**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

**Índice**

[Agradecimentos iii](#_Toc20150566)

[Siglas/Abreviaturas iv](#_Toc20150567)

[Índice de Figuras 5](#_Toc20150568)

[Índice de Quadros/Tabelas 6](#_Toc20150569)

[Introdução 1](#_Toc20150570)

[1. Conceitos do módulo 8 2](#_Toc20150571)

[1.1. Conceito de Programação Orientada a Eventos 2](#_Toc20150572)

[1.1.1. Eventos 2](#_Toc20150573)

[1.1.2. Exemplo de Programação Orientada a Eventos 2](#_Toc20150574)

[1.2. Conceito de Programação Orientada a Objetos 2](#_Toc20150575)

[1.2.1. Objetos 2](#_Toc20150576)

[1.2.2. Exemplo de Programação Orientada a Objetos 2](#_Toc20150577)

[1.3. Conceito de Interface Gráfica 2](#_Toc20150578)

[1.4. Conceito de Programa 2](#_Toc20150579)

[1.5. Conceito de Atributo 2](#_Toc20150580)

[1.6. Conceito de propriedades 2](#_Toc20150581)

Agradecimentos

Siglas/Abreviaturas

A Programação Orientada a Objetos – POO 8

Índice de Figuras

Índice de Quadros/Tabelas

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 SO”,

Com este tema 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.

**Conceitos do módulo 8**

1. Conceitos do módulo 8
   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. Eventos
    2. Exemplo de Programação Orientada a Eventos
  1. Conceito de Programação Orientada a Objetos

A Programação Orientada a Objetos – POO;

* + 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. Exemplo de Programação Orientada a Objetos
  1. Conceito de Interface Gráfica
  2. Conceito de Programa
  3. Conceito de Atributo
  4. Conceito de propriedades

1. **Momentos de trabalho na Programação Orientada a Objetos**
2. **Tema**

**3.1.1 O que é uma API?**

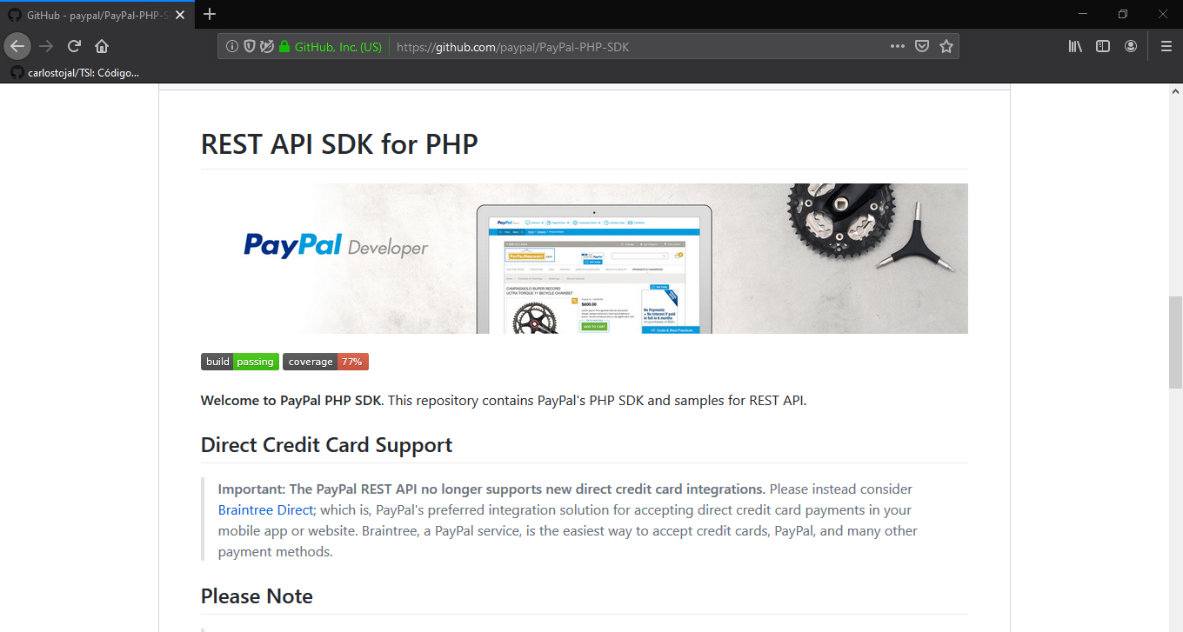
API é o acrónimo de Application Programming Interface ou, em português, Interface de Programação de Aplicativos. (Ciriaco, 2009) .

