Krystian Pośledni, Jakub Pawlicki

System Zarządzania Użytkownikami, Rankingiem i Powtórkami dla Deterministycznych Gier

1. Wprowadzenie

Projekt zakłada stworzenie systemu klient-serwer obsługującego użytkowników oraz podstawowe funkcje wspierające deterministyczne gry. System ma zapewniać:

- Zarządzanie użytkownikami - rejestrację, logowanie, zmianę danych oraz usunięcie konta.
- Obsługę poziomów - przeglądanie oraz wybór poziomów.
- Obsługę rankingów - gromadzenie i przetwarzanie wyników graczy w celu generowania tabel wyników.
- Obsługę powtórek - przechowywanie danych rozgrywek umożliwiających ich weryfikację i odtwarzanie.

System stanowi fundament techniczny dla przyszłej gry, definiując strukturę wymiany danych między klientem a serwerem oraz umożliwiając łatwą rozbudowę o dodatkowe funkcje związane z mechaniką gry.

2. Specyfikacja wykorzystanych technologii

System został stworzony we frameworku ASP.NET Core 8.0 używając szablonu Aplikacja internetowa ASP.NET Core (Model-View-Controller).

Projekt używa systemu zarządzania bazą danych MS SQL Server Express LocalDB.

Interakcje z bazą danych są realizowane za pomocą Entity Framework Core 8.0.

System używa ASP.NET Core Identity UI, który zapewnia gotowe komponenty i mechanizmy do zarządzania użytkownikami, w tym rejestrację, logowanie, zarządzanie hasłami oraz funkcje autoryzacji.

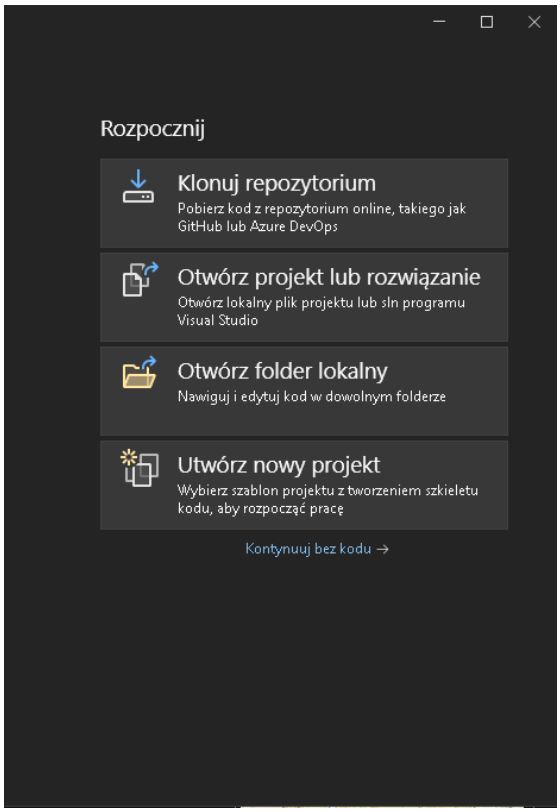
Csharp.Scripting używany jest w celu walidacji wyrażeń logicznych rozwiązywanych przez użytkowników w poziomach gry.

Używane są następujące pakiety:

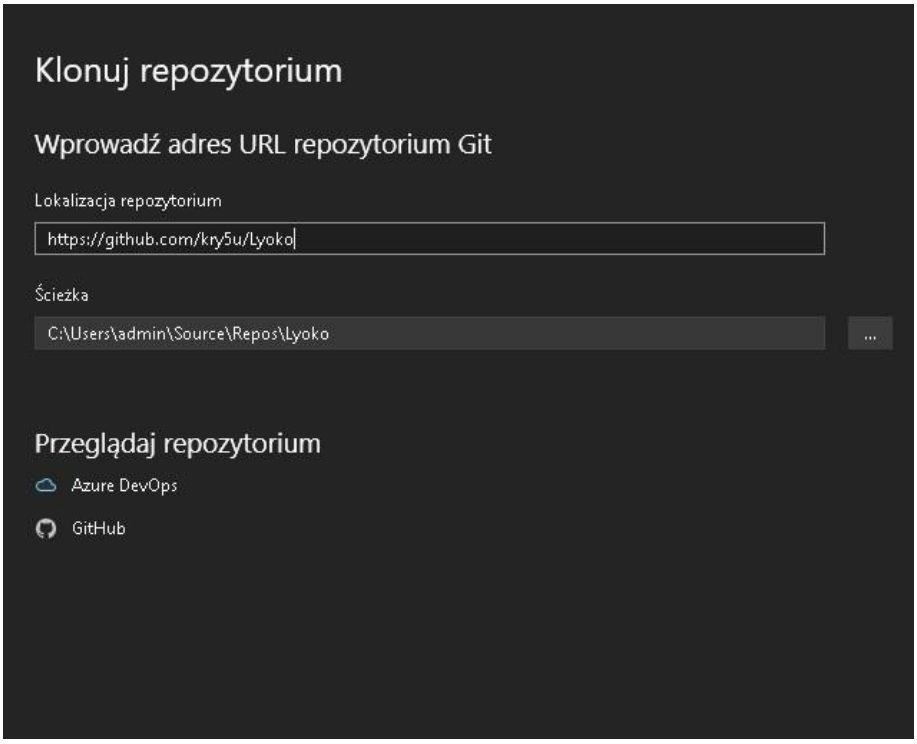
- Microsoft.AspNetCore.Components.QuickGrid
- Microsoft.AspNetCore.Diagnostics.EntityFrameworkCore
- Microsoft.AspNetCore.Identity.EntityFrameworkCore
- Microsoft.AspNetCore.Identity.UI
- Microsoft.CodeAnalysis.CSharp.Scripting
- Microsoft.EntityFrameworkCore.Sqlite
- Microsoft.EntityFrameworkCore.SqlServer
- Microsoft.EntityFrameworkCore.Tools
- Microsoft.VisualStudio.Web.CodeGeneration.Design

3. Instrukcje pierwszego uruchomienia projektu

- Uruchomić Visual Studio 2022.
- Wybrać opcję „Klonuj repozytorium”

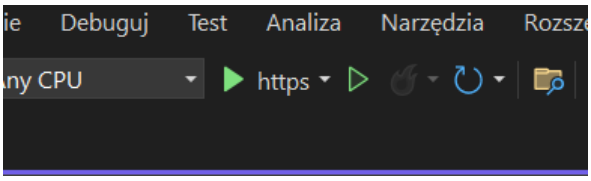


- Wprowadzić ścieżkę do repozytorium na GitHubie.
- Program automatycznie utworzy folder zgodny z nazwą projektu.

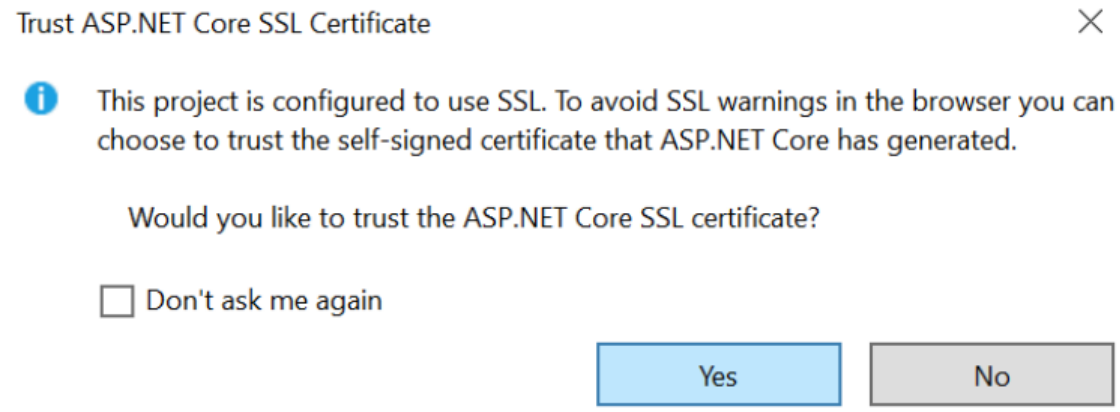


- Otworzenie spowoduje pobranie wszystkich niezbędnych pakietów NuGet w lewym dolnym oknie programu.

- Uruchomić projekt za pomocą F5 lub klikając przycisk https.



- Podczas uruchamiania programu, może zostać wyświetlona informacja o certyfikatach SSL. Można ją zaakceptować, aby uniknąć ostrzeżeń SSL w przeglądarce



4. Opis struktury projektu

Projekt posiada następującą strukturę:

- **wwwroot** – zawiera statyczne pliki, takie jak:
  - **site.css** – arkusz stylów CSS używany w aplikacji.
  - **fonts** – czcionki wykorzystywane w aplikacji.
  - **images** – obrazy używane w aplikacji.
  - **site.js** – plik JavaScript zawierający skrypty dla ikon.
  - **favicon.ico** – ikona strony, wyświetlana na karcie przeglądarki.
- **Areas** – zawiera widoki i kontrolery szkieletowe z **Identity**, odpowiedzialne za zarządzanie użytkownikami, logowaniem, rejestracją oraz innymi funkcjami związanymi z tożsamością.
- **Controllers** – zawiera trzy główne kontrolery aplikacji:
  - **HomeController** – kontroler odpowiedzialny za stronę główną, O grze oraz błędu.
  - **LevelController** – kontroler zarządzający wyświetlaniem listy poziomów, tabel wyników oraz przekierowywaniem do poziomów oraz powtórek.
  - **GameController** – kontroler odpowiedzialny za wyświetlanie poziomu, powtórki oraz obsługę przesyłania oraz weryfikacji wyników.
- **Data** – folder zawierający elementy związane z bazą danych:
  - **ApplicationDbContext** – kontekst bazy danych.
  - **DbSeeder** – klasy do inicjowania i wypełniania bazy danych danymi początkowymi.
  - **Migrations** – folder zawierający migracje bazy danych generowane za pomocą Entity Framework Core.
- **Filter** – folder zawierający filtr blokujący dostęp do nieużywanych stron Razor.
- **Models** – folder zawierający modele danych:
  - **Level** – model reprezentujący poziom gry, zawierający nazwę i dane związane z poziomem.
  - **RedirectViewModel** – model używany do wyświetlania wiadomości i przekierowań użytkownika.
  - **Replay** – model reprezentujący dane powtórek graczy, takie jak wynik, poziom i odpowiedź użytkownika.
- **Views** – folder zawierający widoki aplikacji:
  - **Game** – folder z widokami związanymi z rozgrywką, m.in. "**Play**" i "**Replay**".
  - **Home** – folder z widokami dotyczącymi stron: "**About**", "**Error404**" i "**Index**".
  - **Level** – folder zawierający widoki związane z poziomami: "Index" oraz widok częściowy "**_Ranking**" .
  - **Shared** – folder z widokami współdzielonymi, m.in. **_Layout**, który zawiera wspólne elementy UI, takie jak menu boczne czy stopka.
- **appsettings.json** – plik konfiguracyjny aplikacji.
- **Program.cs** – plik wejściowy aplikacji.

5. Modele

Model Replay - Przechowuje informacje o powtórkach gier użytkowników, w tym wynik, wprowadzone dane oraz powiązanie z poziomem i użytkownikiem.

- Id (int): Unikalny identyfikator powtórki. (Klucz główny)
- LevelId (int): Identyfikator poziomu, dla którego została zapisana powtórka. (Klucz obcy do Level)
- UserId (string): Identyfikator użytkownika, który zapisał powtórkę. (Klucz obcy do IdentityUser)
- User (IdentityUser): Powiązany obiekt użytkownika z tabeli AspNetUsers.
- Input (int): Dane wprowadzone przez użytkownika podczas gry.
- Score (int): Wynik osiągnięty przez użytkownika.
- Level (Level): Powiązany obiekt poziomu.

Model RedirectViewModel - Reprezentuje dane przekazywane do widoku przekierowania, wyświetlając wiadomość oraz link do powrotu.

- Message (string): Wiadomość informująca użytkownika o statusie operacji.
- ReturnUrl (string): Adres URL, do którego użytkownik zostanie przekierowany.

Model Level - Przechowuje dane dotyczące poziomów w grze: nazwę oraz dane poziomu.

- Id (int): Unikalny identyfikator poziomu. (Klucz główny)
- Name (string): Nazwa poziomu widoczna dla graczy.
- Data (string): Dane poziomu, obecnie wyrażenie logiczne definiujące zasady weryfikacji danych dla poziomu.

6. Kontrolery wraz z metodami

Kontroler HomeController

a. Metoda **Index**
   - Metoda HTTP: GET
   - Parametry: Brak
   - Opis:
     Wyświetla stronę główną aplikacji. Przypisuje do ViewData["Title"] wartość "Witamy w Lyoko", która jest używana w widoku.
   - Zwracane dane:
     Widok o nazwie Index.

b. Metoda **About**
   - Metoda HTTP: GET
   - Parametry: Brak
   - Opis:
     Wyświetla stronę informacyjną o grze. Przypisuje do ViewData["Title"] wartość "O grze", która jest używana w widoku.
   - Zwracane dane:
     Widok o nazwie About.

c. Metoda **Error404**
   - Metoda HTTP: GET
   - Parametry: Brak
   - Opis:
     Wyświetla stronę błędu 404 (nie znaleziono strony). Przypisuje do ViewData["Title"] wartość "Error404", która jest używana w widoku.
   - Zwracane dane:
     Widok o nazwie Error404.

Kontroler LevelController

a. Metoda **Index**
   - Metoda HTTP: GET
   - Parametry: Brak
   - Opis:
     Wyświetla listę poziomów gry oraz przypisuje do nich wynik gracza (jeśli istnieje):
     Pobiera wszystkie poziomy z tabeli Levels.
     Identyfikuje bieżącego użytkownika za pomocą User.FindFirstValue(ClaimTypes.NameIdentifier).
     Pobiera wyniki gracza dla każdego poziomu.

Łączy poziomy z wynikami gracza, tworząc nową strukturę danych, która jest przekazywana do widoku.
Dodaje tytuł "Przeglądaj poziomy" do ViewData["Title"].

- Zwracane dane:
  Widok Index z listą poziomów oraz wynikami użytkownika.

b. Metoda **GetRanking**
   - Metoda HTTP: GET
   - Parametry:
     id (int): Identyfikator poziomu, dla którego ranking ma zostać pobrany.
   - Opis:
     Generuje ranking graczy dla wybranego poziomu:
     Pobiera do 20 najlepszych wyników z tabeli Replays dla określonego poziomu (LevelId równe id).
     Wyniki są posortowane malejąco według punktacji (Score).
     Każdy wynik zawiera informacje o użytkowniku (Include(r => r.User)).
   - Zwracane dane:
     Widok częściowy _Ranking zawierający listę najlepszych wyników dla poziomu.

c. Metoda **Play**
   - Metoda HTTP: GET
   - Parametry:
     levelId (int): Identyfikator poziomu, który gracz chce rozegrać.
   - Opis:
     Przekierowuje użytkownika do akcji Play w kontrolerze GameController.
     Wykorzystywane jest przekazanie parametru levelId do kontrolera Game.
   - Zwracane dane:
     Przekierowanie do akcji Play w kontrolerze GameController.

d. Metoda **Replay**
   - Metoda HTTP: GET
   - Parametry:
     replayId (int): Identyfikator powtórki, którą użytkownik chce obejrzeć.
   - Opis:
     Przekierowuje użytkownika do akcji Replay w kontrolerze GameController.
     Wykorzystywane jest przekazanie parametru replayId do kontrolera GameController.
   - Zwracane dane:
     Przekierowanie do akcji Replay w kontrolerze GameController.

Kontroler GameController

a. Metoda **Play**
   - Metoda HTTP: GET
   - Parametry:
     levelId (int): Identyfikator poziomu, który gracz chce rozegrać.
   - Opis:
     Wyświetla poziom, na którym gracz zamierza grać:
     Pobiera poziom z tabeli Levels na podstawie levelId.
     Jeśli poziom nie istnieje, zwraca status Not Found.
     Dodaje nazwę poziomu jako tytuł widoku i przekazuje dane poziomu do widoku.
   - Zwracane dane:
     Widok Play wyświetlający poziom.

b. Metoda **SubmitScore**
   - Metoda HTTP: POST
   - Parametry:
     levelId (int): Identyfikator poziomu, dla którego gracz przesyła odpowiedź.
     input (int): Odpowiedź wprowadzona przez gracza.
   - Opis:
     Przesyła odpowiedź gracza dla danego poziomu:
     Pobiera poziom na podstawie levelId. Jeśli poziom nie istnieje, zwraca status Not Found.
     Sprawdza poprawność wprowadzonej odpowiedzi przy użyciu metody VerifyInput. Jeśli odpowiedź jest niepoprawna, przekierowuje do komunikatu z niewłaściwą odpowiedzią.
     Generuje wynik na podstawie wprowadzonej odpowiedzi i danych poziomu (GenerateScore).
     Dodaje powtórkę gracza, jeśli jeszcze nie istnieje.
     Aktualizuje istniejącą, jeśli nowy wynik jest wyższy niż poprzedni. W przeciwnym wypadku, przekierowuje do komunikatu o niższym wyniku.
     Zapisuje nową powtórkę do tabeli Replays i zwraca komunikat o sukcesie.
   - Zwracane dane:
     Przekierowanie do akcji Redirect z komunikatem o wyniku.

c. Metoda **Replay**
   - Metoda HTTP: GET
   - Parametry:
     replayId (int): Identyfikator powtórki, którą gracz chce obejrzeć.
   - Opis:
     Wyświetla szczegóły zapisanej powtórki gracza:
     Pobiera powtórkę na podstawie replayId wraz z informacjami o użytkowniku i poziomie (Include).
     Jeśli powtórka nie istnieje, zwraca status Not Found.

Dodaje nazwę poziomu do tytułu widoku i przekazuje powtórkę do widoku.

- Zwracane dane:
  Widok Replay wyświetlający szczegóły powtórki.

d. Metoda **Redirect**
   - Metoda HTTP: GET
   - Parametry:
     message (string): Wiadomość wyświetlana użytkownikowi.
     returnUrl (string): Adres URL, do którego użytkownik zostaje przekierowany.
   - Opis:
     Wyświetla stronę z komunikatem, licznikiem oraz przyciskiem przekierowującym użytkownika na określoną stronę.
     Przekazuje dane do widoku MessageRedirect.
   - Zwracane dane:
     Widok MessageRedirect z komunikatem i linkiem do powrotu.

e. Metoda **VerifyInput**
   - Metoda HTTP: Brak (metoda prywatna).
   - Parametry:
     levelData (string): Dane poziomu przechowywane w bazie.
     input (int): Odpowiedź wprowadzona przez gracza.
   - Opis:
     Weryfikuje poprawność wprowadzonej odpowiedzi na podstawie wyrażenia logicznego przechowywanego w levelData.
     Wyrażenie jest przetwarzane za pomocą CSharpScript.EvaluateAsync.
     Jeśli wyrażenie jest nieprawidłowe lub nie może być ocenione, zwraca false.
   - Zwracane dane:
     (bool): true jeśli wynik jest poprawny, false w przeciwnym wypadku.

f. Metoda **GenerateScore**
   - Metoda HTTP: Brak (metoda prywatna).
   - Parametry:
   - input (int): Odpowiedź wprowadzona przez gracza.
   - levelData (string): Dane poziomu przechowywane w bazie.
   - Opis:
     Generuje wynik na podstawie odpowiedzi wprowadzonej przez gracza pomnożonej przez 4576.
   - Zwracane dane:
     (int): Obliczony wynik gracza.

7. Opis systemu użytkowników

Role:

- Użytkownik niezalogowany
- Użytkownik zalogowany

Uprawnienia użytkowników:

- **Użytkownicy niezalogowani**:
    a. Mają dostęp wyłącznie do strony głównej, strony „O grze”, strony rejestracji oraz logowania.
- **Użytkownicy zalogowani** ponadto:
    a. Mogą przeglądać listę poziomów.
    b. Mogą przeglądać do 20 najlepszych wyników dla danego poziomu.
    c. Mogą przechodzić poziomy, przesyłać wyniki oraz przeglądać swoje wyniki.
    d. Mogą odtwarzać do 20 najlepszych powtórek dla danego poziomu.
    e. Mogą zmieniać hasło oraz usunąć swoje konto.
    f. Mogą się wylogować.

Informacje:

- **Powiązane z użytkownikiem**:
    a. Dane dotyczące użytkownika, email (nazwa) oraz hasło
    b. Powtórki użytkownika na danym poziomie
- **Globalne informacje**:
    a. Informacje o grze (opis, zasady) są ogólnodostępne.
    b. Poziomy: Lista dostępnych poziomów w bazie danych jest widoczna dla wszystkich zalogowanych użytkowników.
    c. Wyniki: 20 najlepszych wyników wraz z nazwą użytkownika dla danego poziomu jest widoczna dla każdego zalogowanego użytkownika
    d. Powtórki: 20 powtórek z najlepszymi wynikami dla danego poziomu może zostać odtworzone przez każdego zalogowanego użytkownika.

8. Charakterystyka najciekawszych funkcjonalności

System rankingowy

System wyświetla listę 20 najlepszych wyników graczy dla każdego poziomu. W bazie danych zapisywany jest jedynie najwyższy wynik osiągnięty przez danego gracza.

Cel: Motywowanie graczy do rywalizacji.

Przesyłanie oraz weryfikacja danych rozgrywki po stronie serwera

Użytkownicy mogą przesyłać swoje odpowiedzi na serwer, który je weryfikuje, sprawdzając poprawność na podstawie matematycznych wyrażeń zapisanych w danych poziomu, przyznaje odpowiedni wynik oraz zapisuje powtórkę w bazie danych.

Cel: Zapewnienie uczciwej rozgrywki oraz przechowywanie najlepszych powtórek do późniejszego odtwarzania.

Możliwość odtwarzania 20 najlepszych rozgrywek dla poziomu

System pozwala użytkownikom na odtworzenie najlepszych powtórek dla danego poziomu, co pozwala im zobaczyć, jak osiągnięto najwyższe wyniki.

Cel: Nauka poprzez obserwację najlepszych rozwiązań.