UNIWERSYTET RZESZOWSKI

WYDZIAŁ NAUK ŚCISŁYCH I TECHNICZNYCH

INSTYTUT INFORMATYKI

Oleksii Nawrocki, Tomasz Nowak

131400, 131478

Informatyka

System paczkomatów i logistyki - FasterPost

Praca projektowa

Praca wykonana pod kierunkiem

mgr inż. Marcin Mrukowicz

Rzeszów 2026

Spis treści

1. Wstęp i analiza biznesowa

1.1. Opis świata rzeczywistego i cel systemu

System **FasterPost** stanowi zintegrowaną platformę logistyczną, której głównym celem jest automatyzacja procesów nadawania, transportu i odbioru przesyłek w sieci skrytek paczkomatowych. Interakcja z systemem rozpoczyna się w momencie, gdy użytkownik – niezależnie czy jest to klient indywidualny, czy biznesowy – korzysta z aplikacji webowej w celu wypełnienia formularza nadawczego i realizacji płatności online.

Po pomyślnej transakcji system natychmiastowo rezerwuje odpowiednią skrytkę, umożliwiając nadawcy jej otwarcie za pomocą aplikacji, a następnie umieszczenie paczki wewnątrz maszyny. Zmiana statusu przesyłki następuje automatycznie, co uruchamia proces śledzenia w czasie rzeczywistym.

Logistyczna obsługa przesyłki w **FasterPost** opiera się na skoordynowanej pracy kurierów i systemów magazynowych. Kurier lokalny, wyposażony w dedykowany terminal mobilny, odbiera zgromadzone w paczkomatach przesyłki i transportuje je do magazynu lokalnego. Stamtąd trafiają one do sieci transportu międzymagazynowego (Line Haul). Dzięki zaimplementowanym algorytmom optymalizacji tras, paczki sprawnie przemieszczają się między regionami, by finalnie trafić ponownie w ręce lokalnego kuriera, który umieszcza je w docelowym paczkomacie.

Cała infrastruktura zarządzana jest poprzez system dedykowanych paneli, które dostosowują dostępne funkcjonalności do roli zalogowanego użytkownika. Klienci indywidualni skupiają się na pojedynczych operacjach, natomiast użytkownicy biznesowi mają dostęp do narzędzi masowego generowania etykiet. Nad stabilnością czuwa administrator, monitorujący raporty błędów i konfigurację sieci paczkomatów.

1.2. Analiza rozwiązań istniejących (SOTA)

Analiza rynku usług kurierskich (stan na grudzień 2025) pozwala wskazać kluczowe rozwiązania stanowiące punkt odniesienia dla systemu **FasterPost**:

1.2.1. InPost

Lider rynku oferujący sieć ponad 27 000 paczkomatów dostępnych w trybie 24/7. Nadanie i odbiór odbywa się poprzez aplikację mobilną (kod QR) lub kod PIN. Płatności realizowane są w pełni online. System charakteryzuje się zaawansowanym śledzeniem przesyłek oraz wdrażaniem autonomicznych maszyn.

1.2.2. DPD Pickup

Rozwiązanie hybrydowe, łączące automaty paczkowe z siecią punktów partnerskich (sklepy, stacje paliw). Wiele urządzeń to nowoczesne, bezprzewodowe automaty typu SwipBox Infinity. Aplikacja mobilna DPD Mobile umożliwia precyzyjną lokalizację punktów na mapie.

1.2.3. Orlen Paczka

Sieć obejmująca własne automaty oraz punkty partnerskie. Obsługa procesów nadawczo-odbiorczych realizowana jest przez aplikację Orlen Vitay, a standardowy czas na odbiór wynosi 3 dni z możliwością przedłużenia.

1.3. Wymagania funkcjonalne

Na podstawie analizy potrzeb użytkowników zdefiniowano następujące wymagania funkcjonalne systemu:

1.3.1. Obsługa użytkowników

- Rejestracja i logowanie użytkowników (klientów, kurierów, administratorów) poprzez zunifikowany panel.
- Edycja danych profilowych (dane kontaktowe, ustawienia bezpieczeństwa).
- Weryfikacja tożsamości poprzez wiadomość e-mail.

1.3.2. Obsługa paczek

- Tworzenie przesyłki poprzez interaktywny formularz.
- Generowanie etykiety nadawczej z unikalnym numerem śledzenia (Tracking ID).
- Wybór paczkomatu nadawczego i docelowego z mapy.
- Śledzenie statusu przesyłki w czasie rzeczywistym.
- Obsługa płatności online oraz kalkulator kosztów w zależności od gabarytu (S, M, L).

