Curso Profissional de Gestão e Programação de Sistemas Informáticos

Disciplina de Programação e Sistemas de Informação

Módulo 8 – Conceitos Avançados de Programação

CONCEITOS RELATIVOS À INTERFACE DE DESENVOLVIMENTO DE APLICAÇÕES (API) DO SO

Alunos: Alexandre Miguel; Carlos Tojal; Frederico Bento

Professora: Matilde Vieira

Caldas da Rainha, Outubro de 2019

POCH PORTUGAL 2020 UNIÃO EUROPEIA Fundo Social Europeu

ÍNDICE

AGRADECIMENTOS

Este trabalho só pôde ser realizado com a ajuda de alguns professores, que tanto este ano como no anterior nos transmitiram alguns conhecimentos essenciais para a realização do trabalho, entre eles:

- O professor João Duarte, que nos lecionou a disciplina de Programação e Sistemas de Informação no ano passado, e nos ensinou os conceitos básicos de programação, que foram essenciais para a compreensão da programação orientada a objetos.
- A professora Maria João Anacleto, que nos lecionou a disciplina de Sistemas Operativos, o que a levou a nos ter explicado como é realizada a interação entre os programas, o sistema operativo e o hardware.
- E por fim, o professor Paulo Leonardo, que nos lecionou a disciplina de Redes de Comunicação, onde nos ensinou os conceitos básicos acerca do funcionamento da comunicação de computadores através de rede, o que foi muito importante na realização do trabalho.

Agradecemos também à professora Matilde Viera por nos ter proposto este trabalho, que nos permitiu aprofundar os nossos conhecimentos relativos à programação.

ÍNDICE DE FIGURAS

SIGLAS/ABREVIATURAS

API
Application Programming Interface, 1

GUI
Graphics User Interface, 6

POO
Paradigma Orientado a Objetos, 2

SO
Sistema Operativo, 1

INTRODUÇÃO

O nosso grupo é constituído por Alexandre Miguel, Carlos Tojal e Frederico Bento do 2º ano do curso Técnico de Gestão e Programação de Sistemas Informáticos.

No nosso grupo cada elemento tem funções a desempenhar, a primeira função é desempenhada pelo aluno Alexandre Miguel que consiste em organizar e fundamentar ideias, o aluno Carlos Tojal tem a função de criar exemplos de código para fundamentar a ideia principal do trabalho, tratar a escrita e escrever partes do trabalho que requeiram explicações mais técnicas, e por último o aluno Frederico Bento tem como função pesquisar informação e filtra-la de modo a ter um trabalho consistente.

Este trabalho foi-nos proposto pela professora Matilde Viera no âmbito de adquirir conhecimento acerca de vários temas do módulo 8.

Em vários temas propostos, o tema selecionado para o nosso grupo foi “Conceitos Relativos à Interface de Desenvolvimento de Aplicações (API) do Sistema Operativo (SO) ”.

Com este trabalho os nossos objetivos principais serão:

- O domínio do conceito de API e API do sistema;
- Demonstração da sua finalidade e em que contexto são utilizadas;
- A construção de uma API, com o intuito de fundamentar o trabalho e consolidar os conhecimentos obtidos com a pesquisa.

1. Conceitos do módulo 8

1.1. Conceito de Programação Orientada a Eventos

Para Pinheiro (2017) o paradigma orientado a objetos visa expressar problemas como os objetos ao invés da análise mais tradicional numa perspetiva de programação, em que as rotinas (funções e procedimentos) foram substituídas por métodos e os dados/variáveis por atributos. Neste paradigma, quando dividimos um problema em partes convém pensar em objetos com determinados atributos e métodos. [OU]

"Paradigma Orientado a Objetos (POO) consiste em expressar os Problemas como objetos, ao contrário da análise tradicional os quais eram em rotinas e dados, que aqui foram substituídos por métodos (comportamento) e atributos (propriedades). Assim, quando é colocado o problema de desenvolver um sistema na análise orientada a objetos, deve-se pensar como dividir esse problema em objetos." (Pinheiro, 2017)

Existem diversos paradigmas de programação, sendo o de programação orientada a objetos - POO - um dos mais utilizados. Segundo (Jesus, 2013) este paradigma organiza o código em objetos, que são autónomos e trocam mensagens entre si durante a execução do programa, imitando o comportamento dos objetos de acordo com o que sucede no mundo real.

1.1.1. Eventos

"Um Evento é algo que ocorre numa aplicação e possui um determinado significado para o sistema, desencadeando uma determinada ação como por exemplo carregar num botão ou fechar uma janela." (Pereira)

1.1.2. Exemplo de Programação Orientada a Eventos

Por exemplo, quando clicamos um botão na interface do nosso programa JAVA, o sistema operativo envia um evento, para alertar os programas do acontecimento.

Este evento pode ser recebido pelo nosso programa, e podemos pedir a execução de um dado código nessa ocasião.

1.2. Conceito de Programação Orientada a Objetos

POO (Programação Orientada a Objetos) é um paradigma de programação baseado no conceito de “objetos” compostos por campos ou métodos.

