МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ КЫРГЫЗСКОЙ РЕСПУБЛИКИ

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ КЫРГЫЗСКО-РОССИЙСКИЙ СЛАВЯНСКИЙ УНИВЕРСИТЕТ ИМЕНИ ПЕРВОГО ПРЕЗИДЕНТА РОССИЙСКОЙ ФЕДЕРАЦИИ Б.Н. ЕЛЬЦИНА

ЕСТЕСТВЕННО-ТЕХНИЧЕСКИЙ факультет

Кафедра информационных и вычислительных технологий

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
на тему:

Разработка распределенной системы мониторинга технического состояния экзомассажеров

09.04.04 (710400) – Программная инженерия

Магистерская программа «Разработка программно-информационных систем»

Выполнил студент группы ЕПИМ-2-22

Лазарев Дмитрий Денисович

Руководитель
к.т.н., доц. **Верзунов Сергей Николаевич**

Рецензент
д.т.н., проф. **Брякин Иван Васильевич**

Работа к защите допущена
Заведующая кафедрой
д.т.н., проф. **Лыченко Наталья Михайловна**

БИШКЕК 2024

Аннотация

Цель данной работы заключается в разработке программных средств для создания распределенной системы мониторинга технического состояния экзомассажеров. Работа включает в себя проектирование и реализацию программного обеспечения, способного собирать, анализировать и визуализировать данные, получаемые от удаленных устройств - экзомассажеров. Главная цель состоит в обеспечении оперативного контроля за работой специальных устройств, выявлении проблемных ситуаций и предотвращении возможных отказов в работе.

Для решения поставленной задачи были выполнены следующие работы:

- Проведен анализ предметной области.
- Разработана система, предоставляющая владельцам устройств возможность мониторинга технического состояния экзомассажеров и управления работой экзомассажера.
- Реализована web-платформа мониторинга и управления с простым и понятным интерфейсом.

Разработанная система выполняет следующие функции:

- Получение информации с устройств с помощью разработанной API.
- Создание чата поддержки операторов.
- Удаленное управление устройствами
- Хранение полученных результатов мониторинга в базе данных

Средства разработки:

- Среда разработки: PyCharm, Visual Studio Code
- Язык программирования: Python, HTML (JS, CSS)
- Фреймворк: Django
- База данных: MySQL

Объем пояснительной записки: 65 стр.

Количество приложений: 5

Количество рисунков: 29 шт.

Количество таблиц: 15 шт.

Annotation english

The purpose of this work is to develop software for creating a distributed system for monitoring the technical condition of exomassagers. The work includes the design and implementation of software capable of collecting, analyzing and visualizing data received from remote devices
- exomassagers. The main goal is to ensure operational control over the operation of special devices, identify problem situations and prevent possible operational failures.
To solve the problem, the following work was performed:

- Analysis of the subject area was carried out.
- A system has been developed that provides device owners with the ability to monitor the technical condition of exo-massagers and control the operation of the exo-massager.
- Implemented a web-based monitoring and management platform with a simple and intuitive interface

The developed system performs the following functions:

- Receiving information from devices using the developed API.
- Creating an operator support chat.
- Remote device management
- Storing the obtained monitoring results in a database

Development Tools:

- Development environment: PyCharm, Visual Studio Code
- Programming language: Python, HTML (JS, CSS)
- Framework: Django
- Database: MySQL

Volume of explanatory note: 65 pages.
Number of applications: 5
Number of drawings: 29 pcs.
Number of tables: 15 pcs.

Аннотация кыргызча

Бул иштин максаты - экзомассажерлердин техникалык абалын көзөмөлдөө үчүн бөлүштүрүлгөн системаны түзүү үчүн программалык камсыздоону иштеп чыгуу. Иш алыскы түзмөктөрдөн алынган маалыматтарды чогултуу, талдоо жана визуализациялоого жөндөмдүү программалык камсыздоону иштеп чыгууну жана ишке ашырууну камтыйт
- экзомассажерлер. Негизги максат - атайын түзүлүштөрдүн иштешине оперативдүү көзөмөлдү камсыз кылуу, көйгөйлүү кырдаалдарды аныктоо жана мүмкүн болгон операциялык мүчүлүштүктөрдү алдын алуу.

Маселени чечүү үчүн төмөнкү иштер аткарылды:

- Предметтик аймак талдоого алынды.
- Аппарат ээлерине экзомассажерлердин техникалык абалын көзөмөлдөө жана экзомассажердин иштешин көзөмөлдөө мүмкүнчүлүгүн берген система иштелип чыкты.
- Жөнөкөй жана интуитивдик интерфейс менен веб-негизделген мониторинг жана башкаруу платформасы ишке ашырылган.

Иштелип чыккан система төмөнкү функцияларды аткарат:

- Иштелип чыккан API аркылуу түзмөктөрдөн маалымат алуу.
- Операторду колдоо чатын түзүү.
- Түзмөктү алыстан башкаруу
- Мониторингдин алынган натыйжаларын маалымат базасында сактоо

Иштеп чыгуу куралдары:

- Иштеп чыгуу чөйрөсү: PyCharm, Visual Studio Code
- Программалоо тили: Python, HTML (JS, CSS)
- Framework: Django
- Маалыматтар базасы: MySQL

Түшүндүрмө каттын көлөмү: 65 бет.

Өтүнмөлөрдүн саны: 5

Сүрөттөрдүн саны: 29 даана.

Столдордун саны: 15 даана.

Оглавление

Введение

Современное общество сталкивается с рядом вызовов в области здравоохранения, таких как увеличение числа людей с ограниченной подвижностью и возрастающее внимание к уходу за физическим здоровьем. В связи с этим растет интерес к инновационным технологиям, направленным на улучшение качества жизни и обеспечение эффективного медицинского ухода. Одним из таких технологических решений являются экзомассажеры.

Экзомассажеры представляют собой устройства, разработанные для предоставления массажных процедур с минимальным участием человека. Они обладают способностью к точному и многократному повторению массажных движений, что делает их идеальным инструментом для реабилитации после травмы, улучшения кровообращения, снятия мышечных напряжений и облегчения боли.

Однако, как и любое техническое устройство, экзомассажеры требуют систематического мониторинга и технического обслуживания для обеспечения их эффективной работы, и безопасности пациентов. В этой работе рассматривается разработка программных средств для создания распределенной системы мониторинга и управления техническим состояниям экзомассажеров, которая позволит оперативно контролировать и удаленно управлять их состоянием.

Пояснительная записка к ВКР состоит из введения, 5 глав, заключения и приложений:

- В первой главе описан анализ проблемы и результаты анализа проблем
- В второй главе разработка требований к программному продукту. В данном разделе рассматриваются специфические требования, присущие данному программному продукту.
- В третьей главе описаны проектирование и конструирование ПО. В этом разделе описывается процесс разработки ПО
- Разработка тестов и тестирование программного продукта. Этот раздел состоит из разработки плана
- Руководство пользователя. Данный раздел включает в себя описание необходимой информации для пользователя системы.

Глава 1. Описание исследуемой проблемы

1.1.1. Преимущества и недостатки экзомассажеров

Перевод традиционных методов массажа в цифровую форму делает их более доступными и удобными для широкого круга пользователей. Использование экзомассажеров, как примера таких технологий, обладает рядом преимуществ и недостатков.

Основные преимущества экзомассажеров:

- *Эффективность и точность*: Экзомассажеры обеспечивают точное и многократное повторение массажных движений, что делает их идеальными для реабилитации после травм, улучшения кровообращения, снятия мышечного напряжения и облегчения боли. Их работа не зависит от усталости или опыта человека, что позволяет поддерживать высокое качество массажа на протяжении длительного времени.
- *Снижение затрат:* Использование экзомассажеров позволяет сократить численность персонала, необходимого для проведения массажных процедур. В отличие от услуг профессиональных массажистов, экзомассажеры требуют минимального обслуживания и могут работать автономно, что значительно снижает операционные расходы.
- *Доступность и удобство*: Экзомассажеры могут быть использованы в любое время и в любом месте, где доступно необходимое оборудование. Это особенно важно для людей с ограниченной подвижностью или тех, кто нуждается в регулярном массаже для поддержания здоровья.
- *Персонализация и адаптивность*: Современные экзомассажеры могут быть настроены под индивидуальные потребности пользователя. Это позволяет подобрать оптимальные режимы и интенсивность массажа, что делает процедуры более эффективными и комфортными.

- *Возможность маркетинговых исследований*: Экзомассажеры, оснащенные сенсорами и средствами связи, могут собирать данные о предпочтениях пользователей и эффективности процедур. Это позволяет проводить маркетинговые исследования и улучшать продукт на основе полученной информации без дополнительных затрат.

К недостаткам экзомассажеров можно отнести следующее:

- *Технические требования*: для полноценного функционирования экзомассажеры требуют надежных каналов связи и качественного аппаратного и программного обеспечения. Это может быть препятствием для их использования в регионах с недостаточно развитой инфраструктурой.
- *Дороговизна доставки и установки*: несмотря на сокращение затрат на персонал, доставка и установка экзомассажеров могут быть достаточно дорогими. Эти расходы могут значительно увеличить себестоимость услуги, особенно в удаленных или труднодоступных районах.
- *Ограниченные возможности диагностики*: Экзомассажеры, несмотря на все свои преимущества, не могут полностью заменить квалифицированных специалистов, способных провести комплексную диагностику состояния пациента и адаптировать массаж в соответствии с его индивидуальными потребностями.

Основные преимущества для потребителей:

- *Экономия времени*: Пользователи экзомассажеров могут получать качественный массаж в любое удобное время, не выходя из дома. Это особенно важно для людей с насыщенным графиком, которые не могут часто посещать массажные салоны.
- *Экономия денег*: Затраты на использование экзомассажеров, включая их обслуживание, часто ниже, чем регулярные визиты к профессиональному массажисту. Это позволяет получать качественные массажные услуги при меньших финансовых затратах.
- *Расширенные возможности выбора и информации*: Экзомассажеры могут быть оснащены функциями поиска и оценки различных режимов и программ массажа, предоставляя пользователям доступ к большому количеству информации о доступных опциях. Это позволяет легко подобрать наиболее подходящий режим массажа и получать рекомендации на основе предыдущего опыта использования.

Использование экзомассажеров представляет собой перспективное направление в области здравоохранения и реабилитации, объединяющее высокую эффективность и доступность с возможностью индивидуализации процедур. Однако для полного раскрытия их потенциала необходимо учитывать и решать сопутствующие технические и организационные проблемы.

1.2. Анализ различных подходов к решению

Разработка приложения или веб-сайта с нуля требует значительных усилий. Часто необходимо воссоздавать функции, которые уже были реализованы множество раз, что эквивалентно изобретению колеса заново. Программные фреймворки помогают избежать этой проблемы, предоставляя готовую основу для разработки. В мире программного обеспечения термин "фреймворк" относится к библиотекам файлов, включающим несколько основных функций. Цель фреймворка — предоставить основу, которую можно использовать для более эффективной разработки проектов, включая множество функций, которые иначе пришлось бы кодировать с нуля.

Наиболее популярными фреймворками Python являются Django, Flask и Pyramid.

Django

Django — это высокоуровневый веб-фреймворк на языке Python, способствующий быстрой разработке и чистому, прагматичному дизайну. Созданный для того, чтобы избавить разработчиков от большинства проблем, связанных с веб-разработкой, Django позволяет быстро создавать сложные, функциональные веб-приложения.

Ключевые особенности Django:

- **Высокая безопасность:** Django включает множество встроенных функций безопасности, таких как защита от SQL-инъекций, XSS-атак и CSRF-атак.
- **Аутентификация:** Встроенная система аутентификации, поддерживающая множество форм аутентификации, включая OAuth и другие социальные сети.
- **Открытый код и большое сообщество:** Широкое сообщество разработчиков, активно участвующих в развитии фреймворка и поддержке сторонних приложений.
- **Шаблоны:** Django использует мощный шаблонизатор, позволяющий создавать многоразовые и легко поддерживаемые шаблоны.

- **Миграции баз данных:** Встроенная система миграций позволяет легко изменять структуру базы данных и откатывать изменения при необходимости, что особенно полезно при групповой разработке.

Flask

Flask — это легковесный веб-фреймворк на языке Python, предназначенный для быстрого и простого создания веб-приложений. Flask подходит для проектов любой сложности благодаря своей гибкости и расширяемости.

Ключевые особенности Flask:

- **Минимализм:** Flask предоставляет минимально необходимый набор функций, позволяя разработчикам добавлять только те расширения, которые им нужны.
- **Гибкость:** Flask не навязывает определенную структуру проекта, что позволяет создавать проекты с произвольной архитектурой.
- **Расширяемость:** Flask поддерживает множество расширений для добавления различных функций, таких как аутентификация, миграции баз данных и т.д.
- **Легкость освоения:** Простой и интуитивно понятный интерфейс делает Flask идеальным выбором для начинающих разработчиков.

Pyramid

Pyramid — это гибкий веб-фреймворк на языке Python, предназначенный для создания как простых, так и сложных веб-приложений. Pyramid предоставляет разработчикам возможность выбирать, какие компоненты и библиотеки использовать, что делает его универсальным инструментом для различных проектов.

Ключевые особенности Pyramid:

- **Гибкость и масштабируемость:** Pyramid поддерживает широкий спектр конфигураций, от небольших до крупных и сложных приложений.
- **Поддержка множества баз данных:** Pyramid совместим с различными системами управления базами данных, такими как SQLAlchemy и ZODB.
- **Безопасность:** Встроенные механизмы безопасности, включая аутентификацию и авторизацию, помогают защитить приложения от атак.
- **Документация и сообщество:** Обширная документация и активное сообщество разработчиков, готовых помочь и поделиться своими знаниями.

В табл. 1.1. представлен анализ различных подходов к решению.

Таблица 1.1

Анализ различных подходов к решению

	Django	Flask	Pyramid
Высокая безопасность	+	+	+
Аутентификация	+	+ (через расширения)	+
Открытый код и большое сообщество	+	+	+
Шаблоны	+ (встроенные)	+ (через расширения)	+ (через расширения)
Миграции баз данных	+	+ (через расширения)	+ (через расширения)
Производительность	+	+	+

Каждый из рассмотренных фреймворков Python имеет свои преимущества и недостатки. Django подходит для быстрого создания сложных веб-приложений благодаря своей полноте и встроенным функциям. Flask предоставляет гибкость и легкость освоения, что делает его идеальным выбором для небольших и средних проектов. Pyramid сочетает в себе гибкость и масштабируемость, подходя для проектов любой сложности.

Выбор фреймворка зависит от конкретных требований проекта, предпочтений разработчиков и уровня их опыта.

1.2.1. Деловые преимущества

- **Улучшение обслуживания и поддержки клиентов:**

Быстрое решение проблем: Возможность удаленно блокировать устройства или обновлять их прошивку позволяет быстро реагировать на любые проблемы или ошибки, которые могут возникнуть. Это повышает уровень удовлетворенности клиентов, так как их проблемы решаются оперативно.

Профилактическое обслуживание: Система мониторинга позволяет выявлять потенциальные проблемы до того, как они станут серьезными. Это помогает минимизировать простои и улучшить общее обслуживание.

- **Увеличение безопасности**:

Удаленная блокировка: В случае кражи или потери устройства, возможность удаленной блокировки предотвращает несанкционированное использование, защищая как данные пользователей, так и репутацию компании.

Обновления безопасности: Регулярные обновления прошивки могут включать патчи безопасности, что помогает защитить устройства от новых угроз и уязвимостей.

- **Снижение эксплуатационных затрат:**

Удаленное обслуживание: Снижение необходимости физического обслуживания устройств уменьшает расходы на поездки и логистику для обслуживания или обновления устройств.

Эффективное использование ресурсов: Мониторинг использования устройств позволяет лучше понимать, как и когда они используются, что помогает оптимизировать ресурсное планирование и управление запасами.

- **Увеличение доходов:**

Новые возможности монетизации: Возможность добавления новых функций через обновления прошивки позволяет предлагать пользователям платные обновления или подписки на новые сервисы.

Улучшение качества продукта: Постоянное обновление и улучшение функций устройства через прошивку повышает его ценность для пользователей, что может способствовать увеличению продаж и удержанию клиентов.

- **Соблюдение нормативных требований**:

Соответствие стандартам: Возможность быстрого обновления прошивки позволяет оперативно адаптироваться к изменениям нормативных требований и стандартов безопасности.

Документирование и отчетность: Системы мониторинга обеспечивают ведение журналов и отчетность, что может быть полезно для соответствия внутренним и внешним аудитам.

- **Улучшение пользовательского опыта**:

Персонализация и настройки: Возможность удаленного управления настройками позволяет предлагать пользователям персонализированный опыт, адаптированный к их индивидуальным предпочтениям.

Новые функции и улучшения: Регулярные обновления могут вводить новые функции и улучшения, что сохраняет интерес пользователей и повышает их удовлетворенность.

Внедрение такой системы создает конкурентные преимущества, улучшая качество продукта, безопасность, операционную эффективность и, в конечном счете, увеличивая доходы и удовлетворенность клиентов.

1.2.2. Определение проблемы

Проблема описаны в табл. 1.2 – 1.4

Таблица 1.2

Проблема	Печальный пользовательский опыт взаимодействия с устройствами
Затрагивает	Пользователей.
Ее следствием является	Малый объем продаж, неконкурентный продукт на рынке.
Успешное решение	Разработка распределенной системы мониторинга устройств

Определение проблемы

Таблица 1.3

Проблема	Необходимость обновления прошивки устройств.
Затрагивает	Заказчика.
Ее следствием является	Для решения возникающих дефектов нужно отправлять технического специалиста или другое устройство с нужной прошивкой.
Успешное решение	Обеспечение удаленного обновления прошивки для устранения возникающих проблем

Определение проблемы

Таблица 1.4

Определение проблемы

Проблема	Заказчик распространяет свой продукт без ведома организации разработчика устройств
Затрагивает	Разработчика устройства
Ее следствием является	Нарушение договоренности, перепродажа устройств без предупреждения заказчика, кража технологий.
Успешное решение	Внедрение системы, которая позволяет разработчику отслеживать использование устройств в режиме реального времени, собирая данные о местоположении и состоянии каждого устройства.

1.2.3 Определение позиции изделия

Позиция изделия описана в табл. 1.5.

Таблица 1.5

Определение позиции изделия

Для	Заказчика и поставщика
Которым	Требуется получать информацию от устройства и иметь возможность удаленного управления настройками устройства, обновления прошивки, блокировки устройства
Название продукта	Программные средства для распределенной системы мониторинга технического состояния экзомассажеров
Которые	Используют современные интернет-технологии
В отличие от	Существующих аналогов
наш продукт	Предоставляет возможность удаленного отслеживания и управления каждым из зарегистрированных в системе устройств.

1.3. Описания пользователей

1.3.1. Сведения о пользователях

В системе существуют следующие основные виды акторов:

Оператор – получает всю доступную информацию о техническом состоянии устройств(-а), может удаленно настроить мощность, отключать или включать устройство, регистрировать несколько устройств в своем личном кабинете, а также получить техническую поддержку от сотрудников системы мониторинга.

Администратор – сотрудник системы, который должен регистрировать все устройства, отслеживать их статус, оказывать техническую поддержку пользователям через встроенный онлайн-чат, в случае необходимости блокировать работу устройств (неуплата подписки, нарушение договора и техники безопасности).

Устройство – отправляет данные о техническом состоянии на портал мониторинга, а также получает команды для настройки экзомассажера.

1.3.2. Пользовательская среда

Пользователи сайта это в основном люди в возрасте 18-70 лет. Возможно, пользователи плохо владеют компьютером, для этого необходимо разработать интуитивно простой и удобный интерфейс сайта. Для удобного поиска журналов необходимо реализовать поиск по сайту.

Администратор сайта должен уметь на базовом уровне пользоваться компьютером. Панель администратора тоже должна быть интуитивно понятной и удобной в управлении, иметь возможность управлять сайтом без участия программиста

1.3.3. Профили пользователей

Профили пользователей представлены в табл. 1.6-1.8

Таблица 1.6

Профиль пользователя оператор

Типичный представитель	Оператор
Описание	Зарегистрированный пользователь системы, который подписывается на устройство или группу устройств и может отслеживать их техническое состояние, а также удаленно управлять устройствами.
Тип	Пользователь
Ответственности	Контроль за техническим состоянием устройства.
Критерий успеха	В случае обнаружения проблем обратиться в техподдержку через онлайн-чат.