1.3.3. System kuriera

- Automatyczne planowanie optymalnej trasy na podstawie lokalizacji paczkomatów.
- Podgląd listy przesyłek przypisanych do bieżącego zlecenia (manifest).
- Mobilna obsługa zmiany statusów (np. „Odebrano z magazynu”, „Umieszczono w skrytce”).

1.3.4. Panel administracyjny

- Zarządzanie bazą użytkowników, paczek i kurierów (operacje CRUD).
- Podgląd raportów błędów, statystyk dostaw i obciążenia poszczególnych maszyn.
- Konfiguracja parametrów globalnych systemu.

1.3.5. System paczkomatowy

- Dynamiczna rezerwacja skrytki w momencie opłacenia przesyłki.
- Zdalne otwieranie skrytek z poziomu aplikacji.
- System powiadomień o gotowości paczki do odbioru.

1.4. Wymagania niefunkcjonalne

Aby zapewnić wysoką jakość usług, system **FasterPost** musi spełniać szereg wymagań jakościowych:

1. **Wydajność i skalowalność:** System powinien być przygotowany na obsługę dużej liczby równoległych zapytań, szczególnie w okresach wzmożonego ruchu (np. święta). Czas odpowiedzi API dla kluczowych operacji nie powinien przekraczać 1 sekundy.
2. **Bezpieczeństwo:**
   - Hasła użytkowników muszą być przechowywane w formie zaszyfrowanej (algorytm bcrypt).
   - Wymagana jest implementacja mechanizmów autoryzacji opartych na rolach.
   - Wszystkie działania administracyjne muszą być logowane.
3. **Użyteczność:** Interfejs użytkownika (Web i Mobile) musi być intuicyjny i responsywny (RWD), dostosowując się do urządzeń mobilnych.
4. **Dokumentacja:** System musi posiadać pełną dokumentację techniczną API oraz instrukcję wdrożenia.

2. Analiza systemowa i specyfikacja wymagań

W niniejszym rozdziale przeprowadzono szczegółową analizę funkcjonalną systemu FasterPost, identyfikując kluczowych aktorów oraz definiując scenariusze ich interakcji z aplikacją. Celem tej analizy było stworzenie precyzyjnego modelu wymagań, który posłużył jako fundament dla późniejszego procesu projektowania architektury oraz implementacji poszczególnych modułów oprogramowania.

2.1. Charakterystyka aktorów systemu

Na podstawie analizy dziedziny problemu oraz wymagań biznesowych wyodrębniono trzy główne grupy użytkowników (aktorów), którzy wchodzą w bezpośrednią interakcję z systemem. Każda z ról posiada odrębny zakres odpowiedzialności oraz dedykowany interfejs użytkownika.

Pierwszym i najważniejszym aktorem jest **Klient**. Jest to użytkownik końcowy, będący inicjatorem procesu logistycznego. W ramach systemu FasterPost rola ta została podzielona na użytkowników niezarejestrowanych, którzy posiadają dostęp jedynie do podstawowych informacji ofertowych oraz śledzenia przesyłek, oraz użytkowników zarejestrowanych. Klient zalogowany dysponuje pełnym wachlarzem funkcjonalności, obejmującym nadawanie przesyłek, zarządzanie książką adresową, dokonywanie płatności online oraz przegląd historii zleceń. Interakcja Klienta z systemem odbywa się głównie poprzez responsywną aplikację internetową.

Drugą kluczową rolę pełni **Kurier**. Jest to pracownik terenowy, odpowiedzialny za fizyczną realizację transportu przesyłek w ramach tzw. "Ostatniej Mili"(Last Mile) oraz transportu międzyhubowego. Aktor ten korzysta ze specjalistycznego modułu aplikacji, dostosowanego do urządzeń mobilnych. Jego głównym zadaniem jest odbiór paczek z maszyn nadawczych, transport do magazynu oraz dystrybucja do paczkomatów docelowych. System wspiera Kuriera poprzez automatyczne wyznaczanie optymalnych tras przejazdu oraz weryfikację dostępności skrytek w czasie rzeczywistym.

Trzecim aktorem jest **Administrator**. Rola ta posiada najwyższy poziom uprawnień i odpowiada za utrzymanie ciągłości działania systemu. Administrator nie bierze bezpośredniego udziału w procesie przemieszczania paczek, lecz sprawuje nadzór techniczny i operacyjny. Do jego obowiązków należy zarządzanie kontami użytkowników, monitorowanie stanu technicznego infrastruktury (paczkomatów) oraz interwencja w przypadku wystąpienia błędów krytycznych lub reklamacji.

