



MonDex

Centrum Pókemonów

Dokumentacja projektu

Zespół projektowy

Katsiaryna Kolyshko 276708

Wiktor Konieczny 272956

Prowadzący

dr inż. Marcin Łopuszyński

Przedmiot

Bazy Danych 2 – Projekt

Spis treści

1. Opis wycinka rzeczywistości	4
1.1 Przebieg aplikacji.....	4
1.2 Aktorzy.....	6
1.3 Modelowe scenariusze	7
2. Opracowanie wymagań funkcjonalnych i niefunkcjonalnych	8
2.1 Wymagania funkcjonalne.....	8
2.2 Wymaganie niefunkcjonalne.....	8
3. Tekstowy opis wszystkich przypadków użycia.....	9
1) Zalogowanie do systemu	9
2) Rozegranie pojedynku	9
3) Awansowanie zawodnika (trenera na mistrza).....	10
4) Awansowanie zawodnika (z mistrza na czempiona).....	11
5) Zdobywanie Pokémonów	11
6) Ewolucja Pokémonów (rozszerzenie "Zdobywania Pokémonów")	12
7) Edytowanie bazy danych	13
8) Dodawanie Pokémonów (rozszerzenie "Edytowanie bazy danych")	14
9) Usuwanie Pokémonów (rozszerzenie "Edytowanie bazy danych")	14
10) Dodawanie trenera (rozszerzenie "Edytowanie bazy danych")	15
11) Usuwanie trenera (rozszerzenie "Edytowanie bazy danych")	15
12) Edytowanie Pokémonów (rozszerzenie "Edytowanie bazy danych")	16
13) Przeglądanie bazy danych	17
14) Przeglądanie Pokémonów (rozszerzenie "Przeglądanie bazy danych")	17
15) Przeglądanie trenerów (rozszerzenie "Przeglądanie bazy danych").....	18
4. Sformułowanie wymagań	19
5. Diagram encji.....	20
6. Opis tabel i wypełnienia.....	20
6.1 Typ energii:	20
6.2 Rzadkość:	21
6.3 Regiony:	21
6.4 Ewolucja:	22
6.5 Kolor:	22
7. Identyfikacja Encji	22
8. Opis działania kodu.....	25
9. Testy Bazy.....	30
10. Makieta interfejsu graficznego.....	31
11. Diagram czynności.....	40

12. Diagram klas	41
13. Aplikacja desktopowa	42
13.1 Interfejsy	42
13.2 Opisanie kodu:	50
13.3 Budowa interfejsu	56
13.4 Połączenie z bazą danych.....	67
14. Testy i niezawodność aplikacji.....	69
15. Lista zmian.....	74

1. Opis wycinka rzeczywistości

1.1 Przebieg aplikacji

Centrum Pokémon to instytucja, w której odwiedzający mogą zapoznać się ze światem Pokémonów i ich trenerów. Zdefiniowano wystawę, na której goście mogą oglądać kolekcję Pokémonów i trenerów. Wystawa zawiera profile, gdzie każdy profil przedstawia Pokémona wraz z jego trenerem, w tym szczegóły dotyczące typu Pokémona, mocnych stron i statusu szkolenia. Każdy profil zawiera informacje, do których mogą uzyskać dostęp goście lub, w niektórych przypadkach, zarządzane przez upoważniony personel Centrum Pokémon w imieniu trenerów. Tylko konkretni członkowie personelu mają uprawnienia do aktualizowania i publikowania tych profili, aby zapewnić, że wystawa pozostanie dobrze zorganizowana i dokładnie odzwierciedli aktualną listę Pokémonów i trenerów.

Istnieje wiele typów i odmian Pokémonów, zaczynając od typów energii, rzadkości, ubarwienia, etapu ewolucji, a kończąc na regionach, w których można je spotkać. Im więcej masz doświadczenia, tym większe prawdopodobieństwo, że złapiesz rzadsze stworzenie i pomożesz mu ewoluować.

Pokemon ma 3 etapy ewolucji:

- Etap podstawowy: Jest to początkowa forma Pokémona. Pokémony na tym etapie są zazwyczaj słabsze, ale mogą awansować poprzez trening.
- Pierwsza ewolucja: W miarę jak Pokémon zdobywa doświadczenie, ewoluuje w silniejszą formę ze zwiększonymi umiejętnościami i często bardziej złożonym wyglądem. Na tym etapie jego zestaw ruchów się rozszerza i zyskuje zwiększone statystyki.
- Ostateczna ewolucja: Po znaczącym treningu Pokémon osiąga swoją najpotężniejszą formę. Ten etap zazwyczaj pokazuje drastyczną zmianę rozmiaru, siły i możliwości.

Jest wiele Pokémonów, więc wiele z nich nie zostanie odebranych. Im lepsze jest szkolenie danej osoby, tym rzadszy i lepszy typ może złapać i odebrać dla siebie, co sprawia, że ten gatunek jest niedostępny dla innych. Tak więc jedno konkretne stworzenie może być przypisane tylko do jednego trenera.

Głównym celem Pokémon Center jest zapoznanie odwiedzających z trenerami i Pokémonami, które posiadają. Goście mogą przeglądać listę trenerów z różnych regionów i kategoryzować ich na podstawie typów energii Pokémonów.

Jeśli odwiedzający nie zdecyduje się na rejestrację, może odkrywać różne Pokémony i przeglądać niektóre z ich statystyk, takie jak wzrost, waga, typ energii, rzadkość, a także identyfikować najsilniejsze lub najsłabsze Pokémony w określonych regionach.

Następnie Profesor zapyta odwiedzającego, czy chciałby dołączyć do centrum. Jeśli się zgodzi, otrzyma identyfikator i zostanie poproszony o wybranie preferowanego typu energii Pokémona. Następnie zostanie mu przydzielony pierwszy Pokémon i rozpocznie swoją podróż jako trener.

Jeśli odwiedzający zdecyduje się zalogować, zostanie poproszony o podanie swojego imienia i nazwiska oraz identyfikatora.

Do jednej osoby może być przepisana ograniczona liczba Pókemonów.

Długość życia pokemonów jest uważana za nieograniczoną. Pokemon nie może umrzeć, tylko usunięty z bazy.

Regresja rangi nie występuje w centrum. Gdy osiągniesz rangę — możesz się tylko rozwijać, chyba że to profesor bezpośrednio zdegradował osobę ze stanowiska. Jeśli osoba zostanie usunięta lub zbanowana z centrum, a następnie dołączy ponownie — musi rozpocząć swoją podróż od samego początku.

1.2 Aktorzy

W Pokémon Center mamy pięć poziomów podejmowania decyzji w hierarchii, od minimalnego do pełnego upoważnienia:

Poziom I – Odwiedzający (Visitors)

Odwiedzający mogą przeglądać podstawowe informacje o trenerach i Pokémonach, ale nie mogą wchodzić w interakcje z żadnym z nich. Przede wszystkim eksplorują katalog, aby dowiedzieć się więcej o dostępnych Pokémonach i trenerach.

Poziom II – Trenerzy (Trainers)

Trenerzy mają wszystkie prawa Odwiedzających, z dodatkowymi przywilejami. Mogą przeglądać Pokémony, które posiadają, i odkrywać nieodebrane Pokémony w swoim regionie. Jako mniej doświadczeni trenerzy są ograniczeni do odbierania Pokémonów z etapów ewolucji 1 i 2, odpowiadających typowi energii ich pierwszego przypisanego Pokémona. Ponadto mogą zobaczyć wszystkich trenerów w swoim regionie.

Poziom III – Mistrzowie (Masters)

Mistrzowie posiadają wszystkie możliwości Trenerów, z dodatkowymi ulepszeniami. Mogą odbierać Pokémony na etapie ewolucji 3 i uzyskiwać dostęp do nieodebranych Pokémonów z innych regionów.

Poziom IV – Arcymistrzowie (Champions)

Arcymistrzowie mają wszystkie prawa Mistrzów, a także możliwość zdobywania rzadkich Pokémonów — szczególnie błyszczących, których rzadkość występowania wynosi do 0,05%. Mogą również zdobywać Pokémony z różnych regionów, niezależnie od typu energii.

Poziom V – Profesorowie (Professors)

Profesorowie nadzorują Trenerów i wspierają ich rozwój w centrum. Mogą awansować jednostki na wyższe rangi, wprowadzać nowe Pokémony i przypisywać je innym. Nie mogą jednak mieć przypisanych do siebie żadnych Pokémonów (możemy wyciąć to żeby mieli pełny dostęp). Profesorowie mają również prawo do modyfikowania statystyk Pokémonów, w tym ich wagi, wzrostu lub etapu ewolucji.

1.3 Modelowe scenariusze

Przedstawiamy poniżej przykładowe sytuacje mogące wystąpić w trakcie funkcjonowania naszego zakładu.

Scenariusz 1:

Odwiedzający przybywa do Centrum Pokémonów i przegląda katalog trenerów i Pokémonów. Przegląda dostępne Pokémony i zapisuje, które go interesują. Nie może jednak wchodzić w interakcje z żadnym trenerem ani odebrać żadnego Pokémona bez rejestracji.

Scenariusz 2:

Trener loguje się do swojego profilu i przegląda Pokémony, które posiada. Bada nieodebrane Pokémony dostępne w swoim regionie i decyduje się odebrać Pokémona w etapie ewolucji 1, który pasuje do typu energii przypisanego mu Pokémona.

Scenariusz 3:

Mistrz przegląda aktualną listę Pokémonów i identyfikuje kilka nieodebranych Pokémonów z innego regionu. Decyduje się odebrać Pokémona w etapie ewolucji 3 i aktualizuje swój profil, aby odzwierciedlić tę zmianę.

Scenariusz 4:

Mistrz decyduje się odebrać błyszczącego Pokémona, którego odkrył w innym regionie. Loguje się, sprawdza dostępność i kończy proces odbierania, dodając rzadkiego Pokémona do swojej kolekcji.

Scenariusz 5:

Profesor przeprowadza spotkanie z Trenerami, aby omówić ich postępy. Awansują Trenera do rangi Mistrza na podstawie jego osiągnięć i wprowadzają nowe Pokémony do centrum, aktualizując dostępny katalog dla wszystkich Trenerów.

Scenariusz 6:

Trener błędnie próbuje przejąć Pokémona, który jest już własnością innego Trenera.

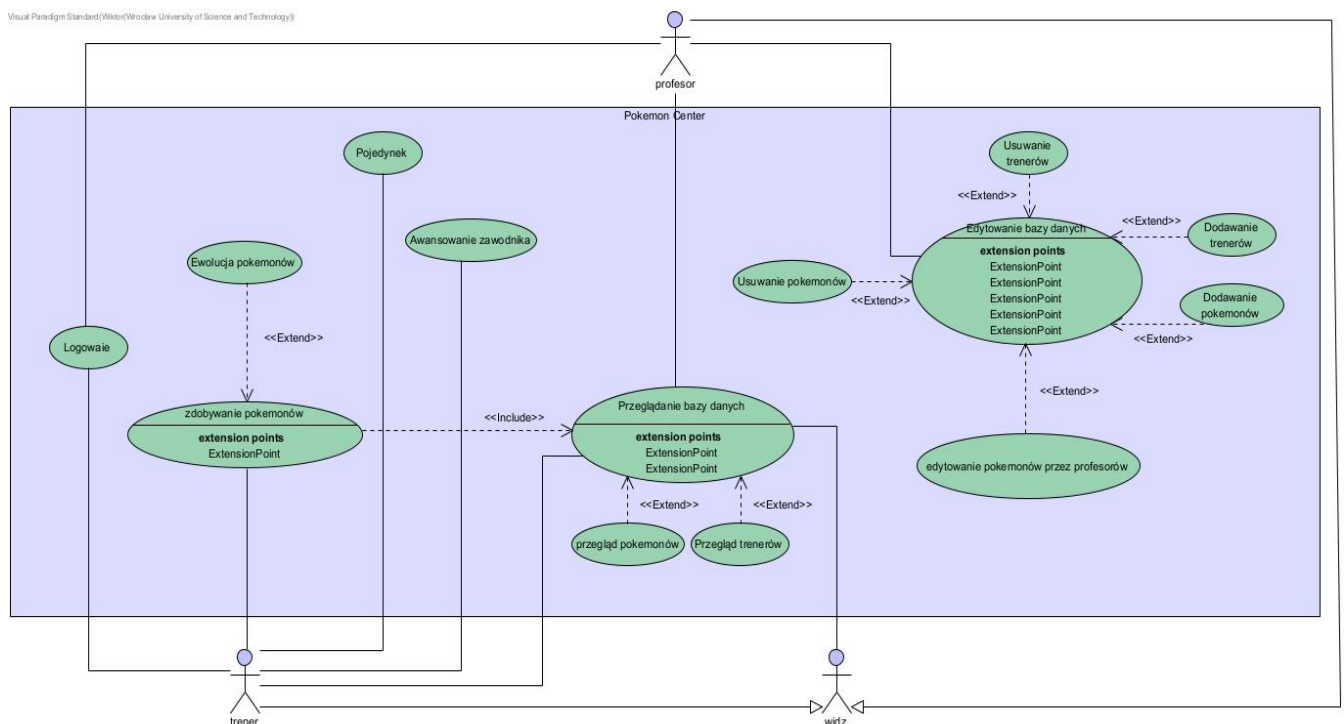
2. Opracowanie wymagań funkcjonalnych i niefunkcjonalnych

2.1 Wymagania funkcjonalne

- Logowanie użytkownika
- Przeglądanie listy pokemonów oraz trenerów
- Odbieranie/przypisywanie/zdobywanie pokemonów
- Mechanizm gry
- Śledzenie i wyświetlanie xp pokemonów oraz zawodników
- Ewolucja pokemonów
- Awansowanie zawodnika
- Dodawanie/usuwanie/edytowanie pokemonów przez profesorów

2.2 Wymagania niefunkcjonalne

- Losowy wybór przeciwników
- Autoryzacja
- Obsługa wyjątków związanych z odbieraniem pokemonów



Rys 1. Diagram przypadków użycia

3. Tekstowy opis wszystkich przypadków użycia

1) Zalogowanie do systemu

- a) **Aktor:** zarejestrowany użytkownik (trener)
- b) **Cel:** Zalogowanie do systemu
- c) **Zdarzenia inicjujące:** Użytkownik uruchamia aplikację i wybiera opcję "Logowanie"
- d) **Warunki wstępne:**
 - Użytkownik musi być zarejestrowany w systemie.
 - Aplikacja jest uruchomiona.
- e) **Warunki końcowe:**
 - Użytkownik jest zalogowany do swojego profilu.
 - W przypadku błędnych danych, użytkownik zostaje poinformowany i pozostaje niezalogowany.
- f) **Scenariusz:**
 - i) Użytkownik uruchamia aplikację.
 - ii) Użytkownik wybiera opcję "Logowanie" z menu.
 - iii) Użytkownik wprowadza swój **nickname** i **hasło**:
 - 1. **Jeśli dane są poprawne:**
 - a) Aplikacja weryfikuje poprawność danych i loguje użytkownika.
 - b) Użytkownik zostaje przekierowany do swojego profilu.
 - c) Następuje powrót do głównego menu aplikacji.
 - 2. **Jeśli dane są błędne:**
 - a) Wyświetlany jest komunikat o niepoprawnych danych logowania.
 - b) Użytkownik ma możliwość ponownej próby wprowadzenia danych.
 - c) Użytkownik pozostaje niezalogowany.
 - d) Aplikacja wraca do ekranu logowania.

2) Rozegranie pojedynku

- a) **Aktor:** zawodnik
- b) **Cel:** Rozegranie pojedynku
- c) **Zdarzenia inicjujące:** Wybranie opcji menu gry
- d) **Warunki wstępne:**
 - Użytkownik jest zalogowany na konto trenera/mistrza/championa
- e) **Warunki końcowe:**
 - Zawodnik bierze udział w grze
 - Następuje powrót do menu gry
- f) **Scenariusz:**
 - i) Gracz wybiera pokemona z listy dostępnych
 - ii) Gracz naciska opcję rozpoczęcia gry

- (1) Jeśli gracz wybrał pokemona:
 - (a) Zostanie rozegrany odpowiedni pojedynek
 - (b) Następuje powrót do menu gry
- (2) Jeśli gracz nie wybrał pokemona:
 - (a) Wyświetlony zostaje komunikat o nie wybraniu żadnego pokemona
 - (b) Pojedynek nie zostanie odegrany
 - (c) Następuje powrót do menu gry

3) Awansowanie zawodnika (trenera na mistrza)

a) **Aktor:** trener

b) **Cel:** Awansowanie na mistrza

c) **Zdarzenia inicjujące:** Trener zdobywa odpowiednią liczbę Pokémonów i zdobywa wymaganą ilość XP.

d) **Warunki wstępne:**

- Trener musi posiadać co najmniej minimalną liczbę Pokémonów (np. 5) oraz minimalną liczbę XP określoną przez system.
- Trener musi być zalogowany w systemie.

e) **Warunki końcowe:**

- Trener automatycznie awansuje na mistrza.
- Trener zyskuje dostęp do nowych opcji i może posiadać więcej Pokémonów (np. zwiększenie limitu z 5 do 15).
- Status trenera zostaje zmieniony na "mistrz" w systemie.

f) **Scenariusz:**

i) Trener zdobył wymaganą liczbę Pokémonów i XP poprzez udział w walkach i inne aktywności.

ii) Aplikacja automatycznie monitoruje liczbę posiadanych Pokémonów oraz XP użytkownika.

1. **Jeśli trener spełnia wymagania do awansu:**

- a) System automatycznie aktualizuje status trenera na **mistrz**.
- b) Wyświetla się komunikat o awansie na mistrza.
- c) Trener zyskuje dostęp do rozszerzonych opcji, takich jak:
 - Możliwość przeglądania większej liczby Pokémonów.
 - Możliwość dodania większej liczby Pokémonów do swojego zespołu (zwiększenie limitu z 5 do 15).
- d) Trener może teraz korzystać z nowych opcji dostępnych dla mistrzów.
- e) System zapisuje zmiany i powraca do głównego menu gry.

2. **Jeśli trener nie spełnia wymagań:**

- a) Trener kontynuuje grę bez zmian w statusie.
- b) System nadal monitoruje postępy trenera, aż spełni wymagania do awansu.

4) Awansowanie zawodnika (z mistrza na czempiona)

a) **Aktor:** mistrz

b) **Cel:** Awansowanie na czempiona (Champion)

c) **Zdarzenia inicjujące:** Mistrz zdobył wymaganą liczbę Pokémonów i XP, w tym Pokémony z różnych regionów.

d) **Warunki wstępne:**

- Mistrz musi posiadać co najmniej wymaganą liczbę Pokémonów (np. 15) oraz odpowiednią liczbę XP.
- Mistrz musi zdobyć Pokémony z różnych regionów.
- Mistrz musi być zalogowany w systemie.

e) **Warunki końcowe:**

- Mistrz awansuje na czempiona.
- Czempion może posiadać do 30 Pokémonów, w tym rzadkie Pokémony typu shiny.
- Status mistrza zostaje zmieniony na "czempion" w systemie.

f) **Scenariusz:**

i) Mistrz zdobył wymaganą liczbę Pokémonów (np. 15) oraz określoną liczbę XP, w tym Pokémony z różnych regionów.

ii) Aplikacja automatycznie monitoruje liczbę posiadanych Pokémonów, XP i różnorodność regionów, z których pochodzą Pokémony.

1. **Jeśli mistrz spełnia wymagania do awansu na czempiona:**

a) System automatycznie aktualizuje status mistrza na **czempion**.

b) Wyświetla się komunikat o awansie na czempiona.

c) Czempion zyskuje dostęp do nowych opcji, takich jak:

- Możliwość przeglądania i łapania do 30 Pokémonów, w tym z różnych regionów.
- Dostęp do rzadkich Pokémonów typu **shiny**.

d) Czempion może teraz korzystać z nowych funkcji, w tym rozszerzonego limitu Pokémonów i dostępu do shiny Pokémonów.

e) System zapisuje zmiany i powraca do głównego menu gry.

2. **Jeśli mistrz nie spełnia wymagań:**

a) Mistrz kontynuuje grę bez zmian w statusie.

b) System nadal monitoruje postępy, aż mistrz spełni wszystkie wymagania do awansu na czempiona.

5) Zdobywanie Pokémonów

a) **Aktor:** trener, mistrz, champion/arcymistrz

b) **Cel:** Zdobywanie Pokémonów do drużyny

c) **Zdarzenia inicjujące:** Użytkownik przegląda listę dostępnych Pokémonów i wybiera nowego Pokémona, którego chce zdobyć.

d) **Warunki wstępne:**

- Użytkownik musi mieć wolne miejsce na nowego Pokémona (np. limit dla trenera to 5 Pokémonów, dla mistrza 15, a dla czempiona 30).
- Użytkownik musi być zalogowany i mieć odpowiedni poziom (trener, mistrz, champion).

e) **Warunki końcowe:**

- Użytkownik zdobywa nowego Pokémona, jeśli spełnia warunki.
- Pokémon zostaje przypisany do użytkownika.

f) **Scenariusz:**

- i) Użytkownik wybiera opcję przeglądania Pokémonów.
- ii) Aplikacja wyświetla listę Pokémonów, które użytkownik może zdobyć, w zależności od jego statusu (trener, mistrz, champion/arcymistrz):
 1. **Trener:** Może wybrać do 5 Pokémonów, z podstawowych gatunków.
 2. **Mistrz:** Może wybrać do 15 Pokémonów, z szerszego wyboru, w tym Pokémony z różnych regionów.
 3. **Czempion:** Może wybrać do 30 Pokémonów, w tym rzadkie i shiny Pokémony.
- iii) Użytkownik wybiera Pokémona, którego chce zdobyć.
 1. **Jeśli użytkownik ma dostępne miejsce na nowego Pokémona:**
 - a) Pokémon zostaje przypisany do użytkownika i dodany do jego drużyny.
 - b) Wyświetla się komunikat potwierdzający zdobycie nowego Pokémona.
 - c) System zapisuje zmiany i powraca do głównego menu.
 2. **Jeśli użytkownik nie ma dostępnego miejsca na nowego Pokémona:**
 - a) Wyświetla się komunikat informujący o braku miejsca na nowego Pokémona.
 - b) Użytkownik nie może zdobyć nowego Pokémona, dopóki nie zwolni miejsca w drużynie.
 - c) System powraca do menu gry.

6) Ewolucja Pokémonów (rozszerzenie "Zdobycia Pokémonów")

- a) **Aktor:** trener, mistrz, champion/arcymistrz
- b) **Cel:** Ewolucja Pokémona po zdobyciu odpowiedniej liczby XP
- c) **Zdarzenia inicjujące:** Pokémon bierze udział w walce i wygrywa, zdobywając XP.
- d) **Warunki wstępne:**
 - Pokémon musi być przypisany do użytkownika.
 - Pokémon musi zdobyć określoną liczbę XP, aby ewoluować.
- e) **Warunki końcowe:**
 - Pokémon ewoluuje, zmieniając swoje parametry (HP, siła, waga, wzrost itp.).
 - Ewolucja zostaje zapisana w systemie.
- f) **Scenariusz:**

- i) Pokémon wygrywa walkę i zdobywa XP.
- ii) System monitoruje zdobywane XP przez każdego Pokémona.
 - 1. **Jeśli Pokémon zdobył wystarczającą liczbę XP do ewolucji:**
 - a) System wyświetla komunikat o możliwości ewolucji Pokémona.
 - b) Pokémon ewoluuje do kolejnej formy:
 - Zmieniają się parametry Pokémona (np. wzrost HP, siła, waga, wzrost).
 - Zmienione zostają jego statystyki w systemie.
 - c) Ewolucja zostaje zapisana i potwierdzona w profilu użytkownika.
 - d) System powraca do głównego menu gry.
 - 2. **Jeśli Pokémon nie zdobył wystarczającej liczby XP:**
 - a) Pokémon kontynuuje zdobywanie XP w kolejnych walkach.
 - b) System nadal monitoruje postępy Pokémona, aż będzie gotowy do ewolucji.

