Dokumentacja Techniczna



Fitally

# Krótki opis działania projektu

**Fitally to aplikacja webowa, która wspiera użytkowników w planowaniu, monitorowaniu i analizowaniu aktywności fizycznej. Umożliwia dodawanie ćwiczeń, zarządzanie sesjami treningowymi . Dzięki funkcjonalnościom, takim jak przypomnienia motywacyjne, spersonalizowane plany treningowe oraz ulubione ćwiczenia, aplikacja pomaga użytkownikom w utrzymaniu regularności i osiąganiu założonych celów fitness. Aplikacja została stworzona w technologii C# .NET 8 i wykorzystuje bazę danych SQLite do przechowywania informacji.**

# Autor projektu

**Klaudia Włoczyk 138438**

# Specyfikacja wykorzystanych technologii

* **Język programowania**: C#
* **Platforma**: .NET 8
* **Baza danych**: SQLite
* **Środowisko IDE**: Visual Studio 2022
* **Narzedzie ORM**: Entity Framework Core 8 (integracja z SQLite)
* **Github :** <https://github.com/klaudiawloc/Fitally_BD_WSB.git>

# Instrukcje pierwszego uruchomienia projektu

Konfigurowanie środowiska oraz ustawienie bazy danych:

* Instalacja Visual Studio 2022 (wraz z komponentami ASP.NET and web development, Entity Framework tools)
* Instalacja .NET 8 SDK
* Opcjonalna instalacja SQLite Browser (do przeglądania lokalnej bazy danych)

Ustawienie bazy danych SQLite:

* Sprawdzenie czy plik bazy danych jest w katalogu projektu.
* Visual Studio automatycznie powinien wykonać migrację.

Uruchomienie aplikacji:

* W Visual Studio wybierz przycisk „Start” lub uruchom aplikację, wpisując dotnet run w terminalu. Spowoduje to uruchomienie aplikacji w trybie klient-serwer.
* W przeglądarce otworzy się strona, gdzie będzie można testować funkcje Fitally.

# Opis struktury projektu

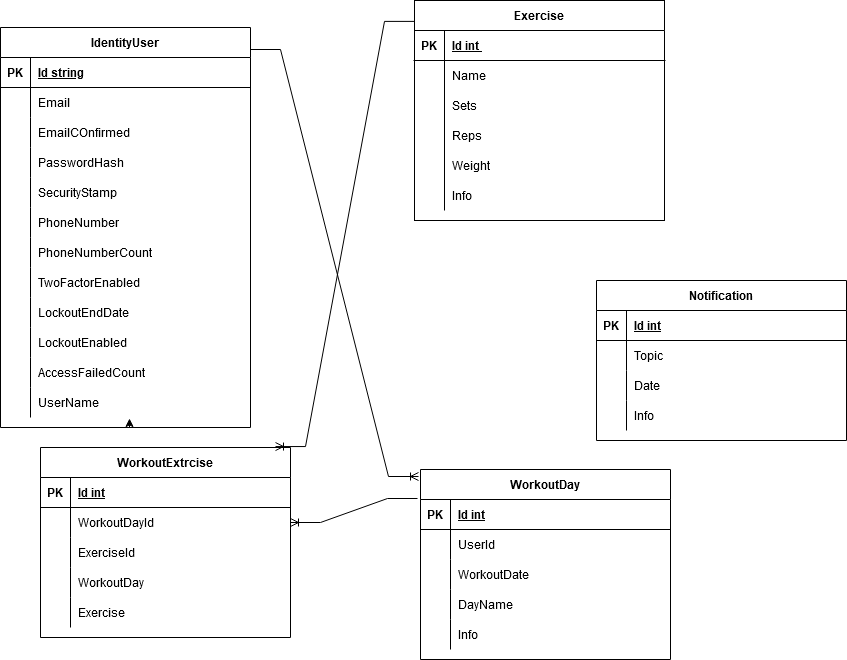
Projekt Fitally został zaprojektowany zgodnie z architekturą ASP.NET Core MVC, co zapewnia - oddzielenie logiki biznesowej (Controllers), danych (Models) i widoków (Views). Skalowalność dzięki Entity Framework Core i migracjom oraz interfejs z wykorzystaniem widoków Razor.

