

Uniwersytet Jagielloński w Krakowie  
Wydział Fizyki, Astronomii i Informatyki Stosowanej

Wojciech Sękowski

Nr albumu: 1155177

# Mikroserwisowa platforma powiadomień o ofertach sprzedażowych

Praca magisterska  
na kierunku Informatyka

Praca wykonana pod kierunkiem  
dra inż. Marcina Zielińskiego  
Instytut Fizyki im. M. Smoluchowskiego

Kraków 2023

*Bardzo dziękuję Panu doktorowi  
inżynierowi Marcinowi Zielińskiemu  
za wszelką pomoc oraz rady udzielone  
w trakcie powstawania pracy*

# Spis treści

<b>1</b>	<b>Wstęp</b>	<b>4</b>
<b>2</b>	<b>Proces ekstrakcji i przetwarzania danych</b>	<b>6</b>
2.1	Proces webscrapingu . . . . .	6
2.2	Praktyki oraz techniki webscrapingu . . . . .	9
2.3	Narzędzia webscrapingu . . . . .	11
2.4	Narzędzia ETL . . . . .	14
<b>3</b>	<b>Powiadomienia jako źródło informacji</b>	<b>16</b>
3.1	Rola powiadomień w platformach sprzedażowych . . . . .	16
3.2	Typy powiadomień i ich znaczenie . . . . .	17
3.3	Mechanizm tworzenia powiadomień . . . . .	19
<b>4</b>	<b>Architektura mikroserwisowa</b>	<b>21</b>
4.1	Opis architektury mikroserwisowej . . . . .	21
4.2	Architektura mikroserwisowa w kontenerach . . . . .	24
<b>5</b>	<b>Implementacja mikroserwisowej platformy powiadomień</b>	<b>26</b>
5.1	Wykorzystane technologie . . . . .	26
5.2	Architektura platformy . . . . .	28
5.2.1	Wprowadzenie wzorca MVC do architektury oprogramowania . . . . .	30
5.3	Elementy implementacji . . . . .	32
5.3.1	Aplikacja kliencka . . . . .	32
5.3.2	Aplikacja serwerowa . . . . .	35
5.3.3	Architektura mikroserwisowa w Node.js . . . . .	44
5.3.4	Komunikacja między kontenerami . . . . .	46
<b>6</b>	<b>Testy i ocena aplikacji</b>	<b>50</b>
6.1	Badanie aplikacji pod kątem defektów: testowanie oraz analiza błędów . . . . .	50
6.2	Ocena efektywności platformy . . . . .	51
<b>7</b>	<b>Podsumowanie</b>	<b>53</b>



# Rozdział 1

## Wstęp

Obecnie informacja czy też komunikacja jest jednym z głównych czynników, który powoduje ciągle rozrastanie się wielu sektorów gospodarki. Ilość informacji, które ludzie konsumują i przetwarzają rośnie z dnia na dzień, więc ważne jest, aby mieć narzędzia, które niemal w czasie rzeczywistym pozwalają na informowanie o danej rzeczy w określonym czasie. Dlatego też systemy powiadamiające są bardzo istotnym elementem ekosystemu wymiany i przepływu informacji.

Dzięki rozwojowi wielu platform sprzedażowych oraz procesów związanych z szeroko pojętym rynkiem *e-commerce*<sup>1</sup> pojawia się coraz więcej rozwiązań będących odpowiedzią na prośby klientów chcących mieć podgląd do swoich produktów w jak najlepszym czasie oraz z jak największą ilością informacji dotyczących konkretnego towaru. W związku z tym, że istnieje coraz więcej takich rozwiązań zwiększa się również ich złożoność oraz utrudniony zostaje dostęp do zarządzania takimi platformami.

Celem niniejszej pracy jest przedstawienie problematyki mechanizmów wysyłających powiadomienia do użytkowników w czasie, w jakim oni sami sobie ustalą. Aplikacja oparta na architekturze mikroserwisowej będzie pobierała dane z różnych serwisów sprzedażowych tj. *OLX*, *Amazon*, *Allegro* czy *Pepper*, na podstawie frazy wpisanej przez użytkownika oraz liczby odpowiedzi, jakie ma zwrócić aplikacja. Następnie będzie ona szukała najnowszych ofert danego produktu i wysyłała powiadomienie typu SMS, E-mail, informacje do serwisu *Discord* lub będzie wyświetlała na bieżąco dane informacje w zależności od konfiguracji użytkownika.

Trzy początkowe rozdziały są wprowadzeniem do tematu oraz omówieniem zagadnień związanych z pobieraniem danych ze stron internetowych, wysyłaniem powiadomień, architekturą mikroserwisową oraz zbiorem informacji teoretycznych wprowadzających czytelnika w powyższe zagadnienia. Piąty rozdział opisuje etapy procesu projektowania, planowania oraz opisu poszczególnych komponentów platformy. Omawia model danych, architekturę systemu oraz analizę wymagań. Została tu opisana kolejno implementacja poszczególnych rozwiązań oraz części związane z pisanem kodu źródłowego. W rozdziale szóstym opisano ocenę platformy oraz wnioski jakie wypłynęły podczas pisania pracy. W rozdziale tym zawarto także przeprowadzenie testów i analizę uzyskanych wyników. W ostatnim rozdziale omówiono po-

---

<sup>1</sup>E-commerce - handel elektroniczny.

szczególne wady i zalety opisywanego rozwiązania oraz podsumowanie punktów, jakie udało się zrealizować, a także sugestie do dalszego rozwoju pracy.

## Rozdział 2

# Proces ekstrakcji i przetwarzania danych

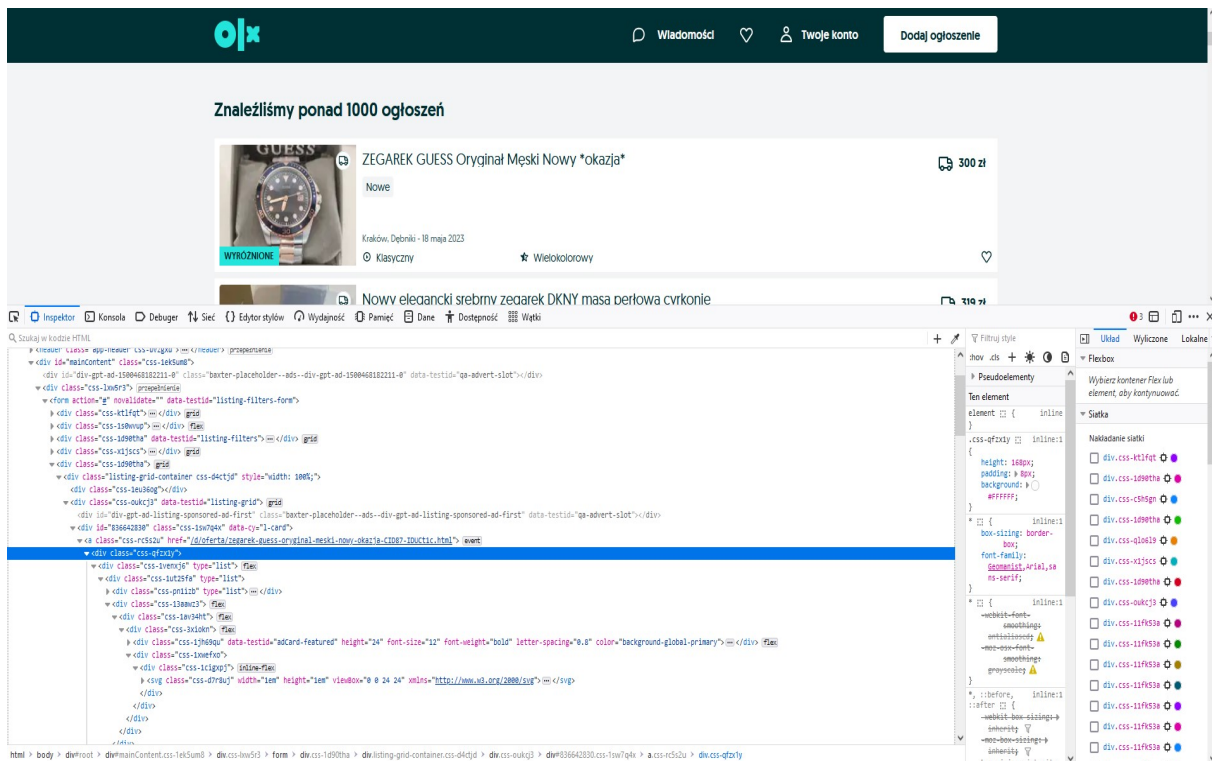
Na każdej witrynie internetowej, na którą wchodzi użytkownik widać zazwyczaj wiele danych. Są one zwracane przez przeglądarkę w postaci kodu *HTML*. Firmy sprzedażowe konkurują ze sobą w pozyskiwaniu coraz to większej ilości danych, tak aby uzyskać przewagę nad innymi. Zazwyczaj dane te są dostępne w różnych formach i mają inną strukturę oraz hierarchię. Niekiedy zdarza się, że niektóre dane są dostępne tylko w formie online. Mogą być one zarówno aktualne jak i nieaktualne. Gdy przegląda się daną stronę internetową to silnik przeglądarki przetwarza zawartość takiej strony, czyli głównie dane (tekst, grafikę, skrypty itd.), a następnie renderuje wynikowy format strony, który jest widoczny dla użytkownika. Jednak nie trzeba wchodzić na daną stronę internetową, aby zobaczyć jej zawartość, gdyż są dostępne narzędzia webscrapingu (jest to proces automatycznego pobierania danych z witryny internetowej w celu dalszej obróbki i analizy) [1], które mogą w tym zadaniu pomóc.

### 2.1 Proces webscrapingu

Proces ten służy do wyłuskiwania konkretnych zawartości wyznaczonych przez użytkownika ze struktury kodu *HTML*. Zazwyczaj wykorzystuje się tutaj określone boty/roboty służące do automatycznego procesu pobierania danych, zamiast robić to ręcznie. Każda strona udostępnia swoje oryginalne treści, zatem są one unikalne, dlatego również ich struktura jest inna. Do pobierania danych ze stron internetowych stosuje się dedykowane oprogramowanie udostępnione w Internecie (darmowe lub płatne). Jednak często developerzy decydują się na napisanie własnych skryptów, dzięki czemu mają pełną kontrolę nad ich działaniem [2].

Generalna zasada działania takiego programu opiera się na standardowych mechanizmach protokołu HTTP przy pomocy których, pobierany jest kod źródłowy danej witryny (przedstawiono na Rys. 2.1), składającej się ze znaczników *HTML*. Następnie program ten filtruje stronę w celu znalezienia interesującej zawartości. Często też uruchomienie programu do pobierania danych jest szybsze niż włączenie strony internetowej i wyrenderowanie jej przez

przeglądarkę, gdyż w tym przypadku pomijamy etap ładowania dostępnej zawartości strony.



Rysunek 2.1: Przykładowy kod źródłowy strony serwisu OLX <sup>1</sup>.

W procesie webscrapingu pobierane dane mogą być zapisane w postaci ustrukturyzowanej np. plików: *CSV*, *XLSX*, *JSON*, *XML*, lub zapisywane do bazy danych (przedstawiono na Rys. 2.2). Dzięki takiemu rozwiązaniu można z łatwością analizować dane czy też sporządzać statystyki lub raporty dotyczące konkretnego zagadnienia.

<sup>1</sup>Adres strony w zapytaniu o zegarek w serwisie OLX: <https://www.olx.pl/krakow/q-zegarek>.





Rysunek 2.2: Proces webscrapingu ukazujący pobranie danych i zapisanie ich do miejsca składowania (zaczerpnięto z [3]).

Często zdarza się, że użytkownik chce pobrać odnośniki (hiperłącza) z danej strony, tak aby mieć dostęp do innych danych. Proces w którym pobierane są dane ze strony internetowej w celu wyodrębnienia odnośników (hiperłączy) określany jest jako indeksowanie (webcrawling) [4]. Proces ten nazywa się indeksowaniem, ponieważ znalezione dane przechowywane są w indeksie, czyli w strukturze, która zwiększa szybkość wyszukiwania lub dostęp do określonych danych. Dobrym przykładem realizacji takiego procesu jest wyszukiwarka *Google*, z której usług korzystało aż 93% stron internetowych w 2020r [5]. Zatem różnica między webscrapingiem, a webcrawlingiem polega na tym, że webscraping wyodrębnia dane ze strony, podczas gdy webcrawling znajduje linki (adresy *URL*) na danej stronie.


Internet jest niesamowitym zbiorem informacji, stąd też dostęp do tych danych wiąże się często z technikami scrapowania. Poniżej przedstawionych zostało kilka zastosowań webscrapingu:

- ciągłe analizowanie rynku i tworzenie statystyk dotyczących konkurencji,
- prognozowanie cen różnych produktów i ich monitorowanie, zbieranie biznesowych informacji,
- agregacja treści z wielu źródeł w jedno miejsce,
- analiza opinii, komentarzy, recenzji klientów w oparciu o dany produkt oraz prognozowanie dalszych korzystnych lub niekorzystnych wyników,
- stworzenie porównywarek cenowych pobierających aktualne ceny i informacje o produkcie,
- działanie robotów, które ustalają ranking widoczności stron.

## 2.2 Praktyki oraz techniki webscrapingu

Poniżej przedstawiono kilka praktyk dotyczących webscrapingu, którymi warto się kierować podczas wykonywania procesów związanych z pobieraniem danych[4]:

1. Gdy dana strona udostępnia interfejs *API*<sup>2</sup> to warto go wykorzystać. Świetnym przykładem tutaj może być dokumentacja *API Allegro* : <https://developer.allegro.pl/>.
2. Większość stron udostępnia plik *robots.txt* (przedstawiono na Rys. 2.3), w którym zapisane są informacje dla automatów (robotów internetowych) jakie operacje mogą wykonywać na stronie, a jakich nie powinny. Przed przystąpieniem do scrapowania ważne jest, aby najpierw przejrzeć podany plik, żeby wiedzieć co scrapować. Linie w pliku *robots.txt* z dyrektywą "Disallow: ..." mówią czego nie scrapować ze strony, zaś linie z dyrektywą "Allow: ..." opisują co możemy scrapować.



```
# sitecode:olxpl-desktop
Host: https://www.olx.pl
Sitemap: https://www.olx.pl/sitemap.xml
User-agent: *
Disallow: */ajax/
Disallow: /adminpanel/
Disallow: /api/
Disallow: */facebook/
Disallow: */rss/
Disallow: */konto/
Disallow: */mojolx/
Disallow: /drukuj/
Disallow: /oferta/ulotka/
Disallow: /oferta/kontakt/
Disallow: /platnosci/
Disallow: /nowe-ogloszenie/confirm/
Disallow: /nowe-ogloszenie/confirmpage/
Disallow: /i2/oferta/abuse/
Disallow: /m/oferta/abuse/
Disallow: /i2/oferta/kontakt/*
Disallow: /api/open/oauth/token/
Allow: */api/v1/seo/listings/*
Allow: /api/v1/seo/offers/
Allow: /api/v1/offers/
Allow: /api/v1/targeting/
Allow: /api/v1/friendly-links/
Allow: /
```

Rysunek 2.3: Przykładowy plik *robots.txt* z serwisu *OLX*.

3. Gdy dana witryna udostępnia swój regulamin lub warunki użytkowania (*TOS*<sup>3</sup>) to warto się z nimi zapoznać i je przestrzegać.
4. Należy zachować zgodność z rozporządzeniem o ochronie danych osobowych *UE* (*RODO*).

---

<sup>2</sup>*API*- interfejs programistyczny aplikacji.

<sup>3</sup>*TOS* - Term of service, warunki korzystania z usługi.

5. Warto ograniczyć ilość zapytań wykonywanych do strony, tak aby zmniejszyć ryzyko obciążenia serwera i nie zakłócać prawidłowego działania strony. Przykładowo lepiej jest scrapować w nocy zamiast w ciągu dnia, wtedy serwery są mniej obciążone.
6. Każdy serwis publikuje na swój sposób unikatowe treści, zatem jeżeli stworzy on jakieś dzieło to jest jego właścicielem. Praktyka webscrapingu odnośnie nie naruszania praw autorskich polega na zbieraniu danych ze stron internetowych w sposób zgodny z obowiązującymi prawami i regulacjami dotyczącymi własności intelektualnej.
7. Dobrą praktyką jest identyfikacja użytkownika poprzez podanie zmiennej *User-Agent*<sup>4</sup> w nagłówku zapytania do strony.

Istnieje wiele różnych sposobów pobrania danych z sieci [6]. Są one głównie uzależnione od struktury danej strony, np.

- ręczne kopiowanie i wklejanie zawartości strony do źródła danych,
- parsowanie *HTML* - wyodrębnianie danych zarówno ze stron statycznych jak i dynamicznych za pomocą żądań *HTTP*, przykładem mogą być tutaj takie parsery *HTML* jak biblioteka *BeautifulSoup*,
- wykorzystanie wyrażeń regularnych w celu dopasowania wzorca do tekstu,
- parsowanie *DOM* (Obiektowy model dokumentu)<sup>5</sup> - struktura *HTML* jest sprowadzana do hierarchii drzewa *DOM*. Często jest tutaj stosowany język taki jak *Xpath*<sup>6</sup>, czy użyte są selektory *CSS*, służące do przeanalizowania wynikowego drzewa *DOM*.

Scrapowanie stron internetowych może być uznane za nielegalne. Gdy dany serwis jest scrapowany pobierane z niego dane mogą być chronione prawem autorskim. Dlatego ze względu na częste łamanie prawa administratorzy sieciowi zazwyczaj stosują różne środki uniemożliwiające działanie takiemu robotowi. Oto przykłady kilku z nich :

- blokowanie botów ze względu na wykonywaną zbyt dużą ilość zapytań do strony,
- blokowanie adresów IP, z których przychodzi zapytanie do strony (czarna lista adresów),
- ustawienie honeypota, czyli pułapki mającej na celu przyciągnięcie i monitorowanie nieautoryzowanych lub złośliwych działań w systemie, gdzie zapytanie do strony zostaje przekierowane w inne miejsce[7],
- stosowanie mechanizmów *CAPTCHA*, czyli techniki, w której dopuszcza się przesyłanie danych wyłącznie przez człowieka, ma to służyć do dodatkowej weryfikacji ze strony użytkownika[8].

---

<sup>4</sup>*User-Agent* - nagłówek służący serwisom do identyfikacji programu klienckiego.

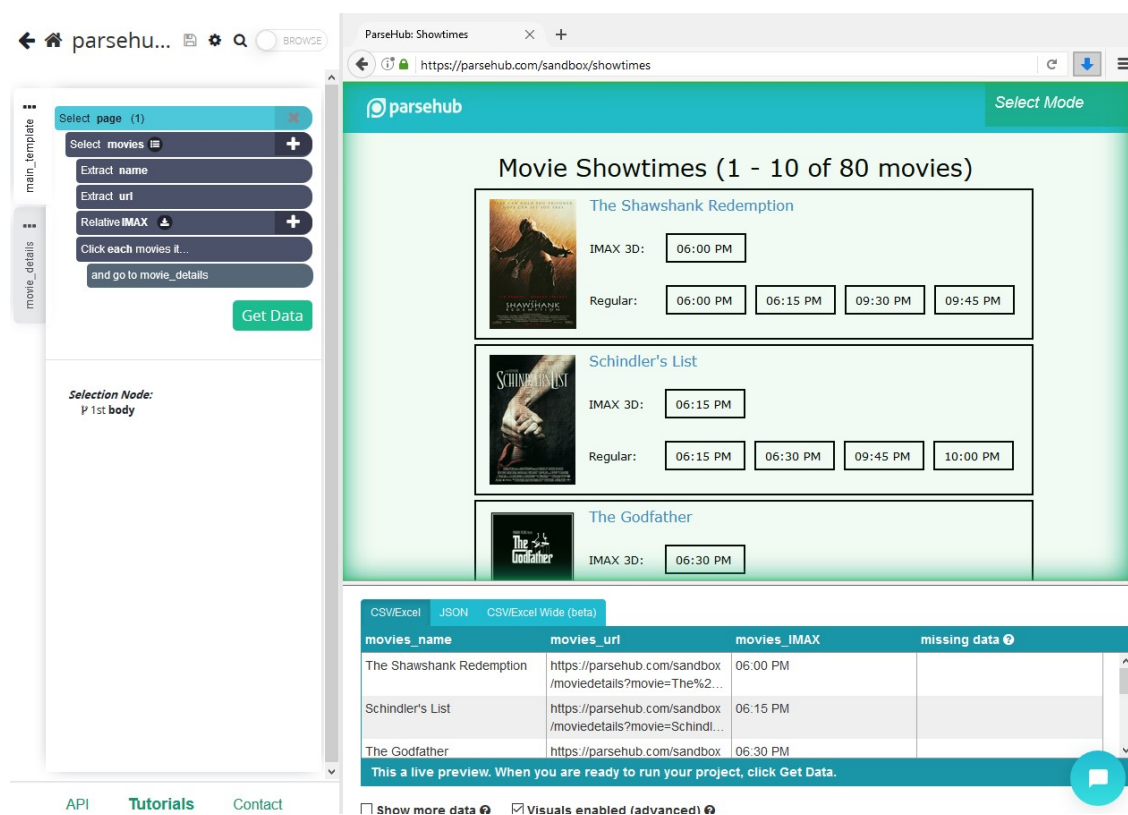
<sup>5</sup>*DOM* - Document Object Model, sposób prezentacji dokumentów *HTML* w postaci obiektowego modelu.

