МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«Рязанский государственный радиотехнический университет имени В.Ф. Уткина»**

«К защите»

Заведующий кафедрой ВПМ

Овечкин Г.В.

«9» июня 2023 г.

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ**

**РАБОТА**

**(бакалавриат)**

на тему

**«Разработка программного обеспечения для автоматизации работы специалистов технической поддержки»**

Направление подготовки: 09.03.04 Программная инженерия

Наименование ОПОП: Программная инженерия

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (Баранчиков П.А.)

Обучающийся \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (Шишкин А.И.)

**РЕФЕРАТ**

Пояснительная записка с.84, таблиц 11, рис. 32, источников 15.

СЛУЖБА ТЕХНИЧЕСКОЙ ПОДДЕРЖКИ, АППАРАТНОЕ ОБЕСПЕЧЕНИЕ, ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ.

Цель проекта – разработка программного обеспечения для автоматизации работы специалистов технической поддержки.

Средства разработки – среда программирования IntelliJ IDEA 2022.2.3, библиотека React, Spring Framework, языки программирования Java и JavaScript, СУБД PostgreSQL.

Назначение разработанных средств – обеспечение эффективного функционирования службы технической поддержки в компаниях, работающих с программным и аппаратным обеспечением.

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 5](#_Toc136885842)

[1 ОБОСНОВАНИЕ АКТУАЛЬНОСТИ РАЗРАБОТКИ 8](#_Toc136885843)

[1.1 Анализ предметной области 8](#_Toc136885844)

[1.2 Анализ существующих подходов и обоснование актуальности разработки. 11](#_Toc136885845)

[1.3 Требования к программному обеспечению 13](#_Toc136885846)

[1.4 Выбор средств разработки 15](#_Toc136885847)

[1.5 Возможность внедрения 16](#_Toc136885848)

[1.6 Подведение итогов исследования. 18](#_Toc136885849)

[2 РАЗРАБОТКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ 19](#_Toc136885850)

[2.1 Анализ требований к системе 19](#_Toc136885851)

[2.1.1 Диаграмма вариантов использования 19](#_Toc136885852)

[2.1.2 Функциональные требования 20](#_Toc136885853)

[2.1.3 Нефункциональные требования 22](#_Toc136885854)

[2.2 Проектирование архитектуры системы 22](#_Toc136885855)

[2.3 Проектирование базы данных 25](#_Toc136885856)

[2.3.1 Концептуальная модель данных 25](#_Toc136885857)

[2.4 Заполнение сущностей атрибутами 34](#_Toc136885858)

[2.5 Разработка серверной части. 41](#_Toc136885859)

[3 ТЕСТИРОВАНИЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ 47](#_Toc136885860)

[4 ПРОГРАММНАЯ ДОКУМЕНТАЦИЯ 48](#_Toc136885861)

[4.1 Применение программы 48](#_Toc136885862)

[ЗАКЛЮЧЕНИЕ 49](#_Toc136885863)

[СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 50](#_Toc136885864)

[ПРИЛОЖЕНИЕ 52](#_Toc136885865)

ВВЕДЕНИЕ

В современном информационном обществе, где техническая поддержка является важной составляющей успешного функционирования организаций, автоматизация этого процесса играет решающую роль. Разработка программного обеспечения для автоматизации работы службы технической поддержки становится неотъемлемым фактором в повышении эффективности, сокращении времени реагирования на запросы клиентов и улучшении обслуживания.

Целью данной дипломной работы является разработка программного обеспечения, которое обеспечит эффективное функционирование службы технической поддержки. В работе будет осуществлен анализ существующих подходов к автоматизации работы службы технической поддержки, а также рассмотрены современные технологии и методы, необходимые для реализации данного программного решения.

В ходе работы будет предложена концепция системы, которая позволит клиентам, являющимся сотрудниками компании, создавать заявки по проблемам, связанным с техникой и программным обеспечением. Заявки будут разделены по категориям, приоритету и возможности удаленного решения проблемы. Для решения задачи автоматического распределения заявок между специалистами будет разработан алгоритм, учитывающий приоритет, нагрузку и доступность каждого специалиста.

В ходе разработки программного обеспечения будет использован набор современных технологий. Для серверной части будет выбран язык программирования Java, а именно, фреймворк Spring, включая его модули Spring Boot, Spring Data JPA и Spring Security, который будет использоваться для упрощения и ускорения разработки, конфигурации и развертывания веб-приложения, а также для обеспечения безопасности и удобного взаимодействия с базой данных.

Для клиентской части программного обеспечения будет применяться React - популярный и мощный JavaScript-фреймворк, позволяющий создавать современные и отзывчивые пользовательские интерфейсы. React обладает большим сообществом разработчиков и широким выбором инструментов для разработки.

В качестве системы управления базами данных будет использован PostgreSQL, что обеспечит надежное хранение и быстрый доступ к данным, а также поддержку расширенных возможностей для масштабирования и обработки больших объемов информации.

В дальнейшем тексте дипломной работы будет детально рассмотрены следующие разделы, необходимые для достижения поставленной задачи.

* Обоснование актуальности разработки.
  + Анализ предметной области.
  + Анализ существующих подходов и обоснование актуальности разработки.
  + Требования к программному обеспечению.
  + Выбор средств для разработки.
  + Возможность внедрения.
  + Подведение итогов исследования.
* Разработка программного обеспечения.
  + Анализ требований к системе.
  + Проектирование архитектуры системы.
  + Проектирование базы данных.
  + Проектирование основных алгоритмов.
  + Разработка приложения с основной бизнес-логикой.
  + Разработка приложения для системы уведомлений.
  + Контейнеризация приложений.
* Тестирование программного обеспечения.
  + Планирование тестирования.
  + Осуществление тестирования.
  + Анализ результатов.
* Составление программной документации.
  + Описание программного продукта.
  + Руководство для сотрудника – клиента системы.
  + Руководство для специалиста технической поддержки.
  + Руководство для администратора системы.

В заключении будут подведены итоги работы, сделан обзор достигнутых результатов и выполнения поставленных задач. Будет оценена эффективность и преимущества разработанного программного обеспечения в контексте автоматизации работы службы технической поддержки. Также будут высказаны рекомендации по дальнейшему развитию и улучшению системы, а также возможности ее применения в реальных условиях.

Таким образом, разработка программного обеспечения с использованием указанных технологий будет способствовать созданию клиент ориентированной системы технической поддержки, обеспечивая эффективное и качественное обслуживание сотрудников компании в их запросах и проблемах, связанных с техникой и программным обеспечением.

# ОБОСНОВАНИЕ АКТУАЛЬНОСТИ РАЗРАБОТКИ

## Анализ предметной области

Анализ предметной области является важным этапом в разработке программного обеспечения для автоматизации работы службы технической поддержки. В данном исследовании мы изучили основные аспекты и особенности предметной области, чтобы лучше понять требования и потребности пользователей, а также определить оптимальные подходы к разработке системы.

Одним из важных аспектов предметной области является эффективное управление заявками и обработка проблем клиентов. В технической поддержке возникает множество запросов от пользователей, связанных с техническими проблемами и неисправностями, а также с запросами на установку, настройку или обновление программного обеспечения. Цель автоматизации работы службы технической поддержки заключается в оптимизации процессов обработки заявок, сокращении времени реакции и улучшении качества обслуживания клиентов.

Также важным аспектом предметной области является взаимодействие между клиентами и специалистами технической поддержки, достичь которого можно посредством интеграции пользователей программного обеспечения с мессенджером Telegram.

Исследование позволило выявить необходимость удобных средств коммуникации, таких как система комментариев и возможность прикрепления файлов к заявкам. Это обеспечивает эффективное взаимодействие, обмен информацией и улучшает качество обслуживания клиентов.

Анализ предметной области является важным этапом разработки программного обеспечения для автоматизации работы службы технической поддержки. Он позволяет более глубоко понять особенности и требования данной области, выявить проблемы, с которыми сталкиваются клиенты и специалисты технической поддержки, а также определить функциональные и нефункциональные требования к системе.

В ходе анализа предметной области были проанализированы основные аспекты работы службы технической поддержки. Рассмотрены типичные проблемы, с которыми могут столкнуться клиенты, связанные как с техникой (например, неисправность оборудования, необходимость замены или ремонта), так и с программным обеспечением (сбои, ошибки, проблемы с установкой или удалением программ).

Также были изучены особенности процесса обработки заявок. Были выявлены этапы, которые включают создание заявки, ее регистрацию в системе, распределение на исполнение, выполнение работ, закрытие и отслеживание статуса заявки. Анализ процесса позволил определить ключевые требования к системе, такие как удобный интерфейс для создания и отслеживания заявок, возможность прикрепления файлов и комментариев, эффективное распределение заявок между специалистами, своевременное получение уведомлений о новых заявках.

Исходя из проведенного анализа предметной области, были выявлены основные проблемы, требования и потребности, которые должны быть учтены при разработке программного обеспечения для автоматизации работы службы технической поддержки. Некоторые из основных выводов анализа предметной области включают.

* Комплексность проблем: клиенты могут сталкиваться с различными проблемами, связанными как с аппаратным обеспечением, так и с программным обеспечением. Это требует разработки системы, способной обрабатывать разнообразные типы заявок и предоставлять соответствующие инструменты для их решения.
* Необходимость оперативного реагирования: клиенты ожидают быстрого и эффективного решения своих проблем. Система должна обеспечивать оперативное распределение заявок между специалистами, отслеживание их статуса, своевременное информирование специалистов о новых заявках.
* Управление процессом: эффективное управление процессом работы службы технической поддержки включает регистрацию и классификацию заявок, планирование и отслеживание работ, а также анализ статистики и отчетность. Система должна предоставлять соответствующие инструменты для этих задач, упрощая процесс управления и повышая его прозрачность.
* Взаимодействие между специалистами и клиентами: коммуникация является важной частью работы службы технической поддержки. Система должна обеспечивать возможность обмена сообщениями и файлами, посредством интеграции с мессенджером Telegram, между специалистами и клиентами, создавая эффективный канал связи и упрощая процесс общения.
* Гибкость и расширяемость: в современной среде бизнеса требования и потребности могут изменяться со временем. Поэтому система должна быть гибкой и легко расширяемой, чтобы адаптироваться к новым требованиям и интегрироваться с другими системами.
* Кроссплатформенность: в современном мире программное обеспечение для удовлетворения потребностей пользователей должно предоставлять возможность работы на любом устройстве, что обеспечит специалистам и клиентам удобный и быстрый доступ к системе.

Исследование предметной области позволяет лучше понять основные аспекты работы службы технической поддержки и определить ключевые требования, которые должны быть учтены при разработке программного обеспечения. Это обеспечивает основу для успешной разработки системы, способной повысить эффективность и качество работы службы технической поддержки, улучшить взаимодействие между специалистами и клиентами, а также обеспечить более оперативное и надежное решение проблем.

## Анализ существующих подходов и обоснование актуальности разработки.

Анализ существующих подходов к автоматизации работы службы технической поддержки позволяет выявить сильные и слабые стороны различных решений и обосновать актуальность разработки программного обеспечения, описанного выше.

Существующие подходы к автоматизации работы службы технической поддержки включают использование тикет-систем, электронной почты, чат-ботов и специализированных программных решений. Каждый из этих подходов имеет свои преимущества и недостатки.

Тикет-системы и электронная почта позволяют организовать процесс обработки запросов клиентов, однако они часто требуют ручной обработки и недостаточно эффективны при большом объеме заявок. Кроме того, в них отсутствует возможность автоматического назначения заявок на специалистов и контроля времени реагирования.

Чат-боты становятся все более популярными и предлагают быстрый и интерактивный способ взаимодействия с клиентами. Однако они не всегда способны предоставить полноценное решение проблемы клиента и требуют наличия заранее подготовленных ответов и сценариев.

Разработка программного обеспечения, описанного выше, предлагает решение, объединяющее преимущества различных подходов и обеспечивающее более эффективную автоматизацию работы службы технической поддержки. Возможность создания заявок, разделение их по категориям, приоритету и возможности удаленного решения проблемы позволяет структурировать процесс обработки запросов клиентов и повысить эффективность работы специалистов.

Особенно актуальной является функциональность автоматического назначения заявок на специалистов. Это позволяет оптимизировать процесс распределения работы между сотрудниками и гарантировать быстрое реагирование на запросы клиентов. Алгоритм автоматического назначения заявок, зависящий от приоритета и занятости специалиста, обеспечивает справедливое распределение нагрузки и минимизацию простоев в обработке заявок.

Для обеспечения эффективного взаимодействия между специалистами технической поддержки и клиентами в разрабатываемой системе будет предусмотрена функциональность комментирования и прикладывания файлов к заявкам.

Такой функционал общения и обмена файлами существенно улучшит качество и эффективность работы службы технической поддержки. Специалисты смогут получать более полную информацию о проблемах клиентов, а клиенты смогут быстрее и удобнее предоставлять необходимые материалы для анализа и решения проблемы.

