Rejestrowanie czasu pracy w sieci lokalnej przy pomocy urządzenia IoT

Bartłomiej Drozd, 127080

1. Wstęp

Projekt ma na celu budowę urządzenia monitorującego czas pracy pracowników identyfikujących się kartami MIFARE wraz z podglądem czasu pracy i panelem zarządzania kartami i pracownikami przez interfejs webowy. Urządzenie rejestrujące przy pomocy czytnika opartego o układ RC522 odczytuje UID karty, a następnie poprzez interfejs bezprzewodowy WiFi przesyła request do lokalnego serwera przez protokół SMQTT (Secure Message Queue Telemetry Transport). Aplikacja serwerowa odczytując zapytania sprawdza, czy wskazany UID jest prawidłowy (znany), rejestruje rozpoczęcie lub zakończenie trwania sesji pracy użytkownika, a następnie zwraca do urządzenia rejestrującego odpowiednią odpowiedź. Urządzenie rejestrujące wyświetla stosowny komunikat na ekranie OLED.

2. Założenia projektowe

- Urządzenie rejestrujące oparte o procesor ESP32,
- Odczyt kart MIFARE przy pomocy modułu z układem RC522,
- Wyświetlanie czasu pracy na wyświetlaczu OLED,
- Monitorowanie czasu pracy pracowników przez aplikację w sieci lokalnej,
- Interfejs webowy do przeglądania czasu pracy pracowników.

3. Wykorzystane komponenty i technologie

1. Urządzenie rejestrujące
    - Heltec 32 WiFI Kit – moduł deweloperski z procesorem ESP32 (dual core 32bit MCU + WiFi + Bluetooth) i ekranem SSD1306 OLED 0.96’ 128x64 (strona producenta),
    - Moduł RFID oparty na układzie RC522 (strona sklepu),
    - Zestaw kart MIFARE,
    - Projekt oprogramowania został skonfigurowany przy pomocy środowiska PlatformIO, kod źródłowy wykorzystuje zarówno elementy Arduino-Core dla ESP32 jak i natywny zestaw bibliotek ESP-IDF.
2. Serwer
    - Mosquitto – broker MQTT,
    - Aplikacja serwerowa napisana w Pythonie opierając się o micro framework Flask,
    - Baza danych SQLite .

4. Szczegóły urządzenia rejestrującego

Oprogramowanie wykorzystuje system czasu rzeczywistego FreeRTOS w celu wykorzystania pełnej możliwości dwurdzeniowego procesora. Poza wątkiem głównym Arduino-Core wykorzystuje drugi rdzeń procesora do obsługi sieci WiFi. Dodatkowo

uruchomione są dwa kolejne wątki do obsługi diody powiadomień oraz sprawdzania stanu połączenia WiFi.

W celu kontroli dostępu do warstwy fizycznej urządzenia, wykorzystano 3 mutexy, aby zabezpieczyć dostęp do portu szeregowego, wyświetlacza i interfejsu WiFi. Kontrola dostępu do interfejsu SPI nie jest konieczna, ponieważ odczyt kart realizowany jest tylko w wątku głównym.

Podczas uruchamiania urządzenia konfigurowany jest najpierw port szeregowy, następnie interfejs I2C do komunikacji z wyświetlaczem OLED, interfejs SPI do komunikacji z układem RC522, na koniec interfejs WiFi i protokół SMQTT. Od momentu zainicjowania wyświetlacza, zostają na nim przedstawione najważniejsze informacje z procesu uruchamiania urządzenia. Przez cały czas uruchamiania urządzenia, dioda powiadomień zmienia swój stan co 100ms.

