



UNIVERSIDADE
LUSÓFONA

Computação distribuída

Projeto Marcação Consultas Clínicas

Miguel Lourenço a22202315

Vasco Pereira a22202735

2024/2025

www.ulusofona.pt

Índice

Índice	2
Lista de Figuras	3
1 Resumo.....	4
2 Introdução	5
3 Arquitectura da solução	6
4 Manual de instalação e configuração	7
4.1 Frontend.....	7
5.1 Backend	12
5.2 Inicialização dos WebServices.....	18
5.2.1 Soup.....	18
5.2.2 Rest.....	20
6 Jpanel	22
7 Guia de Utilização.....	23
7.1 Registo De Novo Usuário.....	23
7.2 Login Usuário.....	24
7.3 Marcar Consulta	25
7.3.1 Listar Consultas	28
7.3.2 Cancelar consultas.....	28
8 Conclusão	29
9 Bibliografia	30

Lista de Figuras

Figura 1	6
Figura 2	6
Figura 3	8
Figura 4	8
Figura 5	9
Figura 6	10
Figura 7 Rest	10
Figura 8 Soup	10
Figura 9	11
Figura 10	11
Figura 11	12
Figura 12	12
Figura 13	13
Figura 14	13
Figura 15 Dbiver	13
Figura 16 Azure	14
Figura 17	16
Figura 18	17
Figura 19	17
Figura 20	17
Figura 21	18
Figura 22	18
Figura 23	19
Figura 24	19
Figura 25	20
Figura 26	20
Figura 27	21
Figura 28	21
Figura 29	21
Figura 30 Registrar Usuário	23
Figura 31 Registo com sucesso	23
Figura 32 Registo Paciente	24
Figura 33 Registo com sucesso	24
Figura 34 Login	24
Figura 35 Login com sucesso	25
Figura 36 marcar consulta	25
Figura 37	25
Figura 38 marcar consulta hora errada	26
Figura 39 erro ao marcar	26
Figura 40 marcar consulta	26
Figura 41 sucesso na marcação	27
Figura 42 tentar marcar consultas a mesma hora	27
Figura 43 listar consultas	28
Figura 44 cancelar consulta	28

1 Resumo

Este projeto tem como objetivo desenvolver um sistema distribuído para a marcação de consultas clínicas em várias especialidades, distribuídas por diferentes clínicas. O sistema será implementado por 2 elementos e terá uma arquitetura composta por um servidor frontend e um servidor backend, com comunicação entre ambos através de RMI. No frontend, serão disponibilizados dois tipos de clientes: um para Web Services REST e outro para Web Services SOAP, ambos suportados por um servidor Apache Tomcat. O backend, que será executado em Linux, será responsável por armazenar as informações das consultas, utilizando uma base de dados. Entre as principais funcionalidades destacam-se: reservar consultas, cancelar consultas, listar consultas reservadas e registrar novos utilizadores.

2 Introdução

O presente relatório apresenta o desenvolvimento de um sistema distribuído para a marcação de consultas, concebido para facilitar o agendamento de diferentes especialidades médicas disponíveis em diversas clínicas. O projeto propõe uma solução tecnológica que integra diversas funcionalidades essenciais para a gestão eficiente de consultas, abrangendo desde a interação com os utilizadores até ao armazenamento e processamento seguro dos dados.

O sistema foi desenhado para suportar as operações de quatro clínicas, cada uma com as suas especialidades médicas:

- **Clínica A:** Especializada em clínica geral (10 médicos), pediatria (5 médicos), cardiologia (1 médico) e ginecologia (2 médicos).
- **Clínica B:** Oferece serviços de clínica geral (5 médicos), ginecologia (1 médico) e hematologia (1 médico).
- **Clínica C:** Focada em medicina tropical (1 médico) e neurologia (1 médico).
- **Clínica D:** Especialidades incluem oftalmologia (2 médicos), oncologia (3 médicos) e otorrinolaringologia (1 médico).

O sistema disponibiliza funcionalidades como a reserva de consultas, o cancelamento de consultas, a listagem de consultas reservadas e o registo de novos utilizadores. As consultas podem ser agendadas em intervalos de 1 hora, no horário compreendido entre as 08:00 e as 20:00.

3 Arquitectura da solução

A arquitetura do sistema foi concebida com dois componentes principais, cada um desempenhando um papel essencial no funcionamento da aplicação:

Servidor Frontend – Este componente é o ponto de interação dos utilizadores com o sistema, disponibilizando duas opções de acesso: através de clientes baseados em Web Services REST ou Web Services SOAP. Ambos são suportados por um servidor Apache Tomcat, que também é responsável pelo registo e autenticação dos utilizadores. Os dados dos utilizadores, como informações de registo e autenticação, são armazenados numa base de dados SQL Server, uma escolha que assegura um armazenamento eficiente e seguro das informações.

O ambiente do frontend foi desenvolvido para ser compatível tanto com sistemas operativos macOS como Windows, garantindo maior flexibilidade e acessibilidade. O ambiente gráfico da aplicação foi implementado com recurso às bibliotecas do Java, utilizando componentes do JPanel para proporcionar uma interface intuitiva e funcional.

	id_usuario ▾	email ▾	senha_hash ▾
1	1	Lucas@gmailcom	-606154167

Figura 1

Servidor Backend – Este componente é responsável por gerir e armazenar todas as informações relacionadas com as consultas, clínicas e médicos. Os dados são armazenados em bases de dados MySQL e SQL Server, uma escolha que se deve à necessidade de compatibilidade e adaptabilidade às diferentes configurações dos servidores, uma vez que o SQL Server não é compatível com arquiteturas ARM64. Este design oferece flexibilidade e fiabilidade no armazenamento dos dados.

No backend estão centralizadas as fichas dos utilizadores associados às clínicas, bem como todos os detalhes das marcações e especialidades disponíveis. A comunicação entre o frontend e o backend é realizada através do protocolo RMI, o que assegura uma ligação eficiente e segura entre os componentes do sistema. O backend foi projetado para operar num ambiente Linux, tirando partido da estabilidade e desempenho deste sistema operativo para garantir uma operação robusta.

Esta arquitetura proporciona uma base sólida para o funcionamento do sistema, assegurando a organização, segurança e acessibilidade dos dados, enquanto permite uma comunicação eficaz e fiável entre os diferentes componentes.

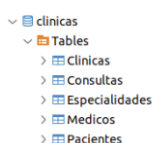


Figura 2

4 Manual de instalação e configuração

Este **Manual de Instalação** estará organizado em duas secções principais, correspondendo ao **Frontend** e ao **Backend**, com instruções detalhadas para cada componente.

