

06.06.2019

**PzL-Z2**

Gierszewska Natalia

Gilewicz Stanisław

Hradowicz Marcin

Klejda Adam

Projekt zespołowy - sprawozdanie końcowe

**Hans - Aplikacja mobilna wiążąca dostawców  
z klientami**

## 1. Wstęp

Aplikacja ma za zadanie wiązać **Klientów**, będących osobami wyrażającymi potrzebę wykonania usługi transportowej pewnych dóbr, a **Dostawcami**, będącymi podmiotami odpowiedzialnymi za dostarczanie dóbr.

Praca z produktem polega w pierwszej kolejności na złożeniu przez **Klienta** zlecenia przewozu towaru (o podanych parametrach tj. waga oraz wymiary) z miejsca początkowego do miejsca docelowego. Przy procesie składania zleceniodawca jest z góry informowany o koszcie jaki będzie musiał ponieść w związku z wykonaniem danego zlecenia. Na cenę usługi składają się parametry towaru oraz długość trasy na jaką dany towar należy przetransportować.

Po zakończeniu procesu składania zlecenia jest ono publicznie udostępniane wszystkim użytkownikom aplikacji, którzy używają jej z profilu **Dostawcy**. Mogą oni przeglądać parametry zlecenia bez informacji kto jest zleceniodawcą. Po wybraniu odpowiedniego zlecenia można je przyjąć w celu wykonania.

Na rynku można już znaleźć podobne rozwiązania. Większość z nich dotyczy się jednak usług przewozu osób. Istnieją jednak produkty podobne lub wręcz identyczne do naszego. Kilka przykładów zostało przedstawione poniżej:

- **Uber** - aplikacja służąca do przewozu osób, podobna do usług taksówek. Klient wskazuje miejsce, z którego chce, aby samochód go odebrał oraz miejsce docelowe. Naszym celem było zrobienie podobnej aplikacji, która służy nie do transportu osób, ale towarów.
- **BlaBlaCar** - aplikacja służąca do przewozu osób. Opierająca się na założeniu, że osoba jadąca samochodem w dłuższą trasę szuka dodatkowych osób, które mogłyby zabrać ze sobą. Osoby te płacą za przewóz ustaloną wcześniej kwotę.
- **Roadie, Weedmaps Driver, Bringg Driver App** - aplikacje bardzo podobne do naszej, lecz żadna z nich nie występuje w Polsce.

Wprowadzenie takiego systemu mogłoby przynieść kilka korzyści. Głównymi z nich mogłyby być:

**Ułatwienie kontaktu między Klientami, a Dostawcami** - aktualnie w Polsce nie ma żadnej wiodącej aplikacji tego typu, która mogłaby ułatwić nieregularny przewóz towarów osobom prywatnym i jednocześnie pozwolić osobom w roli dostawców na korzyści finansowe. Przykładowo: często zdarzają się sytuacje, w których przemieszczamy się własnym środkiem transportu z jednego miasta do drugiego. W takiej sytuacji użytkownik będący dostawcą może sprawdzić w aplikacji, czy nie ma dostępnych zleceń właśnie pomiędzy tymi miastami. Jeśli takie zlecenie występuje, to przy niewielkim dodatkowym nakładzie pracy może zarobić na takim zleceniu - chociażby na paliwo, jakie wykorzysta na przejazd między tymi miastami. Jest to korzyść jednocześnie dla zleceniodawcy, ponieważ jego towar zostanie dostarczony - jak i dla dostawcy, ponieważ zyska na tym finansowo. Można także uznać, iż jest to korzystne dla środowiska, gdyż nie wystąpi potrzeba przewozu towaru dodatkowym środkiem transportu.

**Przechowywanie zleceń w bazie danych** - korzystając z takiej aplikacji, nie trzeba martwić się o dokumenty związane z przewozem towarów. Wszystko jest przechowywane w

bazie danych i nieprzerwanie dostępne w razie potrzeby. Wystarczy posiadać telefon z zainstalowaną aplikacją.

**Obliczanie ceny na podstawie danych podanych przez Klienta** - cena usługi transportowej jest obliczana na podstawie wagi i wymiarów towaru oraz odległości na jaką dany towar trzeba przetransportować. Dobierając odpowiedni algorytm, możemy zapewnić stałą cenę dla każdego użytkownika aplikacji. W takim wypadku, obliczanie kwoty zlecenia nie należy do obowiązków zleceniodawcy, co zwalnia go z oszacowania potencjalnej ceny. Płatność z góry poprzez przelew internetowy lub płatność mobilną jest także dużym ułatwieniem.

**Możliwość nawigowania Dostawcy do miejsc odbioru i dostawy przez Mapy Google** - w dzisiejszych czasach nawigacja samochodowa pozwala na poruszanie się po miejscach zupełnie nam nieznanych. Dlatego, osoba będąca dostawcą z pomocą naszej aplikacji nie musi znać miejsca odbioru czy dostarczenia towaru. Aplikacja pokaże odpowiednią trasę z aktualnej lokalizacji do miejsca docelowego.

## **2. Cel i zakres pracy**

**System ma pozwolić na wykonywanie następujących czynności:**

**Składanie zleceń przez Klientów.** Wiąże się to z uzupełnieniem podstawowych danych o towarze będącym celem przewozu takich jak: waga, wymiary oraz punkt z którego należy odebrać i dostarczyć przedmiot. Następnie Klient musi zapoznać się z obliczoną ceną jaką przyjdzie mu zapłacić za daną usługę i jeśli wyrazi na taką zgodę może złożyć zlecenie. Po tym należy wybrać formę płatności jako płatność z góry lub przy odbiorze. Wybierając płatność z góry użytkownik zostaje przekierowany do systemu płatności. Po zakończonym procesie zlecenie staje się publicznie dostępne dla użytkowników, zwanych Dostawcami.

**Klient powinien mieć możliwość śledzenia statusu zlecenia.** Polega to na wysyłaniu wiadomości e-mail do zleceniodawcy za każdym razem, gdy status zlecenia ulegnie zmianie. Ma to miejsce w przypadku przyjęcia zlecenia przez Dostawcę oraz dostarczenia towaru do miejsca docelowego. Klient może także w każdej chwili sprawdzić status w szczegółach danego zlecenia. Nie jest to zadanie o najwyższym priorytecie, jednak jest to duża wygoda dla osób składających zlecenie. Posiadają oni bieżące informacje o tym co dzieje się z ich towarem.

**Anulowanie złożonego zlecenia.** Do czasu, gdy zlecenie nie zostanie przyjęte przez Dostawcę, Klient w każdej chwili może zrezygnować ze złożonego zlecenia poprzez okno ze szczegółami danego zlecenia.

**Użytkownik musi mieć możliwość przeglądania wszystkich swoich zleceń.** Kontrola jest bardzo ważnym czynnikiem w naszym systemie. Każdy użytkownik, bez względu czy jest on

zalogowany jako Dostawca czy Klient ma możliwość przeglądania wszystkich swoich zleceń. To w jakim stanie znajduje się zlecenie nie ma żadnego znaczenia. Oznacza to także możliwość przeglądania historii zakończonych zleceń.

**Przyjmowanie zleceń przez Dostawcę.** Przed przyjęciem zlecenia Dostawca może przejrzeć wszystkie dostępne, posortować po cenie, odległości od jego lokalizacji oraz dacie złożenia lub dokonać filtracji ze względu z jakiego miasta trzeba odebrać towar lub do jakiego dostarczyć. Można także, wejść w szczegóły, aby zapoznać się z parametrami towaru lub trasą jaką trzeba pokonać.

**Potwierdzanie przekazania towaru do miejsca docelowego.** Po dotarciu do miejsca docelowego Dostawca powinien zakończyć proces realizacji zlecenia. Wiąże się to z wpisaniem imienia oraz nazwiska odbiorcy, następnie system generuje protokół przekazania towaru oraz prosi o złożenie podpisu przez osobę odbierającą towar. Jeżeli zlecenie nie zostało wcześniej opłacone system przed całym procesem przypomni Dostawcy o pobraniu opłaty. Po zakończonym procesie zlecenie przejdzie w stan zakończony, a jego szczegóły wraz z możliwością pobrania protokołu w formacie pdf będą dostępne w historii zleceń.

### Pomijane aspekty

W systemie pomijamy proces przekierowywania do systemu płatności ze względu na trudność realizacji, kwestie prawne oraz brak potrzeby realizacji płatności w warunkach akademickich.

Początkowo zakładaliśmy, iż odległość między punktem początkowym, a docelowym będzie obliczana za pomocą najefektywniejszej trasy wybranej przez Google Maps API. Jednak okazało się, że jest to płatna usługa. W zamian postanowiliśmy wykorzystać darmową możliwość obliczania długości między dwoma punktami w linii prostej.

Tabela 1. Podział prac

Osoba - członek zespołu	Wykonane zadania
Natalia Gierszewska	Wszelkie wyświetlanie list zleceń.
	Wysyłanie maili informujących o zmianie statusu.
	Przyjmowanie zlecenia przez dostawcę.
	Możliwość sortowania list po: <ul style="list-style-type: none"><li>• odległości od bieżącej lokalizacji użytkownika;</li><li>• cenie;</li><li>• dacie złożenia zlecenia.</li></ul>
	Ustalenie lokalizacji używanego urządzenia.
	Okienko symbolizujące system płatności.