7) Edytowanie bazy danych

- a) **Aktor:** profesor
- b) **Cel:** Edytowanie bazy danych Pokémonów i trenerów
- c) **Zdarzenia inicjujące:** Profesor wybiera opcję edytowania bazy danych.
- d) **Warunki wstępne:**
 - Profesor musi być zalogowany do systemu.
 - Profesor musi mieć uprawnienia do edytowania bazy danych.
- e) **Warunki końcowe:**
 - Zmiany w bazie danych zostają zapisane i odzwierciedlone w systemie.
- f) **Scenariusz:**
 - i) Profesor wybiera opcję **Edytuj bazę danych**.
 - ii) System wyświetla dostępne opcje edytowania: dodawanie, usuwanie lub modyfikowanie rekordów.
 - iii) Profesor wybiera, którą operację chce wykonać (dodanie Pokémona, usunięcie trenera, edycja informacji o Pokémonie itp.).
 - 1. **Jeśli profesor wykonuje poprawne zmiany:**
 - a) System zapisuje zmiany w bazie danych.
 - b) Wyświetla się komunikat o pomyślnej edycji bazy danych.
 - c) System powraca do menu głównego.
 - 2. **Jeśli wystąpił błąd (np. niepoprawne dane):**
 - a) System wyświetla komunikat o błędzie.
 - b) Profesor ma możliwość wprowadzenia poprawnych danych.
 - c) System wraca do opcji edytowania.

8) Dodawanie Pokémonów (rozszerzenie "Edytowanie bazy danych")

a) **Aktor:** profesor

b) **Cel:** Dodanie nowego Pokémona do bazy danych

c) **Zdarzenia inicjujące:** Profesor wybiera opcję dodawania nowego Pokémona.

d) **Warunki wstępne:**

- Profesor musi być zalogowany do systemu i mieć uprawnienia do dodawania danych.

e) **Warunki końcowe:**

- Nowy Pokémon zostaje dodany do bazy danych.

f) **Scenariusz:**

- i) Profesor wybiera opcję **Dodaj Pokémona** z menu edytowania bazy danych.
- ii) System wyświetla formularz do wprowadzenia danych nowego Pokémona (nazwa, gatunek, waga, wzrost, ewolucja itp.).
- iii) Profesor wypełnia dane i zatwierdza formularz.

1. **Jeśli dane są poprawne:**

- a) System zapisuje nowego Pokémona w bazie danych.
- b) Wyświetla się komunikat o pomyślnym dodaniu Pokémona.
- c) System powraca do menu edytowania bazy danych.

2. **Jeśli dane są niepoprawne:**

- a) System wyświetla komunikat o błędzie.
- b) Profesor ma możliwość poprawienia danych i ponownego dodania.

9) Usuwanie Pokémonów (rozszerzenie "Edytowanie bazy danych")

a) **Aktor:** profesor

b) **Cel:** Usunięcie Pokémona z bazy danych

c) **Zdarzenia inicjujące:** Profesor wybiera opcję usunięcia Pokémona.

d) **Warunki wstępne:**

- Pokémon musi znajdować się w bazie danych.

e) **Warunki końcowe:**

- Pokémon zostaje usunięty z bazy danych.

f) **Scenariusz:**

- i) Profesor wybiera opcję **Usuń Pokémona** z menu edytowania bazy danych.
- ii) System wyświetla listę Pokémonów dostępnych w bazie danych.
- iii) Profesor wybiera Pokémona, którego chce usunąć, i zatwierdza decyzję.

- 1. **Jeśli Pokémon istnieje:**
 - a) System usuwa Pokémona z bazy danych.
 - b) Wyświetla się komunikat o pomyślnym usunięciu Pokémona.
 - c) System powraca do menu edytowania bazy danych.
- 2. **Jeśli Pokémon nie istnieje:**
 - a) System wyświetla komunikat o braku takiego Pokémona w bazie.
 - b) Profesor ma możliwość wyboru innego Pokémona.

10) Dodawanie trenera (rozszerzenie "Edytowanie bazy danych")

- a) **Aktor:** profesor
- b) **Cel:** Dodanie nowego trenera do bazy danych
- c) **Zdarzenia inicjujące:** Profesor wybiera opcję dodania nowego trenera.
- d) **Warunki wstępne:**
 - Profesor musi być zalogowany do systemu i mieć uprawnienia do dodawania danych.
- e) **Warunki końcowe:**
 - Nowy trener zostaje dodany do bazy danych.
- f) **Scenariusz:**
 - i) Profesor wybiera opcję **Dodaj trenera** z menu edytowania bazy danych.
 - ii) System wyświetla formularz do wprowadzenia danych nowego trenera (nickname, hasło, e-mail itp.).
 - iii) Profesor wypełnia dane i zatwierdza formularz.

- 1. **Jeśli dane są poprawne:**
 - a) System zapisuje nowego trenera w bazie danych.
 - b) Wyświetla się komunikat o pomyślnym dodaniu trenera.
 - c) System powraca do menu edytowania bazy danych.
- 2. **Jeśli dane są niepoprawne:**
 - a) System wyświetla komunikat o błędzie.
 - b) Profesor ma możliwość poprawienia danych i ponownego dodania.

11) Usuwanie trenera (rozszerzenie "Edytowanie bazy danych")

- a) **Aktor:** profesor

- b) **Cel:** Usunięcie trenera z bazy danych
- c) **Zdarzenia inicjujące:** Profesor wybiera opcję usunięcia trenera.
- d) **Warunki wstępne:**
- Trener musi znajdować się w bazie danych.
- e) **Warunki końcowe:**
- Trener zostaje usunięty z bazy danych.
- f) **Scenariusz:**
- Profesor wybiera opcję **Usuń trenera** z menu edytowania bazy danych.
 - System wyświetla listę trenerów dostępnych w bazie danych.
 - Profesor wybiera trenera, którego chce usunąć, i zatwierdza decyzję.
- Jeśli trener istnieje:**
 - System usuwa trenera z bazy danych.
 - Wyświetla się komunikat o pomyślnym usunięciu trenera.
 - System powraca do menu edytowania bazy danych.
 - Jeśli trener nie istnieje:**
 - System wyświetla komunikat o braku takiego trenera w bazie.
 - Profesor ma możliwość wyboru innego trenera.

12) Edytowanie Pokémonów (rozszerzenie "Edytowanie bazy danych")

- a) **Aktor:** profesor
- b) **Cel:** Edytowanie informacji o Pokémonie (ewolucja, rzadkość, waga, wzrost, itp.)
- c) **Zdarzenia inicjujące:** Profesor wybiera opcję edycji danych Pokémona.
- d) **Warunki wstępne:**
- Pokémon musi istnieć w bazie danych.
- e) **Warunki końcowe:**
- Informacje o Pokémonie zostaną zaktualizowane w systemie.
- f) **Scenariusz:**
- Profesor wybiera opcję **Edytuj Pokémona** z menu edytowania bazy danych.
 - System wyświetla listę Pokémonów dostępnych w bazie danych.
 - Profesor wybiera Pokémona, którego chce edytować, i wprowadza zmiany (np. ewolucja, rzadkość, waga, wzrost).

1. **Jeśli dane są poprawne:**
 - a) System zapisuje zmienione informacje o Pokémonie w bazie danych.
 - b) Wyświetla się komunikat o pomyślnym zaktualizowaniu danych.
 - c) System powraca do menu edytowania bazy danych.
2. **Jeśli dane są niepoprawne:**
 - a) System wyświetla komunikat o błędzie.
 - b) Profesor ma możliwość poprawienia danych i ponownego zapisania.

13) Przeglądanie bazy danych

- a) **Aktorzy:** profesor, trener, widz
- b) **Cel:** Przeglądanie dostępnych danych o Pokémonach i trenerach
- c) **Zdarzenia inicjujące:** Użytkownik wybiera opcję przeglądania bazy danych.
- d) **Warunki wstępne:**
 - Widz nie musi być zalogowany.
 - Trener i profesor muszą być zalogowani, aby mieć pełny dostęp do funkcji.
- e) **Warunki końcowe:**
 - Użytkownik przegląda informacje o Pokémonach i trenerach w bazie danych.
- f) **Scenariusz:**
 - i) Użytkownik wybiera opcję **Przeglądaj bazę danych**.
 - ii) System wyświetla opcje przeglądania: **Przeglądaj Pokémony** i **Przeglądaj trenerów**.
 1. **Jeśli użytkownik wybiera opcję przeglądania Pokémonów:**
 - a) System wyświetla listę Pokémonów z informacjami, które są dostępne w zależności od roli użytkownika (trener, profesor, widz).
 - b) Użytkownik może wybrać konkretnego Pokémona do wyświetlenia szczegółowych informacji.
 2. **Jeśli użytkownik wybiera opcję przeglądania trenerów:**
 - a) System wyświetla listę zarejestrowanych trenerów.
 - b) Użytkownik może wybrać trenera, aby zobaczyć szczegółowe informacje (np. liczba Pokémonów, status).

14) Przeglądanie Pokémonów (rozszerzenie "Przeglądanie bazy danych")

- a) **Aktorzy:** profesor, trener, widz
- b) **Cel:** Przeglądanie szczegółowych informacji o dostępnych Pokémonach
- c) **Zdarzenia inicjujące:** Użytkownik wybiera opcję przeglądania Pokémonów.

d) **Warunki wstępne:**

- Widz, trener, i profesor mogą przeglądać Pokémony, ale zakres dostępnych informacji zależy od roli użytkownika.

e) **Warunki końcowe:**

- Użytkownik przegląda dostępne informacje o wybranych Pokémonach.

f) **Scenariusz:**

i) Użytkownik wybiera opcję **Przeglądaj Pokémony** z menu przeglądania bazy danych.

ii) System wyświetla listę Pokémonów.

iii) Użytkownik wybiera Pokémona, aby zobaczyć szczegóły:

1. **Widz:** Widzi zdjęcie, imię Pokémona, typ energii, stopień ewolucji oraz właściciela (dla rzadkich Pokémonów).
2. **Trener i profesor:** Mają dostęp do pełnych danych, w tym wzrostu, wagi, XP, HP, siły, i innych statystyk.
3. **Jeśli Pokémon jest rzadki:**
 - a) Wyświetla się dodatkowa informacja o jego właścicielu.
4. **Jeśli Pokémon nie jest rzadki:**
 - a) Standardowe informacje są wyświetlane zgodnie z rolą użytkownika.

15) Przeglądanie trenerów (rozszerzenie "Przeglądanie bazy danych")

a) **Aktorzy:** profesor, trener, widz

b) **Cel:** Przeglądanie informacji o zarejestrowanych trenerach

c) **Zdarzenia inicjujące:** Użytkownik wybiera opcję przeglądania trenerów.

d) **Warunki wstępne:**

- Trener i profesor muszą być zalogowani, aby przeglądać pełne dane.

e) **Warunki końcowe:**

- Użytkownik przegląda informacje o wybranych trenerach.

f) **Scenariusz:**

i) Użytkownik wybiera opcję **Przeglądaj trenerów** z menu przeglądania bazy danych.

ii) System wyświetla listę zarejestrowanych trenerów.

iii) Użytkownik wybiera trenera, aby zobaczyć szczegóły:

1. **Widz:** Może przeglądać tylko podstawowe informacje, takie jak imię trenera i status (trener, mistrz, arcymistrz).

2. **Trener i profesor:** Mają dostęp do pełnych danych, takich jak liczba posiadanych Pokémonów, XP, status, i inne.
3. **Jeśli trener jest mistrzem lub czempionem:**
 - a) System wyświetla dodatkowe informacje o statusie trenera.

4. Sformułowanie wymagań

Należy precyzyjnie zdefiniować uprawnienia dostępu dla jedyne go użytkownika naszej aplikacji. Możliwe działania, jakie rozpatrujemy w kontekście sformułowania wymagań to wyświetlanie, modyfikowanie, dodawania lub usuwanie rekordów w tabelach.

Jest to przedstawione dla różnych aktorów: Trener (w tym kontekście to będą i trenery, mistrze i czempiony), Profesor, Widz.

Tabela 1.1: Tabela wymagań dla widza

widz			
	lista użytkowników	lista pokemonów	posiadane pokemony
wyświetlanie	✓	✓	
modyfikacja			
dodawanie			
usuwanie			

Tabela 1.2: Tabela wymagań dla trenerów/graczy

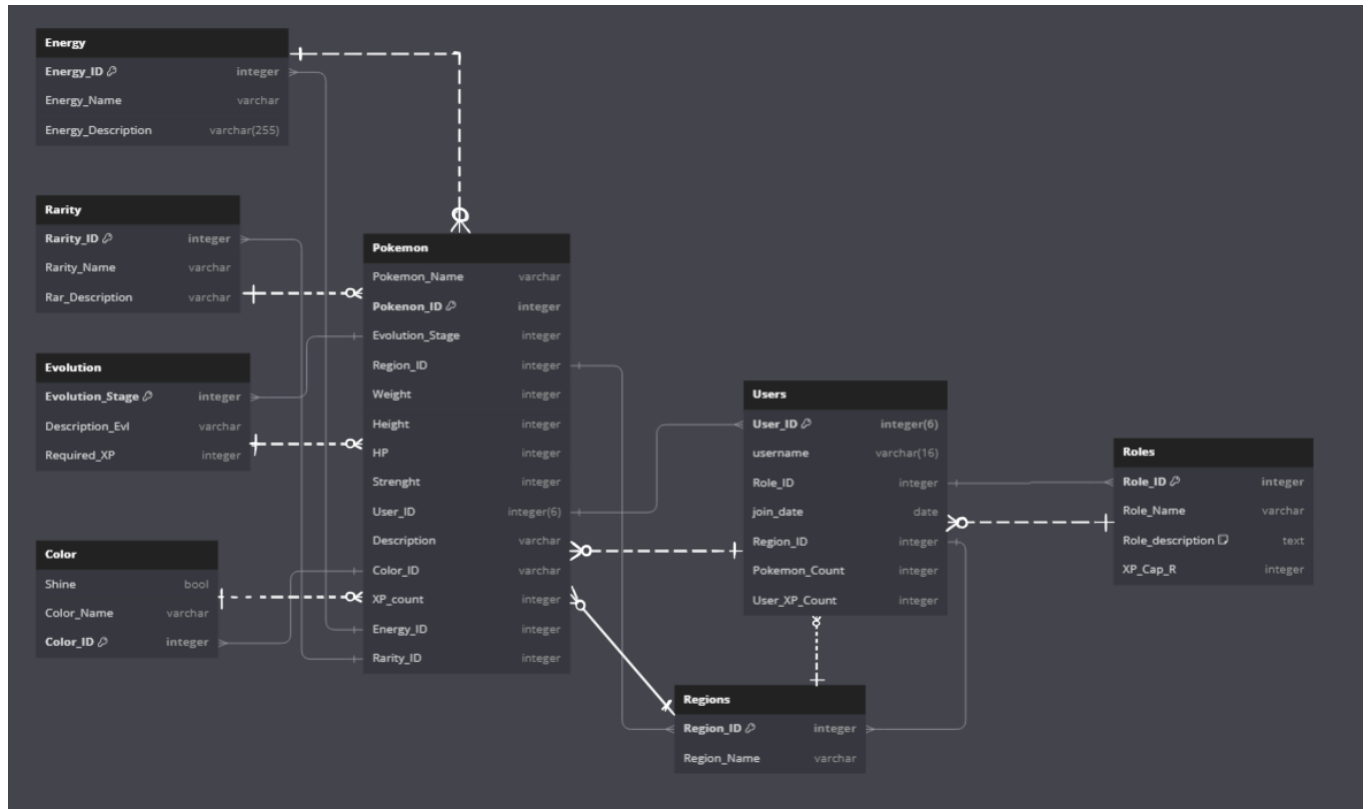
trener			
	lista użytkowników	lista pokemonów	posiadane pokemony
wyświetlanie	✓	✓	✓
modyfikacja			
dodawanie			✓
usuwanie			✓

Tabela 1.3: Tabela wymagań dla profesorów

profesor			
	lista użytkowników	lista pokemonów	posiadane pokemony
wyświetlanie	✓	✓	✓
modyfikacja	✓	✓	✓
dodawanie	✓	✓	
usuwanie	✓	✓	

Z powodu tego, że gra nie jest procesem i częścią bazy danych, a aplikacji, nie jest oznaczony w wymaganiach. Ale to uprawnienie jest tylko u trenerów/graczy.

5. Diagram encji



6. Opis tabel i wypełnienia

6.1 Typ energii:

1. Trawa: Powiązane z naturą i siłą życiową, Pokemony typu trawiastego często skupiają się na regeneracji, leczeniu i umiejętnościach związanych z roślinami i trucizną.
2. Ogień: Energia ognia uosabia ciepło i intensywne płomienie, zapewniając potężne ataki i umiejętności, które mogą powodować oparzenia i poważne obrażenia.
3. Woda: Energia wody jest płynna i adaptacyjna, często kojarzona ze zdolnościami obejmującymi nawodnienie, lód i silne zdolności obronne.
4. Błyskawica/Elektryczność: Ten typ energii jest naładowany elektrycznie, umożliwiając szybkie, paraliżujące ruchy, które polegają na szybkości i nieprzewidywalności.
5. Psychiczna: Energia psychiczna zasila zdolności mentalne, dając Pokémonom moce związane z telekinezą, hipnozą i nadprzyrodzonymi atakami.
6. Walka: Fizyczna i odporna, energia walki wzmacnia bezpośrednie techniki walki wręcz i koncentruje się na sile, wytrzymałości i umiejętnościach.

7. Ciemność: Energia ciemności czerpie z tajemniczych i mrocznych mocy, zapewniając podstępne, destrukcyjne zdolności, często mające na celu dezorientację przeciwników.

8. Wróżka: Energia wróżki jest często mistyczna i czarująca, dając Pokémonom zdolności, które osłabiają lub chronią przed siłami ciemności lub podobnymi do smoków.

9. Smok: Ta energia reprezentuje starożytną moc i wszechstronność, dzięki czemu Pokémony typu smoka są rzadkie i zdolne do złożonych, potężnych ruchów z różnorodnością żywiołów.

6.2 Rzadkość:

1. Zwykle: Pokémony często spotykane w większości regionów, zazwyczaj z podstawowymi zdolnościami. Idealne dla początkujących i łatwe do złapania.

2. Niezwykle: Nieco trudniejsze do znalezienia, niezwykle Pokémony mają ulepszone zdolności lub unikalne cechy, co czyni je bardziej wartościowymi w bitwach.

3. Rzadkie: Rzadkie Pokémony są trudne do znalezienia i często mają wyższe statystyki lub specjalne zdolności, co sprawia, że są bardzo pożądane przez trenerów.

4. Legenda: Legendarne Pokémony są wyjątkowo potężne i zwykle powiązane z mitami. Prawie nigdy nie są widywane na wolności i posiadają niezwykle zdolności. • 5. Shiny Rare: Shiny Pokémon są bardzo pożądane ze względu na swoje unikalne odmiany kolorystyczne i ekstremalną rzadkość. Są identyczne ze swoimi rzadkimi odpowiednikami pod względem siły, ale są cenione za swój wyjątkowy wygląd.

6.3 Regiony:

1. Kanto: Oparty na regionie Kanto w Japonii, Kanto jest oryginalnym regionem, z kultowymi miejscami, takimi jak Pallet Town, gdzie Ash rozpoczyna swoją podróż. Jest znany z mieszanek obszarów miejskich i wiejskich, z punktami orientacyjnymi, takimi jak Mt. Moon i Pokémon League na Indigo Plateau.

2. Johto: Johto jest ściśle związane z Kanto i ma bogate dziedzictwo kulturowe z tradycyjną japońską estetyką. Wprowadza Pokémony z historycznymi i mistycznymi historiami i zawiera Dzwonnicę i Miasto Ecruteak.

3. Hoenn: Oparty na Kiusiu w Japonii, znany z tropikalnego klimatu, Hoenn jest pełen bujnych lasów, plaż i górzystych terenów. Hoenn ma unikalne elementy rozgrywki oparte na pogodzie, skupiając się na naturalnych krajobrazach i legendarnych Pokémonach związanych z lądem, morzem i niebem.

4. Sinnoh: Oparta na japońskiej wyspie Hokkaido, geografia Sinnoh obejmuje góry, obszary śnieżne i starożytne ruiny, kładąc nacisk na tematy mitologii i stworzenia. Ten region jest domem Mt. Coronet i legendarnych powiązanych z czasem i przestrzenią, dodając mistycznego charakteru.

5. Unova: Oparta na Nowym Jorku, Unova jest bardziej przemysłowa i nowoczesna niż poprzednie regiony, z metropolią, Castelia City, jako sercem. Wprowadza Pokémony inspirowane współczesnym życiem i kulturą miejską, symbolizując różnorodność tętniącego życiem miasta.

6. Kalos: Oparta na północnej Francji, Kalos jest znana ze swojej elegancji, historycznej architektury i mody. Posiada zabytki przypominające Wieżę Eiffla (Prism Tower) i wprowadza Mega Evolution,

dając wyrafinowaną i stylową atmosferę. • 7. Alola: Alola, oparta na Hawajach, składa się z tropikalnych wysp o swobodnej kulturze, skupiającej się na pięknie przyrody i szanującej Pokémony jako partnerów.

8. Galar: oparta na Wielkiej Brytanii. Galar to mieszanka obszarów miejskich, wiejskich i przemysłowych, inspirowana kulturą brytyjską. Galar kładzie nacisk na konkurencyjne bitwy Pokémonów na swoich dużych stadionach.

9. Paldea: oparta na Półwyspie Iberyjskim (Hiszpania i Portugalia), Paldea oferuje rozległe, otwarte doświadczenie świata z różnorodnymi ekosystemami, starożytnymi strukturami i kulturowymi akcentami inspirowanymi tradycjami iberyjskimi.

6.4 Ewolucja:

- Etap 1 (podstawowy): To pierwszy etap cyklu życia Pokémona, w którym jest on zazwyczaj najślabszą i najbardziej podstawową formą. Pokémony na tym etapie są często łatwe do znalezienia i złapania.
- Etap 2 (ewolucja): Na tym etapie Pokémon przechodzi ewolucję, stając się silniejszym, z ulepszonymi statystykami i czasami zdobywając nowe umiejętności lub ruchy.
- Etap 3 (ostateczna ewolucja): Ostateczna ewolucja reprezentuje szczyt rozwoju Pokémona, z najwyższymi statystykami i najpotężniejszymi ruchami. Te Pokémony są zazwyczaj rzadsze i trudniej się w nie ewoluje, wymagają określonych metod lub warunków, takich jak wysoki poziom XP lub specjalne przedmioty.

6.5 Kolor:

- Shine (wartość logiczna): Wskazuje, czy Pokémon jest wariantem shiny, czy nie. Shine Pokémon to rzadka wersja o zmienionym kolorze, a ten parametr pomaga odróżnić Pokémony zwykłe od błyszczących.
- Kolor: To pole reprezentuje zwykły kolor Pokémona (np. fioletowy, różowy, niebieski itd.). Pomaga kategoryzować Pokémony według ich typowego wyglądu kolorystycznego.

Aby zebrać informację o pokemonach, bardzo była przydatna strona internetowa "Pokemon.com", rozdział "Pokédex". Na tej stronie można było sprawdzić nazwę, typ energii, fizyczne parametry pokémonów i jeszcze dużo informacji.

