Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего образования
«Уральский федеральный университет
имени первого Президента России Б.Н. Ельцина»

Институт радиоэлектроники и информационных технологий - РтФ

Кафедра/департамент информационных технологий и систем управления

ДОПУСТИТЬ К ЗАЩИТЕ В ГЭК
Заведующий кафедрой
_____ Е.В. Кислицын
(подпись) (Ф.И.О.)
«_____»_____ 2024 г.

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

РАЗРАБОТКА СЕРВЕРНОЙ ЧАСТИ ПРИЛОЖЕНИЯ ДЛЯ СОПРОВОЖДЕНИЯ ВИЧ+ ДЕТЕЙ

Научный руководитель: Юманова Ирина Фарисовна
к.ф.-м.н., доцент _____
подпись

Нормоконтролер: Абакумова Анна Геннадьевна _____
подпись

Студент группы РИМ-220990 Савицкий Владислав Юрьевич _____
подпись

Екатеринбург
2024

Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего образования
**«Уральский федеральный университет
имени первого Президента России Б.Н. Ельцина»**

Институт *радиоэлектроники и информационных технологий - РтФ*
Кафедра/департамент *информационных технологий и систем управления*
Направление подготовки *09.04.04 Программная инженерия*
Образовательная программа *Разработка и управление в программных проектах*

ЗАДАНИЕ

на выполнение выпускной квалификационной работы

студента Савицкого Владислава Юрьевича **группы** *РИМ-220990*

(фамилия, имя, отчество)

1. Тема выпускной квалификационной работы

«Разработка серверной части приложения для сопровождения ВИЧ+ детей»

Утверждена распоряжением по институту от «04» декабря 2023 г. № 33.02-05/298

2. Научный руководитель

Юманова Ирина Фарисовна, старший научный сотрудник, доцент кафедры ИТиСУ, к.ф.-м.н.
(Ф.И.О., должность, ученая степень, ученое звание)

3. Исходные данные к работе *результаты производственной практики (научно-исследовательской работы)*

4. Перечень демонстрационных материалов

Презентационный материал по итогам выполнения ВКР

5. Календарный план

№ п/п	Наименование этапов выполнения работы	Срок выполнения этапов работы	Отметка о выполнении
1.	*1 раздел (глава)*	до 12.04.2024 г.	
2.	*2 раздел (глава)*	до 03.05.2024 г.	
3.	*3-4 разделы (главы)*	до 17.05.2024 г.	
4.	*ВКР в целом*	до 20.05.2024 г.	

Руководитель Юманова Ирина Фарисовна _____
Ф.И.О. (подпись)

Студент задание принял к исполнению _____ В. Савицкий
дата (подпись)

6. Допустить Савицкого Владислава Юрьевича **к защите выпускной квалификационной работы в экзаменационной комиссии**

Зав. кафедрой **ИТиСУ** _____ Е.В. Кислицын
(подпись) Ф.И.О.

СОДЕРЖАНИЕ

РЕФЕРАТ

Магистерская диссертация состоит из введения, трех глав и заключения, изложенных на 59 страницах, а также библиографического списка. В работе имеется 6 рисунков и 3 таблицы. Библиографический список состоит из 50 наименований.

API ДЛЯ МОБИЛЬНОГО ПРИЛОЖЕНИЯ, ВЕБ-РАЗРАБОТКА, МЕДИЦИНСКОЕ ПРИЛОЖЕНИЕ, ASP.NET CORE, DOCKER.

Цель работы – разработка и реализация серверной части приложения для сопровождения ВИЧ+ детей.

Объект исследования – процесс сопровождения с использованием информационных технологий пациентов СПИД-центра до 18 лет.

Методы исследования: изучение научной и технической литературы, методы математической статистики и математической теории, методы алгоритмизации.

Результаты работы: функционирующий web-сервис, развернутый на облачном сервере.

ВВЕДЕНИЕ

СПИД, как одна из важнейших социальных проблем, возникла перед человечеством в конце XX века. В настоящее время в мире официально зарегистрировано более 40 миллионов ВИЧ-инфицированных, а СПИД относится к числу пяти главных заболеваний, уносящих наибольшее число жизней на планете.

Распространение данного инфекционного заболевания затрагивает все сферы жизни общества и касается не только взрослых, но и детей, заражение которых может происходить внутриутробно, во время родовой деятельности, а также при вскармливании грудным молоком женщины, инфицированной ВИЧ. Проблема ВИЧ-инфекции у детей продолжает оставаться крайне актуальной: в настоящее время в Российской Федерации насчитывается около десяти тысяч детей, живущих с ВИЧ-инфекцией, из них 89% получает антиретровирусную терапию (АРВТ), 4% погибло от синдрома приобретенного иммунодефицита (СПИДа).

Учитывая неуклонный рост числа ВИЧ-инфицированных детей в России, необходимо принимать регулярные меры по оптимизации лечения ВИЧ-инфекции. В связи с этим, сопровождение и поддержка ВИЧ-положительных детей и их семей является особенно важной задачей для различных организаций и общественных структур: на данный момент в Российской Федерации развиваются различные технологии и организуется множество программ для поддержки ВИЧ-положительных детей. Одно из направлений – создание мобильных приложений, специально разработанных для сопровождения и поддержки данной категории детей. Основной целью данных приложений является эффективное напоминание о приеме лекарственных препаратов, а также обеспечение точного контроля принятых медикаментов.

Анализ актуальности обусловили выбор темы исследования: «Разработка серверной части приложения для сопровождения ВИЧ+ детей».

Гипотеза разработки состоит в том, что выделение определенного набора из существующих технологий и алгоритмических и архитектурных подходов позволит создать приложение, отвечающее современным требованиям безопасности, производительности, отказоустойчивости и легкости в поддержке.

Целью исследования является разработка и реализация серверной части приложения для сопровождения ВИЧ+ детей.

Для достижения поставленной цели необходимо решить следующие задачи:

– исследование и обоснованный выбор технологий, подходов и архитектур в разработке backend-решений;

– разработка web-сервиса для полноценной работы мобильного приложения;

– развитие инфраструктуры разработки: оптимизация взаимодействия с другими членами команды, документирование backend-решений;

– реализация базового мониторинга и работы по увеличению отказоустойчивости системы;

– экспериментальная проверка разработанных решений через пользовательское тестирование и его анализ.

Объектом исследования является процесс сопровождения с использованием информационных технологий пациентов СПИД-центра до 18 лет.

Предметом исследования является специализированный web-сервис для работы с мобильным приложением.

Методы исследования включают в себя:

1. Методы теоретического исследования, такие как изучение научной и технической литературы;

2. Методы математической статистики для принятия решений и оценки параметров моделей, в частности, корреляционный анализ, тестирование

гипотез и доверительные интервалы;

3. Методы математической теории принятия решений, в частности, применение теории оптимизации, принятие решений в условиях неопределенности;

4. Методы алгоритмизации и программной реализации математических моделей, включая выбор подходящих алгоритмов, структур данных и программных платформ для реализации и тестирования;

5. Пилотное тестирование в целях выявления проблем, с которыми могут столкнуться пользователи, и сбора обратной связи до широкого выпуска приложения.

Теоретической основой исследования стали:

1. Отечественные и зарубежные исследования по проблеме создания безопасных и отказоустойчивых серверных решений, с которыми взаимодействуют мобильные приложения;

2. Современные концепции, раскрывающие сущность создания архитектурно правильных систем, поддержка которых в будущем не будет доставлять больших временных и ресурсных затрат.

Теоретическая и практическая значимость работы заключается в том, что внедрение разработанного приложения для детей способствует формированию навыка принятия лекарств вовремя и в полном объеме. Это поможет детям снизить риск развития болезни до более тяжелых стадий.

База исследования: статистические данные о вовлеченности пациентов СПИД-центра Свердловской области в терапию.

Апробация результатов исследования и публикации были представлены в пояснительной записке к заявке на грант «Старт-1», который был получен командой проекта и на средства которого ведется разработка приложения. Также результаты исследования были опубликованы на студенческой научной конференции «ИНТЕР», где статья «Информационная система для сопровождения ВИЧ+ детей» с этим исследованием заняла призовое место.

Магистерская диссертация состоит из введения, трех глав и заключения, изложенных на 59 страницах, а также библиографического списка, состоящего из 50 наименований. В работе имеется 6 рисунков и 3 таблицы.

Во введении описана актуальность работы, её анализ, а также то, как она повлияла на выбор темы исследования. Поставлена гипотеза и цель исследования совместно с задачами, которые нужно выполнить для их проверки и достижения. Описаны предмет, объект и методы исследования. Обозначена база исследования, а также теоретическая и практическая значимость работы.

В первой главе приведено общее описание всего разрабатываемого проекта. Проведен анализ существующих исследований по теме для постановки проблем решения, а также обусловлена важность проводимого исследования.

Вторая глава посвящена проектированию реализуемой системы, анализу и выбору необходимых технологий и методологий для разработки. Определены требования к серверной части и поставлено частное техническое задание. Отдельными подразделами вынесена информация по управлению проектом. Проведена оценка рисков и затрат проекта.

В третьей части описывается реализация и архитектура системы. Рассказывается о подходах, использованных при создании инфраструктуры вокруг приложения, методах логирования и мониторинга развернутого API.

В заключении приведены выводы по итогам реализации проекта.

1 Анализ предметной области

1.1 Общее описание проекта

В рамках работы произведена разработка API для мобильного приложения, предназначенного для сопровождения детей и подростков, болеющих ВИЧ-инфекцией. Мобильное приложение обеспечивает такие функции, как регистрация и аутентификация пользователей, установка разовых или периодических напоминаний о приеме таблеток, общение пациента с врачом в чате, а также игровой режим.

Отличительной особенностью приложения стала геймификация. В игре реализованы несколько мини-игр, за прохождение которых игрок получает опыт и внутриигровую валюту, необходимую для покупки и открытия различных элементов персонализации и развития. Геймификация должна способствовать регулярности приема медикаментов пациентами: выпил таблетку – получил бонус в игре за её прохождение – быстрее развиваешься в приложении.

1.2 Исследование проблемы регулярности приема медикаментов

Вопрос регулярного приема лекарственных средств, который стоит в центре внимания разработанной системы является известной проблемой. Существуют исследования, посвященные решению подобных задач, особенно в контексте медицинской сферы, которые будут рассмотрены далее. Эти исследования охватывают широкий спектр подходов и методологий, направленных на улучшение соблюдения пациентами назначений врачей, что в свою очередь способствует повышению эффективности лечения и улучшению общего состояния здоровья. Разработанное приложение стремится внести свой вклад в эту область, предлагая новаторские решения и

улучшения, которые могут помочь пациентам более строго следовать медицинским рекомендациям.

Проблема приверженности остается одной из актуальных, и при этом наиболее сложно решаемых для современных медицины и общества. Низкая приверженность может привести к ухудшению состояния здоровья, увеличению количества госпитализаций и приему дополнительных лекарств. Это может также снизить качество жизни детей и их семей. Из-за широкой распространенности, а также серьезных последствий несоблюдения назначенной пациенту терапии, ВОЗ считает явление низкой приверженности к лечению “глобальной проблемой поразительного масштаба” [2–5].

В докладе Всемирной организации здравоохранения 2003 года низкий уровень комплаенса при различных заболеваниях отмечается как причина высокого уровня осложнений, ухудшения качества жизни и смертности [6].

Недостаточный комплаенс – это распространенное явление не только в нашей стране, но и за рубежом. По оценке ВОЗ, в индустриально развитых странах лишь около 50% пациентов, страдающих хроническими заболеваниями, достаточно длительное время точно соблюдают врачебные рекомендации, в развивающихся странах – еще меньше.

Несоблюдение рекомендаций и указаний лечащего врача может иметь серьезные последствия как для здоровья пациента, так и для его социального и экономического благополучия. В результате таких действий или бездействий часто происходит ухудшение эффективности лечения, возникновение осложнений, развитие резистентности к лекарствам, а также увеличение риска осложнений и рецидивов заболевания. Эти прямые последствия могут стать лишь началом цепочки негативных последствий, так как несоблюдение рекомендаций ведет к затратам на повторные визиты к врачам, дополнительные обследования, назначение дорогостоящих лекарств и увеличению числа госпитализаций.

Несоблюдение врачебных предписаний также приводит к косвенным социальным и экономическим последствиям. Пациенты могут испытывать

снижение качества жизни, утрату трудоспособности и связанное с этим уменьшение доходов. Для общества это означает уменьшение производительности труда и увеличение расходов на социальное обеспечение. А для системы здравоохранения – еще более высокие затраты, связанные с повторными обращениями за медицинской помощью, удлинением сроков лечения и ростом числа осложнений. Также отсутствие лечебного эффекта подрывает доверие пациента к системе здравоохранения [7].

