PROGRAMAR

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

EDIÇÃO #58 OUTUBRO 2017

hadoop + RASPBERRY Pi

A PROGRAMAR

AZURE APLICAÇÃO WEB JAVA

JQUERY MOBILE APLICAÇÃO MOBILE

LUA LINGUAGEM DE PROGRAMAÇÃO

LINGUAGEM C TIPOS DE DADOS

SEGURANÇA

BLOCKCHAIN

A REVOLUÇÃO

SEGURANÇA DAS APLICAÇÕES ANDROID

MERKLE TREE

ELECTRÓNICA

ESP32 MICROPYTHON

POMAR MUSICAL

COLUNAS

LIST + DATA TABLE C#

NO CODE

RGPD

A HUMANOIDE SOPHIA

HYDRINEY

ELECTRÓNICA

NODE.JS CONSTRUÇÃO DE APLICAÇÕES WEB

TYPESCRIPT CRIAÇÃO

EDITORIAL

EQUIPA PROGRAMAR

Contacto
revistaprogramar@portugal-a-programar.org

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

ISSN
1 647-071 0

Kernel Panic: Fatal Exception

Existem imensas piadas, umas mais elaboradas outras menos elaboradas, em volta de “exceptions” (excepções)! Uma excepção, indica mais do que ser uma “excepção”, indica algo que não correu como esperado, ou como previsto, ou como suposto, em suma “como”! ;)

No meio de tantas excepções (exceptions), e sem “entrar em pânico”, afinal não somos um “kernel” qualquer, aparece a “exception” à regra, que somos todos que lê-mos a PROGRAMAR, escrevemos, participamos, dedicamos o nosso tempo à comunidade! Somos a “exception”, que alguns acham “fatal exception”, sem retornar e sem “catch”, numa espécie de loop interminável!

Desde a ultima edição imensas coisas aconteceram, a tecnologia evoluiu, tanta coisa mudou, o Verão fez-se Outono (disfarçado), mas Outono, o cheiro a uma bebida quente tornou-se “regra” e não “exception”, a falta desse cheiro quase se torna “fatal exception”! Aproxima-se um tempo mais frio de clima e mais “quente” de espirito e claro sempre cheio de novidades de tecnologia, pois é daquelas alturas do ano em que “aparecem sempre coisas novas”! Enquanto o tempo vai passando, deixamos os nossos leitores com mais uma edição da PROGRAMAR e a promessa de que a próxima será ainda melhor!

Até à próxima edição, boas leituras!

António Santos

ÍNDICE

EVENTOS

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

NOTÍCIAS

Pela primeira vez, investigadores portugueses publicam na MICRO – uma das mais importantes conferências mundiais sobre microarquiteturas de computadores

Com uma investigação realizada na área de processadores (microchips) optimizados para inteligência artificial, os investigadores Pedro Duarte, Gabriel Falcão e Pedro Tomás conseguiram a primeira publicação portuguesa na prestigiada conferência internacional sobre microarquiteturas de computadores – MICRO – que vai já na 50ª edição (https://www.microarch.org/micro50/index.html).

Os três investigadores, desenvolveram uma nova ferramenta de software, que permite analisar de forma automática, um programa e gerar um processador optimizado, em termos de área de chip e consumo energético, de modo a poder ser usado em chips reconfiguráveis do tipo FPGA.

Esta nova tecnologia que conduziu à obtenção desta publicação "*permite desenvolver hardware utilizando uma abordagem próxima do desenvolvimento típico de software. Daí resultam grandes vantagens, nomeadamente ao nível do aumento da comunidade de utilizadores, que tipicamente consiste em programadores convencionais, que não possuem conhecimentos ao nível do desenho de microcircuitos para conceberem os processadores de forma manual*", sublinham os investigadores.

O artigo foi apresentado em Boston, EUA, durante a conferência MICRO (que se realizou entre 14 e 18 de Outubro), cuja taxa de aceitação de artigos é muito competitiva, rondando os 15%, e que apenas publica cerca de 50 artigos por ano.

Além de ser a primeira vez que um artigo português é publicado nesta prestigiada conferência, que reúne investigadores e empresas de topo de todo o mundo, demonstra a qualidade do "made in Portugal"!

Movimento Maker Portugal promove 1º encontro

No passado mês de Setembro realizou-se o primeiro encontro do Movimento Maker Portugal, subordinado ao tema "EU QUERO IOTIZAR A MINHA CASA", na cidade de Leiria.

Tratou-se de um evento centrado no conceito, hands-on, cheio de workshops sobre a temática que lhe deu mote, bem como talks dos mais variados temas interessantes para a comunidade Maker!

Destacamos o facto de ter sido o primeiro encontro de uma comunidade Maker tão geograficamente abrangente, com participantes de todas as zonas do país, facto pelo qual o Movimento Maker Portugal está de parabéns!

TEMA DE CAPA

Raspberry Pi Hadoop

Raspberry Pi Hadoop

Continuando as “aventuras” com o Raspberry Pi e a temática do processamento paralelo e distribuído, sobre o qual escrevi na edição 48 em Março de 2015, decidi desta vez trazer uma outra temática interessante, para quem gosta destas “aventuras”. Desta vez em vez de ser MPICH será Apache Hadoop!

Ao longo deste artigo, em que se prevê que o leitor não disponha de conhecimento prévio sobre hadoop, irá ser montado um cluster Apache Hadoop, recorrendo a unidades Raspberry Pi, e executados alguns exemplos demonstrativos.

Para quem não conhece o Apache Hadoop, é uma framework, de código aberto (open-source) de armazenamento e processamento distribuído de grandes volumes de dados, em clusters de computadores, usando modelos de programação simples. O hadoop foi desenhado para ser escalonado de forma a poder “crescer” desde um simples nó até centenas ou mesmo milhares de nós, onde cada um disponibiliza o seu poder de processamento e armazenamento. Contrariamente a outras soluções, o foi também desenhado de forma a detectar e tratar falhas, ao nível da camada de aplicação, em vez de depender do hardware para disponibilizar a alta-disponibilidade necessária.

“A biblioteca de software Apache Hadoop é um framework que permite o processamento distribuído de grandes conjuntos de dados através de clusters de computadores que usam modelos de programação simples. É projetado para escalar a partir de um único servidor para milhares de máquinas, cada uma oferecendo computação e de armazenamento local. Em vez de confiar em hardware para proporcionar alta disponibilidade, a biblioteca em si é concebida para detectar e tratar falhas na camada de aplicação, de modo a fornecer um serviço altamente disponível em cima de um cluster de computadores, cada um dos quais pode ser propenso a falhas. "- http://hadoop.apache.org/

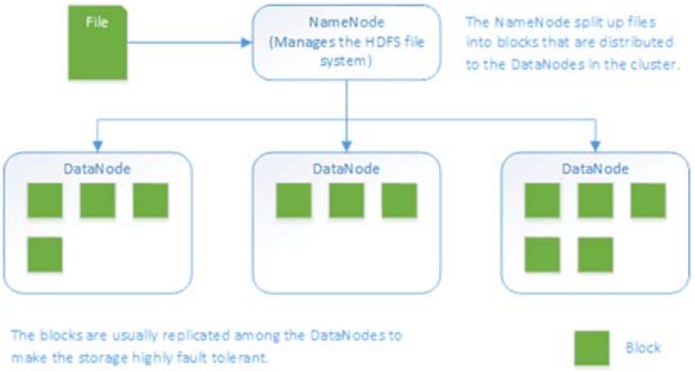
Correr este tipo de software num cluster com raspberry pi, pode parecer “absurdo”, dado o “parco” poder de computação e armazenamento do raspberry. No entanto, e, dependendo do volume de dados, até pode surpreender, uma vez que o consumo de energia deste dispositivo, é bastante baixo, assim como o custo é reduzido e se pensarmos em fazer isto com raspberry pi zero, ainda mais prático se torna! Pelo menos numa lógica de projecto de aprendizagem, ou cluster para testes, por exemplo! No meu caso, inicialmente usei para dados provenientes de sensores montados por mim, para me analisar e gerar informação, com base na informação proveniente dos sensores (a fazerem leituras de 5 em 5 segundos, 24 horas por dia, 7 dias por semana. De qualquer das formas o objectivo deste artigo é apenas apresentar a tecnologia e como a colocar a funcionar!

Apache Hadoop

O Apache Hadoop é constituído por um número de componentes e frameworks de código aberto que o tornam bastante flexível e modular. De forma resumida, encontra-se divido em duas partes:

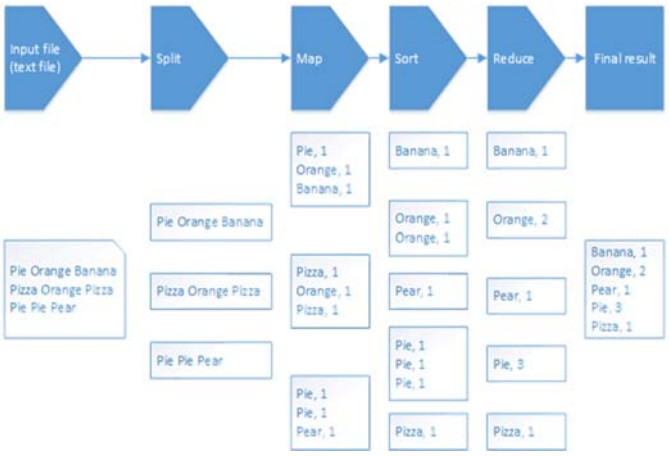
- **Armazenamento de dados (HDFS)** - *Hadoop Distributed File System*, projetado para correr em hardware de baixo custo sendo bastante tolerante a falhas. Os ficheiros são divididos em blocos que são replicados para os nós de dados (*DataNodes*). Por padrão os blocos têm um tamanho de 64MB e são replicados para 3 nós no cluster. Contudo estas definições podem ser ajustadas a necessidades específicas.

Visão geral da arquitectura de HDFS File System:



- **Processamento de dados** (*MapReduce*), é uma framework escrita em Java que é utilizada para criar aplicações que possam processar grandes volumes. Apesar de estar escrita na linguagem de programação Java existem outras linguagens disponíveis para escrever aplicações MapReduce. Tal como com HDFS foi projectada para ser tolerante a falhas e para trabalhar em ambientes de cluster de grande escala. A framework tem a capacidade de dividir dados de input em tarefas mais pequenas (*map tasks*) que podem ser executadas em processos paralelos. O Output das map tasks é então reduzido (*reduce task*) e guardado no sistema de ficheiros. Abaixo poderemos ver o fluxo MapReduce do programa exemplo WordCount que iremos utilizar mais tarde. O WordCount pega num ficheiro de texto como input, divide-o em partes mais pequenas e depois conta cada palavra e devolve ao output um ficheiro com uma contagem de todas as palavras dentro do ficheiro.

Visão geral do fluxo MapReduce (WordCount):



A Instalação

Não irei abordar em detalhes todos os aspectos da instalação do Raspbian Stretch Lite, uma vez que já escrevi sobre ela na edição nº 48 da PROGRAMAR, pelo que aconselho o leitor a consultar a secção "Preparação" do artigo da edição referida, disponível em https://www.revista-programar.info/artigos/criar-um-cluster-de-processamento-paralelo-mpi-com-raspberrys/.

Feita a configuração base, o leitor deve ajustar os nomes dos hosts (*hostname*), de forma a que cada nó do novo cluster tenha um nome que o permita identificar facilmente. Neste caso começamos com o nome "node1", para o primeiro nó e assim sucessivamente.

Terminadas estas configurações iniciais, reiniciamos o raspberry pi, com o comando:

```
sudo shutdown -r now
```

Uma vez reiniciado o raspberry e já com acesso a ele por ssh, procede-se às configurações de rede, recorrendo ao editor de texto GNU nano. Para tal começamos por editar o ficheiro /etc/network/interfaces, recorrendo ao seguinte comando:

```
sudo nano /etc/network/interfaces
```

Dentro do editor, colocamos o ficheiro de forma a ficar idêntico ao seguinte, e gravamos.

```
iface eth0 inet static
address 192.168.0.110
netmask 255.255.255.0
gateway: 192.168.0.1
```

Terminada a edição e gravado o ficheiro, editamos o ficheiro /etc/resolv.conf, para configurar os nameservers correctamente.

Reiniciamos o raspberry pi novamente e assim que esteja reiniciado, podemos continuar com a instalação do restante software.

Antes de mais instalações começamos por instalar o java, pois a distribuição Raspbian Jessie Lite, não traz java instalado. Para tal começamos por obter a chave GPG para o pacote java, recorrendo ao comando seguinte:

```
sudo apt-key adv --recv-key --keyserver keyser-
ver.ubuntu.com EEA14886
```

De seguida editamos o ficheiro de fontes, /etc/apt/source.list.file para acrescentar as fontes necessárias à instalação do java.

```
sudo nano /etc/apt/sources.list
```

E acrescentamos as seguintes linhas no final do ficheiro:

```
deb http://ppa.launchpad.net/webupd8team/java/
ubuntu trusty main
deb-src http://ppa.launchpad.net/webupd8team/java/
ubuntu trusty main
```

Feito isto, procede-se com a instalação do java, recorrendo aos seguintes comandos:

```
sudo apt-get update
```

Posto isto, verificamos se tudo correu como esperado, utilizando o seguinte comando:

```
java -version
```

Que deverá ter um output idêntico ao seguinte:

```
pi@raspberrypi:~ $ java -version
java version "1.8.0_73"
Java(TM) SE Runtime Environment (build 1.8.0_73-b02)
Java HotSpot(TM) Client VM (build 25.73-b02, mixed mode)
pi@raspberrypi:~ $
```

De seguida verificamos a versão do compilador java com o comando seguinte:

```
javac -version
```

```
pi@raspberrypi:~ $ javac -version
javac 1.8.0_73
pi@raspberrypi:~ $
```

Posto isto, avançamos para a preparação do grupo de utilizadores, e conta, recorrendo aos seguintes comandos:

```
sudo addgroup hadoop
sudo adduser --ingroup hadoop hduser
sudo adduser hduser sudo
```

Com esta etapa terminada, avançamos para a configuração das chaves SSH, com palavra passe em branco, para permitir que os nós hadoop possam comunicar entre si, sem estarem constantemente a solicitar a introdução da palavra passe. Para tal, executamos os comandos seguintes:

```
su hduser
mkdir ~/.ssh
ssh-keygen -t rsa -P ""
cat ~/.ssh/id_rsa.pub > ~/.ssh/authorized_keys
```

Verifica se o hduser pode fazer login no SSH

```
su hduser
ssh localhost
```

Instalação do Hadoop

Neste momento, temos tudo pronto para instalar o Apache Hadoop no primeiro nó. A primeira tarefa para se instalar o hadoop, é proceder ao download do mesmo.

```
cd ~/
wget http://mirrors.up.pt/pub/apache/hadoop/
common/hadoop-2.8.1/hadoop-2.8.1.tar.gz
sudo mkdir /opt
sudo tar -xvzf hadoop-2.8.1.tar.gz -C /opt/
cd /opt
sudo mv hadoop-2.8.1 hadoop
sudo chown -R hduser:hadoop hadoop
```

Após a instalação do Hadoop e ainda antes de ser utilizado, são necessárias efectuar algumas configurações. Existem diversas opções disponíveis para a configuração que se segue, no entanto, por opção a configuração será feita no utilizador hduser. Para tal, adiciona-se o seguinte ao ficheiro bashrc que se encontra no directório home do utilizador hduser.

```
export JAVA_HOME=$(readlink -f /usr/bin/java |
                                sed "s:bin/java::")
export HADOOP_INSTALL=/opt/hadoop
export PATH=$PATH:$HADOOP_INSTALL/bin
```

Executada a tarefa, fazemos logout do utilizador pi e login com o utilizador hdusr, para que possamos verificar se o executável do hadoop está acessível fora do fomr /opt/hadoop/bin, recorrendo aos seguintes comandos:

```
su hduser
hadoop version
```

O output do comando deve ser algo parecido com:

```
hduser@node1 /home/hduser $ hadoop version
Hadoop 2.8.1
```

Vamos agora para a configuração seguinte, que é a configuração de variáveis de ambiente do Apache Hadoop. Para tal, recorremos ao nano, como super-user, para editar o ficheiro /opt/hadoop/conf/hadoop-env-sh, e descomentamos algumas linhas. Para editar executamos:

```
sudo nano /opt/hadoop/conf/hadoop-env-sh

E descomentamos as linhas seguintes:

# The java implementation to use. Required.
export JAVA_HOME=$(readlink -f /usr/bin/java
                            | sed "s:bin/java::")

# The maximum amount of heap to use, in MB.
                                  Default is 1000.
export HADOOP_HEAPSIZE=250

# Command specific options appended to HA-
DOOP_OPTS when specified
export HADOOP_DATANODE_OPTS="-
Dcom.sun.management.jmxremote $HA-
DOOP_DATANODE_OPTSi -client"
```

Findas as configurações anteriores, está na hora de configurar o próprio Apache Hadoop. Para tal, usamos o editor nano, para editar o ficheiro /opt/hadoop/conf/core-site.xml, e colocamos o seguinte no ficheiro:

```
<configuration>
  <property>
    <name>hadoop.tmp.dir</name>
    <value>/hdfs/tmp</value>
  </property>
  <property>
    <name>fs.default.name</name>
    <value>hdfs://localhost:54310</value>
  </property>
</configuration>
```

O mesmo se aplica para o ficheiro mapred-site.xml que se encontra no mesmo directório, e o conteúdo deve ser idêntico ao seguinte:

```
<configuration>
  <property>
    <name>mapred.job.tracker</name>
    <value>localhost:54311</value>
  </property>
</configuration>
```

De igual modo para o ficheiro hdfs-site.xml:

```
<configuration>
  <property>
    <name>dfs.replication</name>
    <value>1</value>
  </property>
  <property>
    <name>dfs.block.size</name>
    <value>5242880</value>
  </property>
</configuration>
```

Com todas estas configurações feitas, está na hora de criar o sistema de ficheiros (file system) para o Apache Hadoop (HDFS). Para tal executamos os comandos seguintes, pela seguinte ordem:

```
sudo mkdir -p /hdfs/tmp
sudo chown hduser:hadoop /hdfs/tmp
sudo chmod 750 /hdfs/tmp
hadoop namenode -format
```

E agora iniciamos os serviços do Apache Hadoop! Para tal temos de trocar de utilizador para o hduser, caso não estejamos a usar esse mesmo utilizador.

```
/opt/hadoop/bin/start-dfs.sh
/opt/hadoop/bin/start-mapred.sh
```

Uma vez iniciados os serviços, está na hora de verificar se tudo está a correr como deveria. Para tal executa-se o comando jps, que deve produzir um outpoot parecido com:

```
16640 JobTracker
16832 Jps
16307 NameNode
16550 SecondaryNameNode
16761 TaskTracker
16426 DataNode
```

Com isto tudo terminado, chegou a hora de executar o primeiro teste! Neste caso será apenas com um nó. Para realizar o teste, copiamos um ficheiro qualquer, para o directório /opt/hadoop/, no caso será usado o ficheiro da licença do hadoop (license.txt), mas pode ser usado outro qualquer. Os comandos para o teste são os seguintes:

```
hadoop dfs -copyFromLocal /opt/hadoop/
                          LICENSE.txt /license.txt
hadoop jar /opt/hadoop/hadoop-examples-1.2.1.jar
          wordcount /license.txt /license-out.txt
```

E para ver os resultados, copiamos o ficheiro de volta, para o sistema de ficheiros local (lembro que copiamos o ficheiro para o sistema de ficheiros do apache hadoop.

```
hadoop dfs -copyToLocal /license-out.txt ~/
```

E agora podemos abrir o ficheiro de resultados para ver-mos os mesmos. O dispositivo deve apresentar cada palavra do ficheiro da licença e o seu numero de ocorrências nesse mesmo ficheiro. Para tal vamos novamente recorrer ao nano, para abrir o ficheiro part-r-00000.

```
nano ~/license-out.txt/part-r-00000
```

Chegados a este ponto, está na hora de preparar os restantes nós do cluster!

Antes de passar para a etapa seguinte que consiste em desligar deste "nó" para criar uma imagem do cartão, temos de executar mais algumas configurações que nos irão poupar tempo futuramente. Para tal executamos o comando seguinte e colocamos os endereços IP dos nós que pretendemos usar.

```
sudo nano /etc/hosts
```

```
192.168.0.101 node1
192.168.0.102 node2
192.168.0.103 node3
192.168.0.104 node4
192.168.0.105 node5
```

Seguidamente vamos definir o nome do nosso nó principal! Para tal editamos o ficheiro masters que se encontra em /opt/hadoop/conf

```
sudo nano/opt/hadoop/conf/masters
```

E dentro do ficheiro colocamos o nome do nó:

```
node1
```

De seguida editamos o ficheiro core-site.xml que se encontra em /opt/hadoop/conf e colocamos o seguinte no ficheiro:

```
<configuration>
  <property>
    <name>hadoop.tmp.dir</name>
    <value>/hdfs/tmp</value>
  </property>
  <property>
    <name>fs.default.name</name>
    <value>hdfs://node1:54310</value>
  </property>
</configuration>
```

Editamos também o ficheiro mapred-site.xml e colocamos o seguinte no ficheiro:

```
<configuration>
  <property>
    <name>mapred.job.tracker</name>
    <value>node1:54311</value>
  </property>
</configuration>
```

Por fim, mesmo antes de gerar a imagem, resta-nos limpar o Hadoop FileSystem (HDFS). Para tal executamos o comando seguinte:

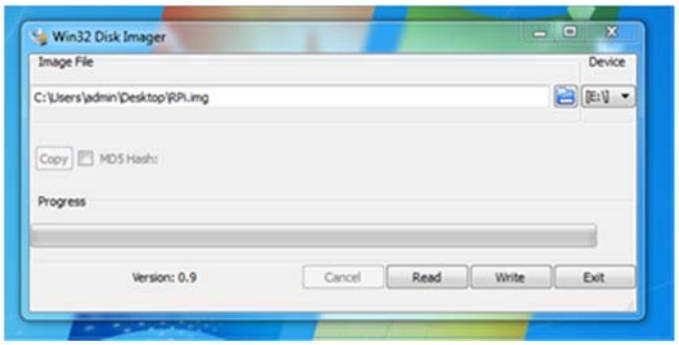
```
rm -rf /hdfs/tmp/*
```

Posto isto avançamos para o passo seguinte que será desligar o nó actual e fazer a imagem que nos irá permitir criar todos os outros nós com menos esforço.

Começamos por desligar o nó actual, retiramos o cartão de memória e fazemos uma imagem do mesmo. Neste caso, será usado o sistema operativo Windows e como tal será usado o software Win32DiskImager, para fazer a imagem do cartão de memória.

O Win32DiskImager pode ser descarregado em: https://sourceforge.net/projects/win32diskimager/.

Uma vez descarregado e instalado, podemos proceder com a produção de uma imagem do cartão de memória, recorrendo ao Win32DiskImager. Escolhemos uma pasta onde vai ficar o ficheiro da imagem (figura seguinte)



Agora escolhemos a unidade correspondente ao cartão de memória, na combobox referente a "device".

De seguida basta carregar "read", para a imagem ser feita.

Para escrever esta mesma imagem num outro cartão de memória o processo é idêntico, mas no sentido inverso, ou seja, escolhemos a imagem e a unidade do cartão de memória de destino, e clica-se no botão "write".

Com a imagem no cartão temos uma base para fazer todos os restantes nodes, basta editar as configurações, alterando o nome do host recorrendo tal como anteriormente explicado a um editor de texto e editando o ficheiro /etc/hostname. De seguida editamos o ficheiro /etc/network/interfaces e colocamos o endereço de ip correspondente ao nó em causa e reiniciamos o raspberry.

Neste momento podemos prosseguir com as restantes configurações do nó, começando por editar o ficheiro slaves que se encontra em /opt/hadoop/conf/ recorrendo a um editor de texto e colocando no ficheiro o seguinte:

```
node1
node2
node3
```

Repetimos os procedimentos anteriores para todos os nós que queiramos criar. Eu apenas fiz com 5, no entanto o leitor pode fazer com quantos pretender.

Testar o cluster

Agora sim, após todas as etapas, chegamos à fase mais apetecível de todo este artigo, vamos então, testar o Cluster! Com todos os raspberry's ligados a um switch, e ligados entre si, chegou o momento de começar o teste.

O primeiro passo será testar se os nós comunicam entre si por ssh, sem necessidade de password. Contudo, antes de procedermos a estas ligações, necessitamos de copiar o ficheiro id_sra.pub do primeiro nó (node1), para a directoria /home/hduser/.ssh/authorized_keys de cada um dos nós do cluster. Existem diversas formas de descarregar o ficheiro para cada nó, pelo que se deixa ao cargo do leitor escolher a que entender por melhor. Feito isto, procedemos para o teste.

Para tal executamos os comandos seguintes, no nó principal:

```
su hduser
ssh node1
exit
ssh node2
exit
ssh node3
exit
ssh node4
exit
ssh node5
exit
```

Com isto feito temos de formatar o HDFS de cada nó e de seguida iniciar os serviços. Para tal usamos os comandos seguintes no primeiro nó (node1):

```
hadoop namenode -format
/opt/hadoop/bin/start-dfs.sh
/opt/hadoop/bin/start-mapred.sh

jps
```

Para os restantes nós o comando será o seguinte:

```
jps
```

Neste momento se não foi indicado nenhum erro, podemos confirmar o estado de cada nó acedendo ao interface web de cada um, nos endereços http://node1:50030 e http://node1:50070 .

Neste momento estamos prontos a usar o cluster Apache Hadoop!

Antes de começar a usar o cluster vamos preparar o primeiro nó (node1) para funcionar como nó mestre (*master node*). Para tal editamos o ficheiro slaves do primeiro nó (node1) e removemos a linha onde consta "node1" e de seguida reiniciamos o cluster com os comandos seguintes:

```
stop-mapred.sh
stop-dfs.sh
start-dfs.sh
start-mapred.sh
```

E por fim, hora de testar! Para tal e apenas a título de exemplo executamos um teste usando o programa wordcount, e o ficheiro de exemplo mediumfile.txt que é disponibilizado pelo Apache Hadoop.

```
hadoop jar /opt/hadoop/hadoop-examples-1.2.1.jar
wordcount /mediumfile.txt /mediumfile-out.txt
```

Conclusão

O Apache Hadoop é uma ferramenta extremamente útil, para trabalho com grandes volumes de dados e oferece muitas mais possibilidades do que as abordadas ao longo deste artigo. A instalação do Apache Hadoop no raspberry, não traz particulares vantagens no desempenho, quando comparado com equipamentos mais poderosos, como computadores pessoais ou até mesmo servidores. No entanto dado o baixo custo dos raspberry e o seu baixo consumo, são uma boa alternativa para quem pretende explorar o Apache Hadoop, ou por exemplo ter o seu próprio cluster hadoop, para análises de dados (maiores, menores, cada um terá um uso diferente). Existem diversas optimizações que ficaram por fazer, mas que creio que saíam claramente fora do âmbito do artigo, que apenas pretende apresentar a tecnologia e um exemplo simples de colocar em funcionamento.

Espero que o leitor se divirta a montar o seu próprio cluster hadoop!

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 PROGRAMAR

Correndo uma Aplicação Web Java em Azure, passo a passo

Junit

Criar uma aplicação móvel com jQuery Mobile

Lua – Linguagem de Programação – Parte 13

Tipos de dados int e variantes na linguagem C

Feed RSS em C# .NET Core no Azure Web App em Linux

Correndo uma Aplicação Web Java em Azure, passo a passo

Como alguns vós sabem, eu pertenço à organização da Comunidade Netponto e há uns tempos organizamos o Visual Studio Launch Party nas instalações da Microsoft Portugal. Nesse evento, tivemos o Miguel Caldas a realizar o Keynote onde ele, entre muitas mensagens, destacou:

"Nós queremos correr o vosso software"

É uma mensagem simples, mas poderosa, que mostra a mudança da postura perante outras tecnologias que não sejam .NET, como seja o PHP ou o Java.

Já há algum tempo que estou tentado em realizar algumas experiencias envolvendo o desenvolvimento de uma aplicação utilizando Java como linguagem de programação.

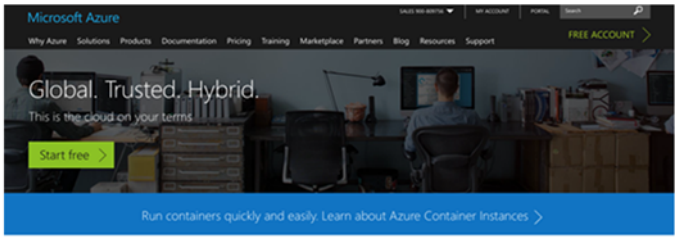
Quem já trabalha em aplicações com Java há algum tempo sabe os desafios que trás consolidar todas as partes para que a aplicação flua sem problemas.

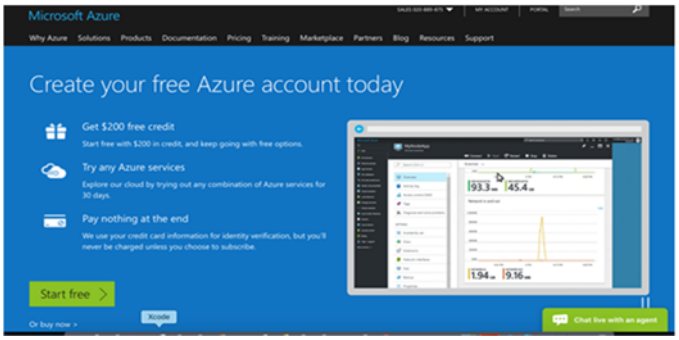
Neste artigo vou descrever os passos que dei ao longo do processo, referindo os desafios que encontrei e como os ultrapassei. Saliente o propósito deste artigo é colocar uma aplicação web em Java a correr no Azure e não as boas práticas do código em si, apesar de algumas serem aplicadas inconscientemente.

Pré-Requisitos

Vamos lá começar....

Antes de tudo precisamos de uma conta no Azure[1] e nada mais simples que aceder ao site e criar uma conta:





E preencher os dados. Simples, certo!?

Após o registo, e validação da conta, recomendo que façam login e deem uma volta pelas opções do portal do Azure. Apenas para ganharem sensibilidade para as opções e os produtos que estão disponíveis. Alerto, desde já, que existem alguns serviços que precisam de um upgrade na conta, pelo que têm alguns custos. É uma questão de analisarem.

Software Utilizado

Para concretizar o desenvolvimento da aplicação de demonstração e publicar no Azure foi utilizado o seguinte conjunto de aplicações:

- Spring Tool Suite 3.9.0 Release
- JDK 1.8.0_122
- Tomcat 8.0.24
- Docker
- MySql (docker image: latest)
- FileZilla 3.27.1

Poderão utilizar o IDE de vossa preferência ou utilizar a instancia de MySql que já tenham instalada. Pessoalmente uso Docker sempre que possível nas minhas demonstrações, pois não preciso de me preocupar muito com a sua configuração ou poluir a minha máquina com o software que tenha que instalar.

Ao investigar pela Internet, e em particular no site de documentação do Azure[2], poderão encontrar referências em instalar o Azure Toolkit[3] para integração com a plataforma. Pela minha experiência, os benefícios de instalar o plugin não são muitos, trazendo um overhead no arranque do IDE e requer um conhecimento de "magia negra" conseguir atingir o objetivo, pelo que opto por manter o IDE mais simples e "limpo" possível para construir a demonstração e mostrar o propósito.

A Aplicação de Exemplo

O propósito da aplicação é muito simples. É uma aplicação do template "Dynamic Web Project", sem configurações adicionais em especial.Figure

Figur1 Dynamic Web Project Template

O projeto é composto por uma página JSP (index.jsp) com um link para invocar uma servlet (HelloWorld.java) que invoca uma chamada à base de dados (SessionConnector.java), processo o resultado e devolve uma lista com todos os resultados da tabela Sessions, expondo o resultado noutra JSP (ListSessions.jsp).

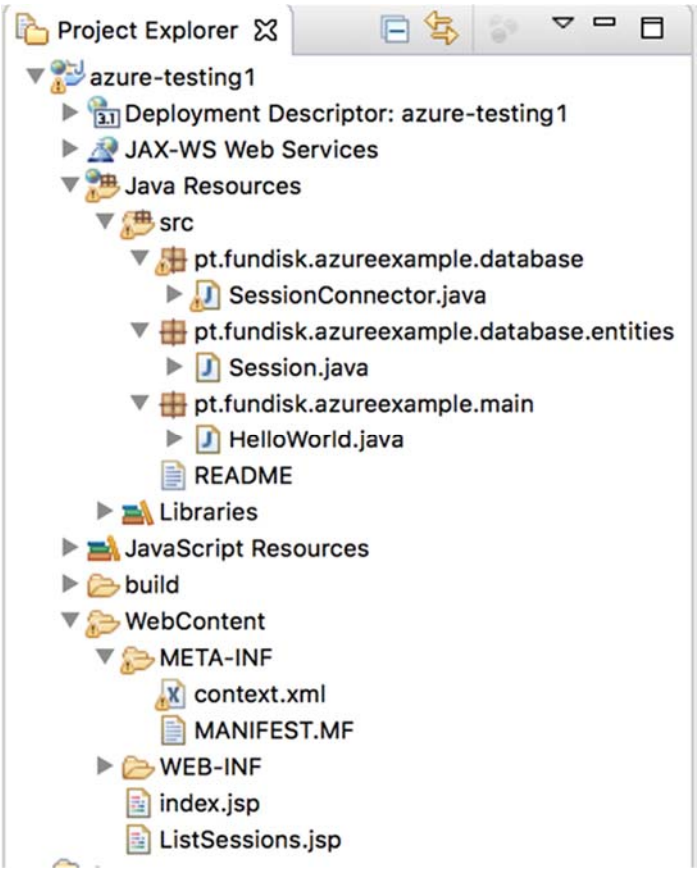


Figure 2: Estrutura do Projeto

Nada de especial.

Algumas considerações

Existem diversas formas de fornecer os dados da ligação à base de dados, sendo as mais utilizadas ter a connection string no código em si ou a ter em ficheiros de propriedades. Para efeitos de demonstração ou para os nossos pequenos projetos poderá não haver problema, mas quando as coisas crescem ou mudam de âmbito é um aborrecimento ter as coisas assim.

Em alternativa, podemos configurar no servidor aplicacional um datasource/resource para a base de dados, e no projeto referenciamos o nome desse datasource/resource para aceder à base de dados e fazer o que queremos.

A grande vantagem desta abordagem é o facto de precisarmos de mexer nas connections strings quando exportamos a aplicação da nossa maquina de desenvolvimento para o servidor onde vai estar alojada.

Outro ponto em ter em atenção, diz respeito ao driver JDBC, neste caso MySql, que deve ser disponibilizado no servidor (no segundo caso) ou deve seguir com o pacote final (no primeiro caso). Parece evidente, no entanto não imaginam o quanto é frequente esquecerem-se dos drivers e quando a aplicação não funciona:

“Não percebo, funciona na minha máquina”.

É-vos familiar?!

Configurações TomCat

Para configurar um datasource/resource no Tomcat, basta realizar duas operações:

- Copiar o driver para a pasta **$CATALINA_BASE/lib**
- Editar o ficheiro **$CATALINA_BASE/conf/context.xml**

Neste ficheiro configura-se o datasource:

```
<Context>
  ...
  <Resource
    name="[JNDI NAME]"
    auth="Container"
    type="javax.sql.DataSource"
    maxActive="50"
    maxIdle="30"
    maxWait="10000"
    driverClassName="com.mysql.jdbc.Driver"
    username="[USERNAME]" password="[PASSWORD]"
    url="jdbc:mysql://[SERVER]:[PORT]/
                                    [DATABASE]"/>
  ...
</Context>
```

Como é que funciona? O Tomcat ao arrancar importa todas as bibliotecas de **$CATALINA_BASE/lib** para o seu classpath e depois carrega as diversas configurações (entre as quais **$CATALINA_BASE/conf/context.xml**) que ficam disponíveis para todas as aplicações que estiver a servir. A aplicação só tem que invocar o JNDI respetivo para proceder à ligação à base de dados.

Agora vamos editar o nosso web.xml e adicionamos a seguinte entrada:

```
<resource-ref>
  <description>MySQL Datasource example</
description>
  <res-ref-name>jdbc/community_events</res-ref-
name>
  <res-type>javax.sql.DataSource</res-type>
  <res-auth>Container</res-auth>
</resource-ref>
```

Base de Dados

Para a base de dados, usei o container Docker de MySql:

```
docker run --name MySqlContainerForTests -p
3308:3306 -e MYSQL_ROOT_PASSWORD=my-secret-pw -d
mysql
```

Depois da base de dados estar disponível, acedi à base de dados com o MySQL Workbench e criei a base de dados, a tabela para o exemplo e inseri dados.

Dynamic Web Project

Para o projeto em si só é preciso a biblioteca jstl-1.2.jar, para se poder utilizar taglibs nas JSPs. Não é relevante em si para este projeto, mas como não estamos a usar Maven para gerir as dependências, precisamos de preocupar com identificação das bibliotecas e a sua exportação especifica.

Moving on ...

Como mencionei no inicio, o projeto tem três classes java sendo que a mais relevante é a classe ***SessionConnector***.

Nesta classe é onde obtemos o datasource, é realizada a operação na base de dados e processados os resultados.