7. Identyfikacja Encji

Tabela 2 Identyfikacja encji wraz z identyfikatorami i atrybutami

Encja	Identyfikator	Atrybuty
-------	---------------	----------

Pokemon	Pokemon_ID	Pokemon_Name, Evolution_Stage, Region_ID, Weight, Height, HP, Strength, Information, Color_ID, XP_Count, Energy_ID, Rarity_ID, User_ID
Users	User_ID	Username, Role_ID, Join_Date, Region_ID, Pokemon_Count, User_XP_Count, Status_ID
Roles	Role_ID	Role_Name, Role_Description, XP_Cap
Energy	Energy_ID	Energy_Name
Evolution	Evolution_ID	Description_Evl, Required_XP
Regions	Region_ID	Region_Name
Rarity	Rarity_ID	Rarity_Name, Rar_Description
Color	Color_ID	Shine, Color_Name
Status	Status_ID	Status_Name, Status_Description

Tabela 3 Opis atrybutów encji

Encja	Atrybut	Opis
Pokemon	Pokemon_ID	Identyfikator Pokémona
	Pokemon_Name	Nazwa Pokémona
	Evolution_Stage	Etap ewolucji Pokémona (1-3)
	Region_ID	Identyfikator regionu, z którego pochodzi Pokémon

	Weight	Waga Pokémona w kilogramach
	Height	Wzrost Pokémona w metrach
	HP	Punkty zdrowia Pokémona
	Strength	Siła Pokémona (1-5)
	Information	Opis Pokémona
	Color_ID	Identyfikator koloru Pokémona
	XP_Count	Liczba zdobytych punktów doświadczenia Pokémona
	Energy_ID	Identyfikator energii przypisanej Pokémonowi
	Rarity_ID	Identyfikator rzadkości Pokémona
	User_ID	Identyfikator użytkownika, który posiada Pokémona

Users	User_ID	Identyfikator użytkownika
	Username	Nazwa użytkownika
	Role_ID	Identyfikator roli użytkownika
	Join_Date	Data dołączenia użytkownika
	Region_ID	Identyfikator regionu użytkownika
	Pokemon_Count	Liczba posiadanych Pokémonów
	User_XP_Count	Liczba punktów doświadczenia użytkownika
	Status_ID	Identyfikator statusu użytkownika

Roles	Role_ID	Identyfikator roli
	Role_Name	Nazwa roli
	Role_Description	Opis roli
	XP_Cap	Limit punktów doświadczenia wymaganych dla tej roli

Energy	Energy_ID	Identyfikator energii
	Energy_Name	Nazwa typu energii

Evolution	Evolution_Stage	Identyfikator etapu ewolucji
-----------	-----------------	------------------------------

	Description_Evl	Opis etapu ewolucji
	Required_XP	Liczba punktów doświadczenia wymaganych do przejścia na etap ewolucji

Regions	Region_ID	Identyfikator regionu
	Region_Name	Nazwa regionu

Rarity	Rarity_ID	Identyfikator poziomu rzadkości
	Rarity_Name	Nazwa poziomu rzadkości
	Rar_Description	Opis poziomu rzadkości

Color	Color_ID	Identyfikator koloru
	Shine	Czy Pokémon jest błyszczący (Y/N)
	Color_Name	Nazwa koloru

Status	Status_ID	Identyfikator statusu użytkownika
	Status_Name	Nazwa statusu użytkownika
	Status_Description	Opis statusu użytkownika

8. Opis działania kodu

Musimy stworzyć tabele, aby dodawać do nich informacje i korzystać z nich. Niżej jest przedstawiony kod generowania tabel:

```
-- // CREATING TABLES

-- Table: Regions
CREATE TABLE Regions (
  Region_ID INTEGER PRIMARY KEY NOT NULL,
  Region_Name VARCHAR2(50) NOT NULL
);

-- Table: Roles
CREATE TABLE Roles (
  Role_ID INTEGER PRIMARY KEY NOT NULL,
  Role_Name VARCHAR2(50) NOT NULL,
  Role_Description VARCHAR2(500),
  XP_Cap INTEGER
);

-- Table: Rarity
CREATE TABLE Rarity (
  Rarity_ID INTEGER PRIMARY KEY NOT NULL,
  Rarity_Name VARCHAR2(50) NOT NULL,
  Rar_Description VARCHAR2(500)
);

-- Table: Energy
CREATE TABLE Energy (
  Energy_ID INTEGER PRIMARY KEY NOT NULL,
  Energy_Name VARCHAR2(50) NOT NULL,
  Energy_Description VARCHAR2(500)
);

-- Table: Color
CREATE TABLE Color (
  Color_ID INTEGER PRIMARY KEY,
  Color_Name VARCHAR2(50)
);

-- Table: Users
CREATE TABLE Users (
  User_ID INTEGER PRIMARY KEY NOT NULL,
  Username VARCHAR2(16) UNIQUE NOT NULL,
  Role_ID INTEGER REFERENCES Roles(Role_ID),
  Join_Date DATE NOT NULL,
  Region_ID INTEGER REFERENCES Regions(Region_ID),
  Pokemon_Count INTEGER,
  User_XP_Count INTEGER,
  Status_ID INTEGER REFERENCES Roles(Role_ID)
);
```

```
-- Table: Pokemon
CREATE TABLE Pokemon (
  Pokemon_Name VARCHAR2(50) NOT NULL,
  Pokemon_ID INTEGER PRIMARY KEY NOT NULL,
  Evolution_Stage INTEGER CHECK (Evolution_Stage BETWEEN 1 AND 3) NOT NULL,
  Region_ID INTEGER REFERENCES Regions(Region_ID) NOT NULL,
  Weight INTEGER,
  Height INTEGER,
  HP INTEGER,
  Strength INTEGER CHECK (Strength BETWEEN 1 AND 5) NOT NULL,
  User_ID INTEGER REFERENCES Users(User_ID),
  Information VARCHAR2(255),
  Color_ID INTEGER REFERENCES Color(Color_ID),
  XP_Count INTEGER,
  Energy_ID INTEGER REFERENCES Energy(Energy_ID),
  Rarity_ID INTEGER REFERENCES Rarity(Rarity_ID)
);

ALTER TABLE Users
  ADD CONSTRAINT chk_professor_no_pokemon_xp CHECK (
    (Role_ID != 4) OR (Pokemon_Count IS NULL AND User_XP_Count IS NULL)
  );
```

```
-- Table: Evolution
CREATE TABLE Evolution (
  Evolution_Stage INTEGER PRIMARY KEY CHECK (Evolution_Stage BETWEEN 1 AND 3),
  Description_Evl VARCHAR2(500),
  Required_XP INTEGER
);
```

Dodamy informacje do bazy, przedstawionej powyżej w dokumencie.

Wypełnienie tabelki „Roles”:

```
-- Roles
INSERT INTO Roles (Role_ID, Role_Name, Role_Description, XP_Cap)
VALUES (1, 'Trainer', 'Trainers are the entry-level users who capture and battle Pokémon. They start with a limited number of Pokémon and gradually
earn XP through battles. Trainers can evolve their Pokémon, participate in battles, and gain experience to move up to higher roles.', 750);

INSERT INTO Roles (Role_ID, Role_Name, Role_Description, XP_Cap)
VALUES (2, 'Master', 'Masters are experienced trainers who have captured a significant number of Pokémon and gained a high level of XP.
They unlock access to more Pokémon and can manage a larger roster.', 1500);

INSERT INTO Roles (Role_ID, Role_Name, Role_Description, XP_Cap)
VALUES (3, 'Champion', 'Champions are the highest level users who have proven themselves by gaining extensive experience and capturing the rarest Pokémon.
They can have up to 30 Pokémon from different regions, including rare and shiny Pokémon. Champions enjoy privileges like access to exclusive content and abilities, along with powerful Pokémon.', NULL);

INSERT INTO Roles (Role_ID, Role_Name, Role_Description, XP_Cap)
VALUES (4, 'Professor', 'Professors manage and curate the Pokémon database, overseeing the creation, modification, and deletion of Pokémon and trainers.
They have full control over the system, allowing them to add new Pokémon, assign evolutions, and edit user and Pokémon details.
Professors ensure the proper functioning of the database and can also track user activity.', NULL);
```

Wypełnienie tabelki „Energy”:

```
-- Inserting values into Energy table
INSERT INTO Energy (Energy_ID, Energy_Name, Energy_Description)
VALUES (1, 'Grass', 'Associated with nature and life force, Grass-type Pokémon often focus on regeneration, healing, and abilities tied to plants and poison.');
```

```
INSERT INTO Energy (Energy_ID, Energy_Name, Energy_Description)
VALUES (2, 'Fire', 'Fire energy embodies heat and intense flames, granting powerful attacks and abilities that can cause burns and severe damage.');
```

```
INSERT INTO Energy (Energy_ID, Energy_Name, Energy_Description)
VALUES (3, 'Water', 'Water energy is fluid and adaptable, often associated with abilities involving hydration, ice, and strong defensive capabilities.');
```

```
INSERT INTO Energy (Energy_ID, Energy_Name, Energy_Description)
VALUES (4, 'Lightning/Electric', 'This energy type is electrically charged, enabling swift, paralyzing moves that rely on speed and unpredictability.');
```

```
INSERT INTO Energy (Energy_ID, Energy_Name, Energy_Description)
VALUES (5, 'Psychic', 'Psychic energy powers mental abilities, giving Pokémon powers related to telekinesis, hypnosis, and supernatural attacks.');
```

```
INSERT INTO Energy (Energy_ID, Energy_Name, Energy_Description)
VALUES (6, 'Fighting', 'Physical and resilient, Fighting energy empowers direct, close-combat techniques and focuses on strength, endurance, and skill.');
```

```
INSERT INTO Energy (Energy_ID, Energy_Name, Energy_Description)
VALUES (7, 'Darkness', 'Darkness energy taps into mysterious and shadowy powers, granting sneaky, disruptive abilities often aimed at confounding opponents.');
```

```
INSERT INTO Energy (Energy_ID, Energy_Name, Energy_Description)
VALUES (8, 'Fairy', 'Fairy energy is often mystical and charming, giving Pokémon abilities that weaken or protect against dark or dragon-like forces.');
```

```
INSERT INTO Energy (Energy_ID, Energy_Name, Energy_Description)
VALUES (9, 'Dragon', 'This energy represents ancient power and versatility, making Dragon-type Pokémon rare and capable of complex, powerful moves with elemental variety.');
```

Wypełnienie tabelki „Evolution”:

```
-- Inserting values into Evolution table
INSERT INTO Evolution (Evolution_Stage, Description_Ev1, Required_XP)
VALUES (1, 'Basic: This is the first stage of a Pokémon's life cycle, where it is typically the weakest and most basic form. Pokémon at this stage are often easily found and captured.', 200);

INSERT INTO Evolution (Evolution_Stage, Description_Ev1, Required_XP)
VALUES (2, 'Evolution: In this stage, the Pokémon undergoes an evolution, growing stronger with improved stats and sometimes acquiring new abilities or moves.', 500);

INSERT INTO Evolution (Evolution_Stage, Description_Ev1, Required_XP)
VALUES (3, 'Final Evolution: The final evolution represents the peak of a Pokémon's growth, with the highest stats and most powerful moves.
These Pokémon are usually rarer and more difficult to evolve into, requiring specific methods or conditions like high-level XP or special items.', 1000);
```

Wypełnienie tabelki „Regions”:

```
--REGIONS:

-- Inserting values into Regions table
INSERT INTO Regions (Region_ID, Region_Name) VALUES (1, 'Kanto');
INSERT INTO Regions (Region_ID, Region_Name) VALUES (2, 'Johto');
INSERT INTO Regions (Region_ID, Region_Name) VALUES (3, 'Hoenn');
INSERT INTO Regions (Region_ID, Region_Name) VALUES (4, 'Sinnoh');
INSERT INTO Regions (Region_ID, Region_Name) VALUES (5, 'Unova');
INSERT INTO Regions (Region_ID, Region_Name) VALUES (6, 'Kalos');
INSERT INTO Regions (Region_ID, Region_Name) VALUES (7, 'Alola');
INSERT INTO Regions (Region_ID, Region_Name) VALUES (8, 'Galar');
INSERT INTO Regions (Region_ID, Region_Name) VALUES (9, 'Paldea');
```

Wypełnienie tabelki „Rarity”:

```
-- Inserting values into Rarity table
INSERT INTO Rarity (Rarity_ID, Rarity_Name, Rar_Description)
VALUES (1, 'Common', 'Pokémon frequently encountered in most regions, typically with basic abilities. Ideal for beginners and easy to catch.');
```

```
INSERT INTO Rarity (Rarity_ID, Rarity_Name, Rar_Description)
VALUES (2, 'Uncommon', 'Slightly harder to find, Uncommon Pokémon have enhanced abilities or unique traits, making them more valuable in battles.');
```

```
INSERT INTO Rarity (Rarity_ID, Rarity_Name, Rar_Description)
VALUES (3, 'Rare', 'Rare Pokémon are difficult to find and often have higher stats or special abilities, making them highly sought after by trainers.');
```

```
INSERT INTO Rarity (Rarity_ID, Rarity_Name, Rar_Description)
VALUES (4, 'Legend', 'Legendary Pokémon are exceptionally powerful and usually linked to myths. They are almost never seen in the wild and possess extraordinary abilities.');
```

```
INSERT INTO Rarity (Rarity_ID, Rarity_Name, Rar_Description)
VALUES (5, 'Shiny Rare', 'Shiny Pokémon are highly coveted for their unique color variations and extreme rarity. They are identical to their Rare counterparts in strength but are prized for their distinct appearance.');
```

Wypełnienie tabelki „Color”:

```
-- Insert values into the Color table
INSERT INTO Color (Color_ID, Color_Name) VALUES (1, 'Red');
INSERT INTO Color (Color_ID, Color_Name) VALUES (2, 'Green');
INSERT INTO Color (Color_ID, Color_Name) VALUES (3, 'Blue');
INSERT INTO Color (Color_ID, Color_Name) VALUES (4, 'Purple');
INSERT INTO Color (Color_ID, Color_Name) VALUES (5, 'Yellow');
INSERT INTO Color (Color_ID, Color_Name) VALUES (6, 'Pink');
INSERT INTO Color (Color_ID, Color_Name) VALUES (7, 'Black');
INSERT INTO Color (Color_ID, Color_Name) VALUES (8, 'Gray');
INSERT INTO Color (Color_ID, Color_Name) VALUES (9, 'Orange');
INSERT INTO Color (Color_ID, Color_Name) VALUES (10, 'White');
INSERT INTO Color (Color_ID, Color_Name) VALUES (11, 'Brown');
INSERT INTO Color (Color_ID, Color_Name) VALUES (12, 'Beige');
INSERT INTO Color (Color_ID, Color_Name) VALUES (13, 'Ivory');
INSERT INTO Color (Color_ID, Color_Name) VALUES (14, 'Silver');
INSERT INTO Color (Color_ID, Color_Name) VALUES (15, 'Gold');
INSERT INTO Color (Color_ID, Color_Name) VALUES (16, 'Teal');
INSERT INTO Color (Color_ID, Color_Name) VALUES (17, 'Cyan');
INSERT INTO Color (Color_ID, Color_Name) VALUES (18, 'Magenta');
```

Wypełnienie tabelki „Pokemon”:

Dlatego, że musimy uwzględnić ewolucje pokémonów, musimy dodać dużo pozycji. Nie jest niżej przedstawiony kod w całości, a tylko przykład dodawania Pokemona „Bulbasaur” i go ewolucje:

```
-- Insert for Bulbasaur and its evolutions (Kanto Region)
INSERT INTO Pokemon (Pokemon_Name, Pokemon_ID, Evolution_Stage, Region_ID, Weight, Height, HP, Strength, Information, Color_ID, Energy_ID, Rarity_ID)
VALUES ('Bulbasaur', 1, 1, 1, 6.9, 0.7, 45, 3, 'A Grass/Poison-type Pokémon with a plant bulb on its back.', 3, 1, 1);

INSERT INTO Pokemon (Pokemon_Name, Pokemon_ID, Evolution_Stage, Region_ID, Weight, Height, HP, Strength, Information, Color_ID, Energy_ID, Rarity_ID)
VALUES ('Ivysaur', 2, 2, 1, 13.0, 1.0, 60, 4, 'The evolved form of Bulbasaur, the bulb on its back begins to bloom.', 3, 1, 2);

INSERT INTO Pokemon (Pokemon_Name, Pokemon_ID, Evolution_Stage, Region_ID, Weight, Height, HP, Strength, Information, Color_ID, Energy_ID, Rarity_ID)
VALUES ('Venusaur', 3, 3, 1, 100.0, 2.0, 80, 5, 'Known for its powerful attacks and large, blooming flower.', 3, 1, 3);
```

Jak widzimy, Pokemony mają własne ID. Dlatego, że możemy znaleźć Bulbasaura w regionie „Kanto” – Region_ID jest ustawiony na 1. Jest zielony – Color_ID = 3. Go typ energii – Trawa, wtedy Energy_ID jest 1. Tabela jest wypełniona około 150 różnych wariacji pokémonów.

Wypełnienie tabelki „Users”:

Dodaaliśmy nowych trenerów, masterzy, czempionów i profesorów do naszej tabelki. Są oni z różnych regionów i później dodaliśmy manualnie pokemonów sposobem: „gdzie ID_Osoby = A”, przypisz my pokemona z ID_Pokemona = B, następnie przypisz do osoby z ID_Osoby = A pokemona z ID_Pokemona = B.

```
--ADD USERS

-- Ash Ketchum (Trainer, Kanto region)
INSERT INTO Users (User_ID, Username, Role_ID, Join_Date, Region_ID, Pokemon_Count, User_XP_Count, Status_ID)
VALUES (1, 'Ash_Ketchum', 1, TO_DATE('1997-04-01', 'YYYY-MM-DD'), 1, 10, 500, 1); -- Role_ID 1 (Trainer), Region_ID 1 (Kanto)

-- Professor Oak (Professor, Kanto region)
INSERT INTO Users (User_ID, Username, Role_ID, Join_Date, Region_ID, Pokemon_Count, User_XP_Count, Status_ID)
VALUES (2, 'Professor_Oak', 4, TO_DATE('1996-04-01', 'YYYY-MM-DD'), 1, NULL, NULL, 4); -- Role_ID 4 (Professor), Region_ID 1 (Kanto)

-- Blue (Champion, Kanto region)
INSERT INTO Users (User_ID, Username, Role_ID, Join_Date, Region_ID, Pokemon_Count, User_XP_Count, Status_ID)
VALUES (3, 'Blue', 3, TO_DATE('1998-05-01', 'YYYY-MM-DD'), 1, 20, 2500, 3); -- Role_ID 3 (Champion), Region_ID 1 (Kanto)

-- Professor Elm (Professor, Johto region)
INSERT INTO Users (User_ID, Username, Role_ID, Join_Date, Region_ID, Pokemon_Count, User_XP_Count, Status_ID)
VALUES (4, 'Professor_Elm', 4, TO_DATE('1999-03-01', 'YYYY-MM-DD'), 2, NULL, NULL, 4); -- Role_ID 4 (Professor), Region_ID 2 (Johto)

-- Red (Master, Kanto region)
INSERT INTO Users (User_ID, Username, Role_ID, Join_Date, Region_ID, Pokemon_Count, User_XP_Count, Status_ID)
VALUES (5, 'Red', 2, TO_DATE('1996-04-01', 'YYYY-MM-DD'), 1, 30, 1200, 2); -- Role_ID 2 (Master), Region_ID 1 (Kanto)

-- May (Trainer, Hoenn region)
INSERT INTO Users (User_ID, Username, Role_ID, Join_Date, Region_ID, Pokemon_Count, User_XP_Count, Status_ID)
VALUES (6, 'May', 1, TO_DATE('2002-01-01', 'YYYY-MM-DD'), 3, 12, 700, 1); -- Role_ID 1 (Trainer), Region_ID 3 (Hoenn)
```

Niżej jest przedstawiony kod na przypisanie pokemona do osoby i osoby do pokemona:

```
--Assign Pokemons

-- Assign Pikachu (Pokemon_ID = 25) to AshKetchum (User_ID = 1)
UPDATE Pokemon
SET User_ID = 1 -- Assigning Pikachu to Ash
WHERE Pokemon_ID = 25;

-- Assign Mewtwo (Pokemon_ID = 150) to Red (User_ID = 2)
UPDATE Pokemon
SET User_ID = 2 -- Assigning Mewtwo to Red
WHERE Pokemon_ID = 150;

-- Assign Dragonite (Pokemon_ID = 149) to Lance (User_ID = 3)
UPDATE Pokemon
SET User_ID = 3 -- Assigning Dragonite to Lance
WHERE Pokemon_ID = 149;
```

Mamy parametr „Pokemon_Count” u użytkowników, czyli musimy zrobić to przez „count”. Możemy to zrobić też jako funkcję, żeby robiło się to automatycznie.

```
-- count
-- Update Pokemon_Count for AshKetchum (User_ID = 1)
UPDATE Users
SET Pokemon_Count = (SELECT COUNT(*) FROM Pokemon WHERE User_ID = 1)
WHERE User_ID = 1;
```

9. Testy Bazy

Sprawdzenie czy Username ma maksymalną długość 16 znaków:

```
SELECT COLUMN_NAME, CHARACTER_MAXIMUM_LENGTH
FROM INFORMATION_SCHEMA.COLUMNS
WHERE TABLE_NAME = 'Users' AND COLUMN_NAME = 'Username';
```

	COLUMN_NAME	CHARACTER_MAXIMUM_LENGTH
▶	Username	16

Sprawdzenie czy User_ID (klucz główny) jest unikalne w tabeli Users:

```
SELECT User_ID, COUNT(*)  
FROM Users  
GROUP BY User_ID  
HAVING COUNT(*) > 1;
```

	User_ID	COUNT(*)

Sprawdzenie, czy kolumna Strength w tabeli Pokemon ma wartość pomiędzy 1 a 5 i nie jest NULL

```
SELECT Pokemon_ID, Strength  
FROM Pokemon  
WHERE Strength NOT BETWEEN 1 AND 5 OR Strength IS NULL;
```

	Pokemon_ID	Strength
*	NULL	NULL

Sprawdzenie, czy Pokemon_ID nie jest NULL w tabeli Pokemon

```
SELECT COUNT(*)  
FROM Pokemon  
WHERE Pokemon_ID IS NULL;
```

	COUNT(*)
▶	0

Sprawdzenie, czy wartości Color_ID w tabeli Pokemon odpowiadają wartościom Color_ID w tabeli Color

```
SELECT p.Color_ID
FROM Pokemon p
LEFT JOIN Color c ON p.Color_ID = c.Color_ID
WHERE c.Color_ID IS NULL AND p.Color_ID IS NOT NULL;
```

Color_ID

Sprawdzenie, czy wartości Energy_ID w tabeli Pokemon odpowiadają wartościom Energy_ID w tabeli Energy

```
SELECT p.Energy_ID
FROM Pokemon p
LEFT JOIN Energy e ON p.Energy_ID = e.Energy_ID
WHERE e.Energy_ID IS NULL AND p.Energy_ID IS NOT NULL;
```

Energy_ID
.

