Wyniki etapu III: Projekt

System monitorowania przejazdów pojazdów autonomicznych „Al-Cab”

Projektowanie oprogramowania

Skład zespołu: Prowadzący:

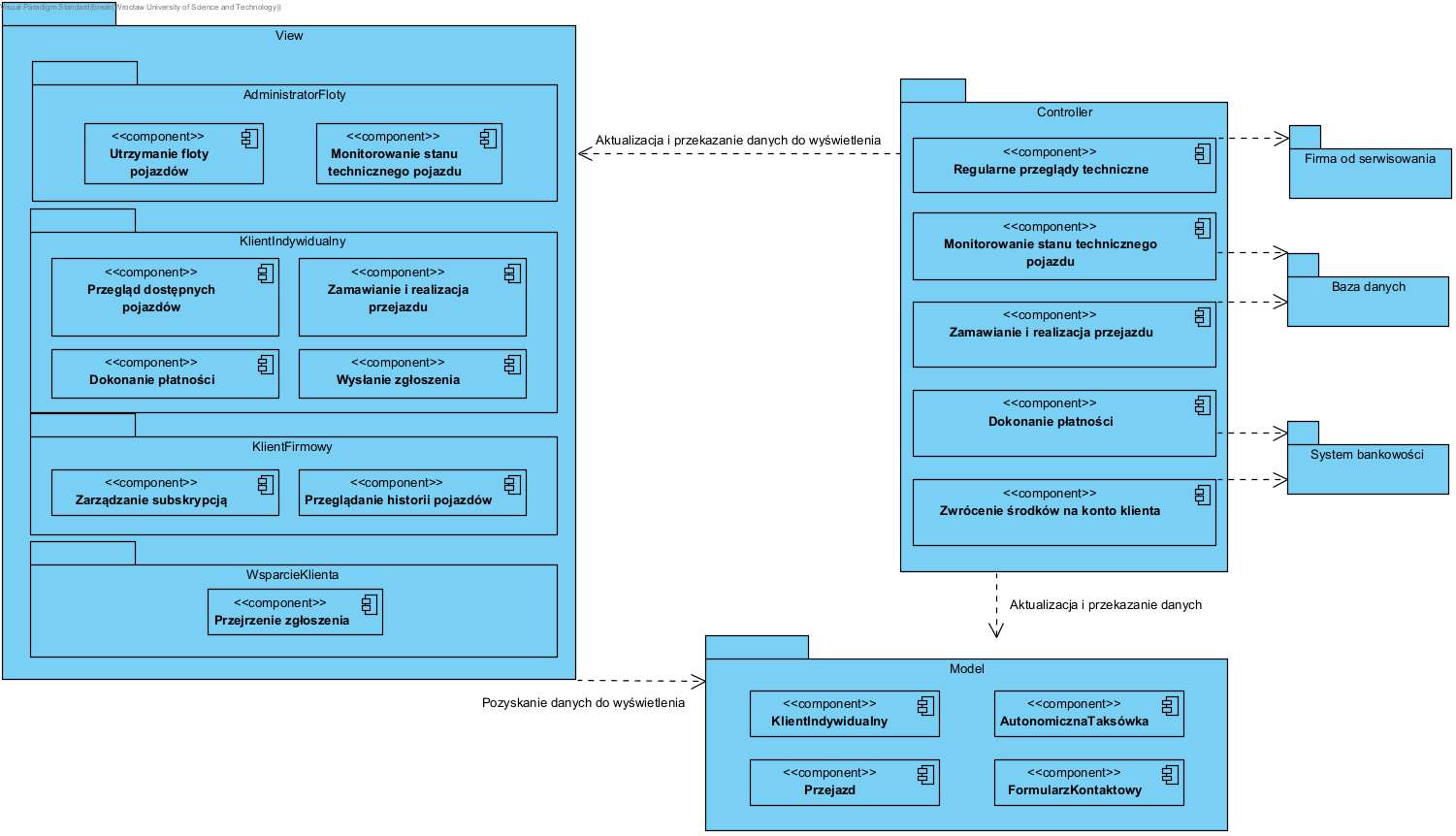
Aleksander Stepaniuk 272644

Kacper Zakrzewski 272693

Bartosz Dusza 272704

Architektura oprogramowania

# Architektura logiczna



# Architektura fizyczna

Opis elementów składowych diagramu:

**Węzły (Node):**

1. Serwer aplikacyjny
   * Centralny punkt przetwarzania danych i logiki systemu. Obsługuje komunikację między klientami (aplikacja mobilna/webowa), bazą danych oraz zewnętrznymi usługami.
   * Zawiera:
     + Backend Python Django - logika backendowa aplikacji, w tym obsługa użytkowników, przejazdów i komunikacji z innymi modułami.
     + Moduł map: komponent odpowiedzialny za integrację z API Google Maps w celu generowania tras i wyświetlania mapy w czasie rzeczywistym.
2. Serwer bazy danych
   * Przechowuje wszystkie dane systemu w strukturze relacyjnej MySQL.
   * Dane przechowywane to między innymi:
     + Informacje o użytkownikach (login, dane osobowe, zaszyfrowane hasło).
     + Przejazdy (data, trasa, pojazd, użytkownik).
     + Punkty serwisowe (lokalizacja, godziny pracy, szczegóły serwisowe).
   * Zawiera:
     + Baza danych MySQL: komponent zarządzający strukturą i operacjami na danych.
3. Pojazd autonomiczny
   * Node reprezentujący komunikację pojazdu z centralnym serwerem aplikacji.
   * Zawiera:
     + MQTT Client: komponent odpowiedzialny za wymianę danych w czasie rzeczywistym (np. lokalizacja pojazdu, status przejazdu).
4. Serwer usług zewnętrznych
   * Odpowiada za obsługę usług dostarczanych przez zewnętrzne API.
   * Zawiera:
     + Google Maps API: integracja z mapami, wyszukiwanie tras i lokalizacji.
     + Twilio SMS Service: wysyłanie SMS-ów w celu potwierdzenia tożsamości użytkownika.
     + SendGrid E-mail Service: wysyłanie e-maili w celu potwierdzenia rejestracji, resetu hasła itd.
5. Aplikacja mobilna
   * Główne narzędzie użytkownika końcowego. Zapewnia dostęp do wszystkich funkcjonalności systemu (zamawianie przejazdów, śledzenie taksówek, płatności).
   * Komunikuje się z backendem za pomocą protokołu HTTPS.
6. Aplikacja webowa
   * Narzędzie wspierające administratorów systemu, zespół wsparcia klienta oraz dział techniczny. Umożliwia zarządzanie użytkownikami, zgłoszeniami technicznymi oraz monitorowanie systemu.
   * Komunikuje się z backendem Django za pomocą protokołu HTTPS.

**Komponenty (Component):**

1. Backend Python Django
   * Główna logika aplikacji, obsługuje funkcjonalności systemu (np. uwierzytelnienie, zarządzanie przejazdami, wyszukiwanie taksówek).
2. Moduł map
   * Pośrednik między systemem, a Google Maps API. Przetwarza dane o trasach i wizualizuje mapy.
3. Baza danych MySQL
   * Odpowiada za przechowywanie danych systemowych, zapytania SQL i relacje między tabelami.
4. GPS + MQTT Client
   * Umożliwia komunikację między pojazdem a serwerem aplikacji w czasie rzeczywistym (np. przesyłanie pozycji GPS pojazdu).
5. Google Maps API
   * Dostarcza dane mapowe i usługi geolokalizacyjne.
6. Twilio SMS Service
   * Usługa do wysyłania wiadomości SMS w celu weryfikacji użytkowników.
7. SendGrid E-mail Service
   * Usługa do wysyłania wiadomości e-mail (potwierdzenia, powiadomienia).

**Artifacty (Artifact):**

1. app.py
   * Plik główny aplikacji Django, zawiera logikę serwera aplikacji.
   * Manifestowany przez komponent „Backend Django”.
2. google\_maps.py
   * Moduł obsługujący integrację z API Google Maps.
   * Manifestowany przez komponent „Moduł map”.
3. database.sql
   * Plik opisujący strukturę bazy danych i zawierający inicjalne dane.
   * Manifestowany przez komponent „Baza danych MySQL”.

