняет необходимые SQL-команды для создания соответствующей схемы в базе данных. Этот подход обеспечивает высокий уровень абстракции, переносимость между различными СУБД и целостность между кодом приложения и структурой базы данных.

Физическая модель данных для проектируемой АИС представлена на Рисунке 10. Она включает таблицы, которые напрямую соответствуют Python-классам из api/models.py, а также служебные таблицы, создаваемые фреймворком Django для управления пользователями и аутентификацией.

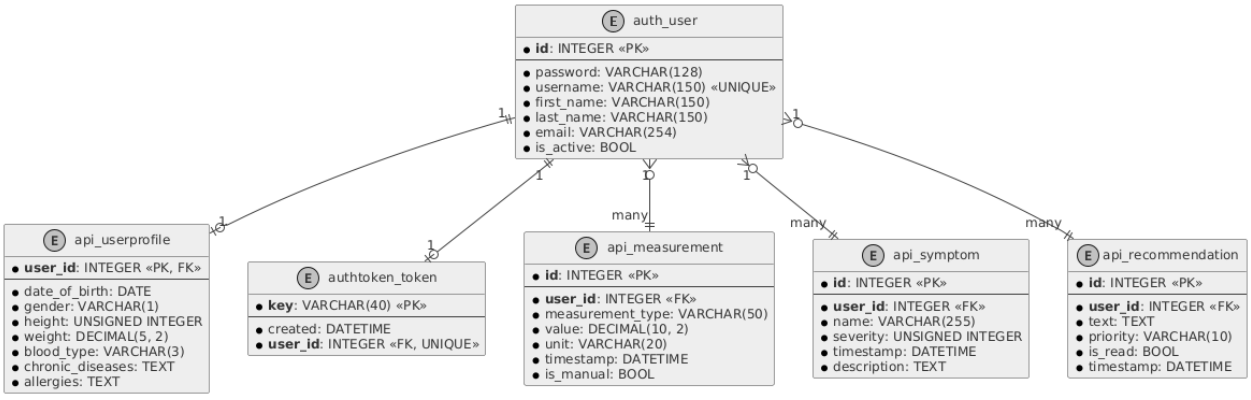

Рисунок 10 – Физическая модель данных (ER-диаграмма)

Характерной особенностью СУБД SQLite является динамическая типизация. В отличие от большинства SQL-СУБД, которые используют статическую типизацию, SQLite применяет так называемое “сродство типов” (typeaffinity). Это означает, что тип данных связан со значением, а не со столбцом. Однако при работе через Django ORM, фреймворк обеспечивает строгую валидацию данных на уровне приложения, гарантируя, что в столбцы, определенные, например, как DateField или DecimalField, будут записываться данные соответствующего формата.

В физической модели отсутствуют хранимые процедуры, а логика, которая в традиционных системах могла бы быть реализована с помощью триггеров, перенесена на уровень приложения. Ярким примером является автоматическое создание записи в таблице api_userprofile при регистрации нового пользователя. Эта функциональность реализована не средствами SQL, а с помощью механизма сигналов Django. Сигнал post_save отлавливает событие сохранения нового объекта User и программно создает связанный с ним объект UserProfile. Такой подход повышает переносимость кода между разными СУБД и сохраняет всю бизнес-логику в рамках Python-кода.

Взаимодействие с базой данных также осуществляется не через прямые SQL-запросы, а через методы Django ORM. Например, для получения последних пяти измерений пульса для текущего пользователя, разработчик пишет следующий код на Python:

```
# Пример кода на Python с использованием Django ORM
from .modelsimportMeasurement
defget_latest_pulse_measurements(user):
returnMeasurement.objects.filter(
user=user,
measurement_type='pulse'
).order_by('-timestamp')[:5]
```

На основе этого кода Django ORM сгенерирует и выполнит SQL-запрос, который для СУБД SQLite будет выглядеть приблизительно так:

```
-- SQL-запрос, сгенерированный Django ORM
SELECT “api_measurement”.”id”,
“api_measurement”.”user_id”,
“api_measurement”.”measurement_type”,
“api_measurement”.”value”,
“api_measurement”.”unit”,
“api_measurement”.”timestamp”,
“api_measurement”.”is_manual”
FROM “api_measurement”
WHERE “api_measurement”.”user_id” = 123 -- ID текущегопользователя
  AND “api_measurement”.”measurement_type” = 'pulse'
ORDER BY “api_measurement”.”timestamp” DESC
LIMIT 5;
```

Для оптимизации запросов в Django активно используются методы select_related и prefetch_related. Эти инструменты позволяют решить распространенную проблему “N+1 запроса”, когда для извлечения связанных данных выполняется один главный запрос и затем N дополнительных запросов. select_related работает для связей “один к одному” и “многие к одному” и использует SQL JOIN для получения связанных объектов в одном запросе. prefetch_related работает для всех типов связей и выполняет отдельный запрос для связанных объектов, а затем “соединяет” их в Python. Кроме того, для ускорения выборок Django автоматически создает индексы на все первичные ключи, а также на внешние ключи, что является базовой, но крайне эффективной мерой оптимизации для реляционных баз данных.

3.5 Разработка программного обеспечения АИС

3.5.1 Схема взаимосвязи модулей приложения АИС

Разработка программного обеспечения является центральным этапом физического проектирования, на котором создаются все программные

компоненты, реализующие бизнес-логику, пользовательский интерфейс и взаимодействие с базой данных в соответствии с ранее разработанными моделями. Программный продукт спроектирован как модульная система, что обеспечивает гибкость разработки, простоту тестирования и возможности для дальнейшего расширения функциональности.

Программное обеспечение АИС построено в соответствии с выбранной трехзвенной архитектурой и состоит из нескольких логически обособленных модулей, каждый из которых выполняет свою специфическую функцию. Взаимодействие между этими модулями осуществляется через четко определенные программные интерфейсы, что обеспечивает их слабую связанность.

На верхнем уровне можно выделить три основных компонента: клиентское приложение, серверное приложение и база данных. Клиентское приложение (streamlit_app), отвечающее за пользовательский интерфейс, не имеет прямого доступа к базе данных. Вместо этого оно взаимодействует исключительно с серверным приложением через программный интерфейс REST API. Серверное приложение (api), в свою очередь, инкапсулирует всю бизнес-логику и является единственным компонентом, который имеет право обращаться к базе данных (db.sqlite3) для чтения и записи информации. Эта схема обеспечивает высокий уровень безопасности и целостности данных.

