”Automatyczna analiza rynku pracy.”

Bartłomiej Sielicki
Łukasz Skarżyński

Praca inżynierska
wersja 0.1

Promotor:
Barbara Rychalska

Wydział Matematyki i Nauk Informacyjnych
Politechniki Warszawskiej
23.11.2017

Streszczenie

Automatyczna Analiza Rynku Pracy IT

Rynek pracy w branży IT jest bardzo dynamiczny. Szczególnie jest to zauważalne w ostatnich latach, kiedy to nieustannie zauważamy wzajemne wypieranie się technologii. Na zmiany reagują wszyscy - począwszy od firm, które aby dorównać konkurencji stale wdrażają nowe rozwiązania, przez pracowników, chcących nieustannie poszerzać własne kompetencje, po uczelnie wyższe, które starają się przekazać swoim przyszłym absolwentom jak najbardziej aktualną wiedzę i umiejętności. Istotna staje się zatem możliwość wglądu w trendy i oparta na niej analiza zachodzących zmian.
Aplikacja której poświęcona jest ta praca jest próbą udowodnienia, że proces taki da się zautomatyzować i bez wymogu nadmiernej ingerencji użytkownika zbierać, analizować, wizualizować oraz udostępniać najbardziej istotne (z punktu widzenia autorów) informacje. W ramach pracy wykonano system informatyczny składający się z dwóch komponentów. Pierwszym z nich jest zaplecze analityczne (*back end*), zajmujące się automatycznym pobieraniem zamieszczanych na jednym z największych polskich serwisów rekrutacyjnych (Pracuj.pl) ofert z branży IT oraz ich późniejszą analizą pod kątem umieszczanych w nich technologii. Drugim wykonanym komponentem jest aplikacja WWW (*front end*)prezentująca wyniki użytkownikowi oraz umożliwiająca eksport zebranych danych do samodzielnej analizy.

Słowa kluczowe: *Rynek pracy IT*, *Automatyzacja*, *Web Scraping*

Abstract

Automated Job Market Analysis

Streszczenie po angielsku.

Spis treści

Wstęp

Słownik pojęć

- **back-end** - Odpowiada za operacje w tle, których przebiegu użytkownik nie widzi bezpośrednio. Zajmuje się przetwarzaniem, wykonywaniem zadań na podstawie otrzymanych danych. W projekcie słowem back-end określamy wszystkie moduły za wyjątkiem aplikacji internetowej z której korzysta użytkownik.
- **front-end** - Warstwa wizualna systemu - interfejs użytkownika. Głównym jego zadaniem jest pobieranie danych od użytkownika oraz przekazywanie ich do back-endu oraz ewentualne pokazanie odebranej odpowiedzi. W systemie którego dotyczy dana dokumentacja front-end jest stroną internetową.
- **web scraper** - program, którego głównym zadaniem jest zbierać określone dane ze stron internetowych. Gdy w dokumentacji używamy słowa "scraper" zawsze odnosi się ono właśnie do "web scraper'a".
- **pipeline** - łańcuch przetwarzania danych (funkcji), w którym wejściem kolejnego etapu jest wyjście poprzedniego.
- **framework** - szkielet do budowy różnego rodzaju aplikacji. Definiuje on strukturę aplikacji oraz ogólny mechanizm jej działania, oraz dostarcza zestaw komponentów i bibliotek ogólnego przeznaczenia do wykonywania określonych zadań.
- **API** - (ang. Application Programming Interface) interfejs programowania aplikacji, jest to ściśle określony zestaw reguł i ich opisów, w jaki programy komputerowe komunikują się między sobą. Najczęściej używamy tego słowa w odniesieniu do WEB API - interfejsu komunikacji korzystającego z HTTP oraz formatu JSON do przesyłania danych.

Rozdział 1

Specyfikacja projektu

1.1 Opis biznesowy

Głównym zadaniem aplikacji jest automatyczna analiza rynku pracy. System zajmuje się agregacją oraz przetwarzaniem ofert zebranych z serwisu zewnętrznego (Pracuj.pl) i przedstawianiem wyników użytkownikowi.

W szczególności proces działania systemu sprowadza się do:

1. Pobierania ofert pracy z serwisu zewnętrznego biorąc pod uwagę jedynie kluczowe elementy, takie jak: tytuł ogłoszenia, opis, datę dodania, itp.
2. Zapisania tak zebranych ogłoszeń do bazy danych
3. Przeprowadzenia analizy na treści ogłoszenia uzyskując listę technologii wymienionych w treści ogłoszenia, których znajomość jest oczekiwana od pracownika.
4. Udostępnienia użytkownikowi interfejsu do wyszukiwania zebranych w systemie ofert oraz wyświetlania statystyk dla wybranych technologii w postaci strony WWW.

Wymienione działania realizują odrębne komponenty systemu.

Użytkownik bezpośrednio korzystać powinien tylko z jednego modułu - wspomnianej wyżej aplikacji WWW.

Przewidzianymi grupami docelowymi użytkowników aplikacji są:

- studenci i osoby szukające pracy - dzięki oferowanym przez aplikację danym, będą mogli ocenić znajomość których technologii staje się pożądana na rynku pracy
- pracodawcy - na podstawie trendów wśród używanych technologii będą mogli podjąć decyzje dotyczące przyszłych projektów
- pozostali użytkownicy zainteresowani zmianami na rynku pracy - możliwość eksportu zgromadzonych przez system danych pozwala na przeprowadzenie samodzielnej analizy

1.2 Wymagania funkcjonalne

Podstawą interakcji użytkownika z systemem jest strona WWW - użytkownik powinien być w stanie otworzyć ją na dowolnym komputerze z dostępem do internetu, wyposażonym w przeglądarkę:

- Google Chrome w wersji 49 lub wyższej
- Mozilla Firefox w wersji 52 lub wyższej
- Safari w wersji 10.1 lub wyższej

Posiadanie przeglądarki innej niż wymienione lub w starszej wersji nie oznacza że strona nie będzie działać, jednak nie da się zagwarantować że będzie to działanie w pełni poprawne.

Na stronie nie przewidziano kont użytkowników, nawet administracyjnego. Każdy z odwiedzających ma dostęp do tych samych danych oraz takie same możliwości.

Funkcjonalność aplikacji WWW rozbita jest na trzy podstrony:

- statystyki technologii
- wyszukiwanie ofert
- informacje o systemie

Możliwości użytkownika w obrębie każdej z sekcji prezentuje załączony diagram przypadków użycia (Rys. 1.1).

1.2.1 Wyświetlenie statystyk technologii

Jest to pierwsza podstawowa funkcjonalność strony. Pozwala ona użytkownikowi na wybór jednej bądź kilku (przy pomocy auto-uzupełniania) technologii oraz wyświetlenie:

- w przypadku wybrania więcej niż jednej technologii, wykresu porównującego z osobna ich popularność (pod względem ilości ofert)
- wykresu ilości ofert zawierających każdą z wybranych technologii
- wykresu procentowej ilości ofert z systemu zawierających te technologie
- wykresu kołowego przedstawiającego najczęściej poszukujących wybranych technologii pracodawców
- listy dziesięciu najbardziej zbliżonych technologii

1.2.2 Wyszukiwanie ofert wg technologii

Kolejną oferowaną użytkownikom możliwością jest wyszukiwanie ofert zebranych i umieszczonych w bazie. Rolę kryterium wyszukiwania, podobnie jak podczas przeglądania statystyk, pełni jedna lub więcej wybranych technologii. Rezultatem jest lista ogłoszeń oraz przycisk oferujący możliwość ich eksportu w formacie JSON.

Każdą z ofert widocznych na liście można rozwinąć, uzyskując dostęp do następujących informacji:

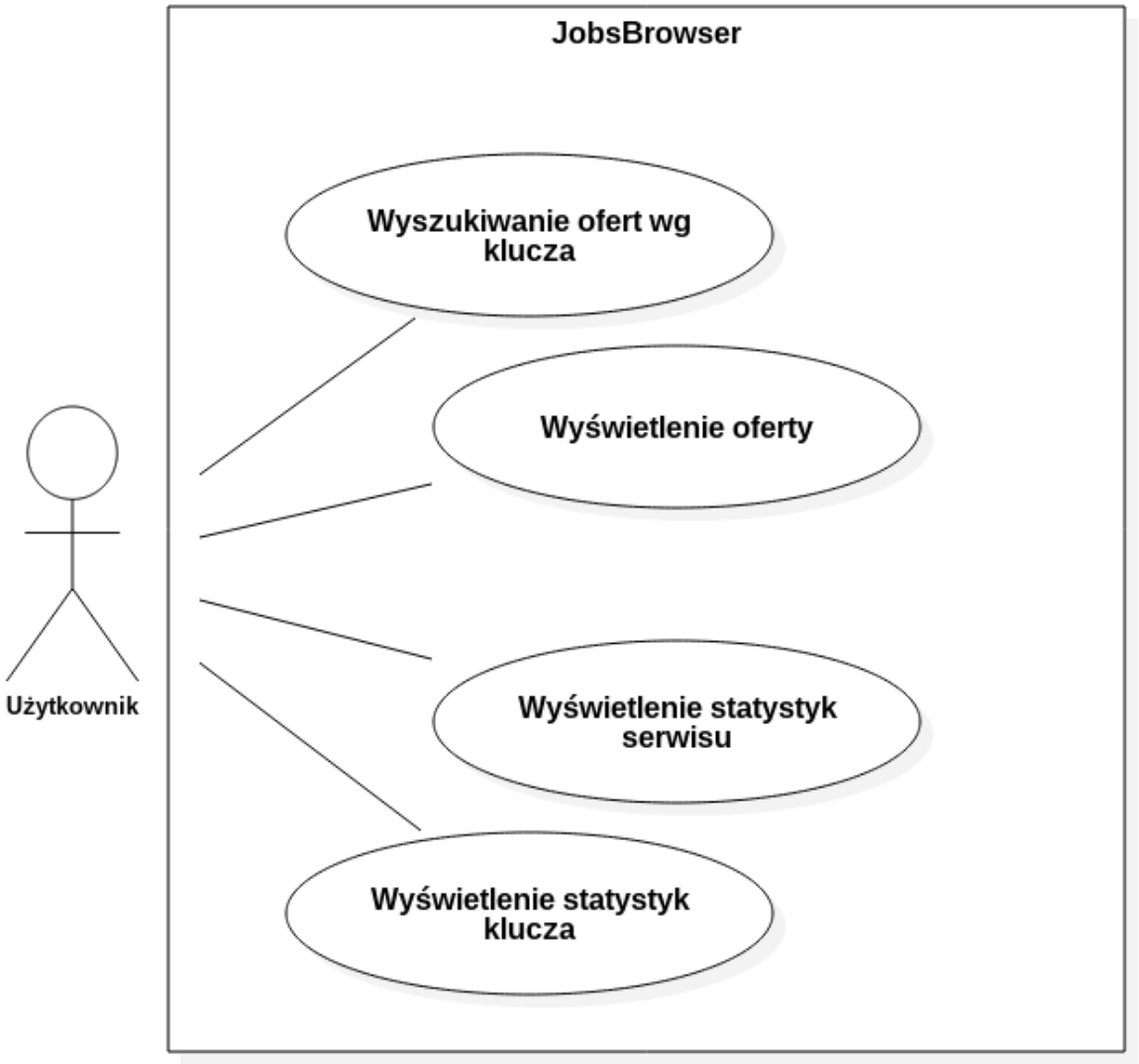
- tytuł oferty

- data dodania i wygaśnięcia oferty w macierzystym serwisie
- nazwa pracodawcy i miejsce pracy
- wszystkie wykryte w treści oferty technologie
- odnośnik do oryginalnego ogłoszenia w macierzystym serwisie

