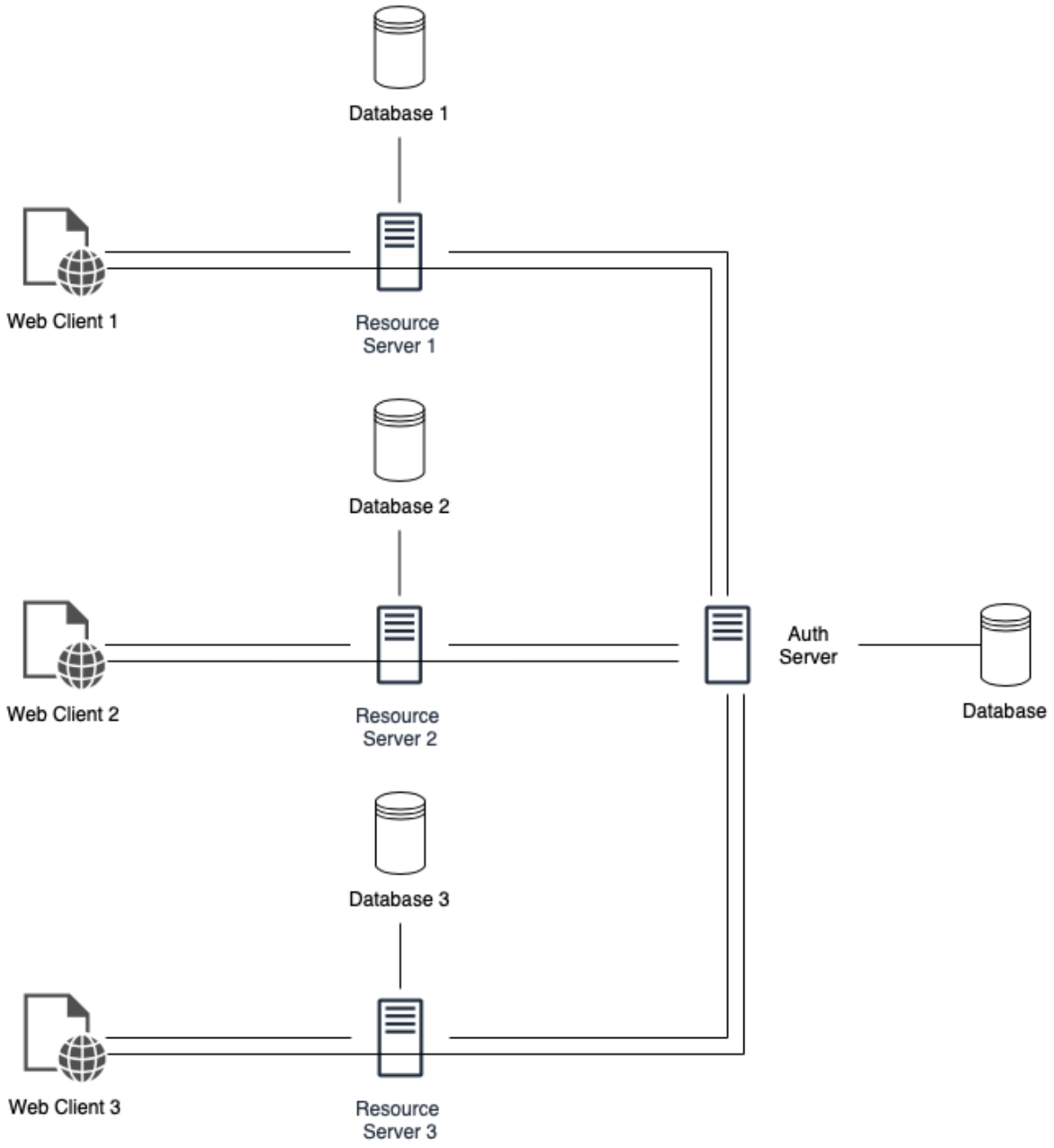
Instytut Informatyki AGH,
Informatyka,
Design Patterns
Semestr V, 2021 r.