Tal como uma GUI (Graphics User Interface) 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.



**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.

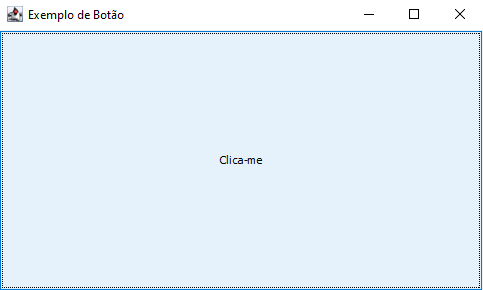
As APIs que vamos abordar são as APIs do sistema.

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

**3.1.3 Em que circunstâncias são usadas APIs do sistema**

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

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



**3.1.4 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.

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.

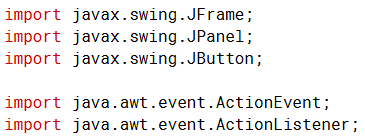
Isto demonstra ainda que se estas APIs não existissem e o desenvolvedor tivesse de criar a conexão entre os seus programas e o sistema operativo todas as vezes que criasse uma aplicação, seria um trabalho muito cansativo, e a evolução da programação e até mesmo da tecnologia (que está dependente da programação) 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.

**3.1.5 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.

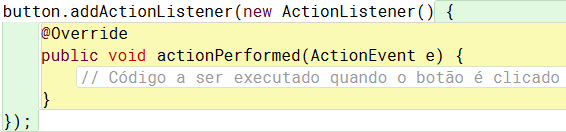
Para podermos criar um botão num programa em JAVA, temos de importar várias classes, de entre elas a JAVAX.SWING.JBUTTON, JAVA.AWT.EVENT.ACTIONEVENT e JAVA.AWT.EVENT.ACTIONLISTENER, como está demonstrado no exemplo abaixo.



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.



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.



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.



**3.1.6 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.



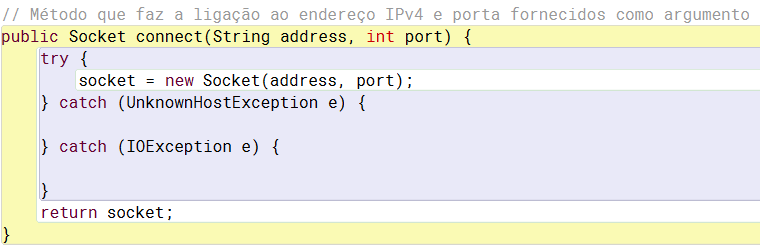
A esses programas chamámos respetivamente *"MyService"*, “*MyApp”* e *"MyService API"*. Demos estes nomes apenas para facilitar a diferenciação de cada um dos códigos.

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

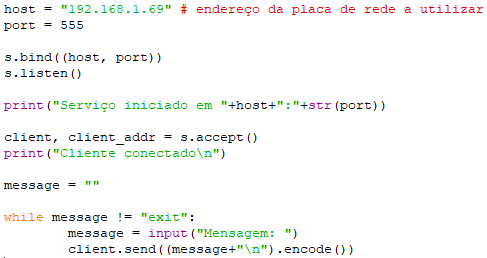
Tanto esta API quanto os 2 outros programas utilizariam também APIs do sistema. No caso do programa *“MyApp”* e da API utilizariam APIs do sistema JAVA, e no caso do programa *“MyService”* da linguagem C. Porquê da linguagem C e não do PYTHON será explicado mais à frente.

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"*.

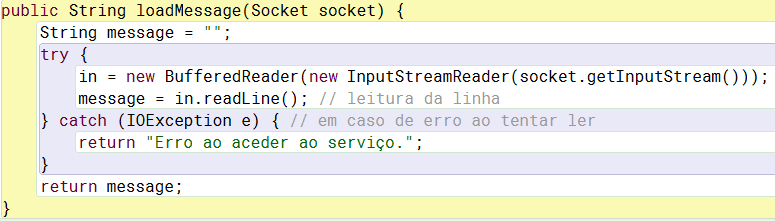
De forma simples, o programa *“MyApp”* conectar-se-ia ao programa *“MyService”* através da classe *“MyService API”*, o programa *"MyService"* enviaria as mensagens através de sockets de rede em PYTHON para o endereço IP e porta do programa *“MyApp”*.