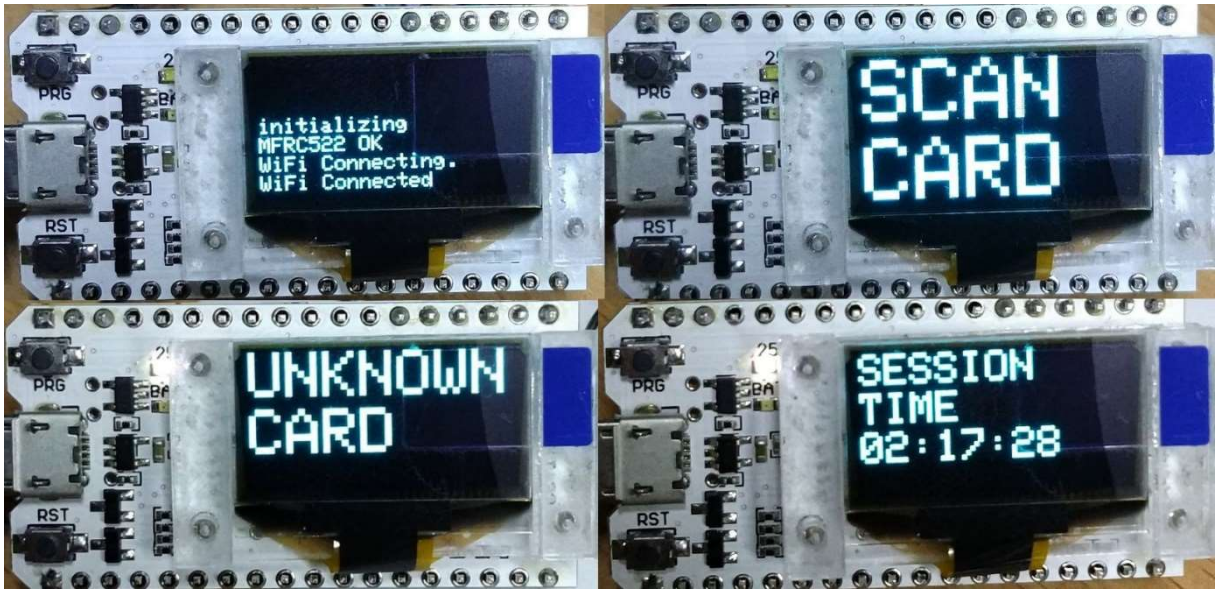
Uruchomione zostają wątki ledBlinkTask, którego zadaniem jest sterowanie diodą powiadomień oraz WiFiCheckTask sprawdzającego okresowo stan połączenia WiFi. Do komunikacji z wątkiem ledBlinkTask wykorzystywana jest jedna zmienna globalna ledDelay typu uint32. Dzięki temu zapis i odczyt jej stanu odbywa się pojedynczą operacją atomową i nie ma potrzeby dodawania kolejnych mechanizmów kontroli dostępu czy wykorzystywania złożonych struktur do komunikacji między wątkami. Ustawienie wartości ledDelay na 0 wyłącza diodę powiadomień do czasu zmiany wartości tej zmiennej. Ustawienie wartości na 1 włącza diodę do czasu kolejnej zmiany wartości. Ustawienie wartości innej niż 0 i 1 przełącza stan diody na przeciwny co wskazany czas. Wątek WiFiCheckTask sprawdza co 30 sekund stan połączenia i w przypadku rozłączenia zmienia stan zmiennej logicznej wifiReconnect na prawdę, co inicjuje mechanizm ponownego połączenia z siecią bezprzewodową. W przypadku nieudanej próby połączenia przez 15 sekund urządzenie zostanie zrestartowane automatycznie.

Parametry połączenia protokołu SMQTT wraz z certyfikatem są na stałe zapisane w kodzie programu. Po ustanowieniu połączenia urządzenie rejestrujące subskrybuje temat o wzorcu mac_address/sub, gdzie mac_address jest adresem mac urządzenia w formacie 6 bajtów w formacie hexadecymalnym oddzielonych dwukropkami. Dane odbierane są na temacie request/mac_address, gdzie mac_address jest taki sam jak w przypadku subskrybowanego tematu.