Кроме того, разработанное программное обеспечение позволяет иметь централизованную систему управления заявками, что способствует лучшей организации и отслеживанию процесса обработки. Специалисты технической поддержки могут видеть все активные заявки, их приоритеты и статусы, что позволяет им эффективно планировать свою работу и сосредотачиваться на наиболее срочных и важных задачах.

Актуальность разработки программного обеспечения для автоматизации работы службы технической поддержки обосновывается не только повышением эффективности и качества обслуживания клиентов, но и сокращением затрат на техническую поддержку. Автоматизация позволяет сократить время на обработку и решение проблем, уменьшить необходимость в привлечении большого количества сотрудников и минимизировать человеческий фактор, который может привести к ошибкам или пропуску заявок.

## Требования к программному обеспечению

Базируясь на особенностях и анализе предметной области, вот некоторые требования к разрабатываемому веб-приложению для автоматизации работы службы технической поддержки.

* Авторизация и аутентификация: пользователи должны иметь возможность регистрации, аутентификации и управления своими учетными записями. Необходимо обеспечить безопасное хранение и передачу пользовательских данных.
* Управление заявками: приложение должно предоставлять возможность создания, отслеживания, редактирования и удаления заявок. Каждая заявка должна иметь уникальный идентификатор, описание проблемы, приоритет, статус, отметку о назначенном ответственном лице, фиксировать время и дату создания и решения проблемы.
* Учет состояние и владельцев техники: система должна иметь возможность отслеживать состояние техники, такое как новое, списанное, ранее использованное или имеющее дефекты. Это позволит эффективно управлять техническими ресурсами компании. Каждая единица техники должна быть связана с конкретным владельцем, который может быть сотрудником компании. Также должна быть возможность указать, что техника находится на складе офиса и не закреплена за каким-либо сотрудником.
* Автоматическое распределение заявок: система должна автоматически назначать специалиста по специальному алгоритму, который будет описан в дальнейшем, в случае если превышен срок ожидания принятие заявки в работу.
* Комментарии и обсуждение: пользователи должны иметь возможность добавлять комментарии к заявкам, обсуждать проблему со специалистами и прикладывать дополнительные файлы, необходимые для решения проблемы. Комментарии должны быть связаны с конкретными заявками и отображаться в хронологическом порядке.
* Масштабируемость и быстрое развертывание: система иметь возможность развертывания в любой среде. Для этого приложение необходимо упаковать в единый образ, который запускается в изолированном контейнере. При повышении трафика такой подход позволит увеличить количество экземпляров приложения и обеспечит горизонтальное масштабирование.
* Информирование о новых заявках: специалисты технической поддержки должны иметь возможность, путем взаимодействия системы с мессенджером Telegram, своевременно получать уведомления о назначенных на них заявках.
* Фильтрация и сортировка: Пользователи должны иметь возможность фильтровать заявки по различным параметрам, таким как статус, приоритет, дата создания, категории, возможности онлайн-решения и т.д. Также следует предусмотреть возможность сортировки заявок по заданным критериям, например, по приоритету или категории.
* Многопользовательская поддержка: система должна поддерживать работу нескольких пользователей одновременно, позволяя им работать с заявками и обмениваться информацией в реальном времени. Разграничение прав доступа должно быть реализовано, чтобы пользователи могли получать доступ только к необходимым им данным и функциональности.
* Безопасность: безопасность данных является важным аспектом разработки программного продукта. Приложение должно обеспечивать защиту данных пользователя, использовать шифрование при хранении паролей и иметь механизмы аутентификации и авторизации для предотвращения несанкционированного доступа.
* Документация и руководство пользователя: разработка должна сопровождаться документацией и руководством пользователя, которые помогут пользователям быстро освоить приложение и эффективно использовать его функциональность.
* Поддержка различных браузеров и устройств: приложение должно быть совместимо с различными популярными веб-браузерами (например, Chrome, Firefox, Safari, Edge) и различными устройствами (например смартфон, планшет, компьютер) и обеспечивать одинаковое качество работы.

## Выбор средств разработки

Определение выбранных технологий для разработки программного обеспечения базируется на особенностях разрабатываемой системы, а также учете преимуществ open-source продуктов. Основываясь на этих критериях, были выбраны следующие технологии.

* Spring: является одним из наиболее популярных и широко используемых фреймворков для разработки приложений на языке Java. Он обеспечивает мощный инструментарий для создания эффективных и надежных приложений, включая модули Spring Boot, Spring Data JPA и Spring Security. Spring Boot облегчает настройку и развертывание приложения, а Spring Data JPA предоставляет удобный способ взаимодействия с базой данных. Spring Security обеспечивает безопасность приложения и управление доступом пользователей.
* React: является популярной библиотекой JavaScript для создания пользовательских интерфейсов. Он позволяет разрабатывать мощные и интерактивные веб-приложения, основанные на компонентном подходе. Благодаря виртуальной DOM и эффективному управлению состоянием, React обеспечивает быструю отрисовку интерфейса и удобное взаимодействие с пользователем.
* PostgreSQL: предлагает мощные возможности хранения и управления данными. PostgreSQL отличается высокой надежностью, производительностью и поддержкой различных средств обеспечения целостности данных. Это позволяет эффективно работать с большим объемом данных, необходимых для работы службы технической поддержки.
* Docker: предоставляет инструменты для развертывания и управления контейнерами, что упрощает процесс развертывания и обеспечивает единообразие окружения между разработкой, тестированием и производством. Docker обеспечивает изоляцию приложения, упрощает масштабирование и повышает гибкость разработки и развертывания программного обеспечения.

Таким образом, выбор open-source технологий, таких как Spring, React, PostgreSQL, Docker, позволяет компании разрабатывать программное обеспечение с использованием мощных инструментов, обеспечивает гибкость, надежность, безопасность и снижает затраты на лицензии.

В качестве среды разработки выбрана IntelliJ IDEA является одной из ведущих интегрированных сред разработки (IDE). Одним из основных факторов при выборе IDE является удобство для команды разработчиков и возможность работать со всем выбранным набором технологий в одном месте. Исходя из этих критериев IntelliJ IDEA выбрана в качестве отличного инструмента для работы с исходным кодом при разработке программного обеспечения для автоматизации работы службы технической поддержки.

## Возможность внедрения

Возможность внедрения системы технической поддержки может принести несколько значительных преимуществ:

* Улучшение обслуживания клиентов: внедрение системы технической поддержки позволит компании оперативно реагировать на запросы и проблемы клиентов. Это повысит качество обслуживания, ускорит процесс решения проблем и улучшит общую удовлетворенность клиентов.
* Эффективное управление запросами и проблемами: система технической поддержки позволит эффективно управлять запросами и проблемами клиентов. Все обращения будут централизованно отслеживаться, адекватно распределяться между специалистами и контролироваться в рамках установленных SLA (уровней обслуживания).
* Улучшение коммуникации и взаимодействия: система технической поддержки обеспечит эффективное взаимодействие между специалистами и клиентами. Будет возможность обмена сообщениями, прикрепления файлов и отслеживания истории коммуникации. Это сократит время и усилия, затрачиваемые на общение и уточнение информации.
* Анализ и отчетность: система будет предоставлять инструменты для анализа данных, отчетности и мониторинга производительности. Это поможет компании выявить общие тенденции, узкие места в процессе обслуживания и принимать информированные решения для улучшения работы службы технической поддержки.
* Повышение производительности и эффективности: внедрение системы технической поддержки позволит автоматизировать и оптимизировать многие рутинные задачи, такие как маршрутизация запросов, отслеживание статусов и приоритетов, управление временем реагирования и т.д. Это сэкономит время и ресурсы, повысит производительность и эффективность работы службы технической поддержки.

В целом, внедрение системы технической поддержки приведет к повышению качества обслуживания клиентов, оптимизации внутренних процессов и повышению эффективности работы службы технической поддержки. Это позволит компании повысить уровень удовлетворенности сотрудников и конкурентоспособность бизнеса.

## Подведение итогов исследования.

Анализ предметной области помог определить основные цели и задачи разрабатываемого программного обеспечения, а также выбрать подходящие технологии и инструменты для его разработки. На основе проведенного исследования предметной области мы можем приступить к следующему этапу - проектированию и разработке программного обеспечения для автоматизации работы службы технической поддержки.

Также был проведен обзор существующих решений на рынке. Исследование показало, что существует ряд программных продуктов, предназначенных для автоматизации работы службы технической поддержки. Однако, большинство из них имеют ограниченный функционал или не удовлетворяют полностью требованиям и потребностям конкретных компаний. Это подтверждает актуальность разработки собственной системы, специально адаптированной к нуждам и особенностям конкретной службы технической поддержки.

# РАЗРАБОТКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

## Анализ требований к системе

Анализ требований к программному обеспечению подразумевает описание функциональности системы, выделение функциональных и нефункциональных требований.

### Диаграмма вариантов использования

Диаграмма вариантов использования (Use Case Diagram) является графическим инструментом моделирования, который используется для описания функциональности системы с точки зрения ее акторов (пользователей или внешних систем) [1]. Она помогает понять, как различные акторы взаимодействуют с системой и как система отвечает на их запросы.

Основная цель диаграммы вариантов использования заключается в идентификации и описании функциональных требований системы. Она помогает разработчикам и заинтересованным сторонам получить ясное представление о том, как система должна взаимодействовать с ее окружением.

В разрабатываемом программном продукте можно выделить следующих акторов.

* Пользователь.
* Клиент.
* Специалист ТХП.

Диаграмма вариантов использования, разрабатываемой системы показана на рисунке 1.

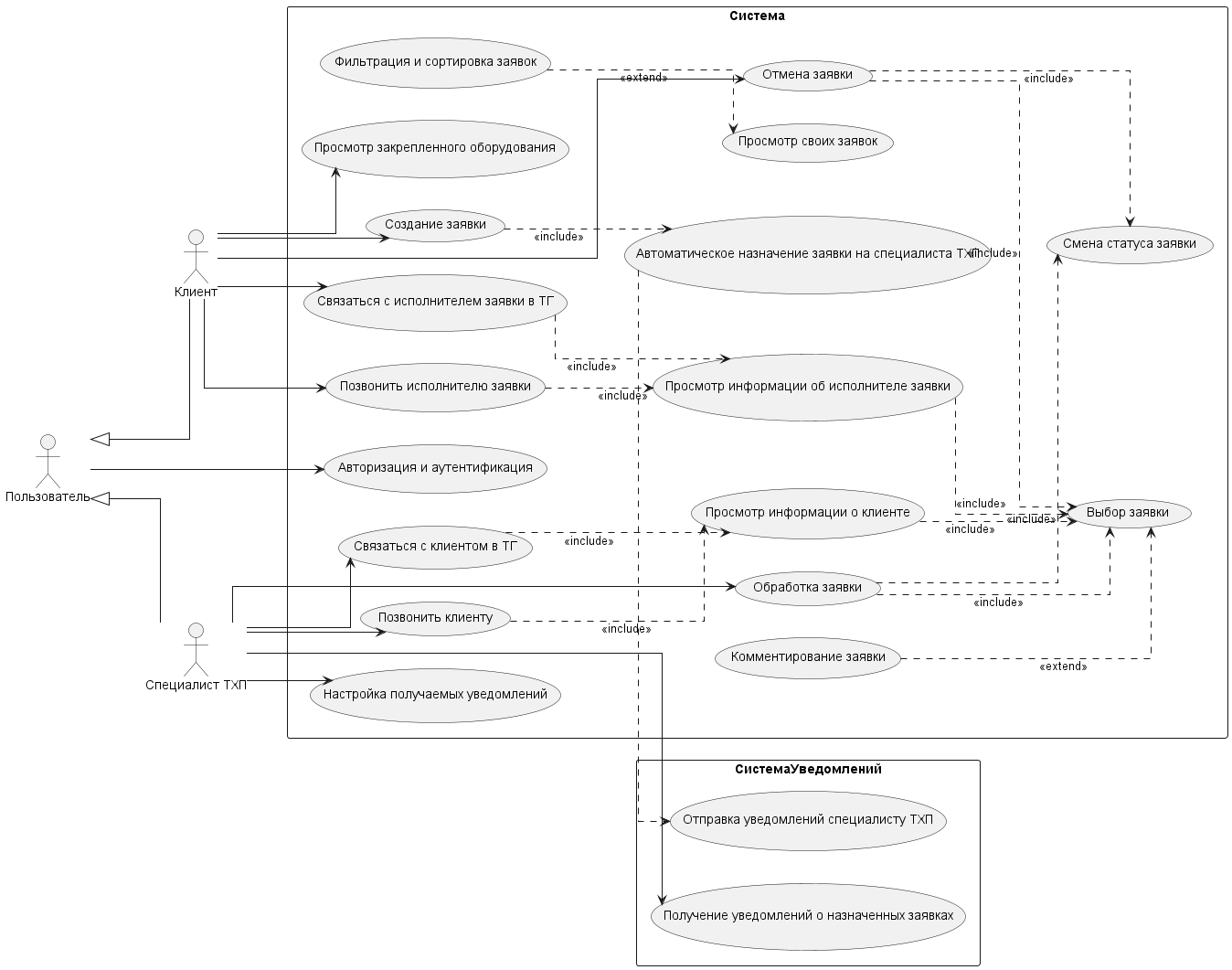


Рисунок - Диаграмма вариантов использования, разрабатываемого программного продукта

### Функциональные требования

Функциональные требования определяют, какая функциональность и возможности должны быть реализованы в разрабатываемой системе. Они описывают, как система должна взаимодействовать с пользователями и выполнять определенные действия и операции. Функциональные требования определяют конкретные задачи, которые система должна выполнять, и описывают ожидаемые результаты и поведение системы в различных сценариях использования.