Общая схема взаимосвязи компонентов на уровне программных модулей представлена на Рисунке 11.

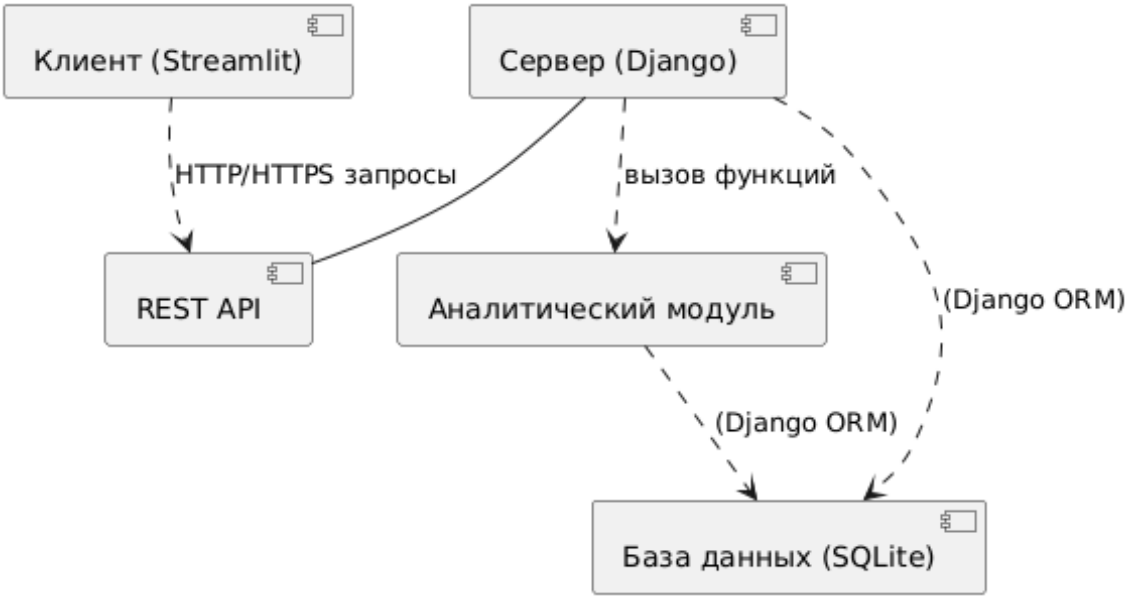

Рисунок 11 – UML-диаграмма компонентов программного обеспечения

Как видно из диаграммы, структура строго иерархична. Клиент взаимодействует с системой через REST API. Сервер Django предоставляет этот API, обрабатывает входящие запросы, при необходимости вызывает функции из Аналитического модуля и взаимодействует с Базой данных через слой абстракции Django ORM. Такая модульная структура является залогом надежности и масштабируемости всего программного комплекса.

3.5.2 Описание модулей приложения АИС с примерами программного кода

Программное обеспечение системы состоит из нескольких ключевых модулей, которые можно разделить на две группы: модули, реализующие основные функции системы (работа с данными), и модули, выполняющие служебные функции (аутентификация, управление доступом).

Модуль служебных функций: Аутентификация и Авторизация отвечает за базовые, но критически важные функции безопасности: регистрацию новых пользователей, проверку их учетных данных при входе (аутентификацию) и управление доступом к ресурсам системы. Он реализован в файле api/views.py с использованием встроенных механизмов

Django и библиотеки Django REST Framework. Ключевыми компонентами являются классы UserRegistrationView для создания новых учетных записей и CustomAuthToken для генерации уникального токена доступа после успешного входа.

Пример реализации класса для регистрации пользователя:

```
# Фрагмент кода из api/views.py
classUserRegistrationView(generics.CreateAPIView):
queryset = User.objects.all()
serializer_class = UserRegistrationSerializer
permission_classes = [permissions.AllowAny]  # Разрешить регистрацию всем
defcreate(self, request, *args, **kwargs):
serializer = self.get_serializer(data=request.data)
serializer.is_valid(raise_exception=True)
user = serializer.save()
token, created = Token.objects.get_or_create(user=user)
      # Возвращаем данные пользователя и токен
user_data = UserSerializer(user, context=self.get_serializer_context()).data
returnResponse({
          'user': user_data,
          'token': token.key
      }, status=status.HTTP_201_CREATED)
```

Данный код принимает от клиента HTTP POST-запрос с данными нового пользователя, проверяет их корректность (валидирует), сохраняет нового пользователя в базу данных и в ответ отправляет как данные созданного пользователя, так и уникальный токен авторизации, который клиент будет использовать для всех последующих запросов.

Модуль управления данными (CRUD-операции). Этот модуль является ядром серверной части и отвечает за все основные операции с данными: создание, чтение, обновление и удаление (CRUD). Он также реализован в

api/views.py в виде наборов представлений (ViewSets), таких как MeasurementViewSet, SymptomViewSet и UserProfileViewSet. Каждый ViewSet предоставляет полный набор конечных точек REST API для управления соответствующей сущностью. Важной особенностью является то, что при создании новой записи об измерении (Measurement), система автоматически инициирует вызов аналитического модуля для пересчета рекомендаций.

Пример реализации MeasurementViewSet, демонстрирующий связь с аналитическим модулем:

```
# Фрагмент кода из api/views.py
classMeasurementViewSet(viewsets.ModelViewSet):
serializer_class = MeasurementSerializer
permission_classes = [permissions.IsAuthenticated, IsOwnerOrReadOnly]
defget_queryset(self):
    # Пользователь видит только свои измерения
returnMeasurement.objects.filter(user=self.request.user)
defperform_create(self, serializer):
    # Автоматически подставляем текущего пользователя
measurement = serializer.save(user=self.request.user)
    # Запускаем анализ после добавления нового измерения
analyze_user_data(self.request.user) # <--- Вызов аналитического модуля
```

Модуль аналитики данных – наиболее наукоемкий модуль системы, реализующий ее интеллектуальную составляющую. Он расположен в файле api/services.py и представлен функцией analyze_user_data. Данный модуль не имеет прямого внешнего интерфейса и вызывается другими частями системы (например, модулем управления данными). Его задача – получить из базы данных актуальные и исторические данные пользователя, применить к ним набор эвристических правил и алгоритмов, основанных на медицинских нормах, и, в случае выявления отклонений, создать в базе данных новые

персонализированные рекомендации. Алгоритм работы этого модуля представлен в виде блок-схемы на Рисунке 12.