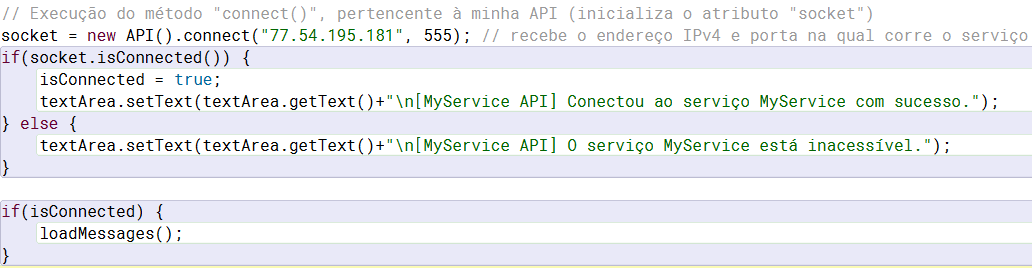


Nesta parte são utilizadas APIs do sistema na classe *“MyService API”*, para se conectar ao programa “MyService”, que também utiliza APIs do sistema para comunicar com a classe *“MyService API”*.



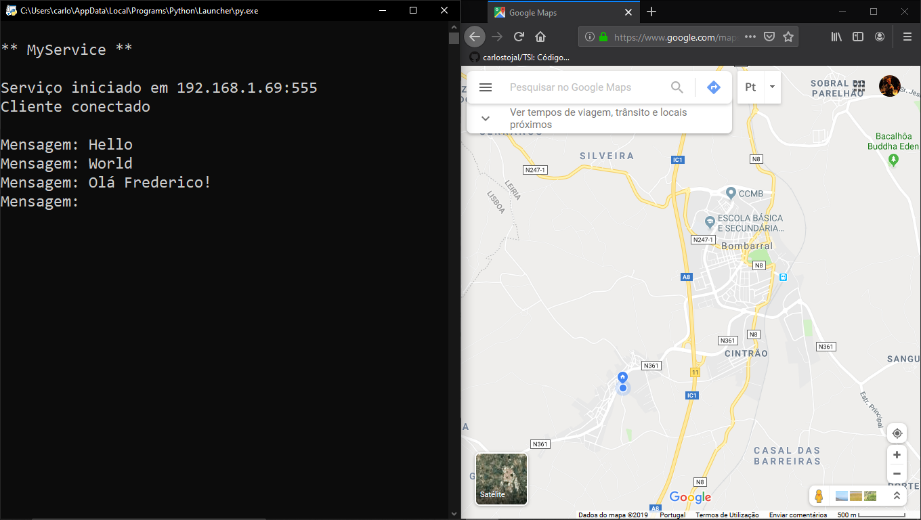
De seguida, as mensagens chegariam ao destino e seriam lidas utilizando a classe JAVA.IO.BUFFEREDREADER, que leria o input stream do socket, obtido através do método SOCKET.GETINPUTSTREAM(), código esse presente na classe *“MyService API”*.

