

Contents

| | |
|--|---|
| Primus Mobile | 1 |
| Architektura | 1 |
| Kluczowe technologie: | 2 |
| Realizacja Wymagań Specyfikacji | 2 |
| 3. Przyjmowanie Asortymentu (Inbound) | 2 |
| 4. Zdejmowanie Asortymentu (Outbound - FIFO) | 2 |
| 7a. Rozpoznawanie Obrazu (AI) | 2 |
| 7b. Asystent Głosowy | 2 |
| 8. Raportowanie | 3 |
| Struktura Projektu | 3 |
| Uruchomienie | 3 |
| Wymagania | 3 |
| Instalacja gotowego pakietu | 3 |
| Kompilacja z kodu źródłowego | 3 |

Primus Mobile

Aplikacja mobilna dla magazynierów systemu **Primus Inter Pares 2026**. Umożliwia przyjmowanie i wydawanie towaru, rozpoznawanie produktów za pomocą AI oraz sterowanie głosowe.

Pełna dokumentacja projektu: Primus Docs

Architektura

Natywna aplikacja mobilna zbudowana w oparciu o **React Native** i **Expo**.

Kluczowe technologie:

| Kategoria | Technologia | Zastosowanie |
|-----------------------|---------------------------|-----------------------------|
| Framework | React Native 0.81 | Natywny UI |
| Platforma | Expo 54 | Build i deployment |
| Routing | Expo Router | File-based routing |
| Stylowanie | NativeWind (Tailwind) | Responsywne style |
| State | TanStack React Query | Zarządzanie cache API |
| HTTP | Axios | Komunikacja z backendem |
| Bezpieczeństwo | Expo Secure Store | Przechowywanie tokenów JWT |
| Kamera | Expo Camera | Skanowanie kodów kreskowych |
| Głos | @react-native-voice/voice | Asystent głosowy |

Realizacja Wymagań Specyfikacji

3. Przyjmowanie Asortymentu (Inbound)

- **Ekran:** app/(protected)/actions/receive.tsx
- **Flow:** Skanowanie kodu kreskowego → Automatyczna alokacja miejsca → Potwierdzenie umieszczenia.

4. Zdejmowanie Asortymentu (Outbound - FIFO)

- **Ekran:** app/(protected)/actions/pick-up.tsx
- **Flow:** Skanowanie kodu → System wskazuje najstarszą partię (FIFO) → Potwierdzenie pobrania.

7a. Rozpoznawanie Obrazu (AI)

- **Ekran:** app/(protected)/actions/ai-recognition.tsx
- **Funkcja:** Zrobienie zdjęcia produktu → Wysłanie do backendu → Model YOLO identyfikuje produkt.
- **Feedback Loop:** Jeśli rozpoznanie jest błędne, użytkownik może poprawić – dane trafiają do zbioru treningowego.

7b. Asystent Głosowy

- **Komponent:** components/VoiceAssistant.tsx
- **Funkcja:** Rozpoznawanie mowy (pl-PL) → Wysłanie do LLM → Parsowanie intencji → Nawigacja do odpowiedniego ekranu.
- **Przykładowe komendy:** "Przyjmij mleko", "Wydaj lody", "Wygeneruj raport ważności".

8. Raportowanie

- **Ekran:** app/(protected)/actions/reports.tsx
- **Funkcja:** Inicjowanie generowania raportów PDF, pobieranie gotowych plików.

Struktura Projektu

```
primus-mobile/
├── app/                               # Expo Router - ekrany aplikacji
│   ├── (auth)/                         # Logowanie
│   ├── (protected)/                   # Ekrany wymagające autoryzacji
│   │   ├── home.tsx                  # Panel główny
│   │   └── actions/                 # Akcje magazynowe (receive, pick-up, ai, reports)
│   │   └── _layout.tsx              # Root Layout + Providers
│   ├── components/                  # Komponenty UI
│   │   ├── VoiceAssistant.tsx      # Asystent głosowy
│   │   └── feedback-modal.tsx    # Modal feedbacku AI
│   ├── services/                    # Integracja z API
│   │   ├── ai.ts                  # Rozpoznawanie produktów
│   │   ├── stock.ts               # Operacje magazynowe
│   │   └── product.ts             # Katalog produktów
│   └── context/                     # React Context (Auth)
```

Uruchomienie

Wymagania

- Node.js 18+
- Expo CLI (`npm install -g expo-cli`)
- Urządzenie mobilne z systemem android

Instalacja gotowego pakietu

Pobierz gotowy pakiet apk z najnowszego release na Githubie lub z folderu z rozwiązaniem konkursu i zainstaluj go na urządzeniu mobilnym. Następnie postępuj zgodnie z instrukcją użytkownika (Primus Docs) aby skonfigurować URL backendu.

Kompilacja z kodu źródłowego

Wymagane narzędzia: * Node.js 18+ * Expo CLI (`npm install -g expo-cli`) * Urządzenie mobilne z systemem android

Instalacja:

```
# Instalacja zależności
npm install

# Kompilacja
expo run:android # dla wersji deweloperskiej
```

```
# lub  
expo run:android --variant=release # dla wersji produkcyjnej
```

Jeśli telefon ma włączony tryb debugowania USB i jest podłączony przez USB, aplikacja powinna się uruchomić automatycznie. W przeciwnym razie, zainstaluj apk ręcznie (/android/app/build/outputs/apk/debug/app-debug.apk).