1.2.3 Wyświetlenie informacji o systemie

W osobnej sekcji użytkownik ma dostęp do wyświetlenia zbiorczych statystyk dotyczących serwisu i dostępnych w nim danych. Zbiór dostępnych statystyk sprowadza się do:

- liczby wszystkich ofert w bazie systemu
- wykresu powyższej wartości względem czasu
- daty ostatniego zebrania danych z serwisu macierzystego



Rysunek 1.1: Przypadki użycia.

1.3 Wymaganie niefunkcjonalne

Tabela 1.1 przedstawia wymagania niefunkcjonalne postawione stworzonej aplikacji.

Tabela 1.1: Wymagania niefunkcjonalne

Obszar wymagań	Opis
Używalność (Usability)	Wymagany dostęp do internetu w celu skorzystania z aplikacji. Aplikacja WWW jest intuicyjna w obsłudze dla użytkownika. Aplikacja WWW jest dostępna dla użytkownika w każdym wybranym dla niego momencie.
Niezawodność (Reliability)	Dostęp do aplikacji WWW powinien być możliwy przez 24 godziny, 7 dni w tygodniu. Za wyjątkiem prac serwisowych nie dłuższych niż 2 h w tygodniu przy założeniu stabilnego połączenia internetowego.
Wydajność (Performance)	Aplikacja WWW powinna działać płynnie na każdym komputerze z dowolnym systemem operacyjnym wyposażonym w odpowiednią przeglądarkę.
Wsparcie	Dane z których korzysta serwis aktualizowane są na bieżąco. W stopce strony umieszczone są adresy e-mailowe autorów aplikacji, oferujących pomoc w przyadku problemów technicznych.

Rozdział 2

Architektura systemu

2.1 Schemat

Przewidziana architektura ma strukturę modułową. Ogólny schemat komponentów oraz ich połączenie przedstawia załączony diagram (Rys. 2.1), natomiast bardziej szczegółowy opis każdego z nich znajduje się w dalszej części tego rozdziału.

W systemie wyróżnić możemy trzy główne, niezależne od siebie (na tyle że mogą, a nawet powinny, być uruchamiane na różnych maszynach) moduły.

Robot zbierający dane

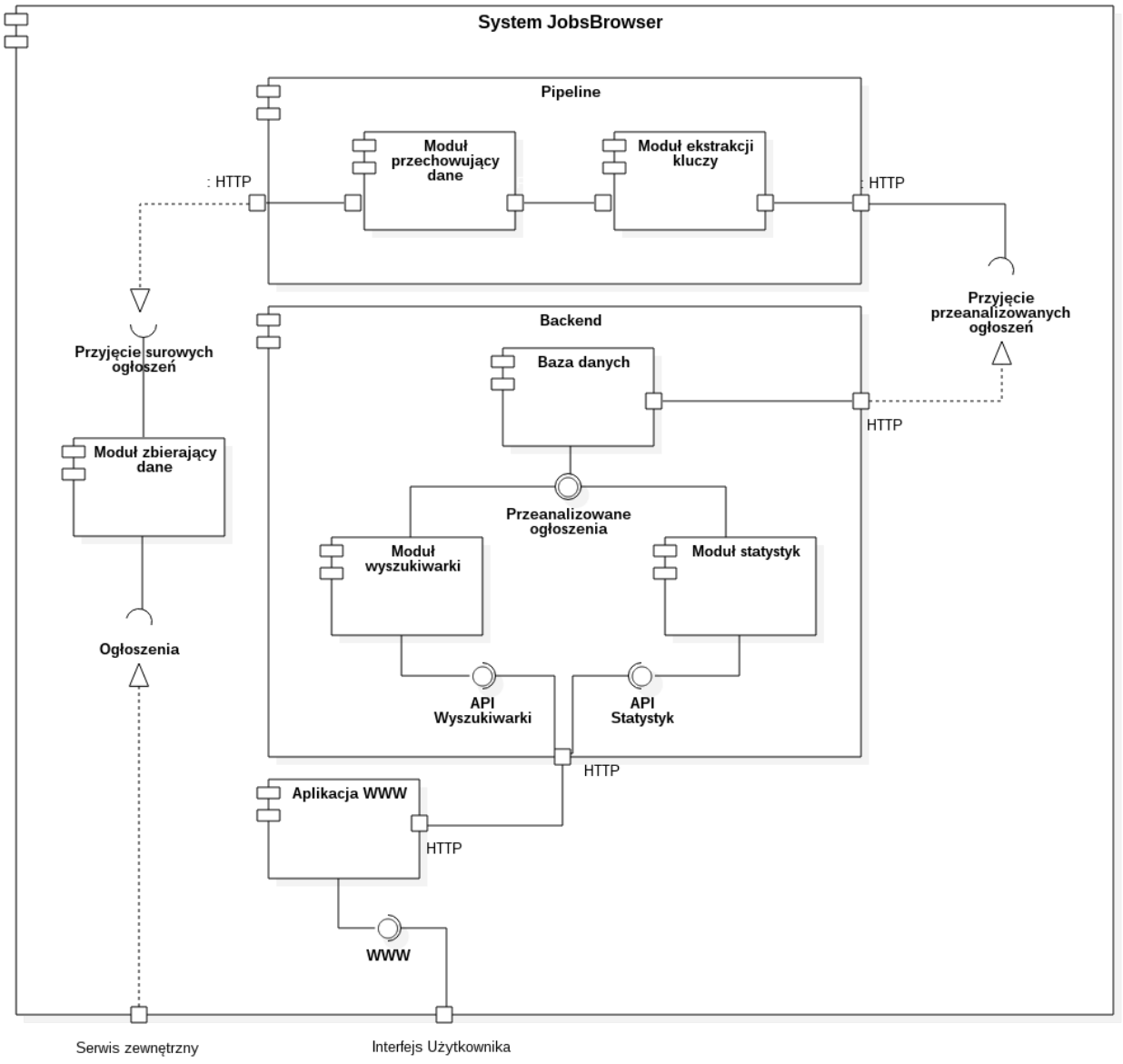
Komponent ten, nazywany też *scraperem*, odpowiada za automatyczne pobieranie ogłoszeń. Jest to program, który cyklicznie łączy się z udostępniającym ogłoszenia serwisem i automatycznie pobiera ich treść. Zebrane oferty przesyła do kolejnego komponentu systemu.

Moduł analityczny

Stosowaną zamiennie nazwą jest *Pipeline* - czyli *łańcuch przetwarzania*. Odbiera on zebrane ogłoszenia, a następnie wykonuje na nich sekwencję operacji obejmujących zapis do bazy danych czy wykrycie zawartych w treści technologii. Komponent ten jest również odpowiedzialny za komunikację z aplikacją WWW, generując wyświetlane przez nią dane.

Aplikacja WWW

Ostatnim elementem systemu, jedynym dostępnym bezpośrednio dla użytkownika jest aplikacja WWW. Nie posiada ona własnej bazy, a co za idzie kont użytkowników. Komunikuje się z modułem analitycznym, pozwalając użytkownikowi na przejrzysty i wygodny dostęp do informacji.



Rysunek 2.1: Diagram komponentów.

2.2 Robot zbierający dane

Kod źródłowy znajduje się pod adresem:

- github.com/jobsbrowser/scraper.

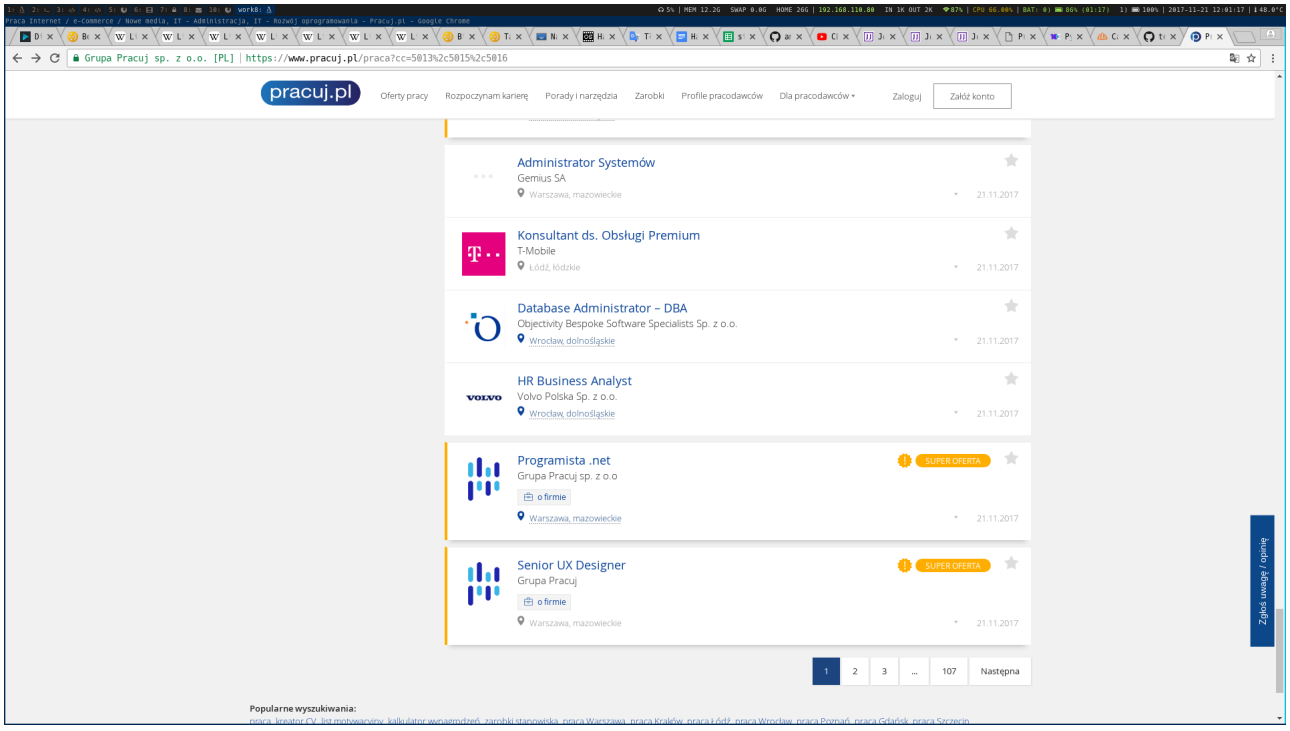
Komponent zajmuje się automatycznym pobieraniem ofert z serwisu zewnętrznego, ekstrakcją podstawowych informacji oraz przesłaniem tak przetworzonych ofert do kolejnego modułu. Zdecydowaliśmy się na serwis Pracuj.pl jako źródło ofert oraz na framework **Scrapy**(Scrapy n.d.) jako bazę naszego modułu.

