UNIVERSIDADE D
COIMBRA

Licenciatura Engenharia Informática

GOOGOL:

MOTOR DE PESQUISA DE PÁGINAS WEB

Sistemas Distribuídos

2022/2023

Mariana Ferreira Sousa – 2020226346 – PL5

Nuno Alexandre Santos Vasques – 2020235483 – PL5

André Marques Florim – 2019218102 – PL5

Conteúdo

Introdução

Este relatório descreve a segunda meta do projeto Googol, que tem como foco a criação de um *frontend web* para a aplicação. O objetivo é permitir que os utilizadores acedam ao serviço a partir de qualquer dispositivo com acesso à Internet, eliminando a necessidade de instalação de *software* cliente. A interoperabilidade é um requisito fundamental, garantindo que os utilizadores da versão web tenham acesso às mesmas informações disponíveis na aplicação de ambiente de trabalho.

Para alcançar esta meta, será utilizado o servidor RMI desenvolvido na primeira etapa do projeto como base para o servidor web. A interface web deverá oferecer as mesmas funcionalidades disponíveis na meta 1, incluindo a indexação de URLs, a pesquisa de páginas e outras operações relacionadas. Além disso, é crucial que a aplicação web apresente alterações em tempo real, especialmente em relação às informações gerais do sistema, como *downloaders* e *barrels* ativos. Para atender às expectativas dos utilizadores cada vez mais exigentes, técnicas menos robustas, como meta-*refresh* e *iframes* ocultas, não serão consideradas.

Por fim, para proporcionar uma funcionalidade mais rica e integrada, a aplicação será integrada à API do *Hacker News* por meio de APIs *REST*. Isso permitirá que, durante uma pesquisa, sejam obtidos URLs das *"top stories"* do Hacker News que contenham os termos pesquisados no Googol. Além disso, os utilizadores com uma conta no Hacker News poderão solicitar ao Googol que recupere todas as suas "stories" e indexe todos os URLs associados.

Este relatório detalhará os passos necessários para alcançar os objetivos da segunda meta do projeto Googol, incluindo a arquitetura proposta, as tecnologias a serem utilizadas e as etapas de implementação.

Arquitetura do Software

Nesta fase da arquitetura do software, o objetivo principal consistiu em desenvolver um frontend web para a aplicação Googol, permitindo que os utilizadores acedam ao serviço a partir de qualquer dispositivo com acesso à Internet, sem a necessidade de instalar um software cliente dedicado. Uma das prioridades foi garantir a interoperabilidade, que foi alcançada através da ligação do servidor web ao servidor de dados utilizando o RMI (Invocação Remota de Métodos). Esta abordagem assegurou que os utilizadores da versão web tivessem acesso às mesmas informações disponíveis na aplicação desktop.

Ao criar o frontend web, procurámos proporcionar aos utilizadores uma experiência intuitiva e de fácil utilização. Os utilizadores podem aceder ao serviço Googol através de navegadores web convencionais, utilizando o protocolo HTTP para enviar pedidos ao servidor web. Estes pedidos podem incluir operações como indexação de URLs, pesquisa de páginas e outras funcionalidades disponíveis na versão desktop.

Um dos objetivos importantes foi melhorar a experiência do utilizador, tornando-a mais dinâmica e responsiva. Para isso, considerámos a utilização de tecnologias como AJAX (JavaScript e XML Assíncronos), que permite a atualização de partes específicas da página sem a necessidade de recarregar a página inteira. Isso resulta numa experiência mais fluída e reduz a necessidade de esperar pelo carregamento completo da página a cada interação.

Além disso, a comunicação em tempo real com o utilizador é uma característica essencial para tornar a aplicação mais interativa. Implementámos essa funcionalidade utilizando WebSockets, permitindo a troca de mensagens em tempo real entre o servidor web e o navegador do utilizador. Dessa forma, informações gerais do sistema, como downloaders e barris ativos, podem ser atualizadas e exibidas instantaneamente no navegador do utilizador, proporcionando uma experiência mais imersiva.

