

Zaawansowane programowanie w Pythonie

Design Proposal

Implementacja ORMa dla języka Python

Zespół 2

Aleksander Drwal

Maksym Bieńkowski

Jędrzej Grabski

1. Założenia projektowe

Naszym zadaniem jest stworzenie silnika migracji. Wymagane funkcjonalności w celu wypełnienia tego zadania to:

1. Tworzenie schematu bazy danych
 - Stworzenie interfejsu który zapewni możliwość dość niezależnej komunikacji między ścieżką (zdefiniowane tabele użytkownika - generator SQLa)
 - Oddzielne adaptory dla różnych wariantów (SQLite3, Postgres, MySQL...)
2. Mechanizm porównywania zmian między kolejnymi schematami bazy danych
 - Zapewnienie funkcjonalności przechodzenia w przód i w tył między różnymi wersjami schematu bazy danych
 - Możliwość dokonania i cofnięcia inwazyjnych migracji bez utraty danych zawartych w bazie.
3. Narzędzie CLI do obsługi całego mechanizmu migracji
 - Poruszanie się między wersjami schematu bazy danych z poziomu konsoli
 - Inicjowanie migracji z poziomu konsoli

2. Narzędzia i technologie

- autoformatter - black
- linter - flake8
- środowisko wirtualne - venv
- oskryptowane budowanie, testowanie, uruchamianie aplikacji
- dokumentacja - sphinx

- instrukcja użytkowania - w pliku .md
- semantic versioning
- budowa paczki - pip, pdm
- testy automatyczne - pytest, tox

3. Harmonogram projektu

| | |
|---------------|---|
| 18.03 - 24.03 | <ul style="list-style-type: none"> - Utworzenie szkieletu projektu - Ustalenie ogólnego interfejsu komunikacji z silnikiem migracji - szkielet modelu i zaczątki implementacji - omówienie szczegółów z prowadzącym |
| 25.03 - 31.03 | <ul style="list-style-type: none"> - Implementacja narzędzia CLI, pozwalającego z poziomu konsoli uruchamiać odpowiednie skrypty i przekazywać dane. - Utworzenie testów do narzędzia CLI - Utworzenie testów do parsowania plików wejściowych i tworzenia modelu - Ukończenie interfejsu komunikacji z silnikiem migracji (dla adaptera SQLite3) |
| 01.04 - 07.04 | <ul style="list-style-type: none"> - Początek prac nad parsowaniem plików wejściowych i na ich podstawie tworzeniem modelu. - Zakończenie prac nad parsowaniem i opracowaniem tworzenia modelu w abstrakcji. - Implementacja abstrakcyjnych metod tworzenia modelu (przejście na SQL). - Funkcjonalna implementacja metod tworzących i przywracających. |
| 08.04 - 14.04 | <ul style="list-style-type: none"> - Utworzenie skryptów łączących i automatyzujących funkcjonalności. - Złączenie narzędzia CLI z skryptami uruchamiającymi - Testy całkowitej funkcjonalności. - PROTOTYP |
| 15.04 - 21.04 | <ul style="list-style-type: none"> - Debugowanie i znajdowanie EDGE-CASES. - Początek pracy nad implementacją innego ekosystemu bazodanowego. |
| 22.04 - 28.04 | <ul style="list-style-type: none"> - Koniec implementacji innego systemu, i sprawdzenie jego sprawności. (te same testy przechodzi) - Finalne poprawki |
| 29.04 - 05.05 | TYDZIEŃ ZAPASOWY 1 |
| 06.05 - 12.05 | TYDZIEŃ ZAPASOWY 2 |
| 13.05 - 19.05 | TYDZIEŃ ZAPASOWY 3 |
| 20.05 - 26.05 | TYDZIEŃ ZAPASOWY 4 |

Tygodnie zapasowe poświęcone na ewentualne przedwczesne ukończenie projektu, lub przedłużenie implementacji docelowych funkcjonalności związanych z silnikiem migracji bazy danych, bądź implementację dodatkowych funkcjonalności niekoniecznie związanych z migracjami, np.:

- Interfejs webowy do edycji zawartości bazy danych (na żywo).
- Funkcjonalność tworzenia Pythonowego schematu "w biegu" na podstawie istniejącej bazy danych.

Bibliografia

Django - <https://docs.djangoproject.com/en/5.0/>
<https://github.com/django/django/tree/main/django/db/>

SQLite - <https://www.sqlite.org/docs.html>

SQLAlchemy - <https://docs.sqlalchemy.org/en/20/>

PDM - <https://pdm-project.org/en/latest/>

Pydantic - <https://docs.pydantic.dev/latest/>

Inspect - <https://docs.python.org/3/library/inspect.html>

Python - <https://docs.python.org/3/>

PyTest - <https://docs.pytest.org/en/8.0.x/>

PyTox - <https://tox.wiki/en/4.14.1/>

PyMy - <https://pypi.org/project/pymy/>