NestJS Microservices Documentation

Mohamed Hussein Abo El-Ela

December 20, 2024

Contents

1 Introduction

This project consists of two independent microservices built with the **NestJS** framework:

1. **Auth Service**: Handles user authentication (signup and login) with a MongoDB database.
2. **Movies Service**: Manages movie data with a separate MongoDB database.

Each microservice is deployed independently and connects to a different MongoDB database.

2 Project Setup

2.1 Prerequisites

Before you start, ensure you have the following installed:

- **Node.js** (v14 or higher)
- **MongoDB** (local or cloud instance like MongoDB Atlas)
- **npm** (v6 or higher)
- **Nest CLI**: Install using the command:

```
    npm install -g @nestjs/cli

```

2.2 Folder Structure

```
.
          auth-service/
                src/
                      auth/
                            auth.controller.ts
                            auth.service.ts
                            auth.module.ts
                            schemas/
                                user.schema.ts
                      app.module.ts
                      main.ts
                .env
          movies-service/
              src/
                    movies/
                          movies.controller.ts
                          movies.service.ts
                          movies.module.ts
                          schemas/
                              movie.schema.ts
                    app.module.ts
                    main.ts
                .env
```

3 Creating the Project

3.1 Setting Up the Auth Service

1. **Generate the Project**:

```
   nest new auth-service

```

2. **Install Dependencies**:

```
npm install @nestjs/mongoose mongoose bcryptjs jsonwebtoken dotenv

```

3. **Create the User Schema**:

```
// src/auth/schemas/user.schema.ts
import { Prop, Schema, SchemaFactory } from '@nestjs/mongoose';
import { Document } from 'mongoose';
import * as bcrypt from 'bcryptjs';

export type UserDocument = User & Document;

@Schema()
export class User {
  @Prop({ required: true, unique: true })
  email: string;

  @Prop({ required: true })
  password: string;
}

export const UserSchema = SchemaFactory.createForClass(User);

UserSchema.pre<UserDocument>('save', async function (next) {
  if (!this.isModified('password')) return next();
  this.password = await bcrypt.hash(this.password, 10);
  next();
});

UserSchema.methods.validatePassword = async function (password: string): Promise<boolean>
  {
  return bcrypt.compare(password, this.password);
};

```

4. **Create the Auth Service and Controller** (for signup and login functionality).

3.2 Setting Up the Movies Service

Follow similar steps to create the movies service, focusing on managing movie data.

4 Environment Variables

4.1 Auth Service .env

```
MONGODB_URI=mongodb://localhost:27017/auth-db
JWT_SECRET=your_secret_key
```

4.2 Movies Service .env

```
MONGODB_URI=mongodb://localhost:27017/movies-db
```

5 API Endpoints

5.1 Auth Service Endpoints

1. **Signup**:
   - **URL**: ‘POST /auth/signup‘

- **Request Body**:

```
{
  "email": "user@example.com",
  "password": "password123"
}
```

- **Response**:

```
{
  "message": "User created successfully",
  "userId": "<user_id>"
}
```

2. **Login**:

- **URL**: 'POST /auth/login'
- **Request Body**:

```
{
  "email": "user@example.com",
  "password": "password123"
}
```

- **Response**:

```
{
  "token": "<jwt_token>"
}
```

5.2 Movies Service Endpoints

1. **Add Movie**:

- **URL**: 'POST /movies'
- **Request Body**:

```
{
  "title": "Inception",
  "description": "A mind-bending thriller",
  "releaseYear": 2010
}
```

6 Screenshots

7 Conclusion

This project demonstrates how to create and deploy microservices using NestJS, each connecting to different MongoDB databases for authentication and movie management.