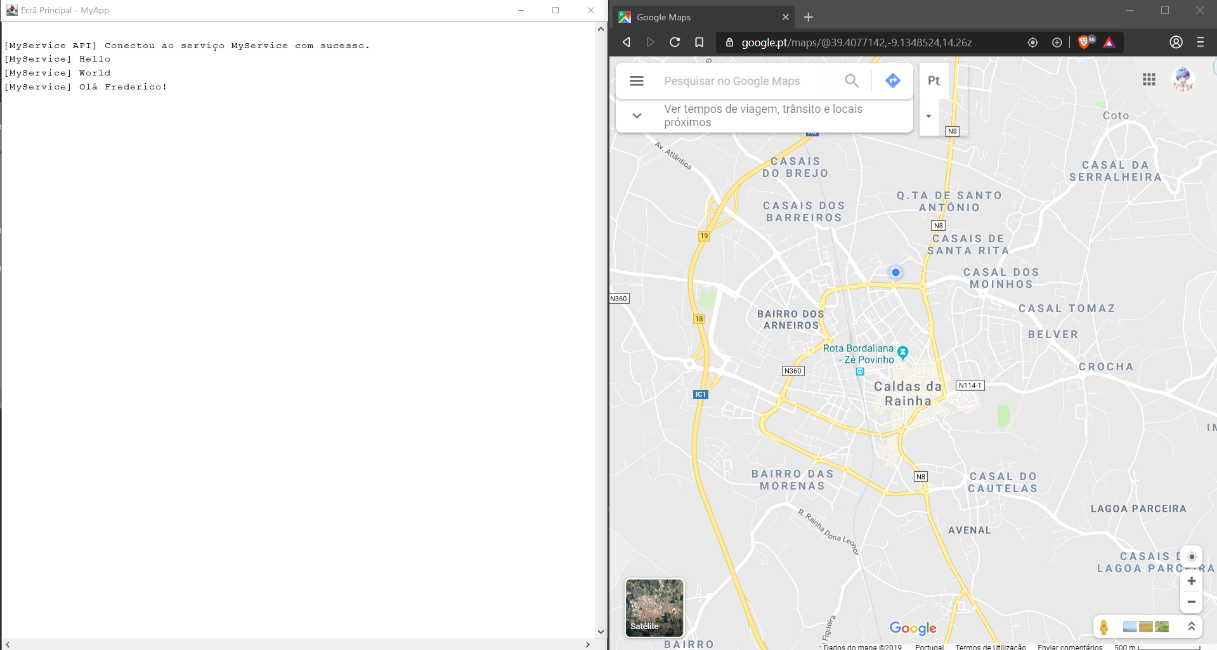


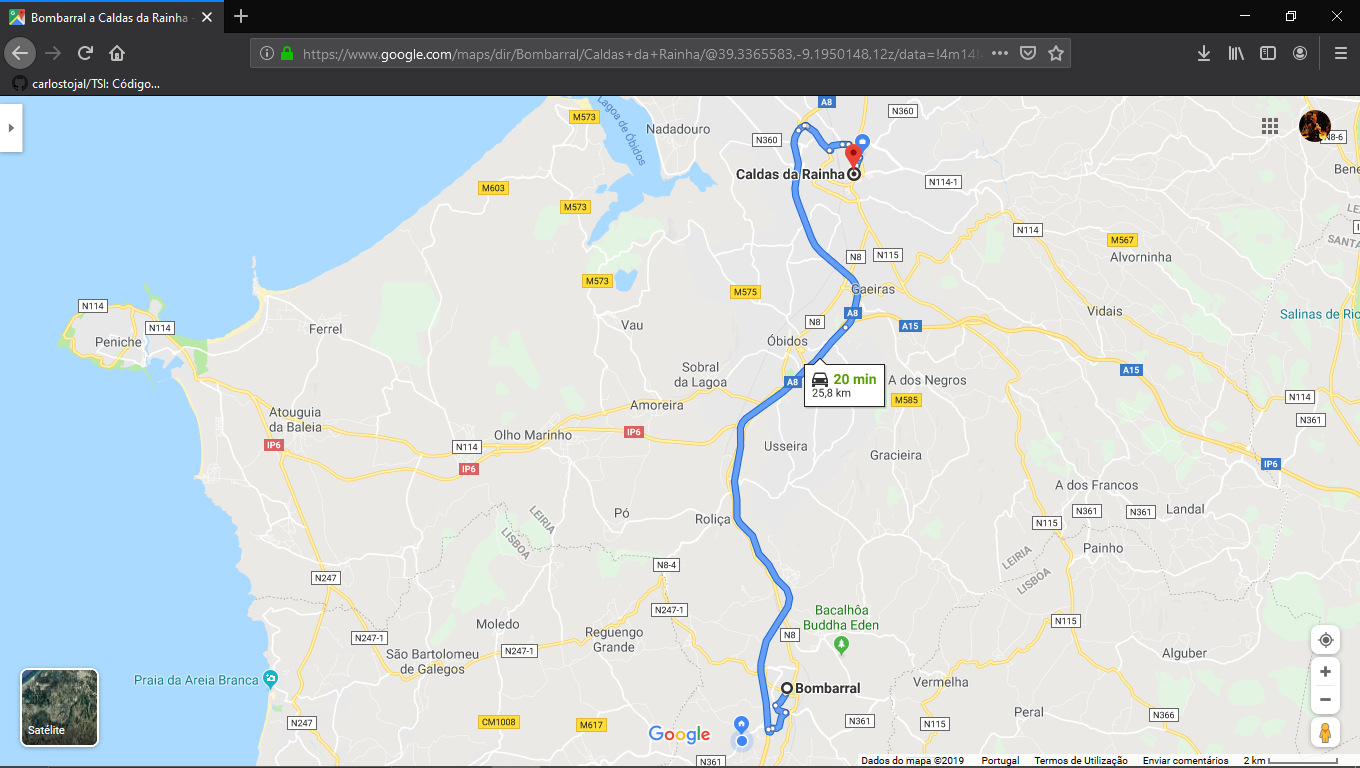
Neste último passo são utilizadas pelo menos duas APIs do sistema: uma para o acesso ao sistema de ficheiros para fazer a leitura, e outra de rede para obter o input stream do socket. Depois disto as mensagens são recebidas pela *"MyService API"*, que as envia para a *"MyApp"*.