Согласно отечественным и зарубежным исследованиям, уровень приверженности при различных заболеваниях варьируется от 20 до 57%. Эти показатели относительно стабильны относительно разных нозологий [8–10].

Так, в статье, опубликованной в журнале «Педиатрическая фармакология», подчеркивается важность оценки и решения проблем приверженности к терапии у детей. Системный подход, учитывающий различные факторы, может помочь улучшить приверженность и, следовательно, результаты лечения [11].

Данное утверждение подтверждается исследованиями многих авторов современной литературы [4, 12–15].

Расширение возможностей для совместного использования приложений и инструментов для обмена сообщениями между врачами и пациентами позволяет улучшить взаимодействие между ними. Это взаимодействие является одним из ключевых для повышения приверженности к лечению [16–18].

Помимо повышения комплаенса, взаимодействие врача с пациентом с помощью приложений также может повысить качество жизни пациентов. Пациенты могут чувствовать себя более вовлеченными в процесс лечения, что может привести к улучшению результатов терапии. Кроме того, мобильные приложения могут обеспечить пациентам легкий и удобный способ доступа к информации о своем здоровье и лечении.

Современные приложения для поднятия приверженности лечения и комплаенса являются незаменимыми инструментами для пациентов, которым

необходимо принимать регулярно лекарства или следить за определенным режимом жизни. Однако, несмотря на их удобство и доступность, некоторые из них оставляют желать лучшего в плане их возможностей и удобства использования.

Одной из основных проблем современных приложений для поднятия приверженности является их недостаточная функциональность. Часто такие приложения предлагают ограниченный набор возможностей, таких как напоминания о приеме лекарств, учет принятых доз и возможность ведения дневника здоровья.

Так авторы исследования, опубликованного в журнале «Российский кардиологический журнал», указывают на ограниченность функционала современных приложений для отслеживания приема лекарств: только 38% приложений имеют функции отслеживания приема лекарств и статистики приверженности; комплексные функции (история терапии, связь с врачом, напоминания об истощении запасов лекарств и т. д.) встречаются редко. Недостаточное количество приложений с русским интерфейсом (53%). Указывается отсутствие специализированных приложений для различных заболеваний, кроме контроля приема оральных контрацептивов. Большинство приложений бесплатны (73%), но часто содержат рекламу (53%).

Несоблюдение медикаментозного лечения связано с увеличением использования медицинских услуг у детей и подростков, страдающих хроническими заболеваниями, и должно рассматриваться в рамках клинической помощи [19].

Оценена важность соблюдения терапевтических процедур для улучшения качества жизни не только у пациентов, но и их окружения, а также для эффективной работы системы здравоохранения. Исследование фокусируется на использовании мобильных приложений как стратегии повышения приверженности к терапии. Результаты показали, что интеграция мобильных приложений в клиническую практику может улучшить регулярность приема лекарственных препаратов [20]. Данный вывод нашел

свое отражение и в систематическом обзоре, проведенным в 2019 году, который показал, что использование компьютерных технологий помогает повысить приверженность лечению, и они являются подходящим методом для управления приемом лекарств в домашних условиях [21].

Многие авторы подтверждают, что мобильные приложения показывают хорошую эффективность в улучшении стабильности лечения у людей с хроническими заболеваниями. Так, в систематический обзор были включены четырнадцать исследований с участием 1785 участников, 940 из которых были рандомизированы в группу вмешательства с использованием мобильных приложений и 845 - в группу обычного ухода. Метаанализ показал, что использование компьютерных технологий было связано со значительным улучшением приверженности пациентов к лечению [22].

Университет Вирджинии (США) разработал специализированное мобильное приложение в сотрудничестве с пациентами. Приложение включает в себя ежедневные текстовые сообщения для самоконтроля настроения, стресса и приема лекарств, а также напоминания о визитах в туберкулезную больницу и центр СПИДа. Через приложение пациенты также могут просматривать результаты медицинских обследований (рентгеновские снимки, анализы на вирусную нагрузку и уровень CD4, микроскопия мокроты), записываться на прием и консультироваться со специалистами. Онлайн-сообщество, интегрированное в приложение, позволяет пациентам анонимно общаться и получать поддержку. Врач-координатор выступает в качестве связующего звена между пациентами и медицинским персоналом, обеспечивая ответы на вопросы и поддержку через приложение. Это приложение служит мостом между пациентами и медицинским персоналом, позволяя им эффективно взаимодействовать для улучшения результатов лечения. Исходя из результатов исследования был сделан вывод о том, что мобильные технологии оказались пригодны для сопровождения амбулаторного этапа ведения больных с туберкулезом и ВИЧ-инфекцией, его

прямое влияние отразилось только на формировании приверженности к АРВТ и снижении числа летальных исходов [23].

В 26 центрах вторичной медицинской помощи Португалии и Испании было проведено многоцентровое обсервационное исследование смешанным методом продолжительностью в 1 месяц. Во время первоначального личного визита врачи сообщили о плане лечения пациентов от астмы в структурированном вопроснике. Во время визитов пациентам было предложено ежедневно использовать приложение для регистрации приема лекарств от астмы. Запланированный прием считался принятым, когда пациенты регистрировали прием (ингалятор, блистер или другую лекарственную форму) с помощью инструмента определения приема лекарств на основе изображений.

Исследователи пришли к выводу о том, что использование компьютерных технологий оказалось осуществимым и приемлемым для мониторинга приверженности к лечению у пациентов с астмой [24].

Использование мобильных приложений оказывает влияние не только на приверженность к лечению, но и на течение и прогноз заболеваний.

В исследование, опубликованное в журнале «Journal of advanced nursing» в 2019 году, были включены 152 ребенка с астмой в возрасте от 6 до 11,9 лет были рандомизированы в экспериментальную группу (модель ведения астмы с мобильным приложением) или контрольную группу (модель ведения астмы под руководством медицинского работника без мобильного приложения). Все дети получали другие рутинные меры лечения, включая ингаляционные кортикостероиды.

Работа показала, что по сравнению с периодом до зачисления, частота обострений астмы снизилась в период после зачисления в обеих группах, с большим снижением в экспериментальной группе. По сравнению с детьми контрольной группы, у детей экспериментальной группы были лучшие вторичные результаты, такие как улучшение приверженности, более высокие баллы контрольных тестов на астму у детей, снижение количества инфекций

дыхательных путей, дней приема антибиотиков, дней отсутствия в школе, потери работы родителями и медицинских расходов [25]. Это исследование подчеркивает потенциал использования компьютерных и интернет-технологий в управлении хроническими заболеваниями.

В ином рандомизированном контролируемом исследовании оценивалось влияние мобильного приложения «Bant» у пациентов с сахарным диабетом 1 типа на уровень гликозилированного гемоглобина (HbA1c), а также на дополнительные показатели (например, самоконтроль уровня глюкозы в крови) для оценки влияния мобильных технологий на поведение подростков с сахарным диабетом 1 типа в целях самоконтроля.

Хотя первичный анализ клинических исходов не продемонстрировал различий между группой, использующей мобильное приложение и контрольными группами, последующий анализ показал, что компьютерные технологии могут положительно влиять на самоконтроль уровня глюкозы в крови среди молодежи. Также в подгруппе пользователей, использующих приложение «Bant» произошло значительное улучшение уровня гликозилировнного гемоглобина на 0,58% (P= 0,02), в то время как в параллельной подгруппе контрольной группы существенных изменений уровня гликозилировнного гемоглобина не произошло (снижение на 0,06%, P=0,84) [26].

В 2021 году исследователи из Бухарского государственного медицинского института провели анализ действенности мобильных приложений для увеличения приверженности к лекарственной терапии. Авторы статистически доказали, что мобильные приложения значительно улучшают приверженность пациентов лекарственной терапии, что подтверждается увеличением регулярности приема медикаментов [27].

Среди медицинских мобильных приложений для контроля за приемом лекарств популярностью у исследователей пользуются приложения с мониторингом глюкозы у пациентов, страдающих сахарным диабетом, которые своевременно должны принимать инсулин. Существует несколько

успешных реализаций таких систем, внедренных в различные процессы в больницах.

Например, опыт интеграции системы GlucoTab в больничные условия демонстрирует, что мобильные устройства могут улучшить мониторинг приема глюкозы у пациентов, позволяют вести записи о результатах обследований, а также оптимизируют рабочие процессы в медицинском учреждении, осуществляя поддержку в принятии решений по лечению. Система поддерживает несколько сценариев работы для необходимых задач в разных больницах [28].

Исходя из всего вышеперечисленного, можно сделать вывод о том, что мобильные приложения могут сыграть важную роль в решении одной из современных проблем здравоохранения. Такие приложения для повышения приверженности к лечению предоставляют пациентам доступ к персонализированной информации о их заболевании и лечении, что способствует лучшему пониманию и осознанию необходимости соблюдения медицинских рекомендаций. Важной функцией таких приложений являются напоминания о приеме лекарств, проведении процедур и других аспектах регулярного лечения. Это помогает пациентам не пропускать назначенные процедуры и соблюдать график приема лекарств, что в свою очередь способствует улучшению результатов лечения.

Эффективное внедрение мобильных приложений в систему ухода за пациентами может привести к значительному улучшению качества жизни больных, снижению медицинских расходов и улучшению результатов лечения. Пациенты, использующие такие приложения, чаще следуют рекомендациям врачей, что ведет к более стабильному состоянию здоровья и уменьшению частоты обострений хронических заболеваний. Это также может снизить необходимость в обращениях в службы экстренной помощи и госпитализации, что сокращает медицинские расходы.

Таким образом, использование мобильных приложений для повышения приверженности к лечению у пациентов с хроническими заболеваниями

является эффективным и перспективным подходом, который может значительно улучшить уход за пациентами и результаты лечения в целом.

1.3 Подходы к реализации аналогичных систем

В современной медицине мобильные приложения и устройства играют ключевую роль в улучшении качества лечения и управления здоровьем пациентов. Разработка и интеграция API для таких систем требует особого внимания к безопасности, совместимости и соответствию нормативным требованиям.

Одним из наиболее важных аспектов при разработке медицинских приложений является безопасность. Это основополагающий фактор, лежащий в основе приложений такого типа. Безопасность должна учитываться и рассматриваться в начале проектирования системы и существовать на каждом уровне взаимодействия между клиентом и сервером. Когда клиент и сервер обмениваются данными, важно использовать надежные методы шифрования для защиты информации от перехвата или несанкционированного доступа.

Зачастую в теме безопасности передачи данных рассматриваются реализации защищенных соединений, основанные на протоколах безопасности, таких как TLS (Transport Layer Security) и его предшественник SSL (Secure Sockets Layer), обеспечивающие шифрование данных во время их передачи между клиентом и сервером. Эти протоколы используют асимметричное шифрование для обмена ключами, что позволяет устанавливать защищённое соединение без необходимости предварительного обмена секретными ключами. В тесной связи с TLS/SSL идет использование протокола HTTPS (Hyper Text Transfer Protocol Secure), который гарантирует, что все данные, передаваемые между веб-браузером и веб-сайтом, зашифрованы, тем самым защищая данные от перехвата и манипуляций злоумышленниками. Этот протокол использует TLS и SSL, о которых говорилось выше. Использование HTTPS стало стандартом не только для

медицинских приложений, но и всех веб-приложений в целом, особенно тех, которые обрабатывают конфиденциальную информацию, такую как данные кредитных карт, личную информацию и медицинские данные [29].

Для обеспечения того, что только авторизованные пользователи имеют доступ к данным, используются различные способы аутентификации и авторизации:

– OAuth – стандарт авторизации, который позволяет пользователям предоставлять приложениям доступ к информации на других веб-сервисах без передачи паролей;

– JWT (JSON Web Tokens) – метод для безопасной передачи информации между двумя сторонами в виде JSON-объекта, который может быть зашифрован для обеспечения целостности.

Помимо передачи данных также нужно рассмотреть их хранение с использованием надежных методов шифрования. Хранящиеся записи должны быть зашифрованы на сервере, используя стандарты шифрования, такие как AES (Advanced Encryption Standard).

Системы также должны быть защищены от различных видов атак:

– SQL-инъекции – использование параметризованных запросов и ORM (Object-Relational Mapping) для защиты баз данных от инъекций;

– XSS (Cross-Site Scripting) – реализация политик Content Security Policy (CSP) для предотвращения XSS-атак;

– CSRF (Cross-Site Request Forgery) – использование токенов CSRF для защиты веб-приложений от поддельных межсайтовых запросов.

Эти меры представляют собой комплексный подход к обеспечению безопасности данных в клиент-серверных системах [30].