2.2.1 Serwis zewnętrzny

Pracuj.pl dzieli zamieszczone w nim oferty na kategorie. My, z racji tematyki pracy skupiamy się wyłącznie na trzech z nich:

- Internet / e-Commerce / Nowe media
  - E-marketing / SEM / SEO
  - Media społecznościowe
  - Projektowanie
  - Sprzedaż / e-Commerce
  - Tworzenie stron WWW / Technologie internetowe
- IT - Administracja
  - Administrowanie bazami danych i storage
  - Administrowanie sieciami
  - Administrowanie systemami
  - Bezpieczeństwo / Audyt
  - Wdrożenia ERP
  - Wsparcie techniczne / Helpdesk
  - Zarządzanie usługami
- IT - Rozwój oprogramowania
  - Analiza biznesowa
  - Architektura
  - Programowanie
  - Testowanie
  - Zarządzanie projektem

Widok listy ogłoszeń w serwisie umożliwia wybór kategorii, z których ogłoszenia chcemy zobaczyć. Skorzystaliśmy z tej możliwości, aby uzyskać bazowy link od którego zaczniemy pobieranie ofert.



Rysunek 2.2: Widok paginacji ogłoszeń w witrynie pracuj.pl.

Pracuj.pl przy zbiorczym wyświetlaniu ofert używa paginacji. Oznacza to, że scraper musi poradzić sobie nie tylko z pobieraniem podstron poszczególnych ofert, ale też z poruszaniem się pomiędzy ponumerowanymi stronami listy.

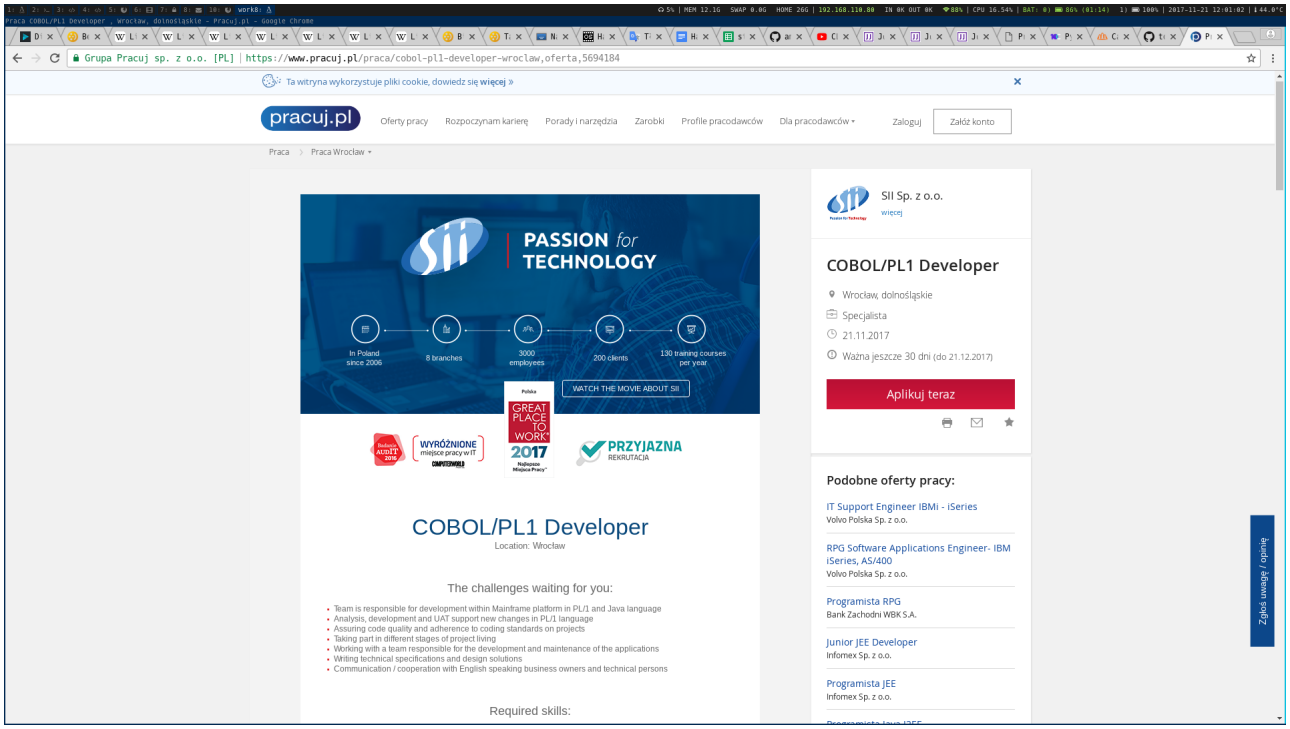
Do każdej z ofert na stronie (których znajduje się ok. 50) prowadzi bezpośredni link, który można wydobyć z kodu HTML listy. Podstrona pojedynczej oferty zawiera wszystkie interesujące nas informacje również możliwe do uzyskania z kodu HTML przy użyciu odpowiednich selektorów CSS. Dostępne w przystępny sposób informacje to:

- Data dodania oferty
- Data wygaśnięcia
- Nazwa dodającego (pracodawca)
- Lokalizacja (miasto i województwo)
- Tytuł oferty
- Treść oferty
- Kategorie w których znajduje się dana oferta

Ponadto, w łatwy sposób można uzyskać również dwie wartości jednoznacznie identyfikujące ofertę:

- Adres URL oferty
- ID (będące częścią adresu URL)

Te dwie wartości z powodzeniem mogą służyć za klucz pozwalający np. szybko sprawdzać czy oferta jest już w bazie systemu - czyli czy została już kiedyś przetworzona.



Rysunek 2.3: Widok oferty w witrynie pracuj.pl.

2.2.2 Napotkane problemy

Zadaniem stojącym przed scraperem jest automatyczne przejście po wszystkich dostępnych stronach listy, wydobycie z nich adresów poszczególnych ofert, a stamtąd wszystkich potrzebnych informacji. Pobrane ogłoszenie z wydzielonymi fragmentami powinno trafić do innego komponentu systemu, który zajmie się jego zapisem czy dalszym przetwarzaniem. Podczas prac nad modułem wynikło kilka kwestii które wymagały opracowania konkretnego rozwiązania.

Aktualizacja ofert w serwisie

Ofert na stronie wraz z upływem czasu będzie przybywać - dla działającego systemu jest to bardzo istotne. Aby zapewnić stały przypływ ogłoszeń z serwisu Pracuj.pl scraper został tak przygotowany, aby mógł być uruchamiany cyklicznie. Przewidzianym interwałem są 24 godziny, choć nie jest to wartość na stałe zdefiniowana w programie. To od uruchamiającego system zależy jak ją ustawi.

Utrzymywanie stanu scrapera

Uruchomienie cykliczne wraz z brakiem stanu (bo przecież wszystkie przetworzone ogłoszenia trafiają do osobnego modułu) wiąże się z możliwością niechcianego przetwarzania jednej oferty wielokrotnie - przy każdym kolejnym uruchomieniu programu. Rozwiązaniem okazało się zaimplementowanie w scraperze możliwości pobrania z modułu do którego trafią dane kluczy (adresów URL) tych danych które już tam są. Oznacza to, że scraper przy każdym uruchomieniu przetworzy wszystkie dostępne strony, ale nie będzie pobierał ani przetwarzał podstron ofert które już ma na liście. Próba pobrania listy wykonywana jest po uruchomieniu programu, przed rozpoczęciem skanowania. Jeżeli nie uda się jej otrzymać (serwer nie odpowie, lub odpowie z błędem), program wypisze w konsoli stosowny komunikat ostrzegający i przejdzie do przetwarzania wszystkich ofert.

Struktura danych

Początkowo dane pobierane były z wersji portalu Pracuj.pl przeznaczonej dla urządzeń o wyższej rozdzielczości ekranu (Laptopy, komputery PC). Problemem okazała się jednak niejednolita struktura danych, utrudniająca automatyczne wydobywanie informacji z treści ogłoszenia. Rozwiązaniem okazało się korzystanie z wersji portalu przeznaczonej dla urządzeń mobilnych.