Dokumentacja projektu

“Implementacja OAuth2 w Pythonie”

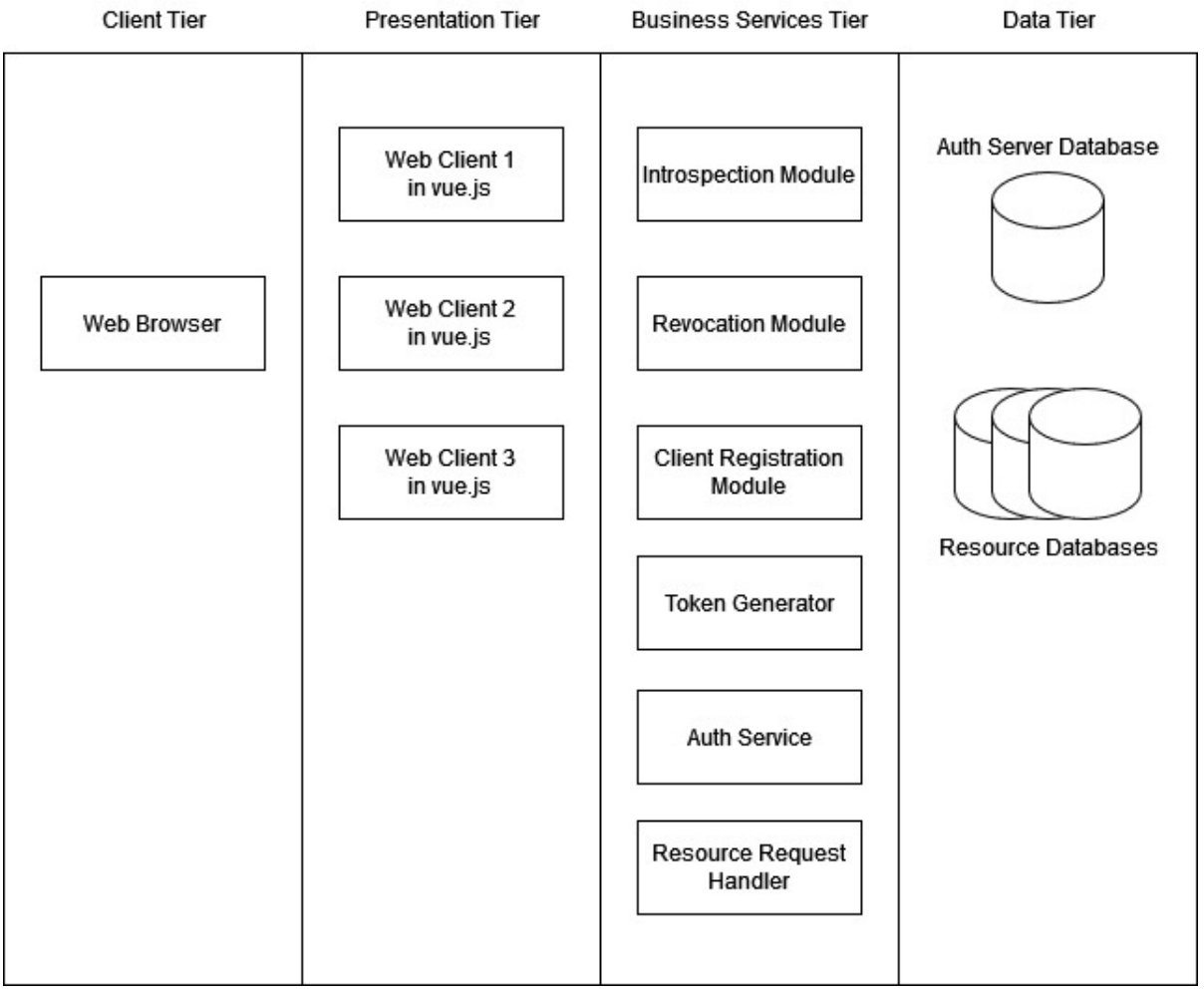
Konrad Kowalczyk
Elżbieta Cymerys
Szymon Frelich
Łukasz Ślęczka

1. Architektura fizyczna



Serwery zasobów (**Resource Server**) komunikują się z bazą danych (**Database**) w celu uzyskania zasobów, o które prosił klient (**Web Client**). Mogą również pytać serwer autoryzacyjny (**Auth Server**) o token przekazany przez klienta poprzez Introspection Endpoint.

