APLIKACJA DO ZARZĄDZANIA KATALOGIEM SZACHOWYM

Opis Projektu:

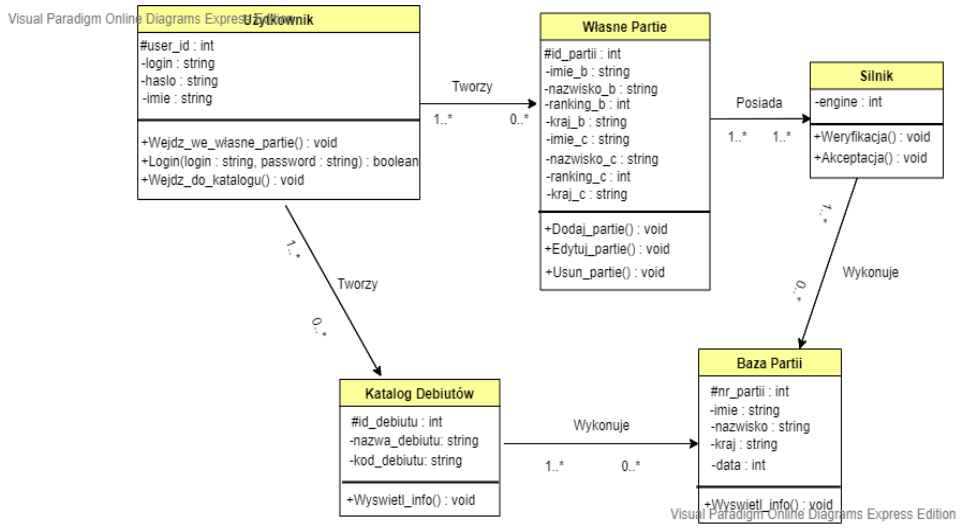
Aplikacja w głównej mierze będzie opierała się na wyszukiwaniu informacji odnośnie rozegranych meczy zawodników. Dany użytkownik będzie mógł uzyskać informacje o danym zawodniku na przykład: jego imię, nazwisko, obywatelstwo czy jego ranking. Korzystanie z aplikacji będzie zabezpieczone za pomocą systemu logowania to znaczy jego login i hasło.

Użytkownik ponadto będzie miał możliwość na przykład:

- Dodanie w odrębnym dziale swoje własne rozegranie partie z zawodnikami wraz z ich edycją.
- Możliwość analizy swoich partii szachowych za pomocą silnika szachowego.
- Wyszukanie debiutu szachowego na podstawie bazy danych partii zawodników

Celem zapewnienia bezpieczeństwa dla użytkownika hasła będą szyfrowane za pomocą specjalnego funkcji szyfrowania. Przechowywanie informacji o partiach szachowych czy danych logowania będą oparte na bazie danych przez specjalny program na przykład **ORACLE**.

1. **DIAGRAM KLAS**



Użytkownik - klasa zawiera dane o użytkowniku takie jak: user_id, login, hasło, oraz imie.

- **Wejdz_we_wlasne_partie()** - katalog własnych partii
- **Login** – zalogowanie się
- **Wejdz_do_katalogu()** - katalog bazy partii i debiutów

Własne Partie – klasa reprezentująca katalog własnych partii zawierająca informację takie jak : id_partii, imię, nazwisko, czy ranking.

- **Dodaj Partie()** - dodanie partii
- **Edytuj Partie()** - edytowanie partii
- **Usun Partie()** - usunięcie partii

Katalog Debiutów - klasa przeszukująca informacji na temat debiutu zawierająca takie informacje jak: id_debiutu, nazwa_debiutu, kod_debiutu.

