# Проектна документация: BarberApp

## Въведение

Настоящата документация описва структурата и архитектурата на проект, разработен с помощта на C# за backend и TypeScript, HTML, CSS, и JavaScript за frontend. Базата данни е оформена с Tailwind CSS за оформление на потребителския интерфейс. Backend частта е разработена с JetBrains Rider, а frontend частта с Visual Studio Code.

## Структура на проекта

### Backend (C# Web API)

Backend частта на проекта се намира в папката "server" и е реализирана с помощта на C# и ASP.NET Core Web API.

#### Основни компоненти

1. BA.Application  
 - Controllers: Съдържа контролери, които управляват логиката на заявките и отговорите.  
 - Models: Съдържа модели на данни, които дефинират структурата на обектите.  
 - Utility: Съдържа полезни функционалности и помощни класове.  
 - Program.cs: Основна входна точка за стартиране на приложението.  
  
2. BA.Common  
 - Съдържа споделени ресурси и класове, които могат да бъдат използвани от други модули в проекта.  
  
3. BA.Data  
 - Data: Съдържа контекста на базата данни и конфигурациите за работа с Entity Framework.  
 - Migrations: Съдържа миграции за базата данни.  
 - Models: Дефинира моделите на данни, които се използват за базата данни.  
  
4. BA.Service  
 - Abstractions: Съдържа интерфейси, дефиниращи абстракции за логиката на услугите.  
 - Realizations: Съдържа конкретни реализации на абстракциите.  
 - DependencyInjections.cs: Управлява инжектирането на зависимости.

#### Технологии и инструменти

- ASP.NET Core Web API  
- Entity Framework Core за работа с бази данни  
- JetBrains Rider за разработка

### Frontend (TypeScript, HTML, CSS, JavaScript)

Frontend частта на проекта се намира в папката "client" и е изградена с помощта на Next.js, който използва React за потребителския интерфейс.

#### Основни компоненти

1. api: Съдържа функционалност за извикване на API на backend частта.  
2. app: Главна структура на приложението, където се дефинира навигацията.  
3. components: Съдържа повтарящи се компоненти, използвани в интерфейса.  
4. public: Съдържа статични ресурси като изображения и файлове.  
5. services: Съдържа клиентски услуги за обработка на данни и логика.  
6. shared: Споделени файлове и конфигурации, които се използват в различни части на приложението.

#### Конфигурационни файлове

- next.config.ts: Конфигурация на Next.js.  
- tailwind.config.ts: Конфигурация на Tailwind CSS за стилизиране.  
- tsconfig.json: Конфигурация за TypeScript.

#### Технологии и инструменти

- Next.js (React)  
- Tailwind CSS за стилизиране  
- TypeScript за типизация  
- Visual Studio Code за разработка

## Интеграция между Frontend и Backend

Frontend и Backend частите на проекта комуникират чрез HTTP заявки към API, предоставени от backend частта. Всички заявки са организирани чрез API endpoint-и, дефинирани в контролерите на backend-а.

### Примерен Workflow

1. Клиентската част (frontend) изпраща заявка чрез методите, дефинирани в папката api.  
2. Backend обработва заявката, използвайки своите контролери, модели и услуги.  
3. Отговорът се връща към frontend-а, където се визуализира чрез компонентите.

## Структура на базата данни

В базата данни са използвани следните таблици:

### Таблици

1. EFMigrationsHistory  
 - MigrationId: Уникален идентификатор на миграцията.  
 - ProductVersion: Версия на продукта.  
  
2. AspNetRoles  
 - Id: Уникален идентификатор на ролята.  
 - Name: Име на ролята.  
 - NormalizedName: Нормализирано име на ролята.  
 - ConcurrencyStamp: Токен за конкурентност.  
  
3. AspNetUserClaims  
 - Id: Уникален идентификатор на заявката.  
 - UserId: Потребител, към който е свързана заявката.  
 - ClaimType: Тип на заявката.  
 - ClaimValue: Стойност на заявката.  
  
4. AspNetUserLogins  
 - LoginProvider: Доставчик на логин.  
 - ProviderKey: Ключ на доставчика.  
 - ProviderDisplayName: Име на доставчика.  
 - UserId: Потребител, към който е свързан логинът.  
  
5. AspNetUsers  
 - Id: Уникален идентификатор на потребителя.  
 - FirstName: Собствено име.  
 - LastName: Фамилно име.  
 - UserName: Потребителско име.  
 - NormalizedUserName: Нормализирано потребителско име.  
 - Email: Имейл на потребителя.  
 - PasswordHash: Хеш на паролата.  
 - Други полета за управление на потребителския достъп.  
  
6. Reservations  
 - Id: Уникален идентификатор на резервацията.  
 - FirstName: Име на клиента.  
 - LastName: Фамилия на клиента.  
 - Date: Дата на резервацията.  
 - Time: Час на резервацията.  
 - Email: Имейл на клиента.  
 - Phone: Телефон на клиента.  
 - Service: Услуга, за която е направена резервацията.  
 - UserId: Потребител, свързан с резервацията.  
  
7. AspNetUserRoles  
 - UserId: Уникален идентификатор на потребителя.  
 - RoleId: Уникален идентификатор на ролята.  
  
8. AspNetRoleClaims  
 - Id: Уникален идентификатор на заявката.  
 - RoleId: Роля, към която е свързана заявката.  
 - ClaimType: Тип на заявката.  
 - ClaimValue: Стойност на заявката.  
  
9. AspNetUserTokens  
 - UserId: Потребител, свързан с токена.  
 - LoginProvider: Доставчик на логин.  
 - Name: Име на токена.  
 - Value: Стойност на токена.

### Графично представяне на базата данни

## Заключение

Проектът е структуриран с ясна архитектура, която осигурява модулност и лесна поддръжка. Използваните технологии и инструменти гарантират висока производителност и добра интеграция между frontend и backend частите.