```java
package rest;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.web.bind.annotation.DeleteMapping;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;
import org.springframework.web.server.ResponseStatusException;

import io.swagger.v3.oas.annotations.Operation;
import io.swagger.v3.oas.annotations.Parameter;
import io.swagger.v3.oas.annotations.media.Schema;
import io.swagger.v3.oas.annotations.responses.ApiResponse;
import io.swagger.v3.oas.annotations.responses.ApiResponses;

@RestController
public class CategoryController {
    @Autowired
    private CatsRepo categoryRepo;

    @GetMapping("/categories")
    @Operation(summary = "Get all categories")
    public List<Category> getAll() {
        return categoryRepo.findAll();
    }

    @GetMapping("/categories/{code}")
    @Operation(summary = "Get category by code",
               description = "Given a category code it retrieves details of category")
    @ApiResponses(value =
           { @ApiResponse(responseCode = "200", description = "Found the category by given code"),
             @ApiResponse(responseCode = "404", description = "Category code not found") })
    public Category getCategoryByCode(
            @Parameter(description = "Category Code", allowEmptyValue = false)
            @PathVariable("code") String code) {
        var cat = categoryRepo.findById(code);
        if (cat.isPresent())
            return cat.get();
        else
            throw new ResponseStatusException(HttpStatus.NOT_FOUND, "Category Code Not Found");
    }

    @GetMapping("/categoriesByDesc")
    public List<Category> getCategoryByDesc(@RequestParam(name = "desc", defaultValue = "") String desc) {
        return categoryRepo.findByDescriptionContaining(desc);
    }

    @DeleteMapping("/categories/{code}")
    public void deleteCategoryByCode(@PathVariable("code") String code) {
        var cat = categoryRepo.findById(code);
        if (cat.isPresent())
            categoryRepo.deleteById(code);
        else
            throw new ResponseStatusException(HttpStatus.NOT_FOUND, "Category Code Not Found");
    }

    @PutMapping("/categories/{code}")
    @Operation(summary = "Update category description", description = "Takes category code and new description and updates description in database")
    @ApiResponses(value = { @ApiResponse(responseCode = "200", description = "Updated category successfully"),
            @ApiResponse(responseCode = "404", description = "Category code not found") })
    public Category updateCategoryDesc(@Parameter(description = "Category Code") @PathVariable("code") String code,
            @Parameter(description = "New desription for category") @RequestParam("desc") String desc) {
        var cat = categoryRepo.findById(code);
        if (cat.isPresent()) {
            var category = cat.get();
            category.setDescription(desc);
            categoryRepo.save(category);
            return category;
        } else
            throw new ResponseStatusException(HttpStatus.NOT_FOUND, "Category Code Not Found");
    }

    @PostMapping("/categories")
    @ApiResponses(value = { @ApiResponse(responseCode = "400", description = "Invalid data passed"),
            @ApiResponse(responseCode = "200", description = "Inserted new category"), })
    //@PreAuthorize(value = "hasRole('ROLE_ADMIN')")
    public Category addCategory(
            @Parameter(description = "Details of a new category", schema = @Schema(implementation = Category.class)) @RequestBody Category category) {
        try {
            categoryRepo.save(category);
            return category;
        } catch (Exception ex) {
            throw new ResponseStatusException(HttpStatus.BAD_REQUEST, "Invalid data!");
        }
    }

}
```