A obtenção do datasource é realizado no construtor da classe:

```
private static final String JNDI_DATASOURCE =
"java:comp/env/jdbc/community_events";

private DataSource ds;

public SessionConnector() {
  try {
    Context ctx = new InitialContext();
      ds = (DataSource)ctx.lookup(JNDI_DATASOURCE);
  } catch (NamingException e) {
    e.printStackTrace();
  }
}
```

Btw, nunca se esqueçam de tratar as exceções. :-)

O seguinte método faz o resto do trabalho, mantendo as coisas simples:

```
public Iterable<Session> getSessionList()
{
  List<Session> sessions = new LinkedList<Session>
();
  try {
    Connection con = ds.getConnection();
    Statement stm = con.createStatement();
    ResultSet rs = stm.executeQuery
(GET_ALL_SESSIONS);
    processResultSet(sessions, rs);
    rs.close();
    stm.close();
  } catch (SQLException e) {
    e.printStackTrace();
  }
  return sessions;
}
```

Decerto reparam que não estou a fechar a Connection e poderão pensar que deveria o fazer. Estamos perante um caso de separação de responsabilidades, se fosse eu explicitamente a abrir a ligação, então deveria ser eu a fechar a mesma. Porém, neste caso, estou a pedir a uma entidade (Datasource) que me disponibilize uma Connection para eu realizar o meu trabalho e como tal não a fecho e ela volta para a pool de ligações.

A outra classe java (HelloWorld.java) é uma servlet que no método GET invoca a classe anterior e devolve os resultados para a JSP.

```
protected void doGet(...) ...{
 SessionConnector sc = new SessionConnector();
 Iterable<Session> sessions = sc.getSessionList();
  request.setAttribute("sessions", sessions);
  request.getRequestDispatcher
  ("ListSessions.jsp").forward(request, response);
}
```

Por fim temos as JSPs:

- index.jsp: com um link para invocar a servlet.

```
<body>
Hello Azure
<a href="HelloWorld">List of sessions</a>
</body>
```

- ListSessions.jsp: apresenta o resultado da bd.

```
<body>
<table>
  <thead>
    <tr>
    <td>RECORD_ID</td>
    <td>FULLNAME</td>
    <td>EMAIL</td>
    <td>SESSION_TITLE</td>
    <td>SESSION_DESCRIPTION</td>
    <td>OBSERVATIONS</td>
    </tr>
  </thead>
  <tbody>

    <c:forEach var="session" items="${sessions}">
    <tr>
    <td>${session.recordId}</td>
    <td>${session.fullName}</td>
    <td>${session.email}</td>
    <td>${session.sessionTitle}</td>
    <td>${session.sessionDescription}</td>
    <td>${session.observations}</td>
    </tr>
    </c:forEach>
  </tbody>
</table>
</body>
```

Para utilizar a taglib, tem que incorporar a seguinte instrução no inicio do ficheiro:

```
<%@ taglib uri = "http://java.sun.com/jsp/jstl/
                                core" prefix = "c" %>
```

E o resultado é o seguinte:

Figure 3: index.jsp

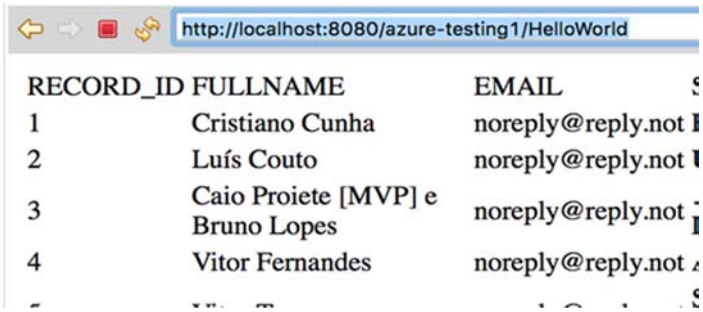


Figure 4: ListSessions.jsp

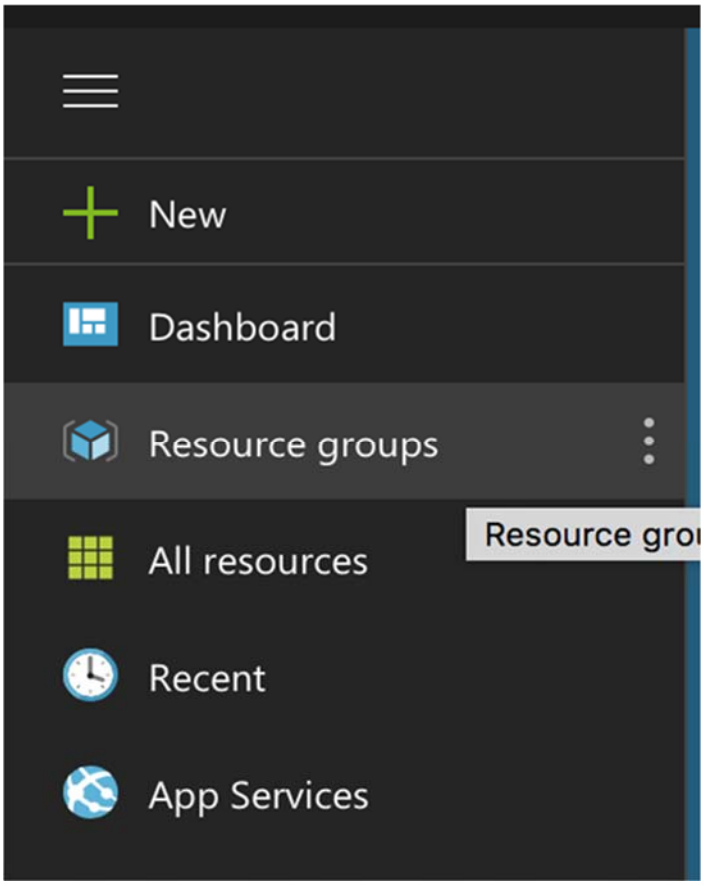
E a secção mais simples está feita. Agora vamos publicar no Azure. Simple, certo?

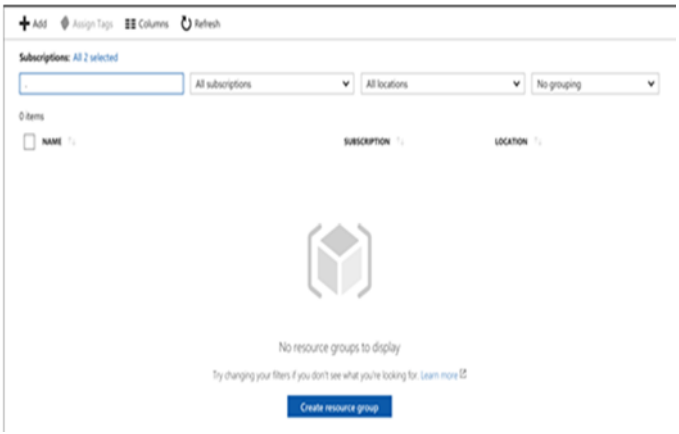
Benvindos ao Microsoft Azure –Parte 1

Vou ser honesto, pela minha experiência, não é simples nem agradável o caminho de publicar a primeira aplicação java no Azure e que precise de ligação a base de dados MySql. Foram precisos mais do que dois cliques para conseguir a aplicação a funcionar. O pessoal esta trabalhar arduamente para integra mais tecnologias no portal, mas (na minha humilde opinião) ainda tem um longo percurso a seguir.

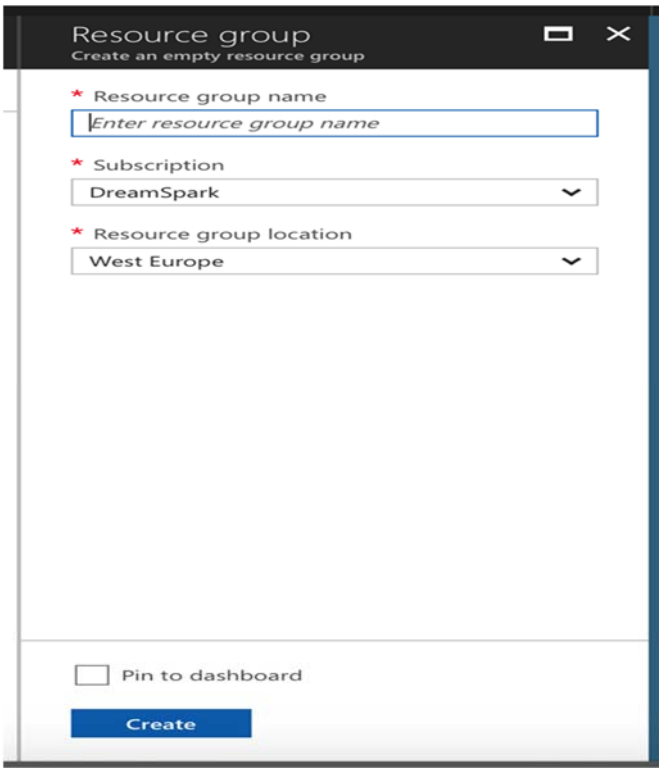
Para publicar a aplicação é necessário ter uma conta no Azure, como mencionado no inicio.

A primeira coisa que vamos fazer, depois do login, é criar um resource group, para que a nossa aplicação e todos os seus recursos estejam arrumados no mesmo grupo.

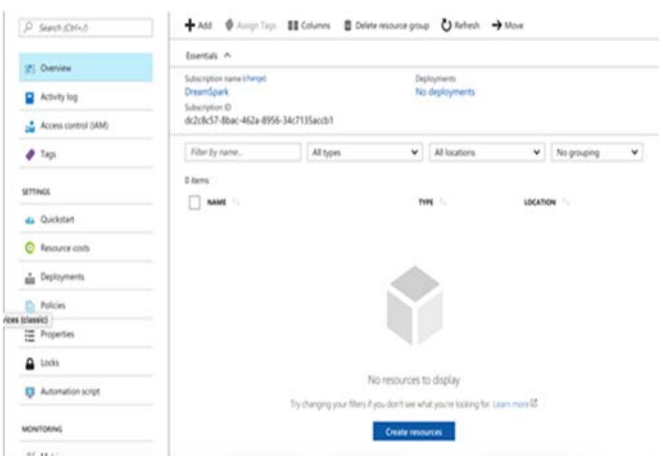




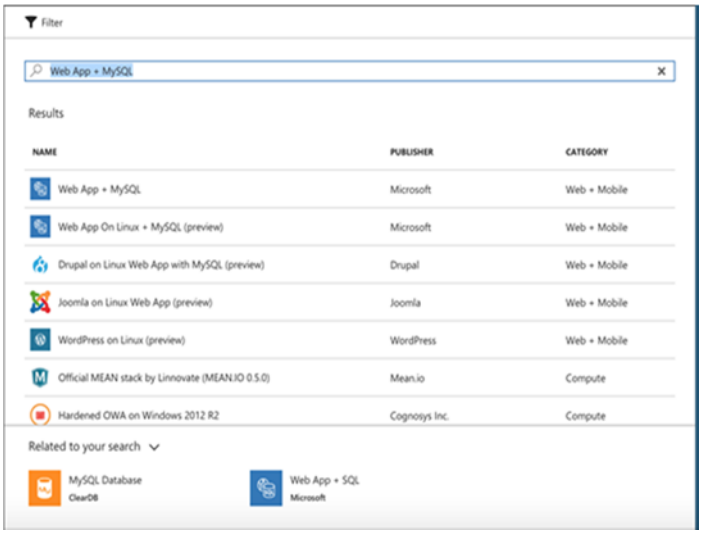
Clicamos no botão ***Add*** e preenchemos os dados.



Após a conclusão vamos para a home page do nosso resource group.

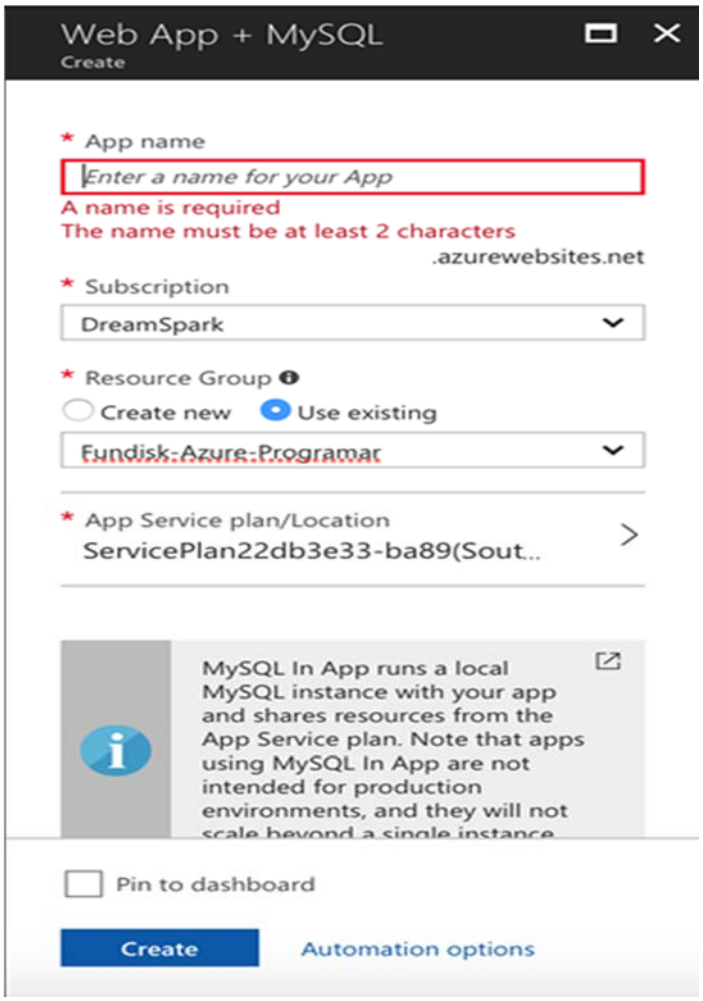


Agora vamos criar um resource do tipo "Web App + MySQL" e para tal voltamos a clicar em ***Add*** e pesquisamos por "Web App + MySQL".



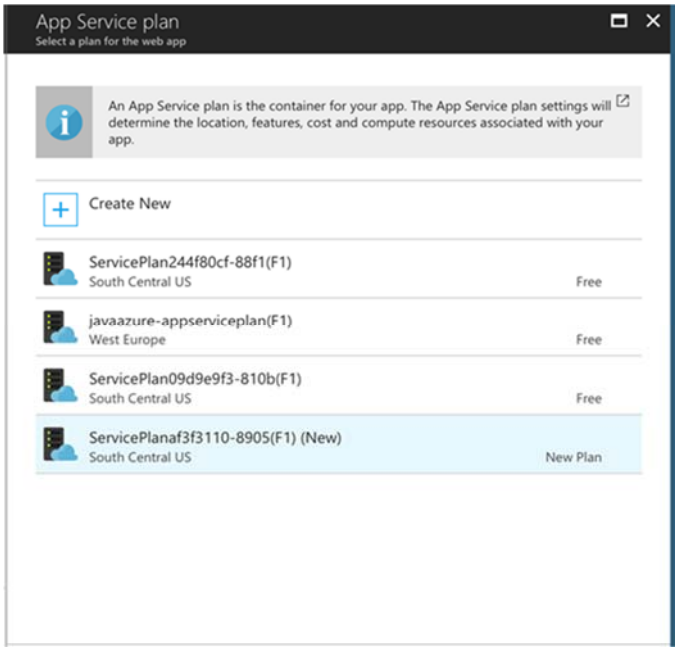
Vamos escolher logo o primeiro resultado e clicamos em ***Create***.

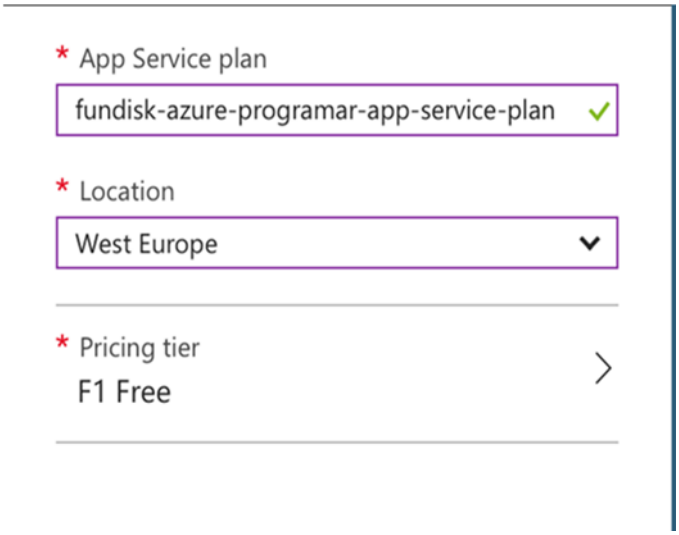
Agora vamos preencher os seguintes dados do formulário.



ANTES de clicar em ***CREATE***, depois de escolhermos o endereço do nosso site, selecionamos ***"App Service plan"*** para podermos escolher a localização do nosso site e dar um nome mais legível.

Neste ecrã, carregamos em ***"Create New"*** e preenchemos os dados às nossas necessidades.





E carregamos em ***OK.***

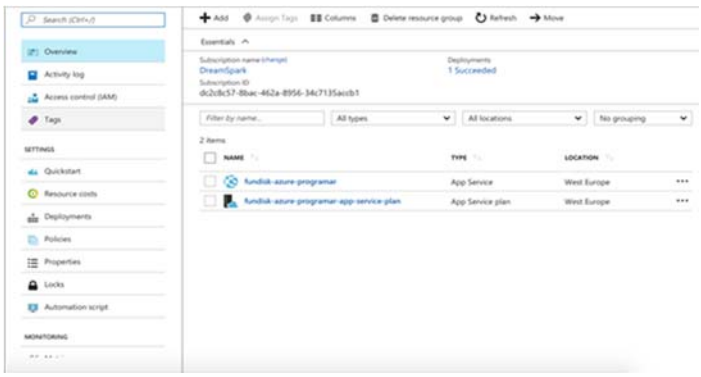
Finalmente carregamos em ***"Create"***. Ao fim de alguns minutos ficamos com o ambiente criado e somos notificados pelo símbolo do sino no topo da página.

Esta primeira parte está concluída. Criamos um resource group para a nossa aplicação e criar uma instancia de webapps com base de dados MySql para servir a nossa aplicação.

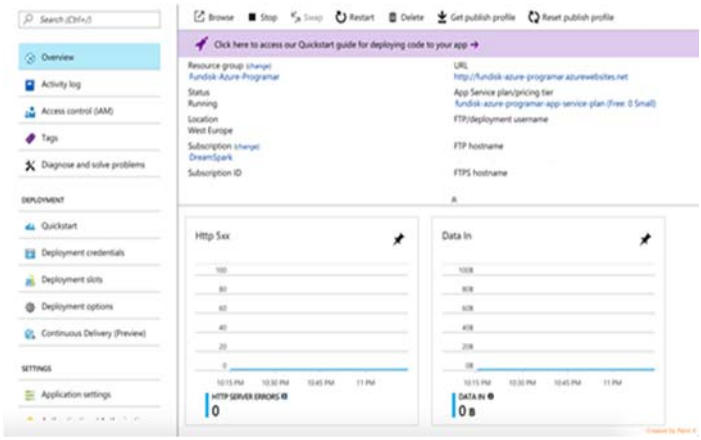
Agora vamos configurar a instância para servir a aplicação que previamente criamos.

Benvindos ao Microsoft Azure –Parte 2

Uma vez concluído, selecionamos novamente Resource Groups no menu e selecionamos o resource group que acabamos de criar. E podemos verificar os recursos que estão associados ao grupo.



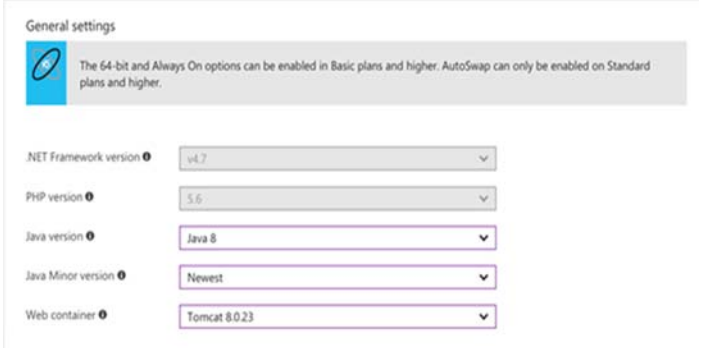
Nesta altura vamos escolher o nosso "App Service" e entramos no painel de controlo do nosso "servidor" aplicacional.



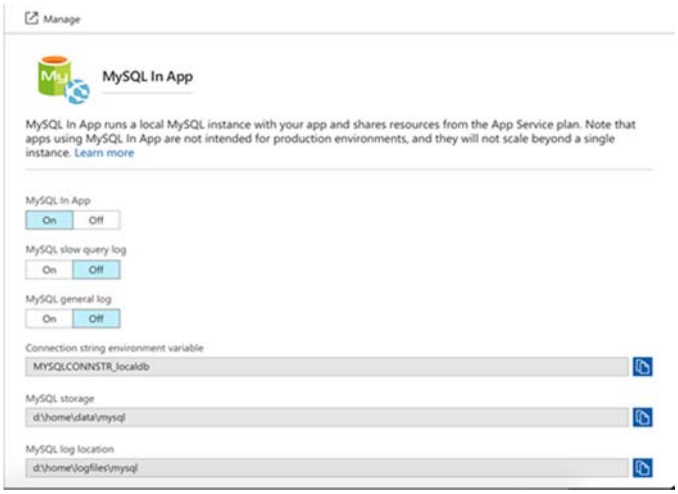
Por omissão, as webapps são em .NET mas como a nossa aplicação é Java vamos ter que reconfigurar.

Depressa vão descobrir que a caixa de pesquisa é o vosso melhor amigo, por isso vamos começar a utilizar desde já.

Pesquisem "**Application settings**" e cliquem no resultado. No painel de opções alterem para as configurações desejadas e carreguem em ***"Save"***.



Agora vamos tratar da base de dados. Voltamos à caixa de pesquisa e pesquisamos por "**MySQL In App**" e selecionamos o resultado.

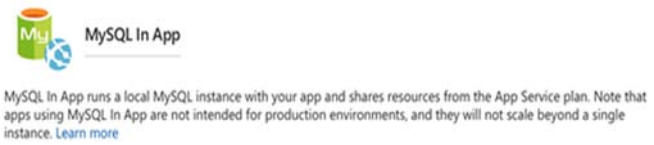


Aqui podemos confirmar que temos uma base de dados ***MySQL in App***, a localização da base de dados e a connection string dela.

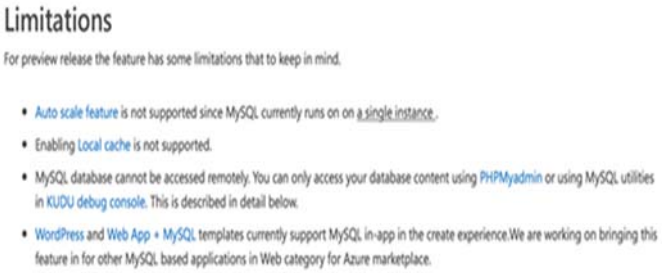
Aqui começam os primeiros desafios.

Desafio #1

O primeiro "alerta" esta no inicio da página:



E torna-se ainda mais desafiante se seguirmos o link indicado:

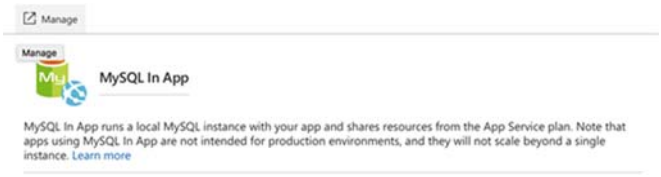


Não é expectável que a utilização desta base de dados seja de longa duração e utilização. Para testar/brincar serve mas para algo mais a "sério" há considerar outras alternativas.

Desafio #2

Precisamos de construir a base de dados para a nossa aplicação e como não podemos aceder remotamente à instância da base de dados vamos ter que utilizar o **PHPMyadmin.**

Para acedermos à instância de PHPMyadmin clicamos no botão **"Manage"** no topo da página.



E vamos para à pagina de login:

Eis que surge a questão:

"**Quais são as credenciais?** "

Houston, we have a problem....

É nesta altura que vamos começar a utilizar o Filezilla, mas antes precisamos de saber quais são os dados de ligação ao servidor para podermos configurar.

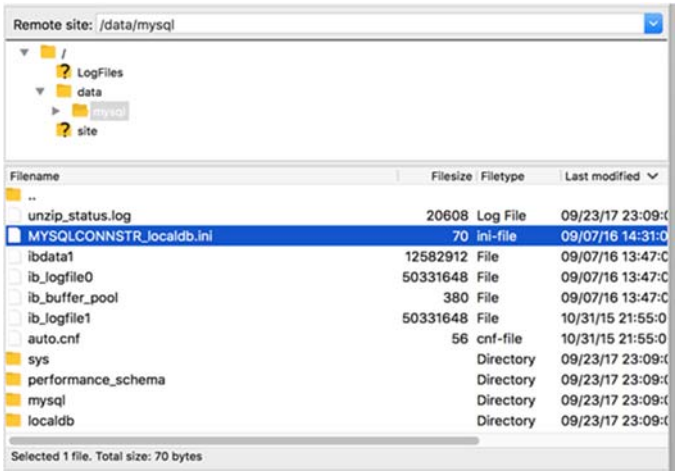
Vamos começar por definir credenciais para aceder. Para configurar, vamos à caixa de pesquisa, pesquisamos por "**Deployment credentials**" e selecionamos o resultado.

Aqui podemos criar um utilizador e definir a password do utilizador.

Voltamos à nossa caixa de pesquisa e pesquisamos "**Properties**" e clicamos no resultado. Na página seguinte temos todos os dados que necessitamos para configurarmos o cliente FTP/FTPS para aceder ao servidor.

Nesta altura abrimos o FileZilla e configuramos um novo servidor com os dados obtidos na página anterior. Após a configuração clicamos em "**Connect**" e estamos dentro do servidor.

Então vamos obter as credenciais de acesso à base de dados. Vamos à localização **"/data/mysql"**, abrirmos o ficheiro "**MYSQLCONNSTR_localdb.ini**" e encontramos as credenciais de acesso.



Voltamos à pagina do PHPMyadmin e introduzimos as credenciais.

Para grande espanto meu, deparei-me com esta imagem:

E roguei muitas pragas... :-)

Long story short. Não se preocupem com a mensagem, apenas quer dizer que o servidor não está a correr, pelo que não se consegue ligar para se autenticar.

Por omissão os servidores não estão ativos para poupar recursos e evitar gastos "acidentalmente". Mas é

relativamente simples ativar os servidores ... Após saber como se faz.

Vamos verificar se o servidor está ou não activo.

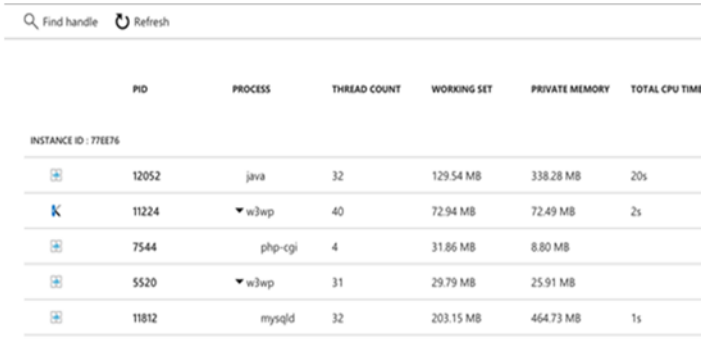
Voltamos à caixa de pesquisa, pesquisamos "**Process explorer**" e clicamos no resultado.

Como esperado o servidor de base de dados não está ativo, mas vamos ativar. Existem duas formas:

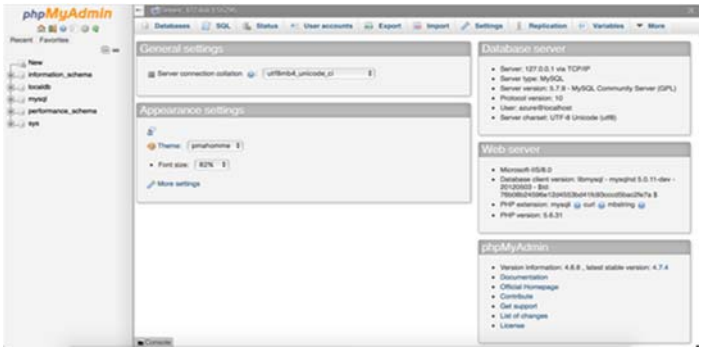
- Fazemos ping a partir da nossa máquina ao nosso site.
- Acedemos ao nosso site.

O endereço do site pode ser obtido na página de "**Overview**".

Após acedermos ao site (ou fazermo ping ao mesmo) voltamos ao "**Process explorer**" e verificamos que a instancia de MySql já se encontra ativa.



Se tentarmos novamente entrar na página do PHPMyAdmin com as mesmas credenciais:



Agora é uma questão de construirmos a nossa base de dados e popular com os dados necessários.

Desafio #3

Já temos o servidor aplicacional Java configurado, já temos a base de dados construída, agora só precisamos publicar, certo?

Bem, não. Ainda temos algum trabalho para fazer.

Como estamos a utilizar uma instancia do Tomcat no Azure não temos acesso à pasta de instalação para copiar o driver para a pasta **$CATALINA_BASE/lib** e editar o ficheiro **$CATALINA_BASE/conf/context.xml.**

Como não temos forma de fazer estas configurações temos que alterar o nosso projeto.

Antes de mais, vamos copiar o driver para o nosso projeto (junto da biblioteca jslt-1.2.jar) na pasta "**WebContent/WEB-INF/lib/**". Depois criamos o ficheiro "**context.xml**" na pasta "**WebContent/META-INF**" com o seguinte conteúdo:

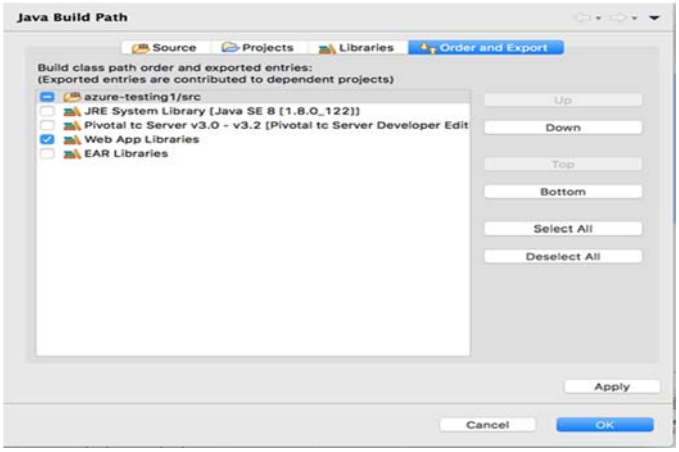
```
<Context>
  <Resource
    name="[JNDI NAME]"
    auth="Container"
    type="javax.sql.DataSource"
    maxActive="50"
    maxIdle="30"
    maxWait="10000"
    driverClassName="com.mysql.jdbc.Driver"
    username="[USERNAME]" password="[PASSWORD]"
    url="jdbc:mysql://[SERVER]:[PORT]/
                                    [DATABASE]"/>
</Context>
```

Em que os dados a preencher são:

- [USERNAME]: username de acesso ao PHPMyAdmin
- [PASSWORD]: password de acesso ao PHPMyAdmin
- [SERVER]: No topo da página do PHPMyAdmin
- [PORT]: No topo da página do PHPMyAdmin
- [DATABASE]: No topo da página do PHPMyAdmin

Exemplo da informação no PHPMyAdmin:

Depois de configurarmos, gravamos o ficheiro e garantimos que quando exportamos o ficheiro war, as bibliotecas são exportadas também:



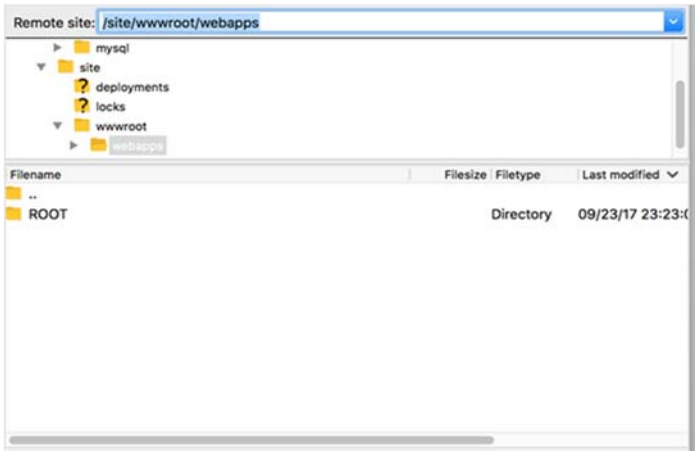
E exportamos o war.

Publicar a aplicação

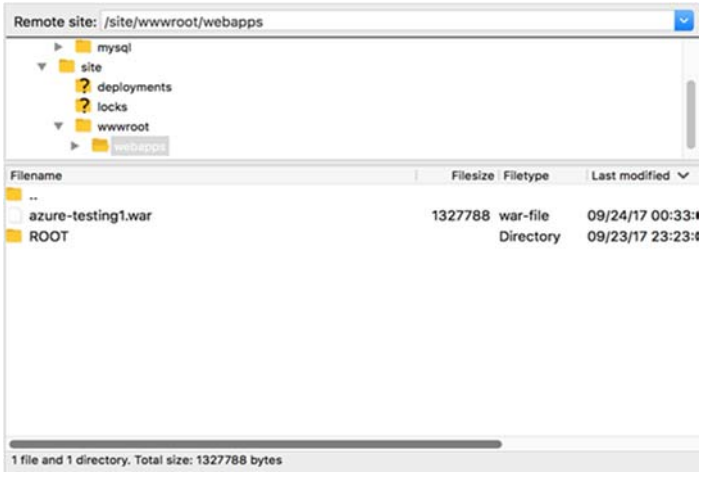
Finalmente chegamos ao que interessa, publicar a nossa aplicação. :-)

Abrimos novamente o FileZilla, acedemos ao servidor e vamos até à seguinte localização:

/site/wwwroot/webapps

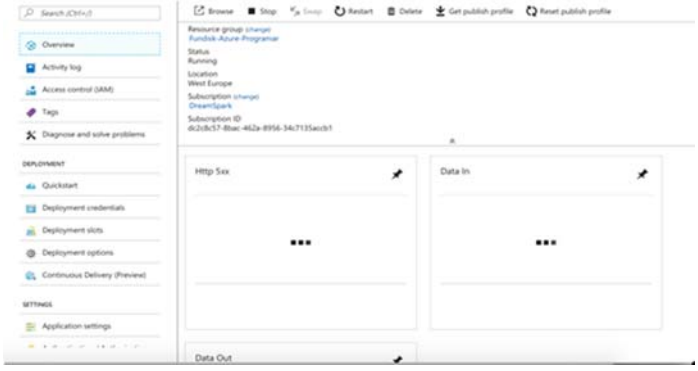


E fazemos upload da nossa aplicação.



Apesar da aplicação estar no servidor, ela só será instalada após o próximo *restart* do servidor.

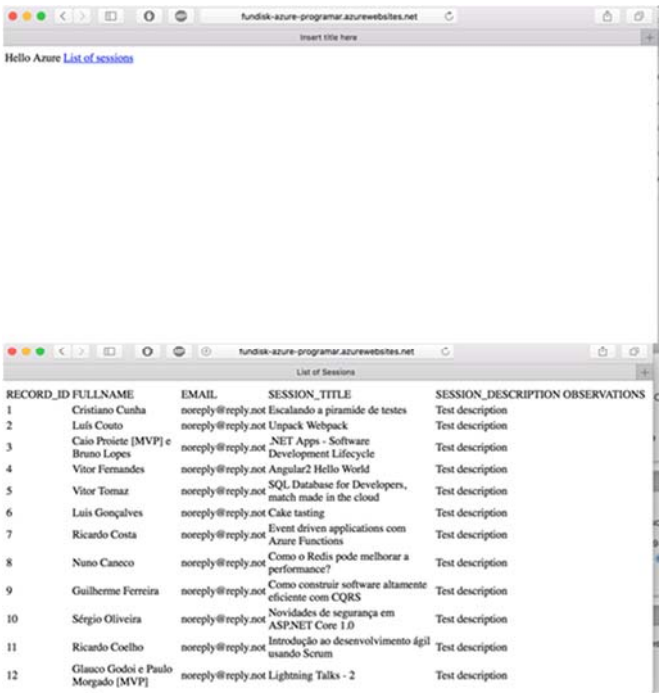
Para fazer restart do servidor, vamos à pagina "**Overview**" e carregamos "**Restart**" no topo da página.



Passados uns minutos a aplicação está instalada.

Se clicarmos no link para o site da aplicação, poderá aparecer o seguinte écran:

Se colocarmos o contexto-root da aplicação acedemos à mesma.



Mas não é isto que se pretende. O objetivo é aceder ao site pelo URL e entrar logo na aplicação.

Para realizar esse objetivo, temos duas hipóteses mais simples:

- Na pasta /site/wwwroot/webapps/ROOT colocamos um index.html que a única coisa que faz é um redirect para a nossa aplicação.
- Quando exportamos o nosso WAR, damos o nome ROOT.war e depois de fazermos upload a aplicação é deployed automaticamente para a raiz do site.

Sugiro esta última abordagem como sendo a que causa menos entropia.

Neste momento, a aplicação encontra-se viva na Internet.

