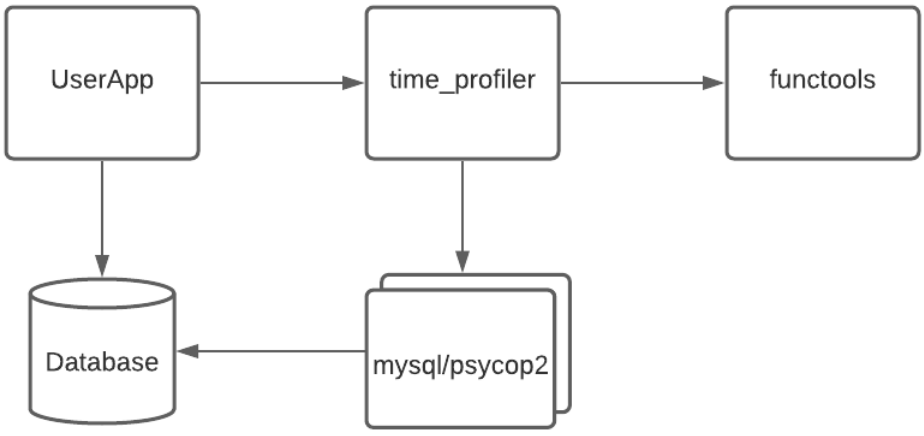
System wyznaczania oraz analizowania wydajności aplikacji klasy enterprise w języku Python – time_profiler

Jan Godlewski, Joanna Partyka, Stanisław Światłoch

1 Architektura fizyczna

Nasza biblioteka łączy się do bazy danych za pomocą bibliotek mysql lub psycop2, w zależności od używanej bazy danych. Aby nie zmieniać atrybutów udekorowanych funkcji używamy również biblioteki functools.



Rysunek 1: Diagram architektury fizycznej

2 Architektura logiczna

Podstawową funkcjonalnością, czyli pomiarem czasu, zajmują się dwie klasy: TimeQuery i TimeExecution. Używane są one jako dekoratory funkcji. Pierwsza z nich wykonuje pomiar wykonania zapytania do bazy danych, druga wykonanie całej funkcji. Używają klasy TimeProfiler do zapisywania wyników pomiaru.

Klasa TimeProfiler przechowuje obiekt Database, który służy do łączenia się z bazą danych, odpowiedniego jej przygotowania i pobierania czasu ostatnio wykonanego zapytania. Do jego tworzenia wykorzystuje DatabaseFactory. Interfejs Database jest implementowany przez PostgresDB, MysqlDB. TimeProfiler jest singletonem, dlatego przechowuje również swoją instancję.

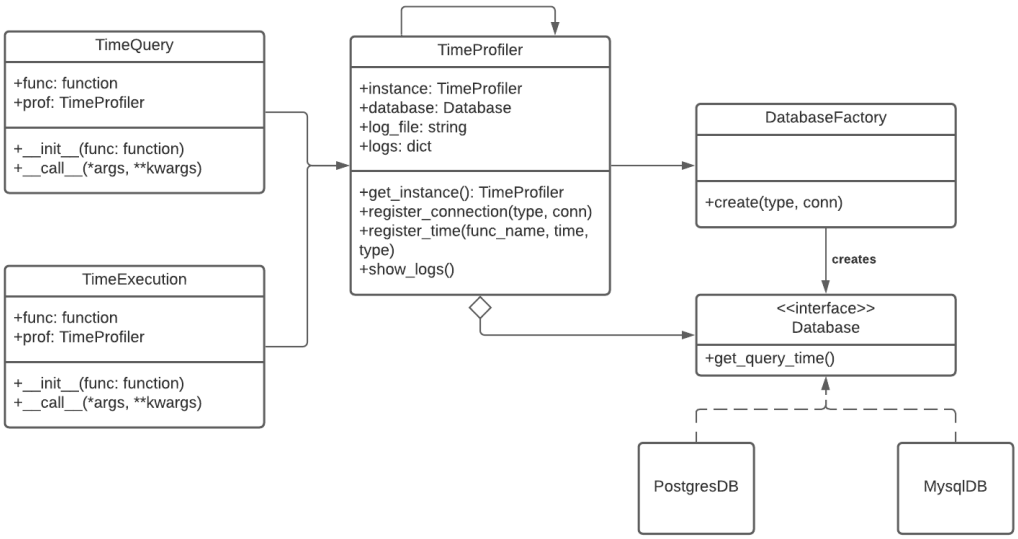
- TimeExecution – dekorator funkcji, dokonuje pomiaru jej wykonania.
- TimeQuery – dekorator funkcji, wykonuje pomiar zapytania do bazy danych w jej wnętrzu.
- TimeProfiler – singleton, używany do zapisywania wyników oraz przechowywania instancji obiektu Database
- Database – interfejs służący do połączenia się z bazą danych, jej odpowiedniego przygotowania oraz pobierania czasu ostatniego wykonanego zapytania. Odpowiednie strategie implementowane są w PostagresDB i MysqlDB.
- DatabaseFactory – fabryka służąca do tworzenia obiektów Database.

