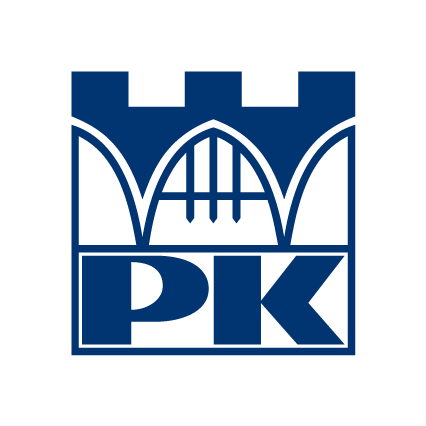
**Skład:** Dawid Orłowski, Konrad Mech, Dawid Plewa

Wydział Inżynierii Elektrycznej i Komputerowej

Informatyka w Inżynierii Komputerowej

***TECHNOLOGIE OBIEKTOWE***

**Sprawozdanie**

**‘’Aplikacja kasyna od strony klienckiej i serwerowej”**

* 1. **Cel i zakres projektu**

Zadaniem projektu jest stworzenie aplikacji kasyna, zapewniającej niezapomnianą rozrywkę i wspaniałe przeżycia.

Aplikacja będzie połączona z bazą danych oraz panel gracza wyświetlany w interfejsie graficznym.

Zespół zajmie się stworzeniem logiki do każdej z poszczególnych gier.

Użytkownik otrzymuje dostęp do wpłacania oraz wypłacania swoich środków w dowolnym momencie.

* 1. **Charakterystyka użytkowników**

Nasza aplikacja będzie przeznaczona dla graczy do różnego sposobu relaksu jak i poczucia adrenaliny do uprawiania hazardu. Do wyboru będzie kilka gier, każdy znajdzie coś dla siebie.

Wyróżniamy również stronę bazy danych, za którą będą odpowiadać administratorzy. W bazie będą przechowywane dane użytkowników, którzy zarejestrowali się w aplikacji.

* 1. **Główne funkcje produktu**

Możemy wyróżnić wiele funkcji aplikacji, natomiast głównymi będą:

-**Panel rejestracji i logowania** – bezpieczny mechanizm rejestracji konta graczy przy użyciu hasła

-**Lobby gier** – główny panel naszej aplikacji wyświetlany w szacie graficznej, możliwość jednej z kilku gier oraz informacje o zasadach każdej gry

-**Portfel i transakcje** – system zarządzania portfelem gracza, wpłaty, wypłaty oraz historia transakcji

\**w planach mamy rozszerzenie aplikacji od strony klienckiej, aby każdy użytkownik mógł sprawdzić swoją historię gier*

* 1. **Wymagania funkcjonalne**

-rejestracja,

-panel logowania,

-panel użytkownika,

-edycja danych klienta,

-lista znajomych,

-przegląd gier,

-przegląd zasad gier,

-wpłaty,

-wypłaty,

-historia transakcji \**in progress*,

-zarządzanie bazą danych (administrator);

* 1. **Wymagania niefunkcjonalne**

-bezpieczeństwo,

-wydajność,

-dostępność,

-zarządzanie bazą danych,

-skalowalność,

-losowość gier – „Fair Play”,

-elastyczność systemu,

-przyjazny dla oka interfejs,

-szyfrowanie danych *\*in progress,*

-zabezpieczenie kont;

**2. Diagram przypadków użycia**

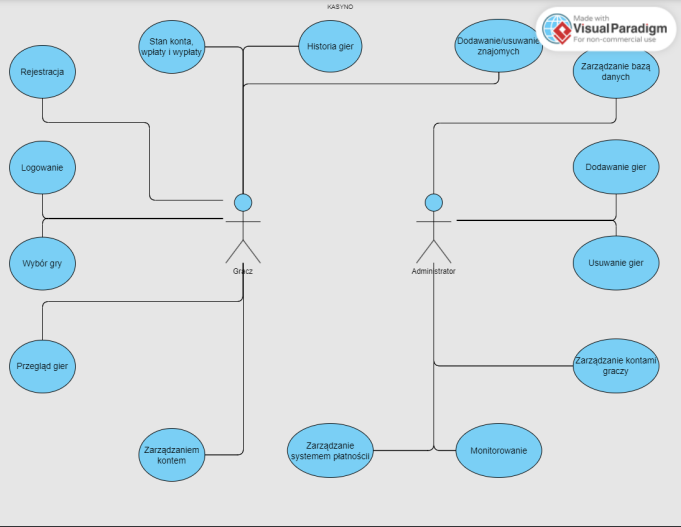
****

Diagram przypadków użycia, który widzimy, reprezentuje różne scenariusze interakcji użytkowników z aplikacją kasyna napisaną w języku Java. Oto krótki opis każdego przypadku użycia:

**Rejestracja**: Użytkownik może utworzyć nowe konto w kasynie.

**Logowanie**: Użytkownik może zalogować się na swoje konto.

**Wybór gry**: Po zalogowaniu użytkownik ma możliwość wybrania gry, w którą chce grać. Przegląd gier: Użytkownik może przeglądać dostępne gry w kasynie.