**Diagram encji i schemat relacji :**

### Opis diagramu encji (ERD – Entity Relationship Diagram):

Diagram przedstawia strukturę bazy danych dla systemu treningowego, zawierającą następujące encje i ich atrybuty:

1. **IdentityUser** (Główna encja użytkownika):
   * **Klucz główny (PK):** Id (typ string)
   * Atrybuty:
     + Email, EmailConfirmed
     + PasswordHash, SecurityStamp
     + PhoneNumber, PhoneNumberCount
     + TwoFactorEnabled, LockoutEndDate, LockoutEnabled
     + AccessFailedCount, UserName
   * Reprezentuje użytkownika systemu z danymi logowania i zabezpieczeń.
2. **Exercise** (Ćwiczenia):
   * **Klucz główny (PK):** Id (typ int)
   * Atrybuty:
     + Name (nazwa ćwiczenia)
     + Sets (liczba serii), Reps (liczba powtórzeń), Weight (ciężar)
     + Info (dodatkowe informacje)
   * Przechowuje dane dotyczące poszczególnych ćwiczeń.
3. **WorkoutExercise** (Relacja między treningami i ćwiczeniami):
   * **Klucz główny (PK):** Id (typ int)
   * Atrybuty:
     + WorkoutDayId (powiązanie z dniem treningowym)
     + ExerciseId (powiązanie z ćwiczeniem)
     + WorkoutDay, Exercise (relacje do innych encji)
   * Łączy konkretne ćwiczenia z konkretnymi dniami treningowymi.
4. **WorkoutDay** (Dni treningowe):
   * **Klucz główny (PK):** Id (typ int)
   * Atrybuty:
     + UserId (powiązanie z użytkownikiem)
     + WorkoutDate (data treningu)
     + DayName (nazwa dnia)
     + Info (dodatkowe informacje)
   * Przechowuje harmonogram treningów użytkownika.
5. **Notification** (Powiadomienia):
   * **Klucz główny (PK):** Id (typ int)
   * Atrybuty:
     + Topic (temat powiadomienia)
     + Date (data powiadomienia)
     + Info (dodatkowe informacje)
   * Przechowuje informacje o powiadomieniach systemowych.

****

### Schemat relacji pomiędzy encjami:

1. **IdentityUser → WorkoutDay (1:N):**
   * Jeden użytkownik (IdentityUser) może mieć wiele dni treningowych (WorkoutDay).
   * Relacja jest zdefiniowana poprzez UserId w tabeli WorkoutDay.
2. **WorkoutDay → WorkoutExercise (1:N):**
   * Jeden dzień treningowy (WorkoutDay) może zawierać wiele ćwiczeń (WorkoutExercise).
   * Relacja jest zdefiniowana poprzez WorkoutDayId.
3. **Exercise → WorkoutExercise (1:N):**
   * Jedno ćwiczenie (Exercise) może być przypisane do wielu treningów (WorkoutExercise).
   * Relacja jest zdefiniowana poprzez ExerciseId.
4. **Pozostałe relacje:**
   * Brak bezpośrednich powiązań między Notification a innymi tabelami, co sugeruje, że powiadomienia są niezależne od użytkownika lub treningu.

**Controllers (Kontrolery):**

Tutaj są kontrolery, czyli miejsca, gdzie obsługiwane są zapytania od użytkownika. Każdy kontroler odpowiada za konkretną funkcję.

1. ExercisesController

* Index
  + http: GET
  + Opis: pobiera listę ćwiczeń i przekazuje ją do widoku Index
* Details
  + http: GET
  + Parametry: int? Id (Identyfikator ćwiczenia)
  + Opis: Wyświetla szczegóły ćwiczenia o podanym id
* Create
  + http: GET
  + Opis: Wyświetla formularz tworzenia ćwiczenia
  + http: POST
  + Parametry: Exercise exercise (Dane przesłane z formularza)
  + Opis: Dodaje nowe ćwiczenia i zapisuje je w bazie danych, następnie przekierowuje do Index
* Edit
  + http: GET
  + Parametry: int? Id (Identyfikator ćwiczenia)
  + Opis: Wyświetla formularz edycji.
  + http: POST
  + Parametry: int id, Exercise exercise (Dane przesłane z formularza)
  + Opis: Aktualizuje dane ćwiczenia i zapisuje zmiany w bazie
* Delete
  + http: GET
  + Parametry: int? Id (Identyfikator ćwiczenia)
  + Opis: Wyświetla szczegóły ćwiczenia do usunięcia
  + http: POST
  + Parametry: int id
  + Opis: Usuwa ćwiczenie z bazy danych
* ExerciseExists
  + Parametry: int id (Identyfikator ćwiczenia)
  + Opis: Sprawdza, czy ćwiczenie istnieje w bazie danych

