ООО «Программы учета»

# Программный комплекс «LIGHTING»

# ОПИСАНИЕ АРХИТЕКТУРЫ И ФУНКЦИОНАЛЬНЫХ ХАРАКТЕРИСТИК ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Хабаровск 2024г.

## Оглавление

# 1. ОПИСАНИЕ АРХИТЕКТУРЫ «LIGHTING»

## 1.1. Основные решения программно-технической архитектуры «LIGHTING»

а) Многоуровневая архитектура «Клиент-сервер»

Согласно функциональным требованиям к программному продукту требуется обеспечить масштабируемость, централизованность данных и высокую оперативность получения необходимой информации, и ее достоверность. Многоуровневая архитектура отвечает данным требованиям, помимо этого данная архитектура обеспечивает безопасность, хранимой в базе данных информации.

б) Серверная часть

Программный комплекс реализован на Blazor UI-фреймворке для создания интерактивных приложений, которые могут работать как на стороне сервера, так и на стороне клиента, на платформе .NET. В своем развитии фреймворк Blazor испытал большое влияние современных фреймворков для создания клиентских приложений - Angular, React, VueJS. В частности, это проявляется в роли компонентов при построении пользовательского интерфейса. В то же время и на стороне клиента, и на стороне сервера при определении кода в качестве языка программирования применяется C#, вместо JavaScript. Также используется EF Core - фреймворк для работы с базами данных (объектно-реляционного отображение (ORM)).

в) Клиентская часть (Web-интерфейс)

Пользовательский интерфейс реализован в форме Web-интерфейса, за основу которого взята платформа Blazor WASM. Страницы описаны языком разметки Razor-pages в связке с JavaScript Выбранный подход позволяет не привязываться к операционной системе устройства, обеспечивая кросс-платформенность программного продукта. В качестве минимальных

требований для работы с приложением выступают требования используемого браузера пользователя.

г) База данных

Выбор СУБД основывается на поставленных задачах и простоте настройки и сопровождения БД, при возможности выполнения больших и сложных операций. Проанализировав текущий рынок СУБД, было выбрано одно из самых популярных и современных решений – PostgreSQL. Данная технология обладает необходимыми для проекта характеристиками: масштабируемость благодаря своим характеристикам с открытым исходным кодом; кластерные решения для хранения данных обеспечивают свободное расширение; поддерживает собственный SSL, который помогает шифровать обмен данными с сервером; является open-source проектом.

д) Интеграция решений с открытым исходным кодом

Использование в проекте open-source решений позволяет избежать сторонних отчислений, что уменьшает затраты на конечный продукт, но при этом упрощает и ускоряет процесс реализации, модернизации и сопровождение проекта.

Выше перечисленные решения были выбраны на основе требований к реализации проекта.

Проект представлен в виде многоуровневой клиент-серверной архитектуры. Пользователь взаимодействует с клиентской частью приложения, которая представляет собой Web-интерфейс, в котором используются элементы фреймворка ASP.NET Core платформы Blazor Webassembly. Сам интерфейс разработан на основе рекомендаций по User experience (UX), описанным в документации фреймворка, то есть в проекте используются ключевые подходы для разработки интуитивно понятного и разборчивого интерфейса. Структура серверной части позволяет обеспечить

аутентификацию для безопасного доступа к данным, а также оптимизировать процессы по логистики.

### 1.2. Описание используемых программных компонентов, платформы и технологий

а) Платформы

- ASP.NET Core
- Blazor

б) Хранение и обработка данных

- PostgreSQL
- EF Core

в) Развертывание

- Nginx
- Apache Tomcat
- IIS

г) Безопасность

- Microsoft Identity

д) Web-интерфейс

- Blazor WASM
- Razor-pages
- Java Script

е) Инструменты разработки (рекомендуемые)

