Googol

Henrique Oliveira, nº2022211169
Pedro Pires, nº2022247126

1 2 9 0

FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE D COIMBRA

dei departamento de engenharia informática

Arquitetura Web

A arquitetura web do Googol é composta por diversos componentes distribuídos que colaboram para permitir a indexação e pesquisa eficiente das páginas tudo numa interface web. Os principais elementos da arquitetura web são:

- Messaging Controller:

  A classe MessagingController representa o componente controlador (Controller) da aplicação Spring Boot responsável por gerenciar interações entre o cliente e os serviços back-end, como o servidor RMI e APIs externas. Ela atua como ponto de entrada para requisições HTTP e WebSocket, orquestrando chamadas a serviços remotos, análise de texto via IA e operações de pesquisa.

- WebSocketConfig:

  A classe WebSocketConfig é um componente essencial da arquitetura de comunicação em tempo real da aplicação, responsável por configurar o suporte a WebSockets com STOMP (Streaming Text Oriented Messaging Protocol) no Spring Framework.

- ChatApp:

  A classe ChatApp é o ponto de entrada principal da aplicação, desempenhando um papel fundamental na inicialização do contexto Spring Boot e no arranque de todos os componentes da infraestrutura da aplicação distribuída.

- Message:

  A classe Message é uma estrutura de dados imutável utilizada para transportar mensagens entre o cliente e o servidor. É um elemento simples, mas essencial na troca de informações via WebSocket e REST, servindo como objeto de transferência.

- App.js:

  O ficheiro app.js constitui o componente central da lógica de apresentação (frontend) da aplicação, sendo responsável por gerir a interação entre o utilizador e o servidor. Este módulo estabelece comunicação bidirecional com o backend através de requisições HTTP REST via fetch. Desta forma, permite o acesso a funcionalidades assíncronas de consulta e processamento, assegurando uma experiência de utilizador dinâmica e responsiva.

- Estatisticas.js:

  No lado das estatisticas, é utilizada a biblioteca SockJS combinada com STOMP para estabelecer a conexão WebSocket com o backend Spring Boot. Após a conexão, o cliente subscreve-se aos tópicos /topic/barrels e /topic/top10, permitindo receber atualizações em tempo real sobre o

estado dos barris e o top 10 de pesquisas. Quando os dados chegam, a interface é atualizada dinamicamente. Além disso, logo após a conexão, o cliente envia mensagens para os endpoints /app/top10 e /app/listar-barrels para obter os dados iniciais.

- Index.html:

  O ficheiro index.html define a interface principal da aplicação “Googol” Inclui uma barra de navegação com opções como envio de mensagens, consulta ao Hacker News, estatísticas e links relacionados. As mensagens são exibidas numa tabela com suporte a paginação. A interação em tempo real com o backend é feita via WebSockets (SockJS + STOMP) e HTTP, enquanto a lógica de funcionamento é gerida pelo ficheiro app.js.

- Estatisticas.html:

  O ficheiro estatisticas.html apresenta uma página dedicada à visualização de estatísticas em tempo real da aplicação “Googol”. Utiliza Bootstrap para o layout e integração com WebSockets via SockJS e STOMP para receber atualizações automáticas. Mostra as estatísticas numa tabela centralizada, com navegação simples e ligação à página principal. O comportamento dinâmico é gerido pelo script app.js.

- MessagingCallBack:

  A interface MessagingCallBack define um método remoto enviarEstatisticasParaClientes, utilizado pelo servidor RMI para notificar o controlador Spring Boot (MessagingController) sempre que existem novas estatísticas a serem enviadas aos clientes via WebSocket. Esta abordagem permite comunicação reativa do servidor para o frontend, garantindo a atualização em tempo real das informações.

Detalhes sobre a integração do Spring Boot com o Servidor RMI da primeira meta

Para a integração da meta 2 com a meta 1 começamos por anotar a classe MessagingController por @RestController (uma junção de duas anotações, a @Controller e a @ResponseBody) o que marca o MessagingController como controlador do springboot e faz com que os métodos retornem dados diretamente no corpo da resposta, depois estabelecemos a ligação ao servidor RMI usando a seguinte linha de código

this.rmiServidor = (Index) LocateRegistry.*getRegistry*(ip, port).lookup(name);

Com isto permite-nos estabelecer a ponte entre RMI e Spring Boot e que a classe MessagingController use os métodos estabelecidos no IndexServer e nas outras classes (Robot e Barrel) usando o próprio IndexServer.
Com o Spring Boot e o servidor RMI interligados, usamos métodos remotos no MessagingController e um método remoto do próprio MessagingController no IndexServer de modo a conseguirmos garantir que notificávamos o Controller de atualizações para as nossas estatísticas em tempo real.