**Połączenia (Connections):**

1. HTTPS
   * Między „Serwer aplikacyjny” a:
     + „Aplikacja mobilna”.
     + „Aplikacja webowa”.
     + „Serwer usług zewnętrznych”.
   * Zapewnia bezpieczną komunikację i szyfrowanie danych.
2. MQTT
   * Między „Serwer aplikacji” a „Pojazd autonomiczny”.
   * Umożliwia komunikację w czasie rzeczywistym między pojazdami, a centralnym systemem.

Obraz zawierający tekst, zrzut ekranu, diagram, Prostokąt

Opis wygenerowany automatycznie

# Model danych

[Class diagram – a refinement of information model when necessary (option).]

# Realizacja przypadków użycia

### Rejestracja użytkownika (Aleksander Stepaniuk 272644)

Obraz zawierający tekst, zrzut ekranu, Czcionka, diagram

Opis wygenerowany automatycznie

Obraz zawierający tekst, Równolegle, numer, diagram

Opis wygenerowany automatycznie

### Wybór pojazdu i miejsca docelowego przejazdu (Aleksander Stepaniuk 272644)

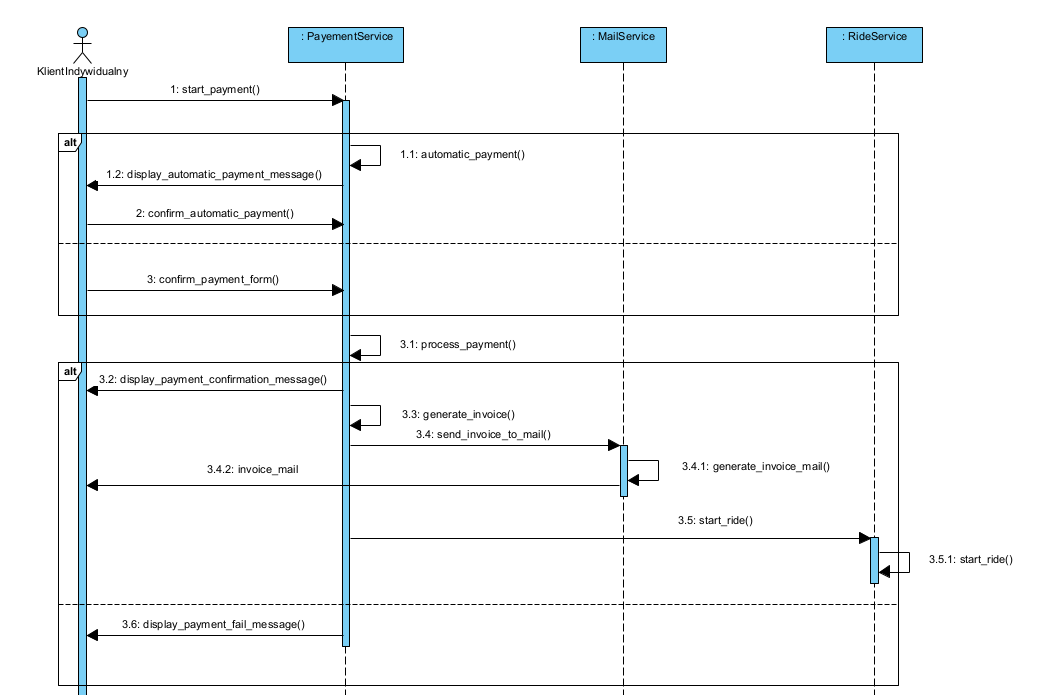
Obraz zawierający tekst, zrzut ekranu, Czcionka, diagram

