Alexandria

Sistema de Gestión Escolar
By: Daniel Mercado Cavazos

Project Description

The School Management System is an application designed to manage student and teacher information in a school. It allows for the creation, updating, deletion, and retrieval of student and teacher records, as well as role assignment and user management.

Requirements

Java 17 (or compatible version)

Spring Boot 3

MySQL (database)

Postman (API testing tool)

Maven (dependency management)

Project Structure

`src/main/java`: Java source code.

- `com.Escuela.Estudiantes.Entity`: JPA Entities.
  - `Student`
  - `Teacher`
  - `User`
  - `Role`
- `com.Escuela.Estudiantes.Repository`: Repository interfaces for data access.
  - `StudentRepository`
  - `TeacherRepository`
  - `UserRepository`
- `com.Escuela.Estudiantes.Service`: Services for business logic.
  - `StudentService`
  - `TeacherService`
- `com.Escuela.Estudiantes.Controller`: REST controllers for exposing APIs.
  - `StudentController`
  - `TeacherController`
  - `UserController`
  - `AuthController`

`src/main/resources`: Project resources.

- `application.properties`: Application configuration.

`pom.xml`: Maven configuration file.

Project Setup

- **Clone the Repository**:

```
git clone
https://github.com/your-username/your-repository.git
```

Project Setup

- **Configure the Database**:

Create a database in MySQL.

Update the `application.properties` file with your database credentials.

```
spring.datasource.url=jdbc:mysql://localhost:3306/your_database
spring.datasource.username=your_username
spring.datasource.password=your_password
```

Project Setup

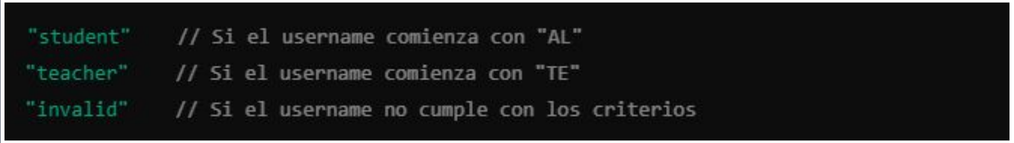
- **Install Dependencies**:

```
mvn install
```

API Endpoints -Authentication

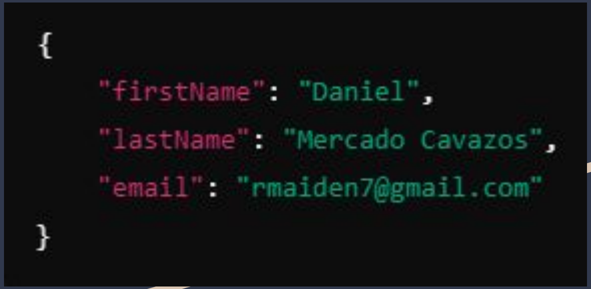
Login

- **Método**: `POST`
- **URL**: `/api/v1/login/login`
- **Descripción**: Realiza la autenticación de un usuario basado en el `username` y `password`.
- **Parámetros**:
  - `username`: Nombre de usuario.
  - `password`: Contraseña.



```
"student"    // Si el username comienza con "AL"
"teacher"    // Si el username comienza con "TE"
"invalid"    // Si el username no cumple con los criterios
```

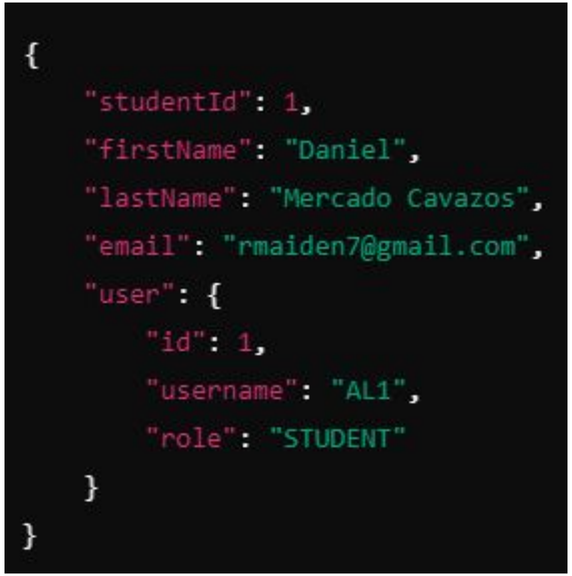
Estudiantes



```
{
    "firstName": "Daniel",
    "lastName": "Mercado Cavazos",
    "email": "rmaiden7@gmail.com"
}
```