Таблица 1.7

Профиль пользователя администратор

Типичный представитель	Администратор
Описание	Сотрудник, наделенный правами на добавление, редактирование, удаление информации пользователей и устройств, регистрация пользователей и устройств.
Тип	Пользователь
Ответственности	Техподдержка пользователей, добавление, блокировка и разблокировка устройств.
Критерий успеха	Получение необходимой информации о состоянии устройств и проблемах пользователей, оперативное решение возникших задач.

Таблица 1.8

Профиль внешней сущности - устройство

Типичный представитель	Устройство мониторинга
Описание	Устройство, которое отвечает за передачу и прием данных.
Тип	Экзомассажер
Ответственности	Сбор данных о техническом состоянии и их дальнейшая отправка, прием команд для настройки устройства.
Критерий успеха	Исправность устройства.

1.3.4. Ключевые потребности пользователей

Ключевые потребности пользователей в контексте разработки программных средств для распределенной системы мониторинга технического состояния экзомассажеров включают:

- Надежность и стабильность: Пользователи ожидают, чтобы система была надежной и стабильной, так как от нее зависит эффективность и безопасность использования экзомассажеров.
- Простота использования: Пользователи, включая желают, чтобы система была интуитивно понятной и легкой в использовании, даже без специальной подготовки.
- Оперативность: важно, чтобы система обеспечивала оперативное обнаружение и реагирование на любые технические проблемы с экзомассажерами, чтобы минимизировать время простоя и финансовые потери.
- Доступность: Пользователи ожидают, чтобы система была доступной из любой точки сети, что позволило бы им мониторить и настраивать состояние экзомассажеров в реальном времени, даже на удаленных объектах или из домашней среды.
- Безопасность данных: Данные являются конфиденциальными, поэтому пользователи ожидают, чтобы система обеспечивала высокий уровень защиты данных и соответствовала стандартам безопасности и конфиденциальности.

1.4. Краткий обзор изделия

1.4.1. Контекст использования системы

Контекст использования разработки программных средств для распределенной системы мониторинга технического состояния экзомассажеров включает следующие сценарии:

- Медицинские учреждения: Больницы, клиники, реабилитационные центры и другие медицинские учреждения могут использовать эту систему для непрерывного мониторинга и обслуживания экзомассажеров, которые используются в процессе реабилитации пациентов после травмы, операции или для облегчения боли.
- Домашнее использование: Пациенты, которые имеют экзомассажеры для лечения хронических болей или для поддержания физической активности в домашних условиях, могут воспользоваться этой системой для мониторинга состояния своего оборудования и своего здоровья.

- Производители медицинского оборудования: Компании, производящие и поставляющие экзомассажеры на рынок, могут интегрировать разработанное программное обеспечение в свои устройства как часть улучшенной службы поддержки и обслуживания клиентов.
- Техническая поддержка и обслуживание: Специалисты по техническому обслуживанию могут использовать эту систему для удаленного мониторинга и управления состояниям экзомассажеров.
- Научные исследования: Ученые и специалисты в области медицинской техники могут использовать данные, собранные этой системой, для анализа производительности и эффективности экзомассажеров, а также для улучшения их конструкции и функциональности.

1.4.2. Сводка возможностей

Возможности программы описаны в табл. 1.9.

Таблица 1.9

Сводка возможностей

Выгоды заказчика	**Поддерживающие возможности**
Непрерывный мониторинг	Система обеспечивает непрерывный мониторинг технического состояния экзомассажеров в режиме реального времени.
Оповещения и предупреждения	Пользователи получают оповещения и предупреждения в случае обнаружения технических проблем.
Удаленное управление	Возможность удаленного управления экзомассажерами.
Аналитика и отчетность	Система предоставляет аналитическую информацию и отчеты о состоянии и производительности.
Графический интерфейс	Интуитивно понятный графический интерфейс для удобного мониторинга и управления системой.
Масштабируемость	Возможность обновления прошивки, которая помогает масштабировать систему для работы с большим количеством устройств и увеличения выполняемого функционала.
Защита данных	Обеспечение высокого уровня защиты данных и

	конфиденциальности в соответствии с нормативами.

1.4.3. Предположения и зависимости

Программа может быть использована при наличии установленного программного обеспечения – веб-браузера(Opera, Mozilla Firefox, Google Chrome) и стабильного интернет соединения В случае увеличения функциональности - добавления новых сервисов, подключение карт, существенных изменений в структуре не планируется.

1.5. Возможности продукта

1.5.1. Регистрация пользователей

Пользователь может зарегистрироваться в системе и получает доступ к добавлению устройств в личном кабинете по уникальному идентификатору каждого из устройств.

1.5.2. Сбор результатов мониторинга

Система собирает всю доступную информацию и предоставляет возможность рассмотреть результаты мониторинга в своем личном кабинете для зарегистрированных пользователей и администраторов.

1.5.3. Сохранение результатов мониторинга

Сохранение полученных данных об устройстве, результатах мониторинга, пользовательских данных (учетная запись, пароль, контактная информация) в базе данных.

1.5.4. Предоставление информации пользователям

Пользователи могут рассмотреть всю доступную информацию об устройстве, результатах мониторинга в удобном графическом виде, также пользователи имеют доступ к чату с сотрудниками для технической консультации.

1.5.5. Удаленное обновление прошивки устройств

Устройства имеют возможность обновить прошивку и в случае необходимости или добавления нового функционала

1.5.6. Блокировка и разблокировка устройств по системе подписки

Каждое устройство имеет статус блокировки, если одним из условий распространения будет система подписки.

1.6. Ограничения применения

Для работы с системой мониторинга необходимо:

1. Доступ в интернет
2. Зарегистрироваться на портале
3. Добавить устройство в личном кабинете
4. Любой браузер из указанных ниже в системных требованиях

1.7. Показатели качества

1.7.1. Применимость

- Время, необходимое для обучения обычных пользователей – 1 рабочий день (8 часов), для обучения продвинутых пользователей (администраторы) – до 5 часов
- Время отклика для типичных задач – не более 5 секунд, для сложных задач – не более 20 секунд.

1.7.2. Надежность

- Доступность – время, затрачиваемое на обслуживание системы не должно превышать 3% от общего времени работы.
- Максимальная норма ошибок или дефектов – 1 ошибка на тысячу строк кода.

1.8. Другие требования к изделию

1.8.1. Системные требования

Система должна соответствовать всем стандартам популярных веб-браузеров:

- Google Chrome
- Opera
- Firefox
- Microsoft Edge
- Yandex.Браузер

1.9. Требования к документации

1.9.1. Руководство программиста

Разработанные программные средства должны иметь руководство программиста по установке, развертыванию, конфигурированию, чтобы в случае необходимости другая команда разработки смогла произвести вышеуказанные действия самостоятельно.

Глава 2. Разработка требований

2.1 Общие сведения

2.1.1. Назначение документа

Назначение документа «Разработка программных средств для распределенной системы мониторинга технического состояния экзомассажеров» - определение требований к разрабатываемым программным средствам и согласование их с заказчиком. Документ включает детальное видение продукта, формализованные требования и описание объектов систем. Данный документ является достаточным для программных средств.

2.1.2. Цели создания системы, решаемые задачи и область применения

Цели создания системы:

1. Обеспечение непрерывного мониторинга технического состояния экзомассажеров.
2. Предоставления возможности удаленного управления экзомассажерами.
3. Предоставление информации пользователям в удобном виде.
4. Повышение эффективности работы устройств за счет анализа полученных данных, которые примут участие в обновлении прошивок устройств.
5. Обеспечение надежности использования системы.
6. Возможность использования экзомассажеров через систему подписки.

Решаемые задачи:

1. Разработка системы мониторинга ключевых технических параметров экзомассажеров с возможностью удаленного доступа для пользователей.
2. Реализация алгоритмов анализа данных для выявления аномалий и формирования рекомендаций по предотвращению отказов.
3. Разработка интерфейса для визуализации результатов мониторинга и управления системой через веб-портал для пользователей.
4. Интеграция механизмов оповещения и автоматического управления для оперативного информирования и вмешательства пользователей при необходимости – в случае нарушения правил и договоренностей, произойдет блокировка устройства.

Документ акцентирует внимание на возможностях, необходимых совладельцам и целевым пользователям, и на том, почему эти потребности существуют.

2.2. Общее описание

2.2.1. Позиционирование программного продукта

Позиционирование программных средств для мониторинга технического состояния экзомассажеров с возможностью удаленного управления, блокировки и обновления прошивки должно учитывать уникальные преимущества, потребности целевой аудитории и отличительные черты, которые делают продукт конкурентоспособным на рынке.

Разрабатываемая система мониторинга технического состояния экзомассажеров включает в себя устройства для управления и с соответственными параметрами, а также аппаратно-программную платформу для обработки результатов. В программно-сервисную часть входит агрегация с открытых источников информации, обработка информации и представление этой информации в понятном пользователю виде.

2.2.2. Функциональность программного продукта

Основные функции данной системы:

- Сбор результатов мониторинга с устройств.
- Удаленное управление настройками устройств.
- Контроль версий и обновление прошивок устройств.
- Регистрация, редактирование, удаление пользователей системы – обычный пользователь, администратор, устройства.
- Возможность подписаться на устройство или группу устройств
- Информативный и удобный интерфейс пользователей, представленный в виде: интерактивной карты с местоположением устройств, графики за период, актуальные данные в виде таблицы.
- Онлайн-чат поддержки пользователей.
- Добавление, редактирование, удаление информации на сайт системы мониторинга при помощи разработанного функционала для администрации.

2.2.3. Пользовательские характеристики

Пользователи системы мониторинга — это люди, которые заинтересованы в получении результатов мониторинга параметров устройств.

2.2.4. Ограничения

- Время на разработку системы
- Информационные данные
- Аппаратно-программные средства

2.2.5. Предположения и зависимости

Для использования функционала системы пользователю необходимо иметь доступ к интернету и смартфон или компьютер.

Для реализации проекта должны быть данные с устройств и постоянный доступ к сети (плата оснащена дополнительным источником связи «4G модем», если устройство не будет подключено к WIFI сети).

Система является достаточно гибкой для дополнения различными сервисами, функционалом и услугами, также в планах остается распространение по системе подписки на сервисы (ежемесячная или годовая оплата), для которой был разработан функционал, позволяющий заблокировать работу устройства.

2.3. Специфические требования

2.3.1 Требования к интерфейсам

- Интерфейс пользователя должен быть простым и понятным, для всех не вызывая вопросов, связанных с ним.
- Должен полностью соответствовать разработанной схеме системы
- Должен соответствовать предоставленному стилю от Заказчика
- Должен делаться в расчете на оптимальное разрешение 1280px по ширине.
- Должен правильно работать в основных веб-браузерах (Google Chrome, Opera, Firefox, Microsoft Edge, Yandex Browser) и быть адаптивным.
- Должен включать в себя безопасные шрифты и соответствовать.
- Должен быть продуман с точки зрения возможности последующей программной реализации задуманного.
- Текстовые материалы должны быть читабельными
- Используемые графические элементы должны быть легальными, понятными и однозначными

2.3.2 Функциональные требования

На рис. 2.1-2.3 представлены функции системы в виде USE-CASE диаграммы. Выявлены трое Акторов: Оператор, Администратор, Устройство.

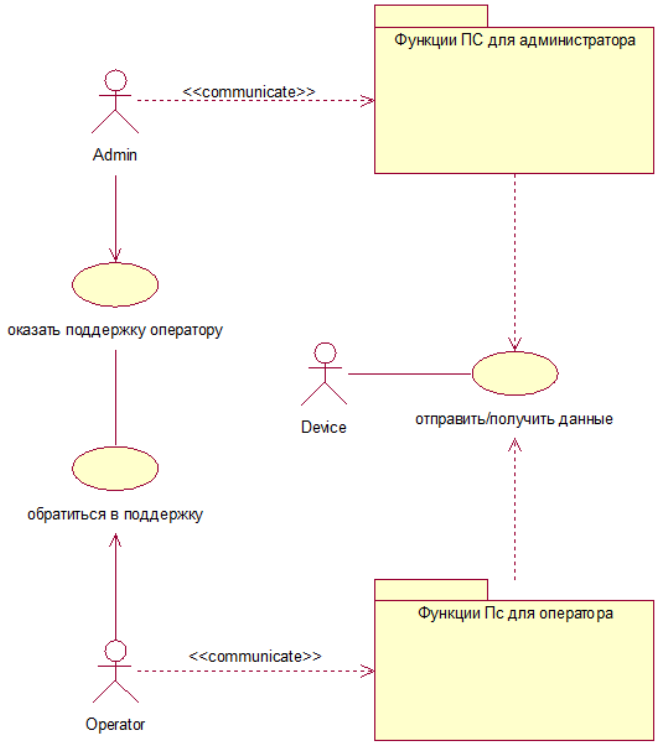


Рисунок 2.1. Пакеты вариантов использования

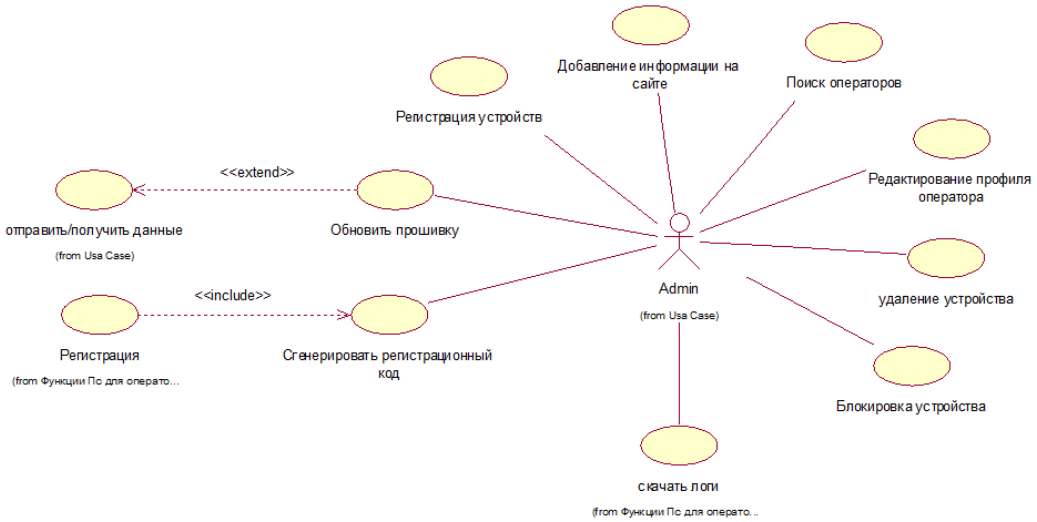


Рисунок 2.2. Детализация пакета функции ПС для администратора

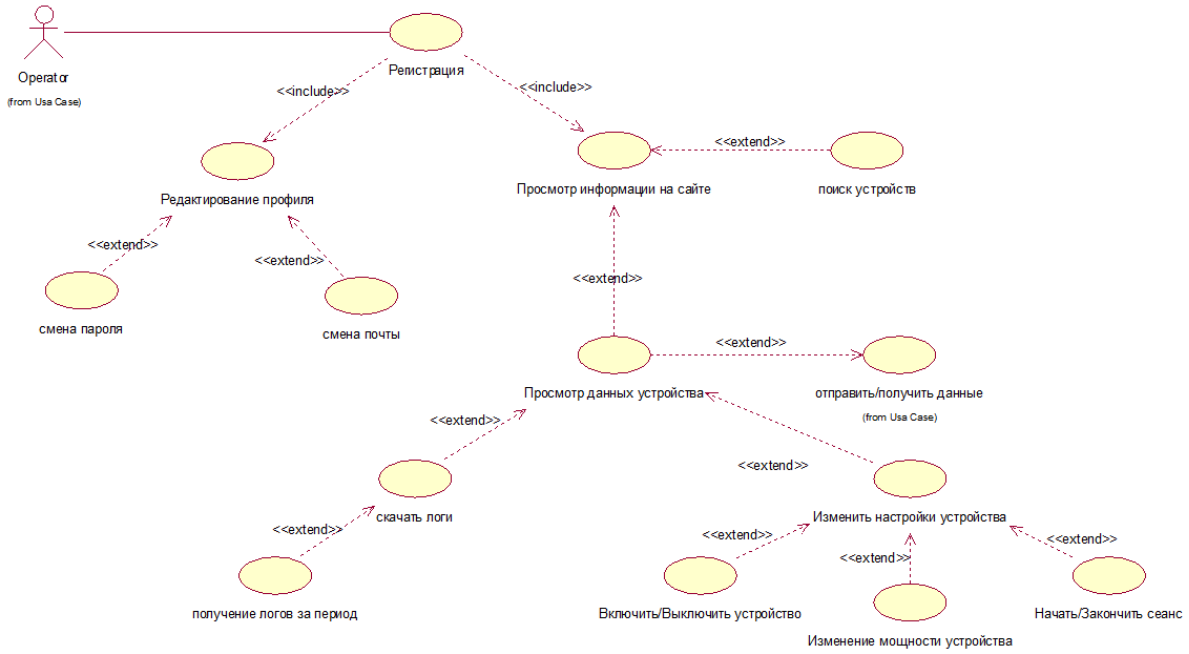


Рисунок 2.3. Детализация пакета функции ПС для оператора

Основные функции Администратора:

- Обновление все доступной информации: добавление, редактирование, удаление данных на сайте и базе данных.
- Предоставление страницы регистрации для гостей при помощи QR или ссылки.
- Отслеживание сообщений от пользователей и оказание помощи.
- Регистрация устройств.
- Контроль версий и обновление прошивок устройств.
- Управление настройками устройств при необходимости.
- Блокировка и разблокировка работы устройств.

Основные функции оператора:

- Зарегистрироваться.
- Добавить устройство или группу устройств.
- Получение результатов мониторинга.
- Управление настройками устройства.

Основные функции устройств:

- Отправка данных о техническом состоянии.
- Получение команд для управления настройками.

Таблица 2.1.

Выявленные акторы системы

Актёр	*Краткое описание*
Оператор	Зарегистрированный пользователь, обладающий правами на добавление устройства в избранное и далее управлять его настройками.
Администратор	Пользователь, наделенный правами добавление и редактирование информации на сайте и в базе данных.
Устройство	Плата модели ESP-8266, которое занимается сбором данных о техническом состоянии экзомассажеров и передает их на сервер для дальнейшего предоставления конечному пользователю и принимает команды для управления настройками экзомассажеров.

2.3.3 Описание вариантов пользования

В табл. 2.2. предоставлен реестр вариантов использования Web-платформы.

Таблица 2.2.

Реестр вариантов использования

Код	**Основной актер**	**Наименование**	**Формулировка**
A1	Администратор	Редактирование информации на сайте	Администратор через админ панель будет редактировать разделы и категории на сайте
A2	Администратор	Регистрация устройств	Администратор через админ панель будет регистрировать новые устройства
A3	Администратор	Обновить прошивку	Администратор через админ панель будет обновлять прошивку устройств
A4	Администратор	Добавление операторов	Администратор через админ панель будет добавлять новых операторов
A5	Администратор	Удаление устройств	Администратор через админ панель будет удалять устройства
A6	Администратор	Блокировка устройства	Администратор через админ панель будет блокировать устройства

A7	Администратор	Скачивание логов	Администратор через админ панель будет скачивать логи устройств
B1	Оператор	Регистрация	Оператор может зарегистрироваться на сайте для дальнейших доступных действий
B2	Оператор	Редактирование профиля	Оператор может редактировать свой профиль, включая смену пароля и почты
B3	Оператор	Просмотр информации на сайте	Оператор может просматривать информацию на сайте
B4	Оператор	Поиск устройств	Оператор может искать устройства по различным параметрам
B5	Оператор	Просмотр данных устройства	Оператор может просматривать данные, связанные с устройствами
B6	Оператор	Отправить/получить данные	Оператор может отправлять и получать данные с устройств
B7	Оператор	Изменить настройки устройства	Оператор может изменять настройки устройства, включая включение/выключение и изменение мощности
B8	Оператор	Начать/Закончить сеанс	Оператор может начинать и заканчивать сеансы использования устройства

2.3.4 Характеристики пользователей

Оператор – это в основном люди в возрасте 20-70 лет. Возможно, пользователи плохо владеют компьютером и телефоном.

Администратор – Продвинутый пользователь компьютера

Таблица 2.3.