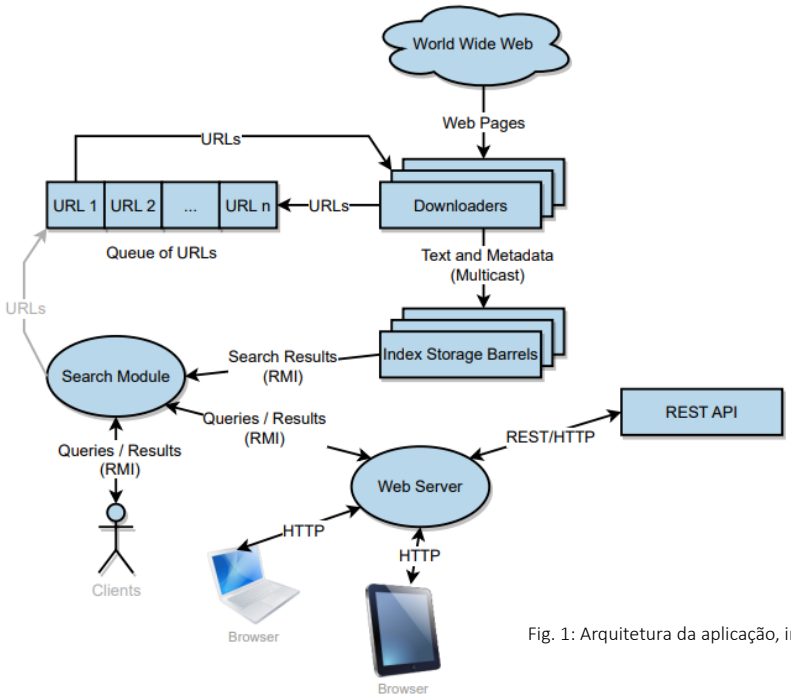


Fig. 1: Arquitetura da aplicação, incluindo os componentes da Meta 1.

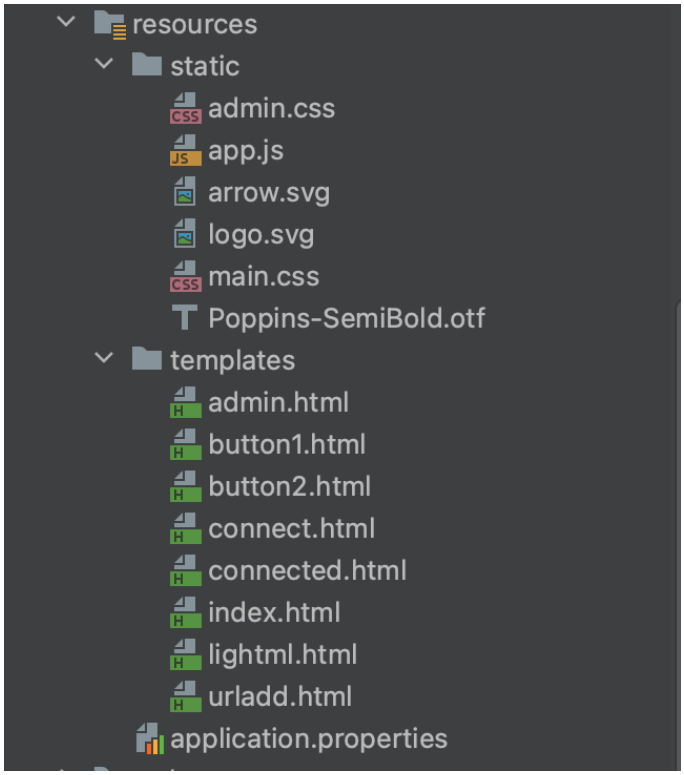
Estrutura

Decidimos, de forma a ser mais organizado, separar os ficheiros java da meta 1 e da meta 2 em packages diferentes. O package meta1 contém os ficheiros da primeira meta e o package com.example.servingwebcontent (utlizámos a aplicação de exemplo fornecida pelo Professor) contém os ficheiros da segunda meta.

