

Spis treści

1. [O projekcie](#)
 2. [Zespół](#)
 3. [Stos Technologiczny](#)
 4. [Struktura Projektu](#)
 5. [Modele Danych](#)
 6. [Logika Działania Aplikacji](#)
 7. [Komunikacja, Autoryzacja i API](#)
 8. [Zadania Asynchroniczne](#)
 9. [Konteneryzacja](#)
 10. [Testy i CI/CD](#)
 11. [Instalacja i Uruchomienie](#)
 12. [Funkcjonalności](#)
 13. [Przegląd Widoków \(Galeria\)](#)
 14. [Dodatkowa Konfiguracja](#)
-

1. Wstęp i Cel Projektu

FasterPost to kompleksowy system logistyczny obsługujący proces dostarczania przesyłek w modelu dwuwarstwowym:

1. **Logistyka Krajowa (Hub-to-Hub):** Transport międzygrodowy pomiędzy głównymi magazynami.
2. **Logistyka Lokalna (Last Mile):** Dostarczanie przesyłek z magazynów lokalnych do paczkomatów.

Celem aplikacji jest optymalizacja tras kurierskich, zarządzanie flotą pojazdów oraz automatyzacja procesu alokacji przesyłek do skrytek paczkomatowych.

Zespół

Profil GitHub	Rola w projekcie
Oleksii Nawrocki	Lider zespołu / Logistics
Tomasz Nowak	Auth / Backend

2. Stos Technologiczny (Tech Stack)

Aplikacja została zbudowana w architekturze klient-serwer (rozdzielony frontend i backend).

Backend

- **Język:** Python 3.x
- **Framework:** Django (z Django REST Framework)
- **Baza danych:** PostgreSQL (relacyjna)
- **Biblioteki algorytmiczne:** NumPy, Scikit-learn (do K-Means), GeoPy (do obliczeń geograficznych).

Frontend

- **Framework:** Next.js (React)
 - **Język:** TypeScript / JavaScript
 - **UI:** Tailwind CSS / Lucide React (ikony)
 - **Komunikacja:** Axios / Fetch API
-

3. Struktura Projektu

Projekt podzielony jest na dwa główne katalogi:

```
fasterpost/
├── backend/
│   ├── accounts/                                # Logika biznesowa i API (Django)
│   ├── packages/                               # Zarządzanie użytkownikami i autoryzacją
│   ├── logistics/                             # Logika przesyłek i śledzenia
│   │   ├── services/                           # Aplikacja: Logistyka Krajowa
│   │   └── models.py                          # Algorytmy routingu (VRP)
│   ├── postmats/                            # Modele Hubów i Tras krajowych
│   │   ├── management/                      # Aplikacja: Logistyka Lokalna
│   │   ├── services/                         # Komendy (np. seed_zones - K-Means)
│   │   └── models.py                        # Algorytmy TSP i alokacji skrytek
│   ├── core/                                 # Modele Paczkomatów i Przesyłek
│   ├── tests/                                # Ustawienia globalne, konfiguracja Celery
│   ├── Dockerfile                            # Testy jednostkowe i integracyjne
│   └── requirements.txt                     # Obraz dla backendu
                                                # Zależności Python
                                                # Interfejs użytkownika (Next.js)
├── frontend/
│   ├── src/
│   │   ├── components/                      # Komponenty React
│   │   ├── pages/                            # Widoki aplikacji
│   │   └── services/                         # Klient API (połączenie z backendem)
│   └── Dockerfile                           # Obraz dla frontendu
└── .github/
    └── workflows/
        └── tests.yml                         # Konfiguracja CI/CD (GitHub Actions)
                                            # Automatyczne uruchamianie testów
├── docker-compose.yml                     # Orkiestracja kontenerów
└── docs/                                  # Dokumentacja projektowa
```

4. Modele Danych (Backend)

System opiera się na relacyjnej bazie danych. Kluczowe encje to:

A. Użytkownicy (Accounts)

- **User:** Rozszerzony model użytkownika Django.
 - `email` (Primary Key), `first_name`, `last_name`.
 - `role`: Enum (`client`, `courier`, `admin`, `warehouse_manager`).
 - `warehouse`: FK do `Warehouse` (tylko dla kurierów i magazynierów - przypisanie do bazy).