Характеристики пользователей

Код	**Основной актер**	**Наименование**	**Формулировка**
A1	Администратор	Редактирование информации на сайте	Администратор через админ панель будет редактировать разделы и категории на сайте

Основное действующее лицо: Администратор

Другие участники прецедента: отсутствуют

Связи с другими вариантами использования: "A3. Изменить категорию / раздел"

"A4 Добавить категорию / раздел"

Краткое описание: Администратор через админ панель будет добавлять и менять категории и разделы на сайт

Код	**Основной актер**	**Наименование**	**Формулировка**
A2	Администратор	Добавление операторов	Администратор через админ панель будет добавлять новых операторов

Основное действующее лицо: Администратор

Другие участники прецедента: отсутствуют

Связи с другими вариантами использования:

Краткое описание: Администратор через админ панель будет добавлять новых операторов

Код	**Основной актер**	**Наименование**	**Формулировка**
B1	Оператор	Регистрация	Оператор может зарегистрироваться на сайте для дальнейших доступных действий

Основное действующее лицо: Оператор

Другие участники прецедента: отсутствуют

Связи с другими вариантами использования: "B2. Редактирование профиля"

"B3. Просмотр информации на сайте"

Краткое описание: Оператор может зарегистрироваться на сайте для дальнейших доступных действий

Код	Основной актер	Наименование	Формулировка
B2	Оператор	Регистрация	Оператор может редактировать свой профиль, включая смену пароля и почты

Основное действующее лицо: Оператор

Другие участники прецедента: отсутствуют

Связи с другими вариантами использования: "B1. Регистрация"

Краткое описание: Оператор может редактировать свой профиль, включая смену пароля и почты

Код	Основной актер	Наименование	Формулировка
B3	Оператор	Регистрация	Оператор может просматривать информацию на сайте

Основное действующее лицо: Оператор

Другие участники прецедента: отсутствуют

Связи с другими вариантами использования: "B1. Регистрация"

"B2. Редактирование профиля

Краткое описание: Оператор может просматривать информацию на сайте

Код	Основной актер	Наименование	Формулировка
B4	Оператор	Поиск устройств	Оператор может искать устройства по различным параметрам

Основное действующее лицо: Оператор

Другие участники прецедента: отсутствуют

Связи с другими вариантами использования: "B3. Просмотр информации на сайте"

"B5. Просмотр данных устройства"

Краткое описание: Оператор может искать устройства по различным параметрам

Код	Основной актер	Наименование	Формулировка
В5	Оператор	Просмотр данных устройства	Оператор может просматривать данные, связанные с устройствами

Основное действующее лицо: Оператор

Другие участники прецедента: отсутствуют

Связи с другими вариантами использования: "В4. Поиск устройств" "В6. Отправить/получить данные"

Краткое описание: Оператор может просматривать данные, связанные с устройствами

Код	Основной актер	Наименование	Формулировка
В6	Оператор	Отправить/получить данные	Оператор может отправлять и получать данные с устройств

Основное действующее лицо: Оператор

Другие участники прецедента: отсутствуют

Связи с другими вариантами использования: "В5. Просмотр данных устройства" "В7. Изменить настройки устройства"

Краткое описание: Оператор может отправлять и получать данные с устройств

Код	Основной актер	Наименование	Формулировка
В7	Оператор	Изменить настройки устройства	Оператор может изменять настройки устройства, включая включение/выключение и изменение мощности

Основное действующее лицо: Оператор

Другие участники прецедента: отсутствуют

Связи с другими вариантами использования: "В6. Отправить/получить данные" "В8. Начать/Закончить сеанс"

Краткое описание: Оператор может изменять настройки устройства, включая включение/выключение и изменение мощности

Код	Основной актер	Наименование	Формулировка
В	Оператор	Начать/Закончить сеанс	Оператор может отправлять и получать данные с устройств

Основное действующее лицо: Оператор

Другие участники прецедента: отсутствуют

Связи с другими вариантами использования: "В6. Отправить/получить данные"

"В8. Начать/Закончить сеанс"

Краткое описание: Оператор может изменять настройки устройства, включая включение/выключение и изменение мощности

2.3.5 Требования к производительности

- Скорость загрузки сайта не должна загружать компьютер пользователя скачиванием стилей и объемных данных.
- Сайт должен одинаково отображаться во всех основных веб-браузерах, в наименее распространенных версиях веб-браузеров допустимы небольшие различия в оформлении блоков, без потери их структуры и функциональности.
- Сайт и все его информационные материалы должны быть доступны всем пользователям.
- Административные функции должны быть доступны только определенной группе лиц, даже если это сотрудник компании.

2.3.6 Логические требования к базе данных

- Целостность базы данных – полнота и непротиворечивость данных
- Многократное использование данных
- Быстрый поиск, получение информации при пользовательских запросах
- Простота обновления данных
- Уменьшений избыточности данных
- Защита данных от несанкционированного изменения и удаления
- Сохранение всех изменений структуры базы данных, возможность отката изменений

2.3.7 Ограничения дизайна

- Должен использоваться простой интуитивно понятный пользователям интерфейс.
- Сайт должен работать на устройствах с различными устройствами и иметь адаптивность масштабирования.

- Сайт должен правильно открываться на основных веб-браузерах, указанных в требованиях к интерфейсу.

2.3.8 Ограничения проектирования

Данная система не имеет ограничений в выборе шаблона проектирования.

2.3.9 Соответствие стандартам

- Разработка данной системы должна соответствовать современным стандартам проектирования программного обеспечения.
- При передаче пользовательских данных должен использоваться защищенный протокол HTTPS.
- На сервере есть действующий SSL-сертификат, подписанный удостоверяющим центром.
- Каждая страница сайта имеет уникальный URL.
- Страницы имеют понятный URL.

2.3.10 Атрибуты программного продукта

Надежность

- Программный продукт отвечает современным требованиям разработки.
- Возможность отката изменений на предыдущую версию.
- Восстановление пользовательских и данных мониторинга.

Доступность

- Данный программный продукт доступен для любого пользователя
- Время, затрачиваемое на обработку запросов и получения ответа не должно превышать 5 секунды.
- Среднее время выполнения запроса – 10 секунд.
- Максимальное время выполнения – 13-20 секунд.

Безопасность

- В случае ошибок системы, здоровью пользователей ничего не угрожает.
- Хранимые данные как пользовательские, так и результаты мониторинга защищены от вмешательства на удаление из базы данных и несанкционированное изменение.

Поддерживаемость

- Все модульные части системы разбиты на классы. Поддержка может обеспечиваться путем добавления новых классов, методов, интерфейсов и представлений.

Глава 3. Проектирование и конструирование ПО

3.1. Разработка диаграммы классов

Диаграмма классов (class diagram) используется для представления статической структуры модели системы в терминологии классов объектно-ориентированного программирования. Диаграмма классов может отражать, в частности, различные отношения между отдельными объектами тематической области, такими как объекты и подсистемы, а также описывает их внутреннюю структуру и типы отношений. Эта диаграмма не показывает информацию о временных аспектах работы системы (рис. 3.1). С этой точки зрения диаграмма классов является дальнейшим развитием концептуальной модели проектируемой системы

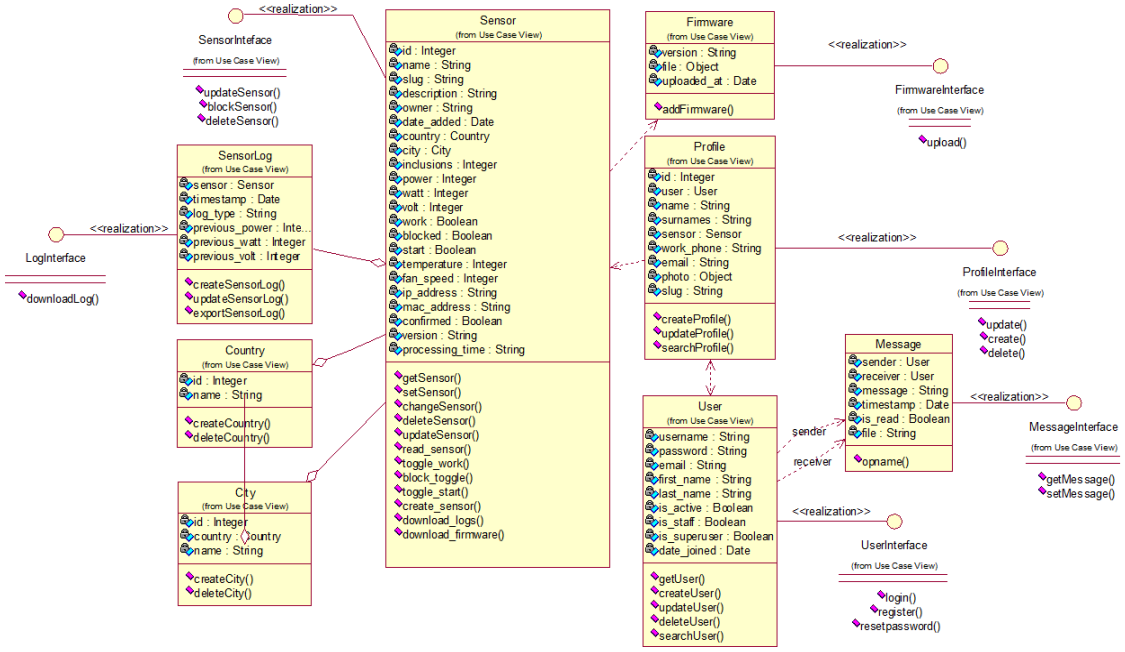


Рисунок 3.1. Диаграмма классов

3.1.1 Разработка класса Sensor

- **Sensor** – это класс описывающий датчик:
- **id**: AutoField – уникальный идентификатор датчика.
- **name**: CharField – имя датчика
- **slug**: SlugField – уникальный идентификатор URL для датчика
- **description**: TextField – описание датчика
- **owner**: CharField – владелец датчика.
- **date_added**: DateTimeField – дата добавления датчика
- **country**: ManyToManyField – страны, к которым относится датчик

- **city**: ManyToManyField – города, к которым относится датчик.
- **inclusions**: IntegerField – количество включений датчика.
- **power**: IntegerField – мощность датчика в процентах.
- **watt**: IntegerField – потребление мощности в ваттах (по умолчанию 0).
- **volt**: IntegerField – электрическое напряжение в вольтах (по умолчанию 0).
- **work**: BooleanField – состояние датчика (онлайн или нет, по умолчанию онлайн).
- **blocked**: BooleanField – заблокирован ли датчик (по умолчанию нет).
- **start**: BooleanField – запущен ли датчик (по умолчанию нет).
- **temperature**: FloatField – температура датчика (по умолчанию 0).
- **fan_speed**: IntegerField – скорость кулера (по умолчанию 0).
- **ip_address**: CharField – IP адрес датчика.
- **mac_address**: CharField – MAC адрес датчика.
- **confirmed**: BooleanField – подтвержден ли датчик администратором
- **version**: CharField – версия прошивки датчика.
- **processing_time**: FloatField – время обработки в миллисекундах.

Интерфейсы и их реализации

SensorInterface:

Реализуется классом Sensor.

Операции: updateSensor(), blockSensor(), deleteSensor().

3.1.2 Разработка класса SensorLog

SensorLog – это класс, описывающий лог записи для датчика с различными атрибутами.

Атрибуты класса SensorLog:

- **sensor**: ForeignKey – ссылка на класс Sensor, представляющая датчик, к которому относится эта запись лога. При удалении датчика все связанные записи лога также удаляются.
- **timestamp**: DateTimeField – временная метка записи лога, устанавливается автоматически при создании записи.
- **log_type**: CharField – тип записи лога.
- **previous_power**: IntegerField – предыдущая мощность датчика в процентах.
- **previous_watt**: IntegerField – предыдущее потребление мощности в ваттах.
- **previous_volt**: IntegerField – предыдущее электрическое напряжение в вольтах

Ассоциация: Один ко многим (один Sensor имеет много SensorLog).

Операции: createSensorLog(), updateSensorLog(), exportSensorLog().

Интерфейсы и их реализации

LogInterface:

Реализуется классом SensorLog.

Операции: downloadLog().

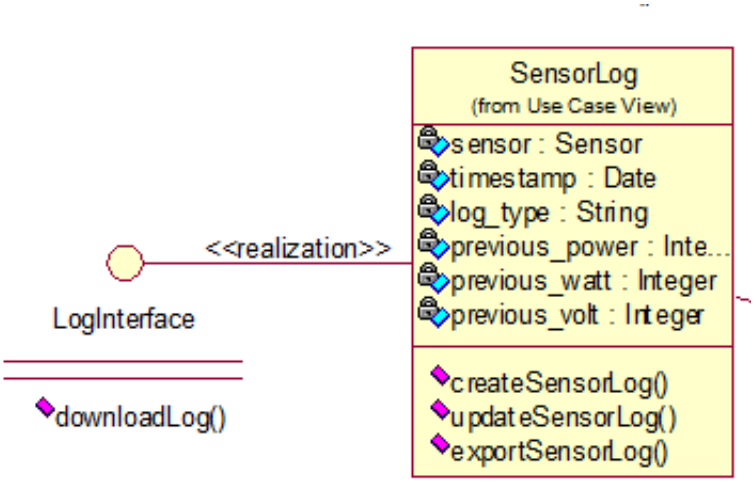


Рисунок 3.2. Диаграмма классов для «SensorLog».

3.1.3 Разработка класса Firmware

Firmware – это класс, описывающий прошивку для датчика с различными атрибутами:

Атрибуты класса Firmware:

- **version:** CharField – версия прошивки
- **file:** FileField – файл прошивки, загружаемый в директорию firmwares/
- **uploaded_at:** DateTimeField – время загрузки файла прошивки, устанавливается автоматически при создании записи

Ассоциация: Один ко многим (один Sensor может иметь много версий прошивок).

Операции: download_firmware().

Интерфейсы и их реализации

FirmwareInterface:

Реализуется классом Firmware.

Операции: upload().

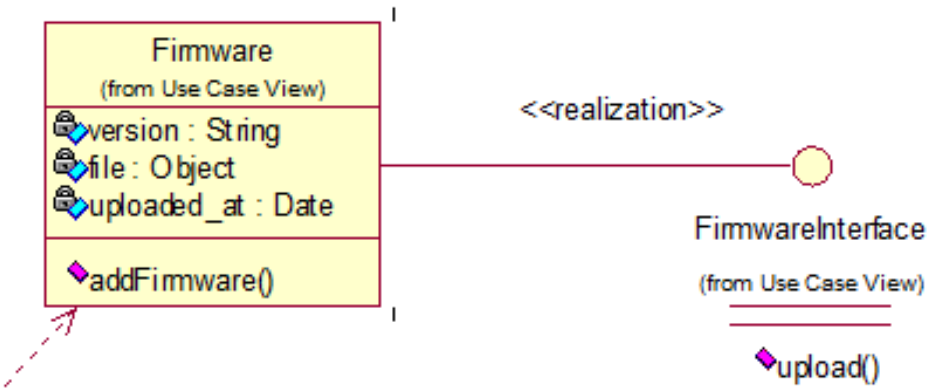


Рисунок 3.3. Диаграмма классов для «Firmware».

3.1.4 Разработка класса Firmware

Message – это класс, описывающий сообщение с различными атрибутами:

Атрибуты класса Message:

- **sender:** ForeignKey – ссылка на класс User, представляющая отправителя сообщения. При удалении пользователя все отправленные им сообщения также удаляются. Поле связано с пользователем с помощью параметра related_name=sender
- **receiver:** ForeignKey – ссылка на класс User, представляющая получателя сообщения. При удалении пользователя все полученные им сообщения также удаляются. Поле связано с пользователем с помощью параметра related_name=receiver
- **message:** CharField – текст сообщения
- **timestamp:** DateTimeField - временная метка отправки сообщения, устанавливается автоматически при создании записи
- **is_read:** BooleanField – статус прочтения сообщения

Message ассоциируется с:

- **User**:
  - Ассоциация: Один ко многим (один User может отправлять и получать много Message).
  - Атрибуты: sender, receiver.

User ассоциируется с:

- **Message**:
  - Ассоциация: Много к одному (много Message принадлежат одному User).
  - Операции: getUser(), createUser(), updateUser(), deleteUser(), searchUser().

Интерфейсы и их реализации

MessageInterface:

Реализуется классом Message.

Операции: getMessage(), setMessage().

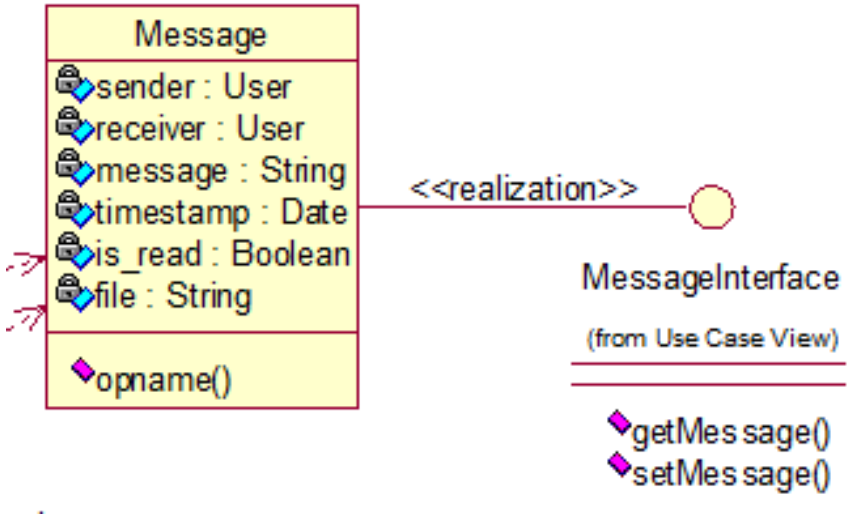


Рисунок 3.4. Диаграмма классов для «Message».

3.1.5 Разработка класса User

User – это класс отвечающий за реализацию зарегистрированных пользователей системы. Какие поля и методы реализованы для взаимодействия с данным классом.

Атрибуты класса User:

- **username: string –** имя пользователя.
- **first_name: string –** имя
- **last_name: string –** фамилия
- **email: string –** адрес электронной почты
- **Password: string –** пароль для входа
- **is_staff:** BooleanField – статус сотрудника.
- **is_active:** BooleanField – активный статус.
- **date_joined:** DateTimeField – дата регистрации.

Ассоциация: Много к одному (много Message принадлежат одному User).

Операции: getUser(), createUser(), updateUser(), deleteUser(), searchUser().

Интерфейсы и их реализации

UserInterface:

Реализуется классом User.

Операции: login(), register(), resetPassword().

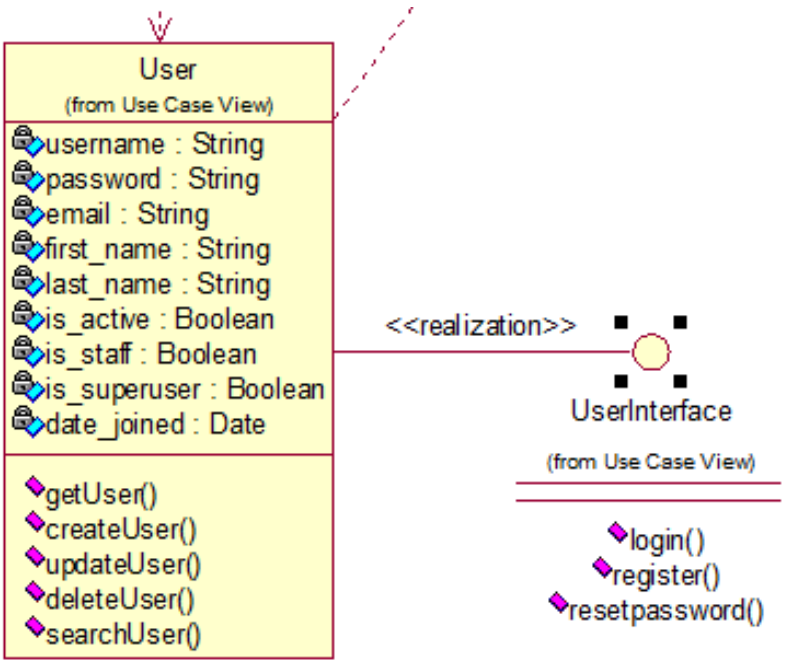


Рисунок 3.5. Диаграмма классов для «User».

