AKADEMIA GÓRNICZO-HUTNICZA
IM. STANISŁAWA STASZICA W KRAKOWIE

Wydział Informatyki, Elektroniki i Telekomunikacji
Katedra Informatyki

INŻYNIERIA OPROGRAMOWANIA - PROJEKT

WERYFIKACJA POPRAWNOŚCI WYNIKÓW ZWRACANYCH PRZEZ SERWER DNS

KONCEPCJA SYSTEMU

MATEUSZ BIELESZ, WOJCIECH KOSIOR, MAREK MORYL, KAMIL SZAREK

KIERUNEK:
Informatyka

OPIEKUN:
mgr inż. Witold Rakoczy

Kraków, 2020

Spis treści

1. Wstęp

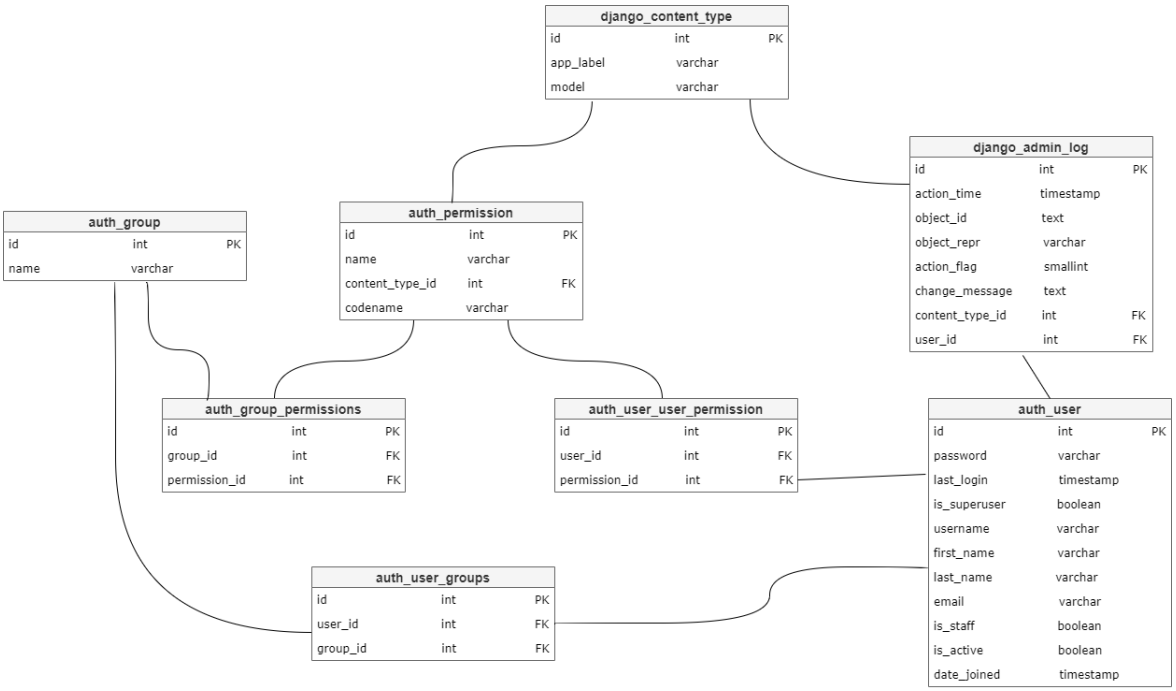
Nasz system będzie złożony z następujących komponentów:

- front-end stworzony przy pomocy frameworku Django. Za jego pośrednictwem użytkownik będzie korzystał z naszego systemu.
- relacyjna baza danych, w której będą przechowywane wszystkie informacje potrzebne do funkcjonowania systemu.
- back-end, który będzie odpowiadał za ustawienia połączeń i zbieranie danych z serwerów DNS oraz za komunikację pomiędzy front-endem, a bazą danych.

