**План проекта “Интерактивный конфигуратор Tion”**

**I. Цели и задачи:**

* **Цель:** Предоставить пользователю удобный инструмент для подбора оптимальной системы вентиляции Tion на основе параметров его помещения.
* **Задачи:**
  + Сбор информации о помещении от пользователя.
  + Выбор типа системы вентиляции.
  + Автоматический подбор моделей оборудования Tion.
  + Расчет необходимой производительности системы.
  + Отображение результатов в понятной и наглядной форме.
  + Реализовать систему авторизации с ролями доступа (основной функционал будет доступен неавторизованному пользователю).

**II. Этапы разработки:**

**1. Общая архитектура:**

Приложение будет иметь трехуровневую архитектуру:

* **Frontend (React):** Отвечает за пользовательский интерфейс, взаимодействие с пользователем, отображение данных и отправку запросов к API.
* **Backend (ASP.NET Core Web API):** Принимает запросы от frontend, обрабатывает бизнес-логику, взаимодействует с базой данных и возвращает данные в формате JSON. Фреймворк для работы с бд Entity Framework
* **Database (PostgreSQL):** Хранит данные об оборудовании Tion, типах помещений и других необходимых сущностях.

**2. Технологии:**

* **Frontend:** React, JavaScript, HTML, CSS, (опционально: TypeScript, Redux/Context API для управления состоянием).
* **Backend:** ASP.NET Core Web API, C#, Entity Framework Core, PostgreSQL.
* **ORM:** Entity Framework Core (для взаимодействия с базой данных).
* **Dependency Injection:** В ASP.NET Core используется встроенный контейнер для внедрения зависимостей.

**3. Принцип работы страницы конфигуратора:**

1. **Загрузка приложения:** Пользователь открывает веб-браузер и переходит по адресу приложения. React-приложение загружается в браузер.

**Ввод параметров помещения:** Пользователь заполняет форму RoomParametersForm (площадь, количество людей, назначение помещения). React обрабатывает изменения в форме и сохраняет их в локальном состоянии. Поля ввода:

* Площадь помещения (м2)
* ?Количество людей?
* Назначение помещения: офис, дом, квартира

1. **Выбор типа вентиляции:** Пользователь выбирает тип вентиляции (приточная, приточно-вытяжная, рекуператор) в компоненте VentilationTypeSelector. React обновляет состояние приложения.
2. **Отправка запроса на сервер:** После ввода параметров и выбора типа вентиляции, React-приложение отправляет HTTP-запрос (обычно GET или POST) к ASP.NET Core Web API. Запрос содержит параметры помещения (площадь, люди, назначение) и выбранный тип вентиляции. Этот запрос может быть отправлен при каждом изменении параметров, либо при нажатии кнопки “Подобрать”.
3. **Обработка запроса на сервере:** ASP.NET Core Web API получает запрос. Контроллер API принимает параметры из запроса.
4. **Расчет производительности и подбор оборудования:** Контроллер вызывает сервисный слой, который выполняет:
   * Расчет необходимой производительности вентиляции на основе параметров помещения.
   * Запрос к базе данных через Entity Framework для получения списка оборудования Tion.
   * Фильтрацию оборудования на основе типа вентиляции и производительности.
5. **Возврат данных:** Сервисный слой возвращает отфильтрованный список оборудования контроллеру. Контроллер сериализует список в формат JSON и возвращает его React-приложению.
6. **Отображение результатов:** React-приложение получает JSON-ответ от API. Оно обновляет состояние с полученными данными и отображает результаты в компоненте ResultsDisplay.
7. **Взаимодействие с пользователем:** Пользователь просматривает список оборудования. Может быть реализована возможность добавления в корзину, отправки заявки или просмотра подробной информации об оборудовании (реализуется через дополнительные запросы к API).

**4. Принцип работы страницы “Каталог”:**

В зависимости кто авторизован в приложении:

* Пользователь (клиент) или анонимный (неавторизованный пользователь): предоставляет полный список оборудования компании Tion
* Администратор: имеет доступ к редактированию, удалению и добавлению оборудования

**5. Взаимодействие между frontend и backend:**

* React-приложение отправляет GET-запрос к API endpoint api/Equipment/GetEquipment с параметрами, переданными в query string (например, api/Equipment/GetEquipment?area=30&people=2&roomType=гостиная&ventilationTypeId=1).
* ASP.NET Core Web API возвращает JSON-ответ, содержащий список подходящего оборудования.

**Проектирование БД**

Основные сущности:

1. Device – информация об оборудовании Tion.
2. VentilationType - справочник типов вентиляции (приточная, приточно-вытяжная, рекуператор).
3. RoomType - справочник типов помещений (спальня, гостиная, кухня, офис). //Не уверен на счет необходимости данной сущности

Таблицы:  
Таблица 1 - Device, ключевое поле «ID»

|  |  |  |
| --- | --- | --- |
| Имя поля | Тип данных | Ограничения |
| ID | Счетчик | Номер по порядку уникален |
| Имя | Текстовый | 50 значений, обязательное поле: да |
| Описание | Текстовый | 50 значений, обязательное поле: да |
| Изображение | BYTEA | 50 значений, обязательное поле: да |
| URL | Текстовый | 255 значений, обязательное поле: да |
| Производительность | Дата/время | Обязательное поле: да |
| ID VentilationType | Числовой | Длинное целое, обязательное поле: да |
| Цена | DECIMAL | 10, 2 |

Таблица 2 – VentilationType, ключевое поле «ID»

|  |  |  |
| --- | --- | --- |
| Имя поля | Тип данных | Ограничения |
| ID | Счетчик | Номер по порядку уникален |
| Название | Текстовый | 50 значений, обязательное поле: да |