3.1.6 Разработка класса **Profile**

Profile – это класс, описывающий профиль пользователя с различными атрибутами:
Атрибуты класса Profile:

- id**:** AutoField – идентификатор пользователя
- user**:** OneToOneField - ссылка на класс User, представляющая пользователя, которому принадлежит этот профиль. При удалении пользователя профиль также удаляется
- name**:** CharField – имя пользователя
- surnames**:** CharField – фамилия пользователя
- sensor**:** ManyToManyField – список датчиков, связанных с профилем пользователя
- work_phone**:** CharField – рабочий телефон пользователя
- email**:** EmailField– электронная почта пользователя
- photo**:** ImageField – фотография пользователя
- slug**:** SlugField – уникальный URL для профиля пользователя.
- last_activity**:** DateTimeField – время последней активности пользователя.

Ассоциация: Один ко многим (один Profile может иметь много Sensor).
Операции: createProfile(), updateProfile(), searchProfile().

Интерфейсы и их реализации

ProfileInterface:

Реализуется классом Profile.

Операции: update(), create().

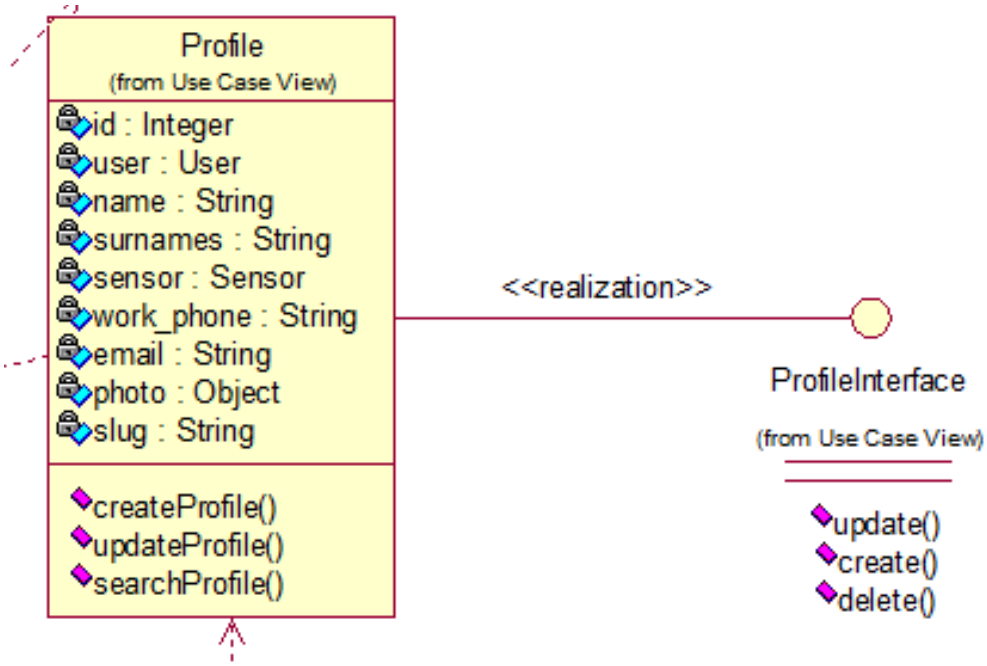


Рисунок 3.6. Диаграмма классов для «Profile»

3.2. Разработка диаграммы последовательности

Диаграмма последовательности — диаграмма, предназначенная. для представления взаимодействия между элементами модели программной системы в терминологии линий жизни и сообщений между ними.

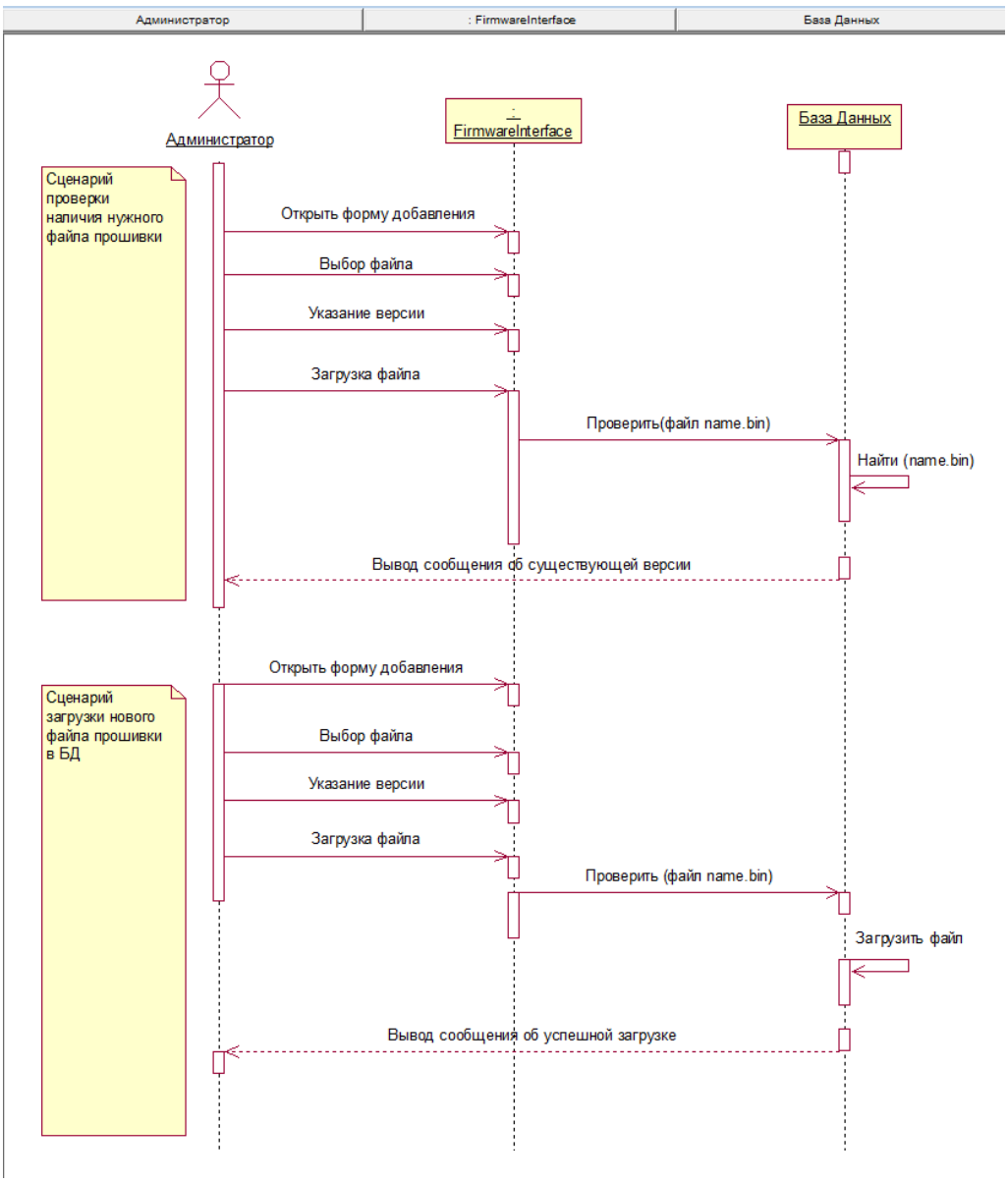


Рисунок 3.7. Диаграмма последовательности загрузки прошивки

Описание последовательности:

1. Администратор открывает форму загрузки прошивки.
2. Администратор выбирает файл в формате .bin.
3. Администратор вводит номер версии и загружает прошивку.
4. Система проверяет наличие такой прошивки в базе данных.
5. Файл загружается в базу данных.

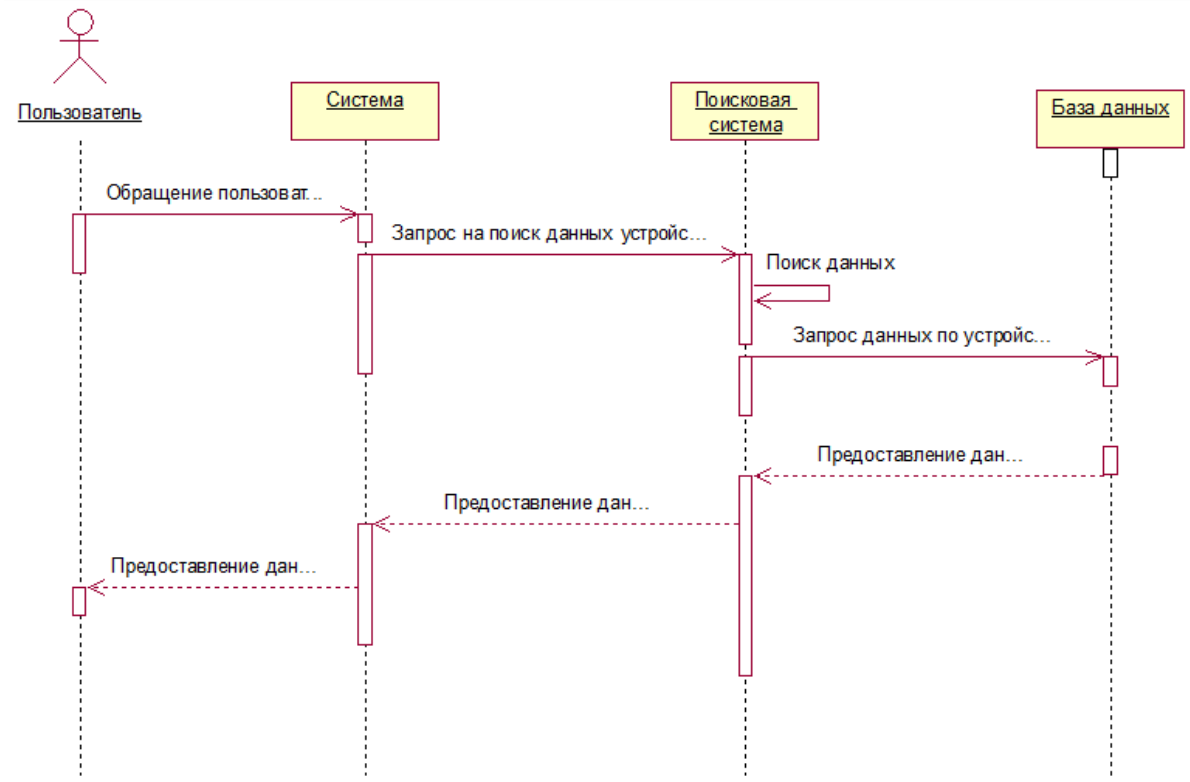


Рисунок 3.8. Диаграмма последовательности поиска

Описание последовательности

1. Пользователь вводит данные для поиска.
2. Системы получает данные.
3. Идет запрос данных в базе данных.
4. Происходит поиск данных по введем данным.
5. Данные передаются пользователю.

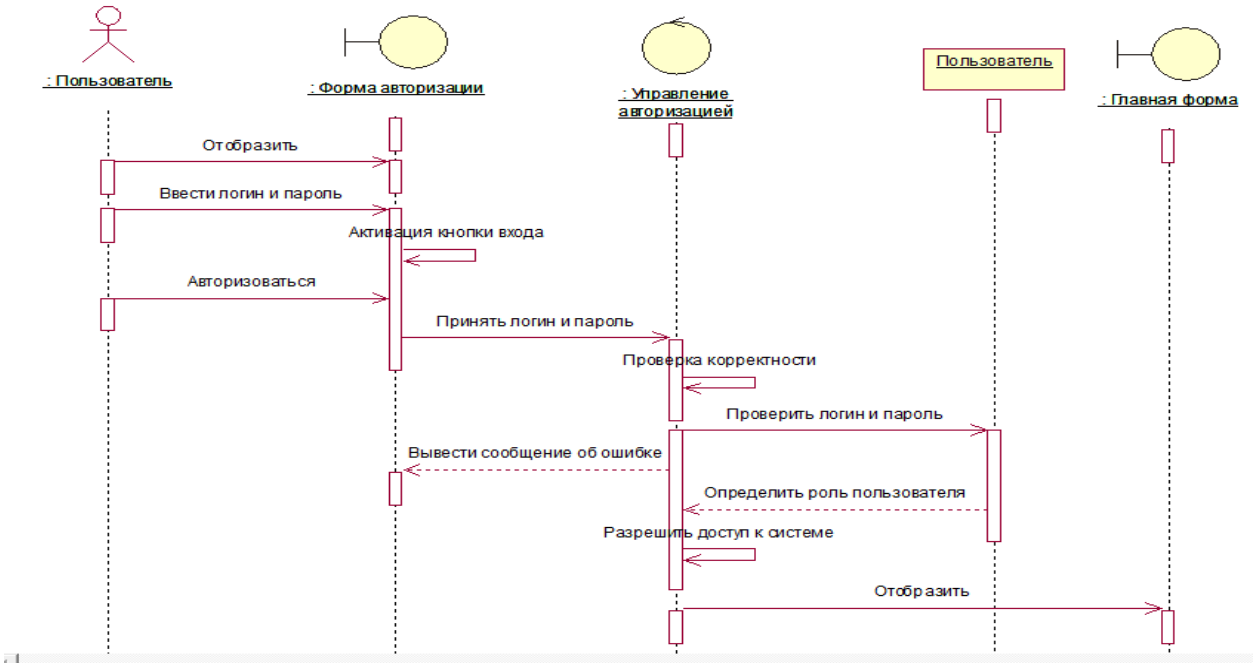


Рисунок 3.9. Диаграмма последовательности авторизации

Описание последовательности:

1. Отображение формы для входа.
2. Ввод логина и пароля в форму.
3. Проверить корректность данных.

   3.1. Вернуть сообщение об ошибке.
4. Определение роли пользователя.
5. Переход на главную страницу.

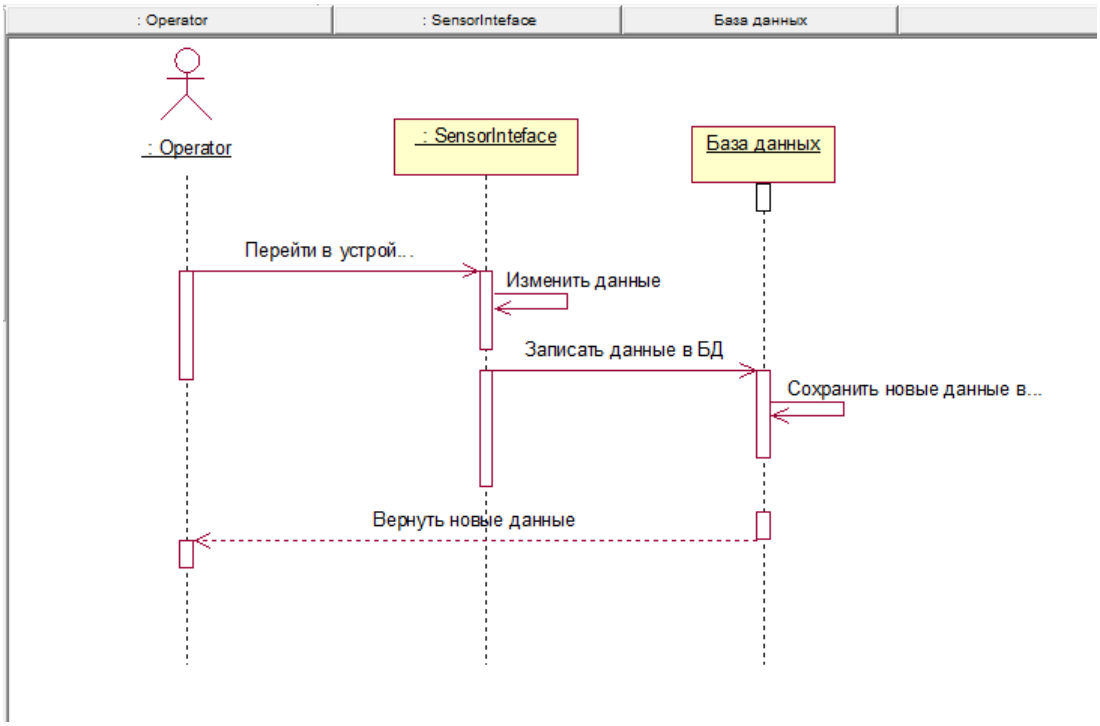


Рисунок 3.10. Диаграмма последовательности управления устройством

3.3. Разработка диаграммы компонентов

Диаграмма компонентов позволяет определить архитектуру разрабатываемой системы, установив зависимости между программными компонентами, в роли которых может выступать исходный, бинарный и исполняемый код. Пунктирные стрелки, соединяющие модули, показывают отношения взаимозависимости, аналогичные тем, которые имеют место при компиляции исходных текстов программ. Основными графическими элементами диаграммы компонентов являются компоненты, интерфейсы и зависимости между ними.

Диаграмма компонентов разрабатывается для следующих целей:

- Визуализации общей структуры исходного кода программной системы.
- Спецификации исполнимого варианта программной системы.
- Обеспечения многократного использования отдельных фрагментов программного кода.
- Представления концептуальной и физической схем баз данных.

При разработке программных компонентов были выявлены следующий компоненты диаграммы.

- **Client** – клиентская среда, где пользователь контактирует с предоставляемыми системой функционалом.

- **Views** – Компоненты, отвечающие за обработку запросов и возврат соответствующих ответов.
- **Templates** – Компоненты, используемые для рендеринга данных, предоставленных представлениями, в HTML.
- **Models (ORM) –** Компоненты, представляющие данные и бизнес-логику.
- **Database –** база данных, куда записываются все приходящие данные.

Таким образом, все взаимосвязи между основными компонентами продемонстрированы на рисунке 3.11

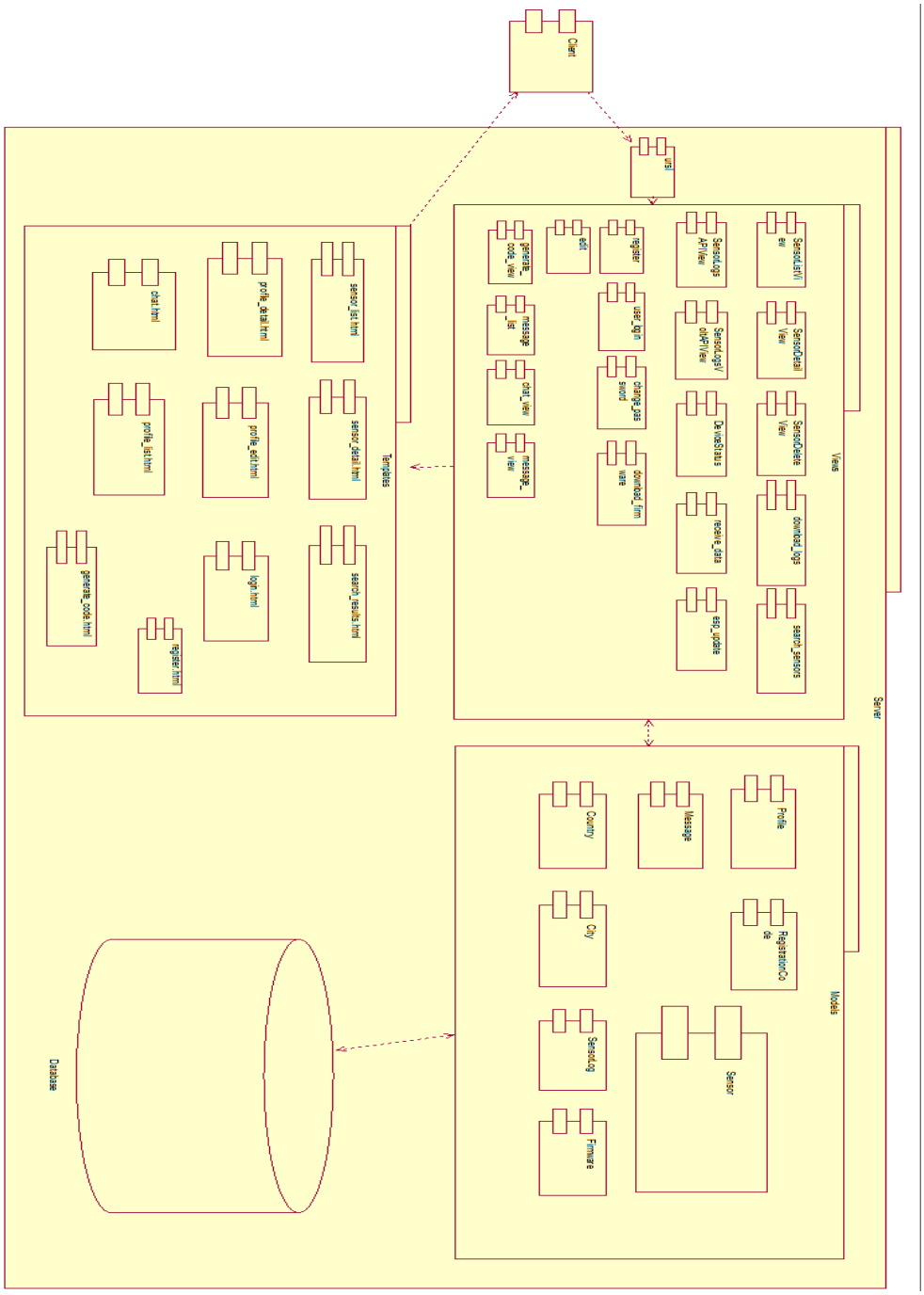


Рисунок 3.11. Диаграмма компонентов системы.

