PROGRAMAR

REVISTA PORTUGUESA DE PROGRAMAÇÃO | WWW.PORTUGAL-A-PROGRAMAR.PT | ISSN 1647-0710

EDIÇÃO #55 - MARÇO 2017

DOCKER: OVERVIEW

A PROGRAMAR

API REST COM SPRING BOOT (PARTE 2)

BOT JOGO DO GALO

AZURE AUTOMAÇÃO COM WINDOWS POWERSHELL DESIRED STATE CONFIGURATION

GALE SHAPLEY O PROBLEMA DO CASAMENTO ESTÁVEL COM O ALGORITMO

ELECTRÓNICA

PROBLEMA A FALTA DE GPIO PINS

COLUNAS

C#

KERNEL PANIC

SQL CURTAS #2 DÚVIDAS COMUNS

ANÁLISES

C# 6 PROGRAMAÇÃO COM PRODUTIVIDADE

PYTHON INTRODUÇÃO À PROGRAMAÇÃO, ALGORITMOS E LÓGICA DE PROGRAMAÇÃO PARA INICIANTES

NO CODE

SHIFT APPENS 4ª EDIÇÃO - COIMBRA

RASPBERRY PI ZERO W

EDITORIAL

EQUIPA PROGRAMAR

Coordenador
António Pedro Cunha Santos

Editor
António Pedro Cunha Santos

Capa
Filipa Peres

Redacção
Augusto Manzano
André Melancia
António Pedro Cunha Santos
Filipa Peres
José Martins
Mónica Rodrigues
Nilo Menezes
Nuno Cancelo
Ricardo Cabral
Rita Peres

Staff
António Pedro Cunha Santos
Rita Peres
Tiago Sousa

Contacto
revistaprogramar@portugal-a-programar.org

Website
http://www.revista-programar.info

ISSN
1 647-071 0

/* A todos os bravos que chegaram tão longe! */

Poderia começar o editorial por escrever o resto do comentário em código, mas seria quase um "abuso" ao qual não me vou dar!

Como um dia disse, um incontornável personagem da história da tecnologia, não se conectam os pontos olhando para a frente, mas sim para traz. A tecnologia é isso mesmo, um movimento "perpétuo", em frente, sem parar, sem esperar, sem pausas, a uma velocidade cada vez mais estonteante. Cheio de surpresas e segredos, cheio de revezes e avanços, cheio de tudo um pouco! Mas acima de tudo, cheio! Cheio porque tem um pouco de todos os que nele trabalham, participam, se envolvem! Sem distinção de géneros, classes ou outras que possam existir!

Recentemente comemorou-se o dia mundial da mulher, daqui a uns meses assinala-se o dia mundial da criança, e lá para novembro, bem despercebido, em alguns países, comemora-se o dia internacional do homem e lá bem perto do final do ano, quase que sem se dar por isso, o dia internacional dos Direitos Humanos, sem distinções! E tudo isto para dizer que a tecnologia existe para servir todos os seres humanos! A Programação é algo acessível a todos, sem excepção! Dos que leem avidamente livros de programação, e aprendem a programar lendo e escrevendo em papel linhas e linhas de código, até aos que usam o último modelo de computador, para dar asas à sua imaginação e programar!

Com tantos dias mundiais, falta mencionar um que nos diz muito a todos, programadores profissionais, programadores aprendizes, e mesmo até aqueles que só programam por desafio de quando em vez! O ducentésimo quinquagésimo sexto dia do ano, que nos anos não bixestos calha no 13 de Setembro e nos bissextos no 12. É o dia do Programador! Pois 2^8 é o número de valores distintos que podem ser representados com um byte de oito bits.

Enquanto esse dia não chega, e os dias vão passando, programem! Que programar comanda a vida! E comentem o código, para vos rirdes quando o leres anos volvidos!

Até à próxima edição, boas leituras!

António Santos

```
/** A todos os bravos que chegaram tão longe!
 *  Que programaram durante tantos dias
 *  Que leram todo este código, e chegaram a este comentário
 *  Sede pacientes! Que a jornada é longa!
 *  Cuidado com os bugs! Que eles não vos atrapalhem!
 *  E lembrem-se deste velho ensinamento:
 *  Quando eu escrevi este código,
 *  só eu e Deus sabíamos o que ele queria dizer!
 *  Agora só um de nós sabe! E não sou eu!!!
 **/

public void Asm()
{
    console.writeln("Feliz é quem programa!");
}
```

ÍNDICE

TEMA DE CAPA

A PROGRAMAR

ELECTRÓNICA

COLUNAS

ANÁLISES

SEGURANÇA

NO CODE

EVENTOS

16 de Março - CiberSegurança e Cibercrime
25 - 27 de Maio - ISELTech'17

Para mais informações/eventos: http://bit.ly/PAP_Eventos. Divulga os teus eventos para o email eventos@portugal-a-programar.pt

NOTÍCIAS

Jovens de Coimbra na final do Google Hash Code 2017

Terá lugar em Paris, no próximo dia 1 de Abril, a final do Google Hash Code 2017, promovido pela Google. O Hash Code é um concurso que visa estudantes e profissionais da Europa, Médio Oriente e África, consistindo na resolução de problemas criados pela Google através de uma linguagem de programação. Os participantes podem formar equipas entre dois e quatro elementos, dividindo-se a competição em duas fases: online e final com os 50 melhores classificados.

Três estudantes naturais de Coimbra participaram na primeira fase que decorreu em Fevereiro e contou com a participação de 2815 equipas. Ricardo Gomes, da Universidade de Coimbra, Miguel Duarte, do Instituto Superior Técnico da Universidade de Lisboa e Pedro Paredes, da Universidade do Porto, vão disputar a grande final do Google Hash Code 2017.

Em entrevista Ricardo Gomes afirma "Sinceramente, não contava que a fase de qualificação corresse assim tão bem. Nós tínhamos uma estratégia inicial e, simplesmente, seguimos essa estratégia. Esperamos superar em Paris o resultado obtido nesta fase".

Imagem 1 - Miguel Duarte, Ricardo Gomes, Pedro Paredes

Imagem 2 - Pedro Paredes, Ricardo Gomes, Miguel Duarte

Fonte: « Assessoria de Imprensa Universidade de Coimbra»

Visual Studio 2017 lançado a 7 de Março

No passado dia 7 de Março, a Microsoft lançou a ultima versão do seu ambiente de desenvolvimento Visual Studio.

Esta nova versão da já famosa IDE, desta feita recheada de novidades, além de algumas melhorias em relação à versão anterior, bem como algumas novidades e continua a ser disponibilizado em diversas versões, sendo que se mantem a existência de uma versão community gratuita.

De todas as novidades desta nova versão do Visual Studio destacamos as seguintes:

- Ajuda Off-line, bastando instalar o Help Viewer
- Código colorido e autocomplete para as linguagens Bat, Clojure, CoffeeScript, CSS, Docker, F#, Groovy, INI, Jade, Javadoc, JSON, LESS, LUA, Make, Markdown ++, Objective-C, Perl, PowerShell, Python, Rust, ShaderLab, SQL, Visual Basic .NET, YAML.
- Snippets de código para as linguagens CMake, C++, C#, Go, Groovy, HTML, Java, Javadoc, JavaScript, Lua, Perl, PHP, R, Ruby, Shellscript, Swift, XML
- Suporte nativo para as linguagens C++, C#, Go, Java, JavaScript, PHP, TypeScript, Visual Basic
- O compilador de C++ melhorado com suporte para C++ 11 e C++ 14, além de suporte preliminar para algumas funcionalidades no futuro C++ 17.
- Suporte para CMake
- Suporte para desenvolvimento para GNU/Linux com C++
- Desenvolvimento móvel com C++ (Android e iOS)
- Suporte para C# 7.0 e Visual Basic 15
- Suporte ao uso de tuples para agrupamento temporário em memória, de conjuntos de valores tipados.
- Suporte para TypeScript 2.1
- Suporte melhorado para Apache Cordova, incluindo um simulador baseado no browser, que permite ver resultados quase imediatos.

A lista de novidades era demasiado longa para incluir aqui, por esse motivo apenas foram focadas algumas, deixando o link para o site do Visual Studio onde poderão ser encontradas todas as novidades e melhorias.

https://www.visualstudio.com/en-us/news/releasenotes/vs2017-relnotes

TEMA DE CAPA

Docker: overview

Docker: overview

Sou muito apologista da metodologia *"set it and forget it"*, configurar as coisas uma vez e reutilizar vezes sem conta a mesma configuração, infraestrutura. Abstrairmos de tal forma, que o foi configurado sirva para o uso geral da nossa aplicação ou projeto. Isto é muito giro, mas pouco realista se tivermos em mente a montanha de projetos e aplicações que estão montadas por Portugal (e não só) a fora.

Tipicamente, a forma como eu fazia, seria criar uma máquina virtual (principalmente em virtualbox) montava a infraestrutura da forma que queria e depois trabalhava sobre ela e partilhava a imagem com quem quisesse. Apesar de funcionar a solução não era em nada elegante, tinha uma imagem com cerca de 20GB, com um sistema operativo (que poderia estar ou não a usar as suas potencialidades), mais o conjunto de ferramentas e ainda tinha que me preocupar com as configurações de rede (para estar exposto para fora da máquina virtual) e ter *mounting points* para poder partilhar ficheiros entre o *host* e a máquina virtual. Quem já fez isto pelo menos uma vez sabe a chatice que dá.

Em 2013 quando comecei a ouvir falar de Docker e o que representava naturalmente fiquei bastante interessado. E ao longo do tempo a empresa tem desenvolvido bastante trabalho e introduzindo bastantes *features*, apresentando um produto de alta qualidade que depressa os gigantes da cloud (AWS, Google Cloud, Microsoft Azure, etc) agarraram a oportunidade de integrar o Docker nos seus seus serviços em forma de *Container Services.*

Afinal o que é o Docker

Docker é um projeto open-source que permite criar um contentor (*container*) com a infraestrutura que se pretender por forma a poder partilha-la em qualquer máquina e manter sempre o mesmo comportamento. Este contentor contem um sistema operativo Linux *lightweight*, com o tradicional sistema de ficheiros e politicas de segurança que caracterizam o sistema operativo Linux. Desta forma usa bastante menos recursos de memória e de espaço em disco.

Um *container* é uma máquina virtual então? Bem a resposta é ... mais ou menos. Não é uma máquina virtual no sentido tradicional a que estamos mais habituados. Vou usar uma das analogias para diferenciar o *Container* Docker de uma máquina virtual:

"Um Container Docker está para uma Máquina Virtual, como uma Thread está para um Processo."

Ou seja, acaba por ser uma *Lightweight Virtual Machine*.

A criação de *Container* segue a filosofia de blocos de construção (*building blocks*). Pensem em peças da Lego, querem construir uma parede e para tal vão colocando peças atrás de peças. O mesmo acontece com o Docker, vão compondo o vosso *container* camada por camada:

1. Instalar o Sistema Operativo (ex: Ubunto)
2. Instalar uma ferramenta (ex: Node)
3. Instalar uma base de dados (ex: Postgres)
4. Etc

Quando estiver composto o vosso *stack* aplicacional está pronto a ser feito build e a distribuírem por quem entenderem.

A imagem Figura 1mostra um *stack* tradicional em Docker.

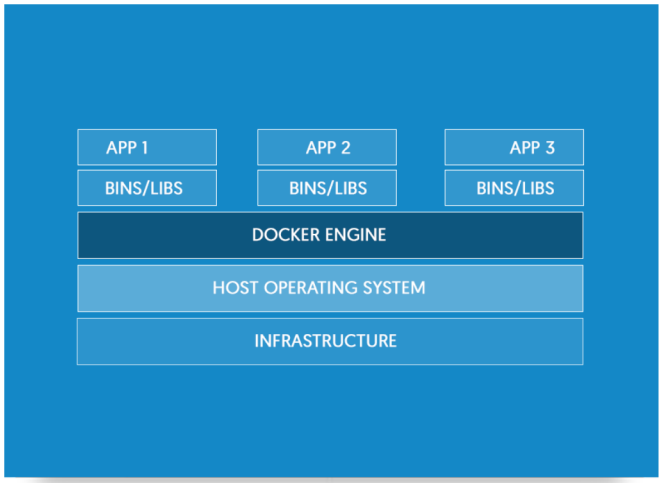


Figura *1: Docker Stack*

Requisitos de Sistema e Instalação

Docker não tem muitos requisitos de sistema, mas os que têm são importantes para temos uma instalação sempre a funcionar.

No endereço http://bit.ly/Docker-System-Requirements podemos encontrar todos os requisitos, sendo os que são mais relevantes os seguintes (no entanto é muito recomendado lerem todos):

- Linux kernel version 3.10 or higher
- 2.00 GB of RAM
- 3.00 GB of available disk space

O URI refere os requisitos para "*Docker Universal Control Plane*", tipicamente para instalar em Cloud ou On-Permisses, mas os requisitos são semelhantes para os nossos portáteis.

Para proceder à instalação basta aceder ao endereço http://bit.ly/Docker-Install e seguir atentamente as instruções. Para os sistemas operativos mais antigos (não demasiado, claro J) cada pagina refere como instalar a Docker Toolbox,

no entanto chamo a atenção que este "produto" não é suportado e as atualizações já não são tão frequentes.

O *wizard* instala três componentes principais:

- Docker Engine: o Docker em si.
- Docker Compose: permite compor imagens Docker
- Docker Machine: permite configurar a *Virtual Machine* onde o Docker instancia os *Containers*

É instalado também uma aplicação Kitematic, ainda está em versão Beta e precisa de algumas melhorias, que permite de uma forma mais gráfica pesquisar imagens no Docker Hub e gerir o ciclo de vida de um *container*.

Nas versões MacOS e Windows o Docker é instalado como serviço, o que permite arrancar o Docker quando o sistema operativo arrancar. Desconheço se em Linux o comportamento é semelhante.

OK, está instalado e agora?

Bem, os primeiros passos passam por conhecer a aplicação, mas recomendo que encontrem um objetivo (por mais simples que seja) para que a aprendizagem seja a mais proveitosa possível.

Se seguiram o manual de instalação, já perceberam que para se trabalhar com Docker é necessário trabalhar em linha de comando. Por incrível que possa parecer não é assim tão intragável como possa parecer J. O passo 2 (versão mac) pede para verificar a versão de cada uma das componentes:

```
HostApple:~ nunocancelo$ docker --version
Docker version 1.13.0, build 49bf474
```

```
HostApple:~ nunocancelo$ docker-machine --version
docker-machine version 0.9.0, build 15fd4c7
```

```
HostApple:~ nunocancelo$ docker-compose --version
docker-compose version 1.10.0, build 4bd6f1a
```

A versão de cada componente é relevante quando se está a "brincar", pois em cada versão nova o comportamento poderá ser diferente e alguns ajustes poderão ter que ser feitos.

O comando mais útil, e que no inicio é usado bastante, é o de ajuda:

```
HostApple:~ nunocancelo$ docker help
```

(Por questões de legibilidade não é apresentado o resultado)

O resultado apresenta todos os comandos suportado por esta versão do Docker, e para obter ajuda de um determinado comando:

```
HostApple:~ nunocancelo$ docker COMMAND --help
```

Pessoalmente desconheço o comportamento de grande parte dos comandos, porem sigo a regra dos 20-80:

Conhecer 20% dos comandos que servem para 80% das vezes.

Não é necessário ser um expert em comandos Linux, mas ajuda já ter trabalhado com ambientes Unix, uma vez que grande parte dos comandos têm uma filosofia semelhante, mas em contexto diferente.

Por exemplo:

O comando:

```
HostApple:~ nunocancelo$ docker ps
```

Mostra os containers ativos que estão a correr na máquina e com o id de um container podemos executar o comando:

```
HostApple:~ nunocancelo$ docker kill CONTAINER_ID
```

Que "mata" o container com esse id.

Esta é uma pequena lista dos comandos que mais uso:

- attach: Attach to a running container
- exec: Run a command in a running container
- images: List images
- info: Display system-wide information
- inspect: Return low-level information on Docker objects
- kill: Kill one or more running containers
- logs: Fetch the logs of a container
- ps: List containers
- pull: Pull an image or a repository from a registry
- rm: Remove one or more containers
- rmi: Remove one or more images
- run: Run a command in a new container
- search: Search the Docker Hub for images
- stats: Display a live stream of container(s) resource usage statistics

Todos restantes também são importantes, mas não os uso com tanta frequência.

Antes de avançarmos é necessário esclarecer os pontos bastantes importantes:

- Como já foi mencionado anteriormente, um *Container* é uma *Lightweight Virtual Machine,* mas é necessário mencionar que uma *Image* é um stack aplicacional previamente definido. Não são a mesma coisa.
- Uma *Image* é readonly, uma vez criada não pode ser alterada. Se for necessário alterar alguma configuração

ou adicionar algo é necessário criar uma nova *Image*.

- Sendo que uma *Image* (que é readonly) vive dentro de um *Container* quando se sai (exit) desse container, toda a nova informação que lá foi criada perder-se aquando a saída do Container. Porém há formas de salvaguardar a informação J.

Docker Hello World

Para não variar, vamos criar um Hello World para começarmos a interagir com o Docker e ver as suas potencialidades.

Vou replicar o exemplo do manual de instalação do passo 3 (versão mac) e instalar uma imagem Hello-World:

```
   HostApple:~ nunocancelo$ docker run hello-
world
   Unable to find image 'hello-world:latest' lo-
cally
   latest: Pulling from library/hello-world
   78445dd45222: Pull complete
   Digest:
sha256:c5515758d4c5e1e838e9cd307f6c6a0d620b5e07e6
f927b07d05f6d12a1ac8d7
   Status: Downloaded newer image for hello-
world:latest

   Hello from Docker!
   This message shows that your installation ap-
pears to be working correctly.

   To generate this message, Docker took the fol-
lowing steps:
    1. The Docker client contacted the Docker
daemon.
    2. The Docker daemon pulled the "hello-world"
image from the Docker Hub.
    3. The Docker daemon created a new container
from that image which runs the
       executable that produces the output you
are currently reading.
    4. The Docker daemon streamed that output to
the Docker client, which sent it
       to your terminal.

   To try something more ambitious, you can run
an Ubuntu container with:
    $ docker run -it ubuntu bash

   Share images, automate workflows, and more
with a free Docker ID:
    https://cloud.docker.com/

   For more examples and ideas, visit:
    https://docs.docker.com/engine/userguide/
```

Quando o comando é executado, o comportamento por omissão é verificar se a imagem pretendida (no nosso caso Hello-World) existe localmente. Se existir executa, senão vai ao repositório (Docker Hub) tentar obter a imagem e instala a imagem como todas as suas dependências.

```
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
78445dd45222: Pull complete
Digest:
sha256:c5515758d4c5e1e838e9cd307f6c6a0d620b5e07e6f9
27b07d05f6d12a1ac8d7
Status: Downloaded newer image for hello-
world:latest
```

Neste exemplo, a imagem Hello-World o que mostra são todos os passos que fez até concluir:

```
Hello from Docker!
This message shows that your installation appears
to be working correctly.

To generate this message, Docker took the follo-
wing steps:
 1. The Docker client contacted the Docker dae-
mon.
 2. The Docker daemon pulled the "hello-world"
image from the Docker Hub.
 3. The Docker daemon created a new container
from that image which runs the
    executable that produces the output you are
currently reading.
 4. The Docker daemon streamed that output to the
Docker client, which sent it
    to your terminal.
```

Para começo está porreiro, mas todos concordamos que não é útil (como todos os Hello-World :P). Então vamos tentar algo mais.

Continuando o manual vamos instalar um servidor NGINX, para quem não sabe é um servidor HTTP de alta performance, e aqui já temos algumas novas features porreiras.

```
HostApple:~ nunocancelo$ docker run -d -p 80:80 --
name webserver nginx
Unable to find image 'nginx:latest' locally
latest: Pulling from library/nginx
5040bd298390: Pull complete
333547110842: Pull complete
4df1e44d2a7a: Pull complete
Digest:
sha256:f2d384a6ca8ada733df555be3edc427f2e5f285ebf46
8aae940843de8cf74645
Status: Downloaded newer image for nginx:latest
6a9862f69870db29401404b40fe4247068c609791160cf60060
f0067bf3c2679
```

Como se pode ver, a máquina tinha a imagem localmente e foi o repositório buscar e instalou três componentes.

Analisando melhor o comando:

```
HostApple:~ nunocancelo$ docker run -d -p 80:80 --
name webserver nginx
```

Vai correr num novo container a imagem nginx, o –d significa que vai correr em modo detach (isto é vai correr em background), o –p 80:80 significa que o container expõe o porto 80 e mapeia internamente com o porto 80 e, por fim, o --name permite atribuir um nome ao container. Por omissão todos os containers têm um nome (e são únicos), podem é não ser nada elucidativos. Em seguida vamos ter um exemplo em que dá para perceber(corri o mesmo comando mas em a opção --name).

A imagem Figura 2 mostra o output:

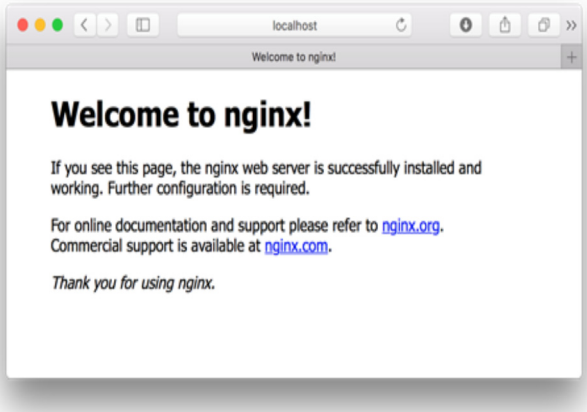


Figura 2: nginx

Vamos correr o comando ps para vermos a informação exposta (por questão de legibilidade algumas colunas menos relevantes foram removidas):

```
HostApple:~ nunocancelo$ docker ps
IMAGE PORTS                                    NAMES
nginx    443/tcp, 0.0.0.0:81->80/tcp
                                               stupefied_clarke
nginx    0.0.0.0:80->80/tcp, 443/tcp           webserver
```

Como se pode ver, a informação mostra a *Image* que estamos a usar, os portos utilizados e o nome atribuído ao *Container*. Na primeira linha temos o nome atribuído automaticamente, apesar de engraçado não é muito útil para identificar o que temos a correr lá dentro.

Para parar o Container:

```
HostApple:~ nunocancelo$ docker stop webserver
```

Para reiniciar o Container:

```
HostApple:~ nunocancelo$ docker start webserver
```

Para remover o Container:

```
HostApple:~ nunocancelo$ docker rm  webserver
```

Para remover a Image:

```
HostApple:~ nunocancelo$ docker rmi  nginx
```

Podemos dar os parâmetros que quisermos pela linha comandos, mapear portos, mapear volumes, executar comandos, etc. Pessoalmente recomendo criar um script que tenha toda a configuração pretendida para executar de uma só vez. Novamente, “Set it and Forget it”.

Next Step: Dockerfile

Isto é muito útil para usar em imagens já feitas, mas e se não encontrar nenhuma imagem que satisfaça as minhas necessidades? Ou então, quero fazer um de raiz? Para atender a esta necessidade foi criado o Dockerfile, que é um ficheiro de texto com um conjunto de diretivas que permite criar imagens.

O manual de referência do Dockerfile pode ser encontrado em http://bit.ly/Dockerfile-Reference.

Como exemplo, vamos supor que temos um site estático que queremos disponibilizar. Para tal, o melhor é começar pelo inicio J.

Criamos uma pasta raiz e dentro desta pasta vamos criar uma outra pasta onde vai estar o nosso site e um ficheiro chamado Dockerfile com o seguinte conteúdo muito simples.

Dockerfile

```
FROM httpd:2.4
COPY ./public-html/ /usr/local/apache2/htdocs/
```

Neste ficheiro usamos duas diretivas:

FROM: Imagem a partir da qual nós vamos trabalhar.

COPY: depois da imagem ser obtida copiamos o nosso site.

Agora vamos compilar o nosso Dockerfile

```
HostApple:pap-httpd nunocancelo$ docker build -t
                                    masterzdran/pap-httpd .
Sending build context to Docker daemon 10.75 kB
Step 1/2 : FROM httpd:2.4
2.4: Pulling from library/httpd
5040bd298390: Already exists
63408554ba61: Pull complete
4dfe9ef0af52: Pull complete
84871cd4d3da: Pull complete
b376204f8aa6: Pull complete
2a8d452c2c14: Pull complete
7cfa9c4a8891: Pull complete
Digest:
sha256:1407eb0941b010035f86dfe8b7339c4dd4153e2f7
                                    b855ccc67dc0494e9f2756c
Status: Downloaded newer image for httpd:2.4
 ---> 67a5f36fd844
Step 2/2 : COPY ./public-html/ /usr/local/apache2/htdocs/
 ---> b32140ef85b6
Removing intermediate container 8fd6889baf0f
Successfully built b32140ef85b6
```

Estamos a fazer build, criamos uma tag (com o -t) e porque usamos a nomenclatura do ficheiro recomendada, indicamos onde está o ficheiro para fazer o build (.).

Ele começa a fazer a sua magia.

Quando termina fazemos:

```
HostApple:pap-httpd nunocancelo$ docker run -dit -
p 80:80 --name pap-httpd masterzdran/pap-httpd
```

E quando formos ao browser obtemos a imagem Figura 3.

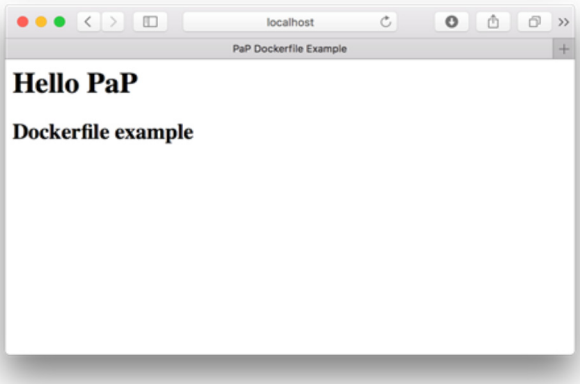


Figura 3: Dockerfile build example

Se repararem o comando tem dois novos parâmetros o –t e o –i, e isto significa que querem que o container tenha um terminal iterativo.

Agora que tem o container online e a correr em background, podem ver os logs do servidor http com:

```
HostApple:pap-httpd nunocancelo$ docker logs  pap
-httpd
```

Porque os nomes dos containers são únicos, tão únicos como o seu ID, podemos referenciar os container pelo seu nome ao invés do seu id. Neste exemplo queremos ver os logs do container pap-httpd.

Por vezes é necessário entrar no container para realizar operações, ver ficheiros de configuração, etc. A maneira mais simples de entrar dentro de um container é através do comando exec:

```
HostApple:~ nunocancelo$ docker exec -ti
                                    pap-httpd bash
root@47d280db122c:/usr/local/apache2#
```

Estamos a dizer que queremos um terminal interactivo (-ti) do container pap-httpd e executa o comando bash. E depois temos uma linha de comandos dentro do container.

Podemos correr um comando com o docker run? A resposta é depende da imagem que estamos a usar. Neste exemplo a nossa imagem “lança” um servidor httpd, se colocarmos um comando no fim para ser executado ele vai fazer overwrite do CMD (último comando de “arranque” da imagem) da aplicação e o servidor não arranca, contudo ficamos com a linha de comandos do container disponível.

E agora para sair? Bem, não executem o “exit” ou carregar em ctrl+c, ctrl+d, porque todos eles indicam ao container que é para fazer shutdown.

Para fazer detach do container pressionem ctrl+p ctrl+q.

Por vezes é necessário ver as configurações de uma imagem para obter informações de networking, mounting points, etc. Para obter essa informação usamos o inspect:

```
HostApple:~ nunocancelo$ docker inspect mas-
terzdran/pap-httpd
```

O output é longo, e por questões de legibilidade não é colocado, mas pode visualizar um vasto conjunto de configurações da image.

Volumes

Como já mencionado, as imagens são readonly e isso quer dizer que quando o container é desligado toda a informação “nova” é perdida. Este comportamento não é o desejado. No nosso exemplo, podemos querer manter os logs do servidor/aplicação para futuras consultas e neste momento não nos é possível, ou mesmo queremos atualizar uma página do nosso site que tem um erro. Então como resolvemos esta situação?

Existem duas formas que podemos contornar este desafio: com Data Volumes ou com Data Containers, ambos funcionam de forma semelhante apesar de serem declarados de forma diferente.

Data Volumes

Imaginemos que queremos então atualizar a página do nosso exemplo. Uma forma é com Data Volumes, em que no arranque da imagem declaramos que existe uma diretoria no nosso sistema de ficheiros que é mapeada com a diretoria no sistema de ficheiros do container (nomeadamente a diretoria onde está o nosso “site”).

```
HostApple:pap-httpd nunocancelo$ docker run -dit
-p 80:80 --name pap-httpd -v [Local Path]:/usr/
local/apache2/htdocs/ masterzdran/pap-httpd
```

Este comando vai mapear a diretoria “Local Path” com a diretoria /usr/local/apache2/htdocs/, onde está o “site” no container. Contudo, ao realizarmos este mapeamento o container deixa de “olhar” para o conteúdo da diretoria e passa a olhar para o volume mapeado. E neste caso a diretoria do host mapeada não é a mesma onde está o site, pelo que obtenho a Figura 4:

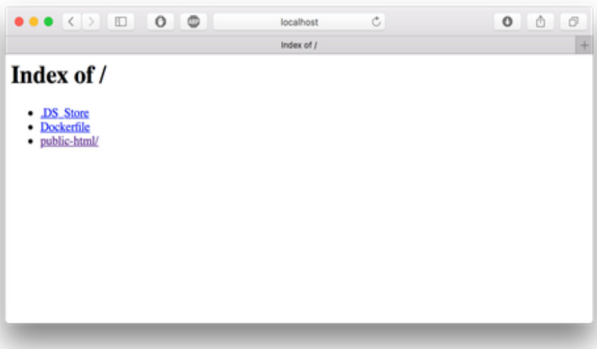


Figura 4: Data Volume Mapping

E navegando para a diretoria "public-html" temos então o nosso site Figura 5.

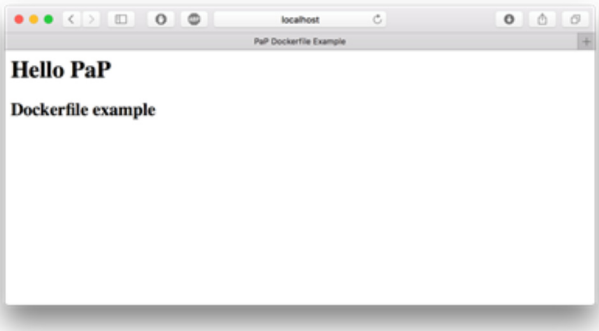


Figura 5: Site num Data Volume

Agora está na altura de modificarmos o nosso site "localmente" e o mesmo ser refletido automaticamente no nosso container.

Após algumas alterações, o nosso site de exemplo ficou assim como na Figura 6:

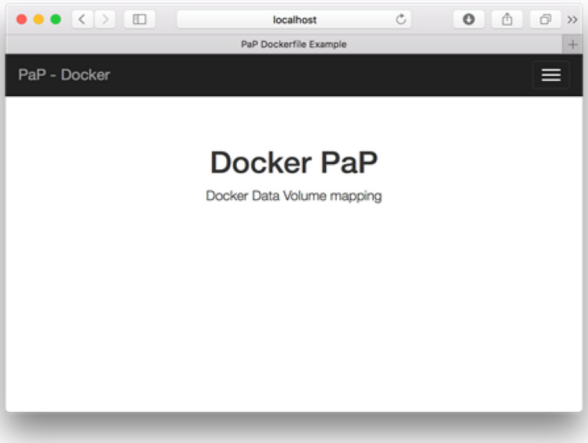


Figura 6: Alterações ao Site

Com pouco esforço o site ficou publicado, bastando fazer refrescar a página para refletir as alterações.

Neste tipo de mapeamento torna-se simples fazer backup do site, bastando criar uma copia de segurança da diretoria mapeada.

Não estamos limitados somente a diretorias, podemos mapear também ficheiros entre host e container, porém pessoalmente ainda não encontrei uma vantagem em fazer isso.

Por vezes existe a necessidade de copiar ficheiros dentro dos containers, como os ficheiros de configurações ou plug-ins que o container tem. Para tal executamos um novo comando:

```
HostApple:pap-httpd nunocancelo$docker cp
<containerId>:/file/path/within/container /host/
path/target
```

Este comando é particularmente útil quando andamos a testar provas de conceito e novas Images e queremos guardar as configurações para depois as usarmos noutros containers.

Data Containers

Assim como havia sido mencionado, existe outra forma de ter os dados guardados – os Data Containers. O paradigma é um pouco diferente dos Data Volumes uma vez que são containers na mesma, mas não têm qualquer imagem associada. Pensem em informação utilizada ou gerada pela nossa aplicação que precisa de ser guardada e posteriormente disponibilizada. Estou a referir-me a logs da aplicação, da infraestrutura, das bases de dados, etc.

É um container em que a única coisa que faz é guardar dados, posteriormente podemos aceder e não está associado a nenhuma imagem ou container, significando que mesmo que os containers "morram" ou sejam removidos a sua informação continua viva no Data Container que lhe foi associado.

Podemos ter N containers para M propósitos e para cada image container que criemos podemos ter associados os Data Containers que necessitarmos.

Para usarmos esta funcionalidade é necessário tomar dois passos:

- Criar o(s) Data Container(s)
- Associar o(s) Data Container(s)

Vamos então criar um volume:

```
HostApple:pap-httpd nunocancelo$ docker create -
v /usr/local/apache2/logs --name pap-httpd-logs
masterzdran/pap-httpd /bin/true
```

Muitos dos parâmetros já nos são familiares e o comando mostra que queremos criar um volume (create –v) com aquela localização (/usr/local/apache2/logs), vamos dar um nome para ser mais legível (pap-httpd-logs). Estamos a usar a imagem para os containers homogêneos e por fim o /bin/true para informar que tudo correu bem (útil se estivemos em modo batch).

Agora, quando associarmos o Data Container a um Container, se este tiver a diretoria mencionada passará a escrever no nosso novo container.

Como o Data Container continua a ser um container, podemos ver o seu estado:

```
HostApple:pap-httpd nunocancelo$ docker ps -a
CONTAINER ID        STATUS                    NAMES
9f48f0ac4f9f              Created
pap-httpd-logs
77deb89d4063          Up About an hour pap-httpd
```

E inspecionar o seu conteúdo:

```
HostApple:pap-httpd nunocancelo$ docker inspect
pap-httpd-logs
```

Agora vamos ao segundo ponto, associar o Data Container ao nosso container e para tal executamos o seguinte comando:

```
HostApple:pap-httpd nunocancelo$ docker run -dit -p 80:80 --name pap-httpd --volumes-from pap-httpd-logs -v [Local Path]:/usr/local/apache2/htdocs/ masterzdran/pap-httpd
```

Como podemos verificar o parâmetro *--volumes-from pap-httpd-logs* associa o nosso *Data Container* ao container que acabamos de criar.

Outra utilização para o mesmo parâmetro é para criar backup ou restore.

Para fazer um backup:

```
HostApple:pap-httpd nunocancelo$ docker run --rm --volumes-from pap-httpd-logs -v $(pwd):/backup ubuntu tar cvf /backup/backup.tar /usr/local/apache2/logs
tar: Removing leading `/' from member names
/usr/local/apache2/logs/
/usr/local/apache2/logs/httpd.pid
```

Neste caso vamos usar uma imagem Ubuntu para executarmos o comando tar para criar um arquivo com o conteúdo da diretoria.

Para depois fazer restore temos que criar um novo Data Container e depois executar um comando parecido para extrair os ficheiros:

```
HostApple:pap-httpd nunocancelo$ docker run -v /usr/local/apache2/logs --name pap-httpd-logs2 ubuntu /bin/bash
HostApple:pap-httpd nunocancelo$ docker run --rm --volumes-from pap-httpd-logs2 -v $(pwd):/backup ubuntu bash -c "cd /usr/local/apache2/logs && tar xvf /backup/backup.tar --strip 1"
usr/local/apache2/logs/
usr/local/apache2/logs/httpd.pid
```

Este acaba por ser o procedimento comum para os processos de migração, em que temos que fazer backup e restore de Data Containers.

Networking

Até agora temos visto um único container, onde reside o nosso servidor web para disponibilizar conteúdo estático.

Os containers são autocontidos e não conhecem outros containers. Podemos ter N containers a correr, tantos os que a nossa virtual machine nos permita ter, e apesar de todos eles poderem ser acedidos pela nossa rede, eles não se conhecem uns aos outros e por isso não conseguem falar entre si. Se tivermos um servidor web que precisa de aceder a uma base de dados que está noutro container, como é que fazemos?

Para ultrapassar este desafio temos que configurar a network. Admito que configuração de redes é algo que me ultrapassa um pouco, sei o essencial para conseguir aceder à Internet. J

No entanto em docker acaba por ser simples. A página de referência do docker (http://bit.ly/docker-network) explica muito bem este tema, assim como a sua configuração.

Para listar as redes disponíveis, executamos o seguinte comando:

```
HostApple:pap-httpd nunocancelo$ docker network ls
```

Como já devem ter reparado o padrão de execução dos comandos é bastante semelhante aos demais comandos já executados. Podemos criar da mesma forma e atribuir um nome em cada uma das novas redes já criadas.

Por omissão todos os containers ficam associados à network Bridge, porém podemos indicar uma outra rede:

```
HostApple:pap-httpd nunocancelo$ docker run --network=<NETWORK>
```

Desta forma permite-nos ter redes (com nomes específicos) para conjunto de aplicações.

Agora, temos os containers dentro da mesma rede e queremos que eles falem uns com os outros, poderia ser um "bicho de sete cabeças", mas a equipa do Docker simplificou a ação.

```
HostApple:pap-httpd nunocancelo$ docker run -dti --link CONTAINER:ALIAS --name LINKED <IMAGE>
```

O parâmetro --link CONTAINER:ALIAS vai dizer que o container que vai ser criado vai está "ligado" ao container CONTAINER:ALIAS. E a partir desse momento ambos os containers falam uns com os outros

> ***"Docker é um projeto open-source que permite criar um contentor (container) com a infraestrutura que se pretender por forma a poder partilha-la em qualquer máquina e manter sempre o mesmo comportamento. (...)"***

Docker Machine

Há pouco mencionei que podemos ter tantos os containers que quisermos como o que a máquina virtual permitir. Por omissão, quando instalamos o Docker é instalado uma Docker Machine com os recursos recomendados (na minha instalação tem dois cores e dois gigabytes de memória RAM), porém por vezes não é suficiente para a infraestrutura que queremos montar.