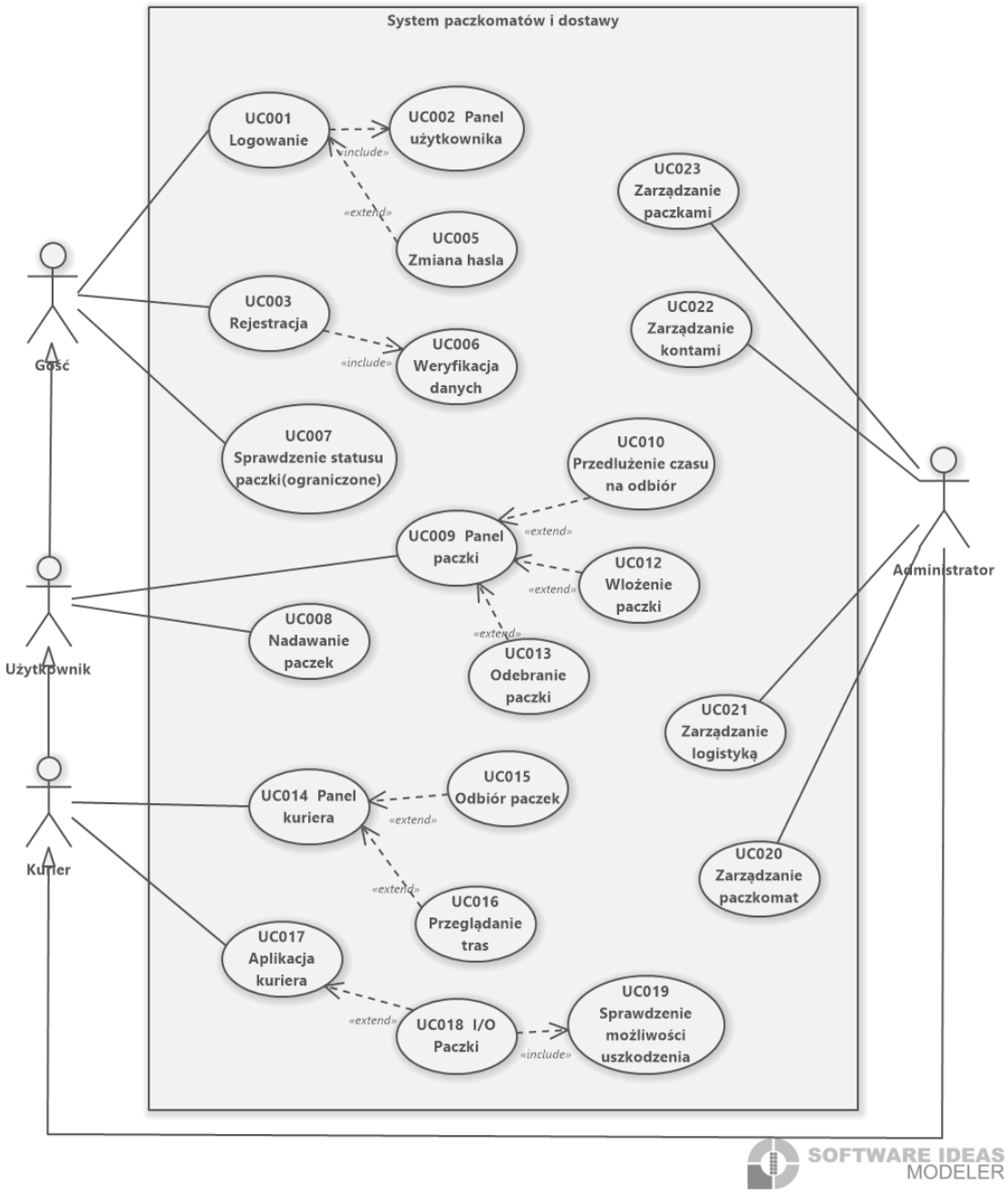
2.2. Model przypadków użycia

Diagram przypadków użycia (UML Use Case Diagram) stanowi wizualną reprezentację funkcjonalności systemu, obrazując relacje między zidentyfikowanymi aktorami a usługami oferowanymi przez aplikację FasterPost. Poniższy rysunek (Rys. 2.1) przedstawia kompleksowy widok procesów biznesowych zaimplementowanych w projekcie.

Centralnym punktem diagramu jest system zarządzania przesyłkami, który integruje działania wszystkich aktorów. Wyraźnie widoczny jest podział na strefę klienta (inicjacja zlecenia) oraz strefę operacyjną (realizacja zlecenia przez kuriera i nadzór administratora).

2.3. Szczegółowy opis scenariuszy biznesowych

Poniżej przedstawiono narracyjny opis kluczowych procesów, które zostały zwizualizowane na diagramie przypadków użycia.