Zabezpieczenia serwisu przed zbieraniem danych

Wartym wspomnienia jest fakt stosowania przez serwis Pracuj.pl zabezpieczeń utrudniających automatyczne zbieranie danych. Podczas pierwszych testów modułu wszystkie wychodzące żądania HTTP miały nagłówek `User-Agent` ustawiony na nazwę projektu - `jobsbrowser`. Nie zauważono wtedy żadnych trudności związanych z uzyskaniem przez scrapera dostępu do zasobów serwisu. Po kilku dniach jednak, sytuacja się zmieniła. Okazało się, że wszystkie żądania

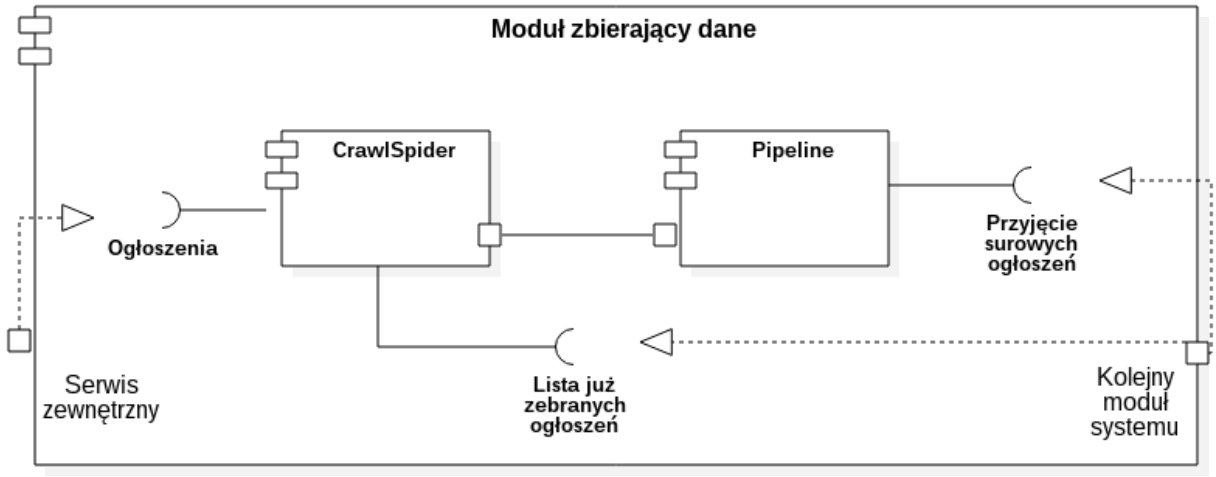
wychodzące z systemu (nawet z innych adresów IP) podpisane jako `jobsbrowser` były odrzucane przez serwer. Konieczne było wprowadzenie poprawki w kodzie modułu, tak żeby nagłówkiem `User-Agent` imitował przeglądarkę internetową. Wybór padł na Google Chrome w wersji 41 na urządzenia mobilne, i to rozwiązało problem.

2.2.3 Wykorzystane technologie

Scrapy

Licencja: BSD3

Do napisania scrapera zdecydowaliśmy się na framework Scrapy(Scrapy n.d.). Jest to framework napisany w języku Python, przy wykorzystaniu biblioteki Twisted. Dzięki temu działa całkowicie asynchronicznie, co czyni go wydajnym przy relatywnej łatwości pisania własnych scraperów.



Rysunek 2.4: Schemat architektury modułu Scraper'a.

2.3 Moduł analityczny

Kod źródłowy znajduje się w repozytoriach:

- github.com/jobsbrowser/pipeline
- github.com/jobsbrowser/backend

Modułem systemu do którego trafiają przetwarzane oferty w następnej kolejności jest moduł analityczny. To tutaj odbywa się, kluczowy z punktu widzenia biznesowego zastosowania projektu, proces ekstrakcji ze zbieranych ofert wartościowych informacji. Wejściem modułu są pobierane z serwisu Pracuj.pl oferty, natomiast rolę wyjścia pełni interfejs HTTP z którego korzysta aplikacja WWW.

Moduł ten rozdzielony jest na dwa komponenty - pierwszym z nich jest usługa obsługująca robota zbierającego dane. Przyjmuje ona pobrane oferty, roboczo zapisuje je w nierelacyjnej bazie danych, a następnie dla każdej z nowo dodanych rozpoczyna sekwencyjny proces analizy. Przetworzone ogłoszenia trafiają w ostatnim kroku tego procesu do drugiego komponentu, którym jest usługa obsługująca aplikację WWW. W odrębnej, już relacyjnej, bazie danych zapisuje ona najważniejsze informacje o przetworzonych ofertach oraz, co najważniejsze, listę technologii które udało się w nich wykryć.

2.3.1 Przeznaczenie modułu

Dzięki robotowi zbierającemu dane do systemu trafiają pojawiające się w serwisie Pracuj.pl oferty. Dla przypomnienia, informacje jakie uzyskiwane są bezpośrednio w trakcie ich pozyskiwania to:

- tytuł
- ID
- treść (w postaci kodu HTML)

oraz kilka mniej znaczących z punktu widzenia tego komponentu, a szerzej opisanych w części dotyczącej robota zbierającego dane. Informacje te same w sobie nie są szczególnie istotne dla użytkownika końcowego aplikacji. Są to bowiem te same dane które zobaczyć możemy korzystając bezpośrednio z serwisu pracuj.pl. Nie niosą więc za sobą póki co żadnych dodatkowych wartości.
Sednem aplikacji i systemu samego w sobie, jest przekształcenie dużego zbioru ofert w statystyki dotyczące ich wspólnych elementów - czyli technologii które są w nich wymieniane. Moduł

ten pozwala właśnie na to - uzyskanie z pojedynczej oferty listy nazw technologii, które są zawarte w jej treści.

2.3.2 Obsługa robota zbierającego dane

Komunikacja między modułem analitycznym a robotem zbierającym dane odbywa się za pośrednictwem protokołu HTTP. Interfejs udostępniany robotowi pozwala na następujące operacje:

- Dodanie oferty do bazy (nadpisując jeśli oferta z takim adresem URL już istnieje)
- Pobranie listy adresów URL ofert zapisanych już w bazie, tak aby uniknąć przetwarzania ich ponownie
- Uaktualnienie wybranej oferty z bazy danych

Wymagania niefunkcjonalne stawiane tej części systemu sprowadzają się zatem do:

- **niezawodności** - usługa powinna być dostępna możliwie cały czas. Nie jest to jednak kwestia kluczowa, ponieważ stosunkowo krótkie braki w dostępności (rzędu maksymalnie kilku dni) nie ciągną za sobą konsekwencji. Jeżeli robot zbierający dane nie uzyska odpowiedzi od modułu zajmującego się ich przechowywaniem i analizą, informacje o tej ofercie nie zostaną nigdzie zapisane. Kiedy usługa przechowywania będzie ponownie dostępna, na zapytanie robota o listę ofert będących już w bazie zwróci tę sprzed awarii, wszystkie pominięte oferty zostaną więc ostatecznie dodane.
- **wydajności** - liczba nadchodzących ofert może być potencjalnie duża, proces zapisu do bazy powinien być więc jak najmniej skomplikowany i efektywny, aby uniknąć spadków na wydajności z powodu niewydajnych zapytań. Podobnie jak w kwestii niezawodności, nie jest to jednak wymaganie kluczowe. Spadki w wydajności nie będą bowiem objawiać się wolniejszym działaniem aplikacji przeznaczonej dla użytkownika końcowego, a jedynie późniejszym pojawianiem się w niej nowych ofert.

Baza danych

Wykorzystanym silnikiem bazy danych do roboczego zapisu ofert jest MongoDB(MongoDB n.d.). Wybór padł na bazę nierelacyjną, ponieważ jedynym typem trzymanych w niej danych są same oferty. Oznacza to że w bazie relacyjnej znajdowałaby się tylko jedną tabela - bez żadnych relacji czy potrzeby zachowania spójności. Nie ma potrzeby również stosowania mechanizmu transakcji, czy skomplikowanych zapytań. Tym co faktycznie jest oczekiwane od bazy jest wydajność, dostępność oraz ewentualna skalowalność.

Struktura dokumentów przechowywanych w bazie jest identyczna jak struktura zebranego ogłoszenia. Dla przypomnienia, pola wyróżniane w dokumencie oferty to:

- Adres URL
- Czas w którym pobrano ofertę
- Kod HTML strony z ofertą
- ID oferty w systemie pracuj.pl
- Data dodania
- Data ważności
- Podmiot dodający
- Tytuł oferty
- Miejsce pracy
- Kategorie oferty
- Kod HTML treści oferty (rozbity na opis, kwalifikacje oraz benefity - wg struktury Pracuj.pl)

Interfejs HTTP

Interfejs HTTP zaimplementowany jest przy użyciu mikro-frameworka Flask(Flask n.d.). Przydatne okazało się dostępne rozszerzenie integrujące go z bazą MongoDB, użytą w module.

2.3.3 Ekstrakcja technologii

Rozpoznawanie technologii

Uznaliśmy, że aby w miarę poprawnie rozpoznawać klucze, powinniśmy w pierwszym kroku zaopatrzyć się w miarę obszerny i sprawdzony ich zbiór.
Podejście takie, jak później się okazało wydaje się być całkiem słuszne, ponieważ stosowane jest także w projektach o znacznie szerszym zasięgu i złożoności niż nasz. Dla przykładu, powstająca w momencie pisania tej pracy platforma **Google Cloud Job Discovery**, zajmująca się automatycznym dopasowywaniem ofert pracy do CV potencjalnych pracowników, do działania wykorzystuje zbudowaną przez zespół Google ontologię zawierającą, jak podają, ok. 50 tys. umiejętności z różnych pól zawodowych.

Z racji tego, że nasz projekt skupia się na ofertach pracy z branży IT, postanowiliśmy skorzystać z faktu że istotne, oczekiwane przez pracodawców umiejętności, pokrywają się z nazwami technologii informatycznych, czy języków programowania. Jako źródło takich danych, postanowiliśmy wykorzystać popularny wśród ludzi zainteresowanych IT portal **Stack Overflow**. API tego serwisu pozwala na pobranie używanych przez jego użytkowników tagów, posortowanych

według popularności. Wszystkich jest blisko 40 tys. W zdecydowanej większości odpowiadają one nazwom technologii.

Na potrzeby naszego projektu, korzystamy ze zbioru dwóch tysięcy najpopularniejszych tagów. Znajdują się wśród nich wszelakie technologie, frameworki, wzorce projektowe i języki programowania. Mając taką listę, możemy każdą przychodzącą ofertę przeszukać pod względem występowania niektórych z nich. Proces ten przeprowadzamy w następując sposób:

1. Po zapisaniu całej oferty do bazy, upraszczamy obiekt przekazując dalej tylko istotne z punktu widzenia komponentu dane, tj.

- ID
- tytuł
- treść

2. Z treści oferty usuwamy tagi HTML
3. Treść oferty rozbijamy na *tokeny*, czyli wyrazy oraz znaki interpunkcyjne
4. Sprawdzamy czy oferta jest w języku polskim, bo tylko takie akceptujemy
5. Usuwamy zbędne z punktu widzenia analizy językowej tokeny, takie jak przyimki, spójniki czy zaimki.
6. Dla listy uzyskanych tokenów sprawdzamy które z nich znajdują się na liście znanych nam kluczy, i te zapisujemy.

W ten sposób każdej ofercie z osobna przypisujemy listę technologii które znaleźliśmy w jej treści.

