1. ОПИСАНИЕ СИСТЕМЫ

3.1 Обоснование технических и программных средств.

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

3.1.1 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.

3.1.2 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.

3.2. Состав и структура системы.

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

3.2.1 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.

3.2.2 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.

3.3 Руководство пользователя. Действия пользователя программы.

3.3.1 Вход в систему (рисунок 1)

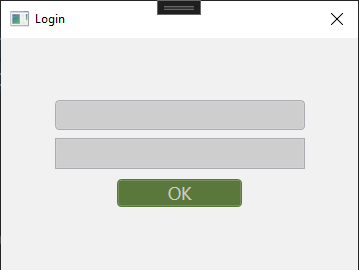


Рисунок 1

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

3.3.2 Критерии поиска (рисунок 2)

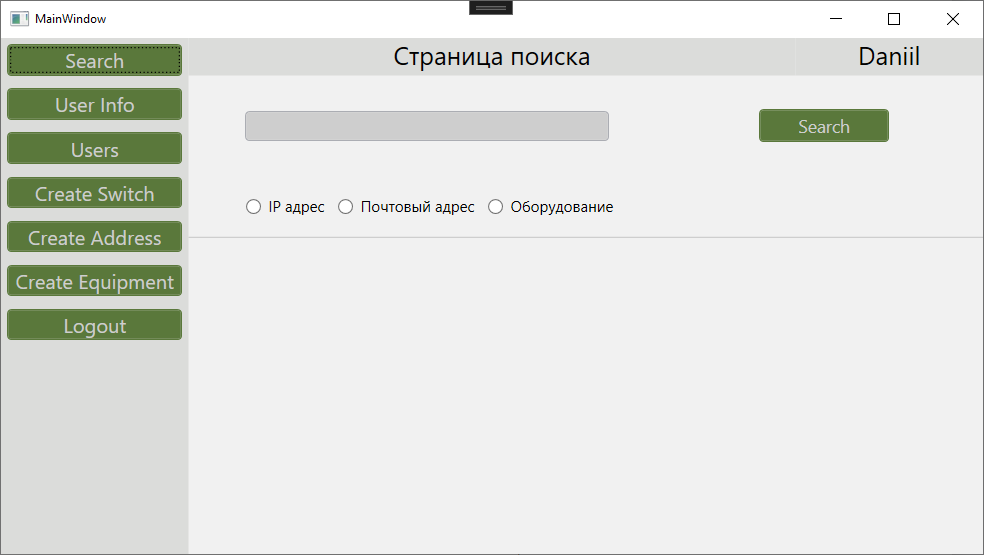


Рисунок 2

* 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.

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

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

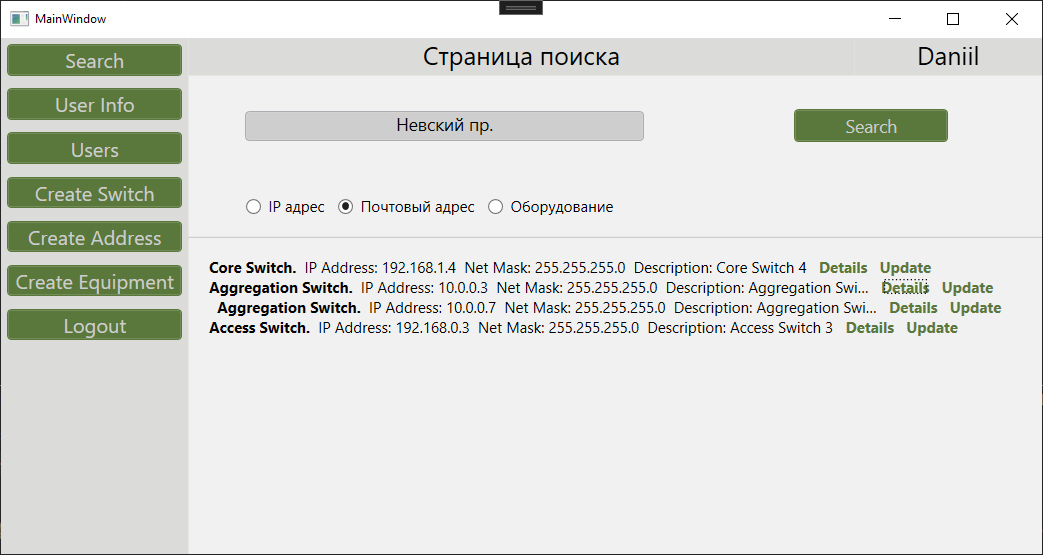


Рисунок 3

* 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).