2. Architektura logiczna



3. Wykorzystane technologie

W projekcie użyte zostaną:

- JavaScriptowa platforma **vue.js** do stworzenia frontendu aplikacji klienckich (**Web Client**),
- Pythonowa biblioteka **FastAPI** do utworzenia backendu systemu (**Resource Server**, **Auth Server**),
- relacyjna baza danych **PostgreSQL** do zarządzania bazą danych (**Database**).

4. Opis aplikacji klienckich

W projekcie użyjemy trzech aplikacji klienckich (**Web Client**), na których zaprezentowane zostanie działanie **Single Sign-On** (**SSO**):

Web Client 1 będzie prostą aplikacją udostępniającą funkcjonalność tworzenia oraz zapisywania, usuwania i edytowania notatek. Notatki danego użytkownika dostępne będą do wglądu tylko dla niego, dlatego aplikacja będzie wymagać od użytkownika uprawnień do odczytania jego notatek z bazy danych.

Web Client 2 będzie aplikacją udostępniającą funkcjonalność tworzenia TODO listy. Użytkownik będzie mógł tworzyć, zapisywać, usuwać oraz oznaczać jako wykonane dane zadanie. Zadania danego użytkownika dostępne będą do wglądu tylko dla niego, dlatego aplikacja będzie wymagać od użytkownika uprawnień do odczytania jego zadań z bazy danych.

Web Client 3 będzie aplikacją wyświetlającą ogłoszenia wszystkich zarejestrowanych użytkowników oraz każdy zalogowany użytkownik będzie mógł dodać nowe ogłoszenie. Co więcej, użytkownik będzie miał możliwość edycji oraz usuwania swoich opublikowanych ogłoszeń.

5. Rejestracja i logowanie się użytkownika

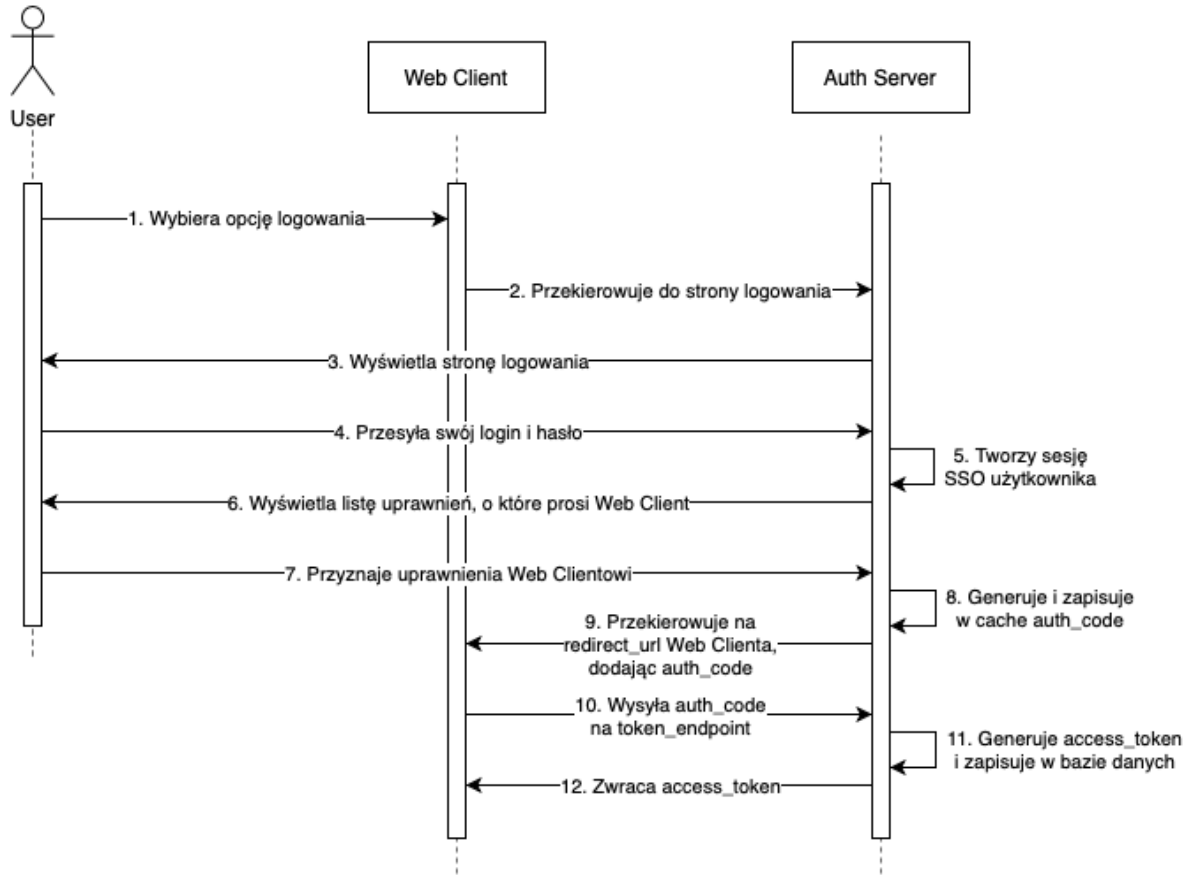
Aby zarejestrować się w systemie użytkownik podaje swój email, login i hasło w formularzu udostępnionym przez serwer autoryzacyjny. Logowanie może być zainicjowane przez użytkownika poprzez wejście pod adres strony logowania, wysyłanej przez serwer autoryzacyjny lub wywołane przez przekierowanie od jednego z klientów, proszących o token dostępowy. Aby zalogować się, użytkownik wpisuje swój email i hasło w formularzu.

6. Rejestracja aplikacji klienckich

Zarejestrowany użytkownik może dodać dowolną ilość aplikacji klienckich. W tym celu musi, będąc zalogowanym, wejść na odpowiednią stronę, udostępnianą przez serwer autoryzacyjny i podać w formularzu następujące informacje o aplikacji:
name - nazwa aplikacji, która wyświetli się użytkownikowi podczas jej autoryzacji,
description - opis aplikacji, wyświetlany użytkownikowi,
redirect_url - adres, pod który API przekieruje użytkownika, po autoryzacji aplikacji.

Po wysłaniu formularza, użytkownikowi wyświetli się wygenerowany, unikalny identyfikator klienta dla aplikacji.

7. Pozyskiwanie tokena przez klienta



1) Użytkownik powiadamia aplikację (poprzez naciśnięcie przycisku “Zaloguj”), że chce się zalogować.
2) Aplikacja kliencka przekierowuje przeglądarkę użytkownika do strony logowania udostępnionej przez serwer autoryzacyjny.
3) Jeżeli użytkownik ma aktywną sesję na serwerze autoryzacyjnym (logował się przez serwer autoryzacyjny), przechodzimy od razu do punktu 6. W przeciwnym wypadku serwer autoryzacyjny wysyła przeglądarce użytkownika formularz logowania.
4) Jeżeli dane logowania są niepoprawne, strona wyświetla użytkownikowi odpowiedni błąd. W przeciwnym wypadku, przechodzimy do następnego punktu.
5) Serwer autoryzacyjny tworzy sesję użytkownika, dzięki czemu przy następnej autoryzacji innej aplikacji, użytkownik nie zostanie poproszony o dane logowania - **Single Sign-On** (**SSO**).
6) Strona logowania wyświetla użytkownikowi informacje, zawierające:
    - nazwę klienta, który prosi o autoryzację,
    - opis klienta
    - listę uprawnień (scope), o które prosi klient.
7) Użytkownik nadaje lub odrzuca dostęp klientowi. W przypadku odmowy dostępu, przeglądarka użytkownika zostanie przekierowana na redirect_url klienta bez kodu autoryzacyjnego.
8) Po uzyskaniu autoryzacji użytkownika, serwer autoryzacyjny tworzy krótkotrwały (1 min) kod autoryzacyjny i zapisuje go w cache.
9) Przeglądarka zostaje przekierowana na redirect_url klienta z kodem autoryzacyjnym umieszczonym w query string o nazwie code.
10) Klient odczytuje kod z URL i wysyła zapytanie HTTP POST na token_endpoint serwera autoryzacyjnego.
11) Jeśli kod jest poprawny, serwer autoryzacyjny generuje nowy token dostępowy, wiąże go z użytkownikiem i klientem i zapisuje w bazie danych. Token użytkownika będzie ważny, dopóki nie zostanie unieważniony.
12) Serwer autoryzacyjny wysyła odpowiedź do klienta zwracając mu token dostępowy.

