Kebab Finder

Mateusz Wisiecki - aplikacja mobilna
Wiktor Wojciechowski - panel administratora
Wojciech Worach - API

Informatyka — PAM — semestr 7

Przedmiot: Projekt Zespołowy

1 Opis przedmiotu zamówienia

1. Użytkownik może pobrać aplikację na systemy Android i iOS.
2. Użytkownik aplikacji mobilnej może zobaczyć listę legnickich kebabów:
   - 2.1 Lista w formie tabeli jest paginowana, sortowalna i filtrowalna.
   - 2.2 Lista w formie mapy jest interaktywna.
   - 2.3 Kliknięcie na miejsce (zarówno na liście, jak i na mapie) otwiera informacje szczegółowe o danym miejscu.
3. Lista obejmuje kebaby obecnie działające, zamknięte oraz planowane.
4. Każdy kebab powinien mieć przypisane informacje podstawowe:
   - 4.1 nazwę,
   - 4.2 logo,
   - 4.3 adres,
   - 4.4 koordynaty geograficzne,
   - 4.5 rok otwarcia i zamknięcia (jeżeli informacje są znane).
5. Każdy kebab powinien mieć przypisane informacje dodatkowe:
   - 5.1 godziny otwarcia w poszczególnych dniach tygodnia,
   - 5.2 dostępne rodzaje mięs (kurczak, wołowina, baranina, falafel itp.),
   - 5.3 dostępne sosy (czosnkowy, ostry itp.),
   - 5.4 informację nt. statusu (punkt 3 OPZ),
   - 5.5 informację, czy kebab jest "kraftowy",
   - 5.6 informację, czy kebab jest ulokowany w nieruchomości, czy w "budzie",
   - 5.7 informację, czy kebab jest oddziałem sieci kebabów,
   - 5.8 informację, jak można zamówić kebab (telefon, Pyszne.pl, Glovo, Uber Eats, własna aplikacja/strona).
6. Informacje z punktów 5.1–5.8 OPZ powinny być bazą dla filtrów z punktu 2.1.
7. Informacje z punktów 4.1, 4.5, 8 OPZ powinny być bazą dla sortowań z punktu 2.1.
8. Każdy kebab powinien mieć przypisane oceny z Google i Pyszne.pl, jeżeli mają tam konta:
   - 8.1 Informacje te są pobierane z właściwych serwisów przynajmniej raz dziennie.
9. Każdy kebab powinien mieć przypisane linki do social media (Facebook, Instagram, własna strona www).
10. Użytkownik widzi liczbę kebabów w aplikacji mobilnej.
11. Użytkownik może zalogować się w aplikacji mobilnej:

11.1 Zalogowany użytkownik może dodawać kebaby do ulubionych.

11.2 Zalogowany użytkownik może komentować kebaby.

11.3 Zalogowany użytkownik może przesyłać administratorowi sugestię dotyczącą zmian danych.

12. Administrator zarządza bazą kebabów poprzez dedykowany panel administracyjny:

12.1 Konto administratora tworzone jest podczas instalacji systemu.

12.2 Przy pierwszym logowaniu administrator musi stworzyć sobie hasło.

13. Administrator widzi listę zgłoszonych sugestii i akceptuje je lub odrzuca.

14. Aplikacja mobilna korzysta z REST API panelu administracyjnego, które jest opisane przez dokumentację OpenAPI wystawioną w endpoincie `/api/documentation`.

15. System posiada zmigrowane dane nt. legnickich kebabów.

2 Opis technologiczny API

2.1 Podstawowe technologie

- **PHP**: Projekt wymaga wersji PHP w wersji `8.2` lub nowszej.
- **Laravel Framework**: Używana wersja `11.9`, która dostarcza zaawansowane funkcjonalności dla aplikacji webowych.
- **SQLite**: Lekka baza danych SQL, która nie wymagająca oddzielnego serwera.

2.2 Główne zależności

- **darkaonline/l5-swagger** (`8.6`): Narzędzie do generowania dokumentacji API w oparciu o standard OpenAPI.
- **guzzlehttp/guzzle** (`7.9`): Klient HTTP do obsługi zapytań do zewnętrznych API.
- **laravel/breeze** (`2.2`): Pakiet upraszczający tworzenie systemu uwierzytelniania.
- **laravel/sanctum** (`4.0`): Pakiet do zarządzania tokenami API oraz uwierzytelnianiem.
- **symfony/css-selector** i **symfony/dom-crawler** (`7.2`): Komponenty do pracy z selektorami CSS oraz przeszukiwania DOM.

2.3 Zależności developerskie

- **fakerphp/faker** (`1.23`): Narzędzie do generowania danych testowych.
- **laravel/pail** (`1.1`): Rozszerzenie do zarządzania plikami logów.
- **laravel/pint** (`1.13`): Narzędzie do formatowania kodu zgodnie ze standardami PSR-12.

- **laravel/sail** (`1.26`): Środowisko do uruchamiania projektu w Dockerze.
- **mockery/mockery** (`1.6`): Biblioteka do tworzenia mocków w testach.
- **phpunit/phpunit** (`11.0.1`): Framework do testów jednostkowych.
- **nunomaduro/collision** (`8.1`): Narzędzie wspomagające debugowanie i raportowanie błędów.