Apesar de podermos alterar estes parâmetros via linha de comandos, eu pessoalmente acabo por usar a interface gráfica fornecida. Para aceder a ela, vamos às propriedades do icon do docker (uma baleia), que no OSX está na barra superior, no Windows na barra de status e aparece a janela da :

Figura 7: Propriedades do Docker

No Tab "Advanced" podemos alterar as configurações da máquina virtual (após a alteração é necessário fazer restart ao serviço, que é uma opção também no icon do docker). Esta interface permite também outras configurações como diretorias partilhas ou definições de proxy. Por omissão não é permitido criar Data Volumes e Containers fora das diretorias partilhadas.

Dependendo do uso que se quer dar aos containers, recomendo que não se exagere no numero de CPUs e memória RAM atribuída, pois isso vai se refletir diretamente na performance do computador no dia a dia. Quando estiverem a testar imagens, novas aplicações, servidores, investiguem os requisitos de sistema e façam umas contas de "merceeiro", pois existe um trade-off entre a performance dos containers e a performance do computador.

Conclusão

Este breve overview sobre o ambiente docker permite sensibilizar sobre as potencialidades deste sistema e de que forma nos pode facilitar, agilizar o nosso trabalho.

É uma excelente forma de ter os sistemas contidos e por componentes tendo um potencial enorme, e porque tudo reside em containers, em muito pouco tempo podemos "migrar" para a Cloud (um serviço de Containers) e ter os nossos sistemas a funcionar.

Repositório:

https://gitlab.com/masterzdran/docker-overview

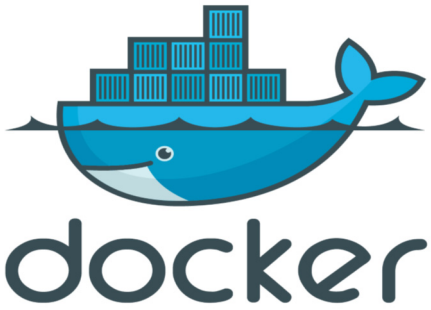
Referencias

Para além das já referidas no texto:

https://docs.docker.com

https://hub.docker.com

http://odewahn.github.io/docker-jumpstart/building-images-with-dockerfiles.html



AUTOR

Escrito por Nuno Cancelo

Curioso por natureza e engenheiro informático por formação. Desde muito cedo me despertou o interesse pelos computadores e informática com o famoso Spectrum. Tenho um gosto peculiar por aprender novas coisas frequentemente mesmo que não venha a trabalhar com elas e optimizar os sistemas aumentando a sua performance.

A PROGRAMAR

API Rest com Spring Boot (parte 2)

Um bot para Telegram com o Jogo da velha (Jogo do Galo).

Automação do Azure com Windows PowerShell Desired State Configuration

O Problema do casamento Estável utilizando o Algoritmo Gale-Shapley

API Rest com Spring Boot (parte 2)

Nesta segunda parte, vamos então adicionar ao nosso projecto um sistema que nos permita criar logs personalizados sobre o acesso à nossa API.

Sempre que desenvolvemos uma aplicação, devemos logo de inicio tratar de providenciar um bom sistema de logs já que ele é uma parte fundamental, seja durante o desenvolvimento, seja durante a operação da aplicação. É através das mensagens de log (em ficheiro ou no ecrã) que podemos determinar o que realmente está a acontecer na nossa aplicação e mais rapidamente determinar a origem de qualquer problema.

Para o programador, muitas vezes a análise do log fornece respostas que de outra forma seriam muito difíceis de obter.

No caso de aplicações cujo acesso remoto é permitido, torna-se também importante poder determinar quais os recursos acedidos, quando e por quem.

No nosso pequeníssimo projecto vamos utilizar o log4J2, uma biblioteca extremamente madura que está debaixo da alçada da Apache Software Foundation. Existem outras bibliotecas para o efeito, mas o log4J2 é extremamente flexível e apresenta uma performance elevada mesmo em sistemas de uso intensivo.

Não faz parte do âmbito deste artigo aprofundar as capacidades desta biblioteca, que são imensas, e a quem estiver interessado em aprofundar os seus conhecimentos aconselha-se a leitura do manual que pode ser encontrado em https://logging.apache.org/log4j/2.x/. A literatura pode ser intimidante, e dificilmente uma única leitura da documentação será suficiente. A utilização e compreensão do log4J exige algum esforço e dedicação, mas o aumento de produtividade que iremos obter sempre que necessitarmos de fazer algum tipo de debug nas nossas aplicações compensa o tempo dedicado ao estudo da biblioteca.

Neste artigo iremos apenas abordar como podemos incluir o log4J2 no nosso projecto, como criar os ficheiros de configuração necessários ao seu funcionamento e como gerar um ficheiro de log muito simples sobre o acesso à nossa API.

Uma vez que o nosso projecto é baseado em Maven, é por aí que vamos começar, a fim de adicionarmos ao projecto todas as dependências necessárias.

O framework Spring, suporta, de raiz, várias bibliotecas para serviços de logging e na versão actual está disponível também um starter para o log4J2 o que nos simplifica bastante a vida (afinal a simplificação é o grande objectivo do springbot), no entanto por defeito é adicionado o logback, e não é esta a biblioteca que pretendemos utilizar aqui.

Assim sendo, o nosso primero passo será remover o logback do nosso projecto e para isso vamos recorrer á possibilidade que o Maven nos oferece de excluir determinadas dependências.

Vamos então ao ficheiro POM.XML do projecto e localizamos as linhas referentes à dependência do Spring-boot-starter, pois é através deste starter que o logback é adicionado no nosso projecto.

```
<dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter</artifactId>
</dependency>
```

Uma vez que pretendemos excluir uma biblioteca que por defeito este starter adiciona ao projecto, vamos adicionar uma exclusão ficando então este grupo com a seguinte redação:

```
<dependency>
      <groupId>org.springframework.boot
                                        </groupId>
      <artifactId>spring-boot-starter
                                      </artifactId>
      <exclusions>
                  <exclusion>

      <groupId>org.springframework.boot
                                        </groupId>
                        <artifactId>spring-boot-
                     starter-logging</artifactId>
                  </exclusion>
      </exclusions>
</dependency>
```

De seguida, e sempre dentro da zona das dependências do ficheiro pom.xml adicionamos a dependência do starter que o *Spring Boot* disponibiliza para adicionarmos o *log4j2* ao nosso projecto. Para isso basta adicionar as seguintes linhas.

```
<dependency>
       <groupId>org.springframework.boot
                                        </groupId>
       <artifactId>spring-boot-starter-log4j2
                                      </artifactId>
</dependency>
```

Em resumo, o que fizemos foi informar que pretendíamos excluir a biblioteca standard de log que o Spring boot disponibiliza e incluir uma outra.

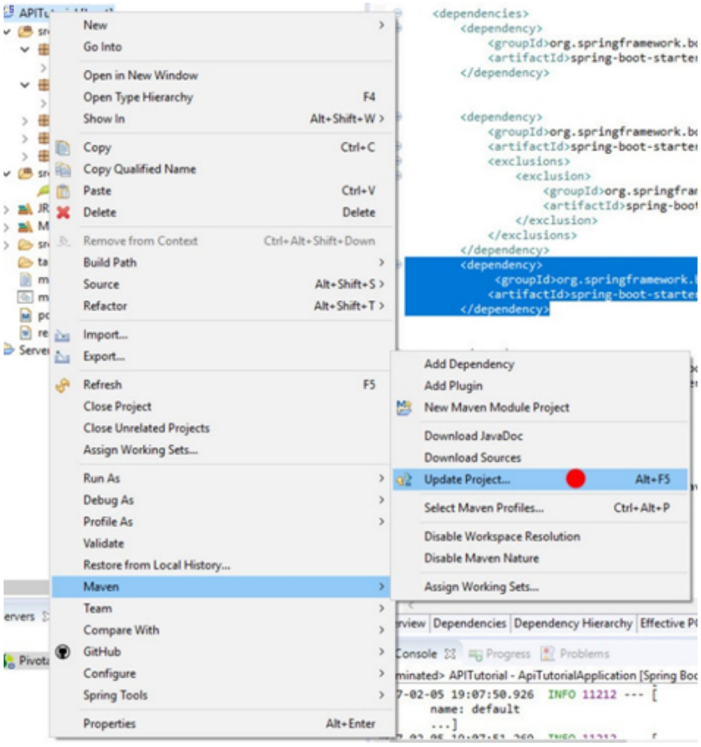
Após adicionarmos a nova configuração pretendida ao ficheiro pom.xml devemos grava-lo.

Como somos adeptos das boas práticas, o passo seguinte é imediatamente ordenar ao Spring Tool Suite que

atualize as nossas dependências.

Podemos aceder ao menu do Maven, ou simplesmente premir ALT+F5.

Após alguns instantes a biblioteca pretendida será adicionada ao projecto.



(pode verificar expandindo o grupo *maven dependencies* do projecto, onde deverá agora encontrar as bibliotecas do *log4J2*)

Uma vez este processo concluído podemos então começar a criar a configuração necessária para que o log4J2 funcione como pretendemos.

Existem pelo menos 3 formas de configurar o *log4j* no *Spring Boot*. Podemos criar as configurações no ficheiro *application.properties*, podemos criar um ficheiro de configuração específico, sendo que este último nos permite um grau de configuração bastante superior que o primeiro método, e ainda podemos criar a configuração programaticamente através de classes.

No nosso caso iremos utilizar um ficheiro de configuração. Um facto importante é que este ficheiro de configuração tem de residir no local correto do projecto. Se usarmos as configurações padrões do framework, esse local será a pasta src/main/resources.

Vamos então criar um novo ficheiro na localização referida, e vamos dar a esse ficheiro o nome de log4j2.xml. (nas configurações padrão este nome é obrigatório, e a utilização de outro nome irá gerar um erro, em que o STS irá indicar que não encontra o ficheiro de configuração). Em alternativa pode-se esquematizar o ficheiro de configuração usando Json ou Yaml, mas a ideia é semelhante.

Este ficheiro, no nosso caso, seguirá uma semântica XML, irá conter a seguinte informação

```
<?xml version="1.0" encoding="UTF-8"?>
<Configuration>
    <Properties>
        <Property name="log-path">logs</Property>
    </Properties>
    <Appenders>
        <Console name="Console-Appender"
                                target="SYSTEM_OUT">
            <PatternLayout>
                <pattern>
                    [%-5level] %d{yyyy-MM-dd
                HH:mm:ss.SSS} [%t] %c{1} - %msg%n
                </pattern>>
            </PatternLayout>
        </Console>
        <File name="Api-File-Appender"
             fileName="${log-path}/api_log.log" >
            <PatternLayout>
                <pattern>
                    %-5level %d [%t] %c:%M(%L): %
                                                m%n
                </pattern>
            </PatternLayout>
        </File>
        <File name="SpringBoot-File-Appender"
   fileName="${log-path}/spring_system_log.log" >
            <PatternLayout>
                <pattern>
                    %-5level %d [%t] %c:%M(%L): %
                                                m%n
                </pattern>
            </PatternLayout>
        </File>
    </Appenders>
    <Loggers>
        <Logger name="org.springframework.web"
                 level="info" additivity="false">
            <AppenderRef ref="SpringBoot-File-
                                        Appender"/>
            <AppenderRef ref="Console-Appender"/>
        </Logger>
        <Logger name="pt.api.controllers"
                 level="info" additivity="false">
            <AppenderRef ref="Api-File-Appender"/
>
            <AppenderRef ref="Console-Appender"/>
         </Logger>
        <Root>
            <AppenderRef ref="Console-Appender"/>
        </Root>
    </Loggers>
</Configuration>
```

Vamos agora analisar algumas particularidades deste ficheiro de configuração. Podemos encontrar algumas secções que destacamos, as ***Properties***, os ***Appenders*** e os ***Loggers***.

Na secção *properties* podemos criar pares do tipo chave/valor que ficam disponíveis para utilização em outras secções do ficheiro. Neste caso apenas utilizamos uma chave a que demos o nome *log-path* e à qual atribuímos o valor *logs (queremos que os nossos logs sejam criados numa pasta com o nome logs)*. Isto irá permitir que em todas as outras secções do ficheiro podemos usar o nome log-path e ele irá ser substituído pelo valor *logs*.

A secção *Appenders* permite-nos definir as possíveis formas de saída para os *loggers*. Criamos *appenders* para a consola (*console*) e para ficheiro (*file*). No caso do *logger*

para a consola, definimos o target como *system_out*, o que dirigirá a mensagem log para o stream de saída que por defeito é a consola. Nos *loggers* do tipo file, as mensagens serão redirecionadas para os ficheiros respetivos de acordo com o que definimos na propriedade *filename*.

Em cada um dos *appenders*, informamos também um padrão de layout que irá definir o formato da mensagem de log. O mais comum é usarmos logs de uma linha num formato legível, no entanto o log4J permite outro tipo de layouts como Json, Xml entre outros. Aqui vamos utilizar o formato mais simples. A definição dos padrões de layout (patternLayout) obedece a regras próprias que pode consultar em *http://logging.apache.org/log4j/2.0/manual/layouts.html#Patterns.* Nestes layouts podemos configurar o formato que queremos para as datas, espaçamentos, nomes ou ID das threads, nomes dos loggers e uma verdadeira miríade de opções que cobrem os casos mais exigentes que porventura possam surgir.

Para finalizar temos a secção dos *Loggers*. O nome que atribuímos aos *loggers* através da propriedade *name* é importante pois esse é o nome que vamos utilizar no código java sempre que necessitamos de evocar um *logger*. Por norma utilizamos nomes na forma *package* ou *package/class* seguindo a hierarquia tradicional do Java.

Ainda dentro dos *loggers* encontramos um que é especial. O *root* . Este irá receber todas as mensagens que o sistema envie e que não estejam enquadradas em nenhum outro *logger*. No caso de nos esquecermos de o criar ele será criado automaticamente pelo *log4J2*, mas isso poderá não ser do nosso interesse, pois assim podemos definir para onde devem ser canalizadas todas as mensagens que não pertençam aos outros *loggers*.

Agora que temos o ficheiro de configuração criado, poderemos começar a utilizar os *loggers* no nosso projecto.

Vamos abrir a classe *ClienteController* que tínhamos criado na primeira parte do artigo e vamos adicionar uma linha imediatamente a seguir à definição da classe, ficando agora esta classe com a redação

```
@RestController
public class ClienteController {
private final Logger logger=LoggerFactory.getLogger
                                (this.getClass());

@Autowired
private ClienteRepository clienteRepository;
```

O que estamos a fazer aqui é solicitar à LoggerFactory que nos disponibilize um logger baseado na nossa classe (que pertence ao package *pt.api.controllers*) e se verificarem na configuração temos um logger definido para esta situação que irá fazer log na consola e log para o ficheiro com o nome api_log.

Agora precisamos de informar o que queremos na nossa linha de log, e vamos então adicionar uma linha aos nossos pontos de entrada, que nos irá permitir registar a informação que pretendendo.

```
logger.info("Classe: "+this.getClass().getName()
+ " ,Método :"+Thread.currentThread
().getStackTrace()[1].getMethodName()
+ " ,URL :" + request.getRequestURI() +
" ,parametros : " + request.getQueryString() +
",IP Origem :" + request.getRemoteAddr() );
```

Caso ainda se recordem da primeira parte deste artigo, foi feita uma pequena menção sobre termos adicionado o HttpServletRequest request como parâmetro aos nossos métodos e de como isso nos poderia ser útil mais tarde. Pois aqui está uma das razões. Através do *request* conseguimos aceder à informação sobre o pedido que foi efetuado pelo cliente chamador.

O que esta linha faz é criar um entrada no log com o nome da classe, nome do método, o URL do pedido, os parâmetros (caso existam) e o endereço IP de origem do pedido. No caso em que não existam parâmetros, o valor registado será *null*

Após adicionarmos esta linha a todos os pontos de entrada da nossa API, a classe *ClienteController* deverá apresentar a seguinte redação:

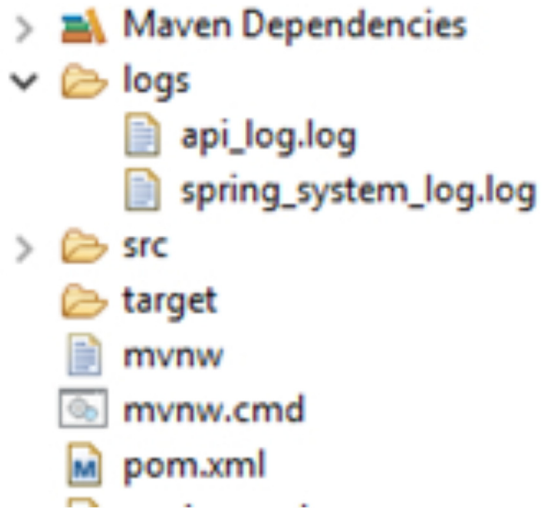
```
package pt.api.controllers;

import java.util.Collection;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import
org.springframework.beans.factory.annotation.Auto
                                              wired;
import org.springframework.http.MediaType;
import org.springframework.http.ResponseEntity;
import org.springframework.http.HttpStatus;
import org.springframework.web.bind.annotation.
                                        PathVariable;
import org.springframework.web.bind.annotation.
                                         RequestBody;
import org.springframework.web.bind.annotation.
                                      RequestMapping;
import org.springframework.web.bind.annotation.
                                        RequestMethod;
import org.springframework.web.bind.annotation.
                                         RequestParam;
import org.springframework.web.bind.annotation.
                                      RestController;

import pt.api.entities.ClienteEntity;
import pt.api.repositories.ClienteRepository;

@RestController
public class ClienteController {
       private final Logger logger =
        LoggerFactory.getLogger(this.getClass());

      @Autowired
      private ClienteRepository clienteRepository;

      // EndPoint #1
        @RequestMapping(value = "/api/v1/cliente/
        {id}", method = RequestMethod.GET,
      produces = MediaType.APPLICATION_JSON_VALUE)

        public ClienteEntity getByClienteId
        (HttpServletRequest request,
```

API REST COM SPRING BOOT (PARTE 2)

```
        HttpServletResponse response,@PathVariable
      (value = "id") String id) throws Exception {
    // verificação do ID
    logger.info("Classe: " + this.getClass().
                          getName() + " ,Método :"
     + Thread.currentThread().getStackTrace()
              [1].getMethodName() + " ,URL :" +
     request.getRequestURI()+ " ,parametros : " +
     request.getQueryString() + ",IP Origem :"
                    + request.getRemoteAddr());
    int idCliente = 0;
    try {
          idCliente = Integer.parseInt(id);
    } catch (Exception ex) {
          response.sendError
                (HttpStatus.BAD_REQUEST.value());
          return null;
    }
    // Fetch de um cliente por ID
    ClienteEntity cliente =
            clienteRepository.findOne(idCliente);
    if (cliente == null) {
          response.sendError
                  (HttpStatus.NOT_FOUND.value());
          return null;
    } else {
          return cliente;
          }
    }

    // EndPoint #2
     @RequestMapping(value = "/api/v1/cliente/
     all", method = RequestMethod.GET, produces =
     MediaType.APPLICATION_JSON_VALUE)

     public ResponseEnti-
     ty<Collection<ClienteEntity>> getAllClientes
     (HttpServletRequest request,     HttpS-
     ervletResponse response) {
    logger.info("Classe: " + this.getClass().
                          getName() + " ,Método :"
     + Thread.currentThread().getStackTrace()
     [1].getMethodName() + " ,URL :" + re-
     quest.getRequestURI()
     + " ,parametros : " + request.getQueryString
     () + ",IP Origem :" + request.getRemoteAddr
     ());

    Collection<ClienteEntity> clientes =
                      clienteRepository.findAll();
    return new ResponseEntity<Collection
       <ClienteEntity>>(clientes, HttpStatus.OK);
    }

    // EndPoint #3
     @RequestMapping(value = "/api/v1/cliente",
     method = RequestMethod.GET, produces = Medi-
     aType.APPLICATION_JSON_VALUE)

    public ClienteEntity getByClienteNIF
  (HttpServletRequest request, HttpServletResponse
 response,@RequestParam(value = "nif") String nif)
throws Exception {
    logger.info("Classe: " + this.getClass().
                          getName() + " ,Método :"
     + Thread.currentThread().getStackTrace()
     [1].getMethodName() + " ,URL :" + re-
     quest.getRequestURI() + " ,parametros : " +
     request.getQueryString() + ",IP Origem :" +
     request.getRemoteAddr());

    if (!nif.matches("[0-9]+") && nif.length()
                                          != 9) {
          response.sendError
                (HttpStatus.BAD_REQUEST.value());
          return null;
    }

    ClienteEntity cliente =
         clienteRepository.pesquisaPorNIF(nif);
    return cliente;
    }
    // EndPoint #4

     @RequestMapping(value = "/api/v1/cliente/
     save", method = RequestMethod.POST, con-
     sumes = MediaType.APPLICATION_JSON_VALUE,
     produces = Medi-
     aType.APPLICATION_JSON_VALUE)

     public int saveCliente(HttpServletRequest
     request, HttpServletResponse response,
     @RequestBody ClienteEntity cliente) throws
     Exception {

     logger.info("Classe: " + this.getClass().
                          getName() + " ,Método :"
     + Thread.currentThread().getStackTrace()
     [1].getMethodName() + " ,URL :" + re-
     quest.getRequestURI() + " ,parametros : "
     + request.getQueryString() + ",IP Ori-
     gem :" + request.getRemoteAddr());

     if (cliente.getId() != null) {
          response.sendError
       (HttpStatus.METHOD_NOT_ALLOWED.value());
    }

    try {
          clienteRepository.save(cliente);
    } catch (Exception ex) {
          response.sendError
              (HttpStatus.BAD_REQUEST.value());
    }

    return HttpStatus.CREATED.value();
    }

    // EndPoint #5

     @RequestMapping(value = "/api/v1/cliente/
     update", method = RequestMethod.PUT, con-
     sumes = MediaType.APPLICATION_JSON_VALUE,
     produces = Medi-
     aType.APPLICATION_JSON_VALUE)

    public int updateCliente(HttpServletRequest
         request, HttpServletResponse response,
    @RequestBody ClienteEntity cliente) throws
                                      Exception {

    logger.info("Classe: " + this.getClass().
                          getName() + " ,Método :"
     + Thread.currentThread().getStackTrace()
     [1].getMethodName() + " ,URL :" + re-
     quest.getRequestURI()
     + " ,parametros : " + re-
     quest.getQueryString() + ",IP Origem :" +
     request.getRemoteAddr());
    // verificar se não é nulo e se existe
    if (cliente.getId() == null ||
     clienteRepository.findOne(cliente.getId())
                                      == null) {
          response.sendError
(HttpStatus.NOT_FOUND.value(), "Erro de ID");
          return 0;
    }

     try {
          clienteRepository.save(cliente);
     } catch (Exception ex) {
          response.sendError
```

```
    (HttpStatus.BAD_REQUEST.value(), "Erro de BD");
            return 0;
      }

      return HttpStatus.ACCEPTED.value();

      }

      // EndPoint #6
       @RequestMapping(value = "/api/v1/cliente/
       delete/{id}", method = RequestMethod.DELETE,
       produces = MediaType.APPLICATION_JSON_VALUE)

      public int deleteCliente(HttpServletRequest
             request, HttpServletResponse response,
      @PathVariable(value = "id") String clienteID)
                                  throws Exception {

      logger.info("Classe: " + this.getClass().
                             getName() + " ,Método :"
       + Thread.currentThread().getStackTrace()
       [1].getMethodName() + " ,URL :" +
                              request.getRequestURI()
       + " ,parametros : " + request.getQueryString
       () + ",IP Origem :" + request.getRemoteAddr
       ());
      // verificação do ID
      int idCliente = 0;
      try {
            idCliente = Integer.parseInt(clienteID);
      } catch (Exception ex) {
            idCliente = 0;
            response.sendError
                  (HttpStatus.BAD_REQUEST.value());
            return 0;
      }

      try {
            clienteRepository.delete(idCliente);
      } catch (Exception ex) {
            response.sendError
    (HttpStatus.BAD_REQUEST.value(), "Erro de BD");
            return 0;
      }
      return HttpStatus.OK.value();
      }
}
```

Testando



Vamos agora correr e testar a nossa API, seguindo os mesmos passos que efetuamos na primeira parte deste artigo, e de seguida verifique que devemos ter uma nova pasta adicionada ao nosso projecto com 2 novos ficheiros.

Vamos abrir o ficheiro api_log.log e nele deveremos encontrar o log de todos os acessos que fizemos à API.

No ficheiro Spring_system_log.log devermos encontrar o log referente ao framework Spring.

Não se assuste se o IP aparecer como 0:0:0:0:0:0:0:1. Caso esteja a desenvolver e testar no mesmo computador, esse é o valor que o Spring vai assumir para *localhost*.

Se tiver estado atento à consola, enquanto efetuou os testes da API, poderá verificar que as mensagens para além de terem ficado armazenadas no ficheiro, aparecem também na consola , pois definimos no ficheiro de configuração que este *logger* ligava com o apender *Api-File-Appender* e com o *Console-Appender*.

Como passar a nossa API para produção ?

Até agora temos estado a trabalhar sempre dentro do Spring Tool Suite (STS) que já inclui um servidor TomCat e nos permite desenvolver e testar rapidamente. No entanto, o objetivo de qualquer aplicação é correr fora do ambiente de desenvolvimento.

Neste exemplo, o que pretendemos fazer é criar uma aplicação que possa ser colocada rapidamente em qualquer computador (que tenha JAVA instalado) e a mesma possa correr imediatamente. Ou seja, não pretendemos instalar o TomCat, bibliotecas, configurar tudo, etc. O único pressuposto que vamos usar aqui, é de que estamos a instalar a nossa aplicação no mesmo servidor onde se encontra instalado o MYSQL e que a base de dados está funcional. (se o MYSQL estiver em outro servidor, bastará alterar a localização no ficheiro Application.properties que está definida em spring.datasource.url)

Como todos sabemos as aplicações java são empacotadas num formato próprio designado por *jar*. Neste caso o que queremos criar, é o que normalmente se chama de ***fat jar*** ou seja um único ficheiro que tem todas as dependências que o nosso projecto necessita para funcionar, e neste caso incluindo também o servidor.

Ou seja, a nossa API vai ter apenas um ficheiro, e assim que o executamos…Voilá. Temos o servidor no ar e a api a funcionar.

A criação de um *fat jar* é extremamente simples, graças ao *Maven* e apenas necessitamos de dar um comando de consola.

API REST COM SPRING BOOT (PARTE 2)

No gestor de ficheiros, navegue até á pasta onde está o seu projecto. A pasta deve ter um conteúdo semelhante ao da imagem, e abra uma linha de comandos nesse local.

Nesta pasta tudo o que tem a fazer é digitar o comando **mvn clean package** seguido de ENTER e aguardar que o Maven trate de baixar todas as dependências necessárias que ainda não estejam no seu computador e de as empacotar.

Quando o processo terminar deverá visualizar uma mensagem de BUILD SUCCESS. De seguida verifique a pasta *target* e nela deverá encontrar 2 ficheiros. Um deverá ter a extensão *.jar* e outro a extensão *.original*. O ficheiro que queremos é o que termina em *jar*. Se seguiu todos os passos como descrevemos desde o início, esse ficheiro deverá ter o nome *apitutorial-1.0.0.jar,* caso contrário terá o nome que lhe tiver atribuido.

> ***Sempre que desenvolvemos uma aplicação, devemos logo de inicio tratar de providenciar um bom sistema de logs já que ele é uma parte fundamental, seja durante o desenvolvimento, seja durante a operação da aplicação.***

Podemos executa-lo imediatamente através do comando ***java -jar apitutorial-1.0.0.jar***. (certifique-se que o servidor do Spring tool suite está parado).

Se tudo tiver corrido bem, deverá visualizar o arranque do Spring e passado alguns segundos a sua API está em funcionamento e acessível a partir de qualquer computador da sua rede, a pasta log será criada e todos os logs serão registados.

No caso de estar a usar um servidor LINUX, o comando para iniciar é o mesmo, só deverá ter em atenção as permissões das pastas.

O código presente neste artigo está disponível em https://gitlab.com/Java-exemplos/apitutorial-parte2.git

Espero que este artigo seja útil para quem se quiser iniciar no framework Spring, utilizando o Spring Boot.

AUTOR

Escrito por José Martins

Natural do Porto, autodidata. Iniciou-se no mundo das tecnologias com um Sinclair ZX-81 onde aprendeu a programar em basic e assembler. Ao longo de 25 anos ligados às tecnologias, passou por quase todas as linguagens de programação, até que decidiu "assentar praça" com JAVA. Atualmente trabalha na PJ Silva Lda, onde desenvolve projetos ligados à monitorização do ensino prático da condução automóvel. (josetabordamartins@gmail.com).

JavaFX: Passos Seguintes

No artigo anterior dei uma breve introdução sobre como programar com JavaFX, conceitos simples que permite começar a desenvolver aplicações gráficas. Neste artigo vou explorar outros temas interessantes do ponto de vista de desenvolvimento e de manutenção de projetos de software com interfaces gráficas de desktop.

Ao longo da minha carreira já desenvolvi e participei em projetos de software de raiz, mas grande parte dela foi a manter e a melhorar aplicações legacy, e deixem-me que diga que existem programadores muito imaginativos J. Um dos temas que mais urticária me causa é o facto de o software desenvolvido não poder ser mantido com facilidade, e ao *ripple effects* das alterações simples que são realizadas.

Como mencionei no artigo anterior, devido à necessidade de alterar uma aplicação em Java Swing, essa mesma alteração não se mostrou nada fácil de realizar, levando-me a procurar alternativas e migrar para JavaFX. No artigo anterior abri a porta à tecnologia e o que foi apresentado não difere muito do Java Swing, pelo que pode ser confuso a razão para a alteração.

Quando andei a investigar, o que estava à procura era um modelo de desenvolvimento que me permitisse definir a interface gráfica em separado da lógica da aplicação, ou seja, um modelo *Model-View-Controller* à semelhança do que existe para aplicações web, ou mesmo para a plataforma *Android*. E JavaFX tem mesmo o que estava à procura e é sobre isso que venho falar.

Antes de começar

Antes de começar é recomendado instalar alguns componentes. Não porque é obrigatório ou necessário (mesmo) mas sim porque auxilia (e muito o desenvolvimento).

É necessário o JDK para ter o JavaFX:

- JDK 7 update 6: com JavaFX 2.2
- JDK 8: com JavaFX 8

Para quem usa um IDE baseado em Eclipse deve instalar este plugin:

- e(fx)clipse: http://bit.ly/e-fx-clipse

Este plugin incorpora um conjunto de funcionalidades ao IDE que facilitam o desenvolvimento e publicação dos projetos JavaFX.

Existe outra ferramenta da Oracle que deve ser instalada:

- Scene Builder: http://bit.ly/JavaFX_Scene_Builder

A ferramenta permite desenhar as interfaces gráficas com simples *drag-and-drop*, configurar eventos etc.

Defining the application layout

Em média, cada controlo JavaFX (checkbox, label, etc) tem quatro linhas de código:

- Instanciar um novo objeto.
- Definir o seu id.
- Definir o seu comprimento
- Definir a sua altura

E depois mais umas quantas linhas para afetar o layout (Pane), configurar a Scene e o Stage.

No entanto em cada controlo pode-se personalizar ainda mais. No caso do controlo textField pode-se definir a fonte, o seu tamanho, as linhas de contorno da caixa, a cor, etc.

Porém o procedimento é aborrecido e se pretender atribuir a mesma personalização para um conjunto de textFields (ex: 20 caixas de texto).

No fim do dia é possível ter umas boas centenas de linhas com código “repetido”, monótono e que se começa a perspetivar difícil de manter quando se começar a trabalhar com os *event handlers*. A solução não é coesa ou mesmo desacoplada e muito menos fácil de manter.

Styling the application

Com o JavaFX veio algo muito fixe, a separação entre a definição da interface gráfica e a sua estilização. E é conseguida através de um ficheiro CSS, sim um simples ficheiro de CSS, à semelhança do que é usado em páginas web.

Tal como em HTML, aqui existem as classes e os ids. As classes identificam o estilo a aplicar a um controlo que tenha aquela classe ou definir por omissão o estilo de um controlo. Para os ids o estilo é atribuído ao controlo que tenha aquele id e é único.

Hello World CSS

Neste breve exemplo, vai ser demonstrado como usar o CSS na aplicação.

```
@Override
public void start(Stage primaryStage)
throws Exception {
  // Stage Title
  primaryStage.setTitle("Hello World");

  // Begin Scene definition
  Label helloLabel1 = new Label();
  helloLabel1.setText("Hello World!");
  helloLabel1.setId("helloLabel1");

  Label helloLabel2 = new Label();
  helloLabel2.setText("Hello World!");
  helloLabel2.setId("helloLabel2");
```

```
    Label helloLabel3 = new Label();
    helloLabel3.setText("Hello World!");
    helloLabel3.setId("helloLabel3");

    VBox vboxPane = new VBox();
    vboxPane.getChildren().addAll(
           helloLabel1,
           helloLabel2,
           helloLabel3);
    vboxPane.setAlignment(Pos.CENTER);

    Scene scene = new Scene(vboxPane,200,200);
    // End Scene definition
    // Add Scene to Stage
    primaryStage.setScene(scene);
    // Show Stage
    primaryStage.show();
  }
```

O exemplo é muito semelhante ao que foi feito no artigo anterior e segue o mesmo padrão. A imagem da Figura 1 mostra o ponto de partida.

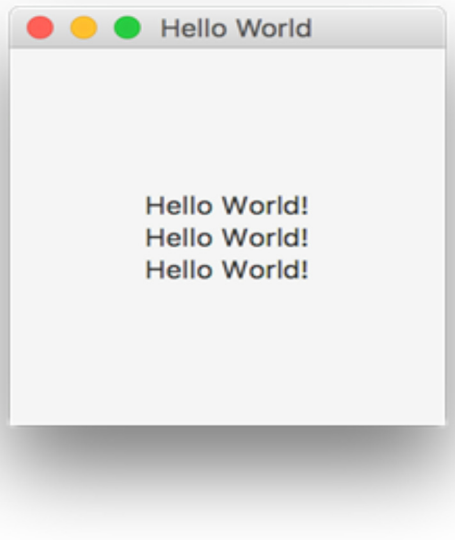


Figura 1: Hello World

Vou começar por criar um novo ficheiro CSS ao qual vou dar o nome "HelloWorld.css" e vou começar a personalizar componentes. Muito em breve vão perceber que não sou designer J.

Vou começar por personalizar o componente raiz do nosso *Stage* (root) que é o controlo base a partir do qual todos os controlos herdam as suas propriedades.

```
/* .root is the base element of the Scene */
.root{
 -fx-background-color: #d08770;
 -fx-font-size: 20px;
}
```

Tal como acontece em HTML, é necessário indicar que este ficheiro com estilos deve ser carregado:

```
    Scene scene = new Scene(vboxPane,200,200);
    scene.getStylesheets()
        .add(getClass()
        .getResource("HelloWorld.css")
        .toExternalForm());
```

A partir deste momento, todos os estilos definidos no CSS estão disponíveis na aplicação. O resultado pode ser visto na Figura 2.

Figura 2: .root styling

Como se vê o tamanho da letra aumentou e fundo mudou de côr. Lindo J.

Agora vou alterar o estilo de todos os controlos do tipo Label:

```
.label{
  -fx-border-color: #3633FF;
  -fx-border-width: 3px;
  -fx-padding: 10px;
}
```

e automaticamente o resultado é visível na Figura 3:

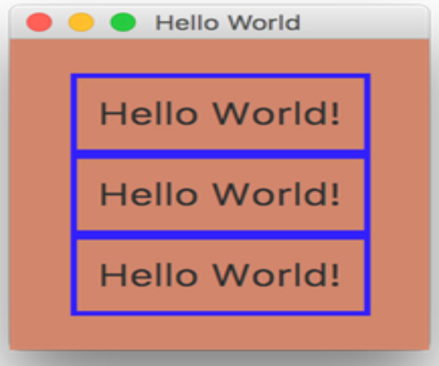


Figura 3: Label styling

Porém é comum que apenas um conjunto de controlos tenham um dado estilo, por isso criamos uma classe especifica para o efeito.

```
.label-mine{
  -fx-border-color: #FF3364;
  -fx-border-width: 5px;
  -fx-font-size: 9px;
}
```

E no controlo definimos qual a classe que ele usa

```
Label helloLabel1 = new Label();
helloLabel1.setText("Hello World!");
helloLabel1.setId("helloLabel1");
helloLabel1.getStyleClass()
   .add("label-mine");
```

E obtém-se o resultado da Figura 4:

Figura 4: Custom Label styling

Por fim, vou definir um estilo para um controlo em especifico, definindo o seu id.

```
#helloLabel3{
  -fx-border-color: #7B241C;
  -fx-border-width: 2px;
  -fx-font-size: 28px;
  -fx-background-color: #F5B7B1;
  -fx-text-fill: #873600;
}
```

E o seu resultado na Figura 5:

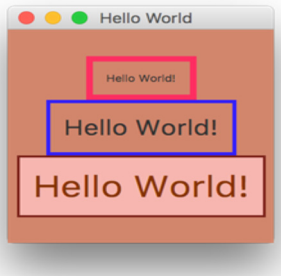


Figura 5: Control Id Styling

De uma forma muito simples foi possível separar a definição dos estilos e a sua personalização do código java em si. No entanto, esta separação não é suficiente.

Hello World - FXML

JavaFX trás outra particularidade fantástica, a possibilidade de descrever em XML toda a interface gráfica. O próximo exemplo descreve o mesmo exemplo mas em FXML.

Assumindo foi instalado o plugin do e(fx)clipse, criar um projeto é tão simples como File -> New -> JavaFX Project e preencher os formulários. No formulário da Figura 6, na combo box *"language"* escolher FXML. E será criada uma estrutura semelhante à Figura 7.