Rys. 2.1. Diagram przypadków użycia systemu FasterPost

2.3.1. Proces nadania przesyłki

Proces nadania jest najbardziej złożonym scenariuszem po stronie Klienta. Rozpoczyna się on w momencie wyboru opcji "Nadaj paczkę"w panelu użytkownika. System wymaga od Klienta zdefiniowania parametrów fizycznych przesyłki (gabaryt S, M lub L) oraz wskazania danych adresata. Kluczowym elementem jest wybór punktów infrastrukturalnych – paczkomatu nadawczego oraz docelowego. System, wykorzystując dane geolokalizacyjne, sugeruje najbliższe dostępne maszyny.

Po zatwierdzeniu danych następuje proces płatności elektronicznej. System integruje się z zewnętrznym operatorem płatności, oczekując na potwierdzenie transakcji. Po pomyślnej autoryzacji płatności, system FasterPost automatycznie generuje etykietę przewozową oraz unikalny kod QR. Równocześnie następuje asynchroniczna komunikacja z wybranym paczkomatem nadawczym w celu wstępnej rezerwacji skrytki.

2.3.2. Obsługa logistyczna i kurierska

Z perspektywy Kuriera, praca z systemem rozpoczyna się od pobrania listy zadań (tzw. manifestu) na dany dzień. Aplikacja mobilna, komunikując się z API backendu, pobiera zoptymalizowaną trasę przejazdu, wyznaczoną algorytmami heurystycznymi. Kurier, docierając do paczkomatu, autoryzuje się w systemie maszyny, co umożliwia mu masowe otwarcie skrytek zawierających paczki do odbioru.

Każda operacja fizyczna na paczce (wyjęcie, załadunek na samochód, umieszczenie w magazynie) jest odnotowywana w systemie poprzez zeskanowanie kodu przesyłki. Dzięki temu status zamówienia jest aktualizowany w czasie rzeczywistym, co pozwala Klientowi na bieżące śledzenie losów przesyłki. W przypadku próby doręczenia paczki do przepełnionego paczkomatu, system dynamicznie weryfikuje dostępność skrytek i w razie potrzeby sugeruje Kurierowi alternatywne działania lub relokację przesyłki do magazynu buforowego.

2.3.3. Zarządzanie i administracja

Scenariusze administracyjne są niewidoczne dla zwykłego użytkownika, lecz kluczowe dla bezpieczeństwa danych. Administrator, logując się do dedykowanego panelu, ma możliwość przeglądu wszystkich transakcji i statusów w systemie. W sytuacjach awaryjnych, np. uszkodzenia drzwiczek w paczkomacie, Administrator ma możliwość zdalnego zablokowania konkretnej skrytki lub całej maszyny, wyłączając ją z puli dostępnych punktów dla nowych zleceń. Proces ten zapobiega frustracji użytkowników i zapewnia płynność działania sieci logistycznej.

3. Architektura i projektowanie techniczne

Niniejszy rozdział poświęcono szczegółowej charakterystyce technicznej systemu FasterPost. Przedstawiono w nim przyjętą architekturę oprogramowania, uzasadniono dobór technologii implementacyjnych oraz omówiono strukturę danych wraz z mapowaniem obiektowo-relacyjnym (ORM). Analizie poddano również organizację kodu warstwy prezentacji.