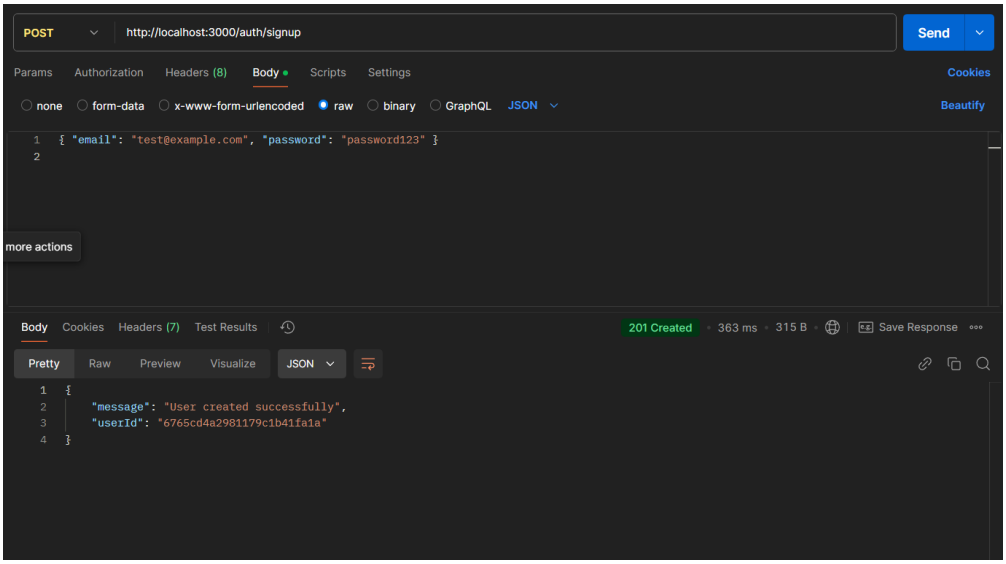

Figure 1: Auth Service Signup in Postman

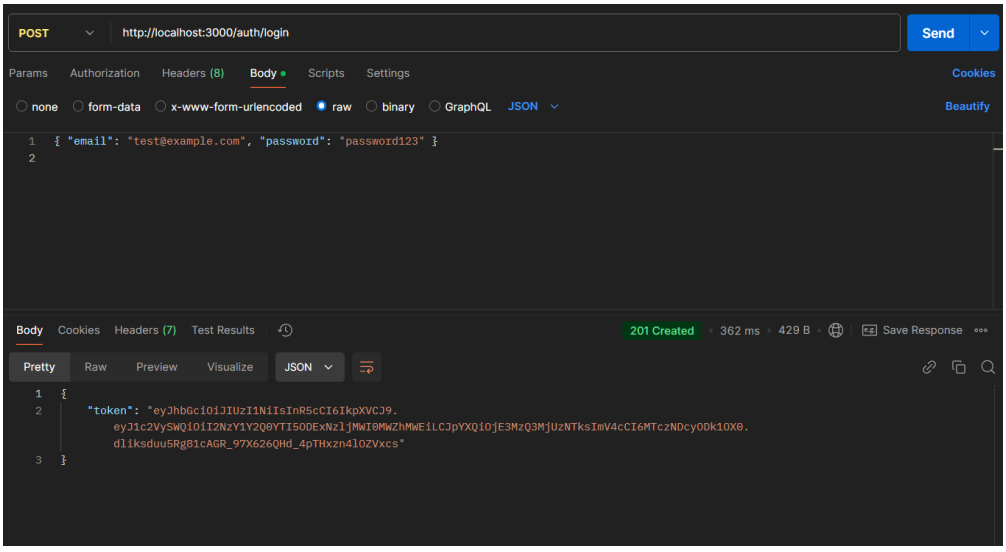

Figure 2: Auth Service Signin in Postman