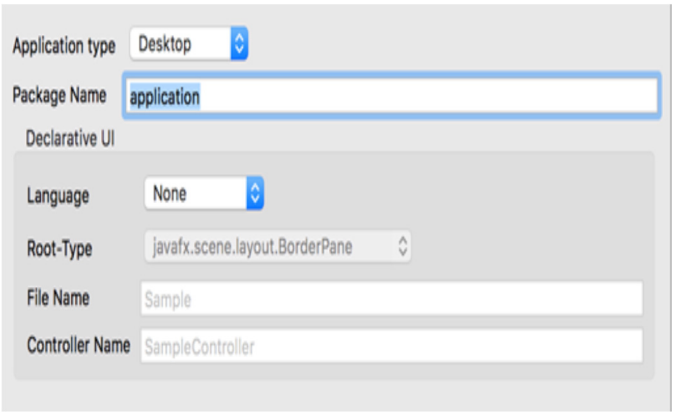


Figura 6: New JavaFX Project

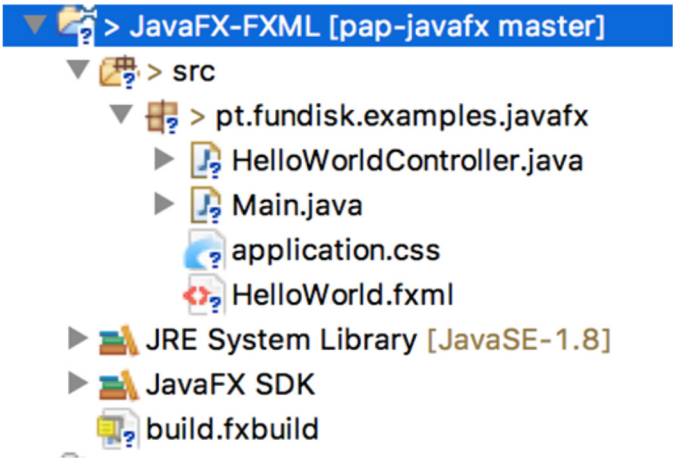


Figura 7: JavaFX Project Structure

O novo projeto contém quatro ficheiros:

- Main.java: é o ponto de entrada da aplicação.
- application.css: é o ficheiro de estilos.
- HelloWorld.fxml: é ficheiro com definição do UI.
- HelloWorldController: é o ficheiro onde estão definidos event handlers

O Main.java gerado é o seguinte:

```
public class Main extends Application {
  @Override
  public void start(Stage primaryStage) {
    try {
      VBox root = (VBox)FXMLLoader
                .load(getClass()
                .getResource("HelloWorld.fxml"));
      Scene scene = new Scene(root,200,200);  sce-
ne.getStylesheets()
              .add(getClass()
              .getResource("application.css")
              .toExternalForm());
                   primaryStage.setScene(scene);
      primaryStage.show();
    } catch(Exception e) {
      e.printStackTrace();
    }
  }
  public static void main(String[] args) {
    launch(args);
  }
}
```

Muito simples, carrega o xml com a definição da UI, o ficheiro de estilos e mostra a janela. Nada de muito relevante a dizer.

O ficheiro de estilos vou utilizado o mesmo conteúdo do exemplo anterior.

Já o ficheiro FXML requer mais atenção. O exemplo a seguir é representação em XML do que foi descrito em código em exemplos anteriores:

```
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.layout.VBox?>

<?import javafx.scene.control.Label?>

<VBox xmlns:fx=http://javafx.com/fxml/1
  fx:controller="….HelloWorldController"
  alignment="CENTER">

    <Label fx:id="helloLabel1"
           text="Hello World!"
           styleClass="label-mine"/>
    <Label fx:id="helloLabel2"
           text="Hello World!"/>
    <Label fx:id="helloLabel3"
           text="Hello World!"/>

</VBox>
```

Logo à partida é visível que a descrição da UI é mais clara e a sua definição mais legível.

Os import statements, à semelhança do Java, indicam o package e o controlo que é utilizado ao longo do ficheiro.

O VBox é o pane que utilizei e é de notar a instrução fx:controller (cujo o nome da classe foi alterado para ser legível). Cada ficheiro FXML tem um e um único controle e para identificar os controlos é utilizado o fx:id.

As restantes linhas apenas caracterizam cada um dos controlos utilizados.

O resultado é o espectável da Figura 8.

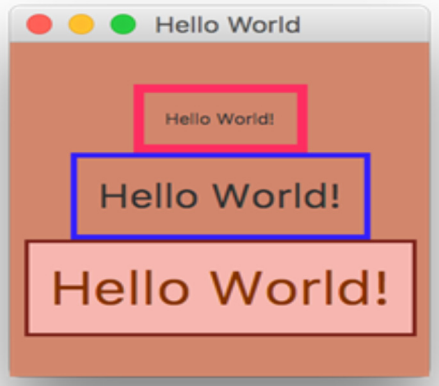


Figura 8: FXML Output

Controllers

Como mencionei, o ficheiro de Controller acaba por ser o ficheiro de gestão e tratamento de eventos.

Uma vez que um ficheiro FXML só tem um único Controller, os atributos podem ter visibilidade pública. Porém, na minha opinião, a visibilidade deve ser mais restrita e coloca-se a anotação @FXML. Para mim, torna o código mais legível.

Aproveito para deixar outra recomendação, o Controller deve implementar a interface Initializable que tem o método initialize. Este método é invocado após todos os nós do FXML terem sido instanciados e é útil no caso de se pretender fazer algo com os controlos antes de começar a aplicação.

O próximo exemplo vou colocar uma label e um botão (Figura 9) , e quando o botão for pressionado a label apresentará outro texto(Figura 10).

Esta é a secção relevante do FXML:

```
<VBox xmlns:fx="http://javafx.com/fxml/1"
  fx:controller="….HelloWorldController"
  alignment="CENTER">
  <Label
  fx:id="helloLabel1"
  text="Hello World!"
  styleClass="label-mine"/>
  <Button text="Click"
  fx:id="clickButton"
  onAction="#clickButtonEvent"/>
</VBox>
```

E a implementação do Controller:

```
public class HelloWorldController
  implements Initializable {
  @FXML private Label helloLabel1;
  @FXML private Button clickButton;
```

```
  @Override
  public void initialize(
            URL location,
            ResourceBundle resources)
  {
    // TODO Auto-generated method stub
  }

  @FXML
  public void clickButtonEvent() {
    Date date = new Date();
    helloLabel1.setText(date.toString());
  }
}
```

Figura 9: FXML with Controller

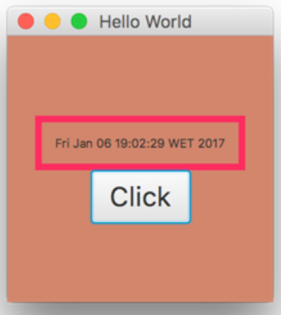


Figura 10: FXML with Controller clicked

Simples e eficaz J

Conclusão

A partir de agora é uma questão de tornar as aplicações mais funcionais e a satisfazer as necessidades.

JavaFX demonstrou-se ser uma agradável surpresa, com as suas componentes MVC que ajudam a simplificar e a manter as aplicações. A separação da aplicação em três camadas (estilos/css, FXML/UI, Controller/Event Handler) tornou-se um thumbs-up para mim, tornando o código da aplicação bastante legível e simpática de se manter.

Depois de se dominar o FXML (e só depois mesmo), recomendo a utilização do SceneBuilder para acelerar o desenvolvimento da UI.

Referências

- http://www.oracle.com/technetwork/java/javase/downloads/javafxscenebuilder-info-2157684.html
- https://dominiks.gitbooks.io/javafx/content/i18n/fxmlbundles.html
- http://www.javafxtutorials.com/tutorials/switching-to-different-screens-in-javafx-and-fxml/
- https://docs.oracle.com/javase/8/javafx/user-interface-tutorial/
- https://docs.oracle.com/javase/8/javafx/api/
- https://docs.oracle.com/javase/8/javafx/get-started-tutorial/
- http://docs.oracle.com/javase/8/javafx/layout-tutorial/index.html
- http://code.makery.ch/library/javafx-8-tutorial/

AUTOR

Escrito por Nuno Cancelo

Curioso por natureza e engenheiro informático por formação. Desde muito cedo me despertou o interesse pelos computadores e informática com o famoso Spectrum. Tenho um gosto peculiar por aprender novas coisas frequentemente mesmo que não venha a trabalhar com elas e optimizar os sistemas aumentando a sua performance.

Um bot para Telegram com o Jogo da velha (Jogo do Galo).

Num mundo com tantas aplicações de chat instantâneo, o Telegram destaca-se pela rica API que disponibiliza para criação de bots. Os bots são pequenos programas que podem interagir com os utilizadores e prestar serviços, como executar comandos, gerir arquivos ou imagens e até mesmo propor jogos!

Há já algum tempo que a comunidade Python explora bibliotecas como a Telebot e mais recetemente, a Telepot. Embora a diferença no nome das duas seja apenas uma letra, o desenho da Telepot parece-me mais robusto e o melhor de tudo: integra chamadas assíncronas!

O objetivo deste tutorial é mostrar como criar um bot assíncrono, usando a Telepot em Python 3.6. Ele é divido em quatro partes: por que assíncrono? obtenção da chave para rodar o bot, criação do bot, o jogo da velha em si (com minimax).

Porquê assíncrono?

Sendo um grande fã de programação assíncrona, não poderia perder a oportunidade de escrever um bot assíncrono. Mas porquê assíncrono?

A programação assíncrona veio para simplificar a vida de programadores que não querem e que não precisam de usar threads. Ela é extremamente útil quando a aplicação passa grandes períodos de tempo à espera de uma resposta de um dispositivo de entrada ou saída, como a rede. A ideia da programação assíncrona é dividir o programa em tarefas. Cada tarefa é executada por um loop de eventos, fornecido pela biblioteca que é responsável por ativar e desativar tarefas. Quando uma tarefa precisa de executar uma outra tarefa assíncrona, ela simplesmente pede ao loop para interromper a sua própria execução e retornar apenas quando a outra tarefa tiver terminado. Este ciclo de ativação e desativação se repete até a conclusão de nosso programa. A grande vantagem é que o processador não fica parado à espera de uma resposta da rede, de um arquivo ou da base de dados. Desta forma, vários pedidos podem ser atendidas simultaneamente e tudo isto sem usar threads. Embora a programação assíncrona não substituia a programação paralela em todos os casos, esta simplifica enormemente a escrita dos programas que não precisam de se preocupar com locking ou concorrência de acesso.

No caso de um bot, vários utilizadores do Telegram podem jogar ao mesmo tempo. Para melhorar a utilização do processador, o melhor é escrever um bot assíncrono que vai responder aos comandos do jogo, mas sem bloquear um jogador quando outro está simplesmente à espera de receber a resposta do bot. Isto é possível porque o bot utiliza chamadas REST via http para comunicar com o Telegram. Assim sendo, os novos comandos chegam pela rede e as suas respostas também são enviadas em requisições http, criando uma ótima oportunidade de suspender a execução durante a transmissão e recepção de comandos e libertando o bot para responder a outros pedidos.

Em Python 3.5, a programação assíncrona teve melhorarias importantes, principalmente em relação à sintaxe. Podemos usar async def para criar uma nova co-rotina assíncrona e await para esperar a sua execução. Veremos mais no código do bot em si. Mas para responder à pergunta desta secção, assíncrono para melhorar a utilização do processador e responder a vários jogadores, alternando entre eles durante o processamento de entra/saída, sem os fazer esperar. Como hoje pagamos instâncias na Amazon, Google Compute Engine ou Azure por hora, executar o máximo de pedidos com o mínimo de instâncias é uma pressão financeira importante para o programador. A programação assíncrona é uma alternativa para aliviar esta pressão, sem aumentar a dificuldade de desenvolvimento da solução em si.

Obtenção da chave

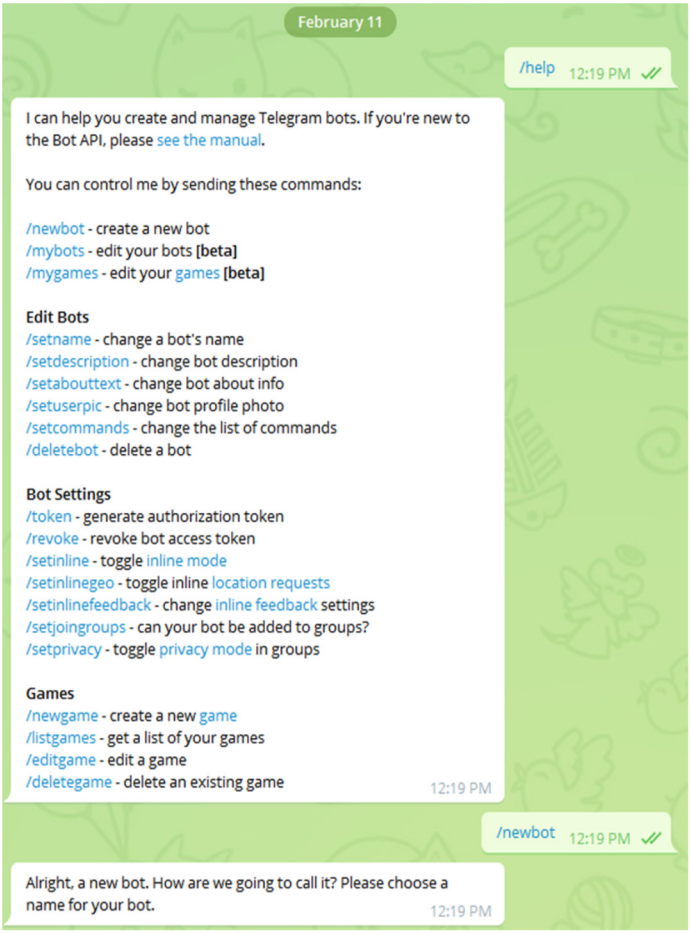
Em primeiro lugar, se ainda não conhece o Telegram, faça o download em https://telegram.org/. Podemos instalá-lo no telefone ou simplesmente criar uma conta para utilizar na web. Basta seguir as instruções do site.

Para criar um novo bot, devemos falar com o BotFather, um bot gerenciador de bots do Telegram. É este bot que vai criar um novo bot e uma chave de acesso que precisamos de obter para usar a API. Podemos falar com o BotFather clicando no link: https://telegram.me/botfather ou procurando por @BotFather.

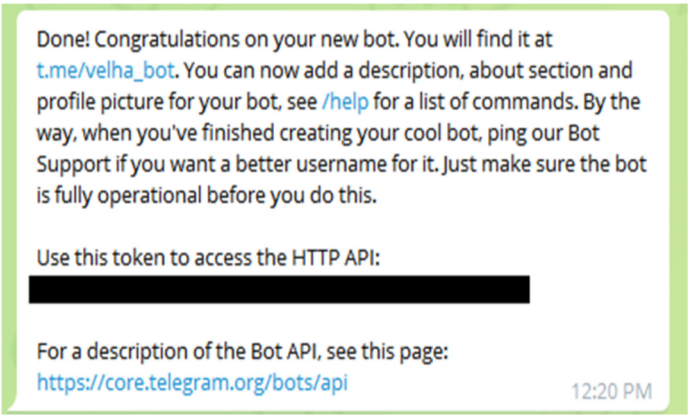
Uma vez com a tela de chat aberta com o BotFather, digitamos o comando:

```
/newbot
```

Não nos podemos esquecer de digitar Enter no fim da linha.



A criação do bot é feita através de uma "conversa" com o BotFather. A primeira coisa que ele pergunta é o nome do bot. Escolhemos um novo nome e escrevemos numa nova linha. Caso o nome já esteja em uso, o bot father pedirá para escolhermos outro.



Depois de aceitar o nome do nosso novo bot, o BotFather exibe esta mensagem. Na segunda linha, ele mostra o novo link para aceder ao nosso bot pela web. No meu caso, o bot chama-se velha_bot. O que procuramos é a chave para aceder à API. Neste caso, a chave para novo bot será mostrada após: "Use this token to access the HTTP API". Eu pintei minha chave de preto, mas deverá aprecer uma string com números e letras. Copiamos esta linha e guardamso num local seguro pois vamos precisar da chave para poder usar a API. A chave parece-se com:

934802948:AODJcjoijcijaoAISDJoiajdoaijAzcWFIS

Pronto, acabamos de criar nosso bot. Agora precisamos de um programa para trazê-lo à vida!

Criação do bot

Vamos começar por fazer o download do bot que desenvolvi para este artigo. O código fonte está disponível no GitHub. É necessário ter o git e o Python 3.6 instalados na máquina para continuar.

Observação: como várias distribuições de Linux ainda não possuem a versão 3.6 disponível nos seus gerenciadores de pacotes, podemos também converter o bot para correr em Python 3.5. As diferenças são pequenas. Apenas substituimos as f-strings como f"{nome}" por "{}".format(nome).

Se utilizarmos Windows, instalamos o git e o Python 3.6, descarregando diretamente de Python.org.

Se utilizarmos Linux (Ubuntu 16.10), instalamos o git e o Python 3.6 com os comandos abaixo:

```
sudo apt update
sudo apt-get install -y make build-essential \
libssl-dev zlib1g-dev libbz2-dev \
libreadline-dev libsqlite3-dev wget \
curl llvm libncurses5-dev libncursesw5-dev \
xz-utils git curl
curl -L https://raw.githubusercontent.com/yyuu/
pyenv-installer/master/bin/pyenv-installer | bash
cat >> ~/.bash_profile <<EOF
export PATH="/$HOME/.pyenv/bin:$PATH"
eval "\$(pyenv init -)"
eval "\$(pyenv virtualenv-init -)"
EOF
source ~/.bash_profile
pyenv install 3.6.0
pyenv virtualenv 3.6.0 velha
pyenv local velha
```

e depois descarregamos o bot com:

```
git clone https://github.com/lskbr/velha.git
```

O git deve descarregar o código fonte e criar um diretório velha no computador.

Digitamos:

```
cd velha
pip install -r requirements.txt
```

Agora vamos precisar da chave que obtivémos do BotFather. Substituímos o texto a vermelho pela chave. Digitamos:

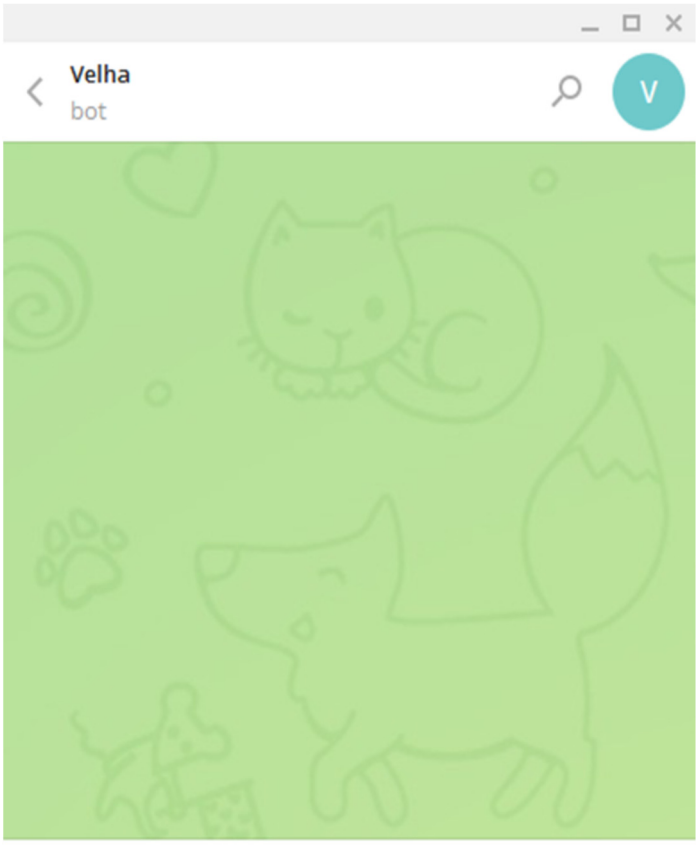
```
export
BOT_TOKEN=934802948:AODJcjoijcijaoAISDJoiajdoaijAzc
WFIS
```

e corremos o bot com:

```
python velha.py
```

Pronto, o bot já deve estar a correr!

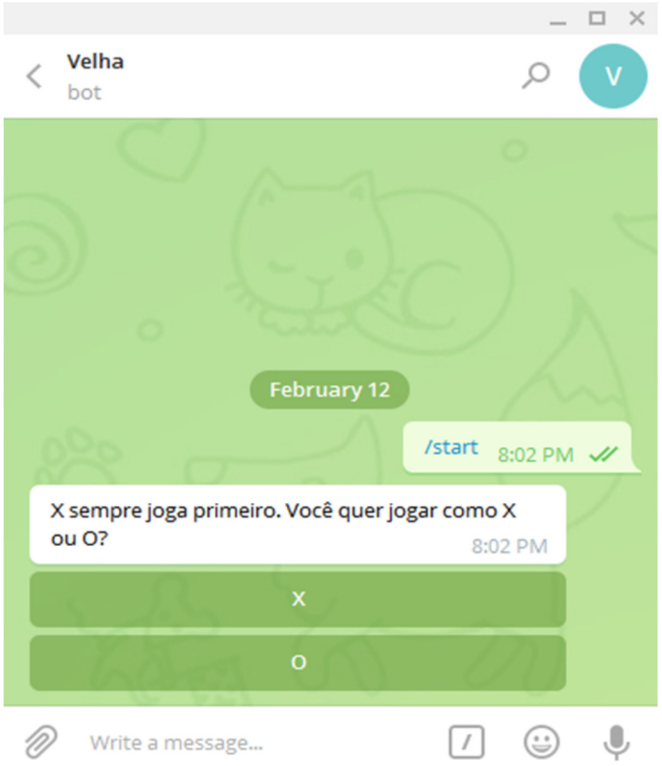
Vamos testá-lo. Usamos a pesquisa do Telegram e procuramos pelo nome do bot. No meu caso é @velha_bot, mas cada um pode substituir pelo nome do seu bot!



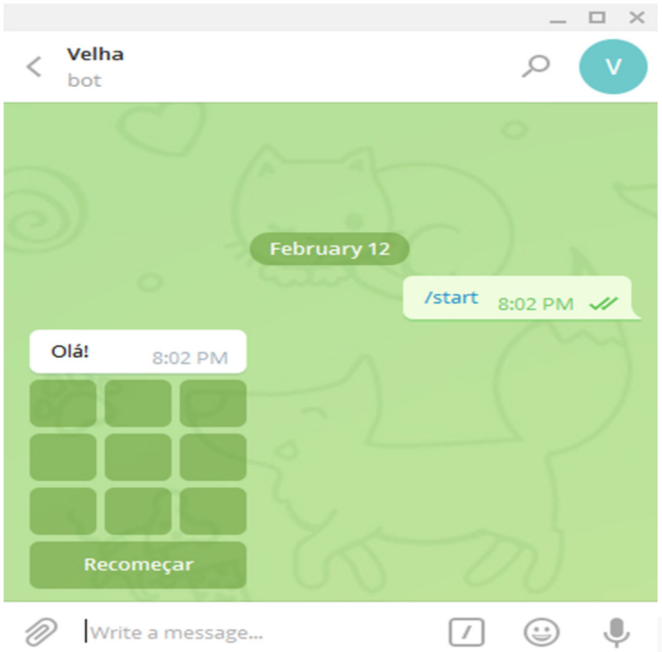
Clicamos ou tocamos em START.

O bot responde e pergunta com qual nível de dificuldade desejamos jogar. Fácil é completamente aleatório, o computador vai simplesmente preencher uma posição vazia. Médio e Difícil usam um algoritmo simples, mas muito interessante do qual vamos falar depois. Eu sugiro que escolhamos Difícil.

O bot então pergunta se queremos começar a jogar (X) ou se queremos jogar depois do computador (O).

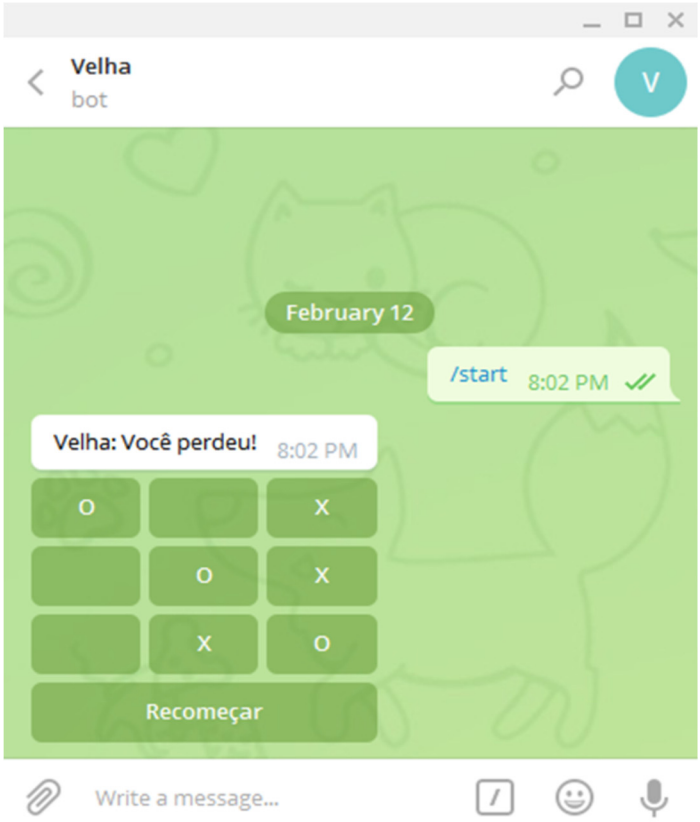


Vejamos que as opções anteriores foram apagadas e que a mensagem foi substituída. Escolhemos X.



Finalmente podemos jogar. Clicamos numa posição disponível e divertimo-nos com o jogo da velha.

Para jogar novamente, clicamos em Recomeçar. O bot avisa o resultado da jogada.



O Jogo da Velha

Agora que instalámos tudo, podemos ver como o programa foi feito! O programa ficou bem maior do que eu esperava, mas vamos por partes.

O programa começa na linha 395, com:

```
# Pega o token da variável de ambiente BOT_TOKEN
TOKEN = os.getenv("BOT_TOKEN")

if __name__ == "__main__":
    logging.basicConfig(level=logging.DEBUG,
                        format='%(asctime)s %
 (levelname)6s %(filename)s %(funcName)s:%(lineno)d
                                          %(message)s')
    loop = asyncio.get_event_loop()
    bot = telepot.aio.Bot(TOKEN, loop=loop)
                                           # Cria o bot
    jogo = JogoDaVelha(bot)               # Cria o jogo
    loop.create_task(jogo.stats(loop))
        # Cria a tarefa que limpa as jogadas velhas
    loop.create_task(bot.message_loop({'chat':
                                  jogo.chat_handler,

'callback_query': jogo.callback_query}))
    try:
        loop.run_forever()
    except KeyboardInterrupt:
        pass
```

Como o TOKEN não deve ser guardado no GitHub (caso contrário seria público), o programa precisa de obtê-lo do ambiente. No caso, usamos os.getenv para obter o token da variável BOT_TOKEN. Por isso precisamos de fazer o export durante a configuração.

Depois de obtermos o TOKEN, configuramos o log do programa e depois o loop de eventos em:

```
loop = asyncio.get_event_loop()
```

Quando escrevemos código para Python assíncrono utilizamos um loop de eventos. Este loop é criado e mantido pelo módulo asyncio. Diferente de uma programa clássico, um programa assíncrono submete rotinas para serem executadas no loop. O loop executa apenas uma rotina de cada vez, o que simplifica o nosso código sem nos preocuparmos com paralelismo. A grande vantagem do loop é que ele pode parar a execução de uma rotina enquanto ela espera por um dispositivo, ou por uma resposta da rede. Veremos em breve como Python faz isto. A contra jogada do loop é que este só executa uma rotina de cada vez, assim, caso uma rotina demore demais a retornar a execução para o loop, todo o programa ficará bloqueado!

Uma vez que temos nosso loop e o token, criamos nosso bot com apenas uma linha!

```
bot = telepot.aio.Bot(TOKEN, loop=loop)
# Cria o bot
```

Utilizamos a teleport.aio que é o módulo da telepot que suporta programação assíncrona. Passamos o TOKEN e o loop como parâmetros.

Finalmente criamos um objeto da classe do Jogo:

```
jogo = JogoDaVelha(bot)  # Cria o jogo
```

Passamos o bot como parâmetro. Isto é importante porque uma vez finalizada a inicialização, é o loop que irá controlar a execução do nosso programa.

Agendamos a execução de uma rotina que exibe quantas jogadas temos em memória com:

```
loop.create_task(jogo.stats(loop))
# Cria a tarefa que limpa as jogadas velhas
```

Vejamos que criámos a tarefa (create_task) no loop,

passando jogo.stats(loop). Aqui as coisas começam a ficar diferentes de um programa tradicional. O método jogo.stats é uma co-rotina e não é executado ao ser chamado. Na realidade, ele cria uma co-rotina que pode ser chamada pelo loop. O código é um pouco diferente de um método normal em Python. O método stats está na linha 386:

```
    async def stats(self, loop):
        """Imprime estatísticas e limpa jogos
                                        antigos"""
        while True:
            jogadas = len(self.jogadas)
            logger.info(f"Jogadas em memória:
                                    {jogadas}")
            self.jogadas.limpa_antigas()
            await asyncio.sleep(60, loop=loop)
```

Vejamos que ele é definido com async def e não com apenas def! É esta mudança que faz com que este método possa ser chamado pelo loop de forma assíncrona. O resto do código é praticamente normal, exceto a última linha:

```
await asyncio.sleep(60, loop=loop)
```

Aqui aparece outra novidade, await. É esta contrução que torna o nosso bot assíncrono especial. Quando usamos await, pedimos ao loop para parar nossa co-rotina atual (criada com async def) para executar uma outra co-rotina, no caso asyncio.sleep que espera 60 segundos para retornar. É importante notar que usamos o sleep de asyncio, pois time.sleep não é uma co-rotina compatível com o loop e se a utilizarmos, todo o programa irá parar enquanto esta não retornar.

Com await, pedimos a execução de asyncio.sleep. O loop então pára a execução de stats, chama asyncio.sleep e volta para stats quando esta terminar. A vantagem é que asyncio.sleep não pára o loop, deixando-o livre para executar outras co-rotinas. E nosso código continua simples, como se esta quebra de execução não tivesse acontecido.

A etapa seguinte é a configuração do bot para chamar métodos do jogo ao receber uma mensagem ou uma resposta (click dos botões):

```
loop.create_task(bot.message_loop({'chat':
jogo.chat_handler,
'callback_query': jogo.callback_query}))
```

Da mesma forma que fizémos com o jogo.stats, vamos criar uma co-rotina que será chamada mais tarde pelo loop. Neste caso, passamos um dicionário com os métodos que queremos chamar quando uma mensagem de chat for recebida e para callback_query. Neste caso, a callback_query é a resposta que o telegram envia quando clicamos nos botões do jogo.

Depois disso, deixamos o loop controlar o nosso programa:

```
    try:
        loop.run_forever()
    except KeyboardInterrupt:
        pass
```

A execução fica dentro de loop.run_forever(), até que a paremos com Control+C.

O loop fica à espera de eventos e executando as co-rotinas que configurámos. No caso do bot, ele periodicamente acede aos servidores do Telegram para saber se há novas mensagens. Quando uma mensagem chega, ele chama jogo.chat_handler:

```
async def chat_handler(self, msg):
        """Processa o chat vindo do utilizador"""
        content_type, chat_type, chat_id =
                                telepot.glance(msg)
        logger.debug(f"Content_type:
    {content_type} Chat type: {chat_type} Messge:
                                        {msg}")
        jogo = self.pega_jogo(msg)
        await self.reply_markup(jogo, chat_id)
```

Vejamos que o método chat_handler foi definido com async def, sendo uma co-rotina. Ele simplesmente recebe a mensagem do telegram, que é um grande dicionário com várias informações sobre a mensagem como: o texto da mensagem, quem a enviou, em que chat, quando foi enviada, etc.

O módulo telepot tem algumas rotinas que ajudam a trabalhar com este dicionário, como a glance que extrai o tipo do conteúdo, o tipo de chat e o chat_id que necessitamos para responder uma mensagem.

Quando bot chama chat_handler, ele não controla nosso jogo em si, ou seja, não sabe que utilizador enviou a mensagem, mas que o bot recebeu uma mensagem. O que fazer com a mensagem é uma responsabilidade do nosso programa. Como podemos ter vários jogos ao mesmo tempo, criamos um dicionário com as jogadas. Cada mensagem tem um user_id de origem e este número identifica cada utilizador de Telegram de forma única (mesmo que ele mude de nome ou de handle). Desta forma, usando o user_id como chave, podemos obter o jogo daquele utilizador. A biblioteca Telepot tem classes muito interessantes que ajudam a manter o contexto por utilizador ou por chat, estas classes não foram utilizadas neste exemplo, consulte a documentação para mais informações.

Chamamos então a reply_markup, passando o jogo e o chat_id da mensgem para que o bot gere uma resposta:

```
    async def reply_markup(self, jogo,
                                    chat_id=None):
        """Dependendo do estado atual do jogo,
                retorna as opções disponíveis"""
        if jogo.tela == 0:
            markup = jogo.nivel_de_dificuldade()
            mensagem = 'Jogo da Velha - Escolha o
                                nível de dificuldade'
        elif jogo.tela == 1:
            markup = jogo.tipo_jogador()
            mensagem = 'X sempre joga primeiro.
                    Você quer jogar como X ou O?'
        elif jogo.tela == 2:
            markup = jogo.constroi_grelha()
            mensagem = jogo.mensagem or 'Jogo da
                                        Velha'

        if jogo.message is None and chat_id is
                                        not None:
            message = await self.bot.sendMessage
          (chat_id, mensagem, reply_markup=markup)
```

```
            jogo.message =
                telepot.message_identifier(message)
        else:
            try:
                await self.bot.editMessageText
      (jogo.message, mensagem, reply_markup=markup)
            except telepot.exception.TelegramError
as te:
                pass
```

Podemos perceber que o bot pergunta o nível de dificuldade, se queremos jogar com X ou O antes de nos deixar jogar. Cada uma destas etapas é controlada por uma propriedade de cada jogo que guarda onde estamos para cada jogador. Quando o ecrã é 0, apresentamos as opções de escolha do nível de dificuldade. Quando é 1, se o jogador quer jogar com X ou O, 2 é o jogo em si e 3 uma mensagem que informa que a jogada já terminou.

Vejamos que jogo.message é uma varíavel que utilizamos para guardar o identificador da mensagem. Isto é necessário, pois para não gerar uma nova grelha a cada resposta, editamos a mensagem precedente (a anterior é substituída por uma nova, dando a impressão de desaparecer). Assim, se jogo.message é nula, enviamos uma nova mensagem com self.bot.sendMessage. É de notar que utilizamos await, pois sendMessage é uma co-rotina do bot. Esta linha é um pouco especial, pois estamos a usar await e atribuindo seu resultado à variável message. Esta é uma outra grande vantagem do await, ele retorna os mesmos valores da co-rotina que acabou de executar. Neste caso, durante o tempo de envio desta mensagem, o loop fica livre para realizar outras tarefas. Quando sendMessage retorna, o programa volta a executar na linha do await e message recebe seu retorno. Vejamos que imediatamente guardamos o identificador da mensagem em jogo.message. Desta forma, na próxima vez, editaremos a mensagem precedente, em vez de enviar uma nova.

No caso de editMessageText, trocamos o conteúdo da mensagem precedente pela nossa nova tela. Um tratamento de exceção básico é realizado, pois um utilizador pode clicar em um jogo antigo e gerar erros (não tratados aqui para simplificar o bot).

Cada ecrã do jogo é composto de uma parte texto e uma série de botões, especificados no parâmetro reply_markup de sendMessage e de editMessageText. O markup que enviamos é gerado em função do estado atual do jogo. Vejamos onde criamos o markup com o nível de diculdade:

```
def nivel_de_dificuldade(self):
        """Retorna as opções dos níveis de
                            dificuldade do jogo"""
        return InlineKeyboardMarkup
                                (inline_keyboard=[
            [InlineKeyboardButton(text="Fácil",
                            callback_data='facil')],
            [InlineKeyboardButton(text="Médio",
                            callback_data='medio')],
            [InlineKeyboardButton(text="Difícil",
                        callback_data='dificil')]])
```

Estamos agora no código da classe Velha, que guarda o estado do jogo para cada jogador (linha 152). O método nível_de_dificuldade é definido na linha 195. Neste caso, simplesmente passamos uma lista de InlineKeyboardButton, cada um com seu texto e callback_data. O texto é mostrado ao utilizador dentro do botão e o callback_data é enviado pelo Telegram caso este botão seja pressionado.

É desta forma que sabemos qual a opção escolhida pelo utilizador. Quando uma resposta do botão é recebida, o bot chama nosso método callback_query (linha 357):

```
async def callback_query(self, msg):
        """Processa a resposta para as escolhas
                                    do utilizador"""
        query_id, from_id, query_data =
     telepot.glance(msg, flavor='callback_query')
        logger.debug(f'Callback Query:
            {query_id}, {from_id}, {query_data}')
        jogo = self.pega_jogo(msg)
        logger.debug(f"Callback query:
     utilizador: {jogo.user_id} mensagem: {msg}")

        if jogo.tela == 0:
            self.configura_dificuldade(jogo,
                                        query_data)
            await self.reply_markup(jogo,
                                self.msg_chat_id(msg))
        elif jogo.tela == 1:
            self.configura_primeiro_jogador(jogo,
                                        query_data)
            if jogo.computador == "X":
                self.joga_pelo_computador(jogo)
            await self.reply_markup(jogo,
                                self.msg_chat_id(msg))
        elif query_data == "recomecar":
            jogo = self.jogadas.novo_jogo(jogo)
            await self.reply_markup(jogo,
                                self.msg_chat_id(msg))
        elif len(query_data) == 1 and
         query_data.isdigit() and jogo.tela == 2:
            posicao = int(query_data) - 1
            if jogo.joga(posicao, jogo.jogador):
                self.verifica_jogada(jogo)
                grelha = jogo.constroi_grelha()
                await self.bot.editMessageText
        (jogo.message, f"Velha: {jogo.mensagem}",
                                reply_markup=grelha)
            else:
                await
              self.bot.answerCallbackQuery(query_id,
                        text='Escolha outra posição')
        elif jogo.tela == 3:
            await self.bot.answerCallbackQuery
               (query_id, text='Jogada terminada.
            Escolha recomeçar para jogar de novo')
```

Da mesma forma que quando recebemos uma mensagem texto, uma mensagem de resposta dos botões é recebida. A resposta em si vem em query_data, extraída pelo telebot.glance (ver que passamos um parâmetro flavor='callback_query':

```
query_id, from_id, query_data = telepot.glance
(msg, flavor='callback_query')
```

O query_data tem o mesmo texto que especificámos no callback_data dos nossos botões. O resto é Python tradicional, onde verificamos o estado do jogo e executamos as configurações necessárias.

