Projekt programistyczny
System przejazdów pracowniczych w komunikacji miejskiej

Podstawy internetu rzeczy
Prowadzący laboratorium: dr inż. Krzysztof Chudzik

Kacper Gaudyn, nr indeksu 266873
Kuba Krąpiec, nr indeksu 266503
Michał Pesta, nr indeksu 266899
Michał Trojanowski, nr indeksu 266864

Data ukończenia pracy: ...

Spis treści

1 Wstęp

1.1 Problem do rozwiązania

W firmie przewozowej komunikacji miejskiej postanowiono ułatwić korzystanie z systemu i połączyć pewne funkcjonalności z Internetem. Do tej pory, pojazdy informacje o kolejnych przystankach i trasach komunikowały tylko lokalnie w pojeździe, a pracownicy którzy płacili za przejazd mogli to robić tylko gotówką.

1.2 Krótki opis implementacji

Do rozwiązania problemu stworzono system który:

- umożliwia pracownikowi płatność za pomocą karty pracowniczej, którą można zasilać środkami,
- umożliwia kierowcy pojazdu zmianę przystanku na którym się znajduje, jak i trasy po której ma zamiar odbyć kurs,
- umożliwia administratorowi systemu zarządzanie danymi w bazie.

Część systemu która znajduje się w pojeździe jest nazywana terminalem.
Terminal jest obsługiwany przez kierowcę i pracownika.
Terminal posiada wyświetlacz, czytnik kart RFID, pasek LED, dwa przyciski i enkoder.
Do bazy danych przekazywane są dane z terminala (zmiana przystanku bądź trasy, płatność).

2 Wymagania projektowe

2.1 Podstawowe wymagania funkcjonalne

2.1.1 Kierowca

1. Jako kierowca chcę zmieniać przystanki za pomocą terminalu w trakcie wykonywania kursu
2. Jako kierowca chcę wybierać trasę kursu w terminalu przy rozpoczynaniu jazdy
3. Jako kierowca chcę za pomocą terminalu rozpoczynać jazdę i ją kończyć na dowolnym przystanku
4. Jako kierowca chcę, aby po dotarciu na ostatni przystanek jazda kończyła się automatycznie

2.1.2 Pracownik

5. Jako pracownik chcę wsiadać na przystanku i płacić za przejazd za pomocą swojej karty pracowniczej
6. Jako pracownik chcę, aby koszt przejazdu był zależny od przejechanych przystanków i był pobierany dopiero po zakończeniu przejazdu (przyłożeniu karty pracowniczej drugi raz)

2.1.3 Administrator

7. Jako administrator chcę zarządzać[1] wszystkimi kursami
8. Jako administrator chcę zarządzać wszystkimi przystankami
9. Jako administrator chcę zarządzać wszystkimi pracownikami, w szczególności ilością pieniędzy na ich koncie

2.2 Podstawowe wymagania niefunkcjonalne

1. System powinien obsługiwać więcej niż jednego pracownika naraz
2. System powinien obsługiwać wiele pojazdów (w tym terminalów) naraz
3. Operacje wykonywane na terminalu (płatność przez pracownika, zmiana trasy i przystanków przez kierowcę) powinny być jak najszybciej przekazywane do bazy danych

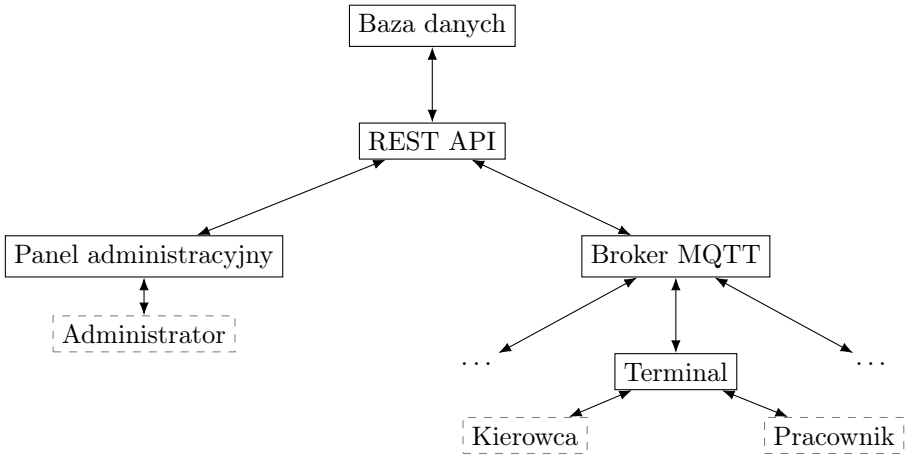
[1] Poprzez zarządzanie rozumiemy operacje: dodawania, odczytu, aktualizowania i usuwania