Исходя из построенной диаграммы можно выделить следующие функциональные требования.

* Система должна предоставлять возможность осуществлять авторизацию и аутентификацию пользователей в системе.
* Система должна реализовывать автоматическое назначение исполнителя для созданной заявки.
* Система должна обеспечивать отправку уведомлений о созданной заявке.
* Специалист ТХП должен иметь возможность отслеживать состояние техники, включая информацию о владельце и статусе (новое, списанное, ранее использованное, с дефектами).
* Специалист ТХП должен иметь возможность осуществлять коммуникацию с клиентом.
* Специалист ТХП должен иметь возможность обрабатывать заявки клиентов.
* Специалист ТХП должен иметь возможность настраивать и получать уведомления о новых назначенных на него заявках.
* Специалист ТХП должен иметь возможность просматривать информацию о клиенте и закрепленном за ним оборудовании.
* Клиент должен иметь возможность создания заявок.
* Клиент должен иметь возможность отслеживания состояния своих заявок.
* Клиент должен иметь возможность отменить заявку.
* Клиент должен иметь возможность просмотреть информацию об исполнителе заявки.
* Клиент должен иметь возможность отслеживать состояние, закрепленной за ним техники.
* Клиент должен иметь возможность дополнять сведения о заявке с помощью комментирования и коммуникации со специалистом по контактным данным.
* Система должна обеспечивать фильтрацию и сортировку заявок.

### Нефункциональные требования

Нефункциональные требования определяют атрибуты и характеристики системы, которые не связаны непосредственно с ее функциональностью, но оказывают влияние на ее общую производительность, качество и эффективность. Они описывают такие аспекты, как надежность, производительность, безопасность, масштабируемость, удобство использования и другие атрибуты системы.

Для разрабатываемого программного продукта можно выделить следующие нефункциональные требования.

* Система должна быть способна масштабироваться горизонтально, чтобы справляться с увеличением нагрузки.
* Система должна обеспечивать аутентификацию и авторизацию пользователе.
* Доступ к конфиденциальным данным, таким как личная информация клиентов, должен быть ограничен и защищен от несанкционированного доступа.
* Веб-интерфейс системы должен быть адаптивным, одинаково хорошо работать на различных браузерах и устройствах.

## Проектирование архитектуры системы

Проектирование системы — это процесс создания архитектурного и технического решения для разрабатываемой системы. Оно включает в себя определение структуры системы, ее компонентов, интерфейсов, а также выбор подходящих технологий, архитектурных шаблонов и принципов, которые будут использоваться при разработке [2].

При проектировании разрабатываемой программной системы было принято решение использовать классическую трехзвенную архитектуру, состоящую из клиентского, серверного уровней и базы данных. Этот выбор обоснован рядом преимуществ, которые такая архитектура предоставляет.

* Распределение ответственности: позволяет четко разделить функциональность и ответственность между различными уровнями системы. Клиентский уровень отвечает за пользовательский интерфейс и взаимодействие с пользователем, серверный уровень обрабатывает бизнес-логику и взаимодействует с базой данных, а базовый уровень отвечает за хранение и обработку данных. Это способствует более четкому распределению задач и обеспечивает модульность и гибкость системы.
* Масштабируемость и гибкость: позволяет гибко масштабировать и расширять систему. Каждый уровень может масштабироваться независимо от других, что обеспечивает горизонтальное масштабирование и легкую добавление новых функциональных модулей. Это позволяет системе эффективно реагировать на изменения в требованиях и нагрузке.
* Удобство сопровождения и разработки: трехзвенная архитектура обеспечивает четкое разделение функций и ответственности между уровнями, что упрощает сопровождение и разработку системы. Каждый уровень может быть разработан и тестирован независимо, что улучшает модульность и повторное использование кода. Это также способствует параллельной разработке и позволяет командам специализироваться на определенных уровнях.
* Безопасность и контроль доступа: позволяет централизованно управлять безопасностью и контролем доступа к системе. Серверный уровень может обеспечивать аутентификацию и авторизацию пользователей, а также применять механизмы шифрования для защиты данных.
* Интеграция и повторное использование: облегчает интеграцию системы с другими внешними сервисами и компонентами. Каждый уровень может быть независимо интегрирован с внешними системами, обеспечивая гибкую и модульную интеграцию. Более того, благодаря четкому разделению функций и модульности каждого уровня, компоненты системы могут быть повторно использованы в других проектах или модулях.

Разрабатываемый программный продукт представляет из себя два клиент-серверных приложения с общей базой данных.

* Веб-приложение с основной бизнес-логикой – пользовательский интерфейс (REACT-UI) и серверная часть (MAIN-SERVER), приложение реализует авторизацию и аутентификацию пользователей, создание заявок, обработку заявок, назначение заявки на специалиста и осуществляет отправку запроса об уведомлении к серверной части (NOTIFY-SERVER) второго приложения.
* Второе приложение – Telegram-бот (TELEGRAM-BOT и NOTIFY-SERVER), реализует авторизацию пользователей и прием запросов на отправку уведомлений от основного приложения. На основании, принятых запросов, формирует сообщение о созданной заявке и отправляет в Telegram специалиста.

Архитектура, разрабатываемого программного обеспечения, приведена на рисунке 2.

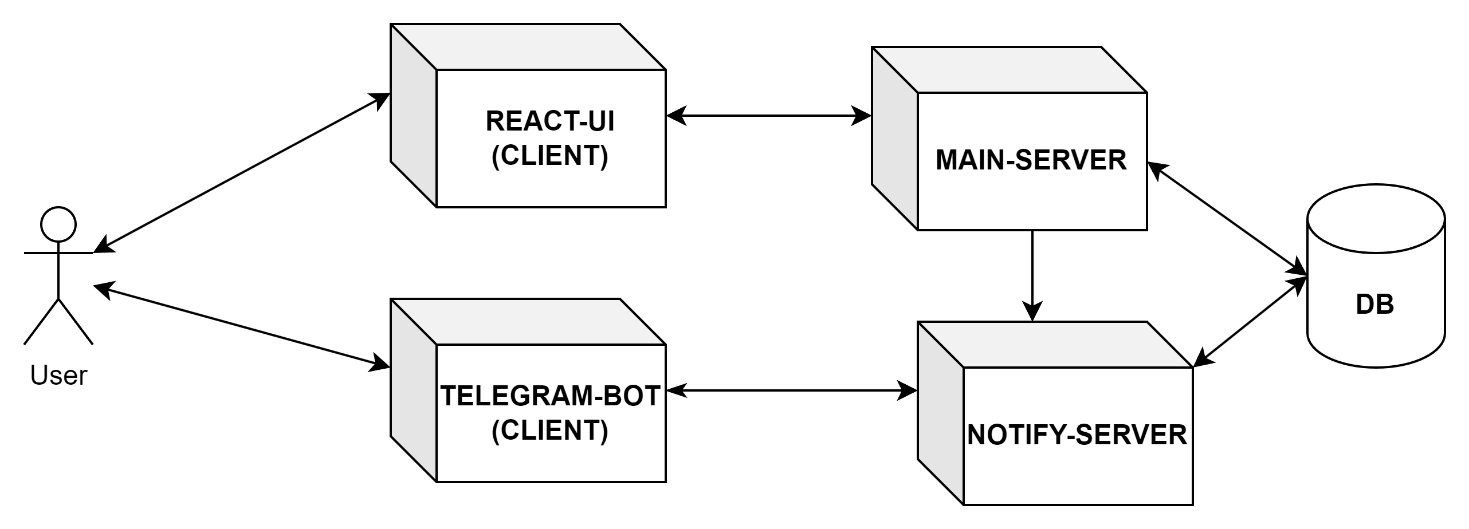


Рисунок 2 - Клиент-серверная трехзвенная архитектура

## Проектирование базы данных

Процесс проектирования базы данных состоит из нескольких связанных этапов: создание концептуальной модели данных, заполнение выделенных сущностей атрибутами, создание логической модели данных и создание физической модели данных.

### Концептуальная модель данных

Концептуальное проектирование базы данных — это процесс разработки абстрактной модели данных, которая представляет собой высокоуровневое описание сущностей, атрибутов и связей между ними в информационной системе. Оно основывается на анализе требований и целей системы, и направлено на создание структуры данных, независимой от конкретных технологий и реализаций.

Основная цель концептуального проектирования базы данных заключается в создании ясной и понятной модели данных, которая будет служить основой для дальнейшей разработки физической базы данных и ее реализации с использованием конкретных технологий, и СУБД [3].

В выбранной предметной области можно выделить следующие сущности.

* Employee – хранит общую информацию о сотрудниках компании. Первичный ключ: Staff\_number.
* Client, Technical\_support\_specialist – отражают роли сотрудников компании в системе: клиент, администратор, специалист ТХП, соответственно. Первичный ключ: Staff\_number.
* Office – хранит информацию об офисах компании, в которых работают сотрудники. Первичный ключ: ID.
* Workplace – хранит информацию о рабочем месте сотрудников в офисе. Первичный ключ: ID.
* Departament – хранит информацию об отделах, в которых работают сотрудники. Первичный ключ: ID.
* Post – хранит информацию о должностях сотрудников. Первичный ключ: ID.
* Device – хранит информацию о технике. Первичный ключ: Serial\_number.
* Device\_type – хранит информацию о типах техники. Первичный ключ: ID.
* Device\_condition – хранит информацию о возможных состояниях техники. Первичный ключ: ID.
* Software – хранит информацию о программном обеспечении. Первичный ключ: ID.
* Sofware\_type – хранит информацию о типах программного обеспечения. Первичный ключ: ID.
* Application – хранит общую информацию о заявках. Первичный ключ: ID.
* Status – хранит информацию о возможных состояниях заявки. Первичный ключ: ID.
* Priority – хранит информацию о возможных значениях приоритета заявки. Первичный ключ: ID.
* Application\_device – хранит информацию о заявках, связанных с техникой. Первичный ключ: ID.
* Application\_software – хранит информацию о заявках, связанных с программным обеспечением. Первичный ключ: ID.
* Application\_device\_type – хранит информацию о типах заявок, связанных с техникой.
* Application\_software\_type – хранит информацию о типах заявок, связанных с программным обеспечением.
* Comment – хранит информацию о комментариях сотрудников, оставленных к заявке. Первичный ключ: ID.
* Attachment – хранит информацию о приложенных сотрудниками файлах, оставленных к заявке. Первичный ключ: ID.
* Notification – хранит информацию о подписках сотрудника на приоритеты заявки.

На основании, выделенных сущностей и требований, в системе можно выделить следующие связи, диаграммы связей показаны на рисунках 3 – 27:

* Рабочее место находится в офисе:

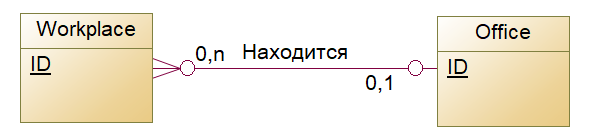


Рисунок - Диаграмма связи «Рабочее место находится в офисе»

* Сотрудник занимает рабочее место.

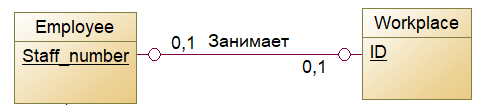


Рисунок - Диаграмма связи «Сотрудник занимает рабочее место»

* Сотрудник работает в отделе.

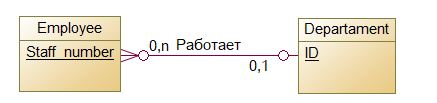


Рисунок - Диаграмма связи «Сотрудник работает в отделе»

* Сотрудник занимает пост.

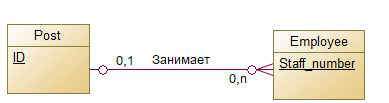


Рисунок - Диаграмма связи «Сотрудник занимает пост»

* Клиент, специалист ТХП являются сотрудником.



Рисунок - Диаграмма связи «Клиент, специалист ТХП являются сотрудником»

* Клиент прикладывает вложения.



Рисунок - Диаграмма связи «Клиент прикладывает вложения»

* Клиент оставляет комментарии.



Рисунок - Диаграмма связи «Клиент оставляет комментарии»

* Специалист ТХП прикладывает вложения.



