

```

package com.dai.webServer.Resources;

import java.net.URI; import java.util.List; import java.util.Optional; import
com.dai.webServer.Mqtt.; import org.json.simple.; 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; import
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.RequestMapping;
import org.springframework.web.bind.annotation.RestController; import
org.springframework.web.servlet.support.ServletUriComponentsBuilder;

import com.dai.webServer.Exceptions.UtilizadorNotFoundException; import
com.dai.webServer.Objects.User; import com.dai.webServer.Repos.UtilizadorRepository;

@CrossOrigin(origins = "http://localhost:5000", maxAge = 3600) @RequestMapping("/") @RestController public class UserResources {

    @Autowired
    private UtilizadorRepository utilizadorRepository;

    @GetMapping("/utilizador")
    public List<User> retrieveAllUsers() {
        return utilizadorRepository.findAll();
    }

    @GetMapping("/utilizador/{id}")
    public User retrieveUser(@PathVariable long id) {
        Optional<User> utilizador = utilizadorRepository.findById(id);

        if (!utilizador.isPresent())
            throw new UtilizadorNotFoundException("id-" + id);

        return utilizador.get();
    }
    //IDK tho
    @PostMapping("/login")

    public User verifyUser(@RequestBody User utilizador) {

        String email = utilizador.getEmail();
        String password = utilizador.getPassword();

        User returned = findUserByEmail(email);
        String passBase = returned.getPassword();

```

```

return returned;

}

@PostMapping("/mail")

public User getUser(@RequestBody User utilizador) {

String email = utilizador.getEmail();

User returned = findUserByEmail(email);

return returned;

}

public User findUserByEmail(String email) {

Optional<User> utilizador = utilizadorRepository.findByEmail(email);

if (!utilizador.isPresent())
    throw new UtilizadorNotFoundException("id-" + email);

return utilizador.get();
} @DeleteMapping("/utilizador/{id}") public void deleteUser(@PathVariable
long id) { utilizadorRepository.deleteById(id); }

@PostMapping("/utilizador")
public ResponseEntity<Object> createUser(@RequestBody User utilizador) {

User savedUser = utilizadorRepository.save(utilizador);

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

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

@PostMapping("/changeLight")

```

```

public void createUser(@RequestBody String light) throws ParseException {

    JSONParser parser = new JSONParser();
    JSONObject jsonObject = (JSONObject) parser.parse(light);
    /*    */
    /* Object dataF = jsonObject.get("id"); */

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

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

    JSONObject rgsJson = (JSONObject) jsonObject.get("rgb");
    String rgb = (String) rgsJson.toJSONString();
    /* String state = (String) jsonObject.get("state"); */
    Publisher pub = new Publisher();
    System.out.println(topic);
    System.out.println(id);
    System.out.println(rgb);

    /* System.out.println(state); */
    /* String mensagem = new StringBuilder().append(state).append(rgb).toString(); */
    /* System.out.println(state); */
    pub.sendMessage(topic, rgb, id);

}

```

```

@PutMapping("/utilizador/{id}")
public ResponseEntity<Object> updateUser(@RequestBody User utilizador, @PathVariable long id) {

    Optional<User> userOptional = utilizadorRepository.findById(id);

    if (!userOptional.isPresent())
        return ResponseEntity.notFound().build();

    utilizador.setId_user(id);

    utilizadorRepository.save(utilizador);
}

```

```
        return ResponseEntity.noContent().build();  
    }  
}
```