“POO usa abstração para criar modelos baseados no mundo real, usa várias técnicas, incluindo por exemplo o encapsulamento que é o princípio pelo qual cada componente” (Ferrari, 2013) de um programa deve agregar toda a informação relevante para sua manipulação como uma unidade (uma cápsula).

1.2.1. Objetos

“Os objetos, em programação, são unidades de código utilizadas no desenvolvimento de aplicações. Classes e objetos podem dizer respeito a qualquer tipo de entidades usadas em programação, tais como: janelas, menus, botões de comando, caixas de texto, imagens, estruturas de dados, etc.”. (Azul, 2011)

1.2.2. Exemplo de Programação Orientada a Objetos

Um exemplo da programação orientada a objetos é a criação de uma janela, que resulta da instanciação de uma classe que permite tal.

Neste caso específico, essa classe deverá também conter métodos que permitem por exemplo a definição do tamanho da janela, estilos ou adição de componentes.

Isto significa ainda que provavelmente esta classe deverá também conter atributos com alguns destes valores.

1.3. Conceito de Interface Gráfica

A interface gráfica é uma forma de interface de utilizador, que funciona por meio de uma tela ou representação gráfica, e é caracterizada pela utilização de dispositivos apontadores para selecionar por exemplo botões ou menus (o Windows é um sistema operativo baseado em interface gráfica, por exemplo).

1.4. Conceito de Programa

Normalmente quando se fala de “programa”, na área da informática trata-se de uma referência a um software, ou seja, uma aplicação que permitem desenvolver diferentes tarefas num computador, telefone ou outro dispositivo eletrónico. Como exemplo de um programa/software temos o Microsoft Word, que permite escrever e ler textos, entre outras funcionalidades.

1.5. Conceito de Atributo

“Atributos de uma classe também conhecidos como propriedades, descrevem um intervalo de valores que as instâncias da classe podem apresentar.

Um atributo é uma variável que pertence a um objeto, os dados de um objeto são armazenados nos seus atributos.” (Soares, 2018)

1.6. Conceito de propriedades

“Propriedades são estruturas em uma classe que definem o acesso a informações dentro dela. Uma propriedade é composta por dois métodos (os famosos Get e Set): um método para retornar a informação Get, e um método para definir o valor da propriedade chamado Set. Geralmente, para cada método existe uma variável dentro da classe que armazena o valor da propriedade”. (Camargo, 2010)

2. Momentos de trabalho na Programação Orientada a Objetos

“Na Programação Orientada a Objetos (POO), como em qualquer outro paradigma de programação, antes de começar a programar ainda há muito para planear. Primeiramente é preciso saber o que programa vai fazer e o que está envolvido nisso. Por exemplo no sistema de uma escola provavelmente será necessário gerir os alunos, então nós conseguimos converter essa realidade para objetos. Seguindo o exemplo anterior cada estudante tem um nome, um número de processo, ano, curso e disciplinas. Estes serão os atributos da classe “Aluno”.

É preciso repetir este processo para tudo o que precisará de gerir no seu programa. Então, só será necessário criar outras classes e métodos que vão gerir e conectar todos os objetos entre si. Neste caso, provavelmente métodos para adicionar, remover ou atualizar a informação do aluno seriam úteis ou necessários”. (Desconhecido, 2017)

3. Tema

3.1.1 O que é uma API?

"API é o acrónimo de *Application Programming Interface* ou, em português, Interface de Programação de Aplicativos". (Ciriaco, 2009)

Tal como uma Graphics User Interface (GUI) facilita a interação entre o utilizador e o computador, uma API facilita a interação entre 2 softwares.

O principal objetivo das APIs é permitir a reutilização de código, facilitando o trabalho dos desenvolvedores, caso contrário estes teriam de desenvolver softwares que permitissem a comunicação entre o programa que estão a desenvolver e o serviço de que necessitam.

Um exemplo de API é a do *PayPal*, que pode ser por exemplo incorporada em sites de vendas online, permitindo transações sem sair do site.

