# Dokumentacja projektu: DataWeaver

## 1. Główny cel projektu

Celem projektu jest stworzenie aplikacji webowej umożliwiającej zarządzanie listą użytkowników w oparciu o bazę danych MySQL. Projekt zapewnia możliwość dodawania, edytowania, usuwania oraz przeglądania użytkowników za pomocą interfejsu webowego oraz REST API. Aplikacja korzysta z frameworka Spring Boot, co ułatwia integrację z bazą danych, budowanie API oraz obsługę widoków opartych na Thymeleaf.

## 2. Założenia projektu

- Stworzenie aplikacji webowej w technologii Spring Boot.
- Przechowywanie danych o użytkownikach w bazie danych MySQL.
- Zapewnienie dwóch interfejsów użytkownika:
    - Interfejs webowy zbudowany przy użyciu Thymeleaf (dla operacji CRUD).
    - REST API umożliwiające integrację z zewnętrznymi systemami.
- Obsługa operacji na użytkownikach, takich jak:
    - Dodawanie nowego użytkownika.
    - Edytowanie istniejącego użytkownika.
    - Usuwanie użytkownika.
    - Wyświetlanie listy użytkowników.
- Pobieranie danych o użytkownikach z zewnętrznego serwisu API (https://jsonplaceholder.typicode.com).

## 3. Struktura projektu

### 3.1 Pakiety

- `com.example.demo.config`: Zawiera konfigurację aplikacji, w tym klasę `RestTemplateConfig`, umożliwiającą korzystanie z `RestTemplate` do komunikacji zewnętrznej.
- `com.example.demo.controller`: Zawiera kontrolery obsługujące żądania HTTP (zarówno dla REST API, jak i interfejsu webowego).
- `com.example.demo.model`: Zawiera model danych (`User`), reprezentujący użytkownika w bazie danych.
- `com.example.demo.repository`: Zawiera repozytorium JPA (`UserRepository`) do zarządzania operacjami na bazie danych.

### 3.2 Główne klasy

- **`DemoApplication`**: Klasa uruchamiająca aplikację Spring Boot.

- **`User`**: Klasa modelu reprezentująca użytkownika (z polami `id`, `name`, `username`, `email`).
- **`UserRepository`**: Interfejs repozytorium JPA umożliwiający wykonywanie operacji na tabeli `User`.
- **`UserController`**: Kontroler obsługujący widoki webowe oraz interakcje użytkownika z przeglądarką.
- **`UserApiController`**: Kontroler udostępniający REST API do zarządzania użytkownikami.