<sup>6</sup>*Xpath* - język zapytań używany podczas pobierania i ekstrakcji danych z węzłów dokumentów *XML* czy *HTML*.

## 2.3 Narzędzia webscrapingu

Na rynku istnieje wiele narzędzi wspomagających procesy webscrapingu. Narzędzia te mogą być rozbudowane pod kątem dostępnych usług. Większość z nich jest darmowa ze względu na podstawowe procesy ale, aby uzyskać dostęp do zaawansowanych usług trzeba za nie dodatkowo zapłacić. Poniżej przedstawione zostanie kilka najbardziej popularnych rozwiązań na rynku.

1. *ParseHub*<sup>7</sup> - narzędzie do webscrapingu z interfejsem *GUI*<sup>8</sup>, pozwala na stworzenie szkicu, dzięki któremu można łatwo wydobyć dane i wyeksportować je w różnych formatach. Pobiera ono dane z dynamicznych (interaktywnych) stron, mimo wyskakujących okienek, czy też obsługuje strony z nieskończonym przewijaniem. Wystarczy otwarcie strony i kliknięcie w dane elementy, które chcemy wyodrębnić. W celu użycia *ParseHuba* należy pobrać aplikację desktopową, która posiada wbudowaną, zintegrowaną przeglądarkę. Narzędzie jest intuicyjne przez co nie wymaga dużej wiedzy ze strony użytkownika.



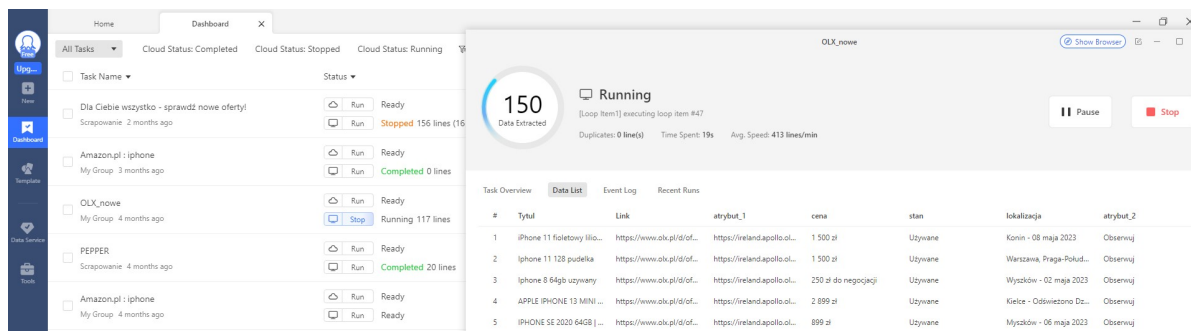
Rysunek 2.4: Interfejs graficzny aplikacji *ParseHub*.

Dane są przechowywane w serwisie przez 14 dni. Występuje także limit maksymalnie dwustu stron na przebieg (stan na dzień 2023.05.22). Program oferuje zapis danych w formacie *JSON* lub *XLSX*. Na Rys. 2.4 przedstawiono graficzny interfejs aplikacji *ParseHub*.

<sup>7</sup>Strona projektu *ParseHub*: <https://www.parsehub.com/>.

<sup>8</sup>*GUI* -graficzny interfejs użytkownika.

2. *Octoparse*<sup>9</sup> - wydajne oprogramowanie służące do webscrapingu, które oferuje interfejs *GUI* (pokazano na Rys.2.5) oparty na chmurze. Umożliwia on użytkownikowi tworzenie reguł w celu pobrania danych ze strony bez większej wiedzy na temat programowania. Automatyzuje proces ekstrakcji danych. Za pomocą dosłownie kilku kliknięć pozwala na wyciągnięcie poszukiwanych treści. Działa podobnie jak przeglądarka, dzięki czemu otwiera strony i naśladuje zachowania użytkownika podczas przeglądania danych w sieci, takie jak: kliknięcie w linki, otwieranie formularzy, wpisywanie tekstu.



Rysunek 2.5: Interfejs graficzny aplikacji *OctoParse*.

Oprogramowanie to umożliwia także działanie w dwóch trybach: ekstrakcja lokalna (działanie na własnym urządzeniu) lub ekstrakcja zdalna (działanie w chmurze). W darmowym trybie zapewnia ono utworzenie maksymalnie dziesięciu zadań, włączenie ekstrakcji lokalnej, ograniczenie do 10000 wierszy na eksport czy też ograniczone wsparcie. Program pozwala na stworzenie harmonogramu do wykonywania zadań i zapewnia zapis do plików: *JSON*, *XLSX*, *HTML*, *TXT* lub bezpośredni eksport do baz danych takich jak *MySQL* czy *Microsoft SQL Server*. Oprogramowanie zaliczane jest do narzędzi klasy *ETL*, czyli procesu składającego się z trzech części, który wspomaga pozyskanie danych z różnych źródeł i umieszczenie ich w jednym miejscu. Narzędzia *ETL* zostaną omówione w następnej sekcji.

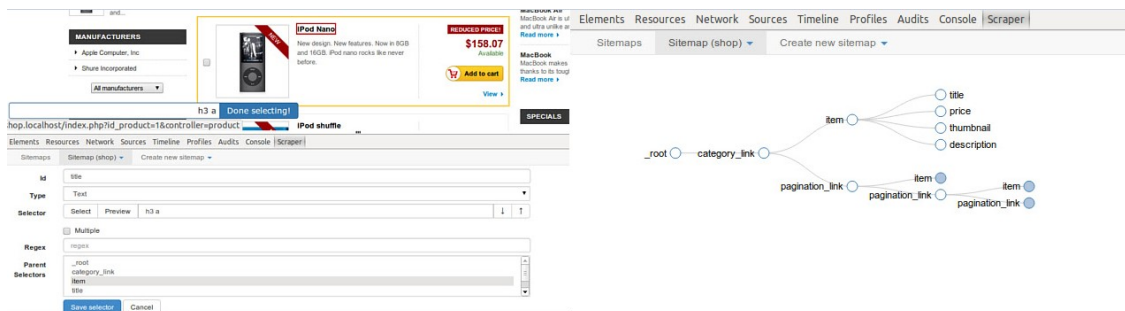
3. *Web Scraper extension*<sup>10</sup> - jest to dodatek do zainstalowania dla przeglądarki *Google Chrome*, dzięki któremu w łatwy i szybki sposób można wyselekcjonować dane na dowolnej stronie. Narzędzie to zapewnia łatwy dostęp do elementów, które chce się pobrać poprzez kliknięcie w dany element. Wtyczka daje możliwość pobrania treści nawet z dynamicznych stron internetowych wykorzystujących asynchroniczną interakcję z użytkownikiem. Aby stworzyć plan ekstrakcji danych ze strony najpierw tworzy się mapę strony (sitemap) tzw. szkic. Proces tworzenia szkicu dla pobrania danych ze strony polega na wybraniu odpowiednich selektorów dla elementów, dla których te dane będą pobierane.

Dodatek działa wyłącznie wtedy kiedy dana zakłada strony pozostaje otwarta. Pozwala on wyeksportować dane do formatu *CSV*, *XLSX* oraz *JSON*. Występuje również wersja płatna wraz z innymi funkcjonalnościami. Widok programu przedstawiono na Rys. 2.6.

#### 4. Narzędzia języka *Python*:

<sup>9</sup>Strona projektu *Octoparse*: <https://www.octoparse.com/>.

<sup>10</sup>Strona projektu *Web Scraper extension*: <https://webscraper.io/>.



Create element selectors

Build sitemaps from multiple selectors

category_link	category_link-href	title	price	th
Laptops	http://shop.localhost/index.php?id_category=5&controller=category	MacBook Air	504.18	htt
Laptops	http://shop.localhost/index.php?id_category=5&controller=category	MacBook	170.57	htt
Accessories	http://shop.localhost/index.php?id_category=4&controller=category	Belkin Leather Folio for iPod...	25.04	htt
Accessories	http://shop.localhost/index.php?id_category=4&controller=category	Shure SE210 Sound-Isolating...	124.58	htt
iPods	http://shop.localhost/index.php?id_category=3&controller=category	iPod Nano	158.07	htt
iPods	http://shop.localhost/index.php?id_category=3&controller=category	iPod shuffle	66.05	htt
iPods	http://shop.localhost/index.php?id_category=3&controller=category	iPod touch 2	241.64	htt
iPods	http://shop.localhost/index.php?id_category=3&controller=category	iPod Nano 2	158.07	htt

Browse and export scraped data

Rysunek 2.6: Widok z programu *Web Scraper* (zaczepnięto z [9]).

- pakiet *Scrapy*<sup>11</sup> - wydajne narzędzie stworzone do ekstrakcji danych z witryn internetowych. Zapewnia narzędzia potrzebne do napisania programów przeszukujących strony internetowe. Jest szybki i wydajny, dzięki czemu programy mogą pobierać dane z wielu stron jednocześnie. Dzięki językowi *Python* pakiet ten jest łatwo rozszerzalny, przez co ciągle można go edytować i dodawać kolejne funkcjonalności. Jest dostępny na platformie zarówno *Windows*, *Linux* jak i *Mac*. Instalacja narzędzia odbywa się za pomocą menedżera pakietów *pip*, przedstawiono we fragmencie kodu 2.1:

```
1 pip install Scrapy
```

Listing 2.1: Instalacja pakietu *Scrapy*.

- biblioteki *Requests* oraz *BeautifulSoup* - te dwie biblioteki służą do pobierania danych z internetu. Biblioteka *Requests* zapewnia wysyłanie żądań do stron internetowych oraz komunikację z nimi poprzez protokół *HTTP*, zaś biblioteka *BeautifulSoup* zapewnia przetwarzanie pobranej treści przez *Requests* oraz wyodrębnianie potrzebnych danych. Tak jak w powyższym przypadku do instalacji bibliotek służy także menedżer pakietów *pip*, przedstawiono na wydruku kodu 2.2:

```
1 pip install requests
2 pip install beautifulsoup4
```

Listing 2.2: Instalacja pakietów *Requests* oraz *BeautifulSoup*.

<sup>11</sup>Strona projektu *Scrapy*: <https://scrapy.org/>.

## 2.4 Narzędzia ETL

Dzisiejsze firmy zmagają się z potrzebą pobierania danych z Internetu oraz z ich analizą i przechowywaniem. Dane zazwyczaj są rozproszone w różnych systemach lub bazach danych. Ich agregacja i dalsze wykorzystanie może przynieść firmie duże korzyści finansowe. Do agregacji i przetwarzania danych wykorzystywane są narzędzia *ETL*. *ETL* jest skrótem od trzech słów *Extract* - ekstrakcja danych, *Transform* - przekształcenie danych i *Load* - załadowanie danych do źródła. Są to trzy kluczowe etapy procesu przetwarzania danych. Dane mogą być pobierane z jednego lub kilku źródeł jak i też wysyłane do jednego lub wielu miejsc. Przede wszystkim narzędzia *ETL* te automatyzują proces przetwarzania danych, więc ich czynności mogłoby wykonywać kilka programów.

*ETL* to narzędzia wspomagające procesy uzyskiwania danych dla baz danych, w szczególności dla hurtowni danych. Stanowią element procesów z zakresu *BI* - *Business Intelligence* [10]. Procesy *ETL* są często nazywane procesami integracji danych, ponieważ następuje tutaj przetwarzanie danych.

W *ETL* najpierw następuje etap ekstrakcji danych, w którym pobierane są dane z różnych źródeł takich jak: bazy danych, katalogi danych, strony internetowe, *API*, języki *RSS*, pobieranie danych z plików i wiele więcej. Jest to jeden z ważniejszych etapów, ponieważ program ten zapoznaje się z danymi, ich jakością oraz ich źródłem. Często może występować tutaj problem *GIGO*<sup>12</sup>, mówiący o tym, że gdy dane na początkowym etapie będą słabej jakości to na końcu otrzyma się również mało jakościowe dane.

Kolejny etap to transformacja danych, w którym dane są czyszczone, odseparowywane, usuwane są duplikaty, a także przekształcane zostają do czytelniejszych formatów. Stosowane są tu operacje filtrowania czy też agregowania danych. Można również używać takich operacji jak zamiana wierszy z kolumnami lub na odwrót. Po zakończeniu tego etapu dane są już zazwyczaj w ustrukturyzowanej formie.

Ostatnim etapem jest etap ładowania danych do określonego źródła. Może to być określona baza danych lub inny magazyn docelowy (typu hurtownia danych), gdzie dostępne są dane dla użytkowników poprzez różne narzędzia analityczne. W tym procesie dane mogą być ładowane w zaplanowanych godzinach lub w czasie rzeczywistym. Na Rys. 2.7 pokazano przepływ danych w narzędziach *ETL*. Bezpośrednimi korzyściami wykorzystania narzędzi *ETL* są: przygotowanie danych do dalszych analiz, zebranie ujednoliconych danych, posiadanie jednego, dobrego jakościowo źródła informacji, generowanie przewidywań/predykcji w biznesie, analizowanie zachowań klientów w oparciu o zbierane dane.

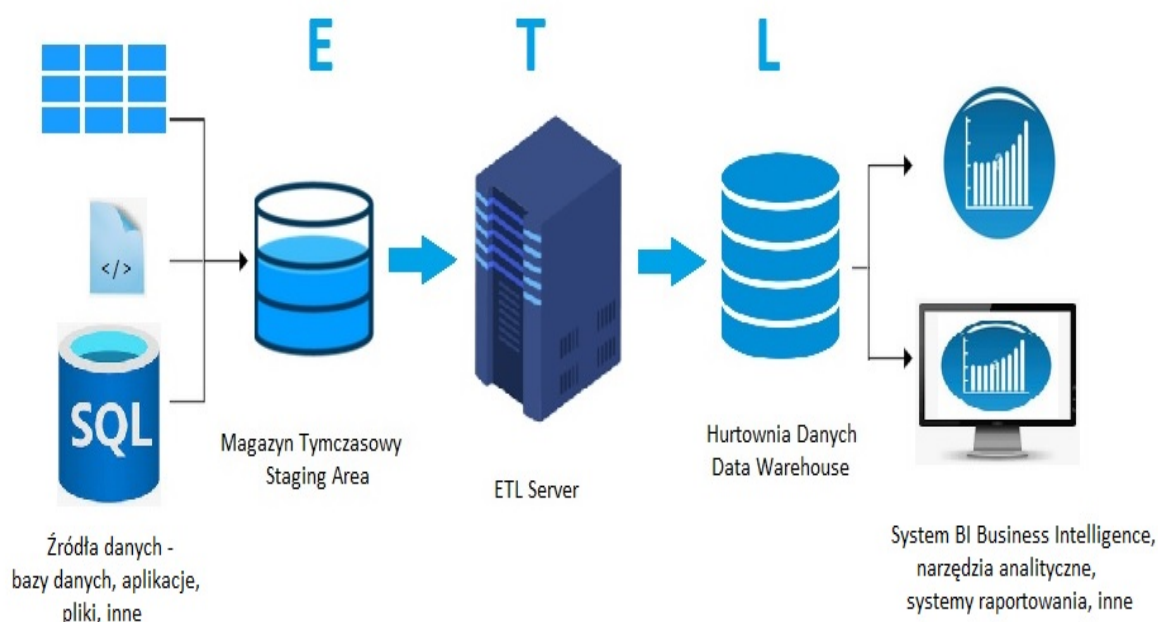
W sytuacji pracy z bardzo dużą ilością danych lub procesem bezpośredniego dostępu do nieprzetworzonych danych wtedy mówimy o procesie *ELT*, czyli jest to skrót od słów *Extract*, *Load*, *Transform*, gdzie w odróżnieniu od procesów *ETL* dane nie są przekształcane po pobraniu, lecz bezpośrednio ładowane do bazy danych [11]. Proces wygląda bardzo podobnie do procesu *ETL*, z tym, że transformację i obróbkę danych zostawia na koniec. W niektórych przypadkach może to przyspieszyć cały proces, gdyż w drugim etapie pomija się proces prze-

---

<sup>12</sup>GIGO - Garbage in, garbage out.



kształcania i przenosi się go na koniec. Poniżej wypisanych jest kilka najpopularniejszych narzędzi *ETL* [12]: IBM Websphere DataStage, Oracle Warehouse Builder, Informatica PowerCenter, Microsoft SQL Server Integration Services (SSIS), Octoparse (wspomniany wcześniej), działający jak *ETL* i wykonujący również rolę webscrapera.



Rysunek 2.7: Przepływ procesu *ETL*, w którym dane są pobierane ze źródeł takich jak bazy danych, pliki i inne, a następnie są przenoszone do magazynu tymczasowego i transformowane przez serwer *ETL*. Po tym procesie są ładowane do hurtowni danych, z której korzystają różne narzędzia analityczne lub systemy raportowania (zaczerpnięto z [13]).



## Rozdział 3

# Powiadomienia jako źródło informacji

Podstawowym celem powiadomień jest przekazanie informacji o określonym zdarzeniu. Obecnie powiadomienia mogą mieć charakter informacyjny, alarmowy, interaktywny czy aktualizacyjny. Ludzie zasypywani są dużą ilością informacji i powiadomień, przez co często tracą skupienie na temacie, nad którym pracują. Powiadomienia pełnią dużą rolę w funkcjonowaniu społeczeństwa informując ludzi o bieżącym stanie konkretnej rzeczy.

### 3.1 Rola powiadomień w platformach sprzedażowych

W technologii system powiadomień to zazwyczaj połączenie sprzętu (hardware) oraz oprogramowania (software) w celu przekazania dalej informacji. Przez media społecznościowe dostarczane są komunikaty powiadomień dotyczących odniesień do ich komentarzy, polubień czy obserwacji. Komunikaty mogą mieć formę alarmową (zazwyczaj), ostrzegawczą czy nawet edukacyjną.

Powiadomienia to po prostu krótkie komunikaty wysłane do użytkownika końcowego mające na celu dostarczyć wskazaną wiadomość [14]. Odgrywają one dużą rolę w ofertach sprzedażowych. Mogą informować użytkowników o bieżących zmianach dotyczących danego produktu, takich jak: status zamówienia, promocja czy dostępność produktu. Przez to zwiększa się zaangażowanie klienta i jego satysfakcja. Klient może podjąć kolejne kroki takie jak zakup produktu czy otrzymanie informacji o niskim stanie towaru. Dzięki takim powiadomieniom może zwiększyć się sprzedaż danego produktu. Poszczególne powiadomienia zazwyczaj są personalizowane dla konkretnej osoby, tak aby skupiły one uwagę. Takie powiadomienia można wysłać po sprawdzeniu historii zakupów, preferencji zakupowych czy miejsca, w jakim najczęściej znajduje się osoba kupująca. Działania te skupiają się na określonych grupach odbiorców, tak aby uzyskać jak najlepsze wyniki sprzedażowe. Wiadomości mogą służyć też poprawie jakości obsługi klienta np. poprzez dostarczanie informacji o planowej dacie dostawy czy ewentualnych opłatach. Ponadto powiadomienia mogą być rodzajem pewnego przypo-

mnienia, np. o nadchodzących płatnościach, trwających pracach w danym systemie, przerwach w dostawie towaru.