2. Baza danych

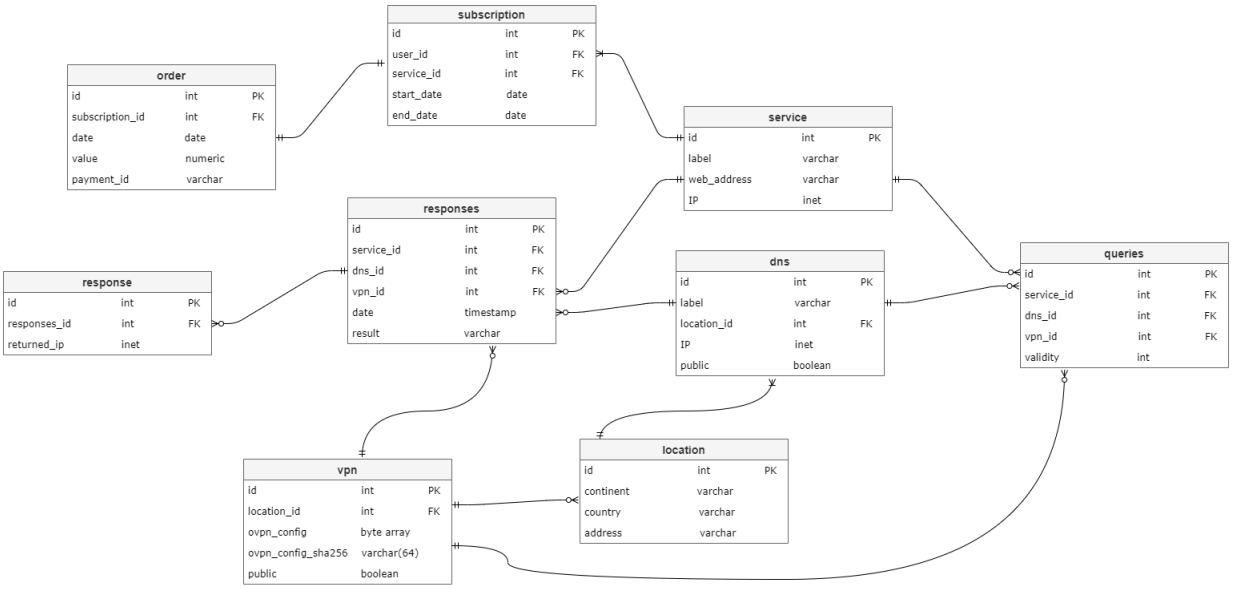
Baza danych jest to element łączący części systemu stanowiące front-end i back-end. Można ją podzielić na dwie części: przechowującą dane o użytkownikach oraz przechowującą dane o systemie. Podczas prowadzenia projektu korzystaliśmy z dwóch baz danych: SQLite, lokalnej bazy danych automatycznie tworzonej przez framework, używanej na początku projektu oraz do przeprowadzenia testów oraz PostgreSQL, dostępnej zdalnie, używanej jako łącznik pomiędzyu backendem i frontendem.

2.1. Baza danych użytkowników



Baza danych użytkowników została automatycznie stworzona przez framework Django. Umożliwia ona korzystanie z systemu dwóm typom użytkowników: normalnym użytkownikom oraz administratorom. Administrator ma dostęp do wszystkich danych przechowywanych w systemie, może je dowolnie dodawać, usuwać lub edytować. Użytkownik ma dostęp tylko do swoich danych.

2.2. Baza danych systemu



Baza danych systemu została stworzona przez nas do przechowywania danych serwisu. Łączy się ona z bazą danych użytkowników poprzez pole *user_id* w tabeli *subscriptions*, odpowiadające polu *id* w tabeli *auth_user*.