Architektura

Z technicznego punktu widzenia proces ten jest ciągiem funkcji, wykonywanych w ustalonej kolejności, gdzie wyjście każdej z nich przekazywane jest jako argument do następnej. Stąd powtarzający się często termin *pipeline*. Argumentem pierwszej funkcji jest zapisana do bazy danych chwilę po otrzymaniu od robota oferta, natomiast wynikiem ostatniej uproszczony obiekt oferty zawierający listę znalezionych technologii. Na koniec oferta jest wyszukiwana w bazie, dodawana jest do niej wspomniana lista (dzięki zastosowaniu nierelacyjnej bazy danych nie wiąże się to z potrzebą jej przebudowy), a następnie wysyłana jest do usługi generującej statystyki.

Do obsługi łańcucha, czyli utrzymania poprawnej kolejności wykonywania funkcji, oraz asynchronicznego przetwarzania wielu ofert jednocześnie korzystamy z frameworka **Celery**(Celery n.d.), którego opis znajduje się w dalszej części tej sekcji.

Znajdowanie podobnych technologii za pomocą modelu Word2Vec

Jedną z funkcjonalności systemu jest generowanie technologii zbliżonych do tych wybranych przez użytkownika. W celu znalezienia takiego zbioru postanowiliśmy skorzystać z modelu Word2Vec (Yoav Goldberg n.d.). Word2vec to grupa modeli reprezentujących słowa jako tzw. zanurzenia słów (Bengio Yoshua n.d.), czyli słowa reprezentowane jako wektory w wielowymiarowej przestrzeni. W naszym przypadku zdecydowaliśmy się wytrenować model word2vec nie na całych korpusach (wszystkich ogłoszeniach), ale na wykorzystywanej przez nas liście technologii. Dzięki takiemu podejściu wynikowy zbiór zawierać będzie tylko pozostałe elementy z tej listy. Do wytrenowania modelu skorzystaliśmy z biblioteki gensim (Gensim n.d.). Poniżej znajdują się wyniki otrzymane dla technologii `JAVA`:

- java-ee
- maven
- junit
- hibernate
- groovy
- soa
- tomcat
- jsp
- jenkins
- eclipse

Jak widać wyniki są zadowalające, wszystkie klucze znajdujące się na powyższej liście mają wiele wspólnego z językiem programowania JAVA. Jednak z powodu stosunkowo małego zbioru treningowego rezultaty nie są tak dobre dla mniej popularnych technologii, mających mało wystąpień. Dla przykładu - `Vue.JS`:

- node.js
- elasticsearch
- jasmine
- testy jednostkowe
- bitbucket
- api
- nosql
- webpack

- mongodb
- user-interface

2.3.4 Próby analizy tekstu i uzasadnienie wyboru algorytmu

Dużo czasu poświęciliśmy na testowanie wielu algorytmów do wyznaczania słów kluczowych z tekstu. Wiele z nich dawało niezadowalające wyniki na zbiorach ogłoszeń napisanych w języku polskim. Poniżej opisujemy krótko algorytmy które przetestowaliśmy wraz z przykładowymi wynikami jakie dawały. Wszystkie przedstawione poniżej wyniki zostały uzyskane z przykładowej oferty (Rys. 2.5).

Wystarczy, że:

- posiadasz doświadczenie w podobnej roli
- znasz na poziomie przynajmniej dobrym język Python (idealnie Python 3) wraz z frameworkiem Django
- znasz również JavaScript (nie tylko jQuery)
- posiadasz dobrą wiedzę na temat relacyjnych baz danych (mile widziana znajomość PostgreSQL)
- sprawnie poruszasz się w HTML5 i CSS3
- cechuje Cię ponadprzeciętna samodzielność oraz umiejętność samoorganizacji
- potrafisz pisać wysokiej jakości kod
- posiadasz umiejętność dekompozycji zadań oraz dostarczania działających rozwiązań
- jesteś skrupulatny (lub przynajmniej pracujesz nad tym, aby takim się stać)

to z dużym prawdopodobieństwem jesteś osobą, której szukamy.

Jeżeli dodatkowo:

- potrafisz pisać testy jednostkowe
- znasz dowolny, inny niż wyżej wymienione, język programowania
- możesz wykazać się znajomością protokołów sieciowych

to z tym większą niecierpliwością oczekujemy na możliwość spotkania z Tobą.

Chcemy powierzyć Ci następujące obowiązki:

- projektowanie oraz implementowanie aplikacji internetowych, w tym wspierających płatności, uwierzytelnianie SMS, jak również aplikacji wyszukujących informacje na wielu stronach
- implementowanie nowych funkcji i poprawek w istniejących aplikacjach internetowych (utrzymanie i rozwój narzędzi stworzonych na potrzeby wewnętrzne)
- ścisłą współpracę z doświadczonymi architektami oprogramowania, wraz z wpływem na kształt oraz sposób działania tworzonych narzędzi

Dodatkowo oferujemy:

- kontakt z najnowszymi technologiami oraz niespotykaną różnorodność tematyczną projektów
- możliwość realizowania własnych, innowacyjnych projektów
- elastyczne godziny pracy
- współpracę z ekspertami z wieloletnim doświadczeniem, którzy chętnie dzielą się swoją wiedzą
- szkolenia wewnętrzne, udział w spotkaniach, seminariach oraz konferencjach
- bardzo dobre warunki pracy, nowoczesny sprzęt, świeże owoce i pyszną kawę
- dostęp do dodatkowych świadczeń (Multisport, Medicover)
- swobodną atmosferę, brak dres code'u, nieformalne relacje
- mnóstwo ciekawych wyzwań i szans rozwoju
- nowoczesne, doskonale skomunikowane biuro w centrum Warszawy

Rysunek 2.5: Oferta wykorzystywana do uzyskania przykładowego wyniku

RAKE(Rapid Automatic Keyword Extraction)(Rake-NLTK n.d.)

Popularny, stosunkowo prosty algorytm do automatycznej ekstrakcji słów kluczowych. Niestety nie przyniósł zadowalających efektów. Poniżej lista zawiera 10 najważniejszych według algorytmu RAKE fraz/kluczy wykrytych w przykładowej ofercie:

- umiejętność samoorganizacji potrafisz pisać wysokiej jakości kod posiadasz umiejętność dekompozycji zadań
- potrafisz pisać testy jednostkowe znasz dowolny
- dostarczania działających rozwiązań jesteś skrupulatny
- poziomie przynajmniej dobrym język python
- język programowania możesz wykazać
- dużym prawdopodobieństwem jesteś osobą
- temat relacyjnych baz danych
- mile widziana znajomość postgresql
- znajomością protokołów sieciowych to
- posiadasz dobrą wiedzę

TF-IDF(Term Frequency - Inverse Document Frequency)(TF-IDF n.d.)

Jeden z bardziej popularnych i szeroko wykorzystywanych algorytmów w świecie przetwarzania języka naturalnego. Niestety również nie dawał zadowalających wyników. Wykryte słowa kluczowe dla tej samej oferty:

- pisać
- implementowanie
- python
- posiadasz
- wyszukujących
- znasz
- uwierzytelnianie
- dres
- niecierpliwością
- niespotykaną

Z racji bardzo szczególnego podzbioru języka polskiego używanego przy pisaniu ogłoszeń oraz relatywnie niewielkiej ich ilości, powyższe algorytmy dawały niezadowalające wyniki, dlatego też zdecydowaliśmy się na korzystanie z algorytmu opartego na tagach pobranych z serwisu *Stack Overflow*.

2.3.5 Generowanie statystyk

Przeznaczeniem tej części modułu jest odpowiadanie na żądania wyszukiwania oraz generowania statystyk. Komunikacja taka powinna odbywać się z wykorzystaniem protokołu HTTP, tak aby łatwo można było zintegrować moduł z aplikacją WWW bądź w przypadku takiej potrzeby innym dowolnym konsumentem.

Z powodu złożoności zapytań (jak np. generowanie statystyk sumarycznych dla wielu dni i wielu technologii jednocześnie) część ta potrzebuje własnej bazy danych na której wykonywane będą zapytania. Wymusza to dodatkowe wymaganie w postaci możliwości dodawania nowych ofert, z którego korzysta proces ekstrakcji technologii (wysyłając na bieżąco w pełni przetworzone oferty).

Wymagania

Wymagania funkcjonalne sprowadzają się więc do obsługi następujących żądań HTTP:

- dodanie nowej oferty do bazy
- wyszukiwanie ofert przy pomocy danego zbioru technologii
- generowanie statystyk, tj. dla każdego dnia z zakresu od 02.12.2018 do daty wykonania żądania obliczenie ilości aktywnych wówczas w serwisie pracuj.pl ofert zawierających podane technologie
- generowanie statystyk całościowych - tj. ilości wszystkich ofert w bazie (również we wspomnianym wyżej zakresie) oraz daty dodania ostatniego ogłoszenia

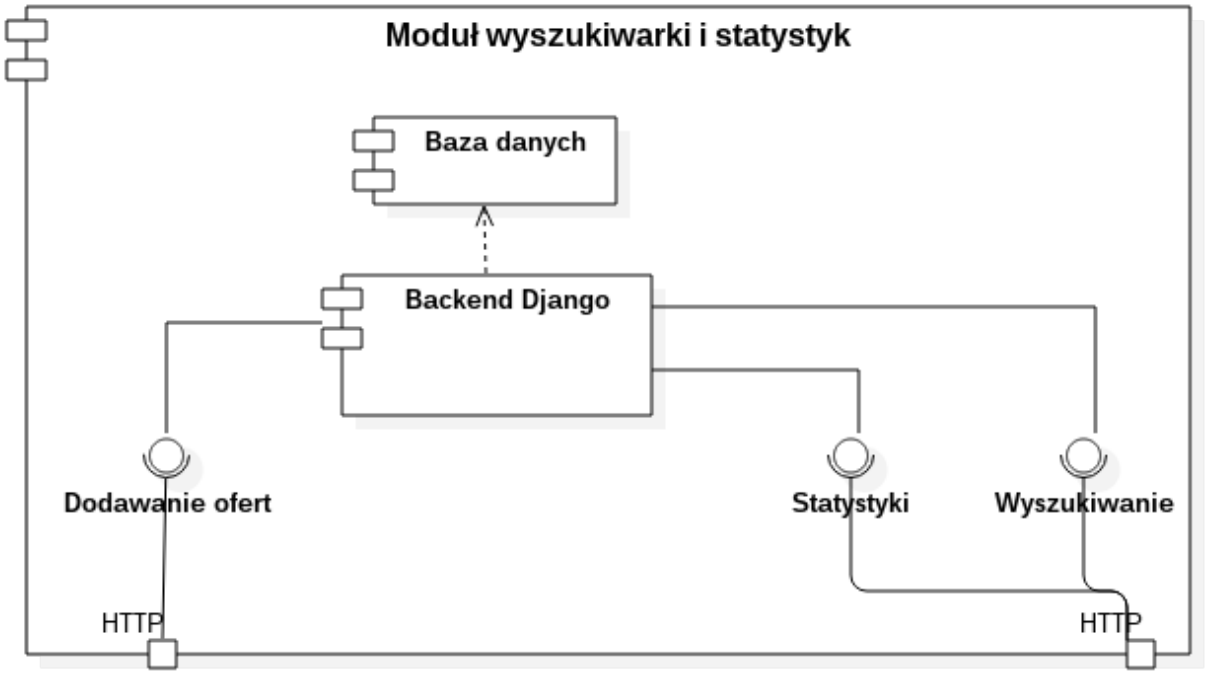
Natomiast wymagania niefunkcjonalne postawione modułowi to:

- **niezawodność** - usługa powinna być dostępna cały czas, ponieważ dostarcza ona danych niezbędnych do pokazania wszystkich statystyk w aplikacji z której korzysta użytkownik. Bez niej aplikacja WWW staje się bezużyteczna.
- **wydajność** - liczba przychodzących do modułu danych może być momentami bardzo duża, dlatego zapisywanie do bazy nie może być przeprowadzane niewydajnie. Największy wpływ na wydajność mają zapytania generujące odpowiednie statystyki. Niektóre z nich są skomplikowane, dlatego niezbędna jest ich optymalizacja. Wydłużony czas generowania statystyk obniży znacznie komfort używania aplikacji WWW

Interfejs

Usługa z pozostałymi komponentami systemu łączy się przez interfejs HTTP. Dane do statystyk są dostarczane przez proces ekstrakcji technologii. Konsumentem generowanych przez usługę statystyk jest aplikacja WWW z której korzysta końcowy użytkownik.

Interfejs zaimplementowany jest przy użyciu frameworka Django(Django n.d.).



Rysunek 2.6: Schemat modułu.

Baza danych

Wykorzystanym silnikiem bazy danych jest SQLite3(SQLite n.d.). Wybór padł na bazę relacyjną ze względu na to, że wykonwane są do niej skomplikowane zapytania, które silnikom nierelacyjnych baz danych zajmują więcej czasu oraz zasobów serwera, o czym przekonaliśmy się, testując te same zapytania na bazie danych MongoDB(MongoDB n.d.). Struktura dokumentów przechowywanych w bazie jest relacyjnym odzwierciedleniem struktury ogłoszenia zebranego przez robota zbierający dane. Baza zawiera trzy tabele:

- Oferty
  - Adres URL
  - Czas w którym pobrano ofertę
  - Kod HTML strony z ofertą
  - ID oferty w systemie pracuj.pl
  - Data dodania
  - Data ważności
  - Podmiot dodający
  - Tytuł oferty
  - Miejsce pracy
  - Kategorie oferty
  - Kod HTML treści oferty (rozbity na opis, kwalifikacje oraz benefity - wg struktury Pracuj.pl)
- Tagi
  - nazwa technologii
- Dodatkowa tabela reprezentująca relację ofert z tagami - jest to bowiem relacja `many-to-many` - czyli wiele do wielu.

W celu optymalizacji często wykonywanych zapytań baza posiada założone indeksy. W tabeli tagów indeksowaną kolumną jest nazwa technologii, zaś w bazie ofert data dodania oferty oraz data wygaśnięcia oferty.

Algorytm generowania statystyk

O ile wyszukiwanie ofert na podstawie listy podanych technologii realizowane jest w wydajny i optymalny sposób przez użyty framework, o tyle generowanie statystyk w określony przez wymagania modułu sposób jest bardziej skomplikowane i wymagało implementacji własnego algorytmu.

Przypominając wymagania, interfejs statystyk ma działać w następujący sposób:

- Przyjmij listę technologii
- Dla każdego dnia z przedziału od 02.12.2018 (początek zbierania ofert) do dzisiaj policz ile tego dnia było w serwisie pracuj.pl aktywnych ofert zawierających te technologie oraz jaki procent wszystkich aktywnych ofert z branży IT stanowiły.
- Zwróć wynik

Ofertę uważa się za aktywną danego dnia, jeśli zachodzą dwa warunki: `dzień dodania <= wybrany dzień` oraz `dzień wygaśnięcia >= wybrany dzień`. Mając te dwie daty w bazie danych można więc konstruować zapytania wybierające odpowiednie oferty.

Naiwna implementacja algorytmu, sprowadzałaby się do wykonania n zapytań zliczających do bazy danych, gdzie n to ilość dni które upłynęły od daty początkowej. Nie jest to rozwiązanie optymalne, ponieważ zapytania do bazy danych są kosztowne a z każdym kolejnym dniem wygenerowanie statystyk wymagałoby ich więcej.

Postanowiliśmy ograniczyć się do jednego zapytania, a algorytm wygląda następująco:

1. Wybierz oferty aktywne przez choć jeden dzień na zadanym przedziale. Wykorzystane warunki to: `dzień dodania <= koniec przedziału`, `dzień wygaśnięcia >= początek przedziału` oraz warunek zawierania się wszystkich podanych technologii wśród tych wykrytych w ofercie.
2. Utwórz n "kubełków", gdzie każdy kubełek odpowiada jednej dacie z przedziału i na początku ma wartość 0
3. Przejdź po liście ofert i dla każdej z nich:
    - Zwiększ wartość kubełka odpowiadającego dacie dodania oferty o 1. Może się zdarzyć że data dodania oferty poprzedza początek zakresu, i nie ma takiego kubełka. Wtedy zwiększ wartość pierwszego kubełka.
    - Jeśli data wygaśnięcia oferty poprzedza koniec zakresu, to zmniejsz wartość w kubełku z dniem następnym po dniu wygaśnięcia.
4. Wykonaj operację sumy skumulowanej na kubełkach.

Dzięki temu w każdym kubełku odpowiadającym każdemu dniu z zakresu znajdzie się ilość pasujących do zapytania, aktywnych tamtego dnia ofert. Do uzyskania wyniku procentowego operacja jest powtarzana, ale w kroku 1 pomijany jest warunek dotyczący technologii. Wtedy wynikiem jest iloraz rezultatów tych dwóch operacji.

2.3.6 Wykorzystane technologie

Flask

Licencja: BSD3

Flask jest napisanym w języku Python mikro-frameworkiem. Głównym jego zastosowaniem jest tworzenie niewielkich aplikacji sieciowych czy REST API. Posiada wbudowany serwer deweloperski i debugger. Stawia duży nacisk na zwięzłość kodu, co pozwala na szybkie prototypowanie ale i późniejsze rozwijanie aplikacji.

Django

Licencja: BSD3

Jest to jeden z największych i najszybciej rozwijanych projektów Open Source. Zdecydowaliśmy się na niego ze względu na to, że w tym module kluczowa jest wydajność generowania odpowiednich statystyk, ale też duża elastyczność w pisaniu kodu. Ważną zaletą Django jest wbudowany ORM który znacząco ułatwia współpracę z bazą danych.

Celery (i Luigi)

Licencja: BSD3

Celery(Celery n.d.) jest asynchroniczną kolejką zadań. Zadania są dystrybuowane do kolejki z różnych źródeł, np. z aplikacji sieciowej. Celery jest przeznaczone głównie do operacji zlecanych oraz wykonywanych w czasie rzeczywistym. Jako kolejkę lub bazę na zlecone zadania możliwe jest wykorzystanie wielu backendów np. redis czy rabbitmq.

Początkowe plany zakładały użycie w tym miejscu frameworka **Luigi**(Luigi n.d.). Jest to stworzone przez twórców aplikacji *Spotify* narzędzie do łączenia ze sobą kolejnych funkcji / etapów, tworząc łańcuch przetwarzania (*Pipeline*). Okazało się jednak, że przewidziane zastosowania narzędzia obsługują etapy innego typu niż te wymagane przez proces ekstrakcji technologii. Luigi przeznaczony jest do łączenia wymagających zadań, angażujących wiele zewnętrznych usług czy języków programowania i wykonujących się nawet kilka dni. W efekcie nie udostępnia chociażby tak podstawowej w mniejszych zastosowaniach możliwości jak uruchamianie zadania z poziomu kodu źródłowego. Możliwe jest uruchamianie ich jedynie za pomocą linii poleceń.

Zdecydowaliśmy się na porzucenie Luigiego na rzecz frameworka *Celery*, którego obsługa zadań jest tym czego potrzebujemy. Minusem takiego wyboru jest utrata odporności na awarie (Luigi zapisuje stan po każdym zadaniu i wraca do niego po awarii), lecz dużym zyskiem jest zwiększona łatwość implementacji.

2.4 Aplikacja WWW

Kod źródłowy znajduje się pod adresem:

- github.com/jobsbrowser/frontend.

Aplikacja będąca ostatnim komponentem systemu oraz jedynym który wchodzi w bezpośrednią interakcję z użytkownikiem.

2.4.1 Wymagania

Głównym wymaganiem funkcjonalnym aplikacji jest prezentowanie danych oraz wyciągniętych z nich statystyk i informacji użytkownikowi w jak najbardziej przystępny oraz przejrzysty sposób. Rozwiązaniem jest organizacja strony na trzy zakładki pomiędzy którymi użytkownik może się swobodnie przełączać. Zakładki odpowiadają odpowiednio możliwościom:

- wyświetlania statystyk na podstawie wpisanych w polu wyszukiwania technologii
- wyszukiwania ofert na podstawie wpisanych w polu wyszukiwania technologii
- przeglądania strony informacyjnej wraz ze statystykami zbiorczymi systemu

Wymagania niefunkcjonalne sprowadzają się natomiast do:

- **niezawodności** - usługa powinna być dostępna cały czas, ponieważ jest ona usługą końcową z której użytkownicy powinni móc korzystać w każdej chwili.
- **wydajności** - liczba użytkowników korzystających ze strony może być bardzo duża, dlatego też aplikacja powinna być napisana wydajnie, aby jej niedostępność lub zbyt wolne działanie nie przynosiły negatywnych odczuć użytkownikowi.