Do wyświetlania tekstu na wyświetlaczu wykorzystano autorską bibliotekę, napisaną na potrzeby prywatnych projektów, opierającą się na bibliotece od Adafruit. Umożliwia ona wypisywanie tekstu w wybranych miejscach na ekranie, wykorzystywanych np. podczas wyświetlania komunikatów dla użytkownika oraz buforowanie tekstu w trybie konsolowym, który widoczny jest podczas uruchamiania urządzenia. Przygotowano łącznie 5 ekranów z powiadomieniami dla użytkownika:

- Ekran „SCAN CARD” informujący o gotowości urządzenia i oczekujący na zeskanowanie karty,
- Ekran „OK” informujący o rozpoczęciu rejestrowania nowej sesji pracy,
- Ekran „SESSION TIME” informujący o zakończeniu rejestrowania czasu aktualnej sesji pracy i wyświetlający czas tej sesji,

- Ekran „UNKNOWN CARD” informujący o nierozpoznaniu UID zeskanowanej karty,
- Ekran wyświetlający informacje z procesu uruchamiania urządzenia, tzw. Ekran konsolowy.



Rysunek 1 Niektóre z ekranów komunikatów dla użytkownika

W pętli głównej programu sprawdzany jest stan urządzenia. Jeżeli jest w trybie oczekiwania i wyświetla na ekranie napis „SCAN CARD”, sprawdzany jest status układu RC522. Jeżeli wykrył on kartę następuje odczyt UID oraz zmienia się stan diody powiadomień na włączony. Następnie sprawdzany jest dodatkowo stan połączenia WiFi, jeżeli jest aktywne wysłana zostaje wiadomość do serwera, której zawartością jest UID w formacie ciągu znaków numeru w postaci hexadecymalnej. Jeżeli stan zmiennej wifiReconnect jest ustawiony naprawdę następuje procedura ponownego połączenia do sieci. Na koniec pętli wyświetlany jest odpowiedni komunikat dla użytkownika.

5. Szczegóły serwera

Oprogramowanie serwera oraz broker MQTT uruchomione zostały na komputerze z systemem operacyjnym Windows 10, lecz nie ma przeszkód w uruchomieniu tej samej aplikacji na innych systemach umożliwiających uruchamianie skryptów w języku Python3.

Do komunikacji między urządzeniem rejestrującym, a serwerem wykorzystano brokera MQTT Mosquitto. Dzięki wykorzystaniu protokołu MQTT możliwe jest podłączenie większej liczby rejestratorów do tego samego serwera, co pozwala na rozpoczęcie lub zakończenie sesji pracy z użyciem dowolnego urządzenia. Do zabezpieczenia połączenia utworzono przy pomocy OpenSSL certyfikat CA oraz certyfikat serwera podpisany certyfikatem CA dla brokera MQTT na potrzeby tego projektu. Ze względu na częste problemy podczas próby podłączenia urządzenia rejestrującego do brokera korzystającego z TLS w wersji 1.2, zmieniono wersję na 1.1.

Skrypt serwera wykonuje 3 główne funkcjonalności:

- Obsługa bazy danych,
- Obsługa MQTT,
- Obsługa zapytań http.

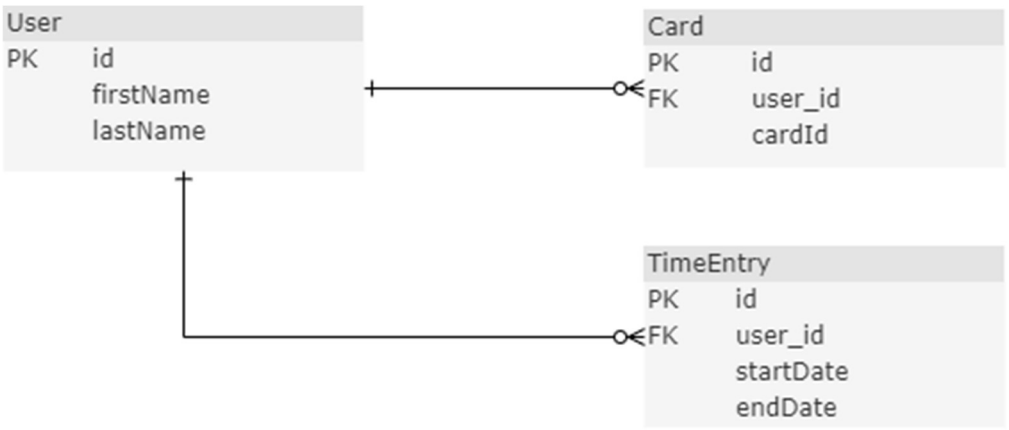
Serwer wykorzystuje microframework Flask w celu realizacji wszystkich powyższych zadań.

