Project Documentation: Task Management System - Microservices Architecture

Table of Contents

Project Overview

The project **Task Management System - Microservices Architecture** comprises multiple components:

- **Auth Service:** Handles user authentication (registration, login, and profile management) using JWT and secure password encryption.
- **Task Service:** Manages tasks with full CRUD operations, including categorization, priority, and status tracking.
- **Frontend:** A responsive Angular application providing a modern UI/UX.

The backend is powered by AWS Lambda, API Gateway, and DynamoDB, forming a serverless architecture that supports scalability and efficiency.

Documentation Overview

This document aggregates all key documentation related to the project:

- **README:** Detailed explanation of the project, features, tech stack, installation, and deployment instructions.
- **Lambda Endpoints Guide:** Best practices for managing multiple endpoints in serverless architectures.
- **Serverless Configurations:** Deployment definitions for both Auth and Task services.
- **AWS Resources:** Explanation of AWS components like Lambda, API Gateway, and DynamoDB used in the project.

AWS and Serverless Architecture

Lambda Functions

AWS Lambda is used to run the business logic for our microservices. Each endpoint defined in our serverless configuration becomes a Lambda function, enabling scalability and cost efficiency.

DynamoDB Tables

- **Users Table:** Stores user records with hashed passwords.
- **Tasks Table:** Stores task records associated with users, along with statuses and priorities.

Service Configurations

Auth Service

The following is an excerpt from the `auth-service/serverless.yml` configuration:

```yaml
service: auth-service
provider:
  name: aws
  runtime: nodejs18.x
  region: us-east-1
  environment:
    USERS_TABLE: ${param:usersTableName}
    JWT_SECRET: ${param:jwtSecret}
functions:
  registerUser:
    handler: src/auth.registerUser
    events:
      - http:
          path: auth/register
          method: post
  loginUser:
    handler: src/auth.loginUser
    events:
      - http:
          path: auth/login
          method: post
```

Task Service

Below is an excerpt from the `task-service/serverless.yml` configuration:

```yaml
service: task-service
provider:
```

```
  name: aws
  runtime: nodejs18.x
  region: us-east-1
  environment:
    TASKS_TABLE: ${param:tasksTableName}
    JWT_SECRET: ${param:jwtSecret}
functions:
  createTask:
    handler: src/tasks.create
    events:
      - http:
          path: /tasks
          method: post
  getTasks:
    handler: src/tasks.getAll
    events:
      - http:
          path: /tasks
          method: get
  updateTask:
    handler: src/tasks.update
    events:
      - http:
          path: /tasks/{taskId}
          method: put
  deleteTask:
    handler: src/tasks.delete
    events:
      - http:
          path: /tasks/{taskId}
          method: delete
```

Lambda Endpoints Guide

For managing large numbers of endpoints, consider the following best practices:

- **Separation by Files:** Divide your serverless configuration into separate files for better maintainability.
- **Single Lambda Router:** Use a single Lambda function with a framework like Express to route multiple endpoints, reducing the number of Lambda functions.
- **API Gateway Integration:** Leverage API Gateway with OpenAPI/Swagger documentation to manage and document your endpoints.

For more details, please refer to the LAMBDA_ENDPOINTS_GUIDE.md file in the project.

Frontend Documentation

The Angular frontend is built using Angular Material and NgRx, ensuring a modern, responsive UI. For installation, configuration, and deployment instructions, see the README.

Conclusion

This documentation provides a comprehensive overview of the Task Management System project, including its architecture, AWS components, Lambda functions, and best practices for managing multiple endpoints. It serves as a single source of truth for developers working on the project.