2.4.2 Struktura kodu

Poniżej prezentujemy drzewo katalogów oraz plików modułu wraz z krótkim omówieniem.

```
index.html            # główny plik HTML
package.json          # plik z zależnościami projektu
package-lock.json
public                # katalog na zasoby niezmieniające się np. zdjęcia, ikony
README.md
```

```
src                    # katalog z kodem źródłowym aplikacji
    App.vue            # główny komponent aplikacji
    components         # katalog z komponentami UI
        LineChart.js   # komponent obsługujący rysowanie wykresów liniowych
        Menu.vue       # komponent reprezentujący menu aplikacji
        Offer.vue      # komponent reprezentujący widok oferty
        TagInput.vue   # komponent obsługujący pobieranie kluczy od użytkownika
    main.js            # plik wejściowy dla aplikacji
    pages              # katalog z komponentami reprezentującymi podstrony aplikacji
        Info.vue       # komponent prezentujący informacje o projekcie
        Search.vue     # komponent umożliwiający interaktywne wyszukiwanie ofert
        Stats.vue      # komponent pokazujący statystyki
    router
        index.js       # konfiguracje routowania aplikacji
    store
        index.js       # konfiguracja oraz inicjalizacja vuex-store
webpack.config.js      # ustawienia babela
```

2.4.3 Główne komponenty aplikacji

Poniżej prezentujemy najważniejsze komponenty aplikacji wraz z krótkim opisem:

- komponenty UI
  - `Menu` - komponent generujący widok górnego menu.
  - `TagInput` - komponent obsługujący pobieranie kluczy (tagów) od użytkownika w interaktywny sposób.
  - `Offer` - komponent generujący widok listy ofert z odpowiednimi tagami pobranymi od użytkownika poprzez komponent `TagInput`.
  - `LineChart` - komponent renderujący wykresy liniowe dla kluczy pobranych dzięki komponentowi `TagInput`.
- komponenty stron
  - `Search` - komponent korzystający z komponentów: `Menu`, `TagInput` oraz `Offer`. Pobiera informacje o ofertach z pobranymi tagami i prezentuje je w formie listy.
  - `Info` - komponent korzystający z komponentów: `Menu`. Pobiera informacje na temat projektu **JobsBrowser** oraz je prezentuje.
  - `Stats` - komponent korzystający z komponentów: `Menu`, `TagInput` oraz `LineChart`. Pobiera statystyki dotyczące wybranych kluczy oraz prezentuje je użytkownikowi w formie wykresów.

2.4.4 Instrukcja użytkowania oraz testy akceptacyjne

Z racji tego, że funkcjonalność aplikacji jest bardzo zwięzła instrukcja jej użytkowania może mieć postać dokumentacji testów akceptacyjnych i tak też chcielibyśmy ją przedstawić.

2.4.4.1 Zakładka statystyk

Jest to pierwsza zakładka którą jako użytkownik widzimy po wprowadzeniu w przeglądarce adresu aplikacji. W górnej części strony znajduje się pasek menu pozwalający na zmianę zakładki, a w środkowej pole do wpisywania kluczy.

Po wprowadzeniu klucza i wciśnięciu klawisza Enter zakoloruje się on, a pod spodem pojawią się dwa wykresy. Wpisany klucz można usunąć lub dodać do niego kolejny pisząc w polu i ponownie wciskając Enter.

Na pierwszym wykresie przedstawiona będzie ilość aktywnych ofert zawierających dany klucz konkretnego dnia, a na drugim jaką procentowo część wszystkich aktywnych wówczas ofert z branży IT one stanowiły. Wykresy są w pewnym stopniu interaktywne. Tzn. nie pozwalają na zmianę skali czy zakresu, ale pozwalają na dokładne zbadanie wartości w określonym dniu najeżdżając na wykres kursorem myszy.

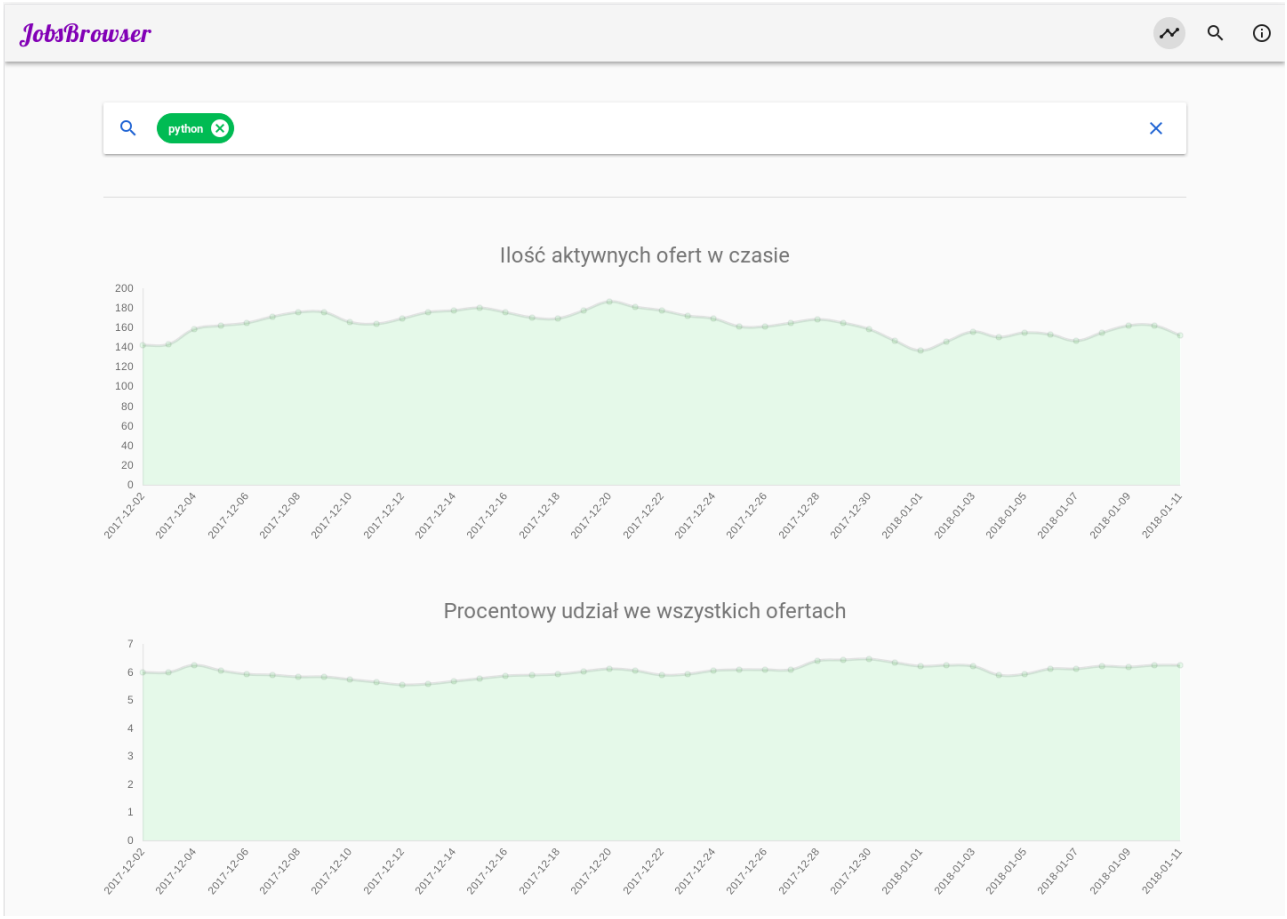
2.4.4.2 Zakładka wyszukiwania

Drugą dostępną z menu zakładką jest ta z widokiem wyszukiwania. Tutaj ponownie zobaczymy identyczne pole do wpisywania kluczy, jednak tym razem pod spodem pojawi nam się lista znalezionych ofert które ten klucz zawierają. Każdą z nich możemy rozwinąć żeby zobaczyć więcej informacji, w tym link do oryginalnego ogłoszenia czy pozostałe wykryte w ofercie klucze. Lista jest paginowana, co zwiększa komfort użytkownika i poprawia szybkość ładowania wyników.

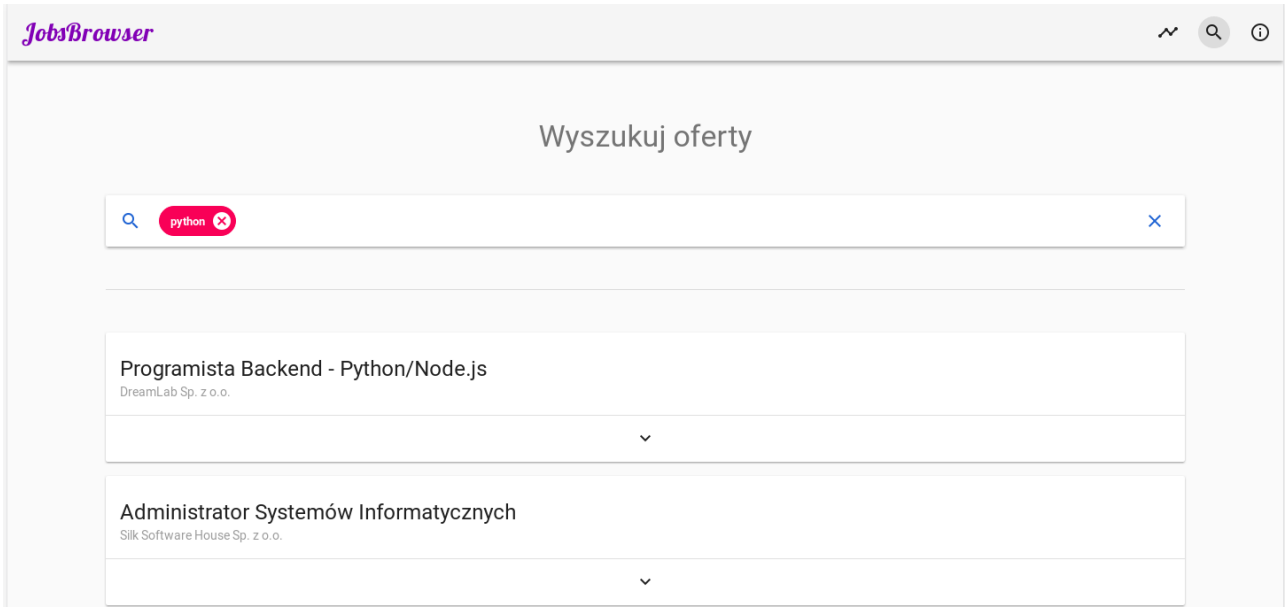
Ogłoszenia na liście posortowane są wg daty ich wygaśnięcia (znaleźć na niej można również ogłoszenia archiwalne). O statusie danej oferty informuje etykieta w prawym górnym rogu.