## 3.2 Typy powiadomień i ich znaczenie

Powiadomienia mogą przybierać różnorakie formy, między innymi takie jak: banery, modale<sup>1</sup>, listy, slajdy, obrazy. Występuje podział na kilka rodzajów powiadomień [15]:

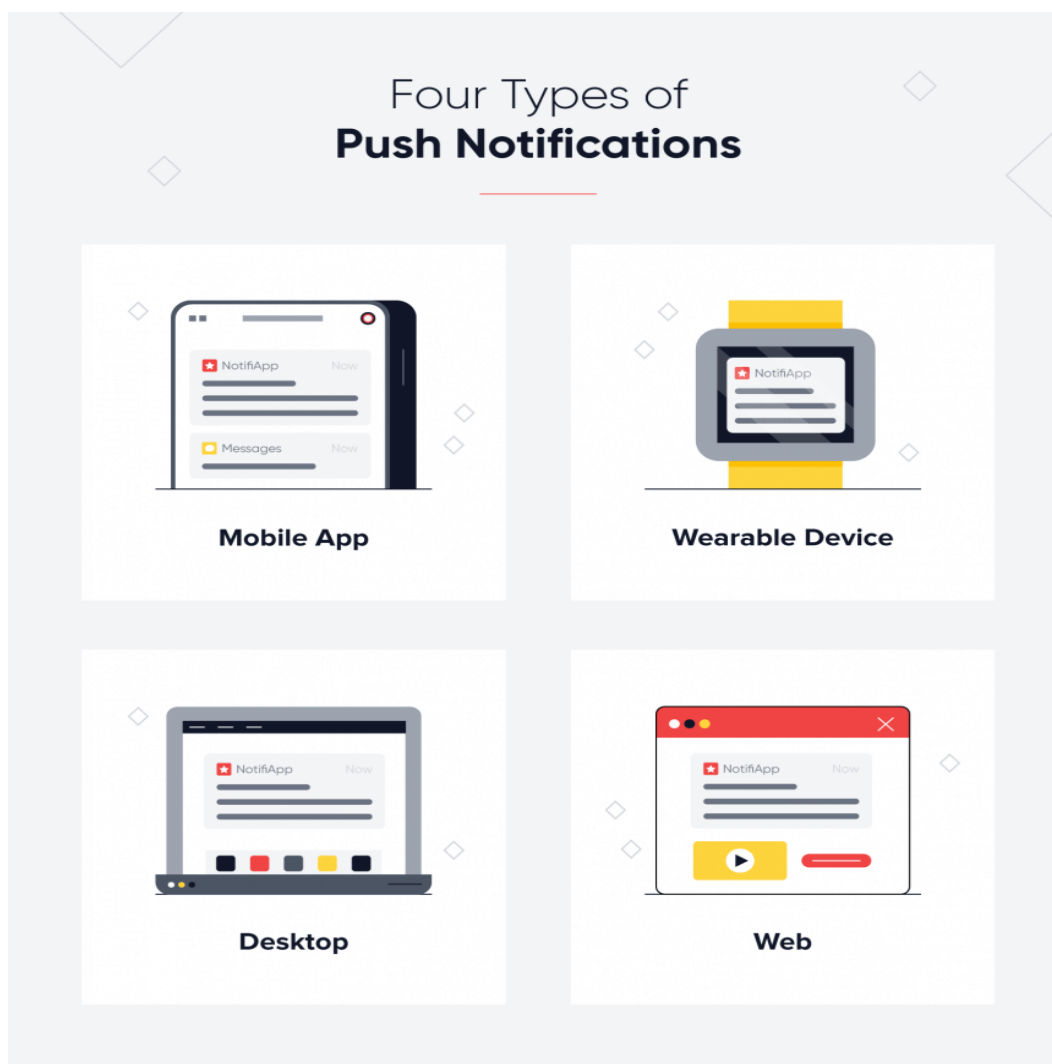
1. Powiadomienia natywne lub systemowe są wysyłane przez systemy operacyjne smartfonów, komputerów czy też urządzeń elektronicznych. Mogą to być komunikaty związane z bieżącymi aktualizacjami czy też przypomnieniami o nadchodzących alarmach jak i nieodebranych połączeniach.
2. Powiadomienia społecznościowe generowane przez platformy takie jak *Facebook*, *Messenger*, *Twitter*, *Instagram*, *Discord*, itd. Są one związane z powiadomieniami o nowych wiadomościach, polubieniach, czy też oznaczeniach użytkownika w danym kontekście.
3. Powiadomienia SMS - krótkie wiadomości tekstowe przesyłane na urządzenia mobilne. Często są to wiadomości mówiące o wystąpieniu jakiegoś zdarzenia lub informujące o zbliżającej się sytuacji, przykładem tutaj może być alert *RCB*, czyli system ostrzegania powiadamiający o zagrożeniach poprzez wiadomości SMS [16].
4. Powiadomienia e-mail - wiadomości wysyłane za pomocą poczty elektronicznej, często są to newslettery, gazetki sklepów, powiadomienia do firm, oferty czy wiadomości od znajomych.
5. Powiadomienia *in-app* (w aplikacji) - powiadomienia, które są wyświetlane tylko podczas korzystania z danej aplikacji. Są stosowane po to, aby zwiększyć zaangażowanie użytkownika podczas korzystania z aplikacji. Mogą przybierać różne formy, jak np. pojawienie się wiadomości na pasku powiadomień czy pokazanie na ekranie okna z grafiką. Są zazwyczaj trudne do usunięcia.
6. Powiadomienia *push* - są to również krótkie wiadomości tekstowe, które występują w przypadku urządzeń mobilnych, jak i w przypadku komputerów osobistych. Często są one przesyłane w czasie rzeczywistym, stąd też użytkownik może je dostać o każdej porze dnia i nocy. Są dostarczane na urządzenia niezależnie od tego czy aplikacja jest uruchomiona czy też nie. Mają wysoką widoczność. Na Rys. 3.1 przedstawiono rodzaje powiadomień typu *push*.

Wielu ekspertów od socjotechniki godzinami pracuje nad wymyśleniem coraz to nowszych rozwiązań, tak aby wysyłane powiadomienia przykuły uwagę użytkownika. Gdy chce się przekonać konkretną grupę osób do unikalnego produktu wiadomości są personalizowane,

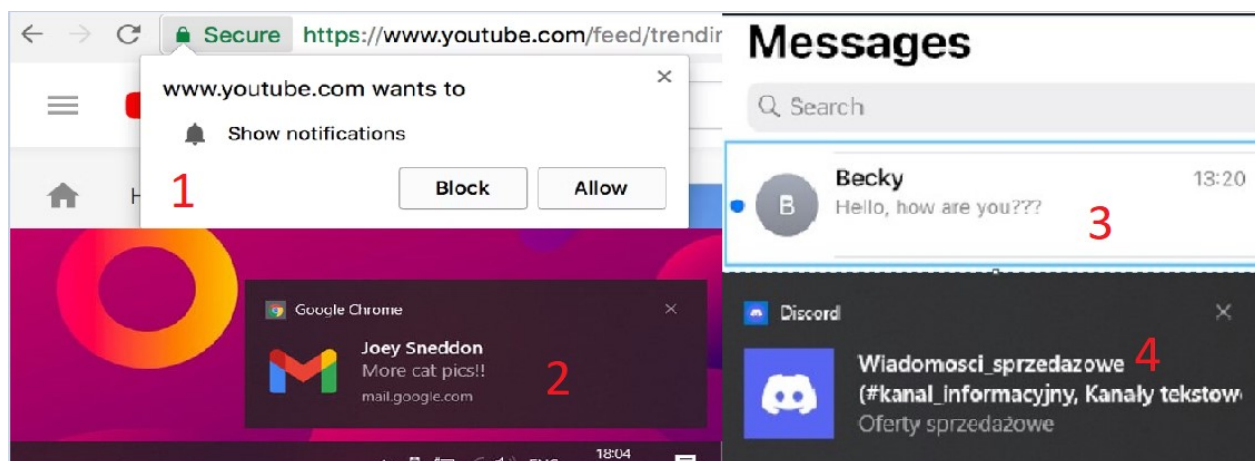
---

<sup>1</sup>Modal - prostokątny element interfejsu użytkownika pojawiający się w treści.

tak aby dostosować je do indywidualnych preferencji użytkownika. Są one również opisywane w sposób przejrzysty i zrozumiały, tak żeby skupić się na najważniejszych aspektach. Uwzględniając kontekst sytuacji można wywnioskować, że dany użytkownik w pewnych godzinach jest mniej skupiony i wtedy wysyłać powiadomienia. Jednak w pewnych przypadkach umożliwia się użytkownikowi kontrolę na swoimi powiadomieniami, tak aby mógł wybrać co jest ważne, a co odrzucić, w jakiej formie mają być otrzymywane powiadomienia i ustalić ich hierarchię. Rys. 3.2 przedstawia powiadomienia z różnych serwisów.



Rysunek 3.1: Powiadomienia *push* dzielą się na cztery rodzaje. licząc o lewej górnej strony są to powiadomienia mobilne, następnie powiadomienia na urządzenia przenośne, powiadomienia na komputery osobiste oraz powiadomienia webowe (zaczepnięto z [17]).



Rysunek 3.2: Powiadomienia z różnych serwisów: 1 - przeglądarka, 2 - powiadomienie e-mail, 3 - SMS, 4 - wiadomość z serwisu *Discord*.

### 3.3 Mechanizm tworzenia powiadomień

Podczas wdrażania aplikacji, która ma dostarczyć jakieś informacje, konieczne jest zaimplementowanie odpowiedniego mechanizmu. Efektywne zaprogramowanie takiego procesu może zwiększyć chęć klientów do korzystania z aplikacji. Najpierw trzeba ocenić jakie rodzaje powiadomień mają być dostarczane przez aplikację do klienta, czy mają to być powiadomienia SMS, e-mail. Kolejnym krokiem jest wybór platformy, która będzie dostarczała wiadomości. Może to być system *Android* czy *iOS*, aplikacja webowa lub inne platformy. Aplikacja wysyłająca powiadomienia często integruje się z usługami powiadomień, czy też korzysta z powiadomień przeglądarki internetowej. Ważne jest tutaj dostosowanie powiadomień, zgodnie z ich preferencjami. Optymalnym rozwiązaniem jest umożliwić wybór użytkownikom, tak aby sami mogli zdecydować o sposobie ich informowania. Dzięki dobremu interfejsowi powiadomienia mogą być łatwiej zarządzane. W wielu przypadkach użytkownicy muszą wyrazić zgodę na otrzymywanie powiadomień. Przed wysłaniem wiadomości trzeba również określić grupę odbiorców, do której będą one trafiać. Na samym końcu należy również przeprowadzić testy, tak aby sprawdzić wszystkie przypadki prawidłowego lub nieprawidłowego wysyłania wiadomości. Przetestowanie odpowiednich scenariuszy, również pod kątem wydajności i przesyłu wiadomości, powinno zapewnić wyższą jakość działania. Zaimplementowanie odpowiedniego mechanizmu powiadomień wymaga zaplanowania szczegółów technologicznych oraz uwzględnienia potrzeb użytkowników. Gdy platforma powiadomień będzie sprawnie funkcjonowała może zwiększyć to wpływ na sukces aplikacji oraz przyciągnie więcej klientów.

W tej pracy również wykorzystano mechanizmy i usługi wspomagające powiadomienia wysyłane do użytkowników. Niniejsza praca umożliwi użycie powiadomień e-mail, SMS, oraz wysyłanie wiadomości do serwisu *Discord*. Dla wysyłania wiadomości mail została użyta biblioteka *Nodemailer*, będąca modułem środowiska uruchomieniowego *Node.js*, w którym

została napisana aplikacja. Dla serwisu *Discord* została wykorzystana biblioteka *Discord.js*, odpowiadająca za wysłanie wiadomości i całą otoczkę z tym związaną. Na samym końcu dla wiadomości SMS wykorzystano serwis *Twilio* (również jako moduł *Node.js*), który jest narzędziem komunikacyjnym w wiadomościach SMS.

## Rozdział 4

# Architektura mikroservisowa

Współczesne systemy komputerowe są rozwinięte pod względem technologicznym jak również mają rozbudowaną architekturę oprogramowania. Gdy podejmuje się wybór architektury, z której chce się skorzystać warto rozważyć popularne technologie w dziedzinie tworzenia aplikacji, gdzie mikroservisy mogą być jedną z takich technologii.

### 4.1 Opis architektury mikroservisowej

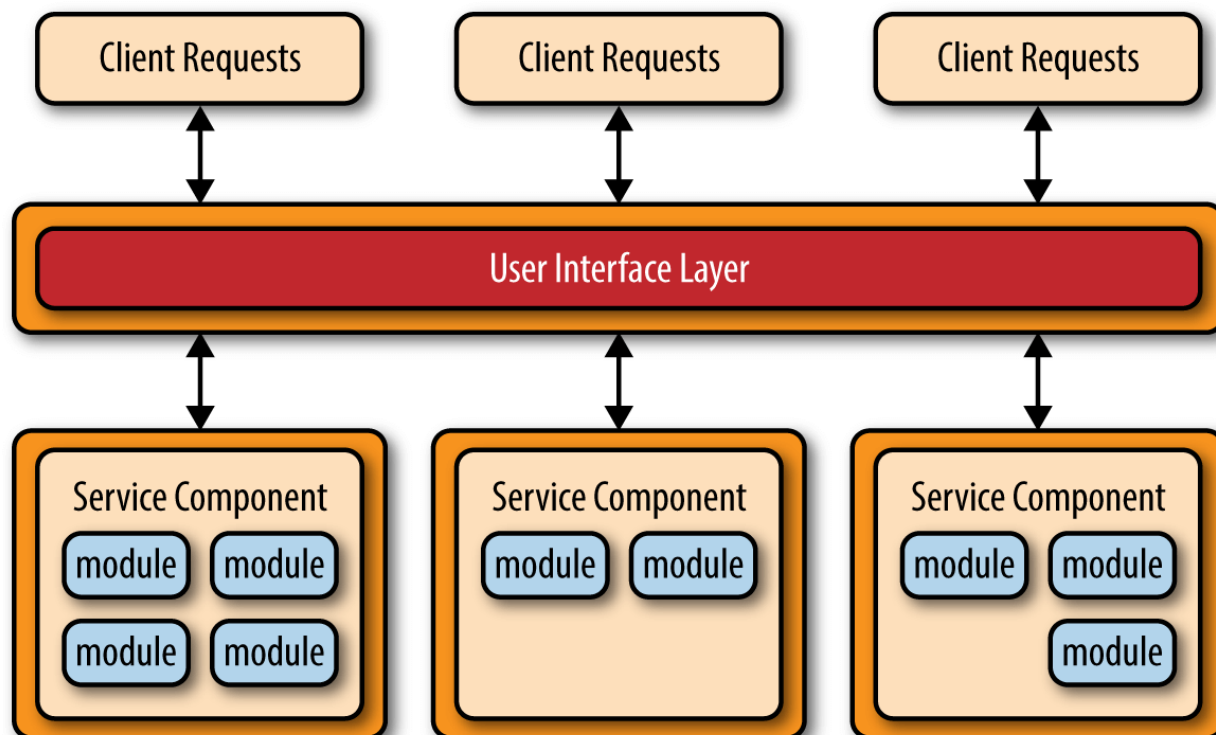
Architektura mikroservisowa to styl tworzenia oprogramowania dla aplikacji internetowych, która implementuje wzorzec architektury zorientowanej na usługi [18]. Styl ten traktuje aplikację jako zbiór małych serwisów, które są ze sobą luźno połączone lub komunikują się z sobą przez określone protokoły. W takiej architekturze projekt podzielony jest na mniejsze usługi, które nazwane są mikroservisami [20].

W przeciwieństwie do systemów monolitycznych, w których cała funkcjonalność jest skupiona w jednym bloku, mikroservisy dzielą się na mniejsze autonomiczne jednostki. Każdy z małych serwisów wykonuje konkretne zadanie lub konkretną funkcję biznesową np. wysłanie maila, zarządzanie użytkownikami czy wykonywanie określonej czynności. Mikroservisy komunikują się między sobą za pomocą protokołów takich jak: *HTTP*, *REST*<sup>1</sup> czy *SOAP*<sup>2</sup>. Mogą być one pisane w różnych językach programowania, jak i korzystać z różnych bibliotek i technologii, dzięki czemu mogą być wdrażane i testowane osobno. Warto podkreślić, że mikroservisy to nie zawsze jedna, pojedyncza usługa. Na jeden mikroservis może składać się kilka mniejszych modułów. Dzięki rozdzieleniu aplikacji na mikroservisy każdy z nich może być wdrażany, testowany i rozwijany niezależnie od pozostałych. Na Rys. 4.1 pokazano zasadę działania architektury mikroservisowej.

---

<sup>1</sup>*REST* - architektura komunikacyjna od wymiany informacji wykorzystująca pliki *JSON*.

<sup>2</sup>*SOAP* - protokół komunikacyjny do wymiany informacji, wykorzystujący pliki *XML*.



Rysunek 4.1: Zasada działania architektury mikroserwisowej. Połączenie poprzez warstwę interfejsu użytkownika między zapytaniami użytkownika, a mikroserwisami, które składają się z kilku modułów. Każdy serwis jest wdrażany jako oddzielna jednostka (zaczepnięto z [19]).

Mikroserwisy to koncepcja składająca się z dwóch rozwiązań, czyli wzorca zorientowanego na usługi (*SOA*<sup>3</sup>) oraz koncepcji tworzenia oprogramowania, gdzie potrzeby finansowe odzworowuje się przez działający mikroserwis (*DDD*<sup>4</sup>) [21]. W podejściu *DDD* nacisk kładzie się przede wszystkim na komponenty systemu, tak aby odzwierciedlały rzeczywistość. Stawia się tutaj na komunikację zespołów: biznesowego i tworzącego oprogramowanie, podział projektu na zespoły i tworzenie oprogramowania w kolejnych etapach, tzw. iteracjach. Wzorzec *SOA*, czyli architektura zorientowana na usługi definiuje usługi, które spełniają wymagania użytkownika. Pojęcie to obejmuje również zestaw metod organizacyjnych, jak i technicznych, które mają połączyć aspekt biznesowy oraz informatyczny [22]. Standard tej koncepcji nie jest zależny od technologii przez co podany wzorzec jest elastyczny. Technologia wprowadza kilka założeń takich jak: zbudowanie aplikacji z mniejszych części odizolowanych od siebie, logikę odizolowaną od interfejsu, ponowne wykorzystanie tej samej usługi, niewymuszanie integracji z innymi komponentami czy systemami. Dodatkowo każda z tych usług powinna być niezależna technologicznie.

<sup>3</sup>*SOA* - Service-oriented architecture.

<sup>4</sup>*DDD* - Domain Driven Design, sposób tworzenia aplikacji, odzwierciedlający założenia biznesowe.

Architektura mikroserwisów jest utożsamiana z terminem dotyczącym *Unixa*: “rób jedną rzecz i rób ją dobrze” [18]. Posiada ona zarówno wiele zalet jak i też wad. Oto niektóre korzyści płynące ze stosowania mikroserwisów [23]:

- mikroserwisy są mniejsze i niezależne przez co mogą być łatwiejsze w późniejszym wdrażaniu oraz rozwoju, elementy są poszczególnymi modułami dzięki czemu można je rozbudowywać, łatwiej je również testować oraz ponownie ich używać,
- poszczególne serwisy mogą być skalowane bez względu na pozostałe,
- każdy komponent może być pisany w dowolnej technologii przez co łatwiej jest analizować błędy w kodzie, choć często może to być też wada tego rozwiązania,
- system staje się bardziej stabilny, ponieważ błąd w jednym module nie powoduje zatrzymania całego systemu,
- łatwo można kopiować (duplikować poszczególne elementy systemu), tak aby zapewnić redundancję<sup>5</sup> w razie awarii systemu,
- gdy jeden z zespołów tworzących oprogramowanie zawiedzie i część kodu pisana przez ten zespół nie nadaje się do użytku, to wtedy nie jest tak kosztowne wyrzucenie tej części projektu “do kosza”, jak w przypadku innych architektur, dzięki temu usprawnia się koordynacja w zarządzaniu zespołem.

Nie zawsze jednak architektura mikroserwisowa jest odpowiednią technologią przy wyborze struktury projektu. Wady płynące ze stosowania architektury mikroserwisowej to m.in.:

- wraz z rozwojem aplikacji rosną poszczególne moduły, przez co wymaga się dobrego zaplanowania infrastruktury, zarówno biznesowej jak i technologicznej,
- nie stosuje się jej przy mniejszych projektach,
- zdarza się, że gdy projekt jest napisany w zbyt wielu technologiach, to następuje wtedy duże zamieszanie i łatwo zgubić się w takim projekcie,
- utrudnione zostaje testowanie interakcji między serwisami oraz analiza działania komunikacji, mikroserwisy wymagają poświęcenia czasu na testowanie.

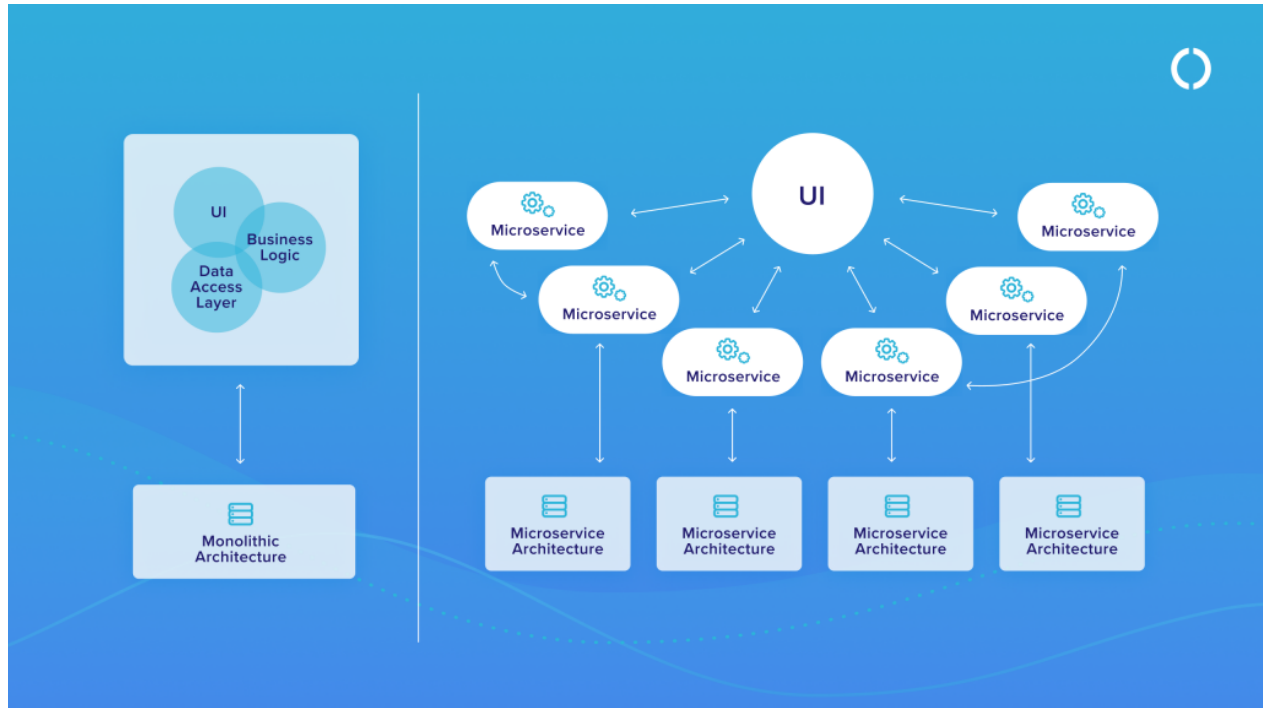
Dla porównania w architekturze monolitycznej pojedyncza jednostka jest bazą dla komponentów funkcjonalnych, takich jak: operacje na bazie danych czy działanie aplikacji w tle. W architekturze monolitycznej aplikacja nie jest rozdzielona na kilka osobnych mikroserwisów, lecz jest ona oparta na jednej bazie kodu. Chcąc wprowadzić zmianę w konkretnym fragmencie kodu aplikacji, która jest rozbudowana, niekiedy ciężko jest zmienić jedną część, bez przebudowy innych fragmentów oprogramowania. Prowadzi to zazwyczaj do tego, że aplikację ciężiej rozbudować i staje się ona mniej elastyczna. Gdy jeden z komponentów czy

---

<sup>5</sup>redundancja - nadmiarowość danego elementu.



też procesów się uszkodzi to niestety cała aplikacja również przestaje działać, może to być wyrządzone przez bardzo małe, niezauważalne błędy. Architektury monolitycznej używa się do realizacji małych aplikacji. Potrzeba przebudowy aplikacji z serwisu monolitycznego na mikroserwisowy spowodowała rozrost wielu nowych programów czy usług, np. AWS (*Amazon Web Services*). Na Rys. 4.2 przedstawiono porównanie architektury monolitycznej oraz architektury mikroserwisowej.