A secção dedicada ao **Backend** especificará os requisitos e os passos necessários para a instalação e configuração do programa de gestão de clínicas. Serão abordados configurações do servidor e bases de dados, bem como instruções para a inicialização do sistema e sua integração com o Frontend.

Esta divisão clara facilita a compreensão e implementação do sistema, permitindo que cada parte seja configurada de forma independente e eficiente.

4.1 Frontend

O **frontend** foi testado e executado tanto em ambiente **Windows** como **macOS**, sendo, portanto, compatível com ambos os sistemas operativos. Para a instalação e configuração do **Frontend**, são necessários os seguintes passos:

5 Requisitos de Instalação

1. **Eclipse 2022** – A IDE utilizada para o desenvolvimento do frontend.
2. **Apache Tomcat 9** – O servidor onde os serviços SOAP e REST serão executados.
3. **Apache CXF 3.6.5** – Necessário para a implementação de Web Services SOAP.

Adicionar o Apache Tomcat 9 no Eclipse

1. Aceder ao Painel de Preferências do Eclipse:
2. Pesquisar por “Server” na barra de pesquisa do painel.
3. No submenu Runtime Environments, clicar em Add.
4. Selecionar “Apache Tomcat v9.0” da lista de servidores disponíveis.

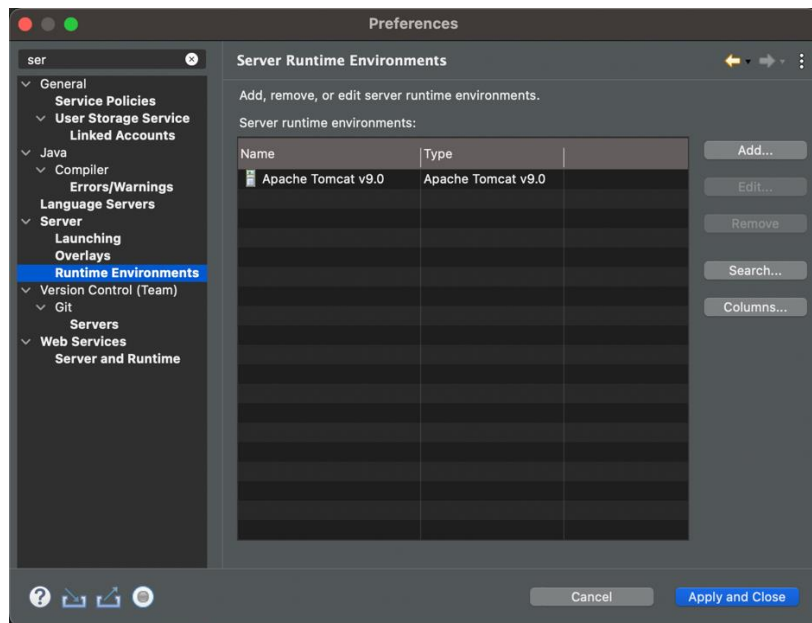


Figura 3

Configurar o Apache CXF no Eclipse

1. Aceder ao Pannel de Preferências do Eclipse:
2. Pesquisar por “Web Services” na barra de pesquisa do painel.
3. Selecionar a opção CXF 2.x Preferences.
4. No campo CXF Runtime, clicar em Browse e adicionar o caminho onde o Apache CXF está instalado.

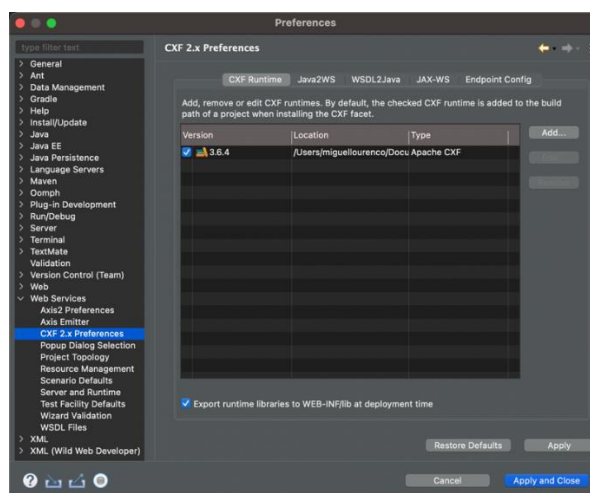


Figura 4

Configurar o Servidor e o Runtime no Eclipse

1. Aceder ao Painel de Preferências do Eclipse:
2. Pesquisar por “Web Services” na barra de pesquisa do painel.
3. Selecionar a opção Server and Runtime no menu de Web Services.
4. Adicionar o Tomcat v9.0 Server e o Apache CXF 2.x Runtime na configuração:
 - No campo Server, seleccionar Tomcat v9.0 Server (configurado previamente no Passo 1).
 - No campo Runtime, seleccionar Apache CXF 2.x Runtime (configurado no Passo 2).

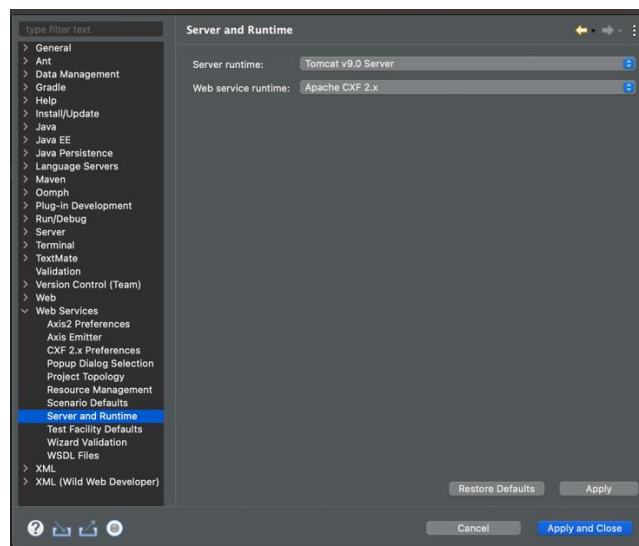


Figura 5

Após a configuração inicial do Eclipse, é possível importar os projetos necessários para o funcionamento do sistema, tanto para os serviços **SOAP** como para os serviços **REST**. Estes projetos estão disponibilizados em dois ficheiros separados, um para cada tipo de serviço.

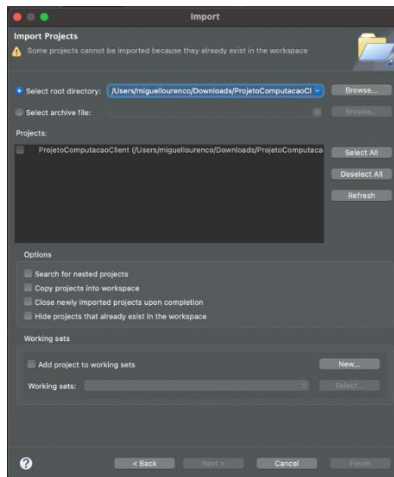


Figura 6

Após a importação, teremos agora os 4 ficheiros dentro do Eclipse, como mostrado na imagem abaixo.



