Raport z realizacji projektu programistycznego

System biblioteczny

Autorzy (grupa C):

Dawid Błaszczyk - nr indeksu

Błazej Kowal - nr indeksu

Alina Lenart - nr indeksu

Bartosz Wacławiak - nr indeksu

Prowadzący laboratorium:

dr inż. Krzysztof Chudzik

Data ukończenia pracy:
07.01.2025

Spis treści

1 Wymagania projektowe

1.1 Wymagania funkcjonalne

System biblioteczny IoT musi spełniać następujące wymagania funkcjonalne:

1.1.1 Obsługa kart RFID

- System musi umożliwiać odczyt kart RFID za pomocą czytnika MFRC522.
- System musi automatycznie wykrywać przyłożenie i zabranie karty RFID.
- Odczytany identyfikator karty (UID) musi być konwertowany do formatu szesnastkowego i przesyłany do serwera centralnego.
- System musi rozróżniać między kartami klientów biblioteki i kartami przypisanymi do książek.

1.1.2 Zarządzanie bazą danych

- System musi przechowywać informacje o klientach (imię, nazwisko, powiązana karta RFID).
- System musi przechowywać informacje o książkach (tytuł, autor, powiązana karta RFID).
- System musi rejestrować wypożyczenia i zwroty książek z datami operacji.
- System musi umożliwiać automatyczne tworzenie nowych rekordów kart przy pierwszym skanowaniu nieznanego UID.

1.1.3 Proces wypożyczania i zwracania

- System musi umożliwiać wypożyczenie książki poprzez zeskanowanie karty klienta, a następnie karty książki.
- System musi umożliwiać zwrot książki poprzez analogiczny proces skanowania.
- System musi wyświetlać informacje o aktywnych wypożyczeniach klienta po zeskanowaniu jego karty.
- System musi weryfikować poprawność operacji (np. czy książka jest dostępna, czy klient już ją wypożyczył).

1.1.4 Interfejs użytkownika

- System musi posiadać webowy interfejs graficzny dostępny przez przeglądarkę.
- Interfejs musi umożliwiać przeglądanie listy wszystkich klientów, książek i wypożyczeń.
- Interfejs musi umożliwiać ręczne dodawanie, edycję i usuwanie klientów oraz książek.
- Interfejs musi wyświetlać informacje o zeskanowanej karcie w czasie rzeczywistym.
- Interfejs musi umożliwiać przeprowadzenie pełnego procesu wypożyczania/zwrotu z graficznym przewodnikiem.

1.1.5 Komunikacja MQTT

- Terminale RFID (Raspberry Pi) muszą komunikować się z serwerem centralnym przez protokół MQTT.
- System musi publikować zdarzenia skanowania kart na topic `raspberry/rfid/scan`.
- Serwer musi odpowiadać z danymi o kliencie lub książce na topic `raspberry/rfid/response`.
- System musi obsługiwać sterowanie diodami LED przez MQTT (topic `raspberry/led`).

1.1.6 Informacje zwrotne dla użytkownika terminala

- System musi sygnalizować gotowość do skanowania zieloną diodą LED.
- System musi sygnalizować przetwarzanie karty czerwoną diodą LED.
- System musi emitować dźwięk buzzera po pomyślnym odczytaniu karty.
- System musi wyświetlać informacje o stanie operacji na wyświetlaczu OLED (oczekiwanie, wykryto kartę, przetwarzanie, dane klienta/książki).

1.2 Wymagania niefunkcjonalne

System musi spełniać następujące wymagania niefunkcjonalne:

1.2.1 Wydajność

- Czas odpowiedzi serwera na żądanie API nie powinien przekraczać 500ms w warunkach normalnego obciążenia.
- System musi przetwarzać zdarzenia RFID w czasie rzeczywistym (opóźnienie poniżej 1 sekundy od momentu skanowania do wyświetlenia informacji).

- Aplikacja webowa musi ładować się w czasie nie dłuższym niż 3 sekundy przy standardowym połączeniu internetowym.

1.2.2 Niezawodność i stabilność

- System musi być odporny na tymczasową utratę połączenia z brokerem MQTT i automatycznie wznawiać komunikację.
- W przypadku błędu odczytu karty RFID, system musi wyświetlić komunikat o błędzie i umożliwić ponowną próbę.
- Baza danych musi zapewniać integralność danych (brak duplikatów wypożyczeń, prawidłowe daty operacji).

1.2.3 Skalowalność

- Architektura systemu musi umożliwiać łatwe dodanie kolejnych terminali RFID bez modyfikacji kodu serwera.
- Baza danych musi być zaprojektowana w sposób umożliwiający przechowywanie tysięcy rekordów klientów i książek.

1.2.4 Bezpieczeństwo

- Komunikacja między komponentami systemu odbywa się w sieci lokalnej (brak wymogu szyfrowania w wersji proof-of-concept).
- Dane w bazie danych muszą być zabezpieczone przed nieautoryzowanym dostępem poprzez odpowiednią konfigurację uprawnień.
- System musi walidować wszystkie dane wejściowe z API, aby zapobiec nieprawidłowym operacjom.