3 Opis technologiczny panel administratora

3.1 Podstawowe technologie

- **React** (`18.3.1`) React to biblioteka javascript umożliwiająca tworzenie aplikacji SPA i tworzenie interfejsu na podstawie komponentów, które można dynamicznie przeładowywać.
- **Node.js** (`20.18.0`): Środowisko uruchomieniowe javascript, wymagane do działania aplikacji React. Razem z npm wymagane do instalacji zależności.

3.2 Główne zależności

- **leaflet** (`1.9.4`): Open-sourceowa biblioteka javascript umożliwiająca tworzenie interaktywnych map
- **react-router-dom** (`6.27.0`): Ułatwia stworzenie routingu w aplikacjach React
- **react-scripts** (`5.0.1`): część konfiguracji Create React App, obejmuje wstępnie skonfigurowane build-tools, narzędzia lintingowe (eslint) i inne narzędzia usprawniające pracę w React.

3.3 Zależności developerskie

- **tailwindcss** (`3.4.14`): framework CSS posiadający wiele klas do stylizacji elementów interfejsu aplikacji

4 Opis technologiczny aplikacja mobilna

4.1 Podstawowe technologie

- **React Native**: W wersji `0.76.2` Framework opracowany przez Facebooka, który umożliwia tworzenie aplikacji mobilnych na systemy iOS i Android za pomocą języka JavaScript i biblioteki React.
- **React**: W wersji `18.3.1` biblioteka JavaScript, opracowana przez Facebook, która dostarcza podstawowy fundament do budowania interfejsów użytkownika. Głównym zadaniem react jest ułatwienie tworzenia komponentowych i deklaratywnych interfejsów użytkownika

- **Expo**: platforma w wersji `52.0.7` do tworzenia aplikacji mobilnych w React Native. Ułatwia proces tworzenia, testowania i wdrażania aplikacji, oferując zestaw narzędzi i bibliotek, które eliminują potrzebę ręcznej konfiguracji środowiska

4.2 Główne zależności

- **react-native-safe-area-context** (4.12.0): Oferuje bezpieczne obszary (safe areas), np. w okolicy wycięć na ekranie.
- **expo/vector-icons** (14.0.2): Zestaw popularnych ikon zintegrowany z Expo
- **react-native-picker/picker** (2.10.2): Komponent do wyboru wartości z listy rozwijanej
- **react-native-maps** (1.18.0): Dostarcza komponenty do wyświetlania map i interakcji z nimi
- **react-native-dots-pagination** (0.3.1): Biblioteka do tworzenia wskaźników paginacji w formie kropek
- **react-native-timer-picker** (2.0.2): Komponent do wyboru czasu w aplikacji
- **react-native-async-storage/async-storage** (1.23.1): Udostępnia miejsce które pozwala na przechowywanie danych lokalnych w formacie klucz-wartość między widokami.
- **axios** (1.7.7): Biblioteka JavaScript używana do tworzenia zapytań HTTP

4.3 Zależności developerskie

- **babel/core** (7.20.0): Główna biblioteka Babel, która odpowiada za transpilację (zamianę jednego języka programowania na inny) nowoczesnego kodu JavaScript do wersji kompatybilnych z przeglądarkami lub środowiskami, które nie obsługują najnowszych standardów.
- **typescript** (5.3.3): Język programowania będący nadzbiorem JavaScript, wprowadzający typowanie statyczne i wiele innych funkcji, które poprawiają jakość i bezpieczeństwo kodu
- **types/react** (18.3.12): Deklaracje typów dla Reacta, umożliwiające korzystanie z TypeScript w projektach React.
- **jest** (29.5.12): Framework do testów jednostkowych i integracyjnych
- **types/jest** (29.5.12): Deklaracje typów dla frameworka testowego Jest, co pozwala używać TypeScript podczas pisania testów.

5 Estymacja zadań

Tabela 1: API

Zadanie	**Estymowany czas**	**Rzeczywisty czas**
Initialize project	2	2
Make database	2	4
Make sauces controller	1	2
Make meat types controller	1	1
Make Kebabs controller	6	7
Make favourite controller	1	0.3
Make comments controller	2	2.5
Make reports controller	3	3
Users controller and admin account	3	5
Get and assign reviews once a day	5	12