Rysunek 4.2: Porównanie architektury monolitycznej oraz mikroserwisowej. Po lewej stronie przedstawiona jest architektura monolityczna, w której logika, interfejs użytkownika oraz warstwa dostępu do danych znajdują się w jednej bazie kodu. Po prawej pokazana jest architektura mikroserwisowa, w której wcześniej wymienione komponenty znajdują się w niezależnych mikroserwisach (zaczepnięto z [24]).

## 4.2 Architektura mikroserwisowa w kontenerach

Wdrażanie mikroserwisów wiąże się z wyborem środowiska, w którym będzie funkcjonował napisany program. Istnieje kilka sposobów wdrażania mikroserwisów m.in.:

- wdrożenia w chmurze, tak by łatwo obsługiwać użytkowników z różnych aplikacji,
- przetwarzanie bezserwerowe (serverless), czyli model usług chmurowych, w których programista zajmuje się wyłącznie tworzeniem logiki aplikacji, a infrastrukturę dla aplikacji dostarcza usługodawca. Pozwala ono zaoszczędzić na kosztach infrastruktury, stosowane

gdy spodziewany jest duży ruch, a programista piszący aplikację nie jest zmuszony do martwienia się zbudowaniem konkretnego serwera (sprawą tą zajmuje się dostawca usług, np. *Amazon*),

- usługa *Paas* (Platform-as-a-Service) <sup>6</sup>, tutaj również wynajmuje się serwer dostawcy, a on dostarcza potrzebne narzędzia,
- stworzenie własnej infrastruktury, co w niektórych przypadkach może być wskazane, gdy projekt jest złożony, a niektóre serwisy nie zapewniają wszystkich funkcjonalności,
- kontenery, które są łatwo skalowalne oraz można dzięki nim szybko rozwiązywać problemy.

Jednym z najpopularniejszych sposobów organizacji aplikacji mikroserwisowych są kontenery. Jest to struktura danych, której zadaniem jest przechowywanie oraz organizacja jakiegoś zbioru danych. Jest pewnego rodzaju wirtualnym buforem (maszyną wirtualną), gdzie mieści się aplikacja wraz ze środowiskiem oraz zestawem narzędzi potrzebnym do jej uruchomienia. Zazwyczaj są to wirtualne systemy operacyjne, w których wnętrzu można łatwo uruchomić napisaną aplikację. Niektórymi z takich systemów są *Kubernetes* czy *Docker*. W kolejnym rozdziale omówione zostanie środowisko *Docker*, ponieważ takie będzie użyte w niniejszej pracy.

---

<sup>6</sup>*Paas* - Platforma jako usługa.

## Rozdział 5

# Implementacja mikroservisowej platformy powiadomień

Pod względem identyfikacji zagadnienia biznesowego mikroservisowa platforma powiadomień o ofertach sprzedażowych jest aplikacją, która powinna być odpowiedzią na potrzeby użytkowników dotyczące prostego informowania ich o ofertach sprzedażowych poprzez wysyłanie powiadomień w określonym czasie na temat danego artykułu. Na rynku istnieje kilka takich serwisów lub rozwiązań podobnych do przedstawionego, np. *Ceneo*. Jednak serwisy te są dosyć złożone, a użytkownik często może mieć problemy, by uzyskać oczekiwany rezultat. Ponadto w większości przypadków rozwiązania te są płatne. Zaprojektowana w ramach pracy aplikacja ma na celu umożliwić użytkownikowi przejrzystą konfigurację oraz dostosowanie powiadomień do platformy, do której chce przesłać dane. Aplikacja posiada następujące założenia funkcjonalne:

1. Możliwość rejestracji i zalogowania się do serwisu - gdy użytkownik poda swoje dane to będą one zapisywane w bazie *MySQL*.
2. Możliwość konfiguracji poszczególnych opcji, czyli wybrania polityk “szukania na bieżąco” aktualnych ofert sprzedażowych lub konfiguracji powiadomień i wysyłania ich o określonym czasie.
3. Możliwość wpisania konkretnej frazy oraz ilości żądanych zapytań w celu uzyskania ofert dotyczących tejże frazy.
4. Wyświetlanie ofert dotyczących wybranych serwisów oraz wysłanie powiadomień w określonej godzinie.

### 5.1 Wykorzystane technologie

W tym rozdziale omówiona zostanie aplikacja i sposób jej implementacji. Przedstawiono przegląd wykorzystanych technologii użytych podczas pisania programu. Ponadto zostanie

zaprezentowany projekt, implementacja platformy, model danych oraz elementy struktury.

Do implementacji omawianej aplikacji użyto następujących technologii serwerowych (backend): *Node.js*, *Python* oraz narzędzia do tworzenia widoków użytkownika *React*. Ponadto użyto silnika bazy danych *MySQL* oraz systemu konteneryzacji *Docker*.

**Node.js** to środowisko uruchomieniowe dla języka *Javascript*, które działa jak serwer. Jest ono bezpłatne oraz jest to środowisko o otwartym kodzie źródłowym. *Node.js* pozwala na wykonywanie kodu *Javascript* poza przeglądarką. Jest to środowisko, które pozwala na równoległe wykonywanie kilku zadań, czyli działa w sposób asynchroniczny. Posiada dużą bibliotekę modułów (w tym framework *Express* wykorzystany w pracy), dzięki czemu można instalować i importować wiele bibliotek do pisanej aplikacji. *Node.js* działa jako serwer przetwarzający wszystkie zapytania, zwracający odpowiedzi oraz włączający skrypty napisane w języku *Python*.

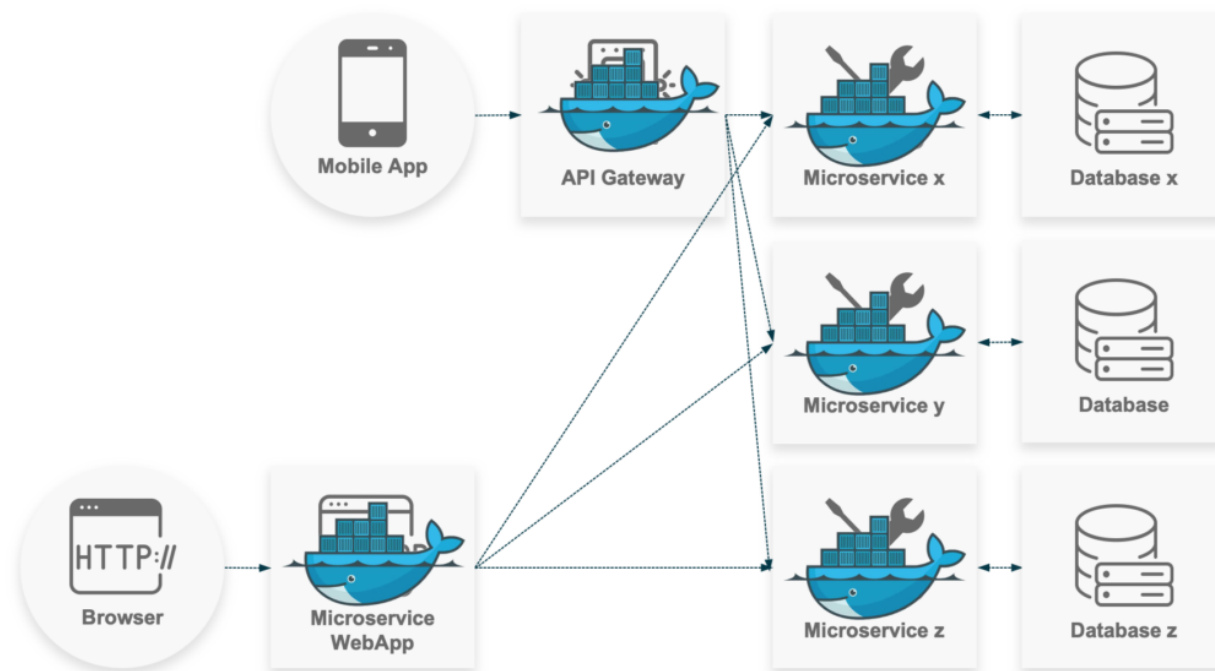
**Python** jest to język programowania wysokiego poziomu, który posiada wiele bibliotek ogólnego przeznaczenia. Wyróżnia się tym, iż jest przejrzysty, a programy w nim pisane zawierają mniej linii kodu niż w innych językach. W *Pythonie* napisano skrypty służące do scrapowania danych z sieci. Wykorzystano też biblioteki języka *Python*: *requests* i *beautifulsoup* - służące do webscrapingu oraz *mysql-connector-python* do połączenia z bazą *MySQL*.

**MySQL** - system zarządzania relacyjnymi bazami danych. Umożliwia tworzenie związków między danymi. Bazy danych są ważnym elementem przy budowie aplikacji internetowych, ponieważ przechowują dane użytkowników, takie jak dane logowania lub dane z zewnętrznych źródeł, np. informacje o ofertach sprzedażowych. Za komunikację między backendem, a bazą danych odpowiada *Sequelize*, czyli narzędzie do mapowania obiektowo-relacyjnego (*ORM*) *Node.js* dla *MySQL*. W tej pracy *MySQL* służy jako główna baza platformy, gdzie zapisywane są dane aplikacji.

**React** jest to biblioteka języka *Javascript*, którą zaprojektowano z myślą o tworzeniu interfejsów graficznych. Wykorzystywana jest przy budowaniu dynamicznych aplikacji oraz aplikacji wymagających częstego odświeżania i aktualizacji. Stosuje koncepcję komponentów, dzięki którym można łączyć szkielet aplikacji w bardziej skompensowaną strukturę. Pozwala na ponowne używanie komponentów w nim napisanych. Wprowadza wirtualne drzewo *DOM* oraz język *JSX*, który jest rozszerzeniem dla składni *Javascript*. W aplikacji odpowiada za graficzny interfejs, w którym porusza się użytkownik.

**Docker** to oprogramowanie, które ułatwia tworzenie aplikacji w t.zw. kontenerach. Umożliwia wirtualizację, wdrażanie i uruchamianie aplikacji w izolowanych środowiskach, które są od siebie niezależne, aczkolwiek mogą się one między sobą komunikować. Kontenery *Docker* można uruchamiać na wielu platformach, przez co są one łatwo przenośne. Umożliwia zarządzanie zasobami, dzięki odpowiedniemu sterowaniu kontenerami. Program dzięki plikowi *Dockerfile* umożliwia szybkie tworzenie systemów w kontenerach. *Docker* - to popularny system maszyn wirtualnych, dzięki którym można wdrażać oraz zarządzać mikroserwisami. Aplikacja jest spakowana w kontenery, przez co łatwiej sterować jej przebiegiem. *Docker* tworzy obrazy, w których zawierają się mikroserwisy, zaś obrazy udostępniane są w kontenerach. *Docker* umożliwia tworzenie obrazów systemów operacyjnych wraz z aplikacją tam działającą na podstawie plików pod nazwą *Dockerfile*. Dzięki temu w pliku można zarządzać

pakietami jakie mają być zainstalowane oraz mikrousługami, które mają być wdrożone. Jeżeli jest mowa o platformie mikroserwisowej, serwisy te muszą się ze sobą komunikować będąc w kontenerach. Odpowiada za to narzędzie *Docker Compose*, które umożliwia komunikację między takimi serwisami. Jest to plik konfiguracyjny, w którym definiuje się komunikację między serwisami. Na Rys. 5.1 przedstawiono architekturę mikroserwisową z użyciem kontenerów *Dockera*.



Rysunek 5.1: Przykład skonteneryzowanych mikroserwisów i połączeń między nimi (zaczepnięto z [25]).

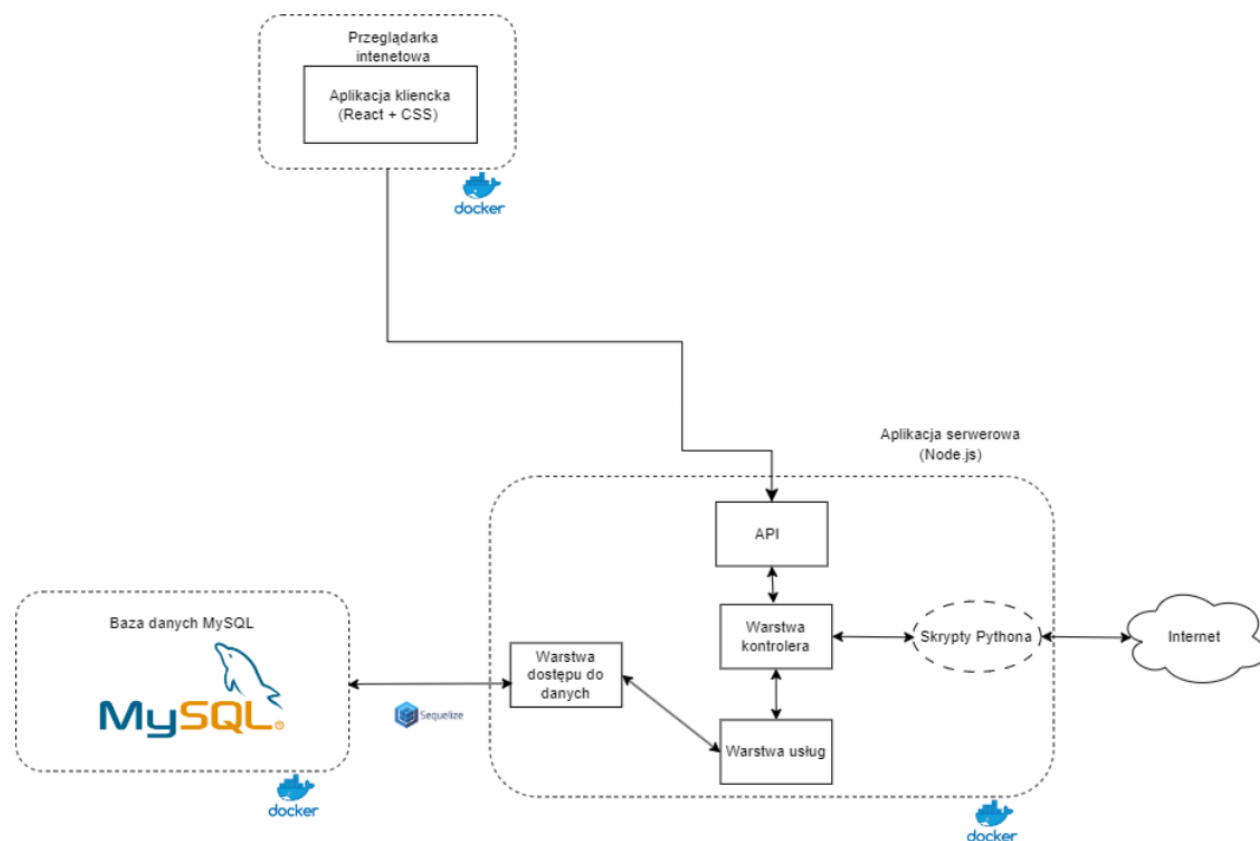
## 5.2 Architektura platformy

Przedstawiona platforma składa się z części klienckiej oraz części serwerowej. Fragment odpowiadający za aplikację kliencką stanowi kod napisany w języku *Javascript*, z wykorzystaniem biblioteki *React*. Interfejs graficzny aplikacji powstał z wykorzystaniem biblioteki *CSS: Bootstrap*.

Część serwerowa to program napisany w języku *Javascript*, którego podstawę stanowi środowisko uruchomieniowe *Node.js* wraz ze środowiskiem *Express.js*. Wykorzystano również bibliotekę *handlebars.js*, odpowiadającą za wysyłanie wiadomości, które są szablonami tworzonymi w aplikacji. Pozostałą część serwerową stanowią skrypty napisane w języku *Python*, które pobierają i zwracają informacje ze stron internetowych. Jako środowisko magazynujące dane wykorzystano bazę danych *MySQL*. Środowisko, zarówno frontendowe jak i backendowe,

a także baza danych są przechowywane w kontenerach *Dockera*, dzięki czemu cała aplikacja jest bardziej zorganizowana.

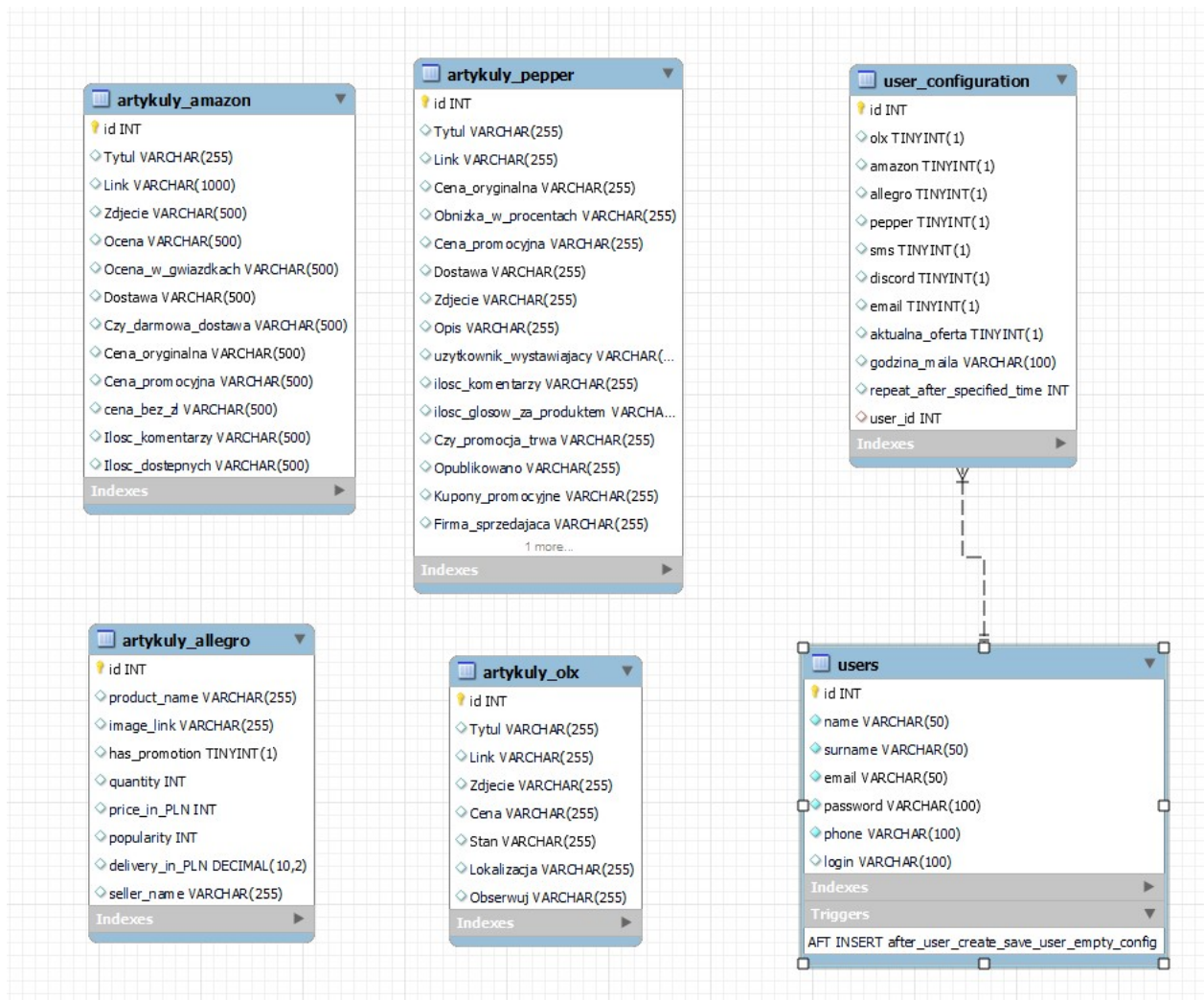
Schemat, przedstawiający połączenie części serwerowej oraz klienckiej znajduje się na Rys. 5.2.