1. WorkoutDaysController
   * Index
     + http: GET
     + Opis: Pobiera listę dni treningowych i przekazuje je do widoku Index
   * Details
     + http: GET
     + Parametry: int? Id (Identyfikator dnia treningowego)
     + Opis: Wyświetla szczegóły dnia treningowego
   * Create
     + http: GET
     + Opis: Wyświetla formularz tworzenia dnia treningowego.
     + http: POST
     + Parametry: WorkoutDay workoutDay (Dane przesłane z formularza)
     + Opis: Dodaje nowy dzień treningowy i zapisuje go w bazie
   * Edit
     + http: GET
     + Parametry: int? Id (Identyfikator dnia treningowego)
     + Opis: Wyświetla formularz edycji dnia treningowego.
     + http: POST
     + Parametry: int id, WorkoutDay workoutDay (Dane przesłane z formularza)
     + Opis: Aktualizuje dane dnia treningowego.
   * Delete
     + http: GET
     + Parametry: int? id (identyfikator dnia treningowego)
     + Opis: Wyświetla szczegóły dnia treningowego do usunięcia
     + http: POST
     + Parametry: int id
     + Opis: Usuwa dzień treningowy z bazy danych i przekierowuje do Index
2. HomeController
   * Index
     + http: GET
     + Opis: Wyświetla stronę główną aplikacji
   * Privacy
     + http: GET
     + Opis: Wyświetla stronę z polityką prywatności
   * Timer
     + http: GET
     + Opis: Wyświetla stronę z funkcją timera
   * Error
     + http: GET
     + Opis: Wyświetla stronę błędu
3. NotificationsController
   * Index
     + http: GET
     + Opis: Pobiera listę powiadomień i przekazuje ją do widoku Index
   * Details
     + http: GET
     + Parametry: int? Id (identyfikator powiadomienia)
     + Opis: Wyświetla szczegóły powiadomienia o podanym id.
   * Create
     + http: GET
     + Opis: Wyświetla formularz tworzenia powiadomienia.
     + http: POST
     + Parametry: Notification notification (dane przesłane z formularza)
     + Opis: Dodaje nowe powiadomienie i zapisuje je w bazie danych. Przekierowuje do Index.
   * Edit
     + http: GET
     + Parametry: int? Id (identyfikator powiadomienia)
     + Opis: Wyświetla formularz edycji powiadomienia
     + http: POST
     + Parametry: int id, Notification notification (Dane przesłane z formularza)
     + Opis: Aktualizuje dane powiadomienia w bazie
   * Delete
     + http: GET
     + Parametry: int? Id (Identyfikator powiadomienia)
     + Opis: Wyświetla szczegóły powiadomienia do usunięcia
     + http: POST
     + Parametry: int id (Identyfikator powiadomienia)
     + Opis: Usuwa powiadomienie z bazy i przekierowuje do Index
4. UserNotificationsController
   * Index
     + http: GET
     + Opis: Pobiera listę powiadomień i przekazuje ją do widoku Index
5. UserWorkoutExercisesController
   * Index
     + http: GET
     + Opis: Pobiera listę ćwiczeń przypisanych do dni treningowych aktualnego użytkownika i przekazuje je do widoku Index
   * Details
     + http: GET
     + Parametry: int? Id (Identyfikator ćwiczenia w dniu treningowym)
     + Opis: Wyświetla szczegóły ćwiczenia przypisanego do dnia treningowego
   * Create
     + http: GET
     + Opis: Wyświetla formularz dodawania ćwiczenia do dnia treningowego. Generuje listy rozwijane dla ćwiczeń (ExerciseId) i dni treningowych (WorkoutDayId) aktualnego użytkownika.
     + https: POST
     + Parametry: WorkoutExercise workoutExercise (Dane przesłane z formularza)
     + Opis: Dodaje nowe ćwiczenie do dnia treningowego i zapisuje je w bazie. Przekierowuje do Index
   * Edit
     + http: GET
