

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 |
|-------------------------|----------------------|------------------------------------|
| Framework | React 18 | Biblioteka UI |
| Język | TypeScript | Statyczne typowanie |
| Bundler | Vite | Szybki serwer deweloperski i build |
| Stylowanie | CSS + shadcn/ui | Komponenty UI |
| State Management | TanStack React Query | Zarządzanie stanem serwera i cache |
| Routing | React Router | Nawigacja |
| HTTP Client | 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/odrzucając wnioski 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](#).

Aplikacja dostępna pod: `http://localhost:5173`

Lokalnie (Development)

Wymagane: Node.js 18+, npm

1. Instalacja zależności:
2. Uruchomienie serwera deweloperskiego:

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