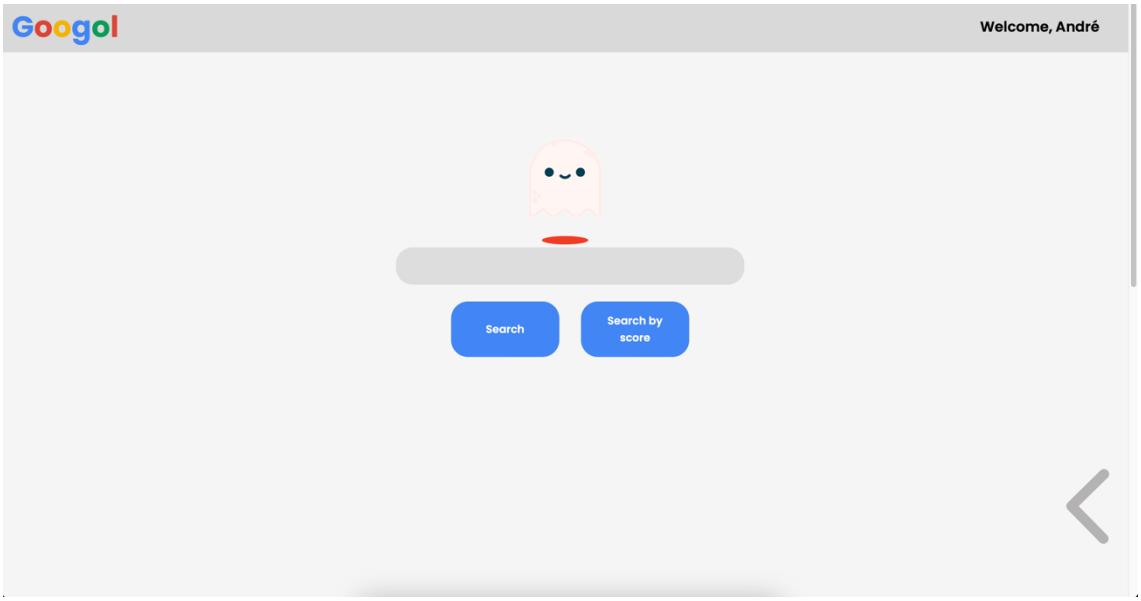
Models

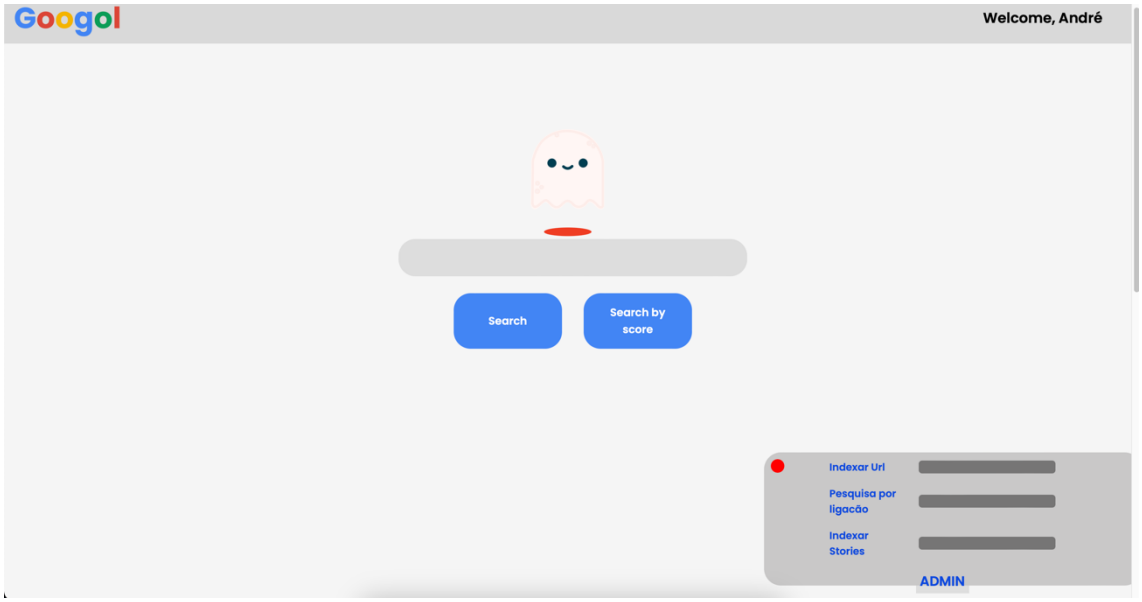
```
@GetMapping("/connect")
public String showConnectForm(Model model) {
    model.addAttribute( attributeName: "username", new UsernameForm());
    return "connect";
}
```

