”Automatyczna analiza rynku pracy.”

Bartłomiej Sielicki
Łukasz Skarżyński

Praca inżynierska

Promotor:
Barbara Rychalska

Wydział Matematyki i Nauk Informacyjnych
Politechniki Warszawskiej
19.11.2017

Rozdział 1

Specyfikacja projektu

1.1 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.

1.2 Opis biznesowy

Głównym zadaniem aplikacji ma być automatyczna analiza rynku pracy. System zajmuje się agregacją oraz przetwarzaniem (analizą) 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 wyłuskując kluczowe elementy: tytuł ogłoszenia, opis, datę dodania, itp.
2. Zapisania tak zebranych ogłoszeń do własnej bazy danych
3. Przeprowadzenia analizy na treści ogłoszenia uzyskując zeń listę *kluczy*, tj:
    - obszaru branży IT którego dotyczy ogłoszenie
    - stanowiska którego dotyczy
    - wymaganych od pracownika opanowanych technologii / umiejętności
4. Udostępnienia użytkownikowi interfejsu do wyszukiwania zebranych w systemie ofert oraz wyświetlania statystyk przy użyciu wyżej wspomnianych kluczy.

Wymienione działania realizują odrębne komponenty systemu. Bardziej szczegółowy podział i opis znajduje się w sekcji wymagań funkcjonalnych oraz rozdziale drugim będącym dokumentacją każdego z modułów.

Użytkownik bezpośrednio będzie korzystał tylko z ostatniego modułu - aplikacji WWW będącej interfejsem dla zebranych i przetworzonych przez system danych.

Aplikacja będzie niosła korzyść duzej grupie odbiorców takich jak:

- studenci - dzięki aplikacji będą mogli znaleźć świetnie dopasowane stanowisko do ich umiejętności lub obserwując dane stanowisko określić jakie umiejętności są na nim elementarne, a także jakiej technologii powinni się nauczyć w najbliższym czasie.
- osoby szukjące pracy - łatwo na podstawie swoich umiejętności znajdą stanowiska dla siebie.
- pracodawcy - na podstawie trendów wśród używanych technologii będą mogli podjąć decyzje dotyczące przyszłych projektów.
- pozostali użytkownicy zainteresowani analizą rynku pracy.

1.3 Wymagania funkcjonalne

Podstawą interakcji użytkownika jest strona WWW - użytkownik powinien być w stanie otworzyć ją na dowolnym komputerze z dostępem do internetu i wyposażonym w aktualną wersję jednej z wiodących na rynku przeglądarek.

- 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 jako twórcy nie możemy zagwarantować że będzie to działanie w pełni poprawne.

Na stronie nie przewidujemy kont użytkowników, nawet administracyjnego. Każdy wchodzący na stronę będzie miał dostęp do tych samych danych oraz takie same możliwości.

Funkcjonalność strony rozbita jest na dwa moduły. Moduł wyszukiwarki oraz moduł statystyk. Możliwości użytkownika prezentuje załączony na końcu sekcji diagram przypadków użycia.

1.3.1 Wyszukiwanie ofert wg klucza

Jednym z podstawowych zastosowań strony jest wyszukiwanie ofert zebranych i umieszczonych w bazie przez nasz system. Jako kryterium wyszukiwania (przez mechanizm filtrowania) może zostać użyty tzw. *klucz*, czyli wyłuskana z opisu ogłoszenia jego cecha. Klucze dzielimy na trzy kategorie:

- **Obszary** (np. Mobile development, Helpdesk)
- **Stanowiska** (np. Software Developer, Data Scientist)
- **Technologie i umiejętności** (np. Java, Docker, AWS)

Pozwoli to na kompleksowe wyszukiwanie ofert ze względu na branżę czy pozycję którą interesuje się użytkownik oraz posiadane przez niego umiejętności. Możliwe jest podanie wielu kluczy jako kryterium.

1.3.2 Wyświetlenie oferty

Wynikiem wyszukiwania jest lista ofert. Każdą z nich użytkownik może wyświetlić uzyskując dostęp do takich informacji jak:

- Tytuł oferty
- Data dodania i wygaśnięcia oferty w macierzystym serwisie
- Nazwa pracodawcy i miejsce pracy
- Wszystkie wyłuskane przez system klucze
- Odnośnik do oryginalnego ogłoszenia w macierzystym serwisie

1.3.3 Wyświetlenie statystyk serwisu

W osobnej sekcji użytkownik ma dostęp do wyświetlenia zbiorczych statystyk dotyczących serwisu i dostępnych w nim danych. Zbiór dostępnych statystyk planowo zostanie rozwinięty podczas pracy nad ostatnimi dwoma modułami systemu. Bazowe, przewidziane już teraz to:

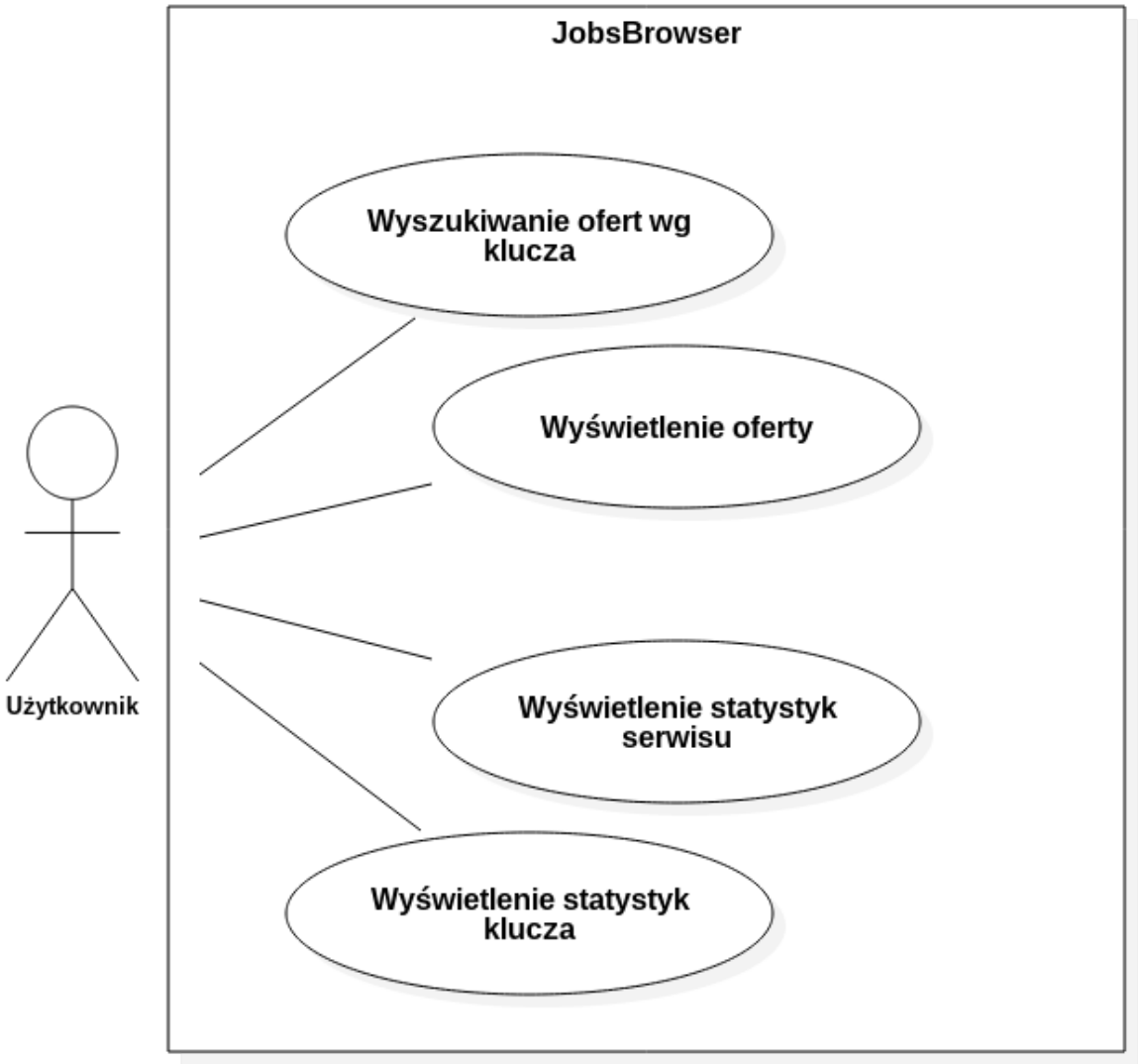
- Liczba wszystkich ofert w bazie systemu
- Wykres powyższej wartości względem czasu
- Datę ostatniego zebrania danych z serwisu macierzystego
- Listę wszystkich istniejących w systemie kluczy

1.3.4 Wyświetlenie statystyk klucza

Jest to druga obok wyszukiwania podstawowa funkcjonalność strony. Pozwala ona użytkownikowi na wybór klucza oraz wyświetlenie:

- Wykresu ilości ofert zawierających ten klucz względem czasu
- Wykresu procentowej ilości ofert z systemu zawierających ten klucz
- Pogrupowanych w kategorie kluczy które występują z tym kluczem najczęściej w jednym ogłoszeniu

Możliwe jest podanie wielu kluczy. Wtedy pod uwagę przy generowaniu powyższych statystyk będą brane tylko ogłoszenia które zawierają każdy z nich.



