МИНИСТЕРСТВО ОБРАЗОВАНИЯ И МОЛОДЁЖНОЙ ПОЛИТИКИ СВЕРДЛОВСКОЙ ОБЛАСТИ

ГАПОУ СО ЕКАТЕРИНБУРГСКИЙ КОЛЛЕДЖ ТРАНСПОРТНОГО СТРОИТЕЛЬСТВА

Специальность 09.02.07:

«Информационные системы и программирование»

Система для работы с заявками о неисправностях оборудования колледжа

Дипломный проект

ДП-ПР-41-01-2023

2023

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И МОЛОДЁЖНОЙ ПОЛИТИКИ СВЕРДЛОВСКОЙ ОБЛАСТИ

ГАПОУ СО ЕКАТЕРИНБУРГСКИЙ КОЛЛЕДЖ ТРАНСПОРТНОГО СТРОИТЕЛЬСТВА

Специальность 09.02.07:

«Информационные системы и программирование»

Заведующий отделением _____ *Е.В.Дудель*

Председатель цикловой комиссии _____ *Г.В. Мирошниченко*

Система для работы с заявками о неисправностях оборудования колледжа

Пояснительная записка

к дипломному проекту

ДП-ПР-41-01-2023-ПЗ

Разработал:

Студент гр. ПР-41 _____/*М.А. Варганов*

Руководитель: _____/*В.А. Михайлов*

Консультант: _____/*В.А. Михайлов*

Н. контроль: _____/*В.А. Михайлов*

Рецензент: _____/*Грибова А. В.*

2023

Содержание

Введение

Развивающийся технологический мир оказывает влияние на многие сферы нашей жизни, включая сферу образования. Любой преподаватель старается идти в ногу со временем и внедрять информационные технологии в обучающий процесс для более глубокого усвоения информации при изучении предметов, также облегчает работу педагогов в ходе проведения занятий. Все это обуславливает использование множества программ, которые облегчают поиск, анализ или создание информации.

Переход образовательных учреждений на цифровые экосистемы означает не только повышение качества обучения, но и рост количества компьютеров, техники и программ для передачи информации, паролей, учетных записей – и всем этим необходимо управлять. ИТ-отделы в учреждениях среднего образования работают в условиях ограниченного бюджетирования, имея на вооружении устаревшие инструменты и ограниченный штат сотрудников. У них просто нет необходимых ресурсов для обеспечения соответствия требованиям отделов образования. Эти технологические сложности препятствуют достижению основной цели колледжа - предоставлению качественного образования.

В сегодняшних реалиях, когда повышаются требования к качеству подготовки выпускников, происходит глобальная информатизация, развитие новейших информационных технологий, что также будет улучшать подготовку будущего специалиста.

Глобальная информатизация учебного процесса привела к появлению новых проблем, связанных с необходимостью обеспечить техническую поддержку пользователей и техники. Вот почему в настоящее время стала весьма актуальной задача разработки информационной системы для службы технической поддержки пользователей, такие системы получили название Service Desk. Информационные системы Service Desk с использованием клиент-серверной технологии стали весьма популярными для реализации в условиях

образовательных учреждений. Данные программы помогают систематизировать и оптимизировать работу технического персонала, ускоряя процесс передачи запросов о проблемах с программным обеспечением или оборудованием, также предоставляя возможности для эффективного управления запросами и заявками. Подобные программы отслеживают и анализируют проблемы эксплуатации, ведут аналитику. Система должна способствовать быстрому решению проблем путем фиксации и отслеживания их статуса в базе данных, контролировать уровни технического обслуживания, отслеживать состояние технических средств, сигнализировать о необходимости выполнения плановых работ.

Безусловно, внедрение информационных систем в работу учебного заведения содержит риски, которые связанны с эффективностью системы, ее гибкостью и техническими аспектами, связанными с оборудованием, линиями связи, программным обеспечением. Особо следует отметить риски, связанные с человеческим фактором. Отсутствие инструментов управления информационными технологиями, уровнем технического сопровождения информационных систем, обслуживание компьютерной техники и линий связи может привести к сбоям или отключениям системы, что, в свою очередь, может привести к уменьшению производительности системы.

Объект исследования - информационная система с использованием клиент-серверной технологии.

Предмет исследования - система для работы с заявками о неисправностях оборудования колледжа.

Цель - разработка системы для работы с заявками о неисправностях оборудования образовательного учреждения ГАПОУ СО «ЕКТС».

Задачи исследования:

- проанализировать возможности автоматизации службы технической поддержки учреждения;
- изучить работу службы технической поддержки в учреждении;

- построить модели автоматизированной системы службы технической поддержки;
- разработать структуру ServiceDesk системы, определить ее основные функции;
- выполнить проектирование автоматизированной службы технической поддержки;
- разработать модули автоматизированной службы технической поддержки предприятия.

1. Описание предметной области

Предметной областью дипломного проекта является система для работы с заявками о неисправностях оборудования в образовательном учреждении ГАПОУ СО «ЕКТС».

Система для работы с заявками о неисправностях оборудования (Service Desk) колледжа предоставляет инструменты для управления и отслеживания обращений сотрудников колледжа по поводу проблем с оборудованием или прикладными программами, которые могут возникнуть в различных компьютерных кабинетах, лабораториях, аудиториях и других помещениях на различных устройствах.

Service Desk - это система автоматизации работы техподдержки с клиентскими обращениям, помогающая ускорить процесс доставки информации о неисправностях от пользователя до технического персонала, отслеживать текущие задачи, а также проводить анализ работы ИТ отдела.

Система позволяет сотрудникам отправлять заявки через веб-интерфейс. Заявка содержит описание проблемы, на каких рабочих устройствах она возникла, место, а также контактную информацию заявителя.

Далее заявка направляется на рассмотрение в техническую поддержку колледжа, которая должна решить проблему в ближайшие сроки. Система также позволяет отслеживать статус заявки и работы, проведенные по ней.

Специалисты в свою очередь, документируют работы, проведенные по заявке в этой же системе, и если проблема устранена, то они её завершают.

Основные функции системы:

- подача заявки через веб-интерфейс с указанием места, описанием проблемы, контактными данными заявителя;
- прием заявок от сотрудников колледжа через веб-интерфейс;
- назначение ответственного специалиста для решения проблемы;

- отслеживание статуса заявки и информирование заявителя об изменении ее статуса;
- добавление проведенных работ по заявке сотрудниками технической поддержки;
- завершение заявки заявителем досрочно, либо технической поддержкой при проведении работ по ней, и устранением проблемы;
- анализ эффективности работы технической поддержки.

Объекты предметной области:

- заявитель - формирует заявку с указанием места, описанием проблемы, на каких устройствах она встретилась, а также свою контактную информацию;
- заявка - содержит информацию о месте, времени и описании проблемы, а также контактную информацию заявителя;
- оборудование - содержит информацию о типе, модели, серийном номере и местонахождении оборудования, и работ, которые когда-либо проводились с этой техникой;
- специалист - ответственный за выполнение заявки и работ, проводимых по ней.

Таким образом, система для работы с заявками о неисправностях оборудования колледжа позволяет эффективно управлять техническим обслуживанием оборудования, улучшать взаимодействие между сотрудниками и тех. поддержкой колледжа и также повышать качество работы технической поддержки.

2. Назначение и область применения программы

Система для работы с заявками о неисправностях оборудования колледжа предназначена для автоматизации процесса приема, регистрации, отслеживания и устранения неисправностей оборудования в колледже. Она позволяет оперативно реагировать на возникающие проблемы с оборудованием и ускоряет процесс их устранения.

Основная область применения такой системы - это поддержка функционирования технических служб колледжа. Система позволяет контролировать поток заявок, следить за их статусами, а также улучшать качество обслуживания пользователей.

Объектом исследования является информационная система с использованием клиент-серверной технологии, в которой функциональные возможности распределены между сервером и клиентскими устройствами. Сервер выполняет роль централизованного хранилища данных и обрабатывает запросы от клиентских устройств, а клиенты используют данные и функции сервера, работая через интерфейс, который предоставляет приложение на клиентском устройстве.

Клиент-серверная технология может быть использована для создания различных информационных систем, например:

- системы управления базами данных;
- корпоративные информационные системы;
- системы электронной коммерции;
- системы управления ресурсами предприятия;
- системы управления проектами и т.д.

Преимущества информационных систем, построенных на клиент-серверной технологии, включают:

- высокую скорость обработки данных и быстрый доступ к информации;

- гибкость и масштабируемость, т.к. сервер и клиентские устройства могут быть легко добавлены или удалены из системы;
- легкое обновление программного обеспечения, т.к. изменения могут быть внесены только на серверной стороне;
- повышенную безопасность, т.к. данные хранятся на сервере и могут быть защищены с помощью различных мер безопасности.

Однако, клиент-серверные системы также имеют некоторые недостатки, например:

- высокие затраты на аппаратное и программное обеспечение, т.к. требуется поддерживать серверную инфраструктуру;
- сложность установки и настройки системы;
- возможность возникновения проблем с соединением между клиентом и сервером, что может привести к нестабильности работы всей системы.

Предметом исследования является система для работы с заявками о неисправностях оборудования колледжа. Эта система разрабатывается для упрощения процесса регистрации и обработки заявок на ремонт или замену оборудования в колледже.

Основная цель системы состоит в том, чтобы обеспечить быстрое и эффективное решение проблем с оборудованием колледжа, что позволит существенно повысить качество обучения студентов и работу преподавателей.

Система будет основана на следующих принципах:

- простота использования. Система должна быть интуитивно понятной и легко доступной для всех пользователей колледжа;
- быстрота и эффективность. Система должна обеспечивать быстрое решение проблем с оборудованием, минимизируя время простоя и максимизируя продуктивность;

- гибкость и адаптивность. Система должна быть гибкой и адаптивной к меняющимся потребностям колледжа, позволяя быстро вносить изменения в процессы работы и функциональные возможности;
- безопасность и конфиденциальность. Система должна обеспечивать высокий уровень безопасности и конфиденциальности данных, предотвращая несанкционированный доступ к информации о заявках.

В ходе исследования будет произведен анализ существующих систем для работы с заявками на ремонт и замену оборудования в образовательных учреждениях, а также проведен опрос технических специалистов колледжа для выявления основных требований и потребностей пользователей.

На основе этого анализа и опроса разработана концепция системы, определены ее функциональные возможности и создан план ее внедрения в колледж.

3. Проектирование задачи

После анализа предмета исследования были выделены следующие требования к системе, которая позволяет:

- сотрудникам колледжа подавать заявки о неисправностях оборудования или прикладных программ;
- заявителю отслеживать состояние заявки и работ, проведенных по ней;
- сотрудникам технической поддержки принимать заявки, добавлять работы, проведенные по ней и завершать её в случае устранения проблемы;
- администратору просматривать текущие задачи каждого сотрудника технической поддержки;
- настройка доступа как к элементам, так и к страницам на веб-сайте в соответствии с ролью пользователя.

Далее после выявления основных требований к системе можно приступать к разработке программы.

В качестве методики разработки программы был выбран поход Database First - это один из подходов к разработке приложений, при котором сначала создается база данных, а затем на ее основе автоматически генерируется код и модели для доступа к данным. Такой подход обычно реализуется с помощью ORM (Object-Relational Mapping) систем, таких как Entity Framework в .NET.

Основные преимущества подхода Database First:

- быстрое создание приложения: благодаря автоматической генерации кода и моделей доступа к данным по мере создания базы данных приложение можно быстро развернуть и начать работу с ним;
- автоматическая синхронизация: изменения в базе данных автоматически отражаются в моделях данных и коде доступа к данным, что позволяет избежать ошибок синхронизации;

- простота использования: поскольку база данных уже существует, она может быть проектирована с учетом требований приложения, что облегчает ее использование;
- читаемость кода: сгенерированный код обычно легче читать, поскольку он соответствует схеме базы данных, что может помочь ускорить разработку;
- поддержка транзакций: код, сгенерированный при помощи подхода Database First, обычно полностью поддерживает транзакции базы данных.

В целом, подход Database First может значительно ускорить процесс разработки приложения, особенно если требования к базе данных уже известны. Однако он может иметь и некоторые ограничения, например, он не идеален для проектов с большим количеством запросов, а также может создавать избыточный код.

После того как мы определились с методологией разработки системы, нужно понять, какой из типов баз данных нам подходит. Лучше всего для разработки данной системы подошла реляционная база данных.

Реляционная база данных (РБД) - это тип базы данных, основанный на модели реляционной алгебры. В РБД данные хранятся в таблицах с определенным набором колонок и строк. Каждая таблица представляет отдельный тип объектов или сущностей, а каждая строка в таблице соответствует одному экземпляру этой сущности, а каждая колонка представляет отдельное свойство или атрибут этой сущности.

РБД имеют ряд преимуществ, таких как стандартизация языка запросов (SQL), простота использования, возможность работы с большими объемами данных и масштабируемость. Они также обеспечивают целостность данных, т.к. каждый элемент данных может быть связан с другими элементами в базе данных через уникальный идентификатор.

Далее можно приступать к проектированию БД. Среди всех решений больше всего выделился Microsoft SQL Server.

В качестве пользовательского интерфейса выбран Angular, и так он не имеет возможности прямого подключения к базе данных нужен протокол обмена информацией, для решения данной задачи выбран REST API созданный в фреймворке ASP.Net Web API.

3.1 Обоснование инструментов разработки

Microsoft SQL Server – это система управления реляционными базами данных, разработанная корпорацией Microsoft. Основной используемый язык запросов - Transact-SQL.

Microsoft SQL Server имеет несколько преимуществ по сравнению с другими реляционными базами данных:

- высокая производительность: SQL Server был оптимизирован для работы с многопроцессорными системами и высокой нагрузкой;
- поддержка транзакций: SQL Server обеспечивает полную поддержку транзакций, что позволяет сохранять целостность данных даже при нештатных ситуациях;
- безопасность: SQL Server достаточно надежен для использования в критических системах, так как предоставляет широкий набор инструментов для защиты данных;
- широкий функционал: SQL Server поддерживает большое число стандартных функций и операторов, а также имеет расширенные возможности для работы с данными, например, для полнотекстового поиска;
- простота установки и настройки: SQL Server имеет понятный интерфейс установки и настройки, что упрощает его использование даже для относительно непрофессиональных пользователей.

Кроме того, Microsoft SQL Server является одним из самых популярных и широко используемых решений для хранения и обработки данных, что обеспечивает большую поддержку со стороны сообщества пользователей и разработчиков.

Предметами исследования являются технологии и программные средства, используемые для создания и поддержки системы. Система основана на индивидуально разработанном приложении на платформе Angular, созданная компанией Google, которая позволяет разработчикам использовать HTML, CSS и TypeScript.

Angular - это один из самых популярных фреймворков для разработки веб-приложений. Его преимущества перед аналогами, такими как React и Vue.js, заключаются в следующем:

- полноценный фреймворк - Angular предоставляет все необходимые инструменты для полноценной разработки веб-приложений, включая маршрутизацию, HTTP-запросы, состояния, формы и многое другое;
- удобный двухсторонний связывание данных - Angular предлагает удобный подход к двустороннему связыванию данных, который автоматически синхронизирует изменения данных пользовательского интерфейса и модели данных;
- мощная система директив - Angular использует директивы для расширения функциональности HTML, что делает его более гибким и мощным;
- компонентная структура - Angular строится на компонентах, что делает код более организованным, легким для понимания и повторного использования;
- удобство тестирования - Angular предлагает удобные инструменты для тестирования приложений, что облегчает процесс отладки и повышает качество кода;
- поддержка Google- Angular разрабатывается и поддерживается компанией Google, что обеспечивает высокую степень надежности и стабильности фреймворка;
- Single Page Application (SPA) - означает приложение одной страницы. Другими словами, SPA – это web-приложение, размещенное на одной web-странице, которая для обеспечения работы загружает весь необходимый код

вместе с загрузкой самой страницы. Приложение такого типа появились сравнительно недавно, с началом эры HTML5 и SPA является типичным представителем приложений на HTML5.

В целом, Angular - это мощный инструмент для разработки веб-приложений, который предлагает большой набор функциональных возможностей, простоту использования и удобство тестирования, что делает его превосходным выбором для многих проектов.

Протоколом обмена информацией между базой данных и веб-интерфейсом является REST API созданный в фреймворке ASP.Net Web API.

ASP.NET Web API - это фреймворк для создания HTTP-сервисов, основанных на платформе .NET. Он имеет ряд преимуществ по сравнению с другими аналогами:

- интеграция с платформой .NET: ASP.NET Web API написан на языке программирования C# и полностью интегрируется с платформой .NET. Это позволяет использовать все возможности .NET Framework, такие как LINQ, Entity Framework и многое другое;
- поддержка RESTful API: ASP.NET Web API предоставляет поддержку RESTful архитектуры, что делает его очень гибким и легко масштабируемым;
- простота в использовании: ASP.NET Web API имеет простую и понятную структуру, которая позволяет быстро создавать и развертывать сервисы;
- поддержка форматов данных: ASP.NET Web API может работать с различными форматами данных, включая XML, JSON, BSON и другие;
- большое сообщество пользователей: ASP.NET Web API имеет большую базу пользователей, что обеспечивает широкую поддержку и доступ к множеству различных библиотек и инструментов;

- высокая производительность: ASP.NET Web API оптимизирован для быстродействия, что позволяет обрабатывать большое количество запросов за короткое время;
- надежность и безопасность: ASP.NET Web API предоставляет множество инструментов для обеспечения безопасности и надежности сервиса, включая аутентификацию, авторизацию, шифрование данных и другие функции.

В целом, ASP.NET Web API является отличным выбором для создания HTTP-сервисов и может быть использован для решения широкого спектра задач, связанных с обработкой и передачей данных через интернет.

3.2 Описание алгоритма решения задачи

Для решения задачи по работе с заявками о неисправностях оборудования колледжа можно использовать следующий алгоритм:

- создание системы для приема и обработки заявок. Для этого необходимо разработать веб-приложение или мобильное приложение, которое позволит студентам и преподавателям подавать заявки на ремонт оборудования. Пользователь должен иметь возможность указывать тип оборудования, описание неисправности и прикреплять фотографии;
- назначение исполнителя. Следующим шагом является назначение исполнителя на ремонт оборудования. Для этого любой из свободных сотрудников технической поддержки может принять заявку;
- выполнение работ. После назначения исполнителя, он может приступить к устранению проблемы. В процессе работы система должна отслеживать статус заявки и обновлять его. Пользователь должен иметь возможность отслеживать прогресс выполнения работ и о завершении работ;
- завершение работ и закрытие заявки. Как только ремонт оборудования закончен, исполнитель должен уведомить систему о завершении работ. После этого заявка будет помечена как выполнена. Система может

собирать данные о времени выполнения работ и другие параметры для последующего анализа эффективности работы системы.

Таким образом, система для работы с заявками о неисправностях оборудования колледжа позволит эффективно управлять ремонтом оборудования и повышать качество его обслуживания.

4. Программа решения задачи

4.1 Логическая структура

В данном разделе описана логическая структура проекта, построенного на платформе Angular.

Логическая структура приложения на Angular представлена следующим образом:

- модули - Angular-приложение состоит из одного или более модулей, каждый из которых содержит компоненты, сервисы и другие файлы, необходимые для работы приложения;
- компоненты - это строительные блоки приложения Angular, которые представляют собой части пользовательского интерфейса. В Angular компоненты отображаются в виде классов TypeScript, которые содержат шаблоны HTML и логику для управления этими шаблонами;
- сервисы - используются для выполнения различных задач, которые могут быть использованы в нескольких компонентах. Сервисы позволяют разделить код на логические блоки и упростить его тестирование;
- роутинг - используется для перенаправления пользователя между различными компонентами приложения. Он позволяет создавать более сложную навигацию между страницами веб-приложения;
- директивы - позволяют изменять поведение элементов DOM в приложении. Angular предоставляет два типа директив - атрибутные и структурные;
- Pipes - используются для обработки данных в шаблонах Angular. Они позволяют форматировать данные, фильтровать или изменять их перед отображением на странице;
- метаданные - используются для добавления дополнительной информации к классам компонентов, сервисов и других объектов в Angular-приложении.

Angular использует большое количество интересных паттернов проектирования, основные из которых представлены ниже.

MVC (Model-View-Controller) - это паттерн проектирования, который используется для разделения приложения на три компонента: модель (Model), представление (View) и контроллер (Controller).

- модель (Model) -это компонент, который отвечает за хранение данных и бизнес-логику приложения. Модель не должна содержать никакой логики, связанной с отображением данных или пользовательским интерфейсом;
- представление (View) - это компонент, который отвечает за отображение данных для пользователя. Представление получает данные из модели и отображает их в удобном для пользователя виде. Представление не должно содержать никакой бизнес-логики;
- контроллер (Controller) - это компонент, который отвечает за управление взаимодействием между моделью и представлением. Контроллер получает данные из модели, отправляет их в представление и обрабатывает действия пользователя, направленные на изменение данных в модели.

Преимущества использования паттерна MVC:

- разделение ответственности - MVC позволяет разделить ответственность между компонентами приложения - модель отвечает за данные и бизнес-логику, представление - за отображение данных, а контроллер - за управление взаимодействием между ними;
- легкая замена компонентов - при использовании MVC компоненты можно легко заменять, не затрагивая другие части приложения;
- упрощение тестирования - каждый компонент приложения может быть протестирован отдельно, что позволяет более эффективно тестировать приложение в целом;

В Angular реализация паттерна MVC отличается некоторыми особенностями. Вместо контроллера используется сервис, который выполняет подобные функции, а представление и модель интегрированы в компоненты.

Observable Pattern - это шаблон проектирования, который используется для создания связи между наблюдателем (observer) и объектом наблюдаемого (observable). Этот шаблон позволяет динамически устанавливать зависимости между объектами и уведомлять все заинтересованные стороны об изменениях в наблюдаемом объекте, он создан для управления асинхронными операциями и обработки потока данных.

Важным аспектом этого шаблона является то, что наблюдаемый объект не имеет прямого доступа к своим наблюдателям. Вместо этого он поддерживает список своих наблюдателей и уведомляет их о любых изменениях, вызывая соответствующие методы у каждого из них.

Такой подход позволяет отделить объекты наблюдаемого от конкретных наблюдателей и обеспечивает более гибкую и расширяемую модель взаимодействия между объектами.

Рассмотрим пример использования Observable Pattern в контексте разработки пользовательского интерфейса. Допустим, у нас есть приложение, которое содержит несколько компонентов, например, кнопку, текстовое поле и список. Каждый компонент может быть наблюдаемым объектом, который уведомляет своих наблюдателей (например, другие компоненты или контроллер) об изменениях в своем состоянии.

Например, когда пользователь нажимает на кнопку, она уведомляет наблюдателей об этом действии. Контроллер может реагировать на это событие, обновляя данные в списке или изменяя содержимое текстового поля.

Таким образом, использование Observable Pattern позволяет создавать гибкие и расширяемые приложения, которые могут динамически реагировать на изменения в своих компонентах и обеспечивать эффективную коммуникацию между объектами.

4.2 Физическая структура

Физическая структура проекта на Angular включает в себя ряд файлов и директорий, которые содержат компоненты, сервисы, модули, роутеры и другие элементы приложения.

Стандартная физическая структура проекта на Angular имеет следующий вид:

```
app/
-- components/
   -- component1/
      -- component1.component.ts
      -- component1.component.html
      -- component1.component.css
   -- component2/
      -- component2.component.ts
      -- component2.component.html
      -- component2.component.css
-- services/
   -- service1.service.ts
   -- service2.service.ts
-- modules/
   -- module1.module.ts
   -- module2.module.ts
-- router/
   -- app-routing.module.ts
-- app.component.ts
-- app.component.html
-- app.component.css
-- app.module.ts
-- main.ts
```

Рис 4.1 - Физическая структура проекта

В директории app/components/ располагаются компоненты приложения. Каждый компонент представлен отдельной директорией, в которой находятся TypeScript-файл компонента, HTML-шаблон и CSS-стили.

В директории app/services/ располагаются сервисы, которые предоставляют доступ к функциональности приложения.

Директория app/modules/ содержит модули, которые используются в приложении и группируют компоненты и сервисы, связанные между собой.

В директории app/router/ находится файл app-routing.module.ts, который определяет маршруты приложения.

Файл app.component.ts представляет главный компонент приложения, а в файлах app.module.ts и main.ts определяются модули и точка входа в приложение соответственно.

Также в проекте могут использоваться другие директории и файлы, например, для хранения статических ресурсов, конфигурационных файлов и т.д. Однако вышеперечисленные каталоги и файлы являются основными элементами физической структуры проекта на Angular.

Данный проект включает в себя 30 основных директорий, внутри которых хранятся либо файлы с исходным кодом, или такие же директории.

Основополагающей директорией является директория MVC.

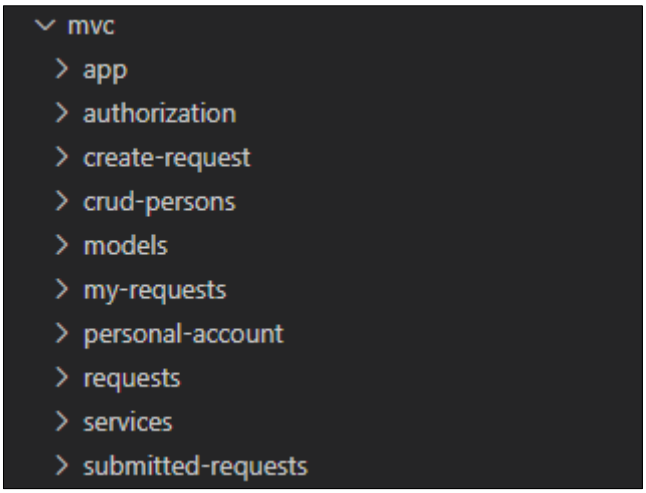

Рис 4.2 - Основополагающая директория

В ней (Рис 4.2 - Основополагающая директория) хранятся все компоненты и модели построенные на основе базы данных, а так же сервисы.

Компоненты в Angular - это основная единица разработки пользовательского интерфейса (UI). Каждый компонент определяет свою собственную логику и отображение внутри приложения Angular.

Компонент содержит четыре основных элемента:

- класс компонента;
- шаблон компонента;
- метаданные компонента;
- стили компонента.

Класс компонента содержит логику для управления поведением компонента, а шаблон компонента определяет, как компонент будет отображаться на странице. Метаданные компонента используются для описания компонента и его свойств. Стили компонента определяют стиль для элементов в шаблоне компонента.

Основным из компонентов данной системы является компонент app.

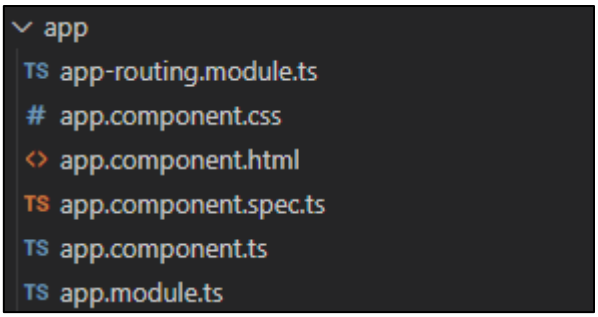

Рис 4.3 - Компонент App

Данный компонент является основным, так как содержит:

- app-routing.module.ts, в котором определено, как проходит маршрутизация на сайте;
- app.component.html содержит «шапку» сайта, которая отображается на каждой странице нашего веб-сайта;
- app.module.ts используется для определения и настройки основного модуля приложения, который является точкой входа для всего приложения. В этом файле определяются все зависимости (dependencies) приложения, такие как компоненты, сервисы, директивы, пайпы и т.д., которые будут использоваться в

приложении. Также в файле app.module.ts определяются настройки, необходимые для конфигурации приложения, например, местоположение корневого компонента, имя приложения, загрузчик данных и т.д. Без файла app.module.ts Angular не сможет запустить приложение, поскольку он отвечает за загрузку всех компонентов и настройку приложения.

Система для работы с заявками о неисправностях оборудования колледжа, как и любая другая имеет авторизацию, для этого создан отдельный компонент:

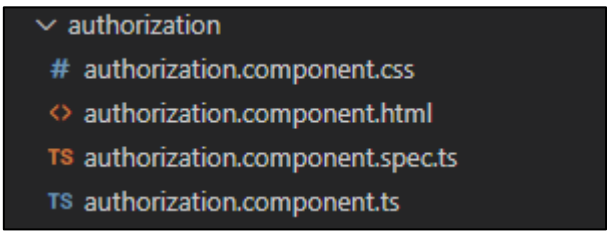

Рис 4.4 - Компонент Authorization

Данный компонент имеет стандартный набор элементов, в нем содержится веб-страница авторизации, а также исходный код в котором проходит аутентификация.

Далее представлен компонент, отвечающий за создание заявки о неисправности (Рис 4.5 - Компонент CreateRequest). View данного компонента содержит форму заполнения заявки, а контроллер отвечает за отправку данных в БД.

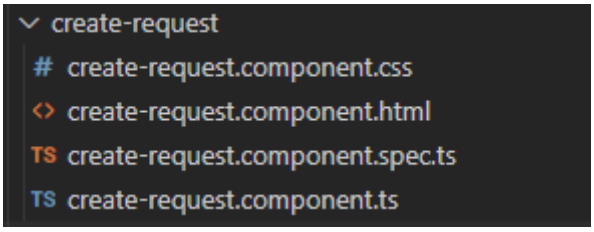

Рис 4.5 - Компонент CreateRequest

Компонент (Рис 4.6 - Компонент MyRequests) отвечает за отображение принятых активных заявок технического персонала, где имеется возможность добавить работы по заявке, а так же её завершить.

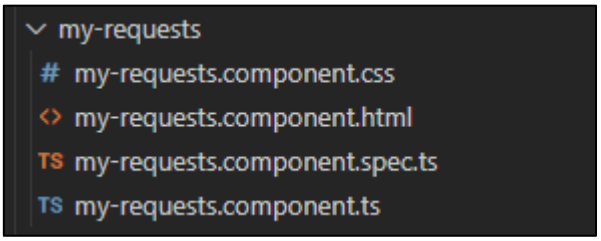

Рис 4.6 - Компонент MyRequests

Компонент (Рис 4.7 - Компонент PersonalAccount) отвечает за отображение личной информации авторизированного в системе пользователя, которую он может поменять. Так же директория имеет файл ChangePassword.html. Данный файл является диалоговым окном для изменения пароля.

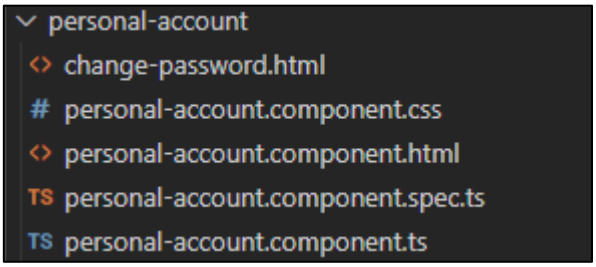

Рис 4.7 - Компонент PersonalAccount

Компонент (Рис 4.8 - Компонент Requests) отвечает за вывод всех активных заявок заявителей для технической поддержки, а так же менять их, посредством диалогового окна ChangeRequestDialogR.html, завершать с помощью окна CompleteRequestDialogR.html, а так же просматривать работы по заявке с помощью RequestInfoDialogR.html.

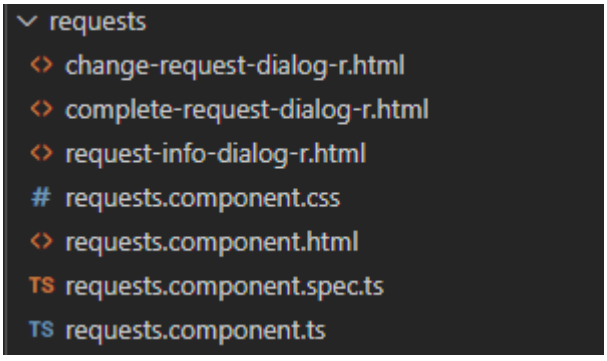

Рис 4.8 - Компонент Requests

Компонент (Рис 4.9 - Компонент SubmittedRequests) отвечает за отображение отправленных пользователем заявок о неисправностях, посмотреть информацию, изменить заявку, а так же её завершит, путем диалоговых окон как и в (Рис 4.8 - Компонент Requests)

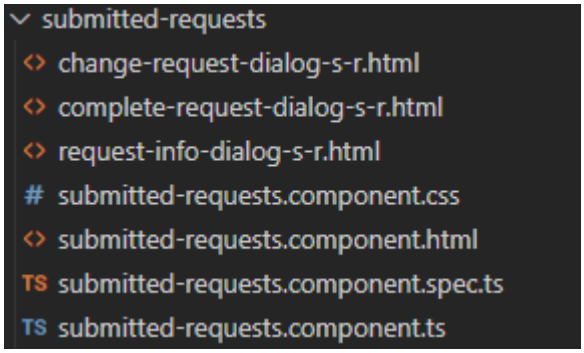

Рис 4.9 - Компонент SubmittedRequests

Компонент (Рис 4.10 - Компонент CrudPersons) отвечает за вывод всех пользователей в виде списка, и дальнейшее их редактирование с помощью диалогового окна EditUserDialog.html.

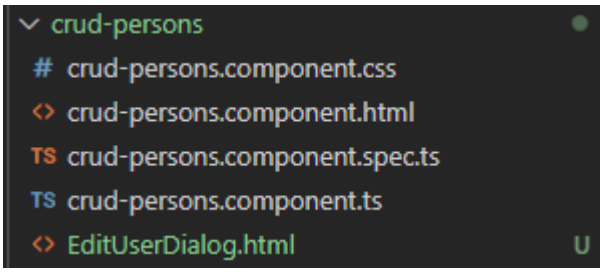

Рис 4.10 - Компонент CrudPersons

Далее представлена директория в которой содержится 17 моделей. Каждая из них построена в соответствии с сущностями в базе данных.

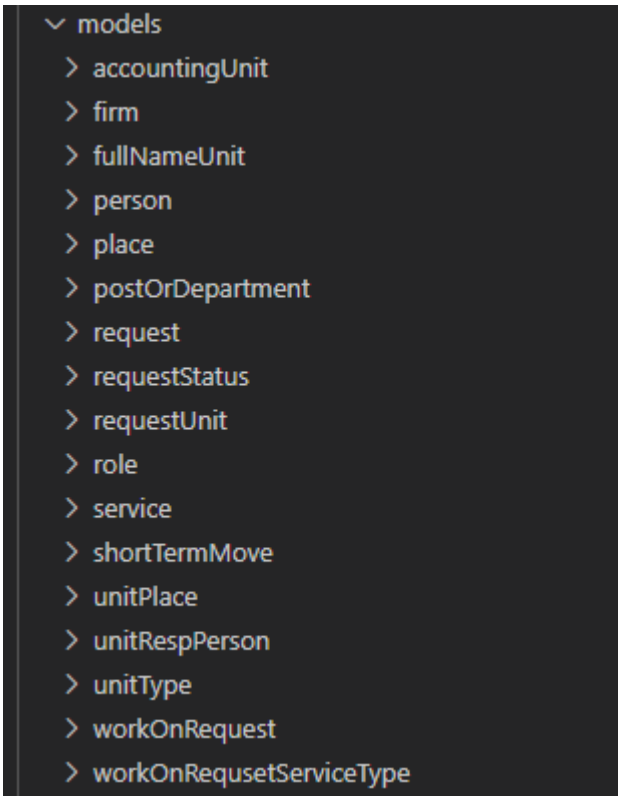

Рис 4.11 - Директория с моделями

На (Рис 4.12 - Https сервисы) приведены основные сервисы, которые использует система, их структура соответствует структуре API. Каждый из них получает определенные данные из БД, которые соответствуют названию сервиса. В сервисах хранятся методы, которые получают данные из БД посредством протокола HTTPS. Такие сервисы созданы из-за того, что Angular не имеет возможности прямого подключения к базе данных.

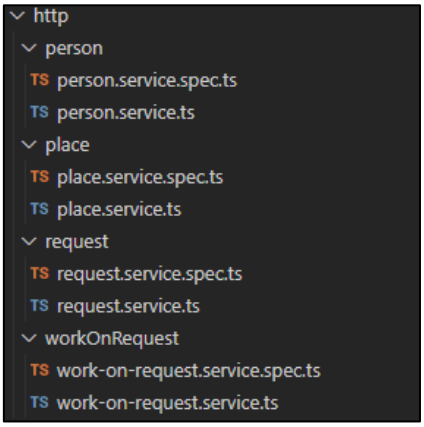

Рис 4.12 - Https сервисы

Так же в ходе проектирования системы было принято решение разработать специальные Гварды, так как она подразумевает авторизацию и управление доступом на основе ролей.

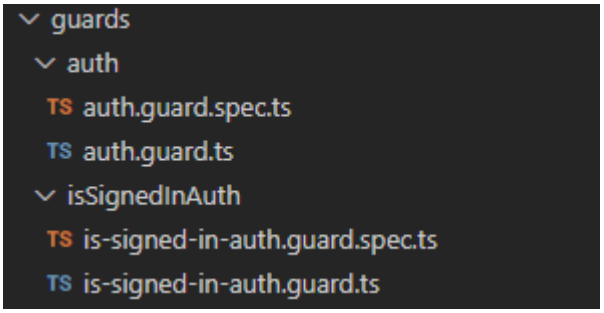

Рис 4.13 - Гварды авторизации

В Angular Guards - это классы, которые используются для контроля доступа к маршруту. Гварды используются для аутентификации пользователей и обеспечения авторизации для доступа к определенным частям приложения. Они также могут использоваться для предотвращения перехода с маршрута, если не выполняются определенные условия.

Существует четыре типа Guards в Angular:

- CanActivate – этот Guard используется для контроля доступа к маршруту. Он проверяет, аутентифицирован ли пользователь и имеет ли он права на доступ к маршруту, прежде чем разрешить ему продолжить;
- CanActivateChild - аналогично CanActivate, но он используется для контроля доступа к дочерним маршрутам;
- CanDeactivate - этот Guard используется для предотвращения перехода пользователя с маршрута, если не выполняются определенные условия. Например, он может использоваться для подтверждения того, хочет ли пользователь покинуть форму с несохраненными изменениями;
- CanLoad - этот Guard используется для предотвращения загрузки лениво загружаемого модуля, если не выполняются определенные условия. Он проверяет, имеет ли пользователь право на доступ к модулю, прежде чем загружать его.

Гварды могут быть добавлены к маршруту путем предоставления их в виде массива в свойствах canActivate, canActivateChild, canDeactivate или canLoad определения маршрута в модуле маршрутизации.

Данный гвард (Рис 4.14 - Гвард Auth) проверят авторизацию пользователя, и его роль, так как система использует управление доступом на основе ролей, пользователь с ролью client, не может иметь доступа к странице, где он не прописан в массиве доступных ролей.

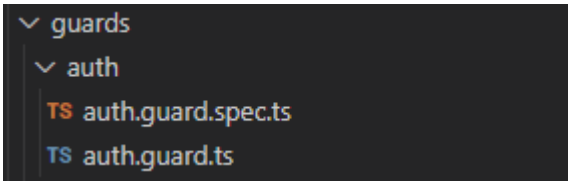

Рис 4.14 - Гвард Auth

Гвард (Рис 4.15 - Гвард IsSigniedInAuth) отвечает за проверку, авторизован ли пользователь, и если да, то он не позволяет пользователю перейти на страницу авторизации.

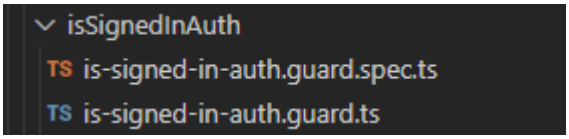

Рис 4.15 - Гвард IsSigniedInAuth

Как в (Рис 4.14 - Гвард Auth), так и в (Рис 4.15 - Гвард IsSigniedInAuth) используется гвард CanActivate.

5. Тестирование и отладка программы

Существует несколько видов тестирования сайтов, которые могут быть применены для проверки их работоспособности и соответствия заданным требованиям. Некоторые из них:

- функциональное тестирование - проверка работоспособности всех функций и возможностей сайта;
- нагрузочное тестирование - проверка способности сайта выдерживать обратную связь при высокой нагрузке на сервер;
- тест на совместимость - проверка работы сайта на различных платформах, браузерах и устройствах;
- тестирование безопасности - проверка защиты сайта от злоумышленников и возможность внедрения вредоносного кода в систему;
- тестирование юзабилити - проверка удобства использования сайта, его простоты и удобства в использовании пользователем;
- тестирование совместимости с поисковыми системами - проверка соответствия сайта правилам поисковых систем, что может повлиять на ранжирование сайта в поисковых системах;
- тестирование на соответствие стандартам - проверка соответствия сайта стандартам HTML, CSS, JavaScript и другим языкам программирования.

Конкретные виды тестирования зависят от целевых задач, требований к сайту и его функциональных возможностей. Важно учитывать все эти факторы при планировании процесса тестирования для достижения наилучших результатов.

После анализа всех видов тестирования, было выбрано функциональное тестирование, так как посчиталось самым эффективным. Функциональное тестирование включает в себя следующие виды тестирования:

- тестирование пользовательского интерфейса - проверка работоспособности графического интерфейса сайта, удобства и простоты использования;
- тестирование функций - проверка работоспособности всех функциональных возможностей сайта, таких как формы обратной связи, поиск, корзина покупок, оплата, регистрация;
- тестирование базы данных - проверка корректности хранения и обработки данных на сервере;
- тестирование производительности - проверка скорости работы сайта при обработке большого количества запросов;
- тестирование совместимости - проверка работоспособности сайта на различных браузерах, операционных системах и устройствах;
- тестирование безопасности - проверка защиты сайта от взлома, атак и утечек конфиденциальной информации;
- тестирование локализации - проверка работоспособности сайта на разных языках и соответствия местным культурным особенностям;
- тестирование сценариев - проверка корректности выполнения комплексных сценариев, таких как процедуры заказа, регистрации и оплаты товаров.

Каждый вид тестирования имеет свои особенности и специфичные требования для проверки работоспособности сайта. Важно выбрать соответствующий подход к тестированию в зависимости от целей и требований к сайту.

Тестирование функциональности проводилось вручную, каждый функциональный блок каждой страницы, был протестирован большое количество раз, в следствии чего были выявлены ошибки, и изменен исходный код.

6. Применение

6.1 Назначение программы

Система для работы с заявками о неисправностях оборудования колледжа является инструментом, который позволяет упростить и автоматизировать процесс обработки и управления заявками на ремонт или замену неисправного оборудования в колледже.

Основной целью данной системы является обеспечение бесперебойной работы оборудования и прикладных программ. Для этого программа предоставляет возможность быстрого и эффективного учета всех поступающих заявок, их распределения между исполнителями и отслеживания статуса выполнения.

Программа позволяет создавать заявки на ремонт оборудования или о неисправностях. Вся информация о заявках хранится в единой базе данных, которая позволяет оптимизировать процесс работы с ними, обеспечивает быстрый поиск и доступ к необходимой информации.

С помощью данной программы можно контролировать всю цепочку выполнения заявок - от их поступления до фактического выполнения работ.

Кроме того, система позволяет получать статистические отчеты о количестве поступающих заявок, времени их выполнения, а также о причинах неисправностей оборудования. Эти данные могут быть использованы для оптимизации работы колледжа и повышения качества обслуживания.

Таким образом, программа для работы с заявками о неисправностях оборудования колледжа является важным инструментом для эффективного управления и контроля за состоянием техники, обеспечения бесперебойной работы колледжа и повышения удовлетворенности сотрудников.

6.2 Требования к аппаратным ресурсам ПК

Angular - это современный инструмент для создания веб-приложений, которые могут работать на различных устройствах и платформах. Для пользования сайтом, построенным на Angular, требуются следующие аппаратные ресурсы:

- процессор. Angular может работать на любом процессоре x86-64, который имеет тактовую частоту не менее 1 ГГц. Это достаточно стандартное требование, которое выполняется большинством современных компьютеров;
- оперативная память. Для запуска приложения Angular требуется, чтобы устройство имело оперативную память объемом не менее 4 Гб. Однако, если вы используете веб-браузеры, которые потребляют большое количество памяти, например, Google Chrome, то рекомендуется иметь более 8 Гб ОЗУ;
- видеокарта. Для работы с Angular необходима видеокарта, поддерживающая WebGL и DirectX 11. WebGL - это технология, которая позволяет рендерить графику на GPU (графическом процессоре), что обеспечивает более быструю отрисовку изображений и более плавную работу интерфейса. DirectX 11 - это библиотека графического программного обеспечения, которая используется для работы с Windows;
- жесткий диск. Для установки и запуска приложения Angular необходимо иметь свободное место на жестком диске не менее 500 Мб. Однако, если вы планируете работать с большим объемом данных, вам может потребоваться больше места на диске.

В целом, эти требования являются минимальными и могут изменяться в зависимости от конфигурации сайта, его функциональности и объема обрабатываемых данных. Также следует помнить, что оптимальная работа сайта будет достигаться при использовании последних версий браузеров, таких как Google Chrome, Mozilla Firefox, Safari, Microsoft Edge или Opera.

6.3 Руководство пользователя

Данный раздел объясняет, как пользоваться системой для работы с заявками о неисправностях оборудования колледжа. Руководство содержит информацию о том, как:

- перейти на сайт;
- авторизоваться;
- оставить заявку;
- посмотреть информацию о проведенных работах по заявке;
- изменить информацию о оставленной заявке, в случае если данные были вписаны неправильно;
- завершить заявку, в случае если неисправность была само устранена.

Для того, чтобы перейти на сайт системы, нужно открыть любой из имеющихся браузеров, в данном случае мы будем использовать Mozilla Firefox. Далее ввести в адресную строку, расположенную в шапке браузера (Рис 6.1 - Ввод URL в адресную строку) следующий URL – “http://localhost:4200”.

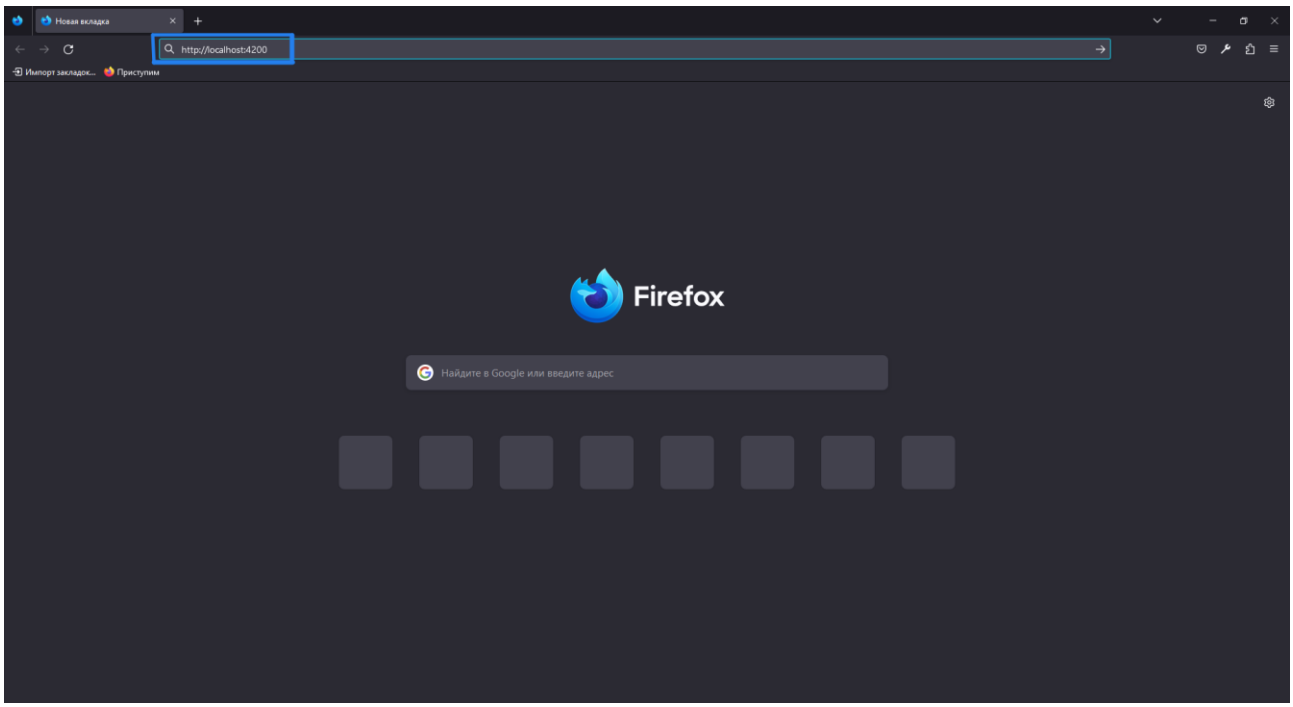

Рис 6.1 - Ввод URL в адресную строку

После чего произойдет переадресация на сайт системы. Первоначальной страницей (Рис 6.2 - Страница авторизации) является страница авторизации, на которой нужно ввести уникальный логин в поле «Логин», и пароль в поле «Пароль», выданный вам техническим персоналом.

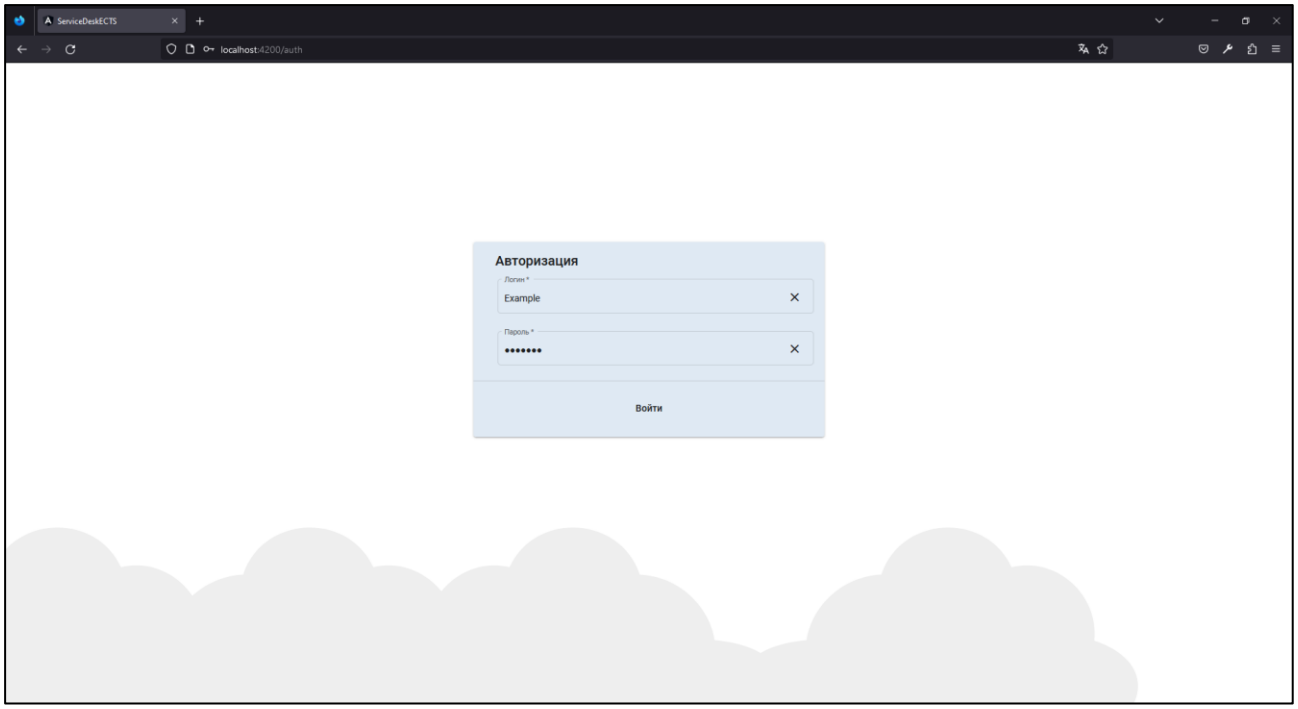

Рис 6.2 - Страница авторизации

В случае, если данные введены неверно, внизу посередине экрана (Рис 6.3 - Уведомление «Данные введены неверно») появиться уведомление, что данные неверны. Если вы уверены, что всё введено верно, то обратить в техническую поддержку.

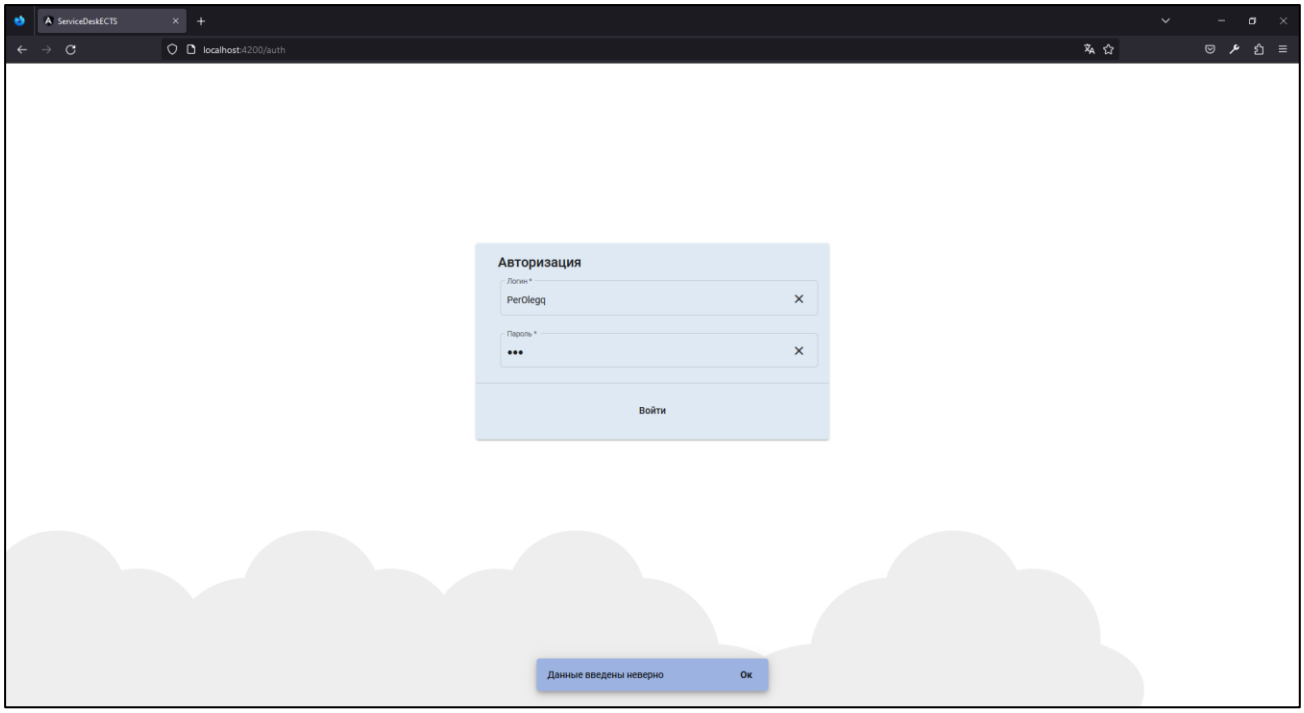

Рис 6.3 - Уведомление «Данные введены неверно»

В случае если всё введено верно, то произойдет переадресация на URL “http://localhost:4200/submittedRequests”, и откроется страница (Рис 6.4 – Пустая страница отправленных заявок) с заявками отправленного от пользователя, под которым мы вошли. На данной странице отображаются заявки, которые когда-либо были оставлены. Так как вход осуществляется первый, страница с заявками не содержит как активных, так и завершенных заявок.

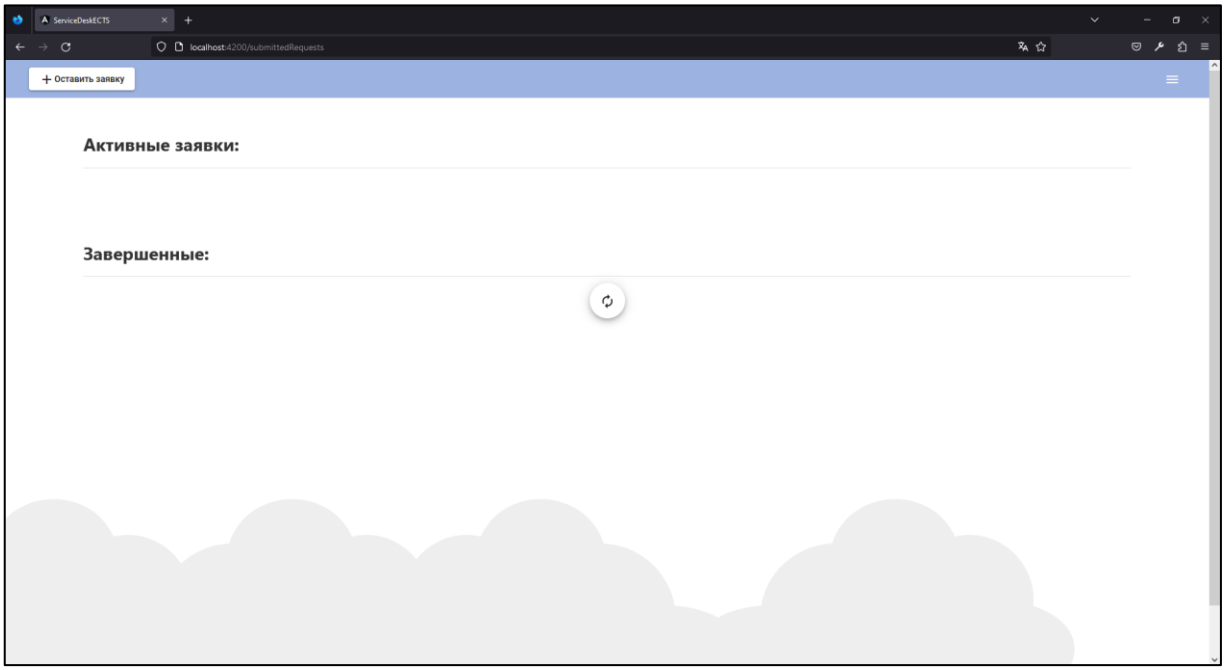

Рис 6.4 – Пустая страница отправленных заявок

Если у вас возникла, любая из технических проблем с оборудованием, которым работаете вы или ваши студенты, то можно оставить заявку нажав на кнопку в левом верхнем углу (Рис 6.5 - Кнопка "Оставить заявку").

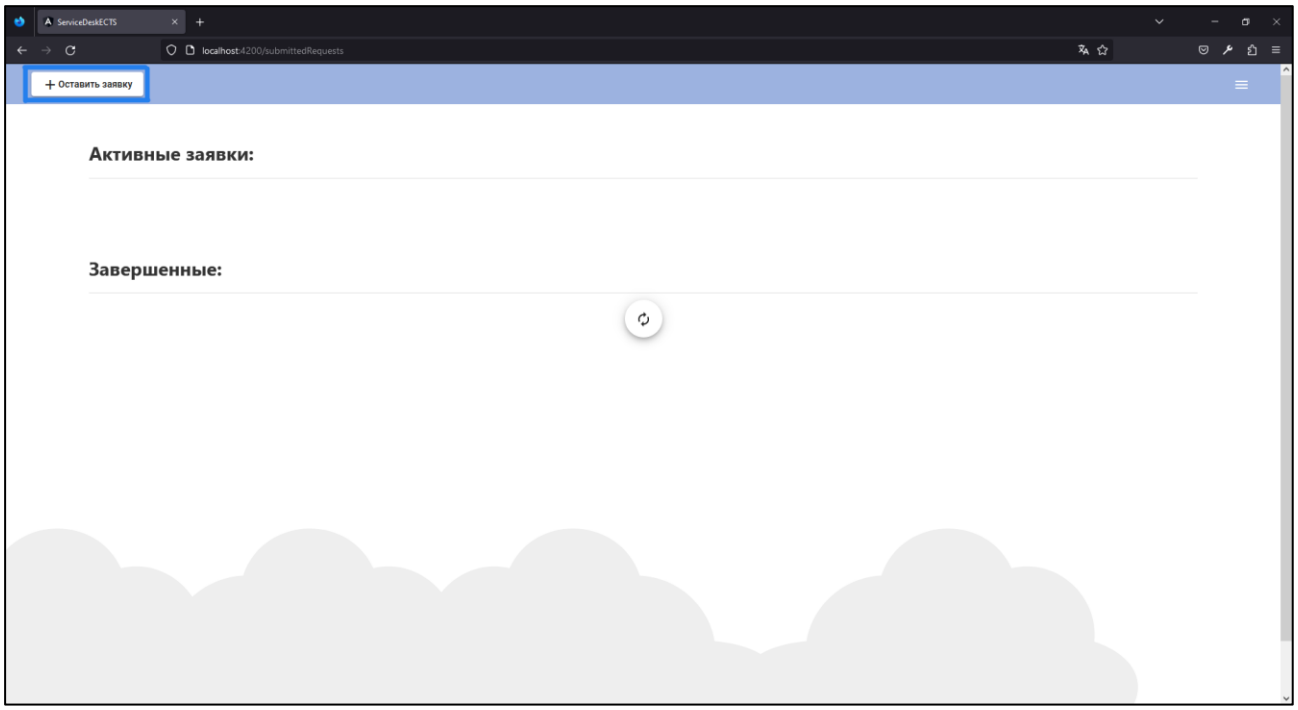

Рис 6.5 - Кнопка "Оставить заявку"

После чего произойдет переадресация на URL “http://localhost:4200/createRequest”, и откроется страница с формой создания заявки (Рис 6.6 - Страница создания заявки).

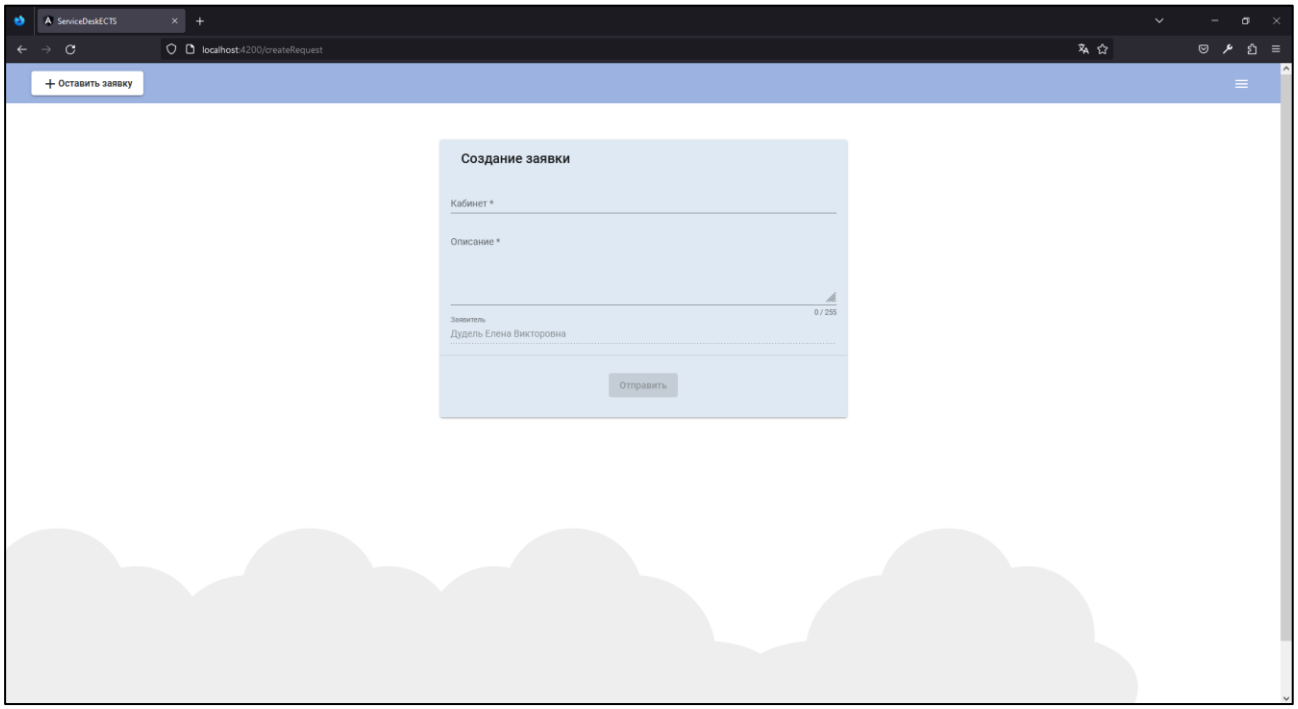

Рис 6.6 - Страница создания заявки

На форме в поле «Кабинет», выбирается тот кабинет, в котором имеется техническая проблема. Кабинет можно выбрать из выпадающего списка (Рис 6.7 - Выпадающий список).

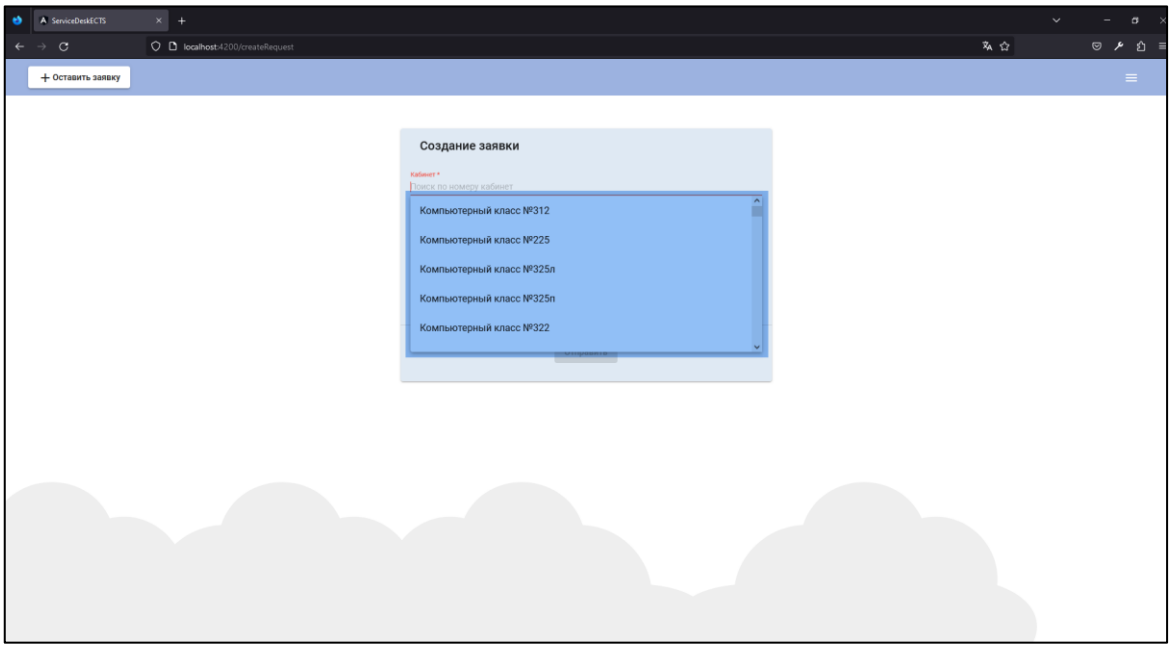

Рис 6.7 - Выпадающий список

Так же в поле «Кабинет» можно произвести поиск кабинета, путем ввода его номера (Рис 6.8 - Поиск кабинета по номеру) и выбора имеющегося кабинета из выпадающего списка.

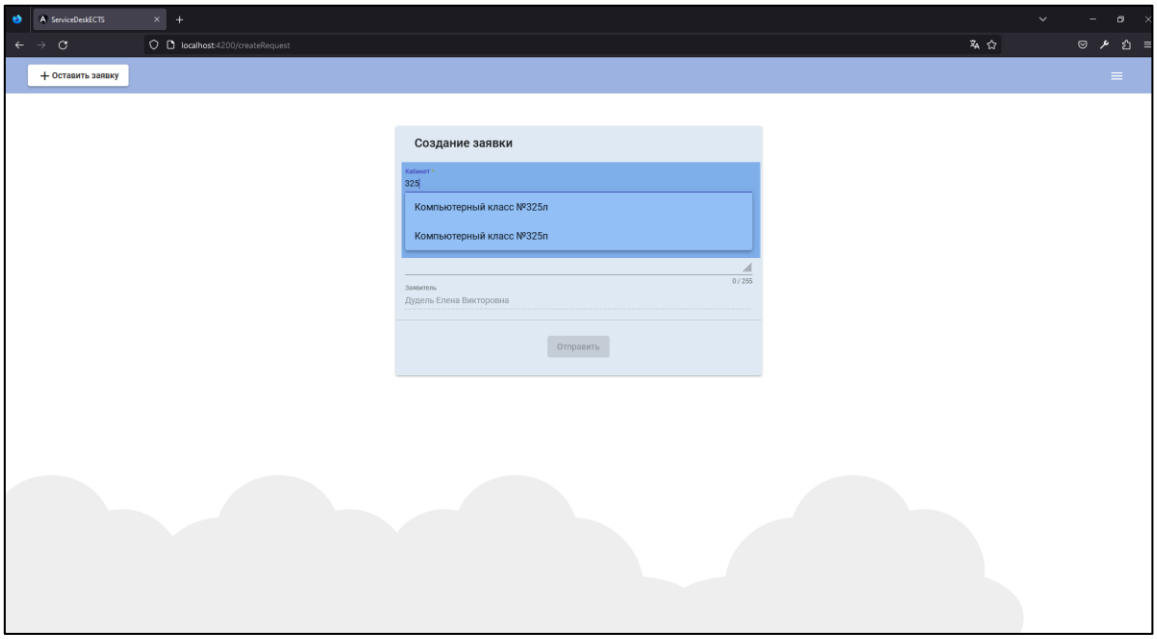

Рис 6.8 - Поиск кабинета по номеру

Далее перейдем к заполнению поля «Описание» (Рис 6.9 - Поле "Описание"), здесь нужно как можно конкретнее описать проблему или проблемы, на каких устройствах она возникла, а также примерное время, когда она произошла. Поле «Заявитель» заполняется автоматически.

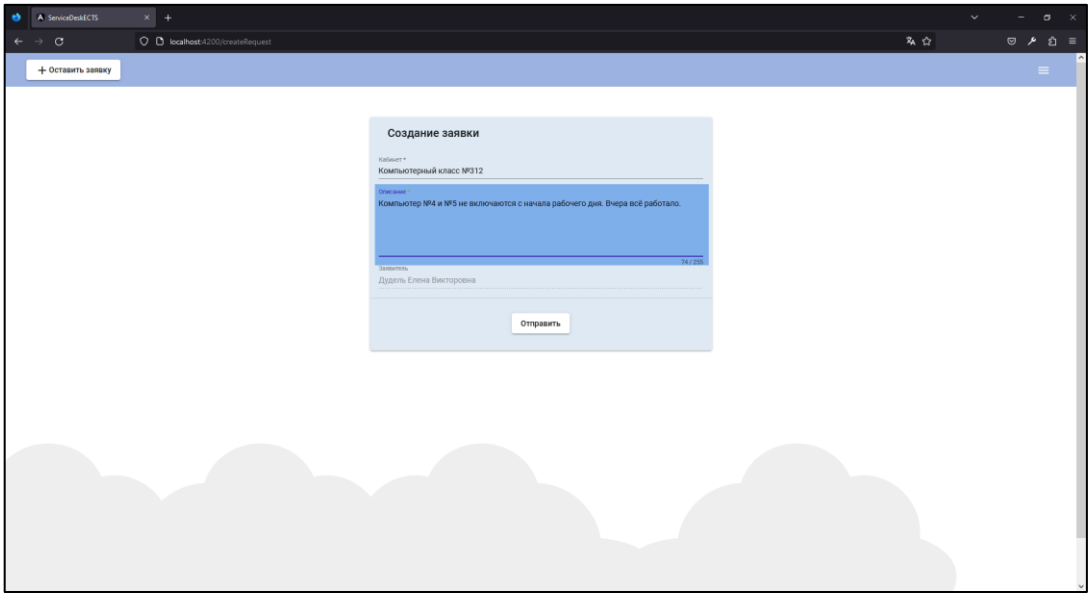

Рис 6.9 - Поле "Описание"

После того, как все поля заполнены, мы можем отправить заявку, путём нажатия кнопки «Отправить» (Рис 6.10 - Кнопка "Отправить").

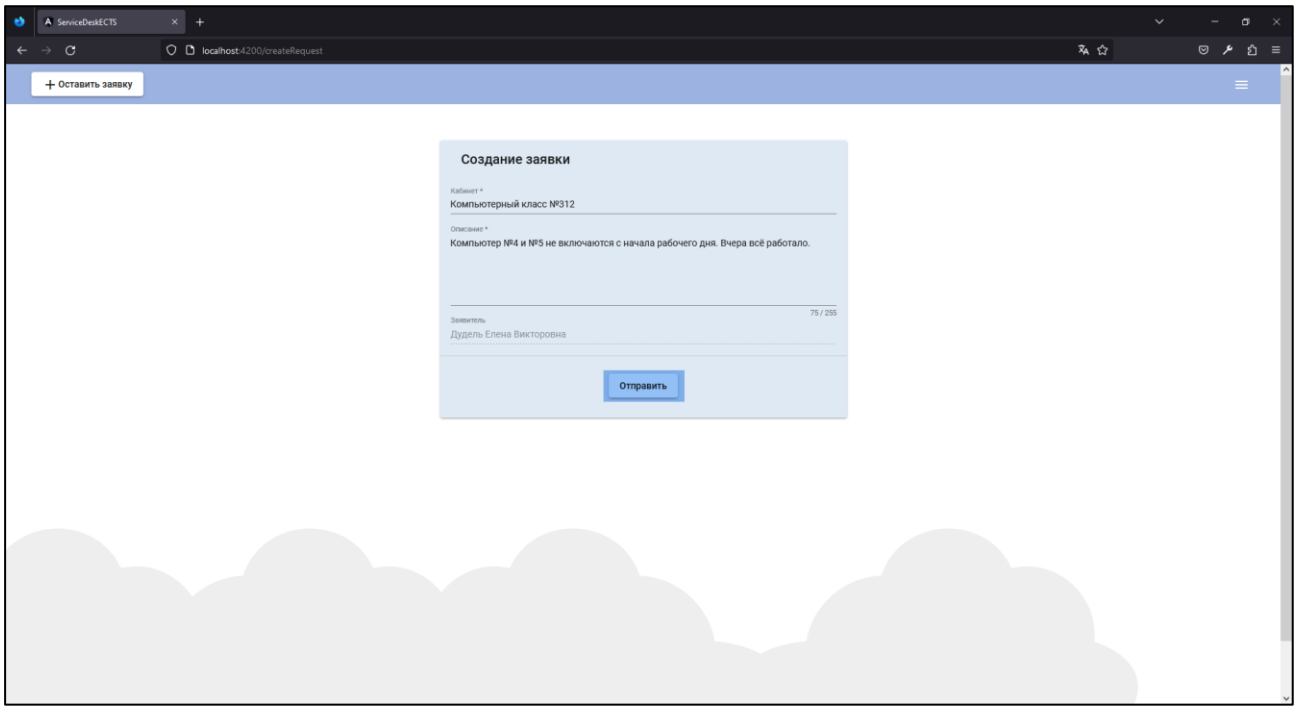

Рис 6.10 - Кнопка "Отправить"

В случае если данные отправлены успешно внизу посередине экрана (Рис 6.11 - Уведомление "Заявка отправлена") вылезет соответствующее уведомление.

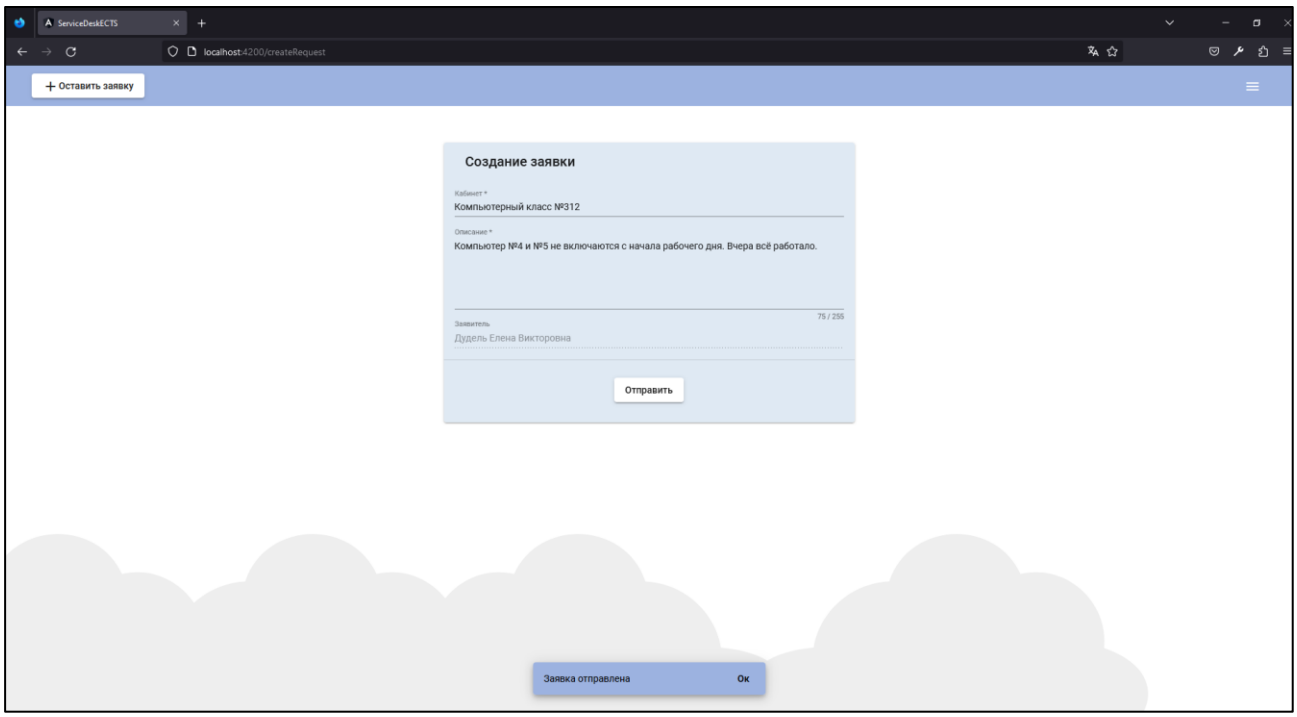

Рис 6.11 - Уведомление "Заявка отправлена"

Спустя 5 секунд или путем нажатия кнопки «Ок», на уведомлении (Рис 6.11 - Уведомление "Заявка отправлена"), произойдет переадресация на URL “http://localhost:4200/submittedRequests”, и откроется страница с отправленными заявками (Рис 6.12 - Страница отправленных заявок).

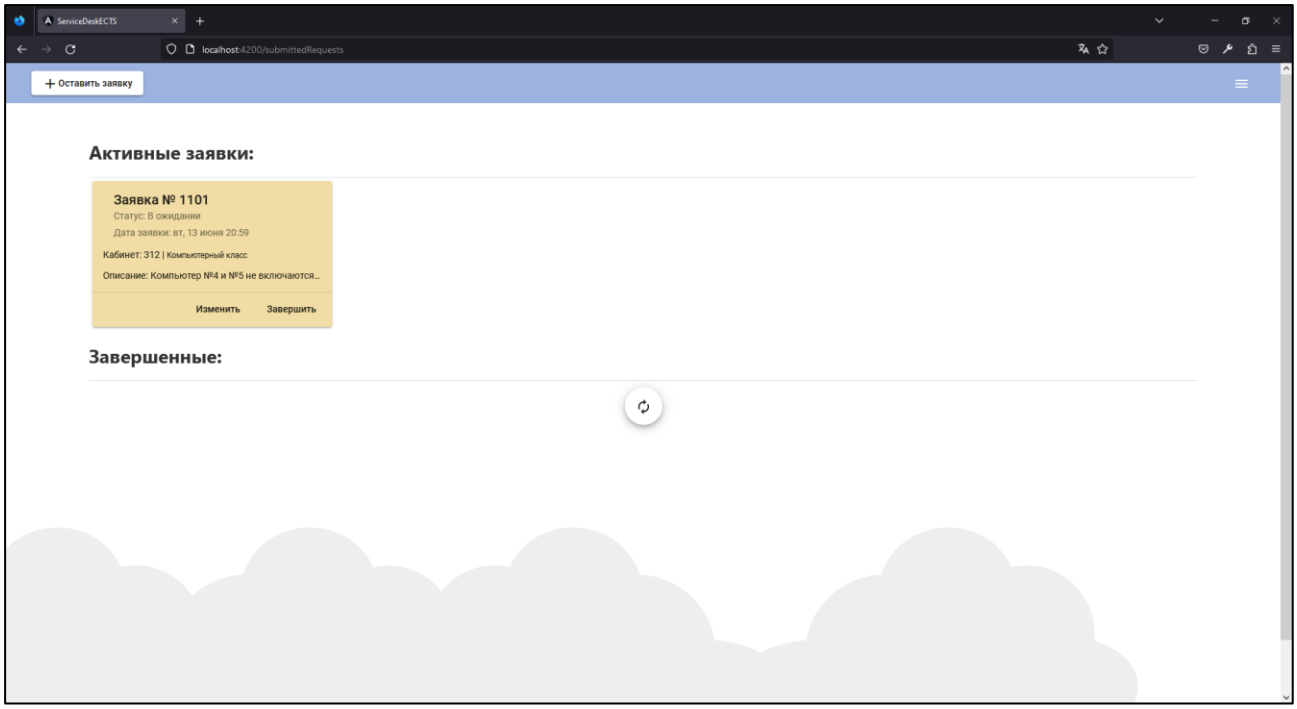

Рис 6.12 - Страница отправленных заявок

Как мы видим список активных заявок пополнился. Так как заявка только что отправлена её статус «В ожидании». Если возникла нужда дополнить заявку или изменить в ней данные, в случае, когда они указаны неверно, то путем нажатия кнопки «Изменить» на нашей заявке (Рис 6.13 - Кнопка "Изменить").

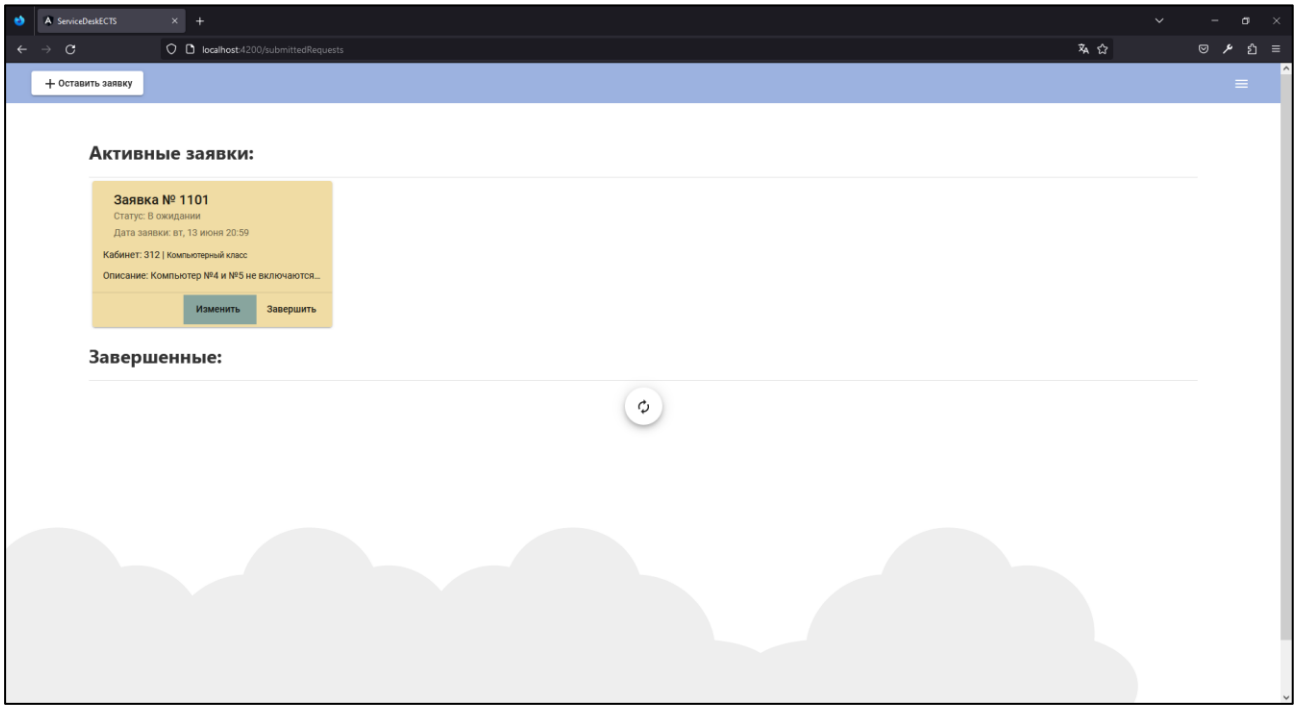

Рис 6.13 - Кнопка "Изменить"

После чего выведется диалоговое окно (Рис 6.14 - Диалоговое окно изменения заявки), в котором указаны данные о заявке, которые мы раннее заполняли. Для того, чтобы изменить данные, надо перейти на нужное поле путем нажатия по нему, и изменить. А после нажать на кнопку «Изменить» (Рис 6.15 - Кнопка "Изменить").

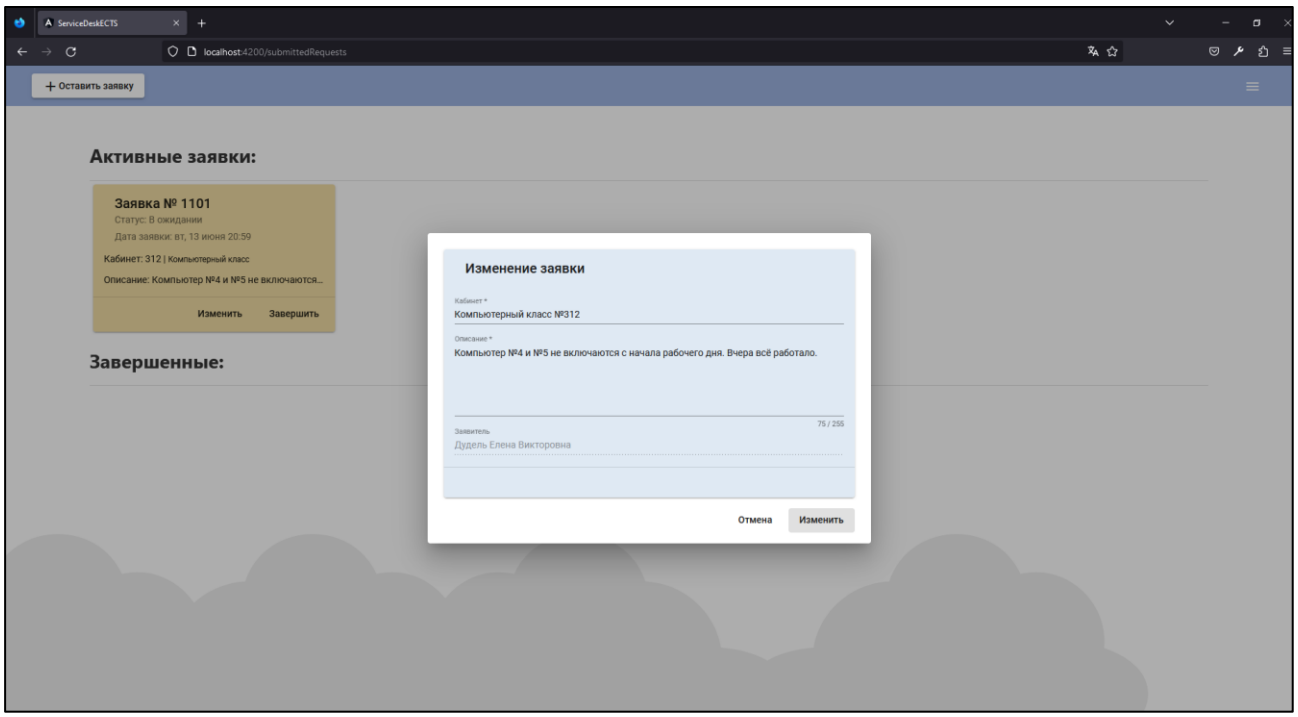

Рис 6.14 - Диалоговое окно изменения заявки

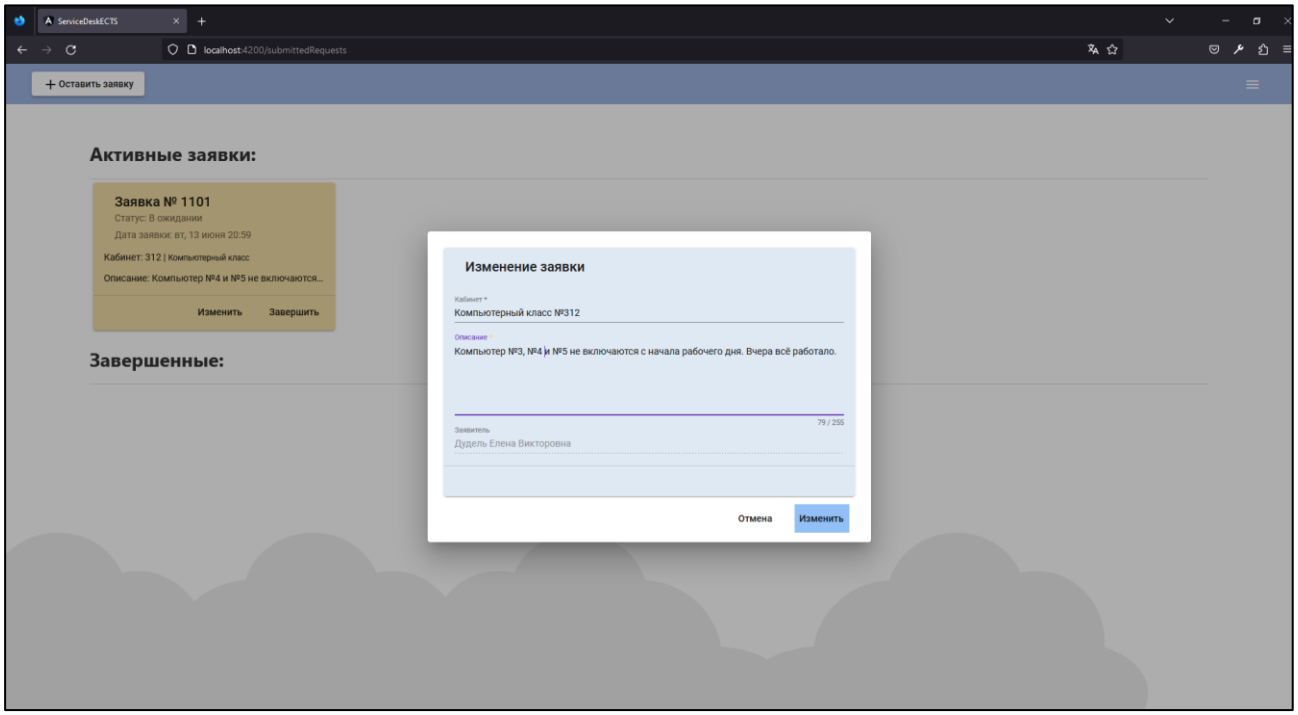

Рис 6.15 - Кнопка "Изменить"

Если потребность в изменении данных пропала, то следует нажать на кнопку «Отмена» (Рис 6.16 - Кнопка "Отмена").

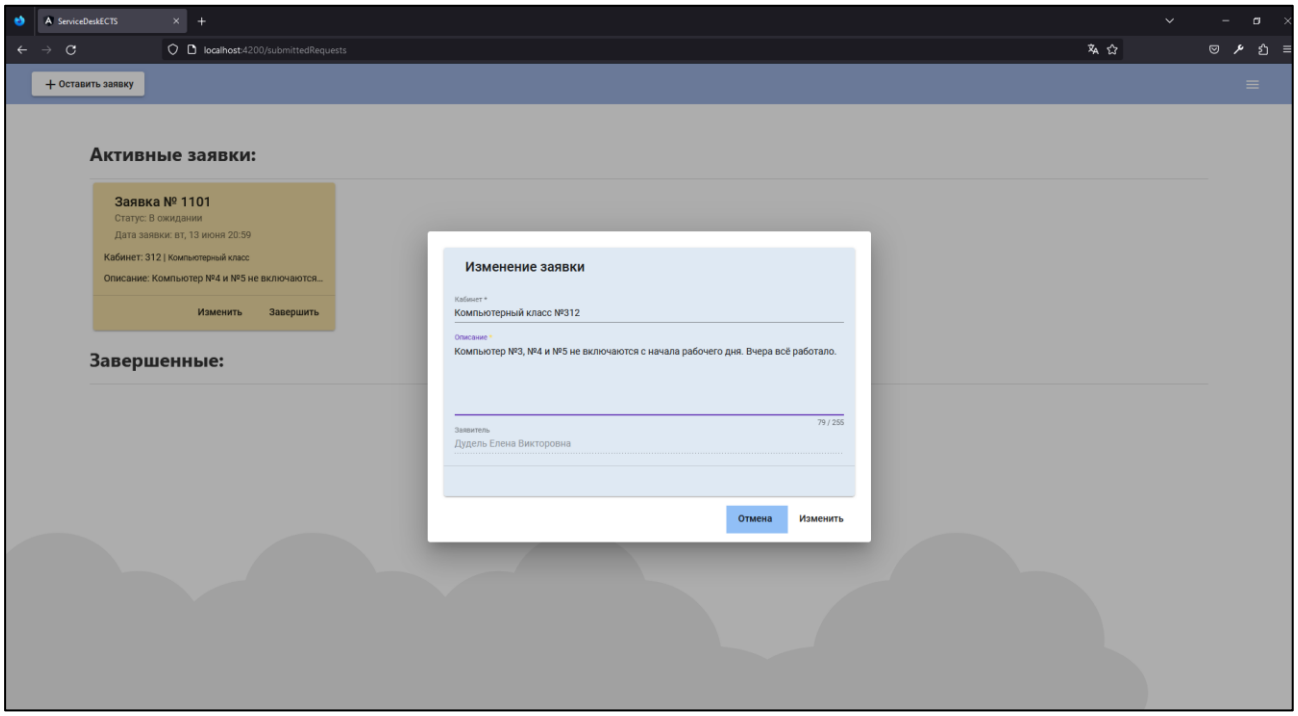

Рис 6.16 - Кнопка "Отмена"

В случае если проблема устранилась сама, или любой другой из пользователей её устранил (Что не рекомендуется делать, если он не является сотрудником технической поддержки), то следует нажать кнопку «Завершить» (Рис 6.17 - Кнопка "Завершить").

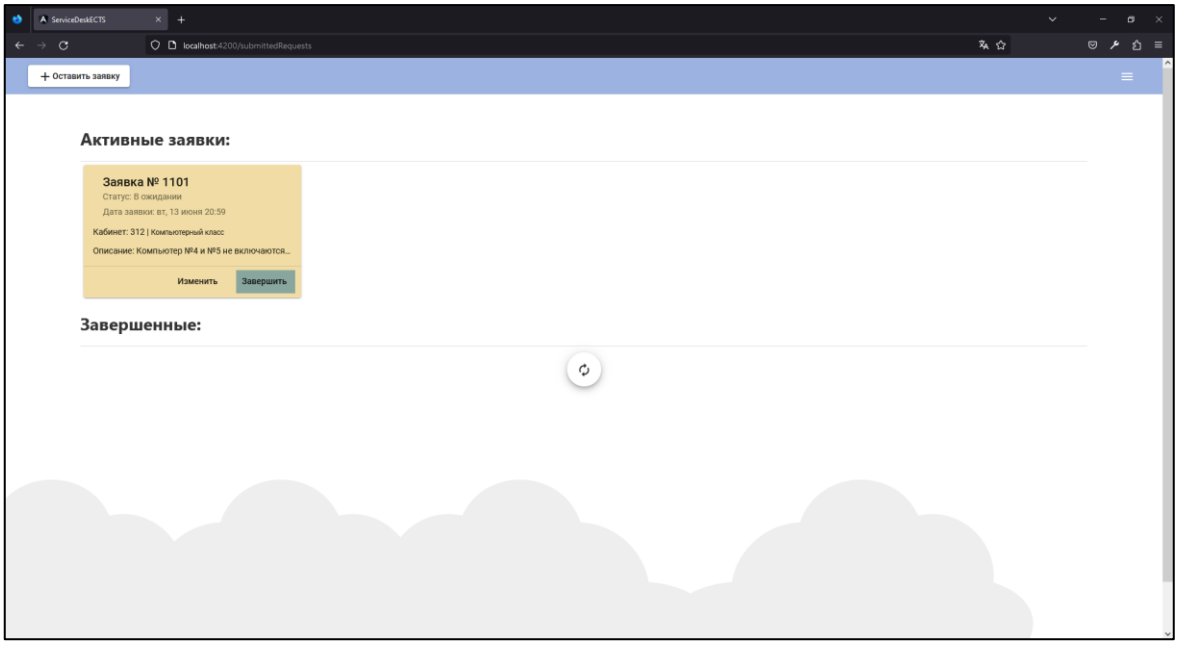

Рис 6.17 - Кнопка "Завершить"

После чего появиться диалоговое окно, в котором предложено завершить задачу, в случае нажатия кнопки «Отмена» диалоговое окно закроется, в случае нажатия кнопки «Да», задача завершится, и работы по ней проводится не будут.

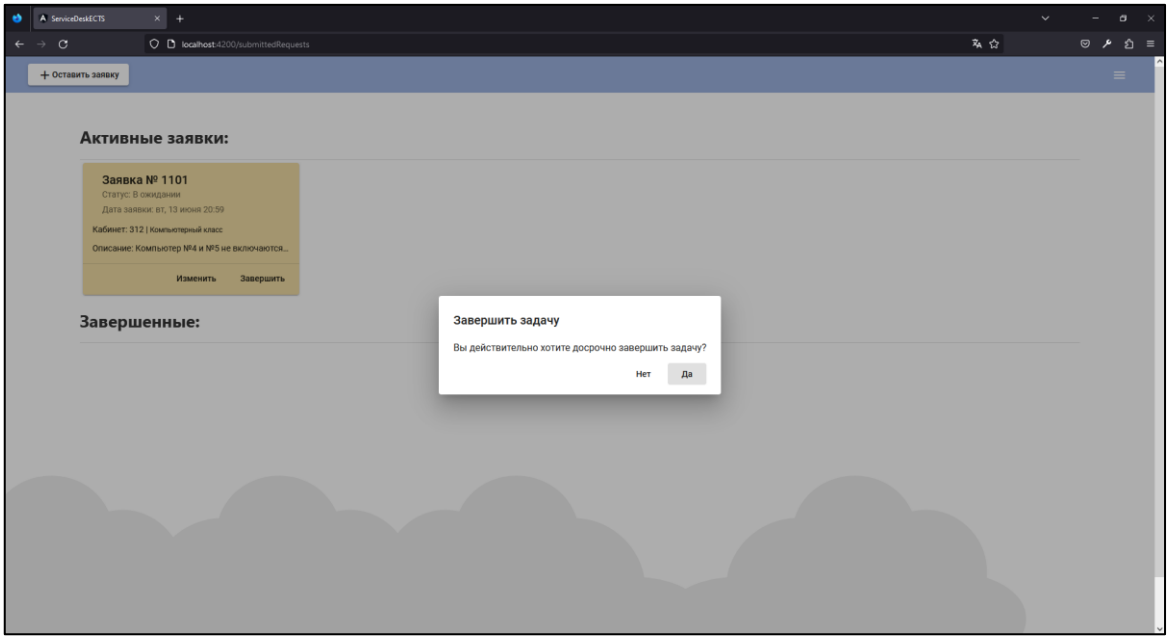

Рис 6.18 - Диалоговое окно "Завершить задачу"

Если же наша проблема до сих пор актуальна и статус заявки «В ожидании», то просто ждём, пока сотрудник технической поддержки её примет.

Когда нашу заявку приняли, её статус поменяется на «Принята в работу», а цвет поменяется с жёлтого на синий.

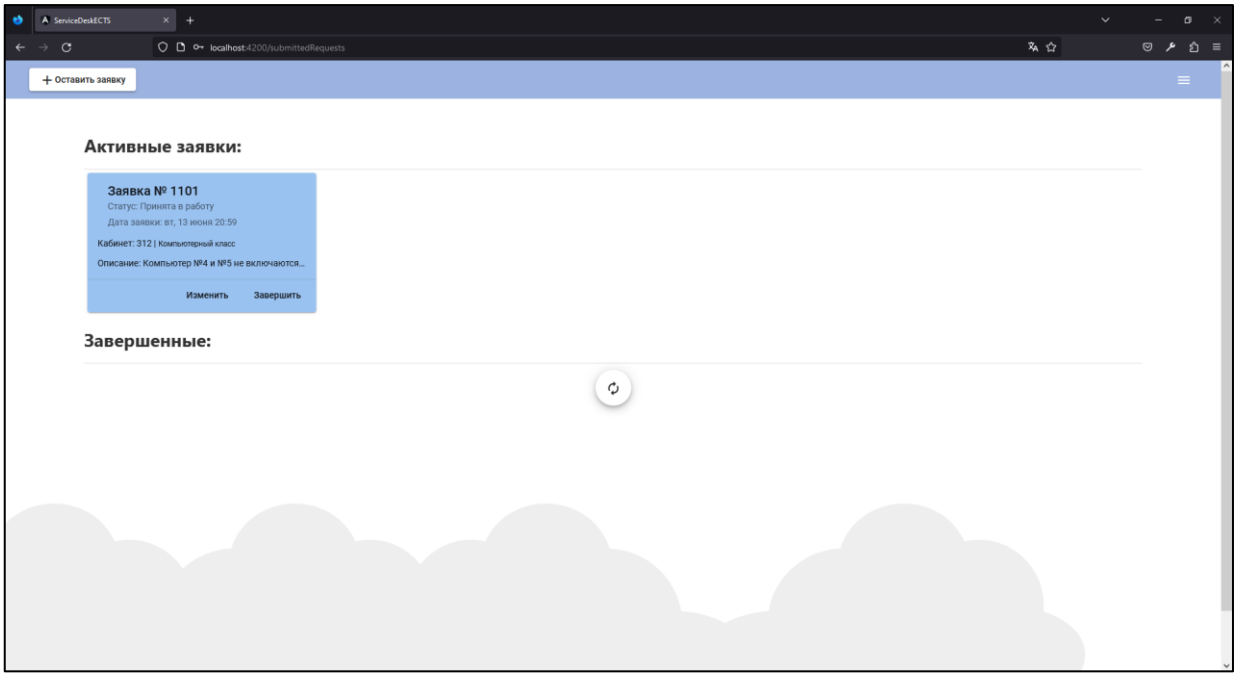

Рис 6.19 - Принятая заявка

Для того, чтобы посмотреть, какие работы проводились по заявке следует кликнуть по ней (Рис 6.20 - Клик по заявке)

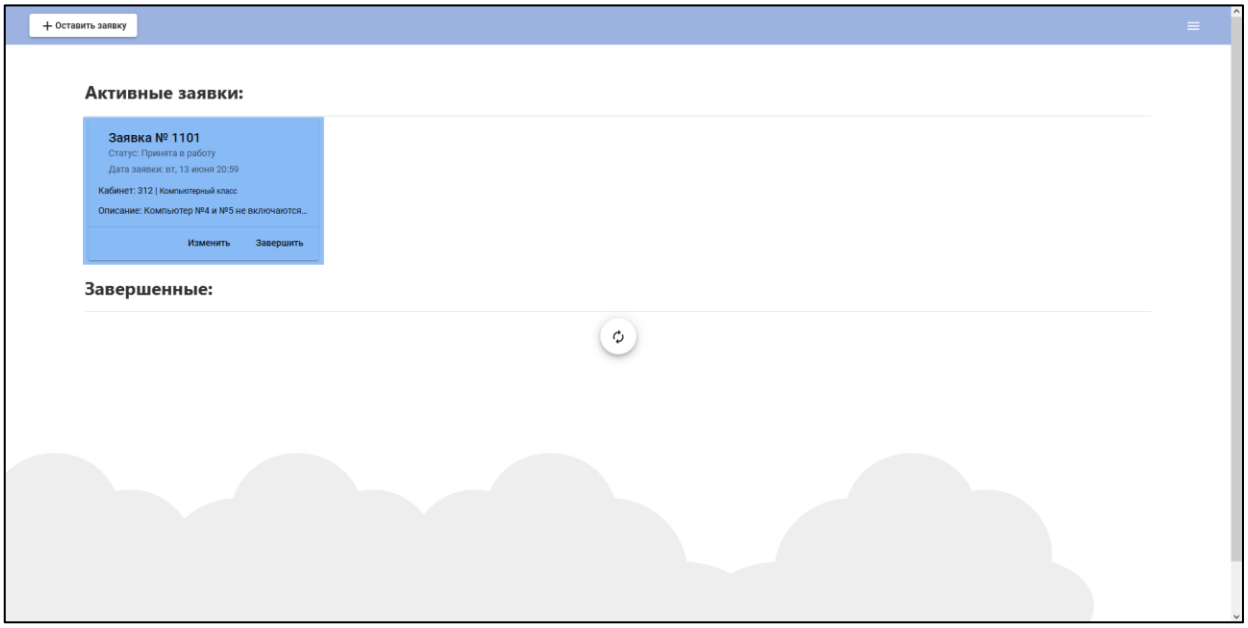

Рис 6.20 - Клик по заявке

После откроется диалоговое окно (Рис 6.21 - Диалоговое окно "Информация о заявке"), в котором будут описаны работы, проведенные по заявке, в нашем случае её только приняли, и никаких работ не проводилось.

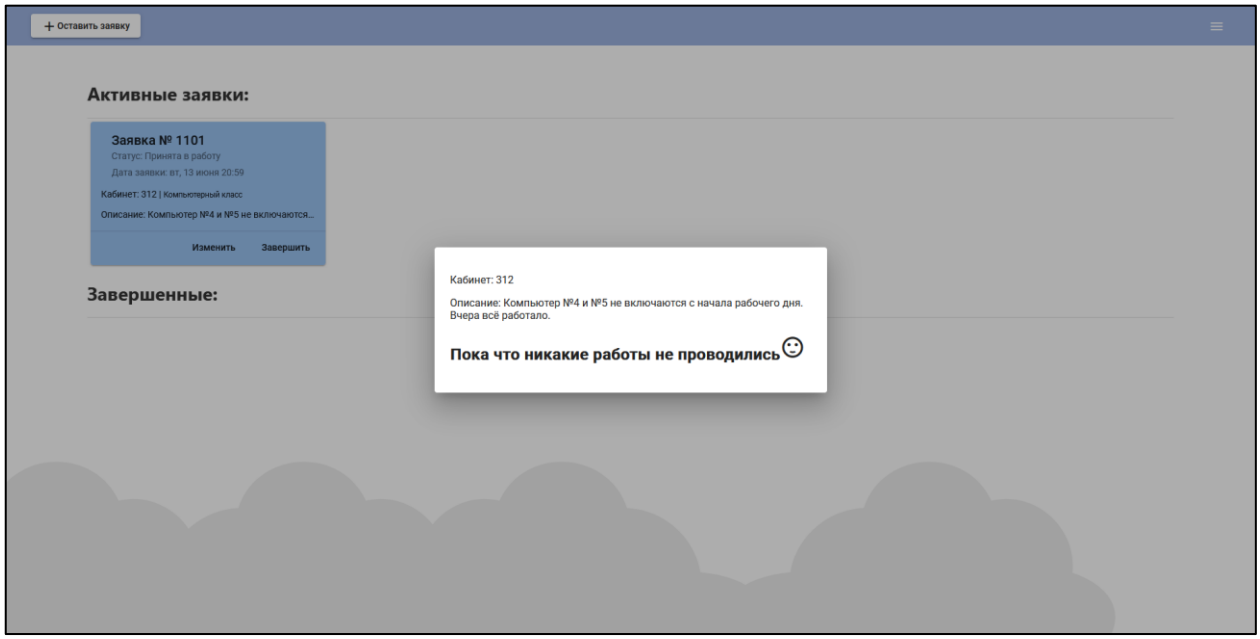

Рис 6.21 - Диалоговое окно "Информация о заявке"

Так же сайт имеет кнопку в правом верхнем углу (Рис 6.22 – Меню), по нажатию на которую появиться меню, можно перейти в личный кабинет, или на страницу с отправленными заявками.

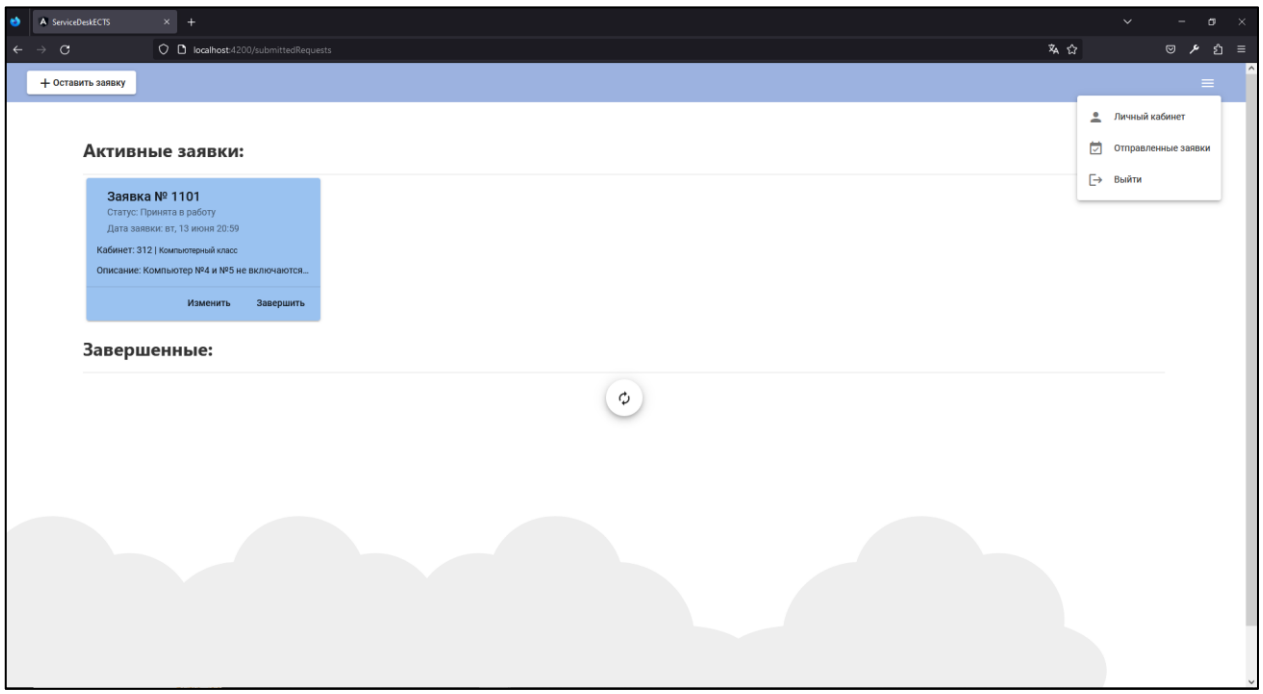

Рис 6.22 – Меню

На странице лично кабинета можно посмотреть свои данные, редактировать их, возможности нету. Если какие-то из них указаны неверно, обратитесь в техническую поддержку.

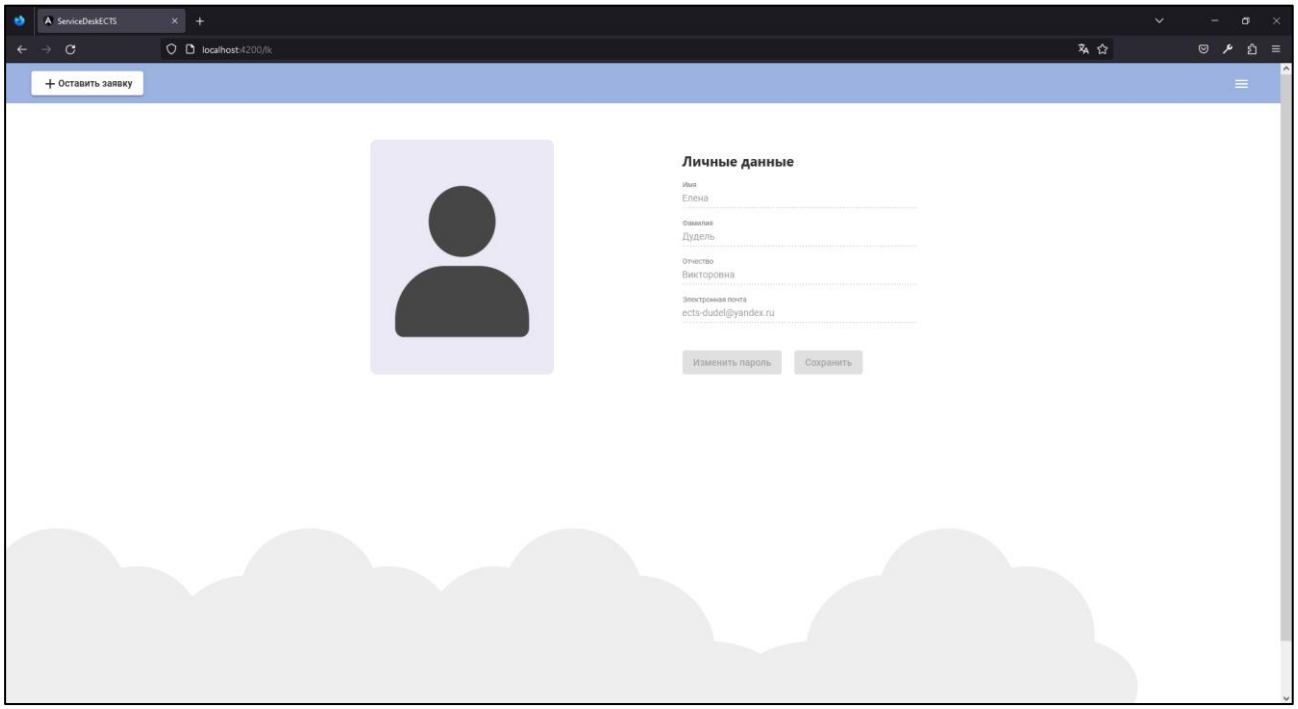

Рис 6.23 - Страница личного кабинета

7. Охрана труда и противопожарная безопасность

Охрана труда - это комплекс мероприятий, направленных на защиту работников от вредного и опасного воздействия производственных факторов. Она осуществляется с целью предотвращения несчастных случаев на производстве, уменьшения количества профессиональных заболеваний и ранений.

В рамках охраны труда проводятся следующие мероприятия:

- обеспечение безопасного и здорового рабочего места - важно обеспечить работникам комфортные и безопасные условия труда, которые не нанесут им вреда и не приведут к различным заболеваниям;
- анализ опасностей и оценка рисков - на предприятии должен быть проведен анализ всех возможных опасностей при работе и оценка рисков, связанных с этими опасностями;
- инструктаж и обучение персонала - работники должны быть обучены правилам охраны труда, а также получить инструктаж по использованию средств индивидуальной защиты и других устройств, которые используются для обеспечения безопасности на рабочем месте;
- организация медицинского наблюдения - работники, которые находятся во вредных условиях труда или производят опасные работы, должны периодически проходить медицинский осмотр для выявления и предотвращения возможных заболеваний.

Противопожарная безопасность - это система мероприятий, направленных на предотвращение пожаров и минимизацию последствий, если они все же возникнут. Включает в себя следующие мероприятия:

- создание системы пожарной безопасности - необходимо обеспечить доступность средств пожаротушения, установить автоматические извещатели, устройства автоматического отключения электроэнергии в случае возникновения пожара и др;

- обучение и тренировка персонала - необходимо проводить инструктажи и тренировки по правилам эвакуации, использованию средств пожаротушения, а также знакомство с планами эвакуации;
- проверка на соответствие - проводить регулярные проверки объектов на соответствие требованиям пожарной безопасности и принимать меры к устранению выявленных нарушений;
- система оповещения - необходимо обеспечить эффективную систему оповещения на случай возникновения пожара.

Таким образом, охрана труда и противопожарная безопасность являются важными элементами обеспечения безопасности на рабочем месте. Рациональное использование средств индивидуальной и коллективной защиты, а также выполнение всех необходимых требований и рекомендаций по охране труда и противопожарной безопасности помогут предотвратить риски и обеспечить безопасность персонала.

Заключение

Первым этапом проекта была постановка задачи и анализ требований к системе. В результате были определены функциональные требования, такие как возможность создания и редактирования заявок, отслеживание статуса заявки. Также были выделены нефункциональные требования, включая надежность, безопасность, удобство использования.

Далее была проведена работа по проектированию системы. Была выбрана архитектура системы, определены ее компоненты и интерфейсы между ними. Был разработан дизайн пользовательского интерфейса, который обеспечивает удобство использования системы.

После этого был выполнен кодинг системы. В процессе разработки были использованы современные технологии и инструменты, такие как базы данных и фреймворки для веб-разработки.

Далее система была протестирована. Было проведено функциональное тестирование, которое позволило выявить и устранить ошибки и недостатки системы.

В результате выполненных работ была разработана система для работы с заявками о неисправностях оборудования колледжа. Система обладает рядом преимуществ по сравнению с аналогами, таких как простота использования, автоматизация процесса поддержки оборудования, что позволяет значительно увеличить эффективность и качество работы технической поддержки колледжа.

Также были достигнуты следующие цели:

- улучшение качества обслуживания сотрудников колледжа по вопросам неисправностей оборудования;
- сокращение времени на обработку заявок и ускорение их исполнения;
- оптимизация процесса управления ресурсами и повышение эффективности работы персонала технической поддержки.

Таким образом, можно заключить, что выполненный дипломный проект является успешным и полностью соответствует заданным требованиям и целям.

Список литературы

1. Бородина В.В. Малое и среднее предпринимательство России: бухгалтерский учет и отчетность: Учебное пособие. М.: – 2013.

2. Троелсен Э. "Язык программирования C# 7 и платформы. NET и NET Core" – Санкт-Петербург: Диалектика/Вильямс, 2018. - 1328 с.

3. Прайс М. "C# 7 и.NET Core. Кроссплатформенная разработка для профессионалов"- Санкт-Петербург: Издательский Дом ПИТЕР,2019 -640 с.

4. Васильев А.Н. "Программирование на C# для начинающих. Особенности языка" – Санкт-Петербург: Бомбора 2019. - 528 с.

5. Маслова Т.С. Бухгалтерский учет в государственных (муниципальных) учреждениях: учеб. пособие. М.: Магистр, ИНФРА-М, 2016.

Приложения

Приложение А Схема алгоритма

Рис А1 - Схема алгоритма авторизации

Приложение Б Текст программы

```
public login(): void {
  this.dataIsLoading = true
  this.authService.authorize(this.userLogin.value, this.userPassword.value)
    .subscribe({
      next: (data) => {
        if (this.authService.getRole() == 'admin' || this.authService.getRole() == 'laborant') {
          this.router.navigate(['/requests'])
        }
        else {
          this.router.navigate(['/submittedRequests'])
        }
      },
      error: (error) => (
        console.log(error.error),
        this.snackBar.open('Данные введены неверно', 'Ок', { duration: 5000, panelClass: ['classicSnackBar'] }), this.dataIsLoading = false),
    });
}
```

Рис Б1 - Метод логирования пользователя

```
private filterPlaces(value: any): Place[] {
  let filterValue = '';
  try {
    filterValue = value.toLowerCase();
  }
  catch {
    if(value.description == null) filterValue = ('Кабинет №' + value.name).toLowerCase();
    else filterValue = (value.description + ' №' + value.name).toLowerCase();
  }
  if(value.description == null) return this.places.filter(place => ('Кабинет №' + place.name!).toLowerCase().includes(filterValue));
  return this.places.filter(place => (place.name!).toLowerCase().includes(filterValue));
}
```

Рис Б2 - Метод фильтрации кабинетов

```
canActivate(
  route: ActivatedRouteSnapshot,
  state: RouterStateSnapshot): Observable<boolean | UrlTree> | Promise<boolean | UrlTree> | boolean | UrlTree {
  if (this.authService.isLoggedIn()) {
    let availableRoles: string[] = route.data['role']
    let currentRole: string = this.authService.getRole();
    return this.checkUserAccess(availableRoles,currentRole);
  }
  this.router.navigate(['/auth']);
  return false;
}
```

Рис Б3 - Гвард CanActive

```
public GetMyAll(): Observable<Request[]>{
  return this.http.get<Request[]>(this.apiUrl + '/GetMyAll', {headers: this.headers});
}

public GetActiveAll(): Observable<Request[]>{
  return this.http.get<Request[]>(this.apiUrl + '/GetActiveAll', {headers: this.headers});
}

public GetMyActiveAll(): Observable<Request[]>{
  return this.http.get<Request[]>(this.apiUrl + '/GetMyActiveAll', {headers: this.headers});
}

public GetMyCompletedAll(): Observable<Request[]>{
  return this.http.get<Request[]>(this.apiUrl + '/GetMyCompletedAll', {headers: this.headers});
}

public GetByImpActiveAll(personId: number): Observable<Request[]>{
  return this.http.get<Request[]>(this.apiUrl + '/GetByImpActiveAll/' + personId, {headers: this.headers});
}
```

Рис Б4 - Http запросы

Приложение В Структура данных

```
export class Person {
  id: number | null;
  name: string | null;
  surname: string | null;
  lastname: string | null;
  postId: number | null
  post: PostOrDepartment | null;
  departmentId: number | null;
  department: PostOrDepartment;
  email: string | null;
  comment: string | null;
  userName: string | null;
  roleId: number | null
  role: Role | null;
  password: string | null;
  passwordSalt: string | null;
  requests: Request[] | null;
  unitRespPeople: UnitRespPerson[] | null;
  workOnRequests: WorkOnRequest[] | null;
```

Рис В1 - Модель класса "Person"