Instituto Superior
de Engenharia
Politécnico de Coimbra

Programação Distribuída

Meta 2

Autores

João Almas - nº2021138417

Vicente Cunha - nº2022139273

Professor

José Marinho

Daniel Moreira

Ramo de Desenvolvimento de Aplicações

Licenciatura em Engenharia Informática

Instituto Superior de Engenharia de Coimbra

Instituto Politécnico de Coimbra

Data: 16 de Dezembro de 2024

Resumo

Este projeto consiste em três componentes principais e guiam-se, resumidamente, por esta arquitetura:

- Cliente (Original)
  - Conecta-se ao servidor e manda os seus pedidos com base na UI.
  - Envia os seus pedidos à api do servidor. Depois analisa as respostas.
- Servidor
  - Conecta-se à base de dados.
  - Quanto à parte do *Springboot*, lida com os clientes, atua na base de dados sobre as suas mudanças, e devolve aos clientes o resultado das suas operações.
  - Quanto à parte da RMI, implementa as funções que os clientes rmi podem chamar, fazendo uma query à base de dados.
  - Ainda sobre RMI, quando um utilizador se regista, faz login, insere ou apaga uma despesa, notifica os clientes rmi que se registaram quando iniciaram o seu próprio programa.
- Cliente RMI
  - Quando inicia, subscreve-se como observer para ser notificado quando um desses eventos acontecer.
  - Lê input do utilizador para executar os comandos que executam os métodos no servidor de listar utilizadores e listar grupos.

Capítulo 1

Cliente

1.1 Análise do Cliente

Esta é a sequência geral do programa.

- Constantemente está à espera do *input* do utilizador para executar os comandos.
- Simultaneamente, enquanto *observer*, será notificado das alterações e mostrá-las-à no ecrã.

Capítulo 2

Servidor

2.1 Análise do Servidor

Esta é a descrição geral do programa

- *Endpoints* feitos no *SpringBoot*:
Método	URI	Operação	Descrição	Tipo de Auth	Request Body	Response Body
POST	/api/auth/register	Criar	Registar um novo utilizador	None	nome, telefone, email e password	Vazio
POST	/api/auth/login	Criar	Autenticar um utilizador	Basic	Vazio	Token JWT
GET	/api/groups	Ler	Receber a lista de grupos onde o user está	Bearer	Vazio	Array de grupos
GET	/api/groups/{group}	Ler	Receber a lista de utilizadores do grupo	Bearer	Vazio	Array de Users
POST	/api/groups/{group}/expenses	Criar	Adicionar uma despesa ao grupo group	Bearer	Despesa nova	Vazio
GET	/api/groups/{group}/expenses	Ler	Receber a lista de despesas no grupo group	Bearer	Vazio	Array de Despesas
DELETE	/api/groups/{group}/expense_id	Apagar	Apagar uma despesa do grupo group	Bearer	Vazio	Vazio

- Métodos disponíveis em rmi://ip-do-servidor:porto/get-app-info:
  - Listar utilizadores
  - Listar grupos
- Métodos disponíveis em rmi://ip-do-servidor:porto/update_server:
  - Adicionar um observer (neste caso o cliente rmi adiciona-se
  - Notificar os observers todos de uma mensagem

Capítulo 3

Cliente RMI

3.1 Análise do Cliente RMI

- O cliente rmi começa por criar uma implementacao de *ObserverClient* e subscrever essa implementação ao servidor.
- A partir daí, cada vez que ocorrer um evento que o notifique, ele vai mostrar a mensagem.
- Depois, o cliente vai lendo o input do utilizador e, se for o caso, chama os métodos remotos do servidor e mostra-os, também, na consola.

Capítulo 4

Conclusão

Este projeto de Programação Distribuída mostra a aplicação prática de conceitos fundamentais da comunicação por *HTTP* através de APIs REST com *SpringBoot* e clientes http.

Foi explorada a comunicação por *RMI*, através da conexão entre o Cliente RMI e o Servidor, onde cada vez que o cliente recebia um comando, executava os métodos do Servidor.

Programação Distribuída

2024/2025