**KAIBURR ASSESSMENT**

**NAME : ABIRAMI R**

**REG NO : 22MCA0023**

**Task 2. Swagger codegen.**

**Create the same REST API as in task #1, but use**

**https://editor.swagger.io/ to create your API definition and generate the server code. Choose any java-based server or server framework,that you like. You can either use the online editor or generate the code manually, e.g. using this document: <https://github.com/swagger-api/swagger>**

**codegen/wiki/server-stub-generator-howto.**

**Make sure that you can deploy/run the generated code. Once your stub is ready - implement the**

**same functionality as described in task #1, but now in java.**

**Finally, be sure that you can show how your application responds to requests using postman,**

**curl or any other HTTP client.**

**SWAGGER API DEFINITION:**

swagger: "2.0"

info:

version: "1.0.0"

title: Server API

paths:

/servers:

get:

summary: Get all servers or find servers by name

parameters:

- in: query

name: name

type: string

description: Server name to search for

responses:

200:

description: Successful response

schema:

type: array

items:

$ref: "#/definitions/Server"

404:

description: Server not found

post:

summary: Create a new server

parameters:

- in: body

name: server

required: true

schema:

$ref: "#/definitions/Server"

responses:

200:

description: Server created

schema:

$ref: "#/definitions/Server"

/servers/{id}:

get:

summary: Get a server by ID

parameters:

- in: path

name: id

required: true

type: string

responses:

200:

description: Successful response

schema:

$ref: "#/definitions/Server"

404:

description: Server not found

put:

summary: Update a server by ID

parameters:

- in: path

name: id

required: true

type: string

- in: body

name: server

required: true

schema:

$ref: "#/definitions/Server"

responses:

200:

description: Server updated

schema:

$ref: "#/definitions/Server"

delete:

summary: Delete a server by ID

parameters:

- in: path

name: id

required: true

type: string

responses:

204:

description: Server deleted

definitions:

Server:

type: object

properties:

id:

type: string

name:

type: string

language:

type: string

framework:

type: string

**SPRING BOOT APPLICATION:**

public class ServerApplication {

public static void main(String[] args) {

SpringApplication.run(ServerApplication.class, args);

}

}

@RestController

@RequestMapping("/servers")

public class ServerController {

private List<Server> servers = new ArrayList<>();

@GetMapping

public List<Server> getServers(@RequestParam(required = false) String name) {

if (name != null) {

return servers.stream()

.filter(server -> server.getName().contains(name))

.collect(Collectors.toList());

} else {

return servers;

}

}

@GetMapping("/{id}")

public ResponseEntity<Server> getServerById(@PathVariable String id) {

Optional<Server> server = servers.stream()

.filter(s -> s.getId().equals(id))

.findFirst();

return server.map(s -> ResponseEntity.ok().body(s))

.orElse(ResponseEntity.notFound().build());

}

@PostMapping

public ResponseEntity<Server> createServer(@RequestBody Server server) {

servers.add(server);

return ResponseEntity.ok().body(server);

}

@PutMapping("/{id}")

public ResponseEntity<Server> updateServer(@PathVariable String id, @RequestBody Server updatedServer) {

Optional<Server> server = servers.stream()

.filter(s -> s.getId().equals(id))

.findFirst();

if (server.isPresent()) {

Server existingServer = server.get();

existingServer.setName(updatedServer.getName());

existingServer.setLanguage(updatedServer.getLanguage());

existingServer.setFramework(updatedServer.getFramework());

return ResponseEntity.ok().body(existingServer);

} else {

return ResponseEntity.notFound().build();

}

}

@DeleteMapping("/{id}")

public ResponseEntity<Void> deleteServer(@PathVariable String id) {

Optional<Server> server = servers.stream()

.filter(s -> s.getId().equals(id))

.findFirst();

if (server.isPresent()) {

servers.remove(server.get());

return ResponseEntity.noContent().build();

} else {

return ResponseEntity.notFound().build();

}

}

}