Wyższa Szkoła Przedsiębiorczości i Administracji w Lublinie

Dokumentacja projektu: API zarządzające pływalnią

Rafał Skowroński

Michał Łuczyński

Informatyka PUW

Semestr: 5

Rok akademicki: 24/25

Lublin 2025

Cel projektu

Celem aplikacji jest stworzenie zaawansowanego API, które umożliwi zarządzanie ruchem na pływalni oraz dostarczy administratorom i użytkownikom szczegółowych informacji o obłożeniu oraz historii użytkowania obiektu. System będzie także obsługiwał różne lokalizacje pływalni.

Wymagania

Wymagania funkcjonalne

1. API umożliwi rejestrację wejścia osoby na pływalnię.
2. API umożliwi rejestrację wyjścia osoby z pływalni.
3. API będzie przechowywało dane o maksymalnej liczbie osób dozwolonych na pływalni.
4. API udostępni endpoint do sprawdzania liczby obecnych osób na pływalni.
5. API zwróci informacje, czy można wpuścić kolejną osobę na pływalnię.
6. API umożliwi zarządzanie wieloma lokalizacjami pływalni.
7. API udostępni endpoint do generowania raportów z historii ruchu na pływalni.
8. API obsłuży autoryzację i autentykację użytkowników, w tym administratorów.

Wymagania niefunkcjonalne

1. Dokumentacja punktów końcowych przy pomocy Swaggera.
2. Logowanie operacji użytkownika przy użyciu Log4j.
3. Obsługa dwóch profilów środowiska: `dev` i `prod`.
4. Skalowalność - API musi obsługiwać równoczesne żądania od wielu użytkowników.

Aktorzy

1. **Administrator**: Odpowiedzialny za konfigurację systemu, w tym ustalanie maksymalnej liczby osób na pływalni, zarządzanie lokalizacjami oraz generowanie raportów.
2. **Użytkownik API**: Może sprawdzać obecną liczbę osób oraz rejestrować wejścia i wyjścia osób.
3. **System zewnętrzny**: Może integrować się z API w celu monitorowania stanu obłożenia pływalni.

Przypadki użycia

Rejestracja wejścia osoby na pływalnię

- **Aktor**: Użytkownik API
- **Opis**: Rejestruje wejście osoby, o ile liczba osób na pływalni nie przekroczyła limitu.
- **Rezultat**: Zwrócenie informacji o sukcesie lub odmowie wraz z kodem HTTP.

Rejestracja wyjścia osoby z pływalni

- **Aktor**: Użytkownik API
- **Opis**: Rejestruje wyjście osoby i zmniejsza licznik obecnych osób.
- **Rezultat**: Zwrócenie informacji o sukcesie wraz z kodem HTTP.

Sprawdzanie liczby obecnych osób

- **Aktor**: Użytkownik API
- **Opis**: Umożliwia sprawdzenie obecnej liczby osób na pływalni.
- **Rezultat**: Zwrócenie liczby osób wraz z kodem HTTP.

Sprawdzanie, czy można wpuścić kolejną osobę

- **Aktor**: Użytkownik API
- **Opis**: Weryfikuje, czy liczba obecnych osób nie przekracza maksymalnego limitu.
- **Rezultat**: Zwrócenie informacji (true/false) wraz z kodem HTTP.

Zarządzanie lokalizacjami pływalni

- **Aktor**: Administrator
- **Opis**: Dodawanie, usuwanie i aktualizacja danych lokalizacji pływalni.
- **Rezultat**: Zwrócenie informacji o sukcesie wraz z kodem HTTP.

Generowanie raportów

- **Aktor**: Administrator
- **Opis**: Generowanie raportów z historią ruchu na pływalni, zawierających dane o godzinach wejść i wyjść.
- **Rezultat**: Plik raportu w formacie CSV.

Rejestracja użytkownika

- **Aktor**: Nowy użytkownik API
- **Opis**: Umożliwia rejestrację nowego użytkownika w systemie.
- **Rezultat**: Zwrócenie obiektu `User` w przypadku sukcesu, kod HTTP 403 w przypadku błędu.

Logowanie użytkownika

- **Aktor**: Użytkownik API
- **Opis**: Umożliwia logowanie do systemu i otrzymanie tokenu JWT.
- **Rezultat**: Zwrócenie tokenu JWT w przypadku sukcesu, kod HTTP 403 w przypadku błędu.

Walidacja tokenu JWT

- **Aktor**: Użytkownik API
- **Opis**: Weryfikuje poprawność tokenu JWT i zwraca dane użytkownika.
- **Rezultat**: Zwrócenie obiektu `User` w przypadku sukcesu, kod HTTP 403 w przypadku błędu.

Kluczowe obiekty

1. **Pool**
    - **Atrybuty**:
        - `id`: Unikalny identyfikator.
        - `maxCapacity`: Maksymalna liczba osób.
        - `location`: Adres lokalizacji pływalni.
    - **Opis**: Reprezentuje pływalnię z jej podstawowymi parametrami.
2. **Log**
    - **Atrybuty**:
        - `id`: Unikalny identyfikator.
        - `timestamp`: Czas operacji.
        - `type`: Typ operacji (wejście/wyjście).
        - `poolId`: Identyfikator pływalni.
    - **Opis**: Rejestruje zdarzenie związane z ruchem osób na pływalni.
3. **User**
    - **Atrybuty**:
        - `id`: Unikalny identyfikator użytkownika.
        - `username`: Nazwa użytkownika.
        - `password`: Zaszyfrowane hasło.
        - `role`: Rola użytkownika (administrator/użytkownik).
    - **Opis**: Przechowuje informacje o użytkownikach systemu.

Struktura bazy danych (MongoDB)

1. **Collection: Pools**

Dokument:

```
 {
  "id": "1",
  "maxCapacity": 50,
  "location": "Warszawa, ul. Basenowa 1"
}
```

2. **Collection: Logs**

Dokument:

```
{
  “id”: “1”,
  "timestamp": "2025-01-01T10:00:00Z",
  "type": "entry",
  "poolId": "1"
}
```

3. **Collection: Users**

Dokument:

```
{
  "id": "admin123",
  "username": "admin",
  "password": "hashed_password",
  "role": "ROLE_ADMIN",
}
```

Struktura bazy danych (Redis)

1. **Klucz POOL_ID:occupancy -> Integer:** Przechowuje obecną liczbę osób na pływalni