Figura 7 Rest

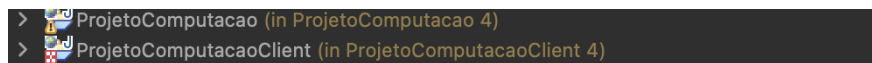
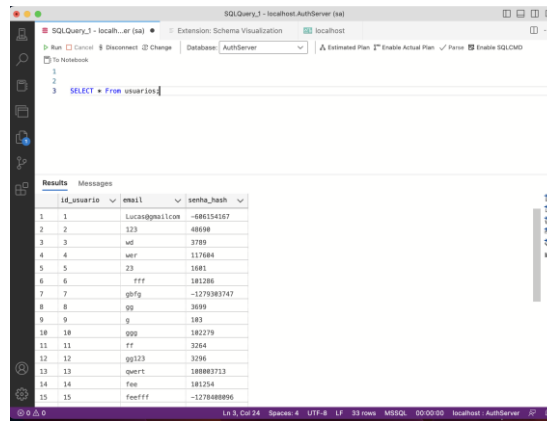


Figura 8 Soup

Passando agora à configuração da base de dados, que contém as informações de login dos utilizadores, a decisão tomada pelo grupo foi implementar uma base de dados em SQL Server. Utilizou-se uma imagem do SQL Server no Docker, que foi usada anteriormente na cadeira de Base de Dados, mas também pode ser instalada diretamente através da linha de comandos.

Para tornar a visualização da base de dados mais fácil, como no exemplo:



The screenshot shows a SQL query interface with the following data:

	id_usuario	email	senha_hash
1	1	lucad@gmail.com	-060154167
2	2	123	40000
3	3	wd	3789
4	4	wer	117684
5	5	23	1061
6	6	fff	181286
7	7	ghfg	-1279383747
8	8	99	3699
9	9	9	183
10	10	999	182279
11	11	ff	3264
12	12	gg123	3296
13	13	quert	188883713
14	14	fee	181254
15	15	feetff	-1279488996

Figura 9

É necessário criar uma base de dados chamada AuthServer, que deverá conter as colunas `id_usuario`, `email` e `senha_hash` (que será a palavra-passe do utilizador), como representado na imagem acima.

Após a criação da base de dados, será necessário configurar tanto no SOAP quanto no REST a ligação à nossa base de dados para que a comunicação funcione corretamente.

No ficheiro AuthServerConn (tanto no projeto REST quanto no SOAP), devemos alterar os seguintes campos para os correspondentes da nossa base de dados, de modo a permitir que o código consiga comunicar corretamente com a base de dados:

- `id_usuario`
- `email`
- `senha_hash`

Com estas alterações, o sistema conseguirá fazer a autenticação dos utilizadores, conectando-se à base de dados e validando os dados fornecidos.

```
private final String userName = "sa";
private final String password = "BDLusofona2023";
private final String dbUrl = "jdbc:sqlserver://localhost:1433;databaseName=AuthServer;encrypt=false";
```

Figura 10

Por fim, para terminar a configuração do Frontend, é necessário fazer alterações tanto no REST como no SOAP para permitir a comunicação com o Backend via RMI. Para isso, é necessário, primeiramente, instalar o Java no computador.

A terminal window titled 'pastatestes -- zsh -- 80x24' showing a series of commands and their outputs. The user 'miguellourenco' is at 'MacBook-Air-de-Miguel'. The commands executed are: 'javac -d bin DatabaseService.java', 'jar cf DatabaseService.jar -C bin .', 'jar tf DatabaseService.jar', and 'java -version'. The outputs show the contents of the jar file and the Java version information (OpenJDK 11.0.25).

```
miguellourenco@MacBook-Air-de-Miguel pastatestes % javac -d bin DatabaseService.j
ava
miguellourenco@MacBook-Air-de-Miguel pastatestes % jar cf DatabaseService.jar -C
bin .
miguellourenco@MacBook-Air-de-Miguel pastatestes % jar tf DatabaseService.jar

META-INF/
META-INF/MANIFEST.MF
com/
com/student/
com/student/DatabaseService.class
miguellourenco@MacBook-Air-de-Miguel pastatestes % java -version

openjdk version "11.0.25" 2024-10-15
OpenJDK Runtime Environment Homebrew (build 11.0.25+0)
OpenJDK 64-Bit Server VM Homebrew (build 11.0.25+0, mixed mode)
miguellourenco@MacBook-Air-de-Miguel pastatestes % java -version
openjdk version "11.0.25" 2024-10-15
OpenJDK Runtime Environment Homebrew (build 11.0.25+0)
OpenJDK 64-Bit Server VM Homebrew (build 11.0.25+0, mixed mode)
miguellourenco@MacBook-Air-de-Miguel pastatestes %
```

Figura 11

Após isso, temos que ir ao ficheiro DatabaseService e alterar para o IP da máquina virtual Linux correspondente, de modo a ser possível conectar ao serviço RMI do backend.

```
Registry registry = LocateRegistry.getRegistry("192.168.64.8", 1099);
DatabaseService service = (DatabaseService) registry.lookup("DatabaseService");
```

Figura 12

5.1 Backend

O Backend foi desenhado para ser instalado e utilizado tanto em Ubuntu x86_64 como em Ubuntu ARM64, o que implica codificação em paralelo para a base de dados do Backend em SQL Server para o Linux x86_64. No entanto, para o Linux ARM64, foi necessária uma adaptação para o MySQL, uma vez que o SQL Server não está disponível nas arquiteturas ARM64.

Neste manual, vamos abordar tanto a instalação e configuração de uma como de outra, porém, o projeto demonstrado será o utilizando MySQL.

Primeiramente, a instalação do Java no Ubuntu é necessária para permitir a comunicação via RMI com o Frontend. Para verificar se o Java já está instalado, basta executar o comando apropriado no terminal, como mostrado na imagem abaixo. Caso o Java não esteja instalado, será necessário proceder com a instalação. Após a instalação, para garantir a comunicação via RMI com o Backend, pode ser necessário alterar o IP do Ubuntu no host, de modo a refletir o IP da máquina, seguido do nome de utilizador, conforme indicado na Figura 14. Somente assim será possível estabelecer a comunicação via RMI entre o Backend e o Frontend.

```
miguel@miguel:~$ java -version
openjdk version "11.0.25" 2024-10-15
OpenJDK Runtime Environment (build 11.0.25+9-post-Ubuntu-1ubuntu122.04)
OpenJDK 64-Bit Server VM (build 11.0.25+9-post-Ubuntu-1ubuntu122.04, mixed mode)
miguel@miguel:~$
```

Figura 13

```
GNU nano 6.2 /etc/hosts
127.0.0.1 localhost
192.168.64.8 miguel

# The following lines are desirable for IPv6 capable hosts
::1 ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters

[ Read 9 lines ]
^G Help ^O Write Out ^W Where Is ^K Cut ^T Execute ^C Location
^X Exit ^R Read File ^\ Replace ^U Paste ^J Justify ^_ Go To Line
```

Figura 14

Após a instalação do Java, devemos então instalar o SQL Server ou o MySQL. Para uma melhor visualização da base de dados, recomenda-se o uso do Azure para o SQL Server e o DBeaver para o MySQL.