Quando estamos na tela 2 e recebemos um número, significa que o utilizador clicou na grelha. A grelha do jogo da velha foi organizada em 9 posições, numeradas de 1 a 9. Como Python indexa a partir do zero, transformamos esta resposta num número inteiro e subtraímos 1 para termos o índice respectivo ou a posição que ele jogou.

Se observarmos o código fonte, veremos que o estado de cada jogo é guardado numa lista de strings com nove elementos. Cada elemento representa uma posição da grelha. Um espaço em branco para posições vazias, X ou O se um dos jogadores já marcou aquela posição anteriormente.

Quando chamamos jogo.joga(posicao, jogo.jogador), pedimos que seja realizada uma nova marcação. Este método retorna False se o utilizador já marcou esta posição e o código gera uma mensagem de erro:

```
await self.bot.answerCallbackQuery(query_id,
text='Escolha outra posição')
```

Vejamos que answerCallbackQuery mostra apenas um overlay no Telegram e não altera a nossa mensagem em si. Experimentemos tentar jogar duas vezes na mesma posícão para ver o efeito. Vamos usar este método para enviar mensagens importantes ao utilizador, especialmente mensagens de erro.

Em jogo.verifica_jogada, testamos se o jogo acabou e retornamos a nova grelha.

Este é o mecanismo geral do jogo. Vejamos um pouco como é que fizémos o computador jogar.

E o computador Joga Velha

O jogo da Velha é muito simples e seria muito chato ter que procurar outro jogador para uma jogada. Vejamos como fazer o computador jogar. Usando uma técnica chamada Minimax (https://pt.wikipedia.org/wiki/Minimax). Uma excelente explicação deste algoritmo para resolver o jogo da velha pode ser encontrada em português aqui: http://www.maawko.com/blog/freecodecamp/jogo-da-velha-entendendo-o-algoritimo-minimax/

O artigo original em Inglês pode ser lido aqui: (http://neverstopbuilding.com/minimax).

Infelizmente, o autor escreveu a solução usando Ruby (brincadeira) e não em Python.

O método consiste em gerar as jogadas possíveis para cada jogador, a partir das posições ainda livres. Depois disso, verificamos o resultado caso jogássemos naquela posição, será que ganhamos? Perdemos? Ou empatamos? A ideia geral é atribuir um certo número de pontos quando ganhamos e o inverso deste número para o jogador oponente.

Este trabalho é feito pela classe Velhus, definida na Linha 24:

```
class Velhus:
    """
    Classe que simula a grelha e permite calcular
                                as jogas possíveis.
    Utilizada para calcular a jogada do computador.

    O Estado contém a grelha como uma lista de
                                          strings.
    Espaço significa que a posição está livre.
    X ou O que o jogador já marcou esta posição.

    Grelha
    Índices   Posições
     0 1 2     1 | 2 | 3
              ---+---+---
     3 4 5     4 | 5 | 6
              ---+---+---
     6 7 8     7 | 8 | 9
    """
    GANHANTES = [set(x) for x in [(0, 1, 2), (3, 4,
                                       5), (6, 7, 8),
                                  (0, 4, 8), (6, 4,
                                                  2),
                                  (0, 3, 6), (1, 4,
                                    7), (2, 5, 8)]]

    def __init__(self, estado=None):
        """estado: estado inicial. Default:
                                          branco"""
        self.estado = estado or [" "] * 9

    def jogadas_possiveis(self):
        """Onde podemos jogar?"""
        return posicoes_de(self.estado, " ")

    def posicoes_por_jogador(self):
        """Retorna uma tupla com as posições do
                        jogador X e do jogador O"""
        return (posicoes_de(self.estado, "X"),
                    posicoes_de(self.estado, "O"))

    def ganhou(self, posicoes, jogador):
        """Verifica se um dos jogadores ganhou a
                                          jogada"""
        s = set(posicoes)
        for p in Velhus.GANHANTES:
            if len(p - s) == 0:
                return True
        return False

    def joga(self, posicao, jogador):
        """Joga pelo jogador em um posição
                                      específica"""
        if self.estado[posicao] == " ":
            self.estado[posicao] = jogador
        else:
            raise ValueError(f"Posição({posicao})
                                        inválida.")

    def resultado(self):
        jX, jO = self.posicoes_por_jogador()
        if self.ganhou(jX, "X"):
            return("X")   # X Ganhou
        elif self.ganhou(jO, "O"):
            return("O")   # O Ganhou
        elif not self.jogadas_possiveis():
            return("*")   # Empate sem resultado
        else:
            return("?")   # Inconclusivo

    @staticmethod
    def alterna(jogador):
        """Inverte o jogodor atual. X --> O e O -->
                                                  X"""
        return "X" if jogador == "O" else "O"

    @staticmethod
    def melhor(result, jogador):
        if jogador == "X":
            return max(result.values())
```

```
        else:
            return min(result.values())

    def proxima(self, jogador, estado, nivel=0,
                                            nmax=3):
        """Cria um dicionário que calcula as possi
                    bilidades de futuras jogadas.
           jogador: jogador corrente (da vez)
           estado: estado do jogo (grelha)
           nivel: nivel atual de recursão, usado
                para diminuir a dificuldade do jogo
           nmax: nível máximo de exploração.
        Retorna caso o nível atual atinja o máximo.
           result: dicionário com a pontuação por
                                        resultado.
        """
        result = {}
        # Percorre as jogadas possíveis
        for possivel in self.jogadas_possiveis():
            j = Velhus(estado[:])  # Cria um
    tabuleiro hipotético, a partir do estado atual.
            j.joga(possivel, jogador)  # joga pelo
                                            jogador
            resultado = j.resultado()  # verifica o
                                resultado da jogada

           if resultado == "X" or resultado == "O":
                rlocal = 10 - nivel    # Atribui
                     pontos com base no nível atual
                result[possivel] = rlocal if
                      resultado == "X" else -rlocal
            elif resultado == "?" and nivel < nmax:
    # Como o resultado não é final, continua a jogar
                outro = self.alterna(jogador)
                lresult = j.proxima(outro,
                          j.estado, nivel + 1, nmax)
                result[possivel] = self.melhor
                 (lresult, outro) if lresult else 0
        return result

    def melhor_jogada(self, jogador, estado, dmax):
        """
        Calcula qual a melhor jogada para o jogador
        jogador: jogador da vez (para qual a melhor
                                jogada será calculada)
        estado: estado atual do jogo
        dmax: nível máximo de profundidade. Usado
                        para diminuir a dificuldade.
        """
        result = self.proxima(jogador, estado,
          nmax=dmax)  # Quais são as possiblidades?
        melhores_jogadas = []
        melhor = self.melhor(result, jogador)

        logger.debug(Velhus.dump_estado(estado))
        logger.debug(f"Jogador={jogador}")

        for posicao, resultado in result.items():
            if resultado == melhor:    # Se esta
                        posição tem o melhor score
                melhores_jogadas.append(posicao)
            logger.debug(f"Melhor: {melhor}
                  {melhores_jogadas} r={resultado}
                                posicao={posicao}")
        return melhores_jogadas
```

Alguns métodos foram removidos para poupar espaço e facilitar o entendimento. Vamos analisar o método próxima. Este método recebe o jogador e o estado da grelha.

Então, para cada posição ainda livre na grelha:

```
for possivel in self.jogadas_possiveis():
```

Criamos uma nova instância de Velhus e jogamos em cada uma destas posições, uma de cada vez. É imortante observar que o estado de Velhus é independente do estado do jogo, pois ainda estamos a verificar onde jogar, ou seja, fazemos tudo isto sem alterar o estado do jogo em si.

```
j = Velhus(estado[:])  # Cria um tabuleiro
                                        hipotético
j.joga(possivel, jogador)   # joga pelo jogador
resultado = j.resultado()  # verifica o resultado
                                        da jogada
```

Cada jogada tem um resultado que pode ser vitória de X, vitória de O, empate ou indeterminado (significando que ninguém ganhou ou perdeu ainda).

Uma vez na posse do resultado, partimos para exeção do minimax:

```
if resultado == "X" or resultado == "O":
    rlocal = 10 - nivel    # Atribui pontos com
                                base no nível atual
    result[possivel] = rlocal if resultado == "X"
                                        else -rlocal
elif resultado == "?" and nivel < nmax:  # Como o
          resultado não é final, continua a jogar
    outro = self.alterna(jogador)
    lresult = j.proxima(outro, j.estado, nivel +
                                            1, nmax)
    result[possivel] = self.melhor(lresult,
                            outro) if lresult else 0
```

Como manda o algoritmo, atribuiremos 10 pontos para a vitória (X indica vitória do jogador X e O de seu oponente). Destes pontos, deduziremos o nível para penalizar os resultados que ocorrem após multiplas jogadas. Um detalhe importante é que o resultado para o jogador X é positivo e para O negativo, daí a inversão de sinal!

Caso o resultado seja indeterminado, chamamos recursivamente próximo, mas alternando o jogador para simular as duas jogadas. Notemos que passamos o novo estado (já com a nossa jogada hipotética) e incrementamos o nível. Utilizamos o nível para limitar a expansão do jogo e criar o nível de dificuldade médio (que não avalia todas as possiblidades). Então, avaliamos o resultado de proxima, mas com a ótica do jogador oponente. Vejamos que o método melhor é importantíssimo, pois decide se estamos a maximizar ou minimizar o resultado:

```
    @staticmethod
    def melhor(result, jogador):
        if jogador == "X":
            return max(result.values())
        else:
            return min(result.values())
```

Como dita o algoritmo de Minimax, o jogador X maximiza as suas jogadas e o jogador O minimiza-as!

O método proxima simplesmente calcula os valores para todas as jogadas possíveis. Desta forma, o método melhor_jogada pode decidir onde o computador vai jogar:

```
def melhor_jogada(self, jogador, estado, dmax):
        result = self.proxima(jogador, estado,
        nmax=dmax)  # Quais são as possiblidades?
```

UM BOT PARA TELEGRAM COM O JOGO DA VELHA (JOGO DO GALO).

```
        melhores_jogadas = []
        melhor = self.melhor(result, jogador)
        for posicao, resultado in result.items():
            if resultado == melhor:   # Se esta
                              posição tem o melhor score
                melhores_jogadas.append(posicao)
        return melhores_jogadas
```

Aplicamos novamente o método melhor para filtrar a melhor jogada. A diferença é que o método melhor_jogada suporta várias posições com o mesmo resultado, ou seja, várias posições onde o resultado é igual ao máximo. As melhores jogadas são retornadas e o computador simplesmente escolhe uma delas aleatoriamente.

> ***Os bots são pequenos programas que podem interagir com os utilizadores e prestar serviços, como executar comandos, gerir arquivos ou imagens e até mesmo propor jogos!***

Este ciclo de jogar, marcar, avaliar os resultados repete-se até o fim da jogada.

Conclusão

Espero que tenham gostado desta solução e que se divirtam criando outros bots para o Telegram. Cada bot é único e apenas arranhamos a superfície do que pode ser feito com a Telepot. O Telegram também disponibiliza toda a documentação da sua api no seu site.

Se não programa em Python, fica o convite para conhecer esta linguagem e a sua comunidade. A programação assíncrona em Python é excelente e o número de bibliotecas cresce todo dia. Base de dados, sites, rede, tupo já pode ser acedido de forma assíncrona.

> ***A criação do bot é feita através de uma "conversa" com o BotFather.***

Escrever bots é divertido, podem ver um outro bot que escrevi usando a biblioteca Telebot no GitHub: https://github.com/PyNorte/pybrtelegrambot. Este bot utiliza uma base de dados SQLite para responder a comandos e guardar os dados dos membros do Grupo Python do Norte do Brasil. Embora não seja um bot assíncrono, tenho certeza que despertará a vossa curiosidade em escrever bots!

AUTOR

Escrito por Nilo Menezes

desenvolvedor de software, especializado em programação paralela, assíncrona e de sistemas distribuídos. Atuou em diversos projetos europeus como pesquisador nas áreas de simulação, telefonia móvel e redes. Coordenou equipes de desenvolvimento de software para indústrias em Manaus, Brasil. Hoje trabalha em sua empresa na Bélgica, prestando consultoria para o desenvolvimento de sistemas escalonáveis e computação em nuvem. É mestre em informática e bacharel em processamento de dados pela Universidade Federal do Amazonas. É autor do livro « Introdução à Programação com Python » editado pela Editora Novatec e disponível tanto no Brasil quanto em Portugal. Nilo pode ser encontrado pelo site: https://python.nilo.pro.br ou no telegram @lskbr e https://t.me/PyCoding.

Automação do Azure com Windows PowerShell

Desired State Configuration

Hoje em dia a automação ajuda muito e é extremamente importante para alguns processos de gestão e administrativos. Um dos principais problemas da automação é não ser aceite por todos. A tecnologia não deve ser usada para substituir ninguém mas sim para ajudar.

Vou mostrar como podem fazer automação com o Windows PowerShell. O Windows PowerShell é uma linguagem scripting da Microsoft que estava reservada aos seus produtos mas isso mudou o PowerShell agora é OpenSource e o código está disponível no GitHub em https://github.com/PowerShell/PowerShell sendo assim possível utilizar em outros sistemas operativos da Apple e Linux. Se pretendem experimentar primeiro tem de instalar o Dot NET Core (https://www.microsoft.com/net/core) e depois o Windows PowerShell (https://github.com/PowerShell/PowerShell/releases/).

O PowerShell já tem todas as funcionalidades da linha de comandos existentes na Microsoft e a vantagem de incluir as bibliotecas e funcionalidades do Dot Net que permite criar scripts de instalação, configuração e gestão de máquinas locais e remotas permitindo aceder ao registo de eventos, processos e serviços, registry e muito mais. Alem dos comandos da Microsoft também inclui comandos do Linux que permite a um administrador de sistemas de Linux conseguir utilizar o Windows PowerShell, por exemplo a listagem de ficheiros.

```
PS C:\pspt> ls


    Directory: C:\pspt


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----        11-01-2017     22:59           1026 00-SecPolicy.PS1
PS C:\pspt> dir


    Directory: C:\pspt


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----        11-01-2017     22:59           1026 00-SecPolicy.PS1
```

Exemplo da listagem de ficheiros

Os comandos de PowerShell que são invocados nos scripts têm a denominação de command-lets (cmdlets). O runtime do PowerShell também os invoca através das API's do Windows PowerShell.

O Windows PowerShell Desired State Configuration (DSC) é um complemento ao PowerShell que inclui novos cmdlets e recursos que permitem criar scripts de instalação, configuração, manutenção ou outros autónomos como por exemplo:

- Instalar novas aplicações e atualizações;
- Executar Windows PowerShell scripts;
- Gerir processos e serviços do sistema;
- Gerir grupos e utilizadores;

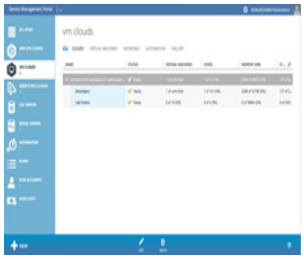
E muito mais.

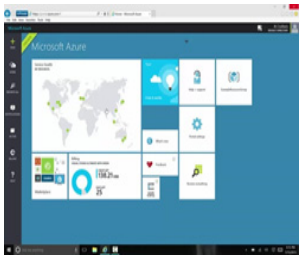
Vou demonstrar como criar uma máquina virtual no Microsoft Azure com o sistema operativo Windows Server 2012 com a funcionalidade Internet Information Services (IIS) ativada e a firewall configurada. O script também funciona com o Windows Server 2016 sendo apenas necessário mudar a imagem do sistema operativo. O script vai criar automaticamente todos os recursos necessários para a máquina como o armazenamento, rede virtual, placa de rede e Internet Protocol (IP) público. Vai instalar IIS e configurar a firewall na máquina virtual e as regras de segurança no Azure.

O Azure como plataforma é um conjunto de serviços que podem ser utilizados para alojar as suas aplicações e cargas de trabalho na nuvem. O Service Management REST API é um serviço web que recebe pedidos para criar, alterar ou configurar os serviços e passa os mesmos para o Microsoft Azure Fabric Controller que toma as decisões com base nesses pedidos e utiliza o Azure Hypervisor para criar novas máquinas virtuais, se necessário, nos datacenters.

O Azure está dividido em dois modelos:

Modelo Clássico	Modelo Atual
Azure Service Management (ASM)	Azure Resource Management (ARM) technology
É utilizado o portal clássico https://	É utilizado o portal atual https://portal.azure.com

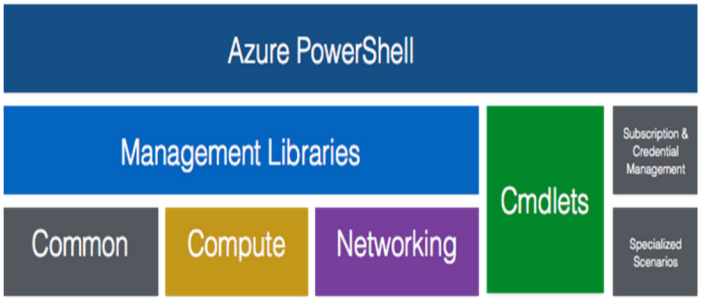




AUTOMAÇÃO DO AZURE COM WINDOWS POWERSHELL

Vai ser utilizado o modelo ARM que introduz uma abordagem para a gestão de recursos do Azure. As instâncias de serviço são referidas como recursos, que podem ser armazenados em grupos de recursos, onde os serviços podem ser criados, geridos, acompanhados, ou podemos excluir todos os serviços simultaneamente. Tudo isto permite uma gestão em grupo em que podem monitorizar-se os serviços em grupo e determinar a taxa de faturação ou recursos de serviços individuais.

Antes de iniciar a criação do script é necessário instalar primeiro os módulos do Azure no PowerShell, que é uma coleção de dois módulos Windows PowerShell que podem ser utilizados para gerir serviços do Azure. O módulo mais comum é Service Management este permite gerir serviços de instâncias no Azure. A seguinte imagem mostra alguns dos componentes contidos no Azure PowerShell



Alguns dos componentes do Azure PowerShell

Pode instalar-se os módulos do Azure de duas formas:

- Instalação Gráfica em https://Azure.microsoft.com/pt-pt/downloads/ na secção PowerShell selecione 'Instalação para Windows'.
- Instalação manual execute uma linha de comandos do Microsoft Windows, Windows PowerShell ou Windows PowerShell ISE para todas estas opções é recomendado executar com privilégios administrativos. Para instalar um módulo utiliza-se o cmdlet 'Install-Module'.

Um módulo pode ser instalado de duas formas:

- Se for instalado por uma conta de administrador o módulo fica disponível para todos os utilizadores e fica instalado em [Unidade de disco local]:\Program Files\WindowsPowerShell\Modules.
- Se um utilizador não tiver privilégios administrativos pode instalar um módulo no seu perfil da máquina local 'Install-Module AureRM -Scope CurrentUser'. A partir do Windows 7 os módulos ficam guardados em '[Unidade de disco local]:\Users\[Nome do utilizador]\Documents\WindowsPowerShell\Modules'.

Além de poder instalar módulos também pode remover e pesquisar outros cmdlest disponíveis:

- 'Find-Script': Procura itens na galeria do Microsoft PowerShell (https://www.powershellgallery.com/);
- 'Save-Module' e 'Save-Script': Guardar os itens da galeria para o sistema;
- 'Install-Module': Instala os itens da galeria;
- 'Publish-Module' e 'Publish-Script': Faz o upload do item para galeria;
- 'Register-PSRepository' : Adiciona um repositório customizado.

Para instalar os módulos execute os seguintes cmdlet, o sinal '#' representa um comentário no script, um conjunto de comentários inicia com '<#' e termina com '#>' todos os comentários estão em escritos em inglês para normalização do script.

```
# Install Azure Resource Manager module/cmdlet
Install-module AzureRM
# Install Azure Classic module/ cmdlet
Install-module Azure
```

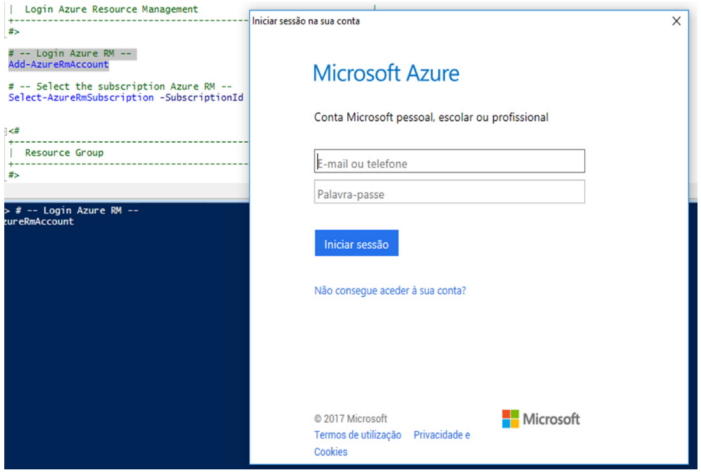
Antes de iniciar a instalação execute o cmdlet 'Set-ExecutionPolicy' que permite alterar a segurança no sistema para permitir correr scripts locais. Pode sempre ativar e desativar esta segurança em modo administrativo ou no utilizador atual.

```
<#
+-----------------------------------------------+
| This is a security change to your system in   |
| order to execute local scripts.               |
+-----------------------------------------------+
#>
# Administrator
Set-ExecutionPolicy -Scope Process -
ExecutionPolicy Bypass
# Current User
Set-ExecutionPolicy -ExecutionPolicy ByPass -
Scope CurrentUser
<#+-----------------------------------------------+
| To reset the permissions back to default run
the following.                                  |
+-----------------------------------------------+
#>
# Administrator
Set-ExecutionPolicy -Scope Process -
ExecutionPolicy Default
# Current User
Set-ExecutionPolicy -ExecutionPolicy Default -
Scope CurrentUser
```

Com a instalação concluída já pode fazer operações no Azure com PowerShell e é partir daqui que inicia o script de automação.

Antes de iniciar qualquer operação no Azure é necessário fazer autenticação no serviço com o cmdlet 'Add-AzureRmAccount'. Vai aparecer uma janela do navegador de internet para introduzir as credenciais da sua conta.

```
<#
+-----------------------------------------------+
| Login Azure Resource Management |
+-----------------------------------------------+
#>
# -- Login Azure RM --
Add-AzureRmAccount
```



cmdlet Add-AzureRmAccount

Após a autenticação se tiver mais que uma subscrição na sua conta é necessário selecionar a subscrição em que pretende trabalhar. Como qualquer linguagem de programação é possível definir variáveis em PowerShell e são definidas como '$NomeDaVariavel = "Valor da Variável"' por exemplo a identificação da sua subscrição do Azure é 'HDJDNHDWN' então devia alterar a variável ' $SubscriptionId = "Your SubscriptionId" ' para ' $SubscriptionId = "HDJDNHDWN" ' antes de selecionar a subscrição.

```
# -- Select the subscription Azure--
$SubscriptionId = "Your SubscriptionId"
Select-AzureRmSubscription -SubscriptionId
                                    $SubscriptionId
```

```
> # -- Login Azure RM --
Add-AzureRmAccount

Environment           : AzureCloud
Account               : [redacted]
TenantId              : 3a8[redacted]975
SubscriptionId        : 066[redacted]F8c
SubscriptionName      : [redacted]
CurrentStorageAccount :

> Select-AzureRmSubscription -SubscriptionId 82[redacted]9

Environment           : AzureCloud
Account               : [redacted]
TenantId              : 44[redacted]8c
SubscriptionId        : 82[redacted]9
SubscriptionName      : Visual Studio Enterprise
CurrentStorageAccount :
```

cmdlet Select-AzureRmSubscription -SubscriptionId

O primeiro serviço que temos de criar no Azure é grupo de resource group (grupo de recursos). Um resource group é um contentor de serviço numa subscrição que permite "armazenar" os vários serviços no Azure. A vantagem é que pode ter vários resource groups separados pelo tipo de serviços. Por exemplo pode ter um resource group apenas para rede virtual um para desenvolvimento e outro para produção. Isto permite separação da manutenção dos serviços. Tudo o que fizer no resource group de desenvolvimento não vai afetar a produção e pode definir administradores para gerir a rede virtual sem que tenha acesso a outros resource groups.

Num resource group é obrigatório definir a localização geográfica e a sua identificação. A Microsoft tem vários data centers espalhados pelo mundo pode consultar em https://Azure.microsoft.com/pt-pt/regions/. Normalmente eu utilizo a Europa do Norte podem existir data centres mais perto/rápido de Portugal mas esta localização geográfica raramente falhou. Para criar um resource group utiliza-se o cmdlet 'New-AzureRmResourceGroup'.

```
<#
+--------------------------------------------+
| Resource Group |
+--------------------------------------------+
#>
# -- Set the Location --
$Location = "NorthEurope"
# -- Create Resource Group --
$ResourceGroup = "PSPTLAB"
New-AzureRmResourceGroup -Name $ResourceGroup -
                    Location $Location -Verbose
```

```
PS C:\> # -- Set the Location --
$Location = "NorthEurope"

# -- Create Resource Group --
$ResourceGroup = "PSPTLAB"
New-AzureRmResourceGroup -Name $ResourceGroup -Location $Location -Verbose
VERBOSE: Performing the operation "Replacing resource group ..." on target "".
VERBOSE: 10:03:35 - Created resource group 'PSPTLAB' in location 'northeurope'


ResourceGroupName : PSPTLAB
Location          : northeurope
ProvisioningState : Succeeded
Tags              :
ResourceId        : /subscriptions/8[redacted]9/resourceGroups/PSPTLAB
```

cmdlet New-AzureRmResourceGroup

Com o resource group criado podem adicionar-se os serviços. O primeiro a ser criado é storage account (conta de armazenamento) que é uma das peças fundamentais num sistema em nuvem sem um sistema de armazenamento não consegue armazenar os registos dos serviços, disco rígido virtual de uma máquina virtual, ficheiros, e muito mais. **Atenção que não existe armazenamento ilimitado até na nuvem há limites**. Em todos os serviços é obrigatório a sua identificação e para grande parte a localização geográfica é também necessária. Para além destes existem vários níveis de serviços no Azure identificado por 'Specific Pricing Tier or Stock Keeping Unit' (SKU) é necessário selecionar o tipo de armazenamento que se pretende pode consultar as diferenças em https://Azure.microsoft.com/pt-pt/pricing/details/storage/blobs/ o mais económico é Locally Redundant Storage (LRS), é este que vou utilizar. Para criar a conta utiliza-se o cmdlet 'New-AzureRmStorageAccount', mas antes de o fazer deve verificar-se se a identificação está a ser utilizada por outra pessoa com o cmdlet 'Get-AzureRmStorageAccountNameAvailability'. A verificação é importante porque a identificação a conta é única em todo o universo do Azure porque pode aceder a sua conta de armazenamento por um endereço de internet.

```
<#
+--------------------------------------------+
| Storage Account |
+--------------------------------------------+
```

```
<#
+-----------------------------------------+
| Storage Account |
+-----------------------------------------+
#>
# -- Set the Storage Account Name and Sku --
$StorageAccountName = "psptlabstorageaccount"
$StorageSkuName = "Standard_LRS" #Specific Pricing
                  Tier or Stock Keeping Unit (SKU)
# -- Verify is the name is available --
# Get-AzureRmStorageAccountNameAvailability $Stora-
geAccountName
# -- Create Storage Account --
$StorageAccount = New-AzureRmStorageAccount
            -ResourceGroupName $ResourceGroup -Name
        $StorageAccountName -SkuName $StorageSkuName
                    -Location $Location -Verbose
```

```
PS C:\>
# -- Set the Storage Account Name and Sku --
$StorageAccountName = "psptlabstorageaccount"
$StorageSkuName = "Standard_LRS"  #Specific Pricing Tier or Stock Keeping Unit (SKU)

# -- Verify is the name is available --
# Get-AzureRmStorageAccountNameAvailability $StorageAccountName

# -- Create Storage Account --
$StorageAccount = New-AzureRmStorageAccount -ResourceGroupName $ResourceGroup -Name $StorageAccountName -SkuName $StorageSkuName -Location $Location -Verbose

PS C:\>
```

cmdlet New-AzureRmStorageAccount

Com o armazenamento criado vai ser criado a rede virtual e gama de endereços para comunicação com os serviços, a placa de rede para a máquina virtual que está na rede virtual criada e o endereço público para a máquina virtual que é definido na placa de rede.

Para criar a rede virtual é necessário primeiro criar o gama do endereçamento da rede com o cmdlet 'New-AzureRmVirtualNetworkSubnetConfig', depois pode criar a rede com o cmdlet 'New-AzureRmVirtualNetwork' e definir a subnet criada.

```
#
+-----------------------------------------------+
| Virtual Network |
+-----------------------------------------------+
#>
# -- Create Subnet --
$Subnet = New-AzureRmVirtualNetworkSubnetConfig
          -Name "PSPTLAB" -AddressPrefix 10.0.0.0/24
# -- Create VNet with Subnet --
$Vnet = New-AzureRmVirtualNetwork -Name
    "PSPTLABVNet" -ResourceGroupName $ResourceGroup
     -Location $location -AddressPrefix 10.0.0.0/16
                        -Subnet $Subnet -verbose
```

```
PS C:\> # -- Create Subnet --
$Subnet = New-AzureRmVirtualNetworkSubnetConfig -Name "PSPTLAB" -AddressPrefix 10.0.0.0/24

# -- Create VNet with Subnet --
$Vnet = New-AzureRmVirtualNetwork -Name "PSPTLABVNet" -ResourceGroupName $ResourceGroup -Location $location -AddressPrefix 10.0.0.0/16 -Subnet $Subnet -verbose
WARNING: The output object type of this cmdlet will be modified in a future release.
VERBOSE: Performing the operation "Creating Resource" on target "PSPTLABVNet".

PS C:\>
```

cmdlet New-AzureRmVirtualNetwork

Com a rede virtual criada pode criar o endereço público no Azure que é definido como Instance-level public IP address (PIP) o cmdlet é New-AzureRmPublicIpAddress deve definir se pretende um endereço estático ou dinâmico.

```
# -- Create a Instance-level public IP address
  (PIP) for Network interface controller (NIC) --
$PublicIp = New-AzureRmPublicIpAddress -Name
"PIP" -ResourceGroupName $ResourceGroup -Location
            $location -AllocationMethod Dynamic
```

```
PS C:\> # -- Create a Instance-level public IP address (PIP) for Network interface controller (NIC) --
$PublicIp = New-AzureRmPublicIpAddress -Name "PIP" -ResourceGroupName $ResourceGroup -Location $location -AllocationMethod Dynamic
WARNING: The output object type of this cmdlet will be modified in a future release.

PS C:\>
```

cmdlet New-AzureRmPublicIpAddress

Agora pode criar a placa de rede com o cmdlet 'New-AzureRmNetworkInterface' é necessário definir a identificação do endereço público.

```
# -- Create NIC with PIP and Subnet --
$NIC = New-AzureRmNetworkInterface -Name "NIC"
     -ResourceGroupName $ResourceGroup -Location
         $location -SubnetId $Vnet.Subnets[0].Id
                -PublicIpAddressId $PublicIp.Id
```

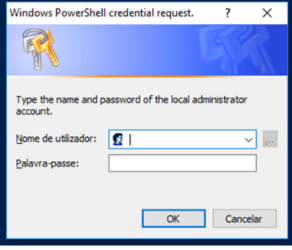
```
PS C:\> # -- Create NIC with PIP and Subnet --
$NIC = New-AzureRmNetworkInterface -Name "NIC" -ResourceGroupName $ResourceGroup -Location $location -SubnetId $Vnet.Subnets[0].Id -PublicIpAddressId $PublicIp.Id
WARNING: The output object type of this cmdlet will be modified in a future release.

PS C:\>
```

cmdlet New-AzureRmNetworkInterface

Com a rede virtual criada pode criar a máquina virtual, a criação da mesma depende todos os serviços anteriores sem os quais não funciona. O Azure disponibiliza várias imagens de sistemas operativos não só da Microsoft mas também de outras marcas como o Red Hat, por exemplo, o utilizador é livre de escolher o que pretende, o valor que vai pagar, mensalmente, já inclui todo o licenciamento. Tal como a conta de armazenamento há também vários tipos de máquinas virtuais e todas elas são diferentes, pode consultar em https://azure.microsoft.com/pt-pt/pricing/details/virtual-machines/. Vou utilizar uma máquina Standard da série Dv2 destinada a computação para fins gerais de próxima geração com o sistema operativo Microsoft Windows Server 2012 R2 Datacenter. Para qualquer sistema operativo é preciso definir uma conta de administrador e as suas credenciais de acesso em PowerShell, pode definir de duas formas.

- Solicitar os dados ao utilizador com o cmdlet 'Get-Credential –Message' e o script fica em standby até serem inseridas as credencias.

```
PS C:\> $cred = Get-Credential -Message "Type the name and password of the local administrator account."
```



cmdlet Get-Credential

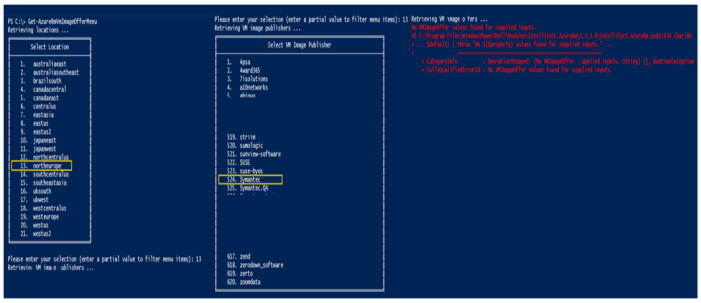
- Definir uma variável e converter para um string protegida com o cmlet 'ConvertTo-SecureString'.

```
PS C:\> # -- Create a PSCredential for Local Administrator Account from a Clear Text Password --
$SecurePassword = ConvertTo-SecureString "PSPTlab+2017" -AsPlainText -Force
$Credential = New-Object System.Management.Automation.PSCredential ("itpro", $SecurePassword);

PS C:\> $SecurePassword
System.Security.SecureString
```

System.Management.Automation.PSCredential

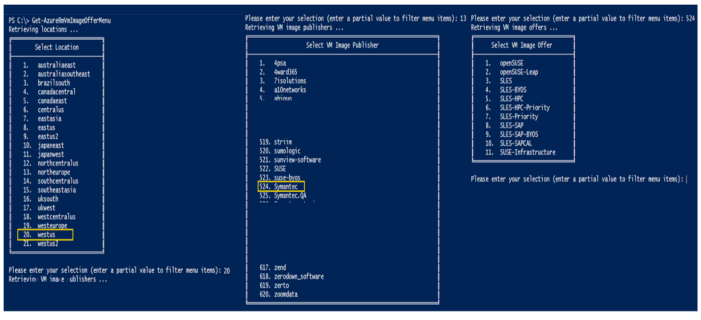
Como se pretende automação vai ser utilizada a segunda opção mas se partilhar o script com alguém lembre-se de apagar a palavra passe, nome de utilizador e de administração. O nome de utilizador que estou a utilizar é 'itpro' existem nomes que são reservados para o Azure como Admin e Root. Para além das credenciais, na criação da máquina virtual vão definir-se todos os serviços criados anteriormente. O processo de criação é mais complexo e vai ser utilizado vários cmdlets. O 'New-AzureRmVMConfig' é para criar uma configuração da máquina virtual. O 'Set-AzureRmVMOperatingSystem' é para definir a identificação da máquina, credenciais de acesso, atualizações automáticas e outras definições. O 'Set-AzureRmVMSourceImage' define-se a imagem que se pretende utilizar; a Microsoft oferece imensas imagens. O seguinte cmdlet permite pesquisar as imagens existentes através PowerShell 'Get-AzureRmVmImageOfferMenu' pode experimentar com a opção 13, 367, 1. Nem todas as imagens estão disponíveis em todas as regiões.

Por exemplo as máquinas virtuais da Symantec não estão disponíveis no Norte da Europa.



Imagens Symantec no norte da Europa

Ao selecionar Oeste dos EUA já tem máquinas virtuais disponíveis.



Imagens Symantec no oeste dos EUA

O 'Add-AzureRmVMNetworkInterface' vai adicionar a placa de rede criada. O 'Set-AzureRmVMOSDisk' vai criar o disco virtual na conta de armazenamento. Por fim para criar a máquina virtual utiliza-se o cmdlet 'New-AzureRmVM' definindo a configuração e localização geográfica.

```
<#
+-----------------------------------------------+
| Virtual Machine |
+-----------------------------------------------+
#>
# -- Virtual Machine Settings --
$VMName = "PSPTLABVM"
$VMSize = "Standard_DS1_v2"
$VMPublisherName = "MicrosoftWindowsServer"
$VMPublisherOffer = "WindowsServer"
$VMSkus = "2012-R2-Datacenter"
$VMOSDiskName = "PSPTLABVMOsDisk"
# -- Create a PSCredential for Local
                        Administrator Account --
#$cred = Get-Credential -Message "Type the name
and password of the local administrator account."
# -- Create a PSCredential for Local
          Administrator Account from a Clear Text
                                        Password --
$SecurePassword = ConvertTo-SecureString "{INSERT
                THE PASSWORD" -AsPlainText -Force
                        $Credential = New-Object
        System.Management.Automation.PSCredential
                    ("itpro", $SecurePassword);
# -- Create the Virtual Machine --
$VM = New-AzureRmVMConfig -VMName $VMName -VMSize
$VMSize
$VM = Set-AzureRmVMOperatingSystem -VM $VM -
Windows -ComputerName $VMName -Credential
         $Credential -ProvisionVMAgent -WinRMHttp
                                -EnableAutoUpdate
#$VM = Set-AzureRmVMOperatingSystem -VM $VM -
        Windows -ComputerName $VMName -Credential
        $cred -ProvisionVMAgent -EnableAutoUpdate
$VM = Set-AzureRmVMSourceImage -VM $VM -
PublisherName $VMPublisherName -Offer
$VMPublisherOffer -Skus $VMSkus -Version "latest"
$VM = Add-AzureRmVMNetworkInterface -VM $VM -Id
                                        $NIC.Id
$VM = Set-AzureRmVMOSDisk -VM $VM -Name
$VMOSDiskName -VhdUri "https://
psptlabstorageaccount.blob.core.windows.net/vhds/
    psptlabvmosdisk1.vhd" -CreateOption fromImage
New-AzureRmVM -ResourceGroupName $ResourceGroup
              -Location $location -VM $VM -Verbose
```

cmdlet New-AzureRmVM

Com a máquina configurada vai ser instado o Internet Information Services (IIS) por DSC.

Para utilizar o DSC no Azure são necessários vários passos. O script é enviado para ser publicado no Azure, antes de ser publicado é verificado a sua configuração, apos á validação o script fica publicado no Azure Automation DSC LifeCycle e pode ser utilizado em qualquer máquina virtual no Azure.