Rysunek 5.2: Schemat prezentujący połączenie platformy powiadomień.

Strukturę bazy danych podzielono na sześć tabel. Cztery z nich odpowiada za przechowywanie informacji związanych z każdym serwisem sprzedażowym, czyli są to tabele: *artykuly\_amazon*, *artykuly\_pepper*, *artykuly\_olx*, oraz *artykuly\_allegro*. Za informacje związane z użytkownikami logującymi się do platformy odpowiadają tabele: *users* (wraz z triggerem wywołującym się po stworzeniu użytkownika) oraz tabela *user\_configuration* odpowiadająca za konfigurację użytkownika.

Schemat bazy danych przedstawiono na Rys. 5.3. Prezentuje on zależności oraz powiązania platformy.



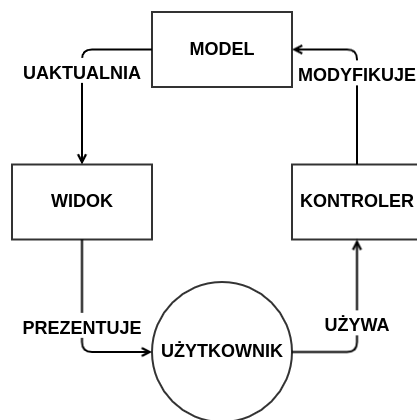
Rysunek 5.3: Schemat prezentujący strukturę bazy danych *MySQL*.

### 5.2.1 Wprowadzenie wzorca MVC do architektury oprogramowania

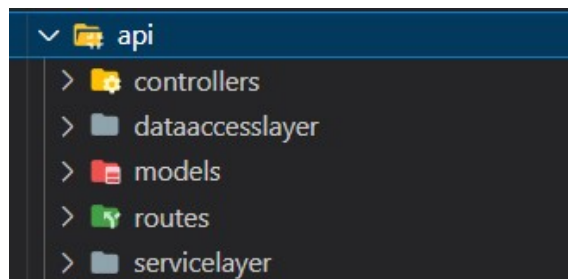
Komunikacja w mikroservisach często odbywa się przez protokół *HTTP*. Mikroservisy mogą udostępniać *API(REST)*<sup>1</sup>, czyli interfejs programistyczny typu klient-serwer służący do przesyłania danych, m.in. dla mikrosług. W projektowanej aplikacji zastosowano podejście zapytań i odpowiedzi, które są zwracane użytkownikowi. Wysyłanie oraz odbieranie danych odbywa się za pomocą formatu *JSON* lub *XML* [26]. Zdarza się też, że mikroservisy wywołują *API* innych mikroservisów, tak aby komunikować się bezpośrednio między sobą. Dzięki wywołaniom metod, mikroservis przesyła dane lub odpytuje inny mikroservis o rzeczy, które chce uzyskać. Do organizacji struktury kodu serwisu *API* można wykorzystać wzorzec projektowy *MVC* (Model - Widok - Kontroler). Wzorzec *MVC*, który służy organizacji całej struktury aplikacji, posiadającej graficzny interfejs [27]. Dzieli on aplikację na trzy

<sup>1</sup>*REST (RESTful)* - Representational State Transfer.

osobne warstwy. Warstwa prezentacji jest oddzielona od logiki biznesowej. Natomiast model odpowiada za reprezentację danych i powiązania między nimi. Widok służy do organizacji interfejsu użytkownika. Kontroler odpowiada za interakcję z użytkownikiem, przyjmując dane wejściowe i na tej podstawie modyfikuje model. Przykładowa struktura *API* wraz ze ścieżkami pokazana jest na Rys. 5.4, zaś struktura *API* aplikacji platformy powiadomień znajduje się na Rys. 5.5.



Rysunek 5.4: Przykład działania frameworka *MVC* (zaczepnięto z [27]).



Rysunek 5.5: Struktura *API* programu serwerowego.



## 5.3 Elementy implementacji

### 5.3.1 Aplikacja kliencka

Widoki aplikacji klienckiej zostały zorganizowane w czterech niezależnych stronach: główna, konfiguracja, panel użytkownika, rejestracja. Organizację wymienionych stron pokazano na wydruku kodu 5.1.

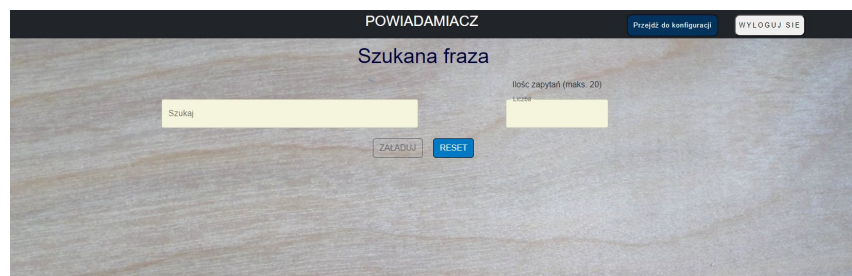
```
1 <BrowserRouter>
2   <Routes>
3     <Route path="/" element={<LoginForm/>} />
4     <Route path="/register" element={<RegistrationForm/>} />
5     <Route path="/dashboard" element={<Dashboard />} />
6     <Route path="/configuration" element={<ConfigurationPage/>} />
7   </Routes>
8 </BrowserRouter>
```

Listing 5.1: Ścieżki w programie, w które może wchodzić użytkownik.

Aplikacja zawiera cztery ścieżki odpowiadające wymienionym stronom funkcjonalnym:

- /configuration - strona z konfiguracją,
- /dashboard - strona z panelem, gdzie są wyświetlane informacje,
- /register - strona z rejestracją użytkownika,
- / - strona z logowaniem jako domyślna strona systemu.

W procesie rejestracji użytkownik musi podać wymagane dane, t.j. imię, nazwisko, e-mail, login, hasło oraz numer telefonu. Po poprawnej rejestracji użytkownik przechodzi do okna logowania i loguje się poprzez podanie loginu i hasła. Dane są zapisywane w *localStorage*.<sup>2</sup> Kolejno wysyłane są zapytania na odpowiedni endpoint, po czym serwer zwraca dane i użytkownik jest zalogowany. Gdy użytkownik zaloguje się poprawnie to wtedy następuje przekierowanie do panelu użytkownika (Rys. 5.6). Następnie użytkownik musi przejść do strony z konfiguracją poprzez naciśnięcie przycisku “Przejdź do konfiguracji”.



Rysunek 5.6: Panel główny aplikacji frontendowej.

<sup>2</sup>*localStorage* - właściwość przeglądarki pozwalająca przechowywać dane, podobna do ciasteczek.

Na stronie konfiguracji (Rys. 5.7) pokazane są dane, a na czerwono podświetlone są wybrane ustawienia. Są tutaj do wyboru platformy sprzedażowe, z których użytkownik będzie mógł pobrać informacje o produktach, czyli serwisy: *OLX*, *Amazon*, *Pepper* oraz *Allegro* (dla *Allegro* praca posiada dostęp do środowiska testowego, opisano to w dalszej części pracy). Następnie pokazane jest z okno z polami wyboru: opcja aktualnych ofert, czyli w czasie rzeczywistym pokazanie na ekranie danych ofert, ustawienie konkretnej godziny wysyłania powiadomień oraz opcja z czasem rutynowego wysyłania powiadomień (co określony czas). Na samym dole znajdują się systemy, do których będą wysłane powiadomienia, czyli SMS, email, *Discord* (można zaznaczyć wszystkie opcje).

**Twoje dane**

Login: **test**

Adres e-mail: **wojtektokoxik@gmail.com**

Telefon: **+48123456783**

Ustawienia: **olx, aktualna\_oferta**

**Wybierz witryny z których chcesz pobrać oferty:**

- ☐ OLX
- ☐ Amazon
- ☐ Pepper
- ☐ Allegro

☐ Aktualne oferty

☐ Ustaw godzinę wysłania

Godzina wysłania: --:--

☐ Co ile minut chcesz wysłać wiadomości

Co ile minut:

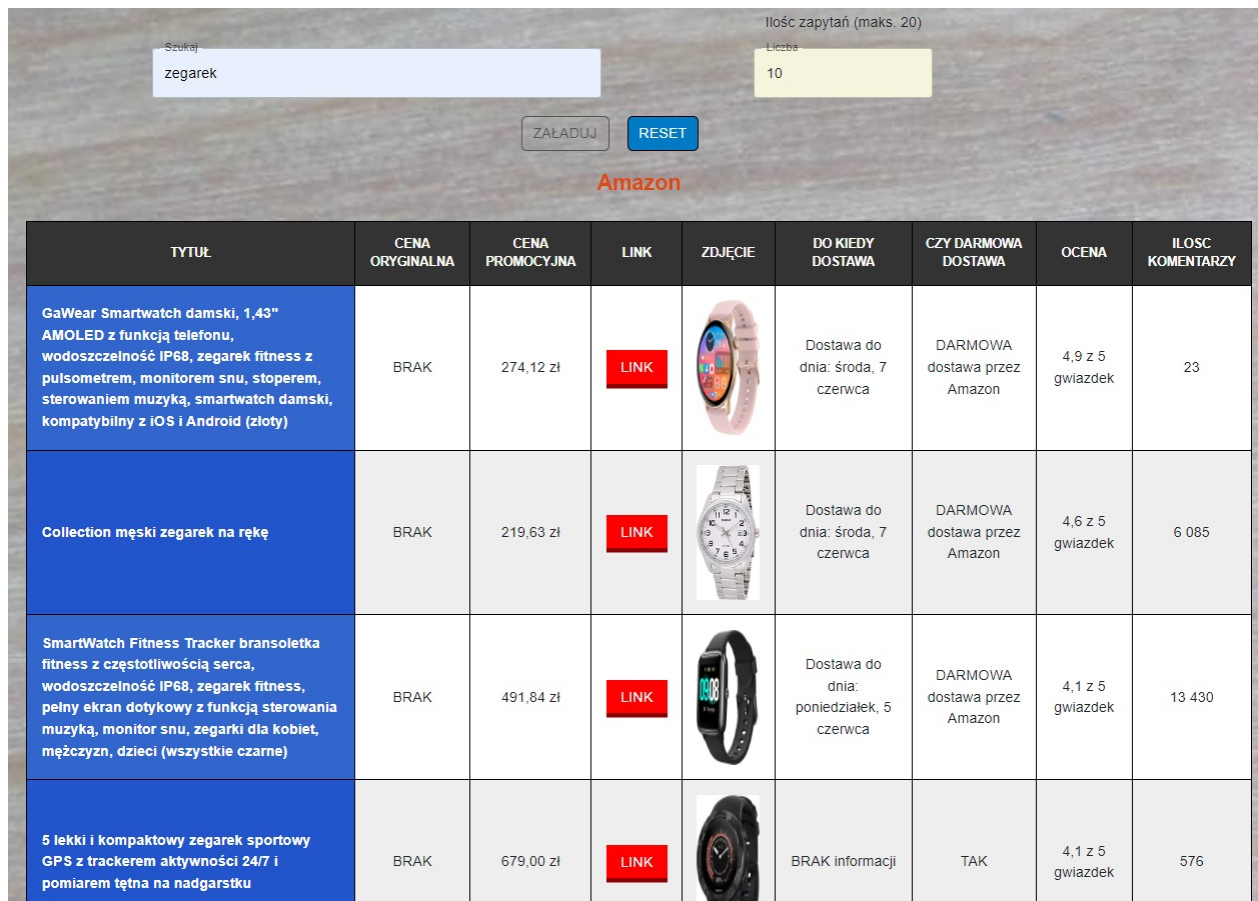
**Wybierz systemy powiadomień, do których chcesz wysłać wiadomość:**





- ☐ SMS
- ☐ Email
- ☐ Discord

**Zapisz konfigurację** **Powrót do panelu**

Rysunek 5.7: Strona z konfiguracją aplikacji dla użytkownika.

Gdy została wybrana konkretna konfiguracja użytkownik przechodzi do panelu z powiadomieniem i wpisuje konkretną frazę określającą produkt oraz liczbę zapytań, które chce uzyskać. Dostępne są dwa przyciski: “Załaduj” - do zatwierdzenia procesu oraz “Reset” do zresetowania widoku strony. Po załadowaniu się strony rezultat wyników widać na ekranie (pokazane na Rys. 5.8) Jeżeli zostaną wybrane inne opcje niż aktualna oferta to na ekranie pojawi się inny komunikat (Rys. 5.9). Przykładowe powiadomienia zostały pokazane na Rys. 5.10.



TYTUŁ	CENA ORYGINALNA	CENA PROMOCYJNA	LINK	ZDJĘCIE	DO KIEDY DOSTAWA	CZY DARMOWA DOSTAWA	OCENA	ILOSC KOMENTARZY
GaWear Smartwatch damski, 1,43" AMOLED z funkcją telefonu, wodoszczelność IP68, zegarek fitness z pulsometrem, monitorem snu, stoperem, sterowaniem muzyką, smartwatch damski, kompatybilny z iOS i Android (złoty)	BRAK	274,12 zł	<a href="#">LINK</a>		Dostawa do dnia: środa, 7 czerwca	DARMOWA dostawa przez Amazon	4,9 z 5 gwiazdek	23
Collection męski zegarek na rękę	BRAK	219,63 zł	<a href="#">LINK</a>		Dostawa do dnia: środa, 7 czerwca	DARMOWA dostawa przez Amazon	4,6 z 5 gwiazdek	6 085
SmartWatch Fitness Tracker bransoletka fitness z częstotliwością serca, wodoszczelność IP68, zegarek fitness, pełny ekran dotykowy z funkcją sterowania muzyką, monitor snu, zegarki dla kobiet, mężczyzn, dzieci (wszystkie czarne)	BRAK	491,84 zł	<a href="#">LINK</a>		Dostawa do dnia: poniedziałek, 5 czerwca	DARMOWA dostawa przez Amazon	4,1 z 5 gwiazdek	13 430
5 lekki i kompaktowy zegarek sportowy GPS z trackerem aktywności 24/7 i pomiarem tętna na nadgarstku	BRAK	679,00 zł	<a href="#">LINK</a>		BRAK informacji	TAK	4,1 z 5 gwiazdek	576

Rysunek 5.8: Wynik wyszukiwań dla konfiguracji: Amazon oraz aktualne oferty.

# Szukana fraza

Szukaj

Ilość zapytań (maks. 20)

Liczba

7

Dane będą przesłane przez serwer do serwisów: **email**

Dane będą przesyłane **co 3 minut(y)**

**Milego dnia!**

Rysunek 5.9: Wynik dla opcji: Amazon, Pepper, E-mail, oraz wysyłanie co 3 minuty.

**1**

**OLX**

Oferty sprzedaży

Tytuł

Renault Megane Renault Megane 1.5 DCI Business

Link

<https://www.olx.pl/haslo/offer/ta/renault-megane-renault-megane-1-5-dci-business-856-687-1000>

Cena

48 500 zł (negocj.)

Lokalizacja

Jaworzno - 22 maja 2023

**2**

**OFERTY SPRZEDAŻOWE**

**OLX**

TYTUŁ	CENA	LOKALIZACJA	LINK	ZDJĘCIA	STAN
Zegarek damski Michael Kors	185 zł	Łódź, Włocławek - 02 czerwca 2023	<a href="#">LINK</a>	BRUK	Używane
Zegarek męski Festina Chrono Elite	350 zł	Warszawa, Targówek - 02 czerwca 2023	<a href="#">LINK</a>	BRUK	Używane
Damski zegarek	50 zł	Warszawa, Śródmieście - 02 czerwca 2023	<a href="#">LINK</a>	BRUK	Używane
Męski zegarek Pulsar	25 zł	Jastrzębie-Zdrój - 02 czerwca 2023	<a href="#">LINK</a>	BRUK	Używane
Zegarek Fossil 17 Buzak, 1 lat 50	149 zł	Łódź - Dziękuję 11.23	<a href="#">LINK</a>	BRUK	Używane
Sz. Corolla 1.8 i Automatic. Small Second zegarek automatyczny	Zamierz	Warszawa, Mokotów - Dziękuję 10.59	<a href="#">LINK</a>	BRUK	Używane
Casio G-shock zegarek męski	190 zł	Olsztyn - Dziękuję 13.26	<a href="#">LINK</a>	BRUK	Używane

**Pepper**

TYTUŁ	CENA ORYGINALNA	CENA PROMOCYJNA	CZY PROMOCJA TRWA	DOŚWIAD	OPIS	LINK	OCENA W %	OPUBLIKOWANO	ZDJĘCIA	ILUŚĆ KOMENTARIÓW	UŻYTKOWNIK WYSTAWIAJĄCY
Smartwatch z GPS, Aktywny GPS 300 Hz, Czerwony	599.99 zł	549.99 zł	TAK	Darmowa dostawa	Multiportowy zegarek z GPS, spracowany przez naszych projektantów zgodnie z Canos. Mięci bagażnik, prywatne, jacy na rowca i twardy. Bateria, włączona, tenora, idy jedne C to.	<a href="#">LINK</a>	-6%	maj 26.		8	razu

**3**

Sent from your Twilio trial account - Penzoo Pure Instinct Rear 3 - Stalowy bagażnik rowerowy do montażu na kłapie. Mieści trzy rowery. <https://www.penzoo.pl/promocje/penzoo-pure-instinct-rear-3-stalowy-bagaznik-rowerowy-do-montazu-na-kłapie-miesci-trzy-rowery-426392>

VHEUHZ Zawias drzwiowy stoper do drzwi przedni lewy/ prawy kompatybilny z Megane 3 Fluence L3 2010-2022 824310007R 80431007R [https://www.amazon.pl/VHEUHZ-kompatybilny-2010-2022-824310007R-80431007R/4p/80B88VYFG/ref=sp\\_1\\_48?keywords=renault+megane+3+kombi&qid=1686666666](https://www.amazon.pl/VHEUHZ-kompatybilny-2010-2022-824310007R-80431007R/4p/80B88VYFG/ref=sp_1_48?keywords=renault+megane+3+kombi&qid=1686666666)

WHEUHZ Zawias drzwiowy stoper do drzwi przedni lewy/ prawy kompatybilny z Megane 3 Fluence L3 2010-2022 824310007R 80431007R [https://www.amazon.pl/VHEUHZ-kompatybilny-2010-2022-824310007R-80431007R/4p/80B88VYFG/ref=sp\\_1\\_48?keywords=renault+megane+3+kombi&qid=1686666666](https://www.amazon.pl/VHEUHZ-kompatybilny-2010-2022-824310007R-80431007R/4p/80B88VYFG/ref=sp_1_48?keywords=renault+megane+3+kombi&qid=1686666666)

WHEUHZ Zawias drzwiowy stoper do drzwi przedni lewy/ prawy kompatybilny z Megane 3 Fluence L3 2010-2022 824310007R 80431007R [https://www.amazon.pl/VHEUHZ-kompatybilny-2010-2022-824310007R-80431007R/4p/80B88VYFG/ref=sp\\_1\\_48?keywords=renault+megane+3+kombi&qid=1686666666](https://www.amazon.pl/VHEUHZ-kompatybilny-2010-2022-824310007R-80431007R/4p/80B88VYFG/ref=sp_1_48?keywords=renault+megane+3+kombi&qid=1686666666)

WHEUHZ Zawias drzwiowy stoper do drzwi przedni lewy/ prawy kompatybilny z Megane 3 Fluence L3 2010-2022 824310007R 80431007R [https://www.amazon.pl/VHEUHZ-kompatybilny-2010-2022-824310007R-80431007R/4p/80B88VYFG/ref=sp\\_1\\_48?keywords=renault+megane+3+kombi&qid=1686666666](https://www.amazon.pl/VHEUHZ-kompatybilny-2010-2022-824310007R-80431007R/4p/80B88VYFG/ref=sp_1_48?keywords=renault+megane+3+kombi&qid=1686666666)

WHEUHZ Zawias drzwiowy stoper do drzwi przedni lewy/ prawy kompatybilny z Megane 3 Fluence L3 2010-2022 824310007R 80431007R [https://www.amazon.pl/VHEUHZ-kompatybilny-2010-2022-824310007R-80431007R/4p/80B88VYFG/ref=sp\\_1\\_48?keywords=renault+megane+3+kombi&qid=1686666666](https://www.amazon.pl/VHEUHZ-kompatybilny-2010-2022-824310007R-80431007R/4p/80B88VYFG/ref=sp_1_48?keywords=renault+megane+3+kombi&qid=1686666666)

WHEUHZ Zawias drzwiowy stoper do drzwi przedni lewy/ prawy kompatybilny z Megane 3 Fluence L3 2010-2022 824310007R 80431007R [https://www.amazon.pl/VHEUHZ-kompatybilny-2010-2022-824310007R-80431007R/4p/80B88VYFG/ref=sp\\_1\\_48?keywords=renault+megane+3+kombi&qid=1686666666](https://www.amazon.pl/VHEUHZ-kompatybilny-2010-2022-824310007R-80431007R/4p/80B88VYFG/ref=sp_1_48?keywords=renault+megane+3+kombi&qid=1686666666)

