

Dokumentacja Inżynierii Wymagań z21

Dokumentacja Inżynierii Wymagań z21

Projekt: System kontroli wejść na teren zakładu pracy

Data: 21.01.2026

Autorzy: Aleksandra Siklucka, Wojciech Kowalczyk, Maciej Droździel

1. Wstęp

1.1 Macierz kompetencji zespołu

Kompetencje	Aleksandra	Wojciech	Maciej
Programowanie w Pythonie	Posiada (podstawy)	Posiada	Posiada (podstawy)
Programowanie w Java	Posiada (podstawy)	Posiada	Posiada
Programowanie w C++	Posiada (podstawy)	Posiada	Posiada
Znajomość FastAPI	Nie posiada	Posiada	Posiada (podstawy)
Znajomość React	Posiada (podstawy)	Posiada	Posiada (podstawy)
Znajomość Vite	Posiada (podstawy)	Posiada	Posiada (podstawy)
Znajomość SQL	Nie posiada	Posiada	Posiada (podstawy)
Tworzenie UML	Posiada	Nie posiada	Nie posiada
Testowanie oprogramowania	Posiada	Posiada (podstawy)	Posiada
Zarządzanie	Posiada	Posiada	Posiada (podstawy)
Rozpoznawanie obrazu	Nie posiada	Posiada (podstawy)	Posiada (podstawy)

1.2 Przeznaczenie dokumentu

Celem dokumentu jest zdefiniowanie i opisanie wymagań funkcjonalnych oraz niefunkcjonalnych systemu kontroli wejść na teren zakładu pracy z wykorzystaniem kodów QR oraz rozpoznawania twarzy. Dokument stanowi podstawę do projektowania, implementacji i testowania rozwiązania.

1.3 Zakres systemu

System obejmuje:

- Weryfikację uprawnień pracowników.
- Identyfikację osób przy użyciu skanowania QR i biometrycznego rozpoznawania twarzy.
- Rejestrację prób wejścia (czas + wynik + zdjęcie).
- Wykrywanie nadużyć (niezgodność twarzy z kodem QR).
- Zarządzanie uprawnieniami dostępu (Panel Administratora).
- Generowanie raportów.

1.4 Pytania (FAQ Projektowe)

- Jak wygląda rejestracja administratora?**

Rejestracja odbywa się poprzez wstępna konfigurację systemu (skrypt `init.sql` lub pierwszy dostęp).

- Jak administratorzy logują się do aplikacji?**

Login (email) + hasło. Hasła są bezpiecznie hashowane w bazie danych.

- Jak identyfikowani są pracownicy?**

Dwuetapowo: Skan kodu QR (identyfikacja wstępna) + automatyczna analiza zdjęcia twarzy (weryfikacja biometryczna) porównywana z wzorcem w bazie.

- Forma aplikacji administracyjnej?**

Aplikacja webowa (SPA) z responsywnym designem.

- Liczba pracowników?**

System skalowalny, wstępnie skonfigurowany na 100+ aktywnych pracowników.

- Jakie dane są rejestrowane przy wejściu?**

ID pracownika, ID bramki, data i czas, wynik weryfikacji QR, wynik weryfikacji biometrycznej, procent podobieństwa, zdjęcie z momentu weryfikacji.

- Co powinny zawierać raporty?**

- Raport wejść
- Raport nadużyć
- Raport efektywności

- Czy wymagany jest panel administratora?**

Tak - zarządzanie pracownikami, zdjęciami, QR, logami, raportami.

- Wymagana dokładność rozpoznawania?**

Min. 90%.

2. Wymagania Funkcjonalne (RF)

RF1: Zarządzanie Pracownikami

System umożliwia administratorowi:

- Dodawanie nowych pracowników (Imię, Nazwisko, Stanowisko, Email, Data zatrudnienia).
- Wgrywanie zdjęć referencyjnych dla każdego pracownika (niezbędnych do treningu modelu AI).
- Edycję i usuwanie danych pracowników.
- Generowanie i wysyłanie kodów QR na email pracownika.

RF2: Zarządzanie Bramkami

- Możliwość definiowania punktów wejściowych (Bramek) w systemie.
- Każda bramka posiada unikalną nazwę i lokalizację.