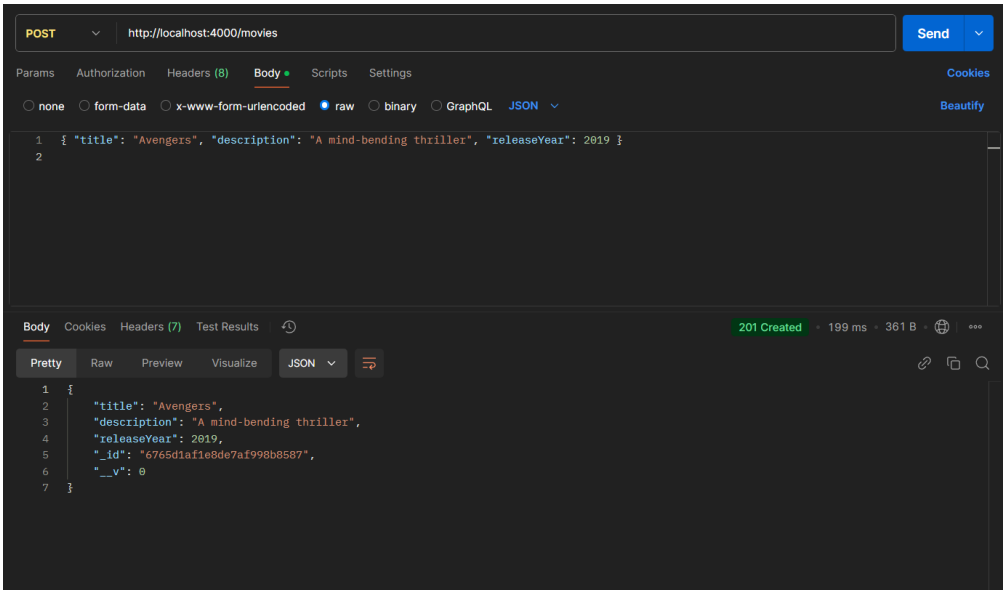

Figure 3: Movies Service - Add Movie in Postman

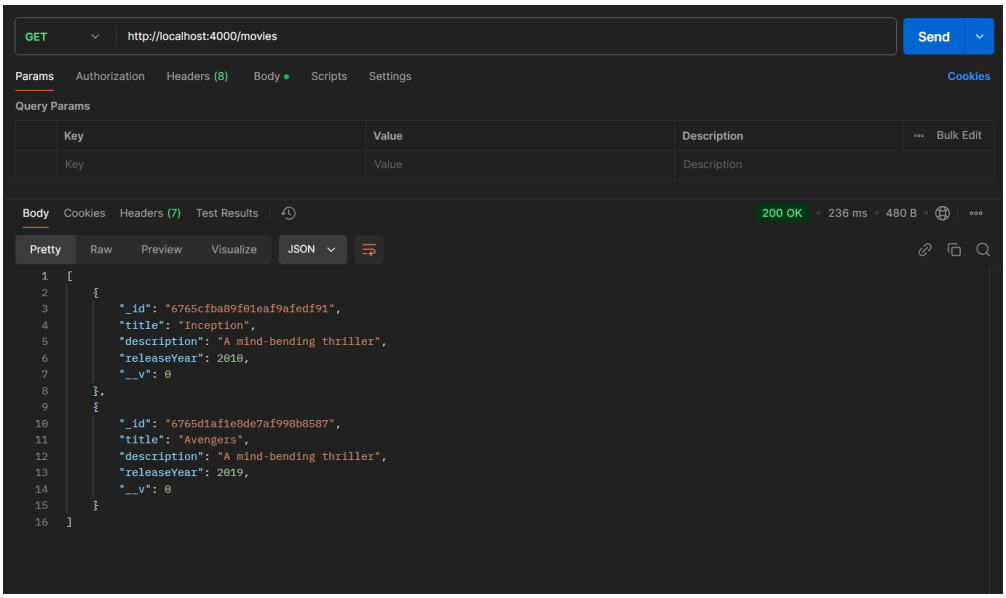

Figure 4: Movies Service - Get Movies in Postman