**Dokumentacja techniczna**

**Opis projektu**

Projekt aplikacji webowej obsługującej listę kontaktów został rozdzielony na dwie osobne aplikacje:

* ContactsApi - API umożliwiające komunikację między aplikacją frontendową a bazą danych. API obsługuje operacje CRUD na rekordach w bazie danych
* ContacstApp - Aplikacja frontendowa napisana z wykorzystaniem wzorca MVVM

Aplikacja niestety nie obsługuje systemu logowania co oznacza, że każdy użytkownik może dodawać, edytować oraz usuwać kontakty

**Wykorzystane biblioteki**

Microsoft.EntityFramework

Microsoft.EntityFrameworkCore.SqlServer

Microsoft.EntityFrameworkCore.InMemory

Microsoft.AspNetCore

DotVVM.Framework

**Sposób kompilacji**

Aby skompilować projekt należy najpierw skonfigurować lokalną bazę danych. Można zrobić to na dwa sposoby. Pierwszy to zaimportowanie pliku z backup’e,m bazy Contacts.bacpac, który jest dołączony do projektu. Drugi sposób to utworzenie nowej tabeli w istniejącej już lokalnej bazy danych. Poniżej kod SQL tworzący potrzebną tabelę:

CREATE TABLE Contacts (

Id INT IDENTITY(1,1) PRIMARY KEY,

FirstName VARCHAR(50) NOT NULL,

LastName VARCHAR(50) NOT NULL,

Email VARCHAR(50) NOT NULL,

PhoneNumber VARCHAR(50) NOT NULL,

Category VARCHAR(50) NOT NULL,

BirthDate datetime NOT NULL,

Password VARCHAR(45) NOT NULL

);

Następnie należy uruchomić API. W projekcie ContactsApi w pliku app settings.json jest string ze ścieżką potrzebną do połączenia do bazy danych:



Należy podać nazwę serwera w którym znajduje się baza danych dołączona do projektu.

Następnie można uruchomić projekt Contacts App. Warto również sprawdzić czy port na jakim zostało uruchomione api jest taki sam jak w stringu URL w klasie Contact Service.cs znajdującym się w katalogu Services projektu Contacts App.



**API - Endpointy**

| **Metoda HTTP** | **Ścieżka endpointu** | **Opis** |
| --- | --- | --- |
| **GET** | **/api/GetContacts** | **Pobiera wszystkie rekordy z tabeli ‘Contacts’** |
| **GET** | **/api/GetContactById** | **Pobiera pojedynczy rekord o podanym identyfikatorze** |
| **POST** | **/api/InsertContact** | **Dodaje nowy rekord** |
| **PUT** | **/api/UpdateContact** | **Aktualizuje rekord o podanym identyfikatorze** |
| **DELETE** | **/api/DeleteContact/{Id}** | **Usuwa rekord o podanym identyfikatorze** |

**API - Klasy**

**Program.cs**

Główny plik konfiguracyjny, który dodaje kontrolery do obsługi żądań HTTP i ustawia połączenie z bazą danych

**ContactsController.cs**

Klasa obsługująca żądania ze strony klienta. Udostępnia endpointy HTTP.

Metody w klasie:

* Konstruktor tworzący ‘context’ bazy danych
* Metoda *‘public async Task<ActionResult<List<Contact>>> Get()’* pobierająca wszystkie kontakty z bazy danych i zwracająca rezultat w postaci listy
* Metoda *‘public async Task<ActionResult<Contact>> GetUserById(int Id)’*  pobierająca pojedynczy kontakt o podanym ID
* Metoda *‘public async Task<HttpStatusCode> InsertContact(Contact contact)’* dodająca nowy kontakt do bazy danych
* Metoda *‘public async Task<HttpStatusCode> UpdateContact(Contact contact)’* aktualizująca poszczególne dane dotyczące pojedynczego kontaktu
* Metoda *‘public async Task<HttpStatusCode> DeleteContact(int Id)’* usuwająca pojedynczy kontakt o podanym ID

**Contacts.cs**

Klasa reprezentująca model kontaktu w aplikacji. Zawiera pola:

* FirstName
* LastName
* Email
* PhoneNumber
* Category
* BirthDate
* Password

**ContactsContext.cs**

Klasa reprezentująca kontekst bazy danych w aplikacji. Zawiera wszystkie informacje o encjach oraz konfigurację z bazą danych

Metody w klasie:

* Konstruktor przyjmujący obiekt konfiguracyjny kontekstu
* Metoda *‘protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)’* konfigurująca połączenie z bazą danych
* Metoda *‘protected override void OnModelCreating(ModelBuilder modelBuilder)’* konfigurująca model danych
* Metoda *‘partial void OnModelCreatingPartial(ModelBuilder modelBuilder)’.* Jest to metoda częściowa, która może być wykorzystana do konfiguracji modelu danych w innych klasach

**ContactsApp - Klasy**

**Program.cs**

Ta klasa to punkt startowy i uruchamia aplikację.

Metody:

* *‘public static void Main(string[] args)’* uruchamia WebHost
* *‘public static IWebHost BuildWebHost(string[] args)’* tworzy instancję klasy IWebHost i wskazuje na klasę Startup.cs jako klasę która konfiguruje aplikację.

**Startup.cs**

Ta klasa przedstawia konfigurację aplikacji. Metody:

*‘public void ConfigureServices(IServiceCollection services)’* dodaje usługi takie jak AddWebEncoders, AddDbContext i AddDotVVM

*‘public void Configure(IApplicationBuilder app, IWebHostEnvironment env)’* definiuje pipeline HTTP oraz dodaje middleware’y UseDeveloperExceptionPage, UseExceptionHandler, UseHttpsRedirection, UseHsts, UseStaticFiles, UseRouting, UseAuthentication i UseAuthorization.

**ContactDetailModel.cs**

Klasa reprezentująca model pojedynczego kontaktu. Zawiera pola:

* Id
* FirstName
* LastName
* Email
* PhoneNumber
* Category
* Password
* BirthDate

**ContactsListModel.cs**

Klasa reprezentująca model pojedynczego kontaktu wyświetlanego w liście kontaktów. Zawiera pola:

* Id
* FirstName
* LastName

**ContactService.cs**

Ta klasa zawiera metody, które umożliwiają interakcję z usługą internetową.

W klasie zdefiniowano cztery metody, które łączą się z API i wysyłają zapytanie tworzące, edytująca lub usuwające kontakty

Metody:

* *‘public async Task<List<ContactsListModel>> GetAllContacts()’* pobiera listę wszystkich kontaktów
* *‘public async Task<ContactDetailModel> GetContactById(int Id)’* pobiera informacje o pojedynczym kontakcie o podanym ID
* *‘public async Task UpdateContact(ContactDetailModel contact)’* Aktualizuje dane o konkretnego kontaktu
* *‘public async Task InsertContact(ContactDetailModel contact)’* tworzy nowy kontakt
* *‘public async Task DeleteContact(int Id)’* Usuwa kontakt

**Klasy CreateViewModel.cs, DetailViewModel.cs, EditViewModel.cs**

Te klasy dziedziczą po klasie MasterPageViewModel i definiują model widoku dla strony dodawania, edycji oraz szczegółów kontaku.

**DefautltViewModel.cs**

Klasa definiująca główny widok strony. Zawiera metody:

* *‘public DefaultViewModel(ContactService contactService)’* konstruktor przyjmujący jako parametr obiekt klasy ConatctService.cs
* *‘public override async Task PreRender()’* pobiera listę kontaktów, która jest pobierana przez DefaultViewModel.dothml