B. Logistyka (Logistics)

- **Hub (Magazyn):** Reprezentuje węzeł w sieci krajowej.
 - `name, city, address.`
 - `latitude, longitude` (Decimal).
 - `connections`: Many-to-Many (graf połączeń między magazynami).
- **Vehicle (Pojazd):**
 - `registration_number, capacity` (domyślnie 50).
 - `max_work_minutes` (720), `current_hub` (FK).
- **Route (Trasa Krajowa):**
 - `courier` (FK User), `scheduled_date`.
 - `status` (`planned, in_progress, completed`).
 - `total_distance, estimated_duration`.
- **RouteStop (Przystanek):**
 - `route` (FK), `order` (int), `warehouse` (FK) lub `postmat` (FK).
 - `arrival_time, departure_time`.

C. Paczkomaty (Postmats)

- **Zone (Strefa):**
 - `name, warehouse` (FK), `color` (do wizualizacji na mapie).
- **Postmat (Paczkomat):**
 - `name, address, latitude, longitude`.
 - `zone` (FK Zone - wynik algorytmu K-Means).
 - `is_active` (bool).
- **Stash (Skrytka):**
 - `postmat` (FK), `size` (`small, medium, large`).
 - `is_empty` (bool), `reserved_until` (DateTime, nullable).
- **Package (Paczka):**
 - `tracking_number` (UUID/String).
 - `sender` (FK User), `receiver_email`.
 - `size, weight`.
 - `status` (`created, paid, in_warehouse, in_transit, delivered`).
 - `origin_postmat` (FK), `destination_postmat` (FK).
- **Actualization (Historia statusów):**
 - `package` (FK), `status, timestamp, location_description`.

Schemat relacji encji (ERD):

!ERD Diagram

5. Logika Działania Aplikacji

5.1. Warstwa Krajowa (Hub-to-Hub)

Logika zaimplementowana w `backend/logistics/services/routing_service.py`.

1. **Problem:** CVRP (Capacitated Vehicle Routing Problem) z oknami czasowymi.

2. **Algorytm:** Heurystyka zachłanna (Greedy Construction).

3. **Przebieg:**

- System grupuje paczki według Huba startowego.
- Dla każdego pojazdu wybierany jest cel metodą **Nearest Neighbor** (Najbliższy Sąsiad).
- Sprawdzane są ograniczenia: czas pracy (12h) i pojemność.
- **Backhauling:** W drodze powrotnej pojazd zabiera paczki zmierzające do jego bazy macierzystej, aby uniknąć "pustych przebiegów".

5.2. Warstwa Lokalna (Last Mile)

Logika zaimplementowana w `backend/postmats/services/routing_service.py`.

1. **Zoning (K-Means):** Miasto dzielone jest na strefy (klastry) w oparciu o lokalizację paczkomatów.

Kurierzy są przypisani do stref.

2. **Alokacja:** Przed wyjazdem system sprawdza dostępność skrytek (`is_empty=True`) w paczkomacie docelowym. Jeśli brak miejsca -> status *DELAYED*.

3. **Routing (TSP):** Wewnątrz strefy trasa wyznaczana jest algorytmem **Nearest Neighbor** z użyciem formuły **Haversine** do obliczania odległości sferycznej.

5.3. Zarządzanie Przesyłką i Płatności

1. **Nadawanie i Wybór Paczkomatu:**

- Użytkownik wybiera paczkomat nadawczy i odbiorczy z interaktywnej mapy (Frontend).
- System filtryuje paczkomaty dostępne w bazie danych, umożliwiając wybór tylko aktywnych punktów.

2. **Płatności:**

- Koszt przesyłki obliczany jest dynamicznie na podstawie wybranego gabarytu (S/M/L) oraz wagi.
- System przewiduje integrację z bramką płatności Stripe. Po pomyślnej transakcji paczka zmienia status na **PAID** i staje się widoczna dla algorytmów logistycznych oraz niemożliwa jest jej późniejsza edycja.