Crear un Estudiante

- **Método**: `POST`
- **URL**: `/api/v1/students`
- **Descripción**: Crea un nuevo estudiante y genera un usuario asociado automáticamente.
- **Cuerpo de la Solicitud**:



```
{
    "studentId": 1,
    "firstName": "Daniel",
    "lastName": "Mercado Cavazos",
    "email": "rmaiden7@gmail.com",
    "user": {
        "id": 1,
        "username": "AL1",
        "role": "STUDENT"
    }
}
```

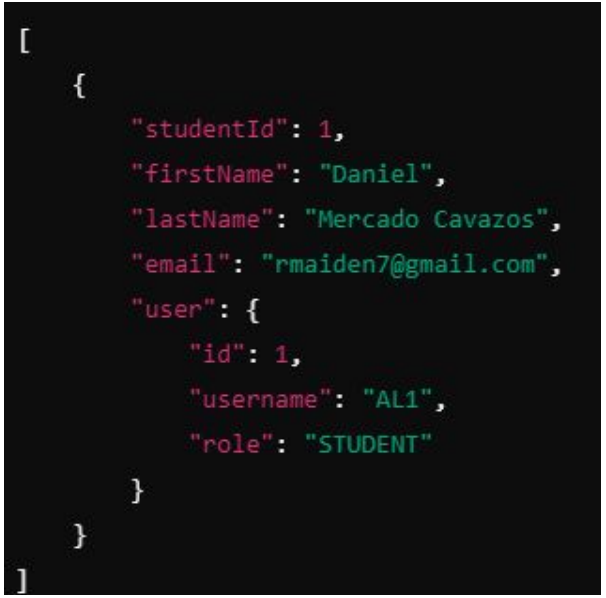
Obtener Todos los Estudiantes

Método: GET

URL: /api/v1/students

Descripción: Obtiene una lista de todos los estudiantes.

Respuesta:



```
[
    {
        "studentId": 1,
        "firstName": "Daniel",
        "lastName": "Mercado Cavazos",
        "email": "rmaiden7@gmail.com",
        "user": {
            "id": 1,
            "username": "AL1",
            "role": "STUDENT"
        }
    }
]
```

