package com.dai.webServer.Resources;

import java.net.URI; import java.util.List; import java.util.Optional;

import org.json.simple.JSONObject; import org.json.simple.parser.JSONParser;
import org.json.simple.parser.ParseException;

import org.springframework.beans.factory.annotation.Autowired; import
org.springframework.http.ResponseEntity; import org.springframework.web.bind.annotation.CrossOrigin;
import org.springframework.web.bind.annotation.DeleteMapping; im-
port org.springframework.web.bind.annotation.GetMapping; import
org.springframework.web.bind.annotation.PathVariable; import org.springframework.web.bind.annotation.PostM
import org.springframework.web.bind.annotation.PutMapping; import
org.springframework.web.bind.annotation.RequestBody; import org.springframework.web.bind.annotation.Reque
import org.springframework.web.bind.annotation.RestController; import
org.springframework.web.servlet.support.ServletUriComponentsBuilder;

import com.dai.webServer.Exceptions.DivisionNotFoundException; import
com.dai.webServer.Objects.Division; import com.dai.webServer.Repos.DivisionRepository;

import com.dai.db.AnalyticsDB;

@RequestMapping(“/”) @RestController @CrossOrigin(origins = “http://localhost:5000”,
maxAge = 1728000)

public class DivisionResources {

```
private AnalyticsDB  db = new AnalyticsDB();
@Autowired
private DivisionRepository divisionRepository;

//listar todas as divisoes
@GetMapping("/division")
public List<Division> retrieveAllHouse() {
    return divisionRepository.findAll();
}

//listar divisao por id
@GetMapping("/division/{id}")
public Division retrieveDivision(@PathVariable long id) {
    Optional<Division> division = divisionRepository.findById(id);

    if (!division.isPresent())
        throw new DivisionNotFoundException("id-" + id);

    return division.get();
}

//Criar Divisao
@PostMapping("/division")
```

```
public ResponseEntity<Object> createDivision(@RequestBody Division division) {

    Division savedDivision = divisionRepository.save(division);

    URI location = ServletUriComponentsBuilder.fromCurrentRequest().path("/{id}")
            .buildAndExpand(savedDivision.getId_division()).toUri();

    return ResponseEntity.created(location).build();

}
@PostMapping("/division/arm")
public String updateDivision(@RequestBody Division division) {

AnalyticsDB  newDb = new AnalyticsDB();
    newDb.updateArm(division.getArmed(), division.getId_division());

return "done";
}




@PostMapping("/division/name")
public String updateName(@RequestBody Division division) {

AnalyticsDB  newDb = new AnalyticsDB();
    newDb.updateName(division.getName(), division.getSensor_id(),division.getId_division());

return "done";
}

//Alterar dados da divisao
@PostMapping("/division/{id}")
public String updateDivision(@RequestBody Division division, @PathVariable String id) {
System.out.println("I'm here boye");
AnalyticsDB  newDb = new AnalyticsDB();

            newDb.updateDivision(division.getSensor_id(), id );
            System.out.println("done");

return "done";
}

@PostMapping("/rfidAdd")
public String addRFID(@RequestBody String json) throws  ParseException {
```

```
    AnalyticsDB newDb = new AnalyticsDB();

    JSONParser parser = new JSONParser();
    JSONObject jsonObject = (JSONObject) parser.parse(json);



    Long user = (Long) jsonObject.get("user_id");

    String tag = (String) jsonObject.get("tag");

    newDb.addCard(user, tag);







    return "done";

}


//Apagar divisao
@PostMapping("/division/delete/{id}")
public String deleteDivision(@PathVariable long id) {
    divisionRepository.deleteById(id);
    return "done";
}
}
```