3. **Edycja Danych:**

- Edycja przesyłki (np. zmiana odbiorcy, rozmiaru, paczkomatu docelowego) jest możliwa **wyłącznie przed dokonaniem płatności** (status **CREATED**).
- Po opłaceniu (**PAID**) dane są "zamrożone", ponieważ system mógł już rozpocząć proces rezerwacji skrytki lub planowania logistyki.

5.4. Szczegóły Systemu Rezerwacji Skrytek

Mechanizm ten zapobiega przepełnieniu paczkomatów (logika w `backend/postmats/services/routing_service.py`).

1. **Weryfikacja przed trasą:** Podczas generowania trasy lokalnej, system sprawdza dostępność skrytek w paczkomacie docelowym dla każdej paczki.

2. **Kryteria dostępności:** Skrytka jest uznana za wolną tylko wtedy, gdy spełnia łącznie dwa warunki:

- Jest fizycznie pusta (`is_empty=True`).
- Nie posiada aktywnej rezerwacji (`reserved_until IS NULL`).

3. **Dopasowanie:** Algorytm dobiera skrytkę odpowiednią do rozmiaru paczki (S do S, M do M itd.).

4. **Rezerwacja (Lock):**

- W momencie utworzenia paczki, system rezerwuje skrytkę na 24 godziny.

- Po opłaceniu przesyłki, rezerwacja jest odnawiana na kolejne 24 godziny od momentu płatności.
- Status skrytki zmienia się na zajęty (`is_empty=False`), co blokuje możliwość przypisania tam innej paczki przez ten czas.

5. **Brak miejsca:** Jeśli paczkomat jest pełny, paczka otrzymuje status `DELAYED` i zostaje w magazynie do następnego cyklu dostaw.

6. Komunikacja, Autoryzacja i API

Aplikacja wykorzystuje architekturę **REST API**. Frontend komunikuje się z backendem za pomocą zapytań HTTP (JSON).

Role w systemie: - **Użytkownik:** Może tworzyć przesyłki i śledzić tylko swoje paczki. - **Kurier:** Ma dostęp do przypisanych mu tras oraz możliwość zmiany statusu paczek (np. przy odbiorze/dostarczeniu). - **Magazynier/Admin:** Pełny dostęp do zarządzania flotą i generowania tras.

6.1. Dokumentacja API (Swagger/OpenAPI)

Pełna lista endpointów, wraz z wymaganymi parametrami i strukturą odpowiedzi, jest generowana automatycznie i dostępna pod adresem:

- **Swagger UI:** </api/schema/swagger-ui/>
- **Redoc:** </api/schema/redoc/>

6.2. System Autoryzacji (HttpOnly Cookie)

System wykorzystuje standardowy mechanizm tokenów autoryzacyjnych, jednak ze względów bezpieczeństwa token jest przechowywany wyłącznie w ciasteczkach **HttpOnly Cookie**. Dzięki temu kod JavaScript (Frontend) nie ma do niego dostępu, co zabezpiecza aplikację przed atakami typu XSS.

Mechanizm działania:

1. **Logowanie:** Użytkownik wysyła `email` i `password` na endpoint logowania.
2. **Generacja Tokena:** Serwer weryfikuje dane i ustawia token w odpowiedzi jako ciasteczko z flagą `HttpOnly`.
 - **Przechowywanie:** Ciasteczko jest niewidoczne dla skryptów JS.
 - **Bezpieczeństwo:** `SameSite=Lax` oraz `Secure=True` (na produkcji).
3. **Komunikacja:** Przeglądarka automatycznie dołącza ciasteczko do każdego zapytania do API. Backend odczytuje token z ciasteczka, a nie z nagłówka `Authorization`.
4. **Wylogowanie:** Serwer wysyła polecenie usunięcia ciasteczka (set-cookie z datą w przeszłości).

Role i Uprawnienia (Permissions):

System wykorzystuje niestandardowe klasy uprawnień (Custom Permissions) w Django:

- `IsCourier`: Sprawdza, czy `user.role == 'courier'`.
- `IsWarehouseManager`: Dostęp do panelu generowania tras.
- `IsOwnerOrReadOnly`: Użytkownik widzi tylko swoje paczki, chyba że jest pracownikiem.