W całym projekcie przyjęta została konwencja nazewnicza używana w Django, w której każda tabela ma pole *id*, będące autoinkrementowanym kluczem głównym, a klucz obcy odnoszący się do klucza głównego w innej tabeli nosi nazwę <*nazwa tabeli*>_*id*. Tabele są tworzone przez framework i noszą nazwy <*nazwa pakietu*>_<*nazwa tabeli*> , w naszym przypadku podzieliliśmy projekt na dwie części:

- public_side - część odpowiada za funkcjonalności dostępne dla użytkownika niezalogowanego, w tym obsługę strony głónej wraz z jej podstronami, obsługę logowania oraz rejestracji,

- user_side - część odpowiada za funkcjonalności dostępne dla użytkownika zalogowanego, ma on dostęp do czterech zakładek: home, profile, statistics oraz buy subscription,

2.2.1. Opis tabel

dns tabela przechowująca znane systemowi serwery dns, każdy z nich ma obowiązkowe pole IP, przechowujące IP serwera do odpytania, pola label i location$_i dsopcjonalne, polepublicmadomylnwartoTrueiokrela, czydanyserwerjestdo$

location tabela przechowująca dane lokalizacyjne, o serwerach DNS lub VPN,

order tabela zawierająca szczegóły zamówienia: datę zatwierdzenia, wartość zamówienia oraz numer płatności, który jest numerem zwrócony przez zewnętrzną stronę, zajmującą się obsługą płatności,

queries tabela łącznikowa, łącząca serwis z serwerami dns które należy sprawdzać oraz vpn'ami, których należy do tego użyć, pole validity określa ilość zapytań do wykonania w ramach wykupionej subskrypcji, zawiera liczbę dodatnią, jest ustawiane przy tworzeniu rekordu i dekrementowane przy wykonaniu zapytania, po uzyskaniu wartości równej 0 rekord jest usuwany,

responses tabela zawierająca historię odpytywania serwerów dns o poszczególne serwisy, z uwzględnioną datą i rezultatem, będącym rezultatem zwróconym przez część back-endową,

service tabela przechowująca dane o serwisach klientów, które należy sprawdzać, obowiązkowe są pola web_address, zawierające pełen poprawny adres strony oraz IP, zawierające oczekiwane IP,

subscription tabela zawierająca informację o subskrypcji, w tym datę rozpoczęcia i wygaśnięcia,

vpn tabela zawierająca dane o VPNach, pole ovpn_config zawiera treśćpliku konfiguracyjnego, pole ovpn_config_sha256 zawiera hash (wyliczany przy dodawaniu pliku na podstawie jego zawartości), pole public określa, czy VPN jest wykupiony jako część systemu i dostęny dla wszystkich użytkowników (wartość True), czy jest to VPN dodany przez użytkownika (wartość False)

2.2.2. Opis przechowywanych obiektów

użytkownik — w systemie są dwa typy użytkowników: klienci oraz administratorzy, administrator ma nieograniczony dostęp do wszystkich elementów systemu, klient tylko do przypasanych do niego rekordów, obsługa logowania, sesji oraz kwestie bezpieczeństwa zapewnione są przez framework Django, wszystkie dane przechowywane są w tabeli pokazanej w rozdziale 2.2.1

serwery DNS — znane systemowi dane o serwerach DNS przechowywane są w tabeli *dns*, lokalizacja serwera w tabeli *location*, rozróżniamy serwery publiczne (z polem public ustawionym na True) - serery dodane przez administratora, oraz serwery prywatne, dodane przez użytkowników jako te, które też mają być sprawdzane

połączenia VPN — w tabeli *vpn* przechowywane są pliki konfiguracyjne potrzebne do nawiązania poleczenia wraz z unikalnymi hashami

zapytania DNS — zapytania tworzone są na podstawie tabeli *queries*, która łączy serwisy z przypisanymi do niech serwerami DNS które należy odpytać i VPNami, których należy użyć, pole *validity* jest liczbą zapytań, które jeszcze należy wykonać w ramach subskrypcji,