Tabela 2: Panel administratora

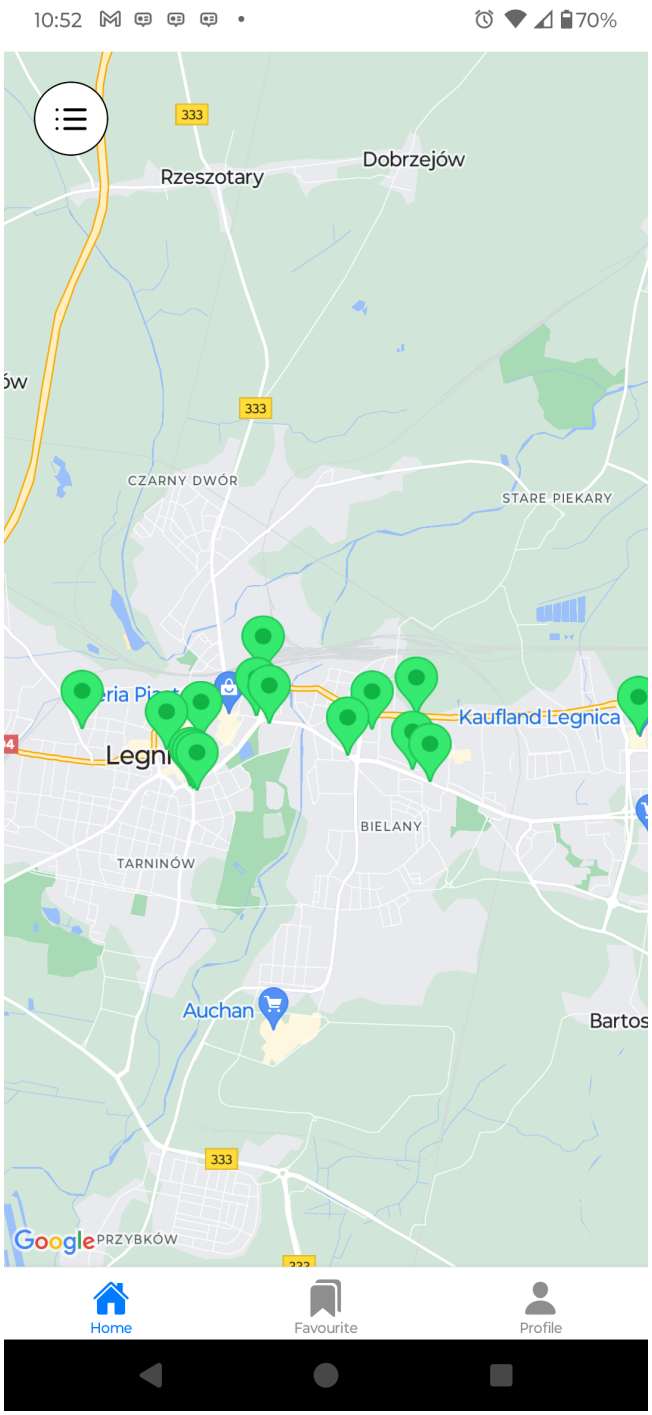
Zadanie	**Estymowany czas**	**Rzeczywisty czas**
Make admin login panel to admin panel	3	4
Add interactive map with kebab information to admin panel	2	3
Admin edit kebab	3	2
Admin add kebab panel	3	4
Admin sauces panel	2	1
Add admin meat types panel	2	0.5
Add navbar to admin panel	1	0.5
Admin reports panel	2	1

Tabela 3: Aplikacja mobilna

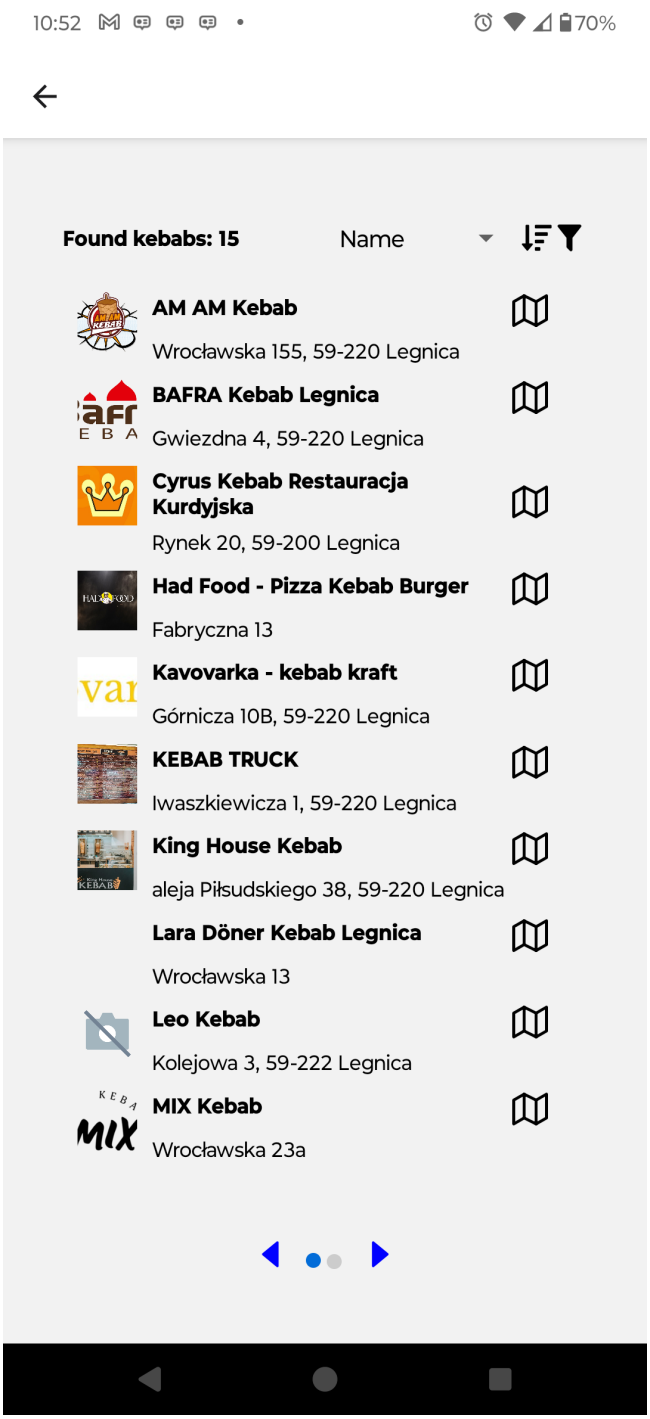
Zadanie	**Estymowany czas**	**Rzeczywisty czas**
Add user registration and login	3	5
Add a kebab map	2	2
Add kebab comments	3	4
Favourite kebabs	2	2
Add user profile edit page	1	1.5
User logout	1	1
Add kebab list	4	4
Kebab reports	2	1
Kebab list filter and sort	5	8
Navbar	1	1

