**Projeto final**



#### Computação Distribuída

Trabalho realizado por: Rafael Silva - a22107578 Henrique Franco - a22107352

Filipe Oliveira - a22109080

01/2024

# Índice

[Introdução 3](#_Toc155104621)

[Arquitetura da solução 4](#_Toc155104622)

[Operações 5](#_Toc155104623)

[Manual de instalação e configuração 6](#_Toc155104624)

[Utilização da aplicação 13](#_Toc155104625)

[Conclusão 17](#_Toc155104626)

## Introdução

O projeto proposto para a disciplina de Computação Distribuída no ano letivo 2023/2024 consiste no desenvolvimento de um serviço de reserva de sombrinhas e espreguiçadeiras em praias do Algarve.

O objetivo é criar uma aplicação distribuída, com servidores frontend e backend, utilizando comunicação RMI em java.

O sistema busca melhorar a experiência dos utilizadores ao oferecer a conveniência de reservas antecipadas.

## Arquitetura da solução

No servidor front-end os clientes vão interagir por um servidor Apache Tomcat através de um Cliente Java Webservices Soap, este cliente emprega o protocolo HTTP e acede a Web Services SOAP disponibilizados pelo servidor Tomcat. O front-end deste trabalho estará no sistema operativo Windows.

A informação das reservas está presente no servidor backend, essa informação é suportada por ficheiros de texto.

O servidor do back-end estará no sistema operativo Linux.

Uma imagem com texto, captura de ecrã, diagrama, Tipo de letra

Descrição gerada automaticamente

Figura 1-Arquitetura

**Back-end:**

No back-end vão existir funções que são chamadas pelo webservice para fazer o que o utilizador pretende.

No back-end também vai existir 1 ficheiro txt com toda a informação necessária sobre as sombras e as reservas que por sua vez será a base de dados desta arquitetura.

**Front-end:**

No front-end, o cliente emprega o protocolo HTTP e acede a Web Services SOAP disponibilizados pelo servidor Tomcat.

É possível realizar o login e o registo e também existe um ficheiro txt que guarda as informações sobre os utilizadores.

## Operações

**Reservar sombrinha:**

A reserva é feita através da escolha da praia (A, B ou C), horário e do número de pessoas estando o utilizador autenticado. As reservas decorrem das 8:00 até às 20:00 por períodos de 1 hora.

**Cancelar sombrinha:**

O cancelamento da sombrinha é feito através do ID da sombrinha estando o utilizador autenticado.

**Listar sombrinhas:**

Lista as sombrinhas não reservadas numa praia, hora proposta e estando o utilizador autenticado.

**Registar Utilizador:**

Procede ao registo de um novo utilizador utilizando o email e password.

## Manual de instalação e configuração

Primeiramente para desenvolver este projeto será necessário no servidor front-end Windows e no servidor back-end Linux.

**Linux:**

No Linux abra um terminal e instale o Java JDK com o comando “sudo apt install default-jdk”.

Após a instalação do JDK instale o GCC com o seguinte comando “sudo apt-get install gcc”.

No caso de estar a utilizar o oracle VM virtual box vá a Dispositivos > Rede > Definições de rede > Associado a: Trocar “NAT” para “apenas adaptador do hospedeiro”.

Guarde num ficheiro os ficheiros “AddServer.java”, “AddServerImpl.java”, “AddServerIntf.java” e “reservas.txt”.

A partir desta pasta abra uma consola e coloque o comando “rmiregistry”

Uma imagem com texto, captura de ecrã, software, Software de multimédia

Descrição gerada automaticamente

Figura 2-Rmi registry launch

Antes de compilar os ficheiros vai ter de alterar o “pathReservas” com o caminho onde tem guardado o ficheiro “reservas.txt” no ficheiro “AddServerImpl.java”.

Tendo já estes passos todos feitos abra um segundo terminal e escreva os seguintes comandos:

Uma imagem com texto, captura de ecrã, software, Software de multimédia

Descrição gerada automaticamente

Figura 3-Compilação e execução

Com isto vamos passar agora para o Front-end.

**Windows:**

No Windows abra um terminal e instale o Java JDK com o comando “sudo apt install default-jdk”.

Após a instalação do JDK instale o GCC com o seguinte comando “sudo apt-get install gcc”.

Posteriormente instale o Eclipse IDE for Enterprise Java and Web Developers, Apache Tomcat 9 e Apache CXF V3.6.2.

Depois de ter tudo instalado abra o Eclipse para configurar o Apache Tomcat e o Apache CXF.

Para isso vá a Window > Preferences > Server > Runtime environment > Add > Apache > Apache Tomcat V9.0 > Next

Uma imagem com texto, eletrónica, captura de ecrã, ecrã

Descrição gerada automaticamente

Figura 4-Apache Tomcat

Depois carregue no “Next” e escolha o ficheiro que instalou.

No final deverá aparecer esta janela:

Uma imagem com texto, captura de ecrã, ecrã, software

Descrição gerada automaticamente

Figura 5-Verificação da instalação da instalação do Apache Tomcat

Na mesma janela, do lado esquerdo, clique em Web Services > CXF 2.x preferences > Add > Browse e selecione o Apache CXF.

Uma imagem com texto, captura de ecrã, ecrã, software

Descrição gerada automaticamente

Figura 6-Verificação da instalação do ApacheCXF

No final deve ter a janela da Figura 6.

Agora dentro do eclipse vamos criar o webservice Soap e o cliente Soap.