**Stan konta, wpłaty i wypłaty**: Użytkownik może sprawdzić stan swojego konta i dokonać wpłat lub wypłat środków.

**Historia gier**: Użytkownik ma dostęp do historii swoich gier, aby śledzić swoje aktywności w kasynie. *\*in progress*

**Zarządzanie kontem**: Użytkownik może zarządzać swoim kontem, np. zmieniać ustawienia lub aktualizować dane osobowe. *\*in progress*

**Zarządzanie systemem płatności**: Użytkownik może zarządzać swoimi metodami płatności, dodawać nowe metody lub aktualizować istniejące.

Dla administratora aplikacji kasyna istnieją następujące przypadki użycia:

**Dodawanie/Usuwanie użytkowników**: Administrator może dodawać lub usuwać użytkowników.

**Zarządzanie bazą danych**: Administrator dba o utrzymanie i zarządzanie bazą danych, w której przechowywane są wszystkie dane użytkowników i gier.

**Dodawanie gier**: Administrator może wprowadzać nowe gry do oferty kasyna.

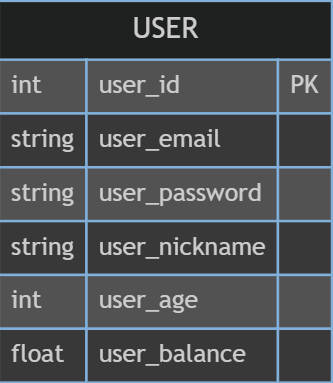
**Usuwanie gier**: Administrator może usuwać gry, które nie są już dostępne lub popularne.

**Zarządzanie kontami graczy:** Administrator ma możliwość zarządzania kontami graczy, w tym blokowania lub dezaktywowania kont.

**Monitorowanie**: Administrator ma możliwość monitorowania działalności w kasynie, w tym gier i transakcji finansowych.

Cały system jest zaprojektowany tak, aby umożliwić płynną interakcję między użytkownikami a aplikacją, a także umożliwić administratorowi zarządzanie kluczowymi aspektami aplikacji kasyna.

**3. Diagram ERD**

****

Obrazek przedstawia diagram ERD (Entity Relationship Diagram) dla tabeli o nazwie "**USER**".

Diagram składa się z sześciu atrybutów:

**user\_id (int)** - Jest to klucz główny tabeli (PK), który jest unikatowym identyfikatorem dla każdego rekordu użytkownika. Typ danych to liczba całkowita.

**user\_email (string)** - Ten atrybut przechowuje adres e-mail użytkownika. Typ danych to ciąg znaków.

**user\_password (string)** - Zawiera hasło użytkownika. Typ danych to ciąg znaków, który najprawdopodobniej jest zaszyfrowany dla bezpieczeństwa.

**user\_nickname (string)** - To pseudonim użytkownika. Typ danych to ciąg znaków.

**user\_age (int)** - Przedstawia wiek użytkownika jako liczbę całkowitą.

**user\_balance (float)** - To pole wskazuje saldo użytkownika, prawdopodobnie na jakimś koncie lub w aplikacji. Typ danych to liczba zmiennoprzecinkowa, co pozwala na reprezentowanie wartości pieniężnych włączając grosze.

**4. Diagram Klas**

**A screenshot of a computer

Description automatically generated**

**Klasa DB:** Ta klasa jest odpowiedzialna za zarządzanie połączeniem z bazą danych. Zawiera metody *connect*(), *disconnect*(), *executeQuery*() oraz prywatne pola przechowujące dane połączenia. Jest używana przez inne klasy kontrolerów do interakcji z bazą danych.

**Klasa LoginPanel:** Reprezentuje interfejs graficzny panelu logowania. Zawiera prywatne komponenty GUI, takie jak *usernameField*, *passwordField* i *loginButton*. Jest to klasa czysto wizualna, bez bezpośrednich zależności od innych klas.

**Klasa LoginPanelController**: Zarządza logiką autentykacji użytkownika. Zawiera metody authenticateUser() oraz cancel(). Ta klasa używa klasy DB do weryfikacji danych logowania, co jest zaznaczone jako zależność (strzałka przerywana).

**Klasa MenuPanelController**: Odpowiada za interakcję z głównym menu aplikacji. Zawiera metody takie jak openGame(), viewProfile(), logout(). Podobnie jak LoginPanelController, ma zależność od klasy DB.

**Klasa SlotsGameController**: Zarządza mechaniką gry w sloty. Zawiera metody spinReels(), calculatePayout(), updateUserBalance(). Ta klasa używa obiektów z klasy User do aktualizacji salda oraz ma zależność od klasy DB.

**Klasa User**: Reprezentuje gracza. Zawiera pola takie jak userID, password, balance oraz metody updateBalance(), playGame(). Ma asocjację z klasą UserData, co wskazuje na to, że każdy użytkownik ma powiązane dane profilu.

**Klasa UserData**: Przechowuje dane profilu użytkownika, takie jak statystyki gier i ustawienia. Zawiera metody getStatistics(), updatePreferences().