Integração de WebSockets com Spring Boot/FastAPI e RPC/RMI

A aplicação implementa comunicação bidirecional em tempo real entre cliente e servidor utilizando WebSockets com STOMP, facilitada pela biblioteca SimpMessagingTemplate. Esta funcionalidade permite que o frontend atualize dinamicamente informações como o estado atual dos barrels e o Top 10 de pesquisas, sem a necessidade de polling constante.
A configuração dos WebSockets está centralizada na classe WebSocketConfig, anotada com @Configuration e @EnableWebSocketMessageBroker. Esta classe implementa WebSocketMessageBrokerConfigurer e define:

- O prefixo /app para mensagens enviadas do cliente para o servidor;
- O prefixo /topic para mensagens enviadas do servidor para todos os clientes subscritos;
- O endpoint /my-websocket, que é o ponto de entrada para conexões WebSocket, com suporte a fallback via SockJS.

A lógica de WebSocket está implementada no controlador MessagingController, que atua tanto como:

- Controlador REST e WebSocket (com métodos anotados com @MessageMapping e @SendTo);
- Cliente RMI, comunicando-se com o servidor da Meta 1 para executar operações distribuídas.

A classe MessagingController recebe uma instância de SimpMessagingTemplate, utilizada para enviar mensagens em tempo real para os tópicos WebSocket. Por exemplo, quando um cliente envia uma mensagem para /app/listar-barrels, o controlador processa o pedido e publica a resposta no tópico /topic/barrels, ao qual o cliente está subscrito.
Além disso, o MessagingController implementa a interface MessagingCallBack, permitindo ser registrado como callback remoto no servidor RMI. Com isso, o servidor pode notificar diretamente o Spring Boot sobre atualizações — como mudanças nas estatísticas dos barrels — que, por sua vez, são repassadas em tempo real ao frontend via WebSocket.

Detalhes sobre a integração com os serviços REST

Recorremos ao uso de REST api's duas vezes no nosso código:

Hacker news: A API do Hacker News é usada para verificar se uma palavra introduzida pelo utilizador aparece nos títulos das notícias mais populares da plataforma (as chamadas *top stories*). Esta funcionalidade é particularmente útil para verificar a relevância ou atualidade de determinados termos.

Passos da integração:

- Através de um GET para https://hacker-news.firebaseio.com/v0/topstories.json, obtemos os identificadores dos artigos mais populares no momento.
- Para cada identificador, fazemos um GET para https://hacker-news.firebaseio.com/v0/item/{id}.json para obter os detalhes de cada artigo (título, URL, etc.).
- Procuramos a palavra inserida nos títulos desses artigos.
- Se houver correspondências, os respetivos URLs são enviados para o servidor RMI para serem posteriormente indexados.
- O resultado (títulos e links encontrados) é retornado ao utilizador.

Gemini Ai: A API do Gemini AI (da Google) é utilizada para gerar uma pequena explicação descritiva sobre uma palavra ou expressão fornecida pelo utilizador. Isto é útil para enriquecer a resposta com conteúdo mais informativo e gerar contexto.

Passos da integração:

- Um prompt é preparado, pedindo uma explicação objetiva sobre o termo inserido.
- O pedido é enviado por POST à API do Gemini, com um corpo JSON.
- A resposta é recebida no formato JSON contendo texto gerado pela IA.
- O texto é extraído e devolvido ao utilizador.

Para a implementação de ambos os métodos recorremos ao RestTemplate, pois considerámos ser uma solução mais simples de implementar em comparação com HttpURLConnection, oferecendo uma sintaxe mais limpa, melhor integração com o Spring, o que tornou o desenvolvimento mais rápido e legível.

Testes Realizados

Testes Realizados	Pass/Fail	Observações
Ao ser enviados links como mensagem, o programa indexa?	✔	O robot sim indexa os links pedidos para indexar.
Ao pedir para indexar links as estatísticas atualizam o tamanho do barrel em tempo real?	✔	Conseguimos ver as estatísticas do tamanho dos barrels a serem atualizadas em tempo real.
Ao pedir para pesquisar alguma palavra que causaria uma mudança no top 10, as estatísticas atualizam em tempo real?	✔	Conseguimos observar que as estatísticas do top 10 atualizam se sim em tempo real.
Ao pesquisar algo vazio para o hacker news, o programa indexa na mesma todos os links recebidos?	✘	Se o programa receber uma palavra vazia, em vez de procurar pelo hacker news todas as top stories, o programa dá logo return e pede para enviar algo para a procura.
Ao passar para a próxima página dos resultados o programa conta como se outra pesquisa foi feita?	✔	O programa conta sim, devido a fazer de novo a função que envia os resultados mas desta vez com os próximos 10.
O utilizador consegue andar para a próxima página e anterior sempre que quiser?	✔ / ✘	O utilizador pode sim andar para a próxima página e para a anterior, mas apenas se no caso de querer avançar, existir mais de 10 resultados restantes e no caso de querer retroceder, não estiver na primeira página.
A REST API de Ai manda outra análise quando mudamos de página?	✘	Não, porque o programa está limitado ao fazer só na primeira página.