Osoba - członek zespołu	Wykonane zadania
<b>Natalia Gierszewska</b>	Sprawozdanie końcowe: <ol style="list-style-type: none"> <li>1. Wstęp               <ol style="list-style-type: none"> <li>a. Ogólna charakterystyka zadania.</li> <li>b. Przegląd podobnych rozwiązań już dostępnych na rynku.</li> </ol> </li> <li>3. Metody konstruowania systemu               <ol style="list-style-type: none"> <li>a. tablica Kanban</li> </ol> </li> <li>4. Budowa systemu               <ol style="list-style-type: none"> <li>a. wykorzystywane klasy</li> <li>b. testy</li> </ol> </li> <li>6. Podsumowanie</li> </ol>
<b>Stanisław Gilewicz</b>	Obliczanie ceny.
	Wyświetlanie punktów odbioru i dostarczenia na mapie. [2]
	Śledzenie statusu zlecenia.
	Możliwość nawigacji do punktów odbioru i dostarczenia.
	Sprawozdanie końcowe: <ol style="list-style-type: none"> <li>4. Opis użytkowanych klas</li> </ol>
<b>Marcin Hradowicz</b>	Sprawozdanie końcowe: <ol style="list-style-type: none"> <li>1. Wstęp               <ol style="list-style-type: none"> <li>a. Tematyka pracy, ogólna charakterystyka zadania.</li> <li>b. Uzasadnienie podjęcia tematu.</li> </ol> </li> <li>2. Cel i zakres pracy               <ol style="list-style-type: none"> <li>a. Przeznaczenie i zadania (podstawowe funkcje) projektowanego systemu.</li> <li>b. Określenie uwzględnianych i pomijanych aspektów zagadnienia.</li> <li>c. Udział poszczególnych członków zespołu w realizacji zadania (w tym sprawozdania).</li> </ol> </li> <li>5. Użytkowanie               <ol style="list-style-type: none"> <li>a. Instrukcja użytkowania - przykładowa sesja, „zrzuty” ekranowe, komentarze.</li> </ol> </li> <li>7. Literatura</li> </ol>
	Kontrola pracy nad całym sprawozdaniem.
	Zaprojektowanie głównego szkieletu aplikacji.
	Nadzór nad pracą.
	Składanie zleceń w tym kontrola nad poprawnością podawanych przez Klienta danych.

Osoba - członek zespołu	Wykonane zadania
Marcin Hradowicz	Możliwość filtrowania zleceń po mieście początkowym lub docelowym.
	Generowanie protokołu przekazania towaru wraz ze składaniem podpisu przez odbiorcę.
	Logowanie do systemu oraz wylogowywanie
	Sprawdzanie dostępności połączenia z internetem oraz uprawnień do lokalizacji urządzenia i zapisywania plików na urządzeniu.
	Animacje ładowania oraz informacje o braku rekordów do wyświetlenia.
	Naprawy błędów.
Adam Klejda	3. Metody konstruowania systemu <ul style="list-style-type: none"> <li>a. Metody pracy zespołowej (np. metody zwinne, Scrum, programowanie ekstremalne).</li> <li>b. Metody modelowania (np. UML).</li> <li>c. Środki implementacji (narzędzia, środowiska, platformy programowania).</li> </ul> 4. Budowa systemu <ul style="list-style-type: none"> <li>a. Ogólna architektura systemu.</li> <li>b. Opis poszczególnych modułów (struktura, funkcje).</li> <li>c. Opis podstawowych struktur danych i struktur sterowania (podprogramów).</li> </ul>
	Anulowanie zleceń.
	Przyjmowanie zleceń.
	Potwierdzanie odbioru towaru.
	Możliwość przeglądania informacji o koncie użytkownika wraz z możliwością edycji.
	Zakładka MyAccount oraz możliwość edytowania danych użytkownika
	Po przyjęciu zlecenia: <ul style="list-style-type: none"> <li>• Klient może sprawdzić dane Dostawcy.</li> <li>• Dostawca może sprawdzić dane Klienta.</li> </ul>
	Obsługa zapytań do bazy Firebase Firestore.
	Positive concurrency podczas komunikacji z bazą danych.

Osoba - członek zespołu	Wykonane zadania
Adam Klejda	Dostęp do zakończonych zleceń.
	Naprawy błędów.

### 3. Metody konstruowania systemu

W projekcie wykorzystywaliśmy metodę Scrum. Jest to metodologia polegająca na iteracyjnym tworzeniu aplikacji. Główną zasadą jest, aby w pewnych sprintach o określonej długości oddawać produkt, który jest zdalny do użytku, a co za tym idzie wszystkie zaplanowane czynności na dany sprint muszą zostać ukończone i być w pełni funkcjonalne, tak aby potencjalny klient tej aplikacji mógł z niej korzystać. W projekcie tym zastosowaliśmy sprinty o długości 2 tygodni. W metodzie tej występuje scrum master jednak w naszym projekcie z powodu na niską liczebność zespołu (4 osoby) postanowiliśmy z tego zrezygnować. Na początku projektu zaczęliśmy od listy wymagań użytkownika w postaci historyjek.

1. Jako klient muszę mieć możliwość złożenia zlecenia, żeby móc przetransportować towar z miejsca startu do docelowego.
2. Jako klient muszę mieć możliwość śledzenia statusu zlecenia aby móc przewidzieć kiedy przesyłka dotrze do celu oraz gdzie się aktualnie znajduje .
3. Jako klient otrzymuję maile potwierdzające zmianę statusu zlecenia aby dostawać informacje nawet gdy nie obserwuję aplikacji.
4. Jako klient muszę mieć możliwość zobaczenia obliczonej ceny przed potwierdzeniem zlecenia aby być świadom podejmowanej decyzji.
5. Jako klient muszę mieć możliwość anulowania zlecenia zanim zostało przyjęte przez dostawcę na wypadek błędnie utworzonego zlecenia, lub zmiany decyzji.
6. Jako klient muszę mieć możliwość zapłacenia za dostawę.
7. Jako dostawca muszę mieć możliwość wyświetlania wszystkich zleceń aby móc wybrać te które są dla mnie odpowiednie.
8. Jako dostawca muszę mieć możliwość przyjmowania zleceń.
9. Jako dostawca muszę mieć możliwość potwierdzenia odebrania towaru ze sklepu.
10. Jako dostawca muszę mieć możliwość potwierdzenia przekazania do miejsca docelowego aby uniknąć komplikacji związanych z próbą wyłudzenia.

Na podstawie historyjek został utworzony plan pracy polegający na wdrażaniu elementów w taki sposób aby można do jak największej ilości działających elementów dojść w jak najkrótszym czasie.

Dodatkowo stwierdziliśmy, że przydatne będzie użycie tablicy Kanban, w celu uporządkowania pracy zespołu oraz łatwego zorientowania się w postępie prac, aby szczegółowe przepytanie pozostałej części zespołu nie było konieczne do wywnioskowania, czy konkretne funkcjonalności już zostały zaimplementowane. Wybraliśmy podział tablicy na “do zrobienia”, “w trakcie” oraz “zrobione”. Taki sposób

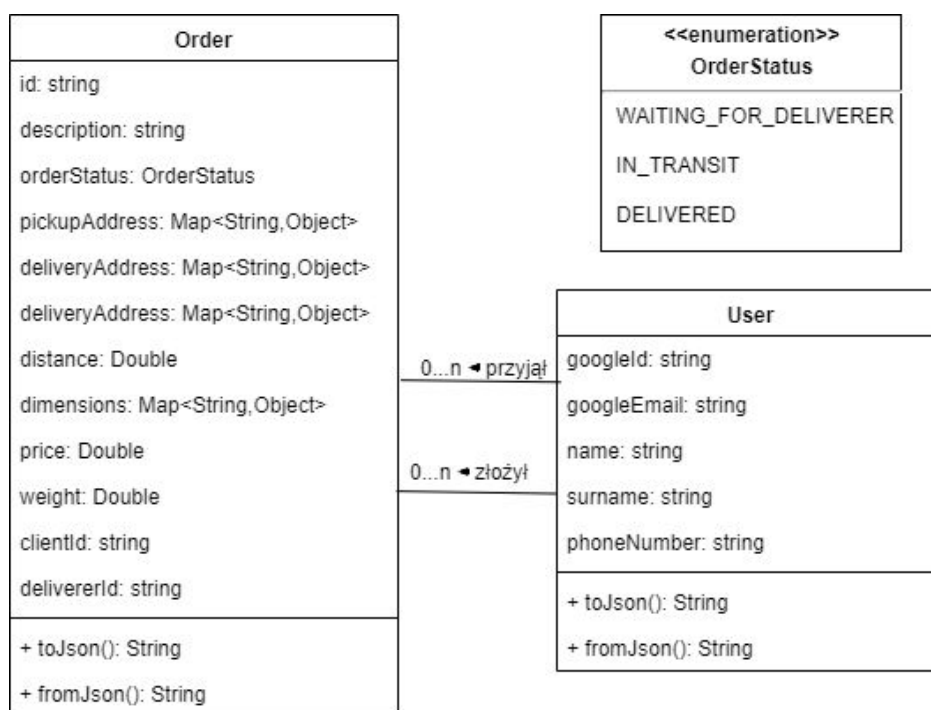
zapisywania zadań posłużył nam również do spisywania błędów aplikacji na bieżąco. Do tego celu wybraliśmy aplikację Trello, ponieważ zadaniu można nadać priorytet oraz przypisać go do osoby z zespołu.