Opis wygenerowany automatycznie

Obraz zawierający tekst, Równolegle, zrzut ekranu, linia

Opis wygenerowany automatycznie

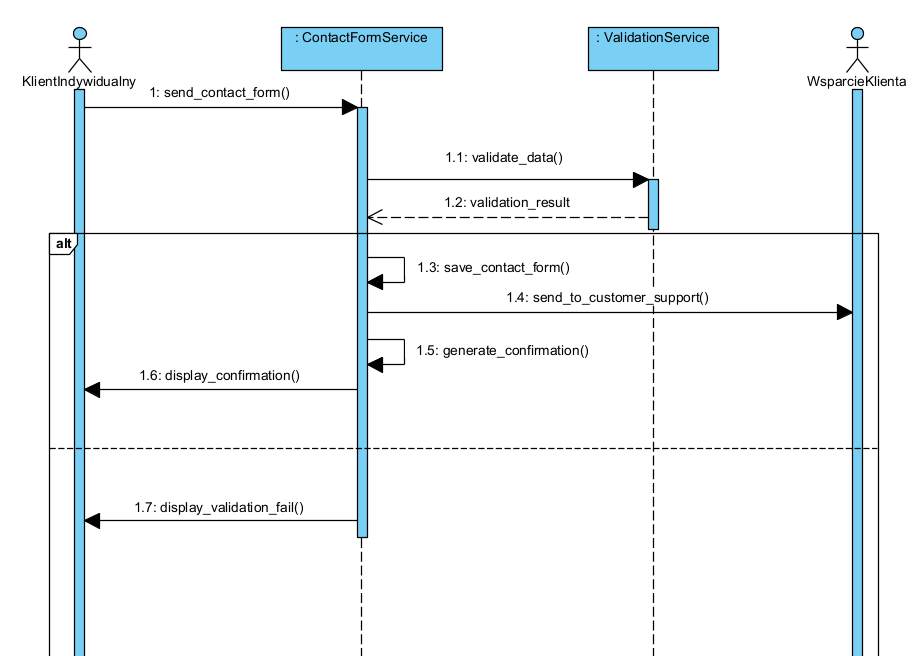
### Płatność za przejazd (Kacper Zakrzewski 272693) (Przepraszam za numerację, jest ona generowana automatycznie i nie wiem jak nad nią zapanować)



Obraz zawierający tekst, zrzut ekranu, Czcionka, diagram

Opis wygenerowany automatycznie

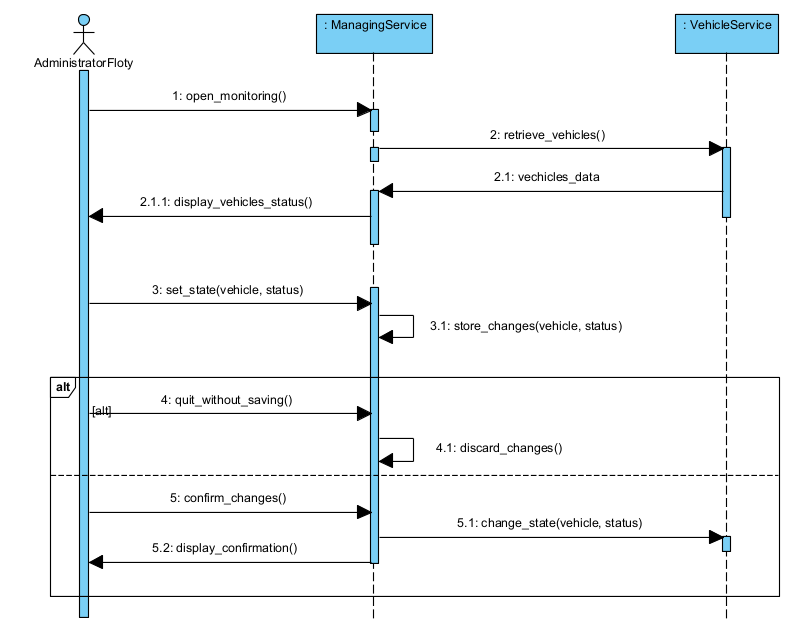
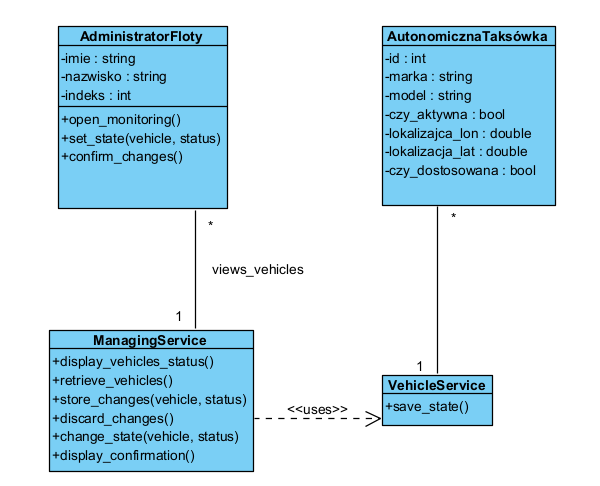
### Zgłaszanie problemów i wsparcie użytkownika (Kacper Zakrzewski 272693)



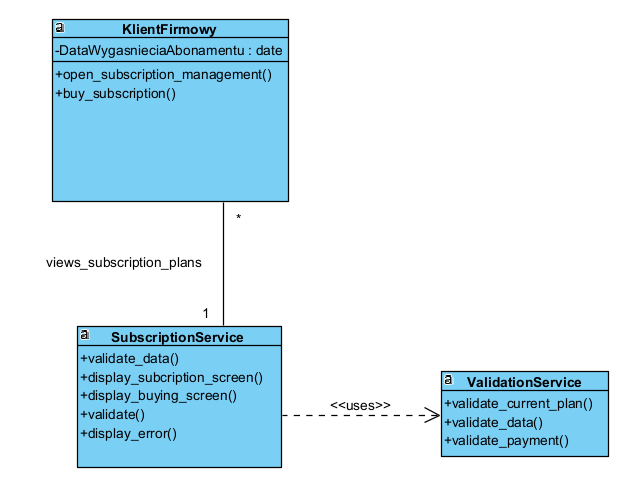
Obraz zawierający tekst, zrzut ekranu, Czcionka, diagram

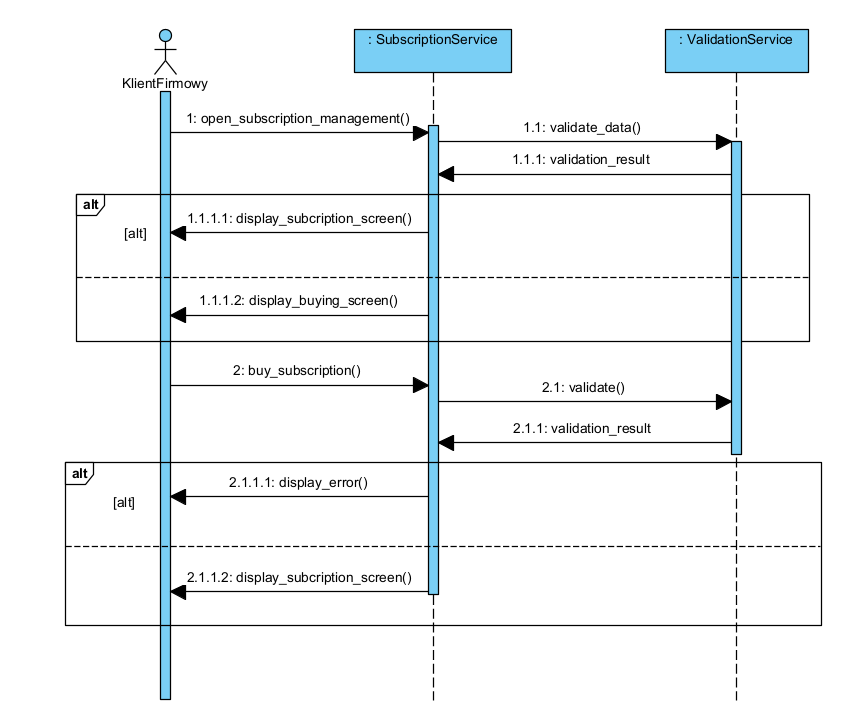
Opis wygenerowany automatycznie

### Monitorowanie stanu technicznego pojazdów (Bartosz Dusza 272704)



### Zakup subskrypcji firmowej (Bartosz Dusza 272704)





Projekt bazy danych

[Type (relational, files, non-relational, object), structure (tables, columns, keys, indexes or files with description of their content).]

[Info if database schema was generated, from which source and how (optional).]

