# SPIS TREŚCI



## 1. Opis działania

Aplikacja umożliwia użytkownikom sterowanie roletami okiennymi za pomocą interfejsu webowego. Użytkownik ma do dyspozycji trzy przyciski:

- **Podnieś rolety** – zwiększa stopień podniesienia rolet o 5%.
- **Zatrzymaj** – zatrzymuje bieżącą operację i zapisuje ostatnią odczytaną wartość do bazy.
- **Opuść rolety** – zmniejsza stopień podniesienia rolet o 5%.

Dane o stanie rolet oraz konta użytkowników są przechowywane w Azure Cosmos DB, co zapewnia synchronizację między instancjami aplikacji.

Aplikacja działa w kontenerach Docker, a użytkownicy mogą sterować roletami z dowolnego miejsca dzięki synchronizacji danych w chmurze Azure.

## 2. Wymagania systemowe

Przed rozpoczęciem instalacji upewnij się, że masz zainstalowane:

- **Docker** – do uruchamiania kontenerów.
- **Azure CLI** – do zarządzania zasobami w chmurze.
- **Node.js i npm** – do uruchomienia aplikacji webowej.
- **Konto na platformie Azure** – wymagane do korzystania z Cosmos DB.

# 3. Instalacja i uruchomienie

### *3.1. Konfiguracja środowiska Azure*

Zaloguj się do swojego konta na platformie Azure:

```
az login
```

Skonfiguruj Azure Cosmos DB:

```
az cosmosdb create --name <nazwa_bazy> --resource-group <nazwa
resource_grupy> --enable-free-tier true --locations regionName <nazwa
regionu>
```

Zapisz dane dostępowe do bazy, które będą używane w aplikacji.

```
  az cosmosdb keys list --name <nazwa bazy> --resource-group <nazwa
resource grupy> --type connection-strings --query
"connectionStrings[0].connectionString" --output tsv
```

Skonfiguruj Azure IoT Hub

```
 az iot hub create --name <nazwa_iot_huba> --resource-group
<nazwa_grupy_zasobow> --sku F1
```

## 3.2. Uruchomienie aplikacji lokalnie

Pobierz repozytorium i przejdź do jego katalogu:

```
git clone <url_repozytorium>
cd <nazwa_folderu>
```

Zainstaluj zależności:

```
npm install
```

Skonfiguruj połączenie z Azure Cosmos DB: Zaktualizuj plik `config.json`, dodając dane dostępowe do bazy:

```
{
  "cosmosDbConnectionString": "<connection_string>"
  "iotHubConnectionString": "<iot_hub_connection_string>"
}
```

Uruchom aplikację lokalnie:

```
npm start
```

Aplikacja będzie dostępna pod adresem http://localhost:3000.

## 3.3. Uruchomienie aplikacji w kontenerze Docker

Zbuduj obraz Docker:

```
docker build -t roller-control-app .
```

Uruchom kontener:

```
docker run -p 8080:3000 roller-control-app
```

Aplikacja będzie dostępna pod adresem http://localhost:8080.

## 4. Struktura danych

Dane o stanie rolet są przechowywane w Azure Cosmos DB w następującym formacie:

```
{
  "id": "rolleta_001",
  "window_id": "window_01",
  "state": 50, // Wartość procentowa podniesienia rolet
  "last_updated": "2025-03-15T12:00:00Z"
}
```

- **id** – Unikalny identyfikator rolety.
- **window_id** – Identyfikator okna, do którego przypisana jest roleta.
- **state** – Aktualny stan podniesienia rolety (0-100%).
- **last_updated** – Znacznik czasu ostatniej aktualizacji stanu.

Dane o użytkownikach są przechowywane w tabeli `users` w Azure Cosmos DB:

```
{
  "id": "user_001",
  "username": "jan_kowalski",
  "password_hash": "hashed_password",
  "email": jan.kowalski@example.com,
  "token": jwt_token, // generowany po zalogownaiu (po wylogowaniu
zostaje usunięty i przegenerowany na nowo)
  "token_exp": "2025-04-15T12:00:00Z", // wazny przez 30 dni
  "isActive": true // okresla, czy uzytkownik jest zalogowaniu
}
```

- **id** – Unikalny identyfikator użytkownika.
- **username** – Nazwa użytkownika.
- **password_hash** – Zahaszowane hasło użytkownika.
- **email** – Adres e-mail użytkownika.

## 5. Logika sterowania

### 5.1. Podnoszenie rolet

1. Użytkownik naciska przycisk "Podnieś rolety".
2. Aplikacja sprawdza aktualny stan rolety.
3. Jeśli stan < 100%, zwiększa wartość o 5%.
4. Aktualizuje stan w bazie Cosmos DB.
5. Wysyła komunikat do IoT Hub z poleceniem podniesienia rolet (MQTT).