Przykładowe zapytanie HTTP z punktu 10:

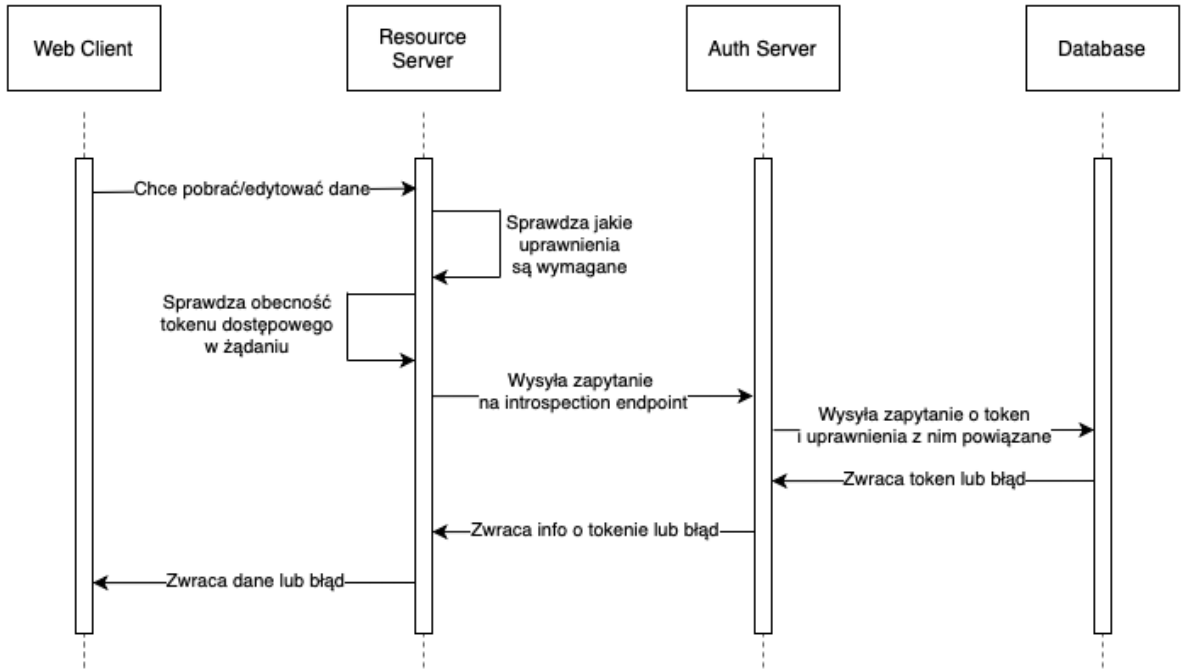
```
HTTP POST {AUTH_SERVER}/access-token
Content-Type: application/json;charset=UTF-8
{
   "client_id": "abc123",
   "code": "xyz789"
}
```

Przykładowa odpowiedź serwera autoryzacyjnego w przypadku sukcesu:

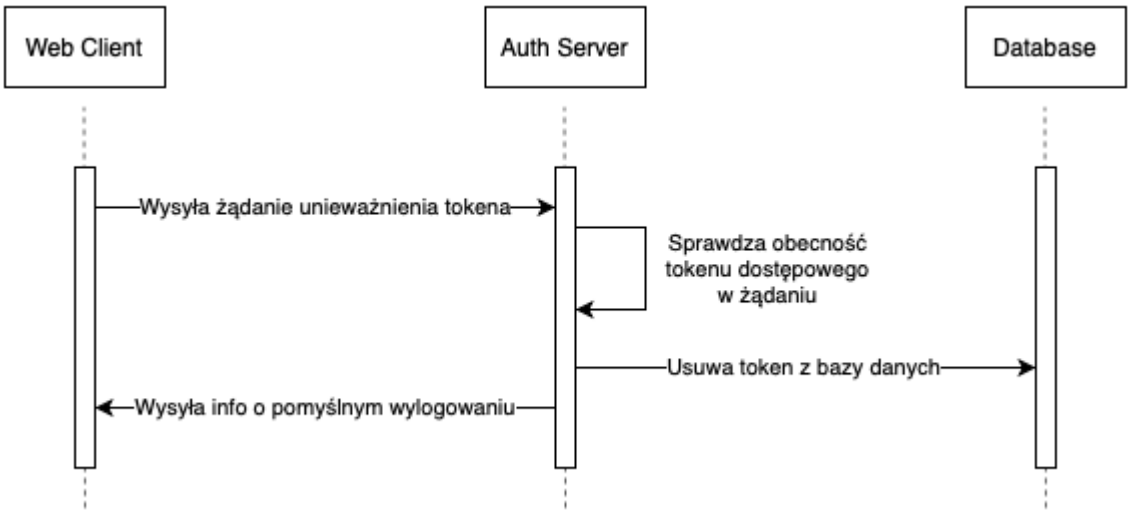
```
HTTP 200 OK
Content-Type: application/json;charset=UTF-8
{
   "access_token": "ghs_82198hdua7i9AhhsB"
}
```

