Dokumentacja techniczna projektu E621Browser

Projekt E621Browser to aplikacja internetowa umożliwiająca przeglądanie i zapisywanie obrazków (artworków) o tematyce furry ze strony E621. Aplikacja została stworzona w oparciu o platformę .NET 8.0 i wykorzystuje zestaw narzędzi dostępnych w ekosystemie .NET, takich jak Microsoft Identity do rejestracji użytkowników oraz SQLite wraz z Entity Framework do obsługi bazy danych.

1. Cel dokumentacji

Niniejsza dokumentacja techniczna ma na celu zapewnienie kompleksowego zrozumienia struktury, działania oraz konfiguracji projektu. Przeznaczona jest zarówno dla programistów pracujących nad projektem, jak i dla osób, które chciałyby zapoznać się z technicznymi aspektami działania aplikacji.

2. Zakres dokumentacji

Dokumentacja zawiera szczegółowe informacje na temat architektury projektu, struktury katalogów, wykorzystanych technologii, konfiguracji środowiska deweloperskiego oraz sposobu uruchomienia projektu na własnym komputerze. Omówione też zostały kluczowe funkcjonalności oraz sposoby ich implementacji.

3. Ogólny opis projektu

E621Browser umożliwia użytkownikom przeglądanie i zapisywanie artworków ze strony E621, poprzez zapewnione przez twórców API, która jest jednym z najbardziej popularnych źródeł contentu furry. Aplikacja umożliwia przeglądanie najnowszych obrazków, jak i wyszukiwanie ich po określonych przez użytkownika tagach, a także rejestrację i logowanie użytkowników w celu zapewnienia spersonalizowanego doświadczenia. Każdy zarejestrowany użytkownik jest w stanie zapisywać artworki które wpadły mu w oko, a także przeglądać listę zapisanych przez niego obrazków i ewentualne usuwanie ich z listy. Dodatkowo użytkownicy są w stanie komentować obrazki i usuwać swoje komentarze.

4. Technologie wykorzystane w projekcie

Projekt został zbudowany przy wykorzystaniu następujących technologii:

- **.NET 8.0 z ASP.NET Core**: Framework programistyczny wykorzystywany do budowy aplikacji internetowych w języku C#. Aplikacja wykorzystuje technologię Model-View-Controller (MVC).
- **Microsoft Identity**: Biblioteka umożliwiająca i ułatwiająca obsługę procesu uwierzytelniania i autoryzacji użytkowników.
- **SQLite**: Lekka baza danych relacyjna wykorzystywana do przechowywania danych aplikacji, takich jak dane logowania użytkowników czy zapisane przez użytkowników artworki. Ta baza danych zapisywana jest w pliku `app.db`.
- **Entity Framework**: Narzędzie Object-Relational Mapping (ORM) służące do mapowania obiektów aplikacji na struktury danych w bazie danych.

5. **Najważniejsze pliki i katalogi występujące w projekcie**

- `.gitignore`: Plik konfiguracyjny dla systemu kontroli wersji Git, który definiuje które pliki mają być ignorowane.
- `E621Browser.sln`: Plik solucji Visual Studio.
- `E621Browser.csproj`: Główny plik projektu aplikacji.
- `Program.cs`: Zawiera kod źródłowy głównej klasy aplikacji, która definiuje metodę `Main` i konfiguruje uruchomienie aplikacji.
- `appsettings*.json`: Pliki konfiguracyjne aplikacji, które zawierają ustawienia dotyczące środowiska, takie jak połączenie z bazą danych.
- `app.db`: Plik bazy danych SQLite, w którym są przechowywane dane aplikacji.
- `wwwroot`: Katalog zawierający zasoby publiczne dostępne dla klienta, takie jak pliki CSS, JavaScript, obrazki, itd.
- `Controllers`: Katalog zawierający kontrolery aplikacji, które przetwarzają żądania HTTP i zwracają odpowiedzi.
- `Data`: Katalog zawierający pliki związane z dostępem do danych, takie jak klasy kontekstu bazy danych.
- `DTOs`: Katalog zawierający obiekty transferu danych.
- `Models`: Katalog zawierający modele danych aplikacji, które reprezentują struktury danych (takie jak `Artwork`) używane w aplikacji.
- `Services`: Katalog zawierający usługi aplikacji.
- `Views`: Katalog zawierający widoki aplikacji, które definiują warstwę prezentacji danych dla użytkownika.
- `Migrations`: Katalog zawierający pliki migracji dla bazy danych, które są generowane przez Entity Framework podczas wprowadzania zmian w modelu danych.