O script é enviado para o repositório.

O repositório importa e valida o scrip e faz a configuração da informação

Implementa a configuração do script que se encontra no repositório

Os scripts podem ter mais de uma configuração e serem aplicados em mais do que uma máquina. Mas, antes de publicar, é necessário compreender a estrutura do ficheiro DSC que é constituído por blocos.

Ambiente da configuração	Configuração, que contem a identificação da configuração.	Configuration ConfigName { ….
	Parâmetros que são usados na configuração. Este grupo é opcional.	param([Parameter (Mandatory=$true)] [String[]]$Param7)
Configuração estrutural	Implantar a configuração nas máquinas definidas. Pode ter um ou mais blocos.	Node localhost {

	Recurso que pretende configurar. Pode ter um ou mais blocos. O recurso é a propriedade do cmdlet 'Get-WindowsFeature'	WindowsFeature IIS {
		Name = "Web-Server"
		Ensure = "Present"
		}
		…

Automação de idempotêntes (Propriedade de uma operação de poder ser aplicada mais do que uma vez sem que o resultado se altere)		foreach -parallel ($featureName in $Name)
		{
		$feature = Get-WindowsFeature -Name $featureName
		if(($Ensure -eq "Present") -and (!$feature.Installed))
		{
		Install-WindowsFeature -Name $featureName
		}
		….
		}
		…

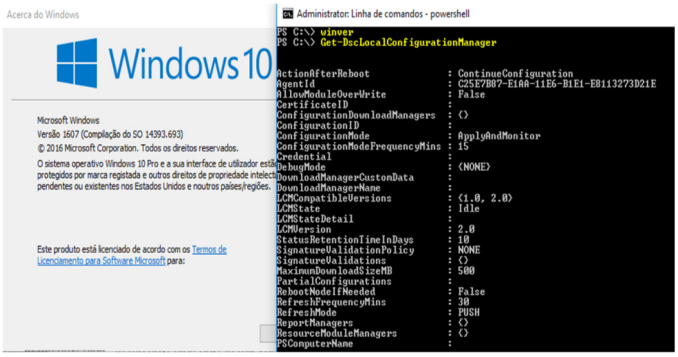
Estes são os módulos disponíveis no DSC, é recomendado que instale os que necessita para utilizar Intellisense na criação dos scripts.

```
PS C:\> Get-DSCResource | Select-Object -Property Name | Format-Wide -Column 2

Loading default CIM keywords
   Processing
Getting module list
   Processing


File                                SignatureValidation
cAzureStorage                       Archive
Environment                         Group
GroupSet                            Log
Package                             ProcessSet
Registry                            Script
Service                             ServiceSet
User                                WaitForAll
WaitForAny                          WaitForSome
WindowsFeature                      WindowsFeatureSet
WindowsOptionalFeature              WindowsOptionalFeatureSet
WindowsPackageCab                   WindowsProcess
xADComputer                         xADDomain
xADDomainController                 xADDomainDefaultPasswordPolicy
xADDomainTrust                      xADGroup
xADOrganizationalUnit               xADRecycleBin
xADUser                             xWaitForADDomain
xAzureAffinityGroup                 xAzureQuickVM
xAzureService                       xAzureSqlDatabase
xAzureSqlDatabaseServerFirewallRule xAzureStorageAccount
xAzureSubscription                  xAzureVM
xAzureVMDscConfiguration            xAzureVMDscExtension
xDefaultGatewayAddress              xDHCPClient
xDnsClientGlobalSetting             xDnsConnectionSuffix
xDNSServerAddress                   xFirewall
xHostsFile                          xIPAddress
xNetAdapterBinding                  xNetAdapterRDMA
xNetBIOS                            xNetConnectionProfile
xNetworkTeam                        xNetworkTeamInterface
xRoute
```

cmdlet Get-DSCResource

Se usar algum destes módulos é necessário definir a sua importação no script a ser utilizado na máquina de destino. Por exemplo para definições na Firewall 'Import-DSCResource -ModuleName xNetworking'. O "x" nos módulos com o xNetworking significa experimental estes recursos são corrigidos e monitorizados pelos autores do módulo.

O DSC não funciona apenas no Windows Server também funciona em máquinas clientes como o Windows 10 Pro.



comando winver e cmdlet Get-DscLocalConfigurationManager

Um dos requisitos necessários para trabalhar com DSC é saber qual a versão do PowerShell que está a ser utilizada e pode verificar-se com o cmdlet $PSVersionTable.PSVersion. O DSC foi implementado na versão 4 no Windows 10 build 14393 é utilizada a versão 5.

```
PS C:\> $PSVersionTable.PSVersion

Major  Minor  Build  Revision
-----  -----  -----  --------
5      1      14393  693
```

cmdlet PSVersionTable

O que significa que estamos a usar a framewrok Windows Management Framework (WMF) 5 esta framework contem o Windows Remote Management, Windows PowerShell e Background Intelligent Transfer Service e introduz o Windows PowerShell OneGet. Em 27 de janeiro de 2017 foi lançada a versão 5.1 que é compatível com o sistema operativo Windows 7, Windows 8.1, Windows Server 2008 R2, Windows Server 2012, e Windows Server 2012 R2, Windows Server 2016 as novidades das nova versão são publicadas no blog oficial da equipa https://blogs.msdn.microsoft.com/powershell e no msdn em https://msdn.microsoft.com/en-us/powershell/.

Antes de utilizar o DSC no Windows 8 e Windows Server 2012 R2 é necessário verificar se tem instalado a correção KB2894179, KB2883200, e KB2894029 para instalar pode executar o cmdlet 'Get-Hotfix' para instalar o KB2883200 execute 'Get-Hotfix –Id KB2883200'.

Este é o conteúdo do ficheiro DSC para instalar o IIS.

Ambiente da configuração	Configuração com identificação DSCIIS.	Configuration DSCIIS {
Configuração estrutural	Implantar a configuração na própria máquina local.	node "localhost" {

	Instalação do IIS.	WindowsFeature IIS
		{
		Ensure = "Present"
		Name = "Web-Server"
		}
		}
		}

Guarde o ficheiro com o nome 'DSCIIS.ps1' na mesma localização do script de automação.

```
Get-Command .\DSCIIS.ps1 | Format-List *
```

```
PS C:\pspt>  Get-Command .\DSCIIS.ps1 | Format-List *


HelpUri             :
Path                : C:\pspt\DSCIIS.ps1
Definition          : C:\pspt\DSCIIS.ps1
Source              : C:\pspt\DSCIIS.ps1
Visibility          : Public
ScriptBlock         : Configuration DSCIIS
                      {

                        node "localhost"
                          {
                              WindowsFeature IIS
                              {
                                  Ensure = "Present"
                                  Name = "Web-Server"
                              }
                          }

                      }
OutputType          : {}
ScriptContents      : Configuration DSCIIS
                      {

                        node "localhost"
                          {
                              WindowsFeature IIS
                              {
                                  Ensure = "Present"
                                  Name = "Web-Server"
                              }
                          }

                      }
OriginalEncoding    : System.Text.UTF8Encoding
Name                : DSCIIS.ps1
CommandType         : ExternalScript
Version             :
ModuleName          :
Module              :
RemotingCapability  : PowerShell
Parameters          : {}
ParameterSets       : {}
```

cmdlet Get-Command

A equipa da Microsoft Windows PowerShell tem um repositório central no GitHuB de recursos do DSC em https://github.com/PowerShell/DscResources.

Voltando ao script de automação a primeira tarefa é a validação e publicação, utiliza-se o cmdlet 'Publish-AzureRmVMDscConfiguration '. O script vai primeiro ser validado antes de ser enviado. Se o mesmo não passar a validado retorna o tipo do erro e a localização do erro.

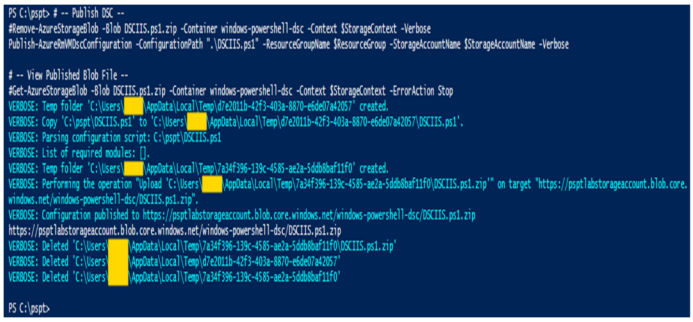
```
PS C:\> Publish-AzureRmVMDscConfiguration -ConfigurationPath ".\DSCIIS.ps1" -ResourceGroupName $ResourceGroup -StorageAccountName $StorageAccountName -Verbose
Publish-AzureRmVMDscConfiguration : The pipeline has been stopped.
At line:1 char:1
+ Publish-AzureRmVMDscConfiguration -ConfigurationPath ".\DSCIIS.ps1" - ...
+ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    + CategoryInfo          : CloseError: (:) [Publish-AzureRmVMDscConfiguration], PipelineStoppedException
    + FullyQualifiedErrorId : Microsoft.Azure.Commands.Compute.Extension.DSC.PublishAzureRmVMDscConfigurationCommand

Publish-AzureRmVMDscConfiguration : Configuration file 'C:\DSCIIS.ps1' not found.
At line:1 char:1
+ Publish-AzureRmVMDscConfiguration -ConfigurationPath ".\DSCIIS.ps1" - ...
+ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    + CategoryInfo          : InvalidArgument: (:) [Publish-AzureRmVMDscConfiguration], ArgumentException
    + FullyQualifiedErrorId : InvalidArgument,Microsoft.Azure.Commands.Compute.Extension.DSC.PublishAzureRmVMDscConfigurationCommand

PS C:\>
```

cmdlet Publish-AzureRmVMDscConfiguration erro no script

Por defeito todos os scirpts publicados vão estar no container 'windows-powershell-dsc' que é criado automaticamente, se não pretender utilizar o container por defeito pode criar um e definir o mesmo no cmdlet. Pode consultar todos os ficheiros que estão no container no portal do Azure, PowerShell, Visual Studio ou Explorador de Armazenamento/ Storage Explorer compatível com sistemas operativos Microsoft, Linux e Mac que está disponível nas transferências na página do Azure em https://azure.microsoft.com/pt-pt/downloads/.

```
<#
+-----------------------------------------------+
| Desired State Configuration (DSC) |
| Install Internet Information Services (IIS) |
+-----------------------------------------------+
#>
```

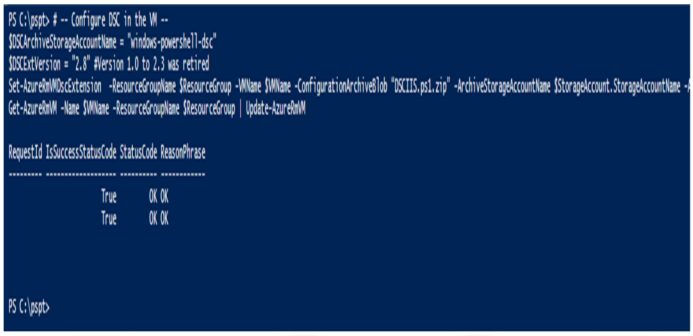
```
# -- Publish DSC --
#Remove-AzureStorageBlob -Blob DSCIIS.ps1.zip -
Container windows-powershell-dsc -Context
                              $StorageContext -Verbose
Publish-AzureRmVMDscConfiguration -
ConfigurationPath ".\DSCIIS.ps1" -ResourceGroupName
$ResourceGroup -StorageAccountName $StorageAccount-
Name -Verbose
# -- View Published Blob File --
#Get-AzureStorageBlob -Blob DSCIIS.ps1.zip -
Container windows-powershell-dsc -Context $Stora-
geContext -ErrorAction Stop
```



cmdlet Publish-AzureRmVMDscConfiguration

Com o DSC corretamente validado e publicado podemos aplicar em qualquer máquina virtual para aplicar utiliza-se o cmdlet 'Set-AzureRmVMDscExtension' é necessário definir a conta de armazenamento, o container onde está o script, o nome do script e da máquina virtual. Um ponto importante é definir a versão da extensão do DSC. Após aplicar na máquina é necessário atualizar a mesma.

```
# -- Configure DSC in the VM --
$DSCArchiveStorageAccountName = "windows-powershell
                                              -dsc"
$DSCExtVersion = "2.8" #Version 1.0 to 2.3 was
                                            retired
Set-AzureRmVMDscExtension -ResourceGroupName
                    $ResourceGroup -VMName $VMName
          -ConfigurationArchiveBlob "DSCIIS.ps1.zip"
                          -ArchiveStorageAccountName
                $StorageAccount.StorageAccountName
                           -ArchiveResourceGroupName
                 $StorageAccount.ResourceGroupName
-ArchiveContainerName $DSCArchiveStorageAccountName
-ConfigurationName "DSCIIS" -Version $DSCExtVersion
-AutoUpdate:$true -Force
     Get-AzureRmVM -Name $VMName -ResourceGroupName
                    $ResourceGroup | Update-AzureRmVM
# -- Remove DSC From the VM --
#Remove-AzureRmVMDscExtension -ResourceGroupName
$ResourceGroup -VMName $VMName -verbose
```



Set-AzureRmVMDscExtension

Enquanto o DSC está a ser aplicado pode monitorizar a sua aplicação com o cmdlet 'Get-AzureRmVMDscExtensionStatus' em que tem que definir o nome da máquina e a que grupo de recursos pertence. Não pode executar esta verificação na mesma janela de PowerShell tem que abrir uma nova janela de linha de comandos, Windows PowerShell ou PowerShell ISE. Fazer uma nova autenticação no Azure RM e executar o cmdlet.

```
$VMName = "PSPTLABVM"
$ResourceGroup = "PSPTLAB"
Get-AzureRmVMDscExtensionStatus -VMName $VMName -
ResourceGroupName $ResourceGroup -Verbose
```

cmdlet Get-AzureRmVMDscExtensionStatus

Com o IIS instalado na máquina virtual é necessário abrir os portos de acesso ao IIS por defeito é 80 para Hypertext Transfer Protoco (HHTP) e 443 para Hyper Text Transfer Protocol Secure (HTTPS). É preciso definir em dois lugares no Firewall da máquina virtual e na segurança no Azure Network Security Groups (NSG).

Para configurar a firewall na máquina virtual vai ser utilizado um outro script DSC.

Ambiente da configuração	Configuração com identificação DSCFirewallConfig.	configuration DSCFirewallConfig {
	Tem um parâmetro que define qual a máquina que vai ser afetada.	param ([string[]]$NodeName = 'localhost')
	Importação dos módulos DSC necessários.	Import-DSCResource -ModuleName xNetworking
Configuração estrutural	Implantar a configuração na máquina definido pelo parâmetro.	Node $NodeName {

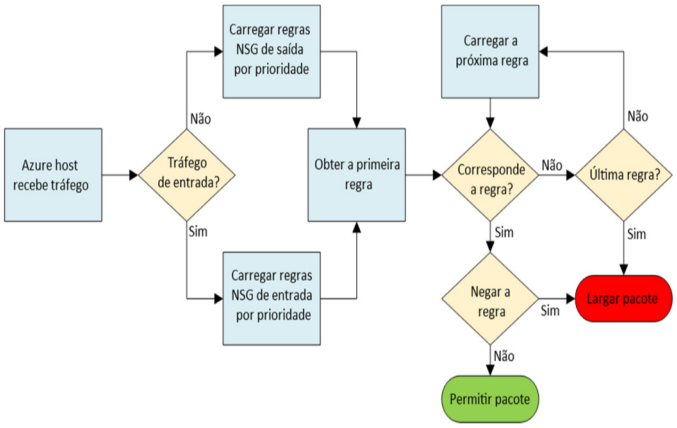
	Configuração da Firewall na máquina Virtual.	xFirewall Firewall
		{
		Name = "IISFirewallRule"
		DisplayName = "Firewall Rule for IIS"
		Ensure = "Present"
		Action = "Allow"
		Profile = ("Domain", "Private", "Public")
		Direction = "OutBound"
		RemotePort = (443, "80", "8080")
		LocalPort = (443, "80", "8080")
		Protocol = "TCP"
		Description = "Firewall Rule for IIS"
		Service = "WinRM"
		}
		}
		}

O método de aplicação é exatamente igual ao anterior DSC do IIS. O script é validado se for válido é publicado e pode ser utilizado.

```
#
+-----------------------------------------------+
| Desired State Configuration (DSC) |
| Firewall Open TCP Ports for IIS |
+-----------------------------------------------+
#>
# -- Install xNetworking module current user --
Install-Module -Name xNetworking -Scope CurrentUser
# -- Publish DSC --
#Remove-AzureStorageBlob -Blob DSCFirewallCon-
fig.ps1.zip -Container windows-powershell-dsc -
Context $StorageContext -Verbose
                    Publish-AzureRmVMDscConfiguration
        -ConfigurationPath ".\DSCFirewallConfig.ps1"
                    -ResourceGroupName $ResourceGroup
    -StorageAccountName $StorageAccountName -Verbose
# -- View Published Blob File --
#Get-AzureStorageBlob -Blob DSCFirewallCon-
fig.ps1.zip -Container windows-powershell-dsc -
Context $StorageContext -ErrorAction Stop
# -- Configure DSC in the VM --
Set-AzureRmVMDscExtension -ResourceGroupName
                        $ResourceGroup -VMName $VMName
                            -ConfigurationArchiveBlob
                          "DSCFirewallConfig.ps1.zip"
                            -ArchiveStorageAccountName
                    $StorageAccount.StorageAccountName
                            -ArchiveResourceGroupName
                        $StorageAccount.ResourceGroupName
-ArchiveContainerName $DSCArchiveStorageAccountName
-ConfigurationName "DSCFirewallConfig" -Version
                $DSCExtVersion -AutoUpdate:$true -Force
Get-AzureRmVM -Name $VMName -ResourceGroupName
                        $ResourceGroup | Update-AzureRmVM
```

execução do DSC para firewall da máquina virtual

Com a firewall definida na máquina virtual define-se agora o Azure NSG. O NSG são listas de acesso ou Access Control List (ACL) que permitem ou negam o tráfego exterior com o Azure de uma sub-rede da rede virtual (VNet). Como qualquer sistema de segurança o NSG contém dois tipos de regras, regras de entrada e regras de saída e é necessário definir as prioridades da regra.



Mostra como as regras do NSG são processadas (docs.microsoft.com).

Antes de criar um NSG primeiro criam-se as regras de acesso só depois se cria o NSG e é associado a um NIC. Vão ser criadas duas regras para permitir tráfego externo do protocolo HyperText Transfer rotocol (HTTP) que utiliza a porta 80 e Remote Desktop Protocol (RDP) para acesso remoto da máquina virtual que utiliza a porta 3389. Para criar as regras utiliza-se o cmdlet 'New-AzureRmNetworkSecurityRuleConfig' que, como indica o nome é a descrição da regra, os portos de origem e destino, e a sua permissão se autoriza ou nega. Um dos pontos fundamentais ao definir uma regra é a sua prioridade que é numerada entre 100 a 4096 a prioridade é definida do menor para o maior.

```
<#
+-----------------------------------------------+
| Azure Network Security Groups (NSG) |
+-----------------------------------------------+
#>
# -- Create NSG Rules --
$NSGRDPRule = New-
AzureRmNetworkSecurityRuleConfig -Name "rdp-rule"
          -Description "Allow RDP" -Access "Allow"
    -Protocol "Tcp" -Direction "Inbound" -Priority
                100 -SourceAddressPrefix "Internet"
    -SourcePortRange * -DestinationAddressPrefix *
                        -DestinationPortRange 3389
$NSGHTTPRule = New-
AzureRmNetworkSecurityRuleConfig -Name "web-rule"
-Description "Allow HTTP" -Access "Allow" -
Protocol "Tcp" -Direction "Inbound" -Priority 101
-SourceAddressPrefix "Internet" -SourcePortRange
                    * -DestinationAddressPrefix * -
                            DestinationPortRange 80
```

cmdlet New-AzureRmNetworkSecurityRuleConfig

Com as regras de acesso criadas pode associar a criação do NSG, para tal utiliza-se o comando 'New-AzureRmNetworkSecurityGroup' que tem que definir o grupo de recursos, localização e as regras a adicionar. Pode adicionar uma ou várias regras em simultâneo.

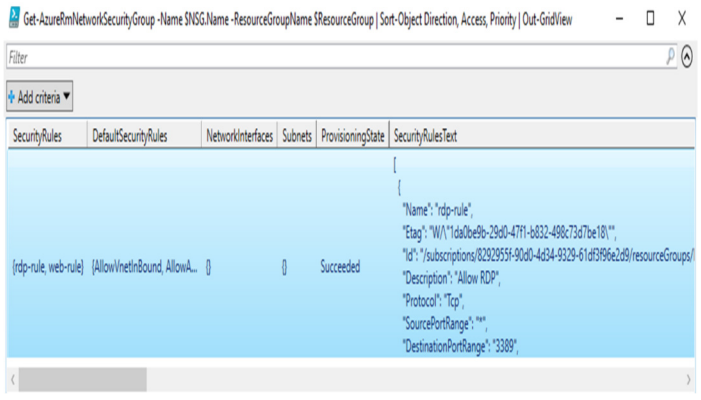
AUTOMAÇÃO DO AZURE COM WINDOWS POWERSHELL

```
# -- Create NSG --
$NSG = New-AzureRmNetworkSecurityGroup -
        ResourceGroupName $ResourceGroup -Location
      $Location -Name "NSG-FrontEnd" -SecurityRules
                              $NSGRDPRule, $NSGHTTPRule
# -- Associate NSG to NIC --
$NIC.NetworkSecurityGroup = $NSG
Set-AzureRmNetworkInterface -NetworkInterface $NIC
```

cmdlets New-AzureRmNetworkSecurityGroup e Set-AzureRmNetworkInterface

Pode sempre ver o estado das regras com o cmdlet 'Get -AzureRmNetworkSecurityGroup' a que grupo de recursos pretende que os resultados sejam mostrados como uma tabela numa janela utilizando 'Out-GridView'

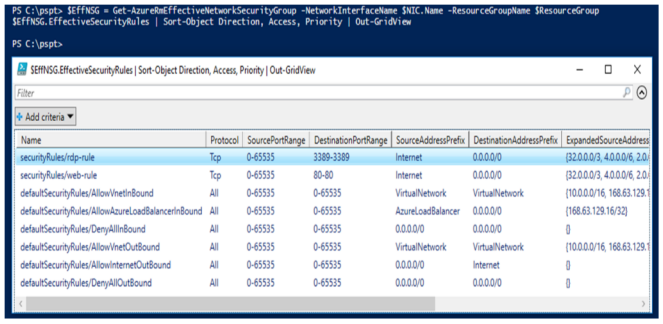
```
# -- View NetWork Security Group --
#Get-AzureRmNetworkSecurityGroup -Name $NSG.Name -
     ResourceGroupName $ResourceGroup | Sort-Object
      Direction, Access, Priority | Out-GridView --
```



cmdlet Get-AzureRmNetworkSecurityGroup com Out-GridVew

Pode fazer um filtro das regras com o cmdlet 'Get-AzureRmEffectiveNetworkSecurityGroup' indicando a NIC e a que grupo de recursos pretende que os resultados sejam mostrados como uma tabela numa janela utilizando 'Out-GridView'

```
# -- View filtering --
#$EffNSG = Get-
          AzureRmEffectiveNetworkSecurityGroup -
NetworkInterfaceName $NIC.Name -ResourceGroupName
                                    $ResourceGroup
#$EffNSG.EffectiveSecurityRules | Sort-Object
       Direction, Access, Priority | Out-GridView
```

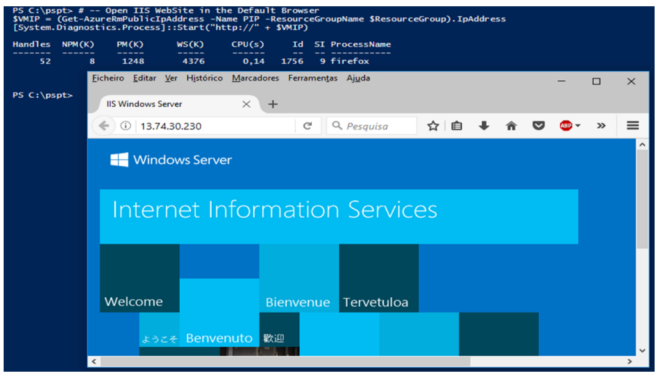


cmdlet Get-AzureRmEffectiveNetworkSecurityGroup

Com as regras criadas está finalizado o script de automação e pode ser utilizado quando quiser apenas tem de mudar as variáveis existentes para o ambiente pretendido.

No entanto o powershell permite fazer muito mais, como abrir a página do IIS através do endereço público de Internet da máquina virtual. Se quiser utilizar o endereço de Internet é necessário configurar o Fully Qualified Domain Name (FQDN) no NIC associado a máquina virtual. Pode abrir o navegador de Internet preferencial através do dot Net com '[System.Diagnostics.Process]::Start()' o endereço IP é obtido com o cmdelt 'Get-AzureRmPublicIpAddress'.

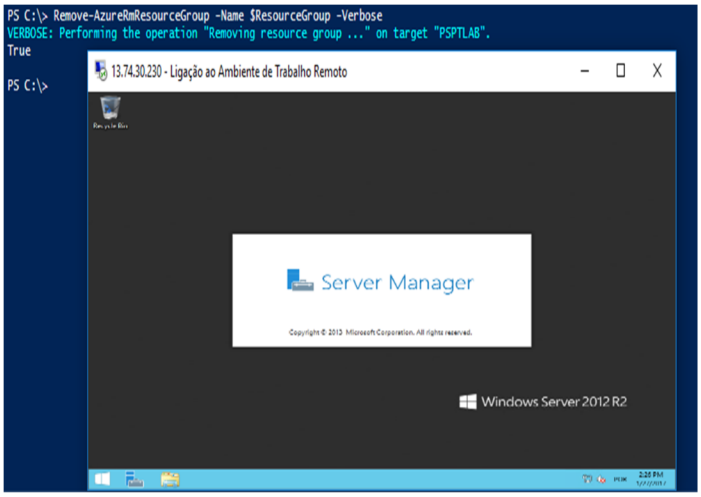
```
<#
+-------------------------------------------+
| Open VM IIS WebSite |
+-------------------------------------------+
#>
# -- Open IIS WebSite in the Default Browser
$VMIP = (Get-AzureRmPublicIpAddress -Name PIP -
     ResourceGroupName $ResourceGroup).IpAddress
[System.Diagnostics.Process]::Start("http://" +
                                        $VMIP)
```



execução do navegador de Internet por PowerShell

Também pode abrir uma ligação de ambiente de trabalho remoto ou outra aplicação com 'Start-Process'.

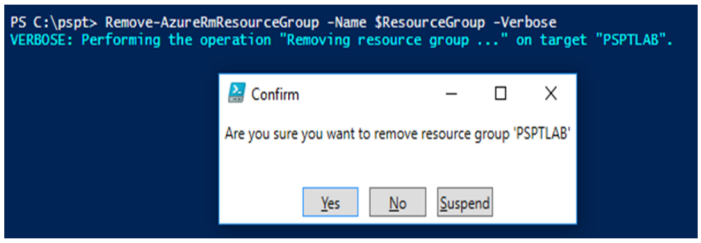
```
<#
+-----------------------------------------------
--------------+
| Open Remote Desktop Session |
+-----------------------------------------------
--------------+
#>
# -- Launh Microsoft Terminal Server Connection
#Start-Process "mstsc" -ArgumentList "/V:$VMIP /
w:1024 /h:768"
```



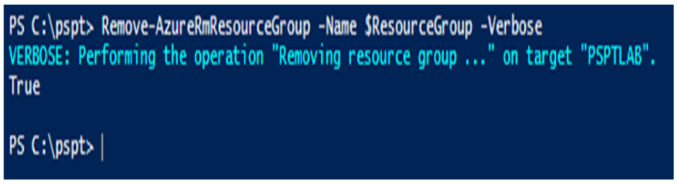
execução do Start-Process

Se pretender destruir o grupo de recurso utilizado utilize o cmdlet 'Remove-AzureRmResourceGroup' indicando o nome do grupo de recursos.

```
<#
+-----------------------------------------------
+
| Destroy module 5 demo environment |
+-----------------------------------------------+
#>
# -- Warning: Delete the Resource Group
#Remove-AzureRmResourceGroup -Name $ResourceGroup -
                                          Verbose
```



confirmação da remoção do grupo de recursos



remoção do grupo de recursos com sucesso

O nível de segurança da conta do Azure é a mesma, nunca é mantido nenhum perfil na máquina a não ser que grave o seu perfil com cmdlet 'Save-AzureRmProfile'.

Em conclusão o DSC é ótimo para automação não apenas para administradores de sistemas mas também para DevOps, administradores de bases dados e muitos mais. Pode fazer quase tudo incluindo implementar uma atualização de segurança em várias máquinas com o xWindowsUpdate o seguinte exemplo vai instalar a atualização KB2908279 que está na própria máquina.

```
Configuration UpdateWindowsWithPath
{
Import-DSCResource -ModuleName
'PSDesiredStatedConfiguration',' xWindowsUpdate'
Node 'NodeName'
{
xHotfix HotfixInstall
{
Ensure = "Present"
Path = "c:/temp/Windows8.1-KB2908279-v2-x86.msu"
Id = "KB2908279"
}
}
}
```

O DSC não funciona apenas no Azure também funciona em máquinas locais. O processo é quase idêntico, enquanto no Azure o repositório e a implementação na máquina são tratados automaticamente, enquanto localmente é necessário gerir o repositório/ficheiros para serem aplicados nas máquinas remotas e a sua implementação. A vantagem do Azure é que trata disso tudo por nós.

Se pretende saber mais sobre Windows PowerShell convido a consultar o blog oficial do PowerShell em https://blogs.msdn.microsoft.com/powershell/ , a documentação da Microsoft em https://technet.microsoft.com/en-us/library/bb978526.aspx e o Microsoft Script Center em https://technet.microsoft.com/pt-pt/scriptcenter/ .

Também está convidado a participar na comunidade PowerShell Portugal (https://pt-pt.facebook.com/PowerShellPortugal/) como participante e/ou orador.

AUTOR

Escrito por Ricardo Cabral

O Ricardo Cabral é apaixonado e autodidata em tecnologia da informação com mais 13 anos de experiência em projetos, desenvolvimento e gestão de TI. Licenciou-se em Engenharia de Informática pela Universidade Autónoma de Lisboa. Participante ativo, voluntariado e/ou orador de reuniões de comunidade portuguesas. Adora partilhar, conviver e aprender. O seu twitter é @rramoscabral.

O Problema do casamento Estável
utilizando o Algoritmo Gale-Shapley

Introdução

O problema do emparelhamento estável (stable marriage problem), é de forma resumida o problema de encontrar um emparelhamento estável entre dois elementos de dois conjuntos de elementos, dada a ordem de preferências de cada elemento do conjunto.

Este problema é normalmente apresentado da seguinte forma: Dados n Reis e n Damas de um conjunto de cartas, cada Rei e cada Dama estabelece uma ordem de preferência para cada um dos elementos "opostos" (reis ou damas), com quem gostaria de estabelecer um "relacionamento", ou por outras palavras, tomar um café e trocar uns bytes de código! Os pares são estabelecidos de forma a que os pares de elementos opostos prefiram estar "juntos" no par estabelecido, do que estar com qualquer outro elemento. Quando não existirem pares que cumpram estes requisitos o conjunto de pares é considerado estável.

Para que um "par" não seja considerado estável, as seguintes premissas são observáveis:

- Um dado elemento A do primeiro conjunto de pares, prefere um elemento B do segundo conjunto de pares, ao elemento com que se encontra emparelhado, e
- O elemento B também prefere A, ao elemento ao qual B se encontra emparelhado.

Notemos que estas duas premissas se encontram ligadas por um operador e (and), o que significa que ambas têm de ser observadas para que um par seja considerado estável.

Em suma um par é considerado estável quando não existe nenhum emparelhamento (A,B) em que ambos A e B estivessem individualmente melhor do que que estão com o par com que se encontram emparelhados num dado momento.

Nota Importante: Neste problema existem duas classes que necessitam de ser emparelhadas, uma com a outra. Caso contrário poderia ser confundido com o problema dos colegas de quarto (*stable roommates problem*)

Existem diversos usos quotidianos do algoritmo Gale-Shapley e do problema do emparelhamento estável, com uma grande abrangência de áreas de conhecimento, desde a economia, passando pela distribuição de alunos por cursos, distribuição de salas de aulas por turmas e horários, gestão de transplantes de órgãos vivos (em uso nos EUA), etc..

A teoria do Algoritmo

Em 1962, David Gale e Lloyd Shapley, dois matemáticos norte-americanos, provaram matematicamente que para cada número igual de homens e mulheres, é sempre possível de resolver o problema do emparelhamento estável, e estabelecer todas as relações de forma estável. Para tal apresentaram um algoritmo que se propunha a realizar tal tarefa. Este algoritmo, realiza um número de iterações, com etapas definidas. Na primeira iteração do algoritmo, cada elemento h, propõem-se a um elemento m) do género oposto, que ele prefere. Por sua vez, cada elemento m) responde "talvez" ao candidato que prefere, e "não" a todos os outros candidatos. Nesta etapa, cada elemento m) fica temporariamente emparelhado com o elemento que lhe é mais preferido até ao momento, tal como o candidato h) fica temporariamente emparelhado com esse mesmo elemento m). Em todas as iterações subsequentes, primeiro cada elemento h) não emparelhado, propõe-se ao elemento m) de sua preferência, a quem não se tinha proposto na iteração anterior. Esta tarefa é realizada independentemente do facto de o elemento m) já estar ou não temporariamente emparelhado. Feita esta etapa, cada elemento m) reponde "talvez" se não estiver emparelhada nesse dado momento, ou se prefere esse proponente por oposição ao seu par provisório. Caso o elemento m) tenha preferido o novo candidato, ao seu par actual, então o candidato passa ao estado "disponível". Este procedimento permite que cada elemento m) troque de par pelo que lhe for "preferido", a cada iteração. Este processo decorre até que todos os pares estejam "emparelhados".

A complexidade de execução deste algoritmo é de O (n^2) onde **n** é o numero de elementos homens ou mulheres.

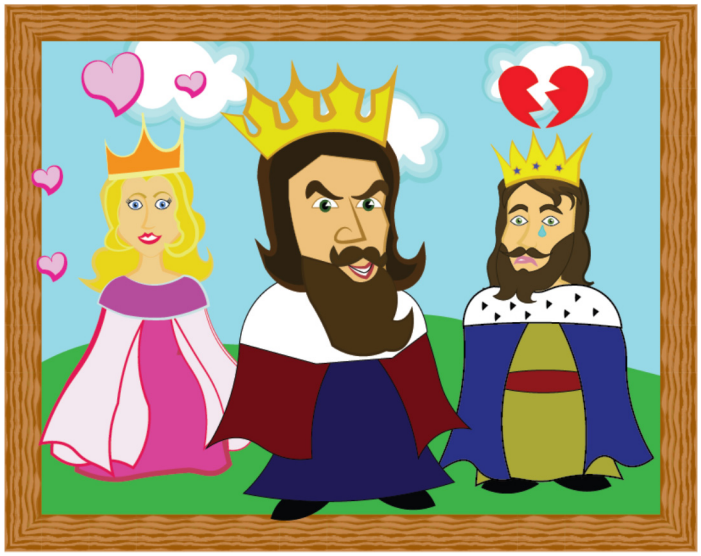
Este algoritmo garante dois resultados essenciais:

1. Todos os pares são estabelecidos (no final não pode ocorrer situação nenhuma em que um par esteja desemparelhado, de tal forma que todos os homens se propuseram a uma determinada mulher, num dado momento no tempo, dado que um homem irá eventualmente propor-se a todas as mulheres se necessário. Situação em que cada mulher teve garantidamente pelo menos uma proposta.

2. Os pares são estáveis. Considerando dois personagens (Pedro e Inês), estão ambos emparelhados, mas não juntos, após todas as iterações do algoritmo. Assim sendo não será possível que eles se prefiram mutuamente por oposição aos seus pares estabelecidos, porque num dado momento se Pedro preferiu Inês, ele ter-se-á proposto, antes de se propor ao seu par actual. Assim, se Inês o tivesse aceite a sua proposta, e ainda assim não tenha terminado emparelhada com ele, no final do algoritmo, isto significa que ela o trocou por um outro elemento de quem ela gostou

mais. Assim sendo, ela não pode preferir Pedro ao seu par actual. Se por algum acaso ela rejeitou a sua proposta, muito provavelmente ela já estaria emparelhada com um elemento de quem ela gostaria mais do que de Pedro. *

* Para os efeitos desta explicação considera-se como "bug" um elemento externo ao grupo com o nome de Afonso IV, (neste caso seria "rei dos glitches" pois a sua intervenção resultaria numa falha!



A versão clássica do problema do casamento estável, (stable marriage problem) pode traduzir-se de forma simples da seguinte forma:

Sendo H = {h_1, . . . , h_n} e M = {m_1, . . . , m_n} os conjuntos de homens e mulheres, um emparelhamento *E* é uma qualquer função injectiva de *H* em *M*. Informalmente, um emparelhamento é, neste caso, um conjunto de *n* casais (monogâmicos e heterosexuais). Um emparelhamento *E* diz-se instável se e só se existir um par (h, m) Ï/E tal que h prefere outro m à sua parceira em E e m também prefere outro h ao seu parceiro em E. Caso contrário, diz-se estável.

Considerando para efeitos de exemplo uma instancia em que n = 4 e as listas de preferências se consideram ordenadas por ordem (estritamente) decrescente da esquerda para a direita,

h_1	m_4	m_2	m_3	m_1		m_1	h_4	h_2	h_1	h_3
h_2	m_2	m_3	m_4	m_1		m_2	h_3	h_1	h_4	h_2
h_3	m_2	m_3	m_4	m_1		m_3	h_2	h_3	h_1	h_4
h_4	m_1	m_3	m_2	m_4		m_4	h_3	h_4	h_2	h_1

pode-se verificar, através duma simples análise de casos, que {(h_1, m_4), (h_2, m_3), (h_3, m_2), (h_4, m_1)} é um emparelhamento estável, passível de ser obtido usando o algoritmo de Gale-Shapley.