RF3: Proces Weryfikacji (Logika Bramki)

- System przyjmuje obraz z kamery i odczytany kod QR.
- Weryfikuje poprawność kodu QR.
- Weryfikuje zgodność twarzy osoby przed kamerą ze zdjęciem wzorcowym przypisanym do kodu QR.
- Rejestruje każdą próbę wejścia w bazie danych.

RF4: Raporty

Administrator ma dostęp do:

- **Rejestru wejść:** Tabela wszystkich prób wejścia (udanych i odrzuconych).
- **Raportu nadużyć:** Filtrowanie prób oznaczonych jako "Podejrzane" (np. poprawny QR, ale inną twarz).

4. Modelowanie systemu

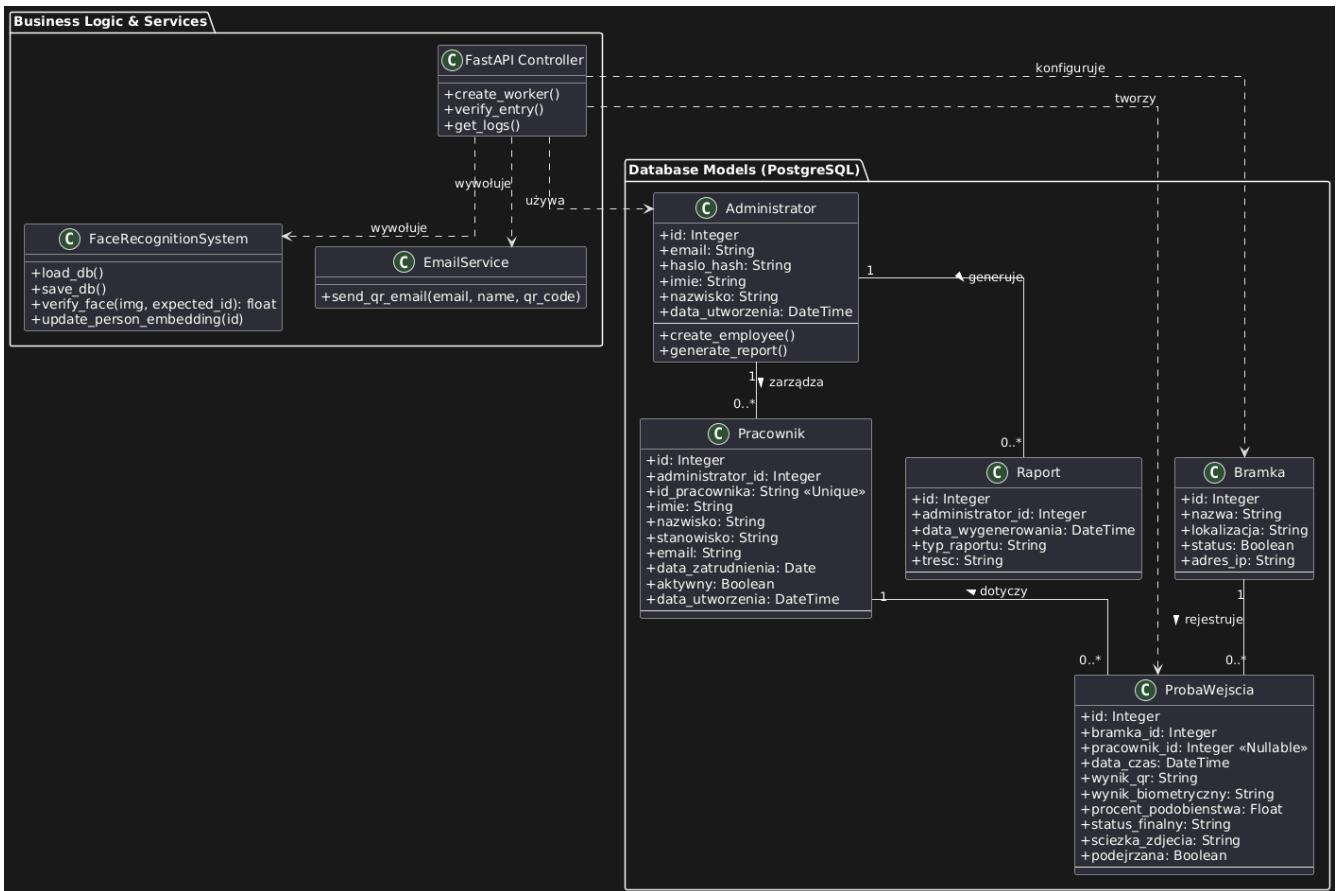
4.1 Use Case

Aktorzy: Administrator, Pracownik, Symulator Bramki (System).