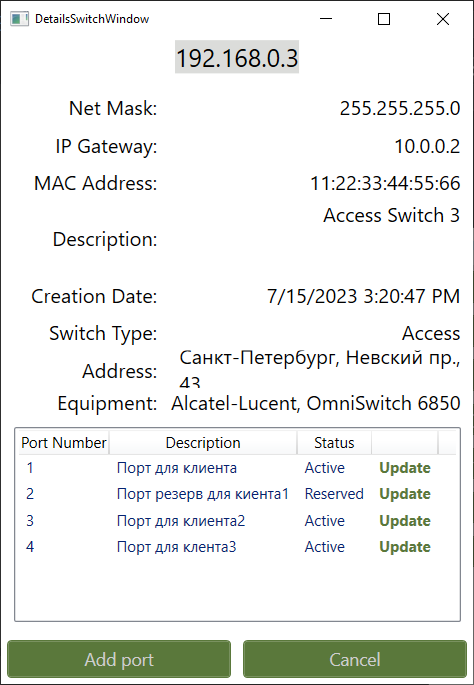


Рисунок 4

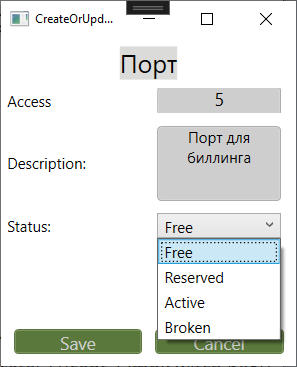


Рисунок 5

3.3.4 Редактирование коммутатора (рисунок 6)

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

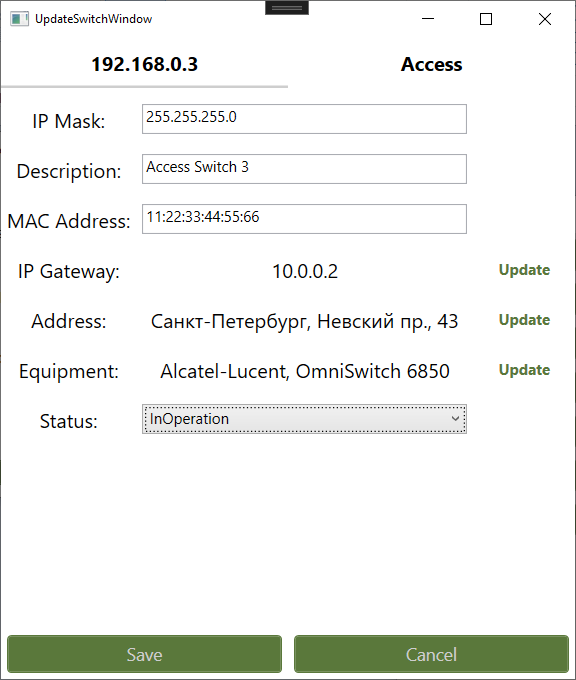


Рисунок 6

* Внесите необходимые изменения в форму редактирования.
* Нажмите кнопку "Save", чтобы обновить информацию о коммутаторе.

3.3.5 Добавление нового коммутатора (рисунок 7)

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

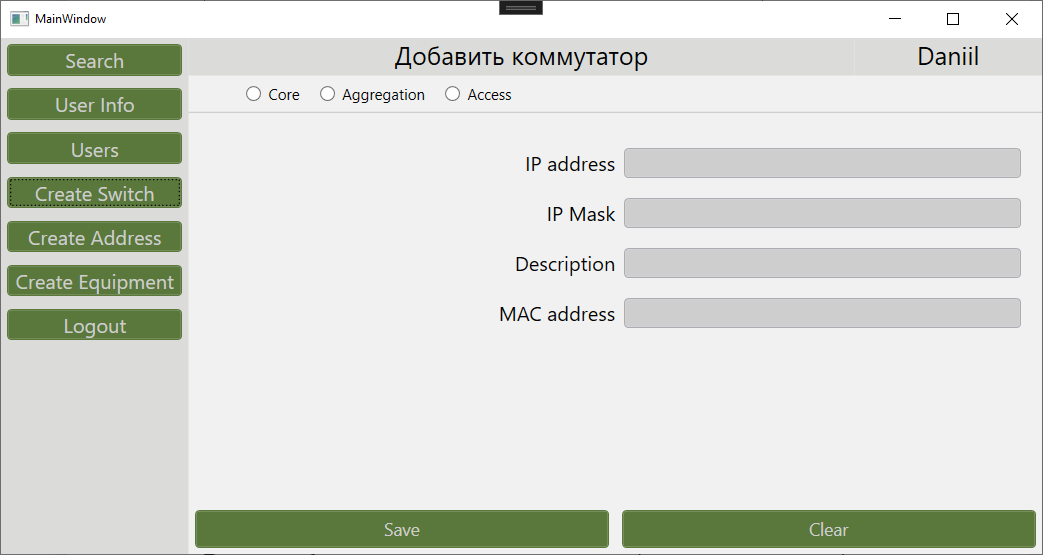


Рисунок 7

* 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.