1. Baza danych

   Do obsługi i zarządzania bazą danych wykorzystano moduły Flask-SQLAlchemy zapewniający mapowanie obiektowo-relacyjne oraz obsługujący połączenia i zapytania do bazy danych oraz Flask-Migrate umożliwiający migracje bazy danych do kolejnych wersji w razie rozwoju aplikacji. W projekcie wykorzystano bazę SQLite3 ze względu na szybkość działania na niewielkich zbiorach danych oraz brak potrzeby instalowania dodatkowego serwera.

   Baza składa się z 3 tabel:

   - User,
   - Card,
   - TimeEntry.



Rysunek 2 Schemat relacji bazy danych

Schemat relacyjny bazy danych został przedstawiony na Rysunek 2. Zastosowany schemat bazy danych pozwala na przypisanie tej samej karty różnym użytkownikom, pod warunkiem, że zawsze będzie najwyżej jeden użytkownik tej karty w danym momencie. Tabela TimeEntry zawiera wszystkie sesje pracy pracowników. W momencie rozpoczęcia nowej sesji tworzony jest nowy rekord zawierający id użytkownika oraz datę i czas rozpoczęcia sesji. W chwili zakończenia sesji dodawana jest data i czas zakończenia. Skrypt mapujący schematy bazy danych znajduje się w pliku models.py w katalogu app. Baza danych znajduje się w pliku app.sqlite w katalogu głównym projektu serwera.

2. MQTT

   Do połączenia z brokerem MQTT wykorzystano moduł Flask-MQTT, który jest wrapperem popularnego modułu paho-mqtt dla microframeworka Flask. Parametry konfiguracji połączenia z brokerem znajdują się w pliku config.py w głównym katalogu

projektu serwera. Klient MQTT subskrybuje kanały request/# umożliwiając odbieranie wiadomości z potencjalnie wielu urządzeń rejestrujących.