3.4. Определение классов-сущностей

Класс - сущность содержит информацию, которая должна храниться постоянно и не уничтожается с уничтожением объектов данного класса или после прекращения работы моделируемой системы. Этот класс соответствует отдельной таблице базы данных. В этом случае его атрибуты являются полями таблицы, а операции – присоединенными или хранимыми процедурами. Этот класс только принимает сообщения от других классов модели.

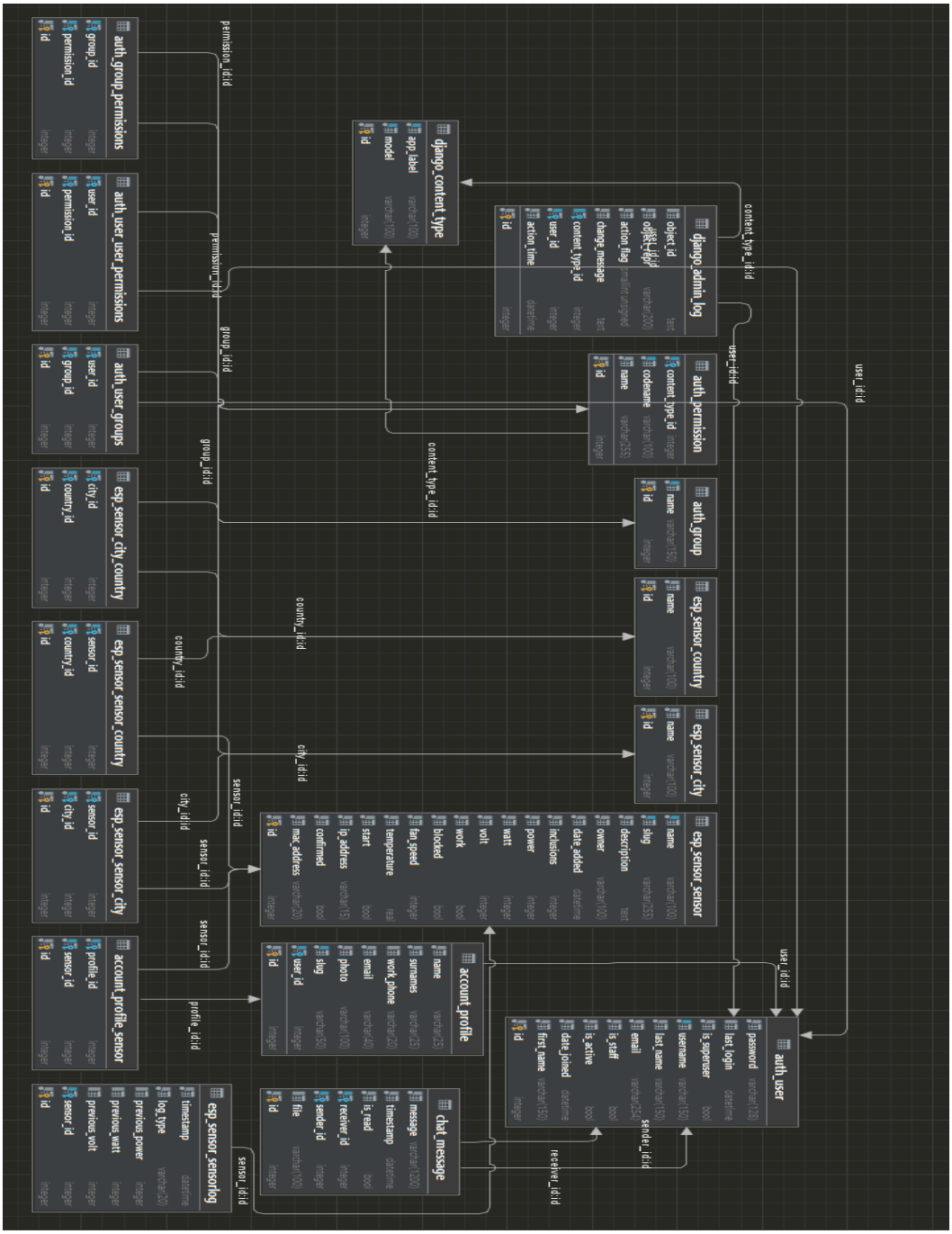


Рисунок 3.12. Классы-сущности и связи между ними

Глава 4. Разработка тестов и тестирование программного продукта

4.1. Разработка плана тестирования

4.1.1 Введение

Данный документ описывает план тестирования для системы «Разработка распределенной системы мониторинга технического состояния экзомассажеров» согласно спецификации. Полная стратегия тестирования программного обеспечения состоит из следующих типов испытаний и выполняется в следующем порядке:

1. Тестирование компонентов (модульное тестирование). Тестируются все программные компоненты (при этом проверяется покрытие кода тестами). Анализ кода.
2. Тестирование интеграции. Тестируется программное обеспечение, чтобы гарантировать, что компоненты взаимодействуют правильно.

4.1.2 Область тестирования

Целью этого тестирования является определить, соответствует ли разработанное программное обеспечение заявленным функциональным требованиям, а также выявить ошибки и представить их исправления, улучшив тем самым качество разработанного программного обеспечения.

Основные области тестирования включают:

Функциональное тестирование: Проверка выполнения всех функциональных требований системы.

Производительность: Оценка времени отклика системы, пропускной способности и масштабируемости.

Надежность: Проверка стабильности и устойчивости системы под различными условиями эксплуатации.

Безопасность: Проверка системы на уязвимости и соответствие требованиям безопасности.

4.1.3 Начальные условия

Перед началом тестирования необходимо выполнить следующие задачи:

1. Имеется законченная программная спецификация: Полное описание функциональности и нефункциональных требований к системе
2. Работающее программное обеспечение: Система должна быть готова к тестированию, включая установку и настройку всех компонентов.

3. Все необходимые компоненты находятся в рабочем состоянии: Все аппаратные и программные компоненты, необходимые для работы и тестирования системы, должны быть правильно установлены и функционировать.

4.2. Типы тестирования

Для всестороннего тестирования системы будут использоваться следующие типы тестирования:

1. Модульное тестирование: Тестирование отдельных модулей или компонентов системы. Целью является проверка правильности работы каждого компонента в отдельности. Модульное тестирование будет выполняться с использованием автоматизированных тестов.
2. Интеграционное тестирование: Проверка взаимодействия между модулями и компонентами системы. Целью является выявление проблем, возникающих при интеграции различных частей системы.
3. Системное тестирование: Комплексное тестирование всей системы целиком для проверки соответствия системы спецификациям и требованиям. Включает в себя функциональное, производительное, нагрузочное и стресс-тестирование.
4. Приемочное тестирование: Финальное тестирование, проводимое с целью подтверждения того, что система соответствует всем требованиям заказчика и готова к эксплуатации. Включает в себя тестирование с реальными данными и сценариями использования.

Составлен следующий набор тестов (табл. 4.1):

Таблица 4.1.

Набор тестов для тестирования системы

№	Тестовый вариант	Ожидаемый результат
1	Добавление нового устройства	Устройство успешно добавлено в базу данных
2	Авторизация пользователя	Пользователь успешно вошел в систему
3	Регистрация пользователя при помощи кода	Пользователь успешно зарегистрирован
5	Регистрация пользователя	Пользователь успешно зарегистрирован
6	Сохранение данных пользователя после редактирования пользователей	Настройки корректно сохранены в базу данных
7	Применение сохраненных настроек	Результаты успешно отправлены в корректной форме
8	Логирование	Успешное отображение логов

4.2.1 Приоритеты тестирования

Проверки перечислены в порядке уменьшения приоритетного уровня:

1. Функционал - все ли заданные функции Web-платформы выполняются, как ожидалось?
2. Удобство и простота - реально ли Web-платформа удобна и легко понимаема для пользователя?
3. Защита – тщательно ли обеспечивается защита данных?
4. Выполнение – Web-платформа соответствует всем критериям выполнения?

4.2.2 Методы тестирования

Будут использоваться **тестовые сценарии** – сценарии вариантов использования (с предопределенным вводом и ожидаемыми выходными данными), варианты использования взяты из диаграммы вариантов использования.

4.2.3 Среда тестирования

Для тестирования данного программного продукта необходимо следующее оборудование:

Персональный компьютер со следующей конфигурацией:

- Ryzen 7 3700x, 32Gb
- Операционная система Microsoft Windows 11
- Установленные программы jetbrains pycharm, Google Chrome

4.3. Модульное тестирование

Модульное тестирование — это процесс программирования, который позволяет проверять правильность блоков исходного кода, правильную работу программных модулей, а также наборов одного или нескольких программных модулей вместе с соответствующими управляющими данными.

Суть метода заключается в создании методов тестирования для тестирования отдельного метода в программе. Таким образом, можно проверить правильность кода при его изменении и добавлении функциональности в программу. Кроме того, модульные тесты позволяют обнаруживать ошибки в работе программных алгоритмов, что, в свою очередь, повышает качество и надежность программного обеспечения.

Цель модульного тестирования — выделить отдельные модули программы и проверить работу этих модулей в соответствии с определенными входными данными.

4.3.1 Тестовые случаи

Название функции: register

Описание тестового случая:

Тестирование функции register () – проверка создания нового пользователя в базе данных (табл. 4.2).

Таблица 4.2.

Тестовые варианты функции register ()

№	Входные данные	Что проверяется	Ожидаемый результат
1	Логин: user	Наличие в базе данных пользователя с таким же логином • Пользователь с таким логином не найден	Сообщение об успешной регистрации.
2	Логин: user	Наличие в базе данных пользователя с таким же логином • Пользователь с таким логином найден	Сообщение, что пользователь с таким логином уже существует.
3	Код регистрации: YHAZM3MO0M	Наличие в базе данных кода с таким же наименованием • Код с таким наименованием не найден	Сообщение, что неверный код регистрации
4	Код регистрации: YHAZM3MO0M	Наличие в базе данных кода с таким же наименованием • Код с таким наименованием найден	Сообщение об успешной регистрации.

Название функции: login

Описание тестового случая:

Шаги тестирования:

1. Пользователь заходит на сайт и нажимает кнопку авторизоваться
2. Вводит данные Логин и пароль

Тестирование функции login () – проверка пользователя в базе данных (табл. 4.3).

Таблица 4.3.

Тестовые варианты функции login ()

№	Входные данные	Что проверяется	Ожидаемый результат
1	Логин: user	Наличие в базе данных пользователя с таким логином • Пользователь с таким логином найден	Переход на главную страницу.
	Логин: user	Наличие в базе данных пользователя с таким логином • Пользователь с таким логином не найден	Сообщение, что пользователь с таким логином не найдет
2	Пароль: QWERTY	Корректность пароля • Пароль соответствует	Переход на главную страницу.
3	Пароль: QWERTY	Корректность пароля • Пароль не соответствует	Сообщение, что пароль не верный.

4.4. Системное тестирование

Системное тестирование — это тестирование программного обеспечения, выполняемое на полной, интегрированной системе, с целью проверки соответствия системы исходным требованиям.

Системное тестирование выполняется методом “черного ящика”.

Было проведено тестирование соответствия системы функциональным требованиям, изложенным в спецификации вариантов использования.

Тестирование системы показало ее соответствие функциональным требованиям.

Система правильно реагирует на неверный ввод данных пользователем, выдавая ему подсказки. На рис. 4.1 показан пример неверного ввода с последующим выводом сообщений об ошибках.

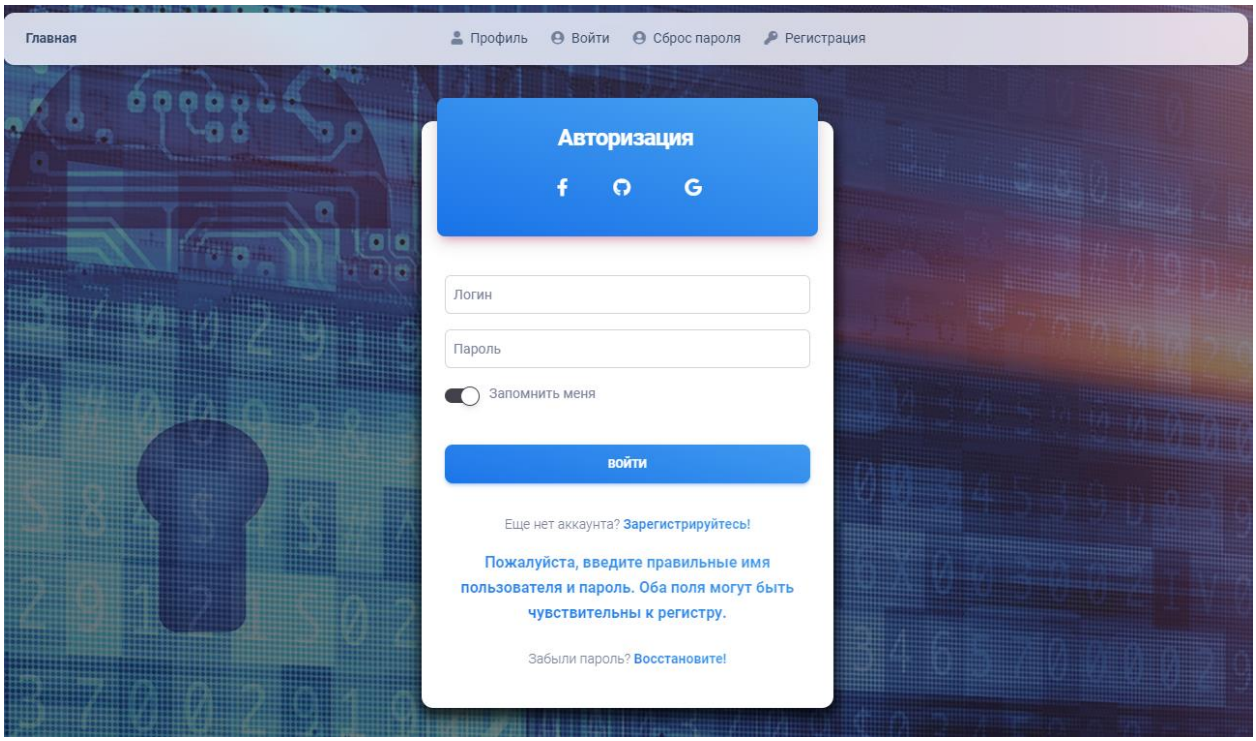


Рисунок 4.1. Попытка входа с неверными данными

На рис. 4.2 показан пример неверного ввода регистрационного кода с последующим выводом сообщений об ошибках.

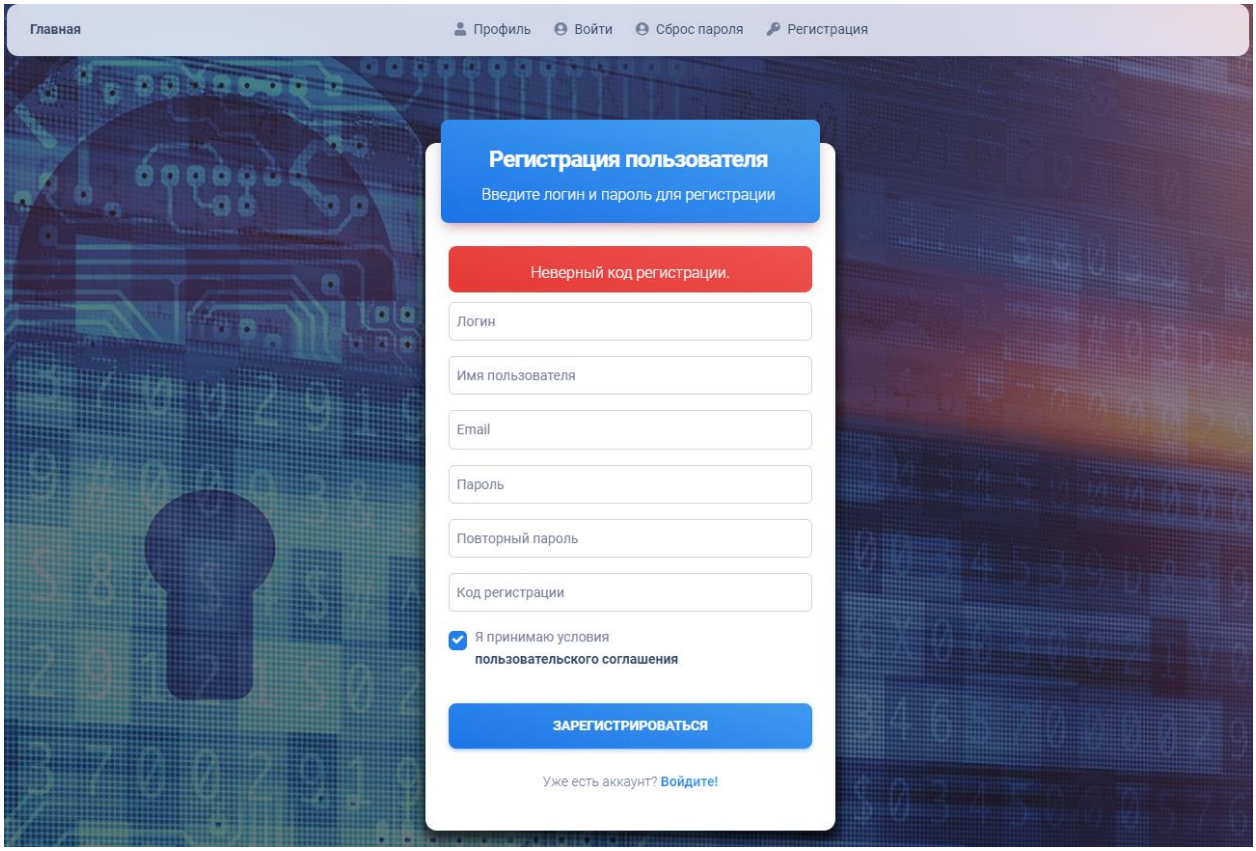


Рисунок 4.2. Попытка регистрации с неверным регистрационным кодом

На рис. 4.3 показан пример ввода просроченного регистрационного кода с последующим выводом сообщений об ошибках.