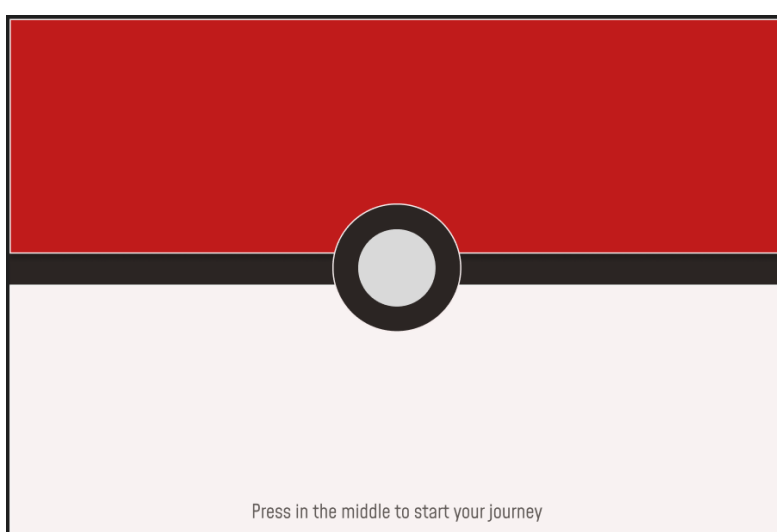
Sprawdzenie, czy kolumna Username ma maksymalną długość 16 znaków

```
SELECT COLUMN_NAME, CHARACTER_MAXIMUM_LENGTH
FROM INFORMATION_SCHEMA.COLUMNS
WHERE TABLE_NAME = 'Users' AND COLUMN_NAME = 'Username';
```

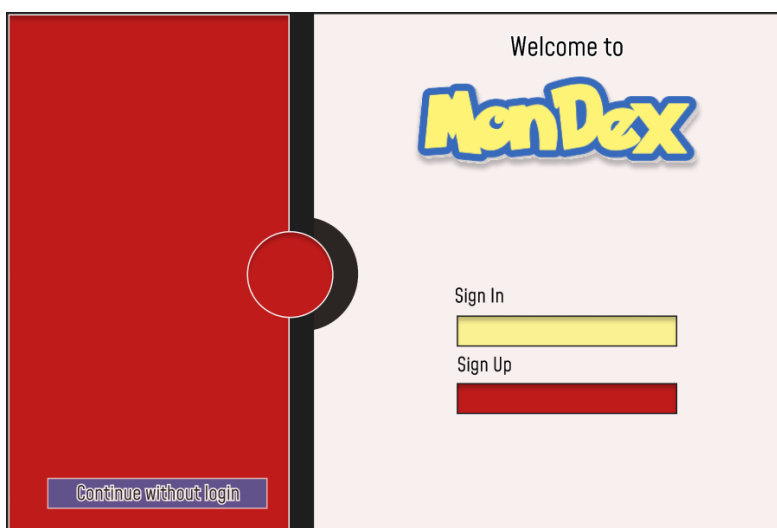
	COLUMN_NAME	CHARACTER_MAXIMUM_LENGTH
►	Username	16

10. Makieta interfejsu graficznego

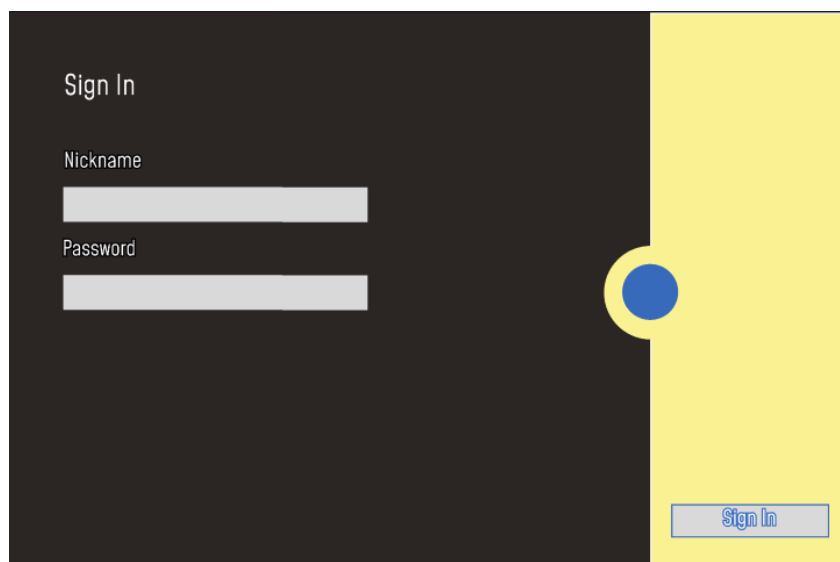
Przed przystąpieniem do implementacji interfejsu graficznego stworzono poglądowe makiety. Aby wyglądało to jako prawidłowa aplikacja, stworzyliśmy następujące makiety interfejsów graficznych: Start aplikacji (Rys. 1.1), Menu wyboru logowania (Rys. 1.2), Interfejs logowania (Rys. 1.3), Interfejs rejestracji (Rys. 1.4), Główne Menu do użytkownika zalogowanego, niezalogowanego i profesora (Rys. 1.5.1, 1.5.2, 1.5.3), Interfejsy wyszukiwania Pokémonów dla użytkownika zalogowanego i niezalogowanego (Rys. 1.6.1, 1.6.2), Interfejs wyszukiwania Trenerów dla użytkownika zalogowanego i niezalogowanego (Rys. 1.7.1, 1.7.2), Interfejs wyświetlania pokemonów trenerów (Rys. 1.8), Interfejs dodawania i usuwania Pokémonów dla trenerów (Rys. 1.9), Interfejs Gry/Walki (Rys. 1.10), Interfejs zmiany roli trenera (dla profesora) (Rys. 1.11), Interfejs dodawania/usunięcia pokemona z bazy (Rys. 1.12). Wszystkie makiety zostały zrobione przez aplikację Figma.



Rys. 1.1 Makieta interfejsu 'Start Aplikacji'



Rys. 1.2 Makieta menu wyboru logowania



Sign In

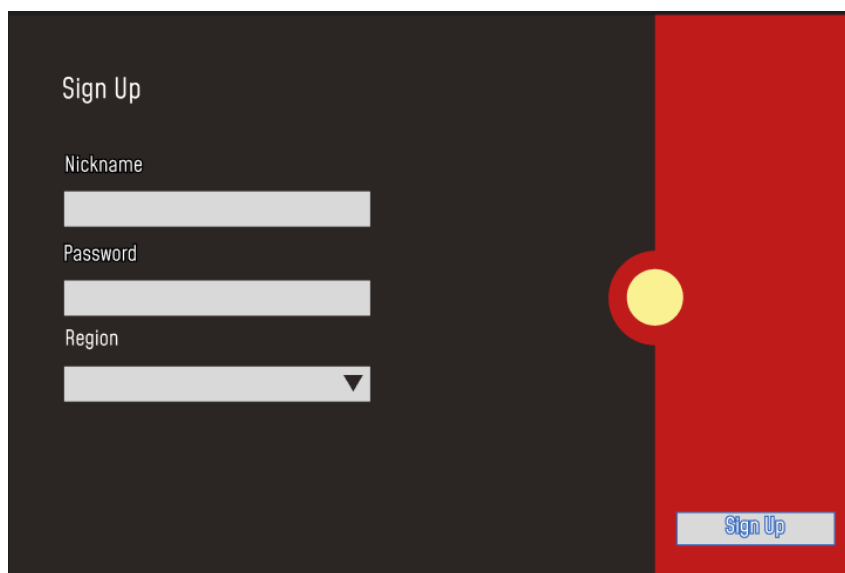
Nickname

Password

Sign In

The image shows a 'Sign In' form on a dark grey background. It features two input fields: 'Nickname' and 'Password'. A yellow vertical bar is on the right, with a blue circle and a yellow ring. A 'Sign In' button is at the bottom right.

Rys. 1.3 Makieta Interfejsu logowania



Sign Up

Nickname

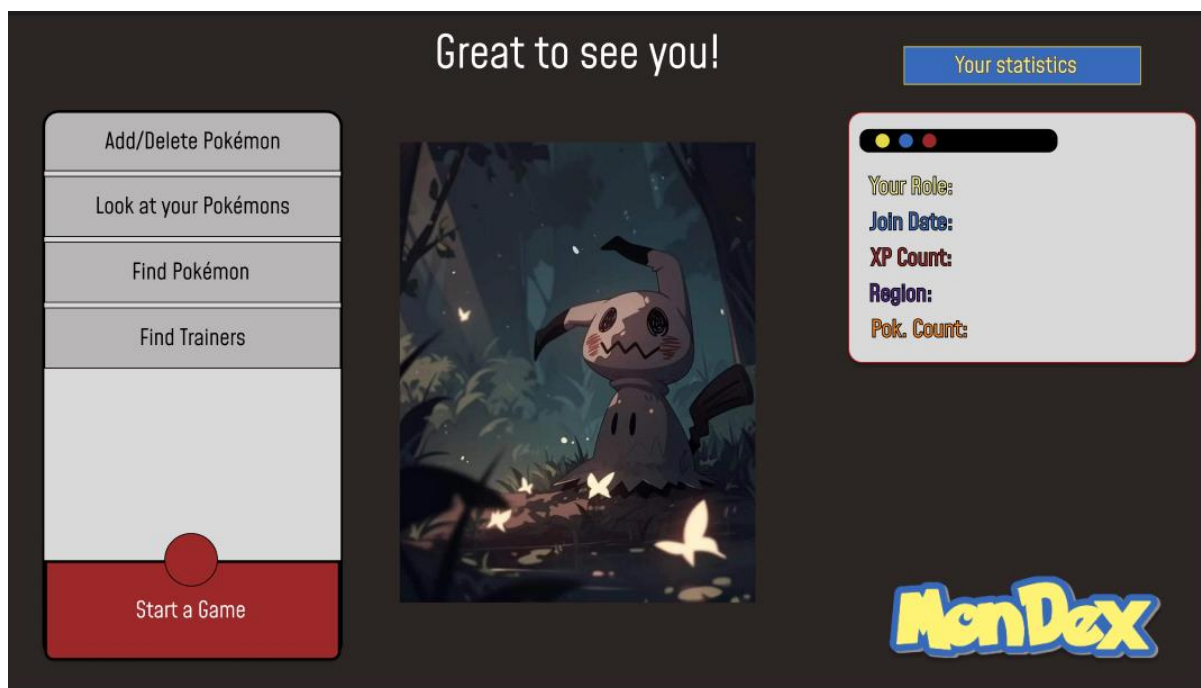
Password

Region

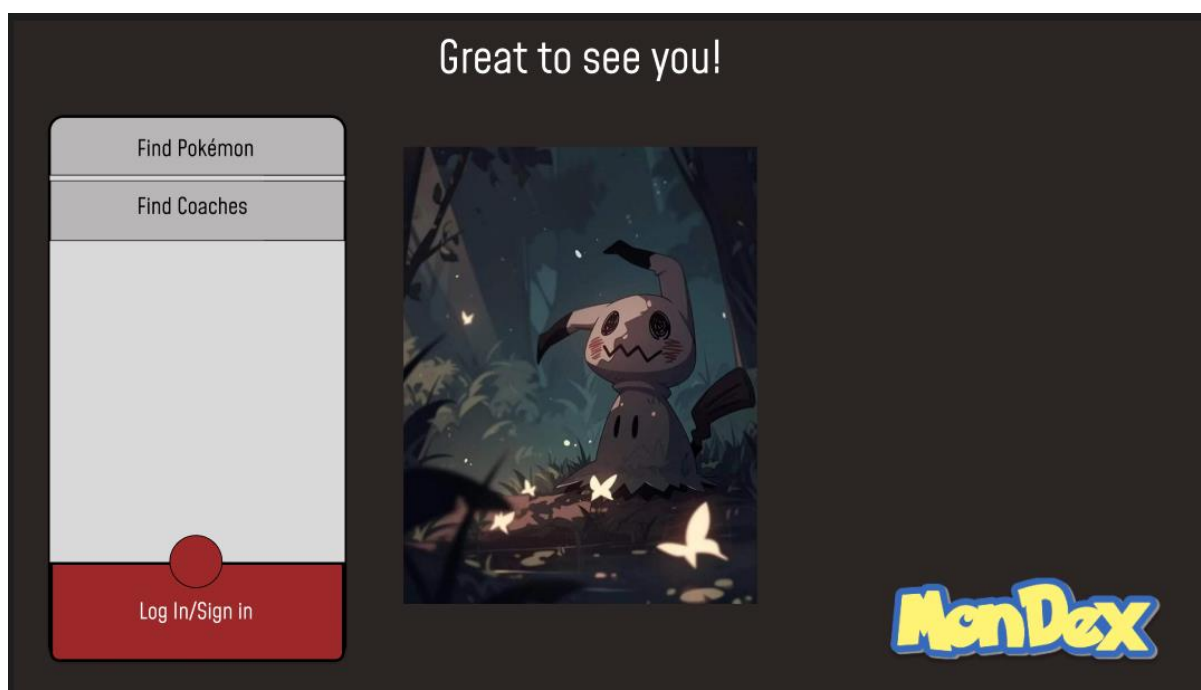
Sign Up

The image shows a 'Sign Up' form on a dark grey background. It features three input fields: 'Nickname', 'Password', and 'Region'. A red vertical bar is on the right, with a yellow circle and a red ring. A 'Sign Up' button is at the bottom right.

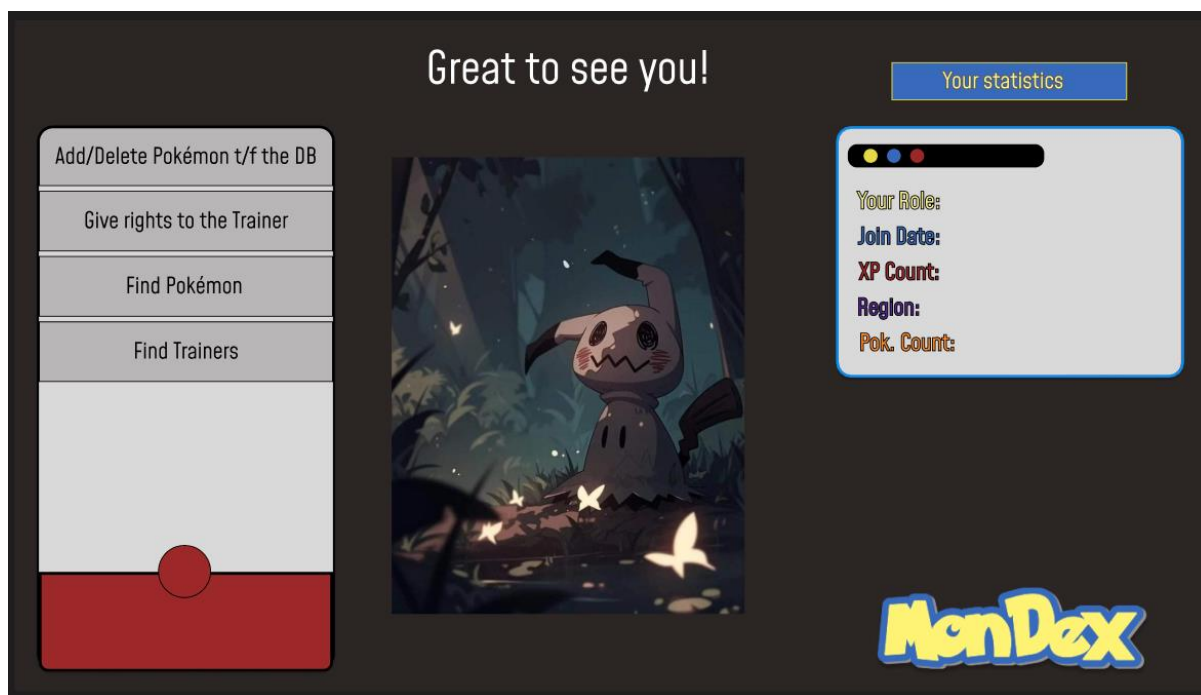
Rys. 1.4 Makieta Interfejsu rejestracji



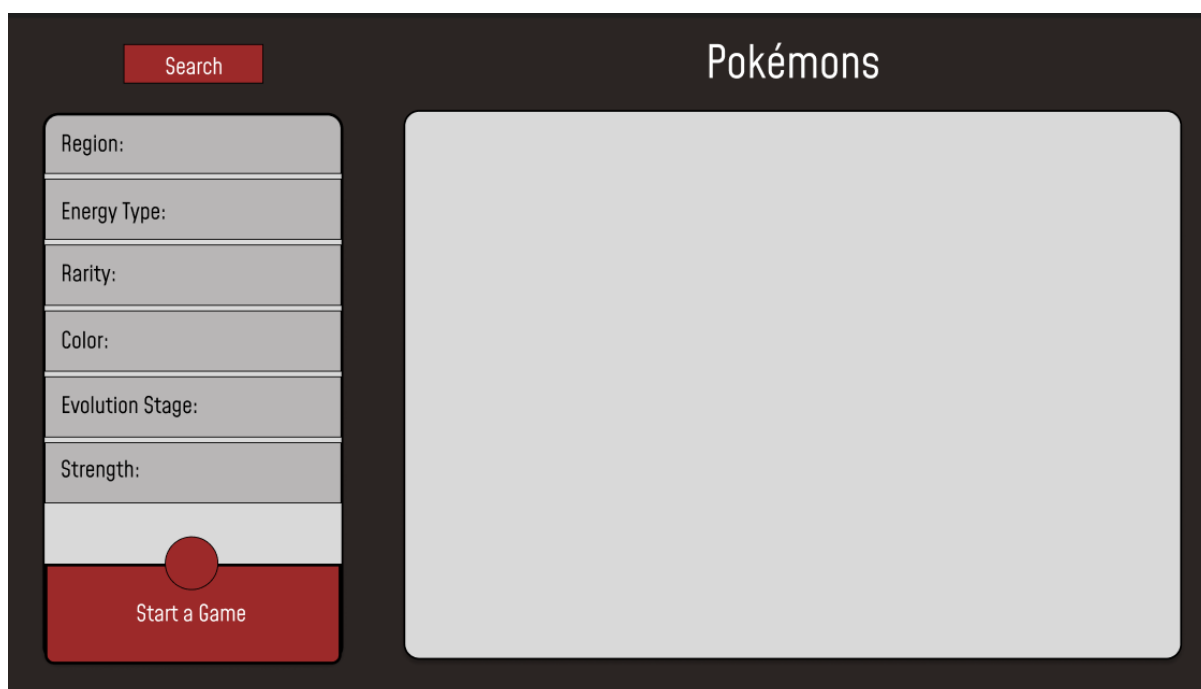
Rys. 1.5.1 Makieta głównego menu do użytkownika zalogowanego



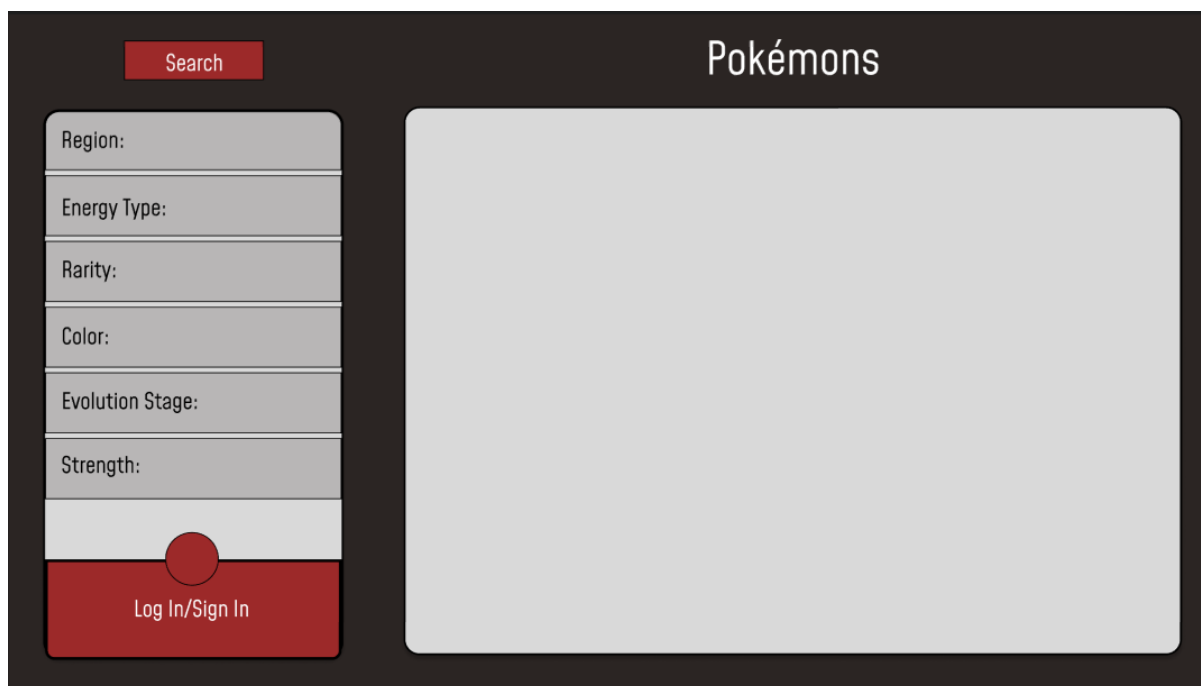
Rys. 1.5.2 Makieta głównego menu do użytkownika niezalogowanego



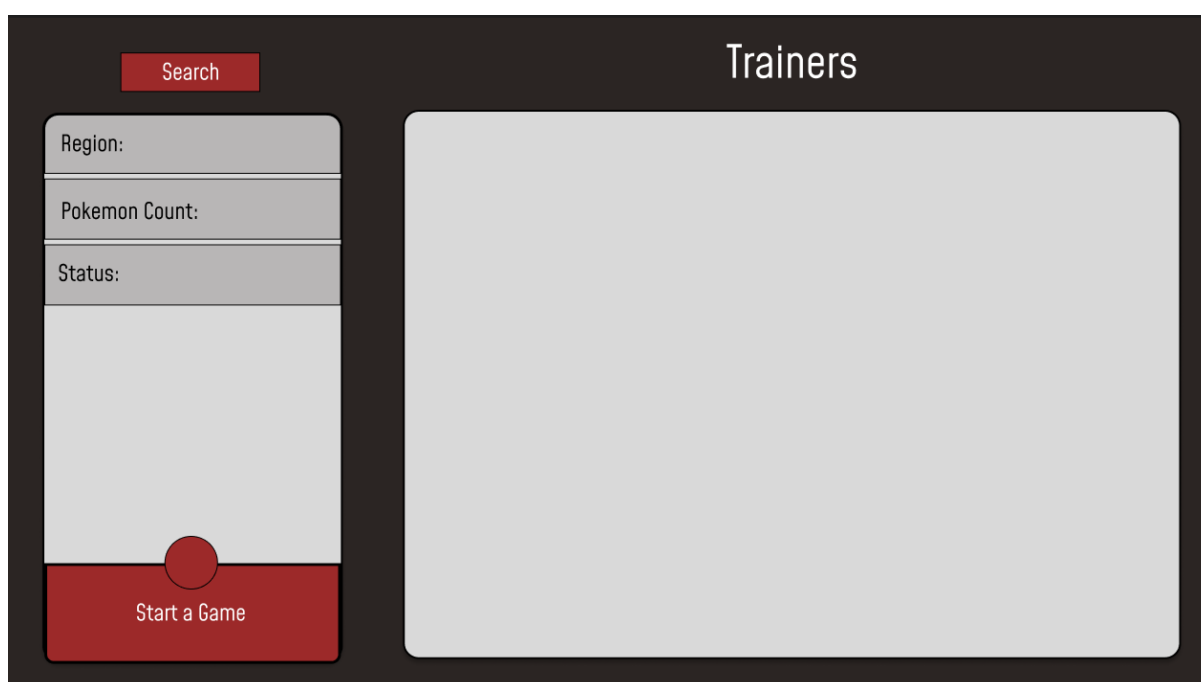
Rys. 1.5.3 Makieta głównego menu do profesora