6 Interfejs

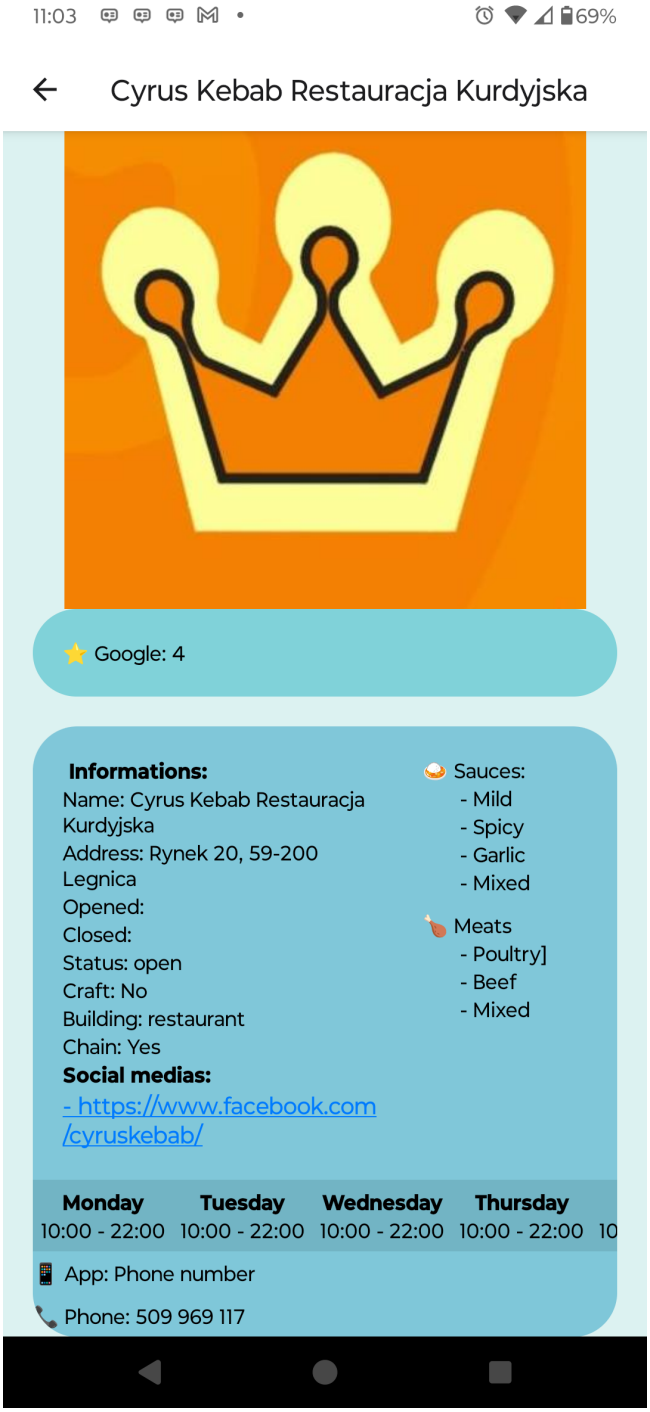
6.1 Interfejs aplikacji mobilnej



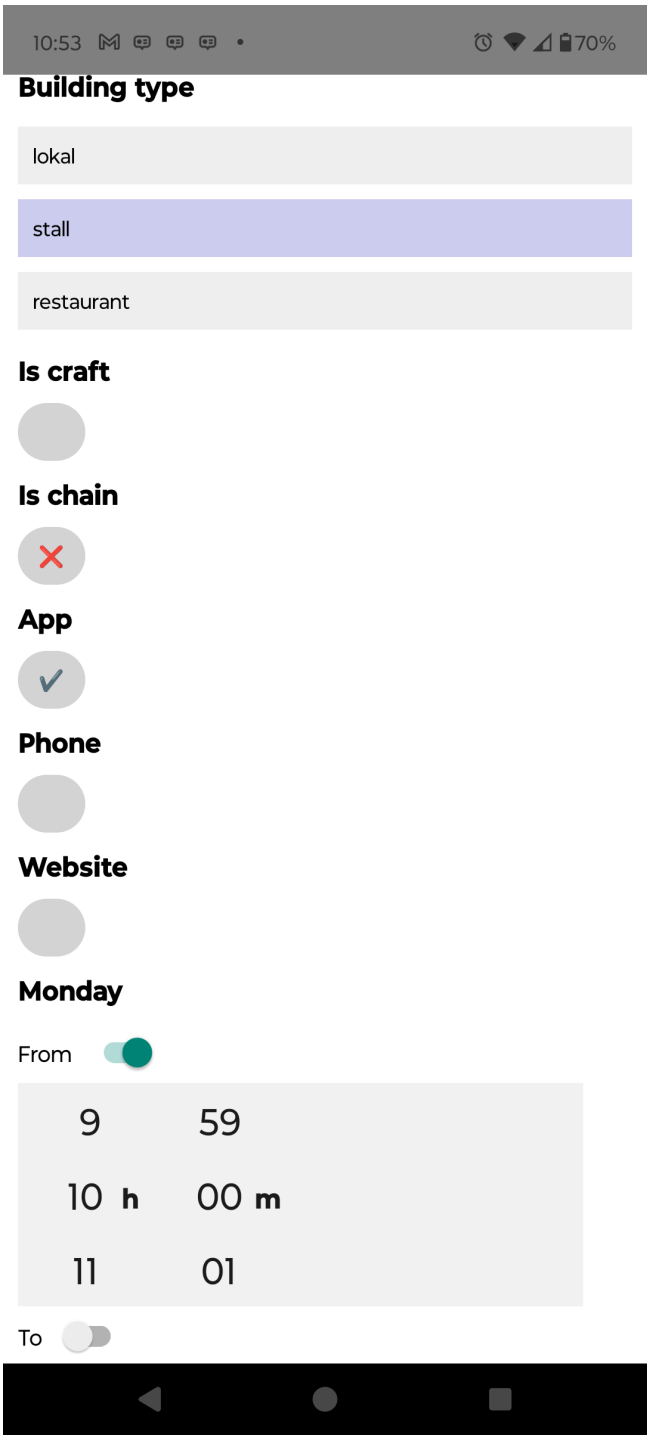
Rysunek 1: Widok mapy kebabów



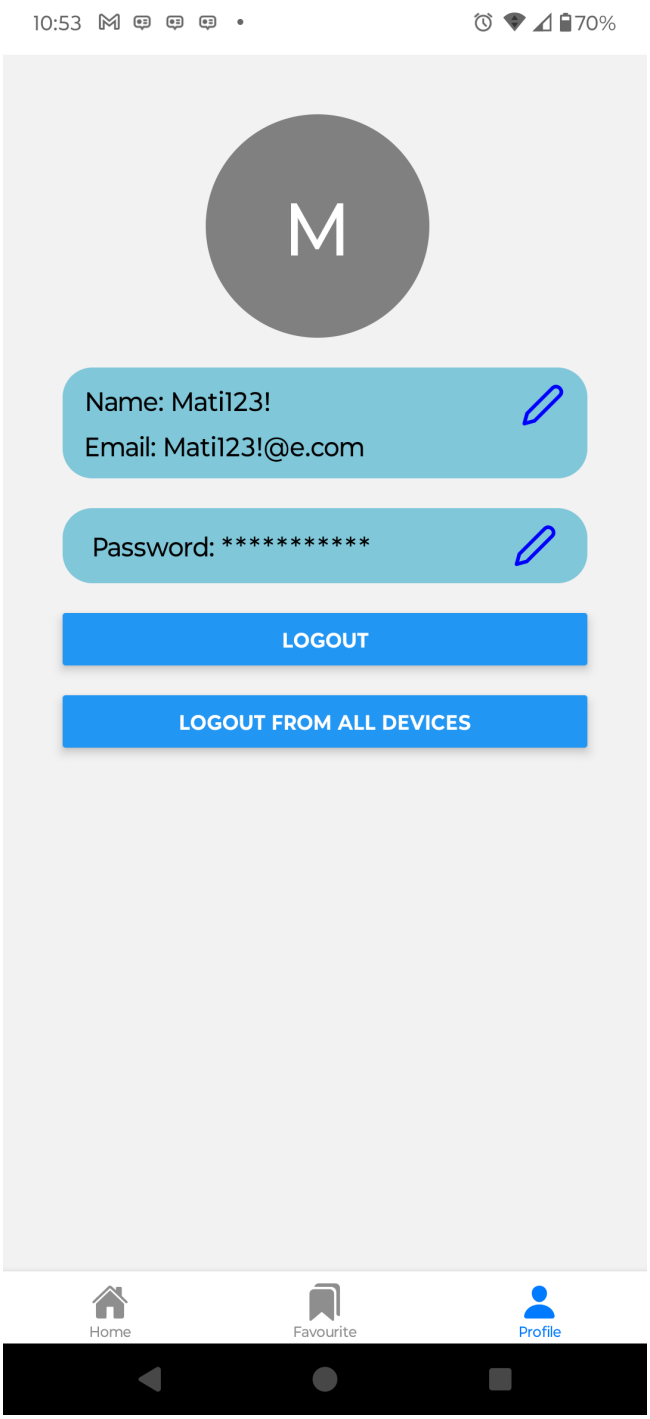
Rysunek 2: Widok listy kebabów



Rysunek 3: Widok szczegółów o kebabie