- **Wyswietl_info()** - wyświetla informacje

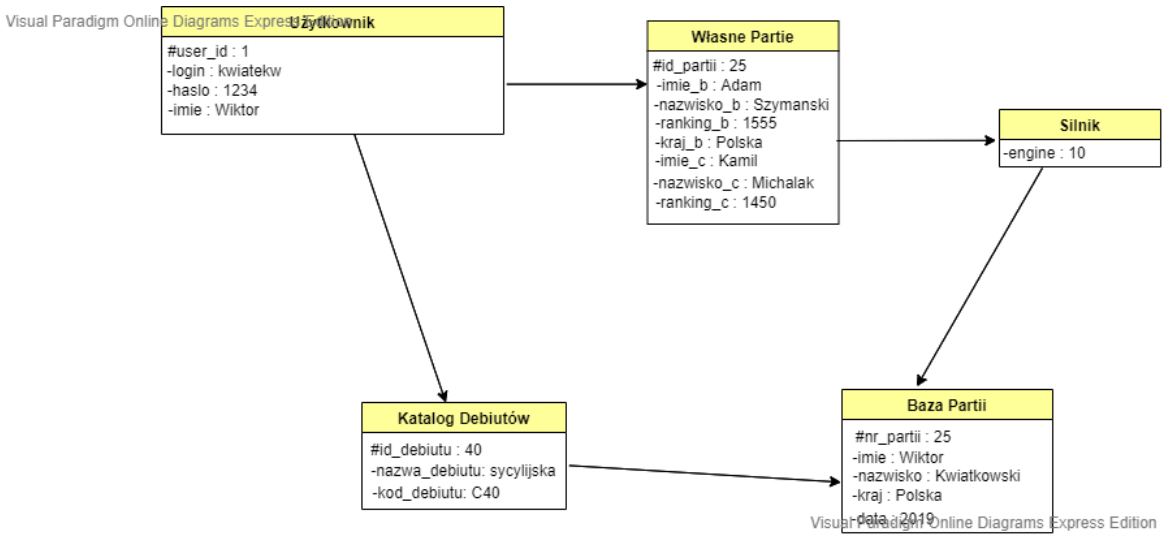
Baza Partii – klasa reprezentująca pojedynczą partię zawierająca indeks nr_partii , imię , nazwisko , kraj, data

- **Wyswietl_info()** - wyświetla informacje

Silnik – klasa reprezentująca silnik szachowy do sprawdzenia weryfikacji poprawności wpisanej partii szachowej.

- **Weryfikacja()** - weryfikacja działania
- **Akceptacja()** - zaakceptowanie wszystkich zmian