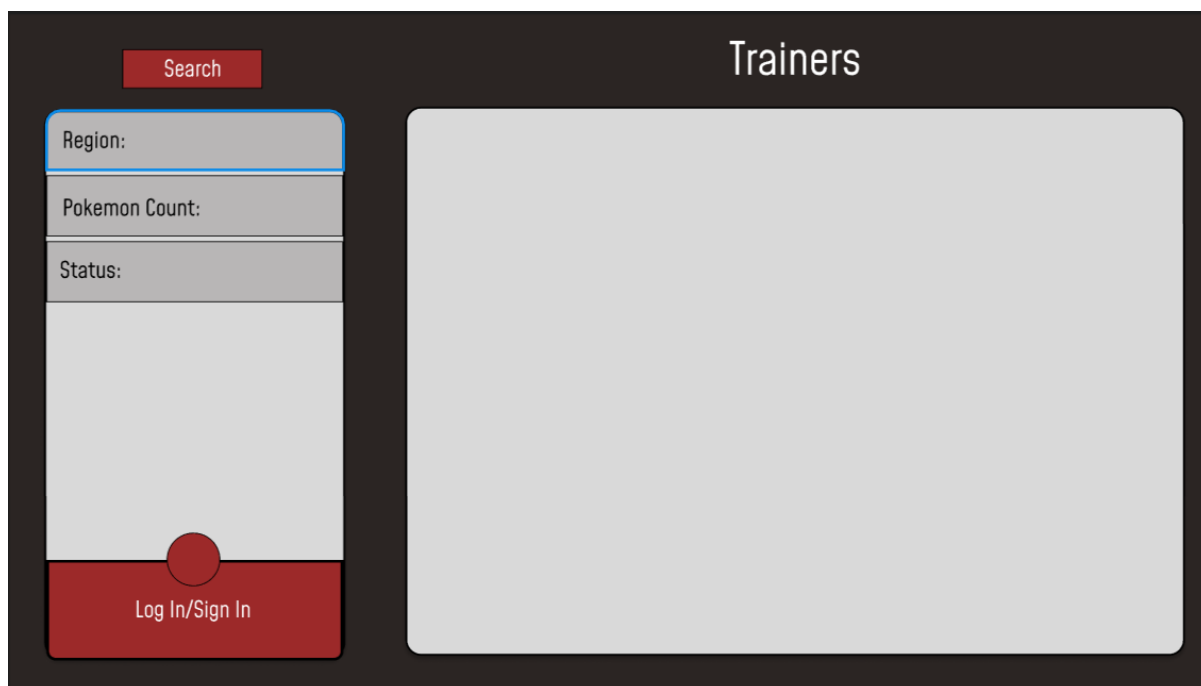
Rys 1.6.1 Makiet interfejsu wyszukiwania Pokémonów dla użytkownika zalogowanego



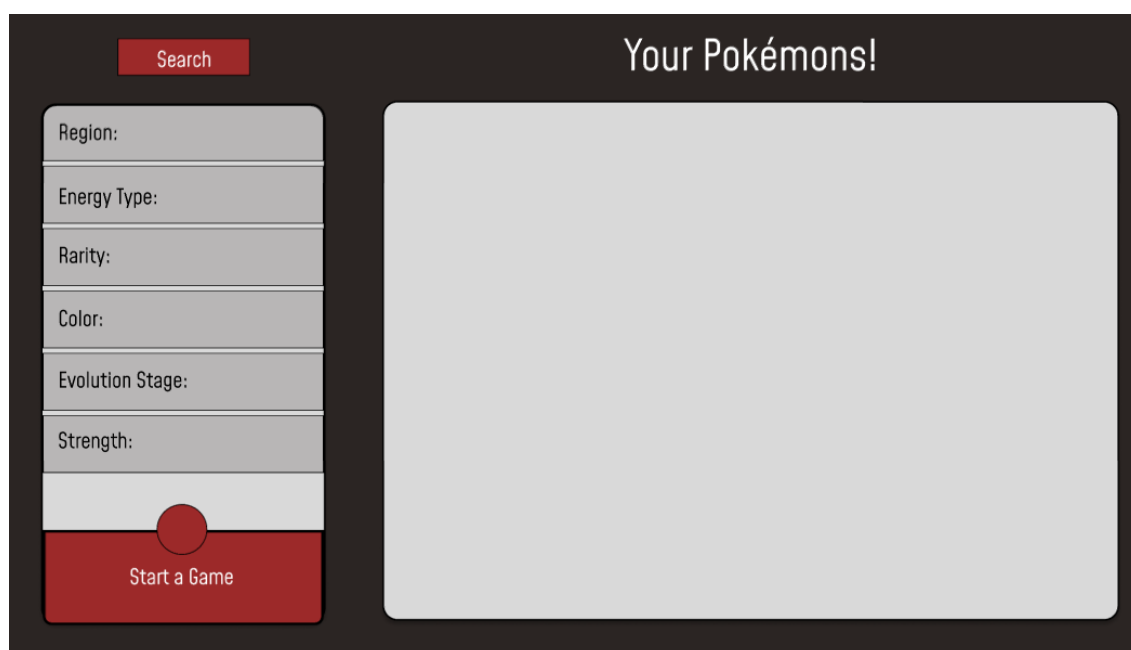
Rys 1.6.2 Makieta interfejsu wyszukiwania Pokémonów dla użytkownika niezalogowanego



Rys 1.7.1 Makieta interfejsu wyszukiwania Trenerów dla użytkownika zalogowanego



Rys 1.7.2 Makieta interfejsu wyszukiwania Trenerów dla użytkownika niezalogowanego



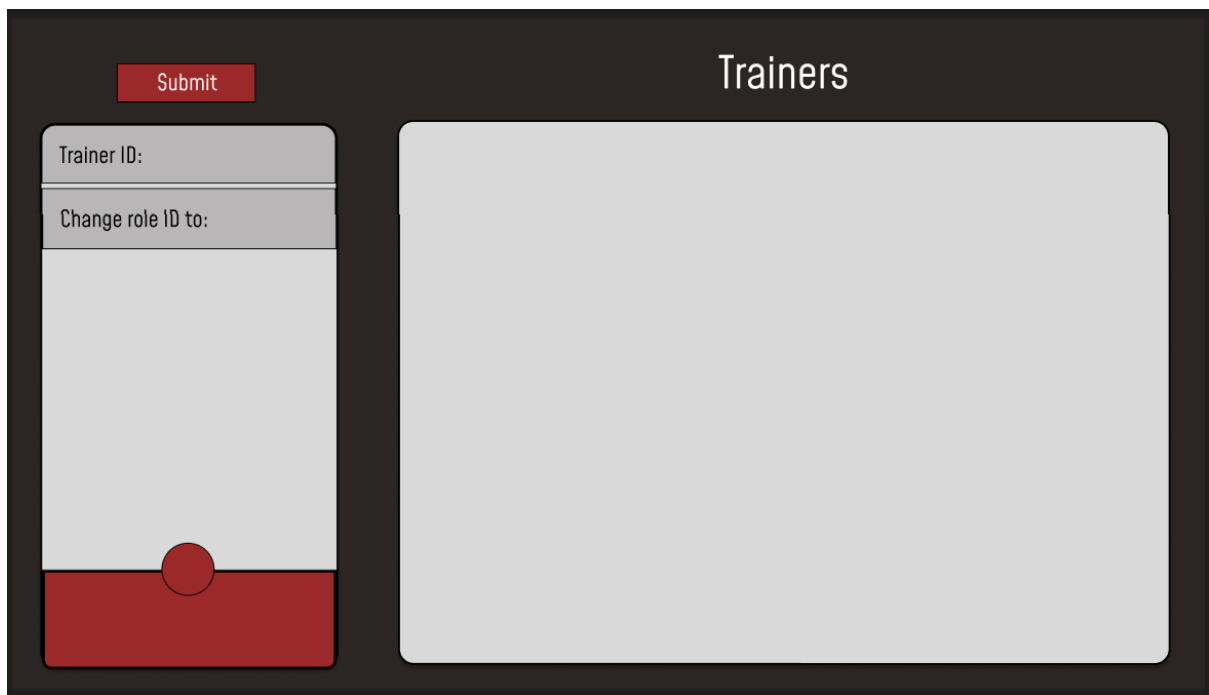
Rys. 1.8 Makieta interfejsu wyświetlania pokemonów trenerów



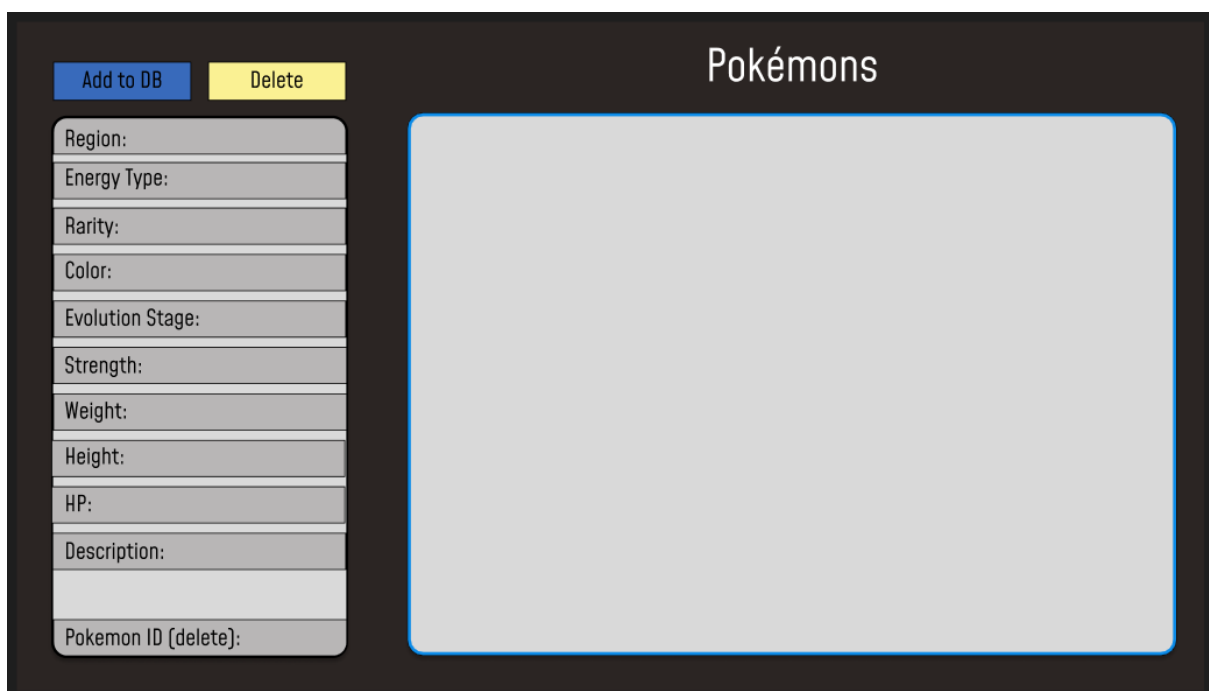
Rys. 1.9 Makiet interfejsu dodawania i usuwania Pokémonów dla trenerów



Rys. 1.10 Makiet interfejsu Gry/Walki



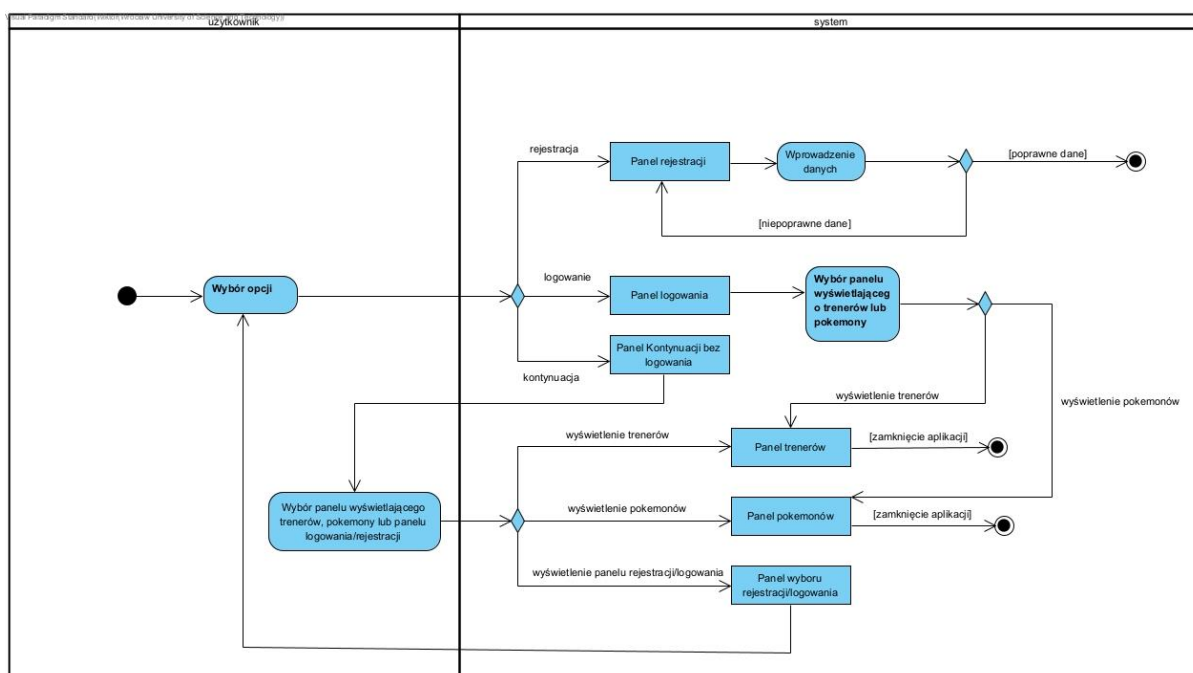
Rys 1.11 Makiet interfejsu zmiany roli trenera



Rys 1.12 Makieta Interfejsu Dodawania/Usunięcia Pokémona z bazy.

11. Diagram czynności

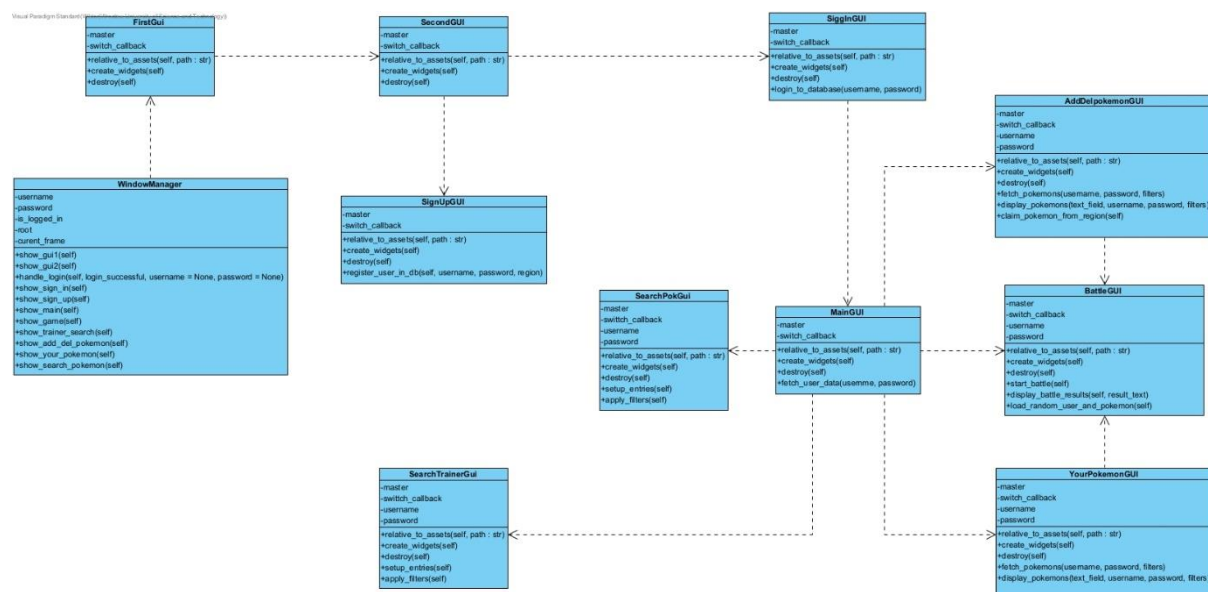
W ramach projektu opracowano diagram czynności, który szczegółowo odwzorowuje działanie i wyświetlanie, zmianę interfejsów i okienek interfejsów.



Wyżek jest przedstawiony diagram czynności do rejestracji, logowania, wyświetlania pokemonów i trenerów.

12. Diagram klas

W ramach projektu opracowano diagram klas, który szczegółowo odwzorowuje strukturę systemu. Diagram przedstawia najważniejsze klasy wraz z ich atrybutami, metodami oraz relacjami, takimi jak dziedziczenie.

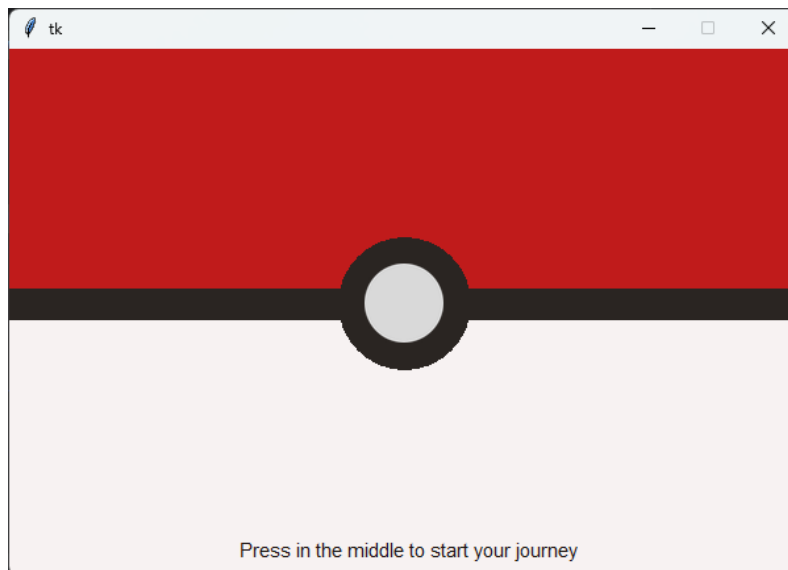


13. Aplikacja desktopowa

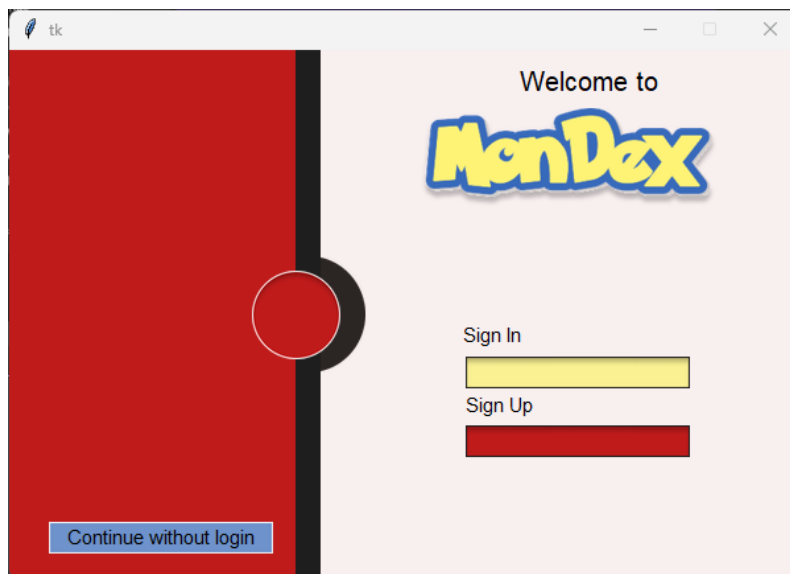
Do stworzenia aplikacji desktopowej wykorzystano język Python oraz serwer bazodanowy MySQL. Interfejs użytkownika został wykonany przy użyciu biblioteki Tkinter, która pozwoliła na stworzenie przejrzystego i funkcjonalnego interfejsu. Program składa się z kilkunastu plików .py odpowiadających za różne okna aplikacji oraz ich funkcjonalności, przy czym głównym plikiem zarządzającym przełączaniem między oknami jest WindowManager.py. Aplikacja wykorzystuje spersonalizowane assety graficzne przechowywane w dedykowanym folderze, co pozwala na utrzymanie spójnej estetyki interfejsu. Połączenie z bazą danych realizowane jest poprzez moduł mysql.connector, który zapewnia bezpieczną komunikację z serwerem MySQL i umożliwia wykonywanie operacji na danych związanych z Pokemonami i użytkownikami.