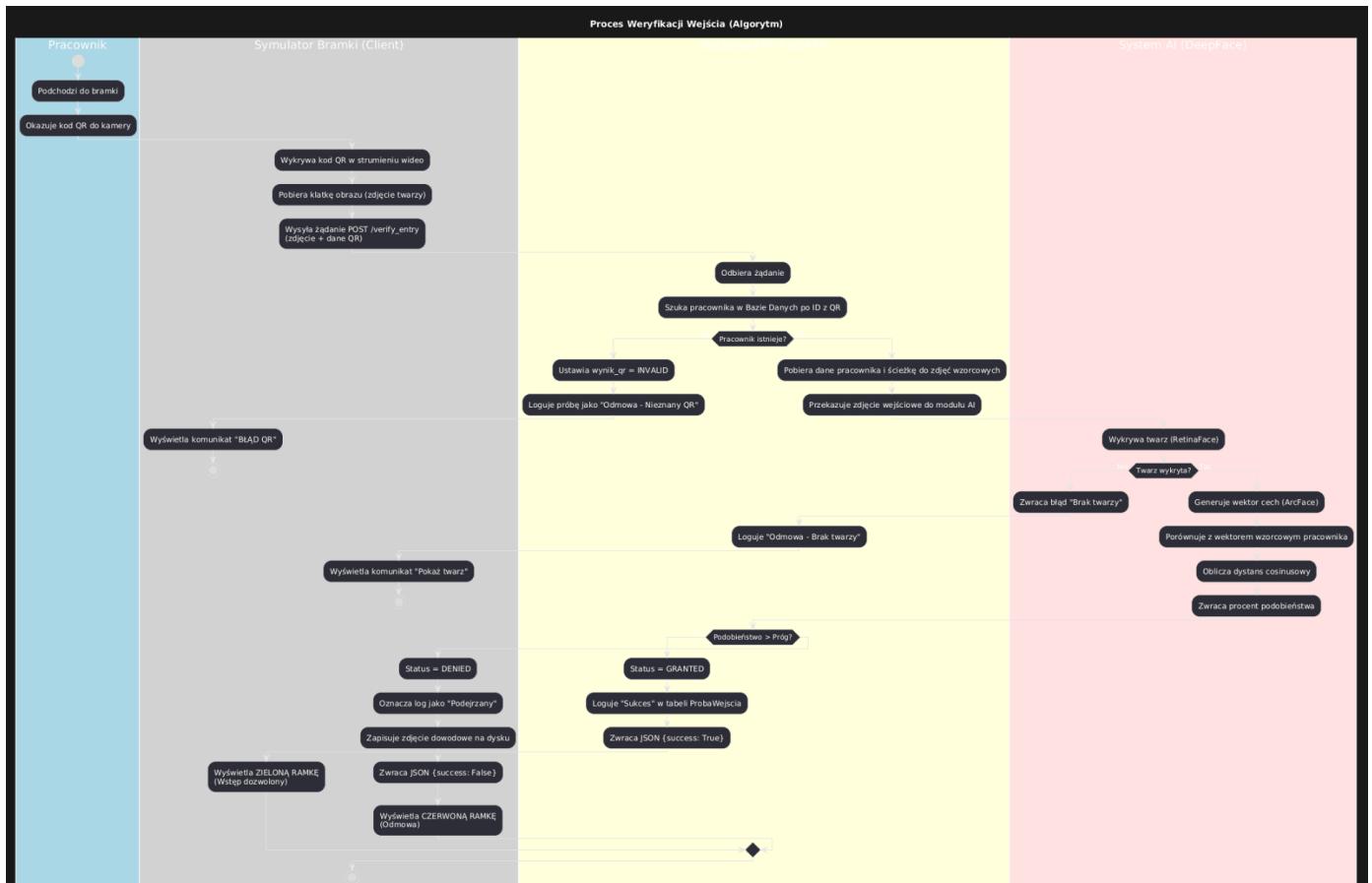
Przypadki użycia:

- **Administrator:** Logowanie, Zarządzanie pracownikami, Przeglądanie logów, Generowanie przepustek.
- **Pracownik:** Okazanie przepustki (QR) do kamery.

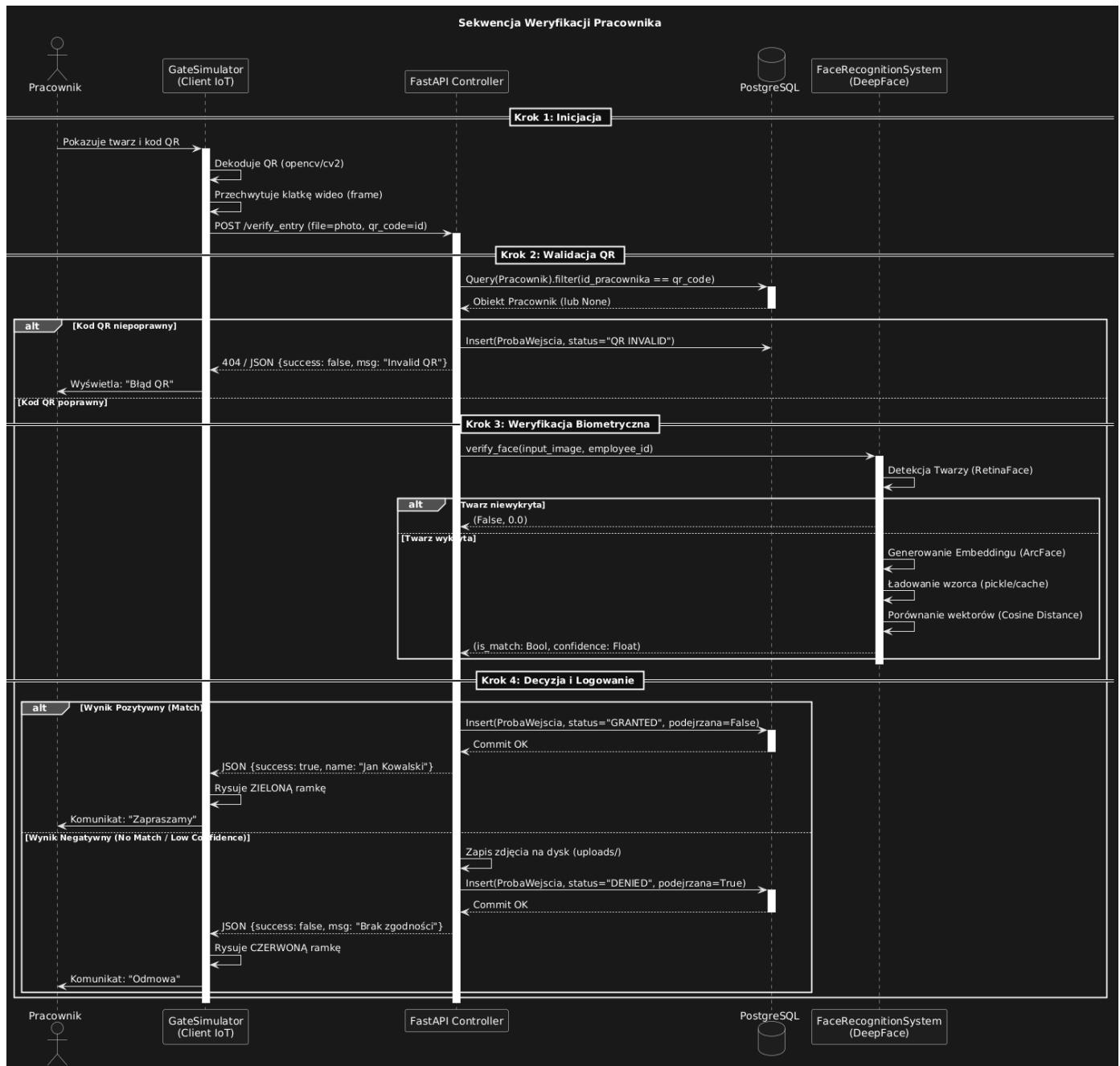
- **System:** Detekcja twarzy, Dekodowanie QR, Porównanie wektorów biometrycznych, Zapis logów.