2. DIAGRAM OBIEKTÓW



3. DIAGRAM PRZYPADKÓW UŻYCIA

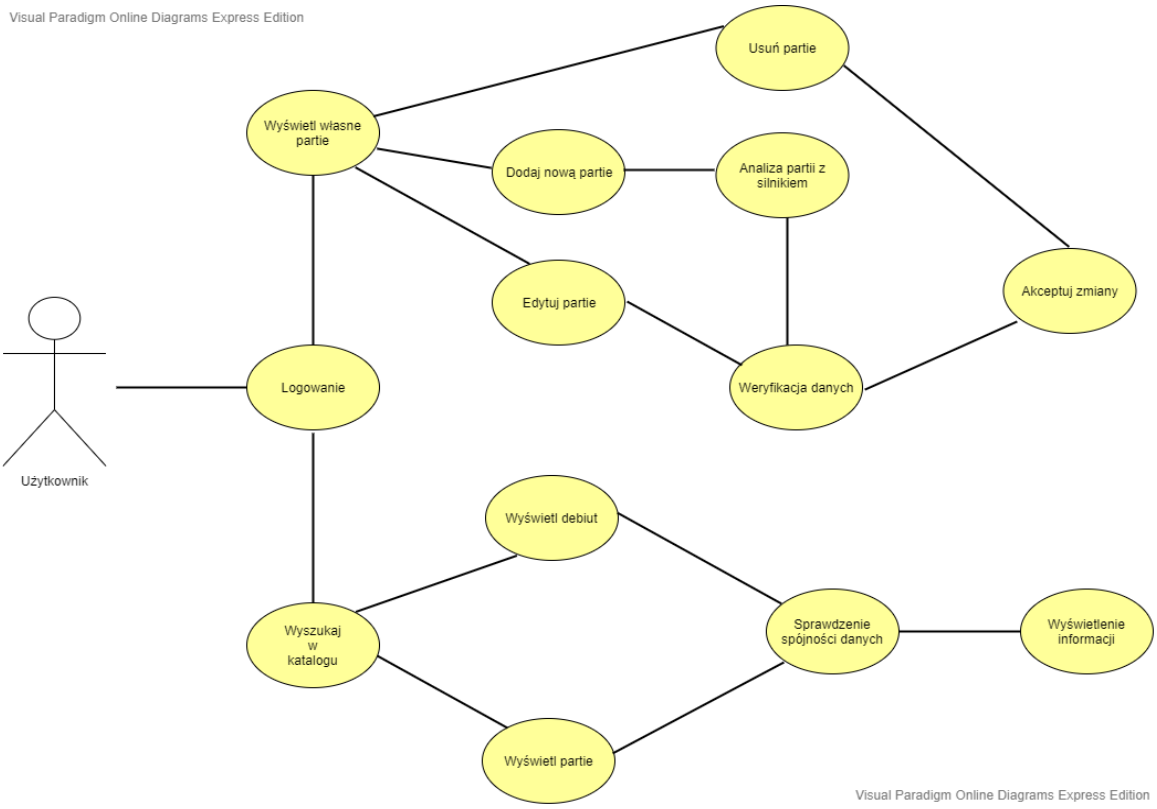
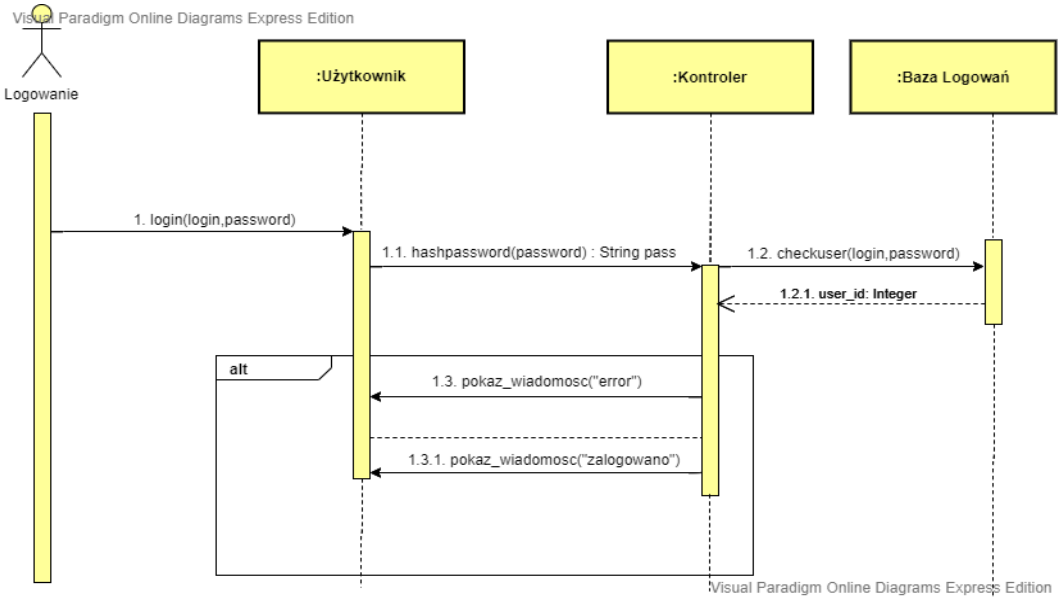


Diagram przypadków użycia odwzorowuje funkcje aplikacji, jaką będą widzieć użytkownicy.