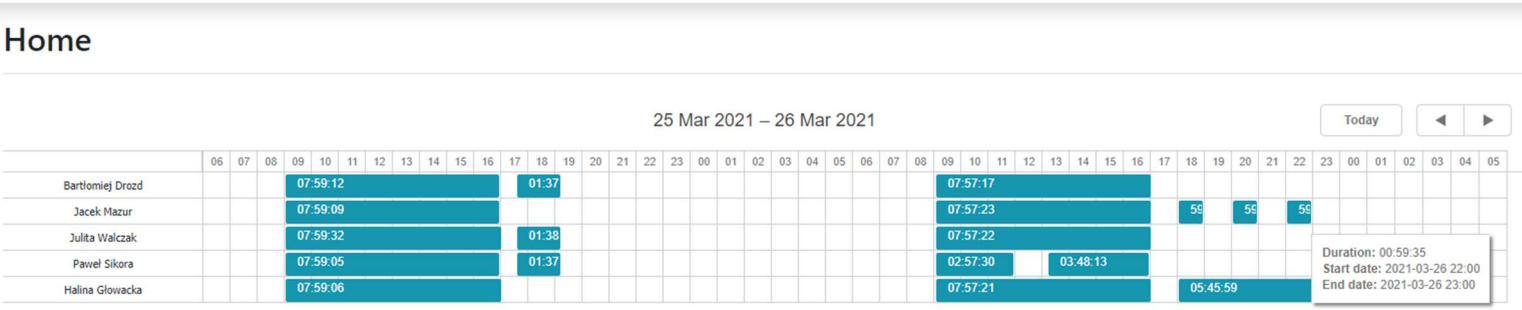
Odbieranie wiadomości MQTT zostało zrealizowane poprzez zarejestrowanie eventu na odebranie wiadomości z kanału request/# w pliku routes.py w katalogu app. Funkcja ta wysyła wiadomość zwrotną o odpowiedniej zawartości na kanał mac_address/sub. Po odebraniu wiadomości z UID zeskanowanej karty funkcja handle_mqtt_message wyszukuje w bazie danych wskazanej karty. Jeżeli karta nie została znaleziona zostaje wysłana wiadomość o nie rozpoznaniu karty. W przypadku, gdy karta zostaje rozpoznana, następuje przeszukanie tabeli TimeEntry, w celu znalezienia rekordu o user_id przypisanego użytkownika do karty oraz nieprzypisanej dacie zakończenia trwania sesji. Jeżeli