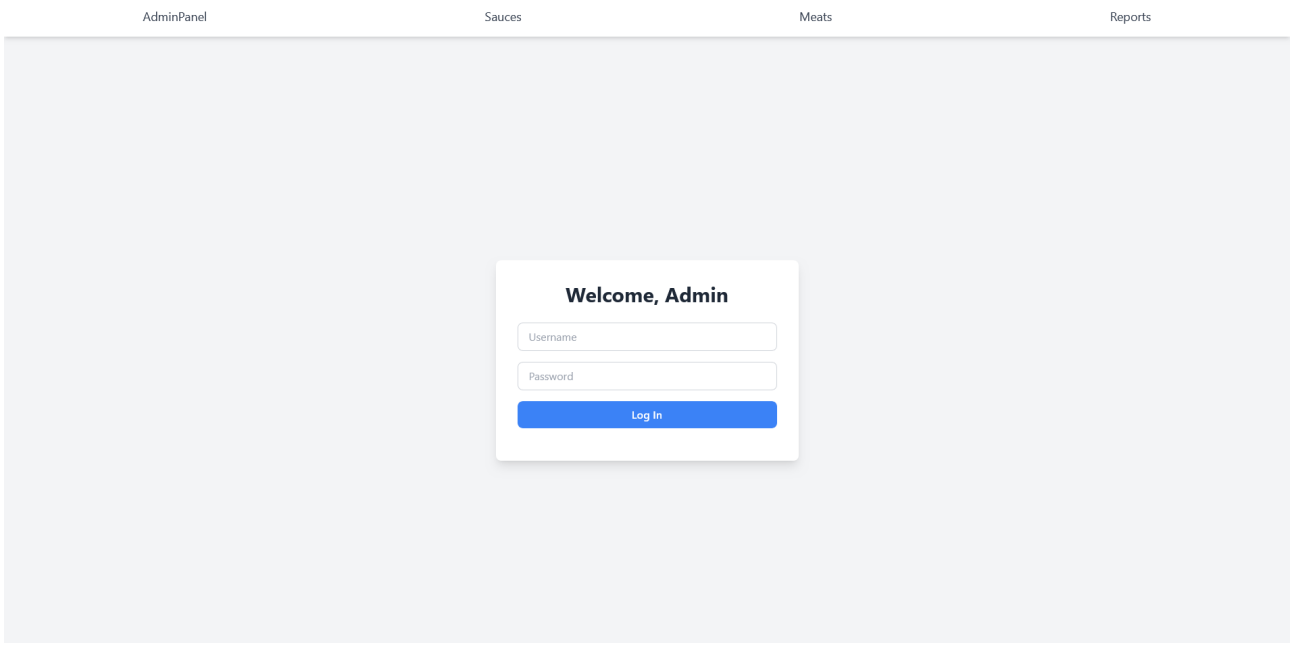


Rysunek 4: Widok filtrów do kebabów

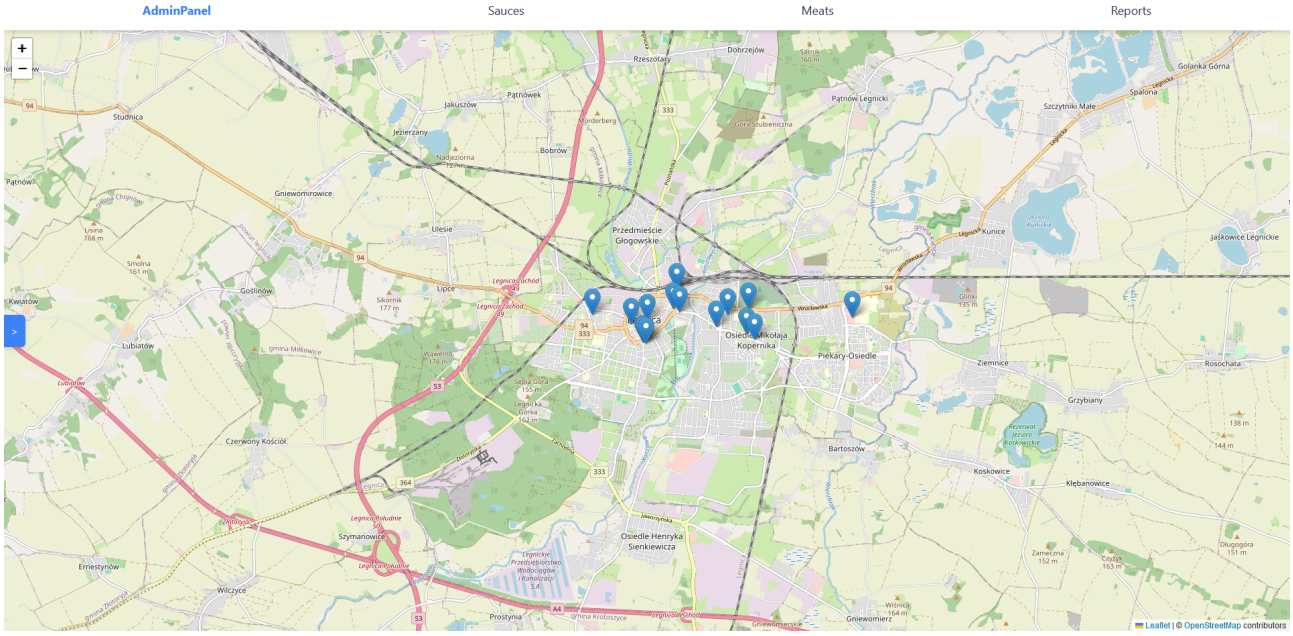


Rysunek 5: Widok profilu użytkownika

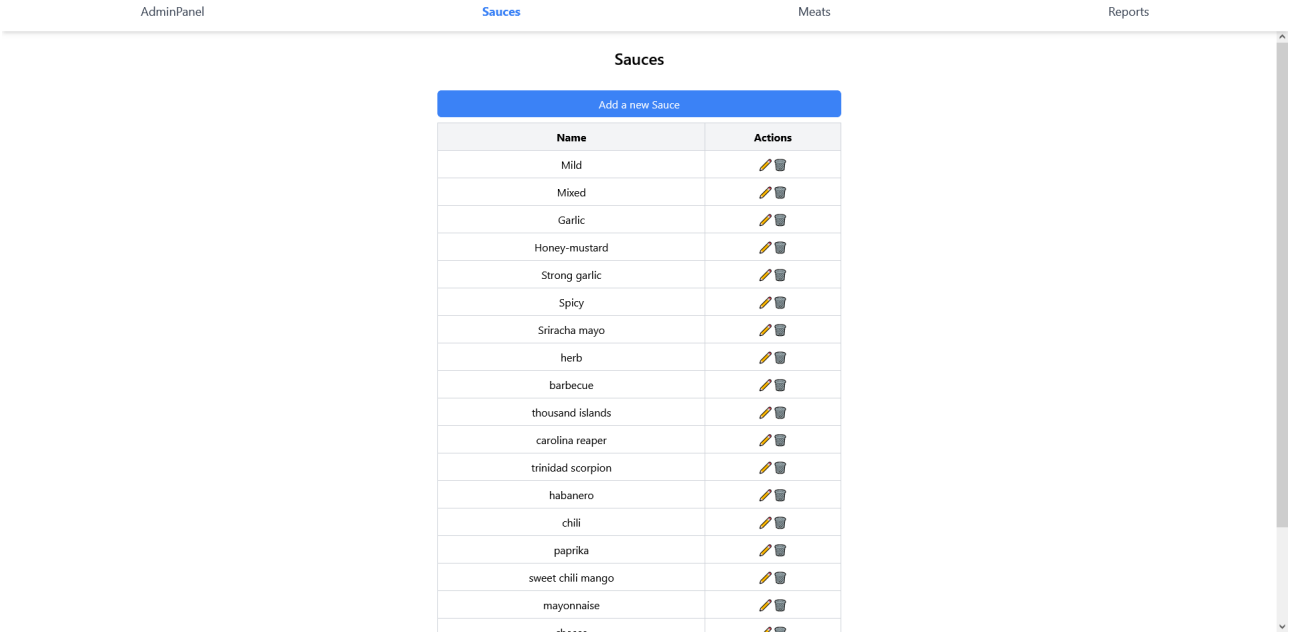
6.2 Interfejs panelu administratora



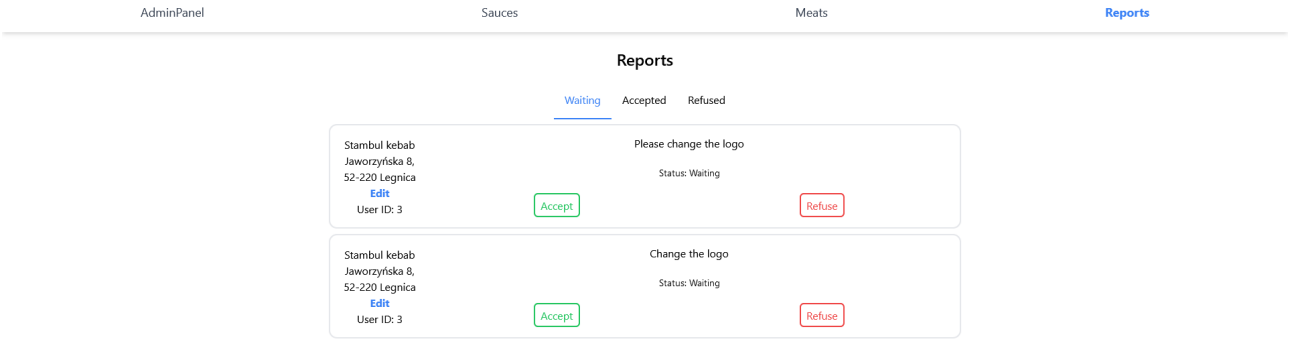
Rysunek 6: Panel logowania



Rysunek 7: Panel główny



Rysunek 8: Panel sosów



Rysunek 9: Panel zgłoszeń

7 Instrukcje uruchomieniowe

7.1 Instrukcja lokalnego uruchomienia API

1. Skopiuj repozytorium na swoje lokalne środowisko.
2. Skonfiguruj plik `.env`.
3. Zainstaluj niezbędne zależności za pomocą komendy `composer install`.

4. Uruchom migracje bazy danych za pomocą komendy `php artisan migrate`.
5. Uruchom seeder bazy danych za pomocą komendy `php artisan db:seed`.
6. Uruchom serwer lokalny za pomocą komendy `php artisan serve`.

