WYŻSZA SZKOŁA
INFORMATYKI i ZARZĄDZANIA
z siedzibą w Rzeszowie

KOLEGIUM INFORMATYKI STOSOWANEJ

Kierunek: INFORMATYKA

Specjalność: Technologie internetowe i mobilne

Marcin Chruściel
Nr albumu studenta w67120

System Zarządzania Graczami (Player Management System)

Prowadzący: mgr inż. Ewa Żesławska

Projekt Programowanie Obiektowe C#

Rzeszów 2024

Spis treści

Wstęp **3**

1 Opis Projektu **4**
1.1 Założenia Aplikacji 4

2 Opis Bazy danych **6**
2.1 Encje i Atrybuty 6
2.2 Związki i Krotności 7
2.3 Diagram ERD 8

3 Struktura projektu **10**
3.1 Klasa GameStats i GameStatsService 10
3.2 Klasa LoginHistory i LoginHistoryService 11
3.3 Klasa PasswordHistory i PasswordHistoryService 12
3.4 Klasa PlayerAccountService 12
3.5 Klasa PlayerDetails i PlayerDetailsService 13
3.6 Klasa PlayerProgress i PlayerProgressService 14
3.7 Klasa Program 15
3.8 Diagram 15

4 Przedstawienie aplikacji **17**

5 Podsumowanie Projektu **23**
5.1 Diagram Ganta 23
5.2 Repozytorium 24
5.3 Podsumowanie 24

Bibliografia **26**

Spis rysunków **27**

Wstęp

W świecie gier komputerowych, zarządzanie zasobami ludzkimi w firmach gamingowych stało się kluczowym elementem sukcesu. Aplikacja GameHRMS (Gaming Human Resource Management System) została stworzona, aby sprostać potrzebom branży gamingowej w zakresie zarządzania personelem. Celem systemu jest zapewnienie efektywnego narzędzia do rekrutacji, zarządzania pracownikami, monitorowania ich postępów i analizowania danych związanych z grami oraz wynikami graczy.

GameHRMS umożliwia firmom gamingowym łatwe śledzenie i zarządzanie pracownikami od momentu rekrutacji aż po ich rozwój w ramach organizacji. Dzięki integracji z bazą danych gier i statystyk graczy, menedżerowie mogą podejmować świadome decyzje dotyczące zespołów projektowych oraz planować strategie rozwoju produktów.

Rozdział 1

Opis Projektu

1.1 Założenia Aplikacji

Opis założenia Projektu

Projekt GameHRMS ma na celu stworzenie kompleksowego systemu zarządzania zasobami ludzkimi w grach komputerowych przy użyciu języka C#. Aplikacja ma umożliwiać skuteczne zarządzanie kontami graczy, śledzenie ich postępu w grze, analizę statystyk oraz zapewnienie odpowiedniego poziomu zabezpieczeń.

Cele Projektu

- **Utworzenie i Zarządzanie Kontami Graczy:** Aplikacja umożliwi tworzenie oraz skuteczne zarządzanie kontami graczy.
- **Personalizacja Profili Graczy:** Zapewnienie możliwości personalizacji profili.
- **Monitorowanie Historii Logowań:** Skuteczne śledzenie historii logowań graczy w celu zabezpieczenia kont przed nieautoryzowanym dostępem
- **Analiza Statystyk Gry:** Gromadzenie oraz prezentacja statystyk związanych z grą, dostarczając informacji o postępie graczy.
- **Mozliwość tworzenia/edycji kont:** Mozliwość tworzneia oraz edycji kont, zmian w statystykach, czy miejscach logowania.

Wymagania Funkcjonalne

- **Zarządzanie Danymi Osobowymi:** Gracze mają możliwość uzupełnienia swojego profilu o dane osobowe, takie jak imię, nazwisko, data urodzenia, adres i numer telefonu.
- **Historia Logowania:** Aplikacja przedstawia historię logowań, zawierając informacje takie jak data, adres oraz imie i nazwisko.
- **Historia Haseł:** System przechowuje historię haseł gracza, z zachowaniem bezpieczeństwa, posiada również date zmiany hasła.
- **Postęp w Grze:** Możliwość śledzenia postępu w grze, obejmującego ilość wszystkich meczy, wygrane, przegrane i średni wynik.

Wymagania Niefunkcjonalne

- **Wydajność:** Aplikacja powinna obsługiwać duże ilości danych, zapewniając szybki dostęp do informacji o graczach.

- **Responsywność:** Interfejs użytkownika powinien być responsywny, umożliwiając wygodne korzystanie.
- **Skalowalność:** Aplikacja powinna być łatwo skalowalna, umożliwiając dostosowanie się do rosnącej liczby graczy.
- **Integracja:** System powinien być łatwo integrowalny z innymi systemami, takimi jak systemy płatności czy platformy do dystrybucji gier.

Rozdział 2

Opis Bazy danych

Wstęp do Dokumentacji Bazy Danych

Celem bazy danych w GameHRMS (Game Human Resources Management System) jest umożliwienie skutecznego zarządzania wszystkimi informacjami związanymi z graczami, ich aktywnością w grach, historią logowania, postępami w grach itp.

2.1 Encje i Atrybuty

1. **PlayerAccount**
   - PlayerID (PK, int): Unikalny identyfikator gracza.
   - Username (varchar(50)): Nazwa użytkownika gracza.
   - Password (varchar(255)): Hasło gracza (hash).
   - Email (varchar(100)): Adres e-mail gracza.
   - RegistrationDate (datetime): Data rejestracji konta gracza.
   - LastLoginDate (datetime): Data ostatniego logowania gracza.
   - AccountStatus (varchar(20)): Status konta (np. "Active").
   - AccountType (varchar(20)): Typ konta (np. "Regular"lub "Admin")..
2. **PlayerDetails**
   - PlayerID (PK, int): Unikalny identyfikator gracza.
   - FirstName (varchar(50)): Imię gracza.
   - LastName (varchar(50)): Nazwisko gracza.
   - DateOfBirth (date): Data urodzenia gracza.
   - Address (text): Adres gracza.
   - PhoneNumber (varchar(20)): Numer telefonu gracza.
3. **LoginHistory**
   - LogID (PK, int): Unikalny identyfikator historii logowania.
   - PlayerID (FK, int): Klucz obcy wskazujący na gracza, którego historia logowania dotyczy.
   - LoginTime (datetime): Data i czas logowania.
   - IPAddress (varchar(15)): Adres IP z którego logowano.
   - Location (varchar(100)): Lokalizacja logowania.

- DeviceType (varchar(50)): Typ urządzenia, z którego logowano.

4. **PasswordHistory**
    - PasswordID (PK, int): Unikalny identyfikator historii haseł.
    - PlayerID (FK, int): Klucz obcy wskazujący na gracza, którego historia haseł dotyczy.
    - PasswordHash (varchar(255)): Hasło gracza (hash).
    - ChangeTime (datetime): Data i czas zmiany hasła.
5. **AccountSecurity**
    - SecurityID (PK, int): Unikalny identyfikator zabezpieczeń konta.
    - PlayerID (FK, int): Klucz obcy wskazujący na gracza, którego zabezpieczenia dotyczą.
    - TwoFactorAuth (boolean): Czy autoryzacja dwuetapowa jest włączona.
    - SecurityQuestion (text): Pytanie zabezpieczające konto.
    - SecurityAnswer (varchar(255)): Odpowiedź na pytanie zabezpieczające konto.
6. **PlayerProgress**
    - ProgressID (PK, int): Unikalny identyfikator postępu gracza.
    - PlayerID (FK, int): Klucz obcy wskazujący na gracza, którego postęp dotyczy.
    - GameLevel (int): Poziom gry osiągnięty przez gracza.
    - ExperiencePoints (int): Punkty doświadczenia gracza.
    - Achievements (text): Osiągnięcia zdobyte przez gracza.
7. **GameStats**
    - • StatsID (PK, int): Unikalny identyfikator statystyk gry.
    - • PlayerID (FK, int): Klucz obcy wskazujący na gracza, którego statystyki gry dotyczą.
    - • TotalGamesPlayed (int): Łączna liczba rozegranych gier.
    - • TotalWins (int): Łączna liczba wygranych.
    - • TotalLosses (int): Łączna liczba porażek.
    - • AverageScore (float): Średni wynik gracza.

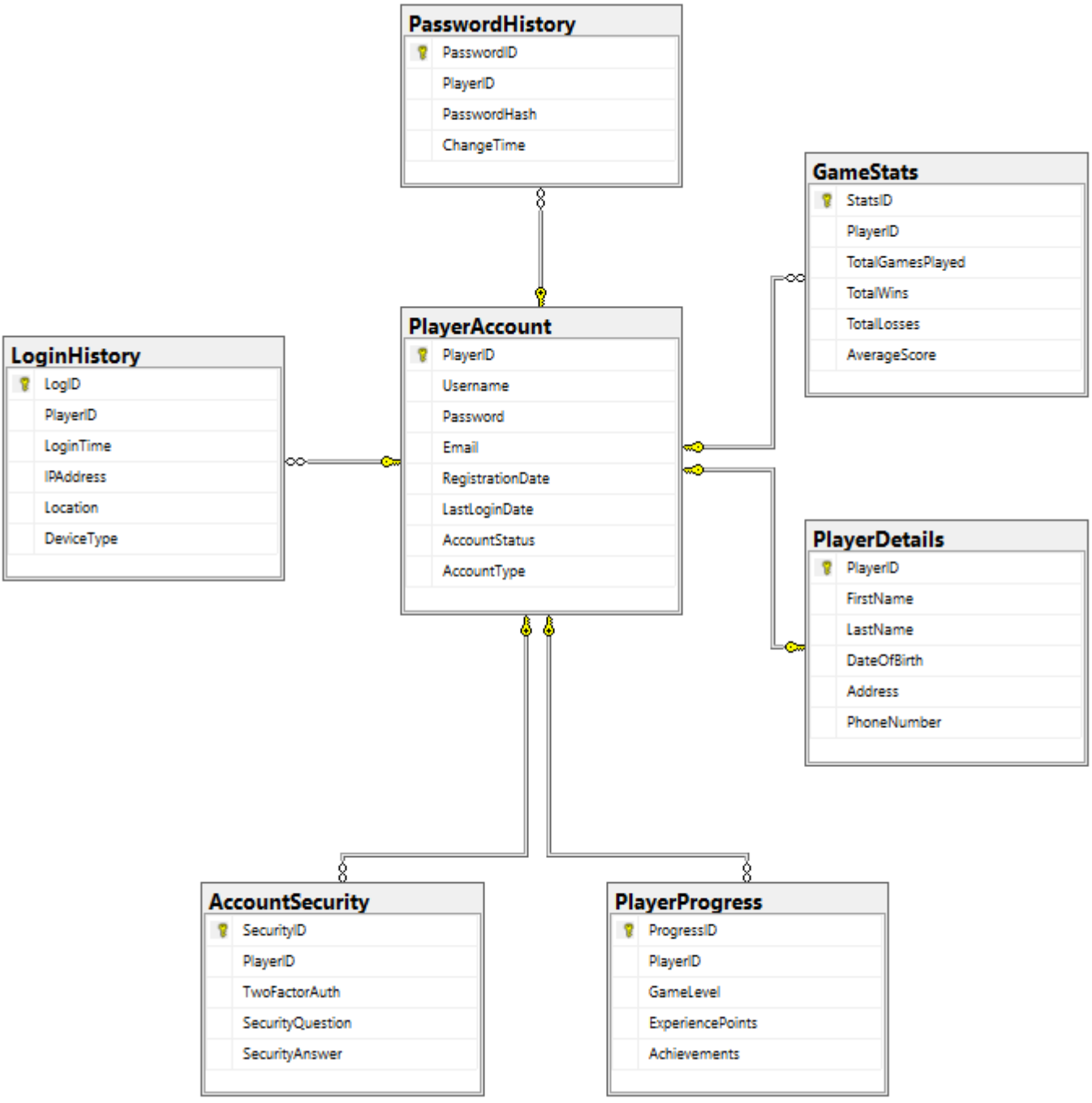
2.2 Związki i Krotności

1. **Gracz - Historia Logowań (Player - LoginHistory):**
    - Związek: Jeden gracz może mieć wiele wpisów w historii logowań.
    - Krotność: Relacja jeden do wielu (1:N).
2. **Gracz - Historia Haseł (Player - PasswordHistory):**
    - Związek: Jeden gracz może mieć wiele wpisów w historii haseł.
    - Krotność: Relacja jeden do wielu (1:N).
3. **Gracz - Zabezpieczenia Konta (Player - AccountSecurity):**
    - Związek: Jeden gracz jest powiązany z jednym zestawem zabezpieczeń konta.
    - Krotność: Relacja jeden do jeden (1:1).

4. **Gracz - Dane Personalne (Player - PlayerDetails)**
    - Związek: Jeden gracz jest powiązany z jednym zestawem danych personalnych.
    - Krotność: Relacja jeden do jeden (1:1).
5. **Gracz - Postęp w Grze (Player - PlayerProgress):**
    - Związek: Jeden gracz jest powiązany z jednym zestawem postępu w grze.
    - Krotność: Relacja jeden do jeden (1:1).
6. **Gracz - Statystyki Gry (Player - GameStats):**
    - Związek: Jeden gracz jest powiązany z jednym zestawem statystyk gry.
    - Krotność: Relacja jeden do jeden (1:1).

2.3 Diagram ERD

Na przedstawionym diagramie encji i związków (ERD) zostały uwydatnione struktury danych dla systemu zarządzania graczami w grach komputerowych (GameHRMS). Diagram precyzyjnie określa relacje między encjami, takimi jak PlayerAccount, PlayerDetails, LoginHistory, PasswordHistory, AccountSecurity, PlayerProgress oraz GameStats. W sposób czytelny przedstawia strukturę bazodanową, uwzględniając kluczowe atrybuty każdej encji, oraz istotne relacje, takie jak te o charakterze 'jeden do wielu' oraz 'wiele do wielu'. Taka reprezentacja umożliwia skuteczne śledzenie postępów graczy, analizę statystyk gry oraz zarządzanie zabezpieczeniami kont, stanowiąc solidną podstawę bazy danych GameHRMS.



Rysunek 2.1: Diagram ERD

Rozdział 3

Struktura projektu

3.1 Klasa GameStats i GameStatsService

Klasa `GameStats`, zdefiniowana w przestrzeni nazw `GAMEHrms0.1`, reprezentuje statystyki gier gracza w systemie GameHRMS, takie jak liczba rozegranych gier, liczba wygranych, przegranych oraz średni wynik. Natomiast klasa `GameStatsService` jest odpowiedzialna za obsługę operacji związanych ze statystykami gier, zapewniając funkcje do tworzenia, odczytywania, aktualizowania i usuwania danych statystycznych gracza.

Struktura

- **Namespace:** GAMEHrms0.1
- **Zależności:** System, System.Collections.Generic, System.Linq, System.Text, System.Threading.Tasks

Klasa GameStats

Konstruktor: `GameStats(int statsID, int playerID, int totalGamesPlayed, int totalWins, int totalLosses, double averageScore)`

Właściwości:

- `StatsID` (publiczny int): Identyfikator statystyk.
- `PlayerID` (publiczny int): Identyfikator gracza.
- `TotalGamesPlayed` (publiczny int): Całkowita liczba rozegranych gier.
- `TotalWins` (publiczny int): Całkowita liczba zwycięstw.
- `TotalLosses` (publiczny int): Całkowita liczba porażek.
- `AverageScore` (publiczny double): Średni wynik.

Klasa GameStatsService

Metody:

- `void Create(GameStats stats)`: Tworzy nowe statystyki gry dla gracza.
- `GameStats Read(int statsID)`: Odczytuje statystyki gry na podstawie identyfikatora.
- `void Update(GameStats stats)`: Aktualizuje istniejące statystyki gry.
- `void Delete(int statsID)`: Usuwa statystyki gry na podstawie identyfikatora.

Opis:

Klasa `GameStats` reprezentuje statystyki graczy w systemie GameHRMS. Zawiera informacje dotyczące liczby rozegranych gier, liczby zwycięstw, liczby porażek i średniego wyniku. Klasyfikuje się do kategorii "Dane"i zapewnia podstawową funkcjonalność związana z gromadzeniem i przechowywaniem statystyk graczy. Klasa `GameStatsService` zapewnia interfejs do zarządzania statystykami gry, umożliwiając tworzenie, odczytywanie, aktualizowanie i usuwanie danych dotyczących statystyk graczy.

3.2 Klasa LoginHistory i LoginHistoryService

Klasa `LoginHistory` przechowuje informacje dotyczące historii logowań użytkowników, w tym identyfikator logowania, identyfikator użytkownika, datę i godzinę logowania oraz adres IP. Z kolei `LoginHistoryService` oferuje funkcjonalności do zarządzania tą historią, w tym tworzenie nowych wpisów, odczytywanie, aktualizowanie i usuwanie istniejących rekordów.

Struktura

- **Namespace:** GAMEHrms0.1
- **Zależności:** System, System.Collections.Generic, System.Linq, System.Text, System.Threading.Tasks

Klasa LoginHistory

Konstruktor: `LoginHistory(int loginID, int userID, DateTime loginDateTime, string ipAddress)`

Właściwości:

- `LoginID` (publiczny int): Identyfikator logowania.
- `UserID` (publiczny int): Identyfikator użytkownika.
- `LoginDateTime` (publiczny DateTime): Data i godzina logowania.
- `IPAddress` (publiczny string): Adres IP logującego się użytkownika.

Klasa LoginHistoryService

Metody:

- `void Create(LoginHistory login)`: Tworzy nowy wpis historii logowania.
- `LoginHistory Read(int loginID)`: Odczytuje historię logowania na podstawie identyfikatora.
- `void Update(LoginHistory login)`: Aktualizuje istniejący wpis historii logowania.
- `void Delete(int loginID)`: Usuwa wpis historii logowania na podstawie identyfikatora.

Opis

Klasa `LoginHistory` i `LoginHistoryService` umożliwiają zarządzanie i śledzenie historii logowań użytkowników w systemie GameHRMS. Dzięki tym klasom, system może monitorować aktywność użytkowników oraz wspierać funkcje związane z bezpieczeństwem i audytem.

3.3 Klasa PasswordHistory i PasswordHistoryService

Klasa `PasswordHistory` rejestruje zmiany haseł użytkowników, przechowując informacje takie jak identyfikator historii, identyfikator użytkownika, datę zmiany oraz nowe hasło.
`PasswordHistoryService` zapewnia metody do zarządzania tą historią, w tym dodawanie nowych wpisów i przeglądanie historii zmian haseł.

Struktura

- **Namespace:** GAMEHrms0.1
- **Zależności:** System, System.Collections.Generic, System.Linq, System.Text, System.Threading.Tasks

Klasa PasswordHistory

Konstruktor: PasswordHistory(int historyID, int userID, DateTime changeDateTime, string newPassword)

Właściwości:

- `HistoryID` (publiczny int): Identyfikator historii hasła.
- `UserID` (publiczny int): Identyfikator użytkownika.
- `ChangeDateTime` (publiczny DateTime): Data i godzina zmiany hasła.
- `NewPassword` (publiczny string): Nowe hasło.

Klasa PasswordHistoryService

Metody:

- `void Create(PasswordHistory history)`: Dodaje nowy wpis do historii zmiany hasła.
- `PasswordHistory Read(int historyID)`: Odczytuje wpis historii zmiany hasła na podstawie identyfikatora.
- `void Update(PasswordHistory history)`: Aktualizuje wpis w historii zmiany hasła.
- `void Delete(int historyID)`: Usuwa wpis z historii zmiany hasła na podstawie identyfikatora.

Opis

Klasa `PasswordHistory` i `PasswordHistoryService` umożliwiają śledzenie historii zmian haseł użytkowników, co jest kluczowe dla utrzymania bezpieczeństwa konta. Pozwalają na monitorowanie i audytowanie zmian haseł, wspierając polityki bezpieczeństwa dotyczące zarządzania hasłami.

3.4 Klasa PlayerAccountService

Klasa `PlayerAccountService` zapewnia metody do zarządzania kontami graczy, w tym tworzenie nowych kont, aktualizację danych konta, usuwanie kont oraz autentykację i autoryzację użytkowników.

Struktura

- **Namespace:** GAMEHrms0.1
- **Zależności:** System, System.Collections.Generic, System.Linq, System.Text, System.Threading.Tasks

Metody

- `void Create(PlayerAccount account)`: Tworzy nowe konto gracza.
- `PlayerAccount Read(int accountID)`: Odczytuje konto gracza na podstawie identyfikatora.
- `void Update(PlayerAccount account)`: Aktualizuje istniejące konto gracza.
- `void Delete(int accountID)`: Usuwa konto gracza na podstawie identyfikatora.

Opis

Klasa `PlayerAccountService` umożliwia efektywne zarządzanie kontami graczy w systemie GameHRMS. Oferuje kompleksowy zestaw operacji do obsługi życiowego cyklu konta gracza, od jego utworzenia po usunięcie.

3.5 Klasa PlayerDetails i PlayerDetailsService

Klasa `PlayerDetails` przechowuje szczegółowe informacje o graczach, takie jak imię, nazwisko, adres e-mail, numer telefonu i wiek. Klasa `PlayerDetailsService` zapewnia funkcjonalności do zarządzania tymi szczegółowymi danymi gracza.

Struktura

- **Namespace:** GAMEHrms0.1
- **Zależności:** System, System.Collections.Generic, System.Linq, System.Text, System.Threading.Tasks

Klasa PlayerDetails

Konstruktor: `PlayerDetails(int playerID, string firstName, string lastName, string email, string phoneNumber, int age)`

Właściwości:

- `PlayerID` (publiczny int): Identyfikator gracza.
- `FirstName` (publiczny string): Imię gracza.
- `LastName` (publiczny string): Nazwisko gracza.
- `Email` (publiczny string): Adres e-mail gracza.
- `PhoneNumber` (publiczny string): Numer telefonu gracza.
- `Age` (publiczny int): Wiek gracza.

Klasa PlayerDetailsService

Metody:

- `void Create(PlayerDetails details)`: Tworzy nowe szczegółowe informacje o graczu.
- `PlayerDetails Read(int playerID)`: Odczytuje szczegółowe informacje o graczu na podstawie identyfikatora.

- `void Update(PlayerDetails details)`: Aktualizuje istniejące szczegółowe informacje o graczu.
- `void Delete(int playerID)`: Usuwa szczegółowe informacje o graczu na podstawie identyfikatora.

Opis

Klasy `PlayerDetails` i `PlayerDetailsService` zapewniają strukturę i logikę niezbędną do zarządzania szczegółowymi informacjami o graczach w systemie GameHRMS. Umożliwiają one efektywne przechowywanie, odczytywanie, aktualizowanie i usuwanie danych osobowych graczy.

3.6 Klasa PlayerProgress i PlayerProgressService

Klasa `PlayerProgress` śledzi postępy graczy w grze, rejestrując takie informacje jak osiągnięte poziomy, zdobyte punkty i czas gry. `PlayerProgressService` oferuje funkcjonalności do zarządzania tymi danymi postępu.

Struktura

- **Namespace:** GAMEHrms0.1
- **Zależności:** System, System.Collections.Generic, System.Linq, System.Text, System.Threading.Tasks

Klasa PlayerProgress

Konstruktor: `PlayerProgress(int progressID, int playerID, int levelNumber, int points, TimeSpan gameTime)`

Właściwości:

- `ProgressID` (publiczny int): Identyfikator postępu.
- `PlayerID` (publiczny int): Identyfikator gracza.
- `LevelNumber` (publiczny int): Numer osiągniętego poziomu.
- `Points` (publiczny int): Zdobyte punkty.
- `GameTime` (publiczny TimeSpan): Czas spędzony w grze.

Klasa PlayerProgressService

Metody:

- `void Create(PlayerProgress progress)`: Rejestruje nowy postęp gracza.
- `PlayerProgress Read(int progressID)`: Odczytuje postęp gracza na podstawie identyfikatora.
- `void Update(PlayerProgress progress)`: Aktualizuje postęp gracza.
- `void Delete(int progressID)`: Usuwa postęp gracza na podstawie identyfikatora.

Opis

Klasa `PlayerProgress` wraz z `PlayerProgressService` umożliwiają zarządzanie postępami graczy w systemie GameHRMS. Ułatwiają one monitorowanie i zarządzanie osiągnięciami graczy, promując motywację i zaangażowanie w grze.

3.7 Klasa Program

Klasa `Program` stanowi punkt wejścia dla aplikacji GameHRMS. Jest odpowiedzialna za inicjalizację środowiska aplikacji, przetwarzanie argumentów wejściowych oraz zarządzanie głównym przepływem sterowania programu. Dzięki niej możliwe jest uruchomienie wszystkich komponentów systemu oraz zapewnienie interakcji użytkownika z aplikacją.

Struktura

- **Namespace:** GAMEHrms0.1
- **Zależności:** System, pozostałe klasy systemu GameHRMS

Metoda główna

- `static void Main(string[] args)`: Jest to główna metoda aplikacji, punkt wejścia, który uruchamia aplikację i inicjuje interakcję z użytkownikiem.

Opis

Klasa `Program` jest sercem aplikacji GameHRMS, uruchamiając i koordynując działanie wszystkich jej komponentów. W metodzie `Main` dokonuje się inicjalizacji niezbędnych zasobów, tworzenia instancji głównych klas serwisowych oraz uruchamiania pętli zdarzeń (jeśli aplikacja jest interaktywna). Ta klasa może również obsługiwać argumenty wejściowe, umożliwiając konfigurację działania programu z poziomu linii poleceń.

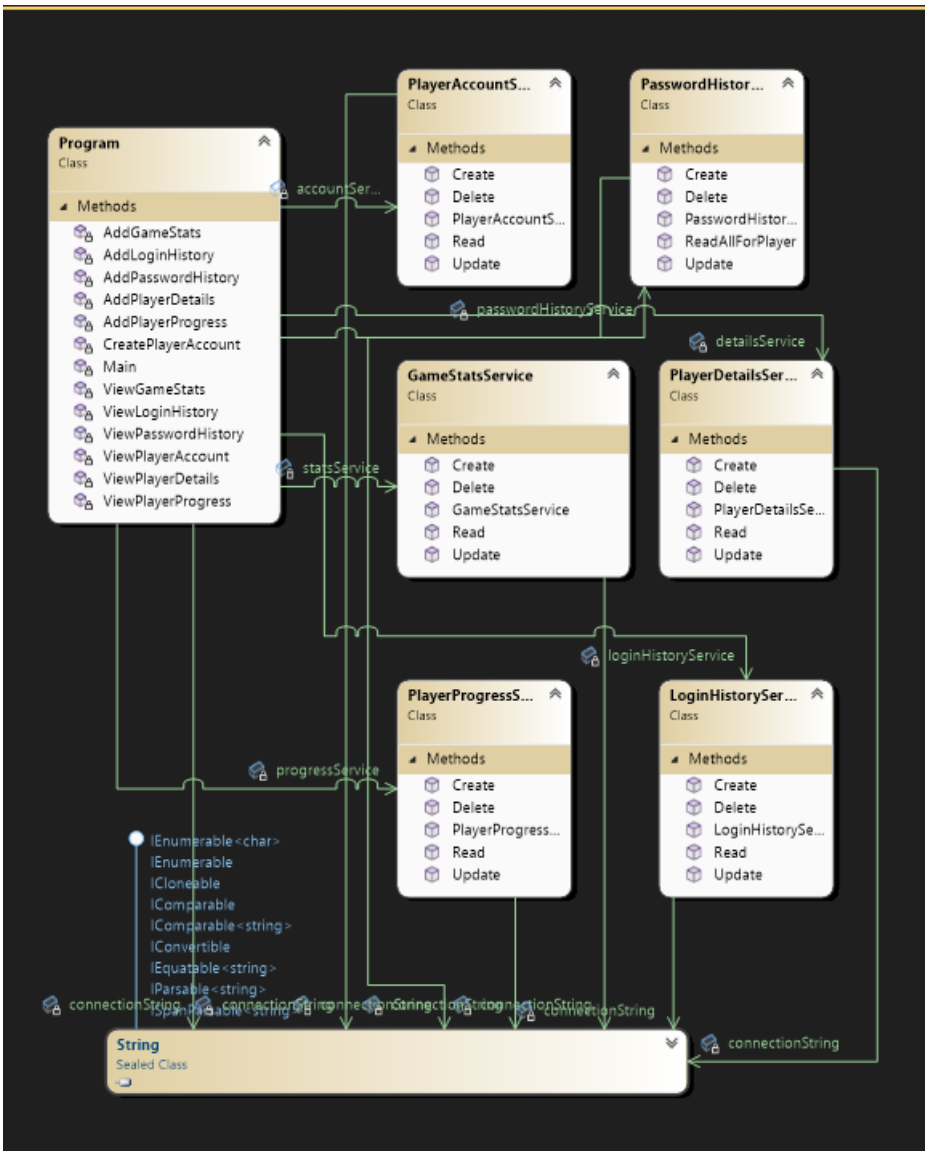
3.8 Diagram

Diagram klas systemu GameHRMS prezentuje strukturę logiczną oraz powiązania pomiędzy głównymi komponentami tego systemu. Pozwala on na zrozumienie organizacji i relacji między różnymi funkcjonalnościami oraz encjami, takimi jak klasy, interfejsy i ich wzajemne zależności.

Dzięki diagramowi klas możemy łatwo zidentyfikować główne komponenty systemu, takie jak klasy reprezentujące użytkowników (PlayerAccount, PlayerDetails), zarządzanie statystykami gry (GameStats, GameStatsService), historię logowania (LoginHistory, LoginHistoryService), historię zmian hasła (PasswordHistory, PasswordHistoryService) oraz postępy graczy (PlayerProgress, PlayerProgressService). Dodatkowo, diagram ten może obejmować klasę Program, która pełni rolę punktu wejścia do aplikacji.

Wymagania systemowe:

- Środowisko wykonawcze .NET Framework w wersji 4.7.2 lub nowszej.
- Serwer baz danych SQL Server w wersji 2016 lub nowszej.
- System operacyjny Windows 10 lub nowszy.
- Procesor Intel Core i5 lub równoważny AMD.
- Co najmniej 8 GB pamięci RAM.
- Wolne miejsce na dysku twardym: minimum 1 GB.



Rysunek 3.1: Diagram klasy Program

Rozdział 4

Przedstawienie aplikacji

Menu główne

Menu główne prezentowane w aplikacji konsolowej GameHRMS jest interfejsem użytkownika, który umożliwia interakcję z systemem przez wykonanie różnych poleceń.
Glowne Menu posiada 12 mozliwych opcji:

Rysunek 4.1: Menu główne

Create Player Account

Ta opcja pozwala użytkownikowi na utworzenie nowego konta gracza w systemie. Po jej wybraniu użytkownik prawdopodobnie zostanie poproszony o podanie danych niezbędnych do rejestracji, takich jak nazwa użytkownika, hasło, adres e-mail itd.

Rysunek 4.2: Create Player Account

View Player Account

Po wybraniu tej opcji użytkownik może wyświetlić szczegóły konta gracza. To może obejmować wyświetlanie informacji o koncie, takich jak nazwa użytkownika, poziom, postęp w grze, statystyki i inne powiązane dane.

```
Enter Player ID:
2
Username: player2, Email: player2@example.com

Choose an option:
1: Create Player Account
2: View Player Account
3: Add Player Details
4: View Player Details
5: Add Player Progress
6: View Player Progress
7: Add Game Stats
8: View Game Stats
9: Add Login History
10: View Login History
11: Add Password History
12: View Password History
0: Exit
```

Rysunek 4.3: View Player Account

Add Player Details

Opcja ta pozwala na dodanie lub edycję szczegółowych informacji o graczu, takich jak imię, nazwisko, wiek, numer telefonu, adres e-mail itp. To rozszerza profil gracza poza podstawowe informacje logowania.

```
Adding Player Details...
Enter Player ID: 5
Enter First Name: Marcin
Enter Last Name: Adam
Enter Date of Birth (YYYY-MM-DD): 2003-03-11
Enter Address: marcin@w67120.com
Enter Phone Number: 111222333
Details added successfully.

Choose an option:
1: Create Player Account
2: View Player Account
3: Add Player Details
4: View Player Details
5: Add Player Progress
6: View Player Progress
7: Add Game Stats
8: View Game Stats
9: Add Login History
10: View Login History
11: Add Password History
12: View Password History
0: Exit
```

Rysunek 4.4: Add Player Details

View Player Details

Umożliwia użytkownikowi przeglądanie już zapisanych szczegółów o graczu. Przydatne do weryfikacji danych osobowych lub do przeglądania aktualnego profilu gracza.

```
Enter Player ID to view details:
7
Name: Marcin Adam, Address: marcin@w67120.com

Choose an option:
1: Create Player Account
2: View Player Account
3: Add Player Details
4: View Player Details
5: Add Player Progress
6: View Player Progress
7: Add Game Stats
8: View Game Stats
9: Add Login History
10: View Login History
11: Add Password History
12: View Password History
0: Exit
```

Rysunek 4.5: View Player Details

Add Player Progress

Dzięki tej opcji użytkownik może zarejestrować nowe osiągnięcia lub postępy gracza, takie jak ukończenie poziomu, zdobycie określonej liczby punktów lub osiągnięcie innych kamieni milowych w grze.

```
Adding Player Progress...
Enter Player ID: 5
Enter Game Level: 12
Enter Experience Points: 4600
Progress added successfully.

Choose an option:
1: Create Player Account
2: View Player Account
3: Add Player Details
4: View Player Details
5: Add Player Progress
6: View Player Progress
7: Add Game Stats
8: View Game Stats
9: Add Login History
10: View Login History
11: Add Password History
12: View Password History
0: Exit
```

Rysunek 4.6: Add Player Progress

View Player Progress

Użytkownik może wyświetlić zapisane informacje o postępie gracza, co może obejmować osiągnięte poziomy, zdobyte punkty, czas gry i inne istotne statystyki.

```
Enter Progress ID to view progress:
5
PlayerID: 5,Player Level 12, Player Exp: 6000,Achivements: Found hidden treasure, Reached level 12

Choose an option:
1: Create Player Account
2: View Player Account
3: Add Player Details
4: View Player Details
5: Add Player Progress
6: View Player Progress
7: Add Game Stats
8: View Game Stats
9: Add Login History
10: View Login History
11: Add Password History
12: View Password History
0: Exit
```

Rysunek 4.7: View Player Progress

Add Game Stats

Ta funkcja umożliwia dodanie nowych statystyk gry dla gracza, takich jak liczba wygranych, liczba przegranych, łączny czas gry, czy inne istotne metryki.

```
Adding Game Stats...
Enter Player ID: 5
Enter Total Games Played: 25
Enter Total Wins: 12
Enter Total Losses: 13
Enter Average Score: 2450
Game Stats added successfully.

Choose an option:
1: Create Player Account
2: View Player Account
3: Add Player Details
4: View Player Details
5: Add Player Progress
6: View Player Progress
7: Add Game Stats
8: View Game Stats
9: Add Login History
10: View Login History
11: Add Password History
12: View Password History
0: Exit
```

Rysunek 4.8: Add Game Stats

View Game Stats

Opcja ta pozwala na przeglądanie zgromadzonych danych statystycznych z gier, umożliwiając analizę wyników i osiągnięć gracza w ramach systemu GameHRMS.

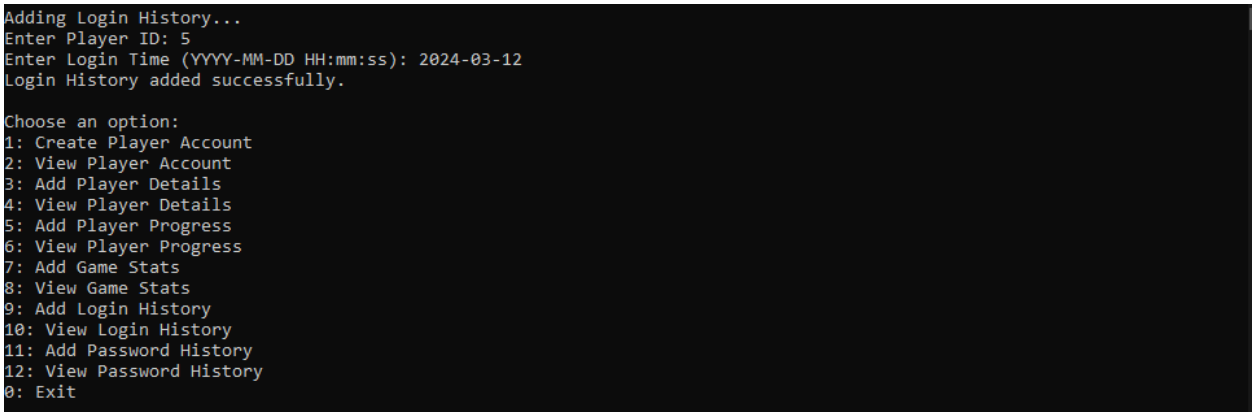
```
Enter Stats ID to view game stats:
4
PlayerID: 4, TotalGamesPlayed: 40, TotalWins: 25, TotalLosses: 15, AverageScore: 79,6

Choose an option:
1: Create Player Account
2: View Player Account
3: Add Player Details
4: View Player Details
5: Add Player Progress
6: View Player Progress
7: Add Game Stats
8: View Game Stats
9: Add Login History
10: View Login History
11: Add Password History
12: View Password History
0: Exit
```

Rysunek 4.9: View Game Stats

Add Login History

Funkcja ta pozwala na rejestrowanie nowych wpisów historii logowania, co może obejmować datę, czas i lokalizację logowania dla każdego gracza



Rysunek 4.10: Add Login History

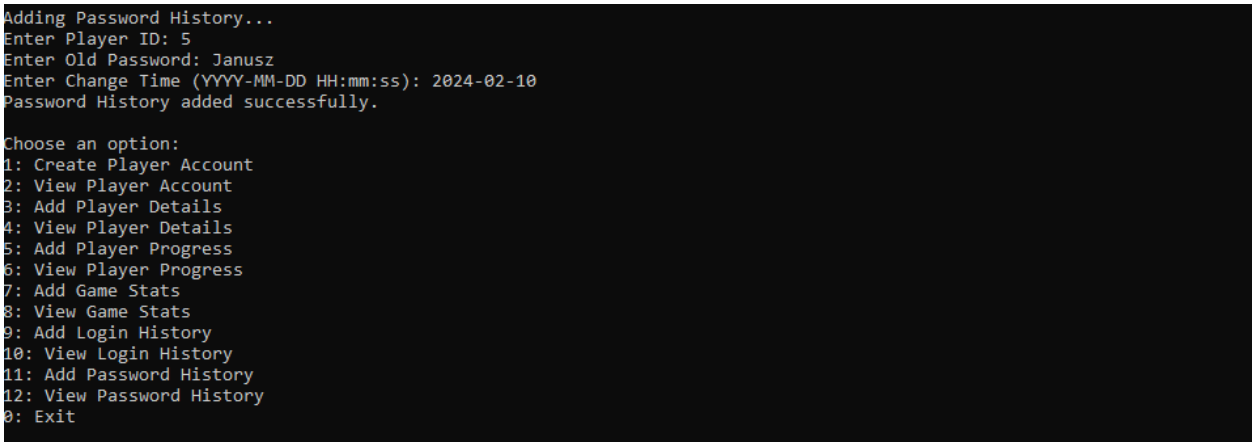
View Login History

Umożliwia użytkownikom przeglądanie historii logowań, co jest przydatne do monitorowania aktywności i bezpieczeństwa kont użytkowników.

Rysunek 4.11: View Login History

Add Password History

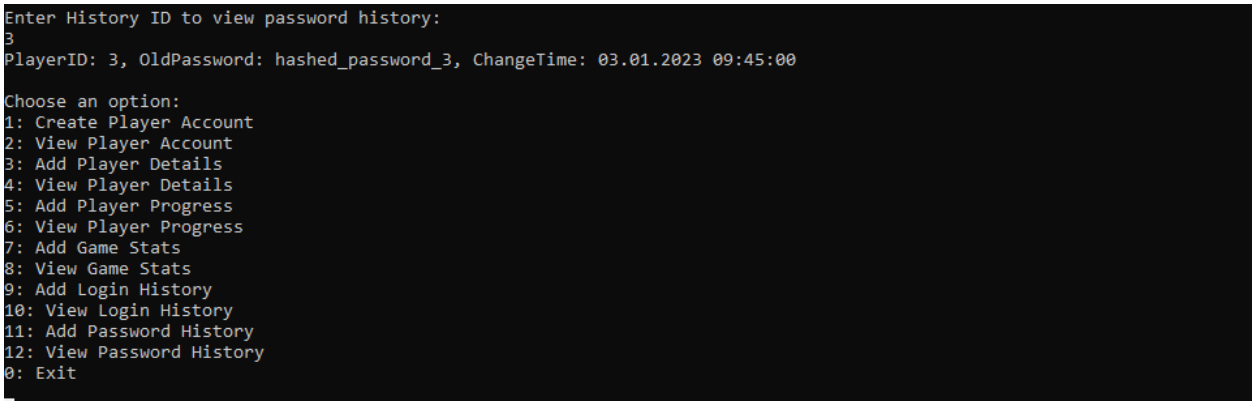
Ta opcja pozwala na zapisywanie historii zmian haseł. To może być wykorzystywane do śledzenia, kiedy i jak często użytkownik zmienia swoje hasło.



Rysunek 4.12: Add Password History

View Password History

Użytkownik może wyświetlać historię zmian haseł, co jest ważne dla audytów bezpieczeństwa i przestrzegania polityk dotyczących haseł.



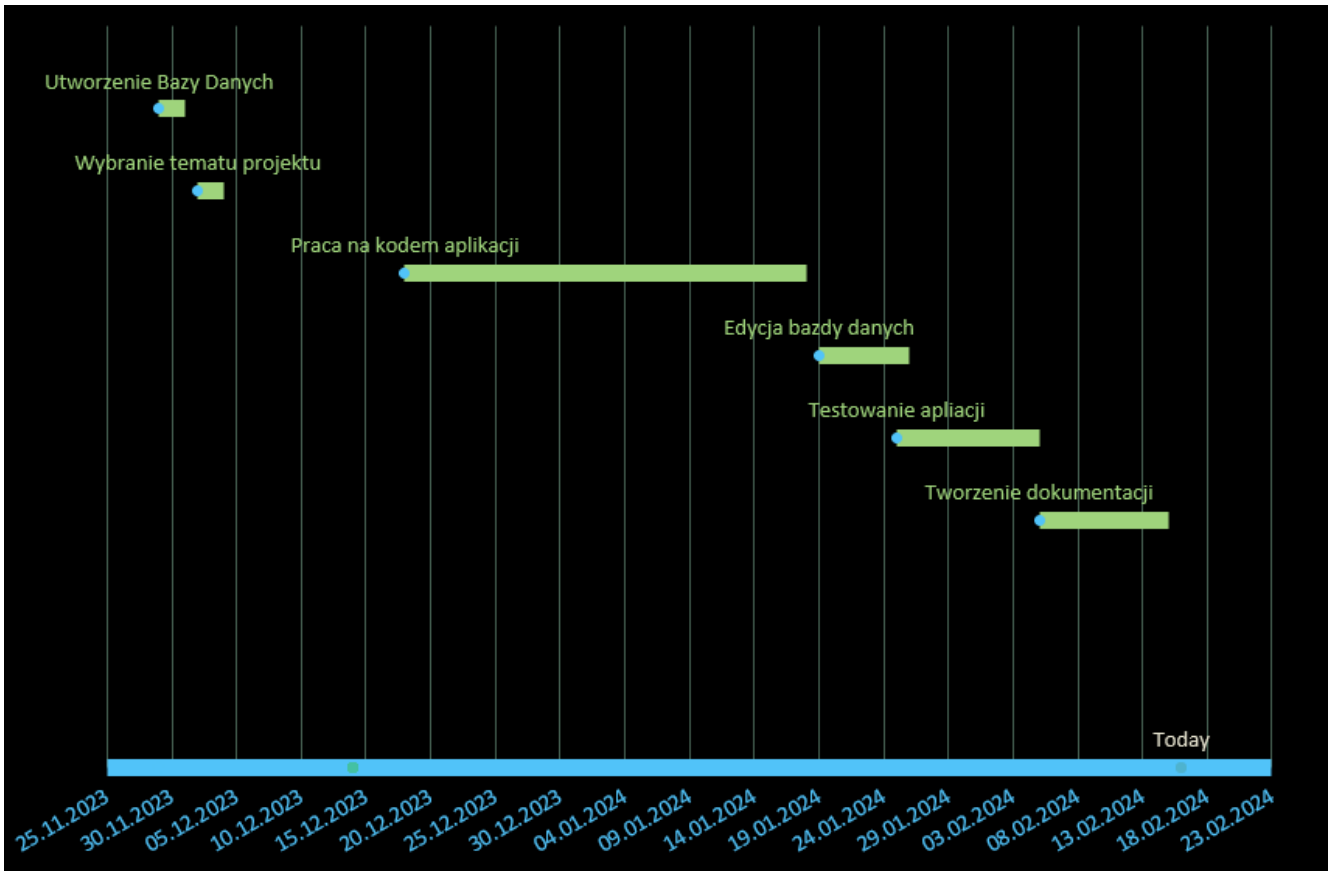
Rysunek 4.13: View Password History

Rozdział 5

Podsumowanie Projektu

5.1 Diagram Ganta

Użyłem diagramu Gantta w moim projekcie, aby uzyskać klarowny wizualny obraz planowanych zadań i ich czasowych ramach. Dzięki niemu mogłem lepiej zarządzać czasem i priorytetami.



Rysunek 5.1: Diagram Grantta

1. **Utworzenie Bazy Danych:** Baza danych jest kluczowym elementem całego programu, utworzenie diagarmu umożliwiło prace na programem.
2. **Wybranie tematu projektu:** Przedstawia porces tworzenia aplikacji na wczesniej utworzonej bazie danych.
3. **Praca nad kodem aplikacji:** Process tworzenia kodu aplikacji, z uwzględnieniem połączenia z bazą danych.
4. **Edycja bazy danych:** Aplikacja nie była w stanie uzywać zapytań, przez błąd przy tworzeniu bazy. Okres ten zmusił mnie do utworzenia bazy danych od początku
5. **Testowanie aplikacji:** Faza testowania aplikacji, czy wszystkie polecenia działają poprawnie z bazą danych.
6. **Tworzenie dokumentacji:** Ostatnia faza pracy nad projektem w której podsumowałem całą aplikacje.

5.2 Repozytorium

W moim projekcie Git pełnił kluczową rolę jako system kontroli wersji, umożliwiając mi efektywne zarządzanie zmianami w kodzie i dokumentacji. Dzięki Git miałem możliwość śledzenia postępów w projekcie, co ułatwiało identyfikowanie i naprawianie błędów, a także eksperymentowanie z nowymi funkcjonalnościami bez ryzyka utraty wcześniejszej pracy. Git zapewniał również łatwą możliwość przywracania wcześniejszych wersji plików, co było niezwykle przydatne przy cofaniu niepożądanych zmian. Użycie Gita pozwoliło mi na utrzymanie porządku i organizacji w mojej pracy, co jest szczególnie ważne w indywidualnych projektach, gdzie samodzielne zarządzanie wszystkimi aspektami projektu jest kluczowe. Repozytorium Git zawierające mój projekt dostępne jest pod podanym adresem. Link do mojego repozytorium Git, gdzie znajduje się kod źródłowy, dokumentacja, co pozwala na pełne zrozumienie i śledzenie rozwoju projektu.

Link Do GitHub

5.3 Podsumowanie

Projekt GameHRMS został stworzony jako kompleksowy system zarządzania zasobami ludzkimi w świecie gier komputerowych. Jego celem jest umożliwienie administratorom oraz użytkownikom skutecznego zarządzania kontami, postępami graczy oraz zapewnienie bezpieczeństwa danych.

Główne cele projektu to zapewnienie użytkownikom możliwości łatwego rejestracji i logowania do systemu, zarządzania swoimi kontami oraz monitorowania swojego postępu w grach. Administratorzy natomiast mają dostęp do narzędzi umożliwiających zarządzanie użytkownikami, ich statystykami oraz historią logowania.

Realizacja projektu skupiła się na stworzeniu struktury bazy danych, która pozwala na efektywne przechowywanie danych użytkowników, ich statystyk oraz historii logowania. Oprócz tego, zaimplementowano klasy i usługi odpowiedzialne za logikę biznesową systemu, zapewniając prosty interfejs użytkownika, który umożliwia wygodne korzystanie z funkcji systemu.

Planowane dalsze prace rozwojowe obejmują rozbudowę interfejsu użytkownika w celu zapewnienia bardziej zaawansowanych możliwości interakcji, wprowadzenie systemu autentykacji i autoryzacji w celu zwiększenia bezpieczeństwa oraz optymalizację i skalowanie systemu, aby sprostać rosnącym wymaganiom i zapewnić płynne działanie nawet przy dużej liczbie jednoczesnych użytkowników.

Podsumowując, projekt GameHRMS ma na celu stworzenie wszechstronnego narzędzia, które ułatwi zarządzanie zasobami ludzkimi w branży gier komputerowych, zapewniając jednocześnie bezpieczeństwo danych i wygodę użytkowników.

Bibliografia

[1] Jacek Matulewski, *C#: lekcje programowania: praktyczna nauka programowania dla platform .NET i .NET Core*, Helion, Gliwice 2021.

[2] Joseph Albahari, Eric Johannsen, *C# 8.0 w pigułce*, Helion, Gliwice 2021.

[3] R. S. Miles, *C: zacznij programować!*, Helion, Gliwice 2020.

[4] https://docs.github.com/en

[5] https://stackoverflow.com

[6] https://www.freecodecamp.org

[7] https://developers.google.com

Spis rysunków

2.1 Diagram ERD 9

3.1 Diagram klasy Program 16

4.1 Menu główne 17
4.2 Create Player Account 17
4.3 View Player Account 18
4.4 Add Player Details 18
4.5 View Player Details 19
4.6 Add Player Progress 19
4.7 View Player Progress 19
4.8 Add Game Stats 20
4.9 View Game Stats 20
4.10 Add Login History 21
4.11 View Login History 21
4.12 Add Password History 21
4.13 View Password History 22

5.1 Diagram Grantta 23