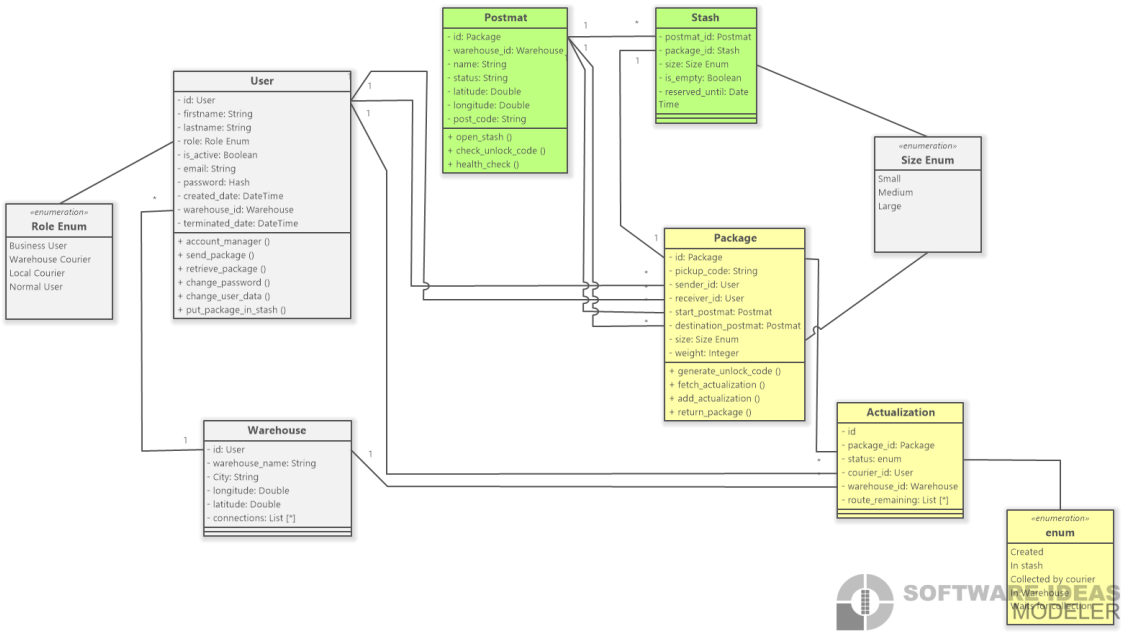
3.1. Architektura systemu

System FasterPost został zaprojektowany w oparciu o architekturę wielowarstwową typu klient-serwer, z wyraźnym odseparowaniem logiki biznesowej (Backend) od warstwy prezentacji (Frontend). Komunikacja między tymi elementami odbywa się w sposób bezstanowy za pośrednictwem interfejsu REST API.

Głównym założeniem architektonicznym było zastosowanie wzorca **Modularnego Monolitu**. Oznacza to, że mimo iż backend funkcjonuje jako pojedyncza jednostka wdrożeniowa, jego wnętrze zostało logicznie podzielone na niezależne domeny biznesowe (aplikacje), takie jak: logistyka, obsługa paczek, płatności czy zarządzanie użytkownikami. Taka separacja ułatwia zarządzanie kodem i ewentualną przyszłą migrację do architektury mikroserwisowej.

3.1.1. Diagram klas

Wizualizację logiczną struktury systemu oraz zależności między kluczowymi komponentami przedstawiono na diagramie klas (Rys. 3.1). Ukazuje on główne serwisy oraz obiekty domenowe wykorzystywane w procesach biznesowych.



Rys. 3.1. Diagram klas systemu FasterPost (widok logiczny)

3.2. Dobór technologii (Technology Stack)

Decyzje technologiczne zostały podjęte w oparciu o wymagania dotyczące wydajności, bezpieczeństwa typów danych oraz szybkości wytwarzania oprogramowania.

3.2.1. Warstwa serwerowa (Backend)

Fundamentem systemu jest język **Python 3.12** współpracujący z frameworkiem **Django 5.0**. Wybór ten podyktowany był dojrzałością ekosystemu oraz wbudowanymi mechanizmami bezpieczeństwa (ochrona przed atakami CSRF, SQL Injection).

- **Django REST Framework (DRF):** Wykorzystany do budowy ustandaryzowanego API, obsługi serializacji danych oraz autoryzacji opartej na tokenach.
- **Celery & Redis:** Zastosowane do obsługi zadań asynchronicznych i kolejkowania (np. generowanie tras, wysyłka e-maili), co zapobiega blokowaniu głównego wątku aplikacji podczas skomplikowanych obliczeń logistycznych.

3.2.2. Warstwa prezentacji (Frontend)

Aplikacja kliencka została zrealizowana przy użyciu frameworka **Next.js** (bazującego na bibliotece React). Wykorzystano nowoczesny model routingu (App Router), co pozwoliło na intuicyjne odwzorowanie struktury URL do fizycznej struktury plików. Za warstwę wizualną odpowiada **Tailwind CSS**, umożliwiający szybkie stylowanie komponentów zgodnie z podejściem *Utility-First*.

3.3. Projekt warstwy danych (ORM)

Warstwa danych została zaimplementowana przy użyciu systemu ORM (Object-Relational Mapping) wbudowanego w framework Django. Pozwoliło to na definicję struktury bazy danych w postaci klas Pythona, zapewniając niezależność od silnika bazodanowego (w projekcie wykorzystano PostgreSQL). W celu zapewnienia unikalności rekordów w systemie rozproszonym, jako klucze główne (Primary Keys) zastosowano standard UUID.

Poniżej omówiono kluczowe moduły danych zaimplementowane w systemie.

3.3.1. Moduł Użytkowników i Autoryzacji (Accounts)

Centralną encją jest model `User`, rozszerzający standardową klasę `AbstractBaseUser`. Model ten przechowuje dane uwierzytelniające oraz informacje profilowe. Zastosowano tu menadżera `MyAccountManager` do obsługi tworzenia kont. Wspierające modele to:

- `EmailVerification`: Powiązany relacją jeden-do-jednego (`OneToOne`) z użytkownikiem, przechowujący tokeny weryfikacyjne.
- `UserTOTP`: Odpowiedzialny za przechowywanie sekretów dla uwierzytelniania dwuskładnikowego (2FA), wykorzystujący bibliotekę `pyotp`.