Szczegółowy przepływ danych:

Scenariusz 1: Wyświetlenie mapy paczkomatów

1. **Frontend:** Użytkownik wchodzi na stronę mapy. Komponent React (np. `MapComponent`) w hooku `useEffect` wywołuje funkcję serwisu.
2. **Request:** `GET /api/postmats/`
3. **Backend:**
 - Django Viewset odbiera zapytanie.
 - Pobiera listę obiektów `Postmat` z bazy danych.
 - Serializer zamienia obiekty Pythonowe na JSON (zawierający `lat`, `lng`, `status`).
4. **Response:** JSON z listą paczkomatów.
5. **Frontend:** Otrzymuje dane i renderuje markery na mapie (np. używając Leaflet lub Google Maps).

Scenariusz 2: Generowanie tras (Panel Administratora)

1. **Frontend:** Administrator kliką przycisk "Generuj Trasy".
2. **Request:** `POST /api/logistics/generate-routes/`
3. **Backend:**
 - Uruchamia `RoutingService`.
 - Algorytm pobiera nieobsłużone paczki.
 - Wylicza trasy (zgodnie z logiką opisaną w pkt 5).
 - Zapisuje nowe obiekty `Route` w bazie danych.
 - Zwraca status operacji.
4. **Response:** `200 OK` + podsumowanie (np. "Wygenerowano 5 tras").
5. **Frontend:** Wyświetla powiadomienie o sukcesie i odświeża listę tras.

Scenariusz 3: Śledzenie paczki

1. **Frontend:** Klient wpisuje numer paczki.
2. **Request:** `GET /api/packages/{id}/track/`
3. **Backend:** Sprawdza status paczki i jej ostatnią lokalizację (Hub lub Paczkomat).
4. **Response:** JSON { "status": "IN_TRANSIT", "location": "Hub Warszawa", "estimated_delivery": "2024-05-20" }.

7. Zadania Asynchroniczne (Celery)

Ze względu na złożoność obliczeniową algorytmów oraz konieczność wysyłania powiadomień, projekt wykorzystuje **Celery** z brokerem wiadomości **Redis**.

Główne zadania (Tasks):

2. **send_status_email_task:**
 - Wyzwalane sygnałem (Django Signals) przy zmianie statusu paczki (np. na `DELIVERED`).
 - Wysyła e-mail do klienta z informacją o zmianie statusu.
3. **release_expired_reservations:**
 - Zadanie okresowe (Celery Beat).
 - Sprawdza skrytki, których `reserved_until` minął, i zwalnia je (`is_empty=True`), jeśli paczka nie dotarła.

8. Konteneryzacja (Docker Compose)

Całe środowisko jest skonteneryzowane, co zapewnia spójność między środowiskiem deweloperskim a produkcyjnym. Plik `docker-compose.yml` definiuje następujące usługi:

1. **db**: Baza danych PostgreSQL (Alpine Linux). Dane są persystowane w wolumenie `postgres_data`.
2. **backend**: Kontener Django (Gunicorn/Uvicorn).
 - Zależy od `db` i `redis`.
 - Uruchamia migracje przy starcie.
3. **frontend**: Kontener Next.js (Node.js).
 - Budowany wieloetapowo (Multi-stage build) dla optymalizacji rozmiaru obrazu.
4. **redis**: Broker wiadomości dla Celery oraz cache.
5. **worker**: Instancja Celery przetwarzająca zadania w tle.
6. **beat**: Harmonogram zadań okresowych Celery.

Uruchomienie całego środowiska:

```
docker-compose up --build
```

9. Testy i CI/CD

Projekt kładzie duży nacisk na jakość kodu, wykorzystując testy automatyczne uruchamiane w potoku CI (GitHub Actions).

Rodzaje testów:

- **Testy Jednostkowe (Unit Tests)**: Testowanie pojedynczych funkcji algorytmicznych (np. czy funkcja Haversine poprawnie liczy odległość).
- **Testy Integracyjne (Integration Tests)**: Testowanie endpointów API (np. czy próba rezerwacji zajętej skrytki zwraca błąd). Używamy `APITestCase` z Django REST Framework.