7.2 Instrukcja lokalnego uruchomienia panelu administratora

1. Jeżeli nie posiadasz środowiska Node.js z npm, to zainstaluj je.
2. Skopiuj repozytorium na swoje lokalne środowisko.
3. W głównym folderze stwórz plik .env ze zmienną REACT_APP_API_URL="twojeapi" w miejscu twojeapi wpisz adres IP swojego api
4. Zainstaluj zależności komendą `npm install`
5. Uruchom aplikację komendą `npm start`

7.3 Instrukcja lokalnego uruchomienia aplikacji mobilnej

1. Skopiuj repozytorium na swoje lokalne środowisko
2. W pliku config.tsx umieść adres serwera
3. Zainstaluj zależności za pomocą komendy npm install
4. Uruchom aplikację za pomocą komendy npx expo start
5. Na telefonie uruchom aplikację Expo Go
6. Zeskanuj telefonem ukazany w terminalu kod QR

7.4 Instrukcja zdalnego uruchomienia API

1. Zaloguj się na serwer poprzez SSH
2. Jeżeli serwer nie posiada serwera apache, pobierz go za pomocą komendy sudo apt-get install apache2 -y
3. Zainstaluj język php wraz z obsługą sqllite
4. W php.ini zmień linię cgi.fix_pathinfo=0 na cgi.fix_pathinfo=1
5. Zainstaluj Composer za pomocą komendy curl -sS https://getcomposer.org/installer — php
6. Pobierz za pomocą git repozytorium API
7. Nadaj uprawnienia do pliku API za pomocą komend: sudo chown -R www-data:www-data /var/www/html/laravel, sudo chmod -R 775 /var/www/html/laravel, sudo chmod -R 775 /var/www/html/laravel/storage oraz sudo chmod -R 775 /var/www/html/laravel/bootstrap/cache

8. Skonfiguruj API tak jak w instrukcji podanej w lokalnym uruchomieniu API

9. Wykonaj restart apache za pomocą sudo systemctl restart apache2

10. Utwórz i skonfiguruj plik apache w ścieżce /apache2/sites-available. Plik ten może wyglądać w ten sposób:

```
<VirtualHost *:80>
ServerAdmin admin@example.com
ServerName mydomain.com
DocumentRoot /var/www/html/laravel/public

<Directory /var/www/html/laravel>
Options Indexes MultiViews FollowSymLinks
AllowOverride All
Require all granted
</Directory>

ErrorLog ${APACHE_LOG_DIR}/error.log
CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>
```

    Głównych zmian wymagają: ServerName gdzie musi zostać podany adres API, DocumentRoot czyli ścieżka do pliku index.php laravel,

11. Przeładuj serwer apache za pomocą komend: sudo a2enmod rewrite, sudo a2ensite plik-konfiugracyjny.conf oraz sudo systemctl restart apache2

7.5 Instrukcja zdalnego uruchomienia panelu administratora

1. Skopiuj repozytorium na lokalną maszynę

2. Otwórz termninal z uprawnieniami administratorskimi

3. Będąc w głównym folderze aplikacji frontendowej uruchom komendę `npm i` aby zainstalować zależności, sprawdź czy zainstalowana jest zależność gh-pages, jeżeli nie, to zainstaluj ją komendą `npm i gh-pages`

4. W pliku package.json w polu `homepage` wstaw link do swojego repozytorium github np. `"homepage": "https://nazwauzytkownika.github.io/repozytorium"`

5. Upewnij się że w pliku `package.json` w polu `scripts` istnieją te pola: `"predeploy": "npm run build", "deploy": "gh-pages -d build",`

6. Uruchom komendę `npm run deploy`

7. Na stronie Github swojego repozytorium w ustawieniach w zakładce Pages zmień gałąź na `gh-pages`

7.6 Instrukcja uruchomienia aplikacji mobilnej na smartfonie

1. Pobierz plik .apk w przypadku systemu Android bądź .ipa w przypadku systemu IOS
2. Uruchom pobrany plik i przejdź przez instalację programu
3. Uruchom aplikację za pomocą odpowiedniej ikony na telefonie

8 Wnioski projektowe

Praca nad projektem zajęła mniej czasu niż poprzednie projekty z którymi musieliśmy się zmagać, dzięki czemu mieliśmy więcej czasu na wspólne decydowanie o kierunku w jakim ma zmierzać projekt. Przygotowanie projektu na samym początku poprzez ustalenie zadań oraz funkcjonalności aplikacji dało większy obraz jak ma wyglądać aplikacja, jednak fakt ustalenia zadań na samym początku prac ograniczał nasze możliwości reagowania na nowe pomysły bądź nieprzewidziane trudności.

Spotkania odbywające się w każdym tygodniu pozwoliły na dobrą komunikację w zespole, była to dobra sposobność na ustalenie kolejnych zadań do wykonania, podzielenia się doświadczeniami oraz dawały okazję na wspólne rozwiązywanie problemów zarówno programistycznych jak i organizacyjnych.

Prace nad projektem przebiegły bez większych problemów. Nauka nowych technologii rozwinęła nasze postrzeganie różnych platform deweloperskich oraz rozszerzyła nasze horyzonty.

9 Repozytoria

- API oraz panel administratora - https://github.com/MatWisie/KebabFinder-API
- Aplikacja mobilna - https://github.com/MatWisie/KebabFinder-Mobile