Universidade de Évora

Departamento de Informática

Sistemas Distribuídos

Controlo de Vacinação

Relatório do Trabalho do segundo trabalho prático

Curso de Engenharia Informática

Docente: Prof. José Saias

Discentes: Leonardo Catarro, nº 43025

Diogo Solipa, nº 43071

2020/2021

Introdução

Imagine um Sistema Nacional de Vacinação, cujo sistema distribuído inclui os seguintes módulos:

1. Aplicação do Cidadão: em que um utente pode escolher um centro de vacinação e proceder ao auto-agendamento;
2. Centro de Vacinação: módulo executado por cada Centro Municipal;
3. Módulo DGS: para a DGS, para operações de âmbito nacional e coordenação dos centros;
4. Módulo central: contém todos os controladores e funcionalidades.

Este trabalho está estruturado em 3 camadas: API, serviço e controladores.

Estruturação do Código

Para este trabalho o módulo central contém todas as implementações das funcionalidades necessárias ao funcionamento do sistema de vacinação.

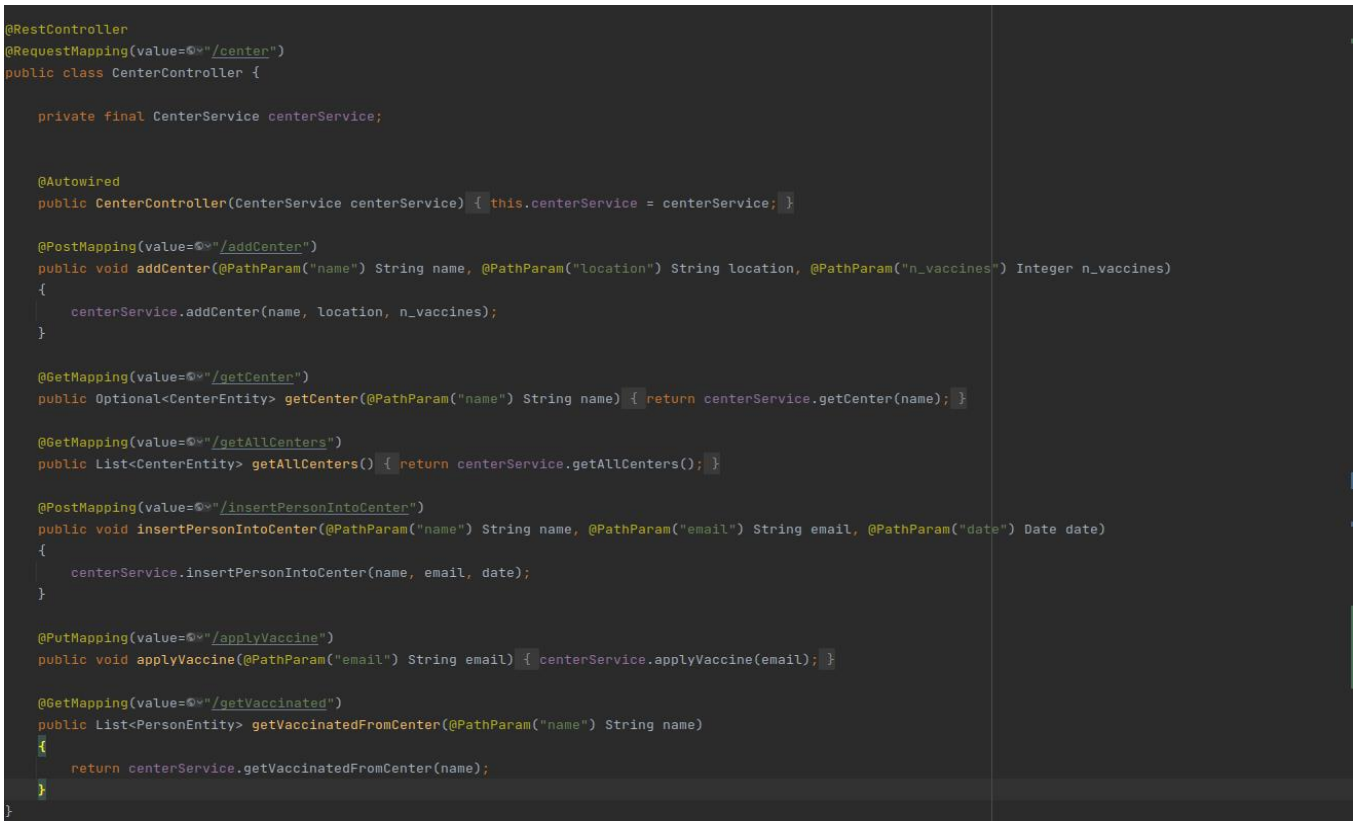
- Módulo Cliente:

```java
@RestController
@RequestMapping(value = "/person")
public class PersonController {

    PersonEntity person;
    private final PersonService personService;

    @Autowired
    public PersonController(PersonService personService) { this.personService = personService; }

    @GetMapping
    public List<PersonEntity> getPeople() { return personService.getPeople(); }

    @PostMapping(value ="/insertPerson", produces = "application/json")
    public void registerPerson(@PathParam("name") String name, @PathParam("age") Integer age, @PathParam("email") String email, @PathParam("date") Date date, @PathParam("centerName") String
    {
        personService.registerPerson(name, age, email, date, centerName);
    }

    @DeleteMapping(value="/testDelete")
    public void deletePerson(@PathParam("id") Integer id) { personService.deletePerson(id); }

    @PutMapping(value="/testPut")
    public void updatePerson(@PathParam("name") String name, @PathParam("age") Integer age, @PathParam("currEmail") String currEmail, @PathParam("newEmail") String newEmail)
    {
        personService.updatePerson(name, age, currEmail, newEmail);
    }
}
```