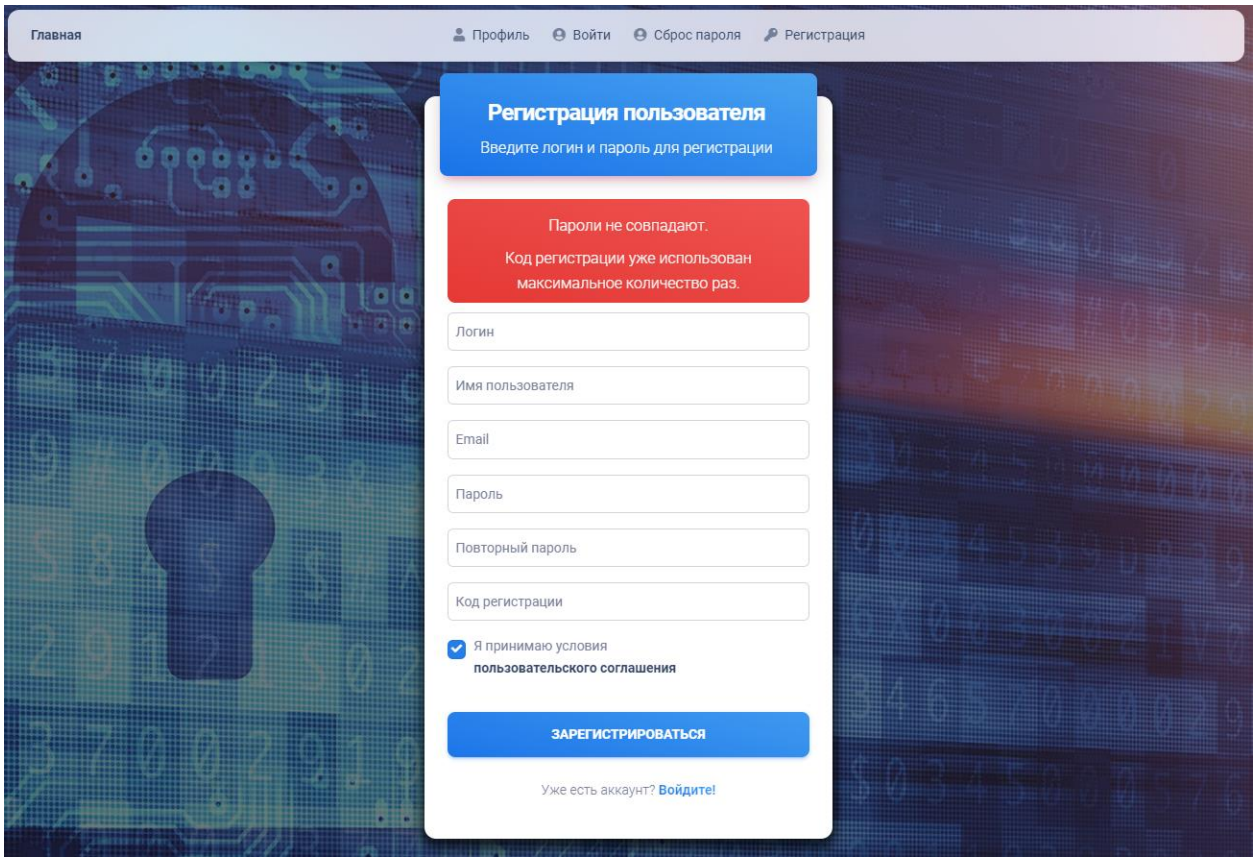


Рисунок 4.3. Попытка регистрации с просроченным регистрационным кодом

Глава 5. Руководство пользователя

5.1. Введение

Любая программная система разрабатывается для определенного конечного пользователя. В следствии этого, необходимо не только корректно спроектировать и протестировать систему, чтобы она удовлетворяла требованиям целевой аудитории, но и объяснить пользователю, как эту систему использовать.

Руководство пользователя предназначено для объяснения пользователю возможностей использования системы и её функций для получения желаемого результата. Все манипуляции с системой расписаны с точки зрения пользователя и представлены в четких инструкция по эксплуатации системы для ее корректной работы.

Данный раздел содержит в себе всю необходимую для пользователя информацию для самостоятельной работы с программой. В нем описываются основные операции, которые система предлагает пользователю.

5.2. Назначение системы

Система предназначена для удаленного управления и контроля за работой экзомассажеров.

Основные цели и функции системы включают:

- *Удаленное управление*: Пользователи могут включать и выключать экзомассажеры, изменять режим работы, настраивать интенсивность массажа и другие параметры, используя веб-интерфейс.
- *Мониторинг состояния*: Система позволяет отслеживать текущие параметры работы экзомассажеров, такие как температура, мощность и другие показатели в реальном времени. Пользователи могут получать уведомления о состоянии устройства, включая предупреждения о возможных неисправностях или необходимости технического обслуживания.
- *Сбор и анализ данных*: Система собирает данные о работе экзомассажеров и сохраняет их для последующего анализа.
- *Обратная связь и поддержка*: Пользователи могут отправлять запросы на техническое обслуживание и получать ответы на вопросы через встроенный интерфейс поддержки.

5.2.1 Условия применения системы

Пользователям системы необходимо иметь персональный компьютер со следующими минимальными системными требованиями:

- 1 GB оперативной памяти
- 10 GB свободного дискового пространства
- процессор с тактовой частотой 1 GHz
- Операционная система Windows 7, 8, 8.1, 10.
- Веб-браузер Opera, Mozilla Firefox, Google Chrome. Версии должны быть не раньше 2019 года.

Также пользователи должны обладать базовыми навыками работы с персональным компьютером и телефоном.

5.3. Подготовка системы к работе

Запуск системы

Для запуска системы нужно:

1. Перейти по ссылке exobed.lazareub.beget.tech.
2. После этого откроется страница аутентификации (рис. 5.1).

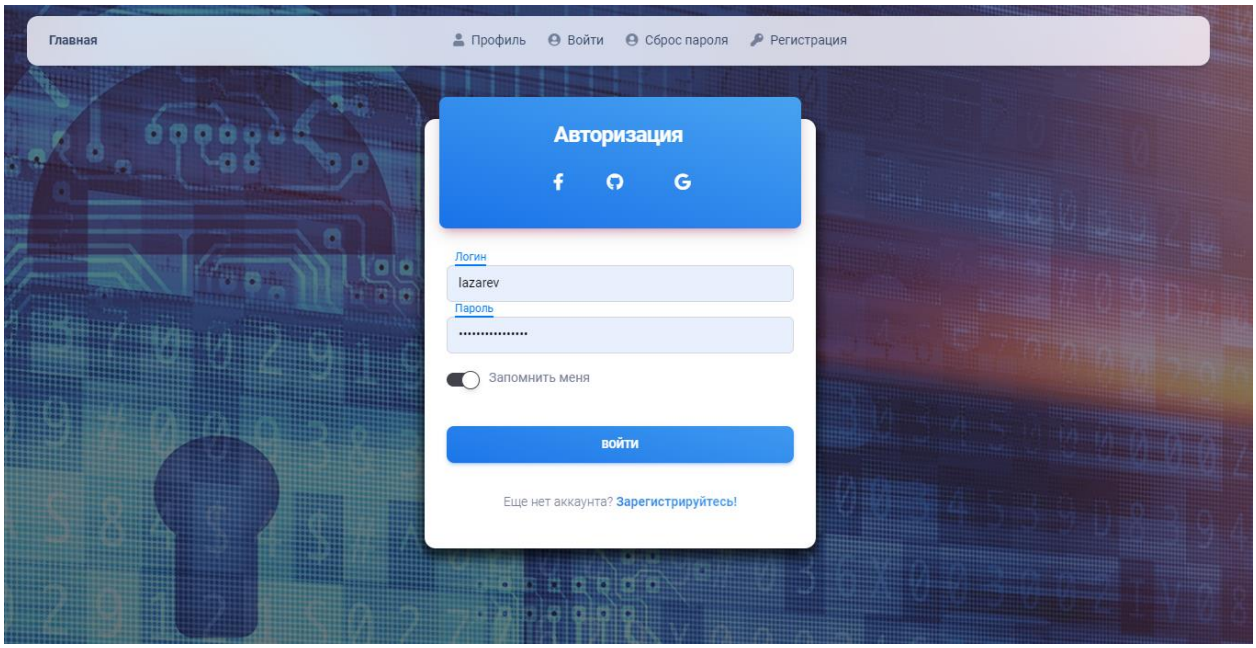


Рисунок 5.1. Страница входа в систему

Для входа нужно ввести свои данные: логин и пароль, а если нет аккаунта, то нужно создать новый аккаунт. (рис. 5.2).

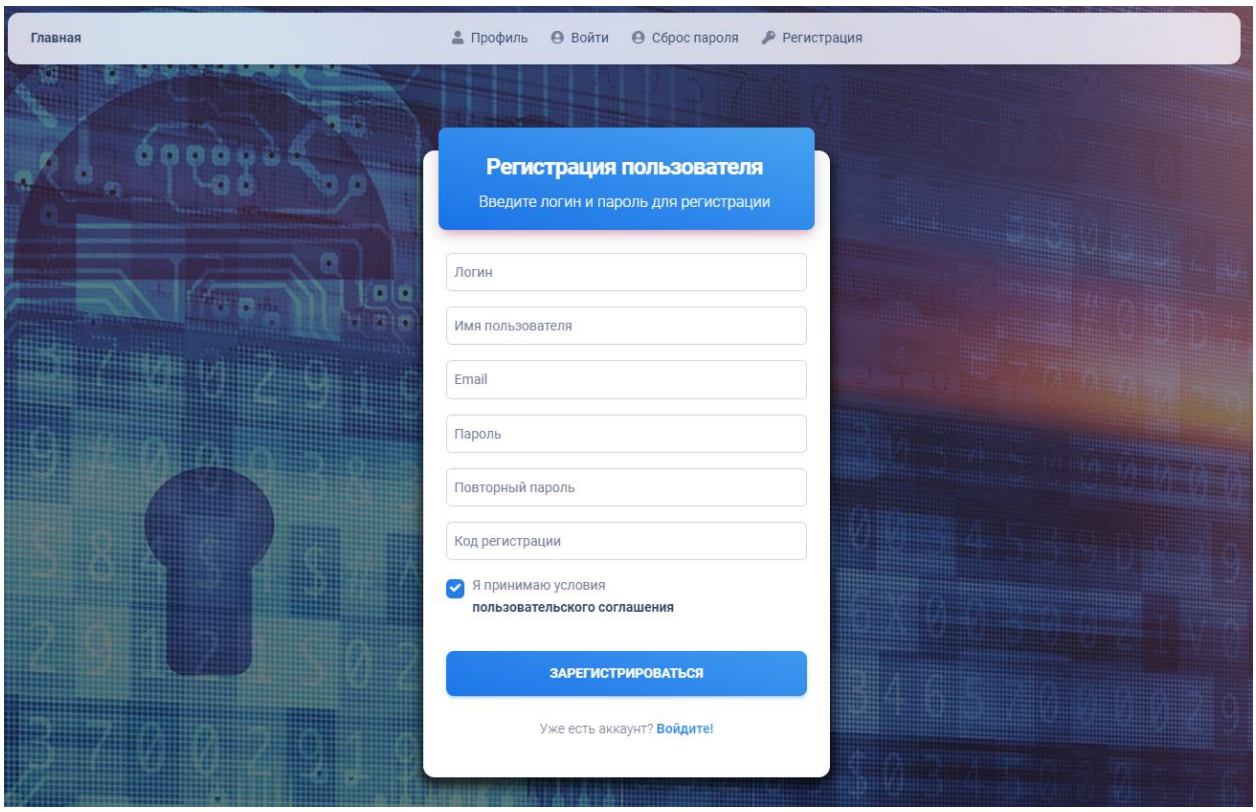


Рисунок 5.2. Страница регистрации

Авторизация

После успешного входа в систему, открывается главная страница сайта, но уже с возможностью просмотра доступных устройств (рис. 5.3).

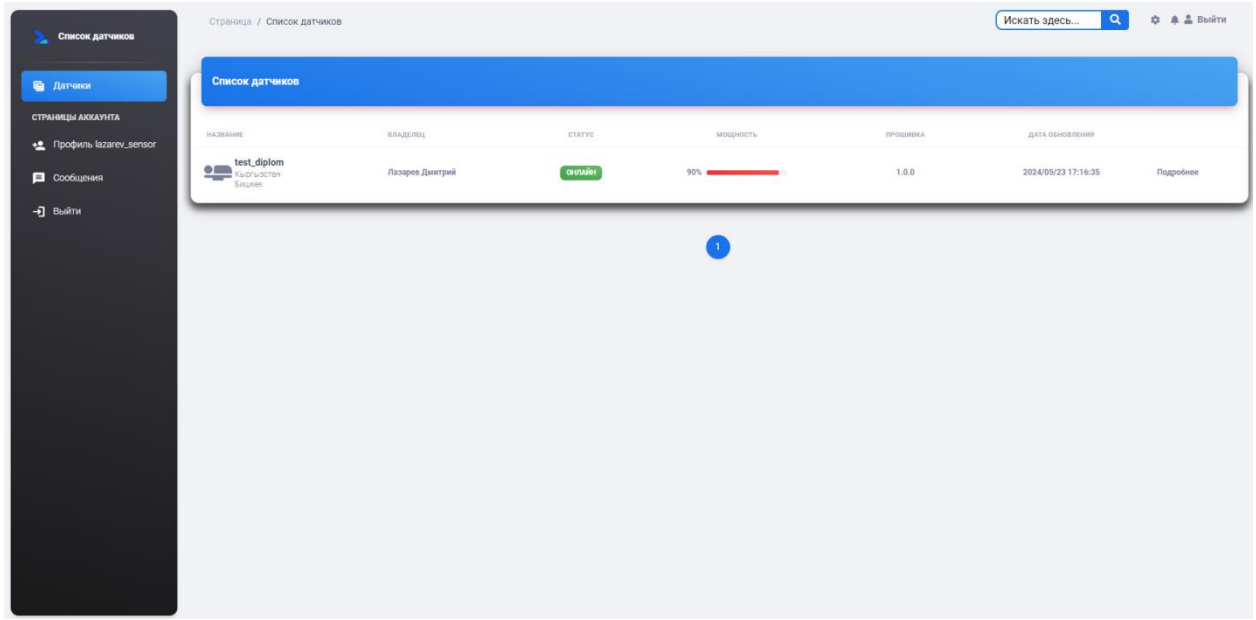


Рисунок 5.3. Главная страница с устройствами

Поиск устройств

Для поиска нужного устройства можно воспользоваться системой поиска по имени устройства, владельцу или описанию (рис. 5.4).

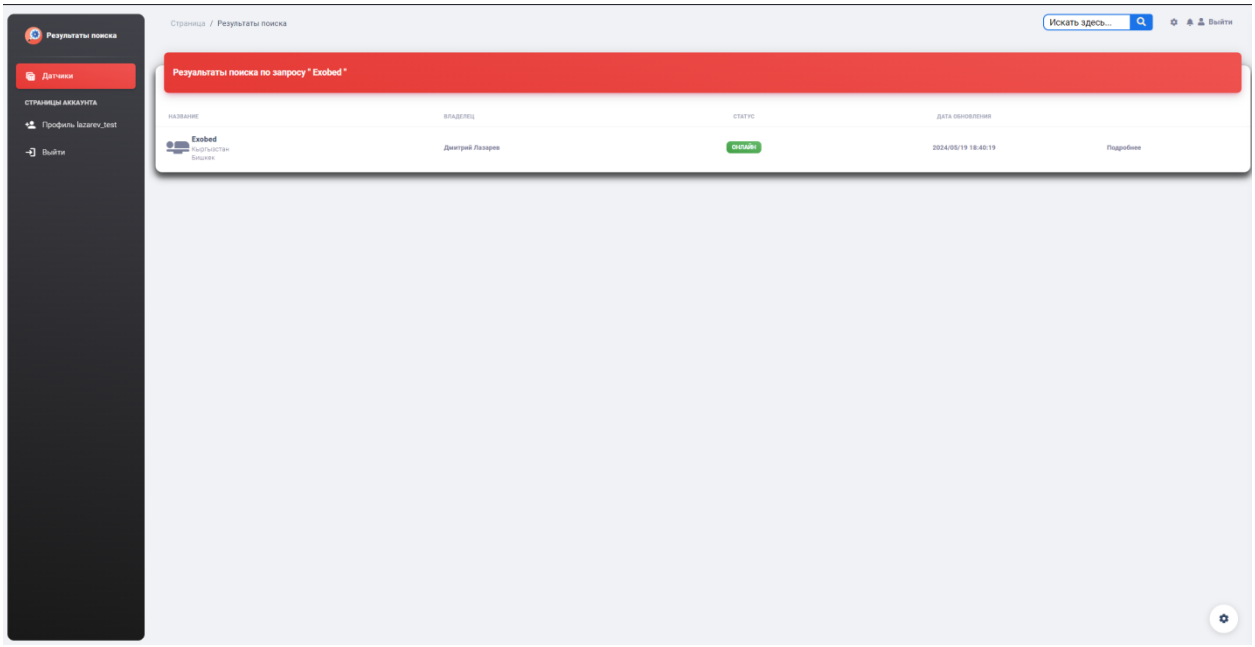
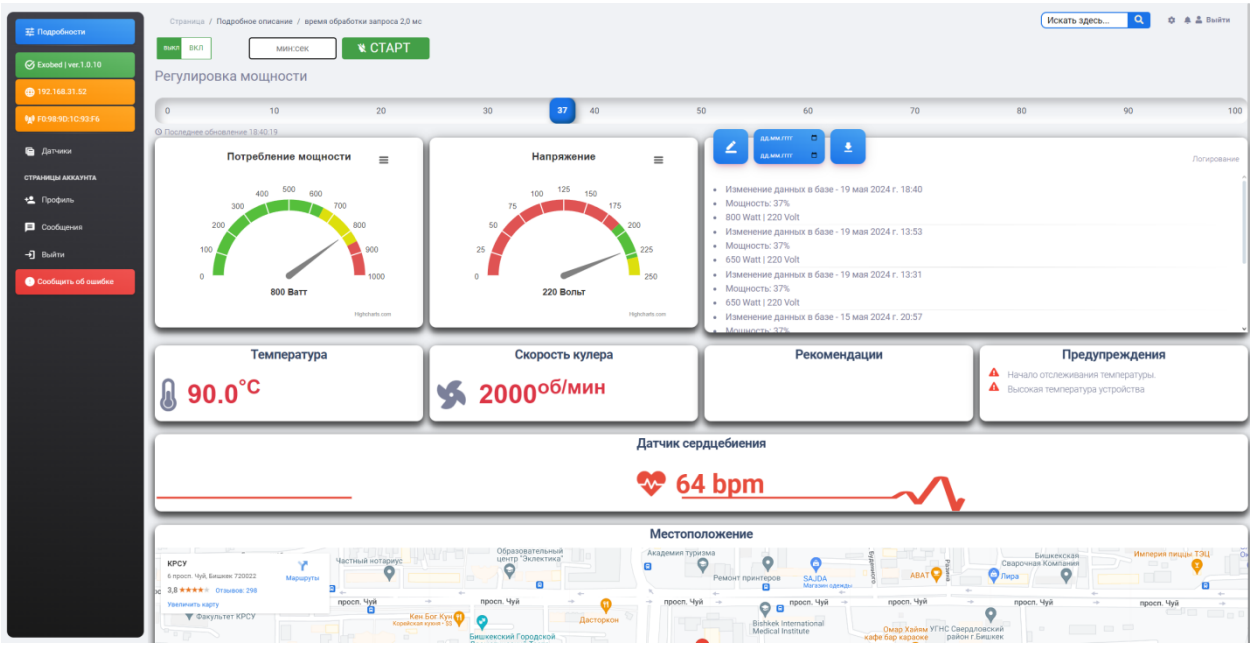


Рисунок 5.4. Страница с поиском

Подробный просмотр устройства

Для подробного просмотра устройства нужно нажать на кнопку «Подробнее» в списке устройств, после чего отобразится подробная информация (рис. 5.5).



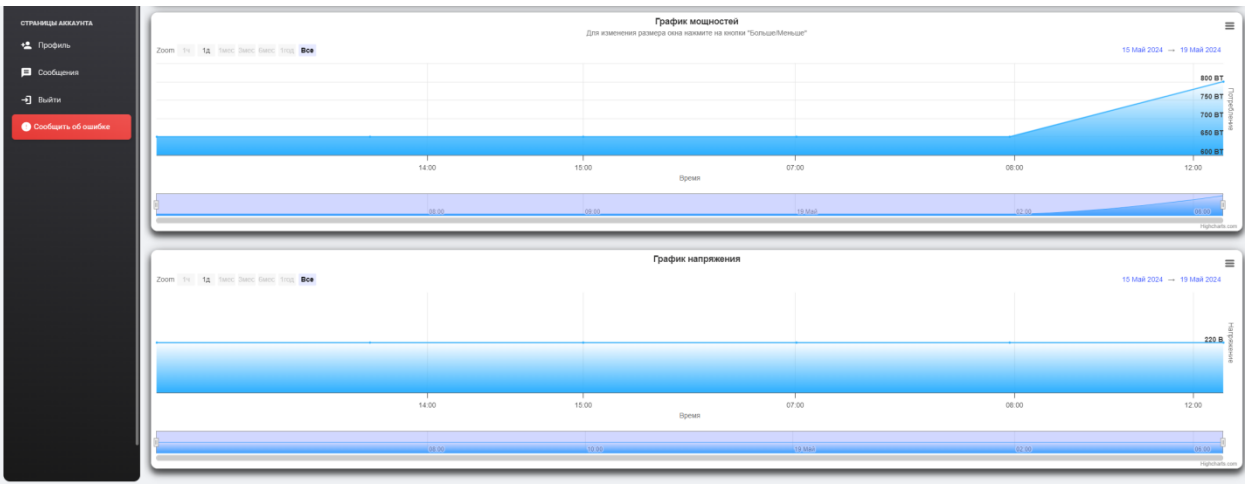


Рисунок 5.5. Подробная информация устройства

Вход от лица администратора

Также можно зайти в роли администратора. Только администратор может подтверждать новые устройства, добавлять коды регистрации, а также обновлять прошивку устройства (рис. 5.6, рис. 5.7 и рис. 5.8).