> ***(...) Saliente o propósito deste artigo é colocar uma aplicação web em Java a correr no Azure e não as boas práticas do código em si, apesar de algumas serem aplicadas inconscientemente.***

> ***"Nós queremos correr o vosso software" (...) É uma mensagem simples, mas poderosa, que mostra a mudança da postura perante outras tecnologias que não sejam .NET, como seja o PHP ou o Java. (...)***

Conclusão

Foi um caminho longo para se publicar esta aplicação no Azure, principalmente pela primeira vez, mas este guia passo-a-passo demonstra cada um dos vários passos e torna mais célere as configurações.

Existem um vasto leque de opções, funcionalidades e ferramentas disponíveis no Azure que não foram abordados. Uns porque ainda não tive oportunidade de utilizar outros porque não se mostraram necessários para o proposito. Uma das funcionalidades é "Deployment option", que permite configurar um SCM para deploys automáticos. É uma funcionalidade muito interessante, que funciona muito bem se o projeto for em .NET. Obtém o código, compila e publica sem problemas. Mas para Java, pela experiência que tive, não funciona, dai optar pelo FTP.

Este artigo é um bom ponto de partida para publicar as nossas aplicações e tirar partido das inúmeras funcionalidades da plataforma. As primeiras vezes poderá demorar mais a configurar ou acertar com as configurações, mas com o tempo vem a experiência e torna-se mais célere.

Recursos

- https://blogs.msdn.microsoft.com/appserviceteam/2016/09/08/troubleshooting-faq-for-mysql-in-apppreview/
- https://www.mkyong.com/tomcat/how-to-configure-mysql-datasource-in-tomcat-6/
- https://docs.microsoft.com/en-us/azure/

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.

Junit

JUnit é um framework open-source para escrever e executar testes em Java. Com ele também é possível executar os asserts que verificam se uma condição é verdadeira. Recentemente (18/07/2017) foi lançada uma versão milestone (provavelmente não é uma versão final e completa) do JUnit 5, que necessita do Java 8.

A primeira aparência é que existem várias funcionalidades legais, tais como testes parametrizados (poder passar vários parâmetros para um mesmo teste executar várias vezes), assert de Exception melhorado, agrupar testes por Tags e etc.

Configuração

- Para poder utilizar a versão milestone 6 do JUnit, basta adicionar as seguintes dependências no pom.xml:

```
<dependency>
      <groupId>org.junit.jupiter</groupId>
      <artifactId>junit-jupiter-api</artifactId>
      <version>5.0.0-M6</version>
      <scope>test</scope>
</dependency>
<dependency>
      <groupId>org.junit.jupiter</groupId>
    <artifactId>junit-jupiter-params</artifactId>
      <version>5.0.0-M6</version>
      <scope>test</scope>
</dependency>
...
<repositories>
      <repository>
        <id>snapshots-repo</id>
         <url>https://oss.sonatype.org/content/
                    repositories/snapshots</url>
         <releases>
            <enabled>false</enabled>
         </releases>
         <snapshots>
            <updatePolicy>always</updatePolicy>
            <enabled>true</enabled>
         </snapshots>
      </repository>
<repositories>
```

O suporte a IDEs ainda é bem escasso, então é necessário configurar um plugin do Maven:

```
<plugin>
      <artifactId>maven-surefire-plugin
                                        </artifactId>
      <version>2.19.1</version>
      <dependencies>
            <dependency>
                  <groupId>org.junit.platform
                                            </groupId>
                  <artifactId>junit-platform-
                    surefire-provider</artifactId>
                  <version>1.0.0-M6</version>
            </dependency>
            <dependency>
              <groupId>org.junit.jupiter</groupId>
                  <artifactId>junit-jupiter-
                              engine</artifactId>
                  <version>5.0.0-M6</version>
            </dependency>
      </dependencies>
</plugin>
```

- Com isso configurado, já é possível executar o comando mvn test para executar os testes do JUnit 5.
- Criei um projeto java-sample-project utilizando algumas funcionalidades do JUnit 5 e vou resumir apenas algumas das funcionalidades:

Testes Parametrizados

- Permite rodar o mesmo teste com parâmetros diferentes com a anotação ***@ParameterizedTest***, vindo de diferentes fontes: Valores puros (String, Integer, etc.), Enum, CSV, Métodos…
- No projeto, utilizei o ***@MethodSource*** para exemplo:

```
@ParameterizedTest
@MethodSource(value="createUsers")
public void testParameterizedUser(User user,
Integer id) {
  Assertions.assertEquals(user,
userService.getById(id));

}
```

- Com isso, o teste irá procurar um método chamado *createUsers* que retorna um ***Stream<Arguments>***:

```
protected static Stream<Arguments> createUsers()
{
    return Stream.of(
            Arguments.of(userOne(), 1),
            Arguments.of(userTwo(), 2)
    );

}
```

- Os parâmetros esperados no meu método do teste são: um *User* e um *Integer*. Nesse Stream eu estou passando os dois. O teste irá executar primeiro com o userOne() e 1, e depois com userTwo() e 2.

Assert Exceptions

- Com o JUnit 4 era possível testar exceções, tanto com o @Test(expected=…) quanto com o Exception Rule (ExpectedException). Esse último ainda era possível validar a mensagem da exceção, mas

acredito que nada se compara com

esse ***assertThrows***:

```
@Test
public void testException() {
  Throwable exception = Assertions.assertThrows
                (IndexOutOfBoundsException.class,
                  () -> { userService.getAll().get
                                          (15); });
  Assertions.assertEquals("Index: 15, Size: 2",
exception.getMessage());
}
```

- **OBS:** Importante notar também o uso de Lambda Expressions e Streams, como no caso dos testes parametrizados. JUnit 5 faz bastante uso das funcionalidades do Java 8.

Asserts Agrupadas

- Antes, asserts em conjunto funcionavam assim: se um falhar, os outros nem são executados. Agora é possível agrupar os asserts de forma que todos são executados e todas as falhas serão reportadas juntas. Para isso, é usado o ***assertAll***:

```
@Test
public void groupedAssertions() {
    final User user = userService.getById(1);
    Assertions.assertAll("user",
        () -> Assertions.assertEquals("First User",
                                    user.getName()),
        () -> Assertions.assertEquals(new Integer
                                (1), user.getId()),
        () -> Assertions.assertEquals(new Integer
                               (20), user.getAge()),
        () -> Assertions.assertEquals("Rio de
                            Janeiro", user.getCity())
    );

}
```

- Se a primeira asserção ou outra(s) falhar(em), ainda assim todas as asserções seriam executadas e no final seria exibido quais falharam.

Assert Timeout

- Permite testar se um método vai executar dentro de um tempo esperado. Para isso, é usado o ***assertTimeout***:

```
@Test
public void timeoutExceeded() {
  Assertions.assertTimeout(ofMillis(100), () -> {
     // Simula uma task que demora menos de 100 ms.
      Thread.sleep(99); // Se demorar mais de 100,
                              //o teste vai falhar
    });

}
```

Nome de Visualização do Teste

- Esse é bem legal para um melhor relatório de testes. Permite que o nome de visualização do teste seja

```
@DisplayName("Testing exception")
@Test
public void testException() {
 ...
}
```

definido através da anotação ***@DisplayName***:

- Ok, eu sei que a descrição do meu teste foi bem estúpida, mas vocês entenderam a ideia.

Repetição de Testes

- Permite executar o mesmo teste mais de uma vez. Usa-se a anotação ***@RepeatedTest***. Para melhor visualização, a anotação possui algumas variáveis para serem utilizadas na descrição, como ***displayName***, ***currentRepetition*** e ***totalRepetitions***.

```
@RepeatedTest(value=2,name="{displayName}
{currentRepetition}/{totalRepetitions}")
@Test
//O teste repetirá 2 vezes, ou seja, será
executado 3 vezes no total
public void testException() {
 ...
```

- Conforme o parâmetro *name* descrito no @RepeatedTest, cada repetição irá mostrar, por exemplo:
  - "Testing exception 1/2"
  - "Testing exception 2/2"

Agrupar Testes em Tags

- Esse é bem legal também. Permite agrupar os testes por tags e rodar apenas os testes de uma tag específica. Para isso, usa-se a anotação ***@Tag***:

```
@Tag("example")
@Test
public void timeoutExceeded() {
   ...
}
```

- Para rodar os testes de uma tag específica (vamos supor *example*), basta executar *mvn test -Dgroups=example*
- A anotação @Tag pode ser usada tanto para classes quanto para métodos e ambos podem ter várias tags definidas no mesmo lugar

Testes Aninhados

- Com a anotação ***@Nested*** é possível criar uma classe de testes dentro de outra classe de testes. Muito bom para organizar melhor os testes.

```
public class UserServiceTest {
    @Nested
    @DisplayName("when new")
    class WhenNew {

        @Test
        @DisplayName("whatever")
        void isNotEmpty() {
            //Seus inner tests
             Assertions.assertTrue(Boolean.TRUE);
        }
     }
}
```

Desabilitar Testes

- Para finalizar, a anotação ***@Disabled*** que desabilita um teste/uma classe de testes.

```
@Test
@Disabled("For demonstration purposes")
public void skippedTest() {
    // não executado
    Assertions.assertTrue(false);
}
```

Para saber mais sobre JUnit 5: http://junit.org/junit5/docs/current/user-guide/

"JUnit é um framework open-source para escrever e executar testes em Java. Com ele também é possível executar os asserts que verificam se uma condição é verdadeira. Recentemente (18/07/2017) foi lançada uma versão milestone (provavelmente não é uma versão final e completa) do JUnit 5, que necessita do Java 8."

AUTOR

Escrito por Raphael Amoedo – a.k.a. **Ralph Avalon**

Graduado em Ciência da Computação – UVA-RJ, Pós-Graduado em Engenharia de Software com JEE – SENAC-RJ, trabalha com Java desde 2011, mas curte outras linguagens como Python e Javascript Desenvolvedor Java na OLX, Dono dos blogs de TI (https://ralphavalonbr.wordpress.com e https://ralphavalon.wordpress.com) e usuário ativo do Stackoverflow, Github e grupos de TI no Telegram

Linkedin: https://www.linkedin.com/in/raphael-amoedo-809427b7

Criar uma aplicação móvel com jQuery Mobile

Introdução

Já muito foi escrito sobre a biblioteca jQuery para JavaScript [https://jquery.com/], incluindo alguns artigos na Revista Programar (por exemplo, "Mitos do jQuery" [https://www.revista-programar.info/artigos/mitos-do-jquery/] e "jQuery: Usar ou Não Usar?" [https://www.revista-programar.info/artigos/jquery-usar-ou-nao-usar/]).

No entanto, existem outros projectos "irmãos" do projecto jQuery que são igualmente interessantes para programadores e designers Web, como as *frameworks* jQuery Mobile e jQuery UI. Neste artigo, foco-me na jQuery Mobile explicando a sua filosofia de programação, e mostrando alguns dos componentes principais para a criação de uma aplicação móvel.

O que é, e porquê usar?

jQuery Mobile [http://jquerymobile.com/] é, ao contrário da biblioteca jQuery, uma *framework* para aplicações móveis. Usar jQuery Mobile implica seguir uma estrutura pré-definida para a nossa aplicação. Esta *framework* permite-nos criar rapidamente aplicações Web com um *look and feel* semelhante a aplicações móveis nativas. A Figura 1 mostra um exemplo do que se consegue facilmente obter com jQuery Mobile. A imagem ilustra uma aplicação com uma gaveta de navegação lateral (Figure 1, à direita) acessível através de um botão no cabeçalho (Figura 1, à esquerda) da aplicação.

Uma das vantagens da jQuery Mobile é que a estrutura da aplicação é definida inteiramente através de HTML. Por exemplo, a gaveta de navegação da aplicação da Figura 1, é construída através do código HTML seguinte:

```
<div data-role="panel" id="mypanel1" data-
display="overlay">
 <img src="space-cube.svg" width="240" >
 <ul data-role="listview">
   <li><a href="#first-page">Page 1</a></li>
   <li><a href="#second-page">Page 2</a></li>
   <li><a href="#third-page">Page 3</a></li>
 </ul>
</div>
```

Essencialmente, a *framework* jQuery Mobile dá-nos um conjunto de elementos de interface gráfica com aspecto visual e comportamento adequados a aplicações móveis. Esta *framework* pode ser usada na construção de aplicações móveis finais, mas pode também ser útil para criação de protótipos funcionais. O facto de ser baseada em tecnologias Web, pode torná-la especialmente útil para designers multimédia interessados em criar e testar um protótipo funcional de uma aplicação móvel: é possível definir todos os ecrãs e transições entre ecrãs que não dependam de lógica específica da aplicação sem necessidade sequer de usar JavaScript.

Figura 1. Possível aspecto de uma aplicação desenvolvida com jQuery Mobile.

Filosofia de programação

A filosofia de programação jQuery Mobile assenta essencialmente na utilização de atributos data-* nos elementos HTML e classes CSS pré-definidas. Por exemplo, a definição de um ecrã da aplicação é feita incluindo simplesmente um com um id e com o atributo data-role="page" (em jQuery Mobile, um ecrã é designado por *page*):

```
<body>
  <div data-role="page" id="pagina-1">
    <!-- conteúdo do ecrã -->
  </div>
</body>
```

Apesar de não ser absolutamente obrigatório, a *framework* jQuery Mobile é orientada principalmente para aplicações de página única (*Single Page Applications* - *SPA*), pelo que uma aplicação com dois ecrãs definiria simplesmente dois como descendentes do <body> do documento HTML:

```
<body>
  <div data-role="page" id="pagina-1">
    <!-- conteúdo do primeiro ecrã -->
  </div>

  <div data-role="page" id="pagina-2">
    <!-- conteúdo do segundo ecrã -->
  </div>
</body>
```

Configuração

A configuração da framework no projecto Web passa simplesmente pela inclusão dos ficheiros CSS e JavaScript (as versões poderão variar):

```
<link rel="stylesheet"
href="http://code.jquery.com/mobile/1.4.5/
                    jquery.mobile-1.4.5.min.css" />
<script src="http://code.jquery.com/jquery-
                          1.11.1.min.js"></script>
<script src="http://code.jquery.com/mobile/1.4.5/
               jquery.mobile-1.4.5.min.js"></script>
```

De notar que a jQuery Mobile depende da biblioteca jQuery pelo que é importante incluir primeiro o ficheiro JavaScript jQuery e só depois o ficheiro jQuery Mobile.

Igualmente importante é a inclusão de um elemento <meta> no cabeçalho documento HTML para garantir que a aplicação é mostrada correctamente num dispositivo móvel:

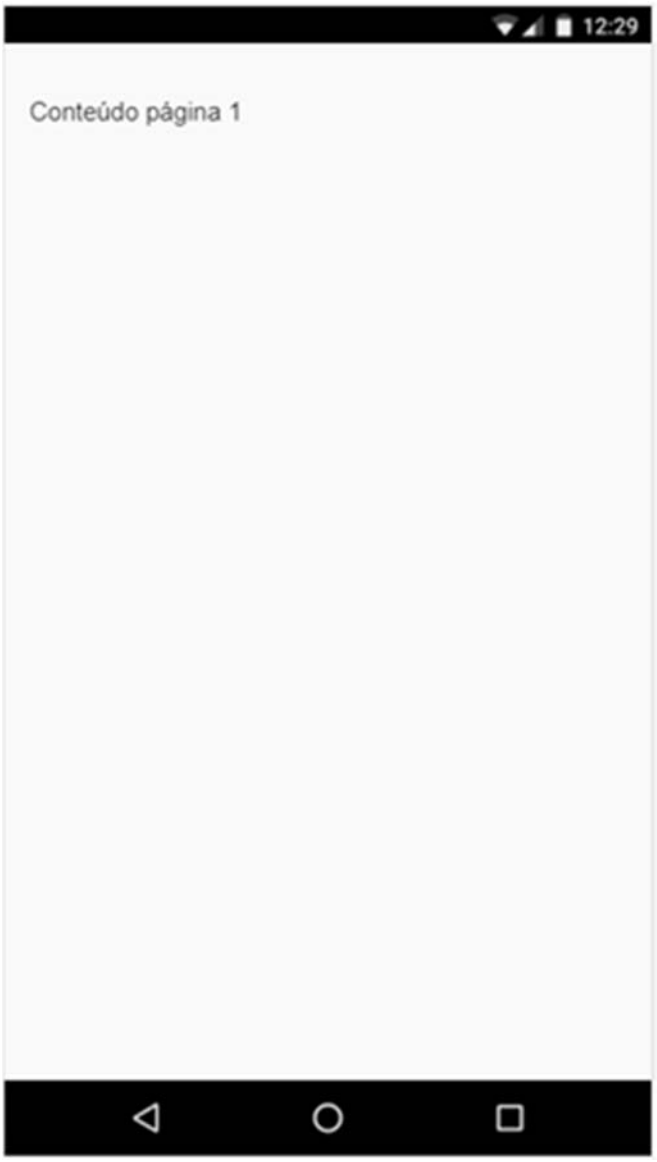
```
<meta name="viewport" content="width=device-width,
initial-scale=1">
```

Principais componentes

Páginas

Uma página é o componente básico em jQuery Mobile que representa um ecrã da aplicação. Todos os outros componentes que iremos ver são inseridos dentro de uma página. Em termos de estrutura HTML, uma página é definida através de um com um atributo data-role="page". É importante atribuir também um "id" ao , uma vez que este será a forma de nos referirmos à página para, por exemplo, permitir navegação entre ecrãs da aplicação. O conteúdo visível da página deve ser colocado dentro de um com role="main" e class="ui-content">.

```
<div data-role="page" id="pagina-1">
  <div role="main" class="ui-content">
    <p>Conteúdo página 1</p>
  </div>
  <!-- /content -->
</div>
<!--/pagina-1 -->

<div data-role="page" id="pagina-2">
  <div role="main" class="ui-content">
    <p>Conteúdo página 2.</p>
  </div>
  <!-- /content -->
</div>
<!-- /pagina-2 -->
```



Exemplo 1 – Páginas.

Código completo: https://goo.gl/Byq1jc | Ver no dispositivo móvel: https://goo.gl/Y9Tfgk

O Exemplo 1 demonstra uma aplicação com apenas duas páginas.

De notar que ao "executar" a aplicação, apenas vemos o conteúdo da primeira página. Para vermos a segunda página temos de adicionar alguma forma de navegação entre ecrãs da nossa aplicação. Uma forma de permitir navegação entre ecrãs é criando simplesmente links internos tal como faríamos numa página web tradicional, usando os ids como URL internos: Link para página 1.

O Exemplo 2 ilustra esta ideia, permitindo-nos navegar entre os dois ecrãs, pressionando nos links respectivos.

Numa aplicação móvel, no entanto, não é usual pressionarmos em links para navegar - tipicamente, usamos botões.

Botões

A criação de botões ilustra bem a vantagem da utilização da framework jQuery Mobile: para criarmos um botão apenas temos de adicionar um atributo ao elemento <a> que usámos para criar a navegação entre ecrãs – o atributo data-role="button". Isto cria um botão que ocupa toda a largura do ecrã, como se pode ver no "Botão 1" do Exemplo 3. Se quisermos que o botão ocupe apenas a largura necessária para mostrar o texto no interior, podemos acrescentar o atributo data-inline="true" ("Botão 2").

```
<div data-role="page" id="pagina-1">
  <div role="main" class="ui-content">
    <p> Conteúdo página 1 </p>
    <a href="#pagina-2">Ir para página 2</a>
  </div>
  <!-- /content -->
</div>
<!--/pagina-1 -->

<div data-role="page" id="pagina-2">
  <div role="main" class="ui-content">
    <p>Conteúdo página 2.</p>
    <a href="#pagina-1">Ir para página 1</a>
  </div>
  <!-- /content -->
</div>
<!-- /pagina-2 -->
```

Exemplo 2

Código completo: https://goo.gl/N9YBZL | Ver no dispositivo móvel: https://goo.gl/7WMS62

```
<div data-role="page" id="pagina-1">
  <div role="main" class="ui-content">
    <p> Conteúdo página 1 </p>
    <a data-role="button" href="#pagina-2">Botão
1</a>
    <a data-role="button" data-inline="true"
href="#pagina-2">Botão 2</a>
    <a data-role="button" data-inline="true" data
-icon="navigation" href="#pagina-2">Botão 3</a>
    <a class="ui-btn ui-btn-icon-left ui-icon-
navigation ui-btn-inline ui-shadow" href="#pagina
-2">Botão 4</a>
  </div>
  <!-- /content -->
</div>
<!--/pagina-1 -->
```

Exemplo 3 - Botões

Código completo: https://goo.gl/NLvn7M | Ver no dispositivo móvel: https://goo.gl/ZY4MPs

```
<div data-role="page" id="pagina-1">
  <div data-role="header" data-position="fixed">
    <h1>Primeira página</h1>
  </div>
  <!-- /header -->

  <div role="main" class="ui-content">
    <p> Conteúdo página 1 </p>
    <a data-role="button" href="#pagina-2">Ir para
Página 2</a>
  </div>
  <!-- /content -->

  <div data-role="footer" data-position="fixed">
    <h1>Primeira página</h1>
  </div>
  <!-- /footer -->
</div>
<!--/pagina-1 -->
```

Exemplo 4 – Cabeçalho e rodapé

Código completo: https://goo.gl/r7BVp7 | Ver no dispositivo móvel: https://goo.gl/AjavMT

Podemos também acrescentar um ícone ao botão indicando o nome do ícone no atributo data-icon ("Botão 3") – a lista de ícones pré-definidos pode ser consultada em https://api.jquerymobile.com/icons/. O Exemplo 3 mostra alguns dos tipos de botões que podemos criar desta forma. De notar que estes atributos "data-*" são na verdade usados pela framework jQuery Mobile para determinar que classes CSS irá atribuir ao elemento. O último botão no Exemplo 3 mostra como podemos obter o mesmo efeito indicando directamente as classes CSS a usar ("Botão 4").

Cabeçalho e rodapé

É comum as aplicações móveis terem um cabeçalho e/ou um rodapé com o logótipo da aplicação e com botões de navegação. Estes elementos podem ser incluídos num ecrã da aplicação jQuery Mobile inserindo um elemento com data-role="header" (ou data-role="footer") dentro da página correspondente ao ecrã. Por exemplo, para adicionar um cabeçalho e rodapé à nossa primeira página teríamos o código do Exemplo 4. (Como normalmente queremos que o cabeçalho e rodapé fiquem sempre no topo e fundo da janela da aplicação devemos usar também o atributo data-position="fixed", caso contrário ficarão no fluxo de conteúdo da página e sujeitos ao scroll.)

De notar que o código do cabeçalho e do rodapé tem de ser repetido em cada página em que pretendamos que estejam presentes, caso contrário alguns ecrãs da nossa aplicação apresentarão cabeçalho/rodapé e outros não.

Botões e navegação no cabeçalho ou rodapé

Os cabeçalhos ou rodapés das aplicações móveis servem, muitas vezes, como ponto de acesso a funcionalidades da aplicação ou como forma de navegação entre os vários ecrãs. Se quisermos incluir botões no cabeçalho ou rodapé podemos fazê-lo através de um elemento com data-role="controlgroup" dentro do elemento que representa o cabeçalho. Este elemento permite-nos incluir vários botões e controlar o seu posicionamento no cabeçalho ou rodapé. O Exemplo 5 mostra o código a incluir no cabeçalho para criar um ecrã com dois botões no cabeçalho. No elemento que define o *controlgroup*, de notar a utilização dos atributos data-type="horizontal" para garantir que os botões são dispostos horizontalmente e da classe CSS "ui-btn-right" para alinhar os botões à direita (podiamos obviamente usar "ui-btn-left" para alinhar à esquerda). De notar também o uso do atributo data-iconpos="notext" nos botões para esconder o texto, ficando apenas o ícone do botão.

Quando o objectivo é criar uma área de navegação entre páginas no cabeçalho ou rodapé, existe um componente especializado – a barra de navegação. A inclusão de uma barra de navegação pode ser feita no cabeçalho ou rodapé (ou até no corpo da página). No Exemplo 6, a inclusão é feita no rodapé. Uma barra de navegação é simplesmente um elemento com data-role="navbar". No interior do inserimos uma lista não ordenada com um link (elemento âncora <a>) em cada item. Automaticamente, os links são

convertidos em botões (neste caso não é necessário adicionar o atributo data-role="button"). Para indicar que um botão está activo (ou seja que o ecrã visível corresponde a um determinado botão na barra de navegação) podemos usar a classe CSS "ui-btn-active". Note-se que no Exemplo 6 o rodapé apenas contém a navegação. No entanto, é possível combinar a barra de navegação com outros elementos.

```
<div data-role="header">
  <div data-role="controlgroup" data-
type="horizontal" class="ui-btn-right">
    <a data-icon="star" data-role="button" data-
inline="true" data-iconpos="notext"
href="#">Favorito</a>
    <a data-icon="gear" data-role="button" data-
inline="true" data-iconpos="notext"
href="#">Settings</a>
  </div>
  <h1>Primeira página</h1>
</div>
```

Exemplo 5 – Botões no cabeçalho

Código completo: https://goo.gl/8LFCd5 | Ver no dispositivo móvel: https://goo.gl/hgoC8W

```
<div data-role="footer" data-position="fixed">
  <div data-role="navbar">
    <ul>
      <li>
        <a data-icon="home" class="ui-btn-active"
href="#">Home</a>
      </li>
      <li>
        <a data-icon="gear" href="#">Settings</a>
      </li>
      <li>
        <a data-icon="info" href="#">Info</a>
      </li>
    </ul>
  </div>
  <h1>Primeira página</h1>
</div>
```

Exemplo 6 – Barra de navegação

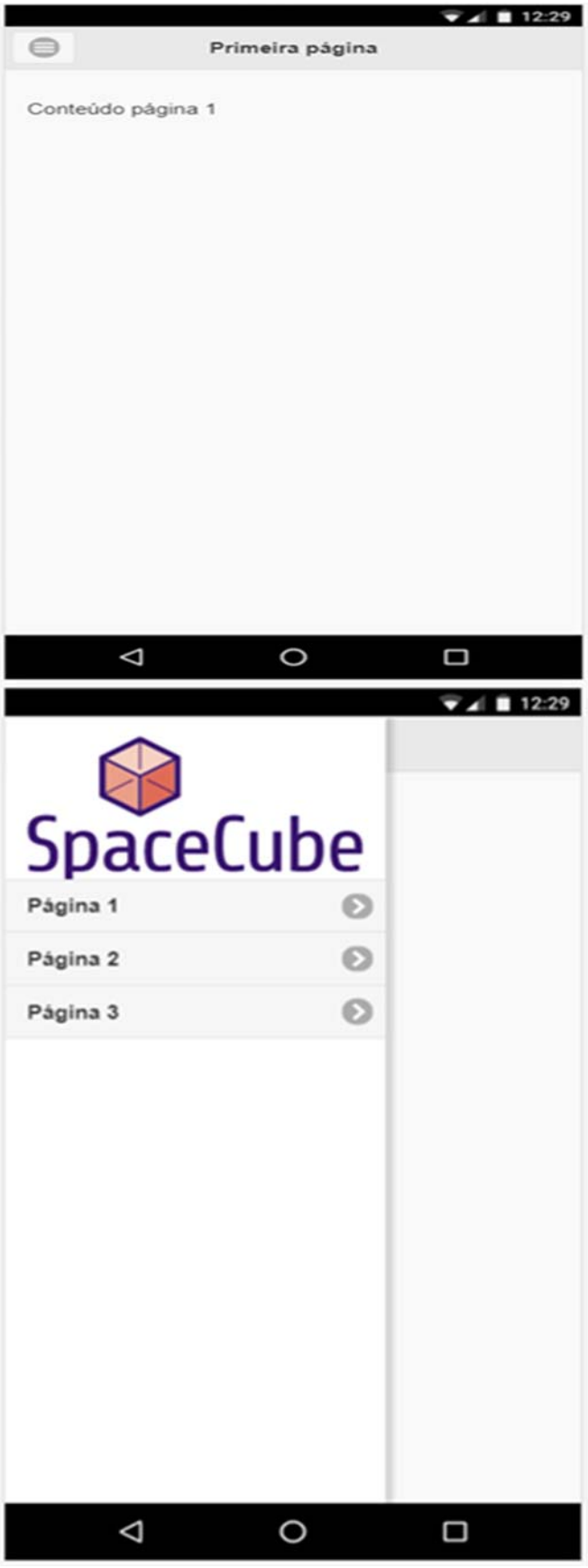
Código completo: https://goo.gl/rLoJVw | Ver no dispositivo móvel: https://goo.gl/c23dCK

Gaveta de navegação

A barra de navegação pode ser usada quando o número de elementos de navegação é relativamente reduzido. Para além disso, os botões na barra de navegação consomem espaço que pode ser importante para mostrar conteúdo mais útil. Uma alternativa à barra de navegação (ou um complemento) é a gaveta de navegação. A gaveta de navegação

é um painel que podemos abrir ou fechar pressionando num botão (tipicamente o botão é apresentado no cabeçalho). Para criarmos uma gaveta de navegação para uma determinada página adicionamos um elemento com data-role="panel" e com um id. Para listar as opções de navegação no interior da gaveta usa-se tipicamente uma *listview* (uma lista não ordenada com data-role="listview") com links em cada item. Para possibilitar a abertura da gaveta de navegação devemos incluir um *controlgroup* no cabeçalho com um botão que refira o id que escolhemos para o painel que representa a gaveta de navegação. O Exemplo 7 mostra o código completo de uma página com uma gaveta de navegação. O atributo data-display no painel que representa a gaveta de navegação permite-nos configurar a forma como a gaveta surge ao utilizador ("overlay" é a forma mais comum, em que a gaveta é puxada para cima do conteúdo do ecrã; "reveal" cria uma gaveta fixa e é o conteúdo do ecrã que se move para mostrar a gaveta por baixo; "push" cria uma gaveta ao lado do conteúdo do ecrã e ambos de movem para mostrar ou esconder a gaveta.)

```
<div data-role="page" id="pagina-1">

  <div data-role="header">
    <h1>Primeira página</h1>
    <div data-role="controlgroup" data-type="horizontal"
                                                  class="ui-btn-left">
      <a data-icon="bars" data-iconpos="notext" data-
                        role="button" href="#gaveta1">Menu</a>
    </div>

  </div>
  <!-- /header -->

  <div data-role="panel" id="gaveta1" data-display="overlay">

    <!-- Fake logo from: https://github.com/pigment/fake-logos
                                                          -->
    <img src="https://s3-us-west-2.amazonaws.com/
                  s.cdpn.io/455884/space-cube.svg" width="240">

      <ul data-role="listview">
        <li>
          <a href="#pagina-1">Página 1</a>
        </li>
        <li>
          <a href="#pagina-2">Página 2</a>
        </li>
        <li>
          <a href="#pagina-3">Página 3</a>
        </li>
      </ul>

    </div>

  <!-- /gaveta navegação -->

  <div role="main" class="ui-content">

    <p> Conteúdo página 1 </p>

  </div>

  <!-- /content -->

</div>
<!--/pagina-1 -->
```



Exemplo 7 – Gaveta de navegação

Código completo: https://goo.gl/P9dEHZ | Ver no dispositivo

móvel: https://goo.gl/Vg99iY

De notar que a gaveta de navegação tem de ser incluída em cada página (com um id diferente para cada página). É possível evitar este tipo de repetição, mas neste momento, isso obriga ao uso de algum código JavaScript, por isso não abordo essa forma de inclusão de gavetas de navegação.

> ***No entanto, existem outros projectos "irmãos" do projecto jQuery que são igualmente interessantes para programadores e designers Web, como as frameworks jQuery Mobile e jQuery UI. Neste artigo, foco-me na jQuery Mobile explicando a sua filosofia de programação, e mostrando alguns dos componentes principais para a criação de uma aplicação móvel.***

Conclusão

Há, obviamente, muito mais a dizer sobre jQuery Mobile, não apenas sobre componentes de interface gráfica disponíveis, mas também sobre a API JavaScript que nos permite instanciar e controlar programaticamente os componentes de interface gráfica, e também sobre a possibilidade de customização da aparência dos componentes.

Com esta introdução, o leitor terá ficado com uma ideia das possibilidades que a framework jQuery Mobile nos dá, da sua estrutura, e dos principais componentes de interface gráfica disponíveis.

AUTOR

Escrito por Jorge Cardoso

Jorge C. S. Cardoso é professor auxiliar convidado no Departamento de Engenharia Informática da Universidade de Coimbra onde lecciona disciplinas relacionadas com tecnologias web, programação, e interacção humano-computador. É autor do livro "Java para Telemóveis" editado pela FEUP Edições. Licenciado em Engenharia Informática pela Universidade do Porto (FEUP), mestre em Sistemas Móveis e doutor em Tecnologias e Sistemas de Informação pela Universidade do Minho (Escola de Engenharia).

Lua – Linguagem de Programação – Parte 13

Neste artigo são apresetadas algumas operações complemetares ao conjuto de informações indicadas em outros artigos, tais como: passagem de parâmetro por matriz, funções anónimas (*lambda*), funções aninhadas (*clousure*), simulação do uso e tratamento de exceções, matrizes internas e compilação de programas.

Passagem de parâmetro por matriz

A linguagem Lua diferencia-se em diversos detalhes de linguagens de programação estruturadas e orientadas a objeto, principalmente o que tange a definição e uso de sub-rotinas (métodos).

É sabido que a linguagem Lua opera com sub-rotinas em estilo função não tendo de forma explícita em sua estrutura operacional a figura dos procedimentos o que, muitas vezes, leva um programador a fazer uso de funções como se fossem procedimentos. No entanto, é possível perceber a figura de um procedimento em Lua como será apontado.

Outro diferencial encontrado na linguagem de programação Lua é o fato desta somente aceitar a passagem de parâmetros por valor, não se utilizando da passagem de parâmetro por referência. No entanto, uma função Lua definida pelo programador consegue retornar mais de um valor a sua chamada. Assim justificando a ausência de passagem de parâmetro por referência.

Essas características operacionais trazem a tona uma questão curiosa na linguagem em relação à passagem de parâmetro com matrizes. Em Lua a passagem de parâmetro por matriz ocorre como referência sem que nada necessite ser definido para tal ação.

Para demonstrar a passagem de parâmetro por matriz tome, como exemplo, um programa que apresente o resultado do somatório dos elementos de uma matriz.

Em seguida escreva o código de programa em um editor de texto, gravando-o com o nome **P13_01.lua** e execute-o com a linha de comando **lua 5.1 P13_01.lua**.

```
-- inicio do programa P13_01

   function somatório(VET, TAM)
     local S = 0
     local X
     for X = 1, TAM do
       S = S + VET[X]
     end
     return S
   end

  local A = {0, 2, 4, 6, 8}

  io.write("Soma = ")
  print(somatorio(A, 5))
```

Ao ser executado o programa **P13_01.lua** é apresentada **Soma = 20**. Neste exemplo não é possível perceber a ocorrência da ação de referência em relação aos valores da matriz, mas percebe-se a ação da passagem dos valores da matriz.

Assim, considere um programa que define uma matriz com um conjunto de valores os quais são elevados ao quadrado.

Em seguida escreva o código de programa em um editor de texto, gravando-o com o nome **P13_02.lua** e execute-o com a linha de comando **lua 5.1 P13_02.lua**.

```
-- inicio do programa P13_02

   function quadrado(VET, TAM)
     local X
     for X = 1, TAM do
       VET[X] = VET[X] * 2
     end
   end

   local A = {0, 2, 4, 6, 8}

   for I = 1, 5 do
     io.write("A = ["..I.."] = ")
     print(A[I])
   end
   print()

   quadrado(A, 5)

   for I = 1, 5 do
     io.write("A = ["..I.."] = ")
     print(A[I])
   end
   print()

-- fim do programa P13_02
```

Observe que o programa **P13_02.lua** faz uso da sub-rotina de função **quadrado()** sem uso de **return**. Uma sub-rotina que não retorna valor é por sua natureza um procedimento.

Ao ser executado o programa são apresentados os valores da matriz **A** antes da execução da ação da função **quadrado()**, sendo então apresentados os valores:

```
A = [1] = 0
A = [2] = 2
A = [3] = 4
A = [4] = 6
A = [5] = 8
```

Na sequência o programa executa a ação da chamada da sub-rotina **quadrado()** como se a sub-rotia fosse um comando da linguagem e apresenta a saída dos valores:

```
A = [1] = 0
A = [2] = 4
A = [3] = 8
A = [4] = 12
A = [5] = 16
```

Indicando a alteração dos valores definidos na matriz **A**.

A partir desta ação nota-se que no uso das passagens de parâmetro na linguagem Lua ocorre passagem de parâmetro por valor nos casos de uso de variáveis simples. No uso de variáveis compostas (matriz) a passagem de parâmetro é por referência.

Funções anónimas (lambda)

Uma função anónima é uma sub-rotina sem nome, declarada e associada diretamente a uma variável. Para maiores detalhes sobre este tema sugere-se a leitura do artigo **FUNÇÕES ANÓNIMAS** de João Silva e Cristinao Ramos publicado na edição 44 desta revista.

Como exemplo deste tipo de ação considere um programa que apresenta o somatório dos valores inteiros de 1 até um limite fornecido pelo utilizador do programa.

Em seguida escreva o código de programa em um editor de texto, gravando-o com o nome **P13_03.lua** e execute-o com a linha de comando **lua 5.1 P13_03.lua**.

```
-- inicio do programa P13_03

   somatorio = function(N)
     local S = 0, I
     for I = 1, N do
       S = S + I
     end
     return S
   end

   io.write("Entre limite: ")
   VLR = io.read("*number")
   io.write("Soma = ")
   print(somatorio(VLR))

-- fim do programa P13_03
```

Ao ser executado o programa, se fornecido o valor **100** será dado como resposta o valor **5050** resultado do somatório de **1** a **100**.

Funções anónimas aceitam recursividade. Tome por exemplo o cálculo da factorial de um valor qualquer.