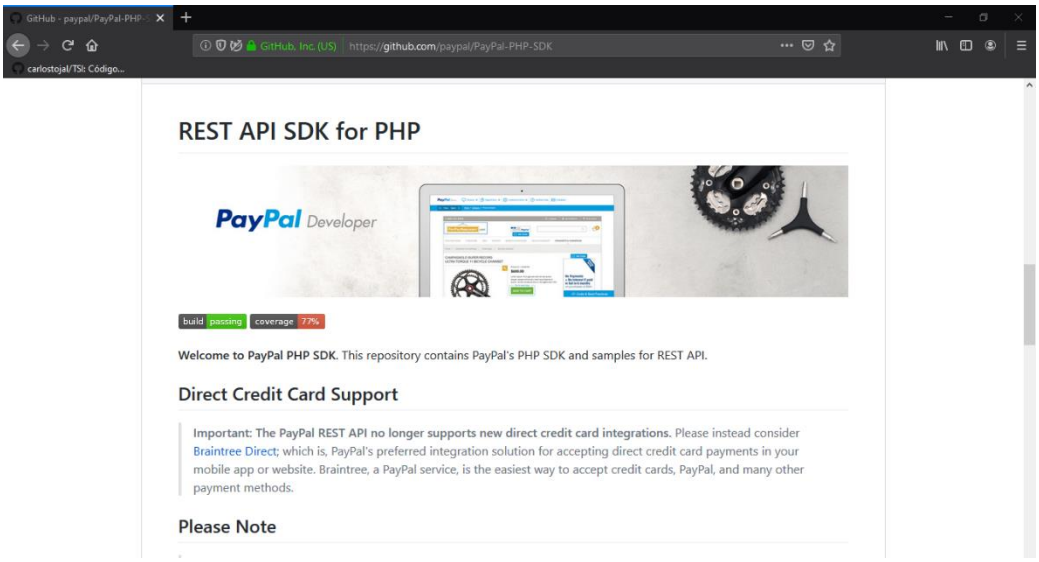

Figura 1 - Página do repositório da API do PayPal no GitHub

3.1.2 Tipos de API

Existem vários tipos de API, como por exemplo as que funcionam através da rede (como a do *PayPal*), ou as do sistema.

As APIs que vamos abordar são as APIs do sistema.

Estas APIs facilitam a interação entre o programa que o desenvolvedor está a criar e o sistema operativo.

3.1.3 O que é uma API do sistema

Uma API do sistema é código, numa dada linguagem, que interage com o sistema operativo, e consequentemente com os seus drivers, o que significa que manipula o hardware.

3.1.4 Em que circunstâncias são usadas APIs do sistema

Os desenvolvedores de software utilizam APIs do sistema frequentemente, muitas vezes sem sequer darem por isso. Aqui abordámos o caso do Java.

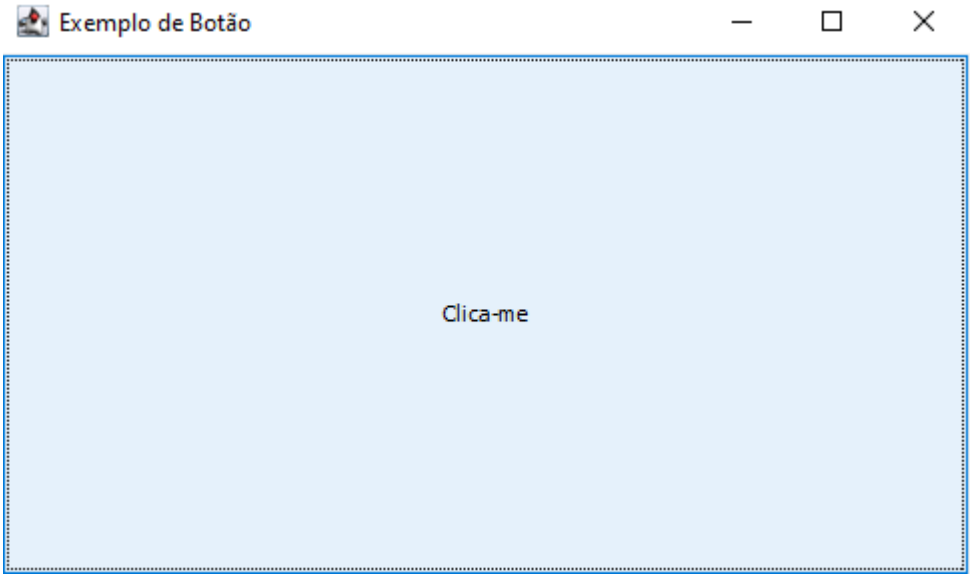

Figura 2 - Botão em JAVA utilizando componentes SWING

Um exemplo básico no desenvolvimento de programas Java em que é utilizada uma API do sistema: o clique de um botão.

3.1.5 Como funcionam as APIs do sistema em JAVA

Em JAVA, as APIs do sistema estão dentro de classes que comunicam com o sistema operativo, daí serem APIs do sistema, tal como já expliquei anteriormente.

Na verdade, as APIs do sistema em JAVA acabam por ser mais complexas do que aparentam inicialmente, pois dentro de classes que são consideradas APIs do sistema, são utilizadas outras classes que também o são, o que gera uma elevada complexidade na compreensão do código.

Isto demonstra ainda que se estas APIs não existissem e o desenvolvedor tivesse de criar a conexão entre os seus programas e o sistema operativo todas as vezes que criasse uma aplicação, seria um trabalho muito cansativo, e a evolução da programação e até mesmo da tecnologia (que está dependente da programação) seria muito mais lenta. As APIs do sistema, nas diversas linguagens, representam de facto um papel de grande importância no desenvolvimento de software.

É este também o objetivo das APIs do sistema: facilitar o trabalho do desenvolvedor, permitindo a reutilização de código e criando alguma uniformidade entre os códigos de todos os desenvolvedores da linguagem.

3.1.6 Exemplos práticos da utilização de APIs do sistema na programação em JAVA

Por exemplo, quando é clicado um botão na interface do utilizador, o sistema operativo envia uma mensagem, que é recebida pelo programa. Essa mensagem chama-se evento.