---

# 4. REST API

## 4.1 Endpoints

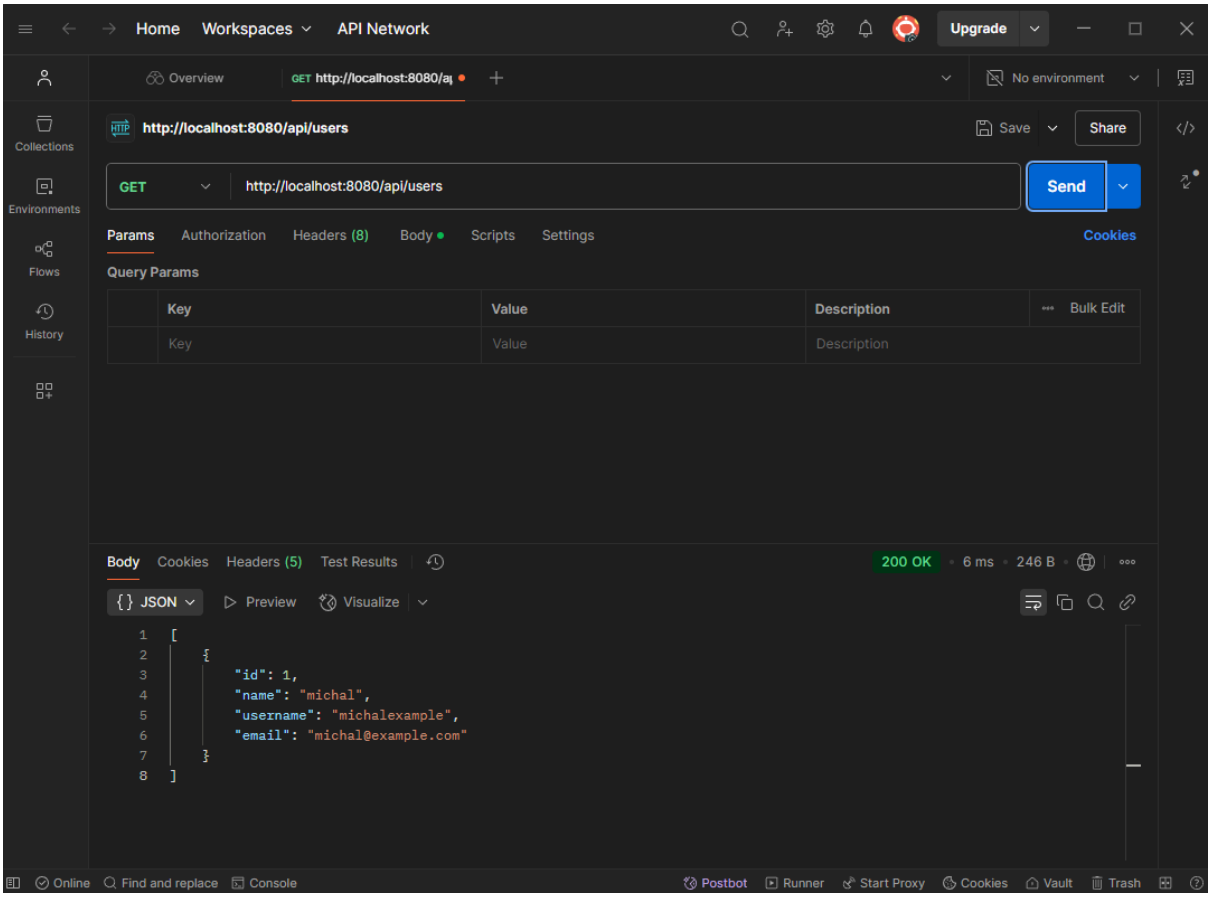
### 1. GET /api/users

- **Opis**: Zwraca listę wszystkich użytkowników zapisanych w bazie danych.

  Do testowania API korzystam z platformy Postman.

**Przykładowa odpowiedź**:

```
[
 {
  "id": 1,
  "name": "Jan Kowalski",
  "username": "jankowalski",
  "email": "jan.kowalski@example.com"
 },
```




Listę użytkowników możemy także zobaczyć z poziomu przeglądarki pod adresem: http://localhost:8080/api/users

### 2. POST /api/users

- **Opis**: Dodaje nowego użytkownika do bazy danych.

  Do testowania API korzystam z platformy Postman.