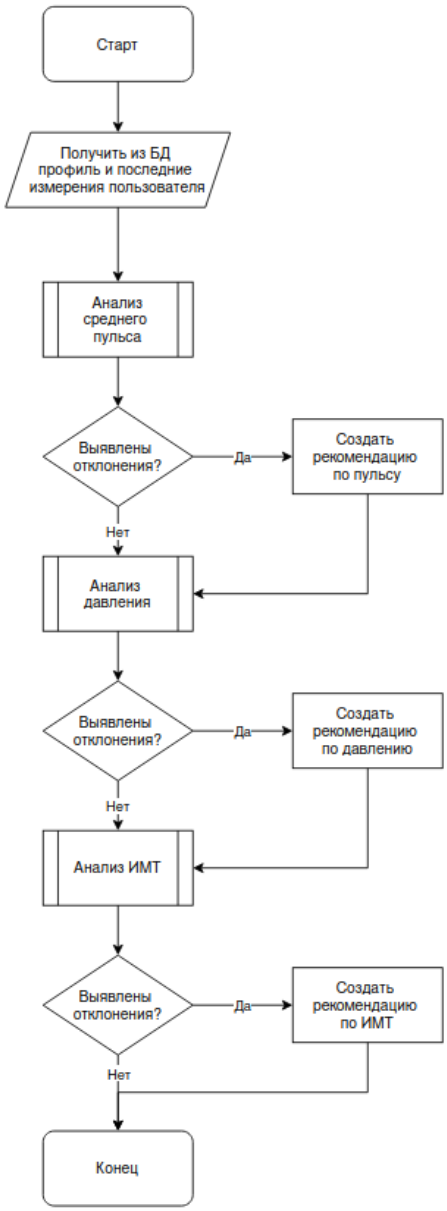

Рисунок 12 – Блок-схема алгоритма работы аналитического модуля (analyze_user_data)

Модуль клиентского приложения отвечает за весь пользовательский интерфейс и организацию диалога с пользователем. Он реализован в виде пакета streamlit_app. Структура диалога организована с помощью бокового меню, которое является основным элементом навигации. В зависимости от

выбора пользователя в этом меню, главный скрипт app.py динамически отображает один из нескольких «экранов»: «Дашборд», «Добавить измерение», «Добавить симптом» или «Профиль». Для получения и отправки данных на сервер, клиентское приложение использует вспомогательный модуль utils.py, который инкапсулирует всю логику HTTP-запросов к REST API.

Пример реализации экрана «Добавить симптом»:

```
# Фрагмент кода из streamlit_app/app.py
elifselection == "😟 Добавить симптом":
st.title("😟 Добавить новый симптом")
with st.form("symptom_form"):
symptom_name = st.text_input("Название симптома (например, Головная боль)")
symptom_severity = st.slider("Степеньтяжести:", min_value=1, max_value=10, value=5)
symptom_description = st.text_area("Описание (необязательно)")
symptom_dt = datetime.combine(st.date_input("Дата:"), st.time_input("Время:"))
symptom_submitted = st.form_submit_button("Добавитьсимптом")
ifsymptom_submitted:
ifsymptom_name:
data = {
"name": symptom_name,
"severity": symptom_severity,
"description": symptom_description,
"timestamp": symptom_dt.isoformat()
}
        # Вызов функции из utils.py для отправки данных на сервер
ifadd_symptom(data):
st.success("Симптом успешно добавлен!")
```

else:

st.error(“Не удалось добавить симптом.”)

else:

st.warning(“Пожалуйста, введите название симптома.”)

Этот код демонстрирует, как организуется диалог: пользователю предоставляется форма для ввода данных, а после нажатия кнопки «Добавить симптом» эти данные упаковываются в словарь и передаются функции add_symptom из модуля utils, которая и выполняет отправку на сервер.

3.6 Описание функциональности АИС

Разработанная автоматизированная информационная система представляет собой комплексный программный продукт, предназначенный для предоставления пользователям удобного и интуитивно понятного инструмента для ведения личного дневника здоровья, отслеживания динамики ключевых показателей и получения персонализированных рекомендаций. Данный раздел содержит краткую инструкцию по эксплуатации АИС и описывает технологический процесс работы с информацией в системе.

Основной сценарий взаимодействия пользователя с системой начинается с процедуры регистрации или аутентификации. Для новых пользователей предусмотрена простая форма регистрации, где необходимо указать имя пользователя, email и пароль. После успешного создания учетной записи или входа в уже существующую, пользователь получает доступ к полному функционалу приложения. Навигация в системе осуществляется через боковое меню, которое является основным элементом управления и обеспечивает быстрый доступ ко всем ключевым разделам. Технологический процесс сбора, передачи, обработки и выдачи информации в АИС представляет собой непрерывный и полностью автоматизированный цикл. Он наглядно представлен на UML-диаграмме деятельности на Рисунке 13.

Рисунок 13 – UML-диаграмма деятельности технологического процесса

Как видно из схемы, процесс инициируется пользователем. Сбор первичной информации осуществляется через два основных экрана:

«Добавить измерение» и «Добавить симптом». На экране добавления измерения, пример которого показан на Рисунке 14, пользователь выбирает тип показателя из выпадающего списка, вводит его значение и указывает дату и время. Для удобства, при выборе артериального давления форма динамически изменяется, предлагая поля для ввода систолического и диастолического значений.

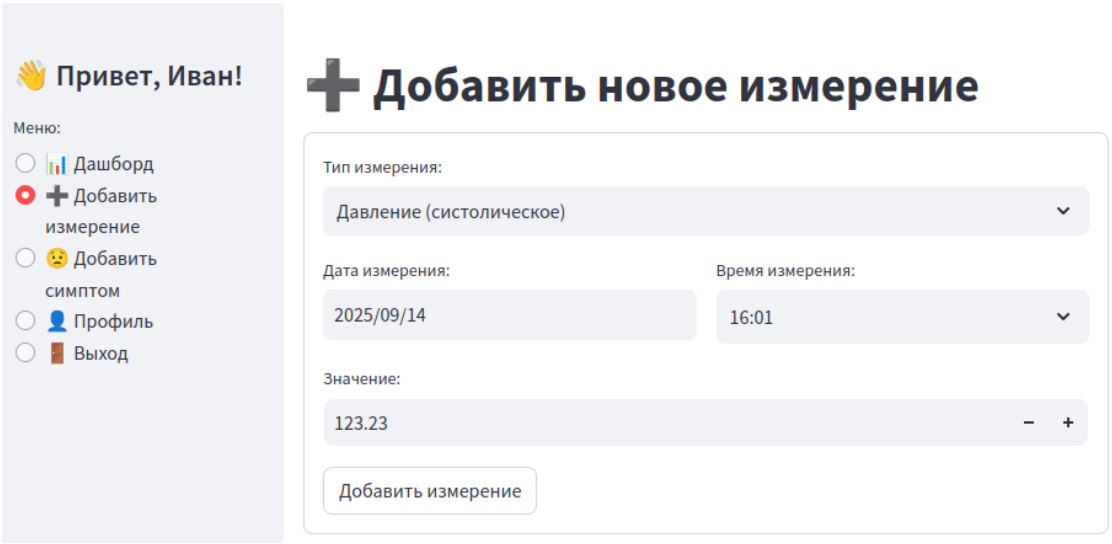

Рисунок 14 – Пример экранной формы «Добавить измерение»