WHEUHZ Zawias drzwiowy stoper do drzwi przedni lewy/ prawy kompatybilny z Megane 3 Fluence L3 2010-2022 824310007R 80431007R [https://www.amazon.pl/VHEUHZ-kompatybilny-2010-2022-824310007R-80431007R/4p/80B88VYFG/ref=sp\\_1\\_48?keywords=renault+megane+3+kombi&qid=1686666666](https://www.amazon.pl/VHEUHZ-kompatybilny-2010-2022-824310007R-80431007R/4p/80B88VYFG/ref=sp_1_48?keywords=renault+megane+3+kombi&qid=1686666666)

WHEUHZ Zawias drzwiowy stoper do drzwi przedni lewy/ prawy kompatybilny z Megane 3 Fluence L3 2010-2022 824310007R 80431007R [https://www.amazon.pl/VHEUHZ-kompatybilny-2010-2022-824310007R-80431007R/4p/80B88VYFG/ref=sp\\_1\\_48?keywords=renault+megane+3+kombi&qid=1686666666](https://www.amazon.pl/VHEUHZ-kompatybilny-2010-2022-824310007R-80431007R/4p/80B88VYFG/ref=sp_1_48?keywords=renault+megane+3+kombi&qid=1686666666)

WHEUHZ Zawias drzwiowy stoper do drzwi przedni lewy/ prawy kompatybilny z Megane 3 Fluence L3 2010-2022 824310007R 80431007R [https://www.amazon.pl/VHEUHZ-kompatybilny-2010-2022-824310007R-80431007R/4p/80B88VYFG/ref=sp\\_1\\_48?keywords=renault+megane+3+kombi&qid=1686666666](https://www.amazon.pl/VHEUHZ-kompatybilny-2010-2022-824310007R-80431007R/4p/80B88VYFG/ref=sp_1_48?keywords=renault+megane+3+kombi&qid=1686666666)

WHEUHZ Zawias drzwiowy stoper do drzwi przedni lewy/ prawy kompatybilny z Megane 3 Fluence L3 2010-2022 824310007R 80431007R [https://www.amazon.pl/VHEUHZ-kompatybilny-2010-2022-824310007R-80431007R/4p/80B88VYFG/ref=sp\\_1\\_48?keywords=renault+megane+3+kombi&qid=1686666666](https://www.amazon.pl/VHEUHZ-kompatybilny-2010-2022-824310007R-80431007R/4p/80B88VYFG/ref=sp_1_48?keywords=renault+megane+3+kombi&qid=1686666666)

WHEUHZ Zawias drzwiowy stoper do drzwi przedni lewy/ prawy kompatybilny z Megane 3 Fluence L3 2010-2022 824310007R 80431007R [https://www.amazon.pl/VHEUHZ-kompatybilny-2010-2022-824310007R-80431007R/4p/80B88VYFG/ref=sp\\_1\\_48?keywords=renault+megane+3+kombi&qid=1686666666](https://www.amazon.pl/VHEUHZ-kompatybilny-2010-2022-824310007R-80431007R/4p/80B88VYFG/ref=sp_1_48?keywords=renault+megane+3+kombi&qid=1686666666)

WHEUHZ Zawias drzwiowy stoper do drzwi przedni lewy/ prawy kompatybilny z Megane 3 Fluence L3 2010-2022 824310007R 80431007R [https://www.amazon.pl/VHEUHZ-kompatybilny-2010-2022-824310007R-80431007R/4p/80B88VYFG/ref=sp\\_1\\_48?keywords=renault+megane+3+kombi&qid=1686666666](https://www.amazon.pl/VHEUHZ-kompatybilny-2010-2022-824310007R-80431007R/4p/80B88VYFG/ref=sp_1_48?keywords=renault+megane+3+kombi&qid=1686666666)

WHEUHZ Zawias drzwiowy stoper do drzwi przedni lewy/ prawy kompatybilny z Megane 3 Fluence L3 2010-2022 824310007R 80431007R [https://www.amazon.pl/VHEUHZ-kompatybilny-2010-2022-824310007R-80431007R/4p/80B88VYFG/ref=sp\\_1\\_48?keywords=renault+megane+3+kombi&qid=1686666666](https://www.amazon.pl/VHEUHZ-kompatybilny-2010-2022-824310007R-80431007R/4p/80B88VYFG/ref=sp_1_48?keywords=renault+megane+3+kombi&qid=1686666666)

WHEUHZ Zawias drzwiowy stoper do drzwi przedni lewy/ prawy kompatybilny z Megane 3 Fluence L3 2010-2022 824310007R 80431007R [https://www.amazon.pl/VHEUHZ-kompatybilny-2010-2022-824310007R-80431007R/4p/80B88VYFG/ref=sp\\_1\\_48?keywords=renault+megane+3+kombi&qid=1686666666](https://www.amazon.pl/VHEUHZ-kompatybilny-2010-2022-824310007R-80431007R/4p/80B88VYFG/ref=sp_1_48?keywords=renault+megane+3+kombi&qid=1686666666)

WHEUHZ Zawias drzwiowy stoper do drzwi przedni lewy/ prawy kompatybilny z Megane 3 Fluence L3 2010-2022 824310007R 80431007R [https://www.amazon.pl/VHEUHZ-kompatybilny-2010-2022-824310007R-80431007R/4p/80B88VYFG/ref=sp\\_1\\_48?keywords=renault+megane+3+kombi&qid=1686666666](https://www.amazon.pl/VHEUHZ-kompatybilny-2010-2022-824310007R-80431007R/4p/80B88VYFG/ref=sp_1_48?keywords=renault+megane+3+kombi&qid=1686666666)

WHEUHZ Zawias drzwiowy stoper do drzwi przedni lewy/ prawy kompatybilny z Megane 3 Fluence L3 2010-2022 824310007R 80431007R [https://www.amazon.pl/VHEUHZ-kompatybilny-2010-2022-824310007R-80431007R/4p/80B88VYFG/ref=sp\\_1\\_48?keywords=renault+megane+3+kombi&qid=1686666666](https://www.amazon.pl/VHEUHZ-kompatybilny-2010-2022-824310007R-80431007R/4p/80B88VYFG/ref=sp_1_48?keywords=renault+megane+3+kombi&qid=1686666666)

WHEUHZ Zawias drzwiowy stoper do drzwi przedni lewy/ prawy kompatybilny z Megane 3 Fluence L3 2010-2022 824310007R 80431007R [https://www.amazon.pl/VHEUHZ-kompatybilny-2010-2022-824310007R-80431007R/4p/80B88VYFG/ref=sp\\_1\\_48?keywords=renault+megane+3+kombi&qid=1686666666](https://www.amazon.pl/VHEUHZ-kompatybilny-2010-2022-824310007R-80431007R/4p/80B88VYFG/ref=sp_1_48?keywords=renault+megane+3+kombi&qid=1686666666)

WHEUHZ Zawias drzwiowy stoper do drzwi przedni lewy/ prawy kompatybilny z Megane 3 Fluence L3 2010-2022 824310007R 80431007R [https://www.amazon.pl/VHEUHZ-kompatybilny-2010-2022-824310007R-80431007R/4p/80B88VYFG/ref=sp\\_1\\_48?keywords=renault+megane+3+kombi&qid=1686666666](https://www.amazon.pl/VHEUHZ-kompatybilny-2010-2022-824310007R-80431007R/4p/80B88VYFG/ref=sp_1_48?keywords=renault+megane+3+kombi&qid=1686666666)

WHEUHZ Zawias drzwiowy stoper do drzwi przedni lewy/ prawy kompatybilny z Megane 3 Fluence L3 2010-2022 824310007R 80431007R [https://www.amazon.pl/VHEUHZ-kompatybilny-2010-2022-824310007R-80431007R/4p/80B88VYFG/ref=sp\\_1\\_48?keywords=renault+megane+3+kombi&qid=1686666666](https://www.amazon.pl/VHEUHZ-kompatybilny-2010-2022-824310007R-80431007R/4p/80B88VYFG/ref=sp_1_48?keywords=renault+megane+3+kombi&qid=1686666666)

WHEUHZ Zawias drzwiowy stoper do drzwi przedni lewy/ prawy kompatybilny z Megane 3 Fluence L3 2010-2022 824310007R 80431007R [https://www.amazon.pl/VHEUHZ-kompatybilny-2010-2022-824310007R-80431007R/4p/80B88VYFG/ref=sp\\_1\\_48?keywords=renault+megane+3+kombi&qid=1686666666](https://www.amazon.pl/VHEUHZ-kompatybilny-2010-2022-824310007R-80431007R/4p/80B88VYFG/ref=sp_1_48?keywords=renault+megane+3+kombi&qid=1686666666)

WHEUHZ Zawias drzwiowy stoper do drzwi przedni lewy/ prawy kompatybilny z Megane 3 Fluence L3 2010-2022 824310007R 80431007R [https://www.amazon.pl/VHEUHZ-kompatybilny-2010-2022-824310007R-80431007R/4p/80B88VYFG/ref=sp\\_1\\_48?keywords=renault+megane+3+kombi&qid=1686666666](https://www.amazon.pl/VHEUHZ-kompatybilny-2010-2022-824310007R-80431007R/4p/80B88VYFG/ref=sp_1_48?keywords=renault+megane+3+kombi&qid=1686666666)

WHEUHZ Zawias drzwiowy stoper do drzwi przedni lewy/ prawy kompatybilny z Megane 3 Fluence L3 2010-2022 824310007R 80431007R [https://www.amazon.pl/VHEUHZ-kompatybilny-2010-2022-824310007R-80431007R/4p/80B88VYFG/ref=sp\\_1\\_48?keywords=renault+megane+3+kombi&qid=1686666666](https://www.amazon.pl/VHEUHZ-kompatybilny-2010-2022-824310007R-80431007R/4p/80B88VYFG/ref=sp_1_48?keywords=renault+megane+3+kombi&qid=1686666666)

WHEUHZ Zawias drzwiowy stoper do drzwi przedni lewy/ prawy kompatybilny z Megane 3 Fluence L3 2010-2022 824310007R 80431007R [https://www.amazon.pl/VHEUHZ-kompatybilny-2010-2022-824310007R-80431007R/4p/80B88VYFG/ref=sp\\_1\\_48?keywords=renault+megane+3+kombi&qid=1686666666](https://www.amazon.pl/VHEUHZ-kompatybilny-2010-2022-824310007R-80431007R/4p/80B88VYFG/ref=sp_1_48?keywords=renault+megane+3+kombi&qid=1686666666)

WHEUHZ Zawias drzwiowy stoper do drzwi przedni lewy/ prawy kompatybilny z Megane 3 Fluence L3 2010-2022 824310007R 80431007R [https://www.amazon.pl/VHEUHZ-kompatybilny-2010-2022-824310007R-80431007R/4p/80B88VYFG/ref=sp\\_1\\_48?keywords=renault+megane+3+kombi&qid=1686666666](https://www.amazon.pl/VHEUHZ-kompatybilny-2010-2022-824310007R-80431007R/4p/80B88VYFG/ref=sp_1_48?keywords=renault+megane+3+kombi&qid=1686666666)

WHEUHZ Zawias drzwiowy stoper do drzwi przedni lewy/ prawy kompatybilny z Megane 3 Fluence L3 2010-2022 824310007R 80431007R [https://www.amazon.pl/VHEUHZ-kompatybilny-2010-2022-824310007R-80431007R/4p/80B88VYFG/ref=sp\\_1\\_48?keywords=renault+megane+3+kombi&qid=1686666666](https://www.amazon.pl/VHEUHZ-kompatybilny-2010-2022-824310007R-80431007R/4p/80B88VYFG/ref=sp_1_48?keywords=renault+megane+3+kombi&qid=1686666666)

WHEUHZ Zawias drzwiowy stoper do drzwi przedni lewy/ prawy kompatybilny z Megane 3 Fluence L3 2010-2022 824310007R 80431007R [https://www.amazon.pl/VHEUHZ-kompatybilny-2010-2022-824310007R-80431007R/4p/80B88VYFG/ref=sp\\_1\\_48?keywords=renault+megane+3+kombi&qid=1686666666](https://www.amazon.pl/VHEUHZ-kompatybilny-2010-2022-824310007R-80431007R/4p/80B88VYFG/ref=sp_1_48?keywords=renault+megane+3+kombi&qid=1686666666)

WHEUHZ Zawias drzwiowy stoper do drzwi przedni lewy/ prawy kompatybilny z Megane 3 Fluence L3 2010-2022 824310007R 80431007R [https://www.amazon.pl/VHEUHZ-kompatybilny-2010-2022-824310007R-80431007R/4p/80B88VYFG/ref=sp\\_1\\_48?keywords=renault+megane+3+kombi&qid=1686666666](https://www.amazon.pl/VHEUHZ-kompatybilny-2010-2022-824310007R-80431007R/4p/80B88VYFG/ref=sp_1_48?keywords=renault+megane+3+kombi&qid=1686666666)

WHEUHZ Zawias drzwiowy stoper do drzwi przedni lewy/ prawy kompatybilny z Megane 3 Fluence L3 2010-2022 824310007R 80431007R [https://www.amazon.pl/VHEUHZ-kompatybilny-2010-2022-824310007R-80431007R/4p/80B88VYFG/ref=sp\\_1\\_48?keywords=renault+megane+3+kombi&qid=1686666666](https://www.amazon.pl/VHEUHZ-kompatybilny-2010-2022-824310007R-80431007R/4p/80B88VYFG/ref=sp_1_48?keywords=renault+megane+3+kombi&qid=1686666666)

WHEUHZ Zawias drzwiowy stoper do drzwi przedni lewy/ prawy kompatybilny z Megane 3 Fluence L3 2010-2022 824310007R 80431007R [https://www.amazon.pl/VHEUHZ-kompatybilny-2010-2022-824310007R-80431007R/4p/80B88VYFG/ref=sp\\_1\\_48?keywords=renault+megane+3+kombi&qid=1686666666](https://www.amazon.pl/VHEUHZ-kompatybilny-2010-2022-824310007R-80431007R/4p/80B88VYFG/ref=sp_1_48?keywords=renault+megane+3+kombi&qid=1686666666)

WHEUHZ Zawias drzwiowy stoper do drzwi przedni lewy/ prawy kompatybilny z Megane 3 Fluence L3 2010-2022 824310007R 80431007R [https://www.amazon.pl/VHEUHZ-kompatybilny-2010-2022-824310007R-80431007R/4p/80B88VYFG/ref=sp\\_1\\_48?keywords=renault+megane+3+kombi&qid=1686666666](https://www.amazon.pl/VHEUHZ-kompatybilny-2010-2022-824310007R-80431007R/4p/80B88VYFG/ref=sp_1_48?keywords=renault+megane+3+kombi&qid=1686666666)

WHEUHZ Zawias drzwiowy stoper do drzwi przedni lewy/ prawy kompatybilny z Megane 3 Fluence L3 2010-2022 824310007R 80431007R [https://www.amazon.pl/VHEUHZ-kompatybilny-2010-2022-824310007R-80431007R/4p/80B88VYFG/ref=sp\\_1\\_48?keywords=renault+megane+3+kombi&qid=1686666666](https://www.amazon.pl/VHEUHZ-kompatybilny-2010-2022-824310007R-80431007R/4p/80B88VYFG/ref=sp_1_48?keywords=renault+megane+3+kombi&qid=1686666666)

WHEUHZ Zawias drzwiowy stoper do drzwi przedni lewy/ prawy kompatybilny z Megane 3 Fluence L3 2010-2022 824310007R 80431007R [https://www.amazon.pl/VHEUHZ-kompatybilny-2010-2022-824310007R-80431007R/4p/80B88VYFG/ref=sp\\_1\\_48?keywords=renault+megane+3+kombi&qid=1686666666](https://www.amazon.pl/VHEUHZ-kompatybilny-2010-2022-824310007R-80431007R/4p/80B88VYFG/ref=sp_1_48?keywords=renault+megane+3+kombi&qid=1686666666)

WHEUHZ Zawias drzwiowy stoper do drzwi przedni lewy/ prawy kompatybilny z Megane 3 Fluence L3 2010-2022 824310007R 80431007R [https://www.amazon.pl/VHEUHZ-kompatybilny-2010-2022-824310007R-80431007R/4p/80B88VYFG/ref=sp\\_1\\_48?keywords=renault+megane+3+kombi&qid=1686666666](https://www.amazon.pl/VHEUHZ-kompatybilny-2010-2022-824310007R-80431007R/4p/80B88VYFG/ref=sp_1_48?keywords=renault+megane+3+kombi&qid=1686666666)

WHEUHZ Zawias drzwiowy stoper do drzwi przedni lewy/ prawy kompatybilny z Megane 3 Fluence L3 2010-2022 824310007R 80431007R [https://www.amazon.pl/VHEUHZ-kompatybilny-2010-2022-824310007R-80431007R/4p/80B88VYFG/ref=sp\\_1\\_48?keywords=renault+megane+3+kombi&qid=1686666666](https://www.amazon.pl/VHEUHZ-kompatybilny-2010-2022-824310007R-80431007R/4p/80B88VYFG/ref=sp_1_48?keywords=renault+megane+3+kombi&qid=1686666666)

WHEUHZ Zawias drzwiowy stoper do drzwi przedni lewy/ prawy kompatybilny z Megane 3 Fluence L3 2010-2022 824310007R 80431007R [https://www.amazon.pl/VHEUHZ-kompatybilny-2010-2022-824310007R-80431007R/4p/80B88VYFG/ref=sp\\_1\\_48?keywords=renault+megane+3+kombi&qid=1686666666](https://www.amazon.pl/VHEUHZ-kompatybilny-2010-2022-824310007R-80431007R/4p/80B88VYFG/ref=sp_1_48?keywords=renault+megane+3+kombi&qid=1686666666)

WHEUHZ Zawias drzwiowy stoper do drzwi przedni lewy/ prawy kompatybilny z Megane 3 Fluence L3 2010-2022 824310007R 80431007R [https://www.amazon.pl/VHEUHZ-kompatybilny-2010-2022-824310007R-80431007R/4p/80B88VYFG/ref=sp\\_1\\_48?keywords=renault+megane+3+kombi&qid=1686666666](https://www.amazon.pl/VHEUHZ-kompatybilny-2010-2022-824310007R-80431007R/4p/80B88VYFG/ref=sp_1_48?keywords=renault+megane+3+kombi&qid=1686666666)

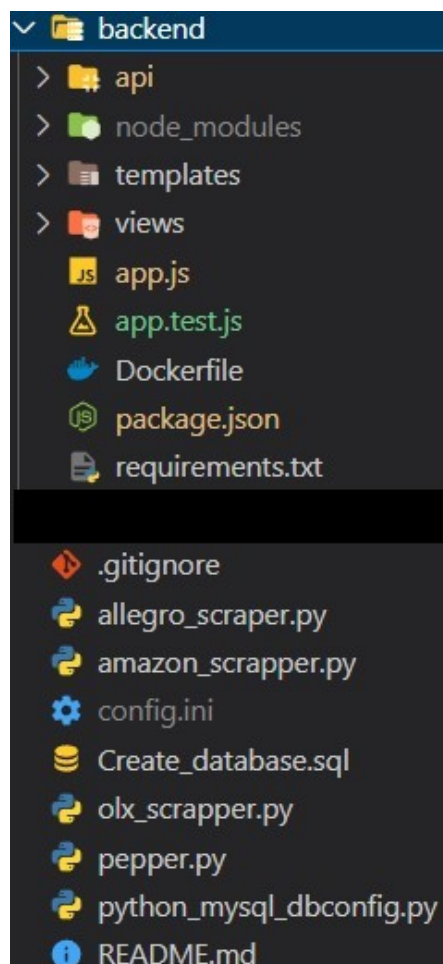
WHEUHZ Zawias drzwiowy stoper do drzwi przedni lewy/ prawy kompatybilny z Megane 3 Fluence L3 2010-2022 824310007R 80431007R [https://www.amazon.pl/VHEUHZ-kompatybilny-2010-2022-824310007R-80431007R/4p/80B88VYFG/ref=sp\\_1\\_48?keywords=renault+megane+3+kombi&qid=1686666666](https://www.amazon.pl/VHEUHZ-kompatybilny-2010-2022-824310007R-80431007R/4p/80B88VYFG/ref=sp_1_48?keywords=renault+megane+3+kombi&qid=1686666666)

WHEUHZ Zawias drzwiowy stoper do drzwi przedni lewy/ prawy kompatybilny z Megane 3 Fluence L3 2010-2022 824310007R 80431007R [https://www.amazon.pl/VHEUHZ-kompatybilny-2010-2022-824310007R-80431007R/4p/80B88VYFG/ref=sp\\_1\\_48?keywords=renault+megane+3+kombi&qid=1686666666](https://www.amazon.pl/VHEUHZ-kompatybilny-2010-2022-824310007R-80431007R/4p/80B88VYFG/ref=sp_1_48?keywords=renault+megane+3+kombi&qid=1686666666)

WHEUHZ Zawias drzwiowy stoper do drzwi przedni lewy/ prawy kompatybilny z Megane 3 Fluence L3 2010-2022 824310007R 80431007R [https://www.amazon.pl/VHEUHZ-kompatybilny-2010-2022-824310007R-80431007R/4p/80B88VYFG/ref=sp\\_1\\_48?keywords=renault+megane+3+kombi&qid=1686666666](https://www.amazon.pl/VHEUHZ-kompatybilny-2010-2022-824310007R-80431007R/4p/80B88VYFG/ref=sp_1_48?keywords=renault+megane+3+kombi&qid=1686666666)

WHEUHZ Zawias drzwiowy stoper do drzwi przedni lewy/ prawy kompatybilny z Megane 3 Fluence L3 2010-2022 824310007R 80431007R [https://www.amazon.pl/VHEUHZ-kompatybilny-2010-2022-824310007R-80431007R/4p/80B88VYFG/ref=sp\\_1\\_48?keywords=renault+megane+3+kombi&qid=16866](https://www.amazon.pl/VHEUHZ-kompatybilny-2010-2022-824310007R-80431007R/4p/80B88VYFG/ref=sp_1_48?keywords=renault+megane+3+kombi&qid=1686666666)





Rysunek 5.11: Struktura części backendowej serwera.