3.3.6 Добавление нового почтового адреса (рисунок 8)

* Нажмите кнопку "Create Address" на приборной панели.

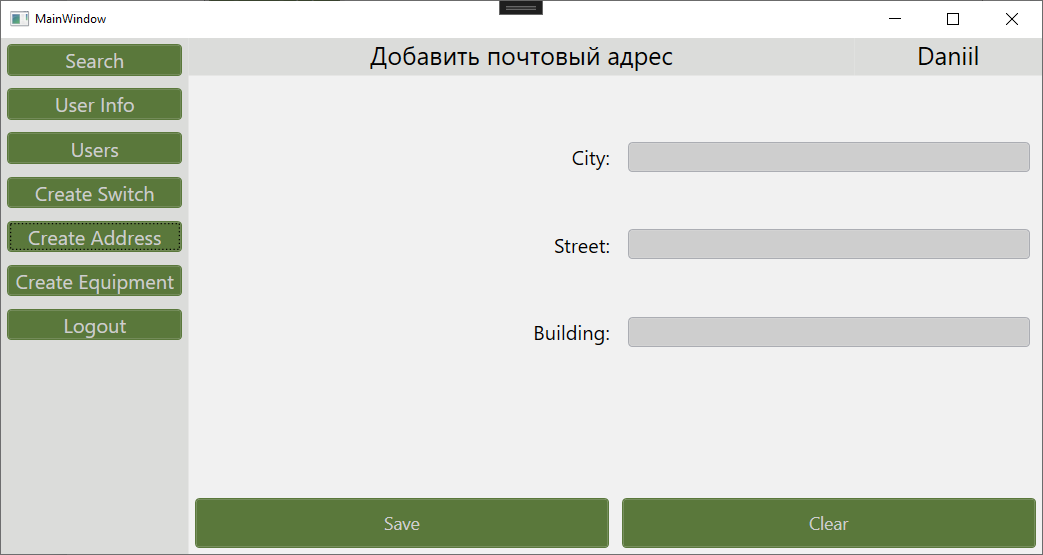


Рисунок 8

* 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.

3.3.7 Добавление нового оборудования (рисунок 9)

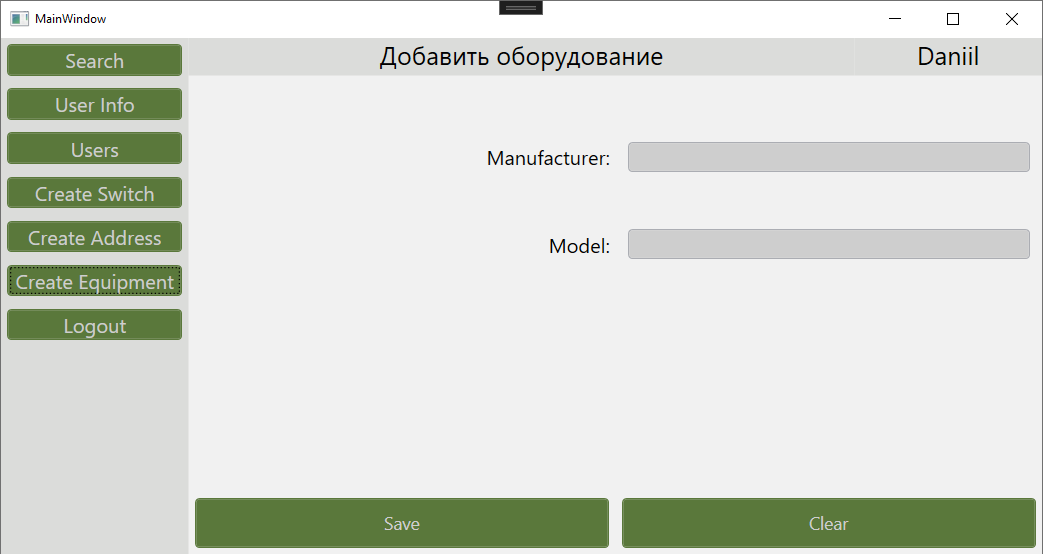


Рисунок 9

* 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.