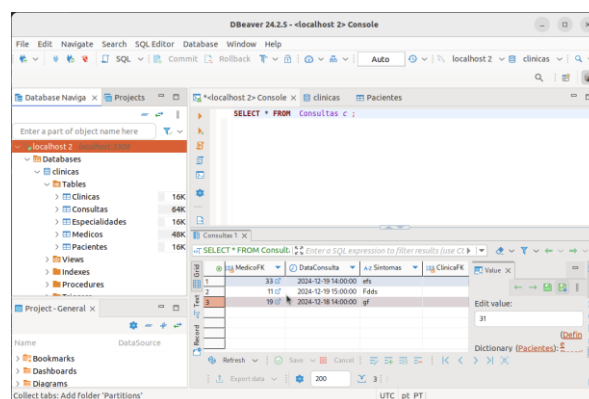


Figura 15 Dbiver

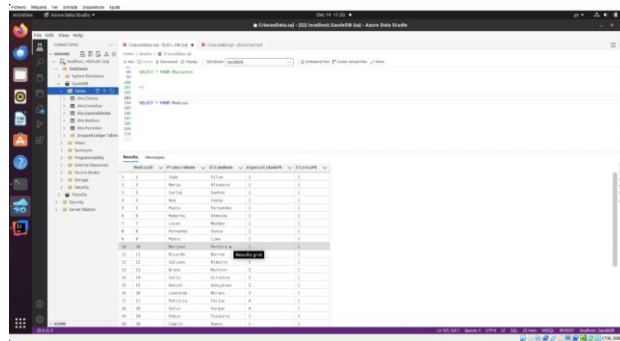


Figura 16 Azure

Onde criamos A tabelas com o seguinte código para **SQLserver**:

```
CREATE DATABASE [SaudeDB]
```

```
CREATE TABLE Clinicas (
    ClinicaID int IDENTITY(1,1) PRIMARY KEY ,
    Nome varchar(255) NOT NULL ,
    Morada varchar(255),);
```

```
CREATE TABLE Especialidades (
    EspecialidadeID int IDENTITY(1,1) PRIMARY KEY,
    Nome varchar(255) NOT NULL);
```

```
CREATE TABLE Medicos (
    MedicoID int IDENTITY(1,1) PRIMARY KEY,
    PrimeiroNome varchar(255) NOT NULL,
    UltimoNome varchar(255) NOT NULL,
    EspecialidadeFK int FOREIGN KEY REFERENCES Especialidades(EspecialidadeID));
```

```
CREATE TABLE Pacientes (
    PacienteID int IDENTITY(1,1) PRIMARY KEY,
    PrimeiroNome varchar(255) NOT NULL,
    UltimoNome varchar(255) NOT NULL,);
```

```
CREATE TABLE Consultas (  
    ConsultaID int IDENTITY(1,1) PRIMARY KEY,  
    PacienteFK int FOREIGN KEY REFERENCES Pacientes(PacienteID),  
    MedicoFK int FOREIGN KEY REFERENCES Medicos(MedicoID),  
    DataConsulta DATETIME NOT NULL,  
    Sintomas varchar(255),  
    ClinicaFK int FOREIGN KEY REFERENCES Clinicas(ClinicalID),);
```

E Para **MySQL** o seguinte código:

```
CREATE DATABASE [SaudeDB]
```

```
CREATE TABLE Clinicas (  
    ClinicalID int IDENTITY(1,1) PRIMARY KEY ,  
    Nome varchar(255) NOT NULL ,  
    Morada varchar(255),);
```

```
CREATE TABLE Especialidades (  
    EspecialidadeID int IDENTITY(1,1) PRIMARY KEY,  
    Nome varchar(255) NOT NULL);
```

```
CREATE TABLE Medicos (  
    MedicoID int IDENTITY(1,1) PRIMARY KEY,  
    PrimeiroNome varchar(255) NOT NULL,  
    UltimoNome varchar(255) NOT NULL,  
    EspecialidadeFK int FOREIGN KEY REFERENCES Especialidades(EspecialidadeID)  
);
```

```
CREATE TABLE Pacientes (  
    PacienteID int IDENTITY(1,1) PRIMARY KEY,  
    PrimeiroNome varchar(255) NOT NULL,  
    UltimoNome varchar(255) NOT NULL,);
```

CREATE TABLE Consultas (

ConsultaID int IDENTITY(1,1) PRIMARY KEY,

PacienteFK int FOREIGN KEY REFERENCES Pacientes(PacienteID),

MedicoFK int FOREIGN KEY REFERENCES Medicos(MedicoID),

DataConsulta DATETIME NOT NULL,

Sintomas varchar(255),

ClinicaFK int FOREIGN KEY REFERENCES Clinicas(ClinicaID),);

Após a instalação e configuração da base de dados que gera as consultas, é necessário a instalação de uma IDE para facilitar a realização das conexões com o **RMI** e a **base de dados**. Neste caso, como exemplo, vamos utilizar o IntelliJ IDEA.

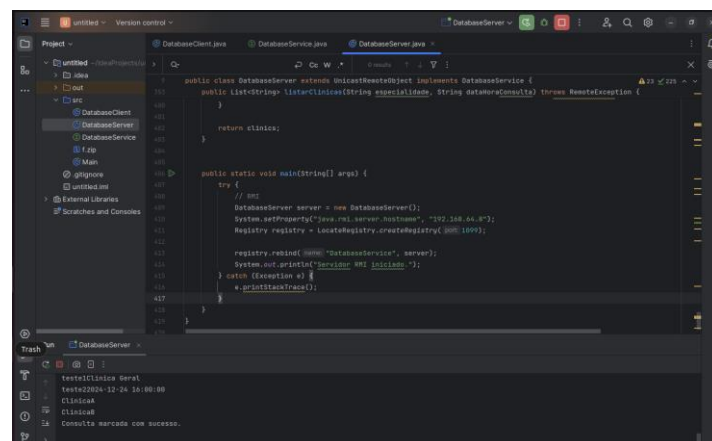


Figura 17

No IntelliJ teremos 2 ficheiros onde o Databaseservice é comum tanto ao backend como ao Front end e ser para Demonstrar as Função que o DatabaseService tem para depois o RMI do Front saber que funções pode chamar

