Escola de Ciências e Tecnologia
Departamento de Informática
Licenciatura em Engenharia Informática
Unidade Curricular Sistemas Distribuídos
Ano letivo 2020/2021

Relatório Trabalho 2

Docente:
Professor José Saias
Discentes:
José Santos - 43017
Filipe Alfaiate - 43315

July 12, 2021

Contents

1 Introdução

No âmbito da unidade curricular de Sistemas Distribuídos lecionada pelo professor José Saias, foi solicitado, como segundo trabalho, que os alunos implementassem sistema distribuído inclui os módulos:

- Aplicação do cidadão: O cidadão através de uma *WebApp* pode agendar, remarcar e verificar detalhes sobre o seu agendamento;
- Centro de vacinação: Sendo possível registar novos centros, entrar na conta de um centro, ver os cidadãos que iram ser vacinados, informar os respetivos cidadãos, vacinar um cidadão e fazer *logout* da conta do centro;
- DGS: Neste módulo é possível atualizar a base dados consoante os novos cidadãos inscritos, distribuir um número nacional de vacinas pelos centros consoante as pessoas mais velhas e listar o número total de vacinas por centro num determinado dia.

Utilizando uma `WebApp` em *Cloud* para a interação com o cidadão, um serviço `REST` interagindo com o módulo Centro e DGS e uma base de dados persistente em postgres e *Google Sheets*.

2 Implementação

2.1 REST

É inicializado um servidor `grizzly`. O `REST` contém a *class* `ListaVacinas`, que tem uma `ArrayList` de `Vacinas` e uma *class* `ListaVacinasResource` com `GET`, `PUT` e `POST`, muito semelhante ao exemplo da atividade 10 das aulas práticas.

2.2 Cidadão

A aplicação Cidadão foi implementada através de uma *Google Apps Script* com as seguintes funções:

- `doGet`: Esta função chama o `getCentros` e retorna o ficheiro `html`;
- `registarVacinacao`: Esta função recebe como argumentos o nome, idade, e-mail, centroVac e data do cidadão, adiciona-o à *sheet* Data e envia um e-mail com o seu código;

- `checkValidCode`: Recebe um código e verifica se este existe;
- `checkData`: Verifica se a data inserida não é antecessor ao dia presente;
- `checkEmailData`: Verifica se o e-mail recebido ainda não se encontra na *sheet* e chama o `checkData`;
- `getCentros`: Devolve o nome de todos os centros;
- `getSheet`: Dado um nome devolve a página com esse nome;
- `getDados`: Dado um código devolve os dados associados a esse utilizador;
- `mudarVacinacao`: Se o utilizador quiser alterar o centro e/ou data da sua vacinação. Envia também um e-mail a informar da mudança;
- `getRowEmail`: Dado um e-mail devolve a linha associada;
- `addCentro`: Recebido um Centro e um número máximo de vacinas insere os dados na página Centros;
- `getRowCodigo`: Dado um código devolve a linha associada;
- `addConfirmacao`: Adiciona um "`y`" se o cidadão puder ser vacinado na hora marcada;
- `addNegacao`: Adiciona um "`n`" se o cidadão não puder ser vacinado na hora marcada;
- `sendEmail`: Dado um e-mail e uma mensagem envia para o e-mail recebido a mensagem.

2.3 Centro de Vacinação

O centro de vacinação foi implementado com `Gradle`. Existe uma conexão com a base dados `postgres`, a `Apps Script` do cliente e o serviço `REST`. As seguintes `class` foram implementadas com os seguintes métodos para realizar as funcionalidades propostas:

- `Centro`: Esta *class* tem os métodos referentes às operações que um centro pode realizar;
  - `getVacinas_XML`: obtém a `ListaVacinas` do dia;

  - `getFilaEspera`: obtém os cidadãos que marcaram ser vacinados no centro no presente dia;
  - `getVacinasDia`: obtém o número de vacinas diárias para o centro;
  - `ordenarIdade`: ordena os cidadãos por ordem decrescente de idade;
  - `infromarCidadaos`: percorre a `listaCidadaos` e envia um e-mail para cada um com a informação de poder ou não ser vacinados;
  - `vacinar`: dado um código, remove o cidadão associado da fila de espera e adiciona-o aos vacinados;
  - `showCidadaosASerVacinados`: mostra informações relativas aos cidadãos a serem vacinados naquele centro no presente dia.
  - `printComandosCentro`: Mostra comandos relativamente ao centro;

- `Cidadao`: Esta *class* guarda informações relativamente a cada cidadão;
- `GoogleCredencial & ScriptAdpter`: Estas funções auxiliam na conexão com a `Apps Script`;
- `ListaVacinas`: Esta *class* contém um `ArrayList` do objeto do tipo `Vacina` e irá ser recebida através do serviço `REST`.
- `Main`
  - `startDB`: Este método lê de um ficheiro de propriedades as informações necessárias para se conectar à base de dados;
  - `main`: É feita a conexão do serviço `REST`, mostra os comando a disponíveis e consoante o `input` do utilizador, realiza a funcionalidade descrita;
  - `getCentros`: Devolve uma `ArrayList` com o nome de todos os centros existentes;
  - `printCentros`: Mostra o nome de todos os centros;
  - `centroExiste`: Dado um centro, verifica se este existe;
  - `registarCentro`: Dado um nome do centro e um número máximo de vacinas, verifica se este já existe, caso não exista, regista um novo centro;
  - `printMenu`: Mostra comandos relativamente ao menu inicial;
  - `entrarCentro`: Dado um nome de um centro faz o *login*, cria uma instância `Centro` e mostra os comando relativamente às funcionalidades de um centro;

- `PostgresConnector`: Recebe os dados referentes à base dados e faz a sua ligação.
- `Vacina`: Esta *class* tem o nome de um centro e o número de vacinas que este irá receber.

