UNIVERSIDADE DE ÉVORA
ESCOLA DE CIÊNCIAS E TECNOLOGIA
DEPARTAMENTO DE INFORMÁTICA

2º Trabalho Prático - Room Rent

Sistemas Distribuídos

André Baião 48092
Gonçalo Barradas 48402

Docente: José Saias

1 Introdução

Este trabalho tem como objetivo implementar um sistema com uma arquitetura distribuída para um serviço de arrendamento de quartos.

Com o objetivo de não depender de uma tecnologia especifica foi utilizada a arquitetura `REST`.

`REST` é uma arquitetura que permite que o transporte dos objetos seja realizado por pedidos `HTTP`. Estes pedidos são efetuados através de métodos, dos quais, `POST`, `GET`, `PUT`, etc. Ajuda a separar responsabilidades entre a interface do cliente e o armazenamento de dados. Esta arquitetura é simples e fornece o acesso de um cliente aos métodos e recursos de uma determinada aplicação, onde os `URI` ou `IDs` globais auxiliam na identificação do recurso pretendido.

A comunicação entre o servidor e o cliente utiliza um formato intermediário universal de representação dos dados (`JSON`), de maneira a lidar com as possíveis diferenças entre as plataformas. `JSON` é uma representação de dados em alternativa ao `XML`. É um formato compacto, independente de uma linguagem e mais legível para o ser humano.

Para o bom funcionamento deste modelo é necessária a utilização de web `frameworks`. Um web `framework` é um recurso de desenvolvimento de software que facilita o desenvolvimento, manutenção, implementação e operações sobre aplicações. O `Framework` abordado neste trabalho é o `Spring Boot`, uma vez que o lado do servidor está implementado em `Java`.

O `Spring Boot` é um `framework` de `OpenSource` e foi desenvolvido para simplificar as configurações de um projeto permitindo aos `developers` abstraírem-se da configuração da aplicação. As suas dependências necessitam de ser incluídas no ficheiro de configuração do projeto.

De forma a implementar um serviço de `publish/subscribe` em que um cliente pode optar por ser notificado das mensagens recebidas do seu anuncio, foi utilizado um protocolo para comunicação assíncrona, o `MQTT`. Este é uma norma baseada em mensagens leves e eficiente.

Autenticação e autorização são dois pontos importantes para a gestão e segurança para uma aplicação. Neste trabalho o serviço de segurança utilizado foi um `token`, de forma a identificar o cliente de gestão, garantindo assim o acesso privilegiado das funcionalidades de gestão.

O armazenamento persistente de dados é realizado através de `PostgreSQL` para uma base de dados.

2 Desenvolvimento

A primeira tarefa foi o desenvolvimento da classe do servidor `REST`. Esta é responsável por abrir a ligação e configuração da segurança.

Foram criadas classes para representar os objetos `Anuncio`, `Mensagem` e `User`.

Na pasta `controller` situam-se as classes que implementam todos os recursos/métodos que o servidor disponibiliza para a aplicação e a classe responsável pela conexão com a base de dados.

A implementação dos clientes foi baseada no trabalho anterior. No entanto foi necessário proceder à adaptação dos códigos para trabalharem com um servidor `REST`, uma vez que no trabalho anterior foi utilizada a tecnologia `Java RMI`.

Para realizar esta adaptação procedeu-se à troca dos acessos ao objeto remoto por pedidos `HTTP`, sendo necessária a criação de objetos `JSON` para enviar os argumentos nos caso do método `HTTP` ser `POST`. O método `GET` envia os argumentos através do `URI`.

Procedeu-se a vários testes antes de desenvolver os pontos extra solicitados pelo professor. Nestes testes foram descobertos alguns erros, erros estes que serão mencionados mais à frente.

Seguiu-se então à implementação do serviço de `publish/subscribe` utilizando o protocolo `MQTT`. As classes `Publisher` e `Subscriber` são as responsáveis pela implementação deste protocolo.

A segurança foi implementada a partir de uma funcionalidade do `Spring Boot`, que permite implementar autenticação e autorização. Foi necessário a utilização de uma classe para gerar o `token`.

3 Discussão de Resultados

Foram detetados alguns erros durante a fase de testes à aplicação. Apesar de muitos serem erros de sintaxe e de fácil resolução, um deles estava relacionado com um dos métodos dos pedidos `HTTP`. Nos recursos que utilizavam o método `POST`, o servidor não conseguia receber os dados do pedido de modo a processá-los. Após alguma pesquisa e de alguns testes percebeu-se que o método `POST` não aceita valores dos tipos `int`, `String`, etc. A solução foi enviar um objeto, em vez dos tipos primitivos.

4 Instruções

Optamos por separar o servidor do cliente, logo o código fonte do servidor encontra-se na pasta `src/rest` e o código fonte dos clientes em `src/restClient`.

Começamos por verificar se as credenciais de acesso à base de dados estão corretas no ficheiro `application.properties` na pasta `resources` do servidor.
Para iniciar o servidor temos de estar na pasta do servidor e executar os seguintes comandos:

```
$ mvn compile
$ mvn spring-boot:run
```

Para iniciar os clientes é necessário estar na pasta dos clientes e executar os seguintes comandos:

- Cliente Geral:

```
$ mvn compile
$ mvn exec:java -Dexec.mainClass=sd.rentRoom.client.Client
  -Dexec.args="<host> <port> <userName>"
```

- Cliente Gestão:

```
$ mvn compile
$ mvn exec:java -Dexec.mainClass=sd.rentRoom.client.ClienteGestao
  -Dexec.args="<host> <port>"
```

5 Conclusão

Sendo a informação cada vez mais digital, a necessidade de utilização deste género de aplicações é cada vez maior. Devido as diferenças nas plataformas e na implementação das mesmas torna-se essencial a criação de serviços web que permitam facilitar a comunicação entre as mesmas, mantendo os requisitos de uma aplicação web robusta e real.

O `REST` é um `webservice` que permite a comunicação utilizando uma única representação de dados intermediária. Esta arquitetura tem como vantagem a capacidade de suportar aplicações implementadas em diferentes linguagens. Podemos ainda concluir que o `maven` é bastante útil na construção e gestão das dependências de um projeto.

Este trabalho foi importante para o nosso percurso académico, dando-nos ferramentas que poderão ser bastante importante no nosso futuro académico e profissional.

Referências

[Saias, 2022] Saias, J. (2022). Aulas de sistemas distribuídos. *in Universidade de Évora*.