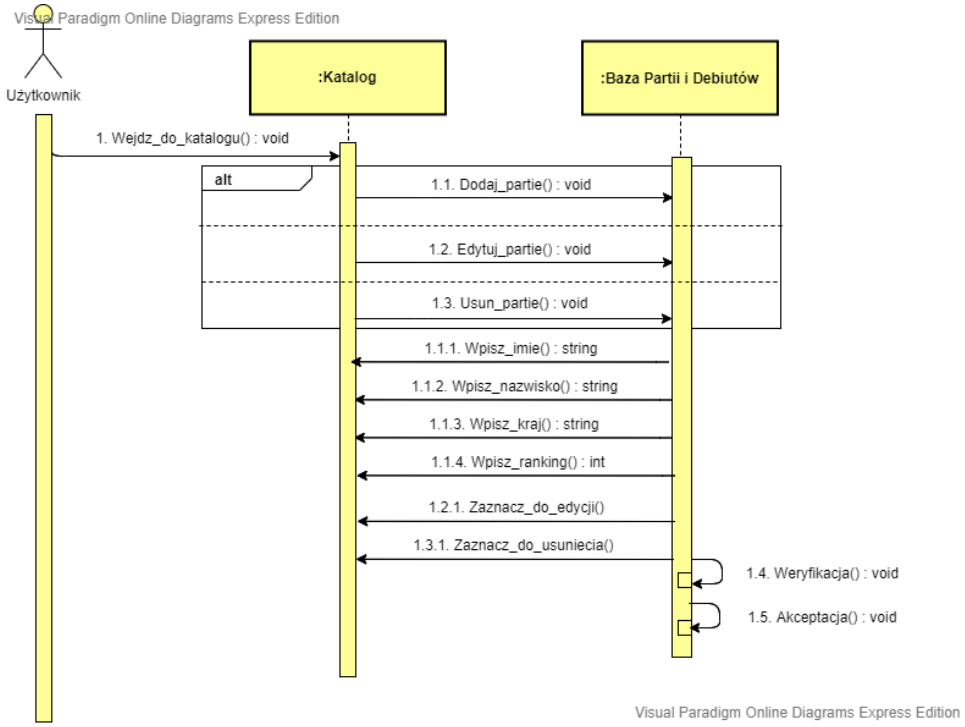
4. DIAGRAM PRZEBIEGU

- **4.1 Logowanie Użytkownika**



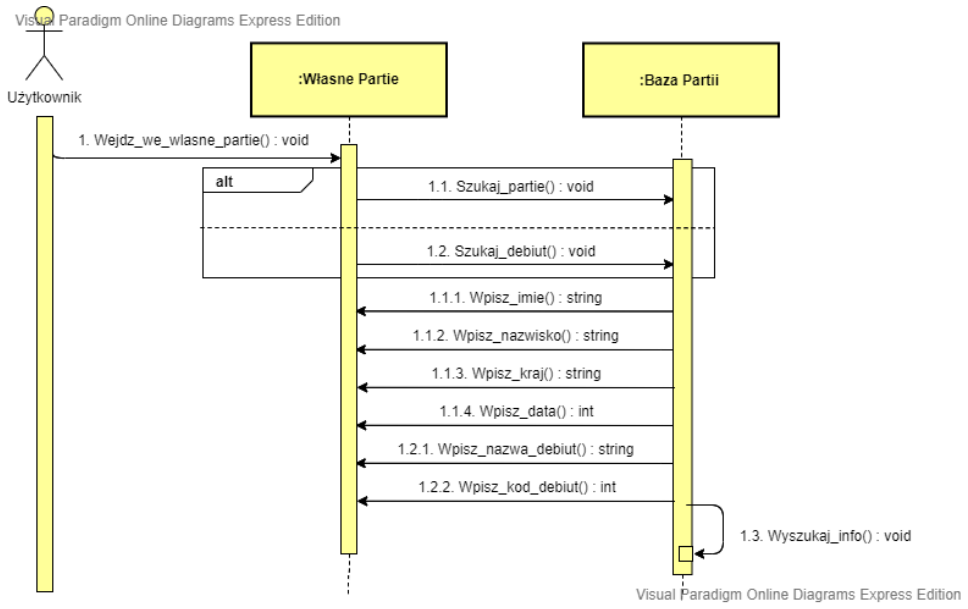
Aplikacja wyszukuje użytkownika o podanym loginie i zaszyfrowanym haśle w bazie danych – jeżeli metoda checkuser() zwróci wartość 1 to dany użytkownik istnieje w bazie. Wtedy wyświetlany jest stosowny komunikat o powodzeniu procesu logowania. W przypadku pojawienia się innej wartości zwracanej przez checkuser() wyświetlany jest komunikat o błędzie.