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.

**3.1.7 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.

****

****

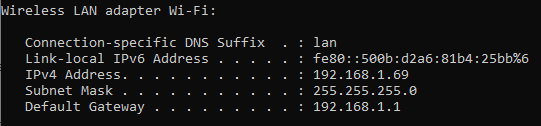
Para ser possível comunicar entre programas fora da mesma rede, tiveram de ser feitas configurações na rede em que ficou hospedado o programa *“MyService”*.

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

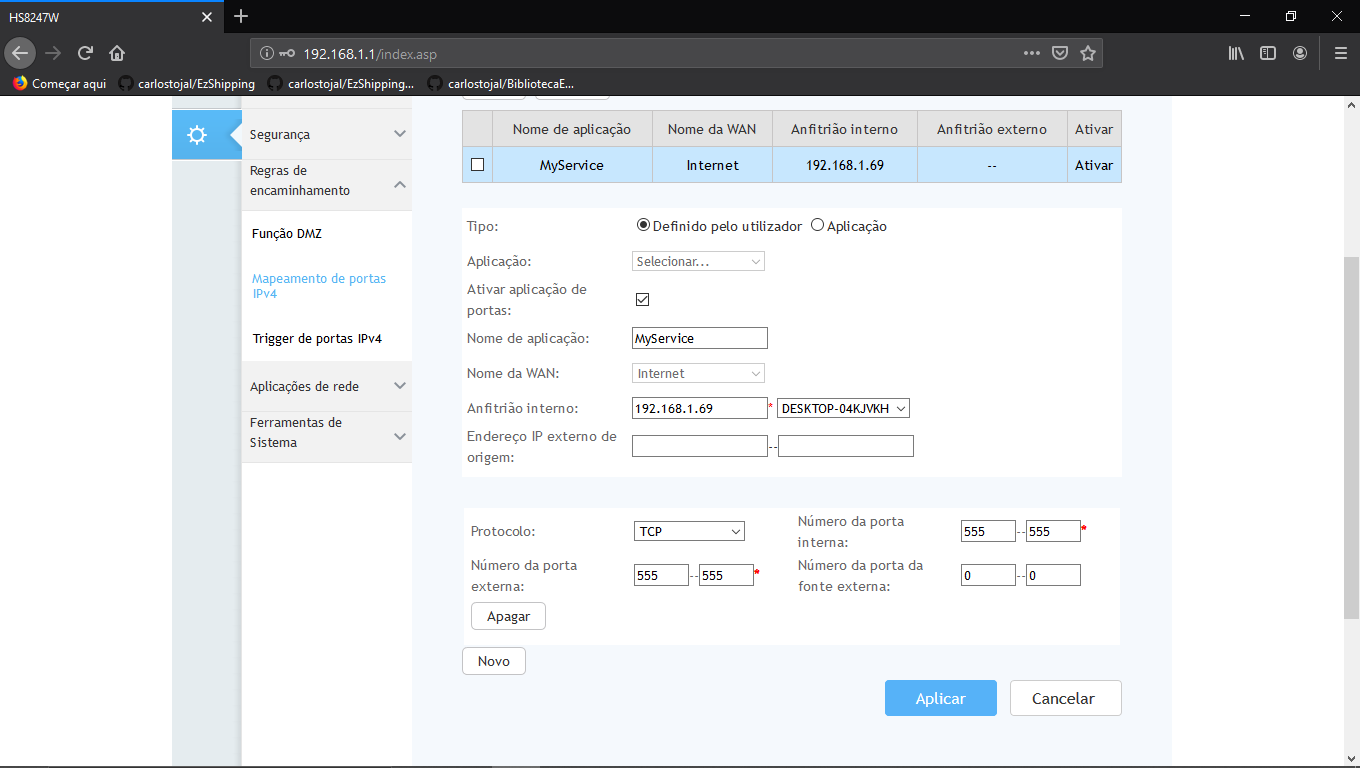
Aqui pode ser visto que o endereço IP do router no qual ficou hospedado o programa *“MyService”* é 77.56.195.181.