Para podermos criar um botão num programa em JAVA, temos de importar várias classes, de entre elas a JAVAX.SWING.JBUTTON, JAVA.AWT.EVENT.ACTIONEVENT e JAVA.AWT.EVENT.ACTIONLISTENER, como está demonstrado no exemplo abaixo.

```
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.JButton;

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
```

Figura 3 - Importação de algumas das bibliotecas mais usadas no desenvolvimento da maioria das aplicações gráficas em JAVA

A classe JAVAX.SWING.JBUTTON é utilizada para criar e personalizar o botão, e a classe JAVA.AWT.EVENT.ACTIONLISTENER é utilizada precisamente para receber o tal evento do sistema.

Assim sendo, podemos dizer que a classe JAVA.AWT.EVENT.ACTIONLISTENER é uma API do sistema, pois comunica com o sistema para receber o evento do sistema, que chega ao programa JAVA como um objeto da classe JAVA.AWT.EVENT.ACTIONEVENT.

Tal pode ser visto abaixo.

```
button.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        // Código a ser executado quando o botão é clicado
    }
});
```

Figura 4 - Código que recebe o evento do sistema operativo ao clicar no botão e permite a execução de código nesse momento

Existem muitas mais classes em JAVA que utilizam APIs do sistema. Até se poderia dizer grande parte delas.

Um outro exemplo de classe JAVA que utiliza uma API do sistema é a classe JAVA.NET.SOCKET.

```
import java.net.Socket;
```

Figura 5 - Importação da biblioteca JAVA que permite comunicação em rede a baixo nível

Esta classe é utilizada para comunicar em rede a baixo nível através do protocolo TCP/IP.

Para isso, esta classe tem de comunicar com os recursos de rede do sistema operativo para poder enviar dados para o endereço IP e porta configurados, assim como receber.

Daí podermos afirmar que esta classe utiliza uma API do sistema.

```
socket = new Socket(address, port);
```

Figura 6 - Instanciação do SOCKET

3.1.7 APIs na prática

Após compreendermos as bases de como são utilizadas APIs do sistema no desenvolvimento de aplicações em JAVA, pudemos avançar para a prática.

Para isso criámos dois programas que vão comunicar entre si, através da nossa própria API.

A ideia foi criar um projeto constituído por um programa em PYTHON, que seria um serviço que enviaria mensagens texto; um programa em JAVA que mostraria esse mesmo texto, e uma API para ligar ambos.

```
import myservice.API;
```

Figura 7 - Importação da minha API

A esses programas chamámos respetivamente *"MyService"*, "*MyApp"* e *"MyService API"*. Demos estes nomes apenas para facilitar a diferenciação de cada um dos códigos.

Depois disto foi mais fácil compreender como é feita a comunicação entre 2 softwares através de uma API.

Tanto esta API quanto os 2 outros programas utilizariam também APIs do sistema. No caso do programa *"MyApp"* e da API utilizariam APIs do sistema JAVA, e no caso do programa *"MyService"* da linguagem C. Porquê da linguagem C e não do PYTHON será explicado mais à frente.

Fazendo uma comparação com o desenvolvimento de um programa em JAVA, o sistema operativo seria o correspondente ao *"MyService"*, o programa a ser desenvolvido a *"MyApp"* e a API do sistema a *"MyService API"*.

De forma simples, o programa *"MyApp"* conectar-se-ia ao programa *"MyService"* através da classe *"MyService API"*, o programa *"MyService"* enviaria as mensagens através de sockets de rede em PYTHON para o endereço IP e porta do programa *"MyApp"*.

```
// Método que faz a ligação ao endereço IPv4 e porta fornecidos como argumento
public Socket connect(String address, int port) {
    try {
        socket = new Socket(address, port);
    } catch (UnknownHostException e) {

    } catch (IOException e) {

    }
    return socket;
}
```

Figura 8 - Método da minha API que cria a conexão com o programa "MyService" e retorna o objeto SOCKET resultante

Nesta parte são utilizadas APIs do sistema na classe *"MyService API"*, para se conectar ao programa "MyService", que também utiliza APIs do sistema para comunicar com a classe *"MyService API"*.

```
host = "192.168.1.69" # endereço da placa de rede a utilizar
port = 555

s.bind((host, port))
s.listen()

print("Serviço iniciado em "+host+":"+str(port))

client, client_addr = s.accept()
print("Cliente conectado\n")

message = ""

while message != "exit":
        message = input("Mensagem: ")
        client.send((message+"\n").encode())
```

Figura 9 - Código do programa "MyService"

De seguida, as mensagens chegariam ao destino e seriam lidas utilizando a classe JAVA.IO.BUFFEREDREADER, que leria o input stream do socket, obtido através do método SOCKET.GETINPUTSTREAM(), código esse presente na classe *"MyService API"*.

```
public String loadMessage(Socket socket) {
    String message = "";
    try {
        in = new BufferedReader(new InputStreamReader(socket.getInputStream()));
        message = in.readLine(); // leitura da linha
    } catch (IOException e) { // em caso de erro ao tentar ler
        return "Erro ao aceder ao serviço.";
    }
    return message;
}
```

Figura 10 - Método da minha API que obtém a primeira mensagem recebida desde o último carregamento