- **4.2 Obsługa katalogu**



W przypadku chęci wejścia do katalogu wywołujemy metodę **Wejdz_do_katalogu**, która wykonuje jedno z podanych trzech poleceń: **Dodaj_partie()**, **Edytuj_partie()**, **Usun_partie()**. Następnie katalog jest poddany weryfikacji i akceptacji poprzez użycie poleceń: **Weryfikacja() i Akceptacja().**

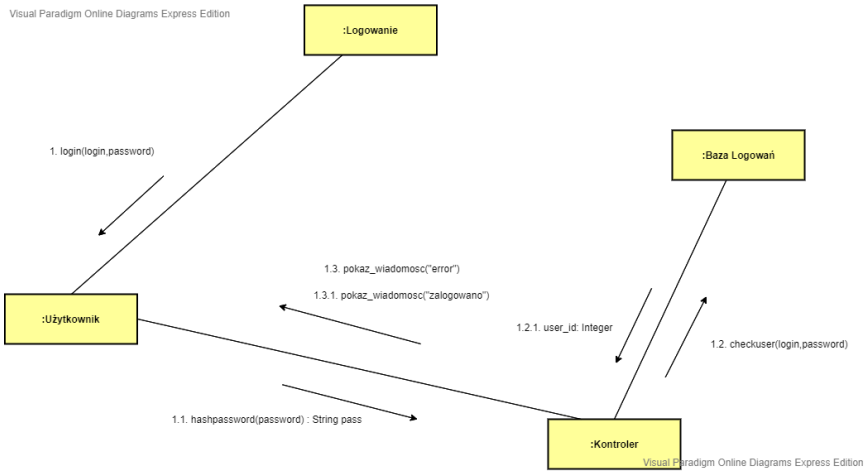
- **4.3 Obsługa własnych partii**



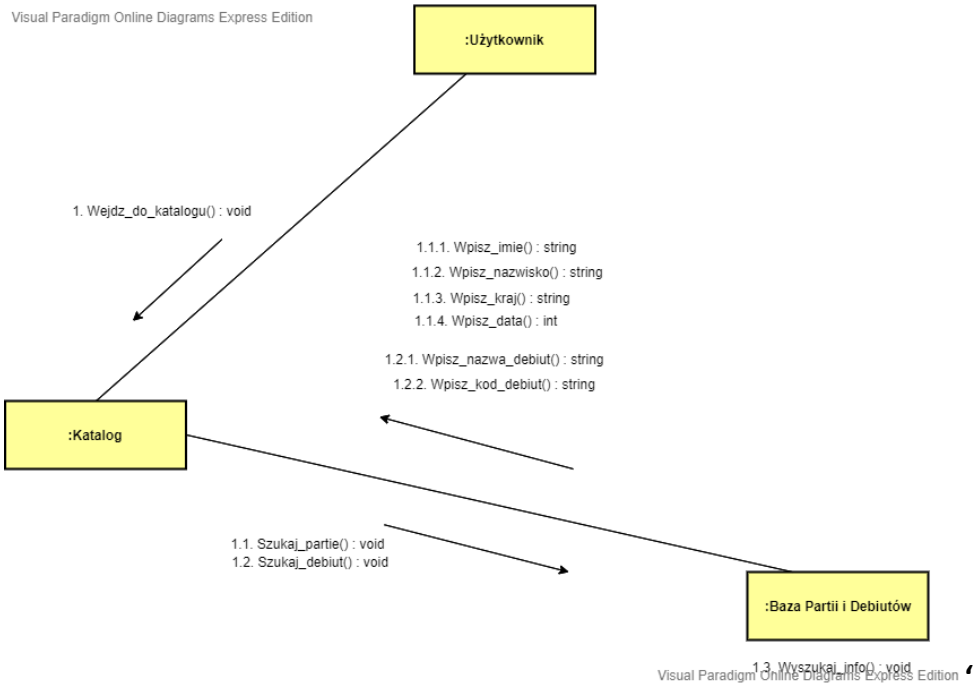
W przypadku chęci wejścia do własnych partii wywołujemy metodę **Wejdz_we_wlasne_partie()**, która wykonuje jedno z podanych dwóch poleceń: **Szukaj_partie()**, **Szukaj_debiut()**. Następnie katalog wyszukuje informacji na temat wybranej opcji poprzez metodę **Wyszukaj_info()**.

