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 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.PasswordErrorException; 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 emailBase = returned.getEmail();
String passBase  = returned.getPassword();

if(password.equals(passBase)){

    return returned;
}else{

    throw new PasswordErrorException("401");
}

}


@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) {

    String mailReturned = utilizador.getEmail();


    Optional<User> exist = utilizadorRepository.findByEmail(mailReturned);
    if (exist.isPresent()){
```

```
    System.out.println("not");

    return  null;

    }
    System.out.println("It kind of worked");

    User savedUser = utilizadorRepository.save(utilizador);

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

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


}



@PostMapping("/changeLight")
public String changeLight(@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 message = (String) jsonObject.get("message");

    Publisher pub = new Publisher();
    System.out.println(topic);
    System.out.println(message);

    String correctedTopic = "request/"  + topic;
    pub.sendMessage(correctedTopic , message ,"lightChange");
```

```
    return "done";

}


@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();
}
}
```