Neste último passo são utilizadas pelo menos duas APIs do sistema: uma para o acesso ao sistema de ficheiros para fazer a leitura, e outra de rede para obter o input stream do socket. Depois disto as mensagens são recebidas pela *"MyService API"*, que as envia para a *"MyApp"*.

```
// Execução do método "connect()", pertencente à minha API (inicializa o atributo "socket")
socket = new API().connect("77.54.195.181", 555); // recebe o endereço IPv4 e porta na qual corre o serviço
if(socket.isConnected()) {
    isConnected = true;
    textArea.setText(textArea.getText()+"\n[MyService API] Conectou ao serviço MyService com sucesso.");
} else {
    textArea.setText(textArea.getText()+"\n[MyService API] O serviço MyService está inacessível.");
}
```

```
if(isConnected) {
    loadMessages();
}
```

Figura 11 – Algum código do ecrã principal do programa "MyApp"

Com isto conseguimos compreender que as APIs do sistema estão em grande maioria dos programas, obviamente que até mesmo no desenvolvimento de outras APIs, tanto como do sistema como outras.

3.1.8 O nosso projeto em prática

Conseguimos ainda com o projeto que criámos ter noção das potencialidades das APIs, pois a combinação de APIs do sistema com os nossos programas e a nossa API permitiu-nos comunicar entre distâncias superiores a 20km.