Em seguida escreva o código de programa em um editor de texto, gravando-o com o nome **P13_04.lua** e execute-o com a linha de comando **lua 5.1 P13_04.lua**.

```
-- inicio do programa P13_04

   factorial = function(N)
     if (N <= 1) then
       return 1
     else
       return factorial(N - 1) * N
     end
   end

   io.write("Entre limite: ")
   VLR = io.read("*number")
   io.write("Factorial = ")
   print(factorial(VLR))

-- fim do programa P13_04
```

Ao ser executado o programa se informado **5** será apresentado o resultado **120**.

Funções aninhadas (clousure)

Uma função do tipo *clousure* (enclausurada) é a função definida dentro de outra função, a qual possui acesso a variáveis que estejam definidas fora do escopo da função.

O exemplo seguinte apresenta a sequência de Fibonacci a partir do uso de função *clousure*.

Em seguida escreva o código de programa em um editor de texto, gravando-o com o nome **P13_05.lua** e execute-o com a linha de comando **lua 5.1 P13_05.lua**.

```
-- inicio do programa P13_05

   function fibonacci()
     local ATUAL = 1
     local ANTERIOR = 0
     local PROX
     return function()
       local PROX = ANTERIOR + ATUAL
       ANTERIOR = ATUAL
       ATUAL = PROX
       return PROX

     end

   end

   local X = fibonacci()

   for I = 1, 10 do

     print(X())

   end
-- fim do programa P13_05
```

Ao ser executado o programa são apresentados os valores **1**, **2**, **3**, **5**, **8**, **13**, **21**, **34**, **55** e **89**.

Observe a definição da função anónima **function()** que efetua o cálculo da sequência de Fibonacci dentro (enclausurada) da função **fibonacci()** efetuando acesso as variáveis locais **ATUAL**, **ANTERIOR** e **PROX** quando do uso da instrução **return function()**.

Tratamento de exceções

Lua não possui instrução para tratamento de exceção do tipo **try … throw … catch**, mas por meio do auxílio das funções internas **pcall()** e **error()** são possíveis estabelecer uma forma alternativa de realizar tal tarefa.

O programa seguinte tem por finalidade indicar mensagem de erro se um valor **0** for fornecido para o divisor de uma operação de divisão.

Em seguida escreva o código de programa em um editor de texto, gravando-o com o nome **P13_06.lua** e execute-o com a linha de comando **lua 5.1 P13_06.lua**.

```
-- inicio do programa P13_05

   local DIVID, DIV, QUOC
   local STATUS, ERRO

   io.write("Entre o dividendo .. ")
   DIVID = io.read("*number")
   io.write("Entre o divisor .... ")
   DIV = io.read("*number")

   STATUS, ERRO = pcall -- try
   (
     function()
       io.write("Resultado = ")
       if (DIV == 0) then
         error("Erro", 0) -- throw
       End

       QUOC = DIVID / DIV
       print(QUOC)
     end
   )
   if not (STATUS) then -- catch

 print(ERRO)

   end

-- fim do programa P13_05
```

Ao ser executado o programa se informado os valores **5** e **2** respetivamente para o dividendo e divisor será apresentada a mensagem **Resultado = 2.5**, mas se forem fornecidos os valores **5** e **0** será apresentada a saída **Resultado = Erro**.

As variáveis **DIVID**, **DIV** e **QUOC** são definidas para operarem respetivamente com os valores de dividendo, divisor e quociente.

As variáveis **STATUS** e **ERRO** são definidas para operarem com os valores lógicos retornados pelas funções **pcall()** e **error()**.

A função **pcall()** captura o erro ocorrido retornando um valor indicando o *status* do erro. Esta função é usada para chamar uma função como parâmetro, neste caso, definida como **pcall (function() ...)**. Se a chamada ocorrer sem erro **pcall** retorna verdadeiro e todos os resultados da chamada. Se ocorrer erro **pcall** retorna falso mais a mensagem de erro.

A função **error()** não retorna valor de erro, retorna uma mensagem de ocorrência que pode ser associada à informação sobre a posição onde o erro ocorreu ou em que linha do programa a função está sendo utilizada. Esta função pode utilizar dois parâmetros sendo o primeiro parâmetro obrigatório e o segundo opcional. O segundo parâmetro quando omitido assume valor 1 (mostra a linha de código onde a função está definida), se informado o valor 0 nenhuma informação do local de ocorrência do erro é indicado, mantendo-se apenas a mensagem definida.

No código do programa, o equivalente a um bloco **try** é processado pela função **pcall**. Se ocorrer um erro de divisão por zero a função **error()** é acionada simulando uma ação **throw**. O processamento de um bloco **catch** é simulado pela instrução **if not (STATUS)**.

Matrizes internas

No segundo exemplo deste artigo foi utilizada no programa a definição de uma matriz interna, em estilo lista (vetor): **local A = {0, 2, 4, 6, 8}**.

Uma matriz, seja de uma ou duas dimensões, é considerada interna quando seus elementos são configurados no próprio código de um programa.

Como no segundo exemplo já foi indicado o uso de uma matriz de uma dimensão, cabe na sequência demonstrar o uso de uma matriz de duas dimensões.

O programa seguinte demonstra a apresentação dos valores subtraídos e obtidos a partir de duas matrizes de duas dimensões.

Em seguida escreva o código de programa em um editor de texto, gravando-o com o nome **P13_07.lua** e execute-o com a linha de comando **lua 5.1 P13_07.lua**.

```
-- inicio do programa P13_06

   local A = {
               {11,22,33},
               {44,55,66},
               {77,88,99}
             }

   local B = {
               { 1, 2, 3},
               { 4, 5, 6},
               { 7, 8, 9}
             }

   local C = {}

   for I = 1, 3 do
     C[I] = {}
     for J = 1, 3 do
       C[I][J] = A[I][J] - B[I][J]
     end
   end

   for I = 1, 3 do
     for J = 1, 3 do
       print(C[I][J])
     end
   end

-- fim do programa P13_06
```

Ao ser executado o programa é apresentada a saída **10**, **20**, **30**, **40**, **50**, **60**, **70**, **80** e **90**.

O programa faz uso das duas maneiras de definição de matrizes de duas dimensões em Lua. A primeira maneira definida de forma interna em relação às variáveis indexadas **A** e **B** e a segunda forma em relação a variável indexada **C** declarada inicialmente como **local C = {}** onde se marca a primeira dimensão e dentro do ciclo de tratamento da variável **I** como **C[I] = {}** para definir-se a segunda dimensão da matriz.

Matrizes internas de duas ou com até mais dimensões são declaradas com a forma:

```
MATRIZ = {
           {L1C1, L1C2, L1C3, L1Cn},
           {L2C1, L2C2, L2C3, L2Cn},
           {L3C1, L3C2, L3C3, L3Cn},
           {LnC1, LnC2, LnC3, LnCn}
         }
```

Onde **MATRIZ** é o nome de identificação da variável indexada, **L1**, **L2**, **L3** e **Ln** indicam as linhas e **C1**, **C2**, **C3** e **Cn** indicam as colunas.

Observe que cada grupo de linhas tem definido suas colunas dentro de chaves **{}** separando-as por vírgulas dentro de um duplo de chaves **{}**.

Compilação de programas

Apesar da linguagem Lua ser do tipo *script*, e por esta razão ser interpretada, ela possui um compilador que gera um código binário no formato *bytecode* que pode ser executado posteriormente pelo próprio interpretador. Para fazer uso deste recurso é necessário utilizar o programa **luac** como segue:

```
luac <opções> <prog.lua>
```

O parâmetro **opções** é opcional, podendo ser omitido e o parâmetro obrigatório **prog.lua** é a indicação de um código de *script* Lua que será compilado.

Em **opções** pode-se fazer uso de algumas chaves de execução, destacando-se principalmente:

- **-l** gera listagem dos *bytecodes* do programa compilado;
- **-o "arquivo"** usado para definir o nome do arquivo do programa compilado (arquivo de saída), Se omitido esta opção o programa compilado será definido com o nome **luac.out**;
- **-v** apresenta a informação da versão do ambiente Lua.

Um programa compilado em Lua não possui maior velocidade de execução se comparado com sua versão no formato *script*, pois quando se indica a execução do *script* Lua com o interpretador, este faz automaticamente uma pré-compilação do código.

Outro detalhe operacional é o fato do compilador gerar um único arquivo de saída, mesmo que se tenha um conjunto de *scripts* a serem executados.

Para mais detalhes sobre este recurso aconselhá-se pesquisar o sítio **LUAC main page** em: **https://www.lua.org/manual/4.0/luac.html**.

No sentido de proceder a um teste simplificado de execução do compilador informe na linha de comando de seu sistema a instrução:

```
luac P13_06.lua
```

Em seguida execute o programa compilado com a instrução:

```
lua luac.out
```

Para conferir que o arquivo gerado ´pelo compilador é binário tente abrir com um editor de texto simples o arquivo **luac.out**.

Para gerar um arquivo compilado com nome definido pelo programador execute a instrução:

```
luac –o teste.cmp P13_06.lua
```

Em seguida execute o programa com a instrução:

```
lua teste.cmp
```

Observe a instrução para listar a estrutura de *bytecodes* usados pelo compilador:

```
luac –l P13_06.lua
```

Após a execução da instrução anterior a listagem é imediatamente apresentada.

A vantagem no uso de *scripts* compilados é a possibilidade de o cliente não ter acesso ao código fonte do programa (*script*), a menos que seja essa a intenção do programador.

Dependendo do pacote instalado, talvez seja necessário indicar junto ao nome do compilador o número da versão em uso do interpretador Lua.

Conclusão

Neste artigo é apresentado o uso de alguns recursos como complementos da série de artigos publicados para a linguagem Lua.

Os exemplos descritos foram gerados em situações decorrentes de um curso da linguagem ministrada pelo autor a partir de dúvidas manifestadas pelos alunos.

Em outras ocasiões poderão ser apresentadas outras possibilidades de uso da linguagem Lua. Até lá.

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.

Tipos de dados int e variantes na linguagem C

Introdução

Este artigo foca os tipos de dados *int* e variantes disponíveis na linguagem de programação C. Na parte inicial, o artigo apresenta os tipos de dados inteiros ditos tradicionais. Seguidamente, o artigo introduz os tipos inteiros orientados para a portabilidade, tais como o int8_t, uint_fast64_t e similares. Os principais conceitos são ilustrados com exemplos, executados, sempre que conveniente, em duas plataformas Linux: uma plataforma de 32 bits Lubuntu 16.04 com a versão 5.3.1 do compilador gcc 5.3.1, e uma plataforma de 64 bits Lubuntu 17.04 com o gcc 6.3.0. A primeira é designada por L32, a segunda por L64. Note-se que ambos as versões do compilador usam nativamente a norma C11 (2011) da linguagem C.

Tipos de dados *int*

Como em muitas outras linguagens de programação, a linguagem C define um conjunto de tipos de dados inteiros. São exemplos os tipos de dados signed char, signed short, signed int e signed long e as variantes sem sinal, unsigned char, unsigned short, unsigned int e unsigned long. Na revisão à linguagem designada por norma C99 foi acrescentado o tipo inteiro long long, nas variantes com (signed) e sem (unsigned) sinal.

O modificador signed é assumido por omissão para os tipos int, com exceção do tipo char como é explicado mais adiante. Assim, a declaração int A designa a variável A como sendo um inteiro com sinal, sendo equivalente a signed int A. Caso se pretenda uma variável int sem sinal, torna-se necessário o uso do modificador unsigned, como por exemplo, na declaração unsigned int B.

Uma variável, qualquer que seja o seu tipo, ocupa sempre um determinado espaço na memória, espaço medido em octetos. A designação octeto deriva do facto de representar oito bits, sendo frequentemente empregue a designação *byte* em sua substituição. Na linguagem C, o operador sizeof devolve o tamanho em octetos, ocupado por um determinado tipo de dado. O tipo de dado pode ser indicado diretamente no sizeof através do nome do tipo de dados (e.g., sizeof(short)) ou indiretamente através de uma variável (e.g., *int A;* sizeof(A)). No caso de se fazer uso de uma variável, os parêntesis podem ser omitidos no sizeof (e.g, *sizeof A* é equivalente a sizeof(A)). O operador sizeof devolve um valor do tipo size_t. Na prática, o tipo size_t corresponde a um inteiro sem sinal. O uso do size_t em lugar de um inteiro sem sinal tem a vantagem de indicar a quem interage com o código fonte que o valor corresponde a um tamanho.

A contabilização do número de bits de um determinado tipo de dados obtém-se multiplicando o valor devolvido pelo operador sizeof por oito. A Listagem 1 mostra o código fonte do programa sizeof, programa que lista os tamanhos, em octetos, reportados pelo operador sizeof para os tipos de dados *inteiro* com sinal e ainda para o tipo ponteiro. Este último representa o tamanho dos endereços na plataforma considerada. Note-se que os tipos de dados *inteiro* sem sinal ocupam o mesmo espaço que as variantes com sinal. Como é abordado mais adiante (Tabela 3 e Tabela 4), o par modificador/especificador de formato do printf (e variantes) para o tipo de dado size_t é o %zu, conforme se pode ver na Listagem 1.

A Listagem 2 mostra o resultado da execução do programa sizeof.c numa plataforma Linux de 32 bits (L32). Por sua vez, a Listagem 3 mostra a saída do programa produzida na plataforma Linux de 64 bits (L64). Em termos de resultados, as diferenças entre as duas plataformas situam-se no tamanho do long e do long long: 4 octetos na plataforma de 32 bits, 8 octetos na plataforma de 64 bits. Observa-se ainda que para cada plataforma, os tipos long e long long ocupam o mesmo número de octetos: quatro na plataforma L32 e oito na plataforma L64.

```
int main(void){
    printf("Tamanho 'int' (octetos) \n");
    printf("char: %zu\n",sizeof(char));
    printf("short: %zu\n", sizeof(short));
    printf("int: %zu\n", sizeof(int));
    printf("long: %zu\n", sizeof(long));
    printf("long long= %zu\n",
                    sizeof(long long));
    printf("sizeof(ponteiro)= %zu\n",
                        sizeof(void*));
    return 0;
}
```

Listagem 1: código fonte do programa sizeof.c

```
Tamanho 'int' (octetos)
char= 1
short= 2
int= 4
long= 4
long long= 8
ponteiro= 4
```

Listagem 2: resultado da execução do programa sizeof.c na plataforma L32

```
Tamanho 'int' (octetos)
char= 1
short= 2
int= 4
long= 8
long long= 8
ponteiro= 8
```

Listagem 3: resultado da execução do programa sizeof.c na plataforma L64

O tamanho exato para os tipos de dados int e variantes não está definido na linguagem C. De facto, com exceção do tipo de dado char que ocupa sempre oito bits (um octeto),

a norma da linguagem C não é rigorosa a respeito do tamanho ocupado por cada um desses tipos de dados. Nesse domínio, a linguagem C especifica que o tipo short ocupa pelo menos dois octetos, o tipo long requer pelo menos quatro octetos e o tipo long long ocupa um mínimo de oito octetos (Tabela 1). Essas regras aplicam-se de igual forma às variantes com e sem sinal. A linguagem C define ainda que em qualquer plataforma, o tipo short deve ocupar menos ou tantos bits que o tipo int. O mesmo relacionamento de tamanho deve ocorrer com o tipo int relativamente ao tipo long (i.e., sizeof(int) <= sizeof (long)) , e com o tipo long em relação ao tipo long long.

Tipo de dado (com/sem sinal)	Requisito
sizeof(short)	>= 2 octetos
sizeof(long)	>= 4 octetos
sizeof(long long)	>= 8 octetos

Tabela 1: tamanhos mínimos para armazenar short, long e long long

Apesar do tamanho dos tipos de dados inteiros não estar definidos, a linguagem C disponibiliza constantes do pré-processador referentes aos valores mínimos e máximos de cada um tipo de dados inteiros. Por exemplo, o valor mínimo suportado por um *int* é representado pela constante INT_MIN, e o valor máximo por INT_MAX. Para um *unsigned int*, existe somente a constante UINT_MAX, pois o valor mínimo é obviamente zero.

Tipo de dado	Mínimo	Máximo
signed char	SCHAR_MI	SCHAR_MAX
unsigned char	0	UCHAR_MAX
short	SHORT_MI	SHORT_MAX
unsigned short	0	USHORT_MAX
int	INT_MIN	INT_MAX
unsigned int	0	UINT_MAX
long	LONG_MIN	LONG_MAX
long long	LLONG_MI	LLONG_MAX
unsigned long long	0	ULLONG_MAX
size_t	0	SIZE_MAX

Tabela 2: constantes que representam os valores mínimos e máximos dos tipos de dados inteiros (com e sem sinal)

A Tabela 2 lista as constantes do pré□processador que definem os valores mínimos e máximos dos tipos de dados inteiros. Estas constantes estão definidas no ficheiro <limits.h>, com exceção de SIZE_MAX que se encontra definida no ficheiro <stdint.h>.

Importa notar que quando são ultrapassados os valores extremos de uma variável int ou variante (char, short, int, long e long long) ocorre uma situação de transbordo (*overflow* na designação anglo-saxónica). No caso da norma da linguagem C, o comportamento no transbordo de variáveis do tipo inteiro apenas se encontra definido para tipos inteiros sem sinal. Para o caso de um tipo de dado com sinal (e.g., *signed short*), o transbordo de uma variável inteira com sinal depende do compilador. De qualquer forma, é fundamental que um programa esteja devidamente preparado para evitar situações de transbordo de variáveis int e variantes. O transbordo de variáveis inteiras pode ter consequências graves como o término súbito da aplicação, ou pior ainda, a produção de resultados errados.

```c
#include <stdio.h>
#include <limits.h>
int main(void){
    unsigned short int a = USHRT_MAX;
    short int b = SHRT_MAX;
    printf("[antes transbordo]\n");
    printf("a=%hu\n", a);
    printf("b=%hd\n", b);
    a++; // provoca transbordo
    b++; //provoca transbordo
    printf("[depois transbordo]\n");
    printf("a=%hu\n", a);
    printf("b=%hd\n", b);
    return 0;
}
```

Listagem 4: código fonte do programa transbordo.c

```
[antes transbordo]
a=65535
b=32767
[depois transbordo]
a=0
b=-32768
```

Listagem 5: saída da execução do programa transbordo

A Listagem 4 mostra o código do programa transbordo.c, no qual é forçado um transbordo em duas variáveis do tipo *short*: uma com sinal (variável **a**), outra sem sinal (variável **b**). O código da Listagem 4 faz uso dos especificadores %hu e %hd na chamada à função printf, respetivamente, para formatar a representação de um unsigned short e de um signed short. A saída da execução do programa transbordo.c na plataforma L32 é mostrado na Listagem 5. No programa transbordo.c verifica-se que o acréscimo de uma unidade à variável **a**, do tipo unsigned short, previamente inicializada com USHRT_MAX, provoca um transbordo, levando a que o valor da variável **a** passe a ser zero.

Para o tipo de dados char, para além de SCHAR_MIN/SCHAR_MAX, existem ainda as constantes CHAR_MIN e CHAR_MAX. A existência dessas duas constantes decorre do facto da norma do C não associar à designação char a existência ou não de sinal. Assim, a designação char poderá corresponder à variante com sinal em determinados compiladores e à variante sem sinal noutros compiladores. Não estando definido na norma, nada deve ser assumido pelo programador. Deste modo, os tipos de dados char, signed char e unsigned char são formalmente diferen-

tes uns dos outros. Assim, sempre que seja necessário guardar valores negativos numa variável do tipo char, deve ser empregue o signed char. Ao invés, caso se pretenda guardar um valor superior a 127 (e inferior a 256), deve ser empregue o tipo unsigned char. É possível determinar através dos valores de CHAR_MAX e SCHAR_MAX, se a plataforma em uso define, por omissão, um char com ou sem sinal. Assim, se CHAR_MAX for igual a UCHAR_MAX, a designação char definida na plataforma não tem sinal, caso contrário, o char definido na plataforma tem sinal, pois CHAR_MAX é igual a SCHAR_MAX. A Listagem 6 apresenta o código do programa sinal_char.c que indica se existe sinal ou não no tipo de dado char. Nas duas plataformas de testes, L32 e L64, o char é um tipo de dado com sinal.

```
#include <stdio.h>
#include <limits.h>
int main(void){
    if( CHAR_MAX == UCHAR_MAX ){
        printf("char: SEM sinal\n");
    }else if( CHAR_MAX == SCHAR_MAX ){
        printf("char: COM sinal\n");
    }else{
        printf("char: indefinido\n");
    }
    return 0;
}
```

Listagem 6: código fonte do programa *sinal_char.c*

Os tipos intmax_t e uintmax_t

A norma C99 requer, com carácter obrigatório, a existência dos tipos de dados intmax_t e uintmax_t. Ambos representam o tipo de dado inteiro que tem o maior número de bits na plataforma considerada. O tipo de dado intmax_t está associado à variante com sinal, e uintmax_t à variante sem sinal. O especificador de formatação do printf e variantes é é o j (%jd ou %ji) para o tipo intmax_t e z para o tipo uintmax_t (%z{u,o,x,X}). Os valores mínimos e máximos do tipo intmax_t são respetivamente, INTMAX_MIN e INTMAX_MAX. Para o tipo uintmax_t, zero é o valor mínimo e UINTMAX_MAX o valor máximo. O programa intmax.c (Listagem 7) exemplifica o uso de intmax_t e uintmax_t, tanto ao nível dos valores máximos como dos modificadores da função printf. O programa mostra ainda o número de octetos necessários ao armazenamento de um intmax_t: 8 octetos em ambas as plataformas, L32 e L64 (Listagem 8).

```
#include <stdio.h>
#include <stdint.h>
int main(void){
    intmax_t intmax_A;
    intmax_A = INTMAX_MAX;
    uintmax_t uintmax_B;
    uintmax_B = UINTMAX_MAX;
    printf("INTMAX_MAX=%jX\n", intmax_A);
    printf("UINTMAX_MAX=%jX\n",
                        uintmax_B);
    printf("sizeof(intmax_t)=%zu"
         " octetos\n", sizeof(intmax_t));
    return 0;
}
```

Listagem 6: código fonte do programa *sinal_char.c*

Os tipos intmax_t e uintmax_t

A norma C99 requer, com carácter obrigatório, a existência dos tipos de dados intmax_t e uintmax_t. Ambos representam o tipo de dado inteiro que tem o maior número de bits na plataforma considerada. O tipo de dado intmax_t está associado à variante com sinal, e uintmax_t à variante sem sinal. O especificador de formatação do printf e variantes é é o j (%jd ou %ji) para o tipo intmax_t e z para o tipo uintmax_t (%z{u,o,x,X}). Os valores mínimos e máximos do tipo intmax_t são respetivamente, INTMAX_MIN e INTMAX_MAX. Para o tipo uintmax_t, zero é o valor mínimo e UINTMAX_MAX o valor máximo. O programa intmax.c (Listagem 7) exemplifica o uso de intmax_t e uintmax_t, tanto ao nível dos valores máximos como dos modificadores da função printf. O programa mostra ainda o número de octetos necessários ao armazenamento de um intmax_t: 8 octetos em ambas as plataformas, L32 e L64 (Listagem 8).

```
#include <stdio.h>
#include <stdint.h>
int main(void){
    intmax_t intmax_A;
    intmax_A = INTMAX_MAX;
    uintmax_t uintmax_B;
    uintmax_B = UINTMAX_MAX;
    printf("INTMAX_MAX=%jX\n", intmax_A);
    printf("UINTMAX_MAX=%jX\n",
                        uintmax_B);
    printf("sizeof(intmax_t)=%zu"
         " octetos\n", sizeof(intmax_t));
    return 0;
}
```

Listagem 7: código fonte do programa *intmax.c*

```
INTMAX_MAX=7FFFFFFFFFFFFFFF
UINTMAX_MAX=FFFFFFFFFFFFFFFF
sizeof(intmax_t)=8 octetos
```

Listagem 8: saída da execução do programa intmax.c na plataforma L32 e L64

Representação de constantes inteiras

A linguagem C admite o uso de três bases numéricas distintas – octal, decimal e hexadecimal – para a representação de constantes inteiras. Uma constante no formato octal inicia-se por um zero à esquerda (e.g., 0123), e apenas contém dígitos de 0 a 7, pois a base tem apenas oito símbolos. Uma constante no formato hexadecimal inicia-se pela sequência de caracteres 0x (e.g., 0x123), com o conjunto de símbolos válidos a incluir os 10 dígitos e as letras de A a F. Por fim, uma constante em formato decimal tem como dígito mais à esquerda, qualquer dígito exceto o zero.

Por omissão, as constantes inteiras em formato decimal são interpretadas pelo compilador como sendo do tipo int. Caso o valor seja maior do que o suportado pelo tipo int, o compilador interpreta a constante como sendo de um tipo de dados com maior abrangência, seja do tipo long, se este for suficiente, ou do tipo long long.

Caso pretenda forçar uma dada interpretação a uma constante inteira, o programador deve fazer uso de sufixos.

Assim, para que uma constante inteira seja interpretada pelo compilador como inteiro sem sinal faz-se uso do sufixo U (e.g. 999999U). Podem ainda ser empregues os sufixos L e LL para solicitar, respetivamente, a interpretação de uma constante como long (L) e long long (e.g., 9999999999LL). Os sufixos de tipo (L e LL) e de sinal (U) podem ser combinados (e.g, 999999999LU ou 99999999UL). Finalmente, os sufixos podem ser representados com letra minúscula ou maiúscula sem qualquer perda de significado.

Especificadores de formatação para tipos de dados *int*

Para representar tipos de dados no formato de string, a linguagem C disponibiliza a função printf e variantes. As variantes incluem, entre outras, a função fprintf (escrita formatada para ficheiro), e a função snprintf (escrita formatada para uma zona de memória). A formatação para string requer que seja indicada na chamada às funções printf e variantes a string de formatação correspondente ao tipo de dado que se pretende representar. Por exemplo, para uma variável do tipo signed int, deve ser indicado %d ou %i, e %u para uma variável do tipo unsigned int, caso se pretenda a representação decimal. Outros especificadores podem ser empregue caso se pretenda outros formatos de saída (e.g., %x ou %X para ter saída em formato hexadecimal). A Tabela 3 mostra os especificadores de formatação para os tipos de dados básicos da linguagem C. Os especificadores indicados podem ainda ser empregues com a função scanf e variantes (fscanf, sscanf, etc.).

Especificador	Saída	Alvo
%d ou %i	decimal	com sinal
%u	decimal	sem sinal
%o	octal	sem sinal
%x	hexadecimal (minúsculas)	sem sinal
%X	hexadecimal (maiúsculas)	sem sinal

Tabela 3: Especificadores de formato para valores inteiros a serem empregues na função *printf* e variantes

Modificador	Tipo de dado inteiro
hh	char
h	short
l	long
ll	long long
j	intmax_t uintmax_t
z	size_t

Tabela 4: modificadores a serem combinados com especificador de formato para tipos de dados inteiros

```
#include <stdio.h>
#include <limits.h>
int main(void){
      char char_A = SCHAR_MAX;
      short int short_B = SHRT_MAX;
      int int_C = INT_MAX;
      long int long_D = LONG_MAX;
      long long int long_E = LLONG_MAX;

      printf("char_A=%hhd\n", char_A);
      printf("short_B=%hd\n", short_B);
      printf("int_C=%d\n", int_C);
      printf("long_D=%ld\n", long_D);
      printf("long_E=%lld\n", long_E);
      printf("--------------------\n");
      printf("char_A=%hho\n", char_A);
      printf("short_B=%ho\n", short_B);
      printf("int_C=%o\n", int_C);
      printf("long_D=%lo\n", long_D);
      printf("long_E=%llo\n", long_E);
      printf("--------------------\n");
      printf("char_A=%hx\n", char_A);
      printf("short_B=%hx\n", short_B);
      printf("int_C=%x\n", int_C);
      printf("long_D=%lx\n", long_D);
      printf("long_E=%llx\n", long_E);
      return 0;
}
```

Listagem 9: código fonte do programa *especificadores.c*

Para os tipos de dados inteiros exceto o int, i.e., para os tipos char, short, long e long long é necessário preceder o especificador de formato com um apropriado modificador. Por exemplo, para mostrar através do printf o valor de uma variável do tipo short em formato decimal, deve indicar-se a string de formatação "%hd" ou "%hi". A lista de modificadores que podem ser combinados com os especificadores de formato na função printf e variante encontra-se na Tabela 4. O programa especificadores.c, mostrado na Listagem 9, faz uso dos especificadores e modificadores para mostrar em três formatos – decimal, octal e hexadecimal – os valores máximos para variáveis com sinal do tipo char, short, int, long e long long. A saída do programa executado na plataforma L32 é mostrada na Listagem 10, e na Listagem 11 para a execução na plataforma L64.

```
char_A=127
short_B=32767
int_C=2147483647
long_D=2147483647
long_E=9223372036854775807
--------------------
char_A=177
short_B=77777
int_C=17777777777
long_D=17777777777
long_E=777777777777777777777
--------------------
char_A=7f
short_B=7fff
int_C=7fffffff
long_D=7fffffff
long_E=7fffffffffffffff
```

Listagem 10: saída do programa especificadores.c executado na plataforma L32

```
char_A=127
short_B=32767
int_C=2147483647
long_D=9223372036854775807
long_E=9223372036854775807
```

```
------------------
char_A=177
short_B=77777
int_C=17777777777
long_D=7777777777777777777
long_E=7777777777777777777
------------------
char_A=7f
short_B=7fff
int_C=7fffffff
long_D=7fffffffffffffff
long_E=7fffffffffffffff
```

Listagem 11: saída do programa especificadores.c executado na plataforma L64 (as diferenças com a plataforma L32 estão sublinhadas)

Tipos de dados *int* com tamanho definido

Embora as constantes do pré-processador indicadas na Tabela 2 permitam determinar os valores mínimos e máximos dos tipos de dados inteiros, a variabilidade do respetivo tamanho torna mais difícil a programação em certas situações. Por exemplo, caso se pretenda fazer uso de uma variável inteira para atribuir a cada bit um determinado significado (e.g., bit **4**: estado da luz da cozinha; bit **5**: estado da luz da sala; etc.) é de todo conveniente fazer uso de um tipo de dados cujo tamanho é conhecido, de modo a ter conhecimento de antemão do número de bits disponíveis. Existem ainda muitas outras situações em que é necessário fazer uso de uma variável com um tamanho exato de bits (e.g., 32 bits), enquanto noutras situações será suficiente o uso de variáveis do tipo inteiro que tenham pelo menos um determinado número de bits. Desde a norma C99, que a linguagem C disponibiliza vários tipos de dados inteiro em que é garantido o tamanho, seja de forma fixa ou através de um majorante. Esses tipos de dados encontram-se acessíveis a partir do ficheiro <stdint.h>, e são descritos nas próximas secções.

Tipos de dados intN_t e uintN_t

Um dos tipos de dados inteiro definido pela norma C99 é o inteiro de tamanho exato. O formato dos identificadores de dados é intN_t para tipos de dados com sinal e uintN_t para tipos de dados sem sinal, em que N indica o número de bits do tipo de dados. Concretamente, são definidos os tipos de dados int8_t, int16_t, int32_t, int64_t e os correspondentes tipos sem sinal, uint8_t, uint16_t, uint32_t e uint64_t. Para esses tipos de dados encontram-se disponíveis constantes do pré-processador indicadoras dos valores mínimos e máximos. O formato das constantes é INTn_MIN, INTn_MAX e UINTn_MAX em que ***n*** corresponde ao número de bits do tipo de dado. Por exemplo, para o tipo int32_t, as constantes dos valores mínimos e máximos são INT32_MIN e INT32_MAX.

A norma define ainda os especificadores de formatação a serem empregues com as funções do tipo printf para os dados do tipo intN_t e uintN_t. Os especificadores de tipo obedecem ao formato PRI + f + n, em que f define o formato pretendido e n corresponde ao número de bits do tipo de dados. O formato pretendido pode ser d (decimal, equivalente a %d), i (inteiro, %i), o (octal, %o), u (sem sinal, %u), x (hexadecimal com as letras em minúsculas, %x) e X (hexadecimal com as letras em maiúsculas, %X). Assim, para um inteiro de 32 bits, ter-se-á, consoante a saída pretendida: PRId32, PRIi32, PRIo32, PRIu32, PRIx32 e PRIX32. Os especificadores estão definidos no ficheiro <inttypes.h>. O programa printf_PRI.c (Listagem 12) exemplifica o uso dos especificadores de formatação PRId8, PRIu8 e PRIx64. Por serem eles próprios *strings*, os especificadores de formatação do tipo PRIfn devem ser indicados de forma especial numa chamada à função printf (ou variante). Assim, o especificador PRIfn deve ser precedido do tradicional símbolo de %, fecho de string ("), sendo a string reaberta (novo ") após a indicação do especificador. A Listagem 12 exemplifica o uso de especificadores do tipo PRIfn na chamada à função printf.

Para a função scanf (e variantes), os especificadores têm a designação SCN em lugar de PRI, mantendo-se o resto do formato.

```c
#include <stdio.h>
#include <limits.h>
#include <inttypes.h> // inclui <stdint.h>

int main(void){
    int8_t int8_var = INT8_MAX;
    uint8_t uint8_var = UINT8_MAX;
    int64_t int64_var = INT64_MAX;
    uint64_t uint64_var = UINT64_MAX;

    printf("int8_var=%" PRId8 " \n",
                                int8_var);
    printf("uint8_var=%" PRIu8 " \n",
                                uint8_var);
    printf("int64_var=%" PRIx64 " \n",
                               int64_var);
    printf("uint64_var=%" PRIx64 " \n",
                              uint64_var);
    return 0;
}
```

Listagem 12: programa printf_PRI.c

```
int8_var=127
uint8_var=255
int64_var=7fffffffffffffff
uint64_var=ffffffffffffffff
```

Listagem 13: resultado da execução do programa printf_PRI.c na plataforma L32

Finalmente, importa observar que a norma C99 indica que os tipos de dados intN_t e uintN_t são opcionais, pelo que poderão não estar disponíveis nalguns compiladores.

Tipos de dados int_leastN_t e uint_leastN_t

Outros dos tipos de dados int definidos pela norma C99 são int_leastN_t e a variante sem sinal uint_leastN_t. Como o nome indica, o tipo de dados int_leastN_t tem, pelo menos, N bits de tamanho. A norma define que os tipos int_leastN_t e uint_leastN_t são obrigatórios para os valores habituais de N, isto é, 8, 16, 32 e 64. Facultativamente, também poderão existir para outros valores de N, como, por exemplo, N=128, sob a forma int_least128_t e uint_least128_t. Similarmente ao que sucede para outros

tipos de dados inteiros, os valores mínimos e máximo encontram-se definido através de constantes do pré-processador com o seguinte padrão: INT_LEASTn_MIN, INT_LEASTn_MAX para a variante com sinal e UINT_LEASTn_MAX para a variante sem sinal. Por exemplo, para n=64, tem-se INT_LEAST64_MIN, INT_LEAST64_MAX e UINT_LEAST64_MAX.

Os especificadores para a função printf e variantes para os tipos int_leastN_t e uint_leastN_t são similares aos anteriormente vistos, obedecendo ao formato PRIfLEASTn, em que f corresponde ao formato pretendido – d ou i para tipo com sinal e o, u, x, X para tipos sem sinal –, e ***n*** indica a largura mínima do tipo de dado. Por exemplo, para o tipo int_least64_t tem-se PRIdLEAST64, e PRIuLEAST64 para o tipo uint_least64_t. Para a função scanf e variantes, os especificadores são os mesmos, exceto que o prefixo PRI é substituído por SCN.

Tipos de dados int_fastN_t e uint_fastN_t

Desde a norma C99 que a linguagem C define ainda o tipo de dados int_fastN_t e a variante sem sinal uint_fastN_t. A palavra *fast* resume o propósito deste tipo de int: disponibilizar a variante mais rápida de inteiro que tenha pelo menos N bits de tamanho. Por exemplo, numa plataforma em que o acesso e manipulação de inteiros se faça nativamente com 64 bits, a variante int_fast32_t pode, por razões de desempenho, ser internamente implementada por um inteiro de 64 bits. Ao contrário do tipo int_leastN_t e variantes, a implementação do tipo int_fastN_t (com e sem sinal) privilegia a velocidade em detrimento, se necessário, do espaço ocupado em memória.

À semelhança do tipo int_leastN_t, um compilador conforme à norma C99 ou C11 deve obrigatoriamente implementar os tipos int_fastN_t/uint_fastN_t e variantes para os valores de N: 8,16, 32 e 64. Os valores mínimos e máximos para as variantes com e sem sinal são expressos pelas constantes INT_FASTn_MIN, INT_FASTn_MAX e UINT_FASTn_MAX. Os especificadores para a função printf e variantes são mostrados na Tabela 5. O programa printf_LEAST_FAST.c (Listagem 14) exemplifica o uso dos especificadores de formato para os tipos int_leastN_t / uint_leastN_t e int_fastN_t / uint_fastN_t.