Tabela 2. Plan wykorzystany przy kolejnych sprintach

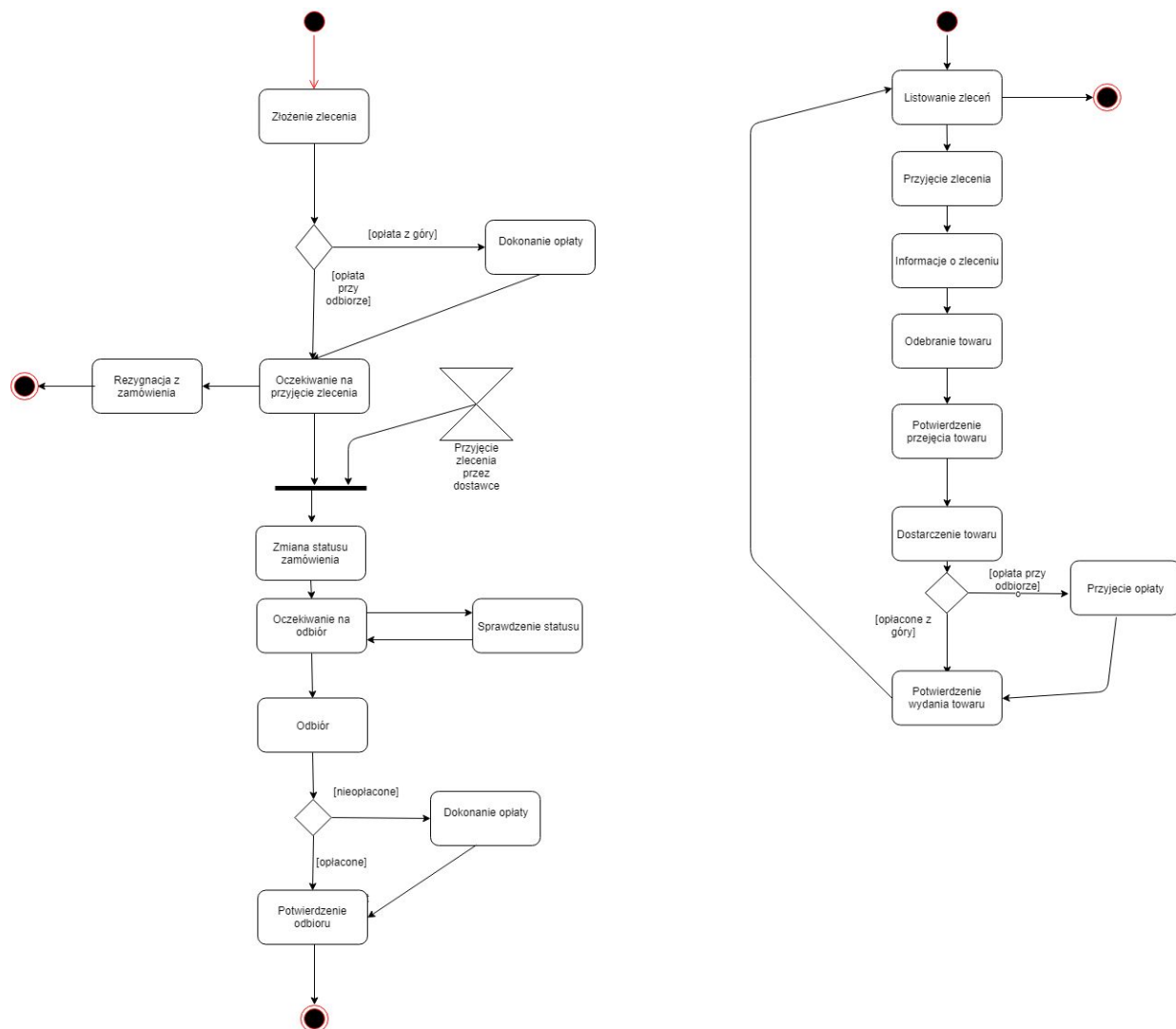
Zajęcia	Funkcjonalność	Osoba wykonująca	Zrobione dodatkowo
I	Jako klient muszę mieć możliwość złożenia zlecenia.	Marcin Hradowicz	<ul style="list-style-type: none"> <li>• logowanie,</li> <li>• wyświetlanie mapy z punktem początkowym i końcowym u dostawcy,</li> </ul>
	Jako klient muszę mieć możliwość zobaczenia obliczonej ceny przed potwierdzeniem zlecenia.	Stanisław Gilewicz	
	Jako dostawca muszę mieć możliwość wyświetlania wszystkich zleceń.	Natalia Gierszewska	
	Jako klient muszę mieć możliwość anulowania zlecenia zanim zostało przyjęte przez dostawcę.	Adam Klejda	
II	Jako klient otrzymuję maile potwierdzające zmianę statusu zlecenia.	Marcin Hradowicz, Natalia Gierszewska	<ul style="list-style-type: none"> <li>• wyświetlanie informacji o kliencie po przyjęciu zlecenia przez dostawcę,</li> <li>• wyświetlenie informacji o dostawcy w szczegółach zlecenia u klienta</li> </ul>
	Jako dostawca muszę mieć możliwość przyjmowania zleceń.	Adam Klejda	
	Jako klient muszę mieć możliwość śledzenia statusu zlecenia.	Stanisław Gilewicz	



Zajęcia	Funkcjonalność	Osoba wykonująca	Zrobione dodatkowo
III	Jako dostawca muszę mieć możliwość wyświetlania trasy za pomocą Google Maps	Stanisław Gilewicz	<ul style="list-style-type: none"> <li>• sprawdzanie połączenia z internetem,</li> <li>• sprawdzanie wymaganych uprawnień,</li> <li>• sortowanie listy aktualnych zleceń u dostawcy,</li> <li>• zakładka Moje konto z możliwością edycji danych</li> </ul>
	Jako dostawca muszę mieć możliwość potwierdzenia odebrania towaru ze sklepu.	Adam Klejda	
	Jako dostawca muszę mieć możliwość potwierdzenia przekazania do miejsca docelowego.	Natalia Gierszewska	
	Jako klient muszę mieć możliwość zapłacenia za dostawę.	Marcin Hradowicz	



Rysunek 1. Diagram klas



Rysunek 2. Diagramy czynności ze strony klienta (po lewej) i Dostawcy (po prawej)

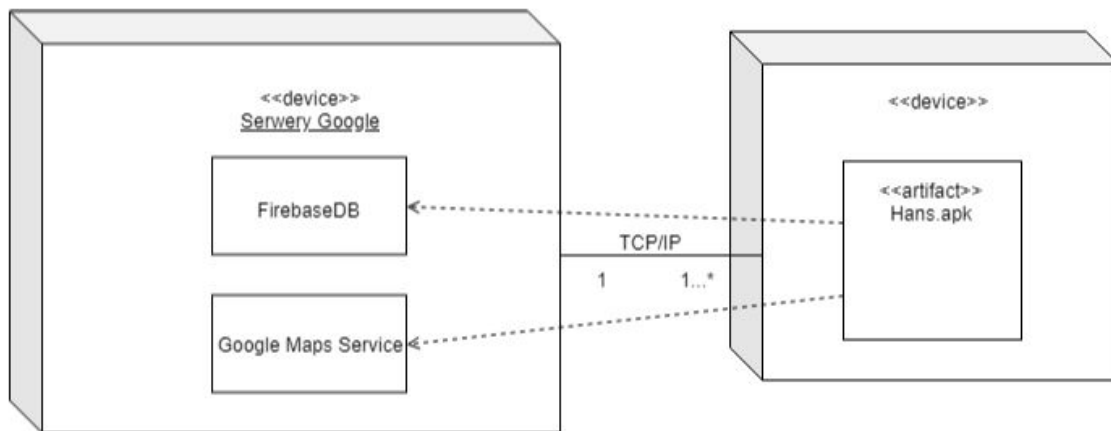
Narzędzia wykorzystane do utworzenia aplikacji:

- Java jako język programowania. Jest to jeden z popularniejszych języków do programowania aplikacji mobilnych na system Android, ma bardzo bogatą i pomocną dokumentację [1].
- Android Studio jako środowisko programistyczne, ponieważ zapewnia dobre warunki do pisania kodu, testowania aplikacji oraz kontroli wersji, co pozwala na efektywną pracę. Jest to wiodące na rynku środowisko do programowania aplikacji na system Android.
- Firebase jako usługę bazy danych do przechowywania informacji o zleceniach i klientach oraz jako przestrzeń do przechowywania potwierdzeń odbioru w formie pdf.
- Google Maps API do obliczania odległości pomiędzy punktami oraz do wyświetlania trasy.
- GitHub jako system kontroli wersji w celu zabezpieczenia postępów podczas pracy oraz dla wygodnej wymiany kodu między współtwórcami.

- Trello jako aplikacja w wersji mobilnej oraz przeglądarkowej do zapisywania aktualnie wykonywanych prac.

#### 4. Budowa systemu

System to aplikacja na urządzenia mobilne z systemem Android w wersji minimum 5.0.



Rysunek 3. Diagram wdrożenia

System można podzielić na dwa główne elementy, aplikację oraz moduły zdalne z których aplikacja korzysta.

- Aplikacja zajmuje się komunikacją klient  $\leftarrow \rightarrow$  system. Pozwala klientowi mieć dostęp do funkcjonalności systemu:
  - fragmenty (klasy rozszerzające klasę Fragment) oraz aktywności (rozszerzające klasę AppCompatActivity) odpowiadające odpowiednim okienkom dostępnym dla użytkowników;
  - klasa obsługująca połączenie z bazą danych firebase;
  - klasa obsługująca połączenie z google maps;
  - klasa reprezentująca zamówienie;
  - klasa reprezentująca użytkownika;
  - klasa zajmująca się wysyłaniem maili;
  - wiele plików w formacie XML będących odwzorowaniem wyglądu interfejsu.
- **Elementy zdalne**, czyli serwery google odpowiadają za przechowywanie danych, logowanie, oraz wyszukiwanie trasy:
  - serwer Firebase z bazą danych;
  - serwer Firebase z dyskiem sieciowym na przetrzymywanie protokołów przekazania towaru w formacie pdf;
  - google maps.

## Wykorzystywane klasy

**Klasa Order** jako reprezentacja zamówienia, przechowuje odpowiednie pola i pilnuje standardu danych, przez co możliwe jest wielokrotne zapisywanie i wczytywanie danych w tej samej strukturze.

Posiada następujące metody:

- settery i gettery,
- toString(),
- toJson() - tworzenie obiektu String w formacie JSON z obiektu Order,
- createFromJson(String orderJSON) - tworzenie obiektu Order z obiektu String w formacie JSON.

Posiada następujące atrybuty:

- id <String> numer identyfikujący zamówienie, nadawany przez bazę danych;
- orderStatus <enum> aktualny status zamówienia:
  - oczekiwanie na przyjęcie przez dostawcę;
  - w drodze;
  - dostarczone;
- pickupAddress <hashMap>{city, number, street, zipCode} - adres odbioru towaru;
- deliveryAddress <hashMap>{city, number, street, zipCode} - adres docelowy;
- length <Double> - odległość z miejsca startowego do docelowego;
- price <Double> - cena w złotych;
- weight <Double> - waga towaru w kilogramach;
- dimensions <hashMap>{depth, height, width} - wymiary towaru w centymetrach;
- description <string> - opis ;
- clientID <string> - identyfikator użytkownika składającego zamówienie;
- delivererID <string> - identyfikator użytkownika przyjmującego zamówienie.