5. DIAGRAM KOOPERACJI:

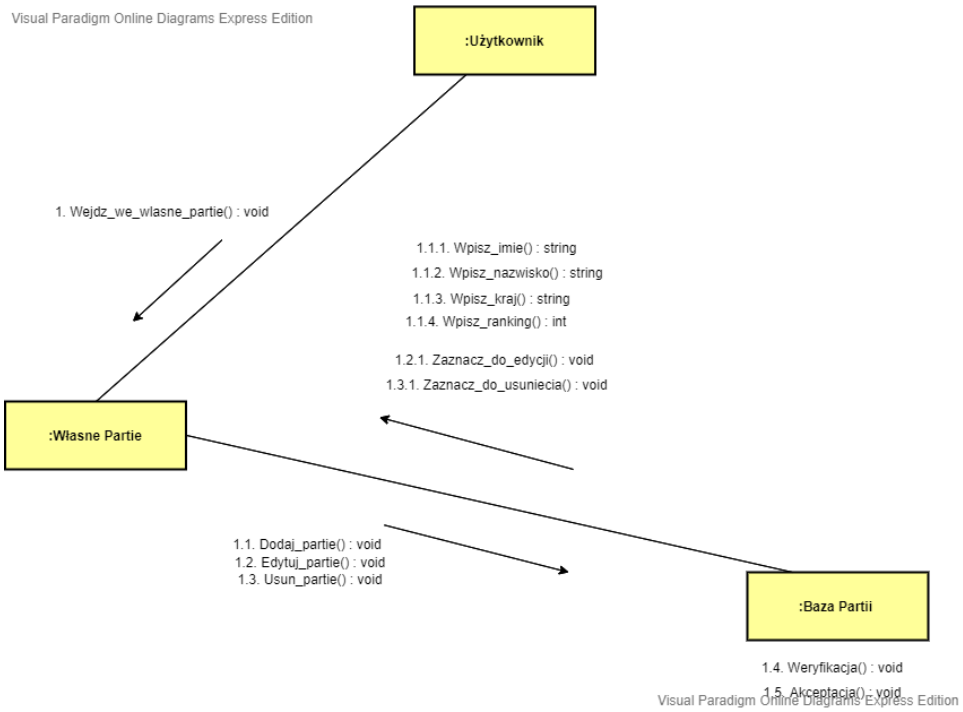
- **5.1 Logowanie Użytkownika**



- **5.2 Obsługa katalogu**



- **5.3 Obsługa własnych partii**



6. DIAGRAM STANU

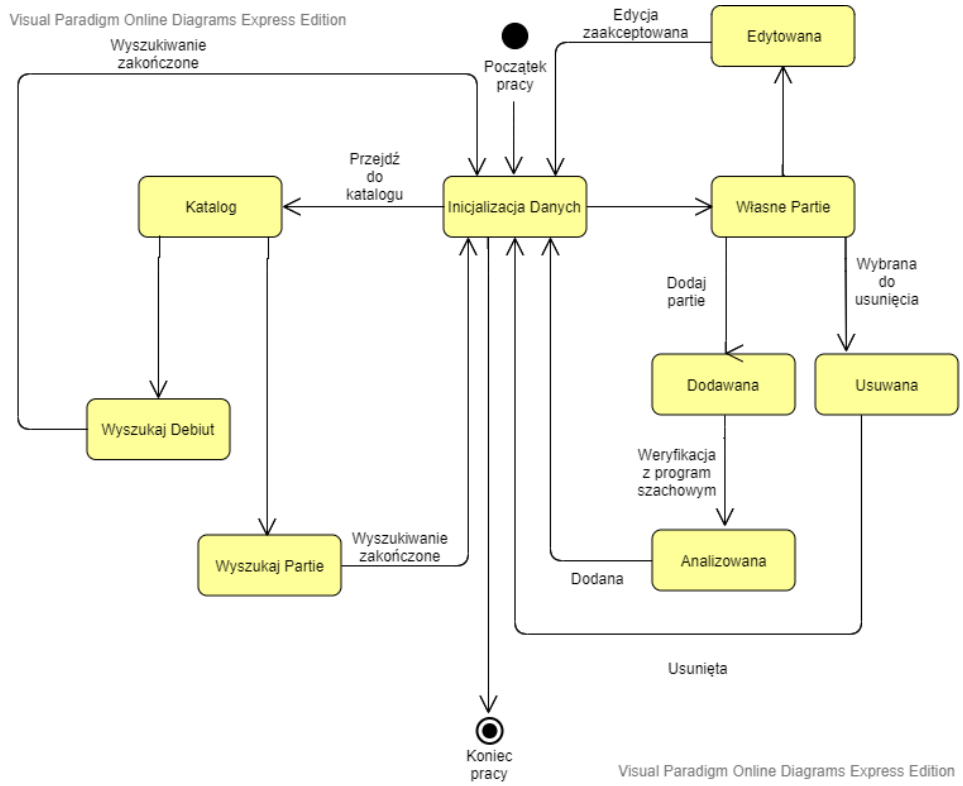


Diagram stanu obrazuje wszystkie możliwe stany przebiegu aplikacji w poszczególnych działach.

