FACULDADE DE
CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE D
COIMBRA

Licenciatura em Engenharia Informática

Relatório da Meta 2 do Projeto de Grupo de Sistemas Distribuídos (SD)

Bruno José Silvério da Silva, nº 2021232021 PL5

Diogo Emanuel Matos Honório, nº 2021232043 PL5

Ano Letivo 2023/2024

Índice

Introdução:

O presente trabalho prático foi realizado no âmbito da cadeira de Sistemas Distribuídos com objetivo de implementar um motor de pesquisa de páginas Web, o Googol. Para alcançar este objetivo, desenvolvemos vários programas em Java e em Html que, em conjunto, chegam ao resultado pretendido.

O relatório visa fornecer uma visão geral do que realizamos em cada um destes programas de modo a facilitar a compreensão do código.

Arquitetura de software

A arquitetura do trabalho pratico desenvolvido consiste em quatro programas Principais em Java: Downloader, Gateaway, IndexStorageBarrel e ClientController.

1. Downloader

O Downloader é responsável por obter dados da fonte desejada pelo cliente. Utilizando a biblioteca jsoup, é capaz de extrair informações de páginas web através de URLs fornecidos. Ele opera de forma independente, aguardando por pedidos em um socket TCP na porta 6000, chamada de "down.PORTClient", e enviando os resultados por multicast para o endereço 224.0.1.2, conhecido como "mc.address", na porta 4321, denominada de "mc.port". Além disso, possui uma fila de URLs para processamento, permitindo a inserção dinâmica de novos links a serem baixados.

Importante referir também que todos os nomes definidos para portas ou endereços, foram definidos num ficheiro de properties de modo a facilitar a sua configuração e melhorar a sua flexibilidade

2. Gateaway

O Gateway atua como um intermediário entre o cliente e os demais componentes do sistema. Ele recebe pedidos dos clientes, como login, registo, pesquisas e inserção de novos URLs. Para a inserção de novos URLs, comunica-se com o Downloader via TCP, utilizando um socket temporário para transmitir a informação para a fila de URLs. As

demais funcionalidades são processadas nos barrels (IndexStorageBarrel), distribuindo tarefas de forma aleatória e utilizando RMI para comunicação entre os componentes, a comunicação RMI é feita através da porta 5001, definida como “rmi.gateaway” no ficheiro “config.properties”.

3. IndexStorageBarrel

O IndexStorageBarrel é responsável por armazenar e gerenciar todos os dados recebidos pelo Downloader através de multicast, usando os mesmos endereços e portas usados anteriormente, na qual este está sempre à escuta. Ele fornece funcionalidades de pesquisa e de armazenamento eficiente dos dados em três ficheiros diferentes: login.txt, urls.txt e words.txt, definidos também no ficheiro de properties.

4. ClientController

O ClientController funciona como MVC, gerenciando assim várias operações de cliente. Ele inclui mapeamentos de endpoints HTTP, estes com a funcionalidade de exibir e processar as páginas HTML de login, register, menu, links, search e url.

As principais funcionalidades do ClientController inclui em operações de login ou registo, usar ligações RMI com o Gateaway para verificar se as credenciais estão corretas, e redirecionar o utilizador para a página HTML apropriada. Assim como quando se deseja fazer uma pesquisa de um link ou de um termo é usado um serviço RMI para chamar as funções de pesquisa do Gateaway e deste modo obter os resultados esperados.

Componente Multicast

A componente multicast desempenha um papel crucial na arquitetura de software desenvolvida, especialmente no contexto do Downloader e do IndexStorageBarrel. Multicast é uma técnica de comunicação em rede que permite que um único pacote de dados seja enviado para vários destinos simultaneamente, reduzindo a carga na rede e otimizando a distribuição de informações.

No âmbito do Downloader, a componente multicast é utilizada para enviar os dados baixados para o IndexStorageBarrel e outros componentes interessados. Após o Downloader obter os dados da fonte desejada, ele os envia por multicast para o endereço específico de grupo (por exemplo, 224.0.1.2) na porta designada (por exemplo, 4321). Essa

abordagem permite que o IndexStorageBarrel receba os dados de forma eficiente e simultânea, garantindo uma comunicação rápida e assíncrona entre os diferentes elementos do sistema.

Por outro lado, no IndexStorageBarrel, a componente multicast é fundamental para receber os dados enviados pelo Downloader. O IndexStorageBarrel está constantemente à escuta do endereço de multicast e da porta designada, aguardando a chegada de novos dados. Assim que os dados são recebidos, o IndexStorageBarrel os processa e os armazena conforme necessário nos arquivos correspondentes.

Componente RMI

A tecnologia de RMI desempenha um papel essencial na comunicação entre os diferentes componentes do sistema, permitindo a invocação de métodos 5 em objetos remotos de forma transparente. Neste projeto, o RMI é utilizado principalmente para a comunicação entre o Cliente, o Gateway e o IndexStorageBarrel, facilitando a interação do usuário final com o sistema.