**Klasa User** jako reprezentacja użytkownika, przechowuje odpowiednie pola i pilnuje standardu danych, przez co możliwe jest wielokrotne zapisywanie i wczytywanie danych w tej samej strukturze.

Posiada następujące metody:

- settery i gettery;
- toString()
- toJson() - tworzenie obiektu String w formacie JSON z obiektu User,
- createFromJson(String userJSON) - tworzenie obiektu User z obiektu String w formacie JSON.

Posiada następujące pola:

- googleEmail <string> - adres e-mail;
- googleID <string> - adres konta google;
- name <string> - imię użytkownika;
- surname <string> - nazwisko użytkownika;

- `phoneNumber <string>` - numer telefonu.

**Klasa `DatabaseFirebase`.** Przy pomocy instancji `FirebaseDatabase` skonfigurowanej i połączonej z kontem HANS oraz api obsługującego powyższą bazę odpowiada za komunikację z bazą danych znajdująca się w chmurze Google'a, posiada metody pozwalające na pobieranie, umieszczanie i aktualizowanie danych.

Posiada następujące metody:

- `insertUserToDatabase(User user)` - umieszczenie użytkownika w bazie danych;
- `insertOrderToDatabase(Order order)` - umieszczenie zlecenia w bazie danych;
- `getAllOrdersForDelivererTask()` - zwraca obiekt `Task` odpowiadający za pobranie wszystkich zleceń dostępnych do wykonania;
- `getAllWaitingOrdersForClient(String userID)` - zwraca obiekt `Task` odpowiadający za pobranie wszystkich zleceń oczekujących na przyjęcie dla danego klienta;
- `getAllOrdersTask()` - zwraca obiekt `Task` odpowiadający za pobranie wszystkich zleceń;
- `deleteOrderByID(Order order)` - usunięcie zlecenia o danym identyfikatorze.

Posiada następujące atrybuty:

- `FirebaseFirestoreInstance <FirebaseFirestore>` - instancja bazy danych.

**Klasa `CheckInternetConnectionReceiver`** - odpowiada za przechwytywanie sygnałów włączenia lub wyłączenia połączenia Wi-Fi oraz transmisji danych w telefonie. Pozwala na reagowanie na natychmiastowy brak połączenia z internetem wyświetlając stosowny komunikat.

**Klasa `PdfGenerator`** - odpowiada za generowanie protokołu przekazania towaru i zarządzanie nim. Metody:

- `signInDocument(Path mPath)` - wstawia podpis do dokumentu;
- `createPdf(User clientOb, User delivererOb, String receiver_firstname, String receiver_lastname)` - tworzy dokument pdf;
- `saveLocal(String filename)` - zapisuje dokument lokalnie;
- `sendToFirebaseStorage(String filename)` - wysyła dokument do bazy danych;
- `downloadFileFromFirebaseStorage(String filename)` - ściąga dokument z bazy danych;
- `sendToFirebaseStorage(String filename)` - tworzy kanał powiadomień dla notyfikacji o pobraniu dokumentu na urządzenie.

**Klasa `NDSpinner`** - klasa rozszerzająca wbudowaną klasę `AppCompatActivity`. Daje ona możliwość tworzenia Spinnera, który wykona akcję po wybraniu elementu aktualnie wybranego. Klasa wbudowana nie daje takiej możliwości.

**DataParser** - klasa służąca przekonwertowaniu JSONowego obiektu, na listę pozycji, przez które przebiega trasa na mapie.

**Klasa FetchURL** - klasa służąca przekonwertowaniu danych otrzymanych od google maps api na listę koordynatów, przez które ma przebiegać wytyczona trasa w formacie JSON.

**Klasa MailSender** - klasa tworząca wiadomości email i wysyłające je na podany adres za pomocą protokołu SMTP i hosta smtp.gmail.com. Obsługuje również autentykację użytkownika, aby maile wysyłane były z utworzonego przez nas adresu hans25164@gmail.com.

**Klasa SortFilterOrders** - klasa wspomagająca sortowanie oraz filtrowanie listy zamówień. Udostępnia metody sortujące lub filtrujące podane listy. Ma również metodę, która pobiera obecną lokalizację urządzenia oraz oblicza odległość do podanego adresu, która jest używana przy sortowaniu zleceń po odległości do miejsca początkowego.

**Klasa OrderListAdapter** - klasa wspomagająca tworzenie widoku listy zamówień.

**Klasa ClientAddOrderFragment** - klasa dziedzicząca po Fragment, jej zadanie to obsługa tworzenia nowego zamówienia. Wywołuje widok pozwalający na wypełnienie informacji dotyczących zamówienia a następnie po wciśnięciu potwierdzenia zapisuje te informacje do bazy danych w formie nowego obiektu typu Order.

**Cechy charakterystyczne:**

- Pozwala na obliczenie dystansu.
- Wylicza przewidywaną cenę zamówienia.

**Klasa ClientArchiveOrderInfoFragment** - klasa dziedzicząca po Fragment. Obsługuje wypisywanie informacji na temat zakończonego zlecenia.

**Cechy charakterystyczne:**

- Wyświetla zakończone zlecenia.
- Pozwala na sprawdzenie danych dostawcy wykonującego dane zlecenie.

**Klasa ClientArchiveOrdersFragment** - klasa dziedzicząca po Fragment. Pobiera z bazy danych wszystkie zakończone zlecenia danego klienta i wyświetla ich listę.

**Cechy charakterystyczne:**

- Listuje zakończone zlecenia danego klienta.
- Pozwala wejść w dane szczegółowe danego, zakończonego zlecenia.

**Klasa ClientInTransitOrderInfoFragment** - klasa dziedzicząca po Fragment. Obsługuje widok przedstawiający informacje dla danego zlecenia o statusie "W TRAKCIE".

**Cechy charakterystyczne:**

- Informacje dotyczące dostawcy który przyjął dane zlecenie.
- Informacja czy towar został odebrany przez dostawcę.

**Klasa ClientInTransitOrdersFragment** - klasa dziedzicząca po Fragment, pobiera i listuje zlecenia o statusie "W TRAKCIE" dla danego klienta.

**Klasa ClientWaitingsOrderInfoFragment** - klasa dziedzicząca po Fragment, klasa zawierająca informacje o konkretnym zleceniu klienta.

**Cechy charakterystyczne:**

- Przycisk pozwalający na anulowanie zlecenia, jeżeli nie zostało jeszcze przyjęte przez dostawcę.

**Klasa ClientWaitingsOrdersFragment** - klasa dziedzicząca po Fragment, implementująca widok okna zawierającego listę oczekujących zleceń klienta.

**Cechy charakterystyczne:**

- Listuje zlecenia posiadające status "OCZEKUJĄCE" które należą do danego klienta.

**Klasa DelivererArchiveOrderInfoFragment** - klasa dziedzicząca po Fragment, implementująca widok okna zawierającego informacje o konkretnym zakończonym zleceniu.

**Cechy charakterystyczne:**

- Pozwala na pobieranie protokołu przekazania towaru.
- Wgląd w dane klienta do którego należało zamówienie.

**Klasa DelivererArchiveOrdersFragment** - klasa dziedzicząca po Fragment, implementująca widok okna zawierającego zakończone zlecenia dostawcy.

**Cechy charakterystyczne:**

- Listuje zlecenia posiadające status "ZAKOŃCZONE", które należą do danego klienta.

**Klasa DelivererAvailableOrderInfoFragment** - klasa dziedzicząca po Fragment, implementująca widok okna zawierającego informacje o konkretnym dostępnym do przyjęcia zleceniu.

**Cechy charakterystyczne:**

- Pozwala na przyjęcie zlecenia przez dostawcę.

**Klasa DelivererAvailableOrdersFragment** - klasa dziedzicząca po Fragment, implementująca widok okna zawierającego listę dostępnych zamówień.

**Cechy charakterystyczne:**

- Listuje zlecenia posiadające status "OCZEKUJĄCE", które są dostępne do przyjęcia.

**Klasa DelivererInTransitOrderInfoFragment** - klasa dziedzicząca po Fragment, implementująca widok okna zawierającego informację o zamówieniu aktualnie wykonywanym przez dostawcę.

**Cechy charakterystyczne:**

- Posiada przycisk pozwalający na potwierdzenie odebrania towaru z miejsca początkowego.
- Posiada przycisk pozwalający na zakończenie, zdanie zlecenia.

**Klasa DelivererInTransitOrdersFragment** - klasa dziedzicząca po Fragment, implementująca widok okna zawierającego listę zamówień.

**Cechy charakterystyczne:**

- Listuje zlecenia posiadające status “W TRAKCIE”, które są aktualnie wykonywane przez danego dostawcę.

**Klasa MapsFragment** - klasa dziedzicząca po Fragment, implementująca widok okna zawierającego mapę.

**Klasa GetReceiverNameFragment** - klasa dziedzicząca po Fragment, implementująca widok okna zawierającego pole do wprowadzenia danych odbiorcy.

**Klasa SignDocumentActivity** - klasa dziedzicząca po Activity, implementująca możliwość podpisania dokumentu.

**Klasa CompleteAccountDataActivity** - klasa dziedzicząca po Activity, obsługująca uzupełnianie danych o użytkownika przy zakładaniu konta.

**Klasa SignInGoogleActivity** - klasa dziedzicząca po Activity, implementująca logowanie do konta google.

**Klasa MyAccountCompleteFragment** - klasa dziedzicząca po Fragment, implementująca widok okna zawierającego informacje o koncie użytkownika.

**Klasa MyAccountEditFragment** - klasa dziedzicząca po Fragment, implementująca widok okna zawierającego możliwość edycji konta użytkownika.