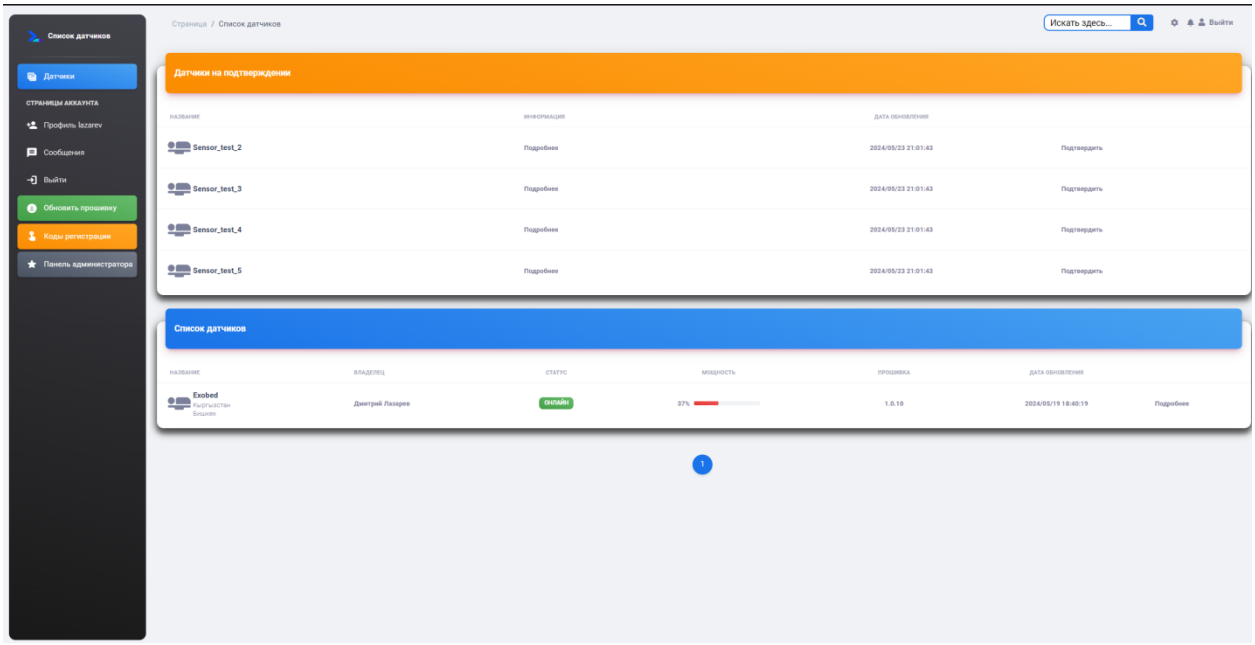


Рисунок 5.6. Вид от лица администратора

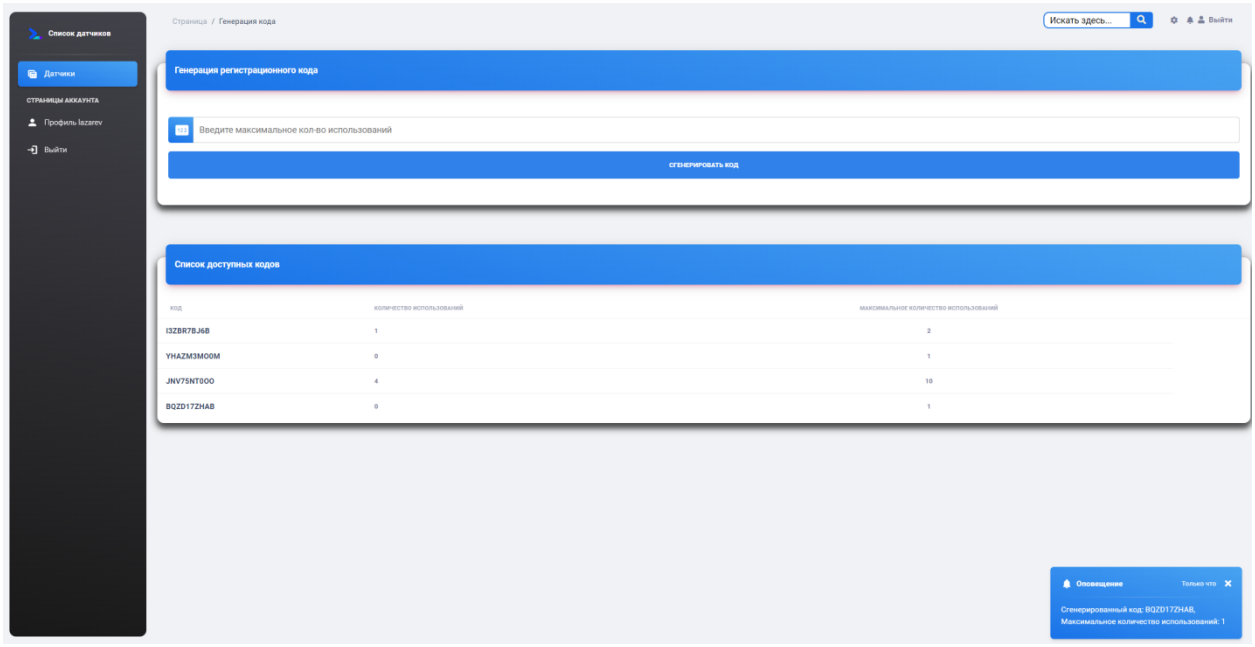


Рисунок 5.7. Добавление кода регистрации

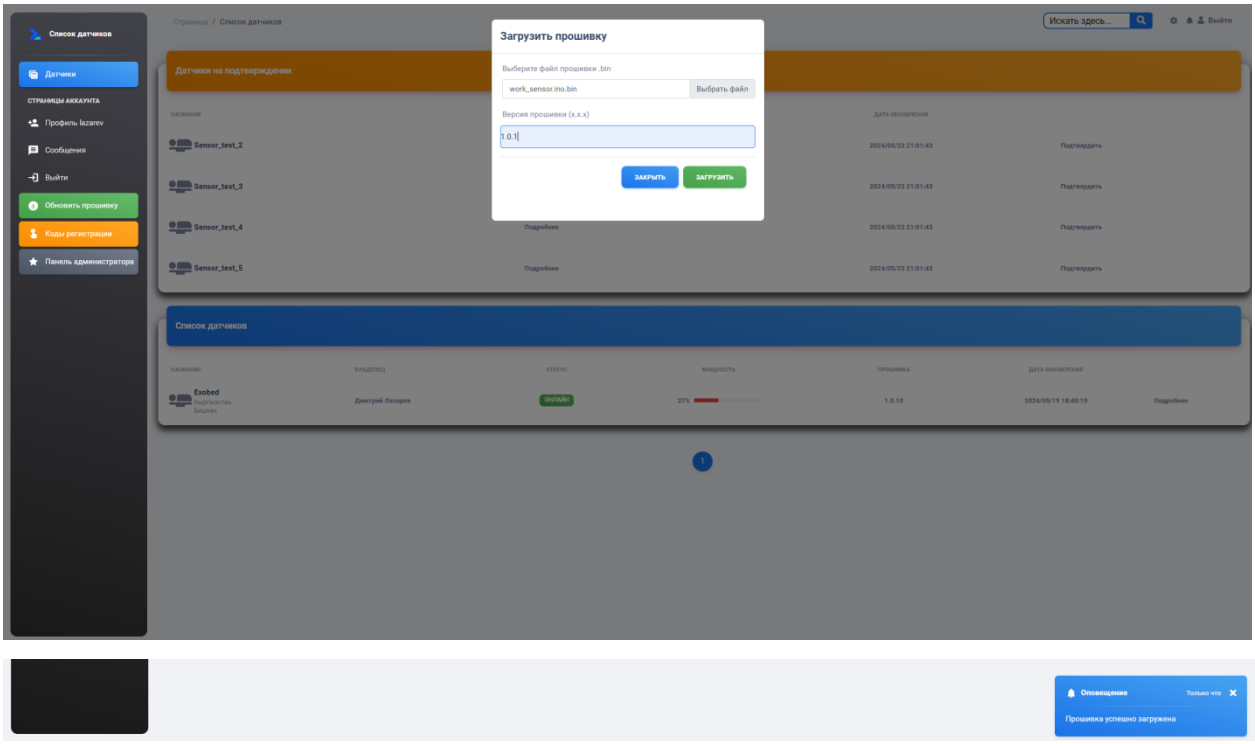


Рисунок 5.8. Добавление прошивки устройства

Редактирование профиля пользователя

Пользователь может редактировать свои данные (рис. 5.9).

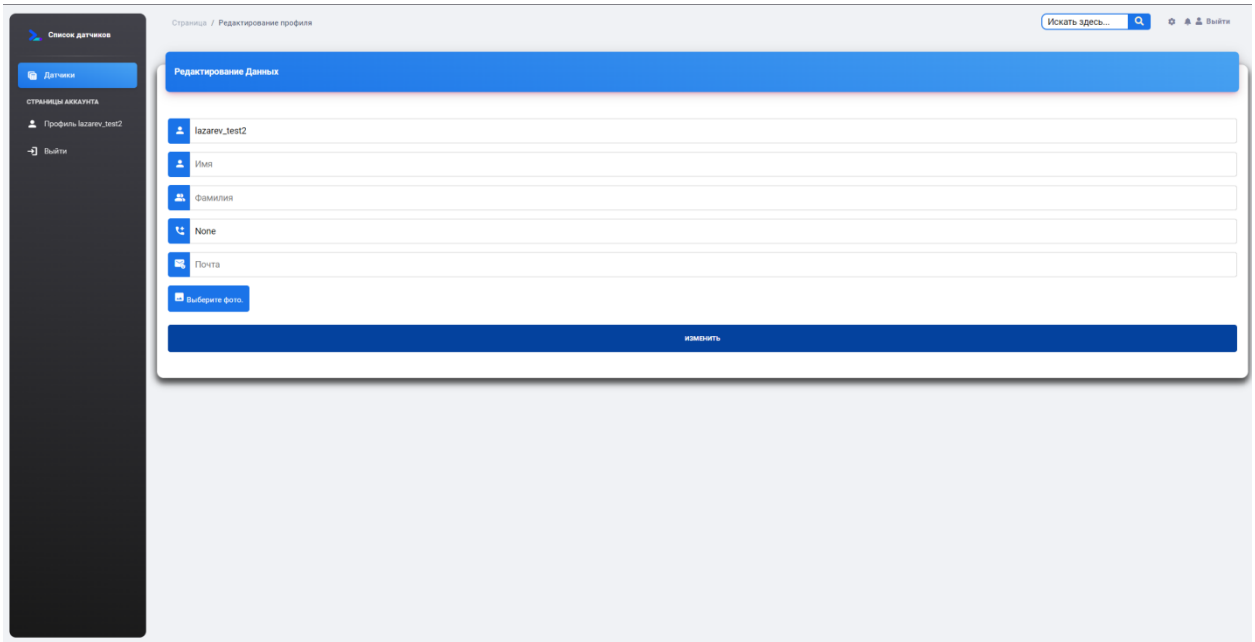


Рисунок 5.9. Редактирование профиля пользователя

Обращение в службу поддержки

Пользователь может обратиться в службу поддержки и получить ответы на свои заданные вопросы (рис. 5.10).

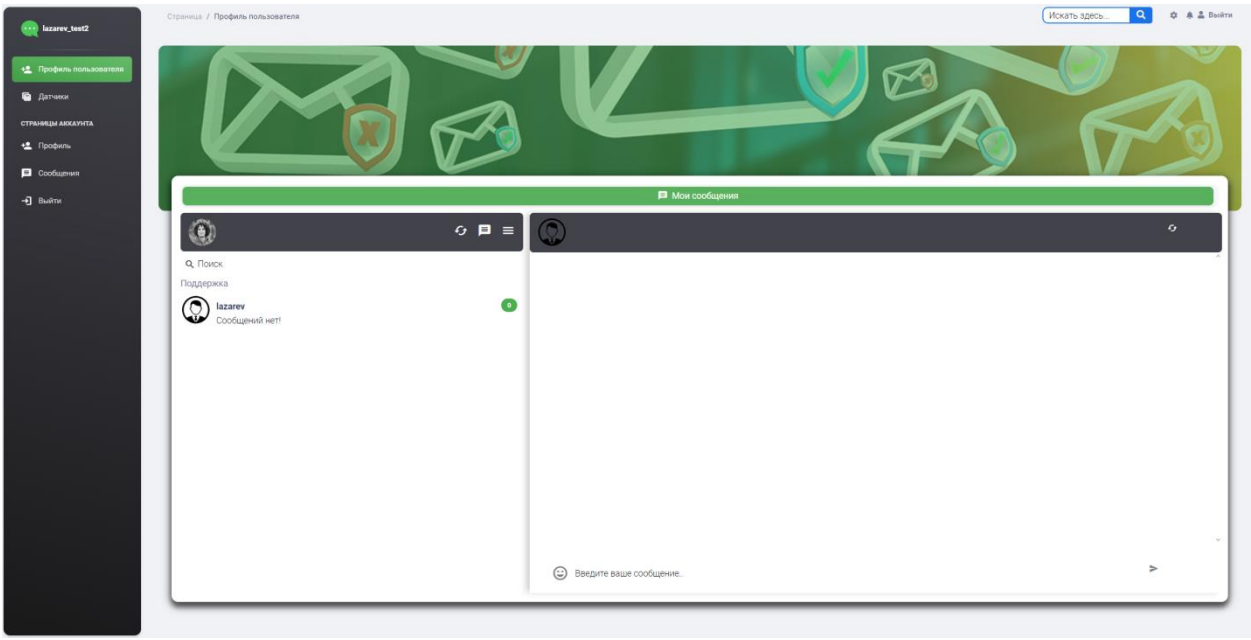


Рисунок 5.10. Страница обращения в службу поддержки

Заключение

В результате разработки программных средств для распределенной системы мониторинга технического состояния экзомассажеров:

1. Проведен анализ предметной области;
2. Разработаны интерфейсы;
3. Разработана модель системы;
4. Реализован алгоритм анализа данных для выявления аномалий и формирования рекомендаций по предотвращению отказов технических устройств.
5. Разработана система мониторинга ключевых технических параметров экзомассажоров с возможностью удаленного доступа для пользователей

Разработанная система выполняет следующие функции:

- Регистрация;
- Авторизация;
- Управление пользователями;
- Регистрация устройств;
- Удаленное управление устройствами;
- Удаленное обновление прошивок устройств;
- Просмотр логов устройств;
- Онлайн-связь с техподдержкой пользователей;

Дальнейшее развитие системы включает в себя:

- Введение системы подписки
- Улучшение клиентского сервиса
- Гибкость и масштабируемость системы

Список литературы

1. Цифровой репозиторий в информационных научно-образовательных системах / О. А. Федотова, А. М. Федотов, О. Л. Жижимов, М. А. Самбетбаева // Труды ГПНТБ СО РАН. – 2019. – № 3(3). – С. 23-28. – DOI 10.20913/2618-7515-2019-3-23-28. – EDN WNPDOH.
2. https://studfile.net/preview/8896197/page:2/ (дата обращения 15.06.2024)
3. https://poznayka.org/s88572t2.html (дата обращения 15.06.2024)
4. https://ru.wikipedia.org/wiki/Модульное_тестирование (дата обращения 15.06.2024)
5. Книга The Django Web Framework for the Python programming language: https://django-book.readthedocs.io/en/latest/
6. Фаулер, Мартин. UML Основы UML Distilled : краткое руководство по стандартному языку объектного моделирования; [пер. с англ. А. Петухова ; предисл. К. Кобрина и др.]. - 3-е изд. - Санкт-Петербург: Символ, 2013. - 184 с
7. Основы объектно-ориентированного представления ПС: Учебник/ С.Орлов. – Спб.: Питер,2016.-186с
8. Леоненков, А. В. Объектно-ориентированный анализ и проектирование с использованием UML и IBM Rational Rose. Курс лекций: учебное пособие для студентов вузов, обучающихся по специальностям в области информационных технологий / А. В. Леоненков. — Москва, Саратов: Интернет-Университет Информационных Технологий (ИНТУИТ), Вузовское образование, 2017. — 318 с.
9. UML диаграммы классов описание: https://habr.com/ru/post/511798/
10. OPM, основы Hibernate. Классы–сущности – URL: https://habr.com/ru/post/29694/
11. Основы UML. Диаграммы последовательности: https://pro-prof.com/archives/2769
12. Основы UML. Диаграммы компонетов: http://khpi-iip.mipk.kharkiv.edu/library/case/leon/gl10/gl10.html
13. Объектно-ориентированная разработка требований: Учебник/ С.Орлов. – Спб.:Питер,2016.-239с
14. Системное тестирование: https://coderlessons.com/tutorials/kachestvo-programmnogo-obespecheniia/ruchnoe-testirovanie/testirovanie-sistemy-2
15. Системные тесты: https://habr.com/ru/post/81226/

Глоссарий

1. Введение

1.1. Цель

Глоссарий содержит описание терминов, используемых при разработке распределенной системы мониторинга технического состояния экзомассажеров.

2. Определения

2.1. Понятия, используемые при описании исходной информации

Экзомассажер

Экзомассажер — это устройство, предназначенное для массажа определенных частей тела. Он может использоваться для улучшения кровообращения, уменьшения мышечных напряжений, а также для расслабления и снятия стресса. Экзомассажеры могут иметь различные формы и функции, включая вибрацию, тепловое воздействие и другие методы воздействия на мышцы и ткани. Они могут быть полезны для людей, которые испытывают напряжение в мышцах или нуждаются в дополнительном способе релаксации.

ESP-8266

ESP-8266 — это микроконтроллер с Wi-Fi модулем, разработанный компанией Espressif Systems. Он позволяет устройствам подключаться к беспроводным сетям Wi-Fi и выполнять различные задачи в интернете вещей (IoT), такие как сбор данных, управление устройствами и передача информации через Интернет. ESP-8266 обладает небольшим размером, низким энергопотреблением и доступной стоимостью, что делает его популярным выбором для создания различных умных устройств и прототипов.

Администратор

Администратор - пользователь системы, наделенный правами изменения в системе. Он может добавлять, удалять, изменять информация на сайте

Web-Платформа – это многоцелевая платформа для разработки веб-проектов и управления содержимым. Она представляет собой набор комплексных решений, направленных на легкую и успешную разработку веб-сайта и его поддержку.

Сайт - информационная система, предоставляющая пользователям сети интернет-доступ к своему содержимому и функционалу в виде упорядоченного набора взаимосвязанных HTML-страниц.

Оператор – может зайти на сайт и ознакомиться со всей информацией о устройствах, выпусках и статьях

Приложение 2.