     + Opis: Wyświetla formularz edycji ćwiczenia. Generuje listy rozwijane dla ćwiczeń (ExerciseId) i dni treningowych (WorkoutDayId) aktualnego użytkownika.
     + http: POST
     + Parametry: int id, WorkoutExercise workoutExercise (Dane z formularza)
     + Opis: Aktualizuje dane przypisanego ćwiczenia w bazie. Obsługuje konflikty zapisu.
   * Delete
     + http: GET
     + Parametry: int? Id (Identyfikator ćwiczenia w dniu treningowym)
     + Opis: Wyświetla szczegóły ćwiczenia do usunięcia
     + http: POST
     + Parametry: int id (Identyfikator ćwiczenia w dniu treningowym)
     + Opis: Usuwa ćwiczenie z dnia treningowego i zapisuje zmiany w bazie. Przekierowuje do Index.
6. UserWorkoutsController
   * Index
     + http: GET
     + Opis: Pobiera listę dni treningowych zalogowanego użytkownika wraz z przypisanymi ćwiczeniami. Przekazuje dane do widoku Index
   * Details
     + http: GET
     + Parametry: int? Id (Identyfikator dnia treningowego)
     + Opis: Wyświetla szczegóły dnia treningowego, w tym przypisane ćwiczenia.
   * Create
     + http: GET
     + Opis: Wyświetla formularz tworzenia nowego dnia treningowego
     + http: POST
     + Parametry: WorkoutDay workoutDay (Dane przesłane z formularza)
     + Opis: Dodaje nowy dzień treningowy i zapisuje go w bazie danych. Przekierowuje do Index
   * Edit
     + http: GET
     + Opis: Wyświetla formularz edycji dnia treningowego.
     + http: POST
     + Parametry: int id, WorkoutDay workoutDay (Dane przesłane z formularza)
     + Opis: Aktualizuje dane dnia treningowego w bazie. Obsługuje konflikty zapisu.
   * Delete
     + http: GET
     + Parametry: int id (Identyfikator dnia treningowego)
     + Opis: Wyświetla szczegóły dnia treningowego do usunięcia.
     + http: POST
     + Parametry: int id (Identyfikator dnia treningowego)
     + Opis: Usuwa dzień treningowy z bazy danych i zapisuje zmiany. Przekierowuje do Index
7. WorkoutExercisesController
   * Index
     + http: GET
     + Opis: Pobiera listę ćwiczeń przypisanych do dni treningowych. Dane są wczytywane z powiązaniem do ćwiczeń (Exercise) i dni treningowych (WorkoutDay) i przekazywane do widoku Index
   * Details
     + http: GET
     + Parametry: int? Id (Identyfikator przypisanego ćwiczenia)
     + Opis: Wyświetla szczegóły przypisanego ćwiczenia
   * Create
     + http: GET
     + Opis: Wyświetla formularz tworzenia przypisanego ćwiczenia. Generuje listy rozwijane dla dostępnych ćwiczeń (ExerciseId) i dni treningowych (WorkoutDayId)
     + http: POST
     + Parametry: WorkoutExercise workoutExercise (Dane przesłane z formularza)
     + Opis: Dodaje nowe przypisanie ćwiczenia do dnia treningowego i zapisuje je w bazie danych. Przekierowuje do Index.
   * Edit
     + http: GET
     + Parametry: int? Id (Identyfikator przypisanego ćwiczenia)
     + Opis: Wyświetla formularz edycji przypisanego ćwiczenia. Generuje listy rozwijane dla dostępnych ćwiczeń i dni treningowych.
     + http: POST
     + Parametry: int id, WorkoutExercise workoutExercise (Dane przesłane z formularza)
     + Opis: Aktualizuje przypisanie ćwiczenia w bazie danych. Obsługuje konflikty zapisu.
   * Delete
     + http: GET
     + Parametry: int? Id (Identyfikator przypisanego ćwiczenia)
     + Opis: Wyświetla szczegóły przypisanego ćwiczenia do usunięcia.
     + http: POST
     + Parametry: int id (Identyfikator przypisanego ćwiczenia)
     + Opis: Usuwa przypisanie ćwiczenia z bazy danych i zapisuje zmiany. Przekierowuje do Index.