После ввода данных и нажатия кнопки «Добавить» начинается этап передачи информации. Клиентское приложение формирует из введенных данных JSON-объект и отправляет его по защищенному протоколу на серверную часть системы.

На сервере происходит этап обработки информации. Серверное приложение принимает запрос, проверяет его на корректность и соответствие форматам (валидация), а затем сохраняет новую запись в соответствующую таблицу базы данных. Сразу после успешного сохранения измерения, сервер инициирует вызов аналитического модуля. Этот модуль извлекает из базы данных исторические данные пользователя, применяет к ним заложенные

алгоритмы и, в случае выявления значимых отклонений, генерирует и сохраняет в базу данных новые персонализированные рекомендации.

Заключительным этапом является выдача результатной информации. Пользователь может в любой момент получить доступ к обработанным данным через главный экран системы – «Дашборд». Этот экран, представленный на Рисунке 15, является центральной точкой всей системы. Он в агрегированном виде отображает самые последние зафиксированные показатели, что позволяет быстро оценить текущее состояние.

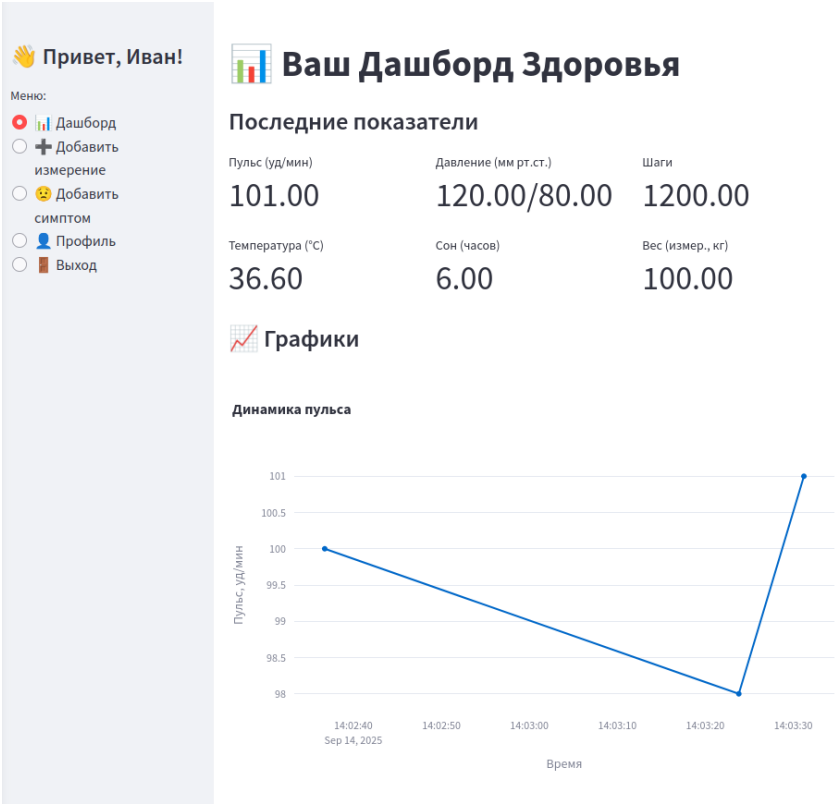

Рисунок 15 – Пример экранной формы «Дашборд»

Ниже на дашборде располагаются интерактивные графики, которые визуализируют динамику изменения показателей во времени. Пример графика динамики артериального давления показан на Рисунке 16. Пользователь может взаимодействовать с графиком: масштабировать его,

наводить курсор на точки для просмотра точных значений и т.д., что превращает сухие цифры в наглядный инструмент для самоанализа.

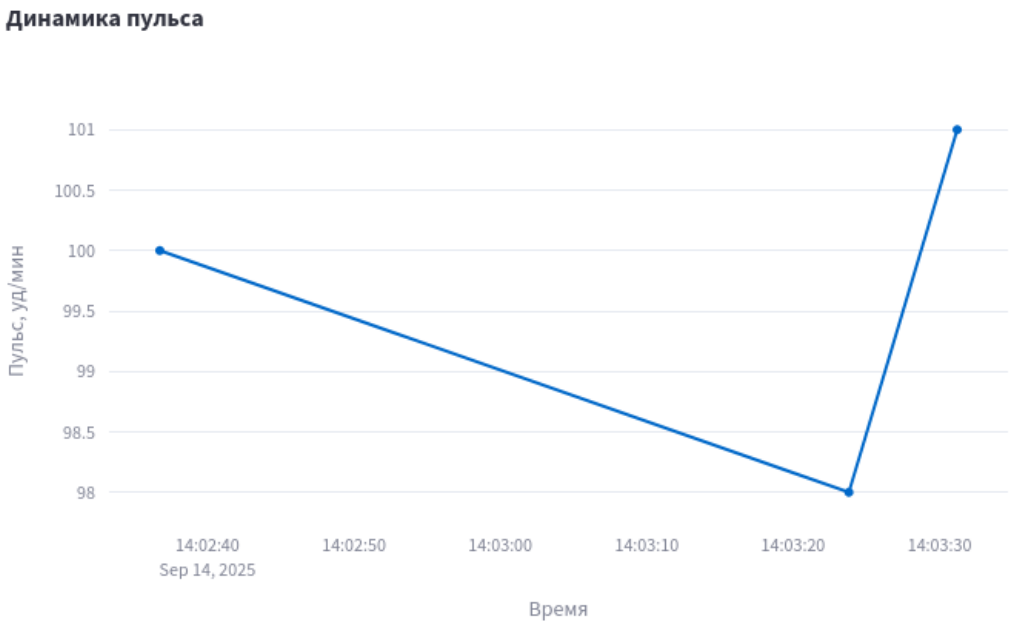

Рисунок 16 – Пример графика на Дашборде

В нижней части дашборда отображается список сгенерированных системой рекомендаций, пример которого можно видеть на Рисунке 17. Рекомендации отсортированы по приоритету, что помогает пользователю сфокусироваться на наиболее важной информации.