Continuous Integration (CI):

Plik `.github/workflows/tests.yml` definiuje proces, który uruchamia się przy każdym `push` do repozytorium:

1. Postawienie kontenera z bazą danych PostgreSQL (Service Container).
2. Instalacja zależności (`pip install -r requirements.txt`).
3. Uruchomienie lintera (np. `flake8`) w celu sprawdzenia stylu kodu.
4. Wykonanie testów: `python manage.py test`.

10. Instalacja i Uruchomienie

Wymagania wstępne

- Docker
- Docker Compose

10.1. Uruchomienie (Docker)

1. **Konfiguracja środowiska:** Skopiuj plik przykładowy `.env`.

```
cd src  
cp .env.example .env
```

2. **Uruchomienie kontenerów:**

```
docker-compose -f backend/docker-compose.yml build --no-cache  
docker-compose -f backend/docker-compose.yml up
```

3. **Inicjalizacja aplikacji:**

```
docker-compose run web sh -c "python manage.py makemigrations"  
docker-compose run web sh -c "python manage.py migrate"
```

4. **Seedowanie**

```
docker-compose run web sh -c "python manage.py seed_warehouses"  
docker-compose run web sh -c "python manage.py seed_logistics"  
docker-compose run web sh -c "python manage.py seed_accounts"  
docker-compose run web sh -c "python manage.py seed_zones"  
docker-compose run web sh -c "python manage.py seed_local_delivery"
```

5. **Testowanie**

```
docker-compose -f test.yml run web sh -c  
"DJANGO_SETTINGS_MODULE=proj.settings_test python manage.py test"
```

Aplikacja frontendowa jest dostępna pod adresem: <http://localhost:80> Aplikacja backendowa jest dostępna pod adresem: <http://localhost:8000>

Przydatne komendy Docker

- Zatrzymanie kontenerów: `docker-compose down`
- Ponowne uruchomienie: `docker-compose up -d`
- Logi: `docker-compose logs -f`

10.2. Uruchomienie Lokalne (Bez Dockera)

Backend

```
cd backend  
pip install -r requirements.txt  
python manage.py migrate  
python manage.py seed_zones # Inicjalizacja stref K-Means  
python manage.py runserver
```

Frontend

```
cd frontend  
npm install  
npm run dev
```

11. Funkcjonalności

System oferuje następujące możliwości w podziale na role:

Użytkownik (Klient)

- Zakładanie konta i logowanie.
- Nadawanie paczek i wybór paczkomatu.
- Śledzenie statusu przesyłki.
- Odbieranie paczek.

Kurier

- Przegląd paczek do odebrania i dostarczenia.
 - Obsługa procesu umieszczania paczki w paczkomacie.
 - Transport między magazynami a paczkomatami.
-

Administrator

- Zarządzanie użytkownikami (CRUD).
 - Zarządzanie paczkomatami i magazynami.
 - Podgląd logistyki.
-

Przegląd Widoków (Galeria)