При разработке различного рода интеграций с медицинскими приложениями в основу взаимодействия между сервисами и клиентами используется REST (REpresentational State Transfer) API благодаря своей простоте, масштабируемости и легкости интеграции. Наиболее важные

аспекты, на которые нужно обращать особое внимание при его проектировании:

– определять четкие и логичные URL-адресов для ресурсов и использование соответствующих HTTP-методов (GET, POST, PUT, DELETE) для операций CRUD;

– создать единообразные и предсказуемые ответы API, включая коды состояния HTTP и форматы сообщений;

– заранее учесть версионирование API и продумать его систему для управления изменениями в API и обеспечения обратной совместимости.

Примеры использования REST API для предоставления медицинских резюме подчеркивают значимость стандартизации данных и интерфейсов [31].

2 Анализ средств и методов разработки

2.1 Общие требования к системе

Перед тем как перейти непосредственно к этапам проектирования и разработки backend системы нужно определить требования как к системе в целом, так и к отдельным её компонентам, а затем на этом основании сформировать частное техническое задание на серверную часть [33].

В приложении должна быть ролевая модель доступа, чтобы пользователи имели доступ только к разрешенным частям системы в соответствии с их ролями.

В системе реализовано 5 ролей:

– администратор – обладает возможностями по добавлению врачей в систему, а также имеет полный доступ к API;

– врач – возможности по приглашению пациентов в приложение, контролю их лечения и коммуникации с ними;

– пациент детского возраста – есть доступ к игровой части приложения, а также к работе с напоминаниями. Коммуникации этой роли с врачом невозможны, они осуществляются через родителя;

– пациент подросткового возраста – те же возможности, что и пациента детского возраста, но способен сам коммуницировать с врачом в чате;

– родитель пациента – основной задачей является наблюдение за лечением ребенка и взаимодействие в лечащим врачом при необходимости.

Это требует реализации авторизации и аутентификации, чтобы гарантировать, что пользователи имеют доступ только к тем данным и функциям, которые соответствуют их роли.

Для обеспечения защиты чувствительных данных приложение должно использовать механизмы шифрования для хранения пользовательской информации, таких как пароли, персональные данные и медицинская информация. Это поможет предотвратить несанкционированный доступ к

чувствительным данным и обеспечит их конфиденциальность. Помимо шифрованного хранения приложение не должно передавать по сети незашифрованные данные, общение должно производиться только по протоколу HTTPS.

Приложение должно обеспечивать целостность связей между пользователями. Для этого необходимо создать механизм, который гарантирует, что связи между пользователями остаются непрерывными и не подвержены случайным или злонамеренным изменениям. Это обеспечит точность и надежность данных, связанных с медицинскими отношениями между клиентами системы.

Для взаимодействия с мобильным приложением backend должен предоставлять API для необходимых логических компонентов – работа с регистрацией и аутентификацией пользователей, работа с игровыми данными, работа с напоминаниями о приеме лекарств, работа с пользовательскими коммуникациями. Ответ от API должен приходить в формате JSON (JavaScript Object Notation).

Основные бизнес-требования, которые сервис должен реализовывать:

– приглашение пациента – на приеме врач должен уметь генерировать код приглашения, которое затем передает пациенту. С полученным кодом пациент должен мочь зарегистрироваться в приложении и получить связь со своим лечащим врачом;

– после регистрации пациент должен уметь входить под своими учетными данными в свой аккаунт, то есть проходить процедуру аутентификации;

– на основе роли аутентифицированного пользователя ему должен быть доступен только определенный пул возможностей, прописанных в техническом задании, то есть в приложении должна работать авторизация;

– авторизованный пользователь должен уметь создавать и редактировать напоминания о приеме медикаментов, содержание модели

расписания приема и медикамента регулируется техническим заданием. Пользователь с ролью врача может также просматривать напоминания и график приема связанных с ним пациентов;

– в мобильном приложение будет реализовываться игровой режим, сервер должен поддерживать его консистентность и согласованность во времени, храня пользовательские игровые данные и проводя необходимые операции над ними по требованиям из технического задания и по согласованию с разработчиками мобильного приложения;

– наличие возможности пользовательской коммуникации в реальном времени через модуль чатов. Возможность начать чат должна регулироваться связями пользователей и опираться на техническое задание.

Отдельным блоком требований стоят технические требования, которые обеспечивают доступность, отказоустойчивость и прозрачность в работе системы. Приложение должно вести подробные журналы логирования и событий для разбора критических ситуаций и ошибок. Также должен происходить мониторинг системы в реальном времени для своевременного реагирования на неожидаемое поведение, например, ошибки при выполнении запросов или превышение квоты ресурсов, выделенных для приложения.

2.2 Технология разработки API

Выбор технологического стека играет ключевую роль в процессе планирования любого проекта, поскольку он определяет основу для разработки. Правильный выбор технологий и методов разработки способствует масштабируемости решения, увеличению скорости разработки и упрощению поддержки приложения в будущем. Этот выбор важен не только для разработчиков, но и для конечных пользователей, поскольку опыт взаимодействия с приложением, включая скорость работы, визуальное оформление и отзывчивость, непосредственно влияет на пользовательский опыт.

С точки зрения проектирования веб-интерфейса необходимо рассмотреть один из самых распространённых подходов в проектировании под названием REST API [34, 35]. REST — это аббревиатура от REpresentational State Transfer и архитектурный стиль для распределенных систем. Клиенты и серверы взаимодействуют через HTTP-запросы для выполнения стандартных функций базы данных, таких как создание, чтение, обновление и удаление записей (также известных как CRUD) в ресурсе. HTTP протокол является наиболее используемым, но REST не обязывает использование именно его. И самое главное отличие подхода в том, что каждое взаимодействие с сервером должно быть без сохранения состояния. Все это позволяет приложениям RESTful быть простыми, легкими и быстрыми.

ASP.NET Core — это кроссплатформенная высокопроизводительная платформа с открытым исходным кодом для создания современных облачных приложений, подключенных к Интернету. Одно из важных преимуществ платформы заключается в возможности создавать и запускать кроссплатформенные приложения, а также присутствует поддержка контейнеризации приложений в Docker.

ASP.NET Core оптимизирован для облака, так как требует мало памяти и обладает высокой пропускной способностью, что предоставляет возможность разместить большее количество инстансов на одном оборудовании, обслужить больше клиентов и платить за меньшее количество ресурсов при использовании услуг облачного хостинга.

2.3 Технологии доступа и хранения данных

Для работы с системой управления базой данных (СУБД) из сервиса могут использоваться разные технологии доступа к данным. А так как сервис написан на платформе .NET, то выбор технологии работы с СУБД сужается довольно сильно до нескольких самых популярных инструментов – ADO.NET и Entity Framework (EF) [50].

ADO.NET является более низкоуровневым инструментом доступа к данным, который предоставляет прямой доступ к базам данных с использованием классов и методов, таких как SqlConnection, SqlCommand и SqlDataReader. ADO.NET предоставляет гибкость и контроль над запросами к базе данных, но требует большего объема кода для выполнения простых операций в сравнении с EF.

Entity Framework (EF) – это ORM (Object-Relational Mapping), который предоставляет абстракцию над базой данных и позволяет работать с данными в виде объектов. EF упрощает операции с базами данных, а также обеспечивает уровень абстракции и переносимость кода между различными провайдерами баз данных (такими как SQL Server, Oracle, MySQL и др.).

Ключевым различием является сложность кода. ADO.NET является более низкоуровневым и требует больше кода для выполнения операций с базой данных, в то время как EF предоставляет простой способ выполнения запросов и операций с базой данных. Важным преимуществом ADO.NET является скорость выполнения кода. ADO.NET несомненно, быстрее, производительность выше, чем у Entity Framework [50].

Однако несмотря на различия в скорости для реализации в проекте было принято решение использовать Entity Framework. Этот фреймворк автоматически формирует запросы для выполнения CRUD-операций (создание, чтение, обновление, удаление) и работает с отношениями между таблицами на основе моделей данных, что на старте приложения значительно увеличивает скорость выполнения задач, а в долгосрочной перспективе упрощает поддержку кода и архитектура работы с СУБД. В ADO.NET необходимо ручное написание кода для отображения данных.

Ещё одним важным преимуществом Entity Framework стала почти безболезненная переносимость данных между использованиями разных СУБД, что облегчает поддержку разных систем управления базами данных. В определенный момент времени это помогло довольно просто сменить СУБД SQLite на Postgres. В ADO.NET в свою очередь требуется более прямая работа

с конкретным провайдером базы данных, что повлекло бы тотальное изменение кода.

Реляционные базы данных становятся все больше и сложнее. Кроме того, современные системы управления базами данных должны эффективно реагировать на операции с их данными. В этом контексте оптимизация базы данных очевидна как процесс совершенствования систем баз данных, направленный на повышение их пропускной способности и производительности.

PostgreSQL широко используется в индустрии и считается стандартом во многих компаниях благодаря своей надежности, масштабируемости и возможностям для разработки.

Так, по результатам исследований, где сравнивалась производительность двух систем управления реляционными базами данных PostgreSQL и MySQL [36], в случае операций добавления, обновления и удаления данных из БД PostgreSQL оказалась эффективнее почти в два раза. Подтверждается это и сравнением с СУБД MongoDB [37], где в статье измерялась производительность с точки зрения времени отклика в кластере из 5 узлов. Результаты показали, что PostgreSQL превосходит MongoDB почти во всех случаях. Кроме того, среднее время отклика значительно сокращается при использовании индексов в случае MongoDB со значительно меньшим положительным эффектом в PostgreSQL. Наконец, размер данных в СУБД PostgreSQL в 4 раза меньше.

Другие источники [38] показывают сравнение производительности систем баз данных Oracle и PostgreSQL с бенчмарком TPC-H, следуя стратегии добавления индексов на основе столбцов для оптимизации выполнения запросов. Десять запросов TPC-H выполняются к таблицам без каких-либо ограничений, с первичными и внешними ключами и с ограничениями индекса. Анализируется производительность в каждом наборе выполнений. Результаты позволяют сделать вывод о положительном эффекте при использовании ограничений со значительным ускорением, а также лучшей пропускной

способностью. Oracle продемонстрировала стабильность при выполнении запросов с наилучшими результатами в сценариях с низкой нагрузкой. Однако PostgreSQL показал более короткое время выполнения после проведенной оптимизации. Глобальные результаты производительности показывают, что Oracle может повысить производительность на 7% с помощью индексов, а PostgreSQL - на 91%.

Таким образом, PostgreSQL действительно заслуживает свою репутацию надежной и масштабируемой системы управления реляционными базами данных, и его широкое использование в индустрии подтверждает его статус стандарта во многих компаниях.

2.4 Технологии доставки и управления приложениями

Во время разработки API важно вовремя задумываться о том, как разработчики, создающие клиент к API, будут своевременно получать обновления сервера. Важно не упустить этот момент, чтобы разработка не затянулась из-за долгой доставки изменений сервера до всех разработчиков команды. Именно в этот момент нужно произвести интеграцию CI/CD – Continuous Integration/Continuous Delivery в пайплайн доставки изменений. Необходимо это по нескольким причинам, основные из которых будут описаны ниже [39].

Одним из главных преимуществ выступает быстрая обратная связь и выявление проблем после развертывания. CI/CD позволяет автоматизировать процессы сборки, тестирования и развертывания кода. Как только разработчик отправляет изменения в репозиторий, CI/CD пайплайн автоматически запускается, компилирует код, выполняет тесты и развертывает приложение в среду, доступную всей команде. Если в процессе возникают ошибки, то можно мгновенно получить обратную связь и быстро исправить проблему. Это помогает избежать накопления ошибок и обеспечивает высокую стабильность и надежность разрабатываемого приложения.

Не менее важным преимуществом является улучшение качества кода. CI предлагает различные этапы тестирования, включая модульные, интеграционные и функциональные тесты. Это позволяет оптимизировать и проверить свой код на ранних этапах разработки до интеграции с удаленным сервером. Автоматическое тестирование также помогает избежать проблем совместимости между различными компонентами системы и повышает качество и стабильность разрабатываемого продукта.

Конвейер CI/CD также позволяет использовать итеративный подход и непрерывное улучшение, где изменения в коде внедряются и тестируются регулярно. Это позволяет быстро реагировать на обратную связь разработчиков и пользователей, вносить коррективы и проводить улучшения.

В целом, внедрение CI/CD позволяет ускорить и автоматизировать разработку программного обеспечения, улучшить качество и стабильность разрабатываемого API, а также снизить время доставки изменений в системе, созданных в результате выполнения задач, до облачного сервера.

Для интеграции с возможностями CI/CD используется GitHub Actions [41]. Это мощный набор инструментов автоматизации и непрерывной интеграции, предоставляемый GitHub. Этот сервис предоставляет возможность эффективно автоматизировать и контролировать различные процессы в рабочем цикле разработки ПО [42].

