Faculdade de Ciências e Tecnologia da Universidade de Coimbra

Departamento de Engenharia Informática

Licenciatura em Engenharia Informática

Sistemas Distribuídos

Relatório do Projeto

------Googol------

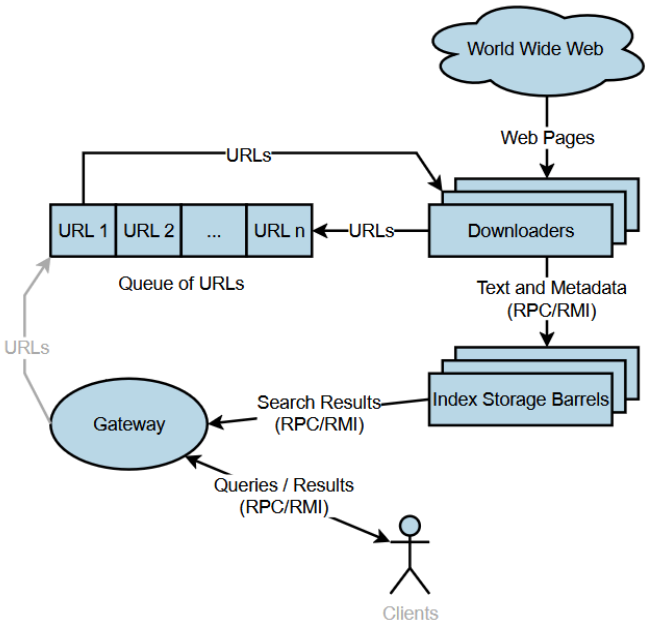
Alunos:

João Novais Saturnino de Matos *nº2021222748*

Bernardo Félix Pedro *nº2021231014*

Índice

Arquitetura de software



Na realização deste trabalho, desenvolvemos 5 classes e 5 interfaces para o backend. A classe **`RMIClient.java`** apresenta o menu do cliente e comunica com a `Gateway` por uma classe intermédia (**`RMIClient.java`**), fazendo pedidos. Ao chamar métodos da classe **`RMIGateway.java`**, obtém-se os resultados pretendidos respondendo ao cliente. A `Gateway` comunica também por RMI com a queue no caso de haver pedidos de indexação de URLs e com **`RMIIndexStorageBarrel.java`** caso seja necessário realizar pesquisas ou obter informações dos `Barrels`. O `Downloader` obtém URLs da queue (que é criada na classe **`RMIGateway.java`**) também por RMI e, após fazer a pesquisa dos mesmos, envia os resultados por reliable multicast através de RMI, os quais guardam a informação que recebem em ficheiro json individual para cada um.

Detalhes do funcionamento da replicação

A replicação de dados entre os `Barrels` é realizada por meio de um reliable multicast implementado sobre RMI. Esse método garante que as mensagens sejam entregues a todos os `Barrels` de forma confiável, evitando perdas de pacotes e garantindo a consistência dos dados. O processo ocorre da seguinte maneira:

1. O `Downloader` processa os URLs obtidos e extrai as informações necessárias.
2. Cada `Downloader` comunica inicialmente com um único `Barrel`, enviando os dados processados.
3. Se houver falha na transmissão para o `Barrel` inicial, o `Downloader` tenta novamente com outro `Barrel` até que a transmissão seja bem-sucedida.
4. Uma vez que um `Barrel` recebe os dados corretamente, ele se encarrega de compartilhar essas informações com os demais `Barrels`, garantindo que todos contenham a mesma informação.
5. Esse mecanismo permite que a redundância e a confiabilidade sejam mantidas dentro do sistema, sem a necessidade de que cada `Downloader` envie os dados diretamente para todos os `Barrels`.
6. Também cada `Barrel` ao ser criado se atualiza com uma `Barrel` já existente.

Detalhes do funcionamento da componente RMI

O sistema utiliza **RMI (Remote Method Invocation)** para a comunicação entre os seus diferentes componentes, permitindo a chamada de métodos remotos entre processos distribuídos. O funcionamento é baseado nas seguintes etapas:

- **Definição das Interfaces RMI:** Cada serviço exposto remotamente é definido através de uma interface que estende `Remote`. Esta interface declara os métodos que podem ser chamados remotamente, especificando que podem lançar `RemoteException`.

- **Implementação dos Servidores RMI:** As classes servidoras implementam as interfaces RMI e fornecem a lógica dos métodos remotos.

- **Uso pelos Clientes RMI:** Os clientes obtêm referências dos objetos remotos através do RMI Registry e invocam os métodos remotamente.

Função de cada Interface RMI no sistema:

1. **`RMIGatewayClientInterface.java`:**
   Responsável por expor os serviços da Gateway para o cliente, como receção de pedidos e envio de respostas.

2. **`RMIGatewayDownloaderInterface.java`:**
   Permite a comunicação entre a Gateway e os Downloaders, gerindo o envio de URLs para serem processados.

3. **`RMIGatewayIBSDownloader.java`:**
   Interface utilizada para gerir a interação entre a Gateway e os Barrels durante o processo de indexação de URLs.

4. **`RMIIndexStorageBarrel.java`:**
   Interface para interação entre os Downloaders e os Barrels, facilitando o envio de dados processados e a sincronização dos mesmos entre os Barrels.

Distribuição de tarefas pelos elementos do grupo

Membro	Tarefa
Bernardo Pedro	Desenvolvimento da Gateway e da componente RPC/RMI dos Barrels
João Matos	Desenvolvimento dos Downloaders e da componente multicast fiável dos Barrels

Testes de software

Teste	Pass	Fail
Indexar um URL	O URL é adicionado à URLQueue que depois é processada.	Se a gateway não estiver com a ligação correta, avisará ao cliente que o servidor está desligado e tentará conexão de 5 em 5 segundos por 5 vezes e depois pergunta se quer continuar tentando.
Listar páginas ligadas à URL.	Mostra as páginas que estão ligadas a esse URL	Se não houver nenhuma página ligada diz que não existe páginas ligadas a esse URL
Pesquisar termo ou termos	Pesquisa qualquer termo e é retornada uma lista de urls que contém esse termo em sua página.	Se a ligação ao barrel estiver comprometida, não é retornado nenhuma url ao cliente. (Não mandará mensagem de erro.)
Ordenação por relevância	A pesquisa por termos e páginas ligadas são organizadas pela relevância. (Número de urls que contém essa mesma url).	
Recuperação do estado de um barrel.	O barrel ao voltar verifica se existe um ficheiro json onde tinha guardado a informação antes de ser desligado e se tiver volta a guardar as informações existentes nele	Se não conseguir ler ele pergunta a outros barrels pela informação disponível no momento.
Carregar e salvar em disco.	Os dados dos barrels são guardados em ficheiros json individuais para cada um	Se não encontrar nenhum ficheiro ele apenas procura um barrel existente para sincronizar.
Página de Administração		Não implementado.