3.3.2. Moduł Paczek (Packages)

Model `Package` stanowi serce systemu operacyjnego. Kluczowym atrybutem jest `pickup_code` (unikalny kod odbioru) oraz relacje kluczy obcych (`ForeignKey`) do:

- `sender` i `receiver`: Wskazujące na użytkowników systemu.

- `origin_postmat` i `destination_postmat`: Określające punkty startowe i końcowe procesu logistycznego.

Historia przesyłki jest przechowywana w modelu `Actualization`, który rejestruje każdą zmianę statusu (np. *IN_TRANSIT*, *DELIVERED*) wraz ze znacznikiem czasowym oraz identyfikatorem kuriera lub magazynu dokonującego zmiany.

3.3.3. Moduł Logistyki (Logistics)

Ten moduł odwzorowuje fizyczną infrastrukturę sieci transportowej.

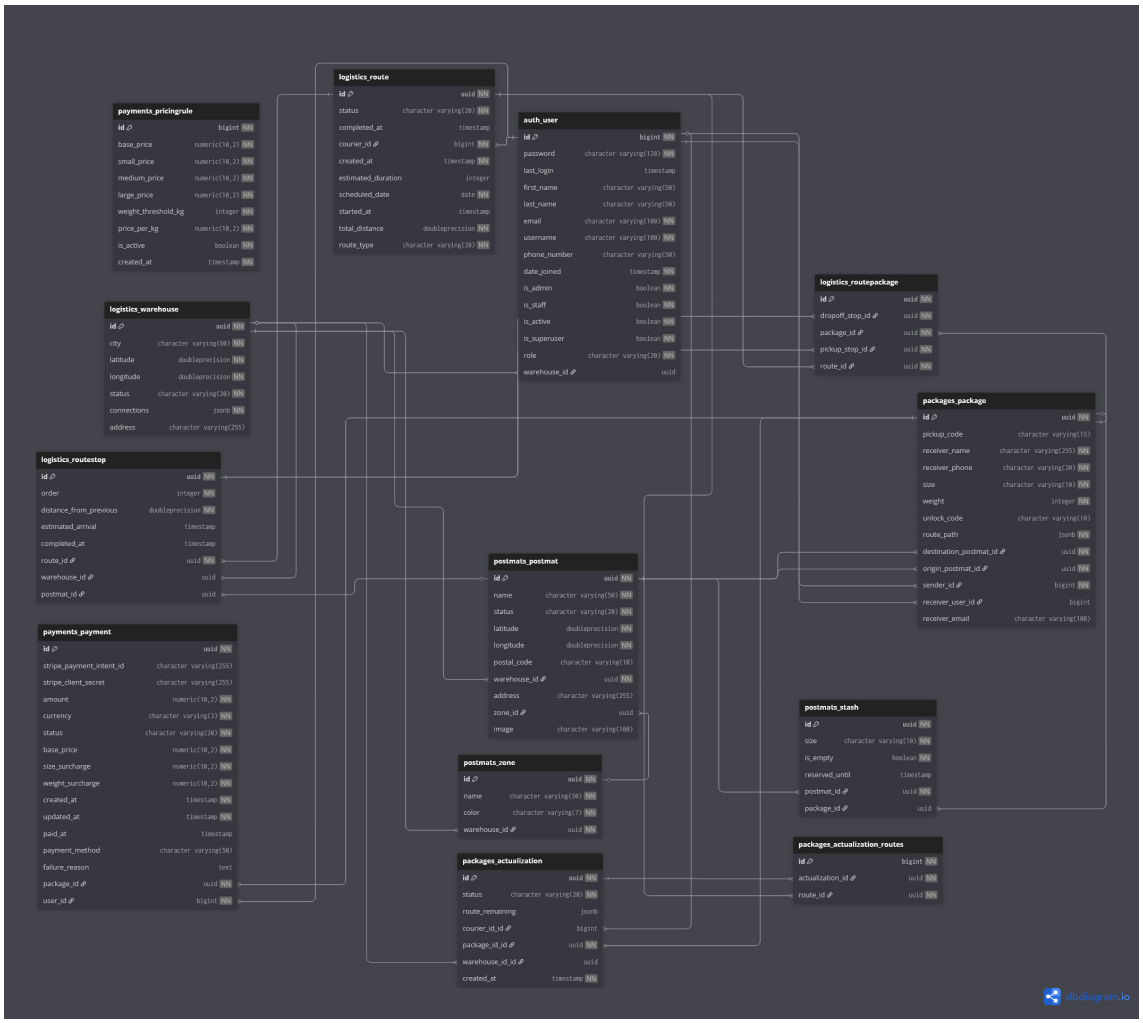
- `Warehouse`: Reprezentuje centra logistyczne (Huby). Przechowuje koordynaty geograficzne (Latitude/Longitude) niezbędne do obliczania dystansów (zaimplementowana metoda Haversine'a).
- `Route`: Definiuje trasę przypisaną do konkretnego kuriera i pojazdu na dany dzień.
- `RouteStop`: Model reprezentujący pojedynczy punkt postoju na trasie. Co istotne, posiada on relacje do magazynu (`Warehouse`) ORAZ paczkomatu (`Postmat`), z których tylko jedna może być aktywna dla danego rekordu. Pozwala to na elastyczne planowanie tras łączących różne typy punktów.