1. Strona Główna i Uwierzytelnianie

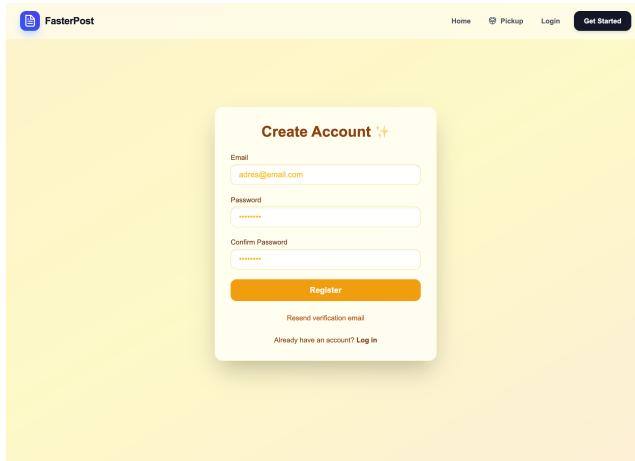
Strona Główna Logowanie

 Home Page

 Login

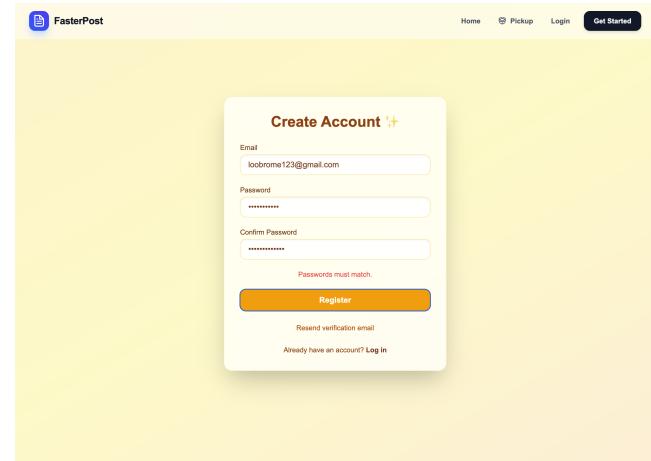
Rejestracja i Weryfikacja:

Rejestracja



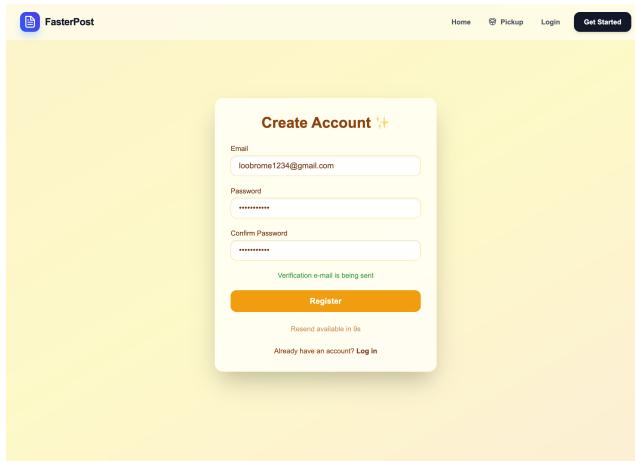
The screenshot shows the 'Create Account' form. The 'Email' field contains 'adres@email.com'. The 'Password' and 'Confirm Password' fields both contain '*****'. A message at the bottom says 'Passwords must match.' Below the form is a 'Register' button.

Walidacja Błędów



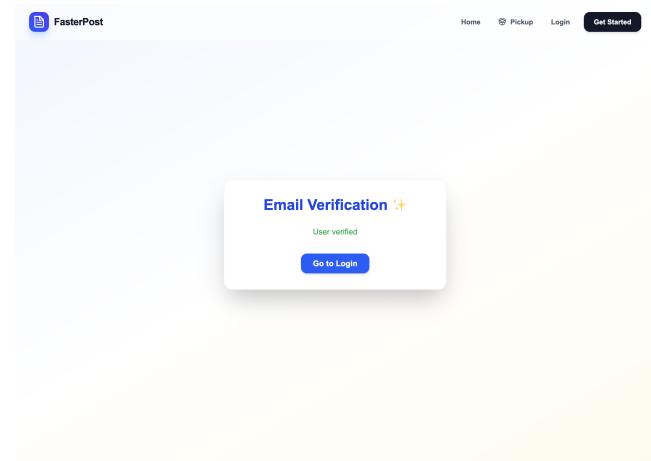
The screenshot shows the same 'Create Account' form as above, but with validation errors. The 'Email' field contains 'loobrome123@gmail.com'. The 'Password' and 'Confirm Password' fields both contain '*****'. A message at the bottom says 'Passwords must match.' Below the form is a 'Register' button.

Poprawne Dane



The screenshot shows the 'Create Account' form. The 'Email' field contains 'loobrome1234@gmail.com'. The 'Password' and 'Confirm Password' fields both contain '*****'. A message at the bottom says 'Verification e-mail is being sent.' Below the form is a 'Register' button.

Weryfikacja Email



The screenshot shows the 'Email Verification' page. It displays the message 'User verified' and a 'Go to Login' button.

2. Panel Użytkownika Indywidualnego

Dashboard i Ustawienia:

Kokpit Użytkownika

Nadchodzące Paczki

Kokpit Użytkownika

Nadchodzące Paczki

The screenshot shows two main sections: "Active Shipments" and "Incoming Shipments".