7. DIAGRAM CZYNNOŚCI

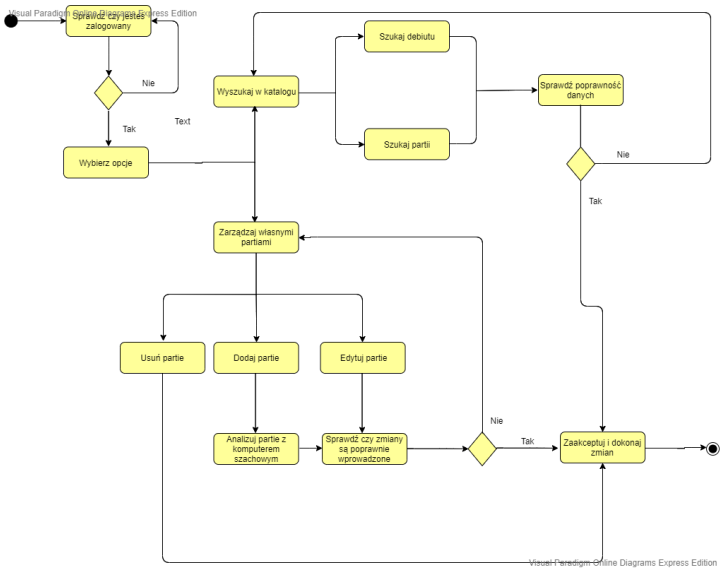
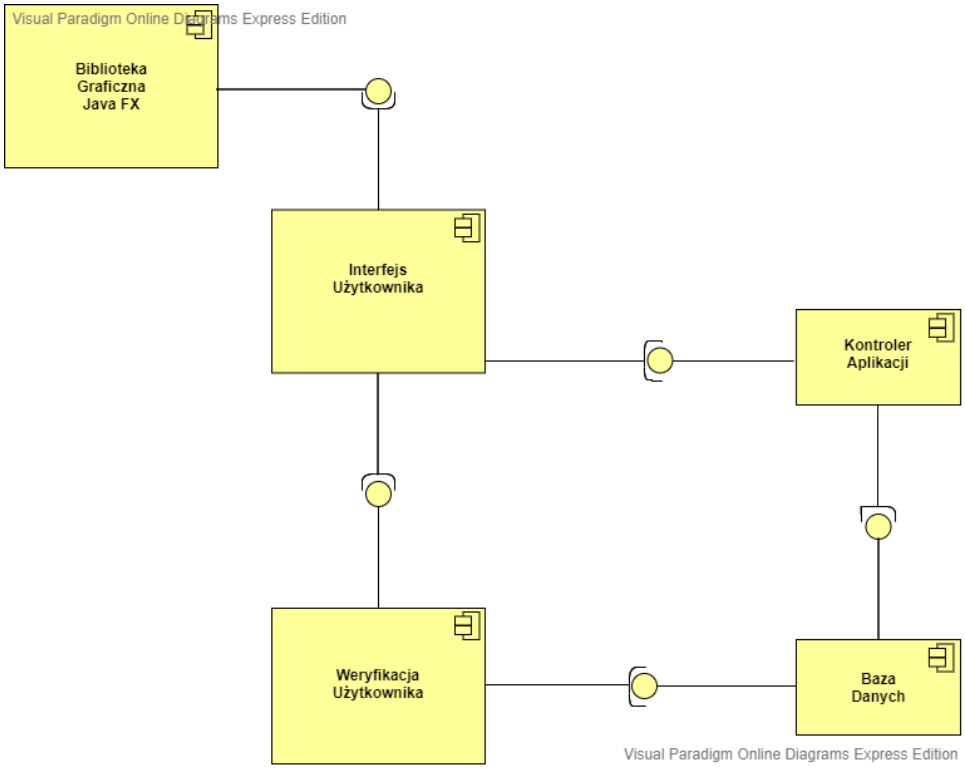