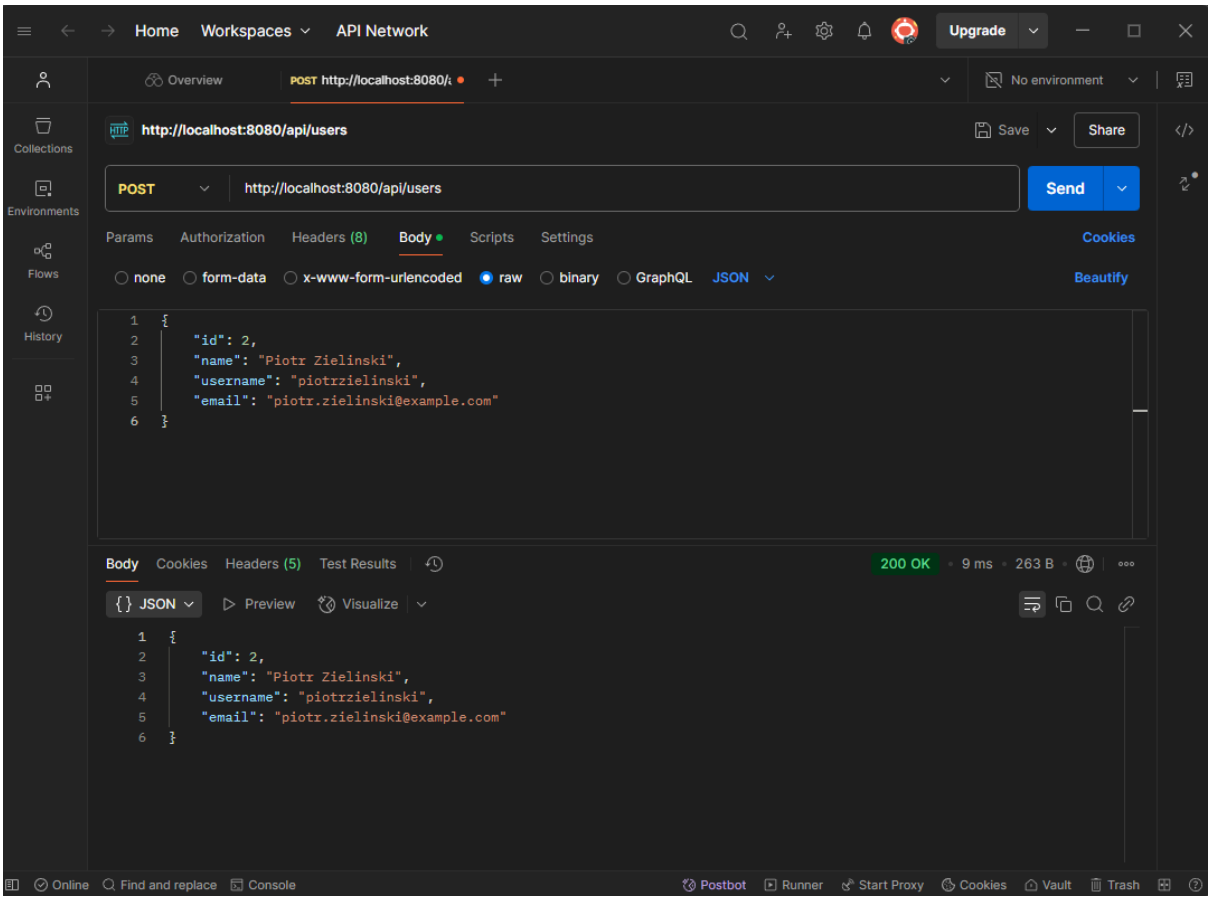
**Parametry**: Dane użytkownika przesyłane w formacie JSON.

**Przykładowa odpowiedź**:

```
 {
  "id": 2,
  "name": "Piotr Zielinski",
  "username": "piotrzielinski",
  "email": "piotr.zielinski@example.com"
}
```

Zastosuj formatowanie stylistyczne


```
[
  {
    "id": 1,
    "name": "michal",
    "username": "michalexample",
    "email": "michal@example.com"
  },
  {
    "id": 2,
    "name": "Piotr Zielinski",
    "username": "piotrzielinski",
    "email": "piotr.zielinski@example.com"
  }
]
```