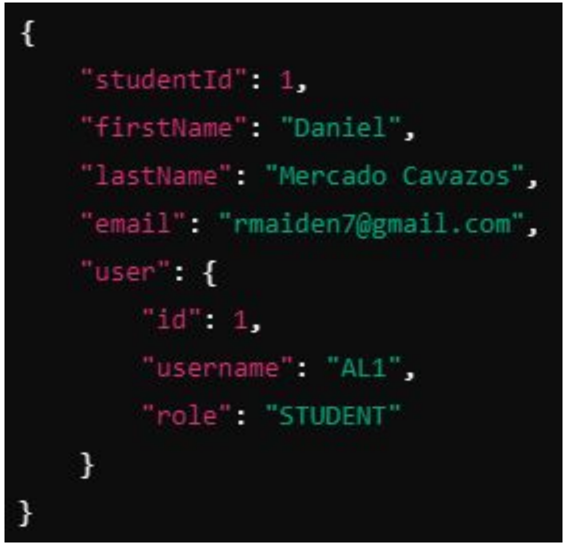
Obtener un Estudiante por ID

Método: `GET`

URL: `/api/v1/students/{studentId}`

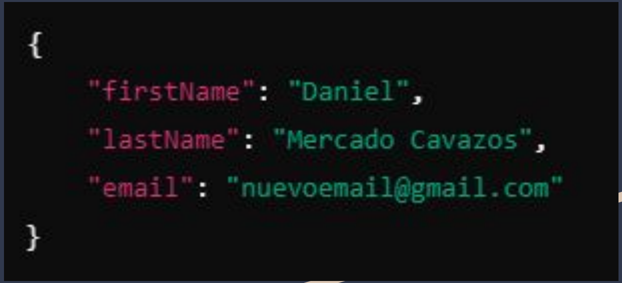
Descripción: Obtiene los detalles de un estudiante específico por su ID.

Respuesta:



```
{
    "studentId": 1,
    "firstName": "Daniel",
    "lastName": "Mercado Cavazos",
    "email": "rmaiden7@gmail.com",
    "user": {
        "id": 1,
        "username": "AL1",
        "role": "STUDENT"
    }
}
```

Actualizar un Estudiante

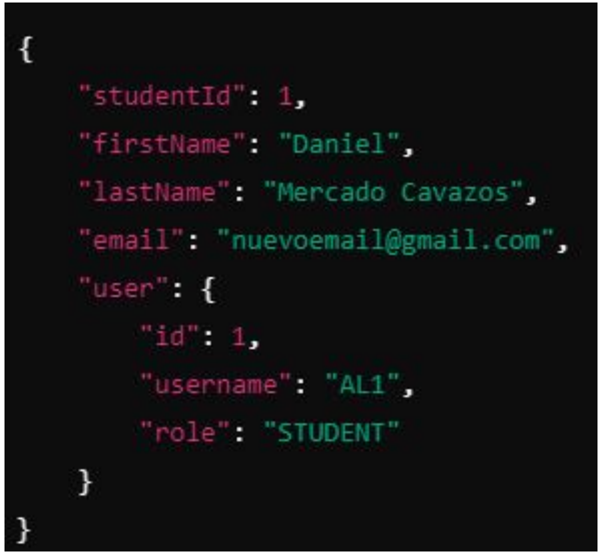


```
{
    "firstName": "Daniel",
    "lastName": "Mercado Cavazos",
    "email": "nuevoemail@gmail.com"
}
```

Método: PUT

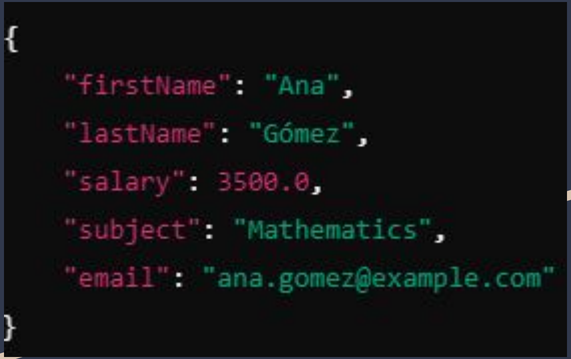
URL: `/api/v1/students/{studentId}`

Descripción: Actualiza la información de un estudiante existente.



```
{
    "studentId": 1,
    "firstName": "Daniel",
    "lastName": "Mercado Cavazos",
    "email": "nuevoemail@gmail.com",
    "user": {
        "id": 1,
        "username": "AL1",
        "role": "STUDENT"
    }
}
```