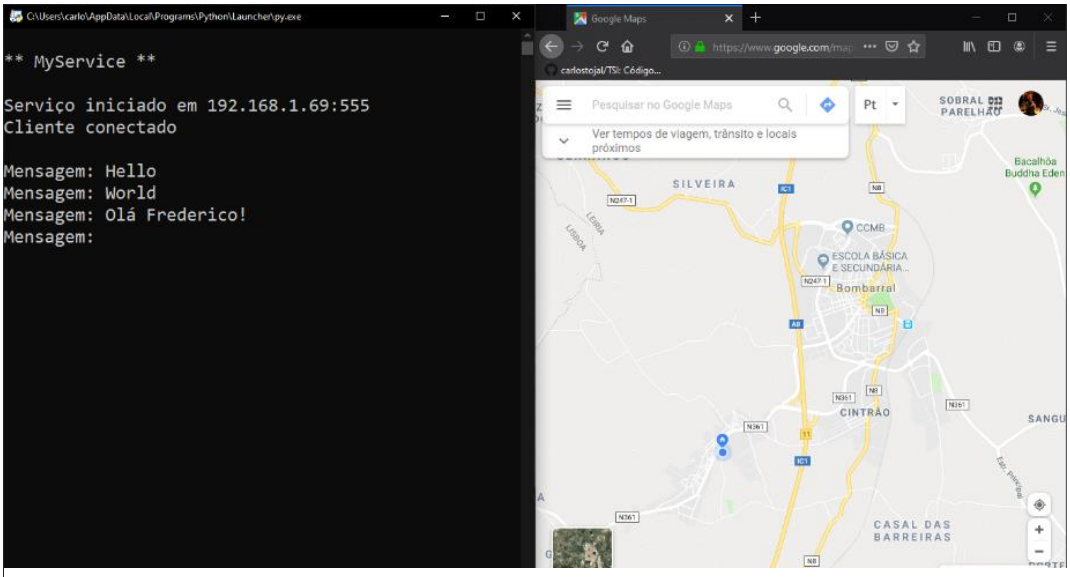

Figura 12 - Execução do programa "MyService" em Bombarral

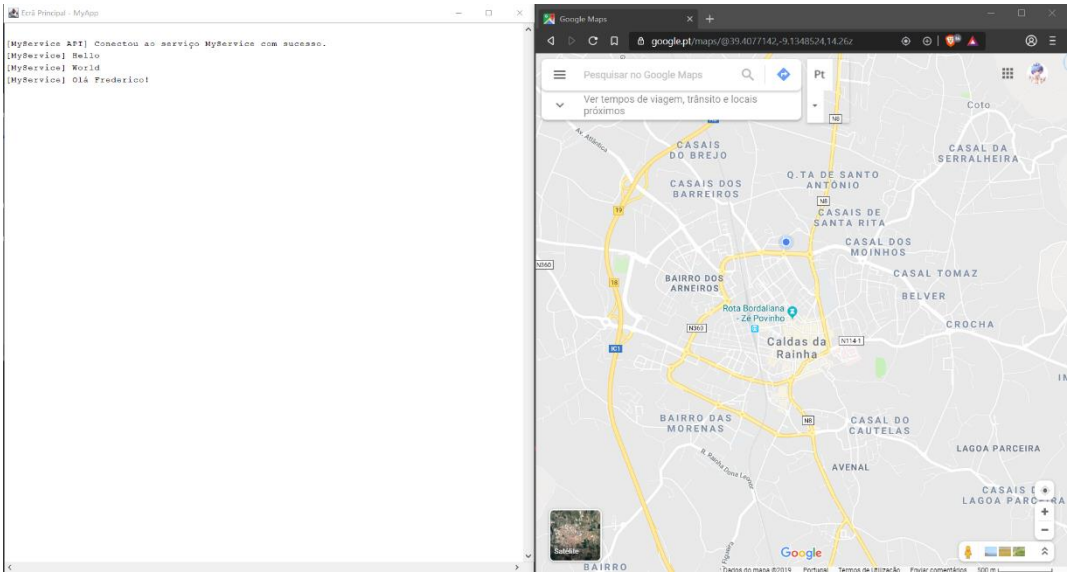

Figura 13 - Execução do programa "MyApp" em Caldas da Rainha

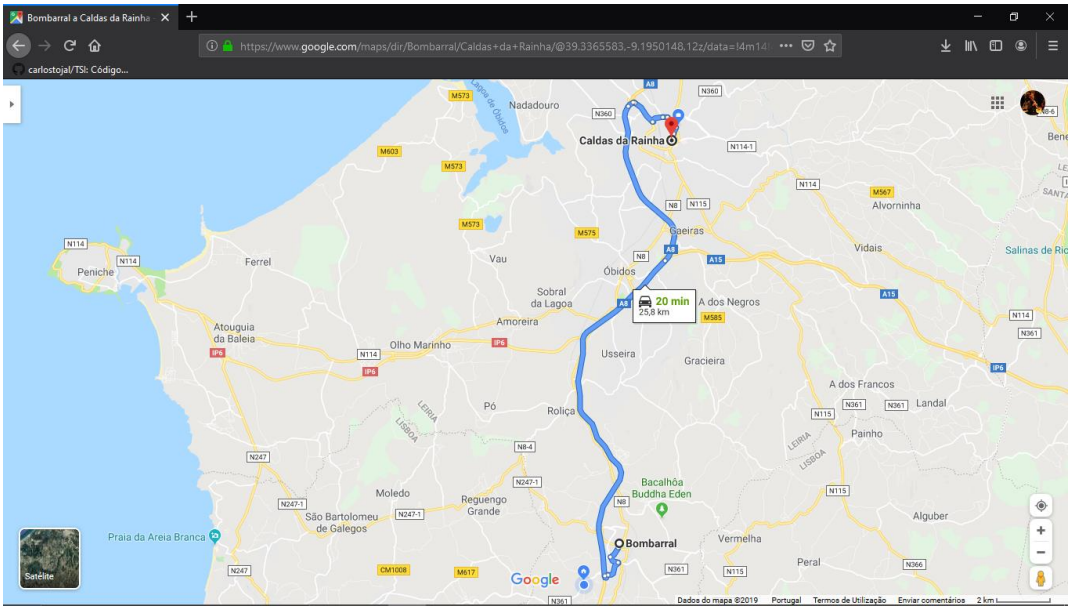

Figura 14 - Distância aproximada (por estrada) entre as máquinas que correram cada um dos programas

Para ser possível comunicar entre programas fora da mesma rede, tiveram de ser feitas configurações na rede em que ficou hospedado o programa *"MyService"*.

Teve de ser feito o redireccionamento da porta no router. Tal pôde ser feito com a configuração mostrada aqui.

Aqui pode ser visto que o endereço IP externo do router no qual ficou hospedado o programa *"MyService"* é 77.56.195.181.

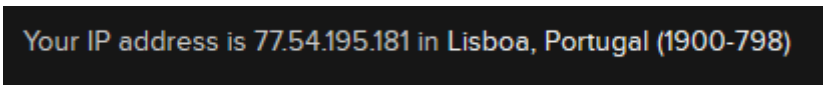

Figura 15 - Endereço IPv4 externo do router da rede que hospedou o programa "MyService"

Aqui foi obtido o endereço IPv4 da placa de rede pretendida para a hospedagem, que neste caso foi 192.168.1.69.

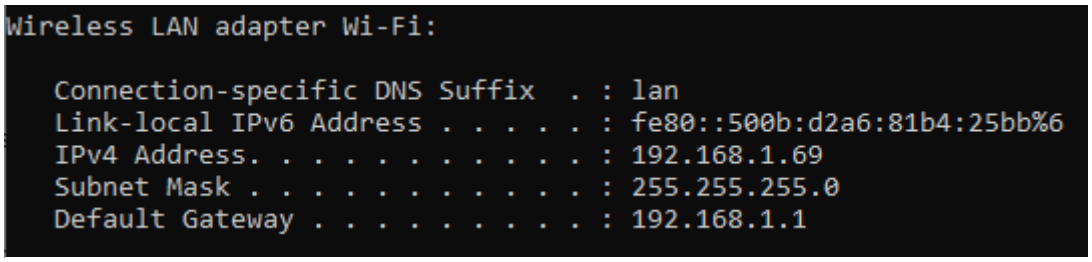

Figura 16 - Resultados do comando "ipconfig" na máquina que hospedou o programa "MyService"

E aqui podem ser vistas as respetivas configurações no painel de controlo do router, que pode ser acedido através do endereço local do router, que normalmente por padrão é 192.168.1.1.