- Módulo DGS:

```java
@RestController
@RequestMapping(value="/dgs")
public class DGSController {

    private final DGSService dgsService;

    @Autowired
    public DGSController(DGSService dgsService) { this.dgsService = dgsService; }

    @PostMapping(value="/registerCenter")
    public void registerCenter(CenterEntity center) { dgsService.registerCenter(center); }

    @PutMapping(value="/updateDateAndVaccines")
    public void updateDateAndVaccines(@PathParam("name") String name, @PathParam("n_vaccines") Integer n_vaccines, @PathParam("date") Date date)
    {
        dgsService.updateDateAndVaccines(name, n_vaccines, date);
    }

    @GetMapping(value = "/getVaccinatedPerDay")
    public List<PersonEntity> getVaccinatedPerDay(@PathParam("date") Date date)
    {
        return dgsService.getVaccinatedPerDay(date);
    }

    @GetMapping(value = "/getListOfPeopleToBeVaccinatedPerDay")
    public List<PersonEntity> getListOfPeopleToBeVaccinatedPerDay(@PathParam("date") Date date)
    {
        return dgsService.getListOfPeopleToBeVaccinatedPerDay(date);
    }

}
```

- Módulo Centro:



```java
@RestController
@RequestMapping(value="/center")
public class CenterController {

    private final CenterService centerService;


    @Autowired
    public CenterController(CenterService centerService) { this.centerService = centerService; }

    @PostMapping(value="/addCenter")
    public void addCenter(@PathParam("name") String name, @PathParam("location") String location, @PathParam("n_vaccines") Integer n_vaccines)
    {
        centerService.addCenter(name, location, n_vaccines);
    }

    @GetMapping(value="/getCenter")
    public Optional<CenterEntity> getCenter(@PathParam("name") String name) { return centerService.getCenter(name); }

    @GetMapping(value="/getAllCenters")
    public List<CenterEntity> getAllCenters() { return centerService.getAllCenters(); }

    @PostMapping(value="/insertPersonIntoCenter")
    public void insertPersonIntoCenter(@PathParam("name") String name, @PathParam("email") String email, @PathParam("date") Date date)
    {
        centerService.insertPersonIntoCenter(name, email, date);
    }

    @PutMapping(value="/applyVaccine")
    public void applyVaccine(@PathParam("email") String email) { centerService.applyVaccine(email); }

    @GetMapping(value="/getVaccinated")
    public List<PersonEntity> getVaccinatedFromCenter(@PathParam("name") String name)
    {
        return centerService.getVaccinatedFromCenter(name);
    }
}
```

Funcionalidades

- Módulo Cliente:

public List<PersonEntity> getPeople() : Esta função vai inserir numa lista todos as pessoas existentes na tabela person da BD.

public void registerPerson(String name, Integer age, String email, Date date, String centerName) : Esta função vai registar o autoagendamento da pessoa ficando registado o seu nome, idade e email, o centro onde deseja ser vacionado e uma data de preferência para a aplicação da vacina.

- Módulo DGS:

public void registerCenter(CenterEntity center) : Esta função vai registar um centro de vacinação, criando um objeto da classe CenterEntity, adicionando-o na tabela de centros para futura aplicação de vacinas.

public void updateDateAndVaccines(String Name, Integer n_vaccines, Date date) : Esta função vai procurar o centro com nome "Name" e atualizar o n_vaccines e a data de vacinação

public List<PersonEntity> getListOfPeopleToBeVaccinatedPerDay(Date date) : Esta função retorna a lista de pessoas vacinadas no dia passado como argumento na variável date.

- Módulo Centro:

public void addCenter(String name, String location, Integer n_vaccines): Esta função adiciona um centro à tabela de centros existentes.

public Optional<CenterEntity> getCenter(String name) : Esta função retorna um objeto do tipo CenterEntity cujo o nome do centro é "name".

public List<CenterEntity> getAllCenters() : Esta função retorna uma lista de todos os centros existentes

public void insertPersonIntoCenter(String name, String email, Date date): Esta função insere a pessoa com o email passado como argumento para ser vacinada no centro “name” no dia “date”.

public void applyVaccine(String email): Esta função regista a realização da vacina para pessoa cujo email é passado como argumento.

public List<PersonEntity> getVaccinatedFromCenter(String name) : Esta função lista o total de vacinados no centro com o nome “name”.

Problemas Encontrados e Observações

No desenvolvimento deste trabalho, um dos maiores problemas encontrados terá sido na utilização do JPA, mais especificamente ao nível dos repositórios e na utilização de listas nas tabelas da Base de Dados. Tentámos usar o "OneToMany" que geraria uma outra tabela, mas acabámos por abandonar a ideia pois não conseguimos que funcionasse como desejado.

Em termos gerais, conseguimos aplicar os conhecimentos aprendidos em Sistemas Distribuídos, conseguindo então ter a maioria dos requisitos implementados e a funcionar corretamente.

Como Compilar e Executar o Programa

1. Abrir 4 terminais (1 para cada módulo)
2. Em cada um dos terminais:
    a. mvn compile
    b. mvn package
    c. java -jar /target / *nomeFicheiro.jar

* Nota: o ficheiro jar é criado na pasta target ao aplicar o comando mvn package