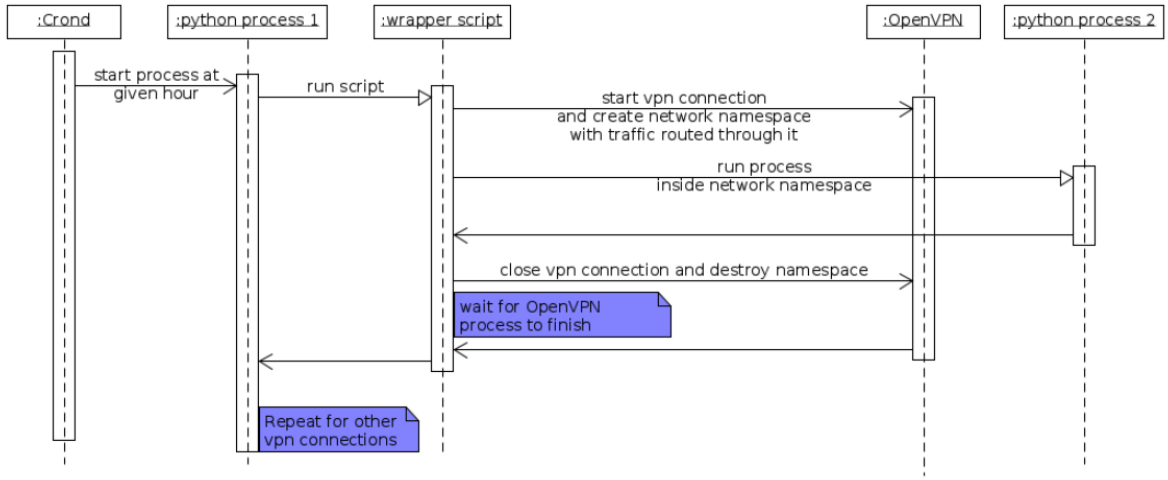
odpowiedzi DNS — za przechowywanie odpowiedzi, które pokazywane są klientowi odpowiadają tabele Responses oraz Response - podział ten wynika z faktu, że jeden serwer DNS może zwrócić kilka adresów IP, pole *result* w tabeli Responses określa rezultat zapytania i ma wartość *successful* jeśli się ono powiodło, jednak jego wynik należy jeszcze sprawdzić z poprawną wartością IP serwisu, z tabeli services,

subskrypcje — dane o subskrypcji przechowywane są w tabeli *subscription* oraz *order*,

3. Back-end

Połączenia z serwerami DNS będą realizowane za pomocą oprogramowania OpenVPN. Na początku pierwszy proces w Pythonie inicjuje połączenie VPN i tworzy dla niego sieciową przestrzeń nazw. Następnie w tej przestrzeni jest uruchamiany drugi proces, który zbiera informację o połączeniu i przesyła je do bazy danych. Po zakończeniu tego procesu przestrzeń nazw jest usuwana, a połączenie VPN zakończone. Te czynności są następnie powtarzane dla wszystkich pozostałych połączeń.

Poniżej przedstawiony jest schemat tej operacji:

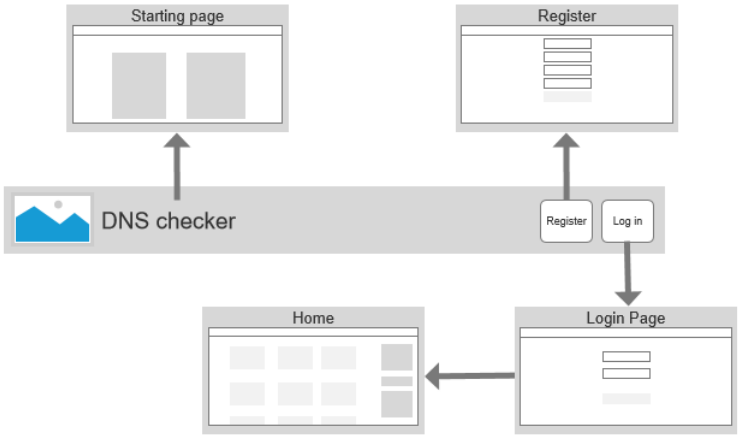


4. Front-end

4.1. Strona użytkownika niezalogowanego

Użytkownicy będą korzystali z naszej aplikacji za pośrednictwem strony internetowej. Nawigacja po niej będzie realizowana poprzez zastosowanie paska nawigacyjnego na górze ekranu. Dostępne będą dla nich następujące komponenty:

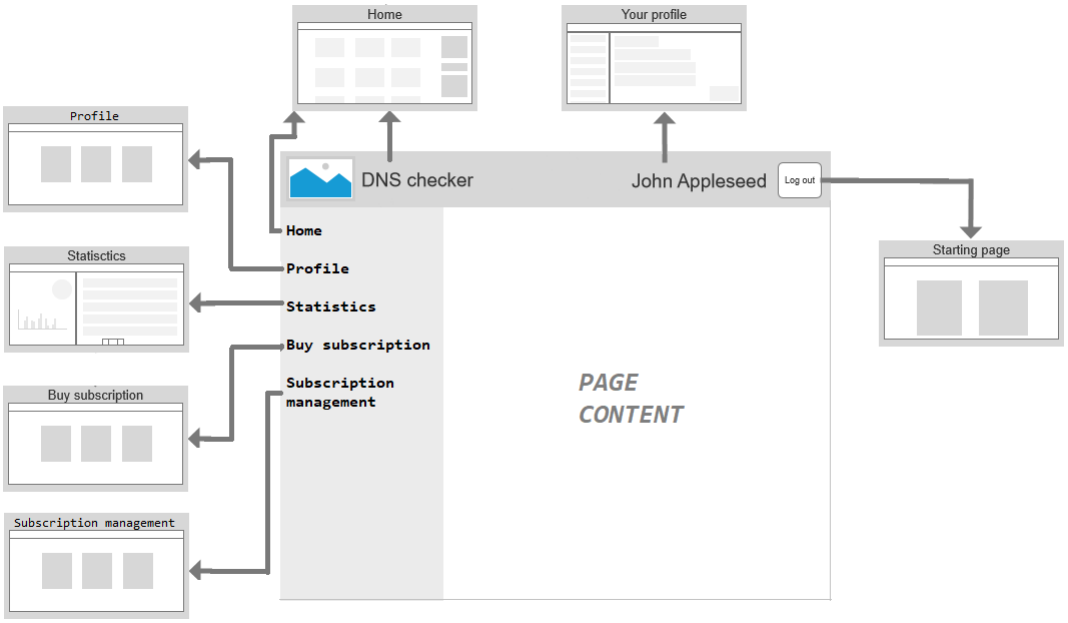
- strona główna - będzie zawierała najważniejsze informacje dotyczące naszego systemu,
- strona logowania - strona umożliwiająca zalogowanie się użytkownika i przekierowująca go do jego strony,
- strona rejestracji - strona umożliwiająca zarejestrowanie się nowego użytkownika



4.2. Strona użytkownika zalogowanego

Strona dostępna po zalogowaniu się użytkownika. nawigacja będzie odbywać się za pomocą paska górnego oraz paska bocznego. Dostępnymi komponentami będą:

- strona główna - będzie zawierała najważniejsze informacje dotyczące aktywnych, wykupionych przez niego subskrypcji, w tym datę ostatniego sprawdzenia i rezultat,
- profil użytkownika - komponent, w którym będą zebrane informacje o koncie zalogowanego użytkownika,
- zakup subskrypcji - miejsce, w którym użytkownik będzie mógł nabyć nową subskrypcję,
- statystki - komponent, gdzie będzie zbiorcze zestawienie danych dotyczących wszystkich subskrypcji użytkownika,


Home
Your profile
Profile
DNS checker
John Appleseed
Log out
Starting page
Statisctics
Home
Profile
Statistics
Buy subscription
Subscription management
PAGE CONTENT
Buy subscription
Subscription management