Rysunek 1.1: Przypadki użycia.

1.4 Wymaganie niefunkcjonalne

W poniższej tabeli przedstawione zostały wymagania niefunkcjonalne tworzonej 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 powinna być 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. Pozostałe moduły powinny działać bez problemów na oddzielnych serwerach.
Wydajność (Performance)	Aplikacja WWW powinna działać płynnie na każdym komuterze z dowolnym systemem operacyjnym na 1 z wcześniej wymienionych przeglądarek. Pozostałe moduły powinny uruchamiać się same co określony odstęp czasu. Początkowo planowane jest uruchamianie ich raz dziennie.
Wsparcie (Supportability)	W razie jakichkolwiek problemów w aplikacji WWW dostępny jest formularz kontaktowy.

1.5 Schemat architektury

Przewidziana architektura ma strukturę modułową. Schemat komponentów oraz ich połączenie przedstawia załączony na końcu sekcji diagram.

W systemie wyróżnić możemy cztery główne, niezależne od siebie (na tyle że mogą, a nawet powinny, być uruchamiane na różnych maszynach) moduły. Ta sekcja zawiera ich ogólny, wysokopoziomowy opis. Szczegóły dotyczące architektury i implementacji znajdują się w następnym rozdziale.

1.5.1 Moduł zbierający dane

Komponent ten odpowiada za automatyczne pobieranie ogłoszeń z serwisu zewnętrznego. Jest to program, który cyklicznie łączy się z udostępniającym ogłoszenia serwisem i automatycznie pobiera ich treść. Z pobranych ogłoszeń w najprostszy sposób (poprzez analizę kodu HTML) wyłuskuje podstawowe elementy, takie jak:

- Tytuł ogłoszenia
- Treść
- Data dodania
- Pracodawca
- Miejsce pracy

W kolejnym kroku tak "rozbite" ogłoszenie przesyła do kolejnego komponentu systemu.

Opisany proces nie obejmuje zapisu do żadnej bazy danych. W tej kwestii moduł zbierający dane polega całkowicie na komponencie, do którego przekazuje pobrane ogłoszenia. To samo dotyczy kwestii rozpoznawania duplikatów - program przed każdym rozpoczęciem skanowania prosi swojego "odbiorcę" o listę już zeskanowanych ogłoszeń aby uniknąć przetwarzania ich ponownie.

1.5.2 Pipeline

Zebrane ogłoszenia trafiają do komponentu *Pipeline* (Łańcucha przetwarzania). Nazwa komponentu odnosi się do zasady jego działania. Odbierane ogłoszenia przekazywane są przez kolejne podmoduły, które wykonują na nich stosowne operacje.

Pierwszym elementem jest moduł przechowujący dane. Odbiera on nowe ogłoszenia od modułu zbierającego i zapisuje je w bazie danych. Jednocześnie oferuje usługę odczytania listy kluczy (adresów URL) dodanych już ogłoszeń, z czego moduł zbierający korzysta przed rozpoczęciem skanowania serwisu zewnętrznego.

Następnie ogłoszenia trafiają do modułu ekstrakcji kluczy. Tutaj zgodnie z wymaganiami funkcjonalnymi systemu z ogłoszenia wyłuskane są elementy z trzech kategorii:

- Obszar branży IT którego dotyczy ogłoszenie
- Stanowisko
- Technologie i umiejętności

Po ukończeniu procesu model obiekt ogłoszenia powiększa się o zestaw kluczy, a następnie zostaje wysłany do kolejnego komponentu.