Одним из важнейших преимуществ GitHub Actions является его простота использования. Другое важное преимущество GitHub Actions – это его гибкость. Сервис позволяет создавать и настраивать собственные рабочие процессы, а также использовать готовые шаблоны, предоставляемые сообществом и репозиторием GitHub. Это существенно ускоряет настройку и облегчает масштабирование автоматизации, позволяя командам разработчиков сосредоточиться на самом коде. Дополнительным преимуществом является интеграция с другими сервисами и экосистемой GitHub, так как хранение кода также осуществляется в этом же сервисе.

Для развертывания приложения выбрана система контейнеризации Docker [49]. Docker является инструментом с открытым исходным кодом, который облегчает создание, доставку и выполнение приложений путем их упаковки в контейнеры. Эти контейнеры действуют как легковесные, автономные пакеты, содержащие все необходимое для работы приложения, что позволяет им работать одинаково в любой среде. Такой подход отделяет приложение от окружающей его инфраструктуры, что ускоряет процессы разработки, тестирования и развертывания.

Контейнеризация с Docker обеспечивает изоляцию и безопасность, позволяя запускать множество контейнеров на одном и том же хосте без конфликтов. Это достигается за счет того, что каждый контейнер работает в изолированной среде и имеет свою копию виртуализированных аппаратных ресурсов.

Виртуализация, с другой стороны, включает в себя создание виртуальной версии аппаратных компонентов, что позволяет запускать приложения, предназначенные для одной аппаратной платформы, на другой. Это обычно управляется операционной системой хоста, которая имитирует аппаратные ресурсы для гостевых систем.

2.5 Выбор архитектуры системы

Целевой архитектурой системы на первом этапе развития проекта выбрана монолитная архитектура. На данный момент в системе относительно немного компонентов:

– компонент аутентификации и авторизации – позволяет пользователям получать права к различным возможностям, а также обеспечивает возможности входа в аккаунт через их идентификационные данные;

– компонент работы с пользователями – отвечает за приглашение пользователей в систему и формирование и поддержку корректных связей между ними;

– компонент общения в чатах – обеспечивает коммуникации пользователей в чатах в реальном времени;

– компонент по работе с игровыми данными – предоставляет возможности по хранению, получению и обработке различных игровых пользовательских данных, который работает, когда пользователь взаимодействует с игровой частью мобильного приложения.

– компонент по работе с напоминаниями – отвечает за работу со всеми видами напоминаний. Через него пациенты создают и редактируют напоминания о приеме лекарств, а врачи могут контролировать прием лекарств пациентов.

Сами компоненты часто изменяются ввиду поступления новых требований, так как проект находится в стадии развития. Поэтому при принятии решения об использовании монолитной архитектуры учитывались следующие преимущества монолитной архитектуры:

– простота развертывания – монолитное приложение состоит из одного самодостаточного блока, что упрощает процесс развертывания, поскольку требуется развернуть только один сервис;

– упрощение разработки и тестирования – можно работать с одним приложением целиком в одном .sln файле, что упрощает понимание кода и тестирование функциональности;

– бонусом стала производительность – монолитное приложение немного более оптимизировано за счёт отсутствия сетевых вызовов между различными сервисами, что особенно важно для операций, требующих интенсивного взаимодействия между компонентами.

Альтернативные архитектурные стили:

– микросервисная архитектура – состоит из набора независимых сервисов, каждый из которых выполняет определённую функцию. Каждый сервис полностью изолирован друг от друга. Это обеспечивает лёгкость масштабирования и обновления, но может усложнить развертывание и

управление;

– serverless-архитектура – при таком подходе разработчики могут сосредоточиться на написании кода, в то время как инфраструктура управляется облачным провайдером. Это упрощает операции, но может привести к проблемам с производительностью и забирает контроль над важными частями инфраструктуры;

– событийно-ориентированная архитектура – этот стиль подразумевает реакцию на события в системе, что может обеспечить высокую отзывчивость и гибкость, но требует сложной координации между компонентами.

Несмотря на текущий выбор монолитной архитектуры, код написан таким образом, что при достаточном развитии проекта, когда потребуется выдерживать более высокие нагрузки или появятся другие требования, например, не независимому развертыванию, архитектура будет пересмотрена в сторону микросервисов.

На текущий момент монолитная архитектура позволяет более быстро доводить задачи до пользователей за счет того, что весь код собран в одном приложении, которое собирается и развертывается самостоятельно. Однако в в долгосрочном периоде это может начать приносить больше неудобств, чем пользы, из-за тех же причин.

Микросервисы обеспечивают гибкость и масштабируемость, поскольку различные компоненты могут быть развернуты и масштабированы независимо друг от друга в зависимости от их потребностей. Это также облегчает поддержку приложения, поскольку каждый сервис может быть модифицирован и выложен отдельно. Кроме того, микросервисная архитектура позволяет использовать различные технологии и языки программирования для каждого сервиса, что способствует выбору наиболее подходящих инструментов для решения конкретных задач. Этот подход также способствует повышению отказоустойчивости.

2.6 Управление проектом

Заявленный проект, в рамках которого ведется разработка API, является комплексным решением для сопровождения пациентов, предоставляя широкий спектр возможностей, поэтому для его разработки нужно несколько ролей, каждая из которых будет заниматься своей частью функциональности. В данном случае над проектом работала команда из 6 человек: 3 мобильных разработчика, 1 бэкенд разработчик, 1 аналитик и 1 дизайнер для приложения.

Разработка проекта происходит по методологии Scrum, что позволяет быстро и эффективно реагировать на меняющиеся требования и проверять новые идеи, которых на начальном этапе развития такого проекта возникает достаточно много. Scrum выбрана из-за её гибкости и ориентированности на командную работу. Согласно Scrum происходит разбиение больших и сложных задач на маленькие и управляемые части, которые мы организуем в спринты. Каждый спринт длится от одной до четырёх недель и включает в себя планирование, разработку, тестирование и демонстрацию результатов.

Для управления задачами и спринтами используется YouTrack. Эта система позволяет не только отслеживать прогресс выполнения задач, но и анализировать рабочую нагрузку, определять приоритеты и управлять ресурсами. YouTrack интегрирован со всеми рабочими процессами, что делает процесс планирования и отслеживания работы максимально прозрачным и эффективным.

2.6.1 Оценка рисков

Анализ рисков проекта является ключевым элементом управления проектами, поскольку он помогает идентифицировать, оценить и приоритизировать потенциальные проблемы, которые могут повлиять на успешное выполнение проекта. В данной работе будут рассмотрены риски, непосредственно относящиеся к реализации серверной части проекта.

Риски проекта могут относиться к различным категориям:

– стратегические риски – связаны с решениями управления и бизнес-стратегией. Например, неправильное позиционирование на рынке или неверный выбор бизнес-модели;

– операционные риски – относятся к внутренним процессам, людям и системам или внешним событиям. Это может быть ошибка сотрудника или сбой в системе;

– финансовые риски – включают риски, связанные с финансовыми аспектами деятельности, такие как кредитные риски, рыночные риски, риски ликвидности и риски процентных ставок;

– репутационные риски – потенциальный ущерб для репутации или бренда компании, который может привести к потере клиентов или дохода;

– юридические риски – связаны с возможностью возникновения судебных исков, штрафов или нарушений законодательства;

– риски со стороны персонала – включают риски, связанные с управлением персоналом, такие как удержание ключевых сотрудников, трудовые споры или недостаток квалифицированных кадров;

– технологические риски – риски, связанные с использованием или зависимостью от технологий, включая устаревание технологий или кибератаки;

– риски внешней среды – включают риски, связанные с политическими, экономическими, социальными и природными факторами, которые могут повлиять на деятельность организации.

Проанализированные риски проекта представлены в таблице 1.

Таблица 1 – Риски реализации серверной части проекта

Риск	Оценка рисков	Планирование реагирования	Реагирование
Выбор ошибочной гипотезы Вероятность: средняя Критичность: высокая	Полное полагание на гипотезу, предложенную заказчиком проекта без собственного анализа	Анализ гипотез и чтение авторитетных источников на эту тему	Приостановка разработки до смены гипотезы на подходящую, перекройка приложения на этапе

			разработки как можно безболезненнее (избегать)
Bus-фактор, зависимость от одного разработчика Вероятность: низкая Критичность: средняя	Отсутствие связи между разработчиками, возможности делегирования задач	Ревью, ведение матрицы компетенций, расширение контекста знаний проекта	Откатиться до последней стабильной версии (избегать)
Опоздание по срокам Вероятность: средняя Критичность: средняя	Параллельная загруженность членов команды основной работой/учёбой	Выделение времени на каждую задачу с запасом	Увеличение кол-ва рабочих часов/уменьшение итогового релизного потенциала (контролировать)
Проблемы с распространением приложения и востребованностью Вероятность: низкая Критичность: средняя	Приложение, недостаточно подходящее для врачей или неинтересное для пользователей	Сбор требований врачебного состава, необходимых для распространения приложения среди ЦА, повторный анализ требований и ЦА	Выяснение и устранение причины отказа врачей (избегать)
Проектирование на смешанной архитектуре Вероятность: средняя Критичность: низкая	Желание успеть в срок и реализовать наибольшее кол-во функционала, грозит сильной привязкой на одну платформу или архитектуру	Определение допустимых в коде временных мер, которые не окажут впоследствии ощутимого влияние на качество кода	Постепенная смена “костылей” на качественный код в процессе дальнейшей разработки и поддержания приложения (принять)
Недоступность инфраструктуры разработки проекта Вероятность: низкая Критичность: средняя	Падение/недоступность хостинга.	Развертывание системы мониторинга, алертов. Нагрузочное и интеграционное тестирование	Отправка информационного сообщения отдельным сервисом (контролировать)
Отсутствие/недостаточность тестирования Вероятность: средняя Критичность: высокая	Ограниченные сроки работ, отсутствие отдельной роли тестировщика в команде	Покрытие кода тестами. Перекладывание роли тестировщика на аналитика в его свободное время или отдача тестирования на аутсорс	Исправление бага и отправка обновлений пользователям (с оповещением)
Необходимость в ресурсах для обеспечения проекта (например оплаты сервера, ремонт) Вероятность: низкая Критичность: низкая	Возникает в случае увеличения затрат, некомфортного для удовлетворения членами команды	Вести медийное освещение проекта, показывать социальную значимость, чтобы собирать деньги с пожертвований и/или краундфандинга	Взять деньги из собранного фонда (контролировать)
Принятие новых норм, законов Вероятность: средняя Критичность: высокая	Принятие государством новых законов, обязывающих разработчиков медицинских	Контроль принятия новых законов, затрагивающих разработчиков медицинских	Внедрение изменений в существующее решение. Обновление приложения у пользователей с

	приложение к каким-либо действиям	приложений; взять у заказчика контакты профильного специалиста	уведомлением. (Контролировать)
Нарушение авторских прав на элементы приложения Вероятность: низкая Критичность: высокая	Элементы дизайна, саундтрек могут быть использованы готовые.	Проверять лицензии, Отдавать создание ресурсов профильным специалистам	Откатить релиз с проблемными ресурсами. (Избегать)

2.6.2 Оценка экономических показателей

Экономический анализ проекта — это фундаментальный инструмент, который позволяет оценить его потенциал и риски. Он необходим для определения финансовой жизнеспособности проекта, предоставляя данные для принятия взвешенных инвестиционных решений. Проведение экономического анализа критично для управления рисками, поскольку он выявляет потенциальные проблемы до того, как они станут угрозой для проекта.

В контексте текущего проекта – медицинского приложения – возможны следующие способы получения прибыли;

– платные консультации – пользователи оплачивают консультации врачей через приложение, создавая тем самым доход для компании;

– торговля медицинскими товарами – в приложении может функционировать платформа для продажи медицинских изделий и лекарственных препаратов, принося доходы от продаж;

– рекламные доходы – в приложение встраивается реклама партнеров, генерируя доход за просмотры и клики;

– партнерские программы – сотрудничество с другими компаниями через различные партнерские соглашения и активности.

Сроки реализации проекта на данный момент рассматриваются в 10 месяцев.

Если рассматривать затраты на серверную часть проекта, то в их основе будет лежать технологическая инфраструктура. Одной из базовых затрат являются арендованные серверы для развертывания API и базы данных, за их поддержку нужно платить около 750 рублей в месяц. В эту же категорию включаются сервисы для инфраструктуры разработки – IDE (Integrated Development Environment), распространяемая по системе подписки – 12 000 рублей в год. За безопасность придется заплатить в год от 6 000 до 8 000 рублей – это покупка домена и SSL сертификата для него, с учетом сроков проекта оформляется сразу на 12 месяцев. К постоянным затратам так же относятся расходы на оплату интернета и электричества, в сумме в среднем получается около 750 рублей в месяц. Существуют так же единоразовые расходы – публикация патента на систему, которые составят 7 800 рублей. Без учета зарплаты backend-разработчика за срок реализации проекта выходит 42 000 рублей. Аналитика по затратам представлена в таблице 2.