Zwracany jest token dostępowy, będący prostym identyfikatorem (bezpiecznie losowo wygenerowanym ciągiem znaków).

8. Token Introspection



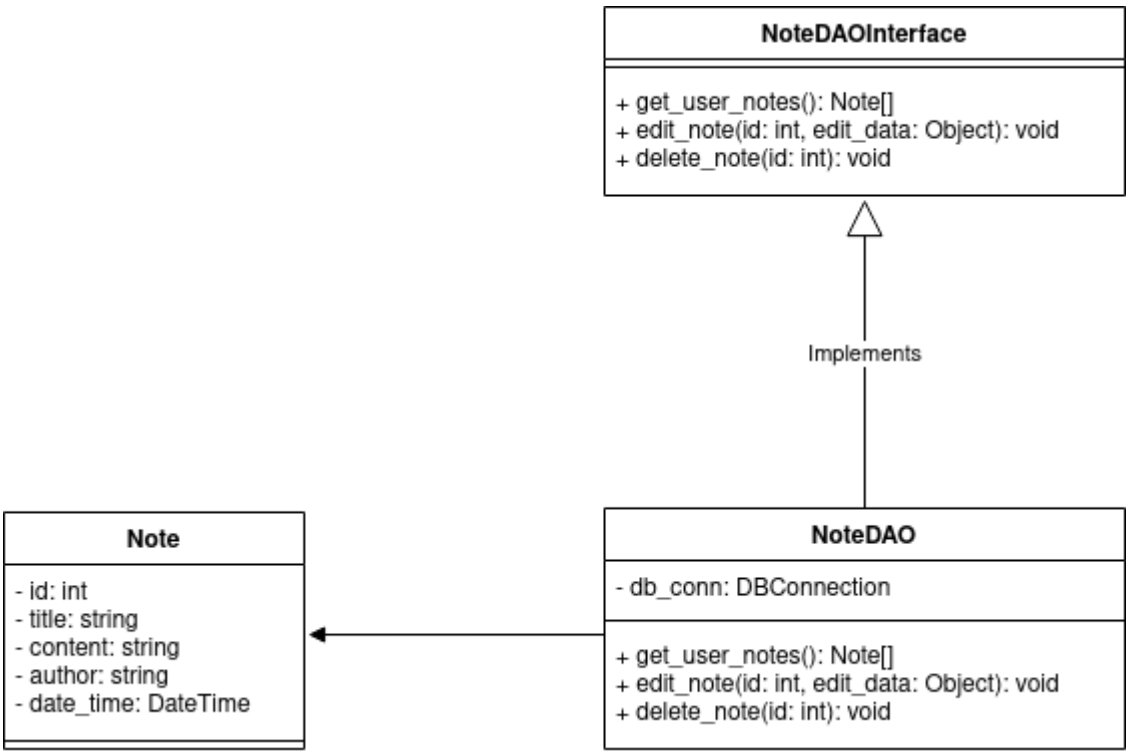
9. Token Revocation



1) Klient powiadamia serwer (po naciśnięciu przycisku “Wyloguj” przez użytkownika), że uprzednio uzyskany token już nie jest potrzebny.
2) Serwer sprawdzi i usunie z bazy token i wyśle do klienta informację o pomyślnym wylogowaniu.