2.4.4.3 Zakładka informacji

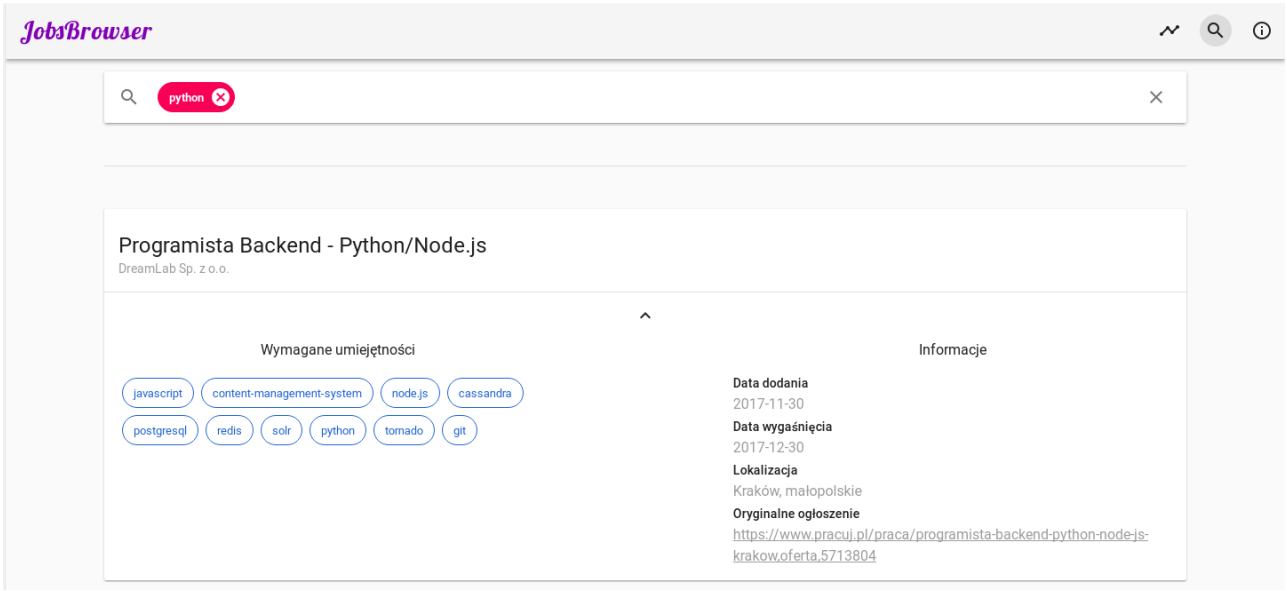
Ostatnią zakładką jest ta poświęcona informacjom o serwisie oraz statystykom zbiorczym. Zobaczymy tam krótki opis projektu oraz wykres ilości wszystkich przetworzonych przez niego ofert względem czasu.



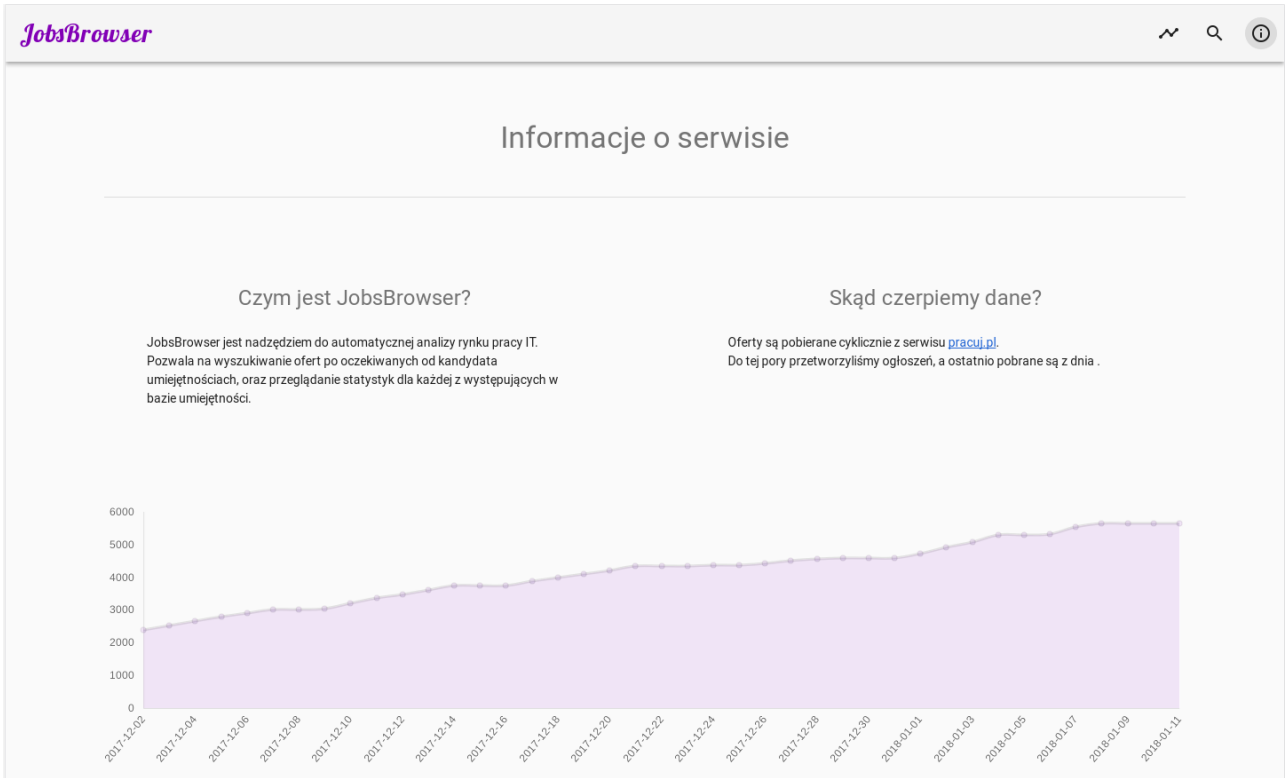
Rysunek 2.7: Zakładka statystyk.



Rysunek 2.8: Zakładka wyszukiwania.



Rysunek 2.9: Rozwinięte ogłoszenie.



Rysunek 2.10: Zakładka informacji.

2.4.5 Wykorzystane technologie

VueJS

Licencja: MIT

Aplikacja została napisana we frameworku VueJS(VueJS n.d.). Jest to framework napisany w języku JavaScript oparty na architekturze MVVVM (Model-View–View-Model). VueJS umożliwia tworzenie lekkich oraz szybkich aplikacji sieciowych. Budowanie aplikacji polega na tworzeniu tzw. komponentów i składaniu z nich całej aplikacji. VueJS wydaje się być dobrym wyborem przy tworzeniu aplikacji, które opierają się na konsumowaniu danych z różnych API oraz prezentowaniu ich użytkownikowi.

Do budowy aplikacji skorzystaliśmy z kilku rozszerzeń usprawniających pracę z Vue. Przedstawiamy je poniżej:

- `vuetify`(Vuetify n.d.) - framework dostarczający gotowe komponenty stworzone w stylu material design
- `vuex`(Vuex n.d.) - system (wzorzec) zarządzania stanem aplikacji napisanej w VueJS. Zapewnia miejsce na przechowywanie danych, które będą dostępne przez każdy komponent oraz kontroluje ich nadpisanie czy zmianę
- `chart.js`(Chart.js n.d.) - biblioteka używana do tworzenia estetycznych wykresów
- `axios`(axios n.d.) - klient HTTP wykorzystany do komunikacji z modułem analitycznym

Historia zmian dokumentu

Tabela 2.1: Historia Zmian

Data	Autor	Opis zmian	Wersja
23.11.2017	Bartłomiej Sielicki	Rozdział 1	0.1
	Łukasz Skarżyński	Rozdział 2 - moduł zbierania danych	

Bibliografia

axios. Klient http napisany w node.js. Dostępne pod adresem: *https://github.com/axios/axios* [2018-01-13].

Bengio Yoshua, V.P., Ducharme Réjean. Advances in neural information processing systems 13 (nips'00). Dostępne pod adresem: *http://www.iro.umontreal.ca/~lisa/publications2/index.php/publications/show/64* [2018-01-27].

Celery. Framework pozwalający na asynchroniczne wykonywanie zadań i łańcuchów przetwarzania (pipeline). Dostępne pod adresem: *http://www.celeryproject.org/* [2017-12-16].

Chart.js. Biblioteka ułatwiająca rysowanie wykresów. Dostępne pod adresem: *http://www.chartjs.org/* [2018-01-13].

Django. Framework wykorzystywany do tworzenia aplikacji webowych oraz rest api. Dostępne pod adresem: *https://www.djangoproject.com/* [2018-01-13].

Flask. Framework wykorzystywany do tworzenia stron internetowych oraz api. Dostępne pod adresem: *http://flask.pocoo.org/* [2017-12-16].

Gensim. Biblioteka przeznaczona do przetwarzania języka naturalnego oraz pozyskiwania informacji w tekscie. Dostępne pod adresem: *https://github.com/RaRe-Technologies/gensim* [2018-01-27].

Luigi. Moduł pozwalający budować skomplikowane łańcuchy(en. pipelines) zadań, operacji czy transformacji na danych. Dostępne pod adresem: *https://luigi.readthedocs.io/en/stable* [2017-12-16].

MongoDB. Nierelacyjna baza danych, przechowująca rekordy w formacie bson. Dostępne pod adresem: *https://www.mongodb.com/* [2017-12-16].

Rake-NLTK. Implementacja w języku python (z użyciem nltk) algorytmu rake (en. rapid automatic keyword extraction). wykorzystywany do automatycznej ekstrakcji słów kluczowych z ogłoszeń. Dostępne pod adresem: *https://github.com/csurfer/rake-nltk* [2017-12-16].

Scrapy. Zestaw narzędzi umożliwiający szybką, wydajną ekstrakcję informacji ze stron internetowych. Dostępne pod adresem: *https://scrapy.org* [2017-12-16].

SQLite. System zarządzania relacyjną bazą danych. Dostępne pod adresem: *https://www.sqlite.org/* [2018-01-13].

TF-IDF. Algorytm wyznaczania wagi słów na podstawie ich wystąpień. Dostępne pod adresem: *https://en.wikipedia.org/wiki/Tf-idf* [2018-01-21].

VueJS. Framework wykorzystywany do tworzenia progresywnych aplikacji internetowych. Dostępne pod adre-

sem: *https://vuejs.org* [2017-12-16].

Vuetify. Zestaw komponentów ui do vuejs zbudowany przy użyciu wyglądu material design. Dostępne pod adresem: *https://vuetifyjs.com/* [2018-01-13].

Vuex. System zarządzania stanem komponentów vuejs. Dostępne pod adresem: *https://vuex.vuejs.org* [2018-01-13].

Yoav Goldberg, O.L. Word2vec explained: Deriving mikolov et al.'s negative-sampling word-embedding method. Dostępne pod adresem: *https://arxiv.org/abs/1402.3722* [2018-01-27].