3 Opis rozwiązania

Biblioteka oferuje dwa rodzaje pomiarów czasu: wykonania funkcji i wykonania zapytania do bazy danych. Pierwszy z nich jest uzyskiwany na podstawie różnicy między czasem przed wykonaniem funkcji i czasem po jej wykonaniu. W przypadku drugiego, wykorzystujemy logi zapytań, w których znajduje się czas wykonania. Do rozpoznania, które logi należy zmierzyć wykorzystujemy id połączenia do bazy danych.

Wykorzystywane są wbudowane do języka Python dekoratory. Użytkownik oznacza wybrane przez siebie funkcje do testowania dodając przed nimi linijkę @TimeQuery i/lub @TimeExecution. Jeżeli będą wykonywane pomiary czasu zapytań, należy również zarejestrować połączenie z bazą danych, które będzie wykorzystywane do ich wykonywania. Po uruchomieniu programu, wyniki pomiarów będą wyświetlane na konsoli oraz zapisywane do pliku.

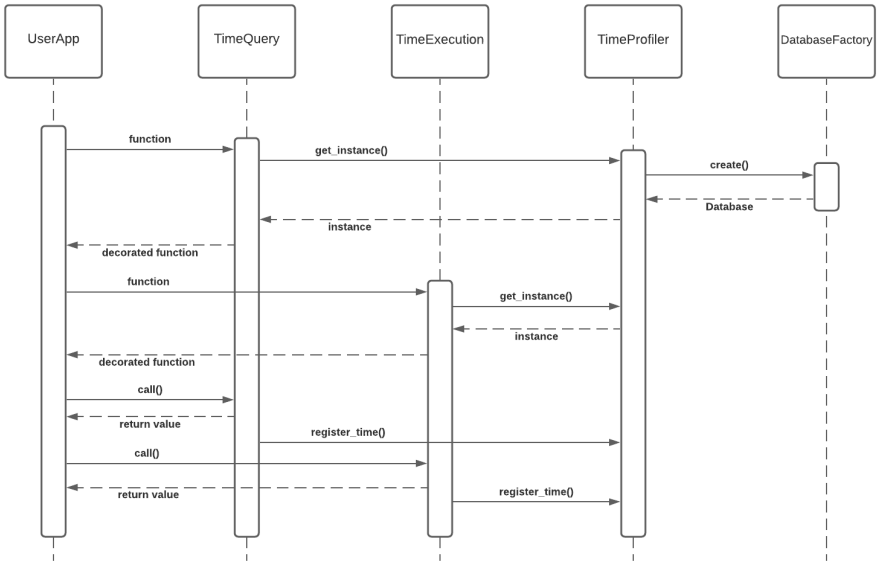
Istnieje również drugi sposób użycia, mianowicie uruchomienie modułu jako skryptu, za pomocą komendy `python -m time_profiler program.py [--output nazwa]`. Skrypt uruchomi program użytkownika, a po jego zakończeniu wyświetli podsumowanie działania. Podając argument output można również wybrać nazwę pliku, do którego zostaną zapisane wyniki.



Rysunek 2: Diagram klas

4 Wykorzystane wzorce projektowe

- Dekorator – TimeQuery oraz TimeExecution są dekoratorami, ponieważ chcemy dodać dodatkową funkcjonalność pomiaru czasu, bez wpływu na wykonanie testowanej funkcji.
- Singleton – Może istnieć wiele instancji TimeQuery oraz TimeExecution. Aby zapisywać informacje w jednym miejscu oraz przechowywać tylko jedno połączenie z bazą danych, TimeProfiler jest singletonem.
- Fabryka prosta – Biblioteka wspiera kilka baz danych. Aby ułatwić tworzenie obiektów do łączenia się z nimi korzystamy z fabryki DatabaseFactory. Pozwala to również na łatwe rozszerzenie wsparcia o kolejne bazy danych.
- Strategia – Dla różnych baz danych, istnieją różne sposoby odczytania czasu wykonania ostatniego zapytania. Dlatego do ich zaimplementowania używamy strategii.



Rysunek 3: Diagram sekwencji