- Active Shipments:** Shows a message: "Manage and track your ongoing deliveries." with a "+ New Package" button. Below it, a box says "No packages found. Use the '+ New Package' button to send your first item."
- Incoming Shipments:** Shows a message: "Track packages sent to you." with a "+ New Package" button. Below it, a box says "No incoming packages found. Packages sent to your email will appear here."

Quick Links Legal Contact Us Home Pickup **Dashboard** Send Package Logout

Personal Information

FIRST NAME	LAST NAME
—	—
USERNAME	PHONE
loobrome1234@gmail.com	—
EMAIL (READ ONLY)	
loobrome1234@gmail.com	

Edit Details

Security & 2FA

Authenticator App
Secure your account with Google Authenticator or similar.

Setup 2FA Now (button)

DISABLED

[→ Sign Out]

Quick Links Legal Contact Us **FasterPost**

Ustawienia konta użytkownika

Proces Nadawania Paczki:

1. Formularz nadania:

Package Info

Receiver Name: Tomasz Nowak

Receiver Phone: 123123123

Receiver Email: loobrome@gmail.com

Size: Small
Max: 8 x 38 x 64 cm

Weight (kg): 120

Estimated Price:

- Base Price: \$5.00
- Size Surcharge: \$5.00
- Weight Surcharge: \$172.50

Total: \$182.50 USD

Selected Origin: YNL15Q
Destination: YNL 2

Create Package & Pay

Select Origin & Destination Postmats

A map showing the location of Rzeszów, Poland. A green checkmark is placed over the city center, indicating it as the selected origin. A red 'X' is placed over the town of Boguchwała, indicating it is not available as a destination.

2. Ostrzeżenia i Walidacja:

Selected origin changed to RZE01P due to availability.

Selected Origin: YNL15Q
Destination: YNL 2

⚠ Nearest stash used. Distance from selected origin: 140.63 km

After payment, you won't be able to edit any parcel information (recipient's email, phone number, parcel locker selection, etc.). If you're unsure about any of the details, complete the payment later in the Account tab.

Complete Payment

Karta

Numer karty: 1234 1234 1234 1234

VISA MasterCard

Data ważności: MM / RR

Kod bezpieczeństwa: 123

Kod CVC: 123

Kraj: Polska

Pay Now

Powered by Stripe • Your payment is secure

3. Płatność:

The screenshot shows the FasterPost user interface. On the left, there's a sidebar with links for 'My Account', 'Send Package', 'My Parcels', 'Incoming Parcels', and 'Settings'. The main area displays 'Active Shipments' with one item: 'Tomasz Nowak' (TRK-RVPBL1L5) with a status of 'CREATED' and a value of '\$182.50 PENDING'. A blue button '+ New Package' is at the top right. A large modal window titled 'Complete Payment' is open, prompting for card details like card number, expiration date, CVC, and country (Poland). A green 'Pay Now' button is at the bottom, with a note 'Powered by Stripe • Your payment is secure'.

4. Potwierdzenie i Umieszczenie w Skrytce:

This screenshot shows the user dashboard after a payment. The 'Active Shipments' section now shows the package as 'PLACED IN STASH' with a status of '\$182.50 SUCCESSFUL'. The payment confirmation message 'Payment successful! You can now deposit the package.' is visible. The right side of the screen shows the 'Umieszczenie w Skrytce' (Placing in Safe) page, which displays tracking details, access codes, and a map.

3. Panel Biznesowy

Dedykowany panel dla klientów biznesowych z obsługą masowych wysyłek.

Strona Biznesowa

Dashboard Biznesowy

Strona Biznesowa

Dashboard Biznesowy

FOR ENTERPRISE

Scale your business with FasterPost

Get access to bulk shipping discounts, API integration for your e-commerce store, and automated CEIDG verification. Join thousands of businesses shipping smarter.

[Open Business Account →](#)

Fast Delivery: Next-day delivery options for business partners.

Secure Handling: Safe insurance and secure locker network.

Bulk Tools: CSV upload and API access for mass shipping.

Verified: Instant NPCEIDG verification for companies.

