Dokumentacja Techniczna Projektu: Gra Wisielec

Spis Treści

---

1. Wprowadzenie

Niniejszy dokument stanowi dokumentację techniczną dla aplikacji desktopowej "Wisielec". Projekt został zrealizowany w języku Python z wykorzystaniem biblioteki Tkinter do stworzenia interfejsu graficznego (GUI) oraz SQLAlchemy jako warstwy ORM (Object-Relational Mapping) do interakcji z bazą danych SQLite.

Aplikacja jest w pełni funkcjonalną grą logiczną, przeznaczoną dla jednego lub dwóch graczy, z systemem rejestracji, logowania i śledzenia statystyk.

2. Główne Funkcjonalności

- **Interfejs Graficzny Użytkownika (GUI):** Aplikacja działa w jednym oknie, z dynamicznie przełączanymi widokami. Stylistyka interfejsu imituje kartkę z zeszytu.
- **System Użytkowników:** Możliwość rejestracji nowych graczy i logowania na istniejące konta.
- **Bezpieczeństwo Haseł:** Hasła użytkowników są hashowane przy użyciu algorytmu `bcrypt` przed zapisaniem do bazy danych.
- **Dwa Tryby Gry:**
  - **Tryb jednoosobowy:** Gracz zgaduje hasło, mając do dyspozycji 10 prób.
  - **Tryb dwuosobowy:** Dwóch zalogowanych graczy zgaduje naprzemiennie to samo hasło. Każdy z nich ma 5 prób.

- **Baza Danych Haseł:** Aplikacja przy pierwszym uruchomieniu importuje 100 haseł z zewnętrznego pliku `hasla.txt` do bazy danych.
- **Statystyki Graczy:** Dostępny jest ranking graczy, sortowany według liczby zwycięstw.
- **Eksport Danych:** Możliwość eksportu statystyk wszystkich graczy do pliku w formacie CSV.
- **Wizualizacja Gry:** Graficzna reprezentacja wisielca, rysowana krok po kroku po każdej błędnej odpowiedzi.

3. Instalacja i Uruchomienie

1. **Wymagania:**
   - Python 3.8+
   - Zależności wyszczególnione w pliku `requirements.txt`.

2. **Instalacja zależności:**
   ```bash
   pip install -r requirements.txt
   ```

3. **Uruchomienie aplikacji:**
   ```bash
   python main.py
   ```
   Przy pierwszym uruchomieniu aplikacja automatycznie stworzy plik bazy danych `hangman.db` i załaduje do niej hasła z pliku `hasla.txt`.

4. Struktura Projektu

Projekt został zorganizowany w sposób modułowy, aby zapewnić separację odpowiedzialności i łatwość w utrzymaniu kodu.

5. Opis Modułów i Komponentów

5.1. `main.py`

Jest to główny punkt wejścia do aplikacji. Jego jedyną odpowiedzialnością jest:
- Sprawdzenie, czy baza danych istnieje i, w razie potrzeby, wywołanie skryptu `database_setup`.
- Utworzenie instancji głównej klasy aplikacji `HangmanApp`.
- Uruchomienie głównej pętli zdarzeń Tkinter (`app.mainloop()`).

5.2. Moduł `gui`

- **`main_app.py`**: Zawiera cały kod odpowiedzialny za interfejs graficzny.
  - `HangmanApp(tk.Tk)`: Główna klasa aplikacji, która dziedziczy po `tk.Tk`. Działa jako kontroler, zarządzając przełączaniem widoków (ramek).

    - `NotebookFrame(tk.Frame)`: Klasa bazowa dla wszystkich widoków, odpowiedzialna za rysowanie tła w stylu kartki z zeszytu.
    - `LoginFrame`, `Player2LoginFrame`, `MainMenuFrame`, `GameFrame`, `StatsFrame`: Każda z tych klas reprezentuje oddzielny "ekran" w aplikacji (logowanie, menu, gra, statystyki).