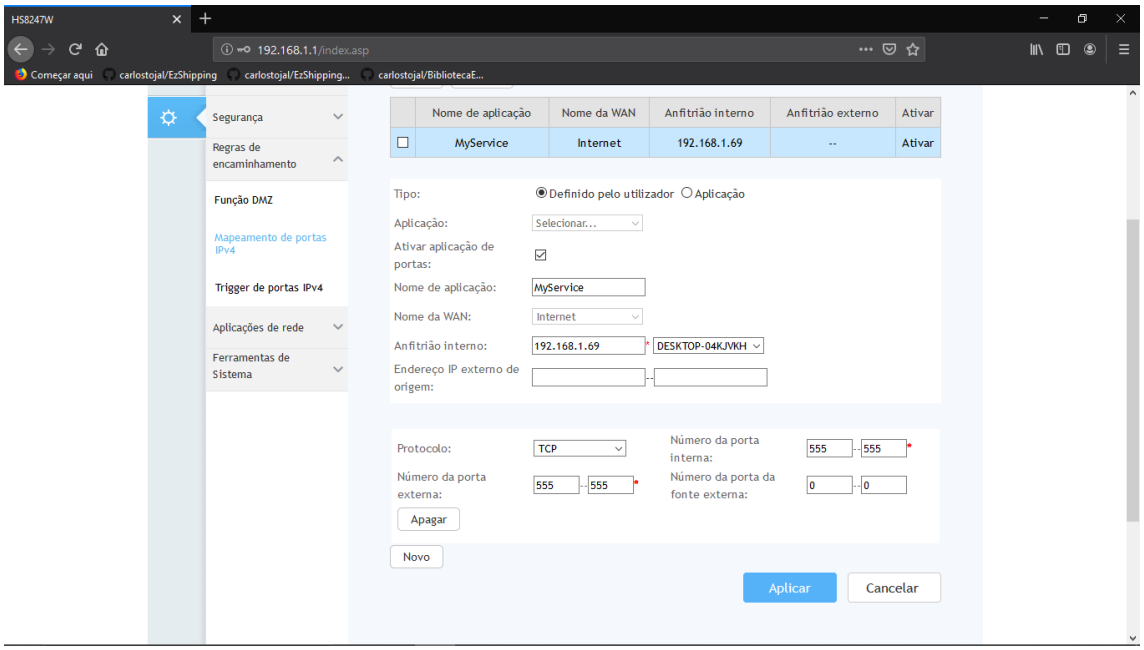

Figura 17 - Configuração do router da rede que hospedou o programa "MyService"

Isto significa que quando o computador de uma rede externa pede para aceder à porta 555 do endereço IP externo do router da rede em que está hospedado o programa *"MyService"*, o mesmo redireciona a ligação para a porta 555 do computador em que está hospedado este mesmo programa.

PYTHON, a linguagem utilizada no programa *"MyService"*, é uma linguagem interpretada e não compilada. Isto significa que as APIs do sistema utilizadas são do interpretador PYTHON, que foi na sua base escrito em C. Os restantes códigos integrantes do interpretador PYTHON que não são C, são na sua maioria códigos PYTHON, que tal como referi anteriormente são interpretados em C. Podemos assim afirmar que as APIs do sistema utilizadas na linguagem PYTHON são na verdade APIs do sistema da linguagem C.

Tudo isto para explicar que o computador que hospeda o programa *"MyService"* utiliza uma API do sistema do interpretador PYTHON para comunicar com o sistema operativo e aguardar uma conexão.

```
s.bind((host, port))
s.listen()
```

Figura 18 - Código do programa "MyService" responsável por aceitar conexão

Tal é possível pois o script PYTHON recorre ao interpretador, que recorre às APIs do sistema da linguagem C e dá instruções ao sistema operativo para aguardar uma conexão à porta e endereço dados. Aqui o endereço IP poderá ser o de qualquer placa de rede do computador que execute este script.

Posteriormente, o programa *"MyApp"* utiliza a *"MyService API"*, que utiliza uma API do sistema para estabelecer uma conexão com o endereço IP externo do router da rede em que está hospedado o programa *"MyService"*, que redireciona a ligação para esse mesmo computador para a porta 555.

A linha abaixo está presente na classe *"MainScreen"*, que chama a execução do método *"connect()"* da minha API.

```
socket = new API().connect("77.54.195.181", 555);
```

Figura 19 - Conexão ao programa "MyService" através da minha API

Esta linha está presente nesse mesmo método, em que *address* é o primeiro argumento (neste caso 77.54.195.181) e *port* o segundo (neste caso 555).

```
socket = new Socket(address, port);
```

Figura 20 - Linha de código presente no método CONNECT() da minha API

Por exemplo aqui a ligação é criada através da minha API, que recorre à API do sistema presente na classe JAVA.NET.SOCKET para conectar ao endereço e porta dados.

A partir daqui pode ser feita a transmissão de dados.

Isto poderá ser mais facilmente compreendido no esquema abaixo.