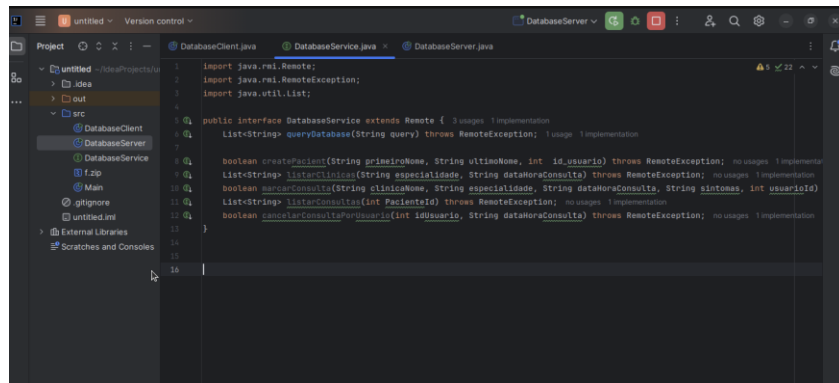


Figura 18

E uma classe chamada DatabaseServer, onde ocorrerá tanto a conexão à base de dados como as funções que interagem com ela, como a criação do serviço RMI.

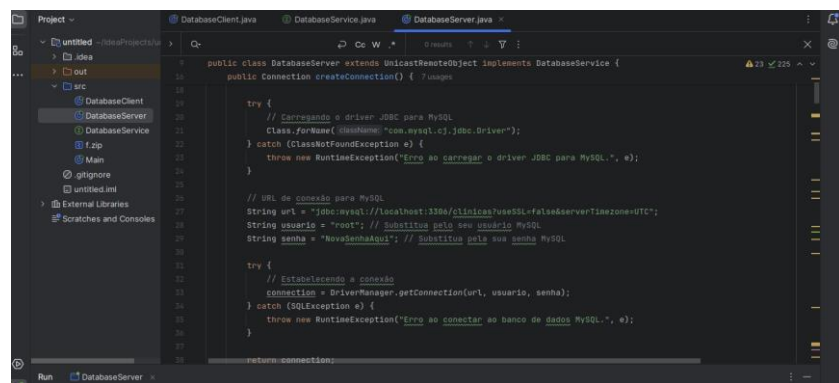


Figura 19

Na imagem acima, temos a conexão com a base de dados, que é similar à explicada no Frontend, onde teremos de alterar os campos do URL, usuário e senha de acordo com os valores definidos durante a criação da base de dados. Além disso, foi necessário colocar o driver do MySQL no diretório do projeto para permitir o acesso à base de dados.

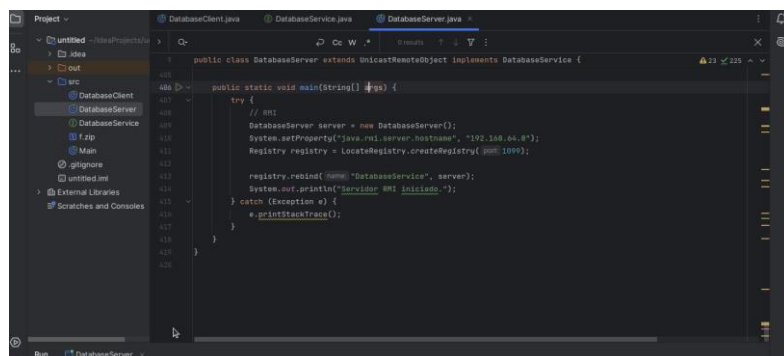


Figura 20

O código acima configura um servidor RMI (Remote Method Invocation) que permite a comunicação remota entre aplicações Java. Ele cria uma instância do servidor DatabaseServer, registra um registry RMI na porta 1099 e associa o serviço ao nome "DatabaseService", tornando-o acessível para chamadas remotas. Em caso de erro, a exceção é capturada e apresentada.

5.2 Inicialização dos WebServices

Para iniciar os Web Services, é necessário seguir alguns passos para conseguir correr a aplicação.

5.2.1 Soup

Para fazer a conexão inicial do SOAP, é necessário criar um Web Service na classe MathUtility dentro da pasta Projeto de Computação, como demonstrado abaixo:

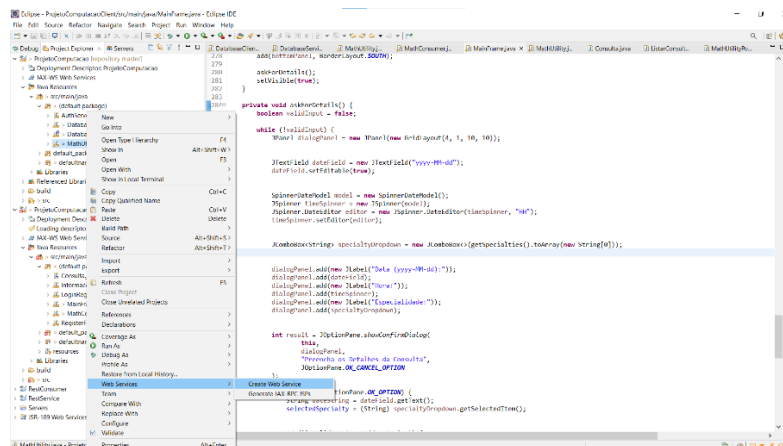


Figura 21

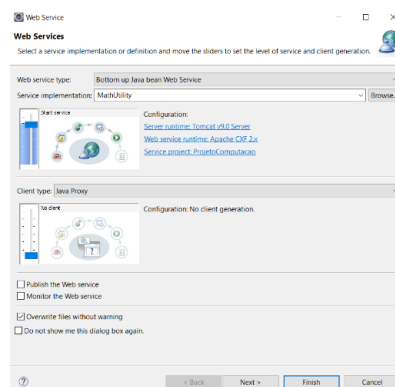


Figura 22

Onde, após fazer o indicado acima, podemos abrir no Google a seguinte página:



Figura 23

Após isso, é necessário então, no ficheiro MathConsumer na pasta ProjetoComputaçãoClient, criar um WebServiceClient, como demonstrado abaixo.