13.1 Interfejsy

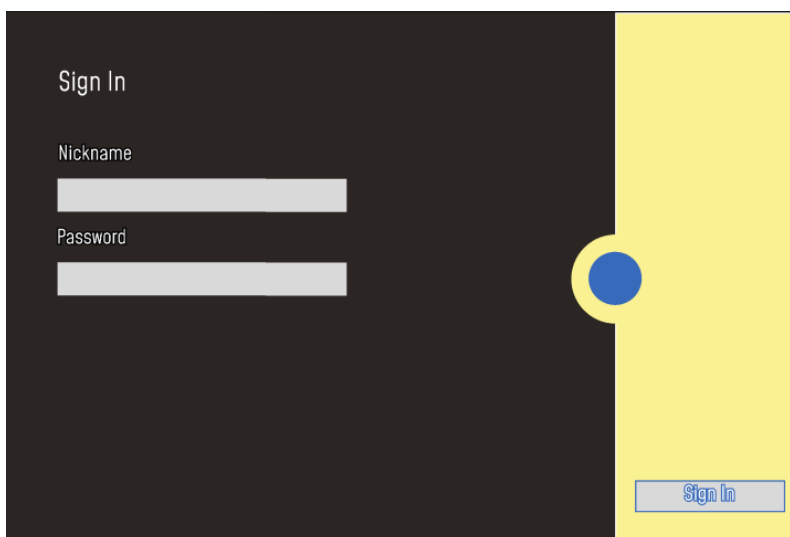
Startowe menu:



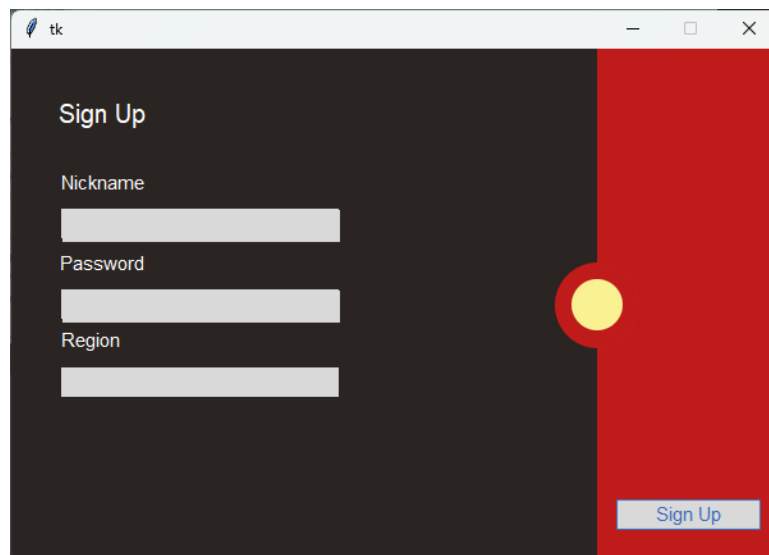
Wybór logowania:



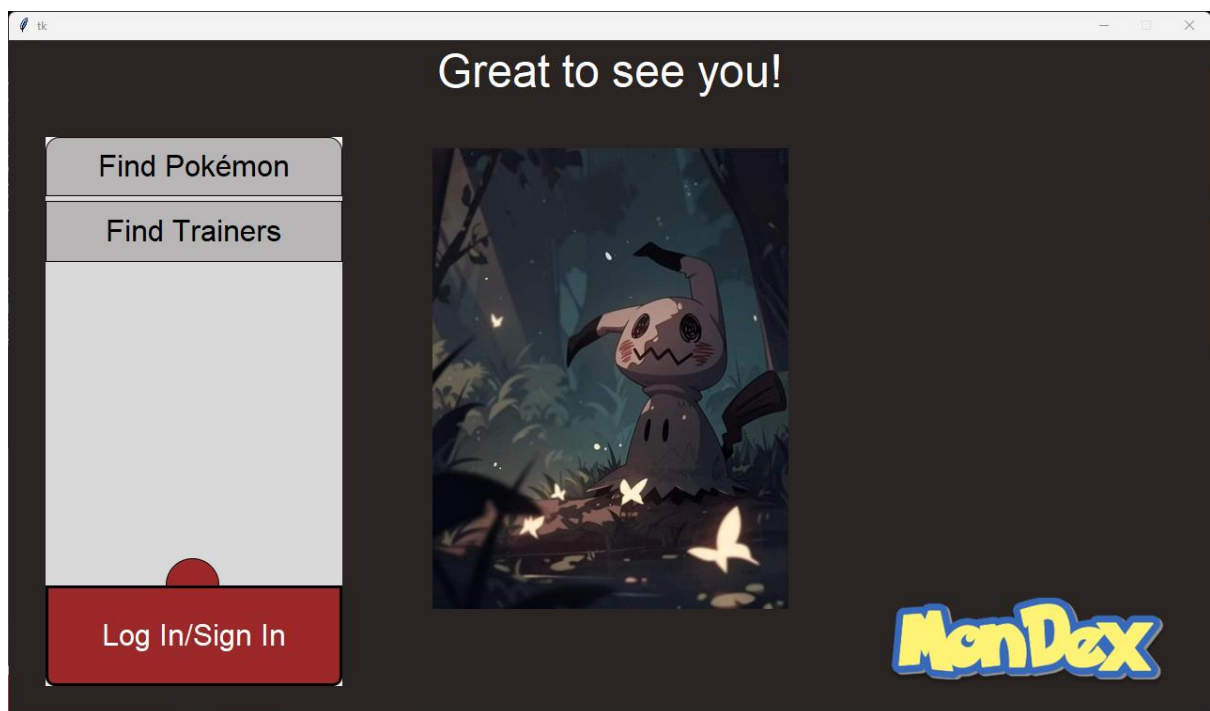
Logowanie:



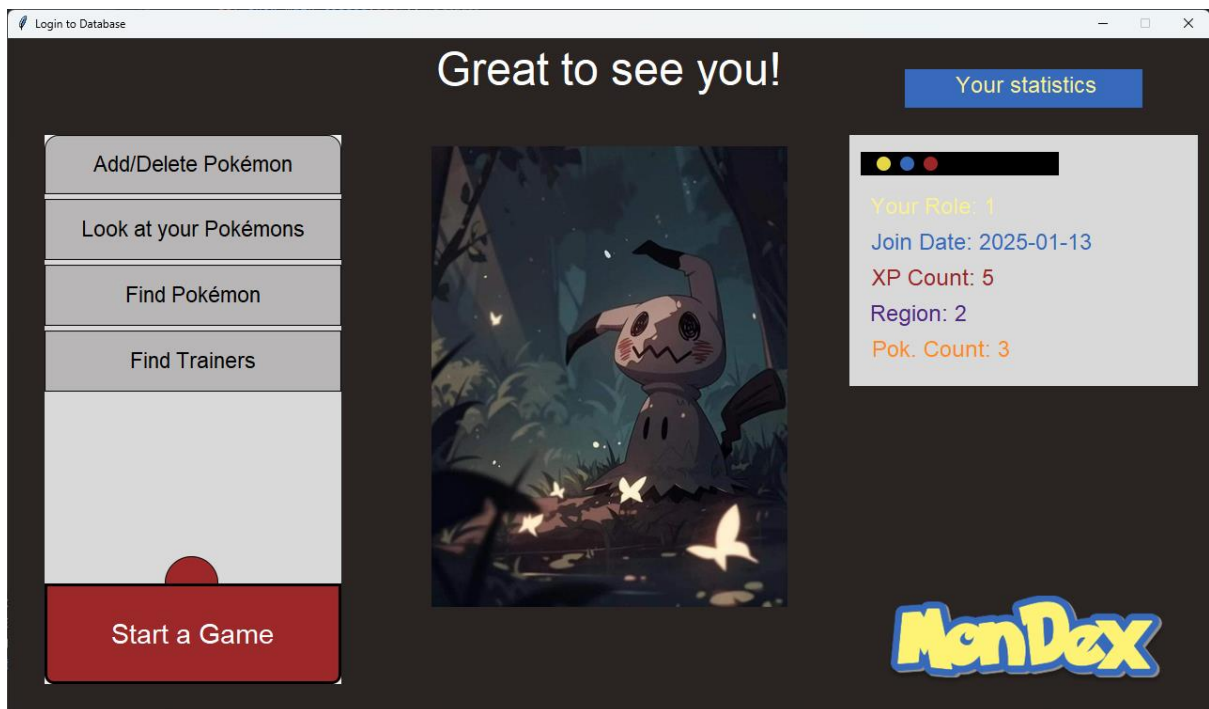
Rejestracja:



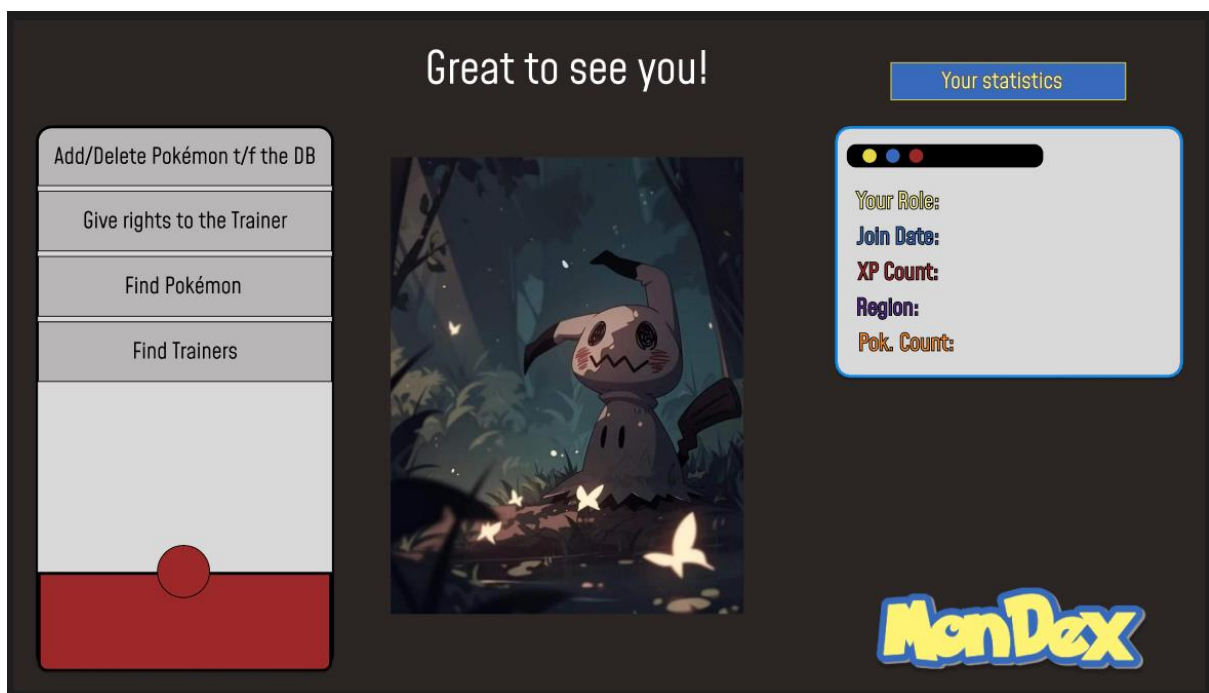
Główne Menu dla użytkownika niezarejestrowanego:



Główne Menu dla użytkownika zarejestrowanego:



Główne Menu dla profesora:



Wyszukiwanie Pokemonów:

Menu wygląda tak samo, ale różnica występuje w następnym: użytkownik niezalogowany nie ma przycisku gry, a może przejść z powrotem do menu logowania, a użytkownik zalogowany – do gry.

Po zmianie dane filtrowane są za pomocą nazw, a nie ID danych. Przykładem są dane filtrowane według typu energii, koloru, a na kolejnym zdjęciu nazwy regionu.

Search

Region:

Energy Type: Fire

Rarity:

Color: Pink

Evolution Stage:

Strength:

Log In/Sign In

Pokémons

Return

	Color_Name	XP_Count	Energy_Name	Rarity_Name
e on its tail, symbolizing its health and vitality.	Pink	None	Fire	Common
and ability to fly; breathes intense flames.	Pink	None	Fire	Rare
on its tail; agile and skilled in climbing.	Pink	None	Fire	Common
both fire and fighting techniques.	Pink	None	Fire	Uncommon
ombat style, combining fire with martial arts.	Pink	None	Fire	Rare

Search

Region: Kanto

Energy Type:

Rarity:

Color:

Evolution Stage: 1

Strength:

Log In/Sign In

Pokémons

Return

Pokemon_Name	Pokemon_ID	Evolution_Stage	Region_Name	Weight	Height	HP	Strength	User_ID
Bulbasaur	1	1	Kanto	7	1	45	3	34
Charmander	4	1	Kanto	9	1	39	3	None
Squirtle	7	1	Kanto	9	1	44	3	34
Caterpie	10	1	Kanto	3	0	45	2	34
Vulpix	37	1	Kanto	10	1	38	3	33
Jigglypuff	39	1	Kanto	6	1	60	2	None
Psyduck	54	1	Kanto	20	1	50	3	None
Mankey	56	1	Kanto	28	1	40	3	32
Poliwag	60	1	Kanto	12	1	40	3	None
Abra	63	1	Kanto	20	1	25	2	33
Machop	66	1	Kanto	20	1	70	3	None
Bellsprout	69	1	Kanto	4	1	50	2	None
Ponyta	77	1	Kanto	30	1	50	3	None
Slowpoke	79	1	Kanto	36	1	90	3	None
Shellder	90	1	Kanto	4	0	30	3	None
Drowzee	96	1	Kanto	32	1	60	3	None
Exeggcute	102	1	Kanto	3	0	50	3	None
Hitmonlee	106	1	Kanto	50	2	50	4	None
Jynx	124	1	Kanto	41	1	65	4	None
Kabuto	140	1	Kanto	12	1	30	3	None
Dratini	147	1	Kanto	3	2	41	4	None
Mew	151	1	Kanto	4	0	100	5	None

Wyszukiwanie Trenerów:

Ta sama różnica w działaniu, co i dla menu wyszukiwania pokémonów.

Search

Region:

Pokemon Count:

Status:

Log In/Sign In

Trainers

User_ID	Username	Role_ID	Join_Date	Region_ID	Pokemon_Count	User_XP_Count	Status_I
1	Ash Ketchum	1	1997-04-01	1	2	501	1
2	Professor Oak	4	1996-04-01	1	None	None	4
3	Blue	3	1998-05-01	1	0	2500	3
4	Professor Elm	4	1999-03-01	2	None	None	4
5	Red	2	1996-04-01	1	30	1200	2
6	May	1	2002-01-01	3	12	700	1
7	Professor Birch	4	2003-02-01	3	None	None	4
8	Steven Stone	3	2002-12-01	3	20	2500	3
9	Dawn	1	2006-01-01	4	15	600	1
10	Professor Rowan	4	2007-02-01	4	None	None	4
11	Cynthia	3	2007-06-01	4	25	3000	3
12	Cheren	2	2010-03-01	5	18	1300	2
13	Professor Junie	4	2011-01-01	5	None	None	4
14	Iris	3	2011-10-01	5	23	2700	3
15	Leon	3	2019-11-01	6	30	3500	3
16	Professor Magna	4	2019-11-01	6	None	None	4
17	Wiktor	None	None	1	None	None	None
19	Wiktor1	None	None	1	None	None	None
21	Maciek	1	2025-01-10	1	None	None	1
22	Szarek	1	2025-01-10	1	None	None	1
23	Szakal	1	2025-01-11	3	None	None	1
24	Kasia	1	2025-01-12	4	None	None	1
26	Maciek2	1	2025-01-12	3	None	None	1
27	Szymon	1	2025-01-12	2	None	None	1
28	Szymon1	1	2025-01-12	5	None	None	1
29	Marko	1	2025-01-13	4	None	None	1
31	PokemonOWNER	1	2025-01-13	3	1	13	1
32	Grisha	1	2025-01-13	2	4	5	1
33	Baton	1	2025-01-15	4	3	22	1
34	CoffeeDrinker	1	2025-01-15	2	3	36	1

Wyszukiwanie własnych pokemonów:

Login to Database

Search

Region:

Energy Type:

Rarity:

Color:

Evolution Stage:

Strength:

Start a Game

Your Pokémons

Pokemon_Name	Pokemon_ID	Evolution_Stage	Region_ID	Weight	Height	HP	Strength	User_ID	I
Golduck	55	2	1	77	2	80	4	32	E
Mankey	56	1	1	28	1	40	3	32	K
Cyndaquil	155	1	2	8	1	39	3	32	A
Mimikyu	778	1	7	1	0	55	3	32	A

Dodawanie/Usuwanie Pokemonów:

Login to Database

Claim!

Delete

Pokemon ID:

Start a Game

Pokémons

Pokemon_Name	Pokemon_ID	Evolution_Stage	Region_ID	Weight	Height	HP	Strength	User_ID
Bulbasaur	1	1	1	7	1	45	3	34
Ivysaur	2	2	1	13	1	60	4	None
Venusaur	3	3	1	100	2	80	5	None
Charmander	4	1	1	9	1	39	3	None
Charmeleon	5	2	1	19	1	58	4	None
Charizard	6	3	1	91	2	78	5	None
Squirtle	7	1	1	9	1	44	3	34
Wartortle	8	2	1	23	1	59	4	31
Blastoise	9	3	1	86	2	79	5	None
Caterpie	10	1	1	3	0	45	2	None
Metapod	11	2	1	10	1	50	2	None
Butterfree	12	3	1	32	1	60	3	None
Pikachu	25	2	1	6	0	35	3	1
Raichu	26	3	1	30	1	60	4	1
Clefairy	35	2	1	8	1	70	3	None
Clefable	36	3	1	40	1	95	4	None
Vulpix	37	1	1	10	1	38	3	33
Ninetales	38	2	1	20	1	73	4	None
Jigglypuff	39	1	1	6	1	60	2	None
Wigglytuff	40	2	1	12	1	140	3	None
Fayduck	54	1	1	20	1	50	3	None
Golduck	55	2	1	77	2	80	4	32
Mankey	56	1	1	28	1	40	3	32
Primeape	57	2	1	32	1	65	4	None
Poliwag	60	1	1	12	1	40	3	None
Poliwhirl	61	2	1	20	1	65	4	None
Poliwrath	62	3	1	54	1	90	5	34
Abra	63	1	1	20	1	25	2	33
Kadabra	64	2	1	57	1	40	4	None
Alakazam	65	3	1	48	2	55	5	None
Machop	66	1	1	20	1	70	3	None
Machoke	67	2	1	71	2	80	4	None
Machop	68	3	1	130	2	90	5	33
Bellsprout	69	1	1	4	1	50	2	None

Dodawanie/Usuwanie Pokemonów z bazy:

Add to DB

Delete

Region:

Energy Type:

Rarity:

Color:

Evolution Stage:

Strength:

Weight:

Height:

HP:

Description:

Pokemon ID (delete):

Pokémons

Menu zmiany roli profesorów:

Submit

Trainer ID:

Change role ID to:

Trainers

Menu Gry:

Login to Database

Choose your Pokémon!

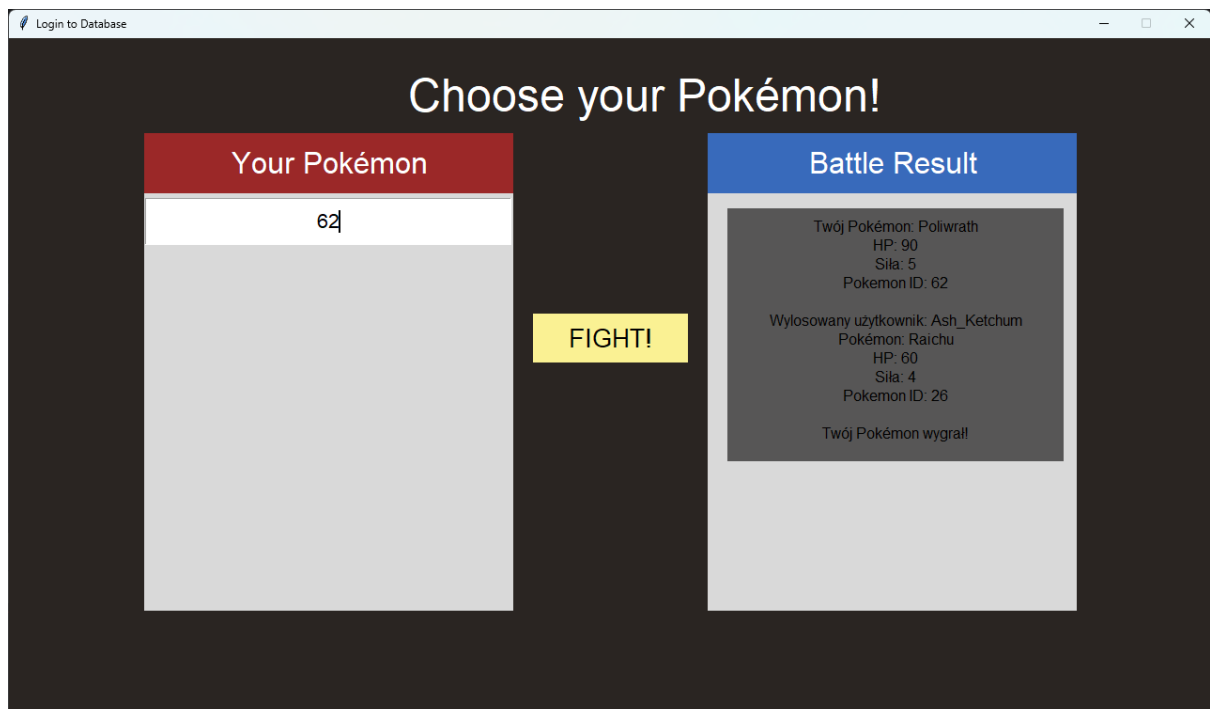
Your Pokémon

Enter Pokémon ID

FIGHT!

Battle Result

Przykład walki:

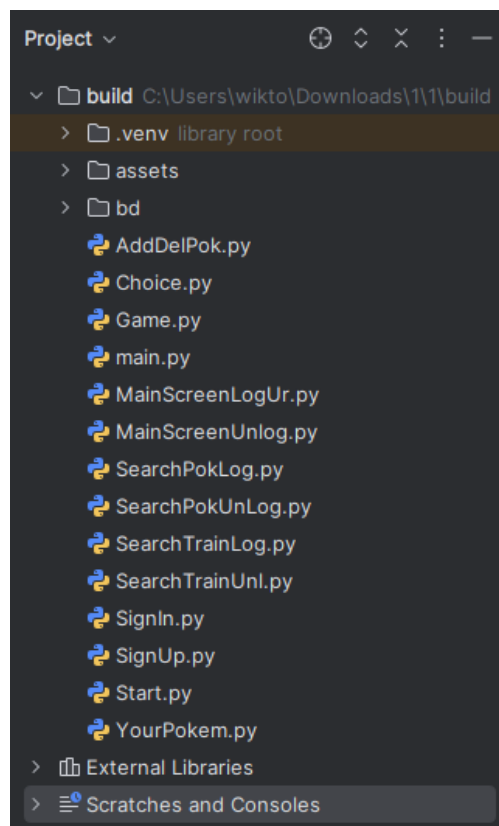


13.2 Opisanie kodu:

Oto wszystkie importy do klasy głównej. Możemy zobaczyć tutaj wszystkie istniejące GUI:

```
from tkinter import Tk, messagebox
from Start import FirstGUI
from Choice import SecondGUI
from SignIn import SignInGUI
from SignUp import SignUpGUI
from MainScreenUnlog import MainGUI
from SearchPokUnLog import MainGUI as PokemonSearchGUI
from SearchTrainUnl import TrainerGUI
from MainScreenLogUr import MainGUI as MainScreenLogUr
from AddDelPok import PokemonGUI
from YourPokem import MainGUI as YourPokemGUI
from SearchPokLog import MainGUI as SearchPokLogGUI
from SearchTrainLog import TrainerGUI as SearchTrainLogGUI
from Game import BattleGUI
```

Struktura projektu jest przedstawiona niżej, gdzie w folderze „assets” są foto do każdego używane GUI:



Opis funkcji dla klasy main:

init:

```
def __init__(self): new *
    self.username = None
    self.password = None
    self.is_logged_in = False
    self.root = Tk()
    self.current_frame = None
```

Inicjalizuje aplikację tworząc główne okno Tkinter. Przechowuje informacje o użytkowniku takie jak nazwa użytkownika, hasło oraz status zalogowania. Ustawia również początkowe puste okno aplikacji.

show_gui:

```
def show_gui1(self): 2 usages new *
    if self.current_frame:
        self.current_frame.destroy()
    self.current_frame = FirstGUI(self.root, self.show_gui2)
```

Wyświetla ekran startowy aplikacji. Jest to pierwszy ekran, który widzi użytkownik po uruchomieniu programu. Zawiera tylko jeden przycisk prowadzący do ekranu wyboru.

show_gui2:

```
def show_gui2(self): 6 usages new *
    if self.current_frame:
        self.current_frame.destroy()
    self.current_frame = SecondGUI(self.root, self.show_gui1)
    # Configure all buttons for the Choice menu
    self.current_frame.button_1.configure(command=self.show_main_unlogged) # Continue without login
    self.current_frame.button_2.configure(command=self.show_sign_in) # Sign In
    self.current_frame.button_3.configure(command=self.show_sign_up) # Sign Up
```

Ekran wyboru zawierający trzy przyciski: kontynuacja bez logowania, logowanie oraz rejestracja. Każdy z przycisków prowadzi do odpowiedniego ekranu: menu głównego dla niezalogowanych, ekranu logowania lub ekranu rejestracji.

handle_login:

```
def handle_login(self, login_successful, username = None, password = None): 2 usages new *
    if login_successful:
        self.is_logged_in = True
        self.username = username # Przechowujemy dane użytkownika w WindowManager
        self.password = password
        print("Logowanie powiodło się!")
        self.show_main_logged()
    else:
        print("Logowanie nie powiodło się.")
        self.show_sign_in()
```

Obsługuje proces logowania użytkownika. Jeśli logowanie jest udane, zapisuje dane użytkownika i przekierowuje do menu głównego dla zalogowanych. W przypadku niepowodzenia, wraca do ekranu logowania.

show_sign_in:

```
def show_sign_in(self): 3 usages new *
    if self.current_frame:
        self.current_frame.destroy()
    self.current_frame = SignInGUI(self.root, self.handle_login)
```

Wyświetla ekran logowania, gdzie użytkownik może wprowadzić swoją nazwę użytkownika i hasło. Po udanym logowaniu przekierowuje do menu głównego dla zalogowanych.