**Models (Modele)**:

W tym folderze trzymamy modele, czyli definicje naszych danych. Są tu klasy opisujące, jak wyglądają nasze obiekty.

1. ErrorViewModel (Model do obsługi błędów w aplikacji, który pozwala na identyfikację błędów na podstawie identyfikatora żądania)
   * RequestId: String? (Identyfikator żądania)
   * ShowRequestId: Bool (Właściwość wyliczeniowa, która sprawdza, czy RequestId nie jest pusty)
2. Exercise (Reprezentuje pojedyncze ćwiczenie)
   * Id: Int (Identyfikator ćwiczenia)
   * Name: String (Nazwa ćwiczenia)
   * Sets: Int (Liczba serii w ćwiczeniu)
   * Reps: Int (Liczba powtórzeń w serii)
   * Weight: Double (Obciążenie przypisane do ćwiczenia)
   * Info: String (Dodatkowe informacje na temat ćwiczenia)
   * WorkoutExercises: List<WorkoutExercise> (Lista powiązanych ćwiczeń w ramach treningów)
3. Notification (Reprezentuje powiadomienia w aplikacji)
   * Id: Int (Identyfikator powiadomienia)
   * Topic: String (Temat powiadomienia)
   * Date: DateTime (Data powiadomienia)
   * Info: String (Treść powiadomienia)
4. SeedUserModel (Reprezentuje użytkownika podczas procesu seedowania danych)
   * UserName: String (Nazwa użytkownika)
   * Password: String (Hasło użytkownika)
5. WorkoutDay (Dzień treningowy użytkownika. Powiązany z ćwiczeniami i użytkownikiem)
   * Id: Int (Identyfikator dnia treningowego)
   * UserId: String (Identyfikator użytkownika, do którego należy dzień treningowy)
   * WorkoutDate: DateTime (Data dnia treningowego)
   * DayName: String (Nazwa dnia treningowego)
   * Info: String (Zawiera informacje na temat dnia)
   * WorkoutExercises: List<WorkoutExercise> (Lista powiązanych ćwiczeń)
6. WorkoutExercise (Model relacyjny, który łączy dni treningowe z ćwiczeniami)
   * Id: Int (Identyfikator powiązania)
   * WorkoutDayId (Identyfikator dnia treningowego)
   * ExerciseId: Int (Identyfikator ćwiczenia)
   * WorkoutDay: WorkoutDay (Powiązany dzień treningowy)
   * Exercise: Exercise (Powiązane ćwiczenie)
7. Data (Dane):

Ten folder zawiera głównie DbContext, czyli klasę, która łączy się z bazą SQLite i zarządza danymi. Znajdują się tu też migracje, które umożliwiają dodawanie nowych tabel czy modyfikowanie istniejących.

1. Views (Widoki):

Tu będą strony, które będą widoczne dla użytkownika.

1. Wwwroot:

Folder z zasobami.

# System użytkowników

Obejmuje różne role użytkowników, ich uprawnienia oraz sposób zarządzania danymi.

**Role**:

* User: Podstawowy dostęp po rejestracji oraz zalogowaniu. Dodawanie, edycja, historia treningów. Wgląd do statystyk, zaplanowane powiadomienia.
* Administrator: Możliwość tworzenia notyfikacji oraz edycja ćwiczeń.

# Najciekawsze funkcjonalności

# Powiadomienia motywacyjne

* Motywujące wiadomości.
* Przypomnienia o treningach.

**Licznik przerw między seriami**

* Timer, który użytkownik może uruchomić po wykonaniu serii ćwiczeń

**Notatki do dni**

* Użytkownik może dodawać notatki/opisy do dni treningowych.