Tipo	Especificadores *printf*
int_leastN_t	PRI{d,i}LEAST**N**
uint_leastN_t	PRI{u,o,x,X}LEAST**N**
int_fastN_t	PRI{d,i}FAST**N**
uint_fastN_t	PRI{u,o,x,X}FAST**N**

Tabela 5: especificadores para printf e variantes para formatos do tipo LEAST e FAST

```
#include <stdio.h>
#include <limits.h>
#include <inttypes.h>
int main(void){
  int_least32_t int_least32_var =
                    INT_LEAST32_MIN;
  uint_least32_t uint_least32_var =
                   UINT_LEAST32_MAX;
  int_fast32_t int_fast32_var =
                    INT_FAST32_MIN;
  uint_fast32_t uint_fast32_var =
                   UINT_FAST32_MAX;
  printf("int_least32_var=%" PRIdLEAST32
                  "\n", int_least32_var);
  printf("uint_fast32_var=%" PRIuLEAST32
                 "\n", uint_least32_var);
  printf("int_fast32_var=%" PRIxFAST32
                   "\n", int_fast32_var);
  printf("uint_fast32_var=%" PRIXFAST32
                 "\n", uint_fast32_var);
  return 0;
}
```

Listagem 14: programa printf_LEAST_FAST.c

Nota final

Este artigo abordou os tipos de dados inteiro disponíveis na linguagem de programação C. Considerando a importância do correto uso e dimensionamento dos tipos de dados inteiro é de todo relevante que os utentes da linguagem C sejam conhecedores das diversas funcionalidades e limitações de cada tipo, bem como das omissões existentes nas várias normas da linguagem. Aos leitores interessados sugere-se a leitura de (Gustedt, 2017) e (Peter & Tony, 2015).

Bibliografia

Gustedt, J. (2017). *Modern C.* Online. Retrieved julho 2017, from http://icube-icps.unistra.fr/img_auth.php/d/db/ModernC.pdf

Peter, P., & Tony, C. (2015). *C in a Nutshell* (2nd ed.). O'Reilly Media, Inc.

AUTOR

Escrito por Sandro Patrício Domingues

é professor do Departamento de Engª Informática na Escola Superior de Tecnologia e Gestão (ESTG) do Instituto Politécnico de Leiria (IPLeiria). Tem lecionado, entre outras, as disciplinas de Programação Avançada e Sistemas Operativos da Licenciatura em Engenharia Informática. É ainda docente da pós-graduação em Informática de Segurança e Computação Forense.

Escrito por Vítor Távora

é professor do Departamento de Engenharia Informática na ESTG / Politécnico de Leiria. Leciona disciplinas de programação a vários cursos de licenciatura da Escola Superior de Tecnologia e Gestão do Politécnico de Leiria. Leciona ainda os módulos de Programação e Programação Segura à Pós-Graduação em Informática de Segurança e Computação Forense (ISCF) do Instituto Politécnico de Leiria.

Feed RSS em C# .NET Core no Azure Web App em Linux

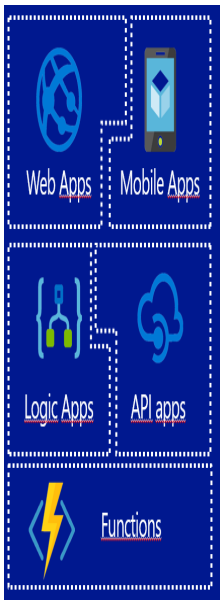
Neste artigo vou demonstrar como criar uma aplicação Web Model-view-controller (MVC) que vai ler o Feed RSS dos artigos da Revista PROGRAMAR em C# .NET Core 1.1 para Docker. A aplicação vai ser disponibilizada no Azure Web App em Linux através Docker Hub.

Azure Web App

O serviço Azure Web App permite que uma aplicação web desenvolvida em .NET, .NET Core, Java, Node.js, PHP, Python e Ruby esteja disponível em qualquer utilizador através da Internet. Para alguns o Web App é uma forma de disponibilizar uma página de Internet ou framework mas permite muito mais do que isso mais a frente vou demonstrar algumas funcionalidades.

O Web Apps está integrado no Azure App Service que é um conjunto de serviços.

- Web Apps, permite alojar websites e aplicações Web.
- Mobile Apps, alojar aplicações mobile a back-ends.
- Logic Apps, automatizar processos empresariais e integrar sistemas e dados no Azure sem escrever código.
- API apps, para alojar RESTful APIs.
- Functions, executa pequenos pedaços de código ou "funções" no Azure.



Os recursos não são reservados a cada serviço os mesmos podem utilizar as funcionalidades dos outros serviços.

O Web App tem tudo o que necessita para construir uma aplicação esta oferece ao administrador de sistema, desenvolvedor ou DevOps uma plataforma totalmente gerida em que se tem acesso as várias funcionalidades por exemplo:

- Always On, mantem a Web App ativa está disponível a partir do Standard tier.
- Definições da aplicação e Connection Strings.
- Registos de auditoria de servidor Web, ficheiros, mensagens de erro detalhadas e pedidos solicitados.
- Backups Manuais e/ou agendados.
- Gestão através de linhas de comandos com Azure PowerShell ou Azure CLI.
- Deployment a partir do Visual Studio Team Services, OneDrive, Git, GitHub, Bitbucker, Dropbox, e outros repositórios externos.
- Monotorização e diagnósticos.
- Múltiplas Framework disponíveis .NET, PHP, Java, Python e Node.js.
- Testes de desempenho.
- Debug remota e por consola.
- Controlo de acesso por funções, Role-Based Access Control (RBAC).
- Scaling Up, Out ou automático.
- Plataforma 32 ou 64 bit.
- Extensões, por exemplo "Let's Encrypt" que está disponível a partir do tier básico requer Server Name Indication (SNI).

  Site Slots. pode ser utilizado como cilo de vida da aplicação.

- Certificados SSL.

  WebJobs, executa tarefas manuais, agendadas, trigger ou contínuo.

Através da plataforma pode ativar ou desativar funcionalidades pretendidas e o Azure automaticamente faz toda essa configuração por nós. Isto é possível porque o Web App é uma plataforma como serviço (Plaform-as-a-Service) apenas gerimos a aplicação e os dados todos. A infraestrutura desde da rede, armazenamento, máquina virtual, sistema operativos, software necessário para executar a aplicação desde do IIS, Apache, Tomcat, ou outro é totalmente gerido pelo Azure e não temos que nos preocupar com manutenção e atualizações, mas a segurança da aplicação é da nossa responsabilidade.

O Azure já dispõe da sua versão em Linux que funciona com máquinas virtuais em Linux, antes estava apenas disponível em Windows. Agora podemos escolher se queremos executar a aplicação num ambiente Windows ou Linux.

A versão em Linux foi um dos pedidos mais solicitadas pelos clientes/utilizadores porque há frameworks como o

Wordpress que funciona muito melhor e mais rapidamente em Linux do que em Windows.

A versão Windows e Linux não são iguais, existem diferenças entre eles. A grande diferença é que em Linux é utilizado o Docker o que permite que se possa utilizar o nosso próprio Container que é um recipiente que contem todo o que é necessário para a aplicação ser executada. O Docker Hub é um repositório permite que a Web App seja atualizada automaticamente quando atualizamos a aplicação no Docker Hub, mas também podemos fazer o deployment da aplicação por FTP, Git, Bitbucket.

Outra diferença entre a opção Windows e Linux é o suporte de linguagens de programação em Linux apenas suporta (quando este artigo foi escrito) Node.js 4.4, 4.5, 6.2, 6.6, 6.9-6.11 e 8.0-8.1, o PHP 5.6 e 7.0, .NET Core 1.0-1.1 e o Ruby 2.3. Enquanto a versão Windows tem mais suporte e inclui o Java 7 e 8 e Python 2.7 e 3.4 mas não suporta Ruby.

Como funciona a arquitetura do serviço

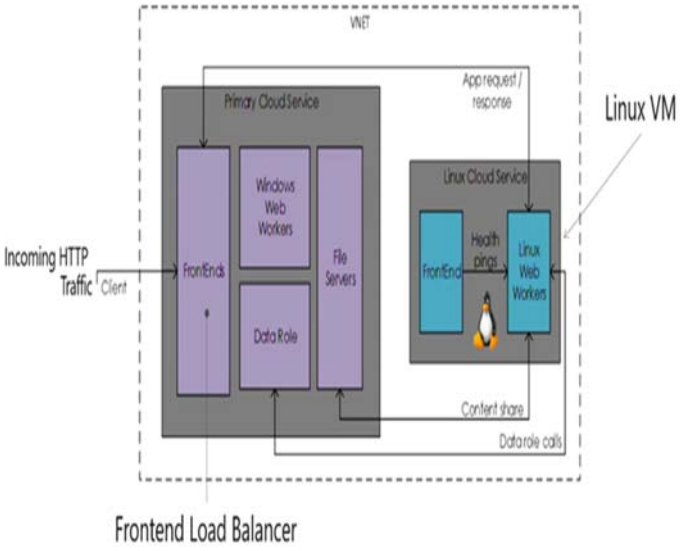


Figura 1 Arquitetura do Serviço

O Azure tem dois clusters de computação na mesma rede virtual (VNET) o cluster primário é composto principalmente por tecnologia Microsoft que dispõem de um frontend load balancer que recebe todo tráfego Hypertext Transfer Protocol (HTTP) recebido que por sua vez reencaminha para máquinas ou serviços em ambiente Windows ou Linux, tem o Windows Web Workers que é utilizado para aplicações Windows, um serviço de armazenamento e muito mais. O segundo cluster apenas tem máquinas virtuais com Linux e é neste cluster que vão estar as aplicações Web App em Linux.

O que contem a máquina virtual em Linux?

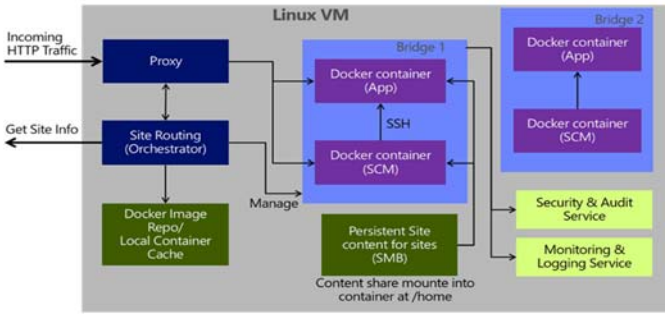


Figura 2 Máquina Virtual Linux

A máquina contem um proxy que recebe todo o tráfego recebido pelo FrontEnd do cluster principal e o mesmo vai encaminhar para a aplicação a informação solicitada e enviada pelo Site Routing mas este faz muito mais do que enviar a informação ao cliente. O Site Routing gera a aplicação, se verificar que existe uma nova versão ele vai atualizar a mesma.

O Container não inclui a informação persistente da aplicação. Cada Web App tem um serviço 'SCM' associado, esse serviço é o KUDU que permite ter acesso a painéis de controlo com linhas de comandos, debug, diagnósticos, processos em execução ou seja permite fazer o deployment rápido. Kudu é um projeto de código aberto e que está disponível no GitHub (https://github.com/projectkudu/kudu) este serviço é executado num contentor a parte.

Figura 3 Azure Kudu

Os dois Containers estão separados através do isolamento de segurança do Docker mas integrados numa única bridge e todas as bridges estão associadas a um plano do serviço (Azure Service Plan).

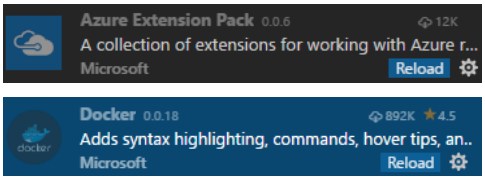
Os dois Containers acedem ao conteúdo da aplicação através do Message Block (SMB) de forma persistente o que significa que quando existe uma alteração é refletida em todos os lugares. Se estiver a executar uma segunda aplicação Web App é criado uma segunda Bridge e por sua vez é criado um novo contentor Docker para aplicação e SCM.

Azure Web App para Containers

Além do Azure Web App em Linux também existe o Azure Web App para Containers mas qual é diferença entre eles? Basicamente o Web App para Containers permite utilizar o nosso próprio Container e efetuar deployment automaticamente, por nós, numa Web App para Linux, ou seja, é uma solução baseada em Web App em Linux mas com a vantagem que todo o processo vai ser feito automaticamente pelo Azure o que permite eliminar as tarefas de gestão. Continuamos a ter acesso à configuração do domínio, certificados SLL e muito mais. É um ambiente ideal para desenvolvedores e DevOps.

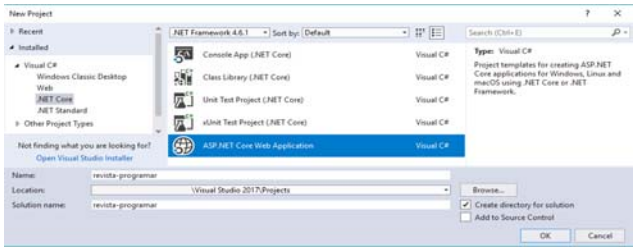
Deployment de um Docker Container para Azure Web App em Linux

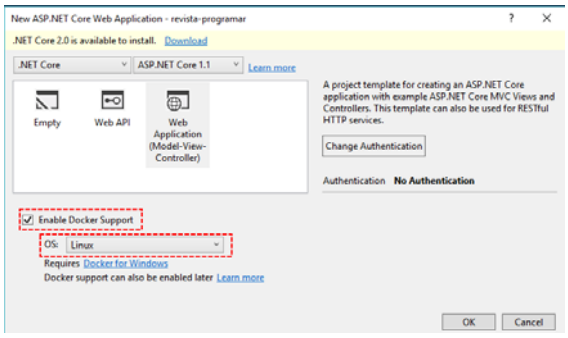
Antes de iniciar é necessário ter instalado o Docker Community Edition que pode obter na página oficial do Docker Store em https://store.Docker.com/ é compatível com vários sistemas operativos, o .NET Core pode ser obtido em https://www.microsoft.com/net/download/core apesar de ser compatível com o Linux por experiência própria não funciona muito bem com o Fedora 23 e 24 devido a bibliotecas que o próprio Fedora utiliza pode alterar mas algumas aplicações instaladas podem deixar de funcionar. O ambiente de desenvolvimento integrado (IDE), recomendo a utilização do Microsoft Visual Studio 2017 se utilizar ambientes Linux recomendo o Microsoft Visual Studio Code. Sobre o Visual Studio a versão Community 2017 é igual à versão Professional mas apenas pode ser utilizado por estudantes, projetos de código aberto e desenvolvedores individuais, o Code pode ser utilizado por todos incluindo qualquer projeto empresarial. Neste artigo utilizei o Microsoft Visual Studio 2017. Se utilizar o visual Studio Code recomendo que instale as seguintes duas extensões Azure Extensions Pack e Docker.



Criação do projeto C# .Net Core 1.1

No Visual Studio é criado um novo projeto ASP.Net Core 1.1 Web Application MVC este projeto já inclui código para efeitos de demonstração pode manter-se esse código ou fazer alterações ao mesmo. O Visual Studio fornece a hipótese de adicionar suporte ao Docker para o sistema operativo em que vai ser executado.





No projeto do Visual Studio criado vai ser adicionado código para ler o RSS do feed dos artigos da página de Internet da revista PROGRAMAR. Primeiro é necessário criar o modelo de dados que vai ser utilizado para listar a informação como se pretende. Por defeito um feed é composto por um título, o endereço de Internet, descrição e a data de publicação. Cria-se uma classe com o nome RSSFeedModel.cs ou outro se preferir no Models que vai contém o seguinte conteúdo.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace revista_programar.Model
{
    public class RSSFeedModel
    {
        public string ArticleTitle { get; set; }
        public string ArticleLink { get; set; }
        public string ArticleDescription { get;
                                           set; }
        public string ArticleDate { get; set; }
    }
}
```

Com o modelo criado é necessário criar o controler, o projeto já tem uma View inicial com à identificação HomeController.cs pode utilizar-se e adicionar o código que vai ler o XML com os artigos e armazenar uma lista do modelo criado.

```
public List<RSSFeedModel> GetFeed()
{
    return Feed().Result;
}

[HttpPost]
public async Task<List<RSSFeedModel>> Feed()
{
    var articles = new List<RSSFeedModel>();
    string rssFeedURL = "https://www.revista-
programar.info/feed/";
    HttpClient httpClient = new HttpClient();
    var httpContent = await httpCli-
ent.GetStringAsync(rssFeedURL);
    XDocument xml = XDocument.Parse(httpContent);
    var RSSFeedData = (from x in xml.Descendants
("item")
    select new RSSFeedModel
    {
        ArticleTitle = ((string)x.Element
("title")),
```

```
public List<RSSFeedModel> GetFeed()
{
    return Feed().Result;
}

[HttpPost]
public async Task<List<RSSFeedModel>> Feed()
{
    var articles = new List<RSSFeedModel>();
    string rssFeedURL = "https://www.revista-
programar.info/feed/";
    HttpClient httpClient = new HttpClient();
    var httpContent = await httpCli-
ent.GetStringAsync(rssFeedURL);
    XDocument xml = XDocument.Parse(httpContent);
    var RSSFeedData = (from x in xml.Descendants
("item")
    select new RSSFeedModel
    {
       ArticleTitle = ((string)x.Element("title")),
       ArticleLink = ((string)x.Element("link")),
       ArticleDescription = ((string)x.Element
("description")),
       ArticleDate = ((string)x.Element("pubDate"))
     });
     articles = RSSFeedData.ToList();
   return articles;
}
```

Em IActionResult Index() adiciona-se o seguinte.

```
public IActionResult Index()
{
   ViewBag.RSSFeed = GetFeed();
   return View();
}
```

Na View principal com a identificação Index.cshtml que é a página inicial pode personalizar à sua maneira para ler o RSS chama-se o ViewBag.RSSFeed do que contém a lista do Controller e transforma-se a lista numa tabela.

```
<div class="row">
    <table class="table table-hover">
        <thead>
            <tr>
                <th>Título</th>
                <th>Link</th>
                <th>Descrição</th>
                <th>Data</th>
            </tr>
        </thead>
        <tbody>
            @if (ViewBag.RSSFeed != null)
            {
                foreach (var item in
                                ViewBag.RSSFeed)
                {
                    <tr>
                        <td>@item.ArticleTitle</td>
                        <td><a href="@item
          .ArticleLink">@item.ArticleLink</a></td>

                      <td>@item.ArticleDescription</td>
                        <td>@item.ArticleDate</td>
                    </tr>
                }
            }
            else
            {
                <tr>
                    <td>Sem informação</td>
                    <td>Sem informação</td>
                    <td>Sem informação</td>
                    <td>Sem informação</td>
                </tr>
            }
        </tbody>
    </table>
</div>
```

E o projeto está concluído. Mas antes de criar o Docker Container para ser enviado para o Docker HUB eu recomendo testar primeiro na própria máquina. Para testar apenas é necessário executar a compilação do projeto e ter o Docker em execução.

Erros de compilação e/ou execução

Se ao executar e der o seguinte erro

```
Docker-compose -f "[Localização do projeto]
\Docker-compose.yml" -f "[Localização do pro-
jeto]\Docker-compose.override.yml" -f
"[Localização do projeto]\obj\Docker\Docker-
compose.vs.debug.g.yml" -p Dockercompo-
se3404700638243216757 config

.IOError: [Errno 2] No such file or directory:
u'[Localização do projeto]\obj\Docker\Docker-
compose.vs.debug.g.yml'
```

Isto deve-se os seguintes motivos não tem o Docker instalado, não tem o Docker em execução ou não permitiu ao Docker aceder à unidade de disco onde se encontra o projeto. Neste último problema para permitir ao Docker o acesso ao disco tem de iniciar o Docker com privilégios administrativos e nas definições em Shared Drives tem que selecionar o disco e aplicar. O Docker vai solicitar as credenciais da conta de administrador.

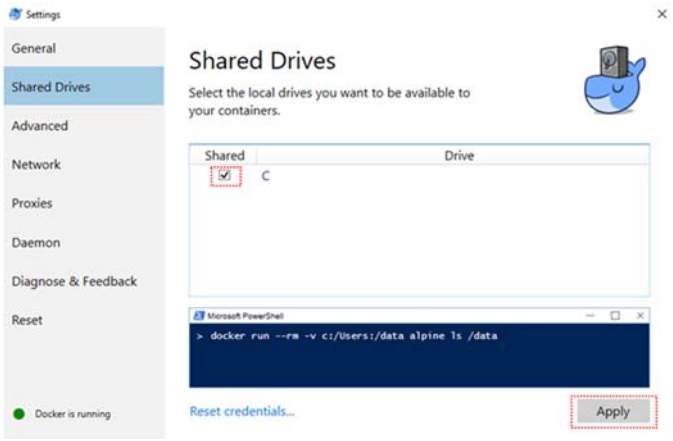


Figura 4 Docker em execução

Com o Shared Drives definidos os utilizadores sem privilégios administrativos que pretendam utilizar o Docker apenas têm que iniciar o Docker selecionar a unidade e aplicar. Não vai ser solicitada nenhuma credencial.

Se der o seguinte erro:

```
UnicodeDecodeError 'ascii' codec can't decote byte [byte] in
position [posição] not in range
```

Isto deve-se ao facto de em todo o caminho do projeto estarmos a usar caracteres especiais como o 'ç'.

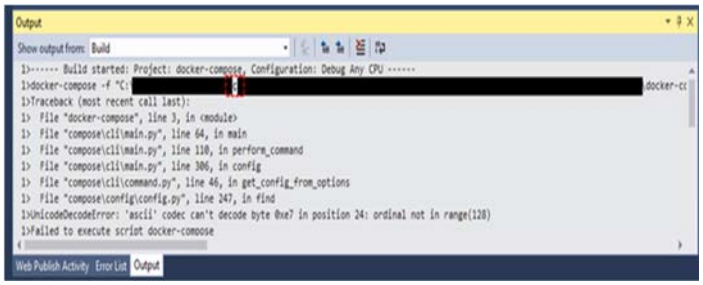


Figura 5 Visual Studio erro de compilação Unicode Decode Error

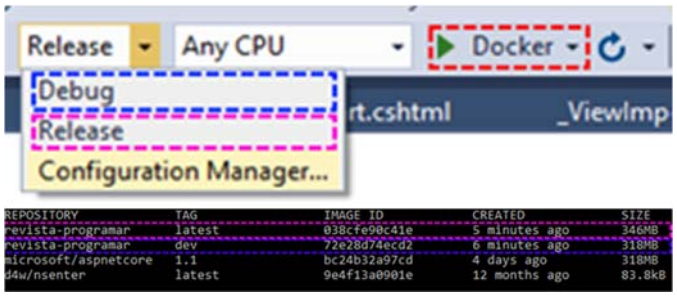
Se encontrar mais erros e não está a conseguir resolver pode consultar a comunidade de desenvolvedores do Visual Studio em https://developercommunity.visualstudio.com.

Com a execução concluída com sucesso o navegador de Internet predefinido vai abrir o projeto MVC criado conforme a seguinte imagem.

Figura 6 Projeto em execução Docker

Em Docker existem imagens Docker e Docker Containers. A imagem Docker é como um executável que inclui tudo o que é necessário para ser executado e é compatível com os vários sistemas operativos. A vantagem das imagens é que são executados isoladamente assim previne conflitos de software e outros problemas. O Docker Container é um recipiente que se encontra na 7ª camada do modelo OSI (Aplicação) que é a camada que está mais próxima do utilizador. Multiplos Containers podem ser executados na mesma máquina e os mesmos partilham o Kernel do sistema operativo, mas cada um deles são executados isoladamente. Um Docker Container não é a mesma coisa que uma Máquina Virtual enquanto nas máquinas virtuais não é possível partilhar recursos com outras máquinas virtuais os mesmos têm de ser reservados, o Docker Container permite partilhar os recursos. Por exemplo em cada máquina virtual é necessário instalar o Sistema Operativo e definir o armazenamento e memória Ram e outros recursos. O Docker utiliza o próprio sistema operativo instalado na máquina e os Container utilizam os recursos disponíveis. Pode utilizar um Docker Container em máquinas locais, virtuais ou em nuvem.

As operações no Docker são executas através de linhas de comandos. O Visual Studio cria automaticamente a imagem Docker e Docker Container. Para criar apenas tem de executar a compilação se fizer em Debug vai criar a imagem de desenvolvedor (dev) se fizer em Release vai criar a imagem final (latest).



Para consultar as imagens Docker na máquina executa-se o comando 'Docker images' se pretende remover uma imagem executa-se o comando 'Docker rmi [identificação da imagem].

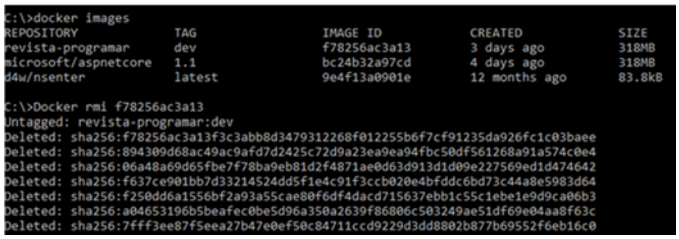


Figura 7 Apagar uma imagem Docker

Para criar um Container a partir de uma imagem executa-se o comando 'Docker run' mas é necessário definir qual a imagem, o porto exterior e interior que se pretende utilizar. Para definir a imagem utiliza-se o parâmetro '- -name [nome da imagem], o porto de entrada e saída é o parâmetro '-d -p [porto exterior]:[porto interior]. Existem mais parâmetros como o utilizador, palava-passe, aceitação dos termos de licenciamento (EULA) e outros pode consultar os comandos a partir da ajuda com o comando 'Docker run - -help'. Para criar o Container com o nome revista-programar em que vai receber pedidos no porto externo 5632 que vão ser reencaminhados para o porto interno 80 e define-se a imagem pela sua identificação o comando é 'Docker run - -name revista-programar -d -p 5632:80 [identificação da imagem]'.

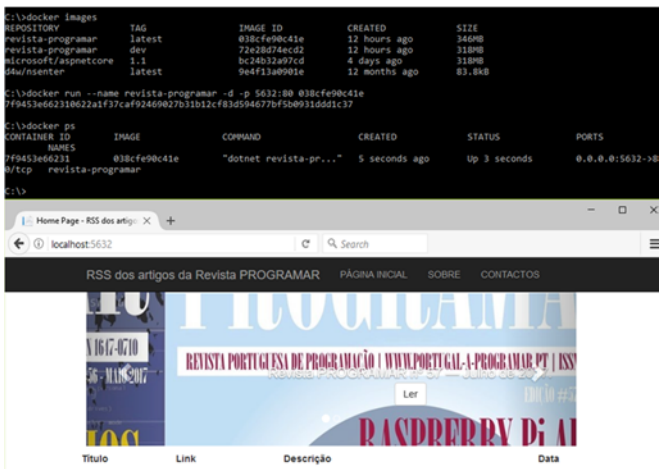


Figura 8 Criação de um Docker Container a partir de uma imagem Docker

Quando cria a compilação no Visual Studio o mesmo cria automaticamente a imagem e a seguir o Container.

Para consultar os Docker Container em execução executa-se o comando 'Docker ps'. Se pretender remover um Container utiliza-se o comando 'Docker rm [identificação do Container]' mas se a mesma estiver em execução não é possível remover. Para parar utiliza-se o comando 'Docker stop [identificação do Container]. Para forçar a remoção de Docker Container utiliza-se o parâmetro '-f'.

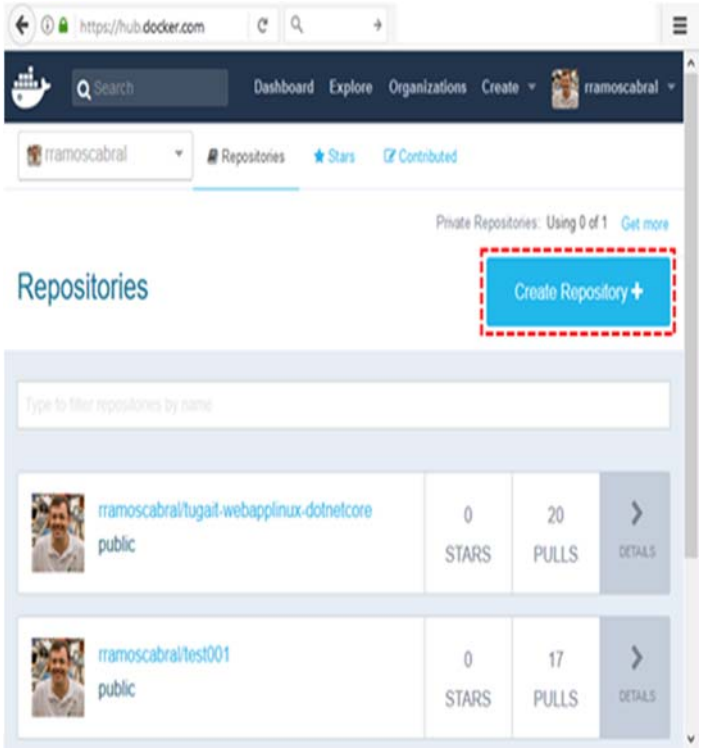
```
C:\>docker ps
CONTAINER ID        IMAGE                  COMMAND             CREATED             STATUS              PORTS
                    NAMES
9d26be422c72        revista-programar:dev  "tail -f /dev/null"  3 days ago          Up About an hour    0.0.0.0:32768-
>80/tcp   dockercompose3404700638243216757_revista-programar_1

C:\>docker rm 9d26be422c72
Error response from daemon: You cannot remove a running container 9d26be422c728320c6d56894b6b6fd73e2183c9fe69f5c6a4b1775
```

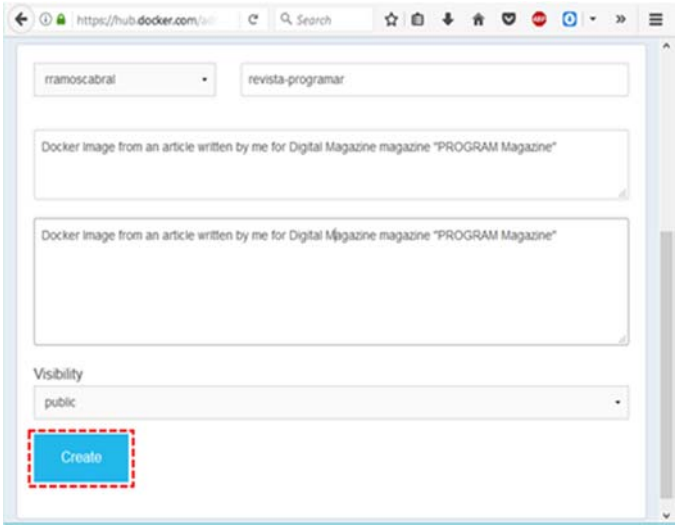
Figura 9 Não é possível apagar Docker Container em execução

Docker Hub

O Docker Hub (https://hub.docker.com/) é um repositório em nuvem onde pode ter as suas imagens Docker. O repositório pode ser público ou privado. Este serviço permite ter um repositório privado gratuitamente e repositórios públicos gratuitos. Com este serviço pode distribuir as suas imagens a partir de um único lugar assim quando atualiza o repositório pode atualizar automaticamente os clientes se os mesmos aceitarem atualizações automáticas.

Para criar um repositório Docker Hub tem de ter uma conta no Docker Hub, selecionar "Create Repositor", escolher namespace, definir o nome do repositório, adicionar uma descrição curta e longa definir se pretende que o repositório seja público ou privado e para finalizar criar em Create.





Com o repositório criado pode enviar a imagem do projeto para o Docker Hub com o comando 'docker push'. Primeiro tem que identificar-se no Docker Hub com o comando 'Docker login', a seguir tem que definir o tag 'docker tag [identificação da imagem existente na máquina] [nome do namespace]/[nome do repositório], para enviar utilize o comando 'docker push [nome do namespace]/[nome do repositório]'.

```
C:\>docker login
Login with your Docker ID to push and pull images from Docker Hub. If you don't have a Docker ID, head over to https://h
ub.docker.com to create one.
Username: rramoscabral
Password:
Login Succeeded

C:\>docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
revista-programar   latest              038cfe90c41e        27 hours ago        346MB
revista-programar   dev                 72e28d74ecd2        27 hours ago        318MB
microsoft/aspnetcore   1.1              bc24b32a97cd        5 days ago          318MB
d4w/nsenter         latest              9e4f13a0901e        12 months ago       83.8kB

