Spring Boot JWT Authentication API

**Overview**

This is a secure RESTful authentication API built using **Spring Boot**, **Spring Security**, and **JWT** (JSON Web Tokens). It supports:

* User registration and login
* Secure token-based authentication
* Role-based access control
* Stateless session handling (ideal for micro-services or frontend integration)
* Central configuration with Spring cloud Config Server

**Project Structure:**

src

└── main

└── java

└── com.tericcabrel.authapi

├── controllers # API endpoints (Auth & User)

├── dtos # Data transfer objects for login/signup

├── entities # JPA entity for User

├── exceptions # Global exception handling

├── configs # Spring Security & JWT configs

├── repositories # JPA repository interfaces

└── services # Core business logic (auth, JWT, user)

**Authentication Flow**

1. **Signup (POST /auth/signup)**
   * Accepts email, password, and full name.
   * Password is hashed using BCryptPasswordEncoder and stored in the DB.
2. **Login (POST /auth/login)**
   * Accepts email and password.
   * If valid, a JWT token is generated and returned.
3. **Accessing protected resources**
   * Clients send the token in the Authorization: Bearer <token> header.
   * JwtAuthenticationFilter extracts and validates the token.
   * Spring Security sets the authentication context if valid.

**Key Endpoints**

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Endpoint** | **Description** | **Auth Required** |
| POST | /auth/signup | Register a new user | No |
| POST | /auth/login | Login and receive JWT token | No |
| GET | /users/me | Get current user profile | Yes |
| GET | /users | List all users | Yes |
| GET | /userprofile | Custom JWT token-based profile | Yes |

**Security Setup**

* Stateless sessions using JWT (SessionCreationPolicy.STATELESS)
* JwtAuthenticationFilter parses and validates JWTs
* SecurityConfiguration defines public vs protected routes
* Custom UserDetailsService loads user by email
* DaoAuthenticationProvider handles login with bcrypt

**Centralised Configuration with Spring Cloud Config Server**

This project integrates with a **Spring Cloud Config Server** for centralized, external configuration.

* A separate app spring-config-server serves configuration files.
* Configs are stored in a Git repo.

**Testing**

* **Auth**
  + POST /auth/signup
  + POST /auth/login
* **User**
  + GET /users/me (with JWT in header)
  + GET /users

# 1. Register a user

POST http://localhost:8080/auth/signup

Content-Type: application/json

{

"email": "user@example.com",

"password": "strongpassword",

"fullName": "User One"

}

# 2. Login

POST http://localhost:8080/auth/login

Content-Type: application/json

{

"email": "user@example.com",

"password": "strongpassword"

}

# 3. Access protected route

GET http://localhost:8080/users/me

Authorization: Bearer <paste\_token\_here>

**Technologies Used**

* Java 17
* Spring Boot 3.x
* Spring Security
* JWT
* MySQL
* JPA / Hibernate