1.5.3 Backend

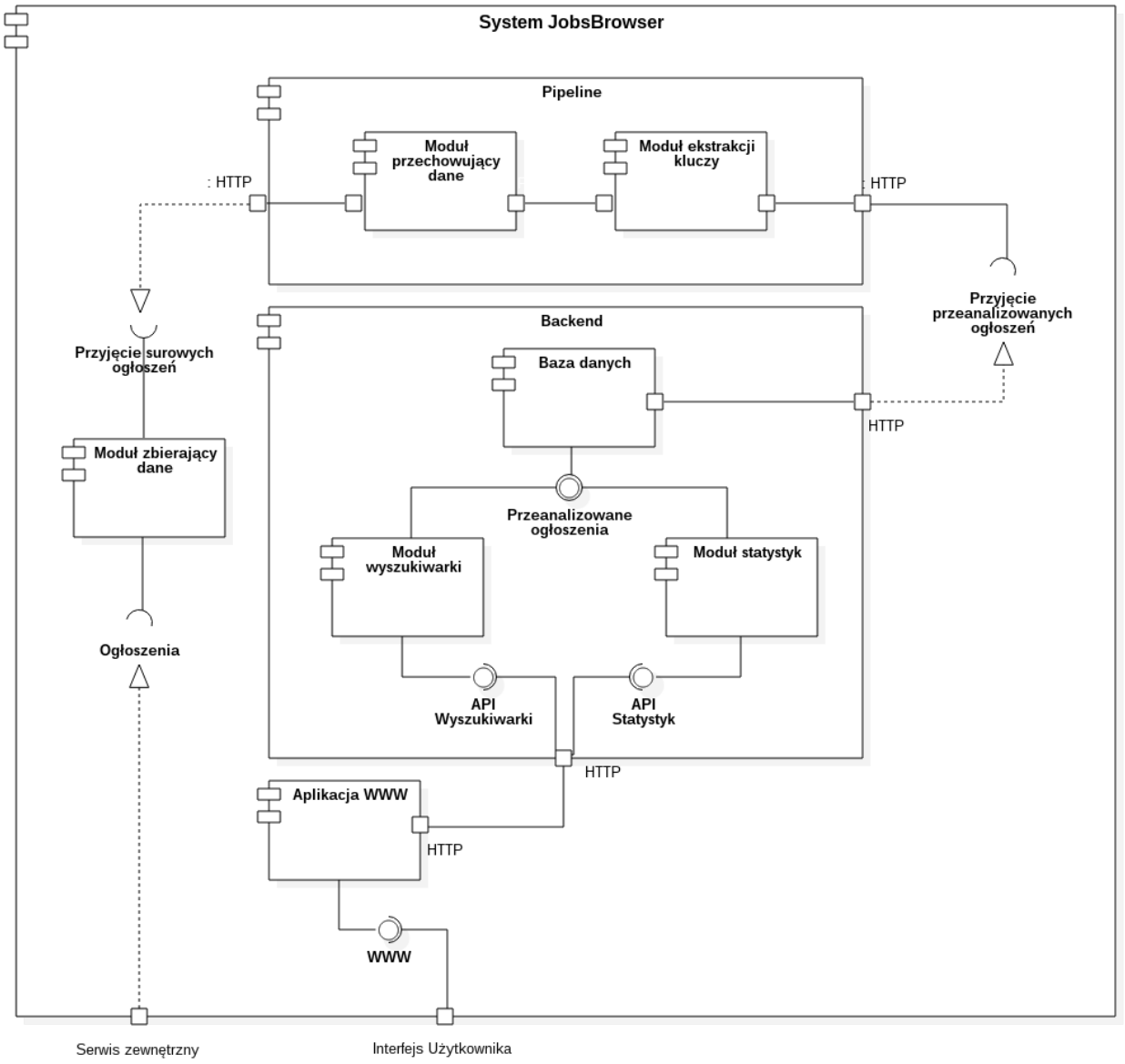
Moduł ten zajmuje się dostarczaniem użytkownikowi (a konkretnie aplikacji WWW) informacji na podstawie przeanalizowanych ofert. Do bazy danych zapisywane są ogłoszenia wraz z kluczami, a podmoduły pozwalają na jej odpytywanie.

Moduł wyszukiwarki wystawia interfejs pozwalający przeglądarce na wykonanie zapytania o ogłoszenia zawierające jednej lub więcej kluczy.

Interfejs modułu statystyk pozwala na dostęp do statystyk systemu jako całości oraz statystyk poszczególnych kluczy. Bardziej szczegółowe wymagania tego interfejsu znajdują się w sekcji wymagań funkcjonalnych.

1.5.4 Aplikacja WWW

Ostatnim elementem systemu, jedynym dostępnym bezpośrednio dla użytkownika jest aplikacja WWW. Nie posiada ona własnej bazy, a co za idzie kont użytkowników. Korzysta wyłącznie z interfejsu wyszukiwania oraz statystyk pozwalając użytkownikowi na przejrzysty i wygodny dostęp do informacji.



Rysunek 1.2: Diagram komponentów.

1.6 Model wytwórczy

Do wytworzenia opisywanego systemu wybraliśmy metodykę zwinną - Agile. Jest ona jedną z najszybszych metod wytwarzania oprogramowania i idealnie wpasowuje się w modułową strukturę architektury systemu. Wybór tej metodyki oznacza prowadzenie prac w modelu iteracyjnym - komponent po komponencie. W naszym przypadku oznacza to, że z każdą kolejną iteracją gotowy projekt będzie powiększał się o nowy, w pełni działający fragment. Po pierwszym etapie będzie to moduł zbierający dane z zewnętrznych serwisów, następnie moduł przechowujący i udostępniający zebrane dane, i tak dalej.