6. **Dokładny sposób działania aplikacji**

Aplikacja E621Browser została zaprojektowana w jak najbardziej optymalny sposób, w celu ograniczenia wykonywania zbyt wielu żądań do API E621. Oto dokładny sposób jej działania:

1. Po uruchomieniu aplikacji użytkownik trafia na stronę główną, która wyświetla listę najnowszych artworków pobranych z API. Wszystkie te operacje są obsługiwane przez kontroler `ArtworkController`.
2. Żądanie API jest obsługiwane przez parser JSON, który interpretuje zwrócone dane. Lista artworków jest analizowana, a artworki nieistniejące w bazie danych są do niej zapisywane, w celu późniejszego ograniczenia odpytywania API E621.
3. Użytkownik jest w stanie przeglądać kolejne strony z obrazkami, ta czynność wykonuje kolejne zapytania do API.
4. Użytkownik ma możliwość wyszukiwania artworków na podstawie tagów. Tagi na stronach typu booru takich jak właśnie E621 są krótkimi

tagami tekstowymi, są rozdzielone spacjami, a każdy artwork ma przypisane określone z góry tagi w celu łatwiejszej kategoryzacji i możliwości wyszukiwania. Ta funkcjonalność jest obsługiwana przez kontroler `ArtworkController` i wyświetla wyniki wyszukiwania na podstronie `Views/Artwork/Search`.

5. Po kliknięciu na miniaturkę danego artworka, użytkownik może go zobaczyć w pełnym rozmiarze na podstronie `Views/Artwork/Details`. Wyświetlone są też dodatkowe dane, takie jak opis i tagi. W tym widoku umieszczony jest przycisk umożliwiający zapisanie lub usunięcie z listy zapisanych danego obrazka.
6. Po kliknięciu na przycisk wyświetlana jest strona z rejestracją, jeśli użytkownik nie jest aktualnie zalogowany, albo wykonywana jest akcja zapisania lub usunięcia obrazka. Zapisanie obrazka polega na wyszukaniu w tabeli powiązanej z modelem `SavedArtwork` w bazie danych odpowiedniego wpisu, jeśli takowy nie istnieje, jest on dodawany wraz z ID użytkownika, a jeśli istnieje, do listy `UserIds` obiektu dodane jest ID zapisującego użytkownika. Usuwanie obrazka z listy przebiega w analogiczny sposób, wyszukiwany jest odpowiedni obiekt w bazie danych, a z listy `UserIds` jest usuwane ID obecnie zalogowanego użytkownika.
7. Na tej stronie zalogowany użytkownik jest również w stanie czytać komentarze innych użytkowników, a także dodawać i usuwać własne. Ta czynność jest obsługiwana przez `CommentsController` i zaprojektowana została w taki sposób, że użytkownicy mogą usuwać wyłącznie swoje komentarze, a także nie są w stanie ich dodać bez rejestracji.
8. Wszystkie metody modyfikujące obiekty w bazie danych powiązane z użytkownikami są chronione przed nieodpowiednim dostępem przez atrybut `[Authorize]`, dzięki czemu tylko zalogowany i upoważniony użytkownik jest w stanie zmienić wyłącznie te dane, które dotyczą jego.

Dzięki zaimplementowanymi funkcjonalnościami użytkownik ma możliwość wygodnego przeglądania i zarządzania artworkami z serwisu E621, jednocześnie zachowując wysoką wydajność i nie wykorzystując w nadmiernym stopniu API oryginalnego źródła.

7. Uruchomienie projektu

Wymagania:

- .NET 8.0 SDK: Wymagane do skompilowania i uruchomienia aplikacji. Można go pobrać ze strony Microsoftu.
- Visual Studio lub JetBrains Rider (opcjonalnie): Są to zintegrowane środowiska programistyczne, w których można uruchomić projekt.

Instrukcja uruchomienia:

1. Sklonuj repozytorium przez stronę na GitHubie lub poprzez komendę:
   `git clone https://github.com/Lukrecjaaa/E621Browser.git`
2. Uruchom projekt
   a. w Visual Studio:
      i. Otwórz projekt wybierając opcję “Open a project or solution”.
      ii. Upewnij się, że domyślny projekt w rozwiązaniu to `E621Browser`.
      iii. Uruchom aplikację, klikając zieloną strzałkę lub wciskając klawisz F5.
   b. w JetBrains Rider:
      i. Otwórz projekt wybierając opcję “Open”.
      ii. Wybierz plik `E621Browser.sln` z katalogu projektu.
      iii. Upewnij się że projekt `E621Browser` jest zaznaczony jako projekt startowy.
      iv. Uruchom aplikację, klikając przycisk “Run” lub wciskając klawisze Shift + F10.
   c. w wierszu poleceń:
      i. Przejdź do katalogu projektu:
      `cd E621Browser`
      ii. Uruchom aplikację za pomocą narzędzia `dotnet`:
      `dotnet run --project E621Browser`.

Po wykonaniu tych kroków powinna zostać automatycznie uruchomiona przeglądarka z widoczną stroną startową aplikacji. Wszelkie potrzebne migracje i aktualizacje bazy danych są już zaaplikowane, dzięki czemu uruchomienie projektu jest łatwiejsze i nie wymaga samodzielnego korzystania z Entity Framework.