1.2.5 Środowisko uruchomieniowe i przenośność

- Terminal RFID musi działać na platformie Raspberry Pi 4B z systemem operacyjnym Raspberry Pi OS.
- Serwer backendowy musi być uruchamialny na systemach Linux, macOS i Windows.
- Aplikacja webowa musi być kompatybilna z nowoczesnymi przeglądarkami (Chrome, Firefox, Safari, Edge).
- System musi wykorzystywać konteneryzację Docker dla brokera MQTT (eclipse-mosquitto).

- Baza danych SQLite musi być przenośna i nie wymagać dodatkowej konfiguracji serwera baz danych.

1.2.6 Technologie i standardy

- Backend: Node.js z frameworkiem NestJS, TypeScript, TypeORM.
- Frontend: React 19 z TypeScript, React Router, Tailwind CSS, Vite.
- Komunikacja IoT: Protokół MQTT z brokerem Eclipse Mosquitto.
- Baza danych: SQLite 3.
- Raspberry Pi: Python 3 z bibliotekami paho-mqtt (klient MQTT) i mfrc522 (obsługa czytnika RFID).
- Komunikacja w czasie rzeczywistym: WebSockets (Socket.IO) dla aktualizacji interfejsu użytkownika.

1.2.7 Użyteczność

- Interfejs graficzny musi być intuicyjny i nie wymagać specjalistycznego przeszkolenia.
- System musi dostarczać jasne komunikaty o stanie operacji (powodzenie, błąd, oczekiwanie).
- Wyświetlacz OLED na terminalu RFID musi prezentować czytelne informacje o aktualnym stanie systemu.
- Kolorowe diody LED muszą jednoznacznie sygnalizować stan systemu (zielony = gotowy, czerwony = przetwarzanie/błąd).

1.2.8 Dokumentacja i kod

- Kod źródłowy musi być czytelny i zgodny ze standardami danego języka programowania.
- Kluczowe funkcje systemu muszą być opatrzone komentarzami wyjaśniającymi logikę działania.
- Projekt musi zawierać pliki konfiguracyjne umożliwiające łatwe uruchomienie systemu.

TODO ciag dalszy

2 Opis architektury systemu

System zostal zaprojektowany w architekturze wielowarstwowej. Sklada się z następujących elementów:

- warstwy klienckiej,
- warstwy serwerowej,
- warstwy komunikacyjnej,
- warstwy danych.

2.1 Schemat architektury aplikacji

Poniżej przedstawiono schemat architektury systemu z uwzględnieniem architektury sieciowej.

Rysunek 1: Schemat architektury systemu

3 Opis implementacji i zastosowanych rozwiązań

3.1 Kluczowe elementy implementacji

W tej sekcji opisano najważniejsze fragmenty kodu odpowiedzialne za kluczowe funkcje systemu.

Listing 1: Przykladowy fragment kodu

```
def example_function():
    print("Przykladowa funkcja systemu")
```

3.2 Implementacja komunikacji MQTT

Opis zastosowanego mechanizmu komunikacji MQTT, struktury topiców oraz sposobu przesylania danych.

Listing 2: Fragment implementacji MQTT

```
client.connect(broker_address)
client.publish("example/topic", payload)
```

3.3 Szyfrowanie i uwierzytelnianie

Opis zastosowanych mechanizmów zabezpieczeń:

- sposób uwierzytelniania użytkowników,
- mechanizmy szyfrowania transmisji danych,
- zabezpieczenia dostępu do zasobów systemu.

3.4 Inne istotne rozwiązania

Opis dodatkowych elementów implementacyjnych uznanych przez Autorów za istotne.

4 Opis dzialania i prezentacja interfejsu

4.1 Instalacja i uruchomienie aplikacji

Opis kroków niezbędnych do uruchomienia aplikacji:

1. Pobranie kodu źródlowego.
2. Instalacja wymaganych zależności.
3. Konfiguracja środowiska.
4. Uruchomienie aplikacji.

4.2 Prezentacja interfejsu użytkownika

Rysunek 2: Przykladowy ekran aplikacji

Opis przedstawionych ekranów oraz sposobu interakcji użytkownika z systemem.

5 Opis wkladu pracy Autorów

5.1 Autor 1 – Imię Nazwisko

- Zakres wykonanych prac.
- Odpowiedzialność za konkretne moduly.
- Szacunkowy wklad procentowy: XX%.

5.2 Autor 2 – Imię Nazwisko

- Zakres wykonanych prac.
- Odpowiedzialność za konkretne moduly.
- Szacunkowy wklad procentowy: XX%.

6 Podsumowanie

Projekt zostal zrealizowany zgodnie z założonymi wymaganiami funkcjonalnymi i niefunkcjonalnymi. W trakcie implementacji napotkano na następujące trudności:

- opis problemów technicznych,
- ograniczenia czasowe lub sprzętowe.

Możliwe kierunki dalszego rozwoju systemu:

- rozbudowa funkcjonalności,
- poprawa bezpieczeństwa,
- optymalizacja wydajności.

7 Literatura

- Dokumentacja użytych technologii.
- Artykuly techniczne.
- Materialy dydaktyczne.

8 Aneks

Kod źródlowy projektu zostal dolączony w formie elektronicznej jako załącznik.