Views

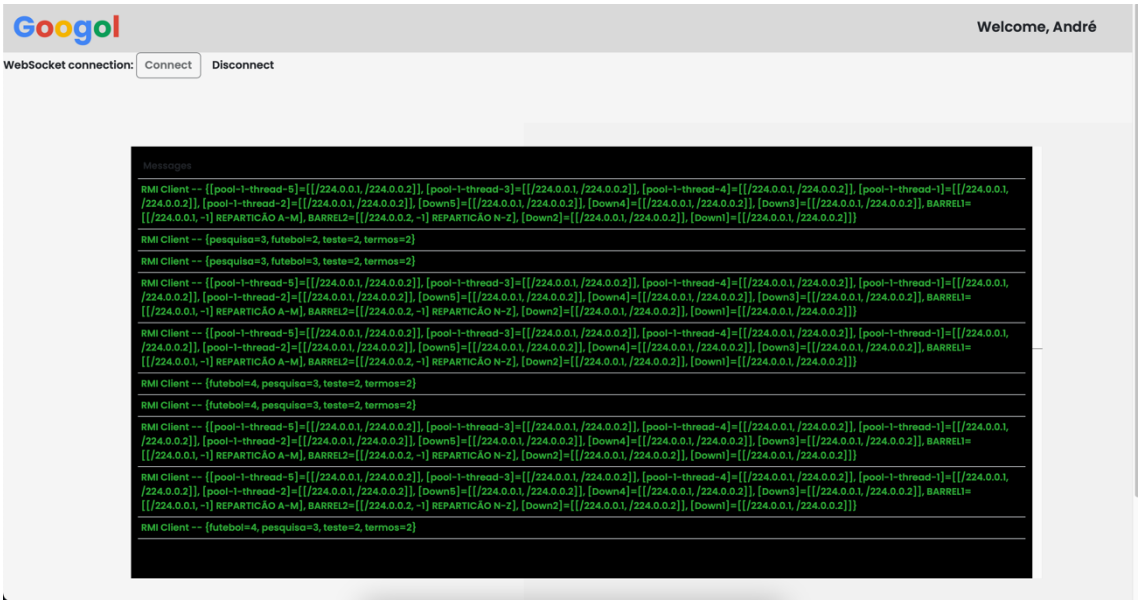



Bem-vindo ao Googol
Username
Password
Login


Googol
Welcome, André
Search
Search by score


Googol
Welcome, André
Search
Search by score
Indexar Url
Pesquisa por ligacão
Indexar Stories
ADMIN


Googol
Welcome, André
Google
google gmail imagens entrar pesquisa avançada publicidadesoluções empresariaistudo sobre a
URL!
Termos
termos de utilização do google – privacidade e termos de
URL!
Termos
termos de utilização do google – privacidade e termos de
URL!
Termos
termos de utilização do google – privacidade e termos de
URL!
Termos
termos de utilização do google – privacidade e termos de
URL!
Termos
termos de utilização do google – privacidade e termos de
URL!
Termos
termos de utilização do google – privacidade e termos de
URL!
Termos
termos de utilização do google – privacidade e termos de
URL!
Termos
termos de utilização do google – privacidade e termos de
URL!
Termos
termos de utilização do google – privacidade e termos de
URL!
Termos



Controllers

```java
@Controller
@Import(RestTemplateConfig.class)
@RequestMapping("/")
public class GreetingController {

    14 usages
    private static RMIClient rmiClient;
    7 usages
    private final SimpMessagingTemplate messagingTemplate;
    2 usages
    private final RMIclientService rmiClientService;


    no usages
    @Autowired
    public GreetingController(RMIclientService rmiClientService,SimpMessagingTemplate messagingTemplate) {...}
    1 usage
    @Autowired
    private HttpSession httpSession;


    no usages
    @PostConstruct
    public void initialize() throws IOException, NotBoundException, InterruptedException {...}
    2 usages
    Map<String, ArrayList<String>> lastFailoverMap = null;

    no usages
    @Scheduled(fixedRate = 5000)
    public  void push() throws IOException {...}
```

```java
    no usages
    @GetMapping("/connect")
    public String showConnectForm(Model model) {...}

    no usages
    @PostMapping("/connect")
    public String connect(@ModelAttribute("username") UsernameForm usernameForm, Model model) throws IOException {...}
    no usages
    @GetMapping("/IND")
    public String individualPage(@RequestParam("url") String url, Model model) throws IOException {...}

    no usages
    @GetMapping("/submit-text/1/{text}")
    public String button1Page(@PathVariable("text") String text, Model model) throws IOException {...}

    no usages
    @GetMapping("/submit-text/2/{text}")
    public String button2Page(@PathVariable("text") String text, Model model) throws IOException {...}

    no usages
    @GetMapping("/LIG")
    public String ligPage(@RequestParam("url") String url, Model model) throws IOException {...}
    no usages
    @GetMapping("/ADM")
    public String ligPage( Model model) throws IOException {...}
    no usages
    private static final Logger logger = LoggerFactory.getLogger(GreetingController.class);

    no usages
    @MessageMapping("/message")
    @SendTo("/topic/messages")
    public Message onMessage(Message message) throws InterruptedException {...}
```