Maestros -Crear un Maestro



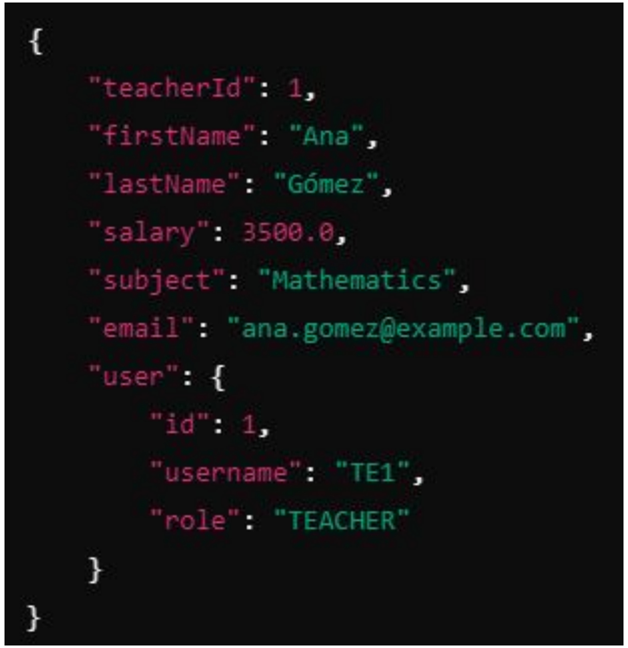
```
{
    "firstName": "Ana",
    "lastName": "Gómez",
    "salary": 3500.0,
    "subject": "Mathematics",
    "email": "ana.gomez@example.com"
}
```

Método: `POST`

URL: `/api/v1/teachers`

Descripción: Crea un nuevo maestro y genera un usuario asociado automáticamente.

Cuerpo de la Solicitud:



```
{
    "teacherId": 1,
    "firstName": "Ana",
    "lastName": "Gómez",
    "salary": 3500.0,
    "subject": "Mathematics",
    "email": "ana.gomez@example.com",
    "user": {
        "id": 1,
        "username": "TE1",
        "role": "TEACHER"
    }
}
```

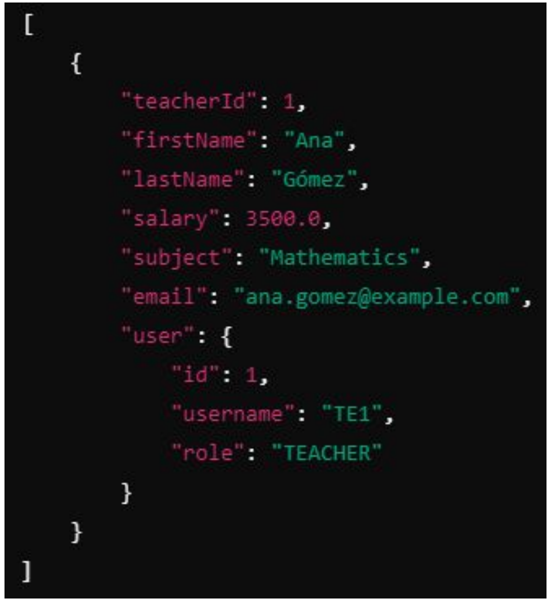
Obtener Todos los Maestros

Método: GET

URL: /api/v1/teachers

Descripción: Obtiene una lista de todos los maestros.

Respuesta:



```
[
    {
        "teacherId": 1,
        "firstName": "Ana",
        "lastName": "Gómez",
        "salary": 3500.0,
        "subject": "Mathematics",
        "email": "ana.gomez@example.com",
        "user": {
            "id": 1,
            "username": "TE1",
            "role": "TEACHER"
        }
    }
]
```

Testing

Tools: Postman or any REST client.

Test Scenarios:

- Create, read, update, and delete students and teachers.
- Login functionality and role-based access.

Troubleshooting

Common Issues:

- **Database Connection Errors**: Ensure that the database is running and credentials in `application.properties` are correct.
- **Invalid Credentials**: Check if the username and password are correctly provided.
- **Missing Dependencies**: Run `mvn install` to ensure all dependencies are included.

Contributing

Guidelines:

- Fork the repository and create a new branch.
- Make changes and test thoroughly.
- Submit a pull request with a detailed description of changes.

Contact

Author: Daniel Mercado Cavazos

Email: danielmercado0947@gmail.com

Github: https://github.com/RusoDan8