**Cechy charakterystyczne:**

- Pozwala na zmianę danych w profilu użytkownika



**Klasa MainActivity** - klasa dziedzicząca po Activity, implementująca główny widok aplikacji, w której wymieniane są klasy dziedziczące po Fragment.

**Klasa SettingsFragment** - klasa dziedzicząca po Fragment, implementująca widok okna zawierającego ustawienia aplikacji.

**Cechy charakterystyczne:**

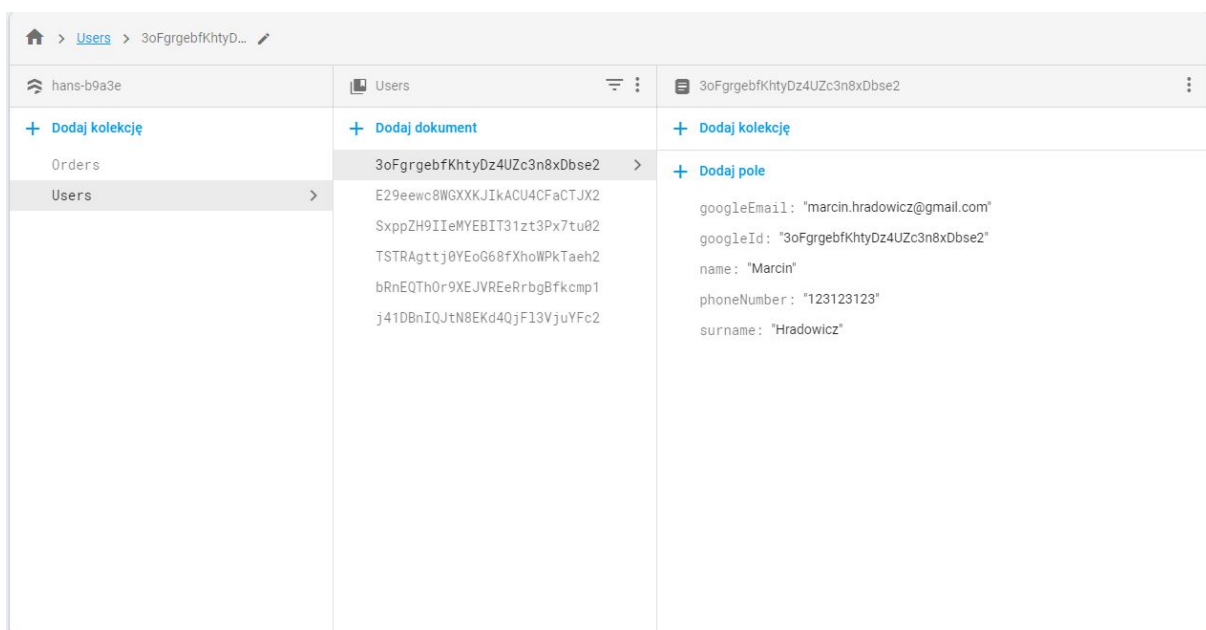
- Posiada przycisk wylogowywania.

**Enum Order status** - pozwala zdefiniować możliwe stany dla zamówienia co zapewnia zgodność wewnątrz systemu:

- WAITING\_FOR\_DELIVERER (OCZKUJĄCE) - oznacza oczekiwanie na dostawcę, aby ten przyjął zlecenie;
- IN\_TRANSIT (W TRAKCIE) - oznacza że dostawca już przyjął dane zlecenie i jest w trakcie jego wykonywania;
- CLOSED (ZAKOŃCZONE) - oznacza zakończone zamówienia, takie które zostały dostarczone do miejsca docelowego.

### Katalog bazy danych Users na platformie Firebase

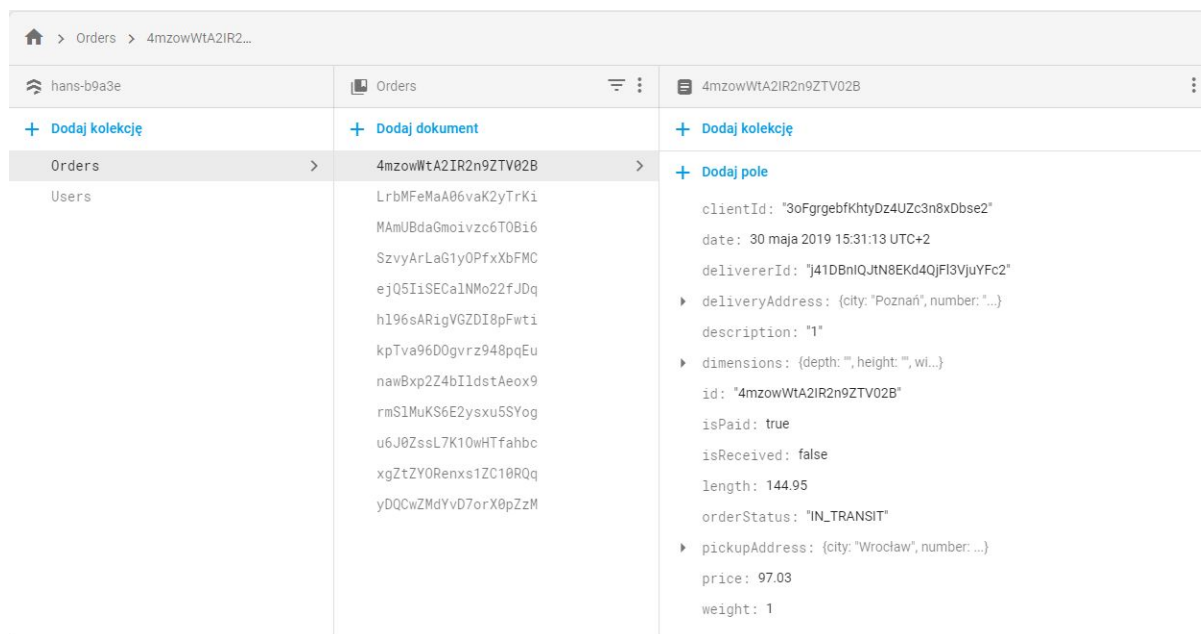
Przechowuje dane w postaci dokumentów skatalogowane jako Users w jednym "pliku". Id dokumentu są generowane na podstawie ID googlowskiego konta, aby łatwo było znaleźć i edytować dane dla konkretnego użytkownika. Dane umieszczane są poprzez googlowskie api do FirebaseFirestore. Pola znajdujące się w dokumencie User w katalogu Users, posiadają taką samą nazwę jak pola klasy User w aplikacji klienckiej co pozwala na automatyczne generowanie obiektu.



Rysunek 4. Widok użytkowników po stronie bazy danych

## Katalog bazy danych Orders na platformie Firebase

Przechowuje dane w postaci dokumentów skatalogowane jako Orders w jednym "pliku". Id dokumentu są generowane automatycznie przez bazę danych. Dane umieszczane są poprzez googlowskie api do FirebaseFirestore. Pola znajdujące się w dokumencie Order w katalogu Orders, posiadają taką samą nazwę jak pola klasy Order w aplikacji klienckiej co pozwala na automatyczne generowanie obiektu. Dodatkowo podczas pobrania orderu przez aplikację zostaje mu ustawiony numer id na odpowiadający dokumentowi co ułatwia odwoływanie się do konkretnego dokumentu.



hans-b9a3e	Orders	4mzowWtA2IR2n9ZTV02B
+ Dodaj kolekcję	+ Dodaj dokument	+ Dodaj kolekcję
Orders >	4mzowWtA2IR2n9ZTV02B >	+ Dodaj pole
Users	LrbMFeMaA06vaK2yTrKi MAmUBdaGmoivzc6TOBi6 SzvyArLaG1y0PfxXbFMC ejQ5IiSECa1NMo22fJDq h196sARigVGZDI0pFwti kpTva96D0gvrz948pqEu nawBxp2Z4bI1dstAeox9 rmS1MuKS6E2ysxu5SYog u6J0ZssL7K10wHTfahbc xgZtZY0Renxs1ZC10RQq yDQCwZMdYvD7orX0pZzM	clientId: "3oFgrgebfkhtyDz4UZc3n8xDbse2" date: 30 maja 2019 15:31:13 UTC+2 delivererId: "j41DBniQJtN8EKd4QJF3VjuYFc2" ▶ deliveryAddress: {city: "Poznań", number: "..."} description: "1" ▶ dimensions: {depth: "", height: "", wi...} id: "4mzowWtA2IR2n9ZTV02B" isPaid: true isReceived: false length: 144.95 orderStatus: "IN_TRANSIT" ▶ pickupAddress: {city: "Wrocław", number: ...} price: 97.03 weight: 1

Rysunek 5. Widok zleceń po stronie bazy danych

## Testy

Testowaliśmy aplikację na własnych telefonach z systemem Android. Każdy z nich różnił się wersją systemu operacyjnego. Przeprowadziliśmy testy możliwych ścieżek użytkowania aplikacji. Braliśmy pod uwagę, że użytkownik może chcieć użyć nieprawidłowych danych (np. o niewłaściwym formacie).

Przypadek, kiedy dostawca i klient mają jednocześnie otwarte okno z zamówieniem, chcąc odpowiednio przyjąć i anulować zlecenie, nie mogą tego zrobić jednocześnie. Działanie, które zostanie wykonane pierwsze, uniemożliwia dokonanie drugiego. Dostawca nie może przyjąć zlecenia, które zostało przed chwilą anulowane, a klient nie może anulować zlecenia, które zostało przed chwilą przyjęte.

Obsłużona jest weryfikacja poprawności formatu wpisywanych danych przy składaniu zlecenia. Konieczne jest wpisanie poprawnego kodu pocztowego, adresu e-mail oraz numeru telefonu przez użytkownika. W przeciwnym wypadku, pojawia się odpowiedni komunikat.

Jeżeli w trakcie użytkowania aplikacji urządzenie straci połączenie z Internetem, pojawia się komunikat o konieczności ponownego połączenia z siecią. Dalsze użytkowanie aplikacji nie jest możliwe, dopóki urządzenie nie połączy się z Internetem ponownie.