Dashboard

Total Packages: 1

Unpaid Packages: 0

Active Magazines: 0

[Create Package](#)

[My Packages](#)

[Magazines](#)

[Payments](#)

Zarządzanie Zasobami:

Twoje Paczki

Płatności i Faktury

Package Tracking

ID	Receiver	Date	Status	Payment
12345678901234567890	John Doe	11.11.2026	Created	Pay

Pending Payments

No unpaid packages found.

Masowe Płatności

Zarządzanie Magazynami

Zapłacić użytkownikowi
22 527,00 USD

Package ID	Size / Weight	Amount
TRK-UNPHDROM	Size: medium, Weight: 1500g	22 504,00 USD
TRK-3YLUBAFU	Size: medium, Weight: 12g	22,00 USD

Płatność kartą

E-mail: email@example.com

Metoda płatności

Informacje o karcie: 1234 1234 1234 1234, MM / RR, Kod CVC

Imię i nazwisko posiadacza karty

Imię i nazwisko

Kraj lub region

Polska

Zapłać

Oświadczenie przed złożeniem: Wysyłka - Przychodź

My Magazines

Add New Magazine

Magazine Name:

Address:

Lat: 52.2297

Lng: 21.0122

Save Magazine

Operacje:

- Wysyłanie paczek:** [docs/screenshots/6_business_sending_packages.png](#)
- Zgłoszenia/Wnioski:** [docs/screenshots/6_business_request.png](#)

4. Obsługa Paczek (Wspólne)

Śledzenie Przesyłki

Odbiór Paczki

Śledzenie Przesyłki

Odbiór Paczki

The image shows two side-by-side screenshots of the FasterPost platform. The left screenshot, titled 'Tracking ID', displays a package with tracking number TRK-RVPBL1L5. It shows the origin (RZE01P) and destination (YNL 2), receiver (Tomasz Nowak), and contact information (123123123). The right screenshot, titled 'Pickup Package', shows a modal for entering details to open a locker, with fields for name, phone number, and unlock code, and a button to 'Open Locker'.

5. Panel Administratora

Zarządzanie całym systemem logistycznym.

Dashboard Admina

Logistyka

The image shows two screenshots of the Admin Panel. The left screenshot is the 'Welcome back, Admin' dashboard with links to 'Logistics Control', 'Postmats', 'Warehouses', 'All Packages', 'User Accounts', and 'Business Requests'. The right screenshot is the 'Logistics Control Tower' showing a map of Europe with route status indicators: ACTIVE ROUTES (0), PLANNED (0), COMPLETED (0), and TOTAL DISTANCE (0 km). It also includes a 'Live Network Map' showing locations across Poland.

Zarządzanie Infrastrukturą:

Paczkomaty i Skrytki

Magazyny

The image shows two screenshots of the Admin Panel. The left screenshot is 'Postmats & Stashes' showing a list of postmats with names RZE01P, YNL1SQ, and YNL1SQ, along with their statuses, locations, and actions. The right screenshot is 'Warehouse Management' showing a map of Poland with warehouse locations marked.

PICKUP CODE	FROM → TO	RECEIVER	SIZE	WEIGHT	STATUS	ACTIONS
TRK-X6V8KCA9	RZE01P → YNL 2	Tomasz Nowak 123123123	small	1g	In Transit	View Details Delete
TRK-RVPBL1L5	RZE01P → YNL 2	Tomasz Nowak 123123123	small	120g	In Transit	View Details Delete

Lista wszystkich paczek w systemie

6. Inne

How FasterPost Works

Everything you need to know about our automated delivery system.

[Home](#)
[Pickup](#)
[Dashboard](#)
[Send Package](#)
[Logout](#)

How do I send a package?

What are Postmats?

How long does delivery take?

Is my package insured?

Can I track my package?

What are the package size limits?

Sekcja Najczęściej Zadawanych Pytań

🔧 Dodatkowa Konfiguracja

Ustawienie SMTP hasła do .env pliku:

<https://myaccount.google.com/apppasswords>

Klucz Stripe

Aby otrzymać klucze stripe należy wejść pod link (z wcześniej założonym kontem):

<https://dashboard.stripe.com/>