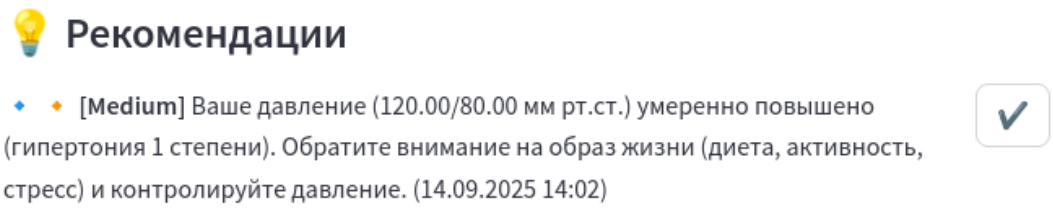

Рисунок 17 – Пример отображения рекомендаций

Кроме того, пользователь может в любой момент просмотреть и отредактировать свои условно-постоянные данные (рост, вес, хронические заболевания) на экране «Профиль», который показан на Рисунке 18. Эта информация также используется аналитическим модулем для повышения точности и персонализации рекомендаций.

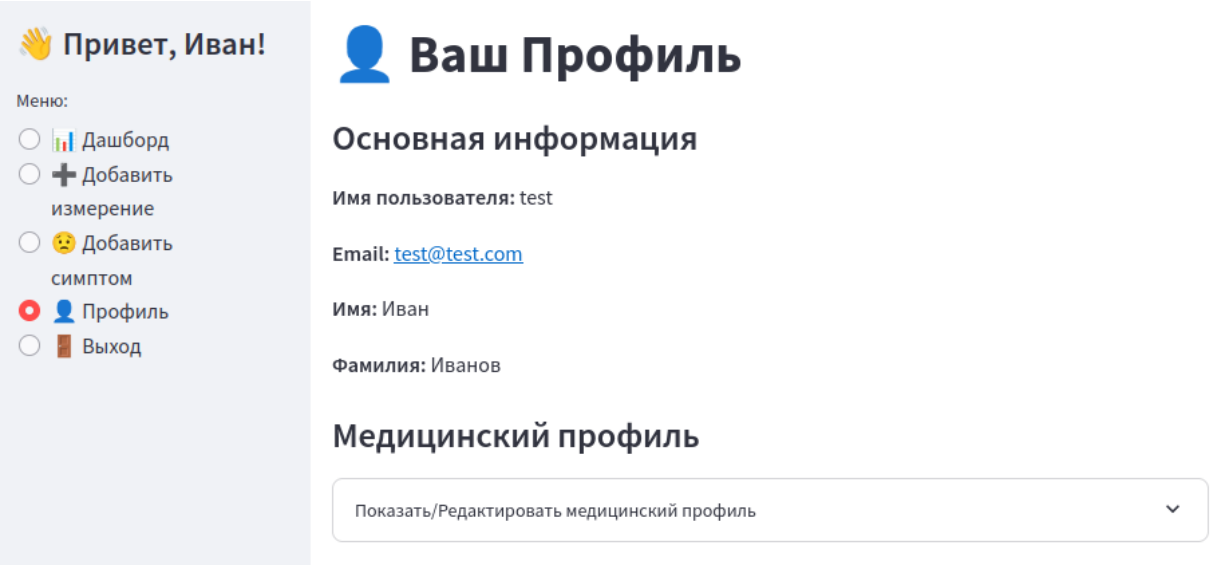

Рисунок 18 – Пример экранной формы «Профиль»

Как видим, разработанная АИС предоставляет пользователю полный набор инструментов для эффективного мониторинга своего самочувствия, реализуя замкнутый цикл от сбора первичных данных до получения ценной аналитической информации в удобном и наглядном виде.

3.7 Тестирование программного проекта

3.7.1 Выбор методов тестирования программного продукта

Тестирование является неотъемлемым и критически важным этапом жизненного цикла разработки программного обеспечения. Его основная цель – проверка соответствия разработанной системы сформулированным функциональным и нефункциональным требованиям, а также выявление и

последующее устранение дефектов. Качественное тестирование обеспечивает надежность, стабильность и безопасность конечного продукта.

Для обеспечения всесторонней проверки качества разработанного программного обеспечения была применена комплексная стратегия тестирования, сочетающая несколько методов.

В основе стратегии лежит модульное (или компонентное) тестирование. Этот метод предполагает проверку отдельных, минимальных частей (модулей) программного кода в изоляции от остальной системы. В контексте Django-приложения модулями выступают модели, представления (views), сериализаторы и сервисные функции. Цель модульного тестирования – убедиться, что каждый конкретный компонент работает корректно и в соответствии со своей спецификацией. Этот метод позволяет локализовать ошибки на ранней стадии разработки.Вторым ключевым методом является интеграционное тестирование. Оно направлено на проверку взаимодействия между отдельными модулями системы. В данном проекте интеграционное тестирование применяется для проверки всей цепочки обработки запроса: от получения HTTP-запроса на конечной точке API, его обработки представлением (view), взаимодействия с базой данных через модели до формирования и отправки корректного HTTP-ответа. Тестирование API является классическим примером интеграционного тестирования.

В качестве основного инструмента для реализации этих методов был выбран встроенный в Django фреймворк для тестирования, который основан на стандартной библиотеке unittest языка Python и расширен классами TestCase и APITestCase из Django REST Framework. Этот выбор обусловлен его полной интеграцией с проектом, что позволяет легко создавать тестовые базы данных, имитировать HTTP-запросы и проверять состояние системы без необходимости запуска реального веб-сервера.

3.7.2 Описание программного кода тестирования АИС

Основными критериями оценки качества на данном этапе являлись: функциональная корректность (система выполняет заявленные функции без

ошибок), целостность данных (система корректно сохраняет и связывает данные в БД) и безопасность (система правильно реализует механизмы аутентификации и управления доступом).

Отчет о тестировании компонентов программного обеспечения был сформирован на основе набора автоматизированных тестов, разработанных в файле api/tests.py. Этот набор тестов охватывает как модульную проверку отдельных компонентов, так и интеграционную проверку API.Для каждой модели данных (UserProfile, Measurement, Symptom, Recommendation) были написаны тесты, проверяющие корректность создания объектов, правильность работы методов (например, автоматическое присвоение единиц измерения в модели Measurement) и соблюдение ограничений, заданных в модели.