Рисунок - Диаграмма связи «Специалист ТХП прикладывает вложения»

* Специалист ТХП оставляет комментарии.



Рисунок - Диаграмма связи «Специалист ТХП»

* Клиент создает заявки.

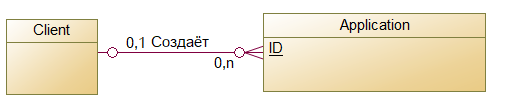


Рисунок - Диаграмма связи «Клиент создает заявки»

* Специалист ТХП обрабатывает заявки.

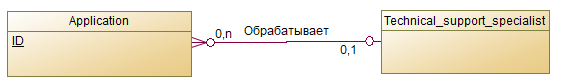


Рисунок - Диаграмма связи «Специалист ТХП обрабатывает заявки»

* Заявки бывают двух видов: связанные с техникой и связанные с программным обеспечением.

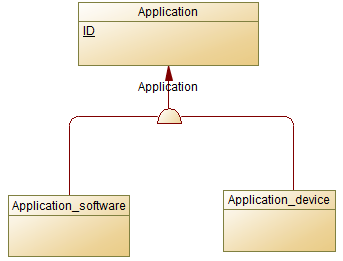


Рисунок - Диаграмма связи «Заявки бывают двух видов: связанные с техникой и связанные с программным обеспечением»

* Заявка содержит комментарии.

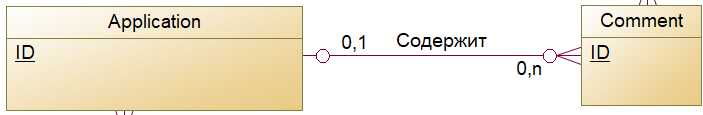


Рисунок - Диаграмма связи «Заявка содержит комментарии»

* Заявка содержит вложения.

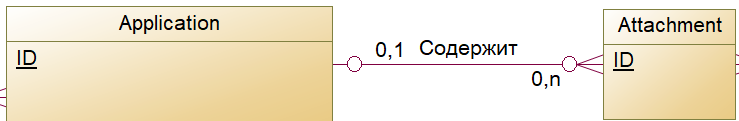


Рисунок - Диаграмма связи «Заявка содержит вложения»

* Заявка имеет приоритет.

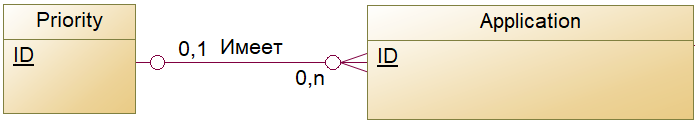


Рисунок - Диаграмма связи «Заявка имеет приоритет»

* Заявка имеет статус.

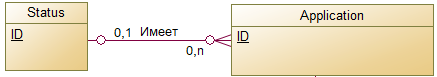


Рисунок - Диаграмма связи «Заявка имеет статус»

* Заявка, связанная с техникой, имеет тип.

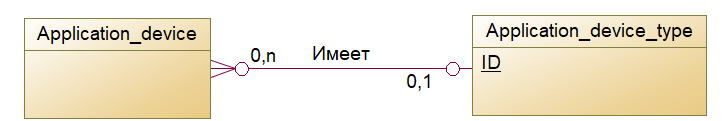


Рисунок - Диаграмма связи «Заявка, связанная с техникой, имеет тип»

* Заявка, связанная с техникой, относится к технике.

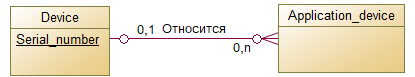


Рисунок - Диаграмма связи «Заявка, связанная с техникой, относится к технике»

* Техника имеет тип.

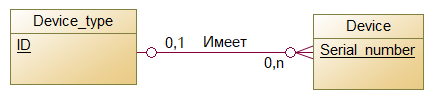


Рисунок - Диаграмма связи «Техника имеет тип»

* Техника имеет состояние.

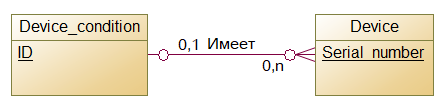


Рисунок - Диаграмма связи «Техника имеет состояние»

* Заявка, связанная с программным обеспечением, имеет тип.

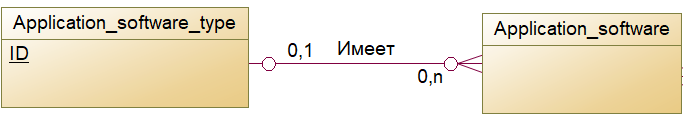


Рисунок - Диаграмма связи «Заявка, связанная с программным обеспечением, имеет тип»

* Заявка, связанная с программным обеспечением, относится к программному обеспечению.

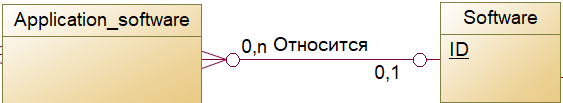


Рисунок - Диаграмма связи «Заявка, связанная с программным обеспечением, относится к программному обеспечению»

* Программное обеспечение имеет тип.

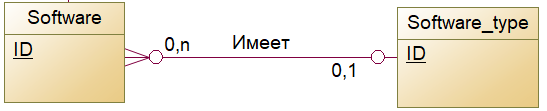


Рисунок - Диаграмма связи «Программное обеспечение имеет тип»

* Специалист ТХП подписывается на уведомления.



Рисунок - Диаграмма связи «Специалист ТХП подписывается на уведомления»

* Уведомление относится к приоритету заявки.



Рисунок - Диаграмма связи «Уведомление относится к приоритету заявки»

На основании выделенных связей между сущностями можно построить концептуальную диаграмму базы данных программного обеспечения для автоматизации работы службы технической поддержки, изображенную на рисунке 28.



Рисунок - Концептуальная модель данных программного обеспечения для автоматизации работы службы технической поддержки.

### Заполнение сущностей атрибутами

Следующим шагом при разработке базы данных является переход от концептуальной модели к логической, этот шаг включает в себя процесс определения структуры данных, их взаимосвязей и организации хранения информации в базе данных, для чего выделим атрибуты из определенных в концептуальной модели сущностях.

* Office.
  + Id – уникальный идентификатор.
  + City – город, в котором находится офис.
  + Street – название улица офиса.
  + Build – номер дома офиса.
* Workplace.
  + Id – уникальный идентификатор.
  + Floor – этаж рабочего места в офисе.
  + Room\_number – номер комнаты на этаже.
  + Table\_number – номер рабочего стола.
  + Office\_id – уникальный идентификатор офиса.
* Role.
  + Id – уникальный идентификатор.
  + Title – название роли в системе.
* Post.
  + Id – уникальный идентификатор.
  + Title – название занимаемой должности.
* Department.
  + Id – уникальный идентификатор.
  + Title – название отдела.
* Employee.
  + Staff\_number – табельный номер сотрудника.
  + Email – электронная почта.
  + Password – хэшированный пароль.
  + Phone\_number – номер телефона.
  + Birthday – дата рождения.
  + First\_name – имя.
  + Second\_name – отчество.
  + Last\_name – фамилия.
  + Is\_online – статус, отражающий находится ли сотрудник в системе.
  + Tg\_chat\_id – уникальный идентификатор чата сотрудника в мессенджере Telegram
* DeviceType.
  + Id – уникальный идентификатор.
  + Title – название типа техники.
  + Icon – название файла с изображением техники.
* DeviceCondition.
  + Id – уникальный идентификатор.
  + Title – название состояния техники.
* Device.
  + Serial\_number – серийный номер техники.
  + Title – название техники.
  + Description – краткое описание техники.
  + Issued\_at – дата выдачи устройства сотруднику.
  + Device\_type\_id – уникальный идентификатор типа техники.
  + Device\_condition\_id – уникальный идентификатор состояния устройства.
  + Owner\_staff\_number – табельный номер владельца устройства.
* SoftwareType.
  + Id – уникальный идентификатор.
  + Title – название типа ПО.
  + Icon – название файла с изображением ПО.
* Software.
  + Id – уникальный идентификатор.
  + Title – название ПО.
  + Description – краткое описание ПО.
  + Software\_type\_id – уникальный идентификатор типа ПО.
* Priority.
  + Id – уникальный идентификатор.
  + Title – название приоритета.
  + Auto\_appointment\_hours – количество часов, через которое заявка с таким приоритетом должна быть переназначена на другого сотрудника.
  + Points – количество баллов, для вычисления занятости сотрудника.
* Status.
  + Id – уникальный идентификатор.
  + Title – название статуса.
* Application.
  + Id – уникальный идентификатор.
  + Description – краткое описание проблемы.
  + Solution – краткое описание решения проблемы.
  + Offline\_solution – требуется ли оффлайн решение.
  + Appointment\_at – дата и время переназначения.
  + Created\_at – дата и время создания.
  + Solved\_at – дата и время решения.
  + Initiator\_staff\_number – табельный номер сотрудника, создавшего заявку.
  + Executor\_staff\_number – табельный номер сотрудника, назначенного на исполнение заявки.
  + Priority\_id – уникальный идентификатор приоритета.
  + Status\_id – уникальный идентификатор статуса заявки.
* ApplicationSoftwareType.
  + Id – уникальный идентификатор.
  + Title – название типа заявки на ПО.
* ApplicationDeviceType.
  + Id – уникальный идентификатор.
  + Title – название типа заявки на технику.
* ApplicationSofware.
  + Id – уникальный идентификатор.
  + Application\_id – уникальный идентификатор заявки.
  + Software\_id – уникальный идентификатор ПО.
  + Application\_software\_type\_id – уникальный идентификатор типа заявки на ПО.
* ApplicationDevice.
  + Id – уникальный идентификатор.
  + Application\_id – уникальный идентификатор заявки.
  + Device\_serial\_number – серийный номер техники.
  + Application\_device\_type\_id – уникальный идентификатор типа заявки на технику.
* Comment.
  + Id – уникальный идентификатор.
  + Text – текст комментария.
  + Created\_at – дата и время создания.
  + Sender\_staff\_number – табельный номер сотрудника.
  + Application\_id – уникальный идентификатор заявки.
* Attachment.
  + Id – уникальный идентификатор.
  + Application\_id – уникальный идентификатор заявки.
  + Sender\_staff\_number – табельный номер сотрудника.
  + File – содержимое файла.
  + Filename – имя файла вложения.
  + Created\_at – дата и время приложения файла.
* Notification.
  + Id – уникальный идентификатор.
  + Active – статус активности подписки.
  + Subscriber\_staff\_number – табельный номер сотрудника.
  + Priority\_id – уникальный идентификатор приоритета.

### Построение логической и физической модели данных.

Целью этого раздела является создание логической модели базы данных, которая будет служить основой для физической реализации базы данных.

Связь-наследование «Клиент, специалист ТХП являются сотрудником» было принято объединить в одну таблицу – Employee, по причине отсутствия специфических связей, а определение роли будет обеспечено за счет таблицы Role, которая также позволит быстро добавить в систему еще одну роль, например, Администратор, путем добавления новой записи в таблицу.

Связи между сущностями заявка и заявка на ПО, заявка и заявка на технику было принято разделить на 3 таблицы и обеспечить связь с помощью внешнего ключа, который будем обеспечивать связь между общими атрибутами любой заявки, хранящимися в таблице Application, и специфическими для заявок атрибутами, хранящимися в таблице Application\_device и Application\_software. Такое решение обосновано тем, что заявка на ПО и заявка на технику имеют несколько специфичных атрибутов.

На основании сущностей, заполненных атрибутами, и принятых решений были построена логическая схема базы данных и физическая схема базы данных, показанные на рисунках 29 и 30.