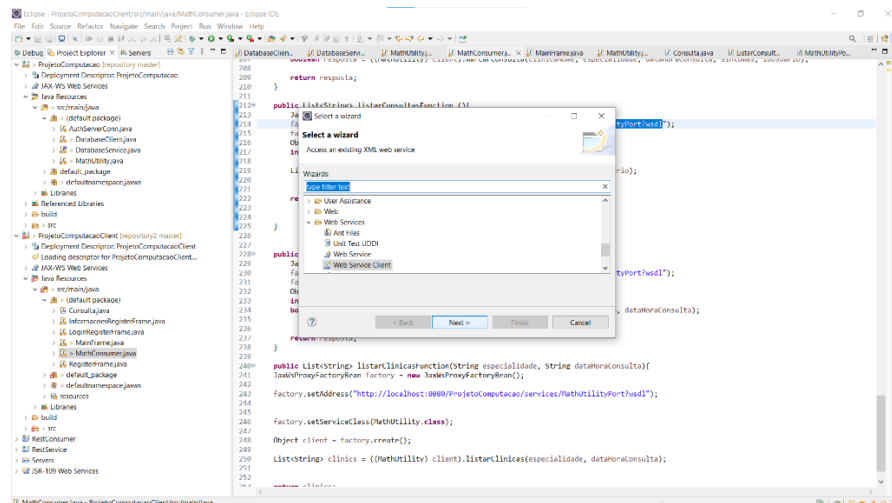


Figura 24

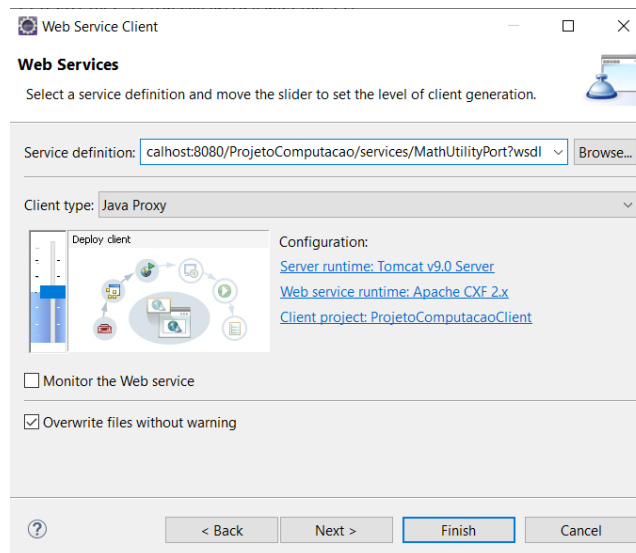


Figura 25

Após isso, podemos então, no mesmo ficheiro, executá-lo como uma Java Application, e isso irá abrir uma aplicação.

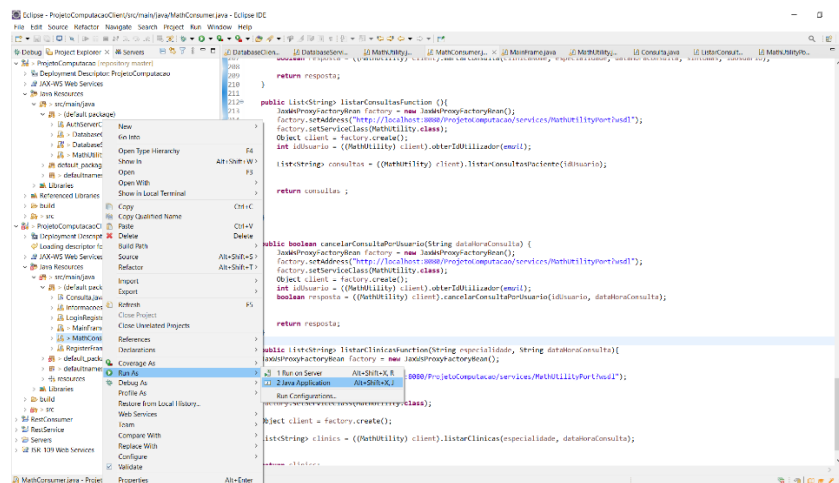


Figura 26

5.2.2 Rest

Para a criação do REST, é mais fácil, uma vez que só temos que correr a pasta novoRest como um servidor e, em seguida, executar o ficheiro MathConsumer da pasta cliente como uma Java Application.

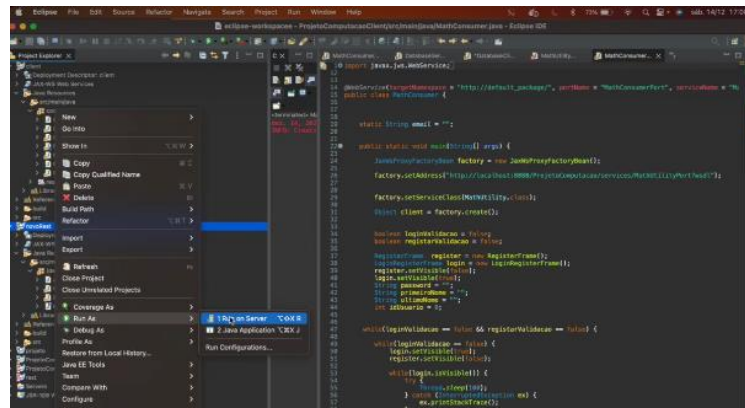


Figura 27

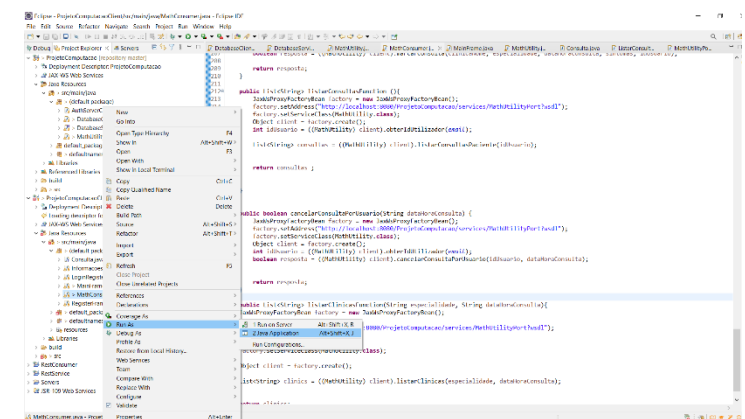


Figura 28

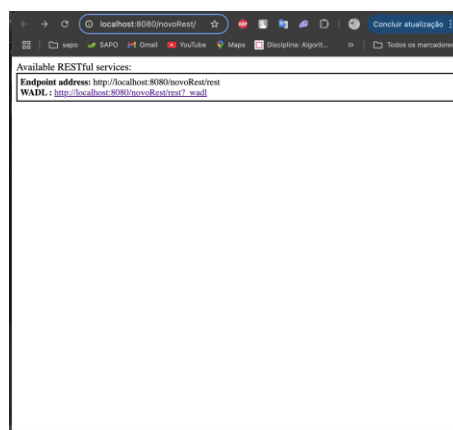


Figura 29

Após correr como uma aplicação Java, vai abrir uma aplicação similar tanto no SOAP como no REST.