Aqui que o endereço IPv4 da placa de rede utilizada na hospedagem deste mesmo programa é 192.168.1.69.



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.



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.



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 sistemapara estabelecer uma conexão com o endereço IP 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.



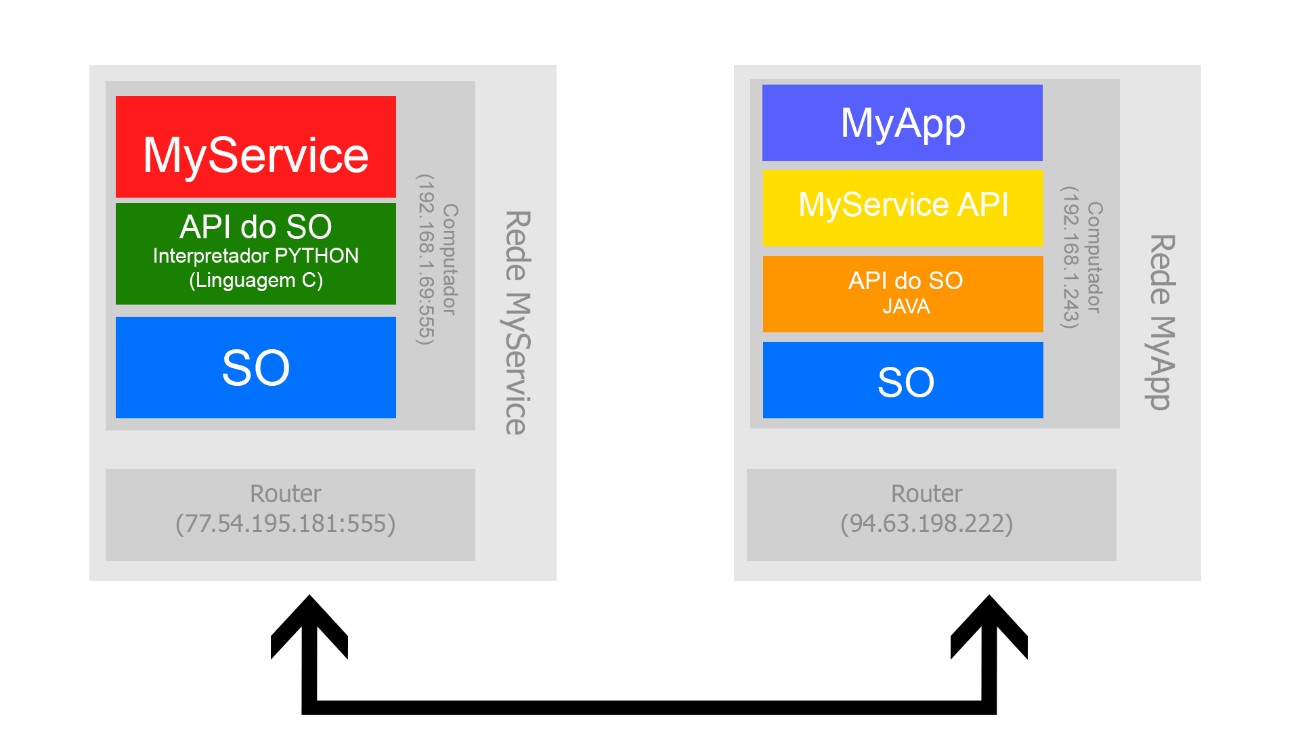
Esta linha está presente nesse mesmo método, em que *address* é o primeiro argumento (neste caso 77.54.195.181) e *port* o segundo (neste caso 555).



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 da página seguinte.



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 manualmente nesta rede.

Todos os códigos de exemplo do trabalho podem ser vistos em <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>.

**CONCLUSÃO**

**REFÊRENCIAS BIBLIOGRÁFICAS**

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

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

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

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