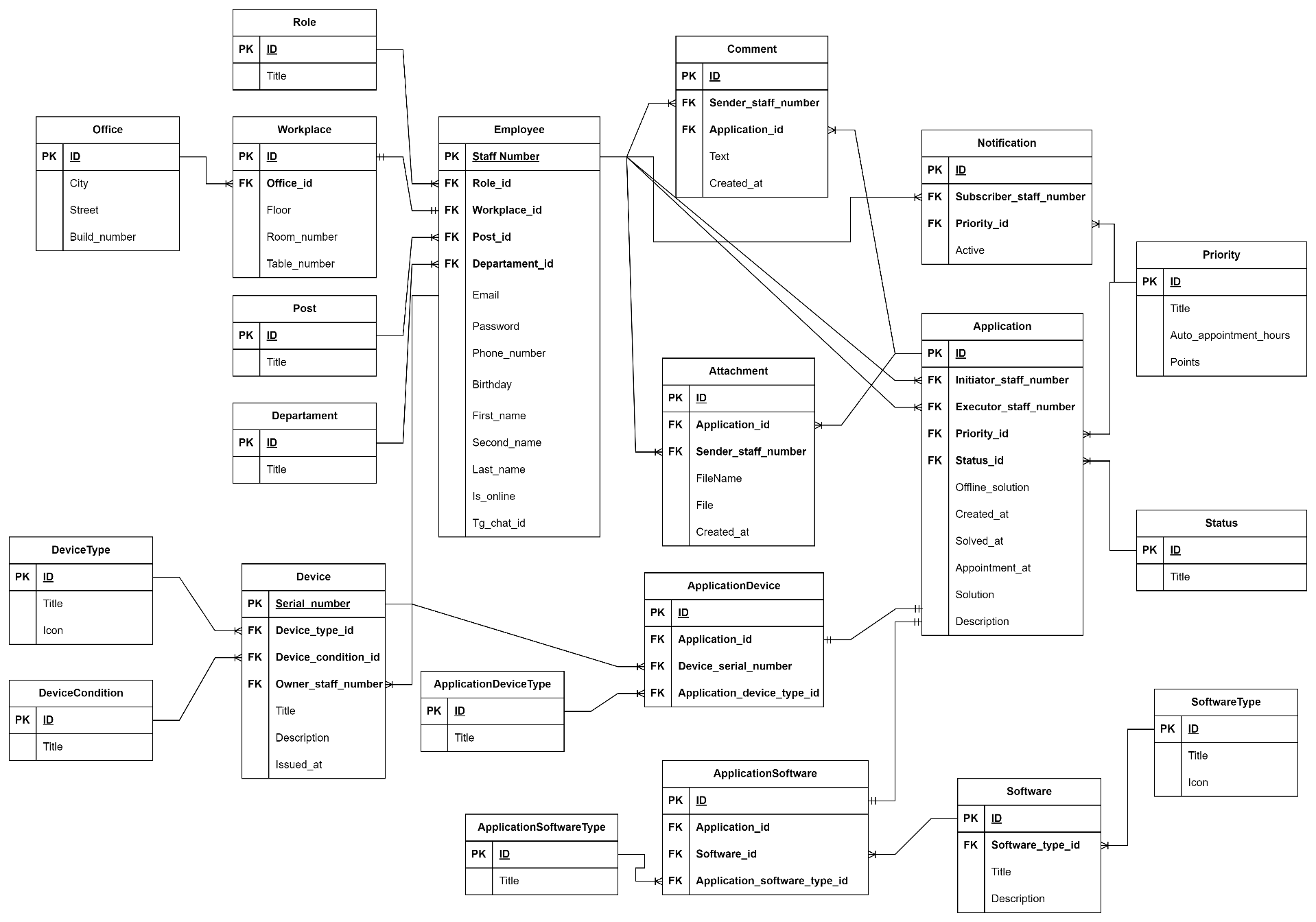


Рисунок 29 - Логическая модель данных программного обеспечения для автоматизации работы службы технической поддержки

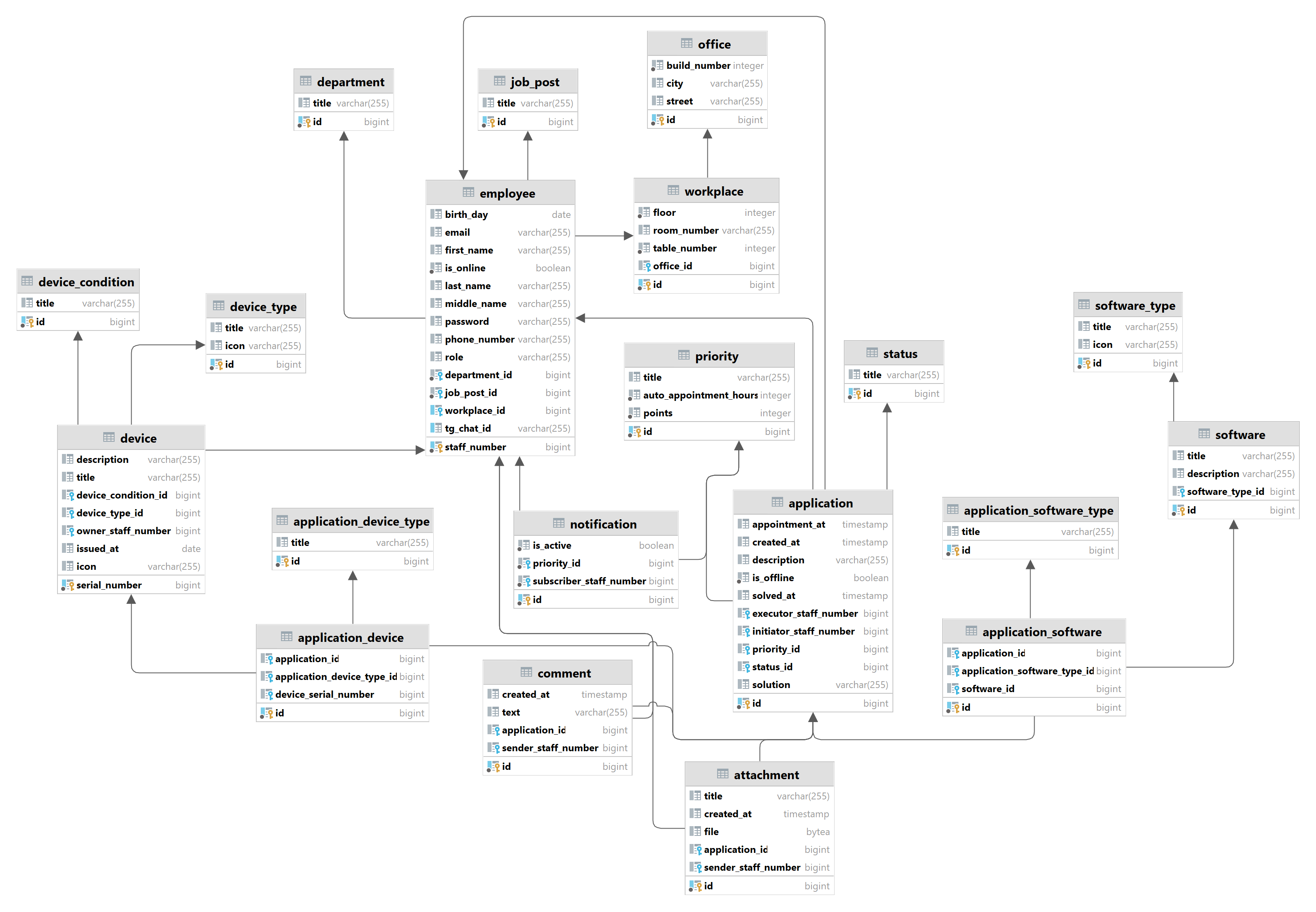


Рисунок 30 - Физическая модель данных программного обеспечения для автоматизации работы службы технической поддержки

## Разработка серверной части основного приложения.

Серверная часть приложения будет разработана на языке программирования Java с использованием фреймворка Spring, его различных модулей. Процесс разработки можно разделить на следующие этапы.

* Разработка базы данных и взаимодействия с ней с использованием Spring Data JPA и Hibernate.
* Разработка процесса авторизации и аутентификации с использованием Spring Security.
* Разработка основной бизнес-логики приложения, включающей в себя необходимые алгоритмы и операции.
* Разработка REST контроллеров с помощью возможностей, предоставляемых Spring Boot.

### Разработка базы данных.

Процесс разработки базы данных с использованием Hibernate и Spring Data JPA включает следующие этапы.

* Определение сущностей: на этом этапе определяются сущности, которые будут представлять таблицы в базе данных. Каждая сущность соответствует отдельной таблице и содержит атрибуты, которые будут столбцами в таблице. Аннотации Hibernate и Spring Data JPA используются для указания связей между сущностями, определения первичных и внешних ключей, а также других аспектов модели данных.
* Создание репозиториев: репозитории представляют интерфейсы или классы, которые определяют методы для выполнения операций с базой данных, таких как добавление, обновление, удаление и поиск данных. Spring Data JPA облегчает создание репозиториев, автоматически генерируя реализацию методов, осуществляющих запросы к базе данных, на основе их названий и типов возвращаемых значений.
* Конфигурация подключения к базе данных: необходимо настроить параметры подключения к базе данных, такие как URL, имя пользователя и пароль. Это обычно выполняется с использованием файлов конфигурации или аннотаций в классе конфигурации приложения. В нашем случае конфигурация будет определяться в файле application.properties следующим образом.

spring.datasource.url=jdbc:postgresql://localhost:5435/technicalSupportDb

spring.datasource.username=postgres

spring.datasource.password=qwe

Из сущностей спроектированной базы данных можно явно выделить общие атрибуты – id и title. Поэтому воспользуемся механизмом наследования и создадим две абстрактные сущности BaseEntity и NamedBasedEntity, которые будет использованы, как родительские для остальных сущностей.

//Аннотация, обозначающая, что это родительская сущность

@MappedSuperclass

@Getter

@EqualsAndHashCode

public abstract class BaseEntity {

// Обозначение столба с уникальным идентификатором, который

// будет являться первичным ключом, также задается

// стратегия генерация значения

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

}

@MappedSuperclass

@Getter

public abstract class NamedBasedEntity extends BaseEntity {

@Column(nullable = false, unique = true)

private String title;

}

И для примера рассмотрим создание двух связанных между собой сущностей Office и Workplace.

// Обозначает, что это сущность

@Entity

// Создание таблицы с одноименным классу названием

@Table

// Создание конструктора без аргументов

@NoArgsConstructor

// Создание конструктора со всеми аргументами (полями класса)

@AllArgsConstructor

// Реализация шаблона проектирования Builder

@Builder

@Getter

public class Office extends BaseEntity {

@Column(nullable = false)

private String city;

@Column(nullable = false)

private String street;

@Column(nullable = false)

private int buildNumber;

// Создания связи 1 ко многим с сущностью класса Workplace,

// mappedBy указывает на название поля, с которым будет связь

@OneToMany(mappedBy = "office")

private Set<Workplace> workplaces;

}

@Entity

// Создание ограничения уникальности для совокупности столбцов

// так как одно и тоже рабочее место

// не может быть в одном офисе на одном этаже и одной комнате

@Table(

uniqueConstraints=

@UniqueConstraint(columnNames={"office\_id", "floor", "room\_number", "table\_number"})

)

@NoArgsConstructor

@AllArgsConstructor

@Builder

@Getter

public class Workplace extends BaseEntity {

@Column(nullable = false)

private int floor;

@Column(nullable = false)

private String roomNumber;

@Column(nullable = false)

private int tableNumber;

// Связь с сущностью Office, которая создаст внешний ключ

// добавив столбец office\_id в таблицу Workplace

@ManyToOne

@JoinColumn(name = "office\_id", referencedColumnName = "id")

private Office office;

@OneToOne(mappedBy = "workplace")

private Employee employee;

}

Все сущности будут находиться в пакете с названием domain, структура которого показана на рисунке 31.

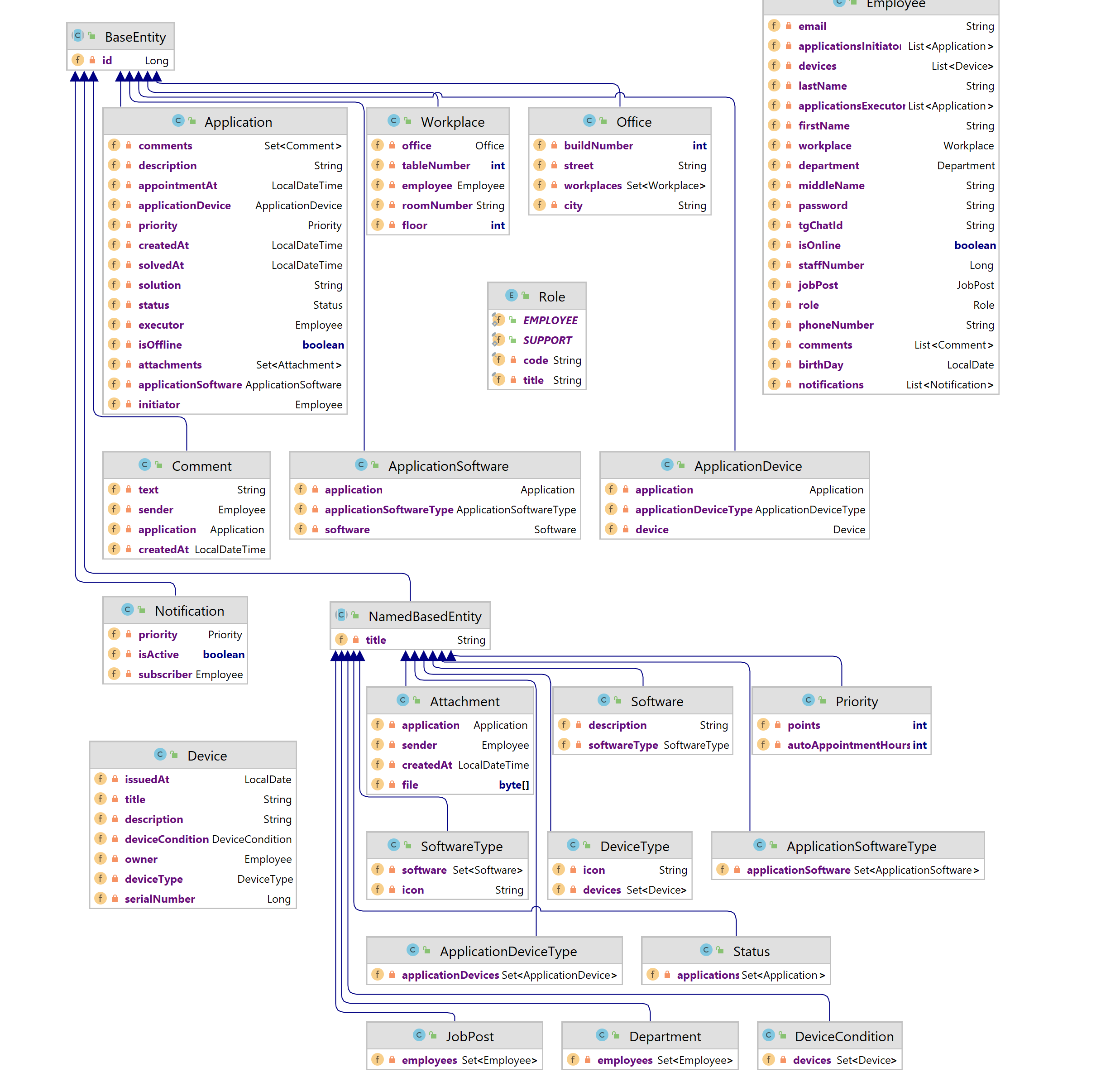


Рисунок - Диаграмма классов созданных сущностей

Теперь рассмотрим создание репозиториев с использованием Spring Data JPA на примере.