6 Jpanel

No nosso projeto, optámos por utilizar a classe JPanel para construir a camada de apresentação da nossa aplicação Java. O JPanel é uma classe da biblioteca Swing que permite o desenvolvimento de interfaces gráficas amigáveis, flexíveis e organizadas, características fundamentais para criar aplicações robustas e de fácil manutenção.

Esta ferramenta possibilita a divisão da interface gráfica em vários JPanel's, o que contribui para uma organização modular e facilita a compreensão por parte dos utilizadores. Além disso, oferece grande liberdade para organizar componentes como botões, tabelas e caixas de texto, permitindo que a interface seja adaptada a diferentes requisitos de design e funcionalidade.

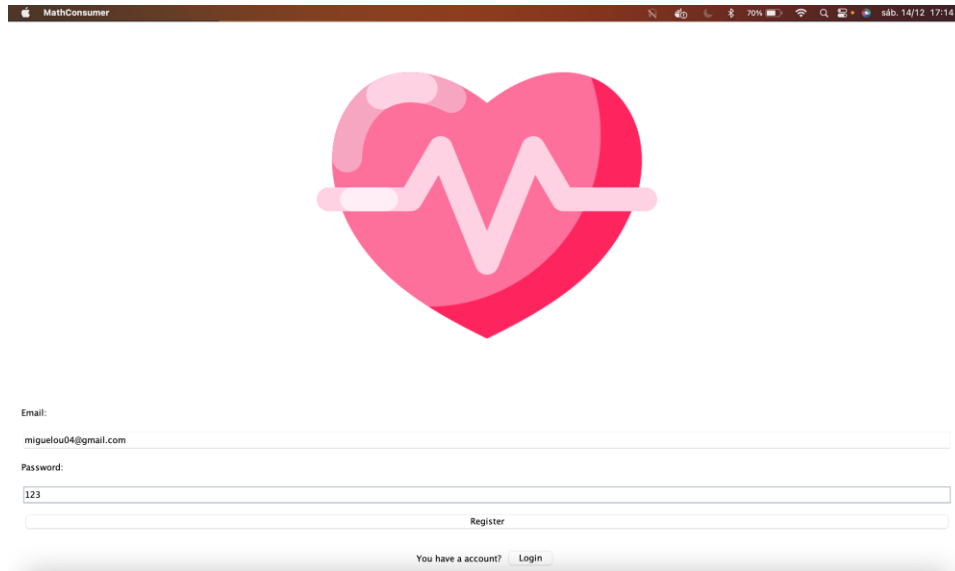
O JPanel destaca-se também pela sua capacidade de personalização, possibilitando a criação de designs únicos e ajustados às necessidades específicas do projeto, seja através da customização de cores, bordas ou da inclusão de elementos visuais dinâmicos.

Mais do que um simples contentor estático, o JPanel é interativo, permitindo que a aplicação reaja a eventos dos utilizadores, como cliques em botões ou seleções em menus dropdown. Esta característica torna-o uma escolha essencial para o desenvolvimento de aplicações que exigem interatividade.

Por fim, sendo parte da biblioteca Swing, o JPanel é independente da plataforma, o que garante a sua compatibilidade com diversos sistemas operativos. Combina simplicidade de implementação com um elevado grau de personalização e funcionalidade, tornando-o ideal para satisfazer as exigências de aplicações modernas.

7 Guia de Utilização

7.1 Registo De Novo Usuário



The screenshot shows a web browser window titled "MathConsumer". The page features a large pink heart icon with a white ECG line. Below the icon, there are two input fields: "Email:" with the value "miguelou04@gmail.com" and "Password:" with the value "123". A "Register" button is positioned below the password field. At the bottom, there is a link "You have a account? Login". The browser's status bar at the top right shows the date and time as "sáb. 14/12 17:14".

Figura 30 Registrar Usuário

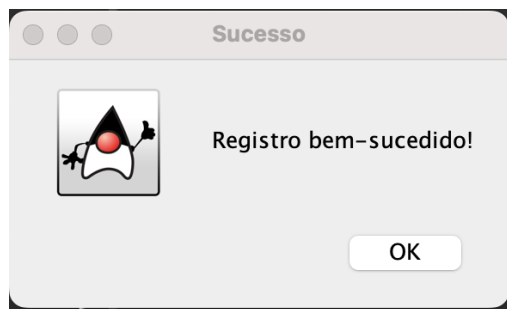


Figura 31 Registo com sucesso

MathConsumer

69% 14/12 17:17

Primeiro Nome:

Miguel

Último Nome:

Laurencio

Enter

Figura 32 Registo Paciente



Figura 33 Registo com sucesso

7.2 Login Usuário

MathConsumer

69% 14/12 17:18

Email:

miguelstudi@gmail.com

Password:

123

Login

Don't have an account? Register

Figura 34 Login

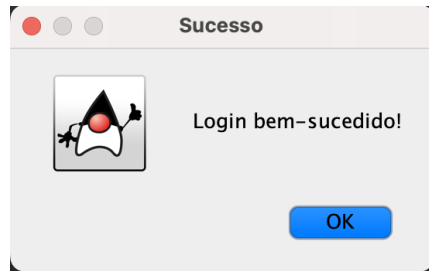


Figura 35 Login com sucesso

7.3 Marcar Consulta

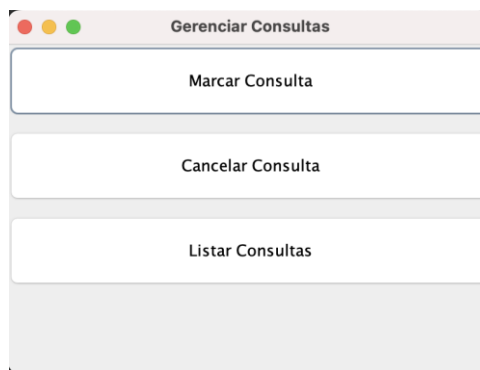


Figura 36 marcar consulta

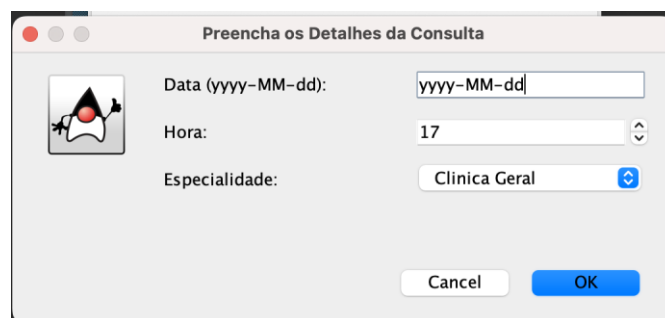


Figura 37

Preencha os Detalhes da Consulta

Data (yyyy-MM-dd): 2024-12-24

Hora: 22

Especialidade: Ginecologia

Cancel OK

Figura 38 marcar consulta hora errada

Marcar Consulta

Erro

Hora inválida! Selecione uma hora entre 08:00 e 20:00.