Таблица 2 – Анализ затрат на проект

Затрата	Сумма, руб / мес	Кол-во месяцев	Итого, руб
Интернет	350	10	3 500
Аренда сервера	750	10	7 500
Электричество	400	10	4 000
Публикация патента	7 800	1	7 800
ПО для работы. IDE JetBrains Rider	1 200	10	12 000
Регистрация домена	200	12	2 400
Покупка сертификата	400	12	4 800
Итого			42 000

2.6.3 Ведение документации и технического задания

Ведение документации в проекте служит основой для обеспечения прозрачности, последовательности и отчетности на всех этапах проекта. В ней зафиксированы все аспекты проектного планирования, выполнения и оценки. Она предоставляет четкую запись целей проекта, задач, сроков и бюджетов.

Это также важный инструмент для новых членов команды, которым необходимо быстро ознакомиться с проектом. На долгой дистанции реализации проекта документация становится особенно важной.

В команде для хранения документации и ТЗ используется YouTrack – тот же инструмент, что используется для отслеживания задач и формирования спринтов.

В большинстве процессов проекта задача в первоначальном необработанном виде сначала попадает аналитику, который формирует конкретные требования и базовое видение её в складывающейся системе, документирует решение в базе знаний команды, а затем передает задачу на уровень дальше – дизайнеру и frontend-разработчику или backend-разработчику. Однако в развивающемся проекте у одного аналитика может не быть времени взять на себя все разрабатываемые задачи, поэтому зачастую разработчики берут на себя эту ответственность и проводят аналитику самостоятельно, формируя ТЗ, которое потом передают на проверку аналитику. Один из таких примеров изображен на рисунке 1, где ТЗ на общение в чатах формировалось разработчиками исходя из пожеланий заказчика.

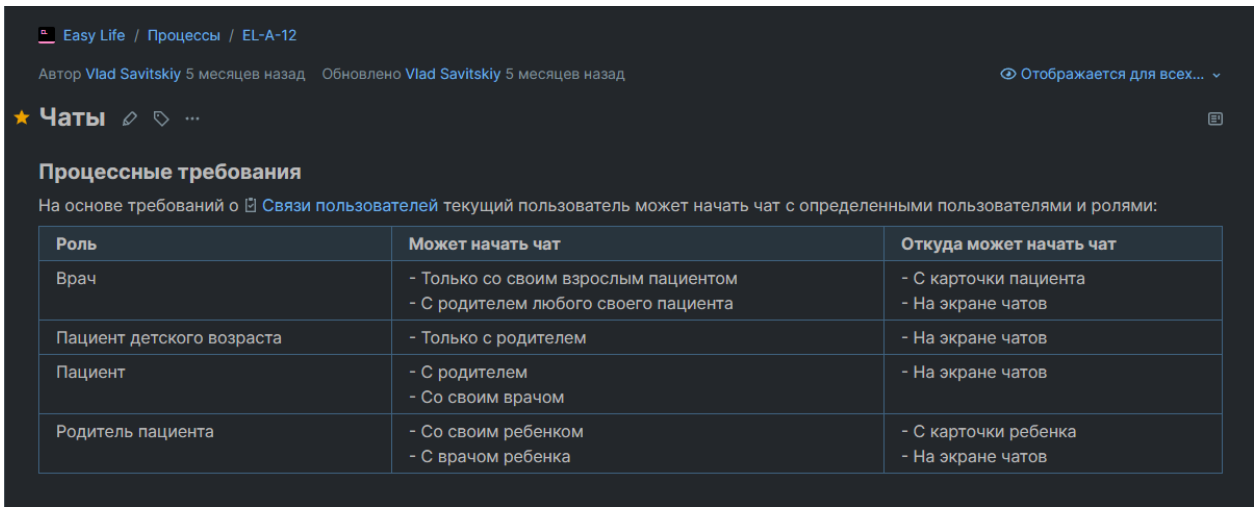

Рисунок 1 – ТЗ для реализации чатов в приложении

Помимо ТЗ на выполняемые задачи в управлении проектом также документировались реализованные технические решения и рекомендации по тестированию этих решений. Такой подход обеспечивает централизованное хранилище знаний, которое может быть использовано для справки, обучения и передачи информации. Когда разработчики документируют свои решения, они создают подробную карту того, что было сделано и почему, что помогает поддерживать последовательность в коде и архитектуре системы.

Кроме разработки важно также подумать о тестировании реализованного функционала, например, о тестировании API. Не всегда эти процессы разработки могут быть удобными и простыми для понимания. В такой ситуации особенно полезно оставить инструкцию по тестированию и разработке, чтобы оптимизировать рабочие процессы, сократив количество вопросов и обсуждений. Пример такой инструкции показан на рисунке 2.

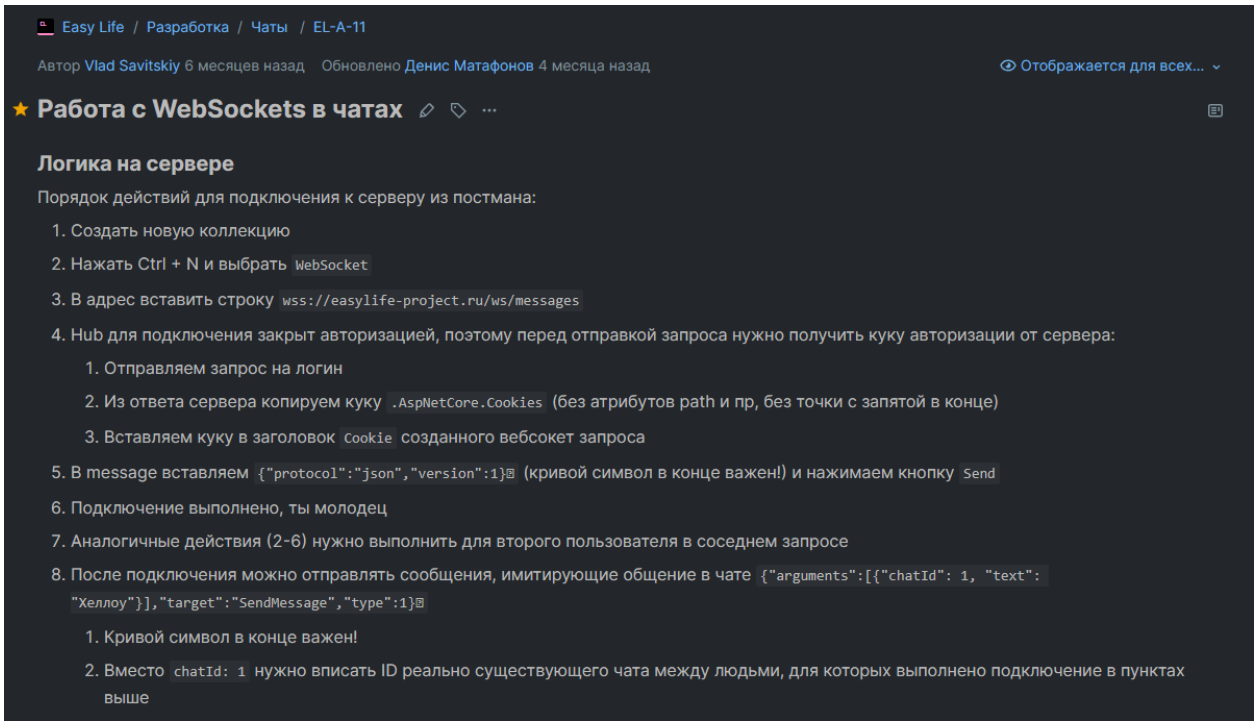

Рисунок 2 – Инструкция по работе с тестированием WebSockets в чатах

3 Результаты разработки системы

3.1 Проектирование системы

Серверное приложение играет ключевую роль в обеспечении взаимодействия между клиентской частью приложения и базой данных, обеспечивая высокую производительность, масштабируемость и безопасность работы приложения. Приложение будет обрабатывать множество запросов от клиентов, и неправильная разработка может привести к сбою в работе, недоступности сервисов и потере клиентов. Приложение должно обеспечивать высокую отказоустойчивость и минимальное время простоя. Неправильная разработка может также привести к утечке данных, взлому системы или использованию приложения для злонамеренных целей. Поэтому особое внимание уделяется построению безопасной архитектуры, контролю доступа, шифрованию данных и другим механизмам защиты.

Вся система разделена на пять основных компонентов, представленных на рисунке 3:

– мобильное приложение;

– основной сервис с API для работы мобильного приложения;

– компонент для аутентификации и авторизации;

– компонент для общения в чатах мобильного приложения;

– база данных.

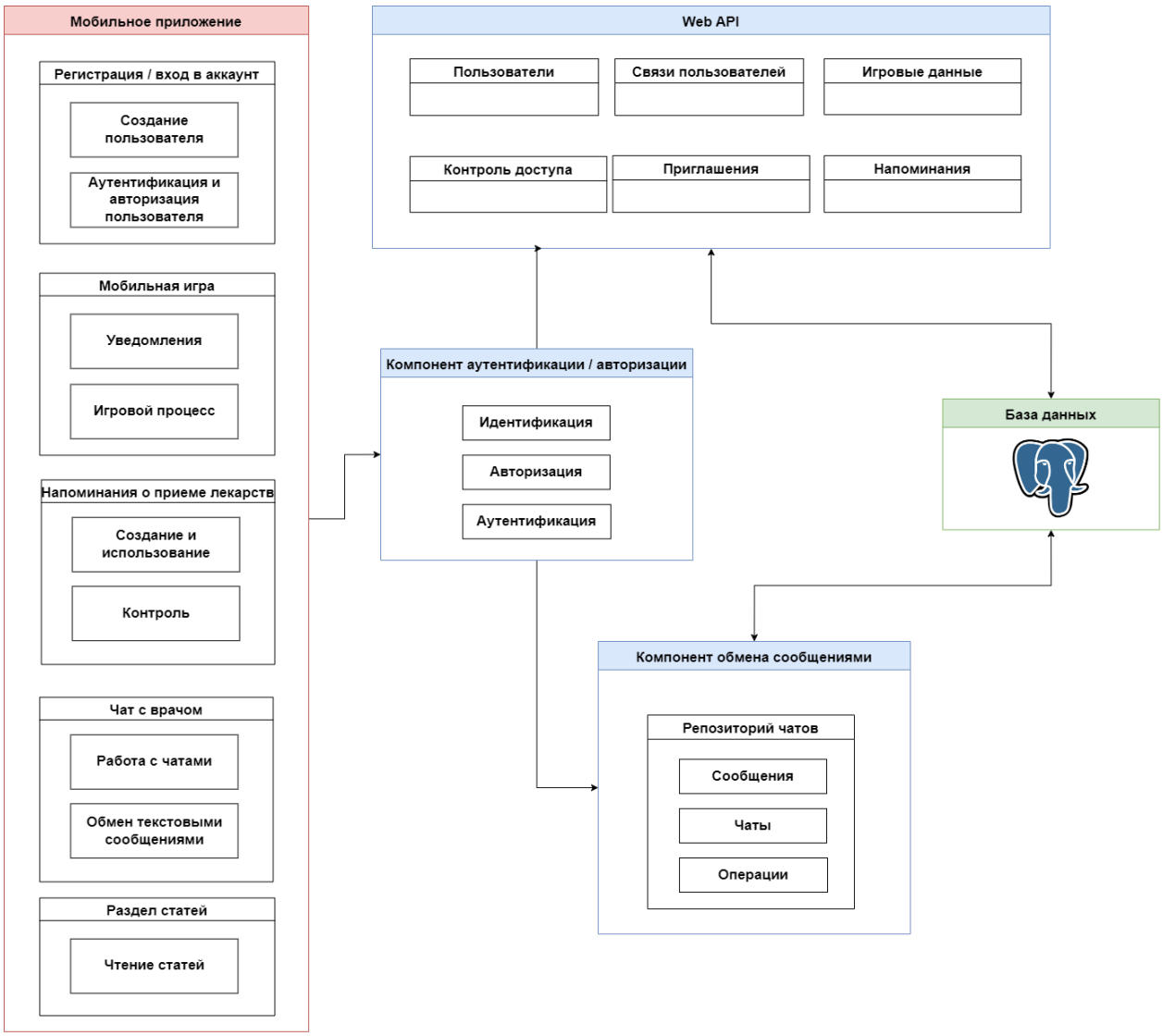

Рисунок 3 – Общая архитектура системы

Мобильное приложение выполняет роль непосредственного взаимодействия с пользователем через интерфейс. Оно включает в себя образовательную, игровую и коммуникативную части. Образовательная часть представлена разделом статей о течении заболевания, мерах его лечения и других полезных материалах – через всё это пациент сможет получать актуальную информацию о своем здоровье и быть более информированным. Коммуникативная часть выражается в возможности непосредственно из приложения общаться с лечащими врачами, обмениваться сообщениями и файлами. Игровая часть – набор мини-игр и механик, выстроенных вокруг

этих игр, направленных на повышение мотивации возвращаться в приложение и регулярно принимать терапию.