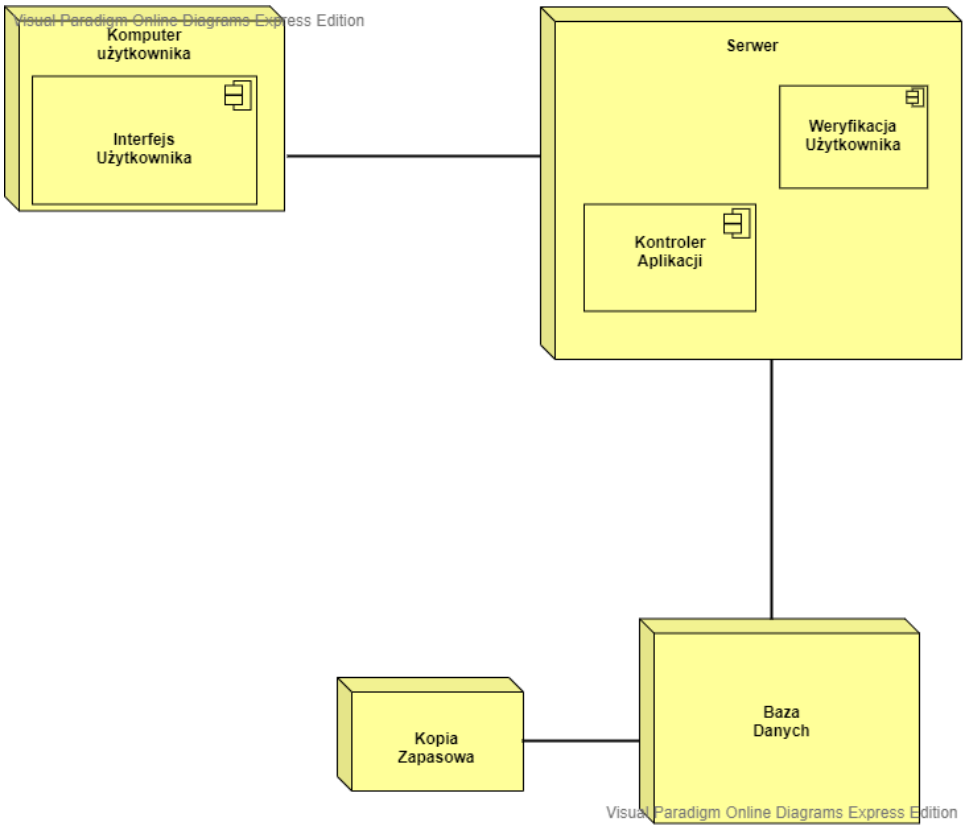
Diagram czynności obrazuje wszystkie możliwe scenariusze dla zwykłego użytkownika.

8. DIAGRAM KOMPONENTÓW



- Interfejs użytkownika – część aplikacji przeznaczona do interakcji z użytkownikiem. Odpowiada również za pobieranie wprowadzanych danych.
- Biblioteka graficzna JavaFX - grafika
- Kontroler – główna część aplikacji, której zadaniem jest walidacja wprowadzanych danych oraz wykonywanie operacji bazodanowych.
- Weryfikacja użytkownika – komponent odpowiedzialny za weryfikację, czy podane na ekranie logowania dane znajdują się w bazie danych, czy też nie.
- Baza danych – magazyn danych zawierający wszystkie informacje na temat naszej aplikacji.

9. DIAGRAM WDROŻENIA



Aplikację można napisać w modelu użytkownik-serwer. W takim przypadku nadostępny będzie tylko graficzny interfejs aplikacji, który będzie komunikował się z serwerem. Serwer będzie zajmował się walidacją wprowadzanych danych, wykonywaniem operacji na bazie danych oraz weryfikacją użytkownika. Ostatnim elementem jest baza danych. Będzie wykonywana kopia zapasowa w celu zabezpieczenia bazy danych przed utratą jakichkolwiek informacji.