Wymagania funkcjonalne projektu zostały opracowane w pierwszej kolejności. Pozwoliło to na nakreślenie zarysu systemu i tego na co musi pozwalać, bez wdawania się w szczegóły implementacyjne. Po wyodrębnieniu komponentów systemu, uznaliśmy że podejście kaskadowe byłoby zbyt ryzykowne. Zaprojektowanie dokładnej struktury i szczegółowych wymagań implementacji każdego z nich już na wstępie, niosłoby za sobą ryzyko niedopasowania się do przewidzianego na implementację czasu. Zachodziłoby również niebezpieczeństwo przeoczenia błędów w planowaniu, które wyszłyby na powierzchnię dopiero w momencie implementacji bądź testów, kiedy metoda kaskadowa nie przewiduje już zmiany wymagań ani planu. Jakość końcowego produktu zdecydowanie bardzo by na tym ucierpiała.

Komponenty systemu są na tyle niezależne i wyizolowane, że mając opracowane ogólne ich założenia i przeznaczenie, podczas każdej z iteracji jesteśmy w stanie skupić się na nich indywidualnie, bez ingerowania w resztę projektu. Pozwoli to na dokładne ich dopracowanie, co jest szczególnie istotne biorąc pod uwagę, że część z nich opiera się w znacznym stopniu na zewnętrznych projektach i technologiach, które do udanej integracji będą wymagały głębszego poznania.

1.7 Harmonogram

Załączona tabela 1.2 przedstawia planowany harmonogram postępów w pracy nad projektem - przewidywane terminy ukończenia każdego z modułów. Wytłuszczone daty to terminy laboratoriów na których przewidziana jest kontrola postępów.

Tabela 1.2: Harmonogram

Rozpoczęcie iteracji	Ukończenie iteracji	Opis	Moduły
26.10.2017	**02.11.2017**	Przygotowanie harmonogramu oraz wstępnych wymagań	
02.11.2017	09.11.2017	Moduł zbierający dane Tydzień 1	
09.11.2017	**16.11.2017**	Moduł zbierający dane Tydzień 2	1/6
16.11.2017	23.11.2017	Moduł przechowujący dane	2/6
23.11.2017	**30.11.2017**	Moduł ekstrakcji kluczy Tydzień 1	
30.11.2017	07.12.2017	Moduł ekstrakcji kluczy Tydzień 2	
07.12.2017	**14.12.2017**	Moduł ekstrakcji kluczy Tydzień 3	3/6
14.12.2017	21.12.2017	Moduł statystyk Tydzień 1	
21.12.2017	28.12.2017	Moduł statystyk Tydzień 2	4/6
28.12.2017	04.01.2018	Moduł wyszukiwarki	5/6
04.01.2018	**11.01.2018**	Strona WWW	6/6

Rozdział 2

Specyfikacja komponentów systemu

2.1 Moduł zbierający dane

Kod źródłowy znajduje się pod adresem: github.com/jobsbrowser/scraper.