Web-сервис поделен на несколько крупных компонентов, имеющих собственные ответственности. Как обсуждалось ранее, в текущей архитектуре – это монолитное приложение, но компоненты разделены таким образом, что вынесение в отдельные сервисы не доставит больших сложностей. На данный момент представлены следующие компоненты:

– компонент аутентификации и авторизации пользователей – это пропускной пункт для всех запросов, попадающих в API. На этом этапе проверяется валидность аутентификации пользователя и доступ к тому ресурсу, на который совершается запрос. Если данные валидны, запрос пропускается далее;

– основной функционал сервиса – действия над пользователями, напоминаниями, игровыми, системными и другими данными;

– компонент, предоставляющий возможность общения в чате в реальном времени.

База данных выполняет роль хранилища для различных системных, пользовательских и игровых данных, необходимых для функционирования различных частей всей системы.

3.2 Настройка хоста под публикацию API

Для того, чтобы к приложению можно было совершать запросы в интернете, оно должно быть развернуто на сервере с выделенным IP адресом, который будет являться уникальным идентификатором сервера в web-пространстве.

На платформе TimeWeb был арендован облачный сервер с конфигурацией: 1 процессорное ядро, 1 ГБ оперативной памяти, SSD накопитель на 15 Гб для хранения файлов, представленный на рисунке 4. Операционная система сервера – Ubuntu 20.04 LTS.

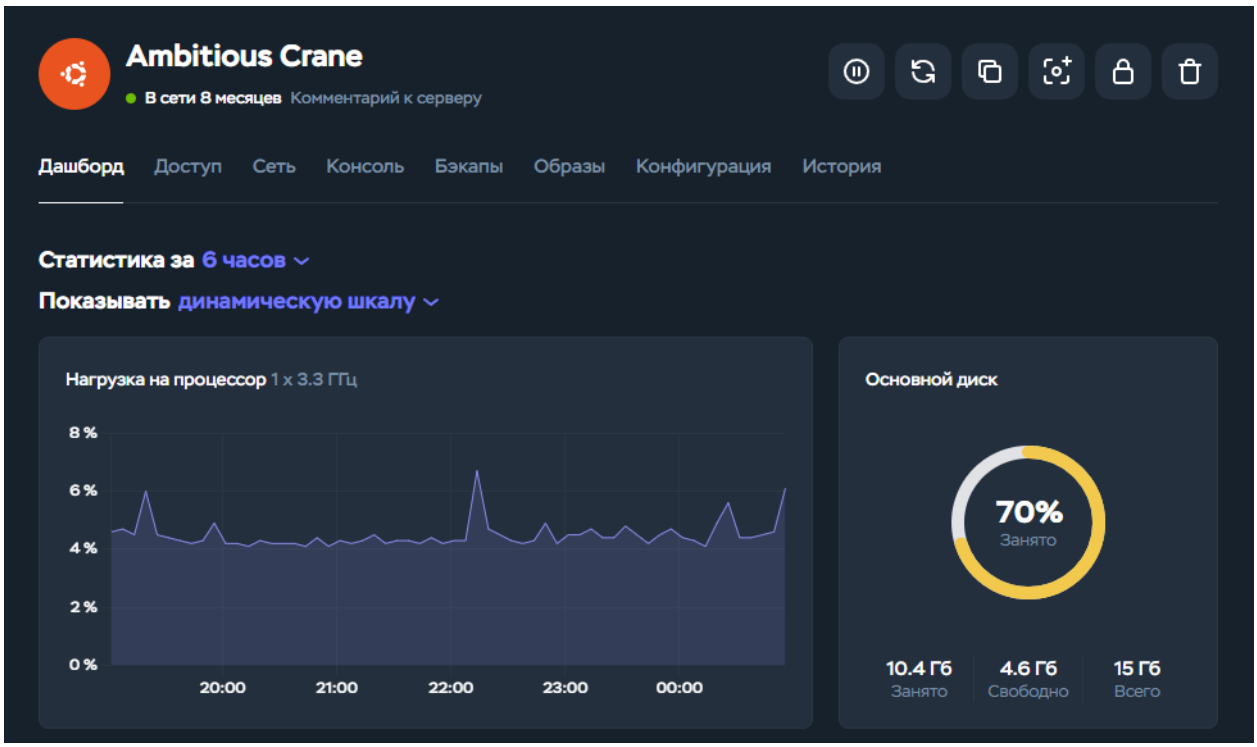

Рисунок 4 – Арендованный облачный сервер для API

После получения доступа к серверу необходимо было настроить его для правильной работы. Первым делом нужно позаботиться о его безопасности и максимально ограничить возможность входа на сервер посторонними лицами. Для этого был сгенерирован SSH-ключ, публичная часть которого помещена на сервер, а возможность аутентификации по паре логин-пароль – отключена. В результате манипуляций подключиться к серверу можно только с персонального компьютера разработчика.

Далее на сервер были установлены зависимости для работы контейнеризации Docker - системы, в рамках которой будут развернуты инстансы приложения с API; веб-сервер Nginx – для балансировки запросов до контейнеров.

Как говорилось ранее в анализе подходов к реализации похожих приложений, очень важно обеспечивать безопасную коммуникацию компонентов в системе, особенно когда дело касается персональной информации пользователей. Исходя из этого был куплен домен и настроен SSL сертификата для него, который обеспечивает следующие преимущества:

– защита информации от перехвата при коммуникации клиента и сервера, предотвращая перехват и изменение данных в процессе передачи между ними;

– подтверждение подлинности веб-сайта и защищает пользователей от фишинговых атак;

– при наличии web-представлений повышение доверия пользователей и улучшение показов сайта в поисковой выдаче.

Домен easylife-project.ru был приобретен у провайдера доменов рег.ру вместе с SSL сертификатом для этого домена.

После подготовки необходимого технического обеспечения можно приступить к настройке роутинга и балансировки. Главные параметры, которые требуется настроить – это проксирование запросов до контейнера с API, расположенном на 5000 порту IP адреса 0.0.0.0, то есть локального IP, а также правильный путь до SSL сертификата подлинности домена, который был загружен на облачный сервер.

```
server {
  listen 80;
  listen 443 ssl http2;
  server_name easylife-project.ru;
  ssl_protocols TLSv1 TLSv1.1 TLSv1.2;
  ssl_certificate      /rsa/easylife/sertchain.crt;
  ssl_certificate_key  /rsa/easylife/private.key;

  location /api {
    proxy_pass           http://0.0.0.0:5000;
    ...
  }
```

Помимо данного облачного сервера был арендован также второй сервер меньшей мощности для развертывания на нем контейнера с СУБД PostgreSQL. На сервере создана БД для проекта.

После описанных действий имеется готовая безопасная инфраструктура сервера для публикации приложения с API, имеющая защищенный доступ в интернет.

3.3 Реализация CI/CD

Ещё один важный этап перед непосредственной разработкой API – подготовить инфраструктуру для развертывания приложения на облачном сервере, чтобы оптимизировать процессы как можно раньше, ведь в совокупности все это скажется на времени разработки проекта в целом.

Для интеграции с GitHub Actions в корне репозитория нужно создать папку .github/workflows, она содержит в себе .yml файлы для запуска различных заданий к выполнению.

Пайплайн развертывания разделен на 5 основных этапов:

– тестирование решения;

– сборка решения;

– сборка и применение миграций;

– развертывание приложения;

– мониторинг приложения.

Этап тестирования – стандартный этап любого конвейера развертывания. Нужно убедиться, что внесенные изменения не сломали отдельные логические части системы или её целиком. По итогам прогона тестов генерируется отчет, который можно удобно посмотреть, используя интерфейс GitHub. Код процесса, выполняющего прогон тестов в конвейере, представлен ниже.

```
jobs:
  build:
    runs-on: ubuntu-latest
    steps:
    - uses: actions/checkout@v3
    - name: Setup .NET
      uses: actions/setup-dotnet@v3
      with:
        dotnet-version: 8.0.x
    - name: Run tests
      run: dotnet test --verbosity normal -c Release
```

После успешного прохождения этапа тестирования конвейер переходит к сборке и применению миграций. Сначала генерируется sql скрипт для последующего исполнения, а затем с помощью утилиты командной строки psql происходит его применение на базе данных. Таким образом локальные изменения схемы попадают в развернутую в облаке БД.

Основная часть конвейера – это сборка и разворачивание самого приложения. Первым этапом происходит сборка исходного кода в исполняемый файл с наоборот dll библиотек, от которых зависит этот файл. Весь полученный материал собирается в контейнер и загружается в хранилище контейнеров – Docker Registry. Затем в финальном этапе развертывания старый контейнер с приложением останавливается, новый контейнер выгружается из хранилища и запускается вместо предыдущей версии.

Завершающим шагом происходит контроль развернутого приложения, скрипт пытается пройти по конечным точкам API, чтобы проверить возвращаемые ответы. Если код ответа – ошибочный, то разворачивание прекращается с уведомлением разработчика о произошедшей ошибке.

Таким образом происходит выкладывание новой версии приложения в облако для получения обновлений другими разработчиками. Весь процесс

занимает не более 2 минут.

3.4 Реализация API

3.4.1 Регистрация, аутентификация и авторизация

Для облегчения разработки и более четкого следования требованиям было принято решение сразу реализовать модель доступа к отдельным частям API согласно требованиям. В основу легла ролевая модель Role-based access control (RBAC) – был создан определенный пул ролей, у каждой из которых появлялся доступ к тем или иным ресурсам. Дальнейшая разработка и тестирование проходило уже с учетом выдаваемых прав.

Для реализации механизма выдачи прав использовалась связка созданной в БД таблицы с возможным набором ролей для системы, а также встроенный механизм авторизации в ASP.NET Core.

```
[ApiController]
[Authorize(Roles = Roles.DOCTOR_ID)]
[Route("invites")]
public class InvitesController : Controller
{ ... }
```

После реализации ролевой модели логично перейти непосредственно к созданию пользователей, которые и имели бы эти принадлежность к этим ролям. В API были добавлены методы для регистрации и аутентификации пользователей по их идентификационным данным – пара логин и пароль. На этом этапе приложение получило возможность создавать пользователей и показывать им те или иные ресурсы в зависимости от их ролей.

На этом этапе важно было позаботиться о корректной генерации связей между пользователями: на текущий момент нельзя, чтобы один пациент смог написать другому пациенту или один родитель связаться с другим родителем,

если у них нет прямой связи через лечащего доктора. Поэтому была спроектирована система связей основной – зависимый пользователь, представленная в таблице 3.

Таблица 3 – Требования по генерации связей между пользователями

Роль	Связи	Направленность
Пациент детского возраста	1. С врачом, который выдал код	Зависимый
	2. С родителем, который делит код с ребенком	Зависимый
Пациент	1. С врачом, который выдал код	Зависимый
	2. С родителем, который делит код с ребенком	Зависимый
Родитель пациента	1. С ребенком, который делит код с родителем	Основной
	2. С врачом, который выдал код ребенку	Зависимый

Согласно этой таблице, выстраиваются отношения между пользователями, она лежит в основе валидации связей при запросах к API при действиях, затрагивающих нескольких пользователей.

В систему пользователя может пригласить только доктор организации, в которой пациент зарегистрирован. Для этого функционала потребовалось добавление ещё нескольких методов в API – для генерации уникального кода каждого пациента и удобных методов для заведения нового пациента в системе уже самим доктором.

После появления пользователя в системе у него должна быть возможность аутентифицироваться в приложении с помощью тех данных, которые он использовал при регистрации. Для этого был реализован механизм аутентификации через JWT токены, представленный на рисунке 5.