Тексты программ

models.py

```
from django.urls import reverse
from django.db import models
from django.core.validators import MinValueValidator, MaxValueValidator
from django.utils.text import slugify


class Country(models.Model):
    id = models.AutoField(primary_key=True)
    name = models.CharField("Название", max_length=100)

    def __str__(self):
        return self.name

    class Meta:
        verbose_name = 'Страна'
        verbose_name_plural = 'Страны'


class City(models.Model):
    id = models.AutoField(primary_key=True)
    country = models.ManyToManyField(Country, blank=True, verbose_name='Страна')
    name = models.CharField("Название", max_length=100)

    def __str__(self):
        return self.name

    class Meta:
        verbose_name = 'Город'
        verbose_name_plural = 'Города'


class Sensor(models.Model):
    id = models.AutoField(primary_key=True)
```

```
    name = models.CharField('Имя датчика', max_length=100, unique=True)
    slug = models.SlugField(max_length=255, unique=True, db_index=True,
verbose_name="URL")
    description = models.TextField('Описание', blank=True)
    owner = models.CharField("Владелец", max_length=100, blank=True)
    date_added = models.DateTimeField('Дата добавления', auto_now=True)
    country = models.ManyToManyField(Country, blank=True, verbose_name='Страна')
    city = models.ManyToManyField(City, blank=True, verbose_name='Город')
    inclusions = models.IntegerField("Количество включений", default=0)
    power = models.IntegerField(default=0, blank=True, verbose_name='Мощность %',
                        validators=[MinValueValidator(0), MaxValueValidator(100)])
    watt = models.IntegerField("Потребление мощности Ватт", default=0)
    volt = models.IntegerField("Электрическое напряжение Вольт", default=0)
    work = models.BooleanField('Онлайн', default=True)
    blocked = models.BooleanField('Заблокирован', default=False)
    start = models.BooleanField('Старт', default=False)
    temperature = models.FloatField('Температура', default=0)
    fan_speed = models.IntegerField('Скорость кулера', default=0)
    ip_address = models.CharField('IP Адрес', max_length=15, blank=True)
    mac_address = models.CharField('MAC Адрес', max_length=20, blank=True)
    confirmed = models.BooleanField('Подтвержден администратором', default=False)
    version = models.CharField('Версия прошивки', max_length=15, blank=True)
    processing_time = models.FloatField('Время обработки (мс)', null=True, blank=True)

    def __str__(self):
        return self.name

    def is_blocked(self):
        return self.blocked

    def get_absolute_url(self):
        return reverse('sensor_detail', args=[str(self.id)])

    class Meta:
        verbose_name = 'Датчик'
        verbose_name_plural = 'Датчики'
```

```
    def save(self, *args, **kwargs):
        if self.blocked:
            self.work = False

        if not self.slug:
            self.slug = slugify(self.name)

        if not self.work:
            self.watt = 0
            self.volt = 0
            self.temperature = 0
            self.fan_speed = 0

        super().save(*args, **kwargs)

        if self.pk:
            log_type = 'Изменение данных в базе'
            SensorLog.objects.create(sensor=self, log_type=log_type,
previous_power=self.power,
                                     previous_watt=self.watt, previous_volt=self.volt)


class SensorLog(models.Model):
    sensor = models.ForeignKey(Sensor, on_delete=models.CASCADE)
    timestamp = models.DateTimeField(auto_now_add=True)
    log_type = models.CharField(max_length=20)
    previous_power = models.IntegerField(null=True, blank=True)
    previous_watt = models.IntegerField(null=True, blank=True)
    previous_volt = models.IntegerField(null=True, blank=True)

    def __str__(self):
        return f'{self.sensor.name} - {self.log_type} - {self.timestamp}'

    def save(self, *args, **kwargs):
        if self.pk:
            try:
                old_log = SensorLog.objects.get(pk=self.pk)
```

```
                if old_log.previous_watt != self.previous_watt:
                    log_type = 'Изменение мощности'
                    SensorLog.objects.create(sensor=self.sensor, log_type=log_type,
                                             previous_power=old_log.previous_watt)
                elif old_log.previous_volt != self.previous_volt:
                    log_type = 'Изменение напряжения'
                    SensorLog.objects.create(sensor=self.sensor, log_type=log_type,
    previous_volt=old_log.previous_volt)
            except SensorLog.DoesNotExist:
                pass
        super(SensorLog, self).save(*args, **kwargs)

    class Meta:
        verbose_name = 'Логи'
        verbose_name_plural = 'Логи'


class Firmware(models.Model):
    version = models.CharField("Версия", max_length=10)
    file = models.FileField(upload_to='firmwares/')
    uploaded_at = models.DateTimeField(auto_now_add=True)

    def __str__(self):
        return self.version

    class Meta:
        verbose_name = 'Прошивка'
verbose_name_plural = 'Прошивки'
```

views.py

```
from django.views.generic import View, ListView, DetailView, DeleteView
from django.views.decorators.csrf import csrf_exempt
from django.contrib.auth.decorators import login_required
from django.contrib.auth.mixins import LoginRequiredMixin
from django.db.models import Q
from .models import *
from django.core.exceptions import ObjectDoesNotExist
from django.urls import reverse_lazy
from django.shortcuts import render, get_object_or_404, redirect
from django.contrib import messages
from django.core.paginator import Paginator
from django.template.loader import render_to_string
from django.middleware.csrf import get_token
from django.utils import timezone
from django.conf import settings
from django.http import HttpResponseRedirect, JsonResponse, HttpResponse,
HttpResponseBadRequest, StreamingHttpResponse, \
    FileResponse

from datetime import datetime
import json
import time
import os
from datetime import timedelta

from rest_framework.views import APIView
from rest_framework.response import Response
from rest_framework import status

from asgiref.sync import async_to_sync
from channels.layers import get_channel_layer



# Класс для отображения списка датчиков
class SensorListView(LoginRequiredMixin, ListView):
    model = Sensor
```

```
    template_name = 'sensor/sensor_list.html'
    context_object_name = 'sensors'
    paginate_by = 8

    def get_queryset(self):
        queryset = super().get_queryset()
        queryset = queryset.order_by('name', 'owner', '-blocked', '-date_added')
        queryset = queryset.prefetch_related('sensorlog_set')

        confirmed_sensors = queryset.filter(confirmed=True)
        unconfirmed_sensors = queryset.filter(confirmed=False)

        return confirmed_sensors

    def get_context_data(self, **kwargs):
        context = super().get_context_data(**kwargs)

        unconfirmed_sensors = Sensor.objects.filter(confirmed=False).order_by('name',
'owner', '-blocked',
                                                    '-date_added')
        for sensor in unconfirmed_sensors:
            latest_log = sensor.sensorlog_set.last()
            sensor.last_updated = latest_log.timestamp if latest_log else sensor.date_added

        context['unconfirmed_sensors'] = unconfirmed_sensors

        confirmed_sensors = context['sensors']
        for sensor in confirmed_sensors:
            latest_log = sensor.sensorlog_set.last()
            sensor.last_updated = latest_log.timestamp if latest_log else sensor.date_added

        context['confirmed_sensors'] = confirmed_sensors
        return context
```

```
class ConfirmSensorView(View):
    def get(self, request, sensor_id):
        sensor = get_object_or_404(Sensor, pk=sensor_id)
        sensor.confirmed = True
        sensor.save()
        return redirect('sensor_list')



# Класс для отображения подробной информации о датчике
class SensorDetailView(LoginRequiredMixin, DetailView):
    model = Sensor
    template_name = 'sensor/sensor_detail.html'
    context_object_name = 'sensor'
    paginate_by = 15

    def get_queryset(self):
        queryset = super().get_queryset()
        return queryset.prefetch_related('sensorlog_set').all()

    def post(self, request, *args, **kwargs):
        sensor = self.get_object()
        previous_power = sensor.power
        previous_watt = sensor.watt
        previous_volt = sensor.volt
        sensor.work = not sensor.work
        sensor.save()


        log_type = 'Включение' if sensor.work else 'Выключение'
        SensorLog.objects.create(sensor=sensor, log_type=log_type,
previous_power=previous_power,
                                 previous_watt=previous_watt, previous_volt=previous_volt)


        return JsonResponse(
            {'success': True, 'power_changed': previous_power != sensor.power,
'current_power': sensor.power})
```

```
def get_context_data(self, **kwargs):
        context = super().get_context_data(**kwargs)
        sensor = self.get_object()
        logs = sensor.sensorlog_set.all().order_by('-timestamp')
        paginator = Paginator(logs, 10)
        page_number = self.request.GET.get('page')
        page_logs = paginator.get_page(page_number)
        context['logs'] = page_logs
        context['sensor_id'] = sensor.id

        temperature = sensor.temperature
        fan_speed = sensor.fan_speed
        context['temperature'] = temperature
        context['fan_speed'] = fan_speed

        self.add_warnings_to_context(sensor, context)
        self.add_recommendations_to_context(sensor, context)

        if logs.exists():
            context[
                'latest_log'] = logs.first()
        else:
            context['latest_log'] = None

        return context

    def add_warnings_to_context(self, sensor, context):
        warnings = []

        previous_temperatures = self.request.session.get('previous_temperatures', [])
        temperature = sensor.temperature
        fan_speed = sensor.fan_speed

        if previous_temperatures:
            last_three_temperatures = previous_temperatures[-3:]
            if all(temp > last_three_temperatures[i - 1] for i, temp in
                   enumerate(last_three_temperatures[1:], start=1)):
```

```
                warnings.append('Температура постоянно растет.')
        else:
            warnings.append('Начало отслеживания температуры.')

        previous_temperatures.append(temperature)
        self.request.session['previous_temperatures'] = previous_temperatures

        if temperature is not None and temperature > 80:
            warnings.append('Высокая температура устройства')
        if fan_speed is not None and fan_speed < 100:
            warnings.append('Низкая скорость кулера')

        context['warnings'] = warnings

    def add_recommendations_to_context(self, sensor, context):
        recommendations = []

        previous_powers = self.request.session.get('previous_powers', [])

        power = sensor.power
        previous_powers.append(power)
        self.request.session['previous_powers'] = previous_powers

        if len(previous_powers) >= 5 and all(p > 90 for p in previous_powers[-5:]):
            recommendations.append('Мощность устройства долгое время превышает
90%. Рекомендуется снизить мощность.')

        context['recommendations'] = recommendations

    def render_to_response(self, context, **response_kwargs):
        if self.request.headers.get('x-requested-with') == 'XMLHttpRequest':
            logs_html = self.render_logs_to_html(context['logs'])
            num_pages = context['logs'].paginator.num_pages
            return JsonResponse(
                {'logs_html': logs_html, 'has_next': context['logs'].has_next(), 'num_pages':
num_pages})
        else:
```

```
            return super().render_to_response(context, **response_kwargs)

    def render_logs_to_html(self, logs):
        return ''.join([self.render_log_to_html(log) for log in logs])

    def render_log_to_html(self, log):
        russian_months = {
            1: 'января', 2: 'февраля', 3: 'марта', 4: 'апреля', 5: 'мая', 6: 'июня',
            7: 'июля', 8: 'августа', 9: 'сентября', 10: 'октября', 11: 'ноября', 12: 'декабря'
        }
        month = russian_months[log.timestamp.month]
        html = f"""
            <li>{log.log_type} - {log.timestamp.strftime('%d')} {month}
{log.timestamp.strftime('%Y г. %H:%M')}</li>
            <li>Мощность: {log.previous_power}%</li>
            """
        if log.previous_watt:
            html += f"<li>{log.previous_watt} Watt | {log.previous_volt} Volt</li>"
        html += "<hr class='dark horizontal my-0'>"
        return html

    @staticmethod
    def get_sensor_logs(sensor_id):
        logs = SensorLog.objects.filter(sensor_id=sensor_id).order_by('timestamp')
        data = [[log.timestamp.timestamp() * 1000, log.previous_watt] for log in logs]
        return data


class SensorDeleteView(LoginRequiredMixin, DeleteView):
    model = Sensor
    template_name = 'sensor/sensor_detail.html'

    def get_object(self, queryset=None):
        return get_object_or_404(Sensor, pk=self.kwargs.get('pk'))

    def delete(self, request, *args, **kwargs):
        self.object = self.get_object()
```

```
        success_url = self.get_success_url()
        self.object.delete()
        return HttpResponseRedirect(success_url)

    def get_success_url(self):
        return reverse_lazy('sensor_list')


@login_required
def block_toggle(request, sensor_id):
    sensor = get_object_or_404(Sensor, pk=sensor_id)
    sensor.blocked = not sensor.blocked
    sensor.save()

    redirect_url = request.META.get('HTTP_REFERER') or reverse('home')
    return HttpResponseRedirect(redirect_url)


@login_required
def toggle_start(request, sensor_id):
    sensor = get_object_or_404(Sensor, pk=sensor_id)
    sensor.start = not sensor.start
    sensor.save()

    redirect_url = request.META.get('HTTP_REFERER') or reverse('home')
    return HttpResponseRedirect(redirect_url)


# Функция для загрузки логов датчика в формате CSV
@login_required
def download_logs(request, device_slug):
    sensor = get_object_or_404(Sensor, slug=device_slug)

    start_date_str = request.GET.get('start_date')
    end_date_str = request.GET.get('end_date')

    start_date = datetime.strptime(start_date_str, '%Y-%m-%d') if start_date_str else None
```

```
    end_date = datetime.strptime(end_date_str, '%Y-%m-%d') if end_date_str else None

    logs = SensorLog.objects.filter(sensor=sensor)
    if start_date and end_date:
        logs = logs.filter(timestamp__range=(start_date, end_date))

    csv_data = render_to_string('logs/csv_template.txt', {'logs': logs})
    filename = f'{sensor.name}_logs.csv'

    response = HttpResponse(csv_data, content_type='text/csv')
    response['Content-Disposition'] = f'attachment; filename="{filename}"'
    return response


# Функция для обработки AJAX запроса с подробной информацией о датчике
def sensor_detail_ajax(request, sensor_id):
    sensor = Sensor.objects.get(pk=sensor_id)
    data = {
        'watt': sensor.watt,
        'volt': sensor.volt,
        'work': sensor.work,
        'temperature': sensor.temperature,
        'fan_speed': sensor.fan_speed,
    }
    return JsonResponse(data)


# Функция для обновления данных датчика (мощности) через AJAX
def update_sensor_power(request, sensor_id):
    if request.method == 'POST' and request.headers.get('x-requested-with') ==
'XMLHttpRequest':
        power = request.POST.get('power')

        sensor = Sensor.objects.get(pk=sensor_id)
        previous_power = sensor.power
        sensor.power = power
        sensor.save()
```

```
            log_type = 'Изменение мощности'
            log_entry = SensorLog.objects.create(sensor=sensor, log_type=log_type,
                                                 previous_power=previous_power,
                                                 previous_watt=sensor.watt, previous_volt=sensor.volt)

            log_entry.previous_power = sensor.power
            log_entry.save()

            return JsonResponse({'success': True})
        else:
            return JsonResponse({'error': 'This endpoint only accepts AJAX requests'},
status=400)


# Функция для потоковой передачи логов датчика
def stream_sensor_logs(request, sensor_id):
    if not sensor_id:
        return HttpResponseBadRequest("No sensor_id provided")

    # Функция-генератор для получения логов датчика
    def generate_logs(sensor_id):
        while True:
            logs = SensorLog.objects.filter(sensor_id=sensor_id).order_by('timestamp')
            for log in logs:
                yield "data: %s\n\n" % json.dumps({
                    'log_type': log.log_type,
                    'timestamp': log.timestamp,
                    'previous_power': log.previous_power,
                    'previous_watt': log.previous_watt,
                    'previous_volt': log.previous_volt
                })
            time.sleep(1)

    # Возвращаем потоковый HTTP-ответ
    response = StreamingHttpResponse(generate_logs(sensor_id),
content_type='text/event-stream')
```

```
    response['Cache-Control'] = 'no-cache'
    response['Connection'] = 'keep-alive'
    return response


# Класс API для получения логов датчика
class SensorLogsAPIView(View):
    def get(self, request, *args, **kwargs):
        sensor_id = self.kwargs['sensor_id']
        logs = SensorLog.objects.filter(sensor_id=sensor_id).order_by('timestamp')
        data = [[log.timestamp.timestamp() * 1000, log.previous_watt] for log in logs]
        return JsonResponse(data, safe=False)


# Класс API для получения логов напряжения датчика
class SensorLogsVoltAPIView(View):
    def get(self, request, *args, **kwargs):
        sensor_id = self.kwargs['sensor_id']
        logs = SensorLog.objects.filter(sensor_id=sensor_id).order_by('timestamp')
        data = [[log.timestamp.timestamp() * 1000, log.previous_volt] for log in logs]
        return JsonResponse(data, safe=False)


def search_sensors(request):
    query = request.GET.get('q')
    results = []

    if query:
        sensors = Sensor.objects.filter(
            Q(name__icontains=query) |
            Q(description__icontains=query) |
            Q(owner__icontains=query) |
            Q(mac_address__icontains=query) |
            Q(ip_address__icontains=query) |
            Q(country__name__icontains=query) |
            Q(city__name__icontains=query) |
            Q(inclusions__icontains=query)
```

```
        ).distinct()

        for sensor in sensors:
            match_field = []
            if query.lower() in sensor.name.lower():
                match_field.append("Название")
            if query.lower() in sensor.description.lower():
                match_field.append("Описание")
            if query.lower() in sensor.owner.lower():
                match_field.append("Владелец")
            if query.lower() in sensor.mac_address.lower():
                match_field.append("MAC адрес")
            if query.lower() in sensor.ip_address.lower():
                match_field.append("IP адрес")
            if query.lower() in [country.name.lower() for country in sensor.country.all()]:
                match_field.append("Страна")
            if query.lower() in [city.name.lower() for city in sensor.city.all()]:
                match_field.append("Город")

            results.append((sensor, '', ''.join(match_field)))
    else:
        sensors = Sensor.objects.all()
        results = [(sensor, "") for sensor in sensors]

    return render(request, 'sensor/search_results.html', {'results': results, 'query': query})


@csrf_exempt
def receive_data(request):
    if request.method == 'POST':
        start_time = time.time()
        try:
            data = json.loads(request.body)
            name = data.get('name', '')
            temperature = data.get('temperature', 0.0)
            humidity = data.get('humidity', 0.0)
            inclusions = data.get('inclusions', 0)
```

```
power = data.get('power', 0)
watt = data.get('watt', 0)
volt = data.get('volt', 0)
work = data.get('work', False)
blocked = data.get('blocked', False)
fan_speed = data.get('fan_speed', 0)
ip_address = data.get('ip_address', '')
mac_address = data.get('mac_address', '')
version = data.get('version', '')
owner = data.get('owner', '')

sensor, created = Sensor.objects.get_or_create(name=name)
sensor.temperature = temperature
sensor.humidity = humidity
sensor.inclusions = inclusions
sensor.power = power
sensor.watt = watt
sensor.volt = volt
sensor.work = work
sensor.blocked = blocked
sensor.fan_speed = fan_speed
sensor.ip_address = ip_address
sensor.mac_address = mac_address
sensor.version = version
sensor.owner = owner
sensor.timestamp = timezone.now()

end_time = time.time()
processing_time = round((end_time - start_time) * 1000, 2)

sensor.processing_time = processing_time
sensor.save()

channel_layer = get_channel_layer()
async_to_sync(channel_layer.group_send)(
    f'device_{name}',
    {
```

```
                'type': 'send_device_state',
                'data': {
                    'name': sensor.name,
                    'blocked': sensor.blocked,
                    'work': sensor.work,
                }
            }
        )

        threshold = timedelta(minutes=5)
        time_since_last_update = timezone.now() - sensor.timestamp
        if time_since_last_update > threshold:
            sensor.work = True
            sensor.save()

        return HttpResponse(f'Новые данные приняты на сервер. Время обработки: {processing_time:.2f} мс')
    except Exception as e:
        return HttpResponse(f'Ошибка при обработке данных: {e}', status=500)

  return HttpResponse('Отказано!')



class DeviceStatus(APIView):
  def get(self, request, slug):
    try:
        sensor = Sensor.objects.get(slug=slug)

        device_status = {
            "name": sensor.name,
            "blocked": sensor.blocked,
            "work": sensor.work,
            "power": sensor.power,
        }

        return Response(device_status, status=status.HTTP_200_OK)
    except Sensor.DoesNotExist:
```

```
            return Response({"error": "Устройство не найдено"},
status=status.HTTP_404_NOT_FOUND)
        except Exception as e:
            return Response({"error": f"Произошла ошибка: {e}"},
status=status.HTTP_500_INTERNAL_SERVER_ERROR)



def get_csrf_token(request):
    csrf_token = get_token(request)
    return JsonResponse({'csrf_token': csrf_token})



def esp_update(request):
    if request.method == 'POST':
        firmware_file = request.FILES['firmware']
        device_name = os.path.splitext(firmware_file.name)[0]
        firmware_path = os.path.join(settings.MEDIA_ROOT, 'firmwares',
f'{device_name}.bin')
        version = request.POST.get('firmware_version', '')


        if Firmware.objects.filter(version=version).exists():
            messages.error(request, 'Прошивка с такой версией уже существует',
extra_tags='bg-gradient-danger')
            return redirect('sensor_list')


        os.makedirs(os.path.dirname(firmware_path), exist_ok=True)


        with open(firmware_path, 'wb+') as destination:
            for chunk in firmware_file.chunks():
                destination.write(chunk)


        actual_filename = f'{device_name}.bin'
        firmware = Firmware(version=version, file=actual_filename)
        firmware.save()


        messages.success(request, 'Прошивка успешно загружена', extra_tags='bg-
gradient-info')
        return redirect('sensor_list')
```

```
        return redirect('sensor_list')


    def get_latest_firmware(request):
        try:
            latest_firmware = Firmware.objects.latest('uploaded_at')
            filename = latest_firmware.file.name
            return JsonResponse({'version': latest_firmware.version, 'url': filename})
        except ObjectDoesNotExist:
            return JsonResponse({'error': 'Прошивки не найдены'}, status=404)


    def download_firmware(request, version):
        firmware = get_object_or_404(Firmware, version=version)
return FileResponse(firmware.file)
```