// Определение интерфейса и наследование от интерфейса

// JpaRepository <T, ID>, где T - класс-сущность,

// с которым будет работать репозиторий,

// ID - тип уникального идентификатора (первичного ключа) в

// классе-сущности T

public interface ApplicationRepository extends JpaRepository<Application, Long> {

// Пример создания нативного запроса с параметром

// аналог классическому PreparedStatement

@Query(

value = "SELECT \* FROM application " +

"WHERE executor\_staff\_number = ?1",

nativeQuery = true

)

List<Application> findAllByExecutorStaffNumber(Long staffNumber);

// Пример генерации запроса с помощью репозитория на основании

// названия, параметров и типа возвращаемого значения метода

List<Application> findAllByPriority(String priorityTitle);

}

### Разработка алгоритма создания заявки.



Рисунок - схема алгоритма создания заявки



Рисунок - Блок-схема функции назначения исполнителя для заявки



Рисунок - Блок-схема алгоритма функции выбора непустого списка



Рисунок - Блок-схема алгоритма функции отправки уведомления исполнителю

### Разработка алгоритма обработки заявки.

Разрабатываемое программное обеспечение для автоматизации работы специалистов технической поддержки предусматривает обработку двух категорий заявок: заявки на технику и заявки на ПО.

Каждая заявка в процессе своего жизненного цикла имеет один из следующих статусов.

* После создания и до момента, пока специалист не взял ее в работу – «Создана».
* После того, как специалист ТХП взял заявку в работу принимает статус «В работе».
* При принятии решения о закрытии заявки специалист ТХП выбирает либо – кратко описывает решение проблемы и выбирает способ решения заявки, либо отменяет заявку. В первом случае заявка принимает статус «Решена», во втором - «Отменена».

Общий алгоритм, который необходимо реализовать в системе для обработки заявок, изображен на рисунке 36.



Рисунок - Блок-схема алгоритма обработки заявки специалистом ТХП

## Разработка клиентской части.

Разработка клиентской части включает в себя создание пользовательского веб-интерфейса, получения данных с сервера и корректного их отображения на формах. В рамках разработки программного обеспечения для автоматизации работы специалистов технической поддержки важными являются следующие составляющие.

* Адаптивный дизайн: дает возможность пользователю использовать приложение с наиболее популярных устройств – смартфона, компьютера и планшета, для тестирования воспользуемся инструментами разработчика в браузере, предоставляющими возможность проверки интерфейса на различных устройствах. Для тестирования будем использовать следующие модели.
  + Смартфон с наименьшим разрешением доступный в инструментах разработчика браузера – iPhone SE.
  + Планшетное устройство – Surface Pro 7 с разрешением 912 на 1368 пикселей, что-то среднее между десктопной версией и версией для смартфона.
  + Компьютерную версию с разрешением 1920 на 1080 пикселей.
* Получение данных с сервера: клиентское приложение для браузера взаимодействует только с серверным приложением с основной бизнес- логикой, соответственно, для получения данных необходимо осуществить REST-запрос на сервер по определенному в контроллерах адресу, а также предоставить, как минимум следующие 2 необходимых заголовка для всех запросов (кроме авторизации).
  + Content-type: application/json – позволяет понять серверу, что в теле сообщения содержится информация в формате JSON.
  + Authorization: значение JWT токена – этот заголовок необходим для того, чтобы сервер осуществил авторизацию запроса.

### Разработка адаптивного пользовательского интерфейса.

Разработаем основные формы с помощью, которых пользователи будут взаимодействовать с системой.

На рисунке

# ТЕСТИРОВАНИЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

# ПРОГРАММНАЯ ДОКУМЕНТАЦИЯ

## Применение программы

Разработанное программное обеспечение предназначено для автоматизации работы специалистов технической поддержки, посредством автоматического назначения заявки, созданной клиентом, на специалиста ТХП и возможностью отправки уведомлений в мессенджер Telegram.

Помимо этого, программное обеспечение реализует следующий функционал для клиента.

* Просмотр закрепленной за ним техники.
* Создание заявки.
* Отслеживание статуса выполнения заявки.
* Просмотр всех своих заявок.
* Фильтрация и сортировка заявок.
* Связь с исполнителем заявки.

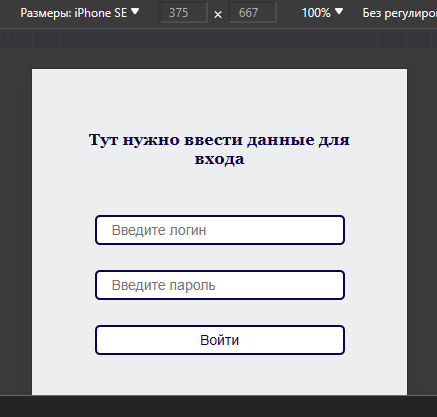
Для специалиста ТХП программное обеспечение предоставляет следующий набор функций.

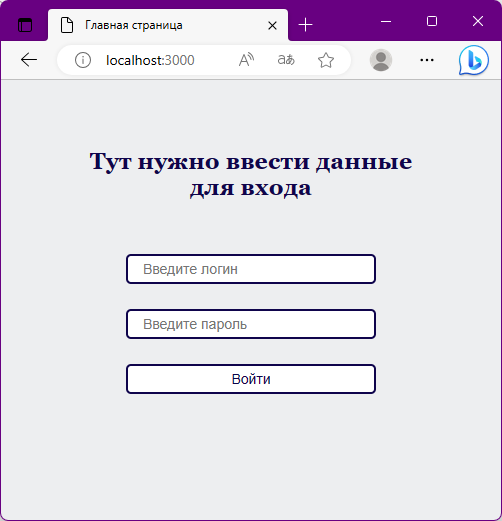
* Обработка заявки.
* Настройка получаемых уведомлений о заявках.
* Получение уведомлений о заявках в Telegram.
* Просмотр всех своих заявок.
* Фильтрация и сортировка заявок.
* Связь с клиентом заявки.

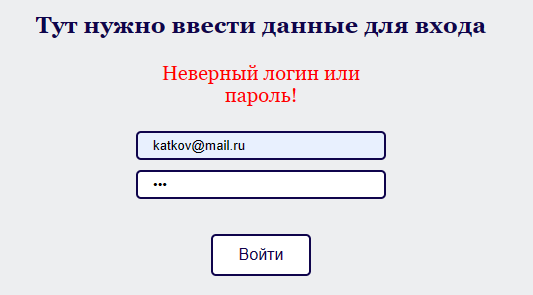
Разработанное программное обеспечение имеет адаптивный пользовательский интерфейс и одинаково работает на мобильных устройствах, планшетах и компьютерах.

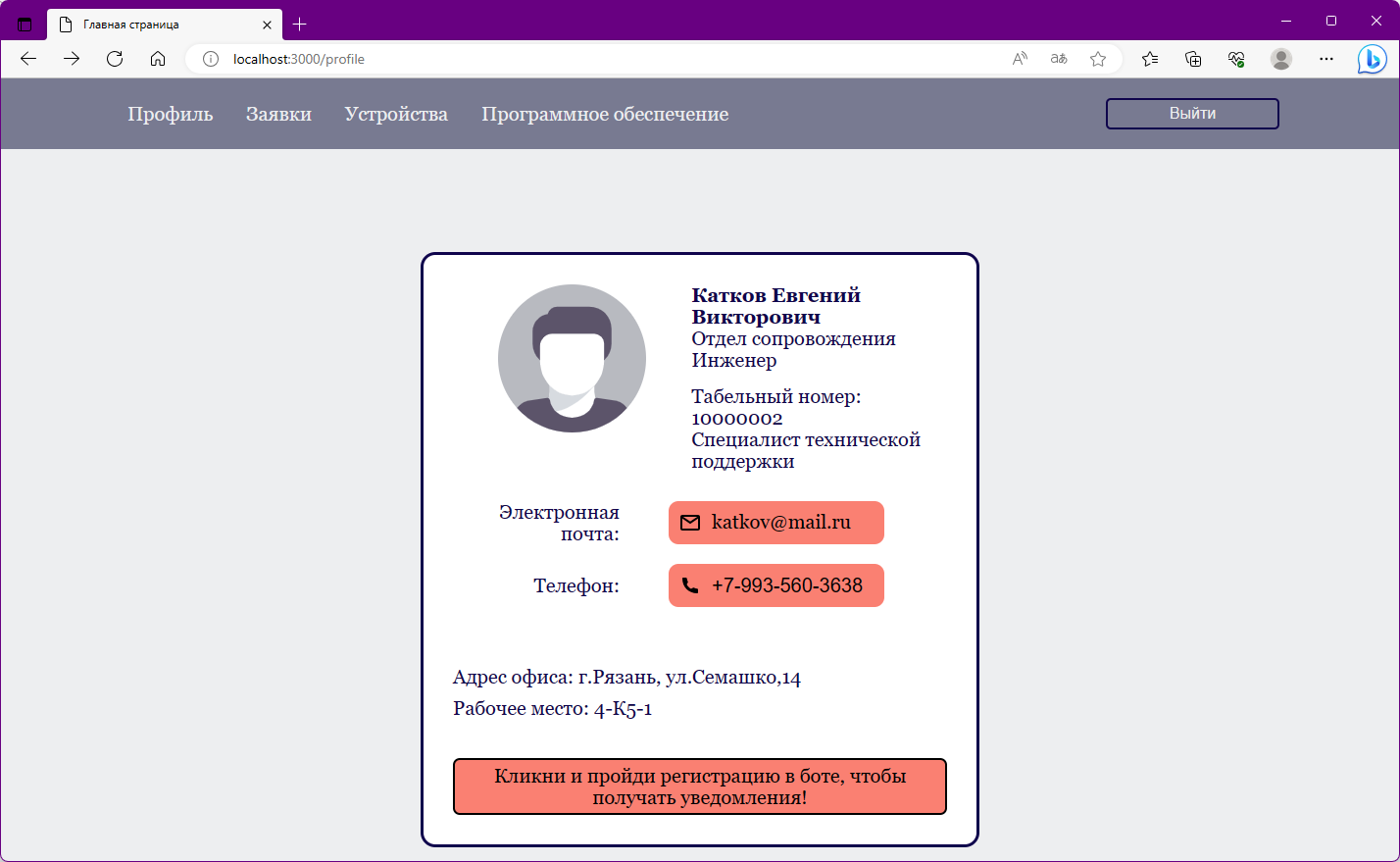
Минимальные системные требования для установки и запуска программного обеспечения:

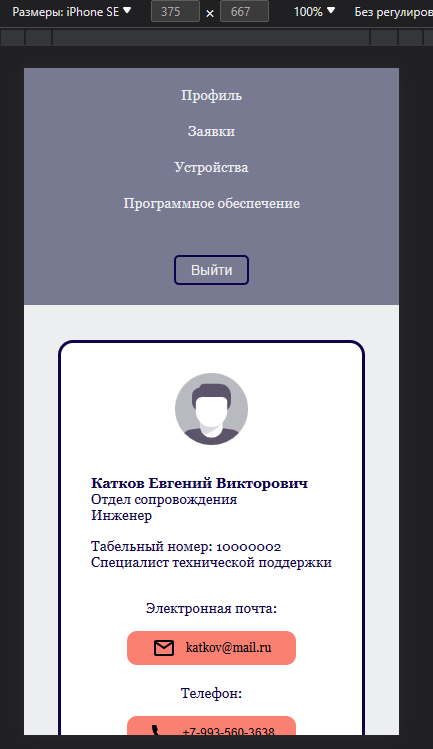
* ОС: Windows 10 (64-битная) или выше, Ubuntu 18.04 LTS или выше.
* Процессор: минимум Intel Core i3 2.4 ГГц или аналогичный процессор.
* Оперативная память: 4 Гб и выше.
* 10 ГБ свободного места и больше, зависит от объема данных.
* Браузер: Google Chrome версии 80 или выше, Mozilla Firefox версии 75 или выше, Safari версии 13 или выше, Microsoft Edge версии 80 или выше, или любой другой аналогичный браузер.
* Дополнительное программное обеспечение: Docker версии 20 или выше.