Apesar da simplicidade do algoritmo na sua implementação inicial, ficou provado que o resultado seria sempre mais vantajoso para o grupo que inicia-se a iteração (quer fossem os homens, quer fossem as mulheres)., tal como podemos ver no pseudo-código seguinte.

```
function emparelhamentoEstavel {
    Initializa todos h ∈ H e m ∈ M para livres
    while ∃ homem h livre que ainda tem uma
                        mulher m a quem se propor{
        m = primeira mulher na lista de h para
                        quem h ainda não se propôs
        If m Is livre Then
          (h, m) ficam emparelhados
       else algum par (h', m) já existe
         if m prefere h a h'
                h() 'fica livre
           (h, m) ficam emparelhados
            Else
           (h', m) permanece emparelhados
    }
}
```

A prova de que existe vantagem para o grupo que inicia as propostas, pode ser obtida, recorrendo à redução ao absurdo. Se existisse no conjunto um par (*h, m*) $\notin E$ que se tornasse num casamento instável, então *h* teria de preferir *m* à noiva *m'* com quem foi emparelhado e *m* teria de preferir *h* ao noivo com quem ficou emparelhada. No entanto se *h* prefere *m* a *m'* então *h* propôs-se a *m* antes de se propor a *m'*. Tal resultaria no facto de *m* só rejeitar *h*, na altura na mesma altura ou posteriormente, para manter ou ficar com o noivo que preferisse. Vejamos um exemplo de código em que após terem sido formados os pares, se perturba o conjunto resultante, para formar um conjunto de relacionamentos instáveis, para posteriormente se verificar a sua estabilidade (*código em linguagem c, segundo o paradigma procedimental*).

```
#include <stdio.h>

int verbose = 0;
enum {
      burro = -1,
      alvin, bart, calvin, donnald, eric, farquad,
      george, hercules, iago, jasper, alice, bela,
     cinderela, daisy, esmeralda, Fiona, giselle,
                        hanna, isabela, jasmine,
};
const char *nome[] = {
      "Alvin", "Bart", "Calvin",  "Donnald",
         "Eric",  "Farquad", "George", "Hercules",
  "Iago", "Jasper", "Alice", "Bela", "Cinderela",
       "Daisy", "Esmeralda", "Fiona",  "Giselle",
                    "Hanna", "Isabela", "Jasmine"
};
int pref[jasmine + 1][jasper + 1] = {
      { alice, esmeralda, cinderela, isabela,
   jasmine, daisy, Fiona, bela, hanna, giselle },
      { cinderela, hanna, alice, daisy, esmeralda,
        Fiona, bela, jasmine, isabela, giselle },
      { hanna, esmeralda, alice, daisy, bela,
   Fiona, isabela, giselle, cinderela, jasmine },
      { isabela, Fiona, daisy, giselle, hanna,
    esmeralda, jasmine, bela, cinderela, alice },
      { jasmine, daisy, bela, cinderela, Fiona,
     esmeralda, alice, isabela, hanna, giselle },
      { bela, alice, daisy, giselle, esmeralda,
    isabela, cinderela, jasmine, hanna, Fiona },
```

O PROBLEMA DO CASAMENTO ESTÁVEL UTILIZANDO O ALGORITMO GALE-SHAPLEY

```
      { giselle, esmeralda, isabela, bela,
  cinderela, alice, daisy, hanna, jasmine, Fiona },
      { alice, esmeralda, hanna, Fiona, isabela,
        cinderela, jasmine, bela, giselle, daisy },
      { hanna, cinderela, daisy, giselle, bela,
       alice, Fiona, isabela, jasmine, esmeralda },
      { alice, Fiona, jasmine, giselle, esmeralda,
          bela, daisy, cinderela, isabela, hanna },

      { bart, farquad, jasper, george, iago, alvin,
              donnald, eric, calvin, hercules   },
      { bart, alvin, calvin, farquad, george,
         donnald, iago, eric, jasper, hercules   },
      { farquad, bart, eric, george, hercules,
           calvin, iago, alvin, donnald, jasper   },
      { farquad, jasper, calvin, alvin, iago,
         hercules, george, donnald, bart, eric   },
      { jasper, hercules, farquad, donnald, alvin,
            george, calvin, eric, iago, bart   },
      { bart, alvin, eric, iago, jasper, donnald,
            farquad, george, calvin, hercules   },
      { jasper, george, hercules, farquad, bart,
          alvin, calvin, eric, donnald, iago   },
      { george, jasper, bart, alvin, iago, donnald,
            hercules, eric, calvin, farquad   },
      { iago, calvin, hercules, george, farquad,
          bart, alvin, eric, jasper, donnald   },
      { eric, hercules, george, alvin, bart,
        jasper, calvin, iago, farquad, donnald   },
};
int pairs[jasmine + 1], propoeric[jasmine + 1];

void parear(int homem, int mulher)
{
      pairs[homem] = mulher;
      pairs[mulher] = homem;
      if (verbose) printf("%4s é emparelhado com
                %4s\n", nome[homem], nome[mulher]);
}

void deixar(int mulher, int homem)
{
      pairs[homem] = pairs[mulher] = burro;
      if (verbose) printf("%4s deixa %4s\n", nome
                              [mulher], nome[homem]);
}

int rank(int this, int that)
{
      int i;
      for (i = alvin; i <= jasper && pref[this]
                                    [i] != that; i++);
      return i;
}

void propoe(int homem, int mulher)
{
      int goce = pairs[mulher];
      if (verbose) printf("%4s propõe-se a %4s\n",
                           nome[homem], nome[mulher]);

      if (goce == burro) {
            parear(homem, mulher);
      } else if (rank(mulher, homem) < rank(mulher,
                                              goce)) {
            deixar(mulher, goce);
            parear(homem, mulher);
      }
}

int desejar(int noivo1, int noiva2)
{
      if (rank(noivo1, noiva2) < rank(noivo1, pairs
   [noivo1]) && rank(noiva2, noivo1) < rank(noiva2,
                                    pairs[noiva2])) {
            printf( "    %4s (c/ %4s) e %4s (c/ %4s)
                                    preferem-se mutuamente"
                  " sobre o par actual.\n",
                  nome[noivo1], nome[pairs
   [noivo1]], nome[noiva2], nome[pairs[noiva2]]);
            return 1;
      }
      return 0;
}

int mulher_do_vizinho(int noivo1, int noivo2)
{
      return desejar(noivo1, pairs[noivo2]) +
                  desejar(noivo2, pairs[noivo1]);
}

int instavel()
{
      int i, j, bad = 0;
      for (i = alvin; i < jasper; i++) {
            for (j = i + 1; j <= jasper; j++)
             if (mulher_do_vizinho(i, j)) bad = 1;
      }
      return bad;
}

int main()
{
      int i, unparearric;

      for (i = alvin; i <= jasmine; i++)
            pairs[i] = propoeric[i] = burro;

      do {
            unparearric = 0;
            for (i = alvin; i <= jasper; i++) {
                  if (pairs[i] != burro) continue;
                  unparearric = 1;
               propoe(i, pref[i][++propoeric[i]]);
            }
      } while (unparearric);

      printf("Pareando:\n");
      for (i = alvin; i <= jasper; i++)
            printf("  %4s - %s\n", nome[i],
                  pairs[i] == burro ? "burro" :
                                    nome[pairs[i]]);

      printf(instavel()
            ? "Casamento não estável\n"
            : "Par estável\n");

      printf("\n Mas se Bart e Farquad
                                    trocassem:\n");
      i = pairs[bart];
      parear(bart, pairs[farquad]);
      parear(farquad, i);
      printf(instavel() ? "Os pares não eram
           estaveis\n" : "Pareamento estável\n");

      return 0;
```

Até aqui, em todos os exemplos apresentados, não utilizamos preferências com empates. Ou seja o mesmo elemento (homem ou mulher), definir um mesmo valor de preferência (*rank*) a dois ou mais elementos do género oposto. Caso o tivesse-mos permitido existiriam três noções de estabilidade. A *fraca estabilidade* , a *super estabilidade* e a *alta estabilidade*.

Um par é chamado *estável-fraco*, a menos que haja um par em que cada um dos elementos prefira estritamente o outro para o seu par no emparelhamento.

O PROBLEMA DO CASAMENTO ESTÁVEL UTILIZANDO O ALGORITMO GALE-SHAPLEY

O cientista Robert W. Irving estendeu o algoritmo de Gale-Shapley por forma a fornecer esse emparelhamento de *fraca estabilidade* em $E(n^2)$ onde *n* é tamanho do problema do casamento estável (*Stable Marriage Problem*) . Os laços da lista de preferências de homens e mulheres são quebrados arbitrariamente e as listas de preferências são reduzidas à medida que o algoritmo prossegue, tal como podemos observar no pseudocódigo seguinte.

```
Assign each person to be free;
      while (some man m is free) do
      begin
            w := first woman on m's list;
            m proposes, and becomes engaged, to w;
            if (some man m' is engaged to w) then
                assign m' to be free;
            for each (successor m'' of m on w's
                                              list) do
                  delete the pair (m'', w)
end;
output the engaged pairs, which form a stable
                                          matching
```

Um emparelhamento é considerado *super-estável* se não existir nenhum par em que ambos os parceiros prefiram estritamente o outro para o seu parceiro ou seja indiferente entre eles.

O mesmo cientista modificou o algoritmo para verificar a existência de uma correspondência *super estável* e resultados correspondentes a $E(n^2)$ no caso de existirem tais emparelhamentos. No exemplo seguinte em pseudo-código é ilustrado este mesmo conceito.

```
atribuir each pessoa to livre;
 repeat
      while (algum homem h is livre) do
     for each (mulher m na cabeça da lista of h) do
        begin
        h propeo, and fica emparelhado, to m;
            for each (sucessor directo de h' of h
                                 on lista de m ) do
            begin
                  if (h' is emparelhado) to m then
                     quebrar emparelhamento;
                  apagar o par (h'. m);
            end
        end
      for each (mulher m que está em emparelhamento
                                          multiplo) do
      begin
            quebrar todos os emparelhamentos
                                       envolvendo M;
            for each (homem h na cauda da lista de
                                                m) do
              apagar o par (h. m)
     end;
 until (lista de algum homem is vazia) or (todos is
                                       emparelhados);
 if todos is emparelhados then
      a relação de emparelhamento is é um
                        emparelhamento super-estável
 else
      não existe emparelhamento super-estável
```

Um emparelhamento é *super estável* se não houver par *h, m* tal que h prefira estritamente m ao seu par e m prefira estritamente y ao seu par ou seja indiferente entre os pares. Robert W. Irving desenvolveu o algoritmo que verifica se existe essa correspondência fortemente estável e produz a correspondência se existir. O algoritmo computa perfeita correspondência entre o conjunto de homens e mulheres, assim, encontrando conjunto crítico de homens que estão envolvidos com várias mulheres. Uma vez que tais emparelhamentos nunca são estáveis, pois todos os pares são apagados a sequencia de propostas será repetida outra vez até que a lista de preferências de algum homem fique vazia, na qual nenhum emparelhamento *super-estável* exista ou seja obtido. De seguida vejamos o pseudo-código que encontra uma correspondência fortemente estável.

```
attribuir each pessoa to livre;
 repeat
      while (algum homem h is livre) do
            for each (mulher m na cabeça da lista
                                          de h) do
        begin
        h propoe, and fica emparelhado, to m;
            for each (sucessor directo h' of h on
                                    lista de m) do
            begin
                  if (h' is emparelhado) to m then
                     quebrar emparelhamento;
                  apagar o par (h'. m)
            end
        end
     if (o emparelhamento não contem um
                     emparelhamento perfeito) then
     begin
          encontrar o conjunto critico Z de homens
          for each (mulher m que está emparelhada
                              com um homem em Z) do
          begin
              quebrar todos os emparelhamentos
                                    envolvendo m;
              for each homem h na cauda da lista
                                          de m do
                  apagar o par (h, m)
          end;
     end;
 until (lista de algum homem is vazia) or (todos
                                    estão emparelhados);
 if todos is emparelhados then
      o emparelhamento is super-estavel
 else
      não existe emparelhamento fortemente estável
```

Ao longo das paginas anteriores, vimos três variantes de implementação do algoritmo de Gale-Shapley, nomeadamente a *clássica*, a *super estável* e a *fortemente estável*, bem como a implementação deste algoritmo em linguagem c segundo o paradigma de programação procedimental. Este algoritmo apesar de parecer quiçá um pouco estranho, ou mesmo maçador, aborrecido ou até de pouca utilidade, na realidade tem utilidades bem mais amplas do que pode aparentar numa primeira abordagem!

Ao longo do tempo, apareceram diversas variantes do problema dos casamentos estáveis, e diversas variantes deste algoritmo e até outros algoritmos semelhantes para resolver problemas específicos. Dos problemas semelhantes ao problema dos casamentos estáveis (*stable marriage problem*) mais conhecidos, destacam-se o problema de atribuição (*assigment problem*), cujo objetivo é encontrar os emparelhamentos com peso máximo, num gráfico bipartido ponderado. Estas combinações não têm de ser estáveis, mas em algumas aplicações, um emparelhamento de ponderação máxima, é mais vantajoso do que um estável.

O PROBLEMA DO CASAMENTO ESTÁVEL UTILIZANDO O ALGORITMO GALE-SHAPLEY

O problema dos companheiros de quarto estáveis (*stable roommates problema*) é similar ao problema dos casamentos estáveis (*SMP*), mas difere no facto de todos os participantes pertencerem a um só conjunto dividido em números iguais. O problema dos internos/hospitais com casais, permite que um conjunto de residentes, inclua casais, que devem ser emparelhados juntos, quer para o mesmo hospital, ou para o mesmo par especifico de hospitais escolhidos pelo casal (por exemplo, um casal quer garantir que vão ficar juntos e não em internatos onde ficariam longe um do outro). A adição de casais ao problema dos residentes / hospitais, torna-o num problema NP-Completo (referente ao tempo polinomial).

Um caso mais "português", possivelmente conhecido do leitor, trata-se da solução proposta para a colocação de docentes em Portugal para o ano letivo 2004/2005, desenvolvido pela ATX para resolver o problema da colocação de professores nos concursos nacionais para a contratação e colucação de docentes nos estabelecimentos de ensino. Apesar das dificuldades no inicio do ano letivo o problema acabou resolvido. Apenas por mera curiosidade de programador, de seguida apresenta-se uma versão resumida, em pseudo-código do algoritmo utilizado.

```
Considerar inicialmente tamb‼em livres as posicoes
iniciais dos professores que pretendem mudar de
posicao.
Colocar os professores pela ordem da lista de
graduacao, na melhor preferência ainda livre de
cada professor (alguns professores podem car por
colocar).
if (professores que estavam inicialmente colocados
                            ficaram todos colocados)
{
    goto end;
}
else
{
            Os professores que estavam inicialmente
      colocados e ficaram por colocar sao colocados
   definitivamente nas suas posi↑coes iniciais, que
   deixam de estar livres. Estes professores já nao
                      entram nas próximas iteracoes.
}
Repete-se a colocacao com menos estes lugares
                                              livres.
```

Entre outras utilizações o algoritmo Gale-Shapley, na variante destinada ao problema *SMP* , está disponível como parte do pacote matchingMarkets, da linguagem R.

Este algoritmo, pode ser aplicado a diversas utilizações, em que se tenham conjuntos que necessitem de ser emparelhados, de forma a obter-se uma melhor relação entre elementos dos conjuntos. Por exemplo a atribuição de órgãos de dadores a pacientes que necessitem de transplante, salas de aula a conjuntos de alunos e professores para efeitos de vigilância de exames, etc…

Neste ultimo exemplo, apenas a titulo de curiosidade, consideremos o seguinte, cada sala, tem como propriedades os lugares, a cada sala encontra-se atribuído um docente, a este conjunto de pares (sala, doente) terá de ser emparelhado uma turma. Utilizando este algoritmo, poderia ser automatizada a atribuição de salas para efeitos de exame a turmas de forma quase automática, minimizando a quantidade de salas necessárias para o efeito.

Conclusão

O problema dos casamentos estáveis e o algoritmo de Gale-Shaply, podem ser aplicados a uma infinidade de áreas do conhecimento e situações quotidianas diversas. O seu vasto uso no dia a dia, ainda que discreto, prova a importância deste algoritmo desenvolvido em 1962.

Referencias

- The prize in Economic Sciences 2012 (Information for the public) Stable matching: Theory, evidence, and practical design
- College Admissions and the Stability of Marriage
- A I And Game Theory The Joy of Matching Paul Harrenstein, David Manlove, Michael Wooldridge
- Emparelhamentos, Casamentos Estáveis e Algoritmos de Colocação de Professores

AUTOR

Escrito por António C. Santos

Programar é criar um novo mundo escrevendo código, cumprindo os mais elementares desígnios da vida, "aprender, ensinar, criar, partilhar, melhorar e seguir". Formou-se no Instituto Politécnico de Viana do Castelo. Membro da Comunidade Portugal-a-Programar desde Agosto de 2007, é também membro da Sahana Software Foundation, onde é Programador Voluntário desde 2012, actualmente exerce funções de mentor voluntário na plataforma MOOC Coursera . Twitter:@apocsantos

A maior concentração de criadores no interior de Portugal

Castelo Branco Mini Maker Faire®

10 junho 2017
Castelo Branco

Inscrições abertas para criadores

inscrições: goo.gl/I9paJL

Robótica Mecânica Eletrónica Artesanato Arte Urbana

Ciência Música Educação DIY E muito mais

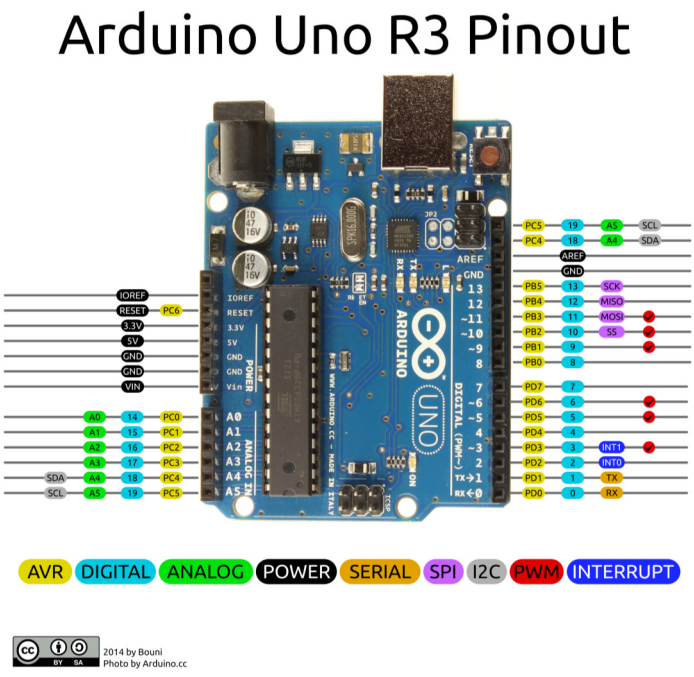
ELECTRÓNICA

O Problema da falta de GPIO Pins

O PROBLEMA DA FALTA DE GPIO PINS

Introdução

Numa esmagadora maioria dos circuitos usados em IoT e em automação de uma forma geral, como o caso do Arduino/Genuino, existem algumas limitações em termos de pinos analógicos, que nos podem complicar a tarefa de ligar sensores. Por exemplo, no Arduino/Genuino Uno [IMG.1], apenas são disponibilizados 6 pinos analógicos que vão de A0 a A5, respetivamente.



Esta limitação torna-se particularmente evidente quando se pretende ligar mais do que 6 sensores analógicos, ficando rapidamente sem pinos disponíveis para ligar os sensores. Se uma solução poderia passar por ligar dois AVR's a comunicarem entre si por UART, outra mais simples, mais eficiente e mais barata, será utilizar um multiplexador analógico/digital, para assim "multiplicar" a quantidade de pinos disponíveis.

O mesmo princípio é válido para pinos digitais.

Como funciona?

Um multiplexador, funciona definindo o fluxo de corrente, entre um determinado ponto e um outro determinado ponto, no caso entre pinos do circuito multiplexador que esteja a ser utilizado. Para tal são utilizados pinos para controlar o circuito, e uma tabela de verdades que permite selecionar o canal por onde a eletricidade deve fluir. [TAB.1]

Para o efeito de seleção da porta do multiplexador a ser utilizada, podemos calcular o valor binário da mesma recorrendo à tabela de verdades do circuito. No circuito usado para exemplo, um CD74HC4067 obtemos 16 pinos usando 4 pinos digitais para controlo e um pino analógico para receber o "sinal" neste caso um valor analógico. Com base na tabela, podemos por exemplo calcular o endereço do pino 10 do nosso multiplexador. Para tal consultamos a tabela e verificamos que para o pino numero 10, o valor em S0 será zero o valor em S1 será 1, o valor em S2 será zero e o valor em S3 será um, e o valor de controlo E será zero ou seja 01010, para tal colocamos a low a corrente no pino S0 a 1 no S1 a zero no S2 a 1 no S3 e a zero e E, obtendo o valor binário 01010 que convertido a decimal seria 10.

TRUTH TABLE

S0	S1	S2	S3	$\overline{E}$	SELECTED CHANNEL
X	X	X	X	1	None
0	0	0	0	0	0
1	0	0	0	0	1
0	1	0	0	0	2
1	1	0	0	0	3
0	0	1	0	0	4
1	0	1	0	0	5
0	1	1	0	0	6
1	1	1	0	0	7
0	0	0	1	0	8
1	0	0	1	0	9
0	1	0	1	0	10
1	1	0	1	0	11
0	0	1	1	0	12
1	0	1	1	0	13
0	1	1	1	0	14
1	1	1	1	0	15

H= High Level
L= Low Level
X= Don't Care

[TAB.1]

Qualquer conversão podia ter sido exemplo a escolha foi sobre o número 10 apenas por mero acaso! De qualquer das formas a tabela ajuda imenso na hora de fazer esta conversão e independentemente do multiplexador escolhido as tabelas por regra, constam na folha de dados do circuito.

Apenas para uma referência de consulta rápida, vamos colocar os valores para cada uma das portas devidamente convertidos abaixo, sendo que para o efeito vamos considerar o valor do pino de controlo como zero para todos os casos.

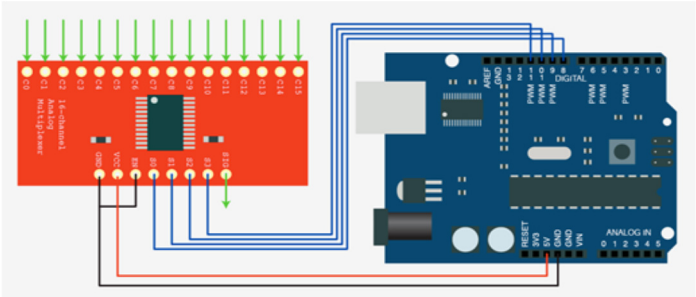
Valor Binário	Decimal correspondente
00000	0
10000	1
01000	2
11000	3
00100	4
10100	5
01100	6
11100	7
00010	8
10010	9
01010	10
11010	11
00110	12
10110	13
01110	14
11110	15

Com estes dados podemos então prosseguir para o exemplo concreto de utilização de um multiplexador, "transformar" 1 pino de dados em 16 pinos permitindo assim utilizar mais sensores ou circuitos sem ocupar tantas portas no circuito base.

O circuito (pinos analógicos)

Tal como referido anteriormente para este artigo é usado um arduino/genuino uno ligado a um multiplexador CD74HC4067 de 16 portas.

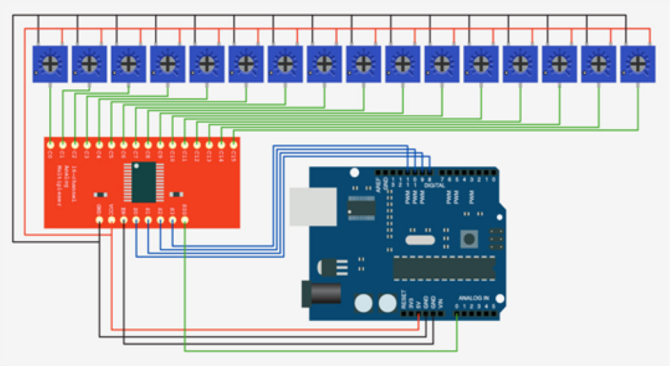
Para este exemplo em concreto os pinos de controlo S0 a S3 do multiplexador são ligados aos pinos digitais, 8, 9, 10 e 11 do arduino, os pinos gnd ligados respetivamente, e o pino vcc do multiplexador é ligado ao pino +5vdc do arduino, fornecendo assim os 5v dc necessários para o multiplexador funcionar e o pino analógico zero do arduino ao SIG do multiplexador para comunicar dados. De notar que o pino de controlo EN do multiplexador permanecerá ligado ao GND em paralelo. [DIAG.1]



[DIAG.1]

Com estas ligações feitas, bastará ligar ao multiplexador os circuitos, componentes ou sensores a que se pretenda aceder, nas portas do multiplexador de C0 a C15, respetivamente. Para não complicar demasiado o exemplo iremos usar 16 potenciómetros, cada um ligado a um dos pinos do multiplexador, e ler cada um de forma independente.

Para tal vamos fazer as ligações da seguinte forma: Ligamos cada um dos potenciómetros ao pino de 5vdc do arduino e o gnd de cada potenciómetro de igual forma ao gnd do arduino. Com isto temos estabelecido um fluxo de corrente entre o arduino e cada potenciómetro. Agora falta ligar os pinos de saída de cada potenciómetro ao multiplexador. Para tal, ligamos cada um ao respetivo pino do multiplexador, entre o C0 para o primeiro dos potenciómetros e o C15 para o ultimo. [DIAG.2]



[DIAG.2]

Programar

O objetivo é fazer a leitura de cada potenciómetro, separadamente. Para tal o circuito deve primeiro colocar os valores corretos em cada pino de acordo com a tabela, antes de efetuar a leitura. Existem diversas formas de abordar este problema em programação, no entanto no exemplo que se segue, a tabela será armazenada numa matriz, para facilitar o trabalho. Uma vez que para cada leitura é preciso primeiramente colocar os valores corretos, em vez de colocar todo o código no "main loop" do programa, utilizamos uma função que recebe um valor inteiro, correspondente ao número do pino que se pretende utilizar, e após ter colocado os valores correspondentes em cada um dos pinos de controlo, efetua a leitura, devolvendo o valore resultante. Além disso esta abordagem acaba sendo útil caso se pretendam usar por exemplo protothreads.

```
//Pinos de controlo do multiplexer
int s0 = 8;
int s1 = 9;
int s2 = 10;
int s3 = 11;

//Pino de comunicação SIG (Signal)
int SIG_pin = 0;
//funcao setup
void setup()
{
      pinMode(s0, OUTPUT);
      pinMode(s1, OUTPUT);
      pinMode(s2, OUTPUT);
      pinMode(s3, OUTPUT);
```

```
      digitalWrite(s0, LOW);
      digitalWrite(s1, LOW);
      digitalWrite(s2, LOW);
      digitalWrite(s3, LOW);

      Serial.begin(9600);
}

//main loop (loop principal do programa)
void loop()
{
//percorre o ciclo e lê cada um dos 16 valores
      for (int i = 0; i < 16; i++)
        {
            Serial.print("Valor na porta ");
            Serial.print(i);
            Serial.print("é : ");
               Serial.println(readMux(i));
               delay(1000);
        }
}
int readMux(int canal)
{
      int controlPin[] = { s0, s1, s2, s3 };
      int muxChannel[16][4] =
        {
            { 0,0,0,0 }, //porta 0
            { 1,0,0,0 }, //porta 1
            { 0,1,0,0 }, //porta 2
            { 1,1,0,0 }, //porta 3
            { 0,0,1,0 }, //porta 4
            { 1,0,1,0 }, //porta 5
            { 0,1,1,0 }, //porta 6
            { 1,1,1,0 }, //porta 7
            { 0,0,0,1 }, //porta 8
            { 1,0,0,1 }, //porta 9
            { 0,1,0,1 }, //porta 10
            { 1,1,0,1 }, //porta 11
            { 0,0,1,1 }, //porta 12
            { 1,0,1,1 }, //porta 13
            { 0,1,1,1 }, //porta 14
            { 1,1,1,1 }  //porta 15
        };
        //percorre os 4 pinos
        for (int i = 0; i < 4; i++)
        {
            digitalWrite(controlPin[i],
                         muxChannel[canal][i]);
        }
        //lê o valor do pino selecionado
        int val = analogRead(SIG_pin);
        //retorna o valor lido
        return val;
}
```

No exemplo acima, o objetivo era ler 16 valores analógicos, provenientes de 16 potenciómetros. No entanto isto é apenas uma utilização para um multiplexador. Por exemplo, poderia ser usado para controlar 16 relés diferentes, mas não só. Pode ser utilizado para por exemplo com dispositivos que comuniquem por uart, etc.

Desmultiplexação

Para não complicar e exemplificar o mesmo efeito, mas desta feita com pinos digitais, pode ser feito com 16 leds.

Em termos de ligações o circuito será em tudo idêntico com algumas diferenças chave. Os pinos de controlo, poderão ser os mesmos, no entanto o pino de dados (SIG) será ligado no pino de 5vdc do arduino e os leds todos eles ligados ao gnd do arduino. Assim em vez de lermos dados, regulamos apenas o envio de corrente entre os 5vdc do arduino e cada uma das portas do multiplexador, acendendo apenas um led de cada vez.

Para exemplificar uma forma diferente de utilizar a tabela de verdades no código desta vez não é usada uma função separada, ficando todo o código no loop principal (main loop).

```
//arduino code
byte controlPins[] =
{
B00000000,
B10000000,
B01000000,
B11000000,
B00100000,
B10100000,
B01100000,
B11100000,
B00010000,
B10010000,
B11010000,
B00110000,
B10110000,
B01110000,
B11110000
};
//setup
void setup()
{
      DDRD = B11111111;
}
//define o pino de output
void setPin(int outputPin)
{
      PORTD = controlPins[outputPin];
}
//main loop
void loop()
{
      for (int i = 0; i < 16; i++)
      { setPin(i);
        delay(250);
      }
}
```

"Numa esmagadora maioria dos circuitos usados em IoT (...) existem algumas limitações em termos de pinos analógicos, que nos podem complicar tarefas (...)"

Conclusão

Ao longo do artigo, mais do que apresentar um circuito apresentaram-se duas formas de executar tarefas idênticas com código diferente, bem como a multiplexação e desmultiplexação de portas recorrendo a um multiplexador simples e de baixo custo. A vantagem de expandir substancialmente o número de portas disponíveis em cada situação ajuda particularmente em circuitos com poucas entradas e saídas, como o caso dos ATMega328, mas também dos ATTiny 45 e dos ATTiny 85 que só dispõem de 3 portas analógicas e 2 portas digitais, ultrapassando assim algumas das limitações impostas pelos controladores.

Fontes

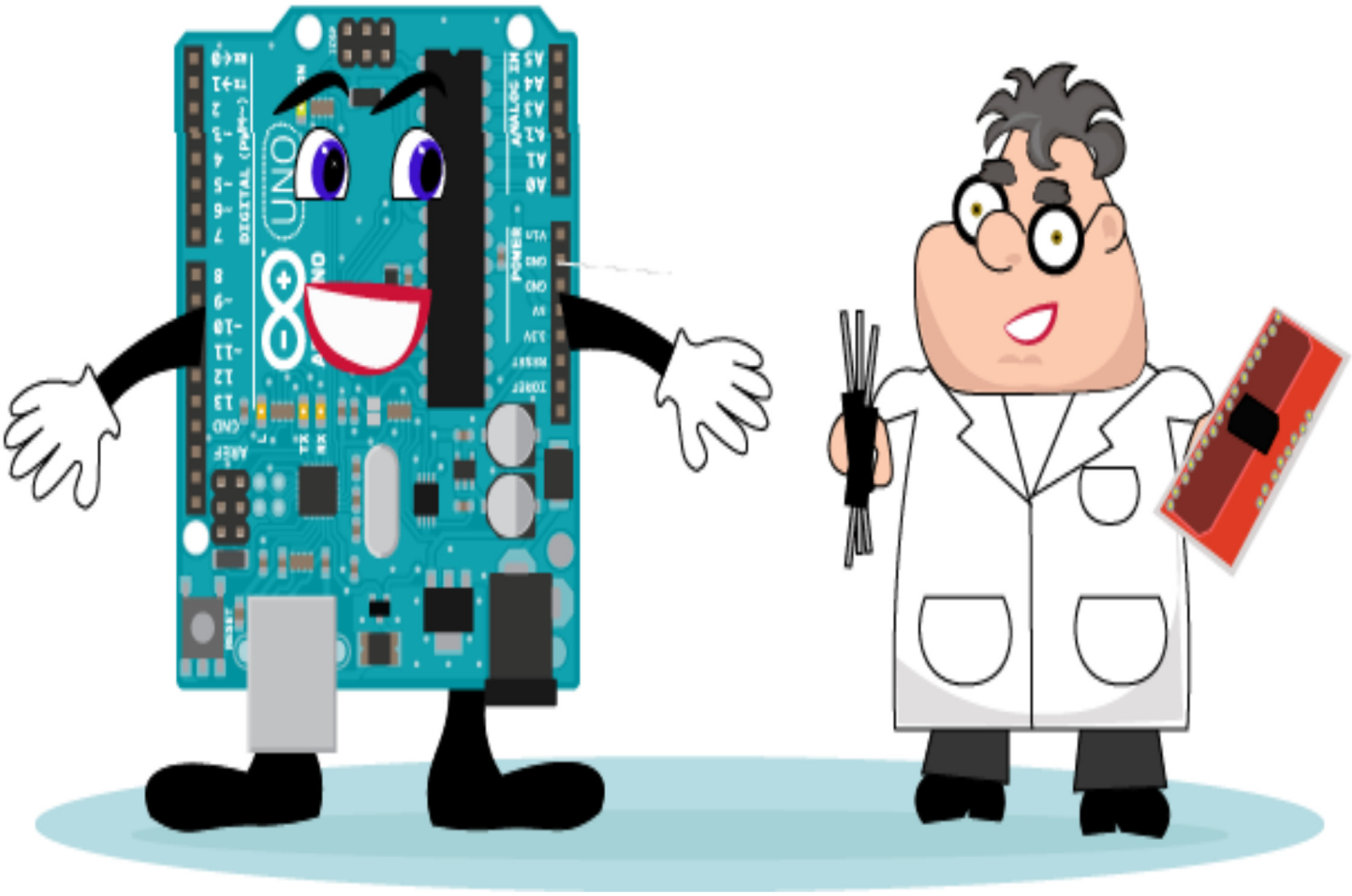
SparkFun https://www.sparkfun.com/products/9056

ARM mbed Developer site https://developer.mbed.org/

Dan Zilinskas Motion Capture for Runners

Texas Instruments CD74HC4067 DataSheet https://www.sparkfun.com/datasheets/IC/CD74HC4067.pdf

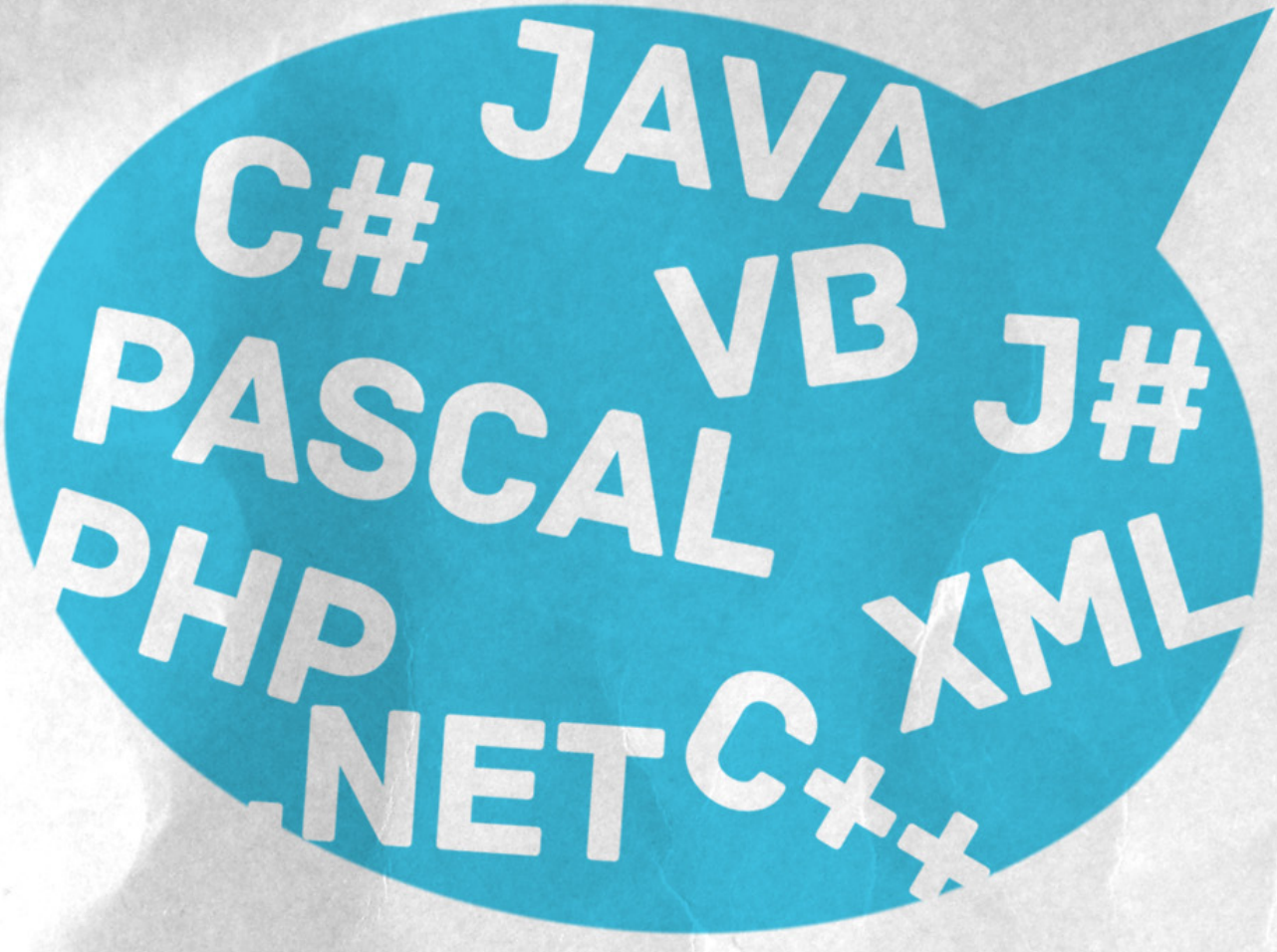
Frontiernerds http://www.frontiernerds.com/



AUTOR

Escrito por António C. Santos

Programar é criar um novo mundo escrevendo código, cumprindo os mais elementares desígnios da vida, "aprender, ensinar, criar, partilhar, melhorar e seguir". Formou-se no Instituto Politécnico de Viana do Castelo. Membro da Comunidade Portugal-a-Programar desde Agosto de 2007, é também membro da Sahana Software Foundation, onde é Programador Voluntário desde 2012, actualmente exerce funções de mentor voluntário na plataforma MOOC Coursera . Twitter:@apocsantos



ENTÃO, SÓ FALAS EM CÓDIGO?