W ogólności, aplikacja działa sprawnie i bez opóźnień. Działa trochę wolniej, gdy aplikacja wczytuje większą ilość danych z bazy danych (długie listy zamówień) oraz gdy sortuje długie listy lub sortuje po odległości od bieżącej lokalizacji urządzenia do punktu początkowego zlecenia. Zadania te zajmują trochę większą ilość czasu. Użytkownik jednak nie ma poczucia, że aplikacja się zawiesza, ponieważ ukazuje się ekran ładowania.

## **5. Użytkowanie**

### **Przykładowa sesja użytkowania:**

1. Klient składa zlecenie, co wiąże się z uzupełnieniem adresu odbioru towaru oraz dostarczenia, a także wagi i wymiarów (rysunek 6).
2. Następnie system pyta użytkownika jakiego rodzaju płatność wybiera (rysunek 7).
3. Na rysunku 8. możemy zauważyć, iż zlecenie zostało dodane i widnieje jako oczekujące na przyjęcie.
4. Rysunek 9. przedstawia listę dostępnych zleceń, z której Dostawca wybiera najbardziej mu odpowiadające.
5. Kolejne rysunki 10. oraz 11. przedstawiają szczegóły zlecenie jakie widzi Dostawca przed przyjęciem. Można zauważyć, iż brak tam informacji na temat zleceniodawcy.
6. Na rysunku 12. widzimy okienko z zapytaniem czy na pewno chcemy przyjąć zlecenie. Po przyjęciu na rysunku 13. widać, że zlecenie pojawiło się jako aktualnie wykonywane.
7. Kolejne dwa rysunki 14. oraz 15. pokazują szczegóły zlecenia. Tutaj już na samym dole pojawiły się informacje o zleceniodawcy.
8. Rysunki 16. oraz 17. ukazują kolejne dwa okna informacyjne dotyczące akcji kończenia realizacji zlecenia. Pierwszy z nich jest potwierdzeniem czy na pewno chcemy zakończyć dane zlecenie, Drugi z nich przypomina o odebraniu zapłaty od odbiorcy, ponieważ została wybrana płatność przy odbiorze.
9. Rysunki 18. oraz 19. dotyczą właściwej części procesu zakończenia zlecenia. Najpierw pojawia się okno uzupełnienia informacji o odbiorcy, a następnie ekran, na którym odbiorca ma złożyć podpis, tym samym potwierdzając odbiór towaru w zadowalającym stanie.
10. Następny rysunek 20. przedstawia informacje o pomyślnie zakończonym zleceniu oraz miejscu, w którym można pobrać wygenerowany protokół przekazania towaru.
11. Jak można zauważyć na rysunku 21. zlecenie widnieje jako zakończone. Po otwarciu możemy przejrzeć jego szczegóły, co przedstawiają rysunki 22. oraz 23. Na samym dole znajduje się przycisk pozwalający na pobranie protokołu przekazania towaru. Po pobraniu pojawia się powiadomienie w formacie pokazanym na rysunku 24. Kliknięcie w powiadomienie otwiera pobrany dokument, który został przedstawiony na rysunku 25.

**Nowe zlecenie**

Punkt początkowy: Poznań  
Punkt końcowy: Warszawa

60 - 965      05 - 077

Piotrowo      Plac Defilad

3      1

Opis: Szafa

Dystans (km)	Cena (zł)
277.15	130.44

OBLICZ CENĘ

Waga (kg)	Szerokość (cm)	Wysokość (cm)	Głębokość (cm)
50	1.5	2	0.5

ZŁÓŻ ZLECENIE

Rysunek 6. Widok składania zlecenia (Klient)

**Nowe zlecenie**

Punkt początkowy: Poznań  
Punkt końcowy: Warszawa

60 - 965      05 - 077

Piotrowo      Plac Defilad

3      1

Opis: Szafa

Czy chcesz dokonać płatności z góry?  
Kwota: 130.44 zł

TAK

WYBIERAM PŁATNOŚĆ PRZY ODBIORZE

ZŁÓŻ ZLECENIE

Rysunek 7. Okno z wyborem rodzaju płatności (Klient)

**Oczekujące zlecenia**

Z: Poznań Piotrowo 3  
Do: Warszawa Plac Defilad 1  
Opis: Szafa

Cena: 130.44 zł  
Waga: 50.0 kg

Rysunek 8. Zaktualizowana lista zleceń oczekujących na przyjęcie (Klient)

**Dostępne zlecenia**

Sortowanie po: czasie złożenia zamówienia..

Filtrowanie po: brak

Z: Poznań Piotrowo 3  
Do: Warszawa Plac Defilad 1  
Opis: Szafa

Cena: 130.44 zł  
Waga: 50.0 kg

Rysunek 9. Lista dostępnych zleceń (Dostawca)

**Dostępne zlecenia**

Bydgoszcz  
Polska  
Poznań  
Łódź  
Warszawa

NAWIGUJ DO PUNKTU STARTOWEGO      NAWIGUJ DO MIEJSCA DOCELOWEGO

Status: **Oczekiwanie na przyjęcie przez dostawcę**

Zapłacone z góry: **Nie**

Punkt początkowy	Punkt końcowy
Poznań	Warszawa
60-965	05-077
Piotrowo	Plac Defilad
3	1
Opis: Szafa	

Rysunek 10. Szczegóły dostępnego zlecenia, część 1 (Dostawca)

**Dostępne zlecenia**

Bydgoszcz  
Polska  
Poznań  
Łódź  
Warszawa

Punkt początkowy: Poznań  
Punkt końcowy: Warszawa

60-965      05-077

Piotrowo      Plac Defilad

3      1

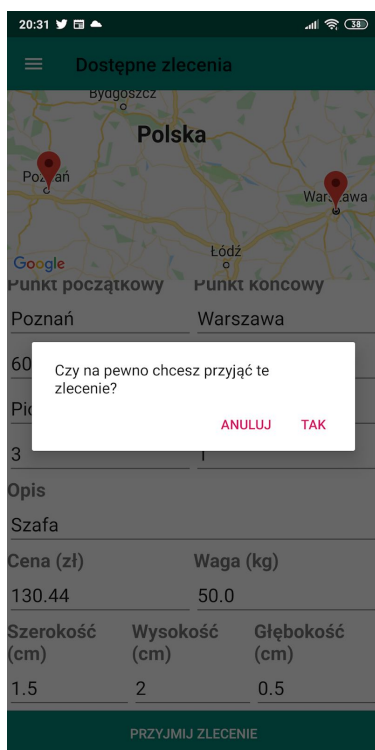
Opis: Szafa

Cena (zł)	Waga (kg)
130.44	50.0

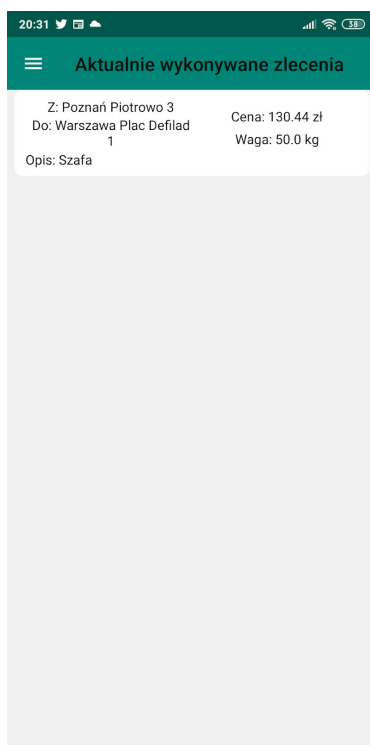
Szerokość (cm)	Wysokość (cm)	Głębokość (cm)
1.5	2	0.5

PRZYJMIJ ZLECENIE

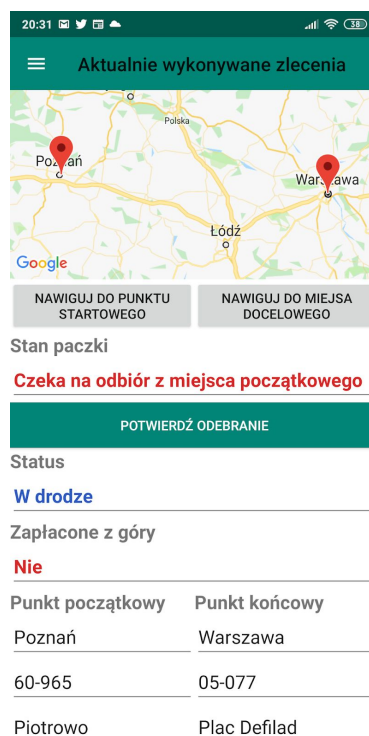
Rysunek 11. Szczegóły dostępnego zlecenia, część 2 (Dostawca)



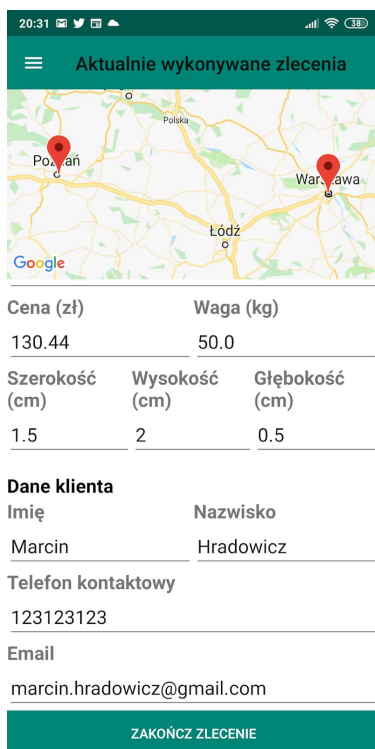
Rysunek 12. Okno z potwierdzeniem przyjęcia zlecenia (Dostawca)