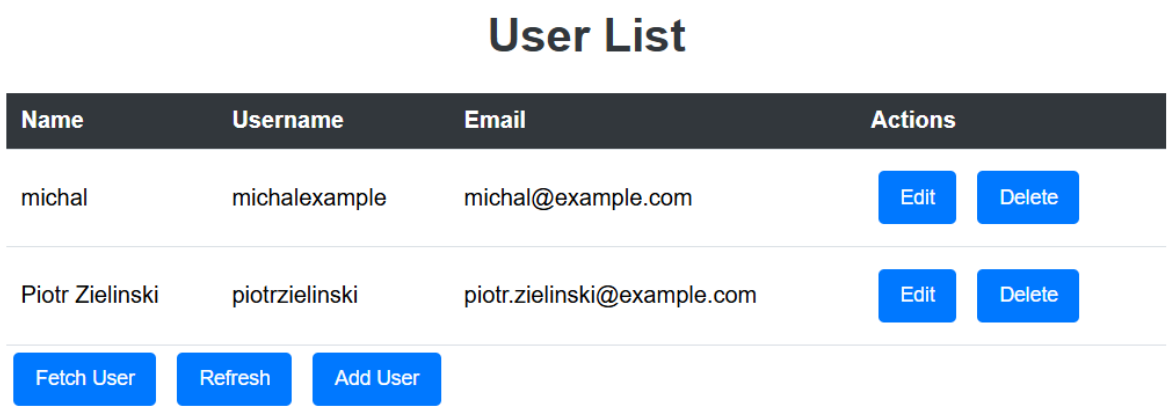




Listę użytkowników możemy także zobaczyć z poziomu przeglądarki pod adresem: http://localhost:8080/api/users

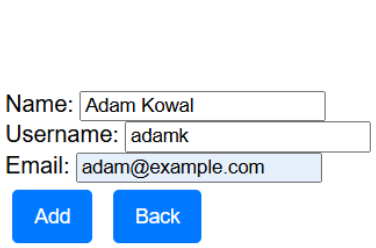
# 5. Interfejs webowy

Aplikacja oferuje widoki webowe zbudowane przy użyciu Thymeleaf:

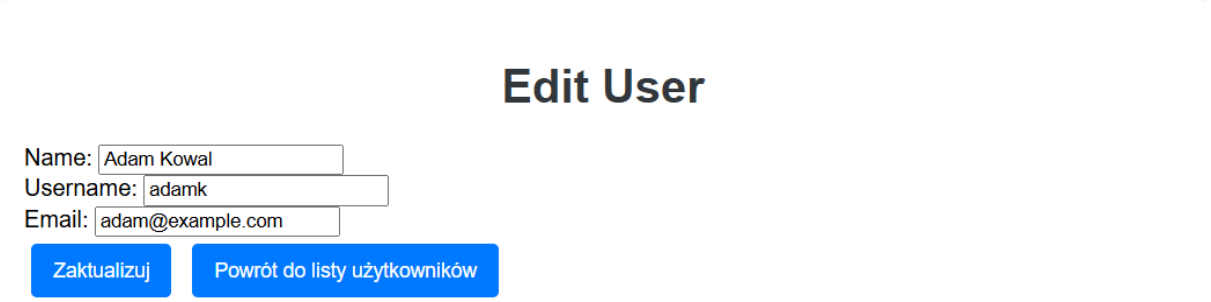
1. **Lista użytkowników**:
    - Adres: `/users`
    - Wyświetla wszystkich użytkowników zapisanych w bazie danych.