TEMOS O REMÉDIO CERTO PARA TI!

portugal-a-programar.pt

A MAIOR COMUNIDADE PORTUGUESA DE PROGRAMAÇÃO, APARECE!

COLUNAS

C# - Padrão de Arquitetura SOLID

SQL Curtas - SQL Curtas #2: Dúvidas Comuns

Kernel Panic -

Padrão de Arquitetura SOLID

Introdução

Existem diversas orientações para programação orientada por objectos, no entanto, neste artigo apenas iremos focar SOLID com exemplos em C#.

SOLID é um acrónimo dos cinco primeiros princípios da programação orientada a objetos e design de código identificados por Robert C. Martin Este mesmo acrónimo foi introduzido por Michael Feathers, após observar que os cinco princípios poderiam se encaixar nesta palavra.

O que significa S.O.L.I.D. ?

- S – Princípio de Responsabilidade Única
- O – Princípio Open Close
- L – Princípio de Substituição Liskov
- I - Princípio de Segregação de Interface
- D – Princípio de Inversão de Dependência

Que princípios ?

Princípio de Responsabilidade Única (SRP)

Diz que cada classe deverá ter responsabilidade única. Uma classe não deverá ter mais que uma razão para mudar.

Exemplo: Vamos supor que criámos uma classe, XmlValidator para validação XML a qual tem a responsabilidade de validar o XML. Se existe necessidade de atualizar o XML, então deverá ser criada uma classe separada para o mesmo. A classe XmlValidator não deverá ser usada para atualizar o XML.

Vejamos como atualizar: Para isso teremos de criar uma nova classe.

```
public class XmlValidator
{
    public void Validate()
    {
        //código
    }
}

public class XmlUpdate
{
    public void DoUpdate()
    {
        //código
    }
}
```

Princípio Open Close (OCP)

Deverá ser aberta para extensão, mas fechada para modificação. Isto significa que se escrevemos uma classe e está a funcionar agora e bem, vem um novo requisito, então não devemos modificar a classe, em vez disso uma nova classe deve ser criada, a qual irá estender a classe anterior.

Vamos supor que temos uma classe chamada Cliente que tem uma propriedade FacturaNumero a qual tem um tipo inteiro.

```
public class Cliente
{
    public int FacturaNumero
    {
        get;
        set;
    }
}
```

Futuramente o requisito mudará, agora FacturaNumero deverá ser alfanumérico em vez de apenas numero inteiro. Assim neste caso, devemos criar uma subclasse ClienteNew com a mesma propriedade mas diferente datatype em vez de modificar a anterior.

```
public class ClienteNew : Cliente
{
    public new String FacturaNumero
    {
        get;
        set;
    }
}
```

Princípio de Substituição Liskov (PSL)

Um objeto parent deverá ser capaz de substituir o seu objeto filho durante o polimorfismo em runtime.

Por exemplo, suponhamos que temos duas classes, Cooler e Fan, ambas são heradadas de um interface comum chamado ISwitch o qual tem três métodos, On, Off e Regulate.

```
public class Fan : ISwitch, ILight
{
    public void On()
    {
    }
    public void Off()
    {
    }
    public void Regulate()
    {
    }
}
public class Bulb : ISwitch
{
    public void On()
    {
    }
    public void Off()
    {
    }
}
```

Cada classe irá usar um interface, o que é requerido. Não existe necessidade de usar um interface que não esteja requerido. Fan é necessária nos três métodos por isso pode usar ambos os interfaces, ISwitch e ILight e Build precisa apenas de um dos dois métodos para implementar On e Off assim precisa de implementar apenas um interface chamado ISwitch.

Princípio de Segregação de Interface (ISP)

Os interfaces específicos de cliente são melhores que um interface de objetivo geral.

Suponhamos que temos um interface para clique:

```
public interface IClick
{
    void onClick(Object obj);
}
```

Com o passar do tempo, solicitam a adição de uma nova função onLongClick. Isto implica adicionar um novo método ao interface:

```
public interface IClick
{
    void onClick(Object obj);
    void onLongClick(Object obj);
}
```

Mais algum tempo passado e eis que chega um novo requerimento para adicionar a função toque e adicionamos o método no mesmo interface.

```
public interface IClick
{
    void onClick(Object obj);
    void onLongClick(Object obj);
    void onTouch(Object obj);
}
```

Neste ponto, decidimos alterar também o nome do interface porque toque é diferente de clique. Desta forma, este interface torna-se um problema, genérico e poluído.

Suponhamos que alguns clientes necessitam apenas da função onClick e outro precisam apenas da função onTouch, então uma será inútil para ambos. O ISP disponibiliza a solução! Divide-se o interface em dois interfaces, ITouch e IClick. O cliente que solicitou onClick pode implementar IClick, o que precisa do onTouch pode implementar ITouch e o que precisa de ambos pode implementar ambos.

```
public interface IClick
{
    void onClick(Object obj);
    void onLongClick(Object obj);
}

public interface ITouch
{
    void onClick(Object obj);
    void onTouch(Object obj);
}
```

Diferença Entre Segregação de Interface e Substituição Liskov

Ambos os princípios parecem o mesmo mas existe uma ligeira diferença, o ISP é mais específico que o LSP. No ISP, até existe método e também a sua definição mas não é relevante para uma classe em particular por isso, de acordo com o ISP, precisamos de separar este método num interface separado. Em LSP, o método existe mas não a sua definição por isso não faz sentido mantê-lo no mesmo interface por isso de acordo com o LSP, precisamos de separa em interfaces diferentes.

Princípio de Inversão de Dependência (ISP)

Este declara dois pontos, o primeiro ponto é que um módulo de nível mais elevado não deverá depender de um módulo de nível baixo. Ambos devem depender da abstração. E o segundo ponto é, Abstração não deve depender do detalhe. O detalhe deve depender da abstração. Por outras palavras, nenhum objeto deve ser criando dentro de uma classe, eles devem ser passados ou injetados a partir de fora. E onde irá receber, será um interface em vez de uma classe.

```
class PasseadorDeLivros
{
    ILogWriter writer = null;

    Public PasseadorDeLivros(ILogWriter writer)
    {
        this.writer = writer;
    }

    public void Notify(string message)
    {
        writer.Write(message);
    }
}

class LogWriter
{
    public void Write(string message)
    {
        //Write it into a file
    }
}

interface ILogWriter
{
    void Write(string message);
}
```

Agora para criar uma nova mensagem utilizar-se-ia o código da seguinte forma:

```
class Main
{
    Private void AddPasseadorDeLivros
    {
        ILogWriter writer = null;

        write = new LogWrite();
    PasseadorDeLivros obj = new PasseadorDeLivros
                                        (write);
    write.Notify(""message");
   }
}
```

No caso de enviar um email em vez do logwrite, podemos introduzir uma nova classe chamada MailWrite e herdar a classe com o mesmo interface ILogwrite. E as alterações serão assim:

```
class Main
{
    Private void AddPasseadorDeLivros
    {
        ILogWriter writer = null;

        write = new MailWrite();
    PasseadorDeLivros obj = new PasseadorDeLivros
                                        (write);
    write.Notify(""message");
   }
}
```

Conclusão

Ao longo do artigo fomos explorando os princípios de SOLID (single responsibility, open-closed, Liskov substitution, interface segregation and dependency inversion), acompanhados com exemplos em C#, sem detalhar nenhum dos princípios, pois isso claramente sairia dos objetivos deste artigo, uma vez que se pretendia apenas uma apresentação simples e sucinta, de SOLID em C#. Como sempre, não defendo estas regras como as melhores do mundo, mas apenas como um bom conjunto, deixando ao cargo do leitor decidir sempre o que utilizar de acordo com as suas necessidades!

AUTOR

Escrito por António C. Santos

Programar é criar um novo mundo escrevendo código, cumprindo os mais elementares desígnios da vida, "aprender, ensinar, criar, partilhar, melhorar e seguir". Formou-se no Instituto Politécnico de Viana do Castelo. Membro da Comunidade Portugal-a-Programar desde Agosto de 2007, é também membro da Sahana Software Foundation, onde é Programador Voluntário desde 2012, actualmente exerce funções de mentor voluntário na plataforma MOOC Coursera . Twitter:@apocsantos

SQL Curtas #2: Dúvidas Comuns

Como mostrar registos por ordem aleatória?

Não existe uma forma "standard" de resolver este problema. Cada SGBD (Sistema de Gestão de Bases de Dados) tem uma forma diferente:

```
CREATE TABLE ProgRand (A INT);
INSERT INTO  ProgRand (A) VALUES (9), (8),
          (7), (6), (5), (4), (3), (2), (1);
-- Ordem natural (armazenada)
SELECT * FROM ProgRand;
-- Ordem pelo campo A ascendente
SELECT * FROM ProgRand ORDER BY A ASC;
-- Ordem aleatória - SQL Server
SELECT * FROM ProgRand ORDER BY NewID();
-- Ordem aleatória com apenas 3 registos -
                                    SQL Server
SELECT TOP (3) * FROM ProgRand ORDER BY NewID
();
-- Ordem aleatória - Oracle
SELECT * FROM ProgRand ORDER BY
                            DBMS_RANDOM.VALUE;
-- Ordem aleatória - MariaDB / MySQL
SELECT * FROM ProgRand ORDER BY RAND();
```

O que é uma TRANSACTION?

Uma transacção é um bloco de comandos que supostamente devem ser executados "ou todos ou nenhum". Exemplo:

```
CREATE TABLE ProgTran (A INT);
      BEGIN TRANSACTION;
      INSERT INTO  ProgTran (A) VALUES (1), (2),
                                          (3);
      SELECT * FROM ProgTran;
      -- Executar outras alterações aqui...
      -- Executar aqui alguma verificação (nota:
                  corre apenas ROLLBACK ou COMMIT)
      -- Se erro:
      ROLLBACK;
      -- Caso contrário tudo ok!
      COMMIT;
      SELECT * FROM ProgTran;
```

Aqui, se a validação efectuada (erros, questões de negócio, etc.) der erro, podemos fazer ROLLBACK e os dados inseridos "provisoriamente" são deitados fora. Apenas fazendo COMMIT serão gravados.

Outro motivo para haver transacções é isolar os pedidos de dois ou mais utilizadores. Se dois utilizadores executarem o código em cima, os registos novos de "outros" utilizadores (ainda não gravados/COMMITted), não aparecerão num SELECT à tabela.

Coloca-se no entanto uma questão: enquanto o ROLLBACK é necessário para cancelar alterações, será que o COMMIT é necessário para garantir que o resultado é efectivamente escrito? A resposta depende do SGBD (definições de auto-commit, etc.). Em qualquer caso, é sempre recomendado incluir explicitamente um COMMIT no final.

Compatibilidade? Consoante o SGBD, poderão haver alterações nos comandos:

- "BEGIN" e/ou "START";
- "TRAN", "TRANS" e/ou "TRANSACTION";
- "COMMIT" e "ROLLBACK" com/sem "TRANSACTION" à frente.

O que é um DEADLOCK?

Um deadlock pode acontecer em bases de dados ou qualquer recurso informático. Para explicar o que é um deadlock, fica um exemplo:

1. O utilizador A começa uma transacção e escreve na tabela X. A tabela X fica bloqueada enquanto não houver COMMIT ou ROLLBACK;
2. O utilizador B começa uma transacção e escreve na tabela Y. A tabela Y fica bloqueada enquanto não houver COMMIT ou ROLLBACK;
3. O utilizador A tenta agora escrever (eventualmente apenas ler) da tabela Y, mas esta está bloqueada pelo utilizador B. Neste caso, fica a aguardar o COMMIT ou ROLLBACK do utilizador B;
4. O utilizador B tenta agora escrever (eventualmente apenas ler) da tabela X, mas esta está bloqueada pelo utilizador A. Neste caso, fica a aguardar o COMMIT ou ROLLBACK do utilizador A;
5. Ambos os utilizadores estão parados à espera do outro, e nenhum conseguirá avançar.

Isto pode acontecer com mais do que dois utilizadores encadeados (e.g. A→B→C→A).

Os SGBD mais recentes têm protecções contra deadlocks. Assim, um destes utilizadores aleatoriamente vai ser considerado "deadlock victim" e a sua transacção vai ter um ROLLBACK automático (e o utilizador recebe um erro). O outro então, não tendo bloqueios alheios, pode completar a sua tarefa.

O comportamento (locks, limitações, monitorização, escolha de vítima, etc.) depende sempre do SGBD.

ALTER TABLE ou DROP TABLE + CREATE TABLE?

Antes de mais, esta pergunta não se aplica apenas a tabelas, mas também a views, stored procedures, bases de dados, logins/utilizadores e muitos outros objectos.

- ALTER altera um objecto já existente;
- DROP remove um objecto já existente (se for uma tabela, inclui remover todos os repectivos dados);

- CREATE cria um novo objecto.

Voltando à pergunta inicial, embora ambas as formas consigam o mesmo resultado relativamente à estrutura base da tabela, apenas o ALTER TABLE mantém as permissões, os índices e constraints previamente definidas (e não expressamente mencionadas no CREATE TABLE), assim como os dados existentes.

Consoante existam outros objectos dependentes do objecto a modificar, poderá não ser possível fazer DROP (note-se que o ALTER poderá também estar limitado).

Se estivermos a falar de logins/utilizadores, remover e recriar um utilizador apaga todas as permissões já definidas em objectos, a pertença a grupos/roles, etc.

Deve, portanto, ser escolhido o ALTER TABLE (ou VIEW, etc.) sempre que possível.

Quais as diferenças entre DELETE, TRUNCATE e DROP?

Nada como uma tabela para esquematizar:

Comando	DELETE	TRUNCATE	DROP
Tipo	DML	DDL	DDL
Registos apagados	Tudo ou com WHERE	Tudo ou por partições	Tudo
Estrutura	Mantém	Mantém	Remove
Logging / Transacções / Rollback	Sim	Depende do SGBD	Não
Foreign Keys	Respeitadas	Depende do SGBD	Depende do SGBD
Triggers	Executam	Ignorados	Ignorados
Permissões	DELETE	Depende do SGBD (tipicamente ALTER TABLE)	ALTER TABLE

- DELETE apaga um ou mais registos (eventualmente todos) de acordo com a respectiva condição WHERE (ou ausência desta). Note-se que o "FROM" é opcional em muitos SGBD mas recomendado por compatibilidade:

```
DELETE FROM TabelaX;
      DELETE FROM TabelaX WHERE CampoY = Z;
```

- TRUNCATE apaga todos os registos na tabela (ou apenas em determinadas partições, dependendo do SGBD):

```
TRUNCATE TABLE TabelaX;
```

- DROP apaga (destroi) a tabela, incluindo todos os registos:

```
DROP TABLE TabelaX;
```

Se o objectivo é apagar todos os registos duma tabela (mantendo a tabela), deve ser utilizado TRUNCATE e não DELETE. Isto porque o DELETE analisa os registos um a um antes de apagar (e teoricamente, remove-os dos índices da tabela também um a um), enquanto o TRUNCATE remove numa única operação todos os registos, primeiro da tabela e depois de todos os índices.

Quando a tabela tem uma coluna de identidade (autonumber), outra diferença é que o DELETE mantém o número de sequência actual, enquanto o TRUNCATE reverte o número para o valor definido incialmente (tipicamente 1). Atenção que este comportamento não acontece em todos os SGBD.

Convém também desmistificar que um TRUNCATE não é necessariamente igual a um DROP TABLE + CREATE TABLE, já que no primeiro todas as permissões e objectos dependentes se mantém inalterados, e que em muitos SGBD o TRUNCATE é logged e revertível com um ROLLBACK.

AUTOR

Escrito por André Melancia

Independent Developer/DBA/Consultant. Microsoft Certified Trainer (MCT) focusing on SQL Server, Azure and IoT. 17+ years' fun developing information and multimedia systems, DBA, project and IT management. PowerShell Portugal, IT Pro Portugal and IoT Portugal communities organiser. IPv6 Portugal, DNSSec Portugal and Windows Development Portugal online communities moderator. Actively volunteering, organising, speaking or just participating at community meetings and events like SQLSaturdays, SQLBits, SQLRelay, Maker Faire Lisbon, Arduino/Genuino Day Lisbon, Global Azure Bootcamp Lisbon, etc. Proud uncle and food devouring expert, with dangerous pussy cat as companion.

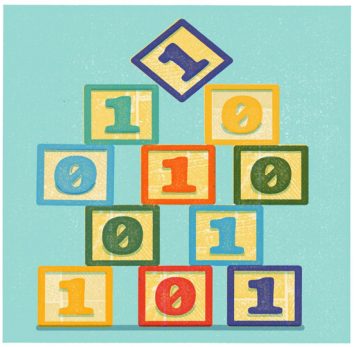
Go to http://Andy.PT and you'll know the same as the NSA...

A arte, engenho, e muita diversão

A Arte

Programar pode ser uma arte, apesar de ser uma acção e não necessariamente uma “expressão” no sentido mais conservador. Nesse caso um programa seria “uma forma de arte” e consequentemente os developers seriam artistas.

Ainda assim, isto nem sempre é observado desta forma! Numa atitude quase que “patológica” ou “desenquadrada” programar é muitas vezes visto como um ofício, uma tarefa, um trabalho e não propriamente uma forma de arte! Por exemplo e sem divagar muito, ensinam-se artes plásticas, expressão dramática, música, etc… nas escolas, no entanto ainda não existe de forma “massificada” a programação como matéria de ensino e estudo! Ainda que pareça precoce ver os mais novos a aprender a programar, certo será admitir que hoje em dia quase todos sabem usar um tablet, ou mesmo um computador!



Logo porque não programar ? Fica a questão!

Ainda que se diga que programar não é uma arte, por este ou aquele motivo, vejamos, programar implica imaginar, criar, construir, transpor o pensamento “abstracto” para algo “real”. Logo em teoria seria uma arte!

Um aprendiz de guitarrista, seguindo a partitura, ou a tablatura com os acordes, consegue reproduzir a música, da mesma forma que um “aprendiz” de programador com um pouco de paciência consegue seguir o código de outro programador e reproduzir o mesmo resultado. Logo em teoria seria uma forma de arte! Ainda que esta afirmação possa ser contestada, não deixa de ter lógica! Assim podemos passar para o engenho!

O Engenho

Se tivermos em conta que programar, tal como montar legos, jogar jogos como o “jogo do burro”, também conhecido por Jenga, implica algum “engenho”, no sentido mais alargado da definição, programar acaba aguçando o engenho! Para os mais “nostálgicos” para não usar outra palavra, que se recordam do tempo dos computadores de 8 bits, como o ZX Spectrum e o Timex TC2048 / TC2068 (estes últimos feitos em Portugal), certamente se lembrará do engenho necessário para carregar um jogo, ou interromper o carregamento, para se fazer aquele “poke” fantástico que apesar de “pouco nobre” já que o objetivo seria ter vidas infinitas no jogo, merecia a nobreza da ingenuidade de o executar. Os que se lembram disto, nessa altura certamente eram crianças, ou adolescentes. Não lhes faltava engenho para inclusive escreverem os seus próprios jogos, em Basic, que era interpretados pelo interpretador do computador e executados, para grande diversão!

Programar exige engenho, aguça o engenho, é uma boa forma de despertar o engenho, latente em toda a gente, para criar coisas novas!

A diversão

Ora aqui chegamos ao final! A diversão! Bem, programar, pode ser considerado tudo menos divertido! Depende do ponto de vista de quem o diz! Na realidade, programar, além de ser uma arte, um “engenho”, é também uma forma de diversão! Primeiramente porque criar é algo inato no ser humano, que tende a criar coisas novas apenas porque se diverte a fazê-lo! Desde criar novos jogos, a criar novos objectos, etc… Logo tudo o que é feito com gosto, acaba sendo divertido! Quando se começa a programar algo, apenas por “desporto”, ou por satisfação pessoal, ou pelo simples desafio, começa um “novo processo” de criação e diversão, entre a tentativa o erro, o estudo e a conclusão! Não é preciso ser algo profissional, nem algo feito por “profissionais” nem tão pouco algo feito por adultos! Para uma criança, programar pode ser algo bastante divertido! Deixo o desafio ao leitor de desenvolver um “pseudo-jogo” até mesmo em scratch e enviar um printscreen do mesmo, para vir a ser incluído num futuro kernel panic!

Aceita o desafio ?

AUTOR

Escrito por António C. Santos

Programar é criar um novo mundo escrevendo código, cumprindo os mais elementares desígnios da vida, “aprender, ensinar, criar, partilhar, melhorar e seguir”. Formou-se no Instituto Politécnico de Viana do Castelo. Membro da Comunidade Portugal-a-Programar desde Agosto de 2007, é também membro da Sahana Software Foundation, onde é Programador Voluntário desde 2012, actualmente exerce funções de mentor voluntário na plataforma MOOC Coursera . Twitter:@apocsantos

Media Partners da Revista PROGRAMAR

Análises

C# 6 - PROGRAMAÇÃO COM PRODUTIVIDADE

Introdução à Programação com Python, Algoritmos e logica de programação para iniciantes

C# 6 - PROGRAMAÇÃO COM PRODUTIVIDADE

Título: C# 6 - PROGRAMAÇÃO COM PRODUTIVIDADE

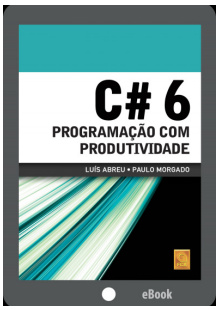
Autores:

Editora: FCA - Editora de Informática

Páginas: 103

ISBN: 978-972-722-835-5

Formato: eBook



IIntrodução

Com o passar dos anos, cada vez mais os programadores consideram o C# como sendo a linguagem da plataforma .Net. A sua evolução tem sido constante e tem contribuído para a simplificação e redução do trabalho com a escrita de código em .Net. As novidades desta nova versão não são tão impressionantes como a do aparecimento do Linq, por exemplo, mas não deixam de ser úteis para aumentar a eficiência do programador.

Este livro que revemos nesta edição é constituído por cinco capítulos e explica as principais novidades introduzidas no C# 6.

Conteúdos

O livro começa por apresentar 3 grandes funcionalidades introduzidas nesta versão do C# nomeadamente as expression bodied members, a directiva using static e por fim os operadores null-conditional. Com as expression bodied members torna-se possível a implementação de propriedades somente de leitura e métodos a partir de expressões lambda. A diretiva using static permite-nos aceder aos membros estáticos públicos de um tipo a partir do contexto actual sem que o nome desses membros tenha que ser precedido pelo nome do tipo que os define. Por fim e uma das funcionalidades preferidas dos programadores são as null-conditional em que o programador vai entender como poderá validar se determinadas expressões têm o valor null sem recorrer das condições de decisão habituais (if's).

O capítulo 2, o autor descreve a inicialização de propriedades implementadas automaticamente e a nova forma de inicialização de dicionários.

O capítulo 3, o leitor irá compreender como podemos remover grande parte das ocorrências das chamadas string "mágicas" do nosso código recorrendo ao novo operador criado: o nameof. Além disso irá conhecer o conceito de strings interpoladas substituindo o uso do nosso conhecido string.Format, tornando-se mais simples a forma como compomos strings nos nossos programas.

O capítulo 4 apresenta melhorias desenvolvidas no uso das excepções, nomeadamente a possibilidade de usar a expressão await no interior de blocos de catch e finally. Além disso, nesta nova versão foi introduzida uma nova funcionalidade, os exception filters. Por fim, o capítulo 5 dá ênfase ao Roslyn (oficialmente designado por .Net Compiler Platform), que trata de "transformar" o compilador numa API que pode ser consumida pelos nossos programas. Com a introdução desta plataforma o programador, se necessitar, terá abertura de analisar e gerar código.

> ***"Com o passar dos anos, cada vez mais os programadores consideram o C# como sendo a linguagem da plataforma .Net."***

Conclusão

Um livro de leitura agradável, bastante sucinto, baseando-se em muitos exemplos práticos para cada funcionalidade. Este livro adequa-se a programadores que já têm conhecimentos sólidos de C# de versões anteriores.

AUTOR

Escrito por Mónica Rodrigues

Licenciada em Engenharia Informática e de computadores pelo ISEL. Software engineer com vasta experiência em desenvolvimento web nas mais variadas tecnologias, desde HTML5, AngularJs, Asp.Net Web API, Asp.Net MVC, WCF, Entity Framework e tantas outras. Gosto igualmente de desenhar soluções de arquitectura aplicando padrões de desenho. Gosto de participar, entre outros, nos eventos da Microsoft, das comunidades Netponto e outras de forma a estar atenda às tecnologias emergentes. Linkedin: https://pt.linkedin.com/in/monicascrodrigues

Introdução à Programação com Python,

Algoritmos e logica de programação para iniciantes

Título: Introdução à Programação com Python, Algoritmos e logica de programação para iniciantes

Autores: Nilo Ney Coutinho Menezes

Editora: Novatec

Páginas: 103

ISBN: 978-85-7522-408-3

Formato: Capa soft

Para a review desta edição, foi-me oferecido pelo autor, o livro Introdução à Programação com Python, Algoritmos e logica de programação para iniciantes 2ª edição.

Dividido em 12 capítulos, o livro apresenta uma estrutura bem organizada e de leitura suave, até para os maus adversos leitores de livros técnicos. Começa por apresentar a motivação para a aprendizagem, capaz de cativar tanto iniciantes como estudantes que recorram ao livro para consolidar conhecimentos.

No segundo capitulo o autor tem o cuidado de explicar como instalar o ambiente de desenvolvimento e o interpretador, nos principais sistemas operativos para computadores, evitando assim potenciais dificuldades, pois nem todos utilizam o mesmo sistema operativo e os processos de instalação diferem entre sistemas operativos. Ainda neste capitulo, o autor apresenta de forma simples uma introdução ao ambiente de desenvolvimento de python, e faculta algumas recomendações essenciais para a utilização deste ambiente.

Nos capítulos seguintes, apresenta as noções elementares de programação, nomeadamente conceitos de variáveis, input/output, instruções condicionais, ciclos (repetições), etc… No entanto, no artigo sexto, apresenta um conceito que apesar de útil, é bastante negligenciado em livros destinados a iniciantes, nomeadamente o conceito de coleção, incluindo listas, filas, pilhas, dicionários, tuplas (tuples), dicionários com listas, listas de objectos, entre outros. Adicionalmente foca também os aspectos de ordenação e enumeração.

O sétimo capitulo é todo ele dedicado a operações com strings, desde a pesquisa, ao posicionamento, quebra, concatenação, substituição, remoção de espaços, validação de conteudos , formatação e termina com um exemplo me cativou a minha atenção por não ser muito comum em livros de programação, um jogo! Incomum em literatura, mas ainda assim bastante interessante, e envolvente de todos os aspectos abordados no livro até ao momento.

Nos capítulos seguintes o autor continua abordando aspectos mais técnicos, começando pelas funções e os aspectos de âmbito (scope / escopo), parâmetros, empacotamento e desempacotamento de parametros, passando por funções lambda, números aleatórios e por fim terminado com a função type.

No capitulo seguinte aborda ficheiros (arquivos), começando pelos parâmetro de linha de comandos, passando por leitura e escrita, e algo interessante, a geração de HTML. É particularmente interessante porque desperta o leitor para outros “voos” com a programação em python.

Não poderia estar completo sem abordar o paradigma de programação orientada a objectos, ainda que de forma muito resumida, termina com um exercício.

Não seria um bom livro de python, sem abordar bases de dados, e neste aspecto devo dizer que o autor se esmerou, dedicando um capitulo inteiro ao trabalho com bases de dados, onde se focam os aspectos mais importantes dessas operações. Ao longo do capitulo os exemplos são apresentados recorrendo ao sgbd SQLite, que cada vez é mais utilizado, pela sua robustez e simplicidade.

Quase no terminar do livro o autor, deixa alguns tópicos que os leitores mais curiosos e havidos, podem aprofundar, não terminando assim de forma abrupta, mas deixando o caminho preparado para uma evolução continua, passando pelo paradigma de programação funcional, até ao desenvolvimento de jogos em python e o desenvolvimento web.

Em conclusão, creio que o livro seja recomendável tanto para os leitores mais iniciados, como para os leitores, mais experientes que desejem iniciar-se na linguagem python. De igual forma o livro é um excelente guia para consulta, apresentando os temas com uma linguagem “suave” e captivante.

AUTOR

Escrito por António C. Santos

Programar é criar um novo mundo escrevendo código, cumprindo os mais elementares desígnios da vida, “aprender, ensinar, criar, partilhar, melhorar e seguir”. Formou-se no Instituto Politécnico de Viana do Castelo. Membro da Comunidade Portugal-a-Programar desde Agosto de 2007, é também membro da Sahana Software Foundation, onde é Programador Voluntário desde 2012, actualmente exerce funções de mentor voluntário na plataforma MOOC Coursera . Twitter:@apocsantos

Segurança

Segredos de Numeração

Segredos de Numeração

Neste mundo moderno, tudo é um número (ou vários). Neste artigo veremos alguns exemplos de numerações utilizadas em aplicações de negócio, o seu significado, como construí-los e como validá-los. Uma explicação mais detalhada da matemática dos dígitos de controlo deixa-se para o leitor.

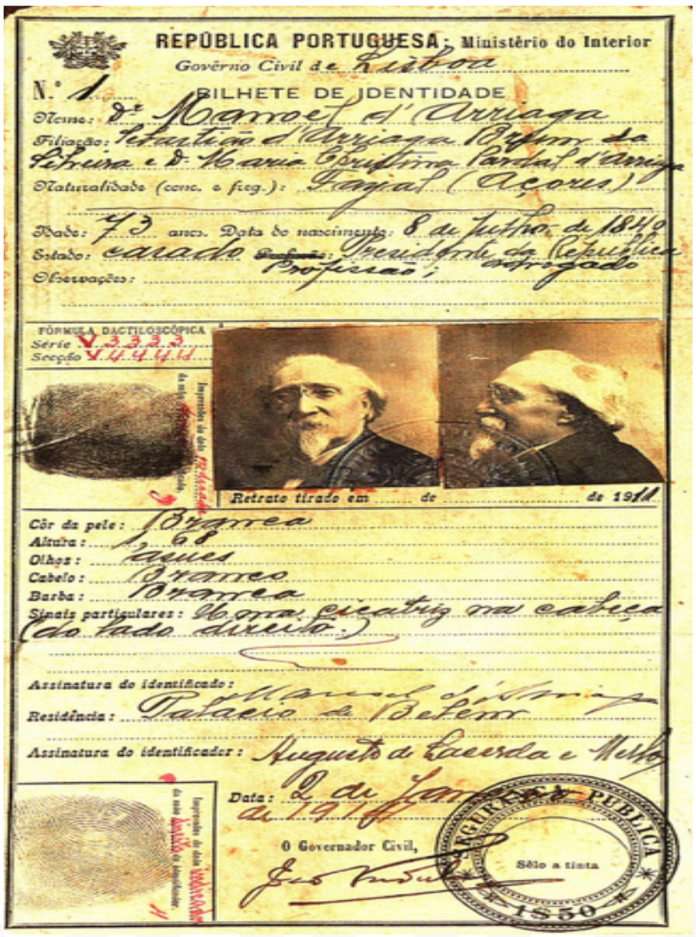
Um dígito de controlo (check digit), que pode ser numérico ou alfanumérico, é um valor que pode integrar ou ser separado do número original e que valida possíveis erros de introdução. O erro mais habitual é a troca acidental da ordem de dois dígitos. E.g. "xxxx12xx" or "xxxx21xx".

Atenção: As funções de validação foram criadas de acordo com a especificação e testadas apenas com uma amostragem reduzida, pelo que podem conter erros. Verificar antes de utilizar.

Número de Identificação Civil (NIC)

O número presente no Cartão de Cidadão (ou no antigo Bilhete de Identidade) é um número sequencial (actualmente com 8 dígitos, por haverem cerca de 10500000 cidadãos em Portugal), com um dígito de controlo em separado.

O primeiro Bilhete de Identidade (número "1") foi emitido em 1914 ao então Presidente da República, Manuel de Arriaga.



Adicionalmente, no Cartão de Cidadão, são adicionados 3 novos dígitos: os primeiros dois (letras) correspondem à emissão individual do cartão ("ZZ" é o primeiro cartão desse cidadão, "ZY" o segundo, etc.), e o último corresponde a um novo dígito de controlo, já que o dígito de controlo anterior não é matemáticamente fiável E.g. "12345678 9 ZY6", correspondente ao 2º cartão pedido pelo cidadão 12345678, com os dígitos de controlo 9 (legacy) e 6 (novo).

Como calcular/validar o primeiro número de controlo? (Exemplo em JavaScript)

```
function CalculaControlo_NIC(nic) {
    if (nic.length > 8) {
        return ("ERRO: O NIC deve ter 8
                                    dígitos.");
    }
    else if (nic.length < 8) {
        nic = ("00000000" + nic).substr(-8);
    }

    Total = 9 * nic.charAt(0)
          + 8 * nic.charAt(1)
          + 7 * nic.charAt(2)
          + 6 * nic.charAt(3)
          + 5 * nic.charAt(4)
          + 4 * nic.charAt(5)
          + 3 * nic.charAt(6)
          + 2 * nic.charAt(7);
    Controlo = 11 - (Total % 11);
    if (Controlo > 9) Controlo = 0;

    return (Controlo);
}

document.write(CalculaControlo_NIC("12345678"));
```

Número de Identificação Fiscal (NIF/NIPC)

O Número de Identificação Fiscal (NIF) partilha a numeração com o Número de Identificação de Pessoa Colectiva (NIPC). É um número de exactamente 9 dígitos numéricos, em que o último corresponde ao dígito de controlo (integrado no número), e os primeiros correspondem ao tipo de entidade:

- 1X: Pessoa singular (completo durante os anos 1980s);
- 2X: Pessoa singular (emissão começou no final dos anos 1980s);
- 3X: Pessoa singular (reservado - seria necessário haver 20 milhões de cidadãos para utilizar);
- 4X: Pessoa singular (reservado - seria necessário haver 30 milhões de cidadãos para utilizar);
- 45: Cidadãos não residentes;
- 5X: Pessoa colectiva (NIPC);

- 6X: Entidade pública;
- 7X: Situações especiais (heranças, fundos, etc.);
- 8X: Empresário em Nome Individual (obsoleto);
- 9X: Situações especiais (condomínios, sociedades irregulares ou sem personalidade jurídica, etc.).

Como o dígito de controlo faz parte do número, é mais fácil evitar falhas. No entanto, o algoritmo de cálculo é idêntico ao do NIC.

Como validar o NIF/NIPC? (Exemplo em JavaScript)

```
function Valida_NIF(nif) {
    if (nif.length != 9) {
        return ("ERRO: O NIF deve ter 9 dígitos.");
    }

    Total = 9 * nif.charAt(0)
          + 8 * nif.charAt(1)
          + 7 * nif.charAt(2)
          + 6 * nif.charAt(3)
          + 5 * nif.charAt(4)
          + 4 * nif.charAt(5)
          + 3 * nif.charAt(6)
          + 2 * nif.charAt(7);
    Controlo = 11 - (Total % 11);
    if (Controlo > 9) Controlo = 0;

    return (Controlo == nif.charAt(8));
}
document.write(Valida_NIF("123456789"));
```

Número de Identificação da Segurança Social (NISS)

Este número tem exactamente 11 dígitos numéricos, sendo o primeiro o tipo, e o último o dígito de controlo:

- 1xxxxxxxxC: Pessoa singular, o número do meio é atribuído internamente;
- 2nnnnnnnnC: Pessoa colectiva, o número do meio corresponde exactamente ao NIPC (9 dígitos).

Como validar o NISS? (Exemplo em JavaScript)

```
function Valida_NISS(niss) {
    if (niss.length != 11) {
        return ("ERRO: O NISS deve ter 11
                                        dígitos.");
    }
    else if (niss.charAt(0) != 1 && niss.charAt
                                        (0) != 2) {
        return ("ERRO: O NISS deve começar por 1 ou
                                        2.");
    }

    Total = 29 * niss.charAt(0)
          + 23 * niss.charAt(1)
          + 19 * niss.charAt(2)
          + 17 * niss.charAt(3)
          + 13 * niss.charAt(4)
          + 11 * niss.charAt(5)
          + 7 * niss.charAt(6)
          + 5 * niss.charAt(7)
          + 3 * niss.charAt(8)
          + 2 * niss.charAt(9);
    Controlo = 9 - (Total % 10);
    return (Controlo == niss.charAt(10));
}
document.write(Valida_NISS("11234567892"));
```

Número de Identificação Bancária Português (NIB)

O NIB é um código com 21 dígitos numéricos no formato "IIII AAAA CCCC CCCC CCC KK". Os espaços são opcionais e omitidos na validação:

- IIII: Código numérico de instituição financeira (banco, instituição de investimento, Direcção Geral do Tesouro (0781));
- AAAA: Código do balcão (referência interna, ou zeros se não utilizado);
- CCCC CCCC CCC: Número e tipo de conta, incluindo quaisquer dígitos de controlo internos da instituição;
- KK: dígitos de controlo.

Como validar o NIB? Na página da Wikipedia existem vários exemplos de validadores em várias línguas. O exemplo seguinte foi retirado daqui e é escrito em PHP:

```
function isValidNib ($nib)
{
    $result = "";
    if (strlen (intval ($nib) ) != 21)
        return "NIB INVALIDO (Standard: 21
                algarismos. Introduzido: " .
                      strlen(intval($nib)) . ")";

    $nnib = str_split (intval ($nib) );

    for($i=0; $i < 19 ; $i++)
    {
        $result = (($result + $nnib[$i]) * 10) %
                                                97;
    }
    $result = 98 - (($result * 10) % 97);

    if($result < 10) $result = "0" + $result;

    if(substr ($nib, 19, 2) != $result)
        return "NIB INVALIDO";
    else
        return "NIB VALIDO";
}
```

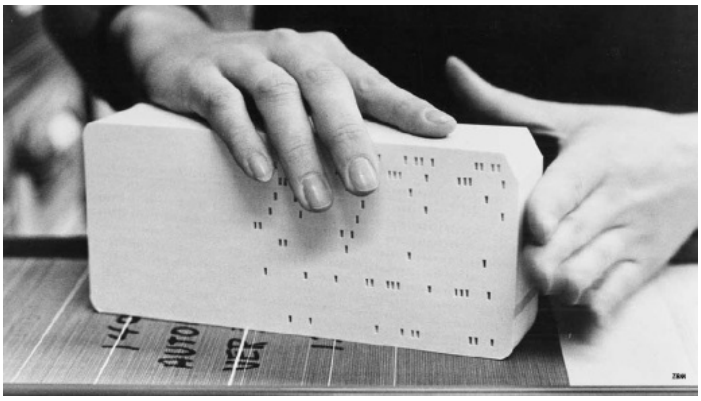
International Bank Account Number (IBAN)

O IBAN é um código alfanumérico de dimensão variável (máximo 34 dígitos). É composto pelo código ISO 3166-1 Alpha-2 do país (e.g. "PT"), dois dígitos de controlo e de seguida o Número de Identificação Bancária nacional (que em alguns países é alfanumérico). Os espaços são opcionais e omitidos na validação.