show_sign_up:

```
def show_sign_up(self): 1 usage new *
    if self.current_frame:
        self.current_frame.destroy()
    self.current_frame = SignUpGUI(self.root, self.handle_login)
```

Wyświetla ekran rejestracji nowego użytkownika. Po udanej rejestracji automatycznie loguje użytkownika i przekierowuje do menu głównego dla zalogowanych.

show_main_unlogged:

```
def show_main_unlogged(self): 3 usages new *
    if self.current_frame:
        self.current_frame.destroy()
    self.current_frame = MainGUI(self.root, self.show_gui2)
    # Configure all buttons in mainScreenUnlog
    self.current_frame.button_1.configure(command=self.show_gui2) # Login button
    self.current_frame.button_2.configure(command=self.show_pokemon_search) # Find Pokémon
    self.current_frame.button_3.configure(command=self.show_trainer_search) # Find Trainers
```

Menu główne dla niezalogowanych użytkowników. Zawiera opcje logowania, wyszukiwania Pokemonów i trenerów. Ma ograniczone funkcjonalności w porównaniu do wersji dla zalogowanych.

show_pokemon_search:

```
def show_pokemon_search(self): 1 usage new *
    if self.current_frame:
        self.current_frame.destroy()
    self.current_frame = PokemonSearchGUI(self.root, self.show_main_unlogged)
    # Configure buttons in Pokemon Search screen
    self.current_frame.button_1.configure(command=self.show_gui2) # Login button
```

Ekran wyszukiwania Pokemonów dla niezalogowanych użytkowników. Zawiera przycisk powrotu do menu głównego oraz opcję logowania.

show_trainer_search:

```
def show_trainer_search(self): 1 usage new *
    if self.current_frame:
        self.current_frame.destroy()
    self.current_frame = TrainerGUI(self.root, self.show_main_unlogged)
    # Configure buttons in Trainer Search screen
    self.current_frame.button_1.configure(command=self.show_gui2) # Login button
```

Ekran wyszukiwania trenerów dla niezalogowanych użytkowników. Podobnie jak wyszukiwarka Pokemonów, zawiera przycisk powrotu i opcję logowania.

show_game:

```
def show_game(self): 5 usages new *
    """Method to show the game screen"""
    if self.current_frame:
        self.current_frame.destroy()
    self.current_frame = BattleGUI(self.root, self.show_main_logged, self.username, self.password)
```

Ekran gry Pokemon, gdzie odbywa się walka. Dostępny tylko dla zalogowanych użytkowników. Zawiera mechanikę walki oraz przycisk powrotu do menu głównego.

show_main_logged:

```
def show_main_logged(self): 6 usages new *
    if not self.username or not self.password:
        messagebox.showerror(title: "Error", message: "You must be logged in to access this screen.")
        self.show_sign_in()
        return

    if self.current_frame:
        self.current_frame.destroy()

    # Przekazujemy dane użytkownika do MainScreenLogUr
    self.current_frame = MainScreenLogUr(
        self.root,
        self.show_gui2,
        username=self.username,
        password=self.password
    )

    # Konfiguracja przycisków
    self.current_frame.button_2.configure(command=self.show_add_del_pokemon)
    self.current_frame.button_3.configure(command=self.show_your_pokemon)
    self.current_frame.button_4.configure(command=self.show_search_pokemon_logged)
    self.current_frame.button_5.configure(command=self.show_search_trainer_logged)
    self.current_frame.button_1.configure(command=self.show_game) # Przycisk do gry
```

Rozszerzone menu główne dla zalogowanych użytkowników. Zawiera wszystkie funkcjonalności: rozpoczęcie gry, zarządzanie Pokemonami, przeglądanie kolekcji oraz wyszukiwanie Pokemonów i trenerów.

show_add_del_pokemon:

```
def show_add_del_pokemon(self): 1 usage new *
    if self.current_frame:
        self.current_frame.destroy()
    self.current_frame = PokemonGUI(self.root, self.show_main_logged, username=self.username, password=self.password)
    self.current_frame.button_1.configure(command=self.show_game) # Start game button
```

Ekran zarządzania kolekcją Pokemonów. Pozwala na dodawanie i usuwanie Pokemonów z kolekcji. Zawiera przycisk powrotu do menu oraz szybki dostęp do gry.

show_your_pokemon:

```
def show_your_pokemon(self): 1 usage new *
    if self.current_frame:
        self.current_frame.destroy()
    self.current_frame = YourPokemonGUI(self.root, self.show_main_logged, username=self.username, password=self.password)
    self.current_frame.button_1.configure(command=self.show_game) # Start game button
```

Wyświetla listę Pokemonów należących do zalogowanego użytkownika. Umożliwia przeglądanie własnej kolekcji. Zawiera przycisk powrotu i rozpoczęcia gry.

show_search_pokemon_logged:

```
def show_search_pokemon_logged(self): 1usage new *
    if self.current_frame:
        self.current_frame.destroy()
    self.current_frame = SearchPokLogGUI(self.root, self.show_main_logged, username=self.username,
        password=self.password)
    self.current_frame.button_1.configure(command=self.show_game) # Start game button
```

Zaawansowana wyszukiwarka Pokemonów dla zalogowanych użytkowników. Oferuje więcej opcji niż wersja dla niezalogowanych. Ma przyciski powrotu i szybkiego dostępu do gry.

show_search_trainer_logged:

```
def show_search_trainer_logged(self): 1usage new *
    if self.current_frame:
        self.current_frame.destroy()
    self.current_frame = SearchTrainLogGUI(self.root, self.show_main_logged, username=self.username,
        password=self.password)
    self.current_frame.button_1.configure(command=self.show_game) # Start game button
```

Zaawansowana wyszukiwarka trenerów dla zalogowanych użytkowników. Podobnie jak wyszukiwarka Pokemonów, ma rozszerzone funkcje i przyciski nawigacyjne.

run:

```
def run(self): 1usage new *
    self.show_gui1() # Start with the first GUI
    self.root.mainloop()
```

Uruchamia aplikację, rozpoczynając od ekranu startowego (show_gui1) i włącza główną pętlę Tkinter.

13.3 Budowa interfejsu

Jako przykład budowania interfejsu i podłączanie do bazy danych, opisano niżej kod dla głównego menu dla zalogowanego użytkownika:

Struktura i styl okna:

Okno aplikacji jest zaprojektowane z ciemnym motywem (tło #2A2522) i wymiarami 1280x720 pikseli. Układ jest podzielony na dwie główne sekcje: lewy panel dla filtrów wyszukiwania (współrzędne 39.103 do 355.683) i prawy panel do wyświetlania wyników (423.100 do 1248.680). Oba panele mają

jasnoszare tło (#D9D9D9). Na górze znajduje się wyśrodkowany tytuł „Your Pokémons” w białym tekście przy użyciu czcionki Mohave Regular. Okno zawiera przycisk „Start a Game” w lewym dolnym rogu i przycisk wyszukiwania w pobliżu góry.

Implementacja okna:

Interfejs okna jest zbudowany przy użyciu Tkinter i podąża za ustrukturyzowanym systemem układu. Główne okno jest inicjowane określonymi wymiarami (1280x720 pikseli) i używa niestandardowego ciemnobrązowego koloru tła (#2A2522). Interfejs jest zbudowany przy użyciu kilku kluczowych komponentów:

System Canvas:

Podstawowym elementem jest widżet Canvas, który rozciąga się na całe okno (1280x720). Ten canvas służy jako podstawa dla wszystkich elementów wizualnych i jest skonfigurowany z:

- Brak obramowania (bd=0)
- Brak grubości podświetlenia
- Styl reliefu grzbietu
- Niestandardowy kolor tła

Struktura układu:

Interfejs jest podzielony na trzy główne sekcje:

1. Obszar Nagłówek (Góra):

- Zawiera tytuł "Your Pokémons" (wycentrowany, biały tekst, czcionka Mohave Regular)
- Posiada przycisk wyszukiwania
- Wykorzystuje pozycjonowanie absolutne dla precyzyjnego umieszczenia elementów

2. Panel Lewy (Sekcja Filtrów):

- Wymiary: 316x580 pikseli
- Tło: Jasnoszary prostokąt (#D9D9D9)
- Zawiera sześć pól do filtrowania:
 - ✧ Każde pole ma etykietę i pole wprowadzania
 - ✧ Wykorzystuje spójne odstępy i wyrównanie
 - ✧ Niestandardowe obrazy tła dla poprawy wizualnej
- Dolna sekcja zawiera duży przycisk "Start a Game"

3. Panel Prawy (Sekcja Wyświetlania):

- Wymiary: 825x580 pikseli
- Tło: Jasnoszary prostokąt (#D9D9D9)
- Zawiera pole tekstowe do wyświetlania wyników z bazy danych

- Skonfigurowany z wyłączonym zawijaniem tekstu dla wyświetlania w formie tabeli

Stylizacja wizualna:

Interfejs wykorzystuje kilka elementów projektowych:

- Niestandardowe obrazy przycisków ładowane z zasobów
- Spójne użycie czcionki (Mohave Regular w różnych rozmiarach)
- Starannie zaplanowana kolorystyka:
 - ✧ Ciemne tło dla kontrastu
 - ✧ Jasnoszare panele dla obszarów zawartości
 - ✧ Białe tekst dla nagłówków
 - ✧ Niestandardowa stylizacja przycisków z tłami obrazów

Pozycjonowanie widgetów:

Wszystkie elementy używają pozycjonowania absolutnego z określonymi współrzędnymi:

(Przykład pozycjonowania)

```
button.place(x=39.0, y=580.0, width=316.0, height=107.0)
```

```
text_field.place(x=423.0, y=100.0, width=825.0, height=580.0)
```

Pola wprowadzania:

Pola wprowadzania są zaimplementowane z:

- Niestandardowymi obrazami tła
- Standardowymi wymiarami
- Dynamicznym pozycjonowaniem etykiet
- Spójnymi odstępami między elementami
- Automatycznym centrowaniem pionowym

Wyświetlanie tekstu:

Obszar wyświetlania wyników zawiera:

- Przewijalny widget tekstowy
- Czcionkę o stałej szerokości dla wyrównania tabeli
- Stan wyłączony, aby zapobiec edycji
- Niestandardowy kolor tła
- Brak zawijania tekstu dla poprawnego wyświetlania tabeli

Zarządzanie zasobami:

Interfejs wykorzystuje ustrukturyzowany system zasobów:

- Zasoby są przechowywane w dedykowanym folderze
- Ścieżki są zarządzane przez funkcję pomocniczą
- Obrazy są ładowane jako obiekty PhotoImage
- Zasoby zawierają tła przycisków i pól wprowadzania

Ta implementacja interfejsu została zaprojektowana z myślą o zachowaniu profesjonalnego wyglądu przy jednoczesnym zapewnieniu intuicyjnej obsługi dla użytkownika. Wykorzystanie systemu pozycjonowania absolutnego umożliwia dokładne kontrolowanie położenia każdego elementu, co w połączeniu ze spójną stylistyką kolorów i czcionek tworzy harmonijny i estetyczny wygląd całej aplikacji. Taki projekt nie tylko zapewnia przyjemne wrażenia wizualne, ale również ułatwia nawigację i interakcję z różnymi funkcjami programu.

Opis funkcji dla interfejsu:

`__init__`:

```
def __init__(self, master=None, switch_callback=None, username=None, password=None): new *
    super().__init__(master)
    self.master = master
    self.switch_callback = switch_callback
    self.username = username
    self.password = password

    # Setup the initial window configuration
    self.master.geometry("1280x720")
    self.master.configure(bg="#2A2522")

    # Setup assets path
    self.output_path = Path(__file__).parent
    self.assets_path = self.output_path / Path(r"assets\frame9")

    self.create_widgets()
```

Inicjalizuje okno aplikacji, ustawia jego rozmiar, tło oraz ścieżkę do zasobów (assets). Przyjmuje parametry master (główne okno), callback do przełączania widoków oraz dane logowania użytkownika.

`relative_to_assets`:

```
def relative_to_assets(self, path: str) -> Path: 4 usages new *
    return self.assets_path / Path(path)
```

Tworzy ścieżkę względną do plików zasobów (obrazów, przycisków) używanych w interfejsie.

apply_filters:

```
def apply_filters(self): 1usage new *
    filters = {}
    for entry_name, entry_widget in self.entries.items():
        value = entry_widget.get().strip() # Pobierz wartość z entry i usuń nadmiarowe spacje
        if value: # Uwzględnij tylko wypełnione pola
            filters[entry_name] = value

    # Wywołaj funkcję wyświetlania danych z uwzględnieniem filtrów
    display_pokemons(
        text_field=self.text_field,
        username=self.username,
        password=self.password,
        filters=filters
    )
```

Zbiera wartości z pól wprowadzania (entries) i używa ich jako filtrów w zapytaniu do bazy danych. Aktualizuje wyświetlane wyniki.

create_widgets:

```
def create_widgets(self): 1usage new *
    # Create and configure the canvas
    self.canvas = Canvas(
        self.master,
        bg="#2A2522",
        height=720,
        width=1280,
        bd=0,
        highlightthickness=0,
        relief="ridge"
    )
    self.canvas.place(x=0, y=0)

    # Create title text (centered)
    self.canvas.create_text(
        1280 / 2, # Centered horizontally
        40.0, # Positioned near top
        anchor="center",
        text="Your Pokémons",
        fill="FFFFFF",
        font=("Mohave Regular", 48 * -1)
    )

    # Create rectangles
    self.create_rectangles()

    # Setup buttons
    self.setup_buttons()
```

```
# Setup entries
self.setup_entries()

# Create text elements
self.create_text_elements()

# Setup images
self.setup_images()

self.master.resizable(False, False)
```

Tworzy wszystkie elementy interfejsu: canvas, tytuł, prostokąty tła, przyciski, pola wprowadzania i pole tekstowe wyników.

create_rectangles:

```
def create_rectangles(self): 1usage new *
    # Left rectangle (search filters)
    self.canvas.create_rectangle(
        39.0,
        103.0,
        355.0,
        683.0,
        fill="#D9D9D9",
        outline=""
    )

    # Right rectangle (display area)
    self.canvas.create_rectangle(
        423.0,
        100.0,
        1248.0,
        680.0,
        fill="#D9D9D9",
        outline=""
    )
```

Tworzy dwa główne prostokąty interfejsu - lewy panel filtrów i prawy panel wyników.

setup_buttons:

```

def setup_buttons(self): 1 usage new *
    # Search button
    self.button_image_2 = PhotoImage(file=self.relative_to_assets("button_2.png"))
    self.button_2 = Button(
        self.master,
        image=self.button_image_2,
        borderwidth=0,
        highlightthickness=0,
        command=self.apply_filters,
        relief="flat",
        text="Search",
        font=("Mohave Regular", 20),
        fg="FFFFFF",
        compound="center"
    )
    self.button_2.place(x=123.0, y=28.0, width=149.0, height=42.0)

```

```

# Login button
self.button_image_1 = PhotoImage(file=self.relative_to_assets("button_1.png"))
self.button_1 = Button(
    self.master,
    image=self.button_image_1,
    borderwidth=0,
    highlightthickness=0,
    command=lambda: print("button_1 clicked"),
    relief="flat",
    text="Start a Game",
    font=("Mohave Regular", 22),
    fg="FFFFFF",
    compound="center"
)
self.button_1.place(x=39.0, y=580.0, width=316.0, height=107.0)

```

Konfiguruje przyciski "Search" i "Start a Game", ustawiając ich wygląd i funkcje.

setup_entries:

```

def setup_entries(self): 1usage new*
    # Entry fields configuration with dynamic positioning
    entry_configs = [
        ("Region_ID", "Region:", 103.0),
        ("Energy_ID", "Energy Type:", 171.0),
        ("Rarity_ID", "Rarity:", 241.0),
        ("Color_ID", "Color:", 311.0),
        ("Evolution_Stage", "Evolution Stage:", 381.0),
        ("Strength", "Strength:", 451.0)
    ]

    self.entries = {}
    self.entry_images = {}
    self.entry_bgs = {}

    for entry_name, label, y_pos in entry_configs:
        # Create and store the entry image
        image = PhotoImage(file=self.relative_to_assets(f"{entry_name}.png"))
        self.entry_images[entry_name] = image

        # Create the entry background
        bg = self.canvas.create_image(
            *args: 197.0,
            y_pos + 31.5,
            image=image
        )
        self.entry_bgs[entry_name] = bg

```

```

    # Calculate center position for the row (background height is 63.0)
    center_y = y_pos + (63.0 / 2)

    # Create the label text (as a canvas text item)
    label_id = self.canvas.create_text(
        56.0,
        center_y, # Centered vertically
        anchor="w", # Left alignment
        text=label,
        fill="#000000",
        font=("Mohave Regular", 24)
    )

    # Get the width of the label text
    bbox = self.canvas.bbox(label_id)
    label_width = bbox[2] - bbox[0]

    # Create the entry widget
    entry = Entry(
        self.master,
        bd=0,
        bg="#B7B5B5",
        fg="#000716",
        highlightthickness=0,
        font=("Mohave Regular", 24)
    )

```

```

# Position entry right after the label text
entry.place(
    x=56.0 + label_width + 5, # 5 pixels of padding after label
    y=center_y - 20, # Offset to center the entry vertically
    width=250 - label_width, # Adjust width based on label size
    height=40.0
)
self.entries[entry_name] = entry

```

Tworzy i konfiguruje pola wprowadzania dla filtrów (Region, Energy Type, Rarity, Color, Evolution Stage, Strength).

setup_images:

```

def setup_images(self): 1 usage new *
    # Login button icon
    self.image_1_photo = PhotoImage(file=self.relative_to_assets("image_1.png"))
    self.image_1 = self.canvas.create_image(
        *args: 195.0,
        579.0,
        image=self.image_1_photo
    )

```

Ładuje i wyświetla obrazy używane w interfejsie.

create_text_elements:


```

def create_text_elements(self): 1usage new*
    # Tworzenie pola tekstowego
    self.text_field = Text(
        bd=0,
        bg="#FFFFFF",
        fg="#000000",
        highlightthickness=0,
        wrap="none", # Wyłączenie zawijania tekstu
        state="normal" # Ustawione na "normal" na początku, aby można było dodać tekst
    )
    self.text_field.place(
        x=423.0,
        y=100.0,
        width=825.0,
        height=580.0
    )

    # Wyświetlanie wyników z bazy danych w polu tekstowym
    display_pokemons(text_field=self.text_field, username=self.username, password=self.password)

    # Ustawienie pola tekstowego na "disabled" po wstawieniu tekstu
    self.text_field.config(state="disabled")

```

Tworzy i konfiguruje pole tekstowe do wyświetlania wyników z bazy danych.

destroy:

```

def destroy(self): new*
    """Properly destroy all widgets"""
    self.canvas.destroy()
    for entry in self.entries.values():
        entry.destroy()
    self.button_1.destroy()
    self.button_2.destroy()
    super().destroy()

```

Poprawnie usuwa wszystkie widgety i zwalnia zasoby przy zamykaniu okna.

Dodatkowo, poza klasą znajdują się dwie ważne funkcje:

fetch_pokemons:

```
# Funkcja do pobierania danych z bazy
def fetch_pokemons(username, password, filters=None): 1 usage new *
    try:
        # Połączenie z bazą danych
        mydb = mysql.connector.connect(
            host="localhost",
            user=username,
            password=password,
            database="bazy_danych-projekt-mysql"
        )
        mycursor = mydb.cursor()

        # Budowanie zapytania SQL
        query = "SELECT * FROM Pokemon AS p JOIN Users AS u ON p.User_ID = u.User_ID WHERE u.Username = %s"
        values = [username] # Dodaj nazwę użytkownika jako pierwszy parametr

        if filters:
            conditions = []
            for column, value in filters.items():
                conditions.append(f"{column} = %s")
                values.append(value)
            if conditions:
                query += " AND " + " AND ".join(conditions) # Dodaj 'AND' przed dodatkowymi warunkami

        # Wykonanie zapytania
        mycursor.execute(query, values)

        # Pobranie wyników i nazw kolumn
        rows = mycursor.fetchall()
        column_names = [desc[0] for desc in mycursor.description]
```

```
# Formatowanie wyników
column_widths = [
    max(len(str(row[i])) for row in rows) if rows else 0
    for i in range(len(column_names))
]
column_widths = [
    max(len(column_names[i]), column_widths[i])
    for i in range(len(column_names))
]

formatted_text = " | ".join(f"{column_names[i]:<{column_widths[i]}}" for i in range(len(column_names))) + "\n"
formatted_text += "-" * (sum(column_widths) + 3 * (len(column_widths) - 1)) + "\n"

for row in rows:
    formatted_text += " | ".join(f"{str(row[i]):<{column_widths[i]}}" for i in range(len(row))) + "\n"

return formatted_text
except mysql.connector.Error as err:
    return f"Błąd bazy danych: {err}"
finally:
    if 'mydb' in locals() and mydb.is_connected():
        mydb.close()
```

Łączy się z bazą danych, wykonuje zapytanie SQL z uwzględnieniem filtrów i formatuje wyniki.

display_pokemons:

```
# Funkcja do wyświetlania wyników w aplikacji
def display_pokemons(text_field=None, username=None, password=None, filters=None): 2 usages new *
    results = fetch_pokemons(username, password, filters) # Przekaż filtry do funkcji pobierającej dane
    text_field.config(state="normal") # Odblokuj pole tekstowe, aby wstawić tekst
    text_field.delete("1.0", "end") # Wyczyszczenie pola tekstowego
    text_field.insert("1.0", results) # Wstawienie wyników
    text_field.config(state="disabled") # Zablokuj pole tekstowe po wstawieniu
```

Wyświetla sformatowane wyniki w polu tekstowym interfejsu.

13.4 Połączenie z bazą danych

Aplikacja łączy się z bazą danych MySQL o nazwie „bazy_danych_projekt_mysql” na localhost. Używa danych uwierzytelniających użytkownika (nazwa użytkownika i hasło) do uwierzytelniania. Główna funkcjonalność koncentruje się na pobieraniu danych Pokémon za pomocą zapytań SQL, które łączą tabele Pokemon i Users. Wyniki są formatowane w uporządkowaną strukturę tabeli z prawidłowo wyrównanymi kolumnami i wyświetlane w polu tekstowym.

Struktura połączenia z bazą danych i przepływ informacji:

Aplikacja wykorzystuje MySQL Connector dla Pythona do nawiązania połączenia z bazą danych. Połączenie jest obsługiwane przez kilka warstw:

Podstawowe połączenie z bazą:

```
mydb = mysql.connector.connect(
    host="localhost",
    user=username,
    password=password,
    database="bazy_danych_projekt_mysql"
)
```

To połączenie jest nawiązywane za każdym razem, gdy aplikacja musi pobrać dane, używając poświadczeń użytkownika do uwierzytelnienia.