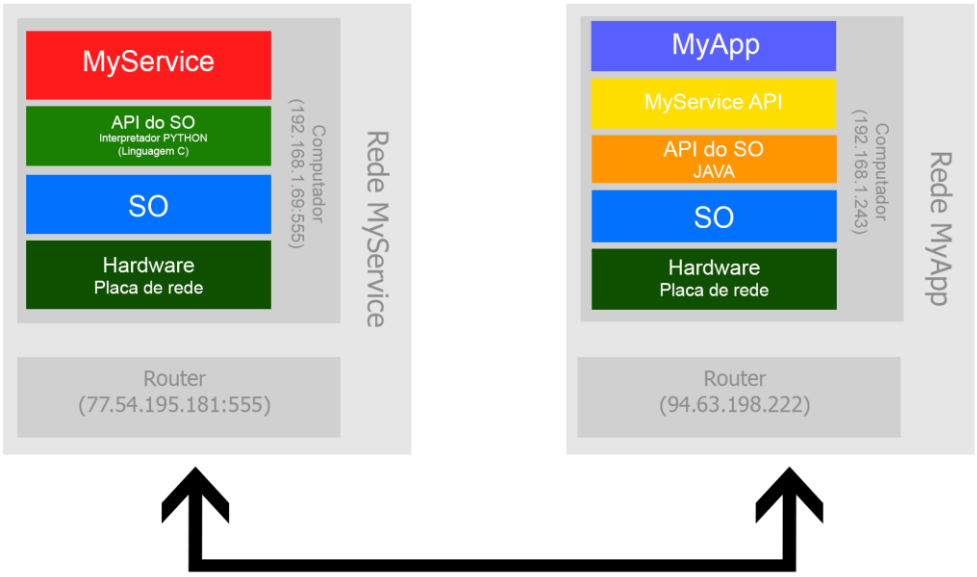

Figura 21 - Esquema representativo do funcionamento dos nossos programas de exemplo

A menção dos endereços IP da rede *"MyApp"* é feita apenas com o objetivo de tornar o esquema mais detalhado. Não é feito qualquer tipo de redireccionamento de portas nesta rede, nem é manipulado qualquer endereço da mesma nos códigos. Os endereços mostrados são os utilizados nas capturas de ecrã dos programas, mas poderiam ser quaisquer outros.

Todos os códigos de exemplo do trabalho podem ser vistos no apêndice A.

CONCLUSÃO

Neste trabalho tínhamos os objetivos de dominar sobre os conceitos relativos aos tipos de API, demonstrar a sua finalidade, e em que situações são utilizadas, e por fim construir uma API para fundamentar o trabalho e aprofundar os nossos conhecimentos, relativos tanto ao conceito de API do sistema como API em geral.

O objetivo mais pretendido pelo grupo foi concluído, que neste caso seria adquirir as noções de API e API do sistema. Conseguimos ainda com o projeto que criámos ter noção das potencialidades das APIs, pois a combinação de APIs do sistema com os nossos programas e a nossa API permitiu-nos comunicar entre grandes distâncias.

Concluindo, pudemos adquirir a noção de que existem vários tipos de API e identifica-las, assim como compreender em que circunstancias são usadas e como funcionam.

APÊNDICE A

O repositório que contém os códigos de exemplo do trabalho pode ser visto neste endereço:

https://github.com/carlostojal/TSI/tree/master/Programa%C3%A7%C3%A3o%20e%20Sistemas%20de%20Informa%C3%A7%C3%A3o/M8%20-%20Conceitos%20Avan%C3%A7ados%20de%20Programa%C3%A7%C3%A3o/APIs

REFERÊNCIAS BIBLIOGRÁFICAS

Azul, A. A. (2011). *Programação e Sistemas de Informação.*

Camargo, W. B. (desconhecido de deconhecido de 2010). *devmedia.com.br.* Obtido de devmedia: https://www.devmedia.com.br/propriedades-e-eventos-classes-programacao-orientada-a-objetos-parte-2/18577

Ciriaco, D. (24 de 3 de 2009). *https://www.tecmundo.com.br/programacao/o-que-e-api-.htm*. Obtido de TecMundo: https://www.tecmundo.com.br/programacao/1807-o-que-e-api-.htm

Desconhecido. (2017). *http://xzueqdro4oqen52i.onion/programming/object-oriented-programming/steps-in-oop.* Obtido de Webculture: http://xzueqdro4oqen52i.onion/programming/object-oriented-programming/steps-in-oop

Ferrari, C. (26 de Agosto de 2013). *pt.slideshare.net.* Obtido de pt.slideshare.net: https://pt.slideshare.net/cleytonferrari/programao-orientada-a-objetos-25598751

Jesus, C. d. (2013). *JAVA - Programação Orientada a Objetos.* Lisboa: FCA.

Pereira, B. (s.d.). *modulo8psdi.blogspot.com.* Obtido de modulo8psdi.blogspot.com: https://modulo8psdi.blogspot.com/p/conceitos-acerca-da-interface-com-o_10.html

Pinheiro, Á. F. (02 de 04 de 2017). *pt.slideshare.net/alvarofpinheiro/paradigma-orientado-a-objetos.* Obtido de https://pt.slideshare.net/alvarofpinheiro/paradigma-orientado-a-objetos-74172656

Soares, T. (28 de Março de 2018). *Medium.com.* Obtido de Medium.com: https://medium.com/@TDamiao/16-conceitos-poo-programa%C3%A7%C3%A3o-orientada-a-objeto-6cdc72ac3ee2