taki rekord istnieje, data i godzina zakończenia są ustawiane na aktualne, obliczany jest czas trwania sesji i wysyłana jest wiadomość z komunikatem o zakończeniu trwania sesji oraz czasie jej trwania. Jeżeli nie istniał rekord ze wskazanym user_id oraz nieprzypisaną datą zakończenia tworzony jest nowy rekord z aktualną datą i godziną rozpoczęcia, a następnie wysyłana jest wiadomość zwrotna z informacją o rozpoczęciu nowej sesji pracy.

3. Http

Serwer http opiera się na głównym module Flask. Składa się łącznie z 7 stron:

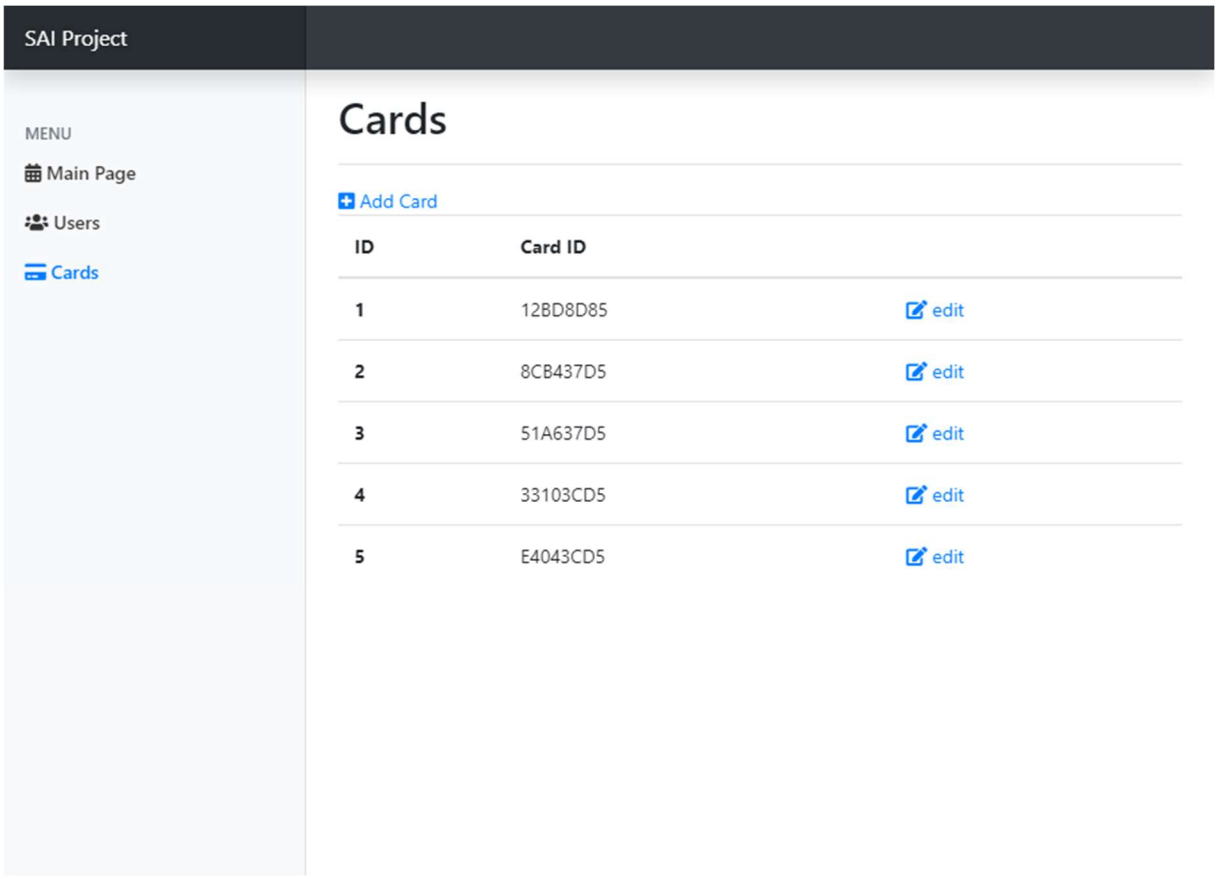
- Strona główna,
- Widok użytkowników,
- Edycja użytkownika,
- Dodawanie użytkownika,
- Widok kart,
- Edycja karty,
- Dodawanie karty.

