Programowanie zaawansowane: Dokumentacja techniczna projektu

Łukasz Michalak (138325) Piotr Namysł (138328)

Wirtualny Ogrodnik.

Opis:

Aplikacja umożliwia użytkownikom śledzenie pielęgnacji roślin w domu, pozwalając na dodawanie roślin do "wirtualnej hodowli", ustawianie przypomnień o podlewaniu, nawożeniu i przesadzaniu, a także prowadzenie dziennika wzrostu rośliny. Funkcje obejmują: dodawanie roślin z bazy lub ręcznie, otrzymywanie powiadomień o pielęgnacji, dokumentowanie wzrostu rośliny oraz śledzenie statystyk związanych z podlewaniem i wzrostem roślin.

Specyfikacja wykorzystanych technologii:

- .NET 8.0

- C# 12

- Blazor

- ASP .NET API

- MS SQL Server

- Entity Framework

Instrukcje pierwszego uruchomienia projektu:

1. Pobierz projekt: https://github.com/PiotrNamysl/VirtualGardener
2. Upewnij się, że masz zainstalowane:
   .NET 8.0 SDK
   MS SQL Server
   Visual Studio 2022 z wymaganymi rozszerzeniami.
3. W pliku appsettings.json zaktualizuj connection string:
   "ConnectionStrings": {
   "DefaultConnection":
   "Server=localhost;Database=VirtualGardener;Trusted_Connection=True;"
   }
4. W konsoli Menedżera pakietów w Visual Studio wykonaj:
   Update-Database

Warstwy aplikacji:

1. Frontend (Blazor Server):

   Widoki i interfejs użytkownika, np. formularze dodawania roślin, listy roślin i harmonogramy pielęgnacji.

2. Backend (ASP.NET Core API):

   Logika aplikacji i połączenie z bazą danych.

3. Baza danych (MS SQL):

   Przechowuje dane użytkowników, roślin, harmonogramów i logów wzrostu.

Foldery projektu:

Models: Definicje modeli danych.

Data: Kontekst bazy danych i konfiguracje migracji.

Controllers: API obsługujące operacje użytkownika.

Pages: Widoki Blazor.

Modele

1. User

- **Opis**: Przechowuje dane użytkowników.
- **Pola**:

  - Id (int): Identyfikator użytkownika.

  - Name (string): Imię użytkownika (max 50 znaków, wymagane).

  - Email (string): Unikalny adres email (walidacja email).

  - PasswordHash (string): Hasło przechowywane jako hash.

  - Role (string): Rola użytkownika (np. Admin, User).

2. Plant

- **Opis**: Reprezentuje roślinę dodaną przez użytkownika.
- **Pola**:

  - Id (int): Identyfikator rośliny.

- Name (string): Nazwa rośliny (max 50 znaków, wymagane).

- Species (string): Gatunek rośliny (opcjonalnie).

- UserId (int): Identyfikator właściciela (klucz obcy).

- AddedDate (DateTime): Data dodania.

3. CareTask

- **Opis**: Zadanie związane z pielęgnacją rośliny.
- **Pola**:

  - Id (int): Identyfikator zadania.

  - PlantId (int): Identyfikator rośliny (klucz obcy).

  - TaskType (string): Typ zadania (np. Watering).

  - DueDate (DateTime): Data wykonania zadania.

  - Completed (bool): Status ukończenia.

4. GrowthLog

- **Opis**: Dziennik wzrostu rośliny.
- **Pola**:

  - Id (int): Identyfikator logu.

  - PlantId (int): Identyfikator rośliny (klucz obcy).

  - Date (DateTime): Data logu.

  - Notes (string): Notatki dotyczące wzrostu.

Kontrolery i metody

1. **UserController**

- **Metody**:

  POST /api/users/register: Rejestracja użytkownika.
  Parametry: UserDto.
  - Name (string): Imię użytkownika.

  - Email (string): Unikalny adres e-mail użytkownika.

  - Password (string): Hasło użytkownika.

  Zwraca:

Status: Informację o zakończeniu rejestracji (200 OK lub 400 BadRequest w przypadku błędu). Jeśli adres email jest już używany, zwróci status 400 BadRequest z odpowiednim komunikatem.

POST /api/users/login: Logowanie użytkownika.
Parametry: LoginDto (JSON).

- Email (string): Unikalny adres e-mail użytkownika.
- Password (string): Hasło użytkownika.

Zwraca:
Status: Informację o powodzeniu logowania (200 OK lub 401 Unauthorized w przypadku niepoprawnych danych). Po poprawnym logowaniu sesja zostaje utworzona, co oznacza, że użytkownik pozostaje zalogowany do momentu wylogowania. Jeśli dane logowania są niepoprawne, zwróci status 401 Unauthorized.

GET /api/users/logout: Wylogowanie użytkownika.
Zwraca: Status operacji.

2. PlantController

- Metody:

  GET /api/plants: Pobierz rośliny użytkownika.
  Parametry: Id użytkownika.
  Zwraca: Listę roślin.

  POST /api/plants: Dodaj nową roślinę.
  Parametry: PlantDto.
  Zwraca: Status operacji.

  DELETE /api/plants/{id}: Usuń roślinę.
  Parametry: Id rośliny.
  Zwraca: Status operacji.

3. CareTaskController

- Metody:

  GET /api/tasks/{plantId}: Pobierz zadania pielęgnacyjne.
  Parametry: Id rośliny.
  Zwraca: Listę zadań.

  POST /api/tasks: Dodaj zadanie pielęgnacyjne.
  Parametry: CareTaskDto.
  Zwraca: Status operacji.

PUT /api/tasks/{id}: Zaktualizuj status zadania.
Parametry: Id zadania.
Zwraca: Status operacji.

System użytkowników

1. Role użytkowników:

   Admin: Może zarządzać wszystkimi użytkownikami i danymi.

   User: Może dodawać, edytować i przeglądać swoje dane.

2. Powiązania danych:

   Dane o roślinach, zadaniach i logach wzrostu są przypisane do zalogowanego użytkownika.

   Publiczne dane obejmują jedynie stronę główną.

3. Sposób nadawania ról:

   Role przypisywane są podczas rejestracji. Admina może ustawić tylko inny admin.

Najciekawsze funkcjonalności:

Algorytm generuje harmonogram pielęgnacji (np. podlewanie co tydzień) na podstawie typu rośliny.

Możliwość automatycznego pobrania informacji o roślinie z zewnętrznej bazy danych.

Wizualizacje wzrostu roślin w formie statystyk, bazujące na dzienniku wzrostu.