

# Contents

|  |   |
|--|---|
| Primus Web Frontend .....                    | 1 |
| Architektura .....                           | 1 |
| Kluczowe technologie: .....                  | 2 |
| Realizacja Wymagań Specyfikacji .....        | 2 |
| 1. Definiowanie Magazynu (Regały) .....      | 2 |
| 2. Definiowanie Asortymentu (Produkty) ..... | 2 |
| 3. Wizualizacja Magazynu .....               | 2 |
| 4. Monitorowanie i Alerty .....              | 3 |
| 5. Raportowanie .....                        | 3 |
| 6. Backupy .....                             | 3 |
| 7. Zarządzanie Użytkownikami (Admin) .....   | 3 |
| 8. Panel AI (Admin) .....                    | 3 |
| 9. Profil Użytkownika .....                  | 3 |
| 10. Uwierzytelnianie .....                   | 3 |
| Struktura Projektu .....                     | 3 |
| Uruchomienie .....                           | 4 |
| Docker (Zalecane) .....                      | 4 |
| Lokalnie (Development) .....                 | 4 |

## Primus Web Frontend

Panel administracyjny systemu **Primus Inter Pares 2026** – interfejs webowy dla administratorów i pracowników magazynu. Aplikacja umożliwia zarządzanie regałami, katalogiem produktów, użytkownikami, backupami oraz monitorowanie stanu magazynu.

**Pełna dokumentacja projektu:** Primus Docs

## Architektura

Aplikacja frontendowa zbudowana jako **Single Page Application** w oparciu o React i TypeScript + Vite.

## Kluczowe technologie:

| Kategoria               | Technologia          | Zastosowanie                       |
|-------------------------|----------------------|------------------------------------|
| <b>Framework</b>        | React 18             | Biblioteka UI                      |
| <b>Język</b>            | TypeScript           | Statyczne typowanie                |
| <b>Bundler</b>          | Vite                 | Szybki serwer deweloperski i build |
| <b>Stylowanie</b>       | CSS + shadcn/ui      | Komponenty UI                      |
| <b>State Management</b> | TanStack React Query | Zarządzanie stanem serwera i cache |
| <b>Routing</b>          | React Router         | Nawigacja                          |
| <b>HTTP Client</b>      | Axios                | Komunikacja z API                  |

## Realizacja Wymagań Specyfikacji

Frontend realizuje interfejs użytkownika dla funkcjonalności wymaganych przez regulamin:

### 1. Definiowanie Magazynu (Regały)

- **Strona:** WarehouseDefinition
- **Funkcje:** Przeglądanie, dodawanie, edycja i usuwanie regałów (MxN, temperatury, waga, wymiary).
- **Import CSV:** Modal do wczytywania definicji regałów z pliku CSV (ImportRacksModal).

### 2. Definiowanie Asortymentu (Produkty)

- **Strona:** ProductDefinitions
- **Funkcje:** Pełny CRUD katalogu produktów z uploadem zdjęć (ProductFormModal).
- **Import CSV:** Masowe dodawanie produktów.

### 3. Wizualizacja Magazynu

- **Strona:** Dashboard
- **Komponenty:**
  - ▶ RackCardGrid – Interaktywna siatka regałów z wizualizacją zajętości slotów.
  - ▶ RackSlotGrid – Szczegółowy widok pojedynczego regału (MxN).
  - ▶ SlotDetailsModal – Podgląd zawartości slotu (produkt, data ważności, operacje).

## **4. Monitorowanie i Alerty**

- **Strona:** AlertsPage
- **Funkcje:** Lista alertów (przekroczenie temperatury, zbliżający się termin ważności, nieautoryzowane zdjęcie wagi), oznaczanie jako rozwiązane.

## **5. Raportowanie**

- **Strona:** Reports
- **Funkcje:** Generowanie raportów PDF (ważność, temperatura, inwentaryzacja), pobieranie wygenerowanych plików.

## **6. Backupy**

- **Strona:** Backups
- **Funkcje:** Tworzenie manualnych backupów, przeglądanie listy kopii, przywracanie stanu magazynu.

## **7. Zarządzanie Użytkownikami (Admin)**

- **Strona:** WarehouseUsers
- **Funkcje:** Lista użytkowników, zatwierdzanie/odrzucanie wniosków o rejestrację, usuwanie kont.

## **8. Panel AI (Admin)**

- **Strona:** AdminAIPage
- **Funkcje:** Zarządzanie modelem rozpoznawania obrazów (YOLO), upload zbioru danych, inicjowanie treningu, podgląd statusu.

## **9. Profil Użytkownika**

- **Strona:** Profile
- **Funkcje:** Zmiana hasła, konfiguracja 2FA (Google Authenticator).

## **10. Uwierzytelnianie**

- **Strona:** Authentication/LoginPage
- **Funkcje:** Logowanie OAuth2, obsługa 2FA (TOTP).

---

## **Struktura Projektu**

```
src/
  └── components/
    └── features/      # Komponenty specyficzne dla funkcjonalności (Dashboard,
Products...)
      └── ui/          # Reużywalne komponenty UI (Button, Card, Modal...)
      └── Navigation/ # Nawigacja i layout
    └── hooks/         # Custom React Hooks (useAuth, useRacks, useProducts...)
    └── pages/         # Strony aplikacji (routing)
    └── context/       # React Context (AuthContext)
```

```
|__ types/          # Definicje TypeScript  
└__ config/        # Konfiguracja (API URL)
```

---

## Uruchomienie

### Docker (Zalecane)

Cała infrastruktura jest definiowana w repozytorium primus-infra.

```
docker compose up -d frontend
```

Aplikacja dostępna pod: <http://localhost:5173>

### Lokalnie (Development)

Wymagane: Node.js 18+, npm

1. Instalacja zależności:

```
npm install
```

2. Uruchomienie serwera deweloperskiego:

```
npm run dev
```

*Uwaga: Upewnij się, że backend jest uruchomiony pod adresem <http://localhost:8000>.*