W programach napisanych w języku *Python* występuje podział na cztery skrypty dotyczące pobierania danych oddzielnie z każdego serwisu, t.j. *allegro\_scrapper.py*, *amazon\_scrapper.py*, *olx\_scrapper.py*, *pepper.py*. Dodatkowo znajduje się tam plik *python\_mysql\_dbconfig.py* odpowiadający za wczytanie danych z pliku *config.ini*. Znajdują się w nim wszystkie dane potrzebne do poświadczeń do poszczególnych serwisów, tak aby połączyć się z bazą danych *MySQL*. Konfiguracja ta została pokazana na Rys. 5.12.

Skrypty utworzono osobno, gdyż każdy serwis posiada inną strukturę i w każdym trzeba wyodrębnić inne elementy. Przykładowo dla serwisu *OLX*, tak jak pokazano we fragmencie kodu 5.2, zostaje wpisany konkretny *URL*, wraz ze zwrotem “final\_phrase”, który jest frazą wpisaną przez użytkownika i podawaną jako argument wejściowy przy uruchomieniu podanego skryptu. Następnie jest doklejany odpowiedni nagłówek i zostaje wysłane zapytanie do strony. Następnie strona zostaje przeszukana w celu znalezienia odpowiednich elementów. Gdy uzyska się potrzebne informacje wywoływana jest funkcja, która umieszcza informacje w bazie danych. Gdy program się zakończy to wypisuje na ekranie liczbę rekordów umieszczonych w bazie.

```

[mysql]
; host = 127.0.0.1
host = 192.168.0.150
database = artykuly
user = root2
; password =
password =

[mysql_dialect]
dialect = mysql

[tokens]
TWILIO_NUMBER=+16063667571
PERSONAL_NUMBER=+48532832333
TWILIO_ACCOUNT_SID=
TWILIO_AUTH_TOKEN=
; Token do serwera discord
BOT_TOKEN=

[discord]
friendsChannelId = 108457
myUserId = 10845

[email]
mailSender=shipgamesender@gmail.com
mailSenderPassword=

[allegro]
CLIENT_ID = 9096efe8383644ee91a44d1de4c637f6
CLIENT_SECRET =

```

Rysunek 5.12: Dane konfiguracyjne aplikacji w pliku *config.ini*.

Bardzo podobnie działają skrypty *pepper.py* pobierając dane z serwisu *Pepper* poprzez URL: [https://www.pepper.pl/search?q={final\\_phrase}](https://www.pepper.pl/search?q={final_phrase}) oraz skrypt *amazon\_scrapper.py*, który pobiera dane z adresu: [https://www.amazon.pl/s?k={final\\_phrase}](https://www.amazon.pl/s?k={final_phrase}). Inne podejście zastosowano dla serwisu *Allegro*.

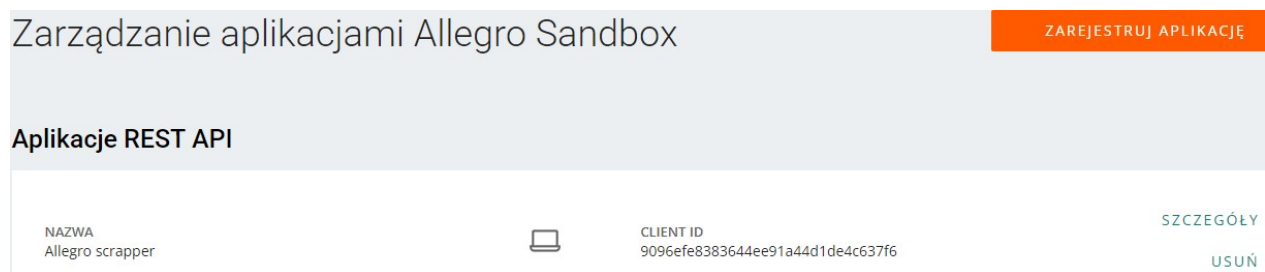
```

1 URL = F"https://www.olx.pl/oferty/q-{final_phrase}/"
2 page = requests.get(URL, headers = {
3     'User-Agent': 'Popular browser\'s user-agent',
4 })
5 page_content = BeautifulSoup(page.content, "html.parser")
6 results = page_content.find_all("div", {"data-cy": "l-card"})
7 for a in results:
8     arr=[]
9     title = a.find_all("h6", {"class": "css-16v5mdi"})
10    link = a.find_all("a", {"class": "css-rc5s2u"})
11    price = a.find_all("p", {"data-testid": "ad-price"})
12    img = a.find_all("img", {"class": "css-8wsg1m"})
13    new=a.find_all("span", {"title": "Nowe"})
14    used=a.find_all("span", {"title": "Uzywane"})
15    location=a.find_all("p", {"data-testid": "location-date"})
16    link_to_page= ''

```

Listing 5.2: Przykład z programu *olx\_scrapper.py*.

Ponieważ regulamin serwisu *Allegro* zabrania bezpośredniego scrapowania danych użyto publicznego *API*. Jednym z endpointów, który pozwala wyszukiwać i przeglądać oferty jest “GET /offers/listing”. Jednak działa on tylko z kontami zweryfikowanymi, więc trzeba najpierw przejść poprawną weryfikację ze strony *Allegro*, aby konto było zweryfikowane. Na potrzeby tej pracy nie udało się zweryfikować konta, lecz dzięki środowisku testowemu (Sandbox) jest możliwość skorzystania z tej metody. Aby uzyskać dostęp do *API Allegro* należy zarejestrować aplikację i powiązać ją z kontem. Wymagana jest nazwa aplikacji, jej rodzaj (koniecznie wybrać działanie aplikacji w środowisku bez dostępu do przeglądarki) oraz uprawnienia aplikacji. Po poprawnym zarejestrowaniu aplikacji zostaną wygenerowane “Client ID” oraz “Client Secret”, dzięki którym możliwe jest uzyskanie tokenu dostępowego do *API* (Rys. 5.13). W programie *allegro\_scraper.py* wykorzystano właśnie powyższe dane, uzyskano token dostępowy poprzez *URL* przedstawiony we fragmencie kodu 5.3. Po uzyskaniu dostępu jest odpytany konkretny endpoint pokazany we fragmencie kodu 5.4 wraz z szukaną frazą wpisaną przez użytkownika i wysyłaną jako argument programu. Tak jak w poprzednich programach dane te zapisywane są w bazie danych.



Rysunek 5.13: Zarejestrowane aplikacje serwisie *AllegroSandbox*.

```

1 CLIENT_ID=config['allegro']['CLIENT_ID']
2 CLIENT_SECRET=config['allegro']['CLIENT_SECRET']
3 TOKEN_URL = "https://allegro.pl.allegrosandbox.pl/auth/oauth/token"
4 def get_access_token():
5     try:
6         data = {'grant_type': 'client_credentials'}
7         access_token_response = requests.post(TOKEN_URL, data=data,
8             verify=False, allow_redirects=False, auth=(CLIENT_ID,
9                 CLIENT_SECRET))
10        tokens = json.loads(access_token_response.text)
11        access_token = tokens['access_token']
12        return access_token
13    except requests.exceptions.HTTPError as err:
14        raise SystemExit(err)
15 access_token = get_access_token()

```

Listing 5.3: Przykład z programu *allegro\_scrapper.py*.

```

1 url = "https://api.allegro.pl/allegrosandbox.pl/offers/listing?phrase
    =f"{phrase}&fallback=false&limit=f"{limit}"
2 headers = {'Authorization': 'Bearer ' + token, 'Accept': "application
    /vnd.allegro.public.v1+json"}

```

Listing 5.4: Przykład z programu *allegro\_scrapper.py*.

Aplikacja frontendowa komunikuje się z serwerem poprzez *API*, *REST*. Struktura danych *API* została przedstawiona na wcześniejszym Rys. 5.5. Struktura wewnętrzna serwisu *API* została zorganizowana następująco:

1. routes - ścieżki konkretnych endpointów.
2. models - modele - struktury, dzięki którym *Sequelize* komunikuje się bazą danych.
3. controllers - kontrolery, odpowiadające za przetwarzanie zapytań i uruchamianie odpowiednich usług.
4. servicelayer - warstwa, odpowiadająca za wywołanie konkretnych metod, które dostają się do warstwy niższej.
5. dataaccesslayer - warstwa, która czyta i zapisuje dane do bazy, używa *Sequelize*.

Poniżej w tabeli 5.1 przedstawiono spis metod udostępnianych przez *API*.



Tabela 5.1: Tabela opisująca poszczególne metody *API*.

Metoda	Ścieżka	Opis
GET	/getUser/:login	Zwraca informację o tym, czy znaleziono lub nie znaleziono użytkownika, oraz zwraca jego ID
GET	/getConfiguration/:login	Zwraca informację o tym, czy znaleziono lub nie znaleziono konfiguracji użytkownika, oraz zwraca jego konfigurację
POST	/registerUser	Zwraca informację o tym, czy można zarejestrować użytkownika, jeżeli tak to rejestruje użytkownika
POST	/login	Zwraca informację o tym, czy można zalogować użytkownika, jeżeli tak to loguje użytkownika
POST	/saveConfiguration	Zwraca informację o tym, czy można skonfigurować (w przypadku nowego użytkownika) lub zaktualizować konfigurację użytkownika istniejącego
POST	/email	Wysyła mail i zwraca informację o poprawnym lub niepoprawnym wysłaniu
POST	/discord	Wysyła wiadomość do serwisu <i>Discord</i> i zwraca informację o poprawnym lub niepoprawnym wysłaniu
POST	/sms	Wysyła wiadomość SMS i zwraca informację o poprawnym lub niepoprawnym wysłaniu
POST	/getData	Pobiera dane z serwisów wyznaczonych dla użytkownika i zwraca informację o pobranych danych
POST	/cronJob	Tworzy zadania wysyłania powiadomień dla użytkownika i zwraca informację o czy udało się poprawnie zarejestrować zadanie
POST	/deleteJobsForUser	Usuwa wszystkie działające zadania dla użytkownika i zwraca informację o ilości usuniętych zadań

Główny plik aplikacji to *app.js*, który poprzez określone ścieżki (routes) łączy się z odpowiednimi serwisami. Przebieg obsługi zapytania dla jednego z punktów końcowych przebiega następująco: wysyłane jest zapytanie na konkretny endpoint, który jest zdefiniowany w routes, następnie jest wyzwalany dany kontroler, który uruchamia program z *serviceLayer*, a z kolei ten wywołuje konkretną funkcję w *dataAccessLayer* (*DAL*), która ma już bezpośredni dostęp do danych.

W zakresie funkcjonalności wysyłania powiadomień platforma oferuje możliwość wysyłania powiadomień e-mail, SMS oraz *Discord*. Dla e-maili wykorzystywana jest biblioteka *nodemailer*, natomiast serwis, który wysyła wiadomości przez protokół *SMTP* to *gmail*. Aby móc wysyłać maile poprzez gmail trzeba wygenerować hasło do aplikacji poprzez wejście w opcje weryfikacji dwuetapowej na koncie użytkownika. Zostało tutaj wygenerowane hasło,

które użyto w konfiguracji *nodemailer* (pokazane na Rys. 5.14). Konfiguracja *nodemailer* pokazana została na we fragmencie kodu 5.5. W przypadku wysłania maila i informacji z nim związanych wykorzystano bibliotekę *handlebars.js*, która pozwala na stworzenie szablonu *HTML*. W folderze *templates*(Rys. 5.11) znajduje się plik *template.hbs*, do którego są przekazywane informacje z bazy danych, a następnie szablon ten w postaci *HTML* jest wysyłany na konkretny e-mail, a mianowicie na e-mail jaki użytkownik podał przy rejestracji.

## ← Weryfikacja dwuetapowa

Weryfikacja dwuetapowa jest WŁĄCZONA od 12 mar 2023

WYŁĄCZ

### Dostępne drugie etapy

Celem drugiego etapu po podaniu hasła jest sprawdzenie, czy to Ty się logujesz. [Więcej informacji](#)

**Uwaga:** potwierdzenia od Google zostaną dodane jako kolejna metoda przy weryfikacji dwuetapowej podczas logowania na konto Google na dowolnym zgodnym telefonie.

### Hasła do aplikacji

Hasła do aplikacji nie są zalecane i w większości przypadków są niepotrzebne. Aby chronić swoje konto Google, łącz je z aplikacjami za pomocą funkcji „Zaloguj się przez Google”.

#### Hasła do aplikacji

1 hasło

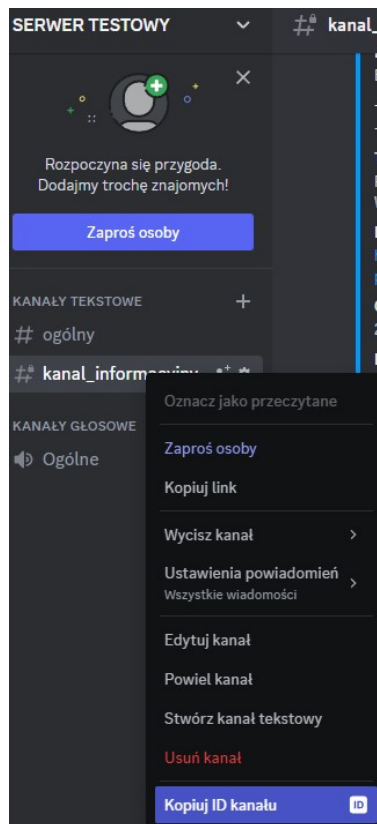


Rysunek 5.14: Widok opcji konfiguracji z poziomu *Gmail*.

```
1 let mailTransporter = nodemailer.createTransport({
2   service: "gmail",
3   auth: {
4     user: config.email.mailSender,
5     pass: config.email.mailSenderPassword
6   },
7   tls: {
8     rejectUnauthorized: false,
9   }
10 });
```

Listing 5.5: Przykład konfiguracji *nodemailer*.

Wysyłanie wiadomości do serwisu *Discord* odbywa się poprzez kolejną usługę. Aby wysłać wiadomość utworzono serwer *Discord* z kanałem tekstowym. Po utworzeniu serwera połączono aplikację na koncie developera (<https://discord.com/developers/applications>) wraz z serwerem *Discord*. Został wygenerowany token, dzięki któremu uzyskano dostęp, do zarządzania botem wysyłającym wiadomości. Następnie pobrano ID kanału, na który będą wysyłane wiadomości (zostało to pokazane na Rys. 5.15). Wysyłanie informacji do serwisu *Discord* odbywa się za pomocą biblioteki *Discord.js*. Przykładowy kawałek kodu wysyłający wiadomości został pokazany w 5.6. Wykorzystano też klasę *EmbedBuilder* z pakietu *discord.js* pozwalającą na podzielenie wiadomości na części, gdyż *Discord* ogranicza wielkość wysyłanej wiadomości do 2000 znaków. W przyszłości serwis zostanie wyposażony w pola do dodania dla konfiguracji, aby każdy użytkownik mógł wysyłać wiadomości do swojego serwera. W tym momencie dane te są brane z pliku *config.ini*.



Rysunek 5.15: Pobranie ID kanału z serwisu *Discord*.

```

1 const channel = client.channels.cache.get(config.discord.channelId);
2 //....
3 return await channel.send({ embeds: arrayOfEmbeds })
4 .then(message => console.log('Wyslano wiadomosc do serwisu Discord!' )
5   )
6 .catch(console.error);
7 //.....
8 client.login(config.tokens.BOT_TOKEN);

```

Listing 5.6: Przykład wysyłania wiadomości do serwisu *Discord*.

Ostatnią opcją wysyłania powiadomień, którą oferuje opracowana platforma jest przesyłanie SMS. Wykorzystano tutaj serwis *Twilio*<sup>3</sup>, czyli popularny system do komunikacji SMS. Tutaj również, aby wysłać wiadomość SMS potrzeba danych dostępowych do aplikacji, czyli *Account SID* (*String Identifier*), *Auth Token* oraz *Twilio phone number* (pokazano na Rys. 5.16). Niestety aplikacja nie pozwala na wysyłanie wiadomości do konkretnych numerów bezpłatnie. Istnieje tylko opcja wysłania wiadomości do własnego numeru telefonu podawanego przy rejestracji w serwisie *Twilio*. W tej pracy wiadomości SMS można wysłać tylko na jeden konkretny numer podany w pliku *config.ini*, gdyż w celu powiązania innych numerów i wysyłania tam wiadomości, trzeba rozszerzyć plan konta *Twilio*. Przykładowy kod wysyłający SMS został pokazany we fragmencie kodu 5.7.

Rysunek 5.16: Dane z platformy *Twilio*.

```

1 async function sendSMStoPhone(textMessage) {
2   sentFrom=config.tokens.TWILIO_NUMBER;
3   sentTo=config.tokens.PERSONAL_NUMBER; //konto trial umożliwia tylko
4     wysyłanie SMS do siebie
5   twilio.messages.create({
6     body: textMessage,
7     from: sentFrom,
8     to: sentTo
9   })

```

<sup>3</sup>Strona projektu: <https://console.twilio.com/>

```

10 .then(message =>{
11     console.log('SMS wyslany z ${sentFrom} do ${sentTo}. SID
        wiadomosci: ${message.sid}');
12 })
13 .catch((error) => {
14     console.error(error);
15 });
16 }

```

Listing 5.7: Przykład wysyłania wiadomości SMS.

W przypadku pozostałych plików lub funkcji wartych omówienia są jeszcze pliki *cron.js* oraz plik *getDataFromScriptsController.js*. Pierwszy odpowiada bezpośrednio za zadanie tworzone przez użytkownika oraz czas wysłania powiadomień. W zależności od tego co wprowadzi użytkownik powiadomienia konkretnych serwisów będą wysyłane w czasie określonym przez użytkownika. W tym celu jest używana biblioteka *node-cron*. Jeżeli chodzi o drugi plik to odpowiada on za uruchamianie skryptów *Pythona* w celu scrapowania danych. Wykorzystuje on paczkę *child\_process* i funkcję *spawn()*, która pozwala na na tworzenie nowego procesu przy użyciu podanej komendy z argumentami wiersza poleceń. Przykładowe wywołanie takiej komendy to “python olx\_scrapper.py "zegarek””. Gdy wszystkie dane zostaną pobrane zwraca odpowiedni komunikat wraz z danymi. Kod działa asynchronicznie przez co procesy się nie blokują.

### 5.3.3 Architektura mikroserwisowa w Node.js

Architektura mikroserwisowa wykorzystuje *API* zgodne z zasadami *REST*, a opracowana platforma udostępnia interfejs *API*. Struktura aplikacji została podzielona na warstwy jak pokazano na Rys. 5.5. W katalogu “routes” są skonfigurowane ścieżki, przez które przyjmują zapytanie *API* i przekierowują je do niższych warstw. Poniżej pokazano działanie jednego z mikroserwisów dla endpointu “/email”. Zapytanie do *API* jest przyjmowane przez serwer i wywoływana jest metoda kontrolera “sendMailController()”. Pokazano we fragmencie kodu 5.8.

```

1 var router = express.Router();
2 const email=require('../controllers/emailController');
3 // .....
4 router.post('/email', email.sendMailController);

```

Listing 5.8: Ścieżka dla endpointu “/email”.

Następnie kontroler przekazuje dane do metody “sendMailService()” i w przypadku pomyślnego wywołania metody zwraca status 201 i komunikat o poprawnym działaniu. W przypadku błędnego działania metody, kontroler zwróci status 500 i odpowiednią wiadomość. Przedstawiono na wydruku kodu 5.9.

```

1 const emailService=require('../servicelayer/email_service');
2 //.....
3 const sendMailController= async (req,res)=>{
4   try{
5     const sendMail= await emailService.sendMailService(req);
6     res.status(201).json({status: 'OK', message: 'Poprawnie
7       wyslano email!'}));
8   }
9   catch(err){
10    res.status(500).send({
11      message: err.message || "Wystapil blad w trakcie
12        wysylania maila!"
13    });
14  }
15 }

```

Listing 5.9: Implementacja metody “sendMailController()” w warstwie kontrolera.

Dalej dane są przekazywane do serwisu, który przekazuje dane do metody warstwy łączącej się bezpośrednio z bazą danych. Metoda “sendMailService()” zwraca wynik funkcji “sendMail()” lub null w przypadku błędu. Pokazano we fragmencie kodu 5.10.

```

1 const emailDal=require('../dataaccesslayer/emailsDal');
2 //.....
3 const sendMailService = async (req)=>{
4   try {
5     const sendMail = await emailDal.sendMail(req);
6     return sendMail;
7   }
8   catch (error) {
9     return null;
10  }
11 }

```

Listing 5.10: Implementacja metody “sendMailService()” w warstwie serwisowej.