Em Portugal, e porque a validação do IBAN é semelhante à do NIB, o prefixo é sempre "PT50", seguido do NIB. E.g. "PT50 IIII AAAA CCCC CCCC CCC KK".

Embora tecnicamente semelhante ao NIB, a validação do IBAN (dois dígitos de controlo) implica somar até va-

lores bastante acima dos inteiros usuais nos computadores, pelo que é necessário utilizar algumas artimanhas matemáticas para calcular. Do ponto de vista Português, basta-nos apenas confirmar o prefixo "PT50" e validar o NIB conforme indicado em cima. Por esse motivo não reproduzimos aqui o algoritmo de validação do IBAN para o caso internacional.



Business Identifier Codes (SWIFT-BIC)

Corresponde ao identificador internacional de instituição financeira, cujo formato é definido no ISO 9362 e gerido pela Society for Worldwide Interbank Financial Telecommunication (SWIFT). É um código alfanumérico (tipicamente letras) que pode ter 8 ou 11 dígitos, no formato "IIIICCLL" ou "IIIICCLLbbb":

- IIII: Código da instituição financeira (letras no caso português);
- CC: Código ISO 3166-1 Alpha-2 do país (e.g. "PT");
- LL: Código de localização (cidade ou outra);
- bbb: Código do balcão, referência interna ou não utilizado;

  E.g. "CGDIPTPL" (Caixa Geral de Depósitos - geral)

  E.g. "CGDIPTPLOSF" (Caixa Geral de Depósitos - offshore da Madeira)

Cartão Bancário

Os cartões bancários são compostos por um número variável de dígitos numéricos (máximo 19, típico 16). Os primeiros dígitos correspondem à entidade emissora e o último ao dígito de controlo (e.g. "MIII IInn nnnn nnnC"):

- M: Major Industry Identifier (MII). Valores típicos são:

  - 3: American Express, Diners Club, etc.

  - 4: Visa, Visa Electron, etc.

  - 5: MasterCard, Maestro, etc.

  - 6: Discover Card, Maestro, etc.

- MIII II: Issuer Identification Number (IIN), incluindo o MII em cima, correspondente ao banco emissor;
- n: Número individual do cartão;
- C: dígito de controlo;

Como validar? Depende do emissor, do tamanho do número e do algoritmo. Na maioria dos casos é utilizado o algoritmo de Luhn.

Código Postal Português (CP)

O código postal Português, conforme definido pela entidade privada CTT, tem o seguinte formato "AAAA-BBB Local", onde:

- AAAA: Código postal principal (1000-9999), correspondente à zona de distribuição postal (não à localidade);
- BBB: Código de proximidade (001-999, 000 ou omisso se desconhecido), permite identificar o bairro, rua, o prédio e/ou empresa;
- Local: Denominação da zona de distribuição e proximidade (não necessariamente a localidade oficial).

Destes, apenas é necessário "AAAA" ou "AAAA-BBB" sendo o "Local" ignorado no processamento automático da correspondência. Quando foi criado, há quase 20 anos, os valores possíveis de "AAAA-BBB" eram cerca de 160000, estando agora perto dos 200000.

Como validar? Devem ser usadas tabelas de base de dados com os valores possíveis. No pior caso, validar apenas as gamas indicadas em cima.

Códigos DCF e NUTS

Os códigos de Distrito, Concelho e Freguesia são utilizados por entidades governamentais para diversos fins, desde as Finanças à Comissão Nacional de Eleições. É um sistema de divisão territorial hierarquico ("DDCCFF"), onde cada componente usa 2 dígitos numéricos:

- DD: Código de distrito, parcialmente baseado no ISO 3166-2:PT

  - 01-18 = Continente;

  - 19-22 = Obsoletos, antigos distritos das Regiões Autónomas;

  - 31-32 = Região Autónoma da Madeira (divisão por ilhas);

  - 41-49 = Região Autónoma dos Açores (divisão por ilhas).

- CC: Código de concelho dentro do distrito. Actualmente existem pouco mais de 200 concelhos;
- FF: Código de freguesia dentro do concelho. Actualmente existem pouco mais de 3000 freguesias, mas os códigos das extintas freguesias (originalmente cerca de 4100) foram mantidos.

A entidade que gere o Código Postal Português fornece tabelas com a correspondência entre código postal e DCF.

Os códigos NUTS (Nomenclaturas de Unidades Territoriais Estatísticas) são uma classificação de zonas administrativas definidas ao nível Europeu:

- NUTS-0: Cada um dos países da União Europeia;
- NUTS-1: Primeira divisão dentro de cada país. Em Portugal há 3 (PT1=Continente, PT2=Açores, PT3=Madeira);
- NUTS-2: Segunda divisão. Em Portugal apenas o Continente usa esta divisão (PT11=Norte, PT16=Centro, PT17=Lisboa, PT18=Alentejo, PT15=Algarve), enquanto as Regiões Autónomas não têm divisão (PT20=Açores, PT30=Madeira);
- NUTS-3: Terceira divisão (E.g. PT11A=Porto, PT170=Lisboa);
- NUTS-4 / LAU-1: Divisão administrativa local, correspondente a Concelhos (compatível com DCF);
- NUTS-5 / LAU-2: Divisão administrativa local, correspondente a Freguesias (compatível com DCF).

Os códigos NUTS não são compatíveis com DCF, já que o mesmo distrito pode ter concelhos em NUTS diferentes.

Código de barras

Existe um número muito elevado de tipos de códigos de barras, desde lineares, a 2D e a formatos "estranhos". Os mais habituais são:

- Code 25 – Interleaved 2 of 5: Conteúdo alfanumérico para fins diversos;
- QRCode: Código com 2 dimensões e conteúdo alfanumérico para fins diversos;
- EAN-13: Internacional, o mais usual em produtos. Tem 13 dígitos numéricos;
- EAN-8: Um sub-conjunto do EAN-13;
- JAN: Usado no Japão, idêntico ao EAN-13;
- UPC-A: Usado nos EUA, compatível com EAN-13 (primeiro dígito EAN-13 é "0"). Tem 12 dígitos numéricos.

Os códigos compatíveis com EAN-13 têm a seguinte composição:

- Prefixo GS1 (3 dígitos): Determina entre outras coisas o país (e.g. 560=Portugal);
- Código do fabricante (dimensão variável): Atribuído pelo GS1;
- Código do produto (dimensão variável até preencher 12 dígitos): Atribuído pelo fabricante;
- Dígito de controlo (1 dígito).

Alguns prefixos têm significado especial:

- Começados por 0 são códigos UPC-A usados nos EUA (restantes 12 dígitos iguais);
- Começados por 2 são de distribuição restrita e podem ser criados por qualquer retalhista para uso local (e.g. fruta a peso no momento);
- Começados por 978 e 979 para o ISBN (restantes 10 dígitos do ISBN);
- Começados por 977 para o ISSN.

Como validar? Depende do tipo de código de barras. Normalmente a validação já é feita nos leitores por hardware mas poderá ser complementada por validação por software.

> ***"Neste mundo moderno, tudo é um número (ou vários). (...) uma explicação mais detalhada da matemática dos dígitos de controlo deixa-se para o leitor."***

ISBN e ISSN

O International Standard Book Number (ISBN), definido no ISO 2108, é um número de 10 ou de 13 dígitos numéricos. Se tiver 13 dígitos, inclui o prefixo GS1 (978 ou 979) tornando-se num EAN-13. Poderá ter espaços ou "-" no meio do número a separar as secções, e eventualmente o prefixo "ISBN":

- Prefixo GS1 (3 dígitos, opcional): Torna o ISBN num EAN-13;
- Grupo (dimensão variável): País, região, grupo de língua, etc.;
- Editor (dimensão variável);

SEGREDOS DE NUMERAÇÃO

- Publicação (dimensão variável);
- Dígito de controlo (1 dígito).

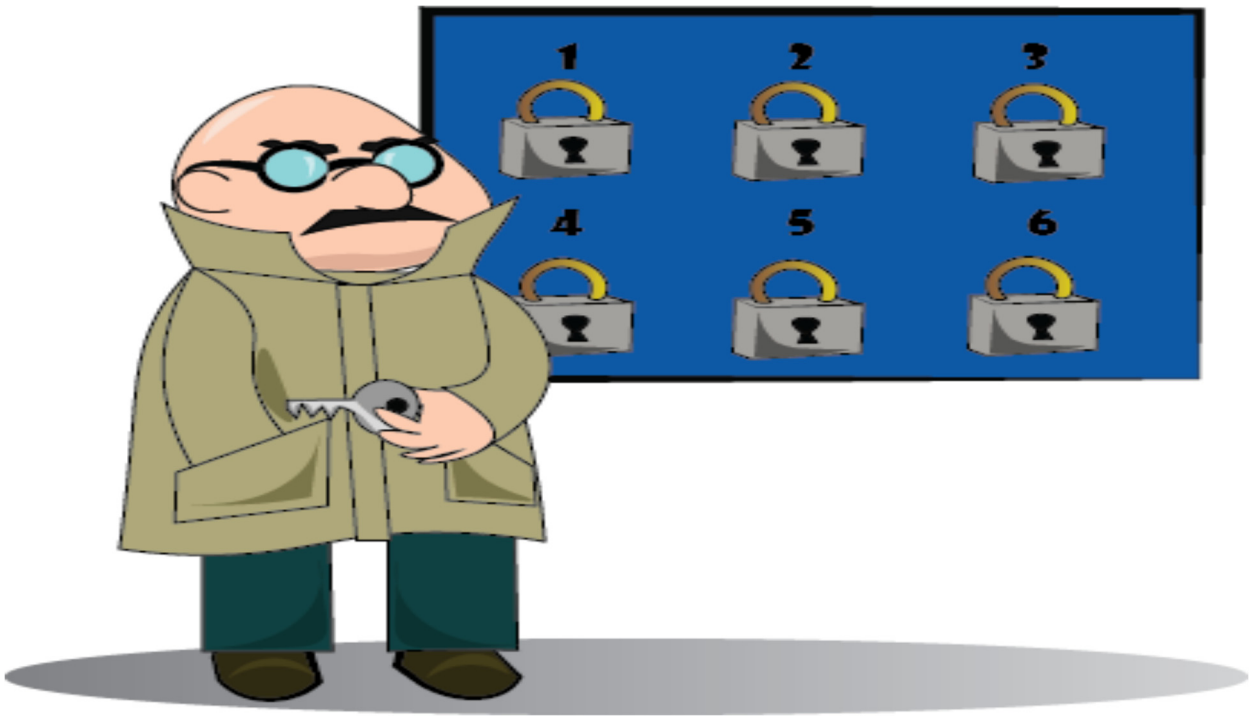
O International Standard Serial Number (ISSN), definido no ISO 3297, é um número de 8 dígitos utilizado para identificar publicações periódicas (revistas, jornais, etc.). Poderá ter o prefixo "ISSN":

- Identificador único da publicação (7 dígitos);
- Dígito de controlo (1 dígito) do ISSN.

Um ISBN pode tornar-se num EAN-13:

- Prefixo GS1 (3 dígitos): "977";
- Identificador único da publicação (7 dígitos);
- Código ISSN (8 dígitos);
- Código comercial (2 dígitos): altera sempre que o preço ou outras condições comerciais mudem;
- Dígito de controlo (1 dígito) do EAN-13.

Tanto o ISBN como o EAN-13 não conseguirão identificar unicamente uma edição específica da publicação periódica, necessitando sempre dum código de barras adicional (colocado ao lado) com o número da edição (2 ou 5 dígitos, EAN-2 ou EAN-5).



AUTOR

Escrito por André Melancia

Independent Developer/DBA/Consultant. Microsoft Certified Trainer (MCT) focusing on SQL Server, Azure and IoT. 17+ years' fun developing information and multimedia systems, DBA, project and IT management. PowerShell Portugal, IT Pro Portugal and IoT Portugal communities organiser. IPv6 Portugal, DNSSec Portugal and Windows Development Portugal online communities moderator. Actively volunteering, organising, speaking or just participating at community meetings and events like SQLSaturdays, SQLBits, SQLRelay, Maker Faire Lisbon, Arduino/Genuino Day Lisbon, Global Azure Bootcamp Lisbon, etc.

Proud uncle and food devouring expert, with dangerous pussy cat as companion.

Go to http://Andy.PT and you'll know the same as the NSA...

No Code

ShiftAppens 2017

Raspberry Pi Zero W

INTERFACE HUMANO-COMPUTADOR, NANOTECNOLOGIA E A DEPENDÊNCIA TECNOLÓGICA.

SHIFTAPPENS 2017

A meio do passado mês de Fevereiro, nos dias 17, 18 e 19, deu lugar no pavilhão Mário Mesquita mais uma edição do SHIFT APPens, mais propriamente a 4º edição do Evento. Como atividade principal era pedido aos participantes que durante os três dias do evento pudessem desenvolver uma aplicação à sua escolha, formando também equipas dinâmicas onde várias personalidades de programação e design se poderiam misturar. No fim dos três dias, e sendo prometido muito código e café por parte da organização do Shift, os participantes poderiam apresentar os seus projetos ao júri do evento (Carlos Mota – Representante do Google Developers Coimbra; Alcides Marques - Representante do Laboratório de Informática do Instituto Pedro Nunes; Joana Brites - Representante da Faculdade de Letras da Universidade de Coimbra; Tiago Henriques - Representante da Wit Software), e serem então premiados pelo seu trabalho (existindo claro, o 1º, 2º e 3º prémio). Em conversa com o relações públicas do evento, Filipe Mendes, podemos perceber que este ano o Hackathon teve bastantes mais participações que nos anos anteriores.

Enquanto circulávamos pelo pavilhão, o relações publicas deu-nos a conhecer o espaço e toda a história anterior ao evento – que começou por ser uma reunião de amigos e estudantes que gostavam de codificar e programar, para depois se tornar num evento que tem alguma dimensão. Este ano, como dizia Filipe, houve a possibilidade de alargar o espaço do evento, devido aos participantes, e realizar o mesmo no pavilhão Mário Mesquita, que conta com instalações preparadas para banhos e descanso, sendo este um dos pontos mais relevantes para o sucesso do Hacktathon este ano. Ao longo do espaço, podíamos ver representantes de empresas como a RedLight Software, a Wit, a Câmara de Coimbra, o Alma Shopping e também outros igualmente importantes.

E como não poderia deixar de ser referido, a marca Licor Beirão preparou um espaço Lounge onde os participantes poderiam recarregar baterias ao longo destes três dias, não esquecendo a Boundi, que auxiliou os participantes com o seu café e o Mr. Pizza e Rei dos Frangos com a habitual comida de Hackathon! Além das Talks (conversas por parte das empresas patrocinantes), existiram também vários quizes de cultura geral neste evento, ajudando os participantes a estender a sua mente. Falemos então de números: O Shift APPens contou com a presença de 160 participantes (30% dos participantes vieram de fora de Coimbra e estima-se que 10% dos participantes eram do sexo feminino), 20 indivíduos de STAFF e ainda 25 voluntários – Juntamente com o IPN (Instituto Pedro Nunes em Coimbra), o STAFF e os voluntários construíram toda a infraestrutura de eletricidade e internet do evento (sendo que os cabos foram cedidos por um dos patrocinadores do evento). Uma grande novidade este ano foi que, através de uma plataforma chamada “Slack”, o STAFF e os participantes do evento puderam estar sempre em contacto – vendo os participantes quais as próximas atividades a desenvolver através da plataforma. No entanto, esta edição teve um formato um tanto diferente das duas primeiras, não existindo agora mentores, mas sim um júri.

Quando questionado sobre esta questão, Filipe Mendes respondeu-nos que para os participantes era mais fácil aceitar indivíduos que tivessem mais experiencia na sua própria equipa, ou seja, todos os participantes preferiam ser ajudados por algum que trabalhasse em conjunto com eles, e daí o evento ter então comtemplado a inscrição de pessoas que não fossem apenas estudantes, mas que também trabalhassem ou tivessem alguma experiência na área. Contando com vários apoiantes de peso ao longo dos anos, não podemos deixar de reparar que nomes como a Microsoft ou a Deloitte deixaram de fazer parte deste evento, o que os participantes e STAFF do Shift vêm como uma perda significativa.

Podemos referir que mais um ano o SHIFT APPens foi um sucesso, e que para as próximas edições se esperam coisas ainda mais random e espetaculares!

AUTOR

Escrito por Filipa Antunes Peres.

Licenciada em Design de Moda e Têxtil e Mestre em Marketing, conta ainda com uma especialização em Web Design. Desde pequena que desenvolveu interesse em várias áreas que tenham a ver com investigação social.

RASPBERRY PI ZERO W

Com o intuito de comemorar o quinto aniversário da família Raspberry Pi, o fim de Fevereiro de 2017 trouxe mais uma novidade a esta conhecida família. Como não podia deixar de ser, aqui na Programar continuamos a ser fãs desta temática, motivo pelo qual não hesitamos em dedicar-lhe algumas linhas nesta edição.

Os leitores mais atentos a estas andanças devem lembrar-se que numa edição anterior já falamos acerca do Raspberry Pi Zero, lançado em Novembro de 2015. Ora esta nova placa é o mesmo Raspberry Pi Zero mas "melhorado" em dois pontos que podem ser essenciais para alguns projectos. A versão com a letra W que se juntou ao nome da primeira placa, vem principalmente acrescentar o Wireless e o Bluetooth 4.0. Estas são de facto as vantagens integradas nesta nova versão de uma das placas mais famosas no mundo da tecnologia.

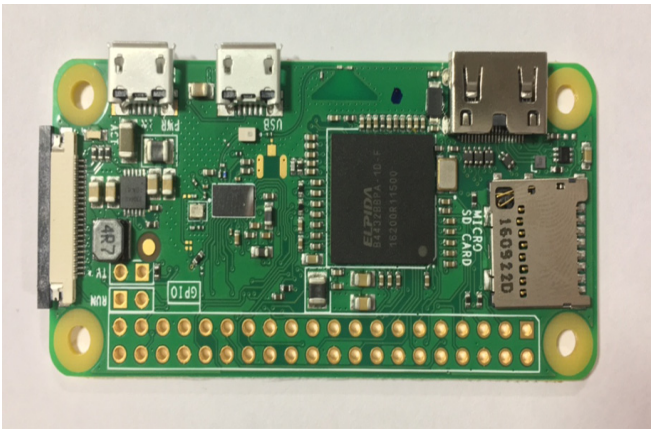


Ilustração 1 - Raspberry Pi Zero W visto de frente

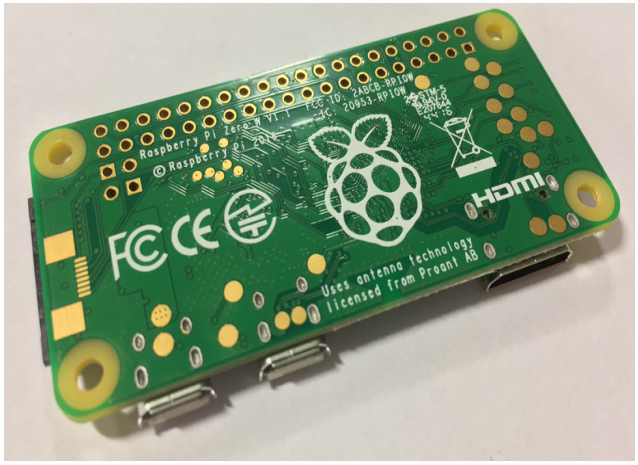


Ilustração 2 - Raspberry Pi Zero W visto de trás

Para recapitular, aqui está a lista completa de recursos para Zero W:

Especificações:

- Dimensões: 65mm × 30mm × 5mm
- SoC: Broadcom BCM2835
- CPU: ARM11 funcionando em 1GHz
- RAM: 512MB
- Wireless: 2.4GHz 802.11n LAN sem fio
- Bluetooth: Bluetooth Classic 4.1 e Bluetooth LE
- Alimentação: 5V, fornecido através de conector micro USB
- Vídeo & Áudio: vídeo 1080p HD e áudio via mini-HDMI
- Armazenamento: Cartão MicroSD
- Saída: Micro USB
- GPIO: GPIO de 40 pinos compatível com HAT
- Interface serial da câmera (CSI)
- 2 portas micro USB

O Zero W pode ter sensivelmente o dobro do preço do Zero, já que foram lançados a 10 dólares e a 5 dólares respectivamente, contudo consoante o nosso objectivo final, o Pi zero W pode ser de facto uma mais-valia a ponderar.

Para não perder o hábito, o Pi W faz jus à sua família, todos os modelos Raspberry possuem um sistema Broadcom num chip (SoC), que inclui uma unidade de processamento central (CPU) compatível com ARM. É importante referir que esta versão W é praticamente idêntica ao anterior Raspberry Pi Zero v1.3 (o que adicionou o conector da câmara), todos os chips e componentes ainda estão apenas no lado superior da placa.

O Pi Zero W usa o mesmo chip Cypress CYW43438 sem fios do Raspberry Pi 3 Model B para o wireless 802.11n LAN e conectividade Bluetooth 4.0.

No entanto apesar do chip ser o mesmo que o Raspberry Pi 3, a antena é completamente diferente. Em vez de ser um pequeno chip branco que o contém, a antena no Pi Zero W é impressa na placa.

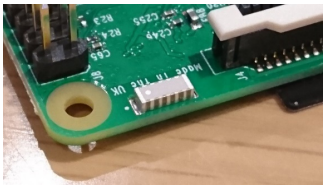


Ilustração 3 - Antena Wireless Raspberry Pi Model 3

Ilustração 4 - Antena Wireless Raspberry Pi Zero W

Em declarações oficiais, Roger Thornton, responsável por este componente da placa, explicou que "A antena do Raspberry Pi 3 é um componente de montagem em superfície, ao passo que a antena Zero W é uma cavidade ressonante que é formada por decapagem de cobre em cada camada da estrutura PCB". Esta tecnologia foi licenciada por uma empresa sueca, a Proant (conforme se pode ver no reverso da placa Zero W).

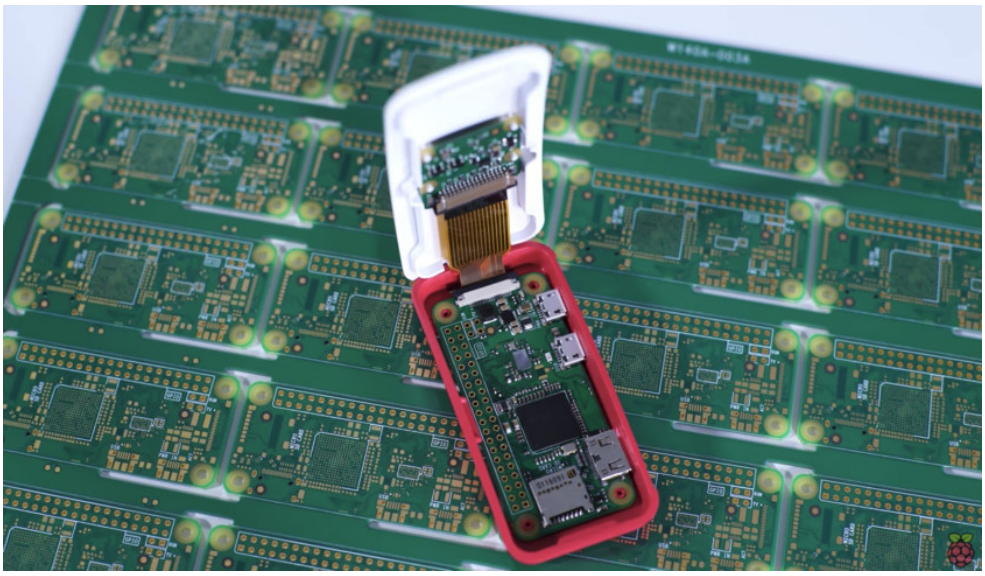
Na prática, na antena no Pi Zero W as ondas de rádio interagem com aquele espaço livre, ressoando na pequena cavidade apenas na frequência certa. Os dois capacitores que estão na parte inferior do plano de terra capturam o sinal de rádio.

Apesar desta novidadade, certamente seria do agrado de muitos utilizadores, um conector sma, que permitisse ligar uma antena wifi externa com maior ganho.

Este novo Raspberry Pi Zero W, apesar de não ser de todo uma novidade estrondosa como outros antecessores seus o foram, continua a ser visto como um elemento a ponderar, uma vez que vem incrementar as possibilidades de conectividade da pequena placa com aparelhos exteriores sem que tenhamos que instalar mais algum módulo externo, uma vantagem principalmente para projecto IoT ou outros projectos que necessitem de conectividade.

Será de esperar uma adopção maior deste novo raspberry pi zero, em projectos como o GameBoy Zero, https://goo.gl/b1VnB8, e outros projectos de hardware recorrendo ao já conhecido software RetroPie, para reproduzir o “aspecto e sensação” dos jogos das décadas de 80 e 90 do século passado.

De igual modo, ao acrescentarem o controlador wifi, o Pi Zero, torna-se também mais interessante para utilização em drones e outros dispositivos autónomos, aliando a conectividade do wifi, com a capacidade de processamento e o gpio do raspberry pi zero.



AUTOR

Escrito por Rita Peres

Natural de Castelo Branco, licenciou-se em Engenharia Informática pela Universidade da Beira Interior. Membro do P@P desde Janeiro de 2010. Embaixadora das Geek Girls Portugal – Núcleo de Lisboa.

25 | 26 | 27 May 2017

O **ISELTech'17** , que decorrerá entre os **dias 25 e 27 de Maio** no campus do **Instituto Superior de Engenharia de Lisboa (ISEL)** , é um evento aberto a estudantes, trabalhadores e entusiastas de tecnologia que queiram adquirir novos conhecimentos e demonstrar as suas capacidades, num espaço aberto à discussão entre participantes e speakers, abordando temas da atualidade tecnológica, como: Mobile Development, Cloud, Web, Gaming, AR/VR, BigData, Internet of Things, Network Engineering e Cibersegurança.

Durante os três dias do evento serão realizados **Workshops** para *ha rd* e *soft skills*, **concursos e sorteios de licenças de software (estágios, licenças JetBrains e Sketch) bem como diversas talks.** O habitual espaço de recrutamento e exposições também estará presente, e contará com a presença de entidades nacionais e internacionais. O último dia fica guardado para o **painel de debate** sobre Cibersegurança, com o tema ***Segurança na Internet: o que ve ndemos de nós?***, procurando abordar temáticas como a produtização do ser humano e a vulnerabilidade da sua privacidade, bem como a ausência de informação e regulamentação, quais os desafios emergentes da segurança na internet e como educar as populações para tal.

Se te interessas por tecnologia, terás certamente várias sessões que te irão interessar.

#SocialMedia

Na divulgação nas redes sociais, recomendamos o uso da seguinte hashtag: #ISELTech

+Info

- Site/Tickets: iseltech.isel.pt
- Sponsors: iseltech.isel.pt#sponsors
- Facebook: /iseltech
- Slack (convite): https://goo.gl/yIFeri
- E mail: iseltech@isel.pt

INTERFACE HUMANO-COMPUTADOR, NANOTECNOLOGIA E A DEPENDÊNCIA TECNOLÓGICA.

Introdução

O presente *release* tem por finalidade abordar como a criação de interfaces avançadas propiciam novos avanços em diferentes áreas do conhecimento. Neste contexto, o foco discutido permeia questões sobre a criação e o desenvolvimento da nanotecnologia que pode estar direta ou indiretamente alterando a interação do homem com o computador.

Um ponto evidente é que o imaginário proposto nas produções cinematográficas, principalmente no que tange a ficção científica apresenta grandes possibilidades de aplicação no uso de Interfaces Humano-Computador (doravante IHC), tanto no aspecto positivo e benevolente como pode ser percebido na produção "Viagem Fantástica" (*Fantastic Voyage*) de 1966, como no aspecto sombrio e devastador na produção "Exterminador do Futuro" (*The Terminator*) de 1984. Cabe como ilustração um breve comentário sobre o filme "*Minority Report*" de 2002 que apresenta um estilo de interface, introduzida tempo depois no console de *videogame* XBOX da empresa Microsoft com o periférico *Kinect* o qual proporciona uma mecânica de jogabilidade semelhante à mostrada no filme.

O estudo do tema IHC não está somente presente no formato visual da tela de um computador ou no formato ergonômico que determinado periférico possa ter para a interação com certo sistema. Este conceito vai a um nível mais além.

Nanotecnologia

Antes de conceituar o que vem a ser *nanotecnologia*, cabe conceituar o que vem a ser *nanômetro*. De acordo com Jordão (2009) nanômetro é uma forma de medida como são o metro e o quilômetro, sendo 1 nm (um nanômetro) equivalente a um milionésimo do milímetro (SETTI, 2012) ou ainda é a bilionésima parte de um metro, sendo esta medida invisível a olho nu e comumente utilizada para a medição de ligações químicas (FUNDACENTRO, 2013).

Jordão (2009) adverte que "o nanômetro não é uma partícula ou um componente da eletrônica, mas é apenas uma mera forma de medida", acrescenta que o nome nanotecnologia, escolhido em 1974 na Universidade Científica de Tóquio decorre do pequeno tamanho de vários itens utilizados para a construção de componentes de alta tecnologia.

De acordo com o Guia do Estudante (2013) nanotecnologia "é a ciência que projeta e desenvolve produtos e processos tecnológicos a partir de partículas minúsculas".

Comenta Jordão (2009) que a nanotecnologia se encontra empregada em mais de oitocentos produtos, destacando como maior uso a produção de microprocessadores, afirmando se encontrar microprocessadores com 45 nm de tamanho. Cita seu uso em outras áreas ao afirmar que:

> *esta tecnologia não foi criada somente para ajudar na informática, mas para revolucionar de maneira geral em qualquer área onde fosse necessário. Atualmente, pode-se relatar a aplicação da nanotecnologia na Medicina, na Química, na Física quântica, nas indústrias que criam protótipos aeroespaciais, refinarias e muitas tantas outras áreas.*

Apesar de o termo nanotecnologia ter sido cunhado em 1974 (Jordão, 2009), referencias a esta possibilidade foram apresentadas no cinema, principalmente no estilo de produção da ficção científica. A primeira referência a esta possibilidade ocorreu no ano de 1966 quando os estúdios *20th Century Fox* produziram o filme "Fantastic Voyage" baseado no livro homônimo, escrito por Isaac Asinov que retrata um grupo de médicos e cientistas que são miniaturizados em um submarino a uma escala microscópica e introduzidos em um paciente para destruírem um coágulo sanguíneo.

Passados os anos a miniaturização de um submarino com tripulação ainda não é possível, mas é possível a construção de pequenos robôs, como os chamados robôs de DNA - *DeoxyriboNucleic Acid* (ADN - Ácido Desoxirribonucleico) feitos de material genético, sendo estes nano robôs que podem carregar determinado anticorpo com a finalidade de atacar células cancerígenas, preservando as células saudáveis de acordo com exposto pelo professor Stevens Rehen em entrevista ao programa Globo News Ciência e Tecnologia no Brasil (GLOBO, 2013).

Na área médica Jordão (2009) aponta que:

> *temos como exemplo aparelhos para diagnosticar determinadas doenças, as quais não podem ser detectadas apenas com base em sintomas e exames comuns. Além disso, a nanotecnologia é muito utilizada para criar remédios, afinal, trabalhar com componentes químicos de tamanho tão pequeno, exige uma tecnologia minúscula o suficiente.*

Para a área de estudo da IHC, segundo Gargin (2010, p. 78) os objetos disponíveis em um ambiente poderão se tornar dispositivos de integração entre os seres humanos como parte de interfaces cada vez mais invisíveis e integradas ao ambiente, graças aos avanços na

miniaturização desses dispositivos, tornando-os mais portáteis e presentes no cotidiano das pessoas e acrescenta:

> *Os dispositivos reagirão ao que os seres humanos desejam, antecipando suas necessidades de informação, numa interação que será mais natural e intuitiva. Eles reconhecerão e contextualizarão a pessoa ou pessoas sobre o ambiente em que estejam inseridas. A obtenção de informações ocorrerá, a partir dos dados existentes nas mais diversas bases, de modo muito rápido e todos os campos das ciências serão afetados pela facilidade e velocidade trazidas ao modelo de colaboração.*

Análise Crítica

A partir desta exposição, leva-se a crer que a nanotecnologia está influenciando a criação de IHC cada vez mais simples, amigáveis e adaptadas as necessidades humanas.

Na área de saúde, já esta proporcionando a médicos cirurgiões maneiras de realizarem cirurgias menos evasivas com o uso de robôs programados para determinada tarefa, além do desenvolvimento de medicamentos a partir dessa tecnologia. Esta é uma maneira de uso de IHC, além das janelas dos ambientes gráficos encontrados nos computadores e dispositivos de comunicação.

Na área da informática e Tecnologia da Informação as IHCs são mais percebidas, principalmente no formato de comunicação que os programas efetuam com o usuário. No entanto, há diversos exemplos de interfaces mal elaboradas que transforma o usuário em um "jóquei de *mouse* (jóquei de rato)". Mas há as bem sucedidas como as existentes nos *tablets* e *smartphones* que proporcionam a sensação de "folhear" a tela.

Apesar das evoluções existentes e daquelas que ainda virão é necessário tomar o cuidado para não ser seduzido pelo "canto da sereia", pois a humanidade está se tornando cada vez mais dependente dessas tecnologias, ficando cada vez mais sedentária e acostumada com essas "mordomias", corre-se o risco de se ter um mundo similar ao mostrado no filme Wall-e dos estúdios Disney.

Cabe, dentro do exposto, questionar um episódio da série *Star Trek* clássica chamada o *Computador Supremo*, onde é proposto instalar na nave *USS Interprise* um computador em substituição a tripulação humana. O curioso no episódio são os questionamentos advindos da experiência proposta, considerada inaceitável pelo Capitão Kirk em ser substituído por um "computador". O curioso é o viés mostrado em relação a uma sociedade acostumada a usar elementos tecnológicos avançados como a forma de comunicação verbal com o computador da nave, onde a IHC é quase que humana no lado do computador. Este episódio mostra bem o fato de se usar os recursos tecnológicos como ferramentas de apoio, mas não como substitutos. Assim sendo, este é de fato o grande desafio na concepção de *hardware*, *software* e na forma de interface destes componentes com o ser humano de maneira que venham a ajudar a sociedade humana a viver melhor, mas não a sucumbir em si mesma.

Bibliografia

FUNDACENTRO. **Nanômetro**. Brasil: Ministério do Trabalho e Emprego, 2013. Disponível em: <http://www.fundacentro.gov.br/conteudo.asp?D=Nano&C=1558&menuAberto=1556>. Acesso em: 3 de mai. 2013.

GARGIN, S. M. **Estudo das Evoluções das Interfaces Humano-Computador**. São Paulo: Escola de Engenharia de São Carlos da Universidade de São Paulo, 2010.

GLOBO. **Globo News Ciência e Tecnologia**. Direção: Eugenia Moreyra. Rio de Janeiro: Direção Geral de Jornalismo e Esporte, 2013 [produção]. 21,38 min, colorido. Disponível em: <http://globotv.globo.com/globo-news/globo-news-ciencia-e-tecnologia/t/todos-os-videos/v/cientista-fala-sobre-a-ciencia-mostrada-no-cinema-e-sua-possibilidade-real-de-existir/2427589/>. Acesso em: 2 de mai. 2013.

GUIA DO ESTUDANTE. **Nanotecnolgia**. Brasil: Editora Abril, 2013. Disponível em: <http://guiadoestudante.abril.com.br/profissoes/ciencias-exatas-informatica/nanotecnologia-687230.shtml>. Acesso em: 8 de set. 2016.

JORDÃO, F. **O que é nanotecnologia?**. Brasil: TecMundo, 2009. Disponível em: <http://www.tecmundo.com.br/amd/2539-o-que-e-nanotecnologia-.htm>. Acesso em: 23 de ago. 2016.

SETTI, R. **1 Milionésimo de Milímetro**. Brasil: Revista Veja, 2012. Disponível em: <http://veja.abril.com.br/blog/ricardo-setti/tag/1-milionesimo-de-milimetro/>. Acesso em: 3 de mai. 2013.

AUTOR

Escrito por Augusto Manzano

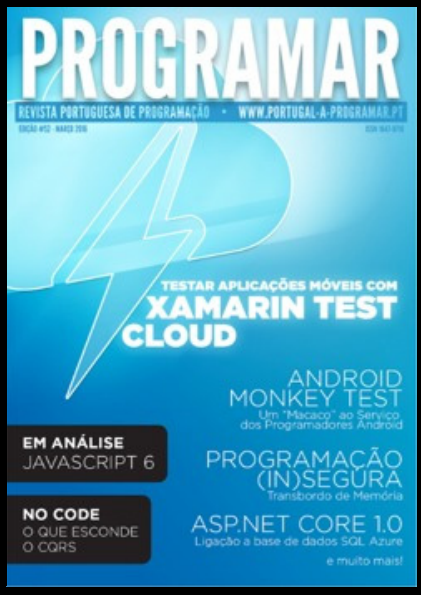
Natural da Cidade de São Paulo, tem experiência em ensino e desenvolvimento de programação de software desde 1986. É professor da rede federal de ensino no Brasil, no Instituto Federal de Educação, Ciência e Tecnologia. É também autor, possuindo na sua carreira várias obras publicadas na área da computação.

Veja também as edições anteriores da Revista PROGRAMAR

54ª Edição - Janeiro 2017

53ª Edição - Agosto 2016

52ª Edição - Março 2016



51ª Edição - Dezembro 2015

50ª Edição - Setembro 2015

49ª Edição - Junho 2015

e muito mais em ...

DUVIDAS?

IDEIAS?

AJUDAS?

PROJECTOS?

portugal-a-programar
.org