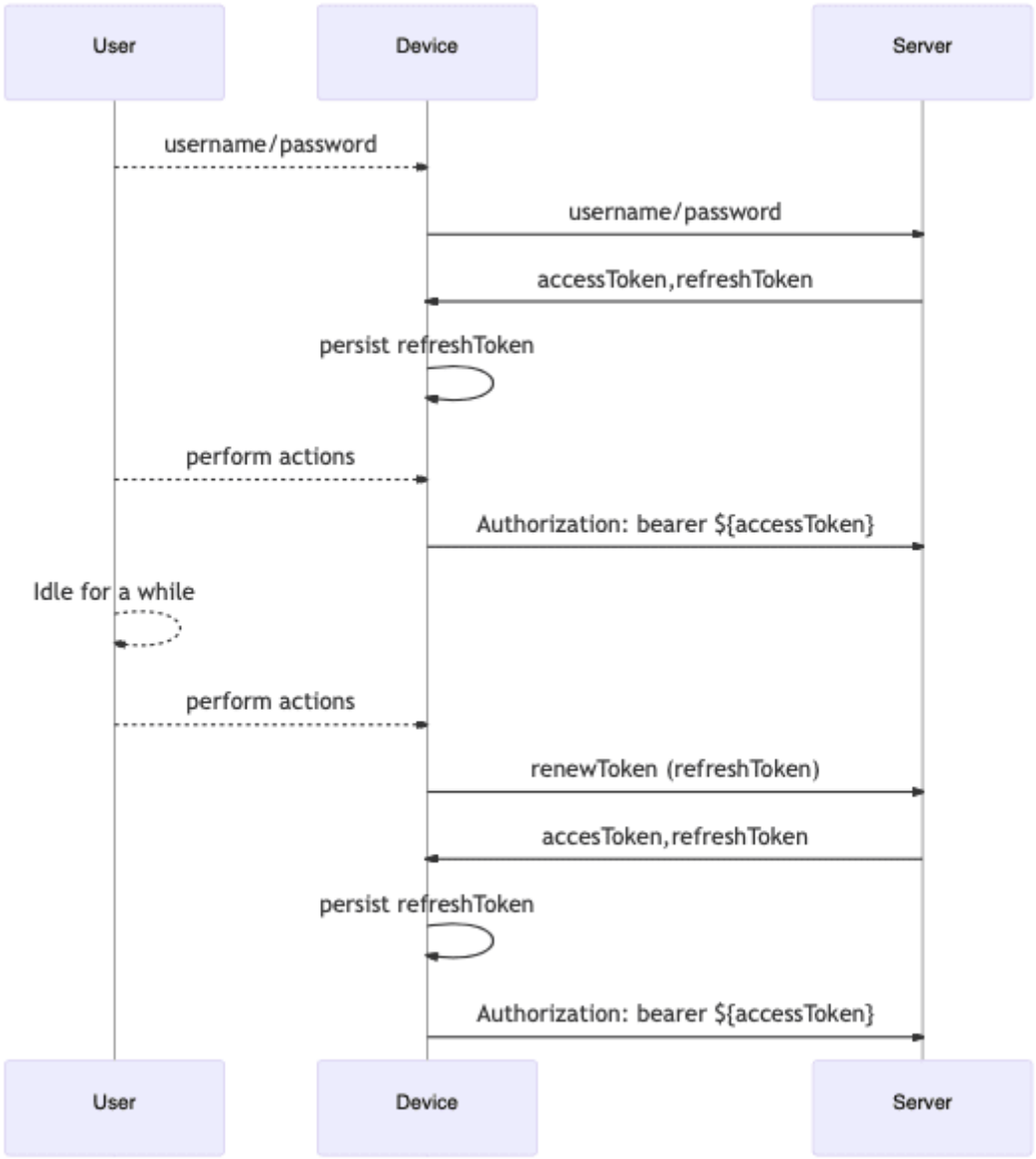

Рисунок 5 – Механизм аутентификации по JWT токену

JWT токены используют пару токенов, access token и refresh token, для аутентификации и авторизации пользователей. Access token – это краткосрочный токен, который предоставляет пользователю доступ к защищенным ресурсам и имеет ограниченный срок действия. Refresh token — это долгосрочный токен, используемый для получения нового access token после истечения срока действия предыдущего. Это разделение повышает безопасность, упрощает управление сеансами и обеспечивает масштабируемость, позволяя пользователям оставаться

аутентифицированными в течение длительного времени без необходимости повторного входа в систему.

По итогу реализации этого этапа в приложении появилась ролевая модель, а также стал доступен сценарий приглашения группы пользователей с последующей их регистрацией и аутентификация и авторизация уже зарегистрированных пользователей.

3.4.2 Пользовательские уведомления и геймификация

Уведомления для пользователя мобильного приложения делятся на два типа: уведомления, источником которых является сервер, и уведомления, источником которых является само мобильное приложение. Это могут быть как напоминания о приеме таблеток, так и уведомления о новом сообщении в чате или других системных событиях.

Учитывая требование о том, что родитель и доктор пациента могут просматривать и редактировать напоминания, а значит должны быть доступны в один момент времени разным пользователям, было принято решение о хранении необходимых данных на серверной части приложения в базе данных, а не на устройстве конкретного пользователя. А уже само мобильное приложение на основе этих данных сформирует необходимые напоминания и запланирует их отправку. К такому типу напоминаний относятся напоминания о приеме медикаментов.

Для хранения необходимой информации были сформированы таблицы «Напоминание о приеме лекарства» и «Лекарства», хранящие частотность и длительность приема, а также принимаемые медикаменты.

Согласно требованиям, были реализованы методы для чтения, создания, редактирования и удаления напоминаний. Причем с этими напоминаниями может взаимодействовать как их автор, так и связанный с автором доктор или родитель.

Вторым типом пользовательских уведомлений являются уведомления, источником которых является серверная часть приложения. Они отправляются в следующих случаях:

– доктору, в случае если пациент совершил действие с напоминанием о приеме лекарств;

– пациенту, если доктор совершил действие с его напоминанием о приеме лекарств;

– доктору о новой регистрации пациента по его коду приглашения;

– любому пользователю при появлении нового сообщения в чате.

Такой тип напоминаний работает через платформу FireBase, используя её API. Платформе для отправки уведомления требуется уникальный идентификатор устройства пользователя, поэтому при каждом входе пользователя в систему происходит сохранение этого идентификатора в базу данных.

На этом этапе в приложении целевые возможности системы – напоминания пользователю о приеме лекарств, способствующие стабилизации регулярности их приема.

Для поддержки ещё одной важной идеи мобильного приложения – геймификации – были созданы несколько таблиц для хранения пользовательского прогресса в мини-играх, а также несколько конечных точек для взаимодействия с клиентами. Мобильные клиенты присылают обновленные данные после каждого прохождения игры, а система на сервере высчитывает прогресс в очках опыта и внутренней валюте приложения, отдавая новые данные обратно для показа пользователю.

3.4.3 Коммуникации в реальном времени

Для коммуникации внутри приложения внедрена возможность общения через чаты, как в стандартных мессенджерах, например, Viber или WhatsApp. Пользователь может начать диалог с другим пользователем на основе их

связей, описанных ранее в разделе «Общие требования к системе». На данный момент в чате можно отправить только текстовую информацию.

Чаты работают в режиме реального времени, то есть пользователь сразу получает сообщение, адресованное ему. В основе такого поведения лежит протокол WebSockets, который позволяет поддерживать двунаправленный туннель связи между клиентом и сервером.

Безопасность общения поддерживает подключенный к домену SSL сертификат, обеспечивающий создание зашифрованному каналу связи. В дополнение к нему доступ к общению закрыт авторизацией, неавторизованный пользователь написать сообщение не сможет.

На сервере за имплементацию протокола отвечает библиотека SignalR, использующая абстракции, именуемые «hub», для инкапсуляции логики подключения клиента к серверу, поддержания и обновления соединений. Библиотека позволяет отправлять сообщения как одному клиенту, так и целой группе пользователей, основанной на определенных правилах группировки.

Для хранения данных по начатым чатам и написанным в них сообщениям в базе данных были созданы таблицы «Чаты» и «Сообщения».

В итоге на этом этапе был получен модуль коммуникации в приложении, который будет использоваться как быстрый способ связи пользователей друг с другом для решения возникающих вопросов и получения обратной связи.

3.5 Логирование и мониторинг

Для обеспечения высокого уровня качества и надежности после размещения работающей системы на облачном сервере важно максимально покрыть её метриками и логами, чтобы в любой момент времени быть уверенным в том, что всё работает корректно.

Для логирования на сервере используется библиотека Serilog – это библиотека ведения структурированных логов для .NET, которая предоставляет быстрый и расширяемый способ записи логов. Она позволяет

создавать собственные форматы логов, фильтровать и обогащать события журнала, и отправлять их в различные выходные потоки, такие как консоль, файлы или удаленные серверы.

Исходя из возможностей Serilog были выбраны два выходных потока для записей структурированных логов – это файл в контейнере в папке, которая присоединена к контейнеру из операционной системы хоста для удобного просмотра логов, и сервер сбора и агрегации логов Seq, который предоставляет веб-интерфейс для изучения, фильтрации и поиска событий в реальном времени. С помощью Seq также получилось построить систему предупреждений разработчика если в системе появилось критическое количество ошибок. Пример уведомления представлен на рисунке 6.

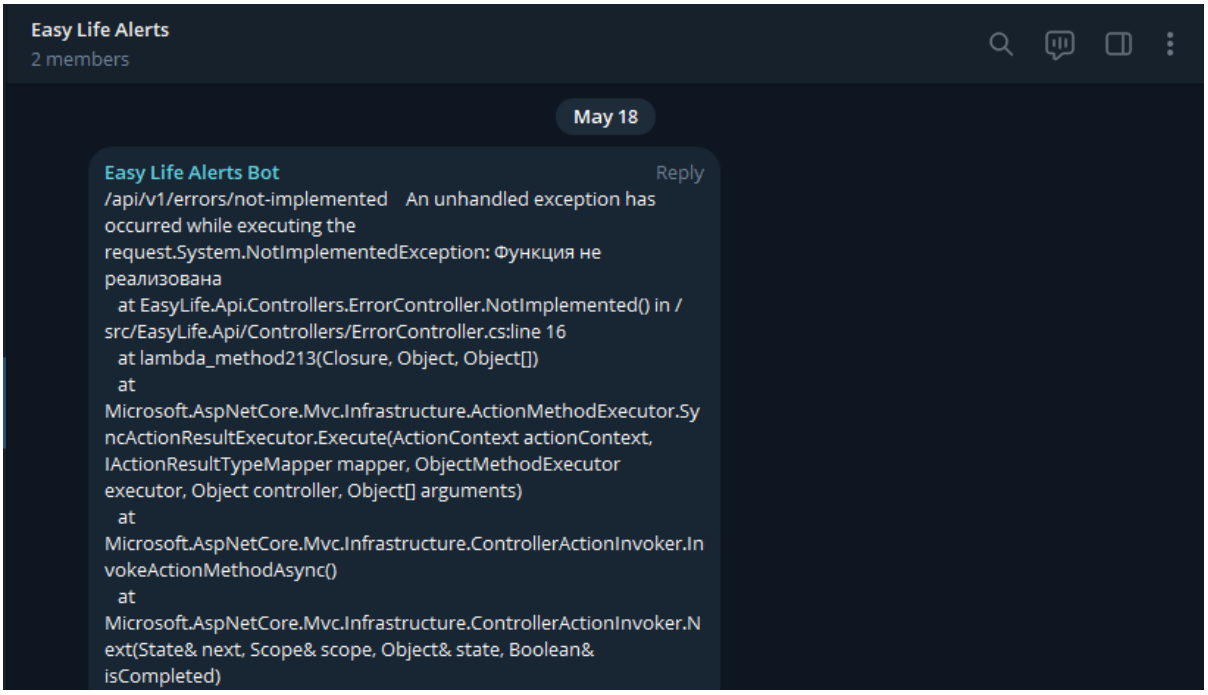

Рисунок 6 – Уведомление в мессенджер об ошибке на сервере

После реализации этих компонентов можно быть уверенным в надежности работы системы, так как в случае непредвиденных обстоятельств она сама предупредит об этом.

3.6 Пользовательское тестирование

После реализации проекта было проведено две итерации пользовательского тестирования. На первой итерации приложение было запущено на небольшой выборке пользователей, состоящих из заинтересованных знакомых, после которого была получена обратная связь по удобству использования и необходимым возможностям. На этом этапе была немного доработана бизнес-логика API: разработаны возможности по добавлению нескольких лекарственных препаратов к одному напоминанию, а также реализована аутентификация посредством JWT-токена для подключения аутентификации по PIN-коду в мобильном приложении. Технических изменений от серверной части на этом этапе не потребовалось.

На второй итерации тестирование уже осуществлялось на выборке из реальных пациентов СПИД-центра по договоренности с администрацией в целях выявления существующих проблем приложения при непосредственном использовании как пациентом, так и врачом.

На обоих этапах серверная часть системы была стабильной и работала согласно поставленным требованиям.

ЗАКЛЮЧЕНИЕ

Аспекты разработки, изученные в ВКР, являются основополагающими для создания гибких и масштабируемых приложений. Такой подход позволяет упростить будущую разработку системы, включающую как доработки существующих компонентов, так и расширение системы в целом, а также её поддержку. Система получилась гибкой и готовой к расширению в случае необходимости.

Цель была полностью достигнута, а поставленные задачи – выполнены. Система реализована согласно поставленным требованиям и отвечает всем функциональным потребностям мобильного приложения.