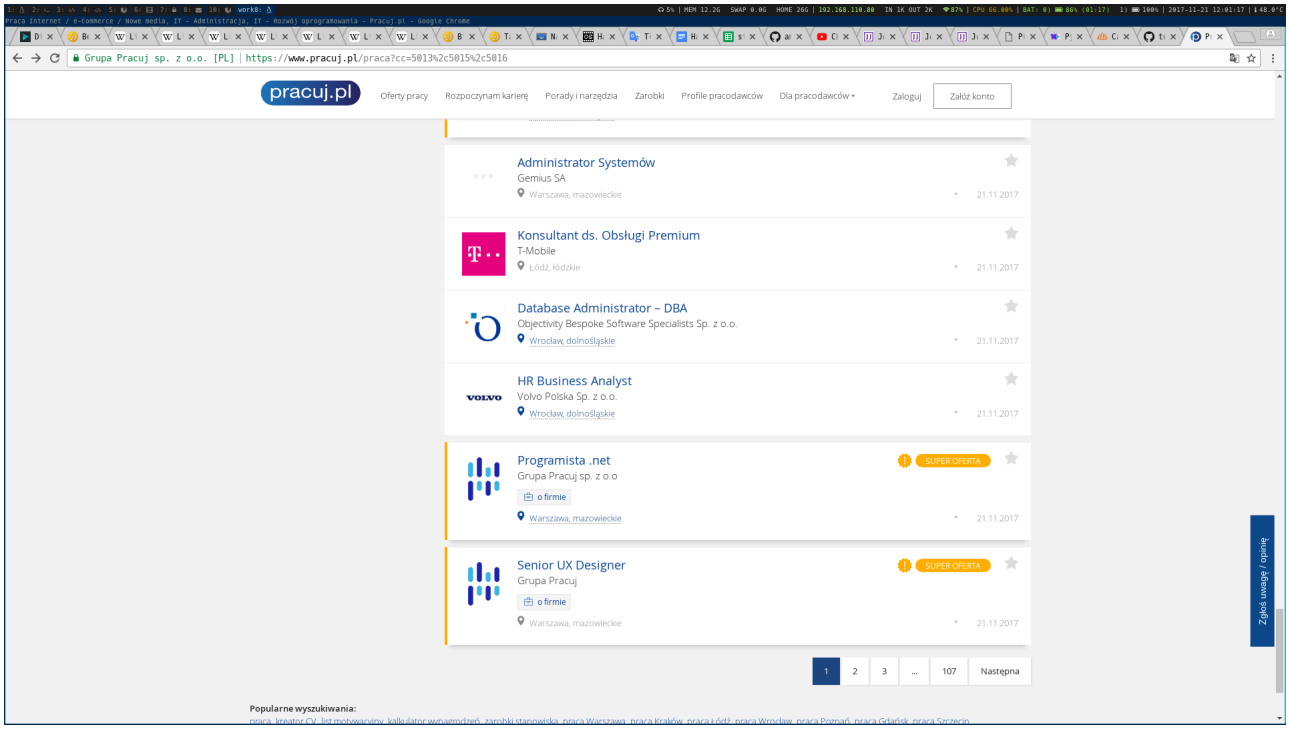
Komponent zajmuje się automatycznym pobieraniem ofert z serwisu zewnętrznego, ekstrakcją podstawowych informacji oraz przesłaniem tak przetworzonych ofert do kolejnego modułu. Zdecydowaliśmy się na serwis Pracuj.pl jako źródło ofert oraz na framework **Scrapy**(Anon n.d.) jako bazę naszego modułu.

2.1.1 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
- IT - Administracja
- IT - Rozwój oprogramowania

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 2.1: 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.

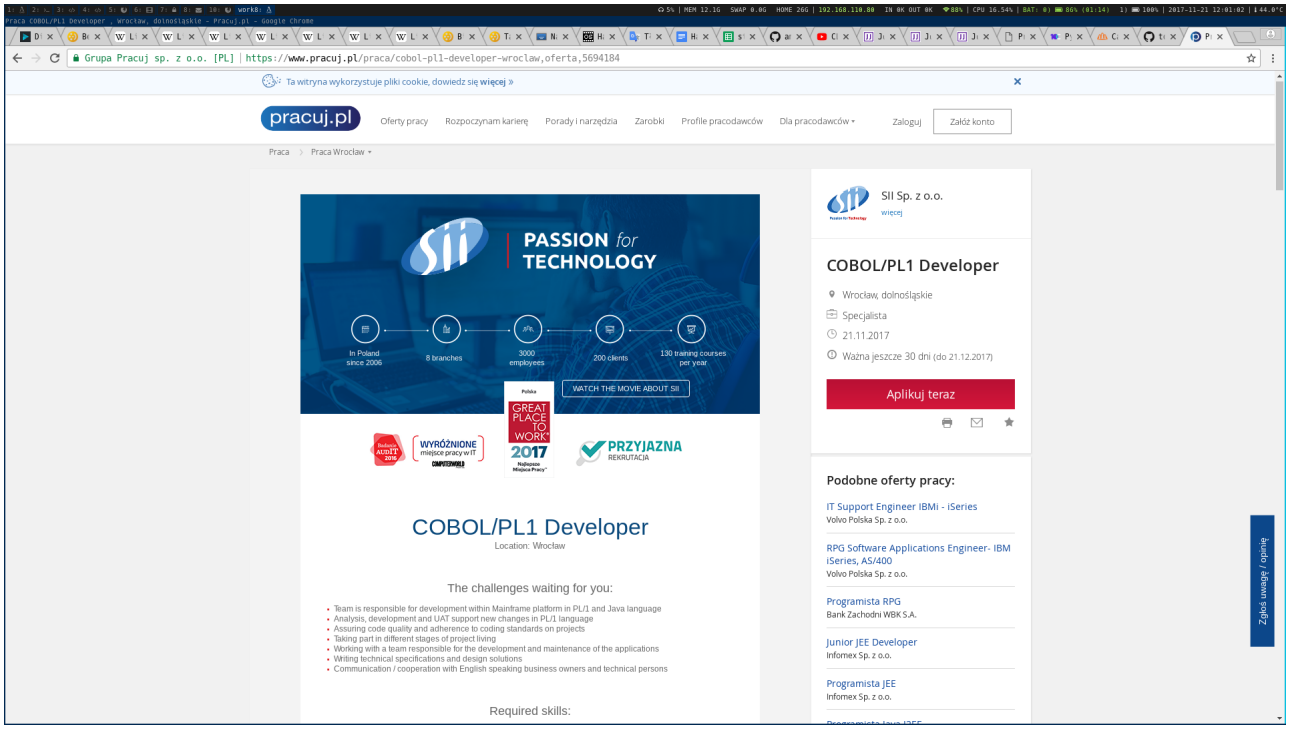
Do każdej z ofert na stronie (których znajduje się ok. 50) prowadzi bezpośredni link, który można wyłuskać z kodu HTML listy. Podstrona pojedynczej oferty zawiera wszystkie interesujące nas informacje również możliwe do wyłuskania z kodu HTML przy użyciu odpowiednich selektorów CSS. Dostępne w przystępny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