3.3.4. Moduł Infrastruktury (Postmats)

Model `Postmat` opisuje automaty paczkowe. Każdy paczkomat posiada zbiór skrytek zdefiniowanych w modelu `Stash`. Skrytki charakteryzują się rozmiarem (S, M, L) oraz statusem zajętości (`is_empty`), co jest kluczowe dla algorytmów alokacji przesyłek.

3.3.5. Diagram ERD

Pełny schemat bazy danych, uwzględniający wszystkie relacje oraz typy atrybutów, przedstawiono na diagramie związków encji (Rys. 3.2).



Rys. 3.2. Diagram związków encji (ERD) bazy danych systemu FasterPost

3.4. Struktura aplikacji klienckiej (Frontend)

Kod źródłowy aplikacji frontendowej został zorganizowany w strukturze katalogowej odzwierciedlającej podział na role użytkowników oraz domeny funkcjonalne. Wykorzystano mechanizm routingu Next.js, gdzie foldery wewnątrz katalogu `src/app` odpowiadają bezpośrednio ścieżkom URL w przeglądarce.

Analiza drzewa projektu (Listing 3.1) wskazuje na następujące kluczowe obszary:

1. **Strefa Publiczna:** Katalogi takie jak `login`, `register`, `track` oraz `find-point` są dostępne dla niezalogowanych użytkowników.
2. **Panel Administratora (`admin/`)**: Wydzielona sekcja zawierająca podmoduły do zarządzania logistyką (`logistics`), użytkownikami (`users`) oraz infrastrukturą (`postmats`).
3. **Panel Kuriera (`courier/`)**: Skupiony wokół dashboardu (`dashboard`), umożliwiającego szybki dostęp do listy zadań w terenie.
4. **Strefa Klienta Biznesowego (`business/`)**: Zawiera dedykowane widoki do zarządzania magazynami własnymi (`magazines`) oraz płatnościami (`payments`).
5. **Strefa Użytkownika (`user/`)**: Umożliwia zarządzanie profilem oraz podgląd szczegółów paczek (`packages/[id]`).

Listing 3.1. Struktura katalogów aplikacji frontendowej

```
src/app
|-- admin
|   |-- business
|   |-- logistics
|   |-- packages
|   |-- postmats
|   |-- routes
|   |-- users
|-- business
|   |-- dashboard
|   |-- magazines
|   |-- packages
|   |-- payments
|-- courier
|   |-- dashboard
|-- user
|   |-- packages
|   |   |-- [id]
|   |-- profile
|   |-- settings
|-- api
|-- login
|-- register
|-- track
|-- send-package
|-- pickup-package
```

Taka organizacja kodu wspiera modularność i ułatwia zarządzanie uprawnieniami (tzw. Route Protection), gdzie dostęp do poszczególnych gałęzi drzewa jest weryfikowany przez warstwę Middleware.

4. Realizacja logiki logistycznej i algorytmów

5. Planowanie, testowanie i wdrożenie

6. Podsumowanie i wnioski

Bibliografia

[1] InPost. Oferta dla biznesu i klientów indywidualnych. $https://inpost.pl$, 2025. Dostęp: 2026-01-31.

[2] Django Software Foundation. *Django Documentation*, 2025. URL: $https://docs.djangoproject.com/$.

[3] Meta Platforms. *React - A JavaScript library for building user interfaces*, 2025. URL: $https://react.dev/$.

Spis rysunków

Spis tabel

Spis listingów

Spis promptów

Streszczenie

Celem niniejszej pracy inżynierskiej było zaprojektowanie i zaimplementowanie kompleksowego systemu logistycznego o nazwie **FasterPost**, wspierającego automatyzację procesów dostaw w modelu „Ostatniej Mili" (Last Mile) oraz transportu międzyhubowego (Line Haul). Projekt stanowi odpowiedź na rosnące zapotrzebowanie rynku e-commerce na wydajne systemy obsługi automatów paczkowych.