3.3.8 Работа с пользователями (рисунок 10)

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

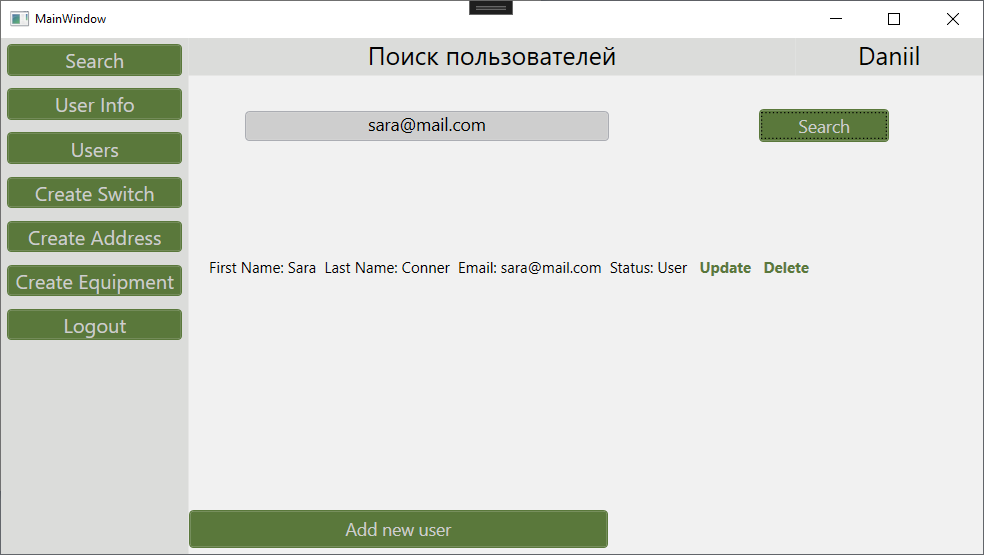


Рисунок 10

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

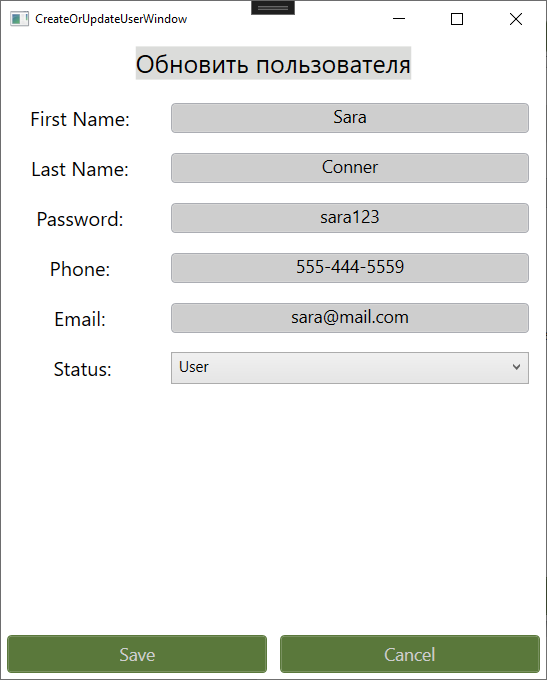


Рисунок 11

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

3.3.9 Информация о текущем пользователе (рисунок 12)

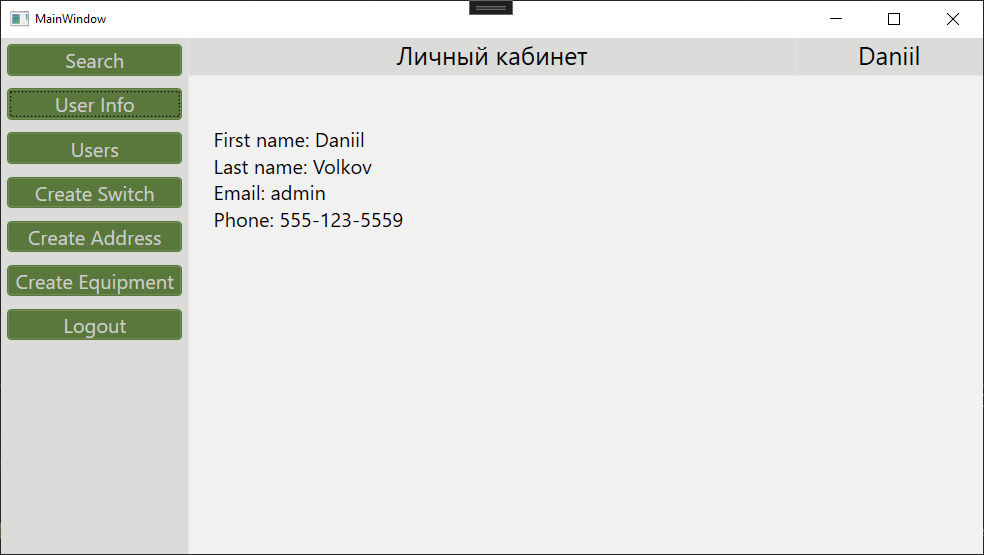


Рисунок 12

3.4 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.