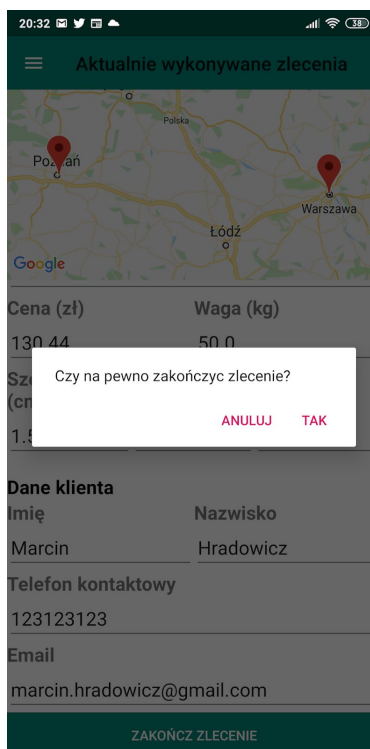
Rysunek 13. Zaktualizowana lista aktualnie wykonywanych zleceń (Dostawca)



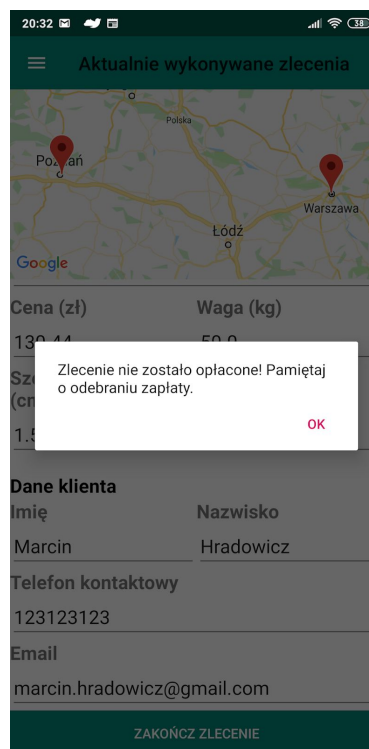
Rysunek 14. Szczegóły aktualnie wykonywanego zlecenia, część 1 (Dostawca)



Rysunek 15. Szczegóły aktualnie wykonywanego zlecenia, część 2 (Dostawca)



Rysunek 16. Okno z potwierdzeniem zakończenia (Dostawca)



Rysunek 17. Okno z przypomnieniem o odebraniu zapłaty za zlecenie (Dostawca)

20:32

☰ Dane odbiorcy

Imię  
Jan

Nazwisko  
Odbiorca

DALEJ

Odbiorca Odbiorcą Odbiorcami

1 2 3 4 5 6 7 8 9 0

q w e r t y u i o p

a s d f g h j k l

z x c v b n m

PL • EN

Rysunek 18. Uzupełnianie danych odbiorcy (Dostawca)

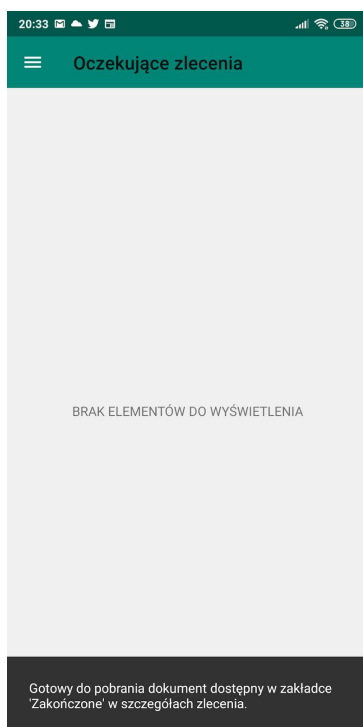
20:33

Podpis odbiorcy

ZATWIERDŹ WYCZYŚĆ

Jan Odbiorca

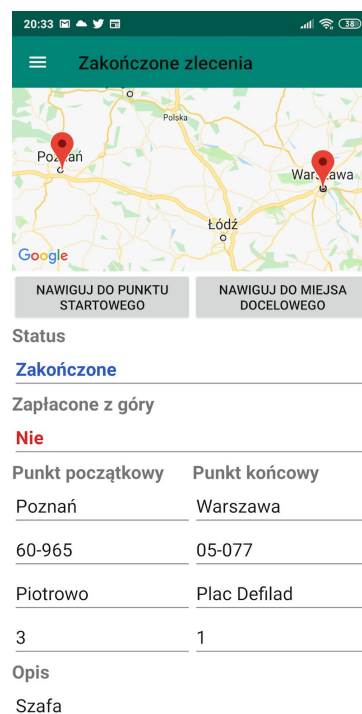
Rysunek 19. Podpis odbiorcy



Rysunek 20. Informacja o zakończonym zleceniu i miejscu z możliwością pobrania protokołu przekazania towaru



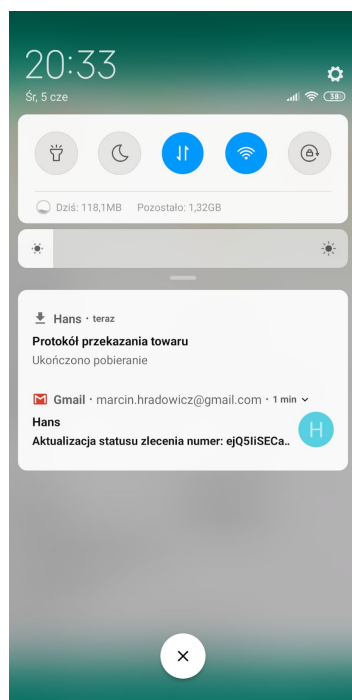
Rysunek 21. Zaktualizowana lista zakończonych zleceń (Dostawca)



Rysunek 22. Szczegóły zakończanego zlecenia, część 1 (Dostawca)



Rysunek 23. Szczegóły zakończanego zlecenia, część 2 (Dostawca)



Rysunek 24. Komunikat o pobraniu protokołu przekazania towaru (Dostawca)




**PROTOKÓŁ PRZEKAZANIA TOWARU**

**Data wystawienia:** 05 cze 2019 20:33  
**Wykonawca usługi transportowej:** Marcin Hradowicz  
**Zleceniodawca usługi transportowej:** Marcin Hradowicz

Tabela 1. Przyjęte towary

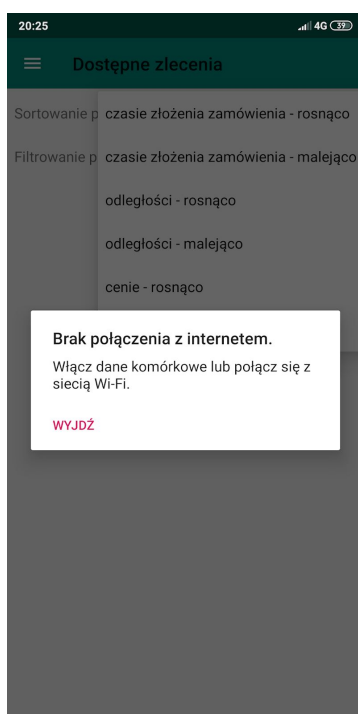
Lp. 1.	Opis: Szafa	Waga [kg]: 50.0	Wymiary [m] 1.5 x 2 x 0.5
* - szerokość[cm] x długość[cm] x głębokość[cm]			

Ja, niżej podpisany Jan Odbiorca oświadczam, że w dniu 05 cze 2019 odebrałem od Marcina Hradowicza wszystkie towary zamieszczone w Tabeli 1. w zadowalającym mnie stanie.

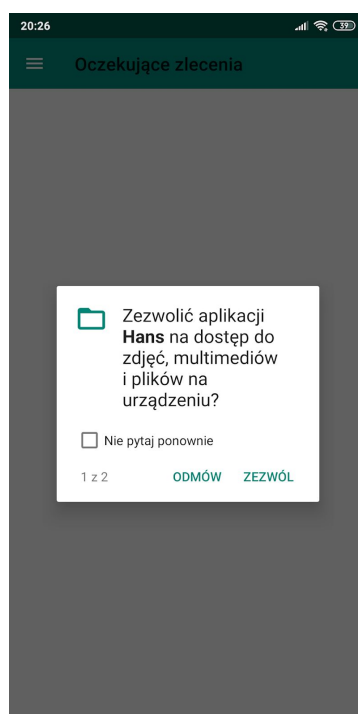
  
 \_\_\_\_\_  
 podpis odbierającego

Rysunek 25. Przykładowy protokół przekazania towaru (Dostawca)

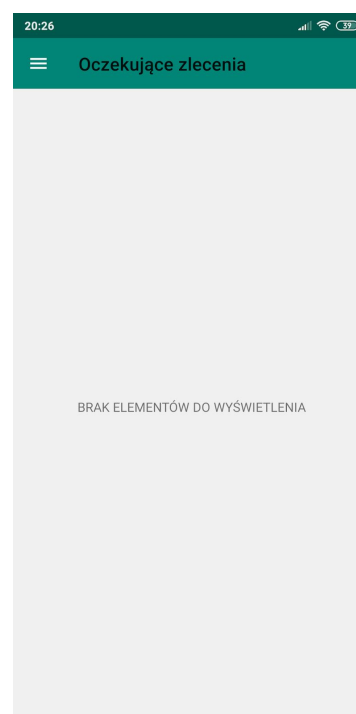
### Reszta dostępnych opcji lub komunikatów:



Rysunek 26. Informacja o braku połączenia z internetem

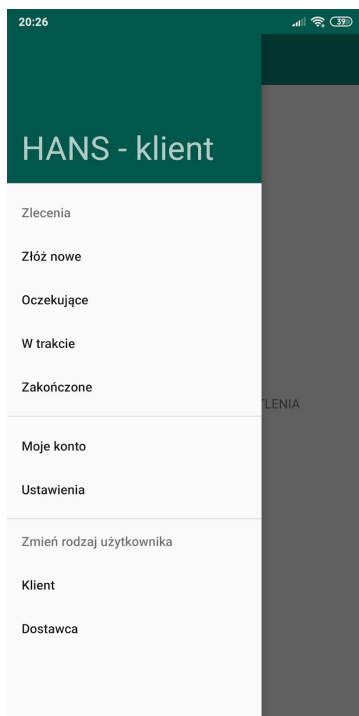


Rysunek 27. Informacja o braku wymaganych uprawnień

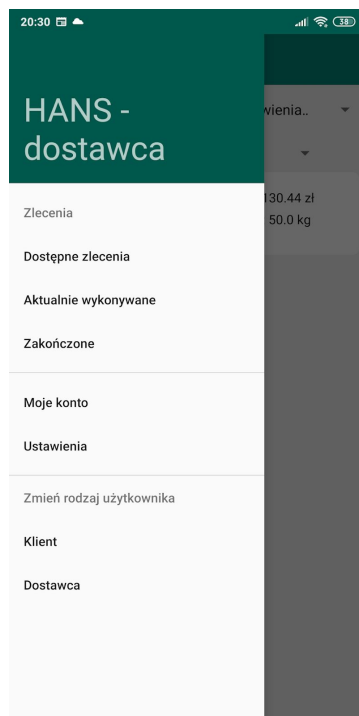


Rysunek 28. Informacja o braku rekordów do wyświetlenia

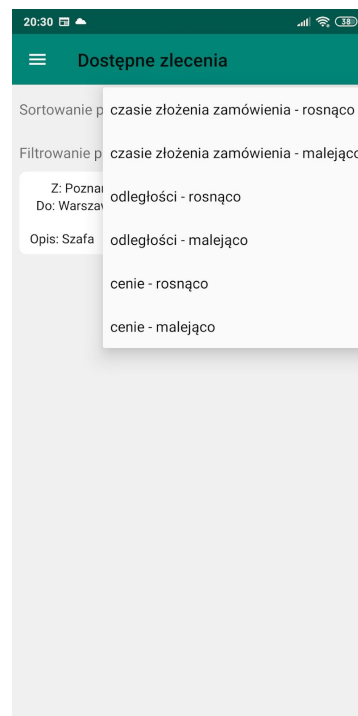




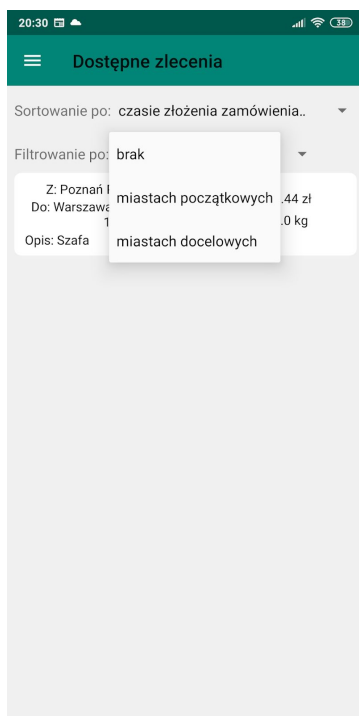
Rysunek 29. Wygląd menu użytkownika w trybie Klienta



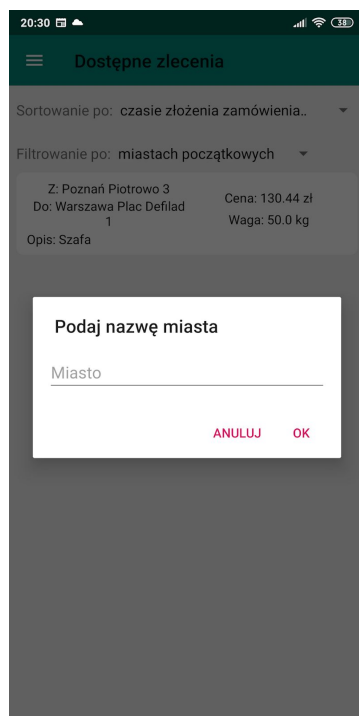
Rysunek 30. Wygląd menu użytkownika w trybie Dostawcy



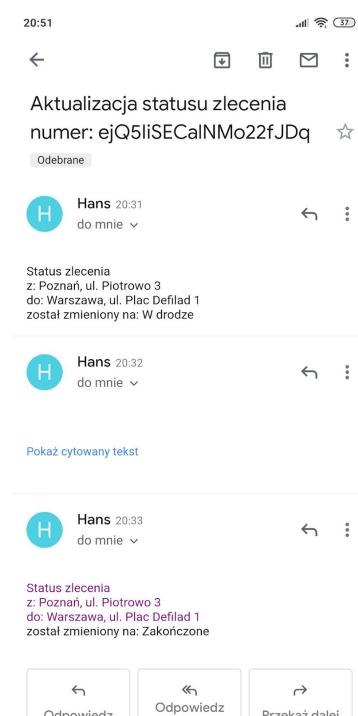
Rysunek 31. Opcje sortowania list



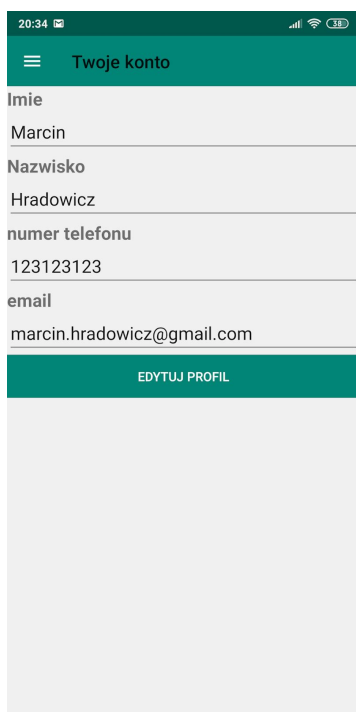
Rysunek 32. Opcje filtrowania list



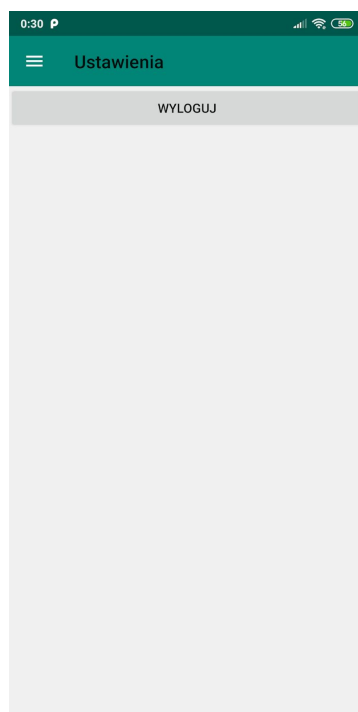
Rysunek 33. Okno z prośbą o wpisanie miasta po jakim użytkownik chce filtrować



Rysunek 34. Powiadomienia e-mail o zmianie statusu zlecenia wysyłane do Klienta



Rysunek 35. Okno z danymi konta



Rysunek 36. Przycisk wylogowywania dostępny w ustawieniach.

## 6. Podsumowanie

Zrealizowaliśmy wszystkie postawione sobie cele. Aplikacja jest funkcjonalna, spełnia nasze założenia. Przeprowadzone testy wskazują, że mogłaby być używana przez szersze grono odbiorców.

Dodatkowo, zaimplementowaliśmy funkcje, które ułatwiają użytkownikowi korzystanie z aplikacji, takie jak wyświetlanie map, filtrowanie i sortowanie list. Ponadto konieczne było wprowadzenie logowania oraz obsługi różnorodnych wyjątków, które mogłyby się wydarzyć podczas użytkowania aplikacji, jak chociażby utrata połączenia z Internetem.

Naszym zdaniem aplikacja mogłaby być wprowadzona na rynek oraz udostępniona większej liczbie użytkowników. Aby to umożliwić, konieczna byłaby implementacja systemu płatności internetowej. Przydatną funkcjonalnością z punktu widzenia odbiorcy, byłoby wysyłanie powiadomień typu push na telefon. Aby aplikacja się wyróżniała i była przyjemna w odbiorze, można popracować nad graficznym interfejsem, trochę go urozmaicić.

Praca zespołowa nad projektem była bardzo efektywna. Małe problemy sprawiło nam stworzenie harmonogramu przed rozpoczęciem prac, ze względu na to, że nie znaliśmy orientacyjnego czasu potrzebnego na zaimplementowanie pewnych funkcjonalności. Nie do końca przemyśleliśmy również kolejność, z jaką powinniśmy wdrażać poszczególne elementy systemu. Wymusiło to na nas konieczność lekkiej zmiany harmonogramu w początkowej fazie prac.

Obecność listy funkcjonalności oraz harmonogramu bardzo pomogła nam w pracy. Łatwo można było się zorientować co należy do obowiązków każdego członka zespołu, co każdy z nas ma do zrobienia jako następne.

Dodatkowo użyliśmy tablicy Kanban. Wykorzystaliśmy aplikację Trello, która pozwala na przypisanie zadania do członka zespołu. Było to bardzo pomocne, np. przy naprawie błędów. Łatwo można było się zorientować, jakie zadania są w trakcie implementacji, a jakie nie zostały jeszcze przypisane do programisty. Wprowadziliśmy również priorytetyzację zadań, aby wiedzieć, co musi być zrobione w danym sprincie, a co nie jest takie pilne.

Byliśmy trochę negatywnie nastawieni do użycia metodyki bazującej na scrumie do uczelnianego projektu. Jednak podsumowując, ma on wiele zalet. Spisanie wymagań, klarowny podział zadań - bardzo to usprawniło pracę nad systemem. Pozwoliliśmy sobie również na pewną elastyczność, lekko zmieniając harmonogram w trakcie prac oraz dodając nowe funkcjonalności, które z czasem okazały się potrzebne. Jeżeli będzie to możliwe, postaramy się użyć podobnej metodyki w innych naszych projektach.

## **7. Literatura**

- [1] Dokumentacja wraz z przewodnikami dla środowiska Android Studio,  
<https://developer.android.com/guide> (dostęp: 06.06.2019)
- [2] Dokumentacja i przewodniki dla Google Maps Api,  
<https://developers.google.com/maps/documentation/android-sdk/intro>  
(dostęp: 06.06.2019)