- Visual Studio 2022 Community/Professional

### 1.3. Описание клиентской и серверной части

а) Одностраничное приложение. Данный подход позволяет осуществлять навигацию по веб-сайту, обновляя содержимое тела веб-документа, без загрузки новых документов и как следствие снижение нагрузки на сервер, повышение производительности. Так же, Blazor WebAssembly позволяет создавать одностраничные интерактивные приложения клиентской стороны, которые запускаются в браузере пользователя и работают с помощью технологии WebAssembly

б) Реализация CRUD-операций с данными в представлениях. Согласно требованиям к программному продукту, пользователи с определенной категорией ролей, должны иметь возможность манипулировать данными. В приложении продумана и реализована логика осуществления операций для удаления, создания, редактирования и просмотра данных при помощи специальных форм заполнения, а так же просмотра истории изменений.

в) Параллельное выполнение задач (например, расчет маршрутов в цепи электроснабжения) без блокировки операций и интерфейса для пользователя.

г) Пагинация, фильтрация и поиск данных. Для обеспечения удобной работы с большими объемами данных, в приложении реализована пагинация данных, поиск и фильтрация по различным параметрам.

д) Валидация введенных данных в формах операций. Для обеспечения безопасности некорректных операций над данными используется трехуровневая валидация: анализ параметров запроса; валидация на уровне модели; валидация на уровне СУБД.

Для разработки клиентской части был выбран фреймворк Blazor WebAssembly, так как он обладает мощным синтаксисом шаблонов, позволяя быстро создавать и модифицировать представления пользовательского интерфейса.

Так же, для построения форм отображения информации, используется

MudBlazor – библиотека Blazor-компонент, распространяющаяся по MIT-лицензии.

В серверной части комплекса используется несколько модулей:

– AspNetCore.Authentication.JwtBearer - компонент ПО промежуточного слоя для проверки подлинности носителя, который добавляется в конвейер

– AspNetCore Authentication – компонент middleware, который сериализует данные пользователя в зашифрованные аутентификационные куки и передает их на сторону клиента. При получении запроса от клиента, в котором содержатся аутентификационные куки, происходит их валидация, десериализация и инициализация свойства User объекта HttpContext.

– Entity Framework Core – модуль для взаимодействия с любыми базами данных.

### 1.4. Безопасность

Для доступа к программному продукту необходимо получить логин и пароль у администраторов проекта, затем пройти авторизацию в самом приложении. В проекте существует система разделения ролей, то есть, каждому зарегистрированному пользователю присваивается роль, например: диспетчер, гость, администратор и т.д. Для каждой роли доступны свои разделы и опции.

Авторизация, аутентификация и защита от распространенных атак обеспечивается платформой AspNetCore Authentication, которая позволяет:

- Проверять подлинность пользователя.

- Реагировать на ситуацию, когда не прошедший проверку подлинности пользователь пытается обратиться к ресурсу, доступ к которому ограничен.

## 2. ОПИСАНИЕ ФУНКЦИОНАЛЬНЫХ ХАРАКТЕРИСТИК

### 2.1. Назначение комплекса

Основным назначением проекта является диспетчеризация сетей 6-10 киловольт.

### 2.2. Задачи и функции

Согласно ТЗ комплекса он выполняет следующие задачи и функции:

а) Ведение базы объектов электросети (питающих центров, трансформаторных подстанций, линий электропередач, опор, участков и т.д.)

б) Графическое представление схемы электросети.

в) Управление схемой электросети (перенаправление мощностей, потоков, включение/выключение объектов электросети).

г) Расчет цепи питания и графическое ее отображение.

д) Ведение журналов переключений, происшествий и т.д.

е) Ведение истории изменений и переключений схемы.

ж) Графическое отображение всей схемы на щит их ЖК панелей, управление схемой с рабочего места диспетчера.

### 2.3. Затрачиваемые ресурсы

Для корректной работы программного комплекса необходимо обеспечить следующие аппаратные требования:

- RAM 32 Гб
- CPU с 4 и более ядрами, с частотой не ниже 1,5 ГГц
- Дисковое пространство 100 Гб
- Swap 4 Гб