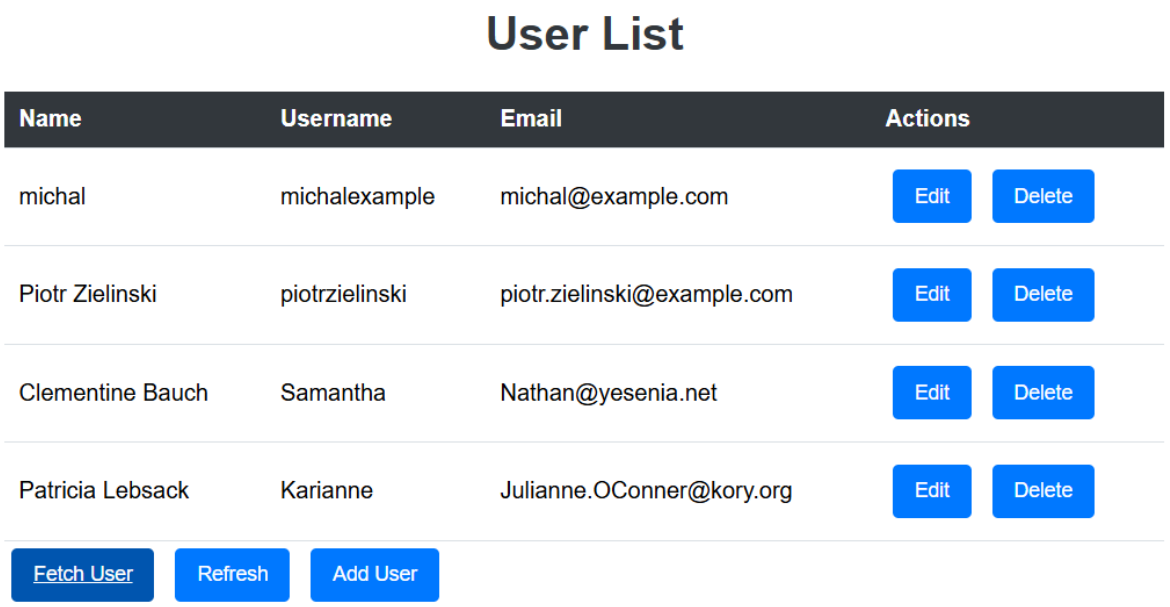

2. **Dodawanie użytkownika**:
    - Adres: `/add-user`
    - Formularz umożliwiający dodanie nowego użytkownika.



3. **Edycja użytkownika**:
    - Adres: `/users/{id}/edit`
    - Formularz edycji istniejącego użytkownika.



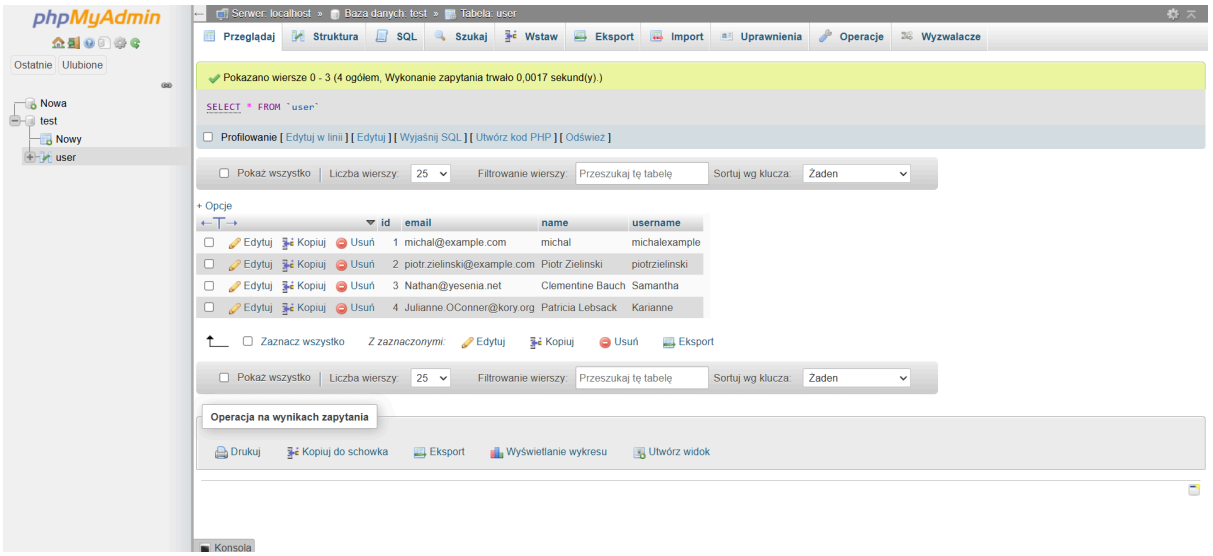

4. **Usuwanie użytkownika**:
    - Adres: `/users/{id}/delete`
    - Usuwa użytkownika z bazy danych oraz z interfejsu webowego.
5. **Pobieranie użytkownika z zewnętrznego API**:
    - Adres: `/fetch-user`
    - Pobiera dane użytkownika z serwisu JSONPlaceholder i zapisuje je w bazie.