Strona główna wyświetla harmonogram zarejestrowanych sesji. Widok prezentuje przedział czasu 24 godzin. Poruszanie się przyciskami ze strzałkami w prawym górnym rogu strony przesuwa oś czasu o 6 godzin do przodu lub do tyłu. Na paskach symbolizujących czas trwania sesji wyświetlony jest czas trwania. Po najechaniu myszką na pasek dodatkowo wyświetlają się szczegółowe dane o dacie i godzinie rozpoczęcia i zakończenia sesji. Do prezentacji danych wykorzystana zastała płatna (na potrzeby projektu zarejestrowano wersję próbną) biblioteka DHTMLX Scheduler. Poza prezentacją sesji pracy umożliwia ona dynamiczne ładowanie danych. System został skonfigurowany tak, aby jednorazowo ściągać z serwera zbiór danych z okresu tygodnia. Zapytania o dane kierowane są przez zapytanie typu POST na adres /scheduler, który zwraca odpowiedź typu application/json z odpowiednim plikiem json generowanym na podstawie parametrów przedziału czasu przekazanych w zapytaniu.

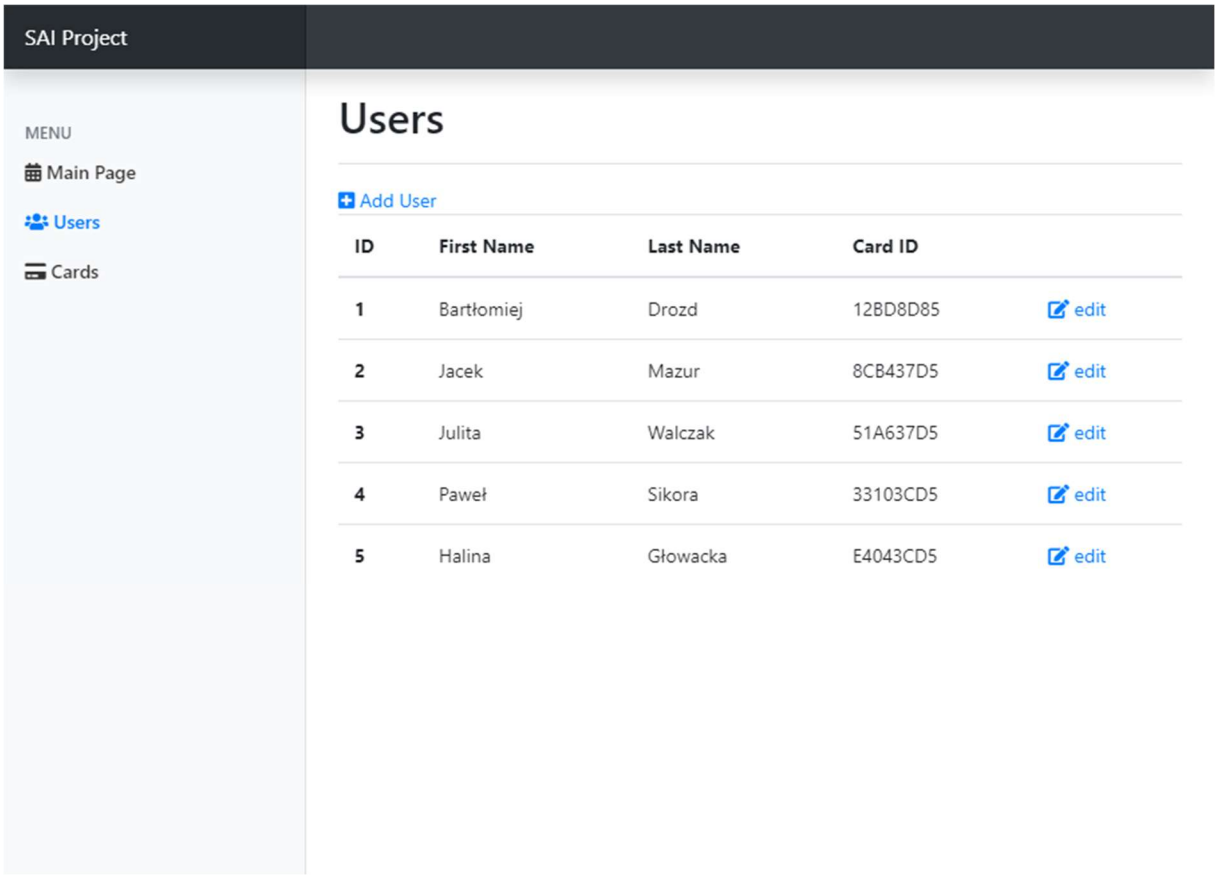


Rysunek 3 Prezentacja sesji pracy pracowników

Widok użytkowników i widok kart przedstawiają proste tabele z odpowiadającymi danymi zawartymi w bazie danych oraz umożliwiają przejście do edycji rekordu lub dodanie nowego. Podczas dodawania lub edycji użytkownika możliwe jest przypisanie mu karty z listy kart bez przypisanego właściciela. Pobieranie odpowiednich danych z bazy danych oraz generowanie widoków odbywa się w pliku routes.py w katalogu app. Szablony widoków znajdują się w folderze app/templates.



Rysunek 4 Widok panelu podglądu kart



Rysunek 5 Widok podglądu pracowników

6. Podsumowanie

Projekt został zrealizowany zgodnie z założeniami. Przygotowano urządzenie rejestrujące oparte na procesorze ESP32 odczytujące karty MIFARE przy pomocy czytnika zrealizowanego na układzie RC522 oraz wyświetlające stosowne komunikaty i informacje użytkownikowi na ekranie OLED. Przechowywanie sesji pracy oraz podgląd i zarządzanie pracownikami i kartami zostały udostępnione przez interfejs webowy na serwerze w sieci lokalnej. Komunikacja między urządzeniem rejestrującym i serwerem zostały zrealizowane poprzez protokół SMQTT.

Projekt ma miejsce na dalszą rozbudowę i optymalizacje. Możliwe jest dodanie systemu logowania do interfejsu webowego, walidacja wprowadzanych danych w formularzach, zmiana biblioteki prezentującej historię sesji pracy na otwartoźródłową w celu uniknięcia opłat licencyjnych, logowanie do brokera MQTT przy pomocy indywidualnych certyfikatów w celu zwiększenia bezpieczeństwa transmisji danych, zapis certyfikatów na kartach MIFARE. Dzięki przemyślanemu sposobowi komunikacji możliwe jest podłączenie większej ilości urządzeń rejestrujących.