- Data dodania oferty
- Data wygaśnięcia
- Nazwa dodającego (pracodawca)
- Lokalizacja (miasto i województwo)
- Tytuł oferty
- Treść oferty

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 2.2: Widok oferty w witrynie pracuj.pl.

2.1.2 Podejście

Zadaniem stojącym przed scraperem jest automatyczne przejście po wszystkich dostępnych stronach listy, wyłuskanie zeń 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.

Ofert w serwisie wraz z upływem czasu będzie przybywać - i dla naszego 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 jest 12 godzin, choć nie jest to wartość zdefiniowana w programie. To od uruchamiającego program na komputerze zależy jak ją ustawi.

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. Zdecydowaliśmy się na rozwiązanie tego programu przez 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.

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

2.1.3 Scrapy

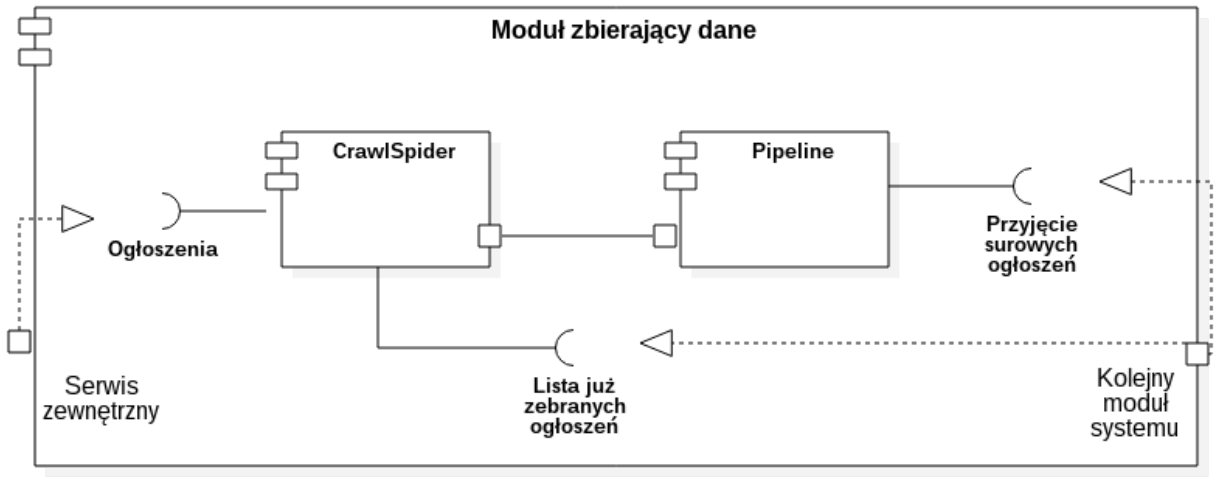
Do napisania scrapera zdecydowaliśmy się na framework Scrapy. Jest to framework napisany w języku Python, przy wykorzystaniu biblioteki Twisted. Dzięki temu działa całkowicie asynchronicznie, co czyni go niezwykle wydajnym przy relatywnej łatwości pisania własnych scraperów. Wydany jest na licencji BSD3.