C:\>docker tag 038cfe90c41e rramoscabral/revista-programar

C:\>docker push rramoscabral/revista-programar
The push refers to a repository [docker.io/rramoscabral/revista-programar]
3d9c2550461e: Pushing [==========>                                        ]  9.17MB/28.05MB
10783d189319: Pushing  1.536kB
488dc49d7d63c: Mounted from microsoft/aspnetcore
9e65f5989552c: Mounted from microsoft/aspnetcore
7c18504757906: Mounted from microsoft/aspnetcore
ffba37508dbc: Waiting
18f9b4e2e1bc: Preparing
```

Figura 10Envio da imagem para Docker Hub

Para sair do Docker utilize o comando 'docker logout'. Se algum dia pretender definir um repositório publico para privado ou apagar o repositório estas operações são executadas nas definições do próprio repositório.

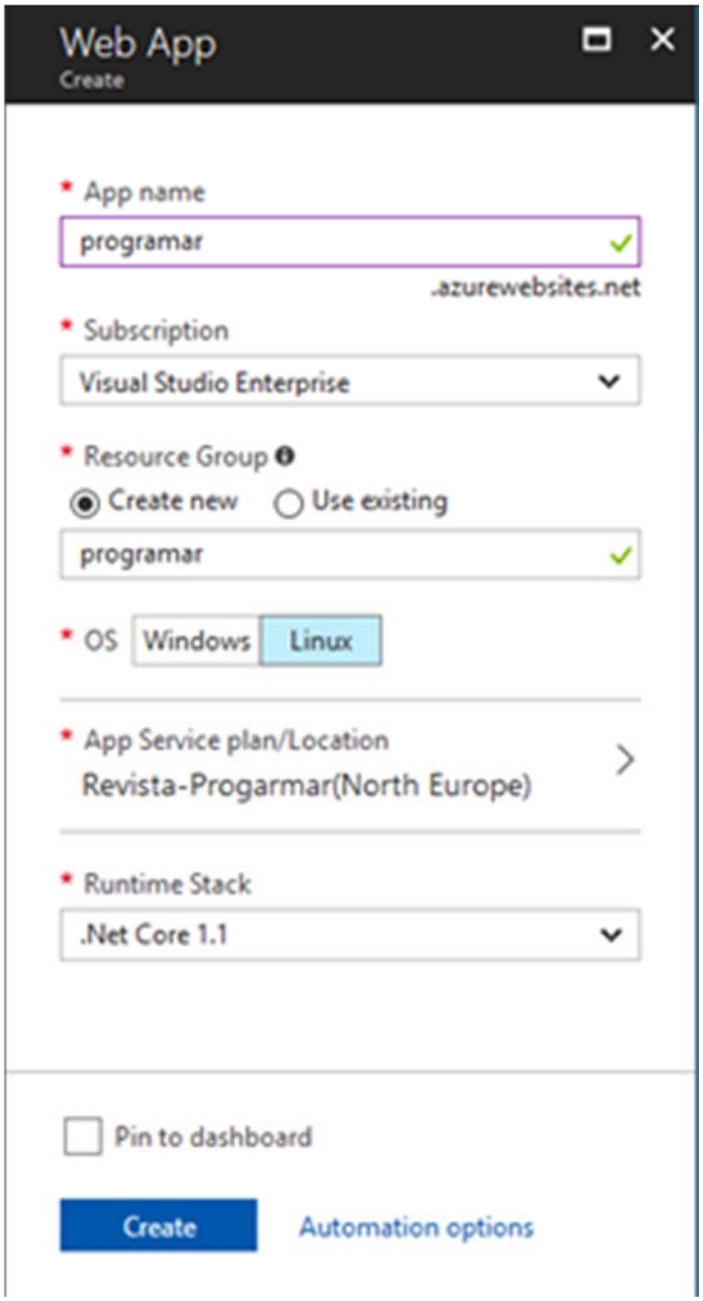
Se pretender descarregar a imagem execute o comando 'Docker pull [nome do namespace]/[nome do repositório]. Pode pesquisar imagens com o comando 'Docker search [nome da imagem]

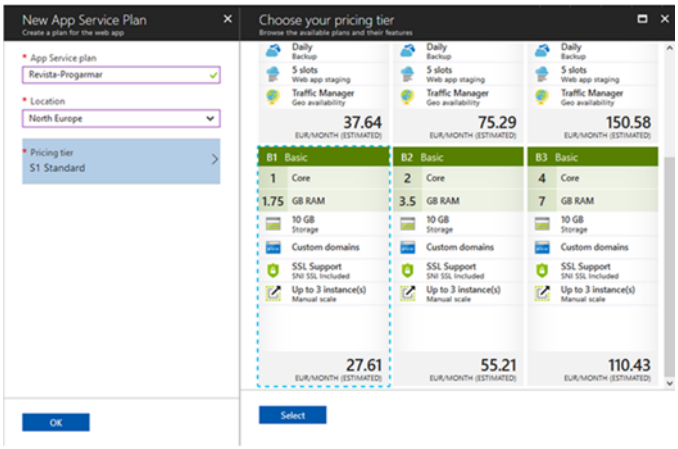
Azure App em Linux

Para consegui utilizar os serviços no Azure é necessário ter uma subscrição, no entanto se nunca utilizou o Azure pode testar o serviço durante 30 dias em https://azure.microsoft.com/pt-pt/free ou experimentar durante 30 minutos sem custo e explicarei adiante com pode faze-lo.

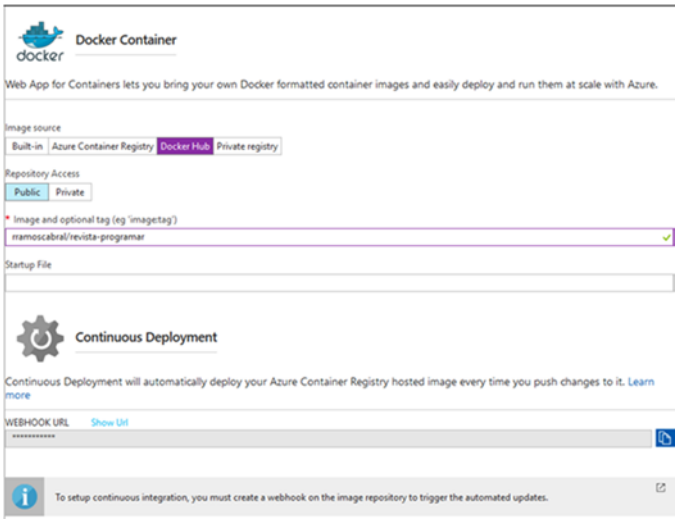
Normalmente trabalho com sistemas com o idioma em Inglês todas as seguintes imagens estão em Inglês. Para criar uma nova Web App em Linux selecione Novo/New e depois selecione Web + Mobile e escolher Web App. Tem de

definir o subdomínio que é único no universo do Azure em azurewebsites.net, se tiver mais do que uma subscrição escolha a subscrição que pretende utilizar, pode escolher um grupo de recursos existente ou criar um novo. Um grupo de recursos é um recipiente de recursos e/ou serviços de Azure quando quiser apagar todos os recursos e/ou serviços no recipiente em vez de apagar um a um pode apagar o grupo de recursos e todos eles serão eliminados. No sistema operativo escolha Linux e tem de escolher ou criar um novo plano de serviço que é o plano de alojamento do serviço. Em Web App em Linux não existe nenhum plano gratuito ao contrário do Web App em Windows que dispõe do plano gratuito o Free Tier. O plano mais baixo em Linux é o Basic B1 mas não tem acesso a todas as funcionalidades como por exemplo slots.

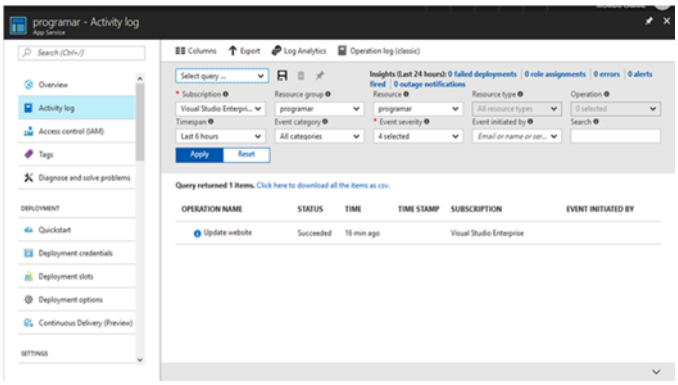




A Web App demora alguns minutos a ser criada e quando finalizar está pronto a ser utilizada. Depois de criar selecione Docker Container em definições para definir o repositório da imagem Docker. Selecione Docker Hub e tem que definir o tipo de acesso do seu repositório e a imagem e tag se utilizar [identificação do namespace]/[identificação do repositório]:[Tag]. Para uma atualização continua pode usar o Web Hock.



Após a definição tem de aguardar que o Azure atualize o Web App, pode demorar uns minutos, no entanto pode verificar o relatório de atividade.



Depois de atualizado com sucesso se aceder ao URI do Web App pode verificar que a mesma já esta em funcio-

namento com a imagem do repositório definido.

Com a aplicação a correr pode experimentar aceder ao KUDU, terá apenas que adicionar o subdomínio scm depois do subdomínio definido [subdomínio definido].scm.azurewebsites.net.

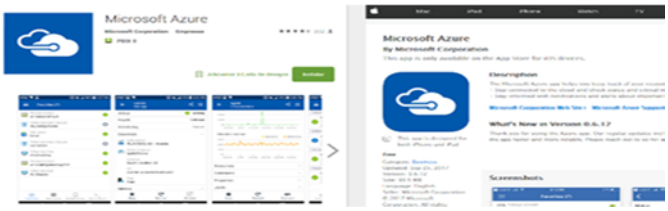
Figura 11 KUDU

Com o KUDU em Linux consegue ver informação do sistema, variáveis de ambiente e parâmetros, executar comandos numa consola Bash. Apesar de permitir o acesso SSH normalmente este encontra-se bloqueado devido à segurança porque consegue fazer qualquer alteração na máquina até apagar definições que o próprio Azure define o que pode danificar a máquina virtual.

Está concluído o deployment da aplicação em Docker para Web App em Linux.

Azure Web App Roadmap

As funcionalidades dos serviços no Azure estão sempre em contante atualização e o Azure Web App em Linux não é exceção. No GitGHUB pode consultar as últimas notas de lançamento em https://aka.ms/linux-release-notes. Na página do Azure consegue consultar outras novidades em https://azure.microsoft.com/en-us/updates/?product=web-sites. Por exemplo desde de maio de 2017 está disponível ao público em geral uma aplicação para Android e Iphone que permite consultar estados, métricas, notificações e alertas dos serviços que estão a ser utilizado na sua subscrição. Esta aplicação ainda se encontra em desenvolvimento.



Quando é devemos escolher o Web App em Windows ou Linux?

Depende dos requisitos da aplicação, se a carga de trabalho funciona muito bem em Linux então deve escolher Linux. Se a aplicação foi desenvolvida em Java então deve escolher Windows por enquanto ainda não é possível usar Java em Linux. Se pretende testar a aplicação em desenvolvimento sem custos apenas o Windows tem o plano de alojamento Free Tier.

É preciso analisar muito bem a aplicação, necessidades e funcionalidades. No entanto pode testar a aplicação durante 30 minutos gratuitos.

Como testar Azure Web App Linux gratuitamente

Para finalizar se pretende testar o Azure Web App Linux gratuitamente durante 30 minutos sem nenhum custo a oferta está disponível em https://aka.ms/tryappservice, apenas tem de escolher Web App e a linguagem de programação. Também pode experimentar o Web App para Containers apenas tem que providenciar a imagem Docker por exemplo 'rramoscabral/revista-programar' e utilizar até a duração expirar. Ambos vão solicitar uma conta Microsoft existente. Recomendo que não abuse desta opção porque serão contactados pela equipa de vendas do Azure.

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.

ELECTRÓNICA

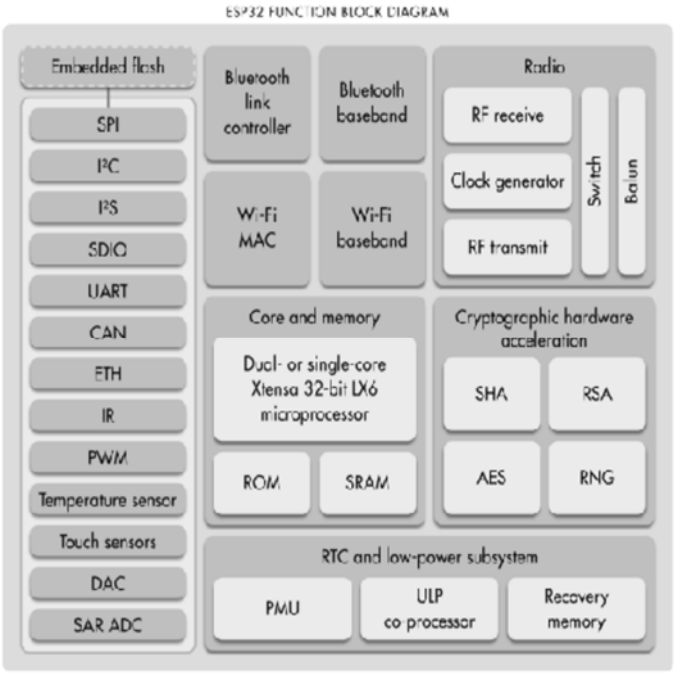
ESP32 - MICROPYTHON

ESP32 - MICROPYTHON

Para aqueles que andam no mundo da Internet das Coisas o micro controlador ESP8266 já deve ser um conhecido, para quem nunca ouviu falar aconselho a experimentar, pois desenvolver dispositivos que tenham que ser ligados a uma rede Wi-Fi nunca foi tão simples.

Então se o ESP8266 é assim tão bom porque é que eu estou aqui a falar do ESP32… bem este super micro controlador… sim super porque conta 2 cores de 240MHz dentro de um microprocessador Tensilica Xtensa de 32 bit Ultra Low Power, 448KiB de ROM para funções internas e de arranque, 520KiB de SRAM para colocarmos o nosso código e por fim a cereja no topo do bolo é o facto ser constituído por 2 módulos de conectividade um Bluetooth Low Energy e Wi-Fi 802.11/b/g/n/e/i, comparando com o ESP8266 tem o módulo BLE a mais, tem muito mais memória e muito mais processamento,e não esquecendo que os 2 cores permitem agora executar código em paralelo, coisa que raramente acontece nos micro controladores comuns de baixo custo, sim este ESP32 custa por volta de 6 euros e pode ser adquirido em http://aliexpress.com.

Na imagem abaixo podemos visualizar a arquitectura do ESP32.



Agora que já conhecemos a "Máquina", voltamo-nos para o tema desta edição em que queremos correr o Sistema Operativo MicroPython dentro do nosso ESP32.

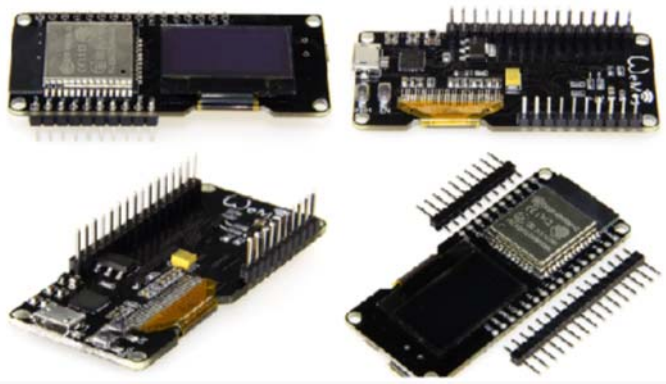
Para aqueles que ainda estão de boca aberta por ter conhecido o ESP32 então não fiquem céticos, porque é mesmo possível correr on sistema operativo e ainda executar código Python directamente numa prompt Python dentro do ESP32.

O que vou aprender neste artigo

- A Instalar Ferramentas necessários para carregar o Sistema operativo para o ESP32
- • A Instalar drivers para o computador comunicar com o ESP32 através do controlador USB-Serial
- CP21XX que acompanha a placa de desenvolvimento que vamos utilizar.
- • Carregar o binário com o Micropython para dentro do ESP32
- • Aceder à consola Python que está a correr dentro do micro controlador
- • Carregar bibliotecas para serem utilizadas dentro do Sistema Operativo
- • Escrever texto para ser mostrado no display OLED SSD1306 que vem incluído na placa.

O que preciso para este artigo

- Placa de desenvolvimento Lolin Wemos ESP32 com display OLED
- Cabo USB
- Computador



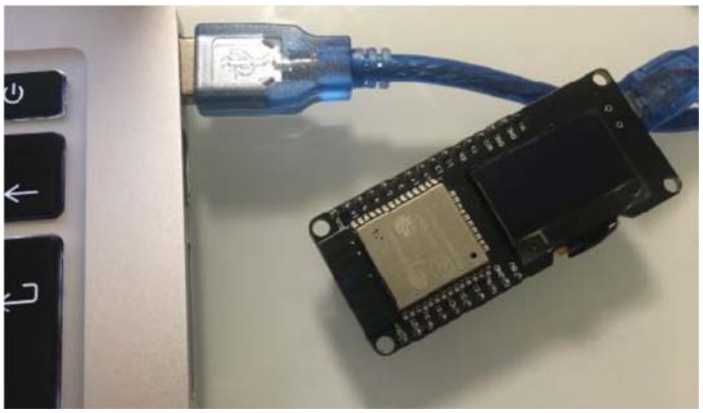
Existem dezenas de versões de placas de desenvolvimento que incluem o ESP32, no nosso tutorial vamos utilizar uma desenvolvida pela LOLIN/WEMOS que inclui um display OLED SSD1306.

Existem diversas formas de instalar o MicroPython, está é uma das formas mais rápidas de carregar o Sistema Operativo e interagir com display OLED.

Eu vou estar a utilizar Python 2.7 e um Macbook Pro com o High Sierra, sobre outros sistemas operativos vou deixar alguns pontos para terem como pesquisar.

Passo 1: Ligar a placa ao computador e determinar o número da porta

Neste passo vamos ligar a placa ao nosso computador utilizando um cabo micro USB.



Aqui um utilizador menos experiente pode ter algumas dificuldades em perceber se tudo está instalado correctamente, mas não desespere porque vou deixar aqui algumas dicas para verificar se está tudo instalado.

Como qualquer dispositivo que ligamos ao nosso computador, este precisa de software adicional para conseguir comunicar com o nosso sistema operativo, seja ele Windows, Mac OS, Linux ou outro. Este tipo de software denomina-se de Driver, ao longo dos tempos os fabricantes tem consolidado com os desenvolveres de sistemas operativos uma forma de incluir os controladores directamente no sistema operativo ou pelo menos o sistema operativo saber onde ir descarregar o driver à internet, sem que utilizador se preocupe com isso, denominamos esses dispositivos de Plug and Play. Bem não é o caso do nosso pequeno ESP32, aqui temos que ir descarregar o driver directamente do site do fabricante, coisa que aconselho sempre ao contrário de drivers que podemos encontrar noutros locais, o do fabricante será decerto o menos propicio a dar problemas. (Utilizadores Linux não deverão necessitar de drivers)

Bom o nosso instinto passa por ir directamente ao google pesquisar “driver wemos lolin esp32…”, esta não é de todo a melhor forma, devemos perceber que a placa de desenvolvimento é constituída por outros controladores que não o nosso ESP32, e se olharmos com atenção vamos ver outros componentes, e perto da porta USB temos um que diz Sililabs CP2102, este é o controlador responsável pela comunicação USB-Serial com o nosso ESP32, isto porque os micro controladores na sua maioria só entendem comunicação serial.

Dica: Aconselho o leitor pesquisar sempre um pouco sobre alguns componentes que se destacam numa placa com o fim de perceber a sua funcionalidade.

Para instalar o driver vamos ao site da Silicon Labs e basta descarregar o driver indicado para o nosso sistema operativo.

Link: https://www.silabs.com/products/development-tools/software/usb-to-uart-bridge-vcp-drivers

Depois de instalado podemos verificar a instalação, numa máquina Linux ou MacOS podemos executar o comando ls /dev/{tty,cu}.* e no Windows podemos abrir o Gestor de Dispositivos e procurar por dispositivos com o nome COMX, ex. COM3.

Agora que já temos uma porta associada ao nosso ESP32 podemos preparar o software necessário para interagir com ele e atualizar o firmware.

Passo 2: Preparação do ambiente de desenvolvimento

Python

Se ainda não tem o Python instalado no seu computador, é necessário instalar, eu pessoalmente utilizo o Python 2.7 mas se quiser instalar o 3.x penso que não deve existir problema.

Para a instalação do Python basta ir ao site oficial https://www.python.org e fazer o download correspondente ao seu sistema operativo e seguir as instruções lá indicadas.

Pip

O Pip é utilizado para instalar novos módulos no Python, nas instalações mais recentes do Python o mesmo já vem incluído. No entanto, caso não esteja instalado pode seguir este tutorial https://pip.pypa.io/en/stable/installing/ .

Dica: o comando pip está relacionado com a versão do Python por isso numa máquina com diferentes versões do Python para utilizar a versão Python 2.7 o comando será pip2.7. Para utilizar apenas o comando pip deve garantir a versão por defeito do Python, pode confirmar a versão executando pip --version

Esptool

Esta ferramenta é recomendada para instalar o novo firmware no ESP32 ou em qual outro chip da familia ESP.

Para fazer a instalação do mesmo vamos utilizar o pip

```
# pip install esptool
```

Se já tiver uma versão antiga pode atualizar o mesmo com o comando:

```
# pip install esptool --upgrade
```

Ampy

O Ampy permite transferir ficheiros do nosso sistema local de ficheiros para dentro do ESP32. Eu utilizo a versão da Adafruit e pode ser facilmente instalado utilizando o pip.

```
# pip install adafruit-ampy
```

Terminal de acesso

Para nos ligarmos à consola do ESP32, que nos vai dar acesso à uma prompt Python vamos necessitar de um terminal. Para utilizadores MacOS ou Linux não é necessário porque já possuem comandos para se ligar a um terminal

serial, nomeadamente o comando **screen**, utilizadores Windows podem fazer o download do PuTTY a partir do site official www.putty.org .

Passo 3: Carregar o binário do MicroPython para o ESP32

O binário com o MicroPython para o ESP32 pode ser descarregado no link https://micropython.org/download#esp32 e guardado numa localização conhecida.

Vamos agora abrir a linha de comandos, caso seja Windows o utilizador deve ir para a pasta onde está o Python instalado, ex: C:\Python27\Scripts. No Linux ou MacOS o python, pip, esptool e ampy são agora um comandos de sistema e podem ser chamados independentemente da directoria.

Teste de ligação

Podemos utilizar o comando esptool.py para verificar a ligação com o ESP32, a forma mais simples de o fazer é pedindo informações relacionadas com a identificação da flash.

Portas em MacOS, por norma a porta tem o nome /dev/cu.SLAB_USBtoUART, caso não aconteça utilize o comando ls /dev/{tty,cu}.* para listar todas as portas

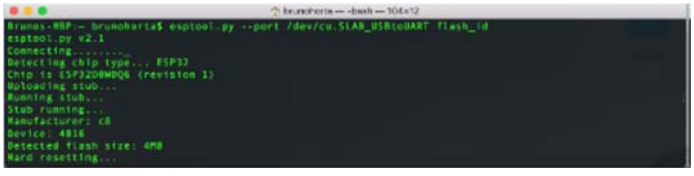
```
# esptool.py --port /dev/cu.SLAB_USBtoUART flash_id
```

Comando em Linux ou Windows

```
# esptool.py --port nome_da_porta flash_id
```

Para sabermos o nome da porta em Windows vamos ao Gestor de Dispositivos e procuramos por COM_X ex: COM3, numa maquina linux podemos executar o comando ls /dev/{tty,cu}.* que nós lista as portas associadas ao dispositivos conectados atualmente ao computador.

Um resultado bem sucedido deve devolver um resultado semelhante ao abaixo demonstrado:

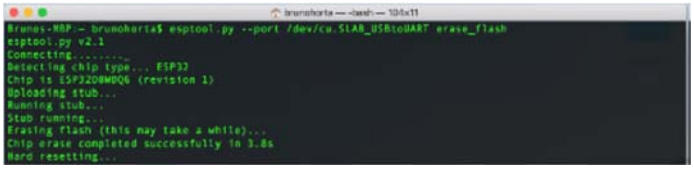


Apagar a Flash

Antes de carregarmos o novo binário devemos garantir que não fica lá nada que nos possa consumir espaço desnecessário, para isso utilizamos o comando

```
# esptool.py --port nome_da_porta erase_flash
```

O resultado do comando deve ser algo idêntico ao abaixo demonstrado:



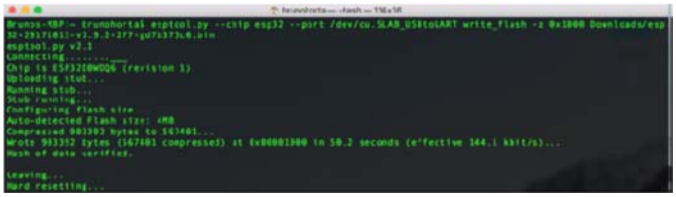
Carregar no binário

Finalmente estamos no passo final, para gravarmos o binário que contêm o MicroPython utilizamos o comando abaixo descrito, alterando o nome da porta para a que foi atribuída ao ESP32, e a localização do ficheiro apontando para o local onde o utilizador guardou o download do ficheiro .bin.

```
# esptool.py --chip esp32 --port nome_da_porta
write_flash -z 0x1000 localização_do_ficheiro.bin
```

Este processo pode e vai demorar alguns segundos por isso não cancele o processo nem deve remover o ESP32 abruptamente.

O resultado do comando deve ser algo idêntico ao abaixo demonstrado:



Verificação e utilização da instalação

Numa maquina Linux ou MacOS para nos ligarmos ao terminal Serial do ESP32 podemos utilizar o comando:

```
# screen nome_da_porta 115200
```

Ao abrir a ligação carregamos no enter e a magia acontece ficamos na prompt interativa do Python com os famosos >>>.

A partir daqui podemos escrever código que ele é executado, por exemplo:

Parte BRUTALLL :)

Bem, o objectivo deste tutorial finaliza com o acesso ao prompt do Python, no entanto já que estamos com a inspiração no pico máximo e completamente delirantes com o ESP32, este que até trás um display OLED mesmo a pedir para receber um belo Hello World, fica aqui a explicação de como podemos carregar Bibliotecas extra para serem utilizadas no nosso código e como enviar texto para o display.

Carregamento da biblioteca para o display OLED SSD1306

Eu utilizo a biblioteca da Adafruit ssd1306.py, pode ser facilmente descarregada em https://github.com/adafruit/

micropython-adafruit-ssd1306, descompacte o zip e guarde o ficheiro ssd1306.py numa localização conhecida.

Vamos então carregar o biblioteca para dentro do nosso ESP32 utilizando o comando.

```
# ampy --port nome_da_porta --baud 115200 put loca-
lização_do_ficheiro_ssd1306.py
```

Importante: para enviar dados para o ESP32 devem fechar todas as ligações abertas para o mesmo.

Dica: se estiver a utilizar uma maquina Windows deve estar no diretório Scripts dentro da pasta da instalação do Python, em Linux ou MacOS o comando pode ser invocado em qualquer directoria.

O resultado do comando deve ser algo idêntico ao abaixo demonstrado:

Terminado o envio voltamos a ligar-nos ao nosso ESP32 utilizando o comando screen em maquinas Linux ou MacOS ou o PuTTY em maquinas windows.

Acesso à prompt Python vamos fazer algum código que nos vais listar os ficheiros gravados na memória do ESP32.

O resultado do comando deve ser algo idêntico ao abaixo demonstrado:

Confirmada a existência da biblioteca ssd1306 vamos ao que interessa e fazer código para enviar texto para o nosso lcd OLED.

Primeiro importamos as bibliotecas machine que nos dá acesso ao GPIOS e aos protocolos de Hardware do ESP32 e a ssd1306 que nos oferece um conjunto de funções de alto nível para interagirmos com o display OLED

```
>>> import machine, ssd1306
```

O passo seguinte é ativar o protocolo I2C para comunicar com o display, pela documentação percebemos que este está ligado ao pino 4 SCL e ao pino 5 SDA

Agora que já temos o I2C vamos criar uma variável

```
>>> i2c = machine.I2C(scl=machine.Pin(4),
sda=machine.Pin(5))
```

que vai referenciar o nosso display, a biblioteca ssd1306 já tem uma função que nos devolve uma instância/referência para o OLED.

```
>>> oled = ssd1306.SSD1306_I2C(128, 64, i2c)
```

Quase a terminar limpamos todo o texto que possa

```
>>> oled.fill(0)
```

estar em preparação.

```
>>> oled.text('MicroPython para', 0, 0)
>>> oled.text('a revista', 0, 10)
```

Começamos a preparar o texto que queremos

Finalmente mandamos mostrar esse texto fazendo com que a magia aconteça, a partir daqui é a loucura .

```
>>> oled.show()
```

"(...)este super micro controlador... sim super porque conta 2 cores de 240MHz dentro de um microprocessador Tensilica Xtensa de 32 bit Ultra Low Power, 448KiB de ROM para funções internas e de arranque(...)"

O resultado deve ser algo idêntico ao demonstrado na imagem a abaixo:

Espero que este artigo vos tenha sido útil e qualquer duvida estão perfeitamente à vontade para perguntar.

Um Grande Abraço e boa PROGRAMAÇÃO
Se gostas deste tipo de informação junta-te a nos no Movimento Maker Portugal, no Facebook

AUTOR

Escrito por Bruno Horta

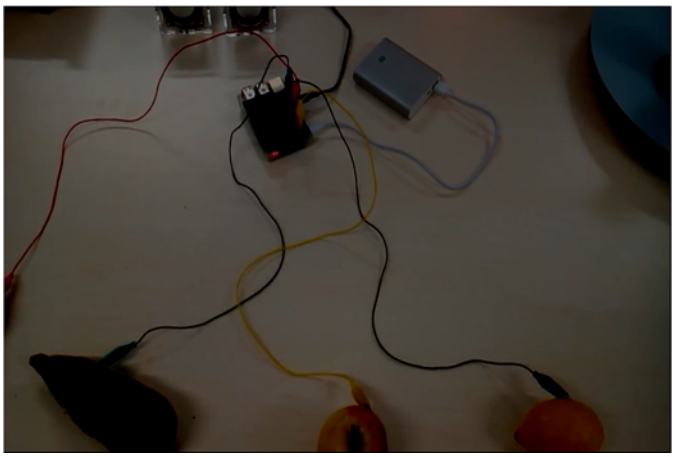
Licenciado em Engenharia Informática pela ESTG. É Senior Java Developer, sendo o seu principal foco o desenvolvimento de aplicações distribuídas e de alto desempenho. Adora desenvolver protótipos para serem utilizados no contexto da Internet das Coisas. É membro fundador do Movimento Maker Portugal.
Linkedin: https://www.linkedin.com/in/brunohorta/

POMAR MUSICAL

O pomar musical foi um projeto feito, inicialmente, para o evento Eletrónica e Informática, organizado pela Associação de Informática de Castelo Branco.

Este evento visava mostrar projetos amadores e profissionais que se faziam, não só em Castelo Branco, mas em todo o país.

O pomar musical usa um Raspberry PI 2 modelo B e o Adafruit Capacitive Touch Shield MPR121 para o Raspberry PI e é programado usando a linguagem Python.



Adafruit MPR121

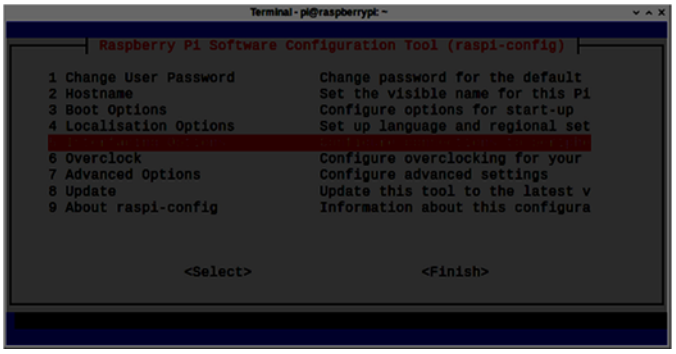
Como funciona ?

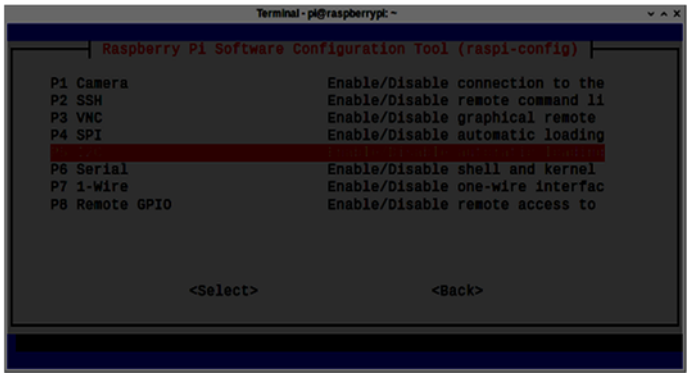
O chapéu capacitivo funciona detetando quando alguém toca num dos 12 elétrodos do sensor. Todos os dias usamos esta tecnologia, nos telefones, tablets e até mesmo eletrodomésticos.

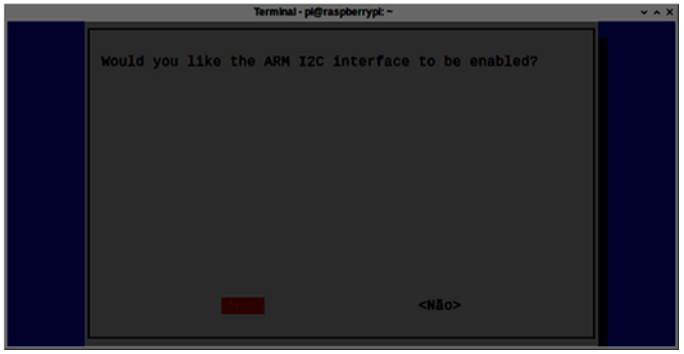
Instalação

Para poder usar o chapéu, é necessário ativar o I2C do Raspberry PI. Para isso, basta executar o raspi-config e ativar.

```
sudo raspi-config
```







De seguida, necessitamos de instalar o suporte para o chapéu. Para tal, basta instalar as bibliotecas da Adafruit disponiveis no github e algumas bibliotecas necessárias para a compilar.

Primeiro, devemos atualizar as definições de software do Raspberry PI

```
sudo apt-get update
```

De seguida, instalar o software necessário

```
sudo apt-get install build-essential python-dev
                     python-smbus python-pip git
```

Após tudo instalado, vamos buscar as bibliotecas para o MPR121. Como o código está alojado no Github, vamos usar o git para clonar o repositório.

```
git clone https://github.com/adafruit/
                        Adafruit_Python_MPR121.git
```

```
Clonar em 'Adafruit_Python_MPR121'...
remote: Counting objects: 73, done.
remote: Total 73 (delta 0), reused 0 (delta 0),
pack-reused 73
Unpacking objects: 100% (73/73), done.
Checking connectivity... terminado.
```

No final, podemos entrar na diretoria e visualizar o seu conteúdo:

```
cd Adafruit_Python_MPR121
ls
```

```
Adafruit_MPR121  examples  ez_setup.py  LICENSE
README.md  setup.py
```

Instalar a biblioteca

```
sudo python setup.py install
```

Se tudo correr bem, temos as bibliotecas instaladas.

Para experimentar, nada como ir à diretoria dos exemplos e executar o teste mais simples:

```
cd examples
sudo python simpletest.py
```

Caso tenham aqui o seguinte erro (ou algo parecido):

```
sudo python simpletest.py
```

```
Adafruit MPR121 Capacitive Touch Sensor Test
chmod: cannot access '/sys/module/i2c_bcm2708/
parameters/combined': No such file or directory
Traceback (most recent call last):
  File "simpletest.py", line 34, in <module>
    if not cap.begin():
  File "build/bdist.linux-armv7l/egg/
Adafruit_MPR121/MPR121.py", line 90, in begin
  File "build/bdist.linux-armv7l/egg/
Adafruit_GPIO/I2C.py", line 80, in requi-
re_repeated_start
```

É porque o módulo não está carregado. Nada mais simples que:

```
sudo modprobe i2c_bcm2708
```

Para o tornar permanente, editem o ficheiro /etc/modules e acrescentem o módulo sudo vi /etc/modules (podem user outro editor mais confortável para voçês) e acrescentem a linha:

```
i2c_bcm2708
```

Guardar o ficheiro e já está. Cada vez que for efetuado reboot, ele será carregado.

Voltem novamente a experimentar e já deverá funcionar

```
sudo python simpletest.py
```

```
Adafruit MPR121 Capacitive Touch Sensor Test
Press Ctrl-C to quit
```

Cada vez que tocarem num sensor, poderão ver a mensagem na consola:

```
9 touched!
9 released!
4 touched!
4 released!
3 touched!
3 released!
2 touched!
2 released!
1 touched!
1 released!
5 touched!
5 released!
6 touched!
6 released!
```

Significa que o sensor X foi tocado e depois libertado (quando tocam e retiram o dedo)

Pomar musical

O Pomar musical não é mais que vários clips de crocodilo ligados nos sensores numa das pontas e na outra ponta ligados a peças de fruta. O importante é que esteja ligado a qualquer material que seja condutivo – metal, água doce ou salgada (quanto mais salgada, mais condutiva) e também, neste caso, fruta. Cada vez que se toca numa peça de fruta, o sensor correspondente dá sinal que está a ser tocado e um ficheiro mp3 com um som é tocado.

Material necessário

Para o Pomar musical, é necessário:

- Um Raspberry PI
- O chapéu da Adafruit MPR121 para o Raspberry PI
- 12 clips de crocodilo
- 12 peças de fruta (ou menos – não esqueçer de comer depois)
- Colunas para ligar ao Raspberry PI
- Ligações

Uma nota importante é, como os fios e os elétrodos têm uma capacitância inerente, significa que, ao ligar um clip de crocodilo (ou trocar alguma fruta numa das pontas), o sensor vai pensar que está a ser tocado. O que é necessário fazer nestes casos é recalibrar os sensores, bastando para isso reiniciar o script de Python. A calibração é efetuada quando o chip é inicializado. O que poderá ser feito é, primeiro efetuar todas as ligações necessárias e só depois executar o script de Python.

POMAR MUSICAL

Sons

Neste caso, foram usados sons livremente disponíveis na Internet, como o som de uma baleia, uma campainha, etc… Os ficheiros de sons foram convertidos para wav para ser mais simples.

Código

O código do Pomar Musical é do mais simples que pode existir. Para tocar os sons, usamos a biblioteca de Python pygame. Esta biblioteca permite criar aplicações multimédia, usando Python, e é programada usando as bibliotecas SDL.

As partes mais importantes do código

Inicializar o mixer com alguns valores para evitar LAG desde que é dado o comando até que o som seja efetivamente tocado. Funcionam bem em Linux

```
# Init mixer
pygame.mixer.pre_init(44100, -16, 2, 2048)
                                # Avoid sound lag
pygame.init()                       # Initialize
```

Posteriormente, criamos um dicionário, com conteúdo de chave:valor, onde a chave será o índice (que irá corresponder ao número do sensor no MPR121) e o valor será o nome (caminho relativo desde onde é executado o script) do ficheiro a tocar. Os ficheiros de som estão colocados numa directoria imediatamente a seguir à localização do script, chamada de Sons (atenção à capitalização, uma vez que Linux é sensível a maiúsculas e minúsculas).

```
# mapear sons
MAPA_SONS = {
        0: 'Sons/1.mp3.wav',
        1: 'Sons/2.mp3.wav',
        2: 'Sons/3.mp3.wav',
        3: 'Sons/4.mp3.wav',
        4: 'Sons/5.mp3.wav',
        5: 'Sons/6.mp3.wav',
        6: 'Sons/7.mp3.wav',
        7: 'Sons/8.mp3.wav',
}
```

Agora, criamos uma lista, inicializada a 0's, que iremos usar para criar os objetos de som.

```
sons = [0,0,0,0,0,0,0,0]
```

Agora, vamos percorrer a lista "sons" - inicializada a 0 - onde vamos criar um objeto de som em cada um dos índices, e definir o volume.

```
import Adafruit_MPR121.MPR121 as MPR121

cap = MPR121.MPR121()
```

Caso não esteja disponivel, dará um erro.

```
if not cap.begin():
        print 'Error initializing MPR121!'
        sys.exit(1)
```

O ciclo principal é onde os sons são tocados. Cada input é lido e verificado se houve transições no estado – tocado ou não tocado.

A função touched é importantíssima aqui. Ela devolve um valor de 12 bits onde cada bit representa um dos 12 inputs do chapéu. Bit 0 (posição) é o input 0, bit 1 (posição) é o input 1 e assim sucessivamente até ao 11 (recordem-se, tudo começa no 0). Se o bit tiver o valor 1, então está a ser tocado naquele momento, e se tiver o valor a 0, então não está a ser tocado.

O ciclo irá comparar o estado anterior para cada bit com o estado atual. Se o valor foi alterado, então um som será tocado.

Neste caso em particular, como só temos 8 ficheiros de som, só nos importa verificar os 8 primeiros sensores (onde foram ligados os clips de crocodilo). Isso é visto pela instrução for i in range(8).

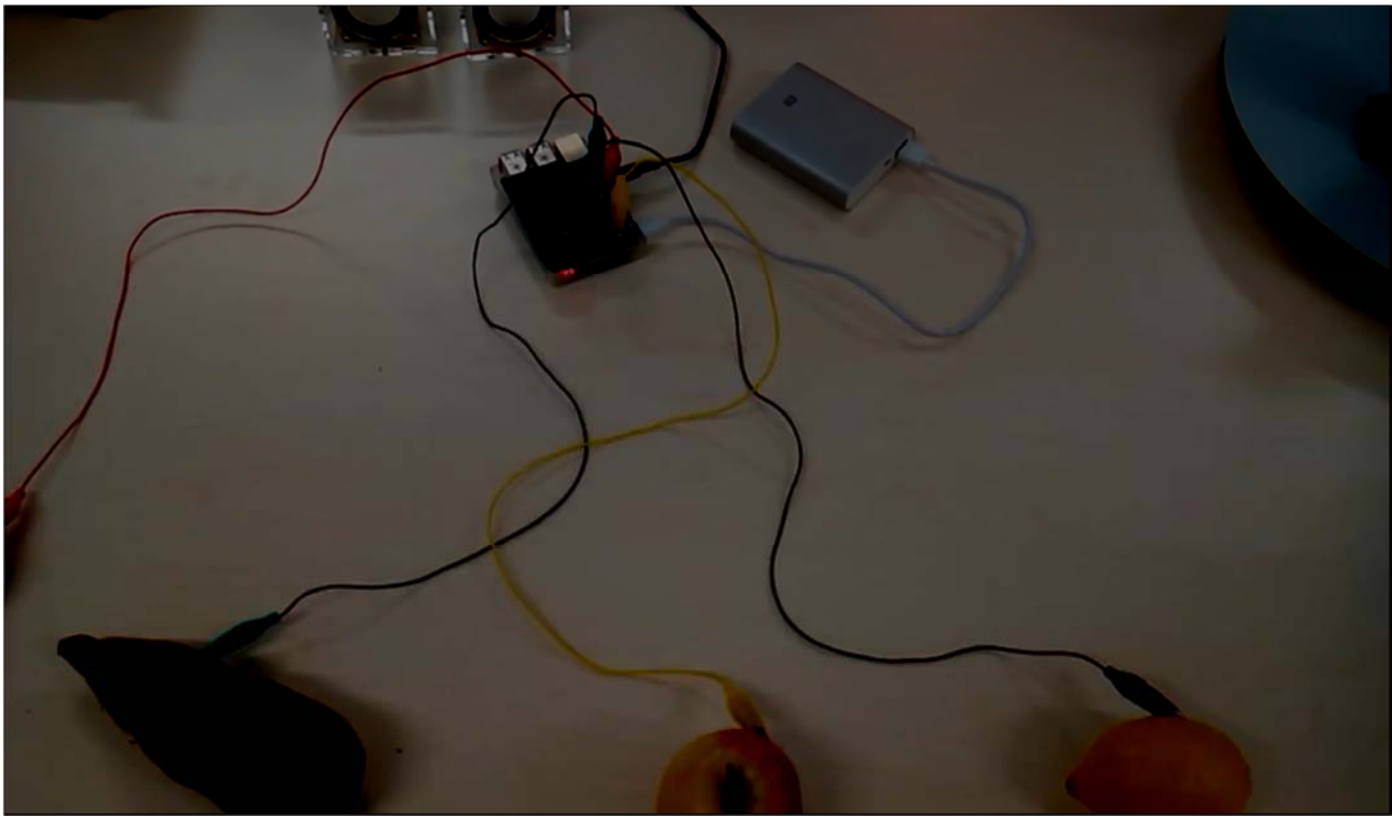
Caso o estado tenha alterado para tocado (neste caso 1), será tocado o respetivo ficheiro correspondente aquele índice. Daí termos igualado os índices aos sensores – torna tudo mais simples.