OK

Figura 39 erro ao marcar



Clinica:

☒ ClínicaA

☐ ClínicaB

Sintomas:

Salvar Consulta

Figura 40 marcar consulta

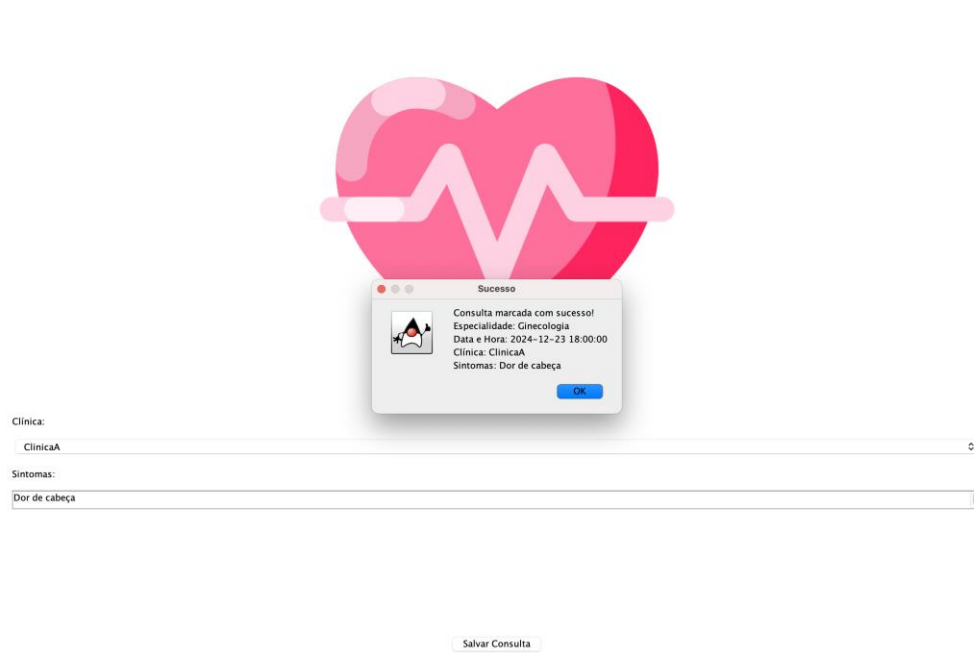


Figura 41 sucesso na marcação

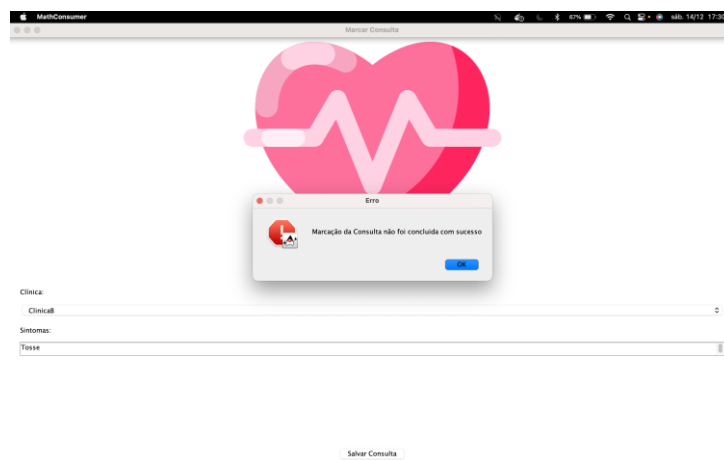


Figura 42 tentar marcar consultas a mesma hora

7.3.1 Listar Consultas

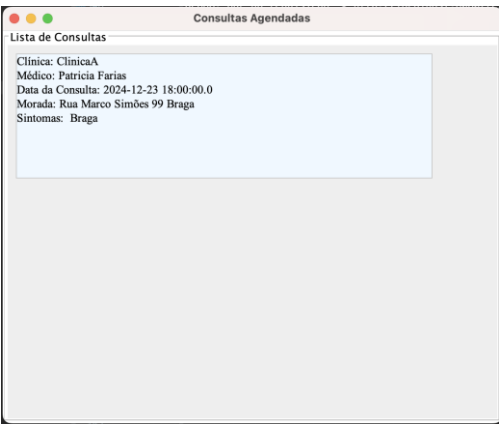


Figura 43 listar consultas

7.3.2 Cancelar consultas

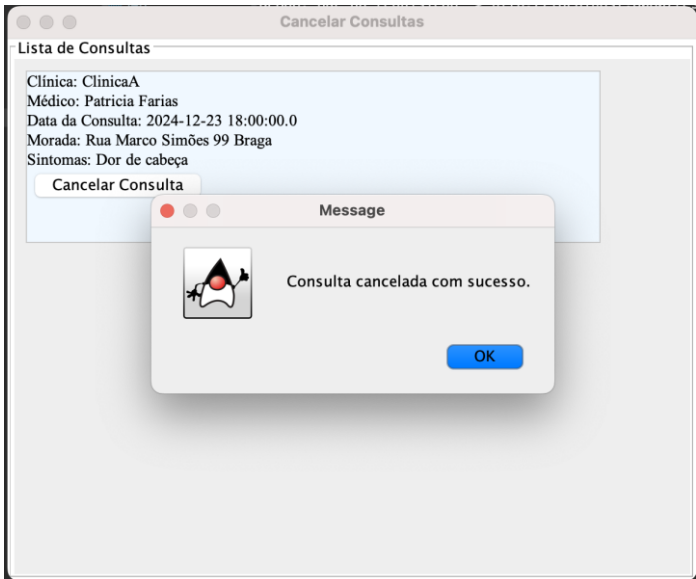


Figura 44 cancelar consulta

8 Conclusão

Com o desenvolvimento deste projeto, visamos aprofundar o conhecimento sobre RMI juntamente com o conhecimento em Web Services, uma vez que ambos foram partes fundamentais do projeto. Também tivemos contato com novas tecnologias, como JPanel, que nos permitiram o desenvolvimento e aprendizagem sobre esses tópicos. Após aplicados e desenvolvidos ao longo de várias horas de trabalho, os resultados esperados foram alcançados, com a criação do cliente, tanto no SOAP como no REST, com conexão ao RMI.

9 Bibliografia

<https://www.eclipse.org/downloads/packages/release/2022-06/r>, acedido em novembro 2024.

<https://tomcat.apache.org/download-90.cgi>, acedido em novembro 2024.

<https://cxf.apache.org/>, acedido em novembro 2024.