Wyższa Szkoła Przedsiębiorczości i Administracji w Lublinie

Dokumentacja projektu “flask-todo-list"

Michał Łuczyński

Informatyka PUW

Semestr: 6

Rok akademicki: 24/25

Lublin 2025

Wprowadzenie

flask-todo-list - webowa aplikacja do zarządzania zadaniami (todo). Aplikacja umożliwia tworzenie, przeglądanie, aktualizowanie oraz usuwanie zadań, a także eksport danych w formacie CSV. Projekt został zrealizowany w ramach zajęć akademickich jako praktyczne zastosowanie zasad inżynierii oprogramowania oraz nowoczesnych technologii webowych.

Źródła informacji

Podczas realizacji projektu korzystałem z następujących źródeł:

- Dokumentacja Flask - podstawowe informacje o tworzeniu aplikacji webowych w Pythonie, routing, obsługa szablonów i formularzy.
- Dokumentacja SQLite - materiały dotyczące projektowania i zarządzania bazą danych w kontekście aplikacji Pythonowych.
- Dokumentacja Docker - materiały na temat konteneryzacji aplikacji.
- Dokumentacja Python - oficjalna dokumentacja języka Python, wykorzystana do implementacji funkcjonalności zgodnie z najlepszymi praktykami.
- Materiały dydaktyczne - treści przekazane w ramach zajęć dotyczących projektowania aplikacji webowych.

Wykorzystane technologie

Backend

- **Python 3.13** - wybór języka wysokiego poziomu ze względu na jego czytelność i łatwość wdrożenia
- **Flask 3.1.1** - mikroframework webowy umożliwiający szybkie tworzenie aplikacji z minimalną konfiguracją
- **SQLite** - lekka, wbudowana baza danych, idealna do prostych aplikacji bez konieczności konfigurowania zewnętrznego serwera baz danych
- **Jinja2** - system szablonów HTML ułatwiający tworzenie dynamicznych stron

Frontend

- **HTML5** - strukturalny szkielet interfejsu użytkownika
- **CSS** - stylizacja interfejsu użytkownika
- **Lucide Icons** - lekkie, skalowalne ikony poprawiające użyteczność interfejsu

Narzędzia deweloperskie i wdrożeniowe

- **Docker** - konteneryzacja aplikacji zapewniająca jednolite środowisko uruchomieniowe
- **Gunicorn** - serwer WSGI dla Pythona, umożliwiający wydajne uruchamianie aplikacji Flask w środowisku produkcyjnym
- **python-dotenv** - zarządzanie zmiennymi środowiskowymi w procesie rozwoju aplikacji

Uzasadnienie wyboru technologii

Flask zamiast większych frameworków (np. Django):

- Mniejsza złożoność dla prostej aplikacji typu todo-list
- Większa elastyczność w projektowaniu struktury aplikacji
- Szybsze wdrożenie dla aplikacji o ograniczonym zakresie funkcjonalności

SQLite zamiast systemów RDBMS:

- Brak konieczności konfiguracji zewnętrznej bazy danych
- Wystarczająca wydajność dla niewielkiej ilości danych
- Łatwość wdrożenia i przenośność (baza danych jako plik)

Docker:

- Izolacja środowiska uruchomieniowego
- Łatwość wdrożenia w różnych środowiskach
- Spójność między środowiskiem deweloperskim a produkcyjnym

Architektura aplikacji

Aplikacja została zaprojektowana zgodnie z wzorcem MVC (Model-View-Controller):

Modele (app/models/)

- **base_model.py** - klasa bazowa dla wszystkich modeli, zapewniająca podstawowe operacje CRUD
- **todo.py** - model reprezentujący zadania todo
- **action.py** - model rejestrujący działania użytkownika

Widoki (app/views/)

- Szablony HTML z wykorzystaniem silnika Jinja2
- Responsywny interfejs z wykorzystaniem własnych styli CSS

Kontrolery (app/controllers/)

- **todo_controller.py** - obsługa operacji związanych z zadaniami
- **index_controller.py** - obsługa głównej strony i funkcji eksportu

Narzędzia pomocnicze (app/utils/)

- **database_connection.py** - menadżer połączeń z bazą danych (z obsługą wątków)
- **csv_encoder.py** - konwersja danych do formatu CSV
- **time_ago.py** - formatowanie czasu w przyjazny sposób

Schemat bazy danych

Aplikacja wykorzystuje dwie główne tabele:

Tabela todos - przechowuje zadania do wykonania

- **id** - unikalny identyfikator
- **text** - treść zadania
- **state** - stan zadania (0 - aktywne, 1 - zakończone)
- **created_at** - data utworzenia

Tabela actions - rejestruje działania użytkownika w systemie

- **id** - unikalny identyfikator
- **type** - rodzaj akcji
- **ip** - adres IP użytkownika
- **created_at** - data wykonania akcji

Aspekty profesjonalizmu i etyki

Bezpieczeństwo

- **Ochrona przed SQL Injection** - parametryzowane zapytania SQL w warstwie modelu
- **Bezpieczne tajemnice** - zarządzanie kluczami aplikacji poprzez zmienne środowiskowe
- **Rejestrowanie działań** - audyt działań użytkownika w celu wykrycia potencjalnych nadużyć

Prywatność

- **Minimalna ilość danych** - aplikacja zbiera tylko niezbędne dane (treść zadań)
- **Rejestrowanie IP** - wyłącznie w celach bezpieczeństwa, z poszanowaniem przepisów RODO

Wydajność

- **Optymalizacja zapytań** - przemyślane indeksowanie i struktura bazy danych
- **Efektywne zarządzanie połączeniami** - zamykanie połączeń z bazą danych po każdym żądaniu
- **Lekki frontend** - minimalistyczny interfejs bez zbędnych zależności zewnętrznych

Dostępność

- **Responsywny interfejs** - dostosowanie do różnych rozmiarów ekranu
- **Semantyczny HTML** - prawidłowa struktura dokumentu dla czytników ekranowych
- **Alternatywne opisy dla ikon** - atrybut alt dla wszystkich obrazów i ikon

Dokumentacja i jakość kodu

- **Typowanie** - stosowanie adnotacji typów w Pythonie dla lepszej czytelności i niezawodności
- **Komentarze docstring** - dokumentacja funkcji i klas zgodna z konwencją
- **Przestrzeganie standardów** - kod zgodny z PEP 8 (Python Enhancement Proposals)

Licencjonowanie

- **Licencja projektu** - projekt udostępniony z jawną licencją
- **Przestrzeganie licencji** - wykorzystanie wyłącznie dozwolonych zasobów i bibliotek

Podsumowanie

Projekt flask-todo-list stanowi przykład nowoczesnej aplikacji webowej stworzonej z wykorzystaniem lekkich, ale potężnych technologii. Zastosowanie Flask, SQLite i Docker pozwoliło na szybkie stworzenie funkcjonalnej aplikacji przy jednoczesnym zachowaniu dobrych praktyk inżynierii oprogramowania. Aplikacja została zaprojektowana z myślą o użytkownikach, zapewniając intuicyjny interfejs i podstawowe funkcje zarządzania zadaniami, a jednocześnie spełnia standardy profesjonalnego oprogramowania w zakresie bezpieczeństwa, wydajności i dostępności.