Para isso vá a File > New > Dynamic webproject > nome à escolha.

Verifique se o target runtime é o Apache Tomcat V9.0

Uma imagem com texto, captura de ecrã, software, número

Descrição gerada automaticamente

Figura 7-Criar Dynamic Web Project

Agora copie os ficheiros (“AddServerIntf.java” e “MathUtility.java”) para src > main > java.

De seguida, dentro do project explorer carregue em webservices > create webservice. Depois verifique se a configuração do servidor tem como server runtime apache 9.0 e se o webservice runtime é o Apache CXF 2.x.

Uma imagem com texto, captura de ecrã, software, Ícone de computador

Descrição gerada automaticamente

Figura 8-Verificação do servidor

Depois de ter completado tudo carregue sempre em next até aparecer esta janela:

Uma imagem com texto, eletrónica, captura de ecrã, software

Descrição gerada automaticamente

Figura 9-Servidor tomcat

Carregue em finish.

A criação do servidor ainda não está completa, pois precisa de eliminar um import que foi criado com a criação do servidor. Para isso vá a source > main > webapp > WEB-INF > CXF-beans.xml

Uma imagem com texto, captura de ecrã, Tipo de letra

Descrição gerada automaticamente

Figura 10-path cxf-beans

Depois elimine o 2 import:

Uma imagem com texto, captura de ecrã, Tipo de letra, file

Descrição gerada automaticamente

Figura 11-import 2

Agora volte a fazer tudo o que fez anteriormente para a criação do servidor Apache Tomcat.

Dito isto entre neste link <http://127.0.0.1:8080/proj2/services>, mas altere o nome “proj2” pelo nome que deu ao projeto.

Depois deverá aparecer o seguinte:

Uma imagem com texto, Tipo de letra, file, captura de ecrã

Descrição gerada automaticamente

Figura 12-Mathutility servisse

Depois clique no link que está em WSDL.

Com esta nova página guarde o link da mesma.

Com isto o webservice Soap disponibilizado pelo servidor tomcat está feito. Agora vamos criar o protocolo HTTP vindo do cliente.

Agora crie um File > new > dynamic web project > nome à escolha > Finish

Para isso vá a Nome do projeto > new > other > webservices > webservice client.

Uma imagem com texto, eletrónica, captura de ecrã, ecrã

Descrição gerada automaticamente

Figura 13-Criar webservice cliente

Depois de carregar em next coloque o link que guardou anteriormente.

Uma imagem com texto, eletrónica, captura de ecrã, software

Descrição gerada automaticamente

Figura 14-link do webservice

Com isto carregue em Finish e copie o “mathconsumer.java” para dentro do src > main > java > default\_package.

De seguida vá a run as > java aplication.

De seguida deverá aparecer na consola o seguinte:

Uma imagem com texto, eletrónica, captura de ecrã, software

Descrição gerada automaticamente

Figura 15-Programa

Com isto a aplicação está pronta a ser utilizada.

## Utilização da aplicação

Para começar, podemos realizar o registo de uma conta.

Uma imagem com texto, captura de ecrã, Tipo de letra, número

Descrição gerada automaticamente

Figura 16 - Registo da conta

E, após o registo, podemos fazer o login com o mesmo registo que fizemos anteriormente.

Uma imagem com texto, captura de ecrã, Tipo de letra

Descrição gerada automaticamente

Figura 17 - Login da conta

Após isto, vamos fazer a reserva de uma sombrinha. Para isso, vamos verificar se, na praia A, no horário das 18 a sombrinha está disponível.

Uma imagem com texto, captura de ecrã, Tipo de letra, número

Descrição gerada automaticamente

Figura 18 - Verificar se a praia está disponível

Como podemos ver, a praia está disponível. Assim, vamos prosseguir para a reserva da mesma.

Uma imagem com texto, captura de ecrã, Tipo de letra, número

Descrição gerada automaticamente

Figura 19 - Reservar a praia pretendida

Agora vamos verificar se a reserva foi mesmo feita.

Uma imagem com texto, captura de ecrã, Tipo de letra, número

Descrição gerada automaticamente

Figura 20 - Verificação sobre a ocupação da sombrinha

E agora vamos cancelar a sombrinha reservada.

Uma imagem com texto, captura de ecrã, Tipo de letra, número

Descrição gerada automaticamente

Figura 21 - Cancelamento da Sombrinha

Para terminar, vamos ver se o cancelamento foi feito com sucesso.

Uma imagem com texto, captura de ecrã, Tipo de letra, número

Descrição gerada automaticamente

Figura 22 - Verificação do cancelamento

Agora, vamos fazer um exemplo em que, numa situação, um utilizador reserva uma sombrinha na praia A as 18 e, de seguida, outro utilizador faz o registo e tenta fazer o cancelamento dessa mesma sombrinha, mostrando que não é possível fazer o mesmo.

Uma imagem com texto, captura de ecrã, Tipo de letra, número

Descrição gerada automaticamenteUma imagem com texto, captura de ecrã, Tipo de letra, número

Descrição gerada automaticamente

Figura 24 - Outro utilizador faz o cancelamento da reserva

Figura 23 - Utilizador faz a reserva

## Conclusão

Com este trabalho, conseguimos perceber as dificuldades que podem existir em sistemas distrubuidos, funcionamento de java RMI e também com o funcionamento de webservice Soap.

Algumas dificuldades encontradas foi em estabelecer a ligação do webservice com o client soap.