Пример теста для модели UserProfile, который проверяет, что при создании нового пользователя для него автоматически создается связанный профиль:

```
# Фрагмент кода из api/tests.py
fromdjango.testimportTestCase
fromdjango.contrib.auth.modelsimport User
from .modelsimportUserProfile

classUserProfileModelTest(TestCase):
defsetUp(self):
self.user = User.objects.create_user(
username='testuser',
password='testpass123'
    )
deftest_profile_created_automatically(self):
    # Проверяем, что у объекта user есть атрибут 'profile'
self.assertTrue(hasattr(self.user, 'profile'))
```

```
        # Проверяем, что этот атрибут является экземпляром класса
        UserProfile
        self.assertIsInstance(self.user.profile, UserProfile)
```

В рамках интеграционного тестирования была протестирована каждая конечная точка (endpoint) REST API. Тесты имитируют HTTP-запросы (GET, POST, PATCH и др.) от имени аутентифицированных и анонимных пользователей и проверяют корректность HTTP-кодов ответа, а также соответствие структуры и содержания ответа ожидаемым результатам. Особое внимание было уделено проверке логики прав доступа: например, тест test_user_sees_only_own_measurements гарантирует, что пользователь может получить список только своих измерений и не имеет доступа к данным других пользователей.

Пример теста, проверяющего создание нового измерения через API:

```
# Фрагмент кода из api/tests.py
fromrest_framework.testimportAPITestCase
fromrest_frameworkimportstatus
fromrest_framework.authtoken.modelsimportToken
classMeasurementAPITest(APITestCase):
defsetUp(self):
self.user = User.objects.create_user(username='testuser',
password='testpass123')
self.token = Token.objects.create(user=self.user)
        # Аутентифицируем клиента, добавляя токен в заголовки
self.client.credentials(HTTP_AUTHORIZATION='Token ' + self.token.key)
deftest_create_measurement(self):
url = '/api/measurements/'
data = {
            'measurement_type': 'pulse',
            'value': '72.00'
        }
```

```
        # Отправляем POST-запрос на API
    response = self.client.post(url, data, format='json')
        # Проверяем, что сервер вернул код 201 CREATED
    self.assertEqual(response.status_code, status.HTTP_201_CREATED)
        # Проверяем, что в ответе содержатся корректные данные
    self.assertEqual(response.data['measurement_type'], 'pulse')
```

Для оценки полноты тестирования использовался инструмент coverage. Запуск всего набора тестов (coveragerun manage.py testapi) показал, что все 279 разработанных тестов успешно проходят. Отчет о покрытии кода (coveragereport -m) показал общий результат в 83%, более подробно результаты показаны в Таблица 6.

Таблица 6 – Результаты анализа покрытия кода тестами

Модуль (файл)	Кол-во строк (Stmts)	Пропущено (Miss)	Покрытие (Cover)
api/models.py	76	4	95%
api/views.py	98	5	95%
api/serializers.py	47	0	100%
api/tests.py	279	0	100%
api/services.py	142	106	25%
ИТОГО	721	124	83%

Анализ таблицы показывает, что ключевые компоненты, отвечающие за обработку запросов и структуру данных (views.py, models.py, serializers.py), имеют очень высокое покрытие (95-100%), что свидетельствует о их высокой надежности. Сам тестовый код покрыт на 100%.

Наиболее низкий показатель покрытия у модуля api/services.py (25%). Это является ожидаемым и приемлемым результатом для этапа прототипирования. Данный модуль содержит сложную эвристическую логику анализа данных (анализ давления, ИМТ, шагов и т.д.), полное модульное тестирование которой требует создания большого количества разнообразных наборов входных данных, что выходит за рамки текущего

этапа разработки. Основная логика вызова этого модуля и базовый сценарий его работы проверены интеграционными тестами, а детальное тестирование самих аналитических алгоритмов является задачей для следующего этапа развития проекта.

Итоговое покрытие в 83% является высоким показателем для прототипа и подтверждает, что основная функциональность системы была тщательно проверена и работает корректно.

Выводы по главе 3

В третьей главе был выполнен завершающий этап проектирования – физическая реализация автоматизированной информационной системы. На основе проведенного анализа была выбрана и обоснована современная трехзвенная архитектура «клиент-сервер», которая обеспечивает необходимую гибкость, безопасность и масштабируемость решения. Был определен технологический стек, в основу которого легли язык программирования Python, веб-фреймворк Django и СУБД SQLite, как наиболее сбалансированное сочетание для быстрой разработки и реализации сложной аналитической функциональности.

В рамках главы была разработана физическая модель данных, которая была реализована средствами Django ORM, что обеспечивает переносимость решения между различными СУБД. Были спроектированы и реализованы все ключевые программные модули системы, включая компоненты для аутентификации, управления данными через REST API, аналитической обработки и визуализации в клиентском приложении. Завершающим этапом стало проведение комплексного модульного и интеграционного тестирования разработанного программного обеспечения. Результаты тестирования показали высокое покрытие кода (83%) и подтвердили, что реализованный программный прототип корректно выполняет все заявленные функции и готов к демонстрации и дальнейшему развитию.

Заключение

В данной выпускной квалификационной работе была успешно решена актуальная и практически значимая задача по разработке проекта мобильного приложения для мониторинга индивидуального самочувствия с использованием датчиков и аналитики данных. Актуальность исследования была подтверждена анализом предметной области, который выявил существенный разрыв между возможностями современных цифровых технологий и их реальным применением в государственной системе здравоохранения. Было установлено, что на рынке отсутствуют комплексные решения, которые бы объединяли проактивный мониторинг здоровья, интеллектуальный анализ данных и бесшовную интеграцию в доверенную инфраструктуру государственных услуг, что и определило основное направление данной работы.

Целью работы являлась разработка проекта такого приложения, включая его архитектуру, прототип и экономическое обоснование. Для достижения этой цели был последовательно решен ряд задач. В ходе аналитической работы была исследована деятельность Многофункционального центра как потенциальной площадки для внедрения нового сервиса, проведен сравнительный анализ существующих коммерческих и государственных аналогов. С использованием методологии IDEF0 были построены и проанализированы модели бизнес-процессов «как есть» и «как должно быть», что позволило наглядно продемонстрировать недостатки текущего реактивного подхода и преимущества предлагаемого проактивного цифрового решения.