Threads

Apesar de inicialmente termos utilizado uma thread na implementação da WebSocket, decidimos alterar e utilizar uma função push que envia a mensagem de 5 em 5 segundos (utilizando fixedrate).

Sockets

```
no usages
@Configuration
@EnableWebSocketMessageBroker
public class WebSocketConfig implements WebSocketMessageBrokerConfigurer  {

    no usages
    @Override
    public void configureMessageBroker(MessageBrokerRegistry config) {
        config.enableSimpleBroker( ...destinationPrefixes: "/topic");
        config.setApplicationDestinationPrefixes("/app");
    }

    no usages
    @Override
    public void registerStompEndpoints(StompEndpointRegistry registry) {
        registry.addEndpoint( ...paths: "/my-websocket").withSockJS();
    }

}
```

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

A integração do Spring Boot com o Servidor RMI envolve a configuração do servidor para expor objetos remotos e permitir a comunicação com os clientes. O Spring Boot oferece anotações e configurações que facilitam essa integração, permitindo que os objetos remotos sejam gerenciados pelo recipiente. Além disso, é necessário definir as interfaces e implementações dos objetos remotos, que devem estender as classes apropriadas do RMI. Através da injeção de dependência, os objetos remotos podem ser acessados nos componentes do Spring Boot, simplificando o desenvolvimento de aplicações distribuídas.

Detalhes sobre a programação de WebSockets e a sua integração com o servidor RMI

A programação de WebSockets envolve a criação de uma comunicação bidirecional entre o servidor e o cliente, permitindo a troca de mensagens em tempo real. No contexto da integração com o servidor RMI, os WebSockets podem ser utilizados para atualizar em tempo real informações sobre o sistema, como a lista de downloaders e barris ativos, além das pesquisas mais comuns realizadas pelos utilizadores. Dessa forma, a integração de WebSockets com o servidor RMI permite atualizar em tempo real as informações do sistema para os utilizadores, proporcionando uma experiência interativa e dinâmica.

Detalhes sobre a integração com o serviço REST

O REST é chamado através de JavaScript para ir buscar os dados à API HackerNews disponibilizada e popular o barrel com a informação correta. Os resultados são atualizados em tempo real na página.

Testes realizados a plataforma

Descrição do teste Realizado	**Passed**	**Failed**
Indexar URL Válido	X	
Indexar URL Inválido (não deve ser aceite)	No backend não é aceite (como esperado)	No entanto, é redirecionado para mesma pagina html que diz que foi adicionado com sucesso
URLs indexados recursivamente	X	

Pesquisar com uma palavra	X	
Pesquisar com várias palavras	X	
Ordenar pesquisa por importância		X
Consultar lista de páginas com ligação para uma página específica	X	
Informação sobre o sistema em tempo real	X	
Particionamento do índice	X	
Lista de Barrels ativos	X	
Top10 Pesquisas mais comuns	X	
Guardar nome do user durante a sessão	X	

Conclusão

No geral, acreditamos que alcançamos o objetivo proposto. Conseguimos implementar tudo o que foi requerido, embora tenhamos enfrentado algumas dificuldades em certos pontos. Consideramos que o trabalho prático foi bem executado no geral.

Para trabalhos futuros, seria interessante melhorar a aparência do frontend, mesmo que, neste momento, a plataforma esteja completamente funcional. Essas melhorias adicionais poderiam contribuir para uma experiência mais agradável para os utilizadores e garantir maior estabilidade e confiabilidade do sistema como um todo.