```
print 'Ctrl+C para sair.'
last_touched = cap.touched()
while True:
        current_touched = cap.touched()
        # verificar cada pin e verificar se foi
tocado
        for i in range(8):
                pin_bit = 1 << i
                # verificar se transitou de nao
tocado para tocado
                if current_touched & pin_bit and
not last_touched & pin_bit:
                        #print 'Tocar qq coisa'
                        if (sons[i]):
                                sons[i].play()

                # nao e preciso verificar se tran-
sitou de tocado para nao tocado
        last_touched = current_touched
        time.sleep(0.1)
```

Codigo completo:

```
__program__     = "PomarMusical"
__author__      = "Bruno Santos (feiticeir0)"
__copyright__   = "Public Domain"

import pygame
import sys
import time

import Adafruit_MPR121.MPR121 as MPR121

cap = MPR121.MPR121()

# Init mixer
pygame.mixer.pre_init(44100, -16, 2, 2048)      #
Avoid sound lag
```

```
pygame.init()                                   #
Initialize

# mapear sons
MAPA_SONS = {
        0: 'Sons/1.mp3.wav',
        1: 'Sons/2.mp3.wav',
        2: 'Sons/3.mp3.wav',
        3: 'Sons/4.mp3.wav',
        4: 'Sons/5.mp3.wav',
        5: 'Sons/6.mp3.wav',
        6: 'Sons/7.mp3.wav',
        7: 'Sons/8.mp3.wav',
}

sons = [0,0,0,0,0,0,0,0]

for key,soundfile in MAPA_SONS.iteritems():
        sons[key] = pygame.mixer.Sound(soundfile)
        sons[key].set_volume(1);


if not cap.begin():
        print 'Error initializing MPR121!'
        sys.exit(1)

print 'Ctrl+C para sair.'
last_touched = cap.touched()
while True:
        current_touched = cap.touched()
        # verificar cada pin e verificar se foi
tocado
        for i in range(8):
                pin_bit = 1 << i
                # verificar se transitou de nao
tocado para tocado
                if current_touched & pin_bit and
not last_touched & pin_bit:
                        #print 'Tocar qq coisa'
                        if (sons[i]):
                                sons[i].play()

                # nao e preciso verificar se tran-
sitou de tocado para nao tocado
        last_touched = current_touched
```



Um exemplo do MPR121 ligado a um kiwi (fora da imagem), uma batata doce, uma maçã e um limão.

AUTOR

Escrito por Bruno Santos

O artigo Pomar Musical – Interface pedagógico com base em Arduino foi gentilmente cedido através da parceira com a Associação de Informática de Castelo Branco

COLUNAS

C# - De List para DataTable em 30 + 2 linhas!

Kernel Panic -

De List para DataTable em 30 + 2 linhas!

Não será de todo estranho, pelo menos para alguns, a necessidade de converter uma lista de um dado tipo para Datatable, em C#. O mais comum, seria criar um novo objecto do tipo DataTable, criar as colunas e iterar a lista, adicionando as linhas ao datatable, a cada iteração. Isto seria no mínimo trabalhoso, além de pouco eficiente. O código seria pouco elegante e semelhante ao seguinte:

```
List<recipes> list = new List<recipes>();

DataTable dt = new DataTable();

PropertyDescriptorCollection props = T
     ypeDescriptor.GetProperties(typeof((recipes));

foreach (var recip in recipes)
{
 dt.Columns.Add(recip.Name,recip.PropertyType);
}
```

Seguido da iteração para colocar os dados no DataTable:

```
foreach (var item in list)
{
 dt.Rows.Add(item.Property1,item.Property2);
}
```

Apesar do código acima funcionar e ser efectivamente simples, num projecto em que se têm diversas listas, de diversos tipos, sempre que se quer passar o conteúdo de uma lista de um dado tipo, para um DataTable, acabar-se-ia por repetir o mesmo código, vez, após vez, após, vez, consumindo mais tempo, criando um código “nada elegante”, demasiado fragmentado e repetitivo.

Como alternativa a isto, podemos utilizar uma solução assente em Gerenics e Reflection! Parece impossível ? Basta recorrer a Extension Methods, que nos permitem “adicionar” métodos a tipos existentes, sem criar novos tipos derivados, recompilar, ou modificar o tipo original.

Os métodos de extensão (Extension methods), são definidos como métodos estáticos , mas chamados usando a sintax de métodos de instância.

Vejamos como se pode fazer isto! Primeiro criamos uma classe, onde iremos colocar o nosso método de extensão (extention method).

```
using System.Collections.Generic;

public static class ListExtensions
{
    public static DataTable ToDataTable<T>(this
                                    List<T>
iList)
    {
        DataTable dataTable = new DataTable();
        PropertyDescriptorCollection
                    propertyDescriptorCollection =
         TypeDescriptor.GetProperties(typeof(T));
        for (int i = 0;
          i < propertyDescriptorCollection.Count;
          i++)
        {
          PropertyDescriptor propertyDescriptor =
                propertyDescriptorCollection[i];
            Type type =
                propertyDescriptor.PropertyType;

            if (type.IsGenericType &&
               type.GetGenericTypeDefinition() ==
                typeof(Nullable<>))

                type = Nullable.GetUnderlyingType
                                        (type);

            dataTable.Columns.Add
                (propertyDescriptor.Name, type);
        }
        object[] values =
            new object
            [propertyDescriptorCollection.Count];

        foreach (T iListItem in iList)
        {
            for (int i = 0;
                      i < values.Length; i++)
            {
                values[i] =
                    propertyDescriptorCollection
                         [i].GetValue(iListItem);
            }
            dataTable.Rows.Add(values);
        }
        return dataTable;
    }
}
```

Com esta classe acrescentada ao nosso projecto, passamos a ter disponível um método chamado “ToDataTable”, nos objectos do tipo *List<T>*. Assim, a passagem do conteúdo de uma lista, para um DataTable, em qualquer parte da solução passa a poder ser feita com as seguintes duas linhas:

```
List<recipes> list = new List<recipes>();

DataTable dt = recipes.ToDataTable();
```

Obviamente o tipo “recipes”, deve ser substituído por o tipo de dados que nos for mais conveniente, uma vez que neste caso o tipo “recipes” foi apenas para exemplo e refere-se a “mil e uma formas de tirar café”!

Esta técnica evita o trabalho de ter de repetir código, ao longo do projecto, bem como de escrever no código cada vez que se pretende transferir dados de uma lista para um DataTable.

Por incomum que possa parecer existem diversas situações em que a transferência de dados de uma lista para um DataTable, pode ser necessária, e será sempre mais fácil o recurso a algo simples como esta solução do que a escrita

e reescrita de código, criando um possível problema de manutenção de código posterior.

A titulo de curiosidade, poderia ser necessário fazer exactamente o inverso, mais concretamente converter um DataTable em lista, e novamente apareceriam uma miríade de formas diferentes para o fazer.

Neste caso e apenas para efeitos de exemplo será feito usando Generics. Apesar de não ser o objectivo deste artigo abordar a transferência de dados de DataTable para List, creio ser de interesse para o leitor um exemplo de como o fazer. Ora para tal criamos um método que irá executar as operações necessárias, e depois invocamos o método sempre que necessário. Vejamos o método:

```
private static List<T> ConvertDataTable<T>
                                    (DataTable dt)
{
    List<T> data = new List<T>();
    foreach (DataRow row in dt.Rows)
    {
        T item = GetItem<T>(row);
        data.Add(item);
    }
    return data;
}

private static T GetItem<T>(DataRow dr)
{
    Type temp = typeof(T);
    T obj = Activator.CreateInstance<T>();

    foreach (DataColumn column in dr.Table.Columns)
    {
        foreach (PropertyInfo pro in
                            temp.GetProperties())
        {
            if (pro.Name == column.ColumnName)
                pro.SetValue(obj, dr
                        [column.ColumnName], null);
            else
                continue;
        }
    }
    return obj;
}
```

Com o método escrito, podemos então chamar este método e converter os dados. Novamente duas linhas são suficientes para realizar esta operação.

```
List<recipe> recipeDetails = new List<recipe>();
recipeDetails = ConvertDataTable<Recipes>(dt);
```

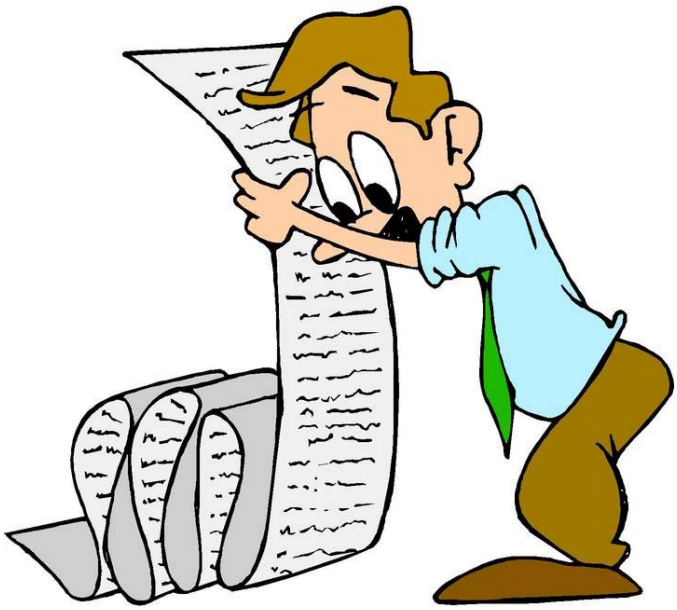
E com isto temos a conversão de List para DataTable e de DataTable para List.

Existem diversas formas de se obter estes mesmos resultados, tal como já foi referido e até pode parecer incomum a necessidade de realizar este tipo de operações. No entanto, por exemplo quando se estão a manipular volumes de dados consideráveis, pode por diversas razões diferentes ocorrer a necessidade de transferir os dados para listas (List) e de volta para DataTable.

Referencias:

https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/classes-and-structs/extension-methods

http://www.asparticles.com/2016/10/different-ways-to-convert-datatable-to-list-in-csharp.html



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

Node.js - Construção de Aplicações Web

Desenvolvimento em Swift para iOS

Node.js - Construção de Aplicações Web

Título: Node.js - Construção de Aplicações Web

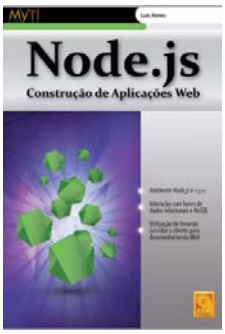
Autores: Luís Abreu

Editora: FCA - Editora de Informática

Páginas: 197

ISBN: 978-972-722-860-7

Formato: Capa Soft



Olá a todos, esta é a primeira vez que faço uma review para a revista Programar, queria desde já agradecer ao António Santos e à FCA pela oportunidade e pelo envio do livro.

Quem me conhece, sabe que dou preferência a um livro impresso aos digitais, apesar de ser um aficionado das tecnologias, a leitura de um livro fisico traz-me um conforto e uma concentração que não consigo nos electrónicos.

O livro Node.js não poderia ter chegado em melhor altura, pois neste momento estou a dedicar mais atenção a esta bela plataforma, que quando aplicada correctamente pode ser uma melhor opção a outras de maior porte, por exemplo o Java EE ou .Net.

Começando pela capa do livro, o leitor mais atento vai verificar que o autor trata as bibliotecas de código como livrarias, bem… para um programador livrarias é um sitio onde compramos livros, já bibliotecas no contexto informático, são boas composições de código que nós facilitam a vida no desenvolvimento de novas aplicações, no entanto é um mero clichê.

Iniciando a leitura do livro começamos com um belo resumo dos excelentes 8 capítulos que compõem o livro, deixando qualquer leitor entusiasmado para ler na totalidade. O termo livrarias parece ter desaparecido nas primeiras paginas, sendo agora tratado pelo termo correcto bibliotecas, mas… aparentemente foi falso alarme já que o autor utiliza tanto um termo como outro ao longo do livro. Esquecendo o termo livrarias, viro o foco para plataforma Node.js, aqui o autor consegue passar perfeitamente a mensagem utilizando poucas palavras e descomplicando o assunto.

No primeiro capítulo a autor justifica o porquê do Node.js, explica a arquitectura assíncrona de eventos, já que esta é uma das grandes potencialidades no Node.js.

De seguida, ainda no capítulo 1 mostra como configurar o ambiente de desenvolvimento, aqui o autor utiliza uma maquina Windows para exemplificar passo a passo a configuração. Ambientes em Mac OS ou Linux não são tratados neste livro, algo que me deixou um pouco desiludido.

No capítulo 2 as coisas começam a ficar interessantes, aqui o módulo *Express* é explicado detalhadamente e com exemplos, este módulo permite ao programador definir rotas e componentes *middleware.* Este é um óptimo capitulo para ficar com boas bases para criar uma aplicação baseada na arquitectura MVC (Model-View-Controller).

O capitulo 3 vem falar sobre motores de templates, nomeadamente o Jade e Vash. O autor explica com mais detalhe o Jade e um pouco mais sucintamente o Vash. Aconselha também ao leitor a utilizar o Jade. No entanto se já efetuou alguma pesquisa sobre motores de templates ou se está a começou a usar, deve ter especial atenção que o motor Jade passou agora a chamar-se *Pug*, o autor não faz referência a isso como é normal, pois o livro começou a ser escrito em 2015 e a alteração de nome de Jade para *Pug* é muito recente. De qualquer forma tudo o que é descrito no livro funciona perfeitamente e mesmo que o leitor tente instalar o Jade o repositório NPM sabe apontar para o módulo correcto.

Para aqueles que adoram criar interfaces espectaculares, então o capítulo 4 é o indicado.

Aqui o autor explica muito detalhadamente como melhorar a nossa aplicação e aumentar a rapidez com que desenvolvemos. Utilitários como Bootstrap, Bower e Grunt são muito importantes para quem vai desenvolver ou desenvolve aplicações Node.js no seu dia a dia.

O Bootstrap permite ao programador criar uma interface que responda bem em qualquer tamanho de ecrã, seja um iMac de 27 polegadas ou um smartphone. O Bower permite gerir bibliotecas otimizando as dependências comuns entre elas, já o Grunt faz com que tarefas rotineiras sejam realizadas automaticamente. Se o leitor ainda não é um *expert*, não tem de ter receio porque tudo é explicado passo a passo com uma linguagem muito simples.

Temas sobre Bases de Dados e *Single Page Applications* são tratados nos capítulos 5 e 6. Para os aficionados do NoSQL o autor explica como utilizar o MongoDB em aplicações Node.js, mas também não deixa de lado as bases de dados relacionais, e explica também mais sucintamente como as utilizar, sendo que é com bases de dados NoSQL que o Node.js se sente como Peixe na água afirma o autor.

Chegando quase ao fim do livro temos o capítulo 7, nos dias que correm este tema não deve ser deixado ao acaso. Nele o autor fala sobre como proteger as aplicações com autenticação e limitar acesso a funcionalidades com base numa autorização. Para o efeito é utilizado o módulo *Passport* que em conjunto com o módulo *Express* já conhecido do capítulo 2, tornam as coisas muito simples para o programador.

Por fim o capitulo 8 é dedicado a tecnologias gerais,

nomeadamente criptografia, socket's, gestão de buffer's e acesso ao sistema de ficheiros. Como leitor e programador recomendo muito este último capítulo, já que nos dá uma ideia do que existe para criarmos aplicações profissionais e robustas.

> ***"No capítulo 2 as coisas começam a ficar interessantes, aqui o módulo Express é explicado detalhadamente e com exemplos, este módulo permite ao programador definir rotas e componentes middleware. Este é um óptimo capitulo para ficar com boas bases para criar uma aplicação baseada na arquitectura MVC (Model-View-Controller)."***

Com isto deixo os meus parabéns ao Eng. Luís Abreu autor deste excelente livro, não só pela escrita directa e clara, mas também pela segurança e certeza com que aborda os temas deste livro, dando sempre o seu parecer com base na sua vasta experiência.

> ***"(...)Aqui o autor explica muito detalhadamente como melhorar a nossa aplicação e aumentar a rapidez com que desenvolvemos. Utilitários como Bootstrap, Bower e Grunt são muito importantes para quem vai desenvolver ou desenvolve aplicações Node.js no seu dia a dia. (...)"***

Recomendo este livro tanto para quem quer iniciar a programação na plataforma Node.js, e para quem quer ter um bom livro de consulta ao lado do teclado.

Obrigado FCA pela publicação e António Santos pelo envio :).

Aguardo pelo proximo :)

Abraço

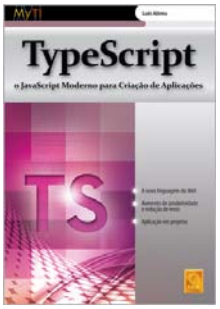
Bruno Horta

AUTOR

Escrito por Bruno Horta

Licenciado em Engenharia Informática pela ESTG. É Senior Java Developer, sendo o seu principal foco o desenvolvimento de aplicações distribuídas e de alto desempenho. Adora desenvolver protótipos para serem utilizados no contexto da Internet das Coisas. É membro fundador do Movimento Maker Portugal. Linkedin: https://www.linkedin.com/in/brunohorta/

TypeScript - O Javascript moderno para criação de aplicações



Título: TypeScript - O Javascript moderno para criação de aplicações

Autores: Luís Abreu

Editora: FCA

Páginas: 168

ISBN: 978-972-722-864-5

O *Javascript (JS)* tem vindo a tornar-se uma linguagem de programação cada vez mais popular, sendo neste momento usada para desenvolver aplicações *standalone*, APIs, aplicações *Web*, entre outras.

Apesar da sua grande versatilidade, o *JS* é diferente de outras linguagens muito utilizadas, como *Java* e *C#.* Esta linguagem não suporta o uso de tipos genéricos, *interfaces* e não é *StrongTyped,* características fundamentais para muitos programadores.

O livro que apresentamos nesta edição dá-nos a conhecer o *TypeScript* (*TS*), uma linguagem/*superset* do *JS*, com o objetivo de aproximar linguagens *como Java e* C# ao *Javascript,* ao disponibilizar algumas características presentes nestas linguagens. O livro intitulado "*TypeScript, O JavaScript Moderno para Criação de Aplicações*" de Luís Abreu, editado pela FCA, permite ao leitor iniciar ou aprofundar o seu estudo em relação ao *TS*, dividindo-se três partes principais: instalação e configuração; descrição técnica; projecto exemplo.

A primeira parte do livro aborda a configuração necessária para compilarmos o *TypeScript* para *JS* utilizando *packages* disponíveis para *NodeJS,* bem como a configuração de ferramentas de *debug* no *Visual Studio Code*. Embora existam outros *IDEs* que suportam a linguagem *TS,* o livro explora apenas o *Visual Studio Code,* sem referenciar outros, como por exemplo o *WebStorm*.

Todas as configurações necessárias são muito bem explicadas, permitindo a leitores sem qualquer experiência em *JS* ou *TS* reproduzi-las sem problemas.

A segunda e principal parte deste livro, aborda a linguagem *TS* de forma técnica, explicando a sua semântica, sintaxe, bem como as capacidades de validação e controlo realizadas aquando da compilação. São apresentados os seguintes temas: tipos básicos; funções; *interfaces*; classes; outros tipos; módulos.

De uma forma muito concisa e com exemplos práticos, toda esta parte do livro demonstra o grande potencial do *TS,* principalmente a nível de prevenção de erros e aplicações técnicas da linguagem. Analisa, ainda, muitas das características próprias desta linguagem que, por exemplo, o *JS* não suporta. Algumas dessas características são a c**apacidade de utilização de tipos genéricos, *interfaces* e *StrongType*.**

Na minha opinião, enquanto programador *Java* com conhecimentos pouco aprofundados em *JS*, penso que alguns utilizadores menos experientes poderão ter dificuldades em entender alguns conceitos adjacentes ao *JS,* pois não são explicados no contexto, apenas apresentados sucintamente.

Outro aspecto que gostaria de ter visto abordado neste livro é o "*Async Await*", conceito familiar para programadores *C#* e *JS*, que já está disponível no *Typescript* desde a versão 1.7.

A terceira, e última, parte lança o desafio de criar um projecto, utilizando o conhecimento obtido na leitura deste livro.

Considero uma iniciativa positiva por parte do autor em criar este desafio aos leitores, pois desta forma permite consolidar os conhecimentos, aspecto que deveria estar mais presente em livros técnicos.

Apesar do *TypeScript,* devido à utilização de *Angular* em alguns projectos pessoais, não ser desconhecido para mim, achei este **livro muito completo e útil para programadores com ou sem experiência em *TypeScript***.

Parabéns ao autor Luís Abreu e à Editora FCA por este excelente livro e por terem abordado o *TypeScript* que, na minha opinião, vem conquistar programadores de *Java*, *C#* e *PHP* e incentivar a utilização do *TypeScript* em projectos.

AUTOR

Escrito por João Sousa

Natural de Leiria, Licenciado em Eng. Informática no Instituto Politécnico de Leiria e Mestre em Computação Móvel pelo Instituto Politécnico de Leiria.

Software Developer na Void Software.

Segurança

A Revolução da Blockchain - A Tecnologia do Futuro

Segurança em Aplicações Android

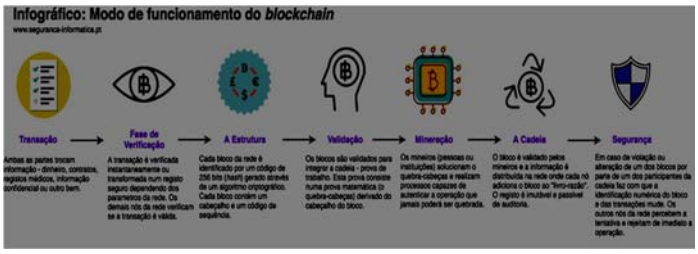
Blockchain and Merkle Tree

A Revolução da Blockchain - A Tecnologia do Futuro

Desde o ínicio do ano de 2009 que uma moeda virtual tem vindo a marcar a diferença no horizonte dos pagamentos digitais. O principal ingrediente do bitcoin [1] é uma tecnologia inderrubável que até então nem tinha sido muito notada — a blockchain [1] [2] (white paper).

A blockchain é uma tecnologia sofisticada que permite salvaguardar registos de uma forma distribuída e não centralizada uma vez que não existe uma entidade de gestão central. Este sistema também conhecido como livro razão, tem o objetivo de guardar registos de transações e os registos de todas essas transações são atualizados em cada nó da rede e não apenas no nó central de gestão (porque efetivamente esse nó não existe). Sempre que é consumada uma transação na rede todos os nós da rede são informados. Ele funciona com base numa cadeia de blocos, nada mais do que um conjunto de informações que são ligadas a cadeias adjacentes. Estes blocos são públicos, no sentido em que todos os nós da rede podem aceder a essa informação. Porém, estes blocos quando são processados não podem ser alterados nem apagados. Além disso, novos registos só podem ser feitos mediante um processo de auditoria.

Em seguida é apresentado um infográfico [3] onde é ilustrado de forma breve o modo de funcionamento DA BLOCKCHAIN.



Esta tecnologia tornou-se de facto tão interessante que foi vista como uma possível implementação de segurança para os mais diversos setores, como por exemplo:

O Dutch Central Bank está a desenvolver um protótipo (DNBCoin) para a sua própria moeda digital [4];

O Deutsche Bank (com sede na Alemanha), o HSBC e o Barclays (ambos com sede na Inglaterra) utilizam tecnologia com base na blockchain desenvolvida pela IBM e com o contributo de outras empresas [5];

O Mizuh Bank no Japão também realizou testes experimentais durante 3 meses [6];

A KRX, operador da bolsa de valores na Coreia do Sul pretende inaugurar uma plataforma blockchain para estimular as transações *off-board* [7];

Também a *startup* norte-americana R3 CEV formou um consórcio de 25 bancos de investimento com o objetivo de implantar uma rede blockchain privada para substituir sistemas internos [8];

A ICAP, uma empresa britânica de serviços financeiros também já usa a *blockchain* [9];

A Nasdaq, onde se negociam ações de empresas de tecnologia em Nova York, usa a tecnologia blockchain para registar troca de ações [10];

MUSE Blockchain, mundo da música, representa um sistema que está reforçando a transparência necessária para a indústria musical, abordando problemas com distribuição de royalties, licenciamento e equidade de artistas [11];

Citizen Ticket visa trazer transparência, segurança e justiça à venda online de bilhetes de eventos [12];

EUA Storj, uma solução de Cloud está atualmente a testar uma rede blockchain para melhorar a segurança e diminuir a dependência de um provedor central [13];

Também os sistemas de votação estão a adotar a tecnologia. A informação relativa ao voto continua registada mas totalmente anónima [14];

A IBM desenvolve projetos que visam aperfeiçoar o sistema blockchain [15].

Ucrânia usa blockchain para identificar propriedades rurais e evitar fraudes [16];

IBM e Samsung desenvolvem um sistema para construir uma rede de dispositivos distribuídos - uma Internet das coisas descentralizada [17];

e Etherify: a primeira empresa portuguesa focada em criptomoeda e 'blockchain' [18].

De entre muitas vantagens, o sistema blockchain oferece:

- O registo em ordem cronológica de todas as transações da rede;
- Todas as novas transações são validadas por cada nó da rede;
- Possui um sistema público, exclusivo, replicado com todos os nós da cadeia;
- A atualização da rede acontece de forma voluntária e descentralizada, isto é, não existe uma entidade central que orquestre toda a rede;

- Recompensa — os nós recebem uma recompensa relativa à tarefa de mineração (embora possa ser alterada a lógica dessa recompensa nas redes públicas e privadas).

E ainda podem ser enumeradas a maior independência, segurança, transparência e agilidade na transmissão das informações uma vez que não existe um terceiro nó de confiança. A nível da banca permite uma redução dos custos de transação e um sistema de pagamento otimizado, com tempo de liquidação acelerado. Dentro dos cuidados de saúde, a blockchain tem também o potencial de proteger os pacientes e as instalações médicas contra percalços de administração e segurança, debates de seguros e infrações de dados clínicos.

"Desde o ínicio do ano de 2009 que uma moeda virtual tem vindo a marcar a diferença no horizonte dos pagamentos digitais. O principal ingrediente do bitcoin [1] é uma tecnologia inderrubável que até então nem tinha sido muito notada — a blockchain [1] [2] (white paper)."

Muitas empresas e até instituições, como os bancos, o próprio Santander (sediado na Espanha), Citibank (sediado nos EUA), Goldman Sachs (sediado nos Estados Unidos), BBVA (sediado na Espanha), Westpac e Commonwealth Bank (sediados na Austrália) e muitos outros, já realizaram experiências com a finalidade de adaptar este sistema criptográfico aos seus sistemas, interesses e finalidades [19].

Afinal de contas, a descentralização acarreta diferentes benefícios para os mais diversos contextos. Para a banca permite, por exemplo:

- Acelerar a liquidação de transações;
- Pagamentos *cross-border* entre bancos "nacionais" e investidores estrangeiros;
- Armazenamento de documentos e contratos;
- *Tracking* de transações;
- Pagamentos entre pessoas que não têm conta bancária; e

"A blockchain é uma tecnologia sofisticada que permite salvaguardar registos de uma forma distribuída e não centralizada uma vez que não existe uma entidade de gestão central. Este sistema também conhecido como livro razão, tem o objetivo de guardar registos de transações e os registos de todas essas transações são atualizados em cada nó(...)"

- Privacidade do cliente final (anonimato).

Hoje em dia a blockchain é uma enorme cadeia de dados digitais, e ela é considerada por muitos "imutável" por uma simples razão: — é extremamente dispendioso contornar uma transação já validada na rede. Atualmente a rede possui um poder de processamento de cerca de 5.988.701.03 TH/s (dados de julho de 2017) [20]. Este número "grande" tem um significado ainda maior — Se algum indivíduo, grupo de indivíduos ou mesmo entidade quiser alterar alguma transação já validada, seria preciso no mínimo 50% + 1 do poder de processamento da rede, isto é, certa de 299.435 TH/s — um valor de processamento gigantesco.

Todavia, um dos maiores problemas na adoção deste modelo criptográfico é que ele é extremamente complexo de aplicar, principalmente porque cada projeto possui o seu próprio esqueleto e também os seus próprios ideais. Existem atualmente esforços de código aberto (open-source), como é o caso do HyperLedger [21] da IBM, o Ethereum [22] e o Corda [23] que são algumas plataformas para o desenvolvimento de sistemas com base na blockchain.

Estima-se que nos próximos anos a blockchain seja largamente usada o que contribuirá para uma melhor segurança na transmissão da informação através da própria Internet. Esta será certamente uma revolução muito semelhante àquela que ocorreu com a Internet.

Referências

[1] https://bitcoin.org/bitcoin.pdf

[2] http://scet.berkeley.edu/wp-content/uploads/BlockchainPaper.pdf

[3] https://seguranca-informatica.pt/infografico-modo-funcionamento-do-blockchain/

[4] http://www.coindesk.com/dutch-central-bank-cryptocurrency-experiments/

[5] http://uk.businessinsider.com/deutsche-bank-hsbc-kbc-natixis-rabobank-socit-gnrale-and…

[6] http://www.coindesk.com/mizuho-blockchain-recordkeeping-digital-currency-trials/

[7] http://www.coindesk.com/korea-exchange-launches-blockchain-powered-private-market-…

[8] https://dailyfintech.com/2016/11/28/why-the-r3cev-blockchain-consortium-is-splintering…

[9] http://www.coindesk.com/icap-blockchain-rosetta-stone/

[10] http://business.nasdaq.com/Docs/Blockchain%20Report%20March%202016_tcm5044-2…

[11] https://bitcoinmagazine.com/articles/three-startups-trying-to-transform-the-music-indust …

[12] https://www.citizenticket.co.uk/

[13] https://storj.io/

[14] http://www.nasdaq.com/article/6-blockchain-applications-that-go-beyond-bitcoin-cm716269

[15] https://www.ibm.com/blockchain/

[16] http://experienceclub.com.br/ucrania-usa-blockchain-em-terras/

[17] http://www.coindesk.com/ibm-reveals-proof-concept-blockchain-powered-internet-things/

[18] http://www.jornaleconomico.sapo.pt/noticias/etherify-nasceu-a-primeira …

[19] http://www.coindesk.com/8-banking-giants-bitcoin-blockchain/

[20] https://blockchain.info/

[21] https://www.hyperledger.org

[22] https://www.ethereum.org

[23] https://www.corda.net

AUTOR

Escrito por Pedro Tavares

Pedro Tavares é atualmente um profissional no ramo da segurança da informação. Desempenha funções como IT Security Engineer, é membro fundador e pentester no CSIRT.UBI e fundador do blog seguranca-informatica.pt.

Segurança em Aplicações Android

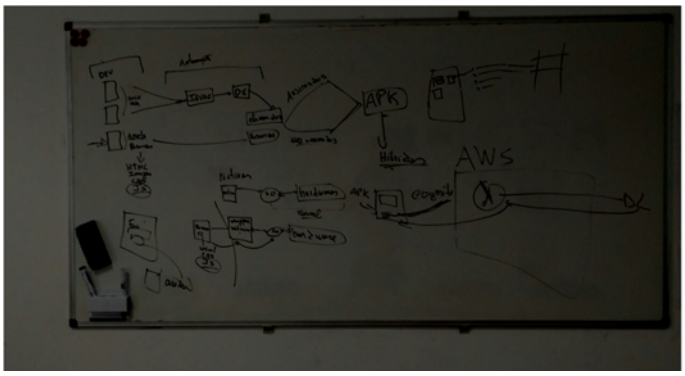
O ficheiro de uma aplicação Android é chamado de Android Package (apk), e não é mais que um ficheiro Zone Information Protocol (ZIP) comprimido.

Começamos com algumas breves questões:

- É possível descomprimir um apk? **Sim**.
- Então, também é possível ler o código-fonte de um apk? **Sim**.
- Os apks são reversíveis através de engenharia reversa? **Sim**.
- Isso quer dizer que, é possível encontrar dados sensíveis como, por exemplo, palavras-passe e Application Programming Interface (API) *keys*, ao longo do código? **Sim**.
- É possível construir um apk totalmente seguro — à prova de bala?

Este artigo tem o objetivo de passar alguns procedimentos de forma a que qualquer *developer*, ou fulano com conhecimentos básicos sobre Android, consiga auditar sua própria aplicação antes que esta seja publicada e maliciosamente explorada.

E respondendo à última questão: -"**Nim**".

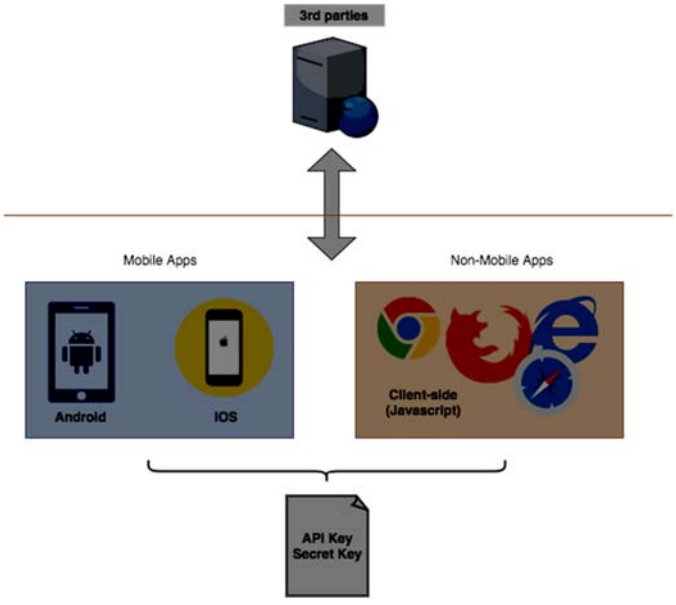


Caça aos Dados Sensíveis

Não existe uma aplicação, sistema, infraestrutura ou até ecossistema à prova de bala e totalmente seguro. Isso deve-se a muitos fatores, p.ex., um desenho deficiente da arquitetura, bugs cometidos em sede de desenvolvimento, software desatualizado, vulnerabilidades *zero-day attack*, fuga de informação, entre outros. Quantos mais parâmetros são mencionados maior se torna o horizonte de ataque. Dito de uma forma mais jocosa, o objetivo da segurança da informação é então dificultar o processo de obtenção de determinado valor no seu estado mais cru. Nesse caso, qualquer fulano que tente explorar e obter informação do sistema, em vez de demorar 30 minutos vai demorar, dias, semanas, talvez anos, extingue-se o sol e o bisneto do bisneto do terceiro filho tenta alcançar o objetivo do tataravô. Confuso? É o objetivo!

O principal segredo para a segurança nasce durante o desenho da arquitetura do sistema. É necessário clarificar a arquitetura, as assunções do sistema, os intervenientes, e depois, começar o puzzle. No caso das aplicações Android existem alguns tópicos que podem e devem ser levantados logo no ínicio do desenvolvimento, p.ex:

- A aplicação irá ser distribuída por centenas de utilizadores, logo o código-fonte vai estar disponível nos smartphones dos utilizadores.
- No caso de comunicação com serviços third-party, é importante pensar como e onde irão ser guardadas as palavras-passe, secret keys e tokens de acesso a esses serviços.
- Consoante o tipo de aplicação, nativa ou não-nativa, é importante incrementar a segurança na forma como o código da aplicação é distribuído na store.



Antes de avançar é importante abordar uns breves conceitos sobre a forma como uma aplicação

Android opera e qual o ciclo de desenvolvimento até à geração do apk final.

APK e o Android

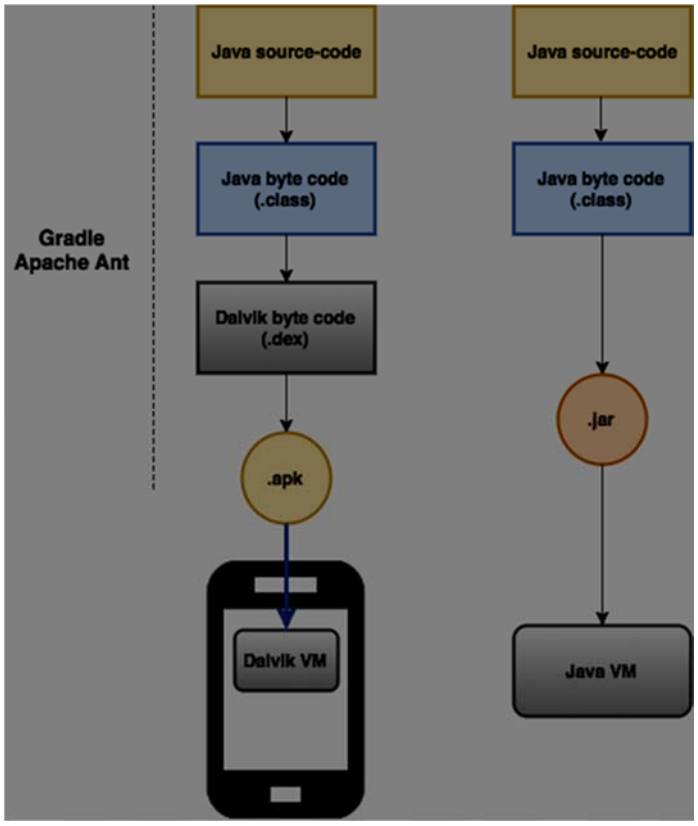
Depois do *download* do apk da store, a aplicação é instalada no smartphone. A forma como o sistema foi desenhado permite:

- Cada apk correr numa Virtual Machine (VM) totalmente isolada;
- Isolamento de processo (UID), i.e., cada processo tem um único ID. Cada processo tem o seu endereçamento de memória, e apenas o processo com o ID XPTO tem acesso à memória XPTO (*shared preferences)*;
- Não há recursos partilhados porque cada *shared preference* só pode ser acedida pelo processo com o ID XPTO (o *owner*); e
- Maior proteção do kernel.

Dalvik Virtual Machine Vs. Java Virtual Machine

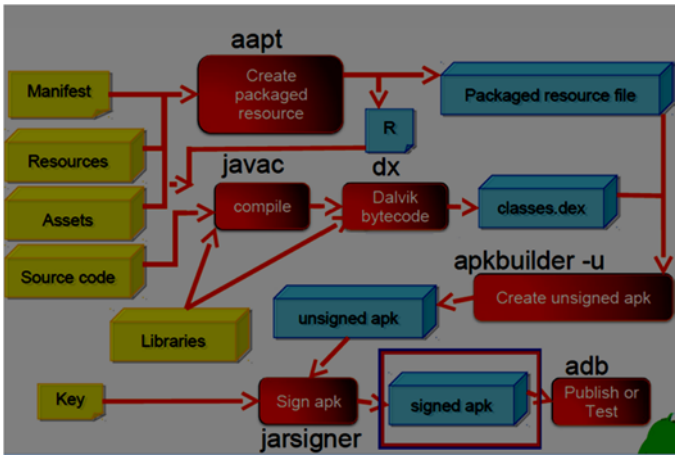
A forma como um apk é interpretado pelo sistema Android é muito semelhante à forma como uma Java Virtual Machine (JVM) interpreta um ficheiro *.jar*. A VM do Android chama-se Dalvik Virtual Machine (DVM).

(Existe também uma outra *runtime* denominada ART e que é bastante mais rápida que a DVM. Pode ser consultada mais informação aqui.)



Existem ferramentas de compilação como o Gradle e o Apache Ant que automatizam o processo de geração da aplicação final, o apk, no caso do Android. De notar que, foram gerados pelo caminho os ficheiros .class, que são os ficheiros com o código Java já compilado. Adicionalmente para que um apk rode numa DVM, é necessário também codificar esses ficheiros em .dex files.

O processo é um pouco mais elaborado do que aquele que foi apresentado acima (ver abaixo).



Os ficheiros Resources, Assets e Manifest não são sequer codificados pelo compilador Java (javac) nem pelo dx. Estes ficheiros são apenas compactados. Está a lembrar-se do ZIP e do UnZIP?

Neste ponto é importante reter o seguinte:

- Os Resources e Assets armazenam outros ficheiros como imagens, fonts, HTML, CSS e Javascript (este último é muito importante) .
- Eles não são compilados (codificados em byte code), apenas são comprimidos junto dos ficheiros .class e .dex.
- A grosso modo, um apk é mesmo um conjunto de ficheiros do tipo: *Resources, Assets, Manifest* + ficheiros compilados (*.class e .dex*).

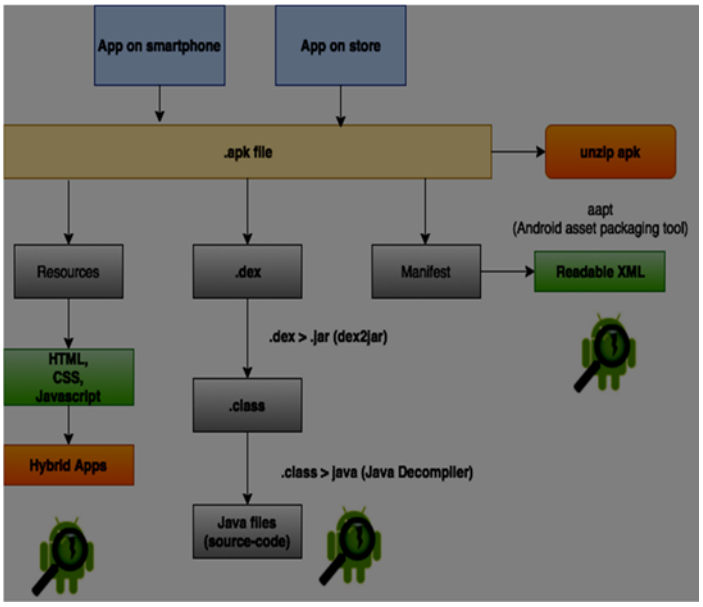
Engenharia Reversa (ER)

Podemos desenhar dois caminhos distintos antes iniciar o trabalho:

- Para aplicações híbridas não é necessário decompilar os ficheiros (.dex e .class), eles apenas vão conter os *plugins* e *libs* dos CPTs. É necessário fazer somente o unzip do apk porque o código está nos Assets.
- Para aplicações nativas é necessário percorrer todo o fluxo de Engenharia Reversa (ER).

SEGURANÇA EM APLICAÇÕES ANDROID

Segue abaixo o fluxo de ER para aplicações Android.



O processo de ER de aplicações Android começa sempre da mesma forma:

- Download do apk do telemóvel ou da store;
- unzip do apk
- Normalmente, para aplicações híbridas, basta o unzip.
- Para aplicações nativas é necessário:
  - Decompilar o ficheiro classes.dex
  - Decompilar os ficheiros *.class para *source -code*
  - Analisar o *source-code*.

1. Download de uma Aplicação Aleatória da Store

De forma a comprovar o que foi dito ao longo do artigo, foi descarregada uma aplicação aleatória da Play Store. Para isso, basta entrar na página da Play Store e selecionar uma aplicação para descarregar.

https://play.google.com/store/apps/details?id=**ID**

Através do SDK do Android, é possível puxar o apk diretamente do smartphone.