3 Opis architektury systemu

3.1 Elementy architektury z opisem

- Baza danych - zawiera wszystkie dane
- REST API - zarządza danymi bazy danych i udostępnia odpowiednie operacje innym podmiotom
- Panel administracyjny - służy administratorom do zarządzania danymi
- Broker MQTT - komunikuje się z terminalami w pojazdach i przekazuje informacje o przejechanych przystankach i płatnościach
- Terminal - znajduje się w każdym pojeździe, umożliwia kierowcy ustalanie trasy i zmianę przystanków, oraz umożliwia pracownikom płacenie za swoje przejazdy
- Użytkownicy
  - Kierowca - kieruje pojazdem i na terminalu może zmieniać trasy, przystanki
  - Pracownik - może wsiadać do pojazdów i płacić w terminalu za przejazd kartą
  - Administrator - zarządza danymi w systemie

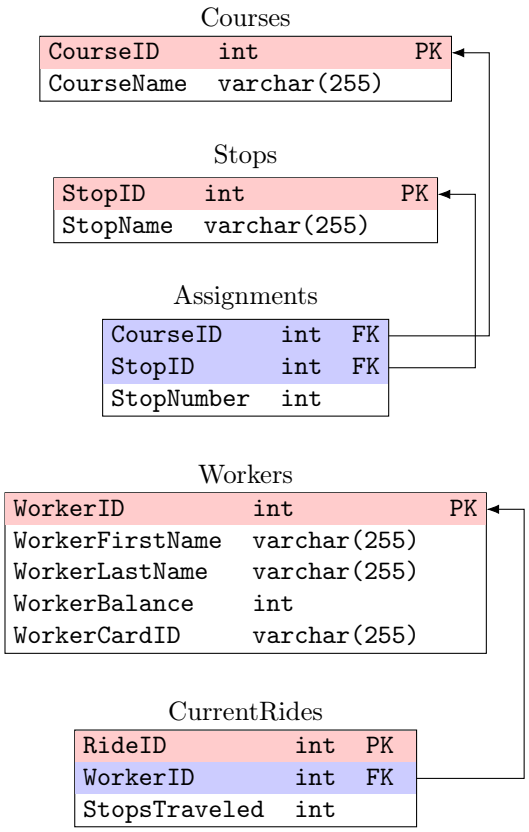
3.2 Graficzna reprezentacja architektury



Rysunek 1: Diagram elementów architektury z kierunkami przekazywania danych

3.3 Baza danych

3.3.1 Schemat bazy danych



Rysunek 2: Tabele bazy danych z zaznaczonymi relacjami między nimi

3.3.2 Scenariusze i ich wpływ na dane

1. Pracownik wsiada na przystanku, przykłada kartę i po przejechaniu 3 przystanków przykłada ją znowu aby zapłacić i wysiada.
   Zakładamy, że każdy przystanek kosztuje 10, czyli pracownik zapłaci 30.

Workers

WorkerID	WorkerFirstName	WorkerLastName	WorkerBalance	WorkerCardID
1	Jan	Kowalski	100	ABC

Rysunek 3: Dane w bazie przed wykonaniem scenariusza

Workers

WorkerID	WorkerFirstName	WorkerLastName	WorkerBalance	WorkerCardID
1	Jan	Kowalski	70	ABC

CurrentRides

RideID	WorkerID	StopsTraveled
⋮		
5	1	3
⋮		

Rysunek 4: Dane w bazie po wykonaniu scenariusza

W tabeli `Workers` zmieniła się kolumna `WorkerBalance`, a w tabeli `CurrentRides` został dodany nowy rekord.

4 Opis implementacji i zastosowanych rozwiązań

4.1 Wykorzystane języki, technologie

- Baza danych - SQLite
- REST API - Python
- Panel administracyjny - Svelte
- Broker MQTT - Python
- Terminal - Python

4.2 Najważniejsze funkcje systemu

4.3 Implementacja MQTT

4.4 Implementacja szyfrowania i uwierzytelniania