2.4 DGS

O centro de vacinação foi implementado com `Gradle`. Existe uma conexão com a base dados `postgres`, a `Sheet` do cliente e o serviço `REST`. As seguintes `class` foram implementadas com os seguintes métodos para realizar as funcionalidades propostas:

- `CidadaoCentro`: Esta *class* associa os dados do cidadão aos dados do centro para auxiliar na distribuição das vacinas pelos centros
- `ListaVacinas`: Esta *class* contém um `ArrayList` do objeto do tipo `Vacina` e irá ser enviada através do serviço `REST`.
- `Main`
  - `putVacinas_XML`: Este método vai fazer o `put` da `listaVacinas` para que os centros possam aceder através do `get`;
  - `startDB`: Este método lê de um ficheiro de propriedades as informações necessárias para se conectar à base de dados;
  - `main`: É feita a conexão do serviço `REST`, mostra os comando a disponíveis e consoante o `input` do utilizador, realiza a funcionalidade descrita;
  - `listarVacinas`: Recebido um data, mostra o número total vacinados por centro;
  - `getCidadaosFromSheet`: Atualiza a base dados `postgres` com os novos cidadãos registados pela *Web*;
  - `distribuirVacinas`: Dado um número de vacinas, obtêm-se os cidadãos que escolheram ser vacinados no presente dia, e organizam-se por ordem decrescente, por idade. Esta lista é percorrida, adicionando, sempre que possível, uma vacina ao centro onde o cidadão mais velho quer ser vacinado;
  - `adicionaVacina`: Dado um objeto `cidadaoCentro`, verifica se o centro pode receber mais vacinas;

  - `getArrayVacinas`: Inicializa uma `ListaVacinas` com o nome de todos os centros e um 0(zero), significa que cada centro recebe 0(zero) vacinas, sendo posteriormente incrementado no `distribuirVacinas`;
  - `printMenuDGS`: mostra os comando disponíveis.

- `PostgresConnector`: Recebe os dados referentes à base dados e faz a sua ligação.
- `SheetsConnection`: Faz a conexão com a `Google Sheets`.
- `Vacina`: Esta *class* tem o nome de um centro e o número de vacinas que este irá receber.

3 Comandos

3.1 Criar Base de Dados

Os seguintes comandos são referentes à criação das tabelas da base de dados:

```
Create table centros( nomeCentro varchar(50) primary key,
nMaxVac integer);

Create table cidadao(nome varchar(50), idade integer, email
varchar(50), codigo varchar(10) primary key);

Create table fila_vacinacao(codigo varchar(10) primary key,
nomeCentro varchar(50), data date, foreign key (codigo)
references cidadao, foreign key (nomeCentro) references
centros);
```

```
Create table vacinado(codigo varchar(10) primary key,
nomeCentro varchar(50), data date, foreign key (codigo)
references cidadao, foreign key (nomeCentro) references
centros);
```

3.2 Centro de Vacinação / DGS

```
$ ./gradlew run --console=plain
```

3.3 REST

```
$ mvn compile && mvn exec:java
```

3.4 Cidadão

Link utilizado pelo cidadão (carregue na frase)

Link do Google Apps Script (carregue na frase)

Link do Google Sheets (carregue na frase)

4 Dificuldades

O presente trabalho demonstrou ser bastante mais complexo do que o inicialmente esperado. Houve especialmente, uma grande dificuldade na conexão entre os programas implementados em `Java` e os serviços `Google`. Embora possa ter sido mais fácil implementar o módulo cidadão em `Java`, preferimos e achamos mais intuitivo que o cidadão interaja com a aplicação via `WebApp`.

Existiu também alguma dificuldade na gerência de dependências, visto termos utilizado o `Graddle`, visto termos encontrado mais documentação e nas aulas práticas termos utilizado maioritariamente `Maven`.

5 Conclusão

Após a realização deste trabalho, conseguimos verificar a importância e a praticabilidade de usar REST e `Google API` e uma base dados persistente.

Acreditamos que o nosso programa dispõe de todas as funcionalidades descritas no enunciado e que muito destes conhecimentos serão de extrema importância para o nosso futuro como Engenheiros Informáticos.

Por fim, consideramos que este trabalho se encontra bem sintetizado, organizado e simples, demonstrando objetivamente os conceitos e conteúdos interiorizados no decorrer das aulas.