### 5.2. Opuszczanie rolet

1. Użytkownik naciska przycisk "Opuść rolety".
2. Aplikacja sprawdza aktualny stan rolety.
3. Jeśli stan > 0%, zmniejsza wartość o 5%.
4. Aktualizuje stan w bazie Cosmos DB.
5. Wysyła komunikat do IoT Hub z poleceniem opuszczenia rolet (MQTT).

### 5.3. Zatrzymanie operacji

1. Użytkownik naciska przycisk "Zatrzymaj".
2. Aplikacja zatrzymuje bieżącą operację podnoszenia/opuszczania rolet.
3. Ostatnia odczytana wartość podniesienia rolet zostaje zapisana do bazy Cosmos DB.
4. Wysyła komunikat do IoT Hub z poleceniem zatrzymania rolet (MQTT).

## 6. Rejestracja użytkownika

1. Użytkownik wypełnia formularz rejestracyjny podając:
    a. **Nazwa użytkownika**
    b. **Hasło** (przechowywane w postaci hasha)
    c. **Adres e-mail**
2. Aplikacja zapisuje dane do tabeli `users` w Cosmos DB.
3. Po pomyślnej rejestracji użytkownik otrzymuje powiadomienie „Success” i może zalogować się do aplikacji.

## 7. Integracja z Azure

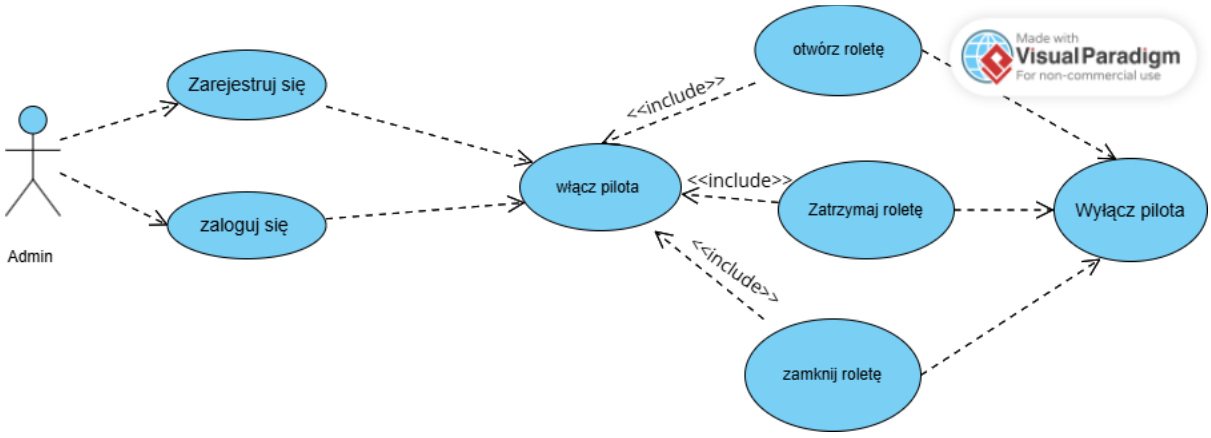
- **Azure Cosmos DB** – Przechowuje aktualny stan rolet i dane użytkowników.

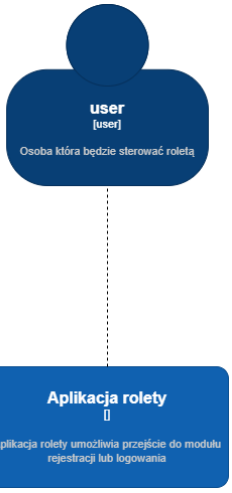
- **Docker** – Aplikacja działa w kontenerze, co zapewnia skalowalność.
- **Azure IoT Hub** – Służy jako broker MQTTs do komunikacji między aplikacją a urządzeniami sterującymi roletami.

## 8. Bezpieczeństwo i uwierzytelnianie

- Użytkownicy logują się za pomocą indywidualnych kont aplikacji przechowywanych w Cosmos DB.
- API aplikacji wymaga tokenów JWT do komunikacji.
- Hasła użytkowników są przechowywane w formie zahaszowanej.
- Wszystkie operacje są rejestrowane w systemie logów dla audytu.
- Komunikacja z IoT Hub odbywa się w sposób bezpieczny przy użyciu MQTTs i SSL/TLS.

## 9. USE CASE DIAGRAM






## 10. DIAGRAMY L1 i L2