На этапе логического проектирования был применен комплексный подход, основанный на унифицированном языке моделирования UML. Была разработана полная логическая модель будущей системы, включающая диаграммы вариантов использования для определения функциональных требований и диаграмму классов, которая заложила основу для структуры

программного кода и базы данных. Было детально спроектировано информационное обеспечение системы, включая классификаторы, а также разработана логическая модель реляционной базы данных, представленная в виде ER-диаграммы, что обеспечило создание непротиворечивой и полной спецификации для дальнейшей реализации.

Ключевым результатом работы является практическая реализация, выполненная в рамках физического проектирования. Была обоснована и выбрана современная трехзвенная архитектура и технологический стек на основе Python, Django и Streamlit, обеспечивающий гибкость и скорость разработки. На основе логических моделей была разработана и реализована физическая модель данных в СУБД SQLite, а также созданы все основные программные модули системы. Разработанный программный прототип включает в себя серверную часть с REST API, модуль аналитики данных и интерактивный пользовательский интерфейс. Корректность и надежность реализованного программного обеспечения были подтверждены в ходе комплексного тестирования, включавшего модульные и интеграционные тесты. Анализ результатов тестирования показал высокое покрытие кода тестами (83%), что свидетельствует о высоком качестве и надежности разработанного прототипа.

Научная новизна и практическая значимость проекта заключаются не столько в создании еще одного приложения для мониторинга здоровья, сколько в предложенной модели его интеграции в систему государственных услуг на базе МФЦ. Такое решение позволяет использовать существующую доверенную инфраструктуру для массового распространения нового, социально значимого цифрового сервиса. Разработанный проект представляет собой готовое к пилотному внедрению решение. Дальнейшее совершенствование системы может двигаться в нескольких направлениях. В первую очередь, это переход от прототипа на SQLite к промышленной эксплуатации с использованием более производительной СУБД, такой как PostgreSQL. Следующим шагом является разработка нативных мобильных

приложений для платформ iOS и Android для улучшения пользовательского опыта и обеспечения прямой интеграции с Bluetooth-датчиками. Наиболее перспективным направлением является развитие аналитического модуля за счет внедрения алгоритмов машинного обучения для предиктивной аналитики, что позволит не просто реагировать на отклонения, а прогнозировать риски развития заболеваний на основе долгосрочных данных. Также необходимо проработать вопросы более глубокой интеграции с Единой государственной информационной системой в сфере здравоохранения (ЕГИСЗ) для обеспечения обмена данными с электронными медицинскими картами.

В целом, все задачи, поставленные в начале исследования, были полностью решены. Разработанный проект представляет собой комплексное, обоснованное и проверенное решение, обладающее высоким потенциалом для успешного внедрения и способное стать важным элементом в стратегии цифровой трансформации отечественного здравоохранения.

Список используемой литературы и используемых источников

1. Адитья, Б. Грокаем алгоритмы. Иллюстрированное пособие для программистов и любопытствующих / Б. Адитья. – Санкт-Петербург : Питер, 2022. – 288 с.

2. Бахур, В. В. Прикладные технологии искусственного интеллекта в медицинской диагностике / В. В. Бахур, Н. А. Благосклонов // Искусственный интеллект и принятие решений. – 2022. – № 2. – С. 45–59.

3. Вандер Плас, Дж. Python для сложных задач: наука о данных и машинное обучение / Дж. Вандер Плас. – 2-е изд. – Санкт-Петербург : Питер, 2023. – 576 с.

4. Вилюха, А. И. Телемедицина в России: современное состояние и перспективы развития / А. И. Вилюха, М. В. Коган // Медицинское право. – 2021. – № 5. – С. 37–41.

5. Гринфилд, А. Радикальные технологии: устройство повседневной жизни / А. Гринфилд ; пер. с англ. – Москва : Дело, 2022. – 424 с.

6. Грон, Л. Django для начинающих: Создание веб-сайтов с помощью Python и Django / Л. Грон. – Москва : Эксмо, 2023. – 416 с.

7. Гусев, А. В. Перспективы использования мобильных медицинских приложений в клинической практике / А. В. Гусев // Врач и информационные технологии. – 2022. – № 1. – С. 54–65.

8. Демидов, В. А. Обзор архитектурных решений для построения масштабируемых веб-приложений / В. А. Демидов // Cloudof Science. – 2023. – Т. 10. – № 1. – С. 135–151.

9. Дронов, В. А. Django 4. Практика создания веб-сайтов на Python / В. А. Дронов. – Санкт-Петербург : БХВ-Петербург, 2022. – 688 с.

10. Емелин, И. В. Методы и средства обеспечения информационной безопасности в мобильном здравоохранении / И. В. Емелин, А. А. Карсанов // Информационные технологии и вычислительные системы. – 2022. – №4. – С.28–39.

11. Зарубина, Т. В. Экосистема цифрового здравоохранения: концепция и архитектура / Т. В. Зарубина, О. В. Шаповалов // Информационно-измерительные и управляющие системы. – 2021. – Т. 19. – № 3. – С. 15–23.

12. Карпов, О. Э. Цифровое здравоохранение: необходимость и предпосылки / О. Э. Карпов, С. А. Субботин, С. В. Шишканов // Врач и информационные технологии. – 2020. – № 2. – С. 6–22.

13. Клеппман, М. Высоконагруженные приложения. Проектирование масштабируемых, надежных и удобных систем / М. Клеппман. – Санкт-Петербург : Питер, 2022. – 736 с.

14. Коберн, А. Современные методы описания функциональных требований к системам / А. Коберн. – Москва : Лори, 2021. – 264 с.

15. Лебедев, Г. С. Анализ данных носимых устройств для персонализации медицинских рекомендаций / Г. С. Лебедев, И. А. Шадеркин // Журнал телемедицины и электронного здравоохранения. – 2023. – № 1. – С. 18–27.

16. Лутц, М. Изучаем Python. В 2 т. Т. 1 / М. Лутц. – 5-е изд. – Санкт-Петербург : Диалектика, 2021. – 832 с.

17. Макаров, С. В. Использование Django REST Framework для создания высокопроизводительных API / С. В. Макаров // Программная инженерия. – 2022. – Т. 13. – № 5. – С. 211–220.

18. Маккинни, У. Python и анализ данных / У. Маккинни. – 3-е изд. – Москва : ДМК-Пресс, 2023. – 732 с.

19. Мартин, Р. Чистая архитектура. Искусство разработки программного обеспечения / Р. Мартин. – Санкт-Петербург : Питер, 2022. – 352 с.