5.3. Moduł `core`

- **`game_logic.py`**: Zawiera czystą, niezależną od GUI logikę gry.
    - `HangmanGame`: Klasa przechowująca stan pojedynczej rozgrywki (słowo do odgadnięcia, zgadnięte litery, maksymalna liczba żyć).
    - `SinglePlayer`: Implementacja logiki dla trybu jednoosobowego.
    - `TwoPlayer`: Implementacja logiki dla trybu dwuosobowego, w tym zarządzanie turami i osobnymi pulami żyć dla graczy.

5.4. Moduł `database`

- **`models.py`**: Definiuje strukturę bazy danych za pomocą klas ORM SQLAlchemy.
    - `User`: Mapuje się na tabelę `users`, przechowując dane o graczach.
    - `Word`: Mapuje się na tabelę `words`, przechowując hasła do gry.
- **`database_setup.py`**: Skrypt odpowiedzialny za inicjalizację bazy danych. Tworzy tabele i ładuje hasła z pliku `hasla.txt`.
- **`crud.py`**: Zestaw funkcji do wykonywania podstawowych operacji na bazie danych (Create, Read, Update, Delete), np. `get_random_word()`, `update_user_stats()`.

5.5. Moduł `services`

- **`auth.py`**: Serwis odpowiedzialny za logikę biznesową związaną z autentykacją użytkowników. Zawiera funkcje do rejestracji (`register_user`), logowania (`login_user`) oraz hashowania i weryfikacji haseł za pomocą `bcrypt`.
- **`export.py`**: Serwis do eksportowania danych. Zawiera funkcję `export_stats_to_csv`, która pobiera dane wszystkich użytkowników i zapisuje je do pliku CSV.

6. Kluczowe Aspekty Implementacji

6.1. Przepływ Sterowania i Danych

Aplikacja działa w oparciu o architekturę sterowaną zdarzeniami (event-driven), co jest typowe dla aplikacji GUI. Interakcja użytkownika (np. kliknięcie przycisku) wywołuje odpowiednią funkcję (callback), która następnie może:
1. Wywołać funkcję z modułu `core`, aby zaktualizować logikę gry.
2. Wywołać funkcję z modułu `database.crud` lub `services`, aby odczytać lub zapisać dane.
3. Zaktualizować stan interfejsu graficznego.

6.2. Zarządzanie Stanem GUI

Zastosowano wzorzec z jedną główną klasą-kontrolerem (`HangmanApp`), która przechowuje wszystkie widoki (ramki `tk.Frame`) w słowniku. Metoda `show_frame()` podnosi wybraną ramkę na wierzch, co daje efekt przełączania się między stronami bez niszczenia i ponownego tworzenia okien.

6.3. Persystencja Danych (ORM)

Zamiast pisać surowe zapytania SQL, projekt wykorzystuje SQLAlchemy ORM. Dzięki temu operacje na bazie danych są wykonywane za pomocą obiektów Pythona (np. `db.query(User)`), co czyni kod bardziej czytelnym, bezpiecznym i niezależnym od konkretnego silnika bazy danych.

6.4. Bezpieczeństwo

Kluczowym aspektem bezpieczeństwa jest ochrona haseł użytkowników. Aplikacja nigdy nie przechowuje haseł w formie czystego tekstu. Zamiast tego, używa biblioteki `bcrypt` do tworzenia silnych, solonych hashy. Podczas logowania, podane hasło jest hashowane i porównywane z hashem zapisanym w bazie.

7. Podsumowanie

Projekt "Wisielec" jest kompletną, modularną aplikacją desktopową, napisaną zgodnie z dobrymi praktykami programistycznymi. Wykorzystanie ORM, odpowiednich wzorców projektowych dla GUI oraz dbałość o bezpieczeństwo danych użytkowników czynią go solidną bazą do dalszej rozbudowy.