Interfaces RMI:

1. RMI Gateaway

A interface RMIGateway é essencial para a interação entre os vários componentes do sistema, facilitando a comunicação com o Gateway e permitindo a execução de diversas funcionalidades por parte dos usuários

check_login(String username, String password) – Permite que o usuário aceda ao sistema.

check_register(String username, String password) – Permite que o usuário se registe no sistema.

send_url(String url) – Permite que o usuário envie para o Downloader urls a serem processados.

search(String words) – Executa a pesquisa por um termo e devolve os resultados.

searchLink(String link) – Realiza a pesquisa por links associados ao pretendido e retorna os resultados obtidos.

addBarrel(RMIIndexStorageBarrel barrel) – Adiciona um novo barrel a lista de barrels ativos.

admin_subscribe(RMIClient admin) – Permite que o usuário entre na lista de clientes que iram receber atualizacoes em tempo real do sistema.

admin_unsubscribe(RMIClient admin) – Permite que o usuário saia da lista anteriormente referida.

showCurrentBarrel() e **getTop10KeysAsString()** – Retorna varias informações relativas ao estado do sistema.

2. RMI IndexStorageBarrel

A interface RMIIndexStorageBarrel permite que o Gateway execute suas funcionalidades e aceda aos dados armazenados nos barrels:

getBarrelReg() – Retorna o nome desse barrel (exemplo: barrel65).

searchURL(String url) – Funcao que executa a pesquisa através de um url de forma a obter links associados ao mesmo.

searchWord(String term) – Metodo que devolve as paginas onde um certo termo esteja presente.

login(String username, String password) e **register(String username, String password)** – Metodos essenciais para efutuar o registo e login do usuário.

Páginas HTML

No presente trabalho, desenvolvemos páginas dedicadas para cada operação existente no ClientController. Isso significa que criamos uma página específica para Registro, outra para Login e uma página de Menu que contém links para as páginas de operações de pesquisa, como a pesquisa de termos e páginas associadas. Além disso, incluímos uma página que envia os URLs para a Queue.

Integração Spring Boot

Para a integração do Spring Boot no nosso projeto, adotamos a arquitetura MVC (Model-View-Controller), essencial para organizar e gerenciar eficientemente as diversas operações do sistema. A estrutura do projeto baseia-se nos três componentes principais do padrão MVC: Model (Modelo), View (Visão) e Controller (Controlador).

Os modelos representam os dados da aplicação e são implementados como classes Java. Para este efeito, foram usados os programas desenvolvidos na meta 1 como base para a componente modelo do programa.

As páginas HTML, desenvolvidas utilizando Thymeleaf, constituem a camada de visualização. Elas permitem a interação do usuário com a aplicação através de uma interface web amigável. Exemplos de páginas HTML incluem as páginas de registo, login, menu e páginas de pesquisa. Por exemplo, a página de login apresenta um formulário onde o usuário pode inserir seu nome de usuário e senha. Esta página é renderizada pelo Thymeleaf, que insere dinamicamente os dados necessários e configura a ação do formulário para enviar as informações para o endpoint apropriado no controlador.

O controlador, implementado na classe ClientController.java, gerencia as requisições HTTP, define os endpoints e coordena a lógica de navegação entre as páginas HTML e os modelos. Ele atua como intermediário entre a visão e o modelo, processando as entradas dos usuários, realizando as operações de negócio necessárias e retornando as respostas apropriadas. Por exemplo, quando um usuário envia o formulário de login, o controlador recebe a requisição, valida as credenciais do usuário através de um serviço RMI que se comunica com o Gateway, e redireciona o usuário para a página de menu se as credenciais forem válidas, ou de volta para a página de login com uma mensagem de erro se não forem.

Além disso, foi implementado o suporte a HTTPS no nosso servidor Spring Boot, garantindo uma comunicação segura entre o servidor e os clientes. Para configurar o HTTPS, geramos um keystore utilizando o comando keytool e configuramos as propriedades do servidor no arquivo application.properties. Isso permite que as páginas possam ser abertas em outros computadores ou dispositivos móveis, garantindo maior segurança e confiabilidade na transmissão de dados.

Para obter os detalhes relativos à instalação e execução, encontra-se tudo esclarecido no ficheiro README.md.

Nota

Embora os testes estejam ausentes no código, é importante ressaltar que todas as componentes foram testadas manualmente para garantir a integração e o funcionamento adequado do programa. Esses testes manuais foram conduzidos de forma cuidadosa e abrangente, abordando diversos cenários e casos de uso para validar o comportamento do sistema como um todo.

Conclusão

Com a realização deste trabalho adquirimos novos conhecimentos ligados ao ramo da programação, relacionado com a cadeira de Sistemas Distribuídos. Ao desenvolver o motor de busca Googol e sua arquitetura de software, adquirimos um entendimento mais profundo sobre os conceitos fundamentais de comunicação entre componentes distribuídos e implementação de sistemas distribuídos