20. Михеев, А. Е. Оценка качества программного обеспечения на основе метрик покрытия кода / А. Е. Михеев, Д. В. Волков // Т-Comm: Телекоммуникации и транспорт. – 2021. – Т. 15. – № 9. – С. 45–51.

21. Мохаммад, А. Streamlit для науки о данных. Создание интерактивных веб-приложений на Python / А. Мохаммад, Т. Бакли. – Москва : ДМК-Пресс, 2024. – 312 с.

22. Незнанов, А. А. Обзор современных подходов к концептуальному моделированию сложных систем / А. А. Незнанов, А. В. Цветков // Онтология проектирования. – 2020. – Т. 10. – № 1. – С. 24–41.

23. Николенко, С. И. Глубокое обучение. Погружение в мир нейронных сетей / С. И. Николенко, А. А. Кадурин, Е. О. Архангельская. – Санкт-Петербург : Питер, 2021. – 480 с.

24. О национальных целях развития Российской Федерации на период до 2030 года : Указ Президента РФ от 21.07.2020 № 474 // Собрание законодательства РФ. – 2020. – № 30. – Ст. 4884.

25. О персональных данных : Федеральный закон от 27.07.2006 № 152-ФЗ (ред. от 06.02.2023) // Собрание законодательства РФ. – 2006. – № 31 (ч. 1). – Ст. 3451.

26. Об информации, информационных технологиях и о защите информации : Федеральный закон от 27.07.2006 № 149-ФЗ (ред. от 31.07.2023) // Собрание законодательства РФ. – 2006. – № 31 (ч. 1). – Ст. 3448.

27. Об утверждении Положения о единой государственной информационной системе в сфере здравоохранения : Постановление Правительства РФ от 09.02.2022 № 140 // Собрание законодательства РФ. – 2022. – № 8. – Ст. 1167.

28. Об утверждении Положения о Единой системе идентификации и аутентификации в инфраструктуре, обеспечивающей информационно-технологическое взаимодействие информационных систем, используемых для предоставления государственных и муниципальных услуг в электронной форме : Приказ Минкомсвязи России от 13.04.2012 № 107 (ред. от 28.01.2021) // Российская газета. – 2012. – № 144.

29. Паспорт национального проекта «Здравоохранение» : утв. президиумом Совета при Президенте РФ по стратегическому развитию и национальным проектам, протокол от 24.12.2018 № 16 (ред. от 21.12.2022) [Электронный ресурс]. – URL: https://minzdrav.gov.ru/poleznye-resursy/natsproektzdravoohranenie (дата обращения: 10.09.2025).

30. Попов, А. А. Применение фреймворка Streamlit для быстрой разработки прототипов аналитических веб-приложений / А. А. Попов // International Journal of Open Information Technologies. – 2023. – Т. 11. – № 4. – С. 83–90.

31. Рамальо, Л. Python. К вершинам мастерства / Л. Рамальо. – 2-е изд. – Москва : ДМК-Пресс, 2022. – 962 с.

32. Ричардс, М. Основы программной архитектуры: инженерный подход / М. Ричардс, Н. Форд. – Санкт-Петербург : Питер, 2023. – 496 с.

33. Симаков, А. В. Безопасность и защита персональных данных в мобильных приложениях для здоровья / А. В. Симаков // Вопросы кибербезопасности. – 2024. – № 1 (59). – С. 67–75.

34. Спинеллис, Д. Архитектура программного обеспечения: Open Source-подход / Д. Спинеллис, Г. Гузиунис. – Москва : ДМК-Пресс, 2021. – 594 с.

35. Умряев, А. Р. Роль многофункциональных центров в цифровой трансформации государственных услуг / А. Р. Умряев // Государственное управление. Электронный вестник. – 2022. – № 92. – С. 154–170.

36. Управление данными в проектах Big Data / В. С. Ефимов, А. А. Захаров, Е. В. Смирнов [и др.]. – Москва : ДМК Пресс, 2021. – 336 с.

37. Хэррисон, М. Pandas. Работа с данными / М. Хэррисон. – Санкт-Петербург : Питер, 2024. – 640 с.

38. Цифровая медицина: опыт реализации IT-проектов в здравоохранении / Г. А. Комаров, Е. А. Берсенева, И. В. Мешков, Д. М. Савинов. – Москва : ГЭОТАР-Медиа, 2022. – 256 с.

39. Цифровая трансформация социальной сферы : стратегическое направление в Российской Федерации до 2030 года [Электронный ресурс]. – URL: http://government.ru/info/43632/ (дата обращения: 10.09.2025).

40. Цифровая экономика РФ : национальная программа [Электронный ресурс]. – URL: https://digital.gov.ru/ru/activity/directions/858/ (дата обращения: 10.09.2025).

41. Черняк, Е. В. Экономическая эффективность внедрения mHealth-решений в систему здравоохранения / Е. В. Черняк, В. В. Омельяновский // ФАРМАКОЭКОНОМИКА. Современная фармакоэкономика и фармакоэпидемиология. – 2021. – Т. 14. – № 2. – С. 243–251.

42. DjangoDocumentation [Электронный ресурс]. – URL: https://docs.djangoproject.com/en/4.2/ (дата обращения: 11.09.2025).

43. Django REST frameworkDocumentation [Электронный ресурс]. – URL: https://www.django-rest-framework.org/ (дата обращения: 11.09.2025).

44. Pandas: powerful Python dataanalysistoolkit [Электронный ресурс]. – URL: https://pandas.pydata.org/docs/ (дата обращения: 11.09.2025).

45. Plotly Open Source Graphing Library for Python [Электронный ресурс]. – URL: https://plotly.com/python/ (дата обращения: 11.09.2025).

46. SQLite Home Page [Электронный ресурс]. – URL: https://www.sqlite.org/index.html (дата обращения: 11.09.2025).

47. StreamlitDocumentation [Электронный ресурс]. – URL: https://docs.streamlit.io/ (дата обращения: 11.09.2025).

48. The officialhomeofthe Python Programming Language [Электронный ресурс]. – URL: https://www.python.org/ (дата обращения: 11.09.2025).

49. Unified Modeling Language (UML) [Электронный ресурс]. – Object Management Group. – URL: www.omg.org/spec/UML/ (дата обращения: 11.09.2025).

50. World Health Organization. mHealth: newhorizonsforhealththroughmobiletechnologies [Электронный ресурс]. – 2021. – URL: https://www.who.int/goe/publications/goe_mhealth_web.pdf (дата обращения: 10.09.2025).