Struktura przepływu danych:

1. Przetwarzanie Danych Wejściowych:

- Interfejs zbiera wartości filtrów z pól wprowadzania (Region_ID, Energy_ID, Rarity_ID, itp.)
- Wartości te są gromadzone w metodzie apply_filters() w formie słownika
- Puste pola są automatycznie pomijane w procesie filtrowania

2. Konstrukcja Zapytania:

- Podstawowe zapytanie zaczyna się od: "SELECT * FROM Pokemon AS p JOIN Users AS u ON p.User_ID = u.User_ID"
- Dodatkowa klauzula WHERE filtruje po aktualnej nazwie użytkownika
- Dynamiczne warunki filtrowania są dodawane na podstawie wprowadzonych danych
- Parametry są odpowiednio zabezpieczone przed SQL injection

3. Proces Pobierania Danych:

```
# Budowanie zapytania SQL
query = "SELECT * FROM Pokemon AS p JOIN Users AS u ON p.User_ID = u.User_ID WHERE u.Username = %s"
values = [username] # Dodaj nazwę użytkownika jako pierwszy parametr

if filters:
    conditions = []
    for column, value in filters.items():
        conditions.append(f"{column} = %s")
        values.append(value)
```

4. Przetwarzanie Wyników:

- Nazwy kolumn są pobierane z opisu kursora
- Dane są formatowane w strukturę podobną do tabeli
- Szerokości kolumn są dynamicznie obliczane na podstawie zawartości
- Wyniki są formatowane z odpowiednim wyrównaniem i separatorami

5. Integracja z Wyświetlaniem:

- Sformatowane wyniki są wstawiane do widgetu tekstowego
- Widget tekstowy jest tymczasowo odblokowywany na czas wstawiania
- Po wstawieniu widget jest blokowany, aby zapobiec edycji
- Wyświetlane dane są automatycznie aktualizowane po zastosowaniu filtrów

Obsługa błędów:

System zawiera kompleksową obsługę błędów:

- Błędy połączenia z bazą danych są przechwytywane i wyświetlane
- Połączenie jest prawidłowo zamykane w bloku finally
- Komunikaty o błędach są formatowane do wyświetlenia użytkownikowi

Mechanizm odświeżania danych:

```
def display_pokemons(text_field, username, password, filters):
    results = fetch_pokemons(username, password, filters)
    text_field.config(state="normal")
    text_field.delete("1.0", "end")
    text_field.insert("1.0", results)
    text_field.config(state="disabled")
```

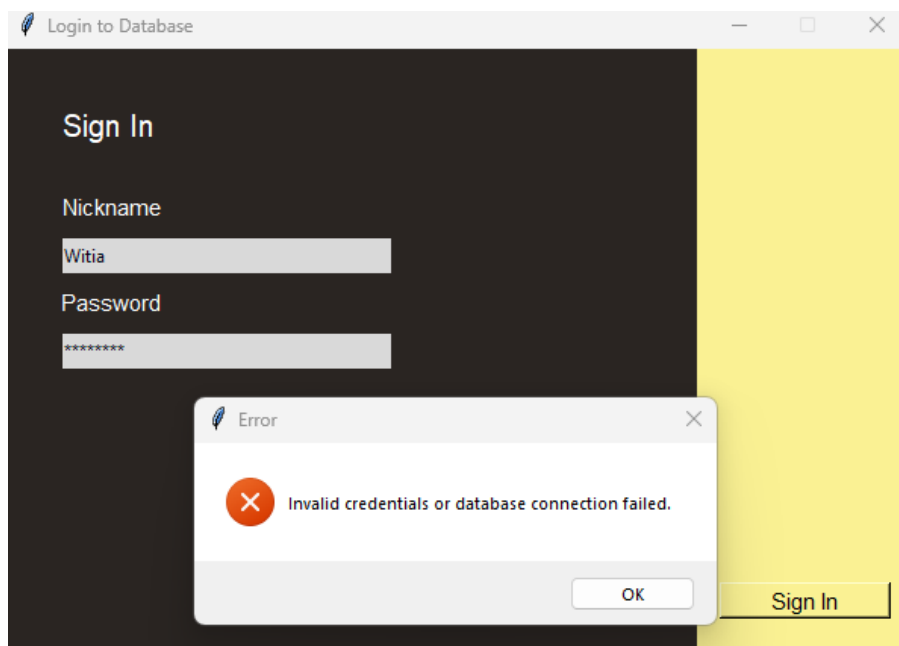
Ta architektura zapewnia:

- Aktualizacje danych w czasie rzeczywistym po zastosowaniu filtrów
- Bezpieczny dostęp do bazy danych przy użyciu poświadczeń użytkownika
- Efektywne formatowanie danych do wyświetlenia
- Prawidłowe zarządzanie zasobami i obsługą połączeń
- Czysty podział między warstwą interakcji z bazą danych a interfejsem użytkownika

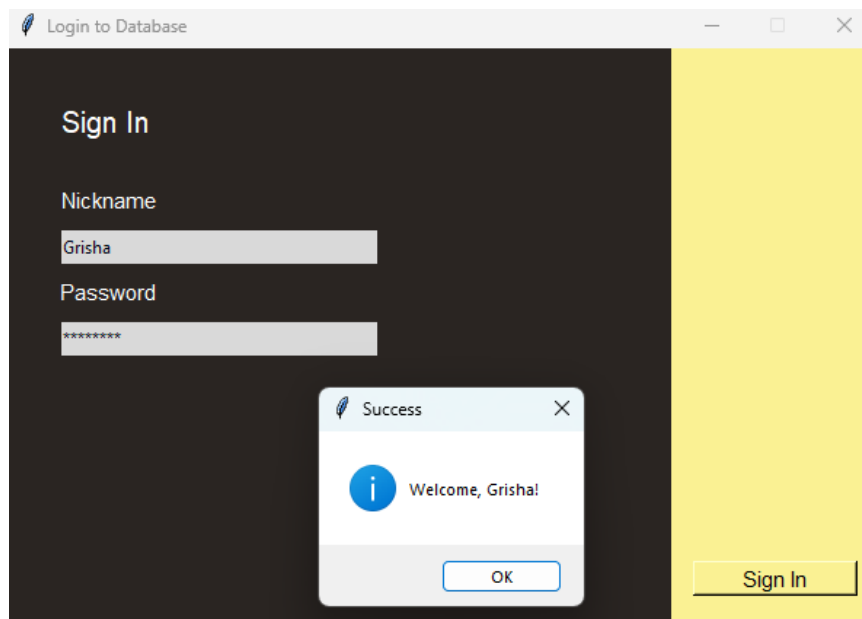
System został zaprojektowany tak, aby był zarówno bezpieczny, jak i wydajny, z odpowiednim podziałem odpowiedzialności między warstwą bazy danych a interfejsem użytkownika. Wszystkie operacje bazodanowe są wykonywane asynchronicznie, aby zachować responsywność interfejsu użytkownika.

14. Testy i niezawodność aplikacji

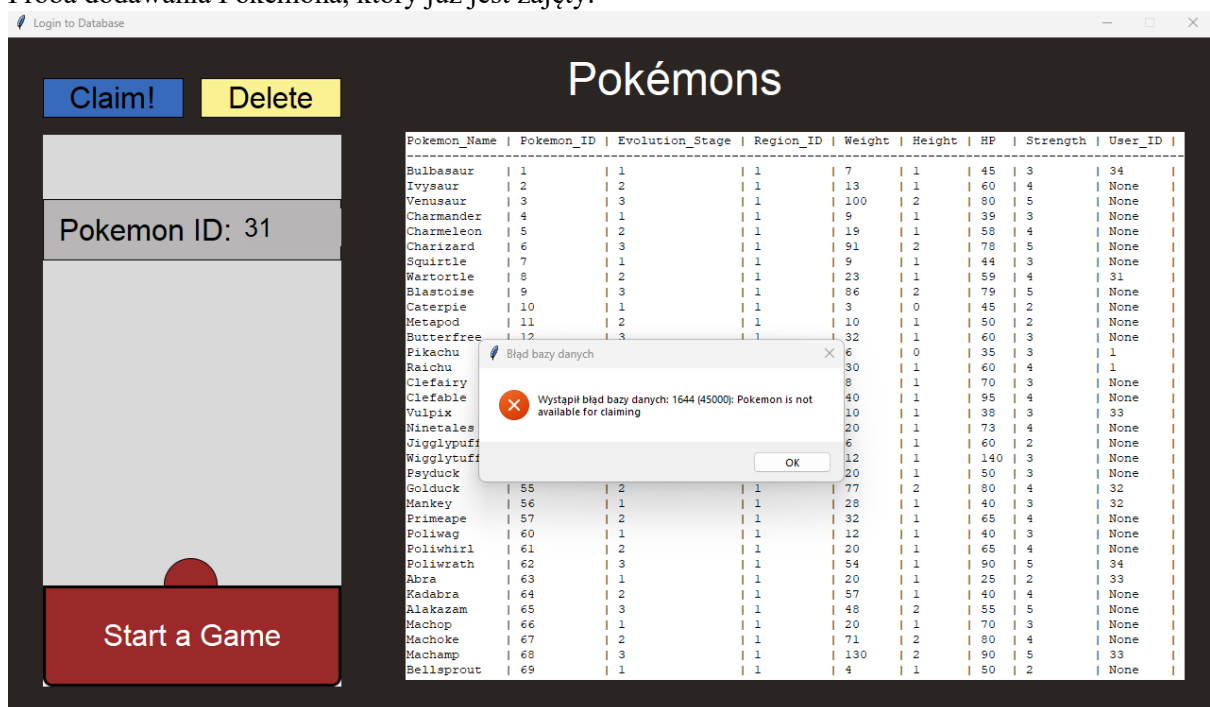
Niepoprawne wpisanie hasła, albo osoba nie istnieje:



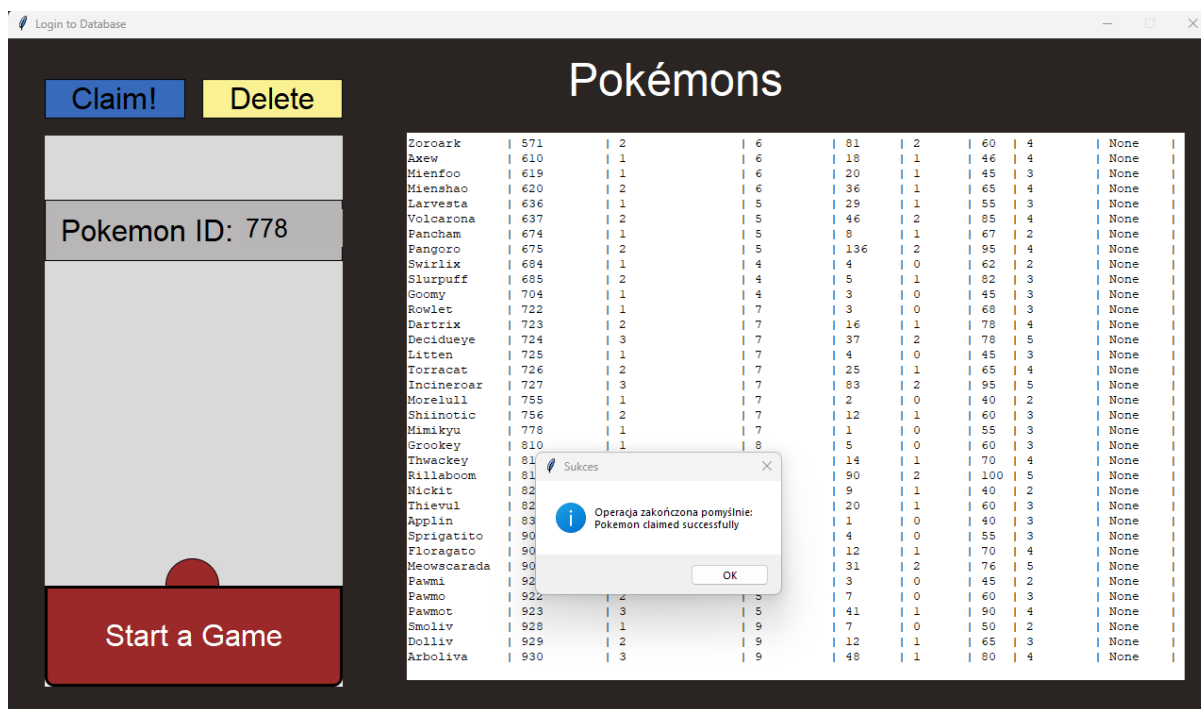
Poprawne Zalogowanie:



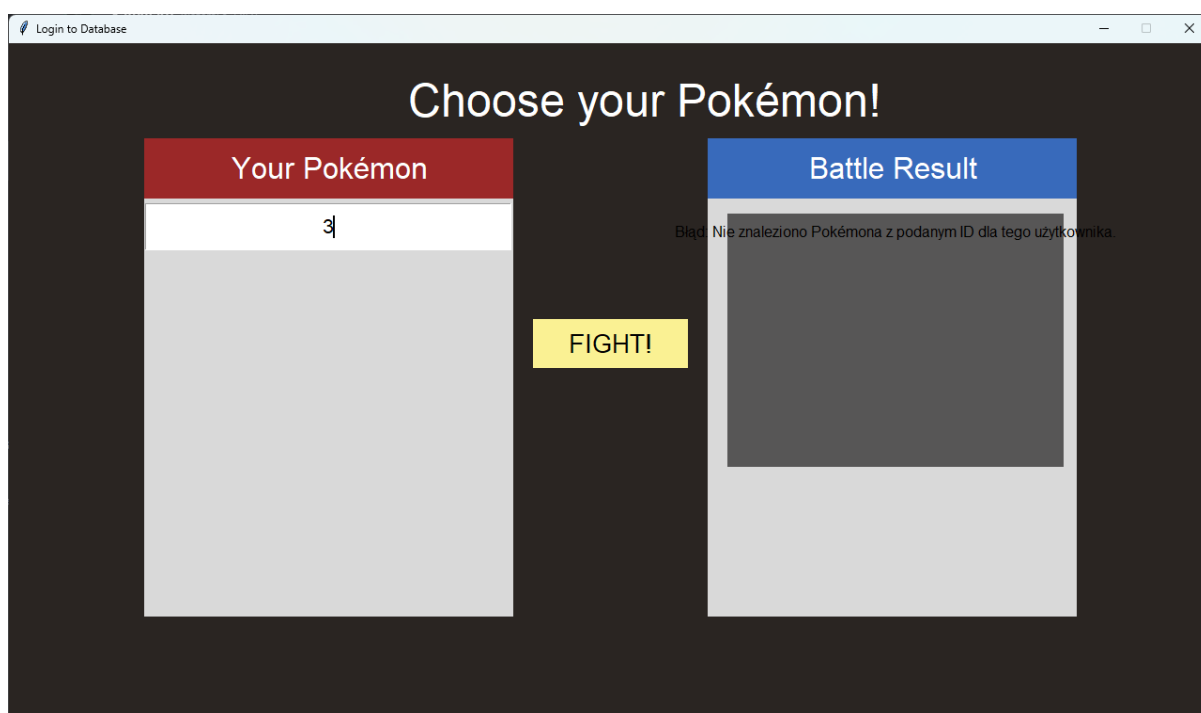
Próba dodawania Pokémona, który już jest zajęty:



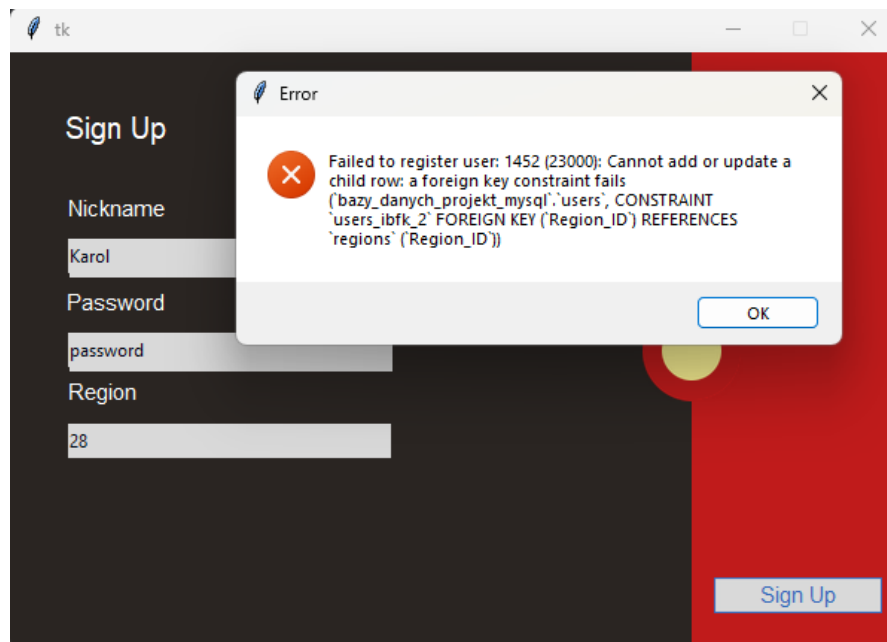
Dodawanie Pokémona, który nie jest zajęty:



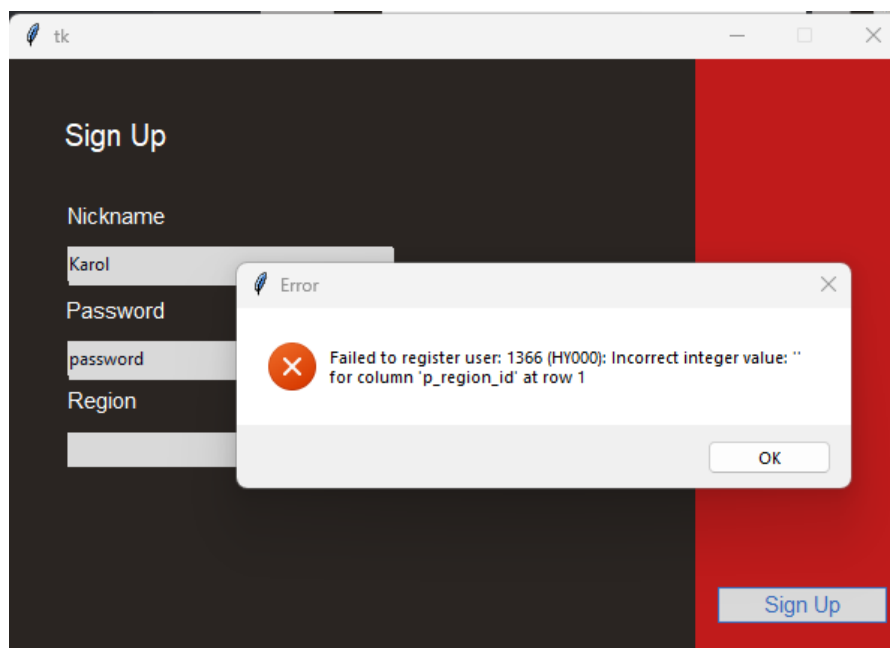
Próba gry, kiedy gracz wprowadzi pokemona, którego nie ma:



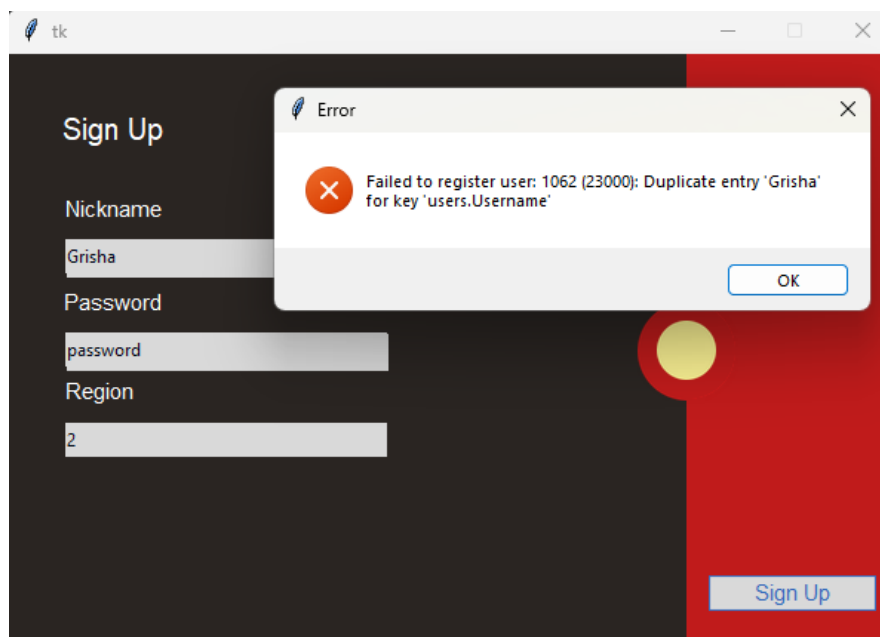
Próba rejestracji osoby do bazy danych, ale wpisywanie złego regionu:



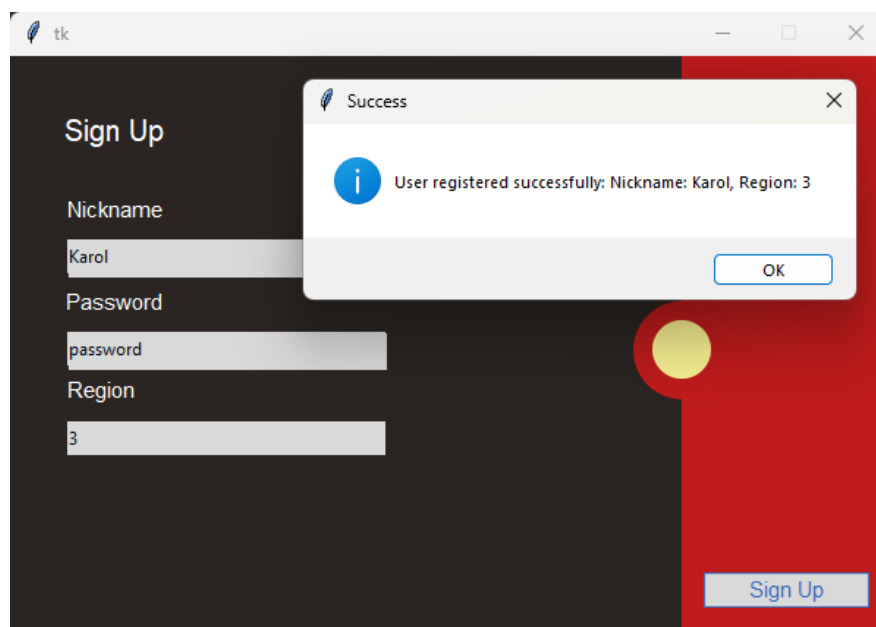
Próba rejestracji osoby bez wprowadzenia regionu:



Próba duplikacji tej samej osoby:



Rejestracja osoby do bazy danych:



Zapisanie tej osoby do bazy:

36	Karol	1	2025-01-15	3	0	HULL	1
----	-------	---	------------	---	---	------	---

15. Lista zmian

Dodano numerację stron, spis treści.

Opis wycinka rzeczywistości:

Dodano do opisu:

Długość życia pokémonów jest uważana za nieograniczoną. Pokémon nie może umrzeć, tylko usunięty z bazy.

Regresja rangi nie występuje w centrum. Gdy osiągniesz rangę — możesz się tylko rozwijać, chyba że to profesor bezpośrednio zdegradował osobę ze stanowiska. Jeśli osoba zostanie usunięta lub zbanowana z centrum, a następnie dołączy ponownie — musi rozpocząć swoją podróż od samego początku.

Zmieniono w opisie dla arcymistrza „Mistrzowie mają wszystkie prawa Mistrzów...” na „Arcymistrzowie mają wszystkie prawa Mistrzów...”

Etap 1:

Trochę powiększono diagram przypadków użycia.

Etap 2:

Trochę powiększono diagram encji. Dodano testy bazy danych, tabele uprawnień.

Etap 3:

Dodano interfejsy i ich makiety dla profesorów. Zmieniono kolor czcionki w grze z czarnego na biały, aby tekst był widoczny w ostatecznej wersji aplikacji. Zmieniono poszukiwanie w bazie po indeksach na poszukiwanie po nazwach, aby użytkownik używał aplikacji bardziej intuitywniej. Na przykład, wpisuje nie „Region: 1” z myślą, że to region Kanto, a wpisuje „Region: Kanto” i wyświetla dane z poprawnym filtrowaniem. Przemieszczono kod dla inicjalizacji interfejsów do opisu budowy interfejsu.