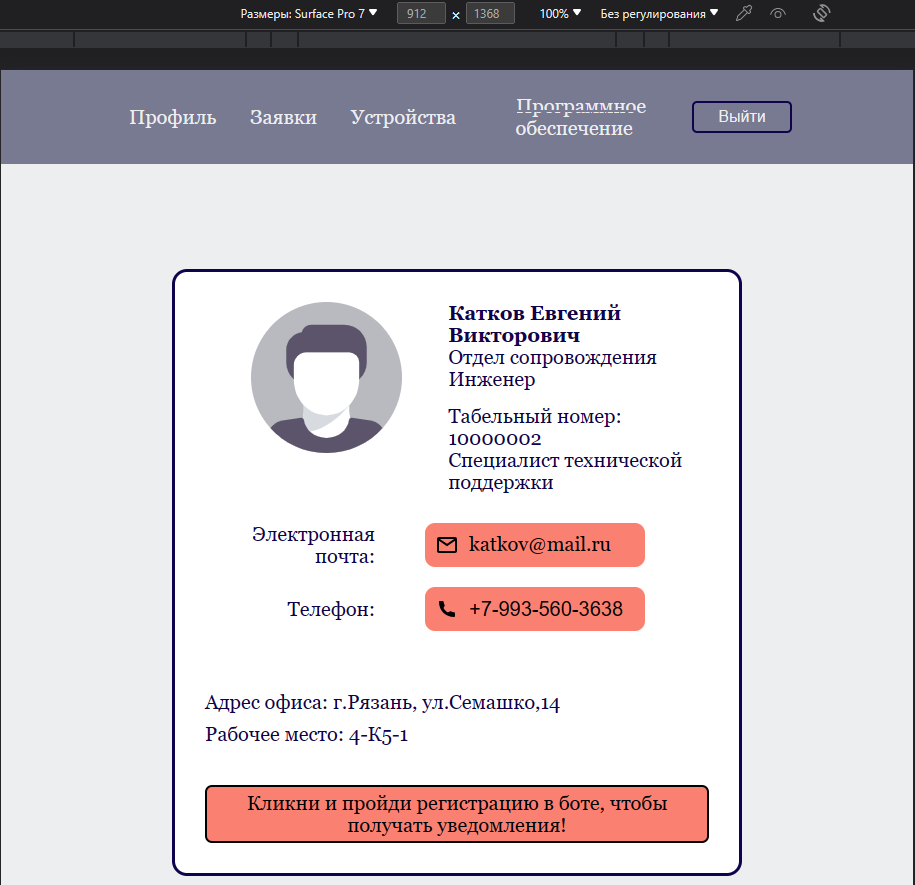
****

****

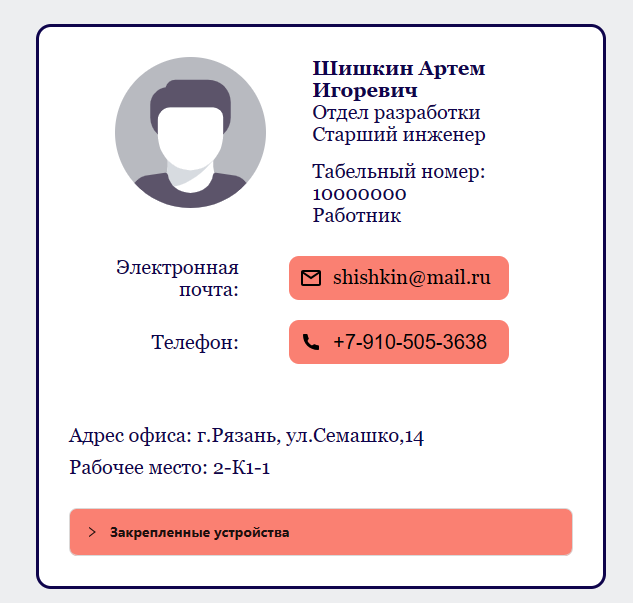
****

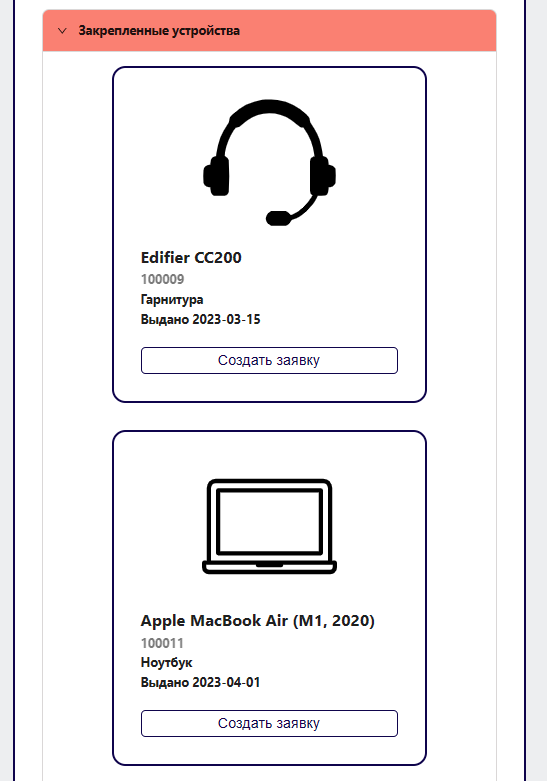
****

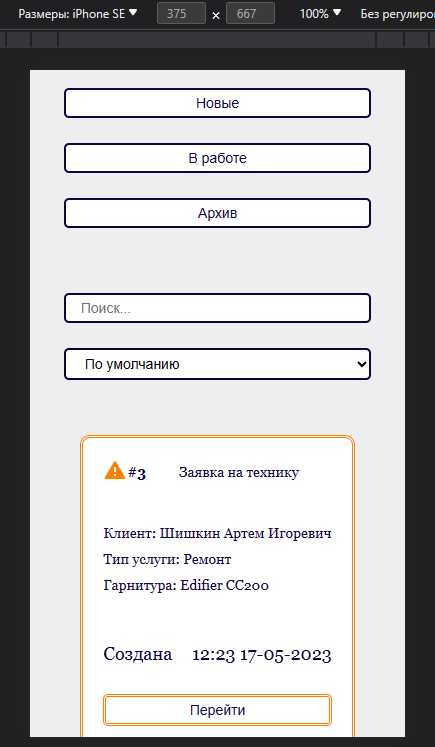
****

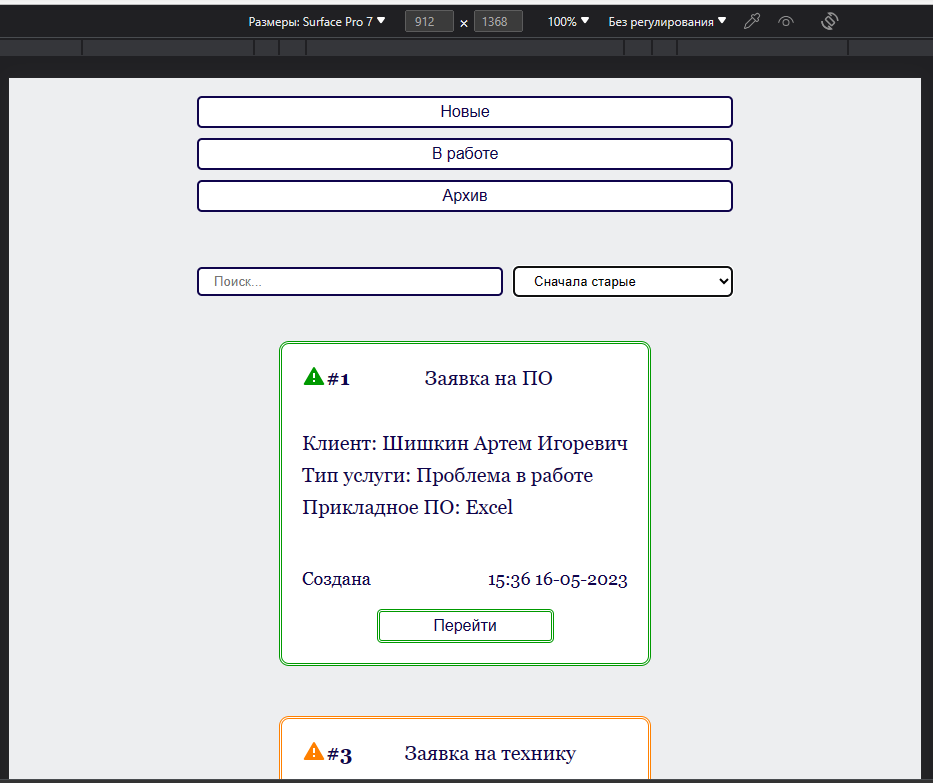
****

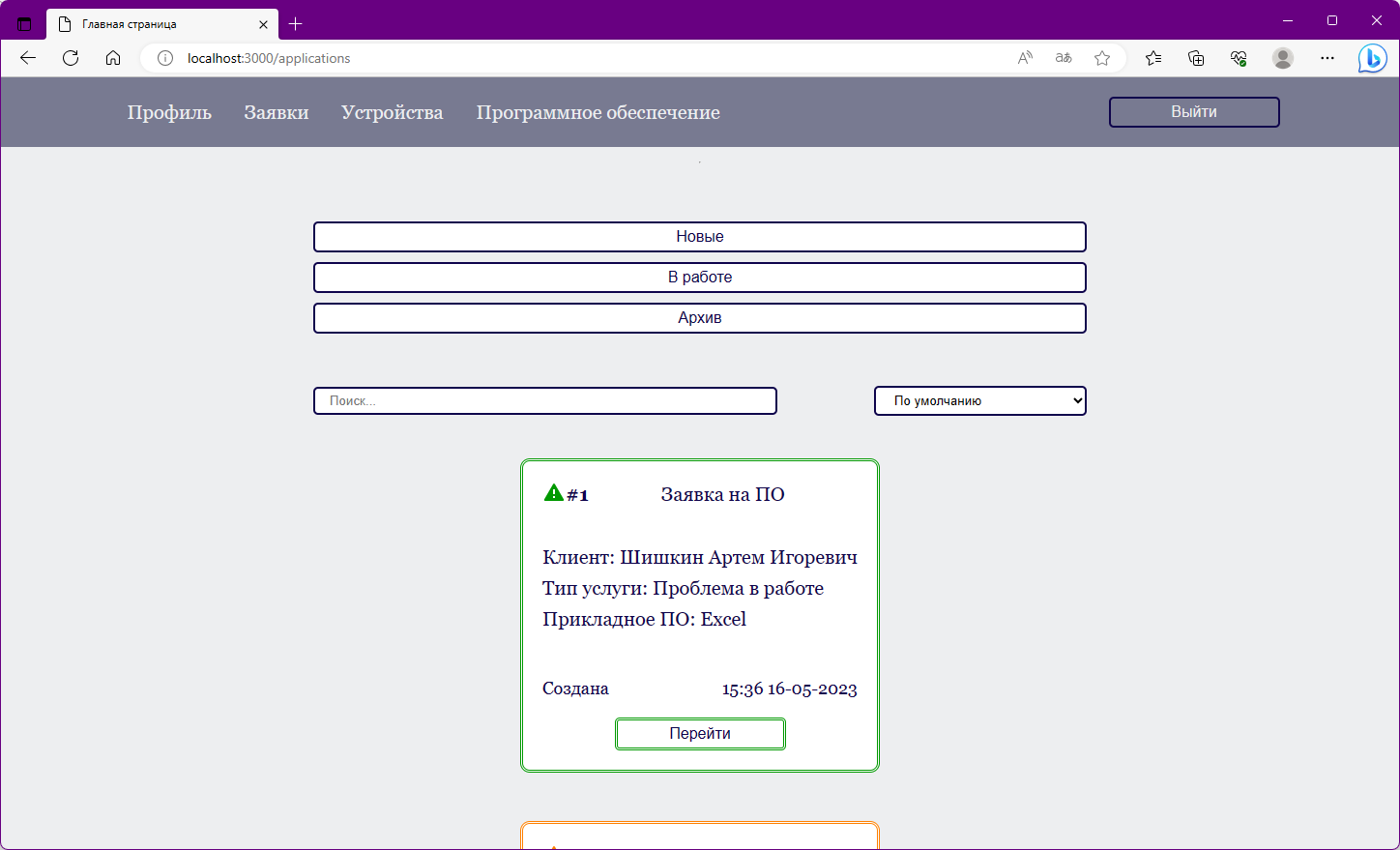
****

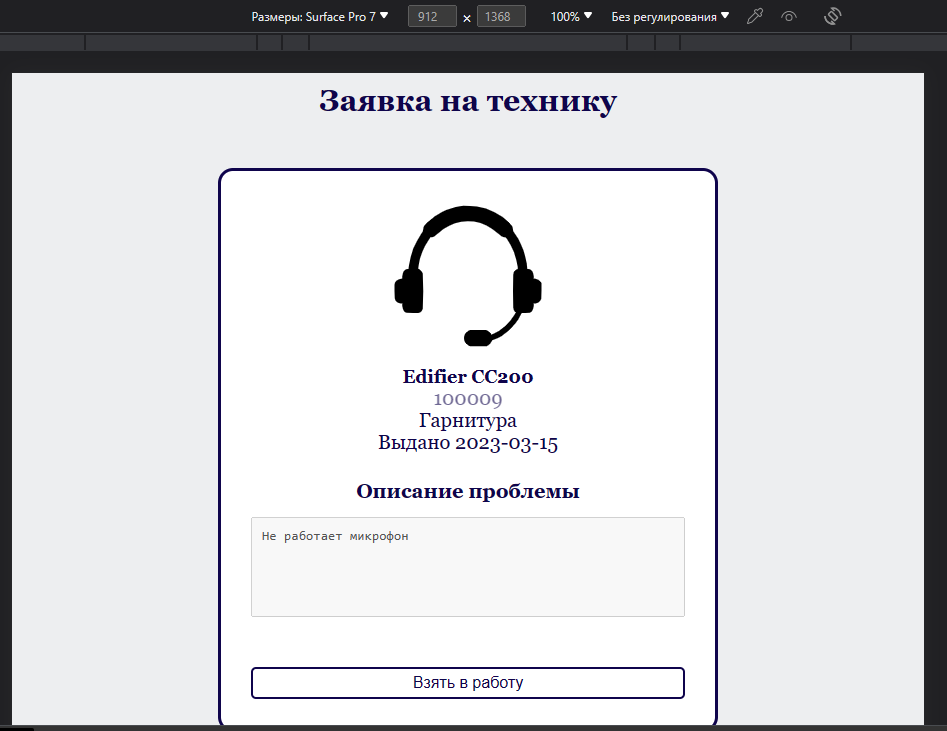
****

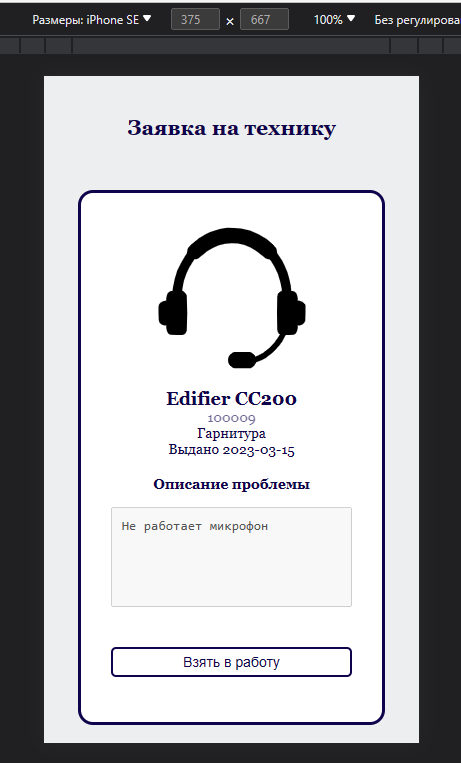
****

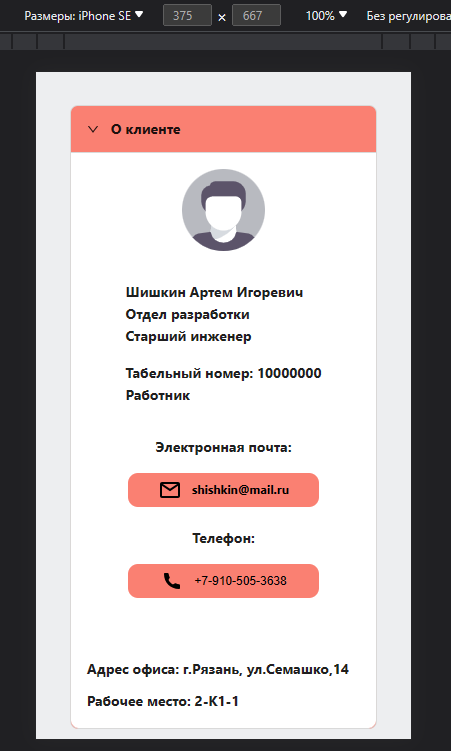
****

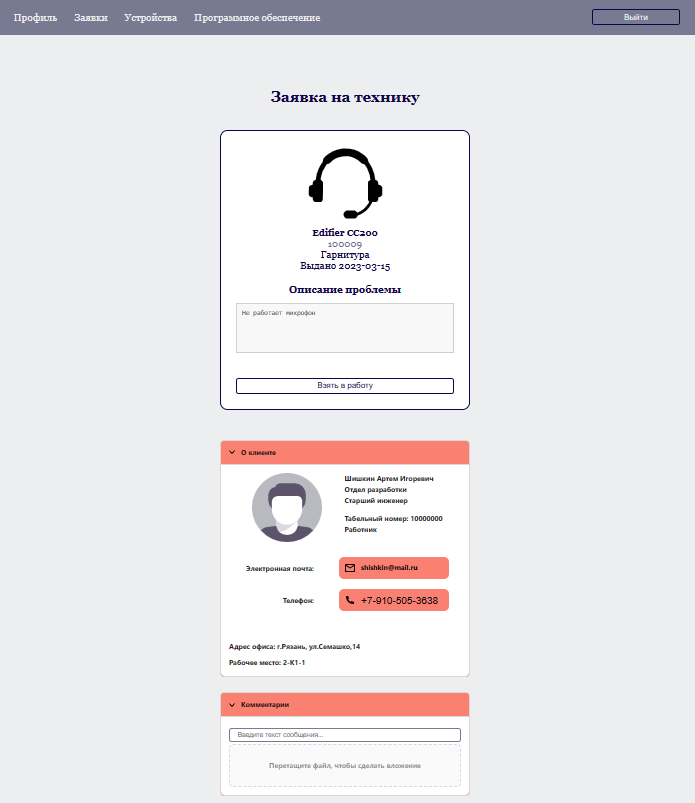
****

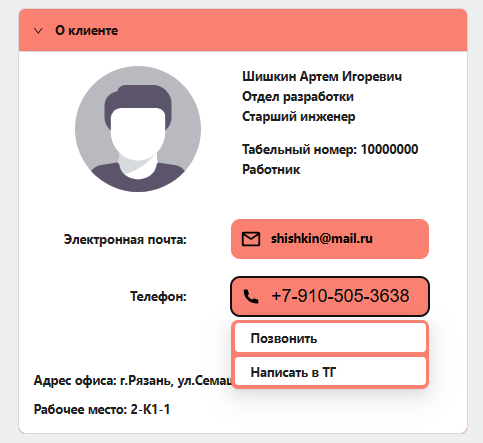
****

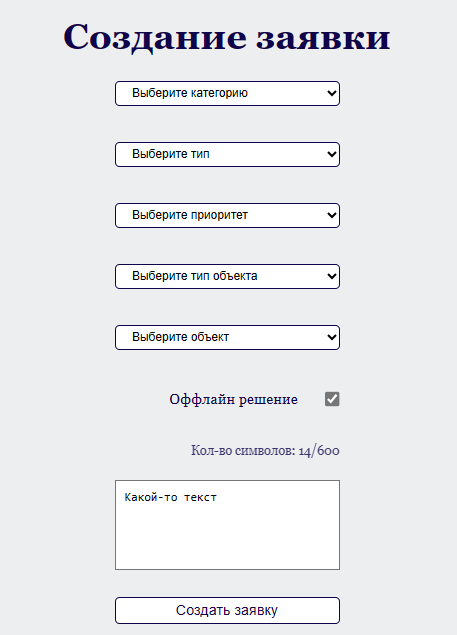
****

****

****

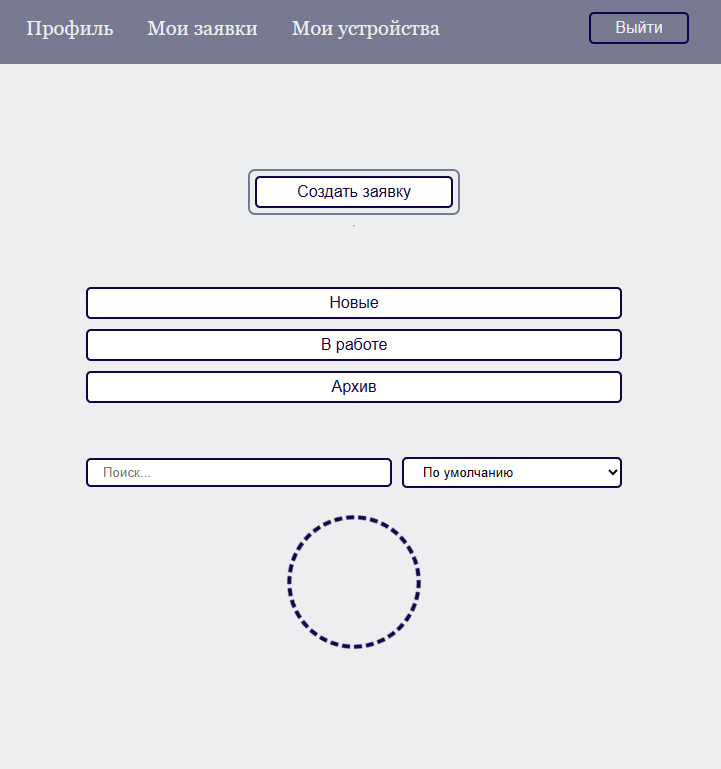