# 6. Opis działania bazy danych w projekcie

1. **Środowisko bazy danych**:
    - W projekcie baza danych działa w środowisku **USBWebServer**, który jest lekkim serwerem webowym. Jest to narzędzie typu „portable”, co oznacza, że nie wymaga instalacji i można je uruchamiać bezpośrednio z dowolnego miejsca na komputerze.
    - USBWebServer zawiera wbudowane oprogramowanie takie jak Apache (serwer HTTP), MySQL (serwer baz danych) oraz phpMyAdmin (interfejs do zarządzania bazą danych).
2. **Obsługa bazy danych**:
    - Do zarządzania bazą danych używany jest **phpMyAdmin** — graficzny interfejs webowy, który umożliwia łatwe tworzenie, edytowanie i zarządzanie strukturą bazy danych oraz jej rekordami.
    - Projekt wykorzystuje **MySQL** jako system zarządzania bazą danych. Dane są przechowywane w relacyjnych tabelach, co pozwala na łatwe wykonywanie operacji CRUD (Create, Read, Update, Delete).
3. **Integracja z projektem**:
    - Projekt oparty na Spring Boot jest połączony z bazą danych MySQL poprzez **JPA (Java Persistence API)** oraz sterownik JDBC dla MySQL (`mysql-connector-j`).
    - W pliku `application.properties` (lub `application.yml`) definiowane są szczegóły połączenia z bazą danych, takie jak:
        - Host (np. `localhost` lub `127.0.0.1`),
        - Port (domyślnie `3306` dla MySQL w USBWebServer),
        - Nazwa bazy danych,
        - Nazwa użytkownika i hasło do bazy.
4. **Funkcjonalność bazy danych w projekcie**:
    - **Tabela `User`**:
        - Baza danych przechowuje informacje o użytkownikach w tabeli `User`.
        - Struktura tabeli:
            - **`id`**: unikalny identyfikator użytkownika (kolumna główna).
            - **`name`**: imię użytkownika.
            - **`username`**: login użytkownika.
            - **`email`**: adres e-mail użytkownika.
    - Dane w tabeli są zarządzane za pomocą operacji JPA w aplikacji, takich jak `save()`, `findAll()`, `findById()` i `delete()`.
5. **Sposób działania z USBWebServer**:
    - Uruchomienie bazy danych:
        - USBWebServer musi być uruchomiony przed startem aplikacji Spring Boot.
        - W panelu kontrolnym USBWebServer należy upewnić się, że usługa MySQL jest aktywna.
    - Dostęp do phpMyAdmin:

        - phpMyAdmin jest dostępny poprzez przeglądarkę, zazwyczaj pod adresem `http://localhost/phpmyadmin`.
        - Użytkownik może logować się do phpMyAdmin i przeglądać dane w bazie lub tworzyć nowe tabele.
    - Automatyczne tworzenie tabel:
        - Jeśli w bazie nie istnieje tabela `User`, Spring Boot automatycznie ją utworzy na podstawie klasy modelu `User`
6. **Przykładowy przepływ danych**:
    - Gdy użytkownik dodaje nowego użytkownika przez interfejs webowy bądź też za pomocą interfejsu Postmana, dane są przesyłane do kontrolera, a następnie zapisywane w tabeli `User` w bazie danych.
    - Przy pobieraniu użytkowników dane są odczytywane z bazy i wyświetlane na stronie w formie tabeli.



---

## 7. Wymagania systemowe

- **Java**: Wersja 21.
- **MySQL**: Działająca baza danych z odpowiednią konfiguracją w pliku `application.properties`.
- **Maven**: Do budowania projektu i zarządzania zależnościami.

---

## 8. Przykładowe konfiguracje

### Plik `application.properties`

```
spring.application.name=projekt-baza
spring.datasource.url=jdbc:mysql://localhost:3306/test
spring.datasource.username=root
spring.datasource.password=usbw
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true
```

---

## 9. Podsumowanie

DataWeaver to aplikacja webowa łącząca operacje CRUD na bazie danych z możliwością integracji z zewnętrznymi API. Jest elastyczna i rozszerzalna, co czyni ją odpowiednią bazą do dalszego rozwoju w kierunku bardziej zaawansowanych funkcjonalności.

Dokumentację przygotował:

Nikodem Franków