W ramach projektu wytworzono wielomodułową platformę internetową, umożliwiającą użytkownikom nadawanie i śledzenie przesyłek, dokonywanie płatności online oraz bezobsługowy odbiór paczek ze skrytek. Kluczowym elementem systemu jest moduł logistyczny przeznaczony dla kurierów, który automatycznie generuje optymalne trasy przejazdu, oraz panel administracyjny służący do zarządzania infrastrukturą i monitorowania stanu sieci. W warstwie inżynierskiej zaimplementowano własne rozwiązania algorytmiczne dla problemu komiwojażera (TSP) oraz problemu routingu pojazdów z ograniczeniami (CVRP), co pozwoliło na dynamiczną optymalizację łańcucha dostaw.

System został zrealizowany w architekturze klient-serwer. Warstwa serwerowa (Backend) powstała w języku **Python 3.12** z wykorzystaniem frameworka **Django** oraz **Django REST Framework**. Jako system zarządzania bazą danych wybrano **PostgreSQL**. Do obsługi zadań asynchronicznych i kolejkowania procesów logistycznych wykorzystano technologie **Celery** oraz **Redis**. Warstwa prezentacji (Frontend) została zaimplementowana w oparciu o framework **Next.js** (React) oraz bibliotekę **Tailwind CSS**, zapewniając responsywność i dostępność na urządzeniach mobilnych. Całość rozwiązania została konteneryzowana przy użyciu platformy Docker.

Suplement: Rejestr wykorzystania narzędzi AI

W niniejszym suplemencie zamieszczono przykładowe zapytania (prompty) skierowane do narzędzi Generative AI, które wspomogły kluczowe etapy realizacji projektu FasterPost.

Kategoria: Logika i Algorytmy

Poniższy prompt posłużył do zaprojektowania wstępnej koncepcji algorytmu optymalizacji tras kurierskich (Last Mile).

```
Jako Senior Python Developer, pomóż mi zaprojektować serwis "RouteOptimizer" w Django.

Dane wejściowe:
- Lista punktów (Paczkomaty) z koordynatami (lat, lon).
- Punkt startowy (Magazyn).

Wymagania:
1. Użyj biblioteki geopy do obliczania dystansu (formuła Haversine).
2. Zastosuj algorytm "Nearest Neighbor" (Najbliższy Sąsiad) do wyznaczenia kolejności
    odwiedzin.
3. Kod ma być czysty, otypowany i uwzględniać obsługę błędów.

Wygeneruj szkielet klasy w Pythonie.
```

Prompt 6.0.1. Prompt inicjujący prace nad modułem optymalizacji tras

Kategoria: Frontend i UI

Prompt wykorzystany do szybkiego prototypowania widoku dla kurierów w technologii Next.js.

```
Stwórz komponent React (Next.js App Router) dla "Courier Dashboard".
Stylizacja: Tailwind CSS (Utility-First).

Funkcjonalność:
- Wyświetl listę przesyłek w formie kafelków ("Task Cards").
- Każdy kafelek musi zawierać: Adres, Status przesyłki i przycisk "Zmień status".
- Widok musi być w pełni responsywny (Mobile First), przystosowany do obsługi na
    smartfonie.
```

Prompt 6.0.2. Prompt wspierający tworzenie interfejsu aplikacji mobilnej kuriera

Załącznik nr 2 do Zarządzenia nr 228/2021 Rektora Uniwersytetu Rzeszowskiego z dnia 1 grudnia 2021 roku w sprawie ustalenia procedury antyplagiatowej w Uniwersytecie Rzeszowskim

OŚWIADCZENIE STUDENTA O SAMODZIELNOŚCI PRACY

……Oleksii Nawrocki, Tomasz Nowak……
Imię (imiona) i nazwisko studenta

Wydział Nauk Ścisłych i Technicznych

……Informatyka……
Nazwa kierunku

……131400, 131478……
Numer albumu

1. Oświadczam, że moja praca projektowa pt.: System paczkomatów i logistyki - FasterPost
    1) została przygotowana przeze mnie samodzielnie*,
    2) nie narusza praw autorskich w rozumieniu ustawy z dnia 4 lutego 1994 roku o prawie autorskim i prawach pokrewnych (t.j. Dz.U. z 2021 r., poz. 1062) oraz dóbr osobistych chronionych prawem cywilnym,
    3) nie zawiera danych i informacji, które uzyskałem/am w sposób niedozwolony,
    4) nie była podstawą otrzymania oceny z innego przedmiotu na uczelni wyższej ani mnie, ani innej osobie.
2. Jednocześnie wyrażam zgodę/nie wyrażam zgody** na udostępnienie mojej pracy projektowej do celów naukowo–badawczych z poszanowaniem przepisów ustawy o prawie autorskim i prawach pokrewnych.

(miejscowość, data) (czytelny podpis studenta)

* Uwzględniając merytoryczny wkład prowadzącego przedmiot

** – niepotrzebne skreślić