4.2 Diagram Aktywności



4.3 Diagram Sekwencji



4.4 Struktura Danych (Klasy)

- **Administrator**

- Pola: `id`, `email`, `haslo_hash`, `imie`, `nazwisko`, `data_utworzenia`.
- Funkcja: Zarządza systemem.

- **Pracownik**

- Pola: `id`, `id_pracownika` (unikalny identyfikator/folder), `imie`, `nazwisko`, `stanowisko`, `email`, `data_zatrudnienia`, `aktywny`.
- Funkcja: Posiadacz przepustki.

- **Bramka**

- Pola: `id`, `nazwa`, `lokalizacja`, `status`, `adres_ip`.
- Funkcja: Punkt kontrolny (fizyczny lub symulowany).

- **ProbaWejscia (Logi)**
 - Pola: `id`, `data_czas`, `bramka_id`, `pracownik_id` (opcjonalne), `wynik_qr` (OK/INVALID), `wynik_biometryczny` (MATCH/NO_MATCH), `procent_podobienstwa` (Float), `status_finalny` (GRANTED/DENIED/SUSPICIOUS), `sciezka_zdjecia`.
 - Funkcja: Główny rejestr zdarzeń audytowych.
- **Raport**
 - Pola: `id`, `administrator_id`, `data_wygenerowania`, `typ_raportu`, `tresc`.

5. Użyte technologie

5.1 Frontend

- React
- Vite
- Tailwind CSS / CSS Modules
- Axios (komunikacja z API)

Uzasadnienie wyboru technologii:

- **Łatwość użycia i szybkość:** Vite zapewnia błyskawiczne budowanie aplikacji i Hot Module Replacement, co znacznie przyspiesza development w porównaniu do starszych narzędzi.
- **Kompetencje zespołu:** Zespół posiada doświadczenie w ekosystemie React, co pozwoliło na sprawne stworzenie interfejsu Panelu Administratora.

5.2 Backend

- Python 3.11+
- FastAPI (zastąpiono Flask)
- PostgreSQL (Baza danych)
- Docker & Docker Compose (Konteneryzacja)

Uzasadnienie wyboru technologii:

- **Wydajność i nowoczesność:** FastAPI jest jednym z najszybszych frameworków Pythonowych, wspiera asynchronicznosc (async/await) i automatycznie generuje dokumentację API (Swagger UI).
- **Stabilność danych:** PostgreSQL to zaawansowany system relacyjnych baz danych, zapewniający integralność danych w środowisku produkcyjnym (w przeciwieństwie do SQLite).
- **Łatwość wdrożenia:** Zastosowanie Dockera pozwala na uruchomienie całego środowiska (Baza + API + Frontend) jedną komendą, niezależnie od systemu operacyjnego hosta.

5.3 Kontrola dostępu i rozpoznawanie

- DeepFace (Wrapper dla modeli AI)
 - Model weryfikacji: ArcFace (Wysoka precyza rozpoznawania).

- Detektor twarzy: **RetinaFace** (Skuteczność w trudnych warunkach oświetleniowych).
- **OpenCV** (Przetwarzanie obrazu).
- **qrcode** (Generowanie kodów).

Uzasadnienie wyboru technologii:

- **Dostępność i jakość:** Wykorzystanie biblioteki DeepFace z modelem ArcFace pozwala na osiągnięcie skuteczności rozpoznawania twarzy na poziomie systemów komercyjnych (ponad 99% na benchmarkach LFW), przy zachowaniu licencji Open Source.
- **Prostota integracji:** Pythonowe biblioteki do wizji komputerowej łatwo integrują się z backendem API.