2.1.4 Uruchomienie i użytkowanie

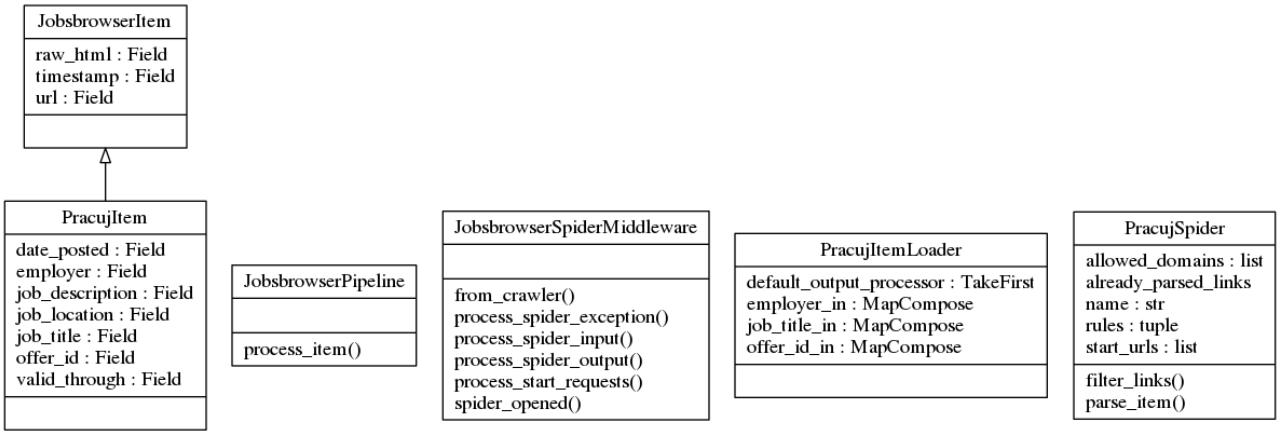
W katalogu `jobsbrowser` repozytorium znajduje się wykonywalny skrypt `run.sh`. Nie przyjmuje on żadnych parametrów, a jego uruchomienie powoduje uruchomienie procesu scrapera. W celu cyklicznego uruchamiania, zalecamy skorzystanie z menadżera *cron*.

W pliku `jobsbrowser/jobsbrowser/settings.py` znajdują się ustawienia programu. Większość z nich to ustawienia framework'a Scrapy. Ich opis znaleźć można w jego dokumentacji. (*Tutaj bibliografia*). Ustawieniami dotyczącymi konkretnie tego projektu są `STORAGE_SERVICE_ADD_URL` oraz `STORAGE_SERVICE_RETRIEVE_URL`, które oznaczają adresy URL pod który program może wykonać żądania w celu odpowiednio, dodania nowo przetworzonej strony oraz uzyskania listy adresów URL ofert których nie powinien przetwarzać.

W celu uruchomienia testów w konsoli należy wpisać polecenie `pytest`.



Rysunek 2.3: Schemat architektury modułu Scraper'a.



Rysunek 2.4: Diagram klas modułu Scraper'a.

2.1.5 Struktura kodu źródłowego

Poniżej prezentujemy drzewo katalogów oraz plików projektu wraz z krótkim omówieniem:

```
jobsbrowser
    jobsbrowser
        __init__.py
        items.py          # definicje klas przechowujących zebrane dane
        loaders.py        # definicje klas ładujących zebrane dane
        middlewares.py
        pipelines.py      # definicje kolejnych etapów przetwarzania danych
        settings.py       # ustawienia pająków
        spiders           # katalog z definicjami pająków zbierających dane
            __init__.py
            pracuj.py     # definicja pająka zbierającego dane ze strony pracuj.pl
    tests/                # katalog z testami projektu
    run.sh                # plik wykonywalny uruchamiający proces zbierania danych
    scrapy.cfg            # konfiguracja całego projektu
requirements.txt          # plik z wymaganiami projektu
```

2.1.6 Główne klasy modułu

- **JobBrowserItem** - klasa bazowa definiująca pola które powinny być dostarczane przez wszystkie spider'y wykorzystywane w projekcie.
- **PracujItem** - klasa przechowująca pola które muszą zostać wypełnione przez scrapery zbierające dane ze strony pracuj.pl.
- **JobsBrowserPipeline** - klasa która po zebraniu danych ze stron wysyła dane do modułu zapisy do bazy danych.
- **JobsBrowserSpiderMiddleware** - klasa wygenerowana automatycznie podczas tworzenia projektu za pomocą Scrapy'ego.
- **PracujItemLoader** - klasa przetwarzająca w prosty sposób zebrane dane.
- **PracujSpider** - klasa wykonująca zapytania do serwisu pracuj.pl oraz zbierająca dane.

Historia zmian dokumentu

Tabela 2.1: Historia Zmian

Data	Autor	Opis zmian	Wersja
04.11.2017	Bartłomiej Sielicki Łukasz Skarżyński	Dodanie harmonogramu.	1.1
18.11.2017	Bartłomiej Sielicki Łukasz Skarżyński	Dodanie specyfikacji technicznej.	1.2

Anon, Scrapy.