Dokmentacja projektu

Opis

Przedmiotem projektu jest prosty serwis społecznościowy zrealizowany jako aplikacja okienkowa.

Program umożliwia użytkownikowi

- Rejestrację (utworzenie nowego konta)
- Zalogowanie i wylogowanie się
- Spersonalizowanie profilu
  - Ustawienie krótkiego opisu (bio) widocznego na czacie
  - Ustawienie zdjęcia profilowego
- Wysyłanie i akceptowanie zaproszeń do grona znajomych innych użytkowników
- Wysyłanie i odbieranie wiadomości od znajomych

Wszystkie informacje są lokalnie utrwalane i użytkownik ma do nich dostęp po ponownym uruchomieniu programu.

Struktura projektu

W katalogu źródłowym znajdują się

- pakiet `core` realizujący główną logikę aplikacji
- pakiet `persistence` realizujący warstwę utrwalania aplikacji
- pakiet `gui` realizujący graficzny interfejs użytkownika
- pakiet `tests` zawierający testy jednostkowe
- plik `requirements.txt` z zależnościami
- katalog `examples` z przykładowym plikiem używanym przez warstwę utrwalania
- pliki konfiguracyjne i pomocnicze narzędzia (flake8, mypy, skrypt generujący komponenty do GUI)

Pakiet `core`

Moduł `authentication`

Klasa `Authentication` realizuje mechanizm uwierzytelniania użytkownika. Wykorzystuje do tego bazę istniejących użytkowników i w bezpieczny sposób przechowuje i porównuje hasła wykorzystując algorytm SHA-256 i losową sól.

Moduł `factory`

Zawiera pomocnicze funkcje ułatwiające tworzenie instancji klas wraz z ich zależnościami

Moduł `identifiers`

Odpowiada za generowanie unikalnych identyfikatorów dla obiektów przechowywanych w bazie danych.

Wykorzystuje do tego moduł `uuid` z biblioteki standardowej, w celu zapewnienia unikalności generowanych identyfikatorów.

Moduł `model`

Zawiera klasy modelowe, przechowujące dane używane przez aplikację.

Do klas modelowych należą

- `User` - użytkownicy aplikacji
- `Message` - wiadomości wysyłane między użytkownikami
- `FriendRequest` - zaproszenia do grona znajomych
- `Photo` - obrazy, zdjęcia profilowe użytkowników

Pola klas podlegają walidacji, podanie niepoprawnych danych powoduje zgłaszanie szczegółowych wyjątków

Moduł `serializers`

Odpowiada za serializację i deserializację instancji klas modelowych do formatu JSON.

Format JSON jest wykorzystywany przez warstwę utrwalania.

Klasy:

- `UserSerializer`
- `MessageSerializer`
- `FriendRequestSerializer`
- `PhotoSerializer`

Moduł `user_service`

Klasa `UserService` stanowi główny interfejs do wykonywania operacji związanych z użytkownikami aplikacji (personalizowanie profilu, wysyłanie zaproszeń, wysyłanie wiadomości).

`UserService` zapewnia kontrolę dostępu do operacji i danych - tylko zalogowany użytkownik może modyfikować swój profil, przeglądać swoje wiadomości itp.

Operacje związane z utrwalaniem deleguje do klas `Repository[T]` . Operacje związane z uwierzytelnianiem deleguje do klasy `Authentication` .

Moduł `validation`

Zawiera pomocnicze funkcje służące do walidacji parametrów i danych podawanych przez użytkowników aplikacji oraz szczegółowe wyjątki sygnalizujące niepoprawność danych.

Pakiet `persistence`

Wastwa utrwalania aplikacji jest zrealizowana przez zapis zserializowanych obiektów modelowych do pliku JSON.

Moduł `interface`

Zawiera interfejsy (`Protocol`) związane z utrwalaniem danych, od których zależą Klasy z warstwy logiki i utrwalania.

Klasy:

- `Database` - operacje odczytu / zapisu
- `JsonSerializer[T]` - serializacja i deserializacja do formatu JSON

Moduł `json_database`

`JsonDatabase` jest implementacją bazy danych w pliku JSON.

Obiekty modelowe są przechowywane w kolekcjach (`users` , `messages` , `friend_requests` , `photos`). Klasa operuje na zserializowanych obiektach.

Moduł `repositories`

Zawiera klasy odpowiedzialne za wyszukiwanie i zapisywanie obiektów modelowych.

Klasa `BaseRepository[T]` jest generyczną, abstrakcyjną klasą bazową odpowiedzialną za wykonywanie podstawowych operacji zapisu, odczytu, usuwania.

Klasy `UserRepository` , `MessageRepository` , `FriendRequestRepository` , `PhotoRepository` dziedziczą po `BaseRepository` i poza generycznymi operacjami, umożliwiają operacje wyszukiwania specyficzne dla obsługiwanego typu danych.

Pakiet `gui`

Zawiera graficzny interfejs użytkownika do aplikacji zrealizowany z użyciem biblioteki `PySide2` .

Aplikacja składa się z okna logowania / rejestracji i z głównego okna realizującego pozostałe funkcjonalności.

Moduł `main`

Stanowi główny punkt wejścia dla programu

Moduł `main_window_tabs`

Zawiera logikę zakładek głównego okna aplikacji

Moduł `login_window_pages`

Zawiera logikę podstron okna logowania / rejestraci

Pakiet `ui_component_templates`

Zawiera pliki XML (z rozszerzeniem .ui) wygenerowane przez Qt 5 Designer stanowiące szablony komponentów GUI.

Do generowania kodu na podstawie szablonów służy skrypt `generate_components.sh`

Pakiet `ui_components`

Zawiera wygenerowane przez skrypt klasy obsługujące komponenty GUI.

Plików nie należy bezpośrednio modyfikować.

Pakiet `resources`

Zawiera dodatkowe zasoby dla aplikacji (zastępcze zdjęcia profilowe)

Pakiet `tests`

Zawiera testy jednostkowe do pakietów `core` i `persistence` .

Wykorzystuje bibliotekę `pytest` do uruchamiania testów i asercji.

Wykorzystuje moduł `unittest.mock` z biblioteki standardowej do mockowania zależności w testach jednostkowych.

Plik `conftest.py` zawiera reużywalne komponenty do użycia w testach jednostkowych wykorzystując mechanizm fixtures z `pytest` .

Instrukcja użytkowania

Projekt testowany na wersji Pythona 3.8.10

W wersji 3.11 pojawiają się problemy z kompatybilnością z PySide2

Uruchomienie GUI

W katalogu projektu

```
python -m gui.main {ścieżka do pliku z bazą danych}
```

Przykładowy plik z pustą bazą danych znajduje się w katalogu `examples`

```
python -m gui.main examples/empty-db.json
```

Zainstalowanie zależności

```
pip install -r requirements.txt
```

Wygenerowanie klas do obsługi widgetów

```
./generate_components.sh
```

Uruchomienie testów

Testy jednostkowe

```
pytest ./tests
```

Sprawdzenie typów

```
mypy core persistence gui
```

Sprawdzenie formatowania kodu i docstringów

```
flake8
```

Sprawdzenie pokrycia testami

```
coverage run -m pytest ./tests
coverage report -m
```

Część refleksyjna

Udało się zrealizować wszystkie wymagane funkcjonalności programu i wykorzystać dodatkowe narzędzia do usprawnienia pracy.

Pokrycie testami wynosi 98%.