```
adb connect xxx.xxx.xxx.xxx:5555
adb shell pm list packages
adb pull /data/app/exemplo.app.pt.apk
```

Para os mais impacientes, podem ser usadas algumas ferramentas *online*, p.ex.:

https://apkpure.com

2. Unzip do apk

Em seguida segue o unzip do apk e a respetiva estrutura dos ficheiros.

```
unzip apk_name.apk
```

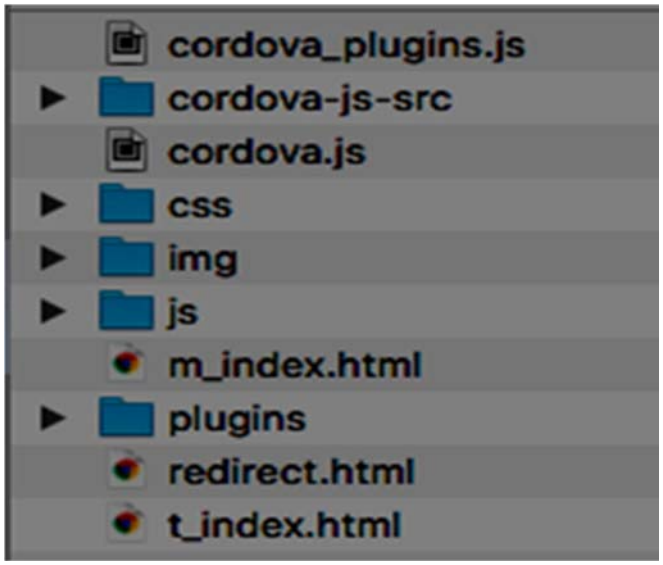
```
-rw-rw-rw-@  1 sirpedrotavares  staff    12876 31 Dec  1979 AndroidManifest.xml
drwxr-xr-x@  5 sirpedrotavares  staff      170 15 Sep 11:32 META-INF
drwxr-xr-x@  3 sirpedrotavares  staff      102 15 Sep 11:32 assets
-rw-r--r--@  1 sirpedrotavares  staff      939 31 Dec  1979 build-data.properties
-rw-r--r--@  1 sirpedrotavares  staff  3404528 31 Dec  1979 classes.dex
drwxr-xr-x@  3 sirpedrotavares  staff      102 15 Sep 11:32 jsr305_annotations
drwxr-xr-x@ 29 sirpedrotavares  staff      986 15 Sep 11:32 res
-rw-rw-rw-@  1 sirpedrotavares  staff   140832 31 Dec  1979 resources.arsc
```

É possível observar dois ficheiros importantes:

- assets, onde vai estar o código de aplicações híbridas (código Javascript); e
- classes.dex, o ficheiro com o source-code em Java.

3. Diretoria Assets

Em aplicações híbridas, como é o caso de apps construídas com o Apache Cordova ou Phonegap, dentro desta diretoria está a estrutura tipicamente de um website.



Analisando o ficheiro m_index.html é possível visualizar uma chamada a um ficheiro Javascript.

```
<script src="js/app.js"></script>
```

Analisando o ficheiro app.js, na diretoria js, é possível encontrar dados sensíveis, secret keys para comunicar com third party applications, como é o exemplo da Amazon Web Services, Linkedin, ou outro tipo de APIs públicas.

```
  linkedIn: {
            callbackUri: "https://####",
            clientId: "####rqbt###rr",
            clientSecret: "####M85####X9"
   }
```

Como uma API não pública necessita de uma API key e uma secret key para que uma comunicação entre o cliente e o servidor seja estabelecida, esses dados são guardados, preferencialmente, em ficheiros do tipo Javascript em aplicações híbridas. Como observado, isto é um problema grave de segurança, uma vez que, foi preciso apenas um simples unzip do apk para chegar aos dados sensíveis.

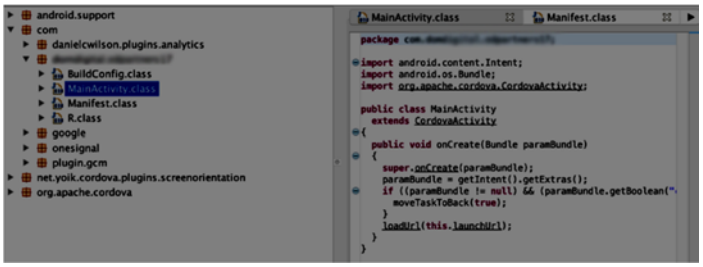
É muito provável que existam bots na Internet a efetuar este tipo de processos de uma forma autónoma.

4. Analisar o Source-code Java da Aplicação - ER

Para uma análise *in-depth*, é necessário seguir o processo de ER mencionado no diagrama mais acima. Para tal, é necessário usar algumas ferramentas disponíveis na Internet, p.ex., dex2jar e Java Decompiler, e efetuar os seguintes passos:

- d2j-dex2jar apk_name.apk (reverter .dex em .jar)
- Abrir o ficheiros com um Java Decompiler

Depois de aberto, é possível visualizar o *source-code* da aplicação.



Como se trata de uma aplicação não-nativa, apenas é feita a invocação do web wrapper, tal como foi mencionado mais acima. O wrapper invoca simplesmente os resources (HTML).

Em aplicações totalmente nativas, seria possível encontrar todas as atividades da app, e procurar recursivamente por palavras-chave dentro do projeto. Dessa maneira, é viável identificar dados sensíveis.

Vantagens de Desvantagens de Aplicações Híbridas

As aplicações híbridas são excelente do ponto de vista da rapidez de desenvolvimento. Torna-se rápido o seu desenvolvimento porque os CPTs já fornecem um conjunto de plugins muito variados. Outra grande vantagem é que, normalmente, o desenvolvimento é cruzado, i.e., multiplataforma, Android, iOS, web, etc. E sobre tudo, o custo de desenvolvimento é muito mais barato que aplicações nativas.

Nem sempre os cuidados de segurança são os melhores. É necessário ofuscar todo o conteúdo, nomeadamente ficheiros Javascript onde vão estar guardados dados sensíveis, e sobretudo, repensar a forma como a app deve ser desenhada.

Relembrar que, o desempenho de uma aplicação híbrida não pode ser comparado ao desempenho de uma aplicação nativa, pois a aplicação não-nativa não fala diretamente com o SO.

Em suma, como vantagens podem ser enumeradas as seguintes:

- Boa experiência de utilização;
- Portabilidade (multiplataforma);
- Baixo custo de desenvolvimento; e
- Rápido desenvolvimento.

Como desvantagens:

- Fraca segurança (sem ofuscação e criptografia, etc);
- Baixa performance (é necessário uma ponte via Javascript); e
- A aplicação é mais pesada que uma aplicação nativa porque agrega um conjunto de plugins do CTP.

Melhorar a Segurança de Aplicações Android

A segurança de qualquer que seja o sistema deve ser pensada desde o seu início. Neste caso em específico, seja uma aplicação Android nativa ou não-nativa, o problema é exatamente o mesmo: — dados sensíveis estão codificados no código e espalhados por centenas de utilizadores.

A ofuscação é uma técnica que pode ajudar a incrementar o tempo de processamento para obter os dados sensíveis, no entanto, eles estão ali (apenas baralhados). Nesse sentido, ofuscação dificulta o processo mas não resolve o problema dos dados codificados no código.

Algumas soluções de ofuscação para aplicações Android são as seguintes:

- Google Code Enclosure (para apps híbridas);
- ProGuard (apps nativas).

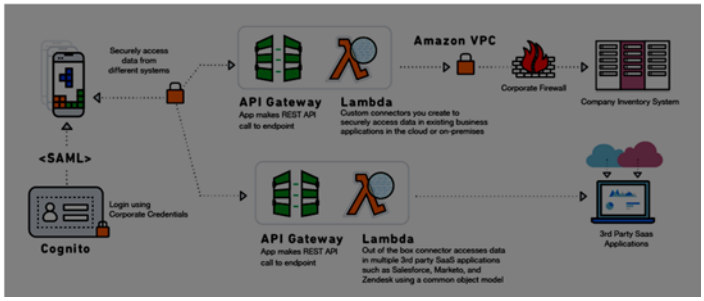
É preciso pensar em outras alternativas para além da ofuscação, p.ex., o Android Key Store System, em que os dados sensíveis são guardados diretamente no processador (hardware).

É uma solução à prova de bala? Não, mas complica em muito a vida de um cracker, e obter dados deste nível de proteção é muito mais complexo do que apenas de um simples “unzip”.

- Existem também outras soluções baseadas em

third parties services, como é o exemplo da AWS e do Meteor. Aqui a computação fica do lado de quem mantém o serviço e a aplicação não precisa sequer de manter registos sensíveis no client-side.

Fica em baixo um caso de uso da AWS para soluções móveis.



Nestes casos, toda a computação fica do lado do provedor de serviço (a AWS, neste caso em específico). A gestão de utilizadores é gerida através do serviço *cognito*, e as callbacks a *third party applications* são geridas dentro do ecossistema do provedor. Esta é uma forma elegante e segura de conceber aplicações móveis.

Tem desvantagens? **Sim**, a aplicação móvel distribuída por centenas de utilizadores é considerada segura (uma vez que não existem dados sensíveis partilhados do lado dos clientes) mas mantem-se "agarrada" a um provedor de serviços (o preço a pagar pela medida de segurança implementada).

Conclusão

Não existem soluções perfeitas para a segurança de um sistema ou de uma aplicação. O segredo está na forma como é conseguido fazer andar o relógio a seu favor. Como foi observado, com uma solução de ofuscação, um indivíduo ao analisar a aplicação, não iria demorar 5 minutos, mas talvez fosse demorar algumas horas a encontrar matéria sensível. Existem muitas outras maneiras que não foram discutidas neste artigo, uma vez que o objetivo principal foi demonstrar e consciencializar a importância do desenho de infraestruturas, sistemas e aplicações de forma a que estes sejam implementados com os mínimos requisitos de segurança.

Estima-se que uma grande percentagem de aplicações distribuídas na Play Store não detenham grandes mecanismos de proteção. Em Portugal, o paradigma não é muito diferente. É importante consciencializar os arquitetos de soluções da importância do uso de mecanismos de criptografia e de segurança. É importante perceber que nos últimos anos os prejuízos por fuga de informação têm sido catastróficos. Basta debruçar um olhar atento sobre um dos últimos grandes *leaks* do verão de 2017, Equifax.

É importante também que as empresas adotem cada vez mais uma mentalidade defensiva, porque já a partir de maio de 2018, existem procedimentos a cumprir, normas europeias que visam penalizar aquelas empresas que não o fizerem (RGPD).

AUTOR

Escrito por Pedro Tavares

Pedro Tavares é atualmente um profissional no ramo da segurança da informação. Desempenha funções como IT Security Engineer, é membro fundador e pentester no CSIRT.UBI e fundador do blog seguranca-informatica.pt.

Blockchain and Merkle Tree

A bitcoin é o nome de batismo de uma criptomoeda que teve um enorme impacto quando foi libertada na Internet e também todo o conceito e tecnologia em seu redor foi alvo de pura análise e investigação nos últimos anos. A aceitação desta criptomoeda foi de tal maneira exponencial que se refletiu de imediato na sua crescente valorização no mercado [1].

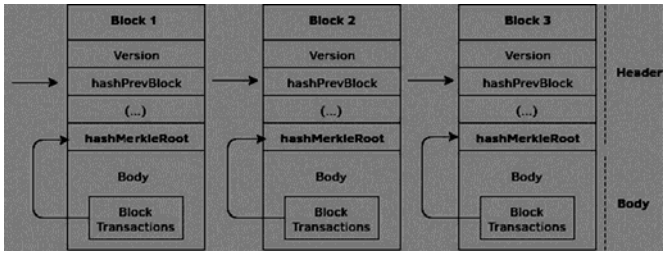
Uma das tecnologias base da bitcoin é a blockchain, que representa uma solução efetiva para resolver o problema das transações duplicadas (*double-spent*) numa rede de pares (peer-to-peer ou p2p). No jargão do bitcoin, a blockchain não é mais que um *ledger* (um 'livro-razão') que guarda o registo de todas as transações ocorridas na rede. Esta tecnologia apareceu de facto na altura certa. Ela permite a implementação de sistemas descentralizados em redes p2p sem a necessidade de uma *trusted third party* como forma de validar transações ou ações num dado ecossistema. Ao invés disso, cada par na rede possui uma cópia do livro-razão onde consegue efetuar todas as validações necessárias sem a necessidade eminente desse terceiro nó de comunicação onde supostamente estaria localizado unicamente o "livro-razão".

Por exemplo, quando o Bob decide efetuar uma transação de 50 BTCs (bitcoins) para a Alice, a transação fica registada na blockchain sem a necessidade de ter que passar por um "banco-central" validador. A transação é automaticamente auditada na rede por todos os nós que a compõem.

Essas transações são armazenadas na cadeia de blocos. Esses blocos têm a seguinte estrutura [2].

FIELD	PURPOSE	SIZE (BYTES)
VERSION	BLOCK VERSION NUMBER	4
HASHPREVBLOCK	256-BIT HASH OF THE PREVIOUS BLOCK HEADER (IN LITTLE ENDIAN FORMAT)	32
HASHMERKLEROOT	256-BIT HASH BASED ON ALL THE TRANSACTIONS IN THE BLOCK	32
TIME	CURRENT TIMESTAMP	4
BITS	CURRENT TARGET IN COMPACT FORMAT	4
NONCE	32-BIT NUMBER (STARTS AT 0)	4

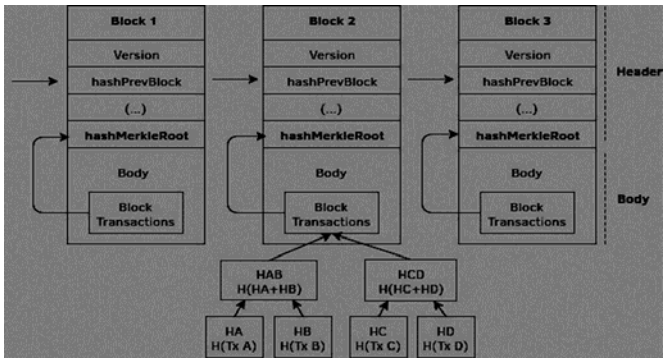
O campo hashPrevBlock guarda a hash do bloco anterior na notação little-endian (invertido). É um registo de 32 bytes que armazena uma hash de 256 bits. O campo hashMerlkeRoot é também um registo de 32 bytes, e armazena uma hash que representa a raiz da Merkle Tree (abordada a seguir). A Merkle Tree é uma árvore que guarda todas as hashes das transações do bloco. O time representa o timestamp de quando o bloco é criado. O campo bits representa a dificuldade do bloco, e por fim, o nonce é um valor incremental que permite aos nós da rede validar um novo bloco candidato para que ele seja adicionado à cadeia. Em seguida é apresentada uma figura que pretende ilustrar como os blocos estão ligados na blockchain.



No Block 2, nomeadamente no campo hashPrevBlock está armazenada a hash do bloco anterior e que permite também estabelecer ligação com os blocos adjacentes. Uma hash identificadora de um bloco é a hash do cabeçalho do bloco, nomeadamente dos campos já citados: Hash(Version, hashPrevBlock, hashMerkleRoot, Time, Bits e Nonce). Essa hash foi a primeira a ser gerada por um par na rede e que satisfazia o desafio colocado.

Merkle Tree

A discussão durante o resto do artigo centra-se no campo hashMerkleRoot do cabeçalho do bloco, que representa a raiz de uma árvore onde são armazenadas as hashes de todas as transações realizadas e associadas ao bloco (ver imagem a seguir), permitindo assim, assegurar também a integridade de todas as transações na cadeia.



BLOCKCHAIN AND MERKLE TREE

As Merkle Tree's foram inventadas em 1979 por Ralph Merkle. O objetivo inicial da sua criação foi introduzido junto das assinaturas de Lamport de tempo-único. Elas representam uma estrutura de dados que contém uma árvore de informações resumidas sobre um pedaço de dados ainda maior, por exemplo: um arquivo, e servem para validar o conteúdo de um arquivo. Elas são usadas para verificar a integridade dos registos, manuseados e transferidos entre computadores ou redes de computadores. Atualmente, o principal uso de Merkle Tree's é para garantir que blocos de dados recebidos de outros pares em uma rede p2p são recebidos intactos e inalterados, e até mesmo para verificar se os outros pares não mentem e enviam blocos falsos.

Uma vez que a transparência é uma das propriedades desta tecnologia, é possível consultar toda a informação referente a blocos e transações de toda a rede desde o seu início (em 3 de janeiro de 2009 às 06:15:05 PM - ver primeiro block -denominado como bloco génesis: -o bloco de altura 0). Em seguida, vamos analisar o bloco com altura 100000 na blockchain, criado a 29 de dezembro de 2010 e identificado com a hash 000000000003ba27aa200b1cecaa-d478d2b00432346c3f1f3986da1afd33e506.

A URL é o seguinte:

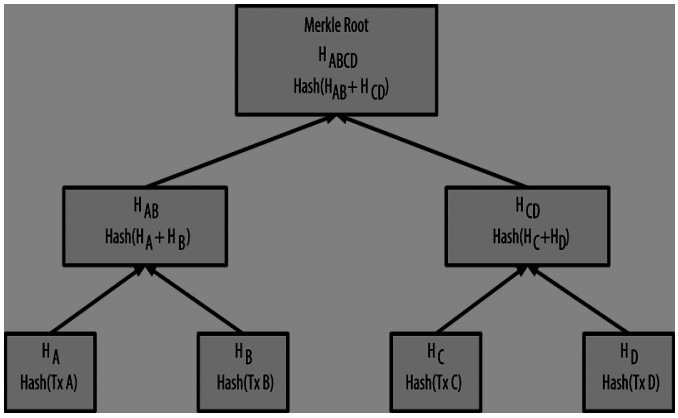
https://blockchain.info/block/000000000003ba27aa200b1cecaad478d2b00432346c3f1f3986da1afd33e506

O valor de hash referente à raiz da Merkle Tree é dado pelo seguinte valor de hash:

f3e94742aca4b5ef85488dc37c06c3282295f-fec960994b2c0d5ac2a25a95766

Este valor é denominado por Merkle Root, e como já foi dito acima, identifica a raiz da árvore. Todas as transações na rede são identificadas através de um hash. A raiz da árvore é formada pela hash da hash de todas as transações em forma de pirâmide. Apenas a Merkle Root é incluída no header do bloco, o que permite verificar a integridade de todas as transações aceites na blockchain. Qualquer alteração ou tentativa de alterar de uma transação anterior irá ser invalidada e rejeitada pela rede.

A seguir fica uma Merkle Tree [4], que tem como Merkle Root (H_{ABCD}) de uma lista de transações: Tx A, Tx B, Tx C e Tx D.



As folhas são os quatro nós de baixo (Tx A, Tx B, Tx C e Tx D). Cada nó possui uma hash, a hash da transação "Hash (Tx A)". O nó superior concatena as hashes das folhas "Hash(H(A) + H (B))". O mesmo processo é aplicado às restantes folhas. A última concatenação de todos os hashess é definido como a raiz da árvore. Este algoritmo tem a forma de uma pirâmide de registos e é a maneira como é verificada a integridade dos blocos e das transações da blockchain.

Usando o exemplo do bloco com altura 100000 da rede bitcoin, as respetivas hashes da árvore são as seguintes:

```
Tx A – 8c14f0db3df150123e6f3dbbf30f8b955a824
                    9b62ac1d1ff16284aefa3d06d87
Tx B – fff2525b8931402dd09222c50775608f75787
                    bd2b87e56995a7bdd30f79702c4
Tx C – 6359f0868171b1d194cbee1af2f16ea598ae8f
                    ad666d9b012c8ed2b79a236ec4
Tx D – e9a66845e05d5abc0ad04ec80f774a7e585c6e8db9
                    75962d069a522137b80c1d
```

A disposição da árvore é então semelhante ao esquema explicado acima. Foi também verificado que a Merkle Root do bloco é dado pela seguinte hash:

```
F3e94742aca4b5ef85488dc37c06c3282295ffec96099
                    4b2c0d5ac2a25a95766
```

A seguir, vamos construir um pequeno programa na linguagem Python de forma a validar a implementação de Merkle Trees para o bloco de altura 100000 da rede, e verificar ainda se o resultado obtido, a Merkle Root, é exatamente o mesmo valor registado nesse bloco.

Implementação em Python

Para começar, é necessário adicionar a lista de transações. Ela contém as quatro transações do bloco 100000 da rede bitcoin.

```
TRANSACTIONS = [
"8C14F0DB3DF150123E6F3DBBF30F8B955A8249B62AC1
D1FF16284AEFA3D06D87",
"FFF2525B8931402DD09222C50775608F75787BD2B87E
56995A7BDD30F79702C4",
```

```
"6359F0868171B1D194CBEE1AF2F16EA598AE8FAD666D9
B012C8ED2B79A236EC4",
"E9A66845E05D5ABC0AD04EC80F774A7E585C6E8DB9759
62D069A522137B80C1D",
]
```

Para implementar o algoritmo é necessário fazer o seguinte:

1. Implementar uma função recursiva para construir a Merkle Tree (def MerkleTree(transactions)).
2. A lista de transações, isto é, o vetor onde estão guardadas as transações. E será iterado com um salto de índice dois. Isto deve-se ao facto de cada ramo ter duas folhas associadas.
3. Em seguida, é concatenada a hash da primeira transação com a da segunda e assim sucessivamente. De notar que, é necessário converter as hashes entre big-endian e little-endian. Esta é uma das muitas particularidade da tecnologia.
4. No final a função retorna o somatório de duas hashes, e essa nova hash é adicionada à árvore "hash(hash (TxA) + hash(TxB))", e por aí em diante.

Função recursiva - Merkle Tree

```
def MerkleTree(transactions):
    global number_of_rounds
    number_of_rounds = number_of_rounds + 1
    if len(transactions) == 1:
        print("\nMerkle Root")
        return transactions[0]
    newTransactionList = []
    #Processing pairs with jump 2
    for i in range(0, len(transactions)-1, 2):
        newTransactionList.append(sum_hash
          (transactions[i], transactions[i+1]))
    if len(transactions) % 2 == 1: # odd, hash
                                   last item twice
        newTransactionList.append(sum_hash
          (transactions[-1], transactions[-1]))
    print "Round [", number_of_rounds, "] -
            Branches [", len(transactions), "]"
    return MerkleTree(newTransactionList)
```

Função para efetuar a soma da hash de duas folhas da Merkle Tree

```
def sum_hash(hash1, hash2):
    #Reverse inputs before and after hashing
                    big-endian / little-endian)
    h1 = hash1.decode('hex')[::-1]
    h2 = hash2.decode('hex')[::-1]
    h = hashlib.sha256(hashlib.sha256
                    (h1+h2).digest()).digest()
    return h[::-1].encode('hex')
```

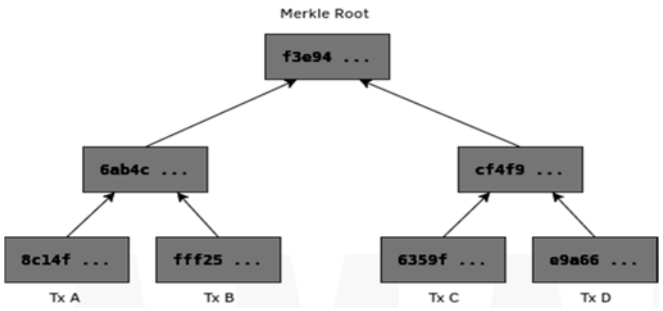
Trecho de código completo

```
#!/bin/python
import hashlib

number_of_rounds = 0

def MerkleTree(transactions):
    global number_of_rounds
    number_of_rounds = number_of_rounds + 1
    if len(transactions) == 1:
        print("\nMerkle Root")
        return transactions[0]
    newTransactionList = []
    #Processing pairs with jump 2
    for i in range(0, len(transactions)-1,
                                        2):
        newTransactionList.append(sum_hash
        (transactions[i], transactions[i+1]))
    if len(transactions) % 2 == 1: # odd,
                          hash last item twice
        newTransactionList.append(sum_hash
        (transactions[-1], transactions[-1]))
    print "Round [", number_of_rounds, "] -
            Branches [", len(transactions), "]"
    return MerkleTree(newTransactionList)

def sum_hash(hash1, hash2):
    #Reverse inputs before and after hashing
                    (big-endian / little-endian)
    h1 = hash1.decode('hex')[::-1]
    h2 = hash2.decode('hex')[::-1]
    h = hashlib.sha256(hashlib.sha256
                    (h1+h2).digest()).digest()
    return h[::-1].encode('hex')

transactions = [

"8c14f0db3df150123e6f3dbbf30f8b955a8249b62ac
1d1ff16284aefa3d06d87",

"fff2525b8931402dd09222c50775608f75787bd2b87
e56995a7bdd30f79702c4",

"6359f0868171b1d194cbee1af2f16ea598ae8fad666
d9b012c8ed2b79a236ec4",

"e9a66845e05d5abc0ad04ec80f774a7e585c6e8db97
5962d069a522137b80c1d",
]

print(MerkleTree(transactions))
```

No final da execução do script o output deve ser exatamente o seguinte para a lista de transações indicadas:

```
Round [ 1 ] - Branches [ 4 ]
Round [ 2 ] - Branches [ 2 ]

Merkle Root
f3e94742aca4b5ef85488dc37c06c3282295f-
fec960994b2c0d5ac2a25a95766
```

No primeiro round, existiam 4 transações para adicionar a Merkle Tree ainda vazia. No segundo round, já só existiam dois. Em seguida, a função recursiva efetuou a última soma de hashess e retornou o valor da sua raiz, nomeadamente a Merkle Root. O esquema abaixo ajuda a perceber esta breve explicação.



Como foi possível verificar, o resultado obtido através da implementação de Merkle Tree's aqui explicada é precisamente a mesma implementação (com algumas alterações) daquela existente na rede bitcoin e que permite guardar o registo das hashess das transações em cada bloco da cadeia. A implementação deste tipo de estrutura em árvore torna-se bastante útil quando é pretendido validar a integridade de um grande conjunto de dados de forma estruturada.

> ***"A bitcoin é o nome de batismo de uma criptomoeda que teve um enorme impacto quando foi libertada na Internet e também todo o conceito e tecnologia em seu redor foi alvo de pura análise e investigação nos últimos anos. A aceitação desta criptomoeda foi de tal maneira exponencial que se refletiu de imediato na sua crescente valorização no mercado [1]."***

> ***"(...) A discussão durante o resto do artigo centra-se no campo hashMerkleRoot do cabeçalho do bloco, que representa a raiz de uma árvore onde são armazenadas as hashes de todas as transações realizadas e associadas ao bloco (ver imagem a seguir), permitindo assim, assegurar também a integridade de todas as transações na cadeia. (...)"***

Referências

[1] https://blockchain.info

[2] https://en.bitcoin.it/wiki/Block_hashing_algorithm

[3] https://pt.wikipedia.org/wiki/%C3%81rvores_de_Merkle

[4] http://chimera.labs.oreilly.com/books/1234000001802/ch07.html#merkle_trees

AUTOR

Escrito por Pedro Tavares

Pedro Tavares é atualmente um profissional no ramo da segurança da informação. Desempenha funções como IT Security Engineer, é membro fundador e pentester no CSIRT.UBI e fundador do blog seguranca-informatica.pt.

No Code

RGPD

Projecto em destaque na PROGRAMAR <Destaque> Hydriney </Destaque>

RGPD

Nos últimos tempos muito se tem falado sobre uma determinada sigla… a *GDPR* ou *RDPD*… aqui na PROGRAMAR decidimos dar uma olhadela neste assunto. Para ficarmos esclarecidos e ajudarmos o leitor a ficar mais esclarecido. E é este o motivo pelo qual este artigo surgiu.

Assim sendo vamos a isto… em português a sigla significa *Regulamento Geral de Proteção de Dados* (RGPD), também conhecida por GDPR - *General Data Protection Regulation.*

Todos nós sabemos que cada vez mais informação é poder. Vivemos num mundo em que tecnologia e informação andam cada vez mais de mãos dadas.

Acredito que praticamente todos nós enquanto cidadãos que fazem as suas escolhas consoante o que mais nos interessa e favorece, que estejamos inscritos em várias bases de dados. Seja porque vamos ao supermercado A e temos um cartão que nos dá descontos, seja porque vamos à loja B e temos mais um registo qualquer acerca da nossa compra enquanto clientes. Ou simplesmente porque estamos registados na nossa operadora móvel ou de internet e mais uma vez nos registamos noutra base de dados. Ou seja, o caro leitor, creio que concordará comigo quando digo que cada vez mais a nossa pegada digital tende a aumentar. E claro, maioritariamente, somos nós os responsáveis por isso.

Então desta foram é normal que esses dados sejam vistos como um aumento benéfico do negócio. Dados criam dados. A loja B sabe que compramos o produto X e Y. Com base nisso podem tentar vender-nos o produto W. Cada vez mais a publicidade é vista em prol da relação emocional que desenvolvemos com determinado produto ou marca. Há uns anos atrás nasceu o conceito de *Brand Value*. Nos tempos que correm nasceu o termo *Data Value*. E porquê? Porque de facto informação é poder.

Imaginemos a empresa A, sendo que esta entidade guarda os dados pessoais dos seus clientes, assim como a lista de produtos adquiridos. Pode usar esses dados, guarda-los, analisa-los, aplicar-lhe algoritmos em prol de quem sabe isso lhe ser útil mais à frente na base do negócio. Mas praticamente nada impedia a empresa A de se juntar com a empresa B, partilharem conjuntos de dados que lhes permitissem estudar melhor as necessidades dos seus próprios clientes para propor mais produtos. A única coisa que poderia eventualmente não permitir isso seria uma cruz naquela parte da ficha de inscrição que informa que “não quero que os meus dados sejam cedidos a terceiros”… claro que como sempre estes pormenores estão nas letras pequeninas. E por norma a ficha de inscrição na maior parte das lojas é um pequeno papel que o operador de loja passa para a base de dados.

Passemos à frente… de toda esta conversa interessa retermos o seguinte… um negócio tem dados, mas pode melhorar se se apoiar nesses próprios dados antecipando as necessidades dos seus clientes. Até aqui nada de estranho.

Para precaver os direitos dos cidadãos europeus para que o tratamento dos dados pessoais seja um direito fundamental à protecção das pessoas singulares, no passado dia 27 de Abril de 2016 foi aprovado no Parlamento Europeu com 95% dos votos, o Regulamento Geral de Proