**Министерство науки и высшего образования Российской Федерации**

**федеральное государственное автономное образовательное учреждение высшего образования**

**“НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО”  
(УНИВЕРСИТЕТ ИТМО)**

**ЦЕНТР АВТОРИЗОВАННОГО ОБУЧЕНИЯ ИНФОРМАЦИОННЫМ ТЕХНОЛОГИЯМ**

**ИТОГОВАЯ КВАЛИФИКАЦИОННАЯ РАБОТА**

СИСТЕМА УЧЁТА И УПРАВЛЕНИЯ IP-АДРЕСАМИ СЕТЕВЫХ УСТРОЙСТВ

Автор\_\_\_Волков Даниил Андреевич\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_

(Фамилия Имя Отчество) (Подпись)

**Центр авторизованного обучения информационным технологиям**

Наименование программы «Разработчик профессионально

ориентированных компьютерных технологий»

Руководитель \_Осипов Н.А.\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_

(ФИО, ученая звание, степень) (Подпись)

**К защите допустить**

|  |  |  |
| --- | --- | --- |
| *Заместитель директора ЦАО ИТ,*  *к.т.н.* |  | / *Т.В. Зудилова*/ |

Санкт-Петербург, 2023г.

Обучающийся\_Волков Д.А.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Группа 124/28

(Фамилия, И. О.)

Работа принята «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2023г.

Работа выполнена с оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата защиты «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2023г.

Секретарь ИАК \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Фамилия, И. О.) (подпись)

Листов хранения \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Демонстрационных материалов \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**ОГЛАВЛЕНИЕ**

[Введение 4](#_Toc256011441)

[Цель 6](#_Toc256011442)

[Описание системы 8](#_Toc256011443)

[Функции системы 21](#_Toc256011444)

[Структура данных 29](#_Toc256011445)

[Заключение 37](#_Toc256011446)

[Литература 40](#_Toc256011449)

# Введение

В современном взаимосвязанном мире эффективное управление и поддержание сетевой инфраструктуры является важнейшей задачей для предприятий и организаций. Одним из важнейших аспектов управления сетью является эффективное отслеживание и систематизация IP-адресов сетевых устройств. В данном дипломном проекте представлено приложение, предназначенное для решения этой задачи и представляющее собой решение для учёта и хранения IP-адресов сетевых устройств.

С ростом сложности и масштабов современных сетевых инфраструктур потребность в надёжном и удобном приложении для управления IP-адресами становится все более значимой. Точный учёт и правильное хранение IP-адресов крайне важны для эффективного поиска неисправностей, планирования сети и распределения ресурсов. Разработка специализированного приложения для этих целей позволит сетевым администраторам и ИТ-специалистам оптимизировать свою работу, снизить количество ошибок и обеспечить стабильность и безопасность своих сетей.

Задача состоит в проектировании, разработке и оценке приложения, позволяющего пользователям создавать, хранить, редактировать и удалять IP-адреса сетевых устройств. Приложение будет предоставлять дополнительные возможности, такие как привязка IP-адресов к почтовым адресам и моделям оборудования, обеспечение уникальности сетевых адресов, реализация иерархической структуры ядра сети, агрегации и доступа. Кроме того, приложение будет поддерживать авторизацию пользователей с различными уровнями доступа, что позволит обеспечить безопасное управление и контроль информации, касающейся IP-адресов.

Для реализации данного приложения были выбраны две программные технологии: C# REST API для серверной части и WPF для клиентской части. Выбор C# REST API обусловлен тем фактом, что данная технология представляет надёжный и масштабируемый фреймворк на стороне сервера для обработки API-запросов, хранения данных и бизнес-логики. Он обеспечивает гибкость, безопасность и простоту интеграции с различными компонентами. С другой стороны, WPF (Windows Presentation Foundation) предусматривает богатый и интерактивный пользовательский интерфейс для клиентской части, позволяя пользователям интуитивно взаимодействовать с приложением и эффективно выполнять различные задачи.

Благодаря использованию этих технологий приложение может предложить надёжное, масштабируемое и удобное решение для учёта и хранения IP-адресов сетевых устройств. Сочетание C# REST API и WPF обеспечивает бесперебойную связь между клиентом и сервером, эффективную работу с данными и визуально привлекательный пользовательский интерфейс.

В заключение следует отметить, что целью данного дипломного проекта является разработка приложения, решающего задачи управления IP-адресами в сетевых инфраструктурах. Благодаря расширенным возможностям, иерархической организации, авторизации пользователей, а также использованию технологий C# REST API и WPF приложение предлагает комплексное решение для упрощения управления IP-адресами и повышения общей эффективности и безопасности сети.

# Цель

Цель данной итоговой квалификационной работы заключается в разработке комплексного приложения учёта и хранения IP-адресов сетевых устройств.

В процессе достижения поставленной цели необходимо решить следующие задачи:

1. Разработка приложения для учёта и хранения IP-адресов, что предполагает создание специализированного приложения для учёта и хранения IP-адресов сетевых устройств, в процессе разработки которого будет использован C# REST API для серверной части и WPF для клиентской части, как было обосновано ранее. Приложение предназначено для создания, хранения, редактирования и удаления IP-адресов, а также для предоставления дополнительных возможностей, таких как привязка IP-адресов к почтовым адресам и моделям оборудования. Кроме того, будет обеспечена уникальность сетевых адресов и реализована иерархическая структура ядра сети, агрегации и доступа. В целях гарантии безопасности управления информацией, касающейся IP-адресов, приложение будет поддерживать авторизацию пользователей с различными уровнями доступа (просмотр/редактирование).

1. Оценка функциональности и эффективности приложения. Выполнение данной задачи подразумевает, что после разработки приложения будет проведена оценка его функциональности и эффективности в решении поставленных задач. В процессе оценки будут проведены тесты, симуляции для оценки производительности, удобства использования и общего влияния приложения на управление IP-адресами. Результаты этой оценки позволят понять сильные и слабые стороны приложения и определить возможные пути его улучшения или усовершенствования в будущем.

В конечном итоге, данный проект призван внести вклад в область управления сетями, обеспечив практичное и эффективное решение для учёта и хранения IP-адресов. Проведенные исследования, разработки и оценки предоставят сетевым администраторам и ИТ-специалистам возможность получить ценные сведения, которые позволят им усовершенствовать процессы управления IP-адресами и повысить общую эффективность и безопасность своих сетевых инфраструктур.

# Описание системы

1 Обоснование технических и программных средств.

С целью создания системы управления сетевой инфраструктурой предполагается использование следующих технических и программных средств:

1.1 Серверная часть

Язык программирования C# - надежный и широко распространенный язык с отличной поддержкой разработки на стороне сервера, что делает его идеальным выбором для создания REST API.

Фреймворк REST API: C# Rest API может быть реализован с использованием ASP.NET Core, зрелого и многофункционального фреймворка для построения современных веб-приложений и API.

Доступ к базам данных: Entity Framework Core - мощный фреймворк Object-Relational Mapping (ORM), который упрощает взаимодействие с базами данных и обеспечивает бесшовную интеграцию с ними.

1.2 Клиентская часть

Язык программирования C# также представляет собой универсальный язык, подходящий для разработки на стороне клиента. Он обеспечивает согласованность с языком серверной части, что упрощает сопровождение кода.

Пользовательский интерфейс: C# WPF (Windows Presentation Foundation) - надежный фреймворк для создания настольных приложений с богатыми и интерактивными пользовательскими интерфейсами.

Связь с сервером: C# WPF может использовать JSON-запросы для взаимодействия с REST API на сервере и эффективного получения или обновления данных.

2. Состав и структура системы.

Система управления сетевой инфраструктурой состоит из двух основных частей: серверной и клиентской.

2.1 Серверная часть представляет собой REST API с использованием C# и ASP.NET Core, которое обрабатывает HTTP-запросы от клиентского приложения и предоставляет соответствующие ответы; использует Entity Framework Core для доступа к данным и манипулирования ими; реализует различные конечные точки API для выполнения CRUD-операций (Create, Read, Update, Delete) над коммутаторами, почтовыми адресами и типами оборудования.

2.2 Клиентская часть включает в себя C# WPF-приложение, которое

использует C# для разработки логики и пользовательского интерфейса на стороне клиента и взаимодействует с REST API на сервере для получения данных и выполнения операций.

Таким образом обеспечивается удобный и отзывчивый интерфейс для поиска коммутаторов, просмотра подробной информации, редактирования и добавления данных.

3. Руководство пользователя. Действия пользователя программы.

3.1 Вход в систему (рисунок 1)

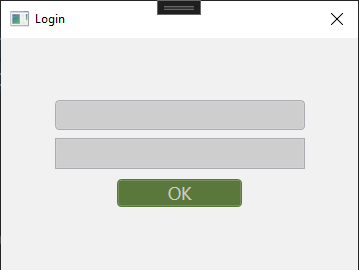


Рисунок 1. Авторизация пользователя

* Введите имя пользователя и пароль на экране входа в систему.
* Для авторизации нажмите кнопку "OK".

3.2 Критерии поиска (рисунок 2)

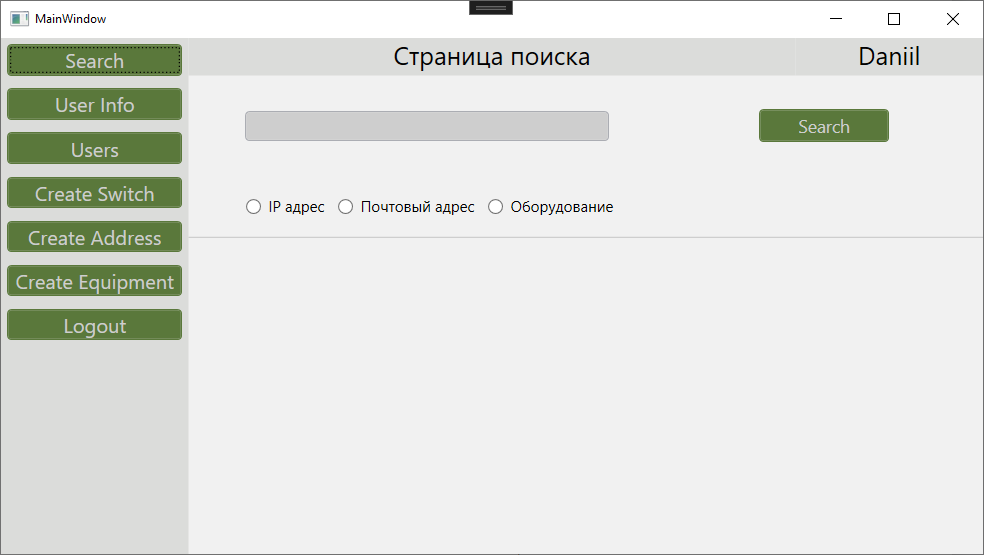


Рисунок 2. Поиск коммутатора.

* Выберите один из критериев поиска: поиск по IP-адресу (“IP адрес”), поиск по почтовому адресу (“Почтовый адрес”) или поиск по типу оборудования (“Оборудование”).
* Введите соответствующие данные в зависимости от выбранного варианта поиска (например, IP-адрес, почтовый адрес, тип оборудования).
* Нажмите кнопку "Search".
* Приложение отобразит список коммутаторов, соответствующих критериям поиска.

3.3 Просмотр подробной информации (рисунки 3,4,5)

* Щелкните на ссылку “Details” из результатов поиска, чтобы просмотреть его подробную информацию (рисунок 3).

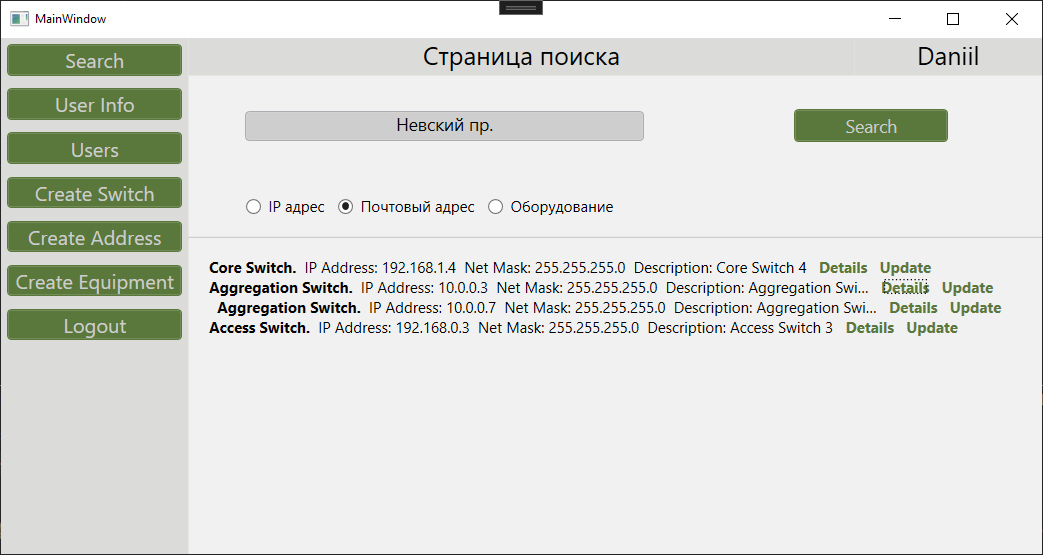


Рисунок 3. Результат поиска по почтовому адресу.

* Приложение отобразит исчерпывающую информацию о выбранном коммутаторе, такую как IP-адрес, тип оборудования, производитель и т.д., информацию о портах (рисунок 4), где возможно редактировать задействованный порт, добавить описание для нового порта, например, номер порта, описание, статус (рисунок 5).

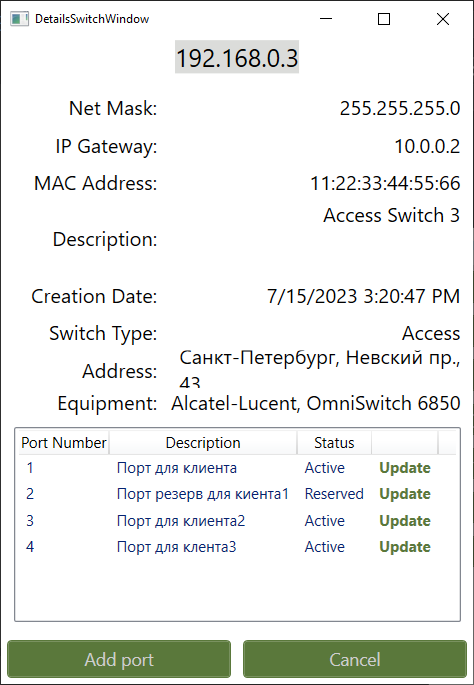


Рисунок 4. Детальная информация о коммутаторе

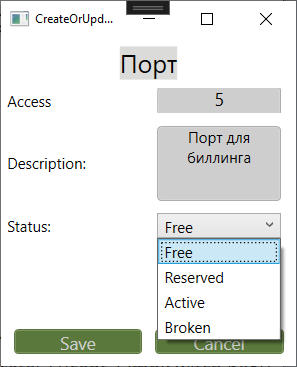


Рисунок 5. Добавления нового порта коммутатора

3.4 Редактирование коммутатора (рисунок 6)

* Просмотрите подробную информацию о коммутаторе.
* Для изменения сведений о коммутаторе нажмите кнопку "Update".

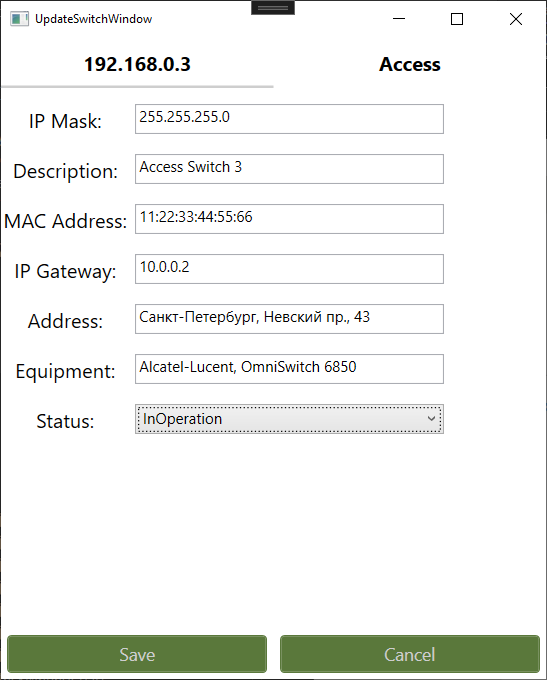


Рисунок 6. Редактирование данных коммутатора

* Внесите необходимые изменения в форму редактирования.
* Адрес вводится через запятую в формате: Город, улица, дом, если такого адреса не существует в базе данных он добавится автоматически.
* Оборудование вводится через запятую в формате: Производитель, модель. если такого оборудования не существует в базе данных он добавится автоматически.
* При добавлении/изменении шлюза, если вышестоящего коммутатора с таким IP адресом не существует в базе данных, будет ошибка (остальные данные добавятся) необходимо создать новый вышестоящий коммутатор через форму "Create Switch"
* Нажмите кнопку "Save", чтобы обновить информацию о коммутаторе.

3.5 Добавление нового коммутатора (рисунок 7)

* Нажмите кнопку "Create Switch" на приборной панели (рисунок 7).

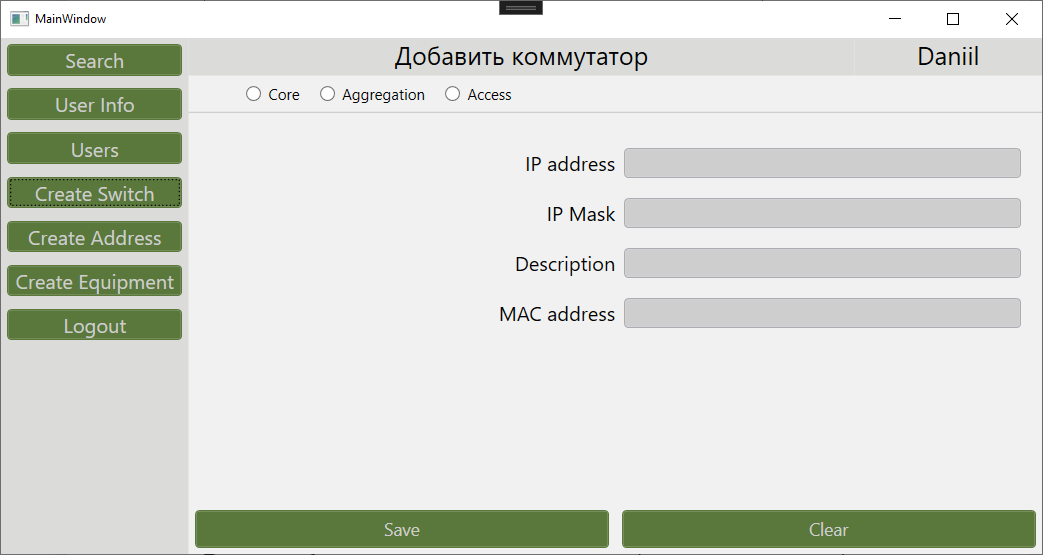


Рисунок 7. Добавление нового коммутатора

* Заполните необходимые данные в форме добавления нового коммутатора.
* При попытке добавить коммутатор с уже существующим IP адресом в базе данных возникнет ошибка.
* Нажмите кнопку "Save", чтобы сохранить новый коммутатор.

3.6 Добавление нового почтового адреса (рисунок 8)

* Нажмите кнопку "Create Address" на приборной панели.

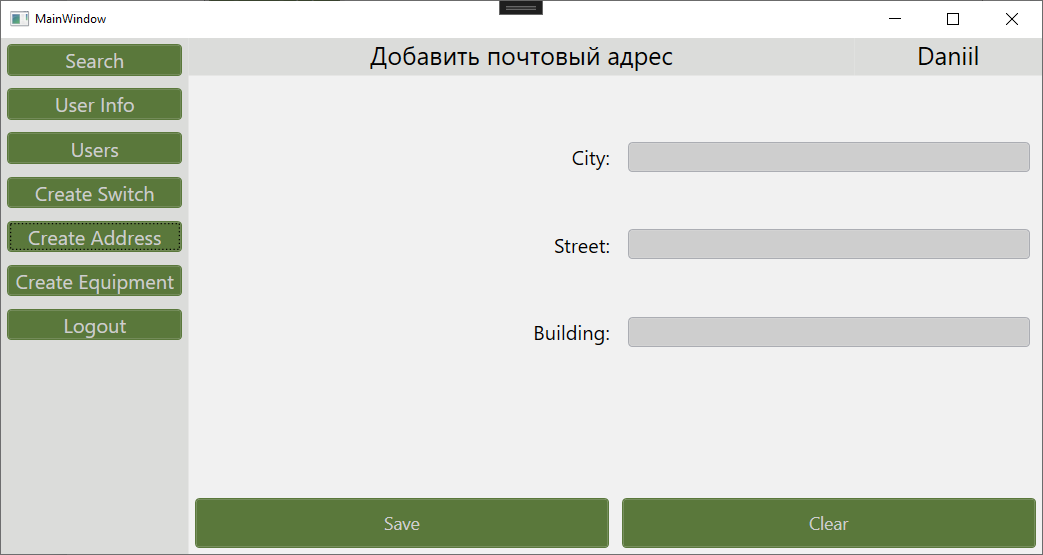


Рисунок 8. Добавление нового постового адреса

* Введите в форму данные о почтовом адресе (город, улица, здание).
* При попытке добавить уже существующий адрес в базе данных, возникнет ошибка.
* Нажмите кнопку "Save", чтобы сохранить новый почтовый адрес.

3.7 Добавление нового оборудования (рисунок 9)

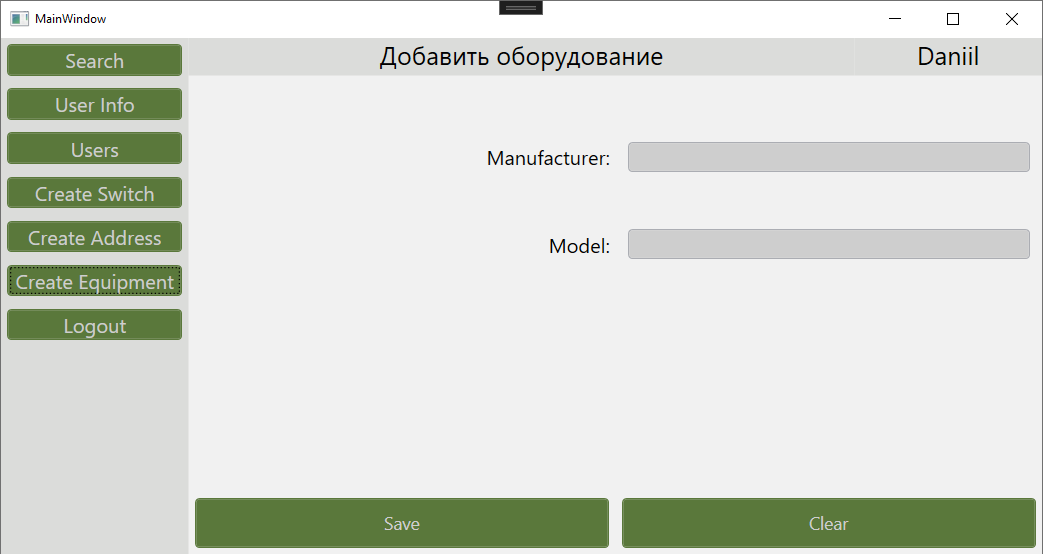


Рисунок 9. Добавление производителя, модели коммутатора

* Нажмите кнопку "Create Equipment" на приборной панели (рисунок 9).
* Введите данные о новом типе оборудования.
* Нажмите кнопку "Save", чтобы сохранить новое оборудование.
* При попытке добавить уже существующее оборудование в базе данных, возникнет ошибка.

3.8 Работа с пользователями (рисунок 10)

* Нажмите кнопку "Users" на приборной панели (рисунок 10).
* Введите логин (email) пользователя

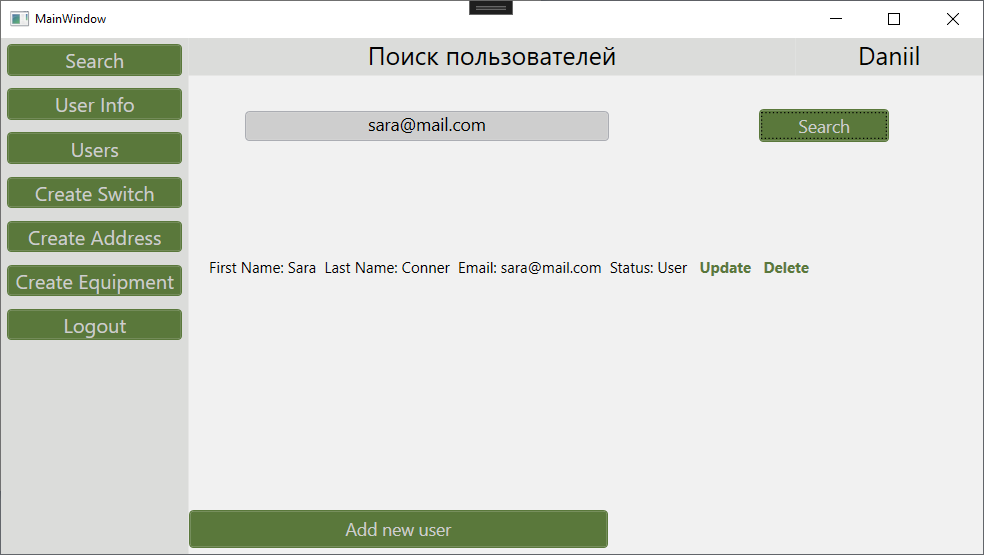


Рисунок 10. Поиск пользователя

* Для создания нового пользователя нажмите кнопку “Add new user”
* Для удаления пользователя нажмите “Delete”
* Для обновления пользователя нажмите “Update” (рисунок 10)

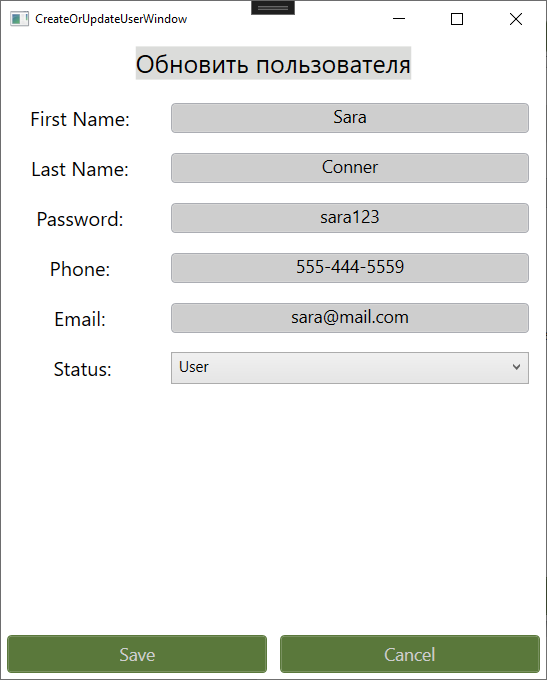


Рисунок 11. Обновление пользователя

* Внесите необходимые изменения в форму редактирования.
* Нажмите кнопку "Save", чтобы обновить информацию о пользователе (рисунок 11).

3.9 Информация о текущем пользователе (рисунок 12)

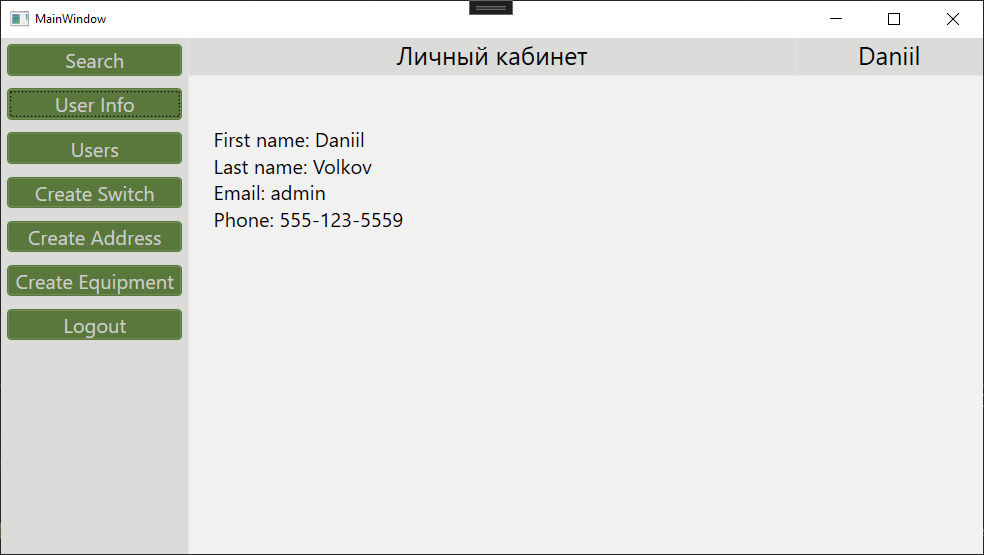


Рисунок 12. Информация о текущем пользователе.

4. Практическая значимость работы:

Система управления сетевой инфраструктурой, построенная с использованием C# Rest API, Entity Framework Core и C# WPF, имеет практическое значение для управления и организации сложных сетевых инфраструктур, предоставляя следующие возможности для пользователя:

* Эффективный доступ к данным. C# Rest API с Entity Framework Core обеспечивает беспрепятственный и эффективный доступ к данным, позволяя пользователям получать и изменять информацию о сетевой инфраструктуре.
* Опыт работы с настольными приложениями. C# WPF предоставляет богатый пользовательский интерфейс с интерактивными элементами, обеспечивая привычный и эффективный опыт для пользователей настольных систем.
* Обновления в реальном времени. Приложение взаимодействует с сервером в режиме реального времени, что позволяет пользователям получать самую свежую информацию о сети.
* Масштабируемость. Модульная и хорошо структурированная архитектура обеспечивает легкую масштабируемость, что позволяет использовать ее в растущих сетевых инфраструктурах.
* Безопасность данных. Механизмы аутентификации и авторизации пользователей обеспечивают доступ и модификацию критически важной сетевой информации только авторизованным сотрудникам.
* Оптимизированное управление. Удобный интерфейс и интуитивно понятные действия упрощают задачи управления сетевой инфраструктурой, повышая производительность и эффективность.

В целом система управления сетевой инфраструктурой, разработанная на базе C# Rest API и C# WPF, представляет собой надежное и практичное решение для эффективного управления и обслуживания сложных сетевых инфраструктур, способствующее повышению производительности, безопасности и оптимизации работы сети.

# Функции системы

1. Поиск коммутаторов

1.1. Search

1.1.1. Отображение формы поиска

1.1.1.1. SearchPage

Описание: данная функция отображает удобную форму, в которую пользователь может ввести IP-адрес для поиска коммутаторов в сетевой инфраструктуре.

Входные данные: нет

Выходные данные: форма поиска с полем ввода IP-адреса.

1.1.2. Выполнение поиска по IP-адресу

1.1.2.1. IPSearch (IpAddress)

Описание: данная функция обрабатывает введенный пользователем IP-адрес и запрашивает базу данных для поиска коммутаторов с совпадающими IP-адресами.

Входные данные: IP-адрес в виде строки.

Выходные данные: список коммутаторов, соответствующих указанному IP-адресу.

2. Поиск по почтовому адресу

2.1. SearchPage

Описание: данная функция отображает форму, в которую пользователь может ввести почтовый адрес для поиска коммутаторов в сетевой инфраструктуре.

Входные данные: нет

Выходные данные: форма поиска с полями для ввода данных о почтовом адресе.

2.1.1. PostalAddressSearch(postalAddress)

Описание: данная функция обрабатывает указанный пользователем почтовый адрес и запрашивает базу данных для поиска коммутаторов, связанных с указанным почтовым адресом.

Входные данные: данные почтового адреса (город, улица, здание) в виде строк.

Выходные данные: список коммутаторов, связанных с указанным почтовым адресом.

3. Поиск по типу оборудования

3.1. SearchPage

Описание: данная функция отображает форму, в которой пользователь может выбрать тип оборудования для поиска коммутаторов в сетевой инфраструктуре.

Входные данные: нет

Выходные данные: список коммутаторов, связанных с указанным типом оборудования.

3.1.1. EquipmentManufacturerSearch(equipment)

Описание: данная функция обрабатывает выбранный пользователем тип оборудования и запрашивает базу данных для поиска выключателей указанного типа.

Входные данные: тип оборудования, выбранный пользователем.

Выходные данные: список коммутаторов выбранного типа оборудования.

4. Просмотр подробной информации

4.1. Отображение подробной информации о коммутаторе

4.1.1. SwitchDetails(switchId)

Описание: данная функция извлекает из базы данных подробную информацию о коммутаторе с заданным идентификатором и отображает ее пользователю.

Входные данные: ID коммутатора в виде целого числа.

Выходные данные: подробная информация о коммутаторе, такая как IP-адрес, тип оборудования, производитель и т.д.

5. Редактирование коммутатора

5.1. Отображение формы редактирования

5.1.1. UpdateSwitchWindow

Описание: эта функция извлекает из базы данных текущую информацию о коммутаторе с заданным идентификатором и отображает форму редактирования, предварительно заполненную существующими данными, позволяя пользователям изменять информацию о коммутаторе.

Входные данные: ID коммутатора в виде целого числа.

Выход: форма редактирования с предварительно заполненными полями, содержащими текущую информацию о коммутаторе.

6. Сохранить правки

6.1. save(switchId, updatedDetails)

Описание: данная функция сохраняет отредактированную информацию о коммутаторе с заданным идентификатором в базу данных после того, как пользователь отправит форму редактирования.

Входные данные: ID коммутатора в виде целого числа, updatedDetails, содержащий измененную информацию о коммутаторе.

Выход: нет

7 Добавление нового коммутатора

7.1. Отображение формы нового коммутатора

7.1.1. CreateSwitchPage

Описание: данная функция отображает форму, в которую пользователь может ввести данные для добавления нового коммутатора в сетевую инфраструктуру.

Входные данные: нет

Выходные данные: форма добавления нового коммутатора с полями ввода данных о коммутаторе.

8. Сохранение нового коммутатора

8.1. saveSwitch(newSwitch)

Описание: эта функция сохраняет в базе данных только что добавленные данные о коммутаторе после того, как пользователь отправит форму нового коммутатора.

Вход: SwitchDetails, содержащий информацию о новом добавленном коммутаторе.

Выход: нет

9. Добавление нового почтового адреса

9.1. Отображение формы добавления нового почтового адреса

9.1.1. PostalAddressPage

Описание: данная функция отображает форму, в которую пользователь может ввести данные для добавления нового почтового адреса в сетевую инфраструктуру.

Входные данные: нет

Выходные данные: форма добавления нового почтового адреса с полями для ввода данных об адресе.

10. Сохранение нового почтового адреса

10.1. postalAddress

Описание: данная функция сохраняет в базе данных данные о новом почтовом адресе после отправки пользователем формы нового почтового адреса.

Вход: PostalAddress, содержащий информацию о новом почтовом адресе.

Выход: нет

11. Добавление нового типа оборудования

11.1. Отображение формы добавления нового типа оборудования

11.1.1. EquipmentPage

Описание: данная функция отображает форму, в которую пользователь может ввести данные для добавления нового типа оборудования в сетевую инфраструктуру.

Входные данные: нет

Выходные данные: форма добавления нового типа оборудования с полями ввода данных о типе.

12. Сохранение нового типа оборудования

12.1. SaveEquipment(equipment)

Описание: данная функция сохраняет в базе данных данные о новом типе оборудования после того, как пользователь отправит форму нового типа оборудования.

Вход: EquipmentDetails, содержащая информацию о вновь добавленном типе оборудования.

Выход: нет .

13. Аутентификация пользователя

13.1. Вход в систему

13.1.1. login(username, password)

Описание: эта функция проверяет введенные имя пользователя и пароль по базе данных на предмет успешного входа в систему.

Входные данные: имя пользователя и пароль в виде строк.

Выходные данные: Булевский результат, указывающий на успешный вход в систему или неудачу аутентификации.

14. Управление пользователями

14.1. Регистрация пользователей

14.1.1. registerUser(newUserDetails)

Описание: данная функция добавляет в систему нового пользователя с указанными данными о нем (например, имя пользователя, пароль, электронная почта).

Вход: newUserDetails, содержащая информацию о новом пользователе.

Выход: нет

15. Обновление профиля пользователя

15.1. updateUserProfile(userId, updatedProfileDetails)

Описание: эта функция позволяет пользователям обновить информацию о своем профиле, например, имя, электронную почту или пароль.

Входные данные: ID пользователя в виде целого числа, updatedProfileDetails, содержащий измененную информацию о профиле пользователя.

Выходные данные: нет

16. Управление ролями пользователей

16.1. updateUserStatus(userId, status)

Описание: эта функция позволяет администраторам обновить роль пользователя (например, администратор, обычный пользователь).

Входные данные: ID пользователя в виде целого числа, newRole - новая роль, которую необходимо присвоить пользователю.

Выходные данные: нет .

17. Управление доступом

17.1. Проверка авторизации

17.1.1. Authorization(userId, requiredPermission)

Описание: эта функция проверяет, имеет ли вошедший в систему пользователь с заданным идентификатором пользователя требуемое разрешение на выполнение определенного действия.

Входные данные: ID пользователя в виде целого числа, requiredPermission - разрешение, необходимое для выполнения действия.

# Структура данных

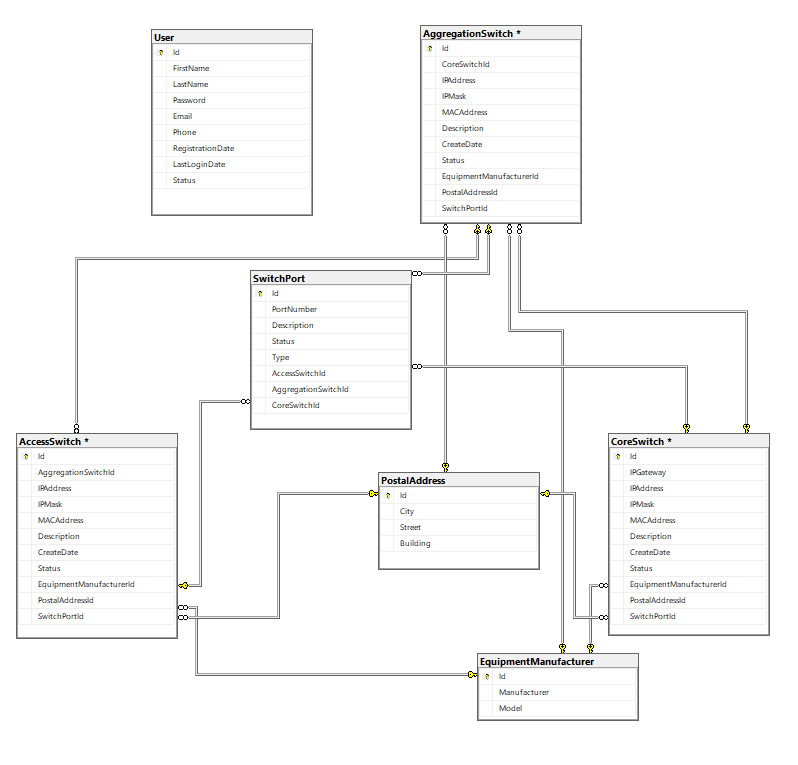


Рисунок 13. Структура Базы данных.

1. Таблицы базы данных

1.1 Таблица EquipmentManufacturer

Структура: Таблица EquipmentManufacturer содержит информацию о различных производителях оборудования в сетевой инфраструктуре. Она состоит из трех столбцов:

* "Id", который служит первичным ключом и однозначно идентифицирует каждого производителя;
* "Manufacturer", в котором хранится название производителя;
* "Model", содержащий название модели оборудования, выпускаемого производителем оборудования.

1.2 Таблица PostalAddress

Структура: В таблице PostalAddress хранятся сведения о почтовых адресах, связанных с различными сетевыми элементами. Она состоит из четырех столбцов:

* "Id", служащий первичным ключом;
* "City", в котором хранится город почтового адреса;
* "Street", содержащий информацию об улице,
* "Building", включающий в себя номер здания.

1.3 Таблица SwitchPort

Структура: Таблица SwitchPort представляет порты коммутатора в сети. Она содержит несколько столбцов:

* "Id" - первичный ключ;
* "PortNumber" - уникальный номер, присвоенный каждому порту коммутатора;
* "Description" - дополнительная информация о порте;
* "Status" - текущее состояние порта, которое может принимать значения: Free, Reserved, Active, Broken;
* "Type" - принадлежность порта к коммутатору (необязательно), может принимать значения: Indeterminate, Core, Aggregation, Access.

Кроме того, имеются три столбца внешних ключей:

* "AccessSwitchId";
* "AggregationSwitchId";
* "CoreSwitchId".

Каждый из них ссылается на конкретный коммутатор (AccessSwitch, AggregationSwitch или CoreSwitch) в зависимости от местоположения порта.

1.4 Таблица User

Структура: Таблица User содержит информацию о пользователях, связанных с сетью. Она включает в себя различные столбцы:

* "Id" - первичный ключ, однозначно идентифицирующий каждого пользователя;
* "FirstName" и "LastName" - имя и фамилия пользователя;
* "Password" - пароль пользователя;
* "Email" - адрес электронной почты пользователя, который, также служит логином для входа в приложение;
* "Phone" - номер телефона пользователя (необязательно);
* "RegistrationDate" - дата и время регистрации пользователя;
* "LastLoginDate" - дата последнего входа пользователя в систему;
* "Status" - текущий статус пользователя, который может принимать значения: Admin, Operator, User.

1.5 Таблица AccessSwitch

Структура: Таблица AccessSwitch представляет коммутаторы доступа в сети. Она состоит из нескольких столбцов:

* "Id" - первичный ключ, однозначно идентифицирующий каждый коммутатор доступа;
* "AggregationSwitchId" - внешний ключ, ссылающийся на таблицу AggregationSwitch для обозначения коммутатора агрегации, выполняющего роль шлюза, к которому подключен данный коммутатор доступа (необязательно);
* "IPAddress" и "IPMask" для хранения IP-адреса и маски подсети коммутатора доступа соответственно;
* "MACAddress" для хранения MAC-адреса коммутатора доступа (необязательно); "Description" для хранения необязательной дополнительной информации о коммутаторе доступа;
* "CreateDate" для отслеживания даты создания коммутатора доступа;
* "Status", который может принимать значения: InOperation, Inaccessible, Dismounted, Scheduled для указания текущего статуса коммутатора доступа; "EquipmentManufacturerId" в качестве внешнего ключа, ссылающегося на таблицу EquipmentManufacturer, для указания производителя коммутатора доступа (опционально);
* "PostalAddressId" в качестве внешнего ключа, ссылающегося на таблицу PostalAddress, для привязки коммутатора доступа к конкретному почтовому адресу (опционально);
* "SwitchPortId" в качестве внешнего ключа, ссылающегося на таблицу SwitchPort, для привязки портов к коммутатору доступа (опционально).

1.6 Таблица AggregationSwitch

Структура: Таблица AggregationSwitch представляет коммутаторы агрегации в сети. Она имеет структуру, аналогичную таблице AccessSwitch, включая такие столбцы, ка "Id", "CoreSwitchId", "IPAddress", "IPMask", "MACAddress", "Description", "CreateDate", "Status", "EquipmentManufacturerId", "PostalAddressId1" и "SwitchPortId". Параметр "CoreSwitchId" является внешним ключом, связывающим таблицу CoreSwitch с коммутатором ядра сети (шлюзом), к которому подключается данный коммутатор агрегации (опционально).

1.7 Таблица CoreSwitch

Структура: Таблица CoreSwitch содержит сведения о коммутаторах ядра сети. Она имеет схожую структуру с таблицами AccessSwitch и AggregationSwitch, включая такие столбцы, как "Id", "IPGateway", "IPAddress", "IPMask", "MACAddress", "Description", "CreateDate", "Status", "EquipmentManufacturerId", "PostalAddressId" и "SwitchPortId". В столбце "IPGateway" хранится информация об IP-шлюзе для коммутатора ядра сети.

2. Табличные отношения:

* Таблицы "AccessSwitch" и "AggregationSwitch" связаны друг с другом через колонку внешнего ключа "AggregationSwitchId" в таблице "AccessSwitch", которая ссылается на колонку Id в таблице "AggregationSwitch". Эта связь означает, что коммутатор доступа может быть подключен к коммутатору агрегации, образуя иерархическую структуру сети.
* Таблицы "AggregationSwitch" и "CoreSwitch" связаны между собой через колонку внешнего ключа "CoreSwitchId" в таблице AggregationSwitch, которая ссылается на колонку Id в таблице CoreSwitch. Эта связь указывает на то, что агрегационный коммутатор может быть подключен к основному коммутатору, что позволяет реализовать иерархическую архитектуру сети.
* Таблица "EquipmentManufacturer" связана с таблицами "AccessSwitch", "AggregationSwitch" и "CoreSwitch" через соответствующие столбцы внешних ключей "EquipmentManufacturerId". Данные связи указывают на то, что каждый тип коммутатора (доступ, агрегация, ядро) может быть связан с определенным производителем оборудования.
* Таблица "PostalAddress" связана с таблицами "AccessSwitch", "AggregationSwitch" и "CoreSwitch" через колонку внешнего ключа "PostalAddressId" в каждой таблице коммутаторов. Эти отношения позволяют точно связать каждый коммутатор с конкретным почтовым адресом. Каждый коммутатор имеет уникальный почтовый адрес, а ссылки на внешние ключи позволяют эффективно находить соответствующую информацию о почтовых адресах при запросах к базе данных для получения сведений о коммутаторах. Такой структурированный подход повышает эффективность управления и организации данных сетевой инфраструктуры, поскольку информация о почтовых адресах может быть легко связана и сохранена для каждого типа коммутаторов.

3. Иерархия базы данных

На основе представленных таблиц и связей между ними можно сделать вывод о иерархиях базы данных. Иерархии баз данных помогают организовать и представить отношения между таблицами в системе баз данных.

* База данных (верхний уровень)

Представляет собой всю систему базы данных, содержащую множество связанных таблиц.

* EquipmentManufacturer (уровень 1)

В данной таблице хранится информация о различных производителях оборудования. Она находится на том же уровне, что и другие первичные таблицы.

* PostalAddress (уровень 1)

Данная таблица содержит информацию о почтовых адресах различных сетевых компонентов. Она находится на том же уровне, что и другие первичные таблицы.

* User (Уровень 1)

В этой таблице хранится информация о пользователях, связанных с сетью.

Она находится на том же уровне, что и другие первичные таблицы.

* SwitchPort (уровень 1)

В данной таблице представлены порты коммутатора и их свойства.

Она находится на том же уровне, что и другие первичные таблицы.

* AccessSwitch (уровень 2) В этой таблице содержится информация о коммутаторах доступа, используемых в сети.

Она связана с таблицами EquipmentManufacturer и PostalAddress. Данная таблица находится на более низком уровне, чем первичные таблицы EquipmentManufacturer, PostalAddress, User и SwitchPort.

* AggregationSwitch (уровень 2)

В данной таблице представлены сведения о коммутаторах агрегации, используемых в сети. Она связана с таблицами EquipmentManufacturer и PostalAddress и находится на более низком уровне, чем первичные таблицы EquipmentManufacturer, PostalAddress, User и SwitchPort.

* CoreSwitch (уровень 2):

В этой таблице представлены коммутаторы ядра сети, используемые в сети. Она связана с таблицами EquipmentManufacturer и PostalAddress и находится на более низком уровне, чем первичные таблицы EquipmentManufacturer, PostalAddress, User и SwitchPort.

В этой иерархии верхний уровень - это вся система баз данных, первый уровень содержит первичные таблицы, в которых хранится информация о производителях, почтовых адресах и портах коммутаторов. Второй уровень содержит таблицы, представляющие различные типы коммутаторов, и эти таблицы связаны с таблицами EquipmentManufacturer и PostalAddress первого уровня.

Иерархия базы данных отражает логические связи между таблицами, где некоторые таблицы имеют прямые связи друг с другом, образуя хорошо организованную структуру для управления данными сетевой инфраструктуры.

# Заключение

В заключение следует отметить, что работа по созданию системы управления сетевой инфраструктурой с использованием C# Rest API, Entity Framework Core и C# WPF позволила успешно решить поставленные задачи. Система эффективно управляет и организует сложную сетевую инфраструктуру, предоставляя пользователям удобный интерфейс для поиска коммутаторов, просмотра детальной информации и выполнения редактирования или дополнения. Использование C# Rest API и Entity Framework Core обеспечивает эффективный доступ к данным и манипулирование ими, а C# WPF - многофункциональность настольного приложения.

1. Трудности и решения

В процессе разработки возникли некоторые трудности, например, с осуществлением процедуры аутентификации пользователей, а также с созданием интуитивно понятного пользовательского интерфейса. Однако эти проблемы были решены за счет использования надежных механизмов аутентификации, предоставляемых C# Rest API, правильной валидации данных и обработки ошибок, а также проведения тщательного тестирования с использованием платформы API тестирования Postman и Unit Tests Microsoft Visual Studio.

2. Дополнительные возможности

На основе проделанной работы можно реализовать ряд дополнительных возможностей с целью увеличения потенциала системы, максимального использования ее ресурсов/преимуществ и повышения удобства ее использования на практике. Таковыми являются, например:

* Создание мобильного приложения. Разработка версии мобильного приложения для системы управления сетевой инфраструктурой расширит возможности системы и позволит пользователям управлять сетевой инфраструктурой "на ходу".
* Онлайн-мониторинг оборудования. Реализация мониторинга сетевого оборудования в режиме реального времени позволит пользователям получать ценную информацию о состоянии коммутаторов, что обеспечит активное обслуживание и решение проблем.
* Привязка почтовых адресов на картах. Интеграция картографических сервисов в приложение предоставит пользователям возможность визуализировать географическое распределение сетевого оборудования, что, в свою очередь, облегчит анализ и планирование на основе местоположения.

3. Перспективы.

Система управления сетевой инфраструктурой имеет многообещающие перспективы в области управления сетями. При дальнейшем развитии и совершенствовании она может стать ценным инструментом для сетевых администраторов и ИТ-специалистов. Модульная архитектура и масштабируемость системы позволяют использовать ее для управления как малыми, так и крупными сетевыми инфраструктурами.

Создание мобильного приложения расширит сферу применения системы, ориентируясь на выездных инженеров, которым необходим оперативный доступ к сетевой информации. Кроме того, значительную ценность системе придаст включение функций онлайн-мониторинга, которые позволят инженерам получать информацию в режиме реального времени для активного управления сетью.

Помимо этого, интеграция картографических сервисов для почтовых адресов обеспечит пространственный контекст сетевой инфраструктуры, позволяя администраторам планировать и оптимизировать размещение оборудования с учетом географического распределения.

В заключение следует отметить, что система управления сетевой инфраструктурой продемонстрировала свою эффективность в организации и управлении сетевыми инфраструктурами. Успешная реализация C# Rest API, Entity Framework Core и C# WPF заложила прочный фундамент для дальнейшего развития и расширения, а последующее исследование предложенных перспектив и постоянное совершенствование приложения может привести к тому, что система станет незаменимым инструментом для эффективного и активного управления сетью.

# Литература

1. Албахари Дж. C# 9.0. Справочник. Полное описание языка / пер. с англ. - СПб.: ООО “Диалектика”, 2021. 1056 с.
2. Осетрова И.С. Разработка баз данных в MS SQL Server 2014. - СПб: Университет ИТМО, 2016. – 114 с.
3. Троелсен Э., Джепикс Ф. Язык программирования C# и платформа .NET. Основные принципы и практики программирования. 10-е изд.; пер. с англ. Ю.Н. Артеменко. - Киев.: “Диалектика”, 2022. - 770 с.
4. Принципы построения REST JSON API [Электронный ресурс]. 2019. URL: <https://habr.com/ru/articles/447322> (Дата обращения 29.07.2023).
5. Руководство разработчика Prism — часть 1. Введение. 2013. [Электронный ресурс]. URL: <https://habr.com/ru/articles/176851> (Дата обращения 30.07.2023).
6. Создание веб-API с помощью ASP.NET Core. 2023. [Электронный ресурс]. URL:<https://learn.microsoft.com/ru-ru/aspnet/core/web-api/?view=aspnetcore-7.0> (Дата обращения 28.07.2023).
7. Техническая документация Microsoft SQL Server. 2023. [Электронный ресурс]. URL:<https://docs.microsoft.com/ru-ru/sql/sql-server/?view=sql-server-2017> (Дата обращения 26.07.2023).