Na końcu dane są przekazywane do warstwy “DataAccessLayer”, w której występuje odwołanie do modelu komunikującego się z bazą danych. Gdy dane zostaną pobrane z bazy danych to przekazywane są przez kolejne warstwy, czyli warstwę serwisu, warstwę kontrolera i na końcu bezpośrednio do routera, który jest odpowiedzialny za mapowanie konkretnych żądań na odpowiednie funkcje obsługujące te żądania. Przedstawiono na wydruku kodu 5.11.

```

1 const Users=require('../models/Users_model');
2 //.....
3 const sendMail = async (req)=>{
4
5   let receiver=null;
6   let {user,data}=req.body;
7
8   try {
9     //.....
10    receiver= await Users.Users_models.findOne({

```

```

11         where: {
12             id: parseInt(user)
13         }
14     });
15     //.....
16     let mailOptions = {
17         from: config.email.mailSender,
18         to: receiver.email,
19         subject: "Oferty sprzedazowe - mailing!",
20         html: htmlToSend
21     };
22
23     return mailTransporter.sendMail(mailOptions, function (err,
24         data) {
25         if (err) {
26             console.error("Wystapil blad:", err.message);
27         } else {
28             console.log("-----");
29             console.log("Email zostal wyslany prawidlowo,
30                 komunikat: " + data.response);
31         }
32     });
33 } catch (err) {
34     console.error(err);
35 }

```

Listing 5.11: Implementacja metody “sendMail()” w warstwie dostępu do danych (“DataAccessLayer”).

W ten sposób jeden mikroservis wykonuje określoną czynność poprzez przejście od routera do bazy danych i w odwrotnym kierunku zwracając końcową odpowiedź. Pozostałe mikroservisy również spełniają podobne funkcje do przedstawionego powyżej tworząc w ten sposób architekturę mikroservisową.

Aplikację frontendową oraz backendową można również potraktować jako dwa osobne serwisy komunikujące się ze sobą za pomocą stylu architektury *REST*. Dodatkowo te dwa środowiska są uruchamiane za pomocą kontenerów *Docker* dzięki czemu są od siebie odseparowane i łatwiejsze w zarządzaniu.

### 5.3.4 Komunikacja między kontenerami

Używając oprogramowania *Docker* zarówno część frontendowa jak i backendowa jest zamknięta w osobnych kontenerach. Dla potrzeb pracy baza danych *MySQL* również została umieszczona w kontenerze (przedstawia to Rys. 5.2). Środowisko jest tworzone przez silnik *Dockera* za pomocą plików *Dockerfile*. Pliki przedstawiono na wydrukach kodu 5.12 oraz 5.13.

```

1 FROM node:16-alpine
2
3 RUN mkdir -p /home/react/app/node\_modules && chown -R node:node /
  home/react/app
4
5 WORKDIR /home/react/app
6
7 COPY --chown=node:node /frontend/my-test-app/package*.json ./
8
9 USER node
10
11 RUN npm install
12
13 COPY --chown=node:node /frontend/my-test-app/ .
14
15 ENV NODE_ENV production
16
17 RUN npm run build
18 EXPOSE 3000
19 CMD ["npm", "start"]

```

Listing 5.12: Plik *Dockerfile* dla frontendu.

```

1 FROM ubuntu:22.04
2
3 RUN mkdir -p /usr/src/app/backend/node_modules
4
5 WORKDIR /usr/src/app/backend
6
7 RUN apt-get update && apt-get install -y \
8   locales \
9   python3-pip \
10  curl
11
12 COPY /backend/package*.json ./
13 COPY /backend/requirements.txt ./
14
15 RUN apt-get update -y
16 RUN apt-get upgrade -y
17 RUN apt install nodejs -y
18 RUN apt install npm -y
19
20 RUN npm cache clean -f
21 RUN npm config set strict-ssl false
22 RUN npm install -g n
23 RUN n stable
24 RUN ln -sf /usr/local/n/versions/node/7.8.0/bin/node /usr/bin/node
25
26 RUN npm install
27
28 RUN apt-get install -y python3

```



```

29
30 COPY --chown=node:node /backend/ .
31 COPY --chown=node:node /*.py /usr/src/app/
32 COPY --chown=node:node /*.ini /usr/src/app/
33
34 RUN pip install -r requirements.txt
35
36 EXPOSE 9005
37
38 CMD [ "node", "app.js" ]

```

Listing 5.13: Plik *Dockerfile* dla backendu.

Aby stworzyć kontener *Dockera* najpierw należy zbudować obraz. Robi się to za pomocą polecenia “docker build”. Gdy taki obraz zostanie utworzony wykonując polecenie “docker run” stworzymy kontener i uruchamiamy go na podstawie przygotowanego wcześniej obrazu. Komendy budujące obraz i uruchamiające kontener wykonywane są z głównego katalogu projektu. Dla środowiska frontendowego kod tworzący kontener pokazano na wydruku 5.14.

```

1 $ docker build -t frontend_app_react -f .\frontend\my-test-app\
  Dockerfile .
2 $ docker run -p 3000:3000 frontend_app_react

```

Listing 5.14: Kod tworzący kontener dla frontendu.

Za pomocą flagi “-t” nadaje się tag(nazwę) obrazowi. Za pomocą flagi “-f” specyfikuje się ścieżkę, gdzie znajduje się plik *Dockerfile*, dzięki któremu zostanie utworzony obraz *Dockera*. Kropka oznacza bieżący katalog roboczy. Następnie za pomocą drugiej komendy uruchamia się kontener z podaną nazwą i za pomocą flagi “-p” specyfikuje się port zewnątrz oraz wewnętrzny kontenera. Dla środowiska backendowego proces wdrażania odbywa się za pomocą poleceń pokazanych na wydruku kodu 5.15.

```

1 $ docker build -t node_server -f .\backend\Dockerfile .
2 $ docker run -p 9005:9005 -it --add-host=host.docker.internal:host-
  gateway node_server

```

Listing 5.15: Kod tworzący kontener dla backendu.

Komendy wyglądają bardzo podobnie jak w przypadku środowiska frontendowego z tą różnicą, że przy uruchamianiu kontenera dodawane są dwie flagi: “-it”, która pozwala na odpalenie kontenera w trybie interaktywnym z możliwością wejścia do środka kontenera oraz “-add-host=host.docker.internal:host-gateway”, która umożliwi bezpośrednie połączenia kontenera z lokalną maszyną i usługami tam działającymi, np. system *Windows*, dzięki czemu będzie możliwe połączenie się z bazą danych i środowiskiem frontendowym. Dla bazy danych proces wdrażania odbywa się za pomocą poleceń pokazanych we fragmencie kodu 5.16:
















```

1 $ docker pull mysql/mysql-server:latest
2 $ docker run -p3306:3306 --name=mysql_container -e MYSQL_ROOT_HOST=%
  -e MYSQL_ROOT_PASSWORD=my-secret-pw -d mysql/mysql-server:5.7

```

Listing 5.16: Kod pobierający i uruchamiający obraz bazy danych MySQL.

Za pomocą komendy “docker pull” pobierany jest obraz bazy danych *MySQL* z repozytorium *Dockera* w serwisie *Dockerhub*. Następnie uruchamiany jest kontener na porcie 3306 pod nazwą *mysql\_container*. Za pomocą flagi “-e” ustawia się zmienne środowiskowe *MYSQL\_ROOT\_HOST* i *MYSQL\_ROOT\_PASSWORD*, czyli zmienne, które utworzą użytkownika *root* z hasłem. Przez podanie % w zmiennej *MYSQL\_ROOT\_HOST* zezwala się na logowanie z dowolnego adresu IP. Po pierwszym uruchomieniu kontenera z bazą danych należy wejść do środka tego kontenera i wykonać skrypt *create\_database.sql* tworzący tabele, powiązania oraz trigger na bazie danych. Kolejność uruchomienia kontenerów to kolejno kontener z bazą danych, kontener ze środowiskiem backendowym oraz kontener ze środowiskiem frontendowym. Działające kontenery przedstawiono na Rys. 5.17.

Name	Image ↑	Status	Port(s)	Last started ↓	Actions
 priceless_blackburn 1c22b6a048be 	<a href="#">frontend_app_react</a>	Running	<a href="#">3000:3000</a> 	20 seconds ago 	
 vigorous_driscoll 6110fa721df5 	<a href="#">node_server</a>	Running	<a href="#">9005:9005</a> 	25 seconds ago 	
 mysql_container 5b8cde7647df 	<a href="#">mysql/mysql-server:5.7</a>	Running	<a href="#">3306:3306</a> 	30 seconds ago 	

Rysunek 5.17: Działające kontenery w środowisku Docker.

## Rozdział 6

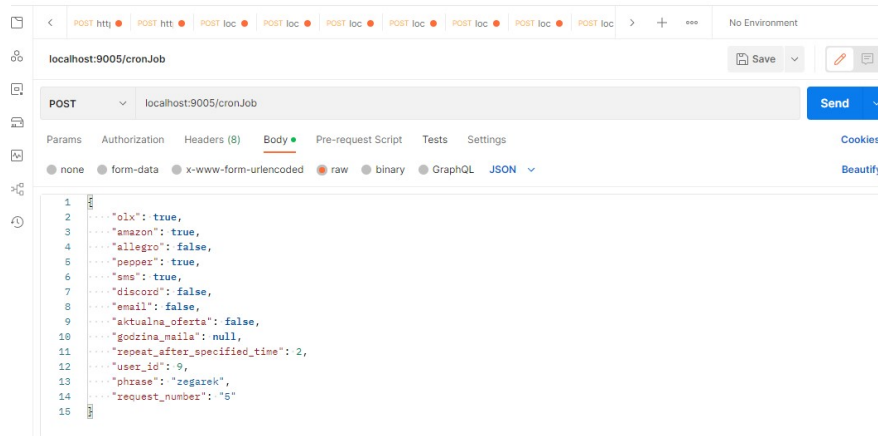
# Testy i ocena aplikacji

W tym rozdziale przedstawiona została ocena wydajności aplikacji oraz pokazane zostały testy platformy. Przed przystąpieniem do wdrożenia aplikacji na serwer produkcyjny wymagane jest wykonanie testów danego programu. Testowanie można podzielić na dwa typy: testowanie ręczne oraz testowanie automatyczne. Różnią się one między sobą tym, że w testowaniu manualnym tester nie używa narzędzi do testów automatycznych, tylko sprawdza wszystkie cechy aplikacji i wykonuje przypadki testowe przechodząc przez aplikację samodzielnie. Zaś w testach automatycznych, dzięki odpowiednim bibliotekom czy środowiskom, testy wykonywane są przez skrypty.

### 6.1 Badanie aplikacji pod kątem defektów: testowanie oraz analiza błędów

Dzięki testom automatycznym udało się zautomatyzować testowanie szczególnych przypadków, w których mogło dochodzić do błędów. Ponieważ program udostępnia *API*, zatem pierwszym narzędziem, który służył do testowania był *Postman*, czyli aplikacja służąca do testowania endpointów udostępnianych przez aplikację. Widok programu przedstawiono na Rys. 6.1.

Do zautomatyzowania testów potrzebne jest jednak dodatkowe oprogramowanie. Taką biblioteką jest *Jest*, który służy do testowania programów napisanych w języku *Javascript*. Biblioteka zapewnia tworzenie zarówno testów jednostkowych jak i funkcjonalnych. Posiada wiele funkcji, które ułatwiają testowanie napisanego kodu. Narzędzie *Jest* zostało zaprojektowane z myślą o prostocie, więc nie wymaga konfiguracji, a dzięki testom, które są zapisane w odpowiednich plikach automatycznie uruchamia testy. Jest jednym z najbardziej popularnych narzędzi do testowania w środowisku *Javascript*. Na Rys. 6.2 przedstawiono pomyślne przejście testów dla dwóch przypadków testowych. Testy uruchamia się komendą “run npm test”, gdy w pliku *package.json*, jest wpisana linia ““test”:”jest””. Przykładowy kod dla testu jednego z endpointów pokazano we fragmencie kodu: 6.1



Rysunek 6.1: Widok z programu Postman dla konkretnego zasobu.

```

1 describe("Testy dla uzytkownikow", () => {
2     it('Pobranie uzytkownika o loginie test', async() => {
3         const response = await requests(app).get("/getUser/test");
4         expect(response.body.user_id).toEqual(9);
5         expect(response.statusCode).toBe(201);
6     });
7 });

```

Listing 6.1: Przykładowy kod dla testu endpointu /getUser/:login.

## 6.2 Ocena efektywności platformy

Po przeprowadzonych testach platformy działa ona zgodnie z założeniami. Umożliwia rejestrację oraz logowanie użytkowników. Po wybraniu odpowiedniej konfiguracji przez użytkownika platforma dostarcza powiadomienia w czasie rzeczywistym lub wysyła wiadomości do serwisów wybranych przez użytkownika. Platforma jest skalowalna i elastyczna, przez co umożliwia jej to dalszy rozwój. Po przeanalizowaniu dostępnych na rynku rozwiązań, stwierdzono, że istnieją aplikacje podobne do tej przedstawionej w pracy, jednak nie są one takie same. Aplikacja jest prosta i intuicyjna przez co będzie łatwa w użytkowaniu dla nowych klientów.

```
PASS ./app.test.js
  Console

    console.log
      Początek testow
        at Object.log (app.test.js:11:13)

    console.log
      Listening on 9005 ...
        at Sequelize.log (node_modules/sequ...

    console.log
      Koniec testow
        at Object.log (app.test.js:16:13)

Test Suites: 1 passed, 1 total
Tests:       2 passed, 2 total
Snapshots:   0 total
Time:        1.78 s, estimated 2 s
Ran all test suites.
```

Rysunek 6.2: Pomyślne przejście testów z użyciem narzędzia *Jest*.

## Rozdział 7

# Podsumowanie

Aktualnie coraz więcej osób korzysta z programów służących do powiadamiania o aktywnych ofertach sprzedażowych. Aplikacja do powiadomień o ofertach sprzedażowych to rozwiązanie, które sprawdzi się zarówno dla ludzi wymagających oraz tych, którzy chcą uzyskać jak najszybszy efekt. Umożliwia ona spersonalizowanie konkretnych wiadomości wysyłanych do systemów wybranych przez użytkownika. Wykorzystanie architektury mikroserwisowej pozwala na łatwe zarządzanie strukturą aplikacji przez co w każdej chwili można edytować lub dodawać nowe funkcjonalności.

Niniejsza praca dotyczyła przygotowania aplikacji, której tematem była mikroserwisowa platforma powiadomień o ofertach sprzedażowych. Założeniem pracy było napisanie programu, dzięki któremu użytkownik będzie mógł zarejestrować się i zalogować do aplikacji, a następnie w konfiguracji wybrać platformy, z których chce pobierać informacje o produktach. Kolejno użytkownikowi udostępniono wybranie możliwości powiadomień do konkretnych serwisów. Za pomocą szukanej frazy i ilości zapytań platforma miała dostarczyć użytkownikowi szukane oferty w czasie rzeczywistym lub dostarczać je w formie powiadomień co określony czas lub o określonej godzinie i minucie. Zaimplementowana aplikacja zrealizowała koncepcje charakteryzując się również wydajnością i niezawodnością. Serwer może przetwarzać wiele zapytań wysyłanych do niego jednocześnie. Również dzięki zastosowaniu podziału na moduły umożliwia się platformie rozrastanie bez konieczności dużych zmian w kodzie aplikacji.

Jednymi z głównych ograniczeń funkcjonalnych platformy jest liczba serwisów, z których dane będą pobierane oraz usług, do których wiadomości te będą przesyłane. Obecnie wybór ogranicza się tylko do kilku opcji. Ograniczeniem są również serwisy, z których dane te są scrapowane, tak jak np. *Allegro*, które blokuje pobieranie poprzez kod programu zawartości z ich stron. Kolejną możliwą niedogodnością jest czas przetwarzania wiadomości przez serwer, który może się wydłużyć w przypadku dużej ilości zapytań oraz wybrania wszystkich dostępnych serwisów.

Możliwości rozwoju tego rodzaju aplikacji jest wiele. Obecnie platforma działa tylko dla kilku serwisów (*OLX*, *Amazon*, *Pepper*, *Allegro*), przy czym korzystanie z serwisu *Allegro* odbywa się tylko przez platformę testową. Planowany jest podział platformy na działy takie

jak motoryzacja (szukanie samochodów), budownictwo (szukanie domów na wynajem i na sprzedaż), praca (oferty pracy) i wiele innych. Również dla powiadomień przewidziane są nowe opcje, jak dodanie nowych serwisów, do których można przysyłać wiadomości w postaci *Twittera* czy *Telegrama*. Istnieje także możliwość dodania opcji, aby aplikacja sama wysyłała powiadomienia. W planach przewiduje się również dołączenie dodatkowych workerów<sup>1</sup>, dzięki którym można byłoby ustawić wiele zadań powiadamiania użytkownika. Istnieje też pomysł dodania szablonów, w których użytkownicy będą mogli specyfikować wygląd powiadomień.

Dzięki łatwej konfiguracji platformy konkretni użytkownicy lub nawet firmy korzystające z tego oprogramowania będą miały rozwiązanie, dzięki któremu będą mogły trafić do większej liczby klientów. Przez to zwiększy się także sprzedaż poszczególnych produktów oraz zadowolenie klientów. Użytkownicy będą mieli szybszy dostęp do interesujących ofert i nie stracą czasu na niepotrzebne przeszukiwanie wielu stron.

---

<sup>1</sup>worker - proces wykonujący określoną czynność.

# Bibliografia

- [1] [https://en.wikipedia.org/wiki/Web\\_scraping](https://en.wikipedia.org/wiki/Web_scraping) (dostępność 25.05.2023r.)
- [2] **Hirschey, J. K. (2014):** Symbiotic Relationships: Pragmatic Acceptance of Data Scraping
- [3] <https://digitalmoves.co.uk/web-scraping/> (dostępność 25.05.2023r.)
- [4] <https://mirosławmamczur.pl/web-scraping-co-to-i-jakie-sa-dobre-praktyki/> (dostępność 25.05.2023r.)
- [5] <https://adboosters.pl/pozycjonowanie/pozycjonowanie-torun/> (dostępność 25.05.2023r.)
- [6] <https://scrapingrobot.com/blog/scraping-technique/> (dostępność 25.05.2023r.)
- [7] <https://pl.wikipedia.org/wiki/Honeypot> (dostępność 25.05.2023r.)
- [8] <https://pl.wikipedia.org/wiki/CAPTCHA> (dostępność 25.05.2023r.)
- [9] <https://chrome.google.com/webstore/detail/web-scraper-free-web-scr/jnhgnonknehpejjnehehlkklplmbmhn> (dostępność 25.05.2023r.)
- [10] <https://pl.wikipedia.org/wiki/ETL> (dostępność 25.05.2023r.)
- [11] [https://en.wikipedia.org/wiki/Extract,\\_load,\\_transform](https://en.wikipedia.org/wiki/Extract,_load,_transform) (dostępność 25.05.2023r.)
- [12] <https://etl-tools.info/pl/etl.html> (dostępność 25.05.2023r.)
- [13] <https://datacamel.pl/etl/> (dostępność 25.05.2023r.)
- [14] <https://icproject.com/blog/cenna-wiedza/powiadomienia-czym-sa-i-co-warto-o-nich-wiedziec/> (dostępność 25.05.2023r.)
- [15] <https://www.marketing-automation.pl/kompletny-przewodnik-po-powiadomieniach-push/> (dostępność 25.05.2023r.)
- [16] [https://pl.wikipedia.org/wiki/Alert\\_RCB](https://pl.wikipedia.org/wiki/Alert_RCB) (dostępność 25.05.2023r.)



- [17] <https://clevertap.com/blog/what-are-push-notifications/> (dostępność 25.05.2023r.)
- [18] <https://pl.wikipedia.org/wiki/Mikroserwisy> (dostępność 25.05.2023r.)
- [19] <https://www.bdabek.pl/architektura-mikroserwisow> (dostępność 25.05.2023r.)
- [20] **Krause L. (2015):** Microservices: Patterns and Applications. Symbiotic Relationships: Pragmatic Acceptance of Data Scraping
- [21] <https://bykowski.pl/mikroserwisy-wprowadzenie-i-praktyczny-przyklad> (dostępność 25.05.2023r.)
- [22] [https://pl.wikipedia.org/wiki/Architektura\\_zorientowana\\_na\\_uslugi](https://pl.wikipedia.org/wiki/Architektura_zorientowana_na_uslugi) (dostępność 25.05.2023r.)
- [23] <https://www.jcommerce.pl/jpro/artykuly/mikroserwisy-nowa-jakosc-w-miedzynarodowych-projektach-it> (dostępność 25.05.2023r.)
- [24] <https://oktawave.com/pl/blog/chmura-obliczeniowa/technologie-chmury/co-to-sa-mikroserwisy> (dostępność 25.05.2023r.)
- [25] <https://www.statworx.com/en/content-hub/blog/why-you-should-use-containerized-microservices-when-deploying-your-data-science-application> (dostępność 25.05.2023r.)
- [26] **Fowler M. (2002):** Patterns of Enterprise Application Architecture
- [27] <https://pl.wikipedia.org/wiki/Model-View-Controller> (dostępność 25.05.2023r.)