К системе такого формата можно подключать не только мобильные клиенты, но и любые другие – будь то приложение для браузера или приложение для персонального компьютера. Благодаря универсальному интерфейсу система на сервере будет работать одинаково для всех.

Успешно проведены две итерации пользовательского тестирования. Ценные рекомендации, высказанные участниками тестирования, были приняты к сведению и будут учтены при дальнейшем развитии приложения. В целом, результаты пользовательского тестирования подтвердили: высокое качество технической реализации приложения, продуманность и удобство архитектуры, соответствие функционала приложения ожиданиям целевой аудитории.

Реализованное приложение позволит успешно закрывать потребности медицинских центров по контролю приверженности лечению за счет сформированного базового набора компонентов, способствующих развитию привычки принимать терапию, а также коммуницировать с лечащим врачом.

Подготовлена вся необходимая инфраструктура для масштабирования системы при необходимости – скрипты для конвейера CI/CD, сервер с установленными зависимостями, инфраструктура с логированием, мониторингом и уведомлениями в мессенджер.

Планы на будущее включают в себя развитие инфраструктуры системы: балансировку запросов на основе состояния реплик, гибкое масштабирование, кэширование для повышения производительности. Помимо улучшения вышеописанной инфраструктуры планируется также улучшить качество тестирования: написать новые тесты, внедрить слоты на основе веток и возможность открыть их для предпросмотра.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Подготовка и оформление магистерской диссертации [Текст]: Методические рекомендации / Васина В. Н., Хлебников Н. А., Т.И. Алферьева, Шадрин Д.Б. - Екатеринбург. 2023. - 96 с.

2. Лукина Ю. В., Кутишенко Н. П., Марцевич С. Ю. Проблема приверженности в современной медицине: возможности решения, влияние на результативность терапии и исходы заболевания //Рациональная фармакотерапия в кардиологии. – 2017. – Т. 13. – №. 4. – С. 519-524.

3. World Health Organisation: Adherence to Long-Term Therapies, Evidence for Action. Geneva: WHO. – 2003. – С. 230.

4. Наумова Е. А., Семенова О. Н. Современный взгляд на проблему приверженности пациентов к длительному лечению //Кардиология: Новости. Мнения. Обучение. – 2016. – №. 2 (9). – С. 30-39.

5. Лукина Ю. В., Кутишенко Н. П., Марцевич С. Ю. Приверженность лечению: современный взгляд на знакомую проблему //Кардиоваскулярная терапия и профилактика. – 2017. – Т. 16. – №. 1. – С. 91-95.]

6. World Health Organisation: Adherence to Long-Term Therapies, Evidence for Action. Geneva: WHO. – 2003. – С. 110.

7. Захарова Е. В. Теоретические концепции и методы исследования комплаенса и приверженности лечению //Теоретическая и экспериментальная психология. – 2019. – Т. 12. – №. 3. – С. 96-110.

8. Мищенко М.А., Кононова С.В. Анализ факторов, влияющих на приверженность к гиполидемической терапии // Медицинский альманах. – 2014. – № 1(31). – С. 95–98.

9. Олейников В.Э., Елисеева И.В., Томашевская Ю.А., Борисова Н.А., Фадеева С.С. Эффективность антигипертензивной терапии у пожилых пациентов и анализ приверженности лечению // Рациональная фармакотерапия в кардиологии. – 2014. – Т. 10. – № 4. – С. 391–396.

10. Приверженность рекомендованной терапии больных, перенесших

острый коронарный синдром, и риск развития сердечно-сосудистых осложнений в течение года после госпитализации //Рациональная фармакотерапия в кардиологии. – 2011. – Т. 7. – №. 5. – С. 567-573.

11. Ильенкова Н. А., Черепанова И. В., Вохмина Т. А. Проблемы приверженности терапии у детей с бронхиальной астмой //Педиатрическая фармакология. – 2016. – Т. 13. – №. 6. – С. 565-570.

12. Николаев Н. А. и др. Управление лечением на основе приверженности //Consilium Medicum. – 2020. – Т. 22. – №. 5. – С. 9-18.

13. Белостоцкий А. В. и др. Проблема приверженности больных туберкулезом к лечению //Туберкулез и болезни легких. – 2015. – №. 4. – С. 4-9.

14. Панов В. П., Логунов Д. Л., Авдеева М. В. Приверженность пациентов лечебно-профилактическим мероприятиям и здоровому образу жизни: актуальность проблемы и возможности преодоления //Социальные аспекты здоровья населения. – 2016. – Т. 48. – №. 2. – С. 8.

15. Кувшинова Н. Ю. Проблема приверженности терапии в различных областях медицины //Известия Самарского научного центра Российской академии наук. – 2015. – Т. 17. – №. 5-3. – С. 1014-1020.

16. Тулепбергенов Г. К. и др. ЭФФЕКТИВНОСТЬ МОБИЛЬНЫХ ПРИЛОЖЕНИЙ ДЛЯ ПОВЫШЕНИЯ ПРИВЕРЖЕННОСТИ ДЛЯ ПАЦИЕНТОВ С ФИБРИЛЛЯЦИЕЙ ПРЕДСЕРДИЙ //Вестник Казахского Национального медицинского университета. – 2022. – №. 1. – С. 165-171.

17. Дрёмова Н. Б. Проблемы приверженности лекарственной терапии у современных пациентов //Фармакоэкономика: теория и практика. – 2018. – Т. 6. – №. 1. – С. 47-47.

18. Шмонин А. А. и др. Проблемы приверженности лекарственной терапии в медицинской реабилитации //Доктор. Ру. – 2017. – №. 11 (140). – С. 19-26.

19. Ноздрачев Д. И., Замятин К. А., Таратухин Е. О. Цифровые средства повышения приверженности к лечению //Российский

кардиологический журнал. – 2019. – №. 12. – С. 96-102.

20. Jiménez-Chala E. A. et al. Use of Mobile Applications to Increase Therapeutic Adherence in Adults: A Systematic Review //Journal of Medical Systems. – 2022. – Т. 46. – №. 12. – С. 87.

21. Pérez-Jover V. et al. Mobile apps for increasing treatment adherence: systematic review //Journal of Medical Internet Research. – 2019. – Т. 21. – №. 6. – С. e12505.

22. Peng Y. et al. Effectiveness of mobile applications on medication adherence in adults with chronic diseases: a systematic review and meta-analysis //Journal of managed care & specialty pharmacy. – 2020. – Т. 26. – №. 4. – С. 550-561.

23. Жданова С. Н. и др. Опыт использования мобильного приложения для повышения приверженности к лечению больных туберкулезом и ВИЧ-инфекцией //Туберкулез и болезни легких. – 2021. – Т. 99. – №. 11. – С. 17-24.

24. Jácome C. et al. Feasibility and acceptability of an asthma app to monitor medication adherence: mixed methods study. JMIR Mhealth Uhealth. 2021 May 25; 9 (5): e26442. doi: 10.2196/26442.

25. Lv S. et al. A randomized controlled trial of a mobile application-assisted nurse-led model used to improve treatment outcomes in children with asthma //Journal of advanced nursing. – 2019. – Т. 75. – №. 11. – С. 3058-3067.

26. Goyal S. et al. A mobile app for the self-management of type 1 diabetes among adolescents: a randomized controlled trial //JMIR mHealth and uHealth. – 2017. – Т. 5. – №. 6. – С. e7336.

27. Ходжиева Г. С. Внедрение мобильных приложений в процесс контроля комплаенса пациентов при лечении анемии //BARQARORLIK VA YETAKCHI TADQIQOTLAR ONLAYN ILMIY JURNALI. – 2023. – Т. 3. – №. 12. – С. 358-366

28. Spat S. et al. Development, integration and operation of mobile, Android-based medical devices in hospitals: Experiences from the GlucoTab® system //2014 4th International Conference on Wireless Mobile Communication and

Healthcare-Transforming Healthcare Through Innovations in Mobile and Wireless Technologies (MOBIHEALTH). – IEEE, 2014. – С. 128-131.

29. Акбарова М. Р. Безопасность и защита данных в облачных технологиях //Universum: технические науки. – 2022. – №. 10-1 (103). – С. 17-19.

30. Нестеренко В. Р., Маслова М. А. Современные вызовы и угрозы информационной безопасности публичных облачных решений и способы работы с ними //Научный результат. Информационные технологии. – 2021. – Т. 6. – №. 1. – С. 48-54.

31. Roziqin M. C. et al. Implementation of REST API in Web Service System for Medical Resume Provision //International Journal of Healthcare and Information Technology. – 2023. – Т. 1. – №. 1. – С. 34-48.

32. Мартин Р. Чистая архитектура. Искусство разработки программного обеспечения [Текст]: Техническая литература / Р. Мартин; Н. Римицан; А. Киселев. - Санкт-Петербург: Питер, 2022 - 352 с.

33. Анализ предметной области [Электронный ресурс]. Анализ предметной области. Выявление функциональных требований к приложению. Режим доступа: https://intuit.ru/studies/courses/574/430/lecture/9749 (дата обращения: 09.03.2024).

34. Reynders, Fanie. Modern API Design with ASP.NET Core 2. Berkeley, CA: Apress, 2018. https://doi.org/10.1007/978-1-4842-3519-5.

35. REST API [Электронный ресурс] // IBM Cloud Education: [сайт]. URL: https://www.ibm.com/cloud/learn/rest-apis (дата обращения: 11.03.2024). ILIĆ M. et al. The difference between ADO .NET and Entity Framework in software. – 2021.

36. Klimek B., Skublewska-Paszkowska M. Comparison of the performance of relational databases PostgreSQL and MySQL for desktop application //Journal of Computer Sciences Institute. – 2021. – Т. 18. – С. 61-66.

37. Makris A. et al. MongoDB Vs PostgreSQL: A comparative study on performance aspects //GeoInformatica. – 2021. – Т. 25. – С. 243-268.

38. Schüle M. E. et al. Freedom for the sql-lambda: Just-in-time-compiling user-injected functions in postgresql //32nd International Conference on Scientific and Statistical Database Management. – 2020. – С. 1-12.

39. Weil A. Learn Microservices-ASP. NET Core and Docker. – Lulu. com, 2018.

40. SOLID [Электронный ресурс]. SOLID: The First 5 Principles of Object-Oriented Design. Режим доступа: https://www.digitalocean.com/community/conceptual_articles/s-o-l-i-d-the-first-five-principles-of-object-oriented-design (дата обращения: 12.03.2024).

41. Kinsman, Timothy, Mairieli Wessel, Marco A. Gerosa, и Christoph Treude. «How Do Software Developers Use GitHub Actions to Automate Their Workflows?» В 2021 IEEE/ACM 18th International Conference on Mining Software Repositories (MSR), 420–31, 2021.

42. Singh, Charanjot, Nikita Seth Gaba, Manjot Kaur, и Bhavleen Kaur. «Comparison of Different CI/CD Tools Integrated with Cloud Platform». В 2019 9th International Conference on Cloud Computing, Data Science & Engineering (Confluence), 7–12, 2019. https://doi.org/10.1109/CONFLUENCE.2019.8776985.

43. Technology stack [Электронный ресурс]. What factors go into choosing a technology stack. Режим доступа: https://www.aha.io/roadmapping/guide/it-strategy/technology-stack (дата обращения: 11.03.2024).

44. Разработка мобильного сервиса в сфере здравоохранения для детей, живущих с ВИЧ / Савченко Н.В., Николаева К.И., Уфимцева М.А., Жунисова Д.С., Бочкарев Ю.М. / Современные проблемы науки и образования, г. Екатеринбург. – 2020. – С. 170.

45. Об информации, информационных технологиях и о защите информации [Текст]: Федеральный закон РФ от 27.07.2006 N 149-ФЗ // http://www.consultant.ru. – 2006. – 14 июля.

46. О персональных данных [Текст]: Федеральный закон РФ от 27.07.2006 N 152-ФЗ // http://www.consultant.ru. – 2006. – 14 июля.

47. ГОСТ 19.201–78. Единая система программной документации (ЕСПД). Техническое задание. Требования к содержанию и оформлению [Текст]: дата введения 1980-01-01. – М.: Стандартинформ, 2010. – 4 с.

48. ГОСТ 19.102–77. Единая система программной документации (ЕСПД). Стадии разработки [Текст]: дата введения 1980-01-01. – М.: Стандартинформ, 2010. – 4 с.

49. Docker overview [Электронный ресурс] // Docker docs: [сайт]. URL: https://docs.docker.com/get-started/overview/ (дата обращения: 15.03.2024).

50. ADO.NET and Entity Framework [Электронный ресурс] // Meduim: [сайт]. URL: https://medium.com/@sadigrzazada20/ado-net-and-entity-framework-faf43f66675c (дата обращения: 12.03.2024).