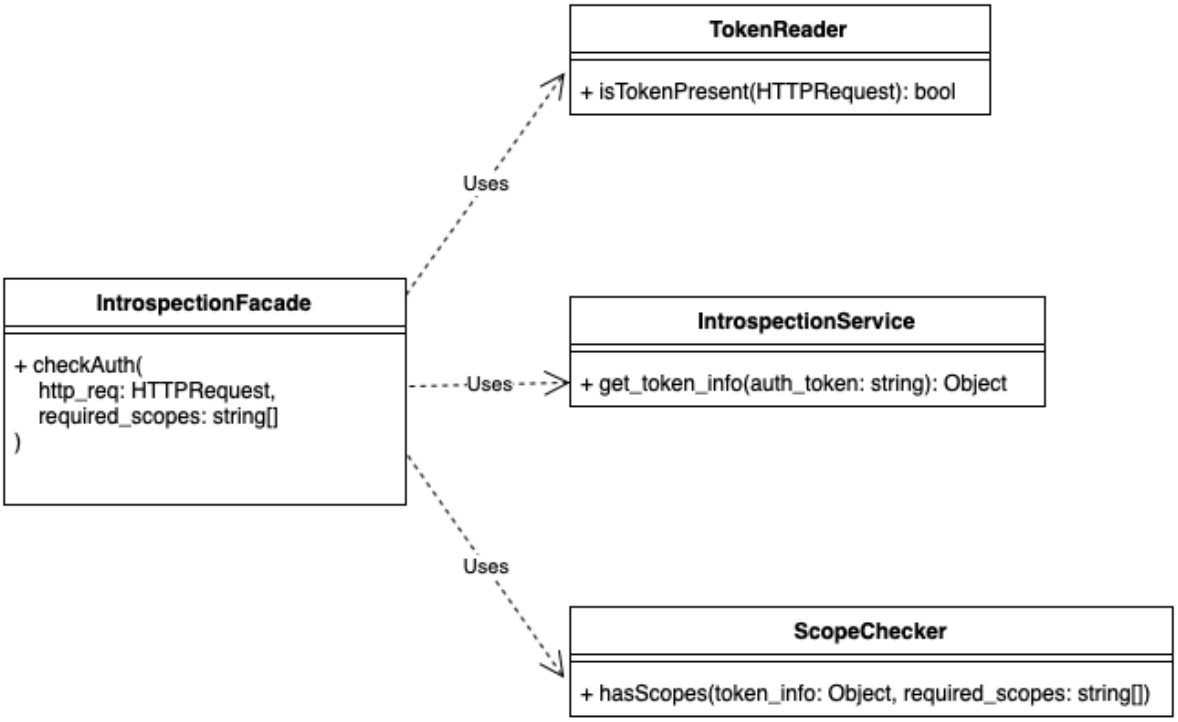
10. Wzorce projektowe

1) **Data Access Object** (**DAO**) - udostępnia interfejs do komunikacji z bazą danych do osobnej klasy oraz wprowadza warstwę abstrakcji między aplikacją a bazą danych. Dzięki temu aplikacja nie wie jak ani gdzie składowane są jej dane, co ułatwia ewentualną zmianę bazy danych.



2) **Fasada** - wprowadza prosty interfejs do wykonywania złożonych operacji. W tym przypadku chodzi o operację sprawdzania autoryzacji aplikacji. Takie sprawdzenie składa się nawet z kilku kroków:
    - sprawdzenia obecności tokena w zapytaniu,
    - wysłania zapytania na introspection endpoint,
    - sprawdzenia czy aplikacja ma nadane odpowiednie uprawnienia (scope).

    Fasada może uprościć wywoływanie powyższego, np. poprzez udostępnienie metody checkAuth(http_req, required_scopes), wykonującej wszystkie powyższe czynności.



3) **Builder** - posłuży do budowania skomplikowanych adresów URL, zawierających dużo zmiennych query string. Zamiast składać cały URL jako string, dodając do niego ręcznie kolejne zmienne, builder udostępni metody dodające wykorzystywane w OAuth2 zmienne, takie jak redirect_url, scope, itd.