****

****

****

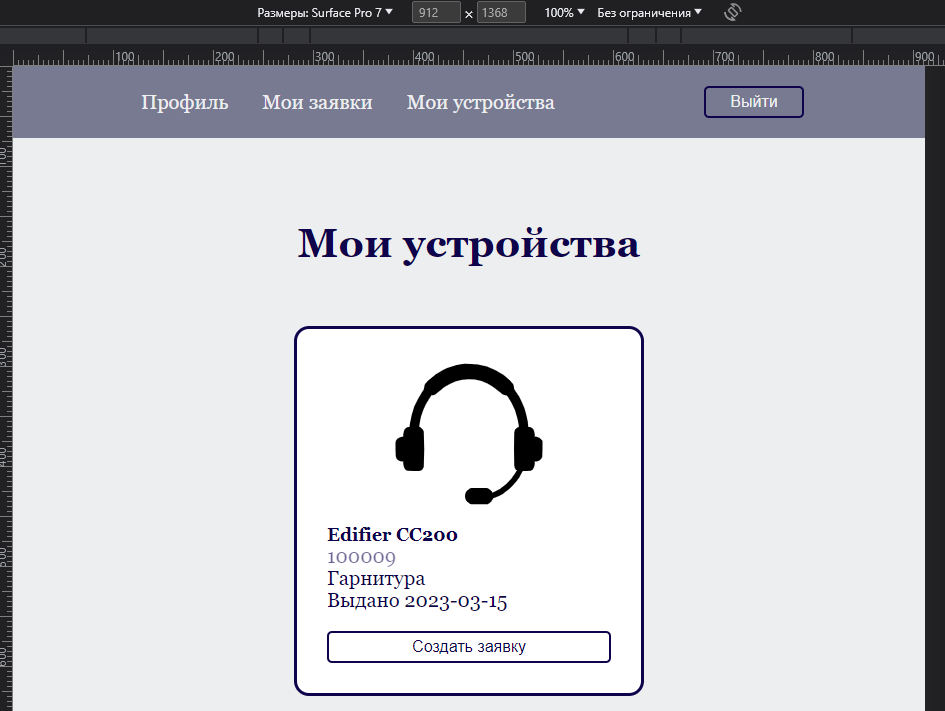
****

****

****

****

****

****

ЗАКЛЮЧЕНИЕ

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Черткова, Е. А. Программная инженерия. Визуальное моделирование программных систем. Учебник для среднего профессионального образования. Издательство Юрайт, 2023. – 147 с.
2. Роберт С. Мартин Чистая архитектура. Искусство разработки программного обеспечения. Издательство Питер, 2018. – 352 с.
3. Шитов, В. Н. Основы проектирования баз данных. Учебное пособие. Издательство ИНФРА-М, 2023. — 236 с.

ПРИЛОЖЕНИЕ