**Техническое задание на разработку веб-приложения для спа-салона**

**Название проекта:** Spa Management System  
**Версия:** 1.0  
**Дата создания документа:** 15.09.2025  
**Ответственный за документ:** Мухамадиев Э.А.

**1. Введение**

**1.1. Цель документа**

Настоящий документ определяет цели, задачи, требования и порядок разработки веб-приложения для автоматизации процессов спа-салона. ТЗ является основным документом для команды разработки и служит для оценки результатов на всех этапах проекта.

**1.2. Контекст проекта**

Существующая работа салона основана на бумажном журнале или простых электронных таблицах, что приводит к ошибкам, сложностям в учете рабочего времени сотрудников, формировании отчетности и информировании клиентов. Данное приложение призвано решить эти проблемы.

**1.3. Роли и ответственность**

* **Администратор БД (АБД - Эмиль):** Проектирование, развертывание, настройка и поддержка базы данных. Обеспечение безопасности, производительности и целостности данных.
* **Backend-разработчик (Эмиль):** Разработка серверной части приложения (API) на C#, взаимодействие с БД, реализация бизнес-логики.
* **Vue Backend (Максим):** Разработка клиентской административной панели на Vue.js (компоненты, маршрутизация, состояние).
* **Vue Frontend (Даниил):** Разработка клиентского интерфейса для пользователей (Клиентов) на Vue.js (бронирование, личный кабинет).
* **Тестировщик (Максим):** Проведение функционального, интеграционного и пользовательского тестирования. Составление тест-кейсов и баг-репортов.
* **Архитектор/Схемы взаимодействия (Даниил):** Проектирование общей архитектуры приложения, схем взаимодействия между фронтендом, бэкендом и БД. Визуализация пользовательских сценариев.

**2. Описание системы**

**2.1. Типы пользователей и их возможности**

1. **Неавторизованный пользователь:**
   * Просмотр списка услуг спа-салона с описанием и стоимостью.
   * Просмотр расписания работы салона и информации о свободных слотах.
   * Регистрация в системе (статус: "Клиент").
2. **Клиент:**
   * Все возможности неавторизованного пользователя.
   * Вход в личный кабинет.
   * Онлайн-запись на доступную процедуру к выбранному мастеру.
   * Просмотр, изменение и отмена своих будущих записей.
   * Просмотр истории своих посещений.
3. **Менеджер:**
   * Все возможности Клиента.
   * Полное управление расписанием: просмотр, создание, редактирование, отмена записей для любых клиентов.
   * Управление учетными записями клиентов.
4. **Администратор БД (АБД):**
   * Доступ к системе предоставляется через UI.
   * Ответственность за выполнение резервного копирования, мониторинг производительности и применение патчей к БД.

**3. Функциональные требования**

**3.1. Модуль каталога услуг**

* **FR1. Просмотр каталога:** Система должна предоставлять публичную страницу со списком всех активных услуг.
* **FR2. Управление услугами (Менеджер):** Система должна позволять менеджеру добавлять, редактировать (название, описание, длительность, стоимость, категорию) и деактивировать услуги.

**3.2. Модуль расписания и записи**

* **FR3. Просмотр доступных слотов:** Система должна отображать календарь с доступными временными слотами для записи, учитывая длительность процедур и график работы мастеров.
* **FR4. Онлайн-запись (Клиент):** Система должна позволять клиенту забронировать выбранную услугу на свободный слот к конкретному мастеру.
* **FR5. Управление записями (Менеджер):** Система должна позволять менеджеру создавать, редактировать и отменять записи от имени любого клиента.

**3.3. Модуль пользователей и аутентификации**

* **FR6. Регистрация и аутентификация:** Система должна предоставлять функционал регистрации и входа по email и паролю.
* **FR7. Ролевая модель:** Система должна разграничивать права доступа на основе ролей (Клиент, Менеджер).

**3.4. Модуль личного кабинета**

* **FR8. История посещений (Клиент):** Клиент должен видеть список своих выполненных записей.
* **FR9. Управление своими записями (Клиент):** Клиент может отменить или перенести свою будущую запись.

**4. Технические спецификации**

**4.1. Архитектура**

* **Клиентская часть (Frontend):** Фреймворк Vue.js 3 (Composition API). Сборка - Vite.
* **Серверная часть (Backend):** Веб-API на языке C# с использованием фреймворка ASP.NET Core (версия 8+).
* **База данных:** Реляционная СУБД (MySQL).
* **Взаимодействие:** Frontend и Backend взаимодействуют по протоколу HTTP/HTTPS через RESTful API. Формат данных - JSON.

**4.2. Схема взаимодействия (Даниил)**

* Пользователь открывает страницу в браузере (Vue App).
* Vue компонент инициирует асинхронный HTTP-запрос к эндпоинту API.
* API на C# принимает запрос, обращается к БД через ORM (Entity Framework Core).
* API обрабатывает данные и возвращает ответ в формате JSON.
* Vue компонент получает ответ и отображает данные пользователю.

**5. Этапы разработки и ответственность**

1. **Этап 0: Подготовка (Все)**
   * Согласование ТЗ.
   * Настройка рабочих окружений и репозитория (Git).
2. **Этап 1: Проектирование БД и API (Эмиль)**
   * Создание ER-диаграммы БД.
   * Проектирование основных моделей данных (Service, User, Appointment).
   * Определение схемы API ( endpoints: /api/services, /api/appointments).
3. **Этап 2: Разработка ядра API (Эмиль)**
   * Реализация моделей C# и контекста БД.
   * Создание контроллеров для основных сущностей.
4. **Этап 3: Разработка административной панели (Максим)**
   * Создание компонентов Vue для управления услугами и записями.
   * Реализация роутинга и состояния.
   * Интеграция с API.
5. **Этап 4: Разработка клиентского интерфейса (Даниил)**
   * Создание публичных страниц (услуги, расписание).
   * Реализация функционала личного кабинета и записи.
   * Интеграция с API.
6. **Этап 5: Тестирование и отладка (Максим)**
   * Написание тест-кейсов.
   * Функциональное тестирование.
   * Баг-фиксинг.
7. **Этап 6: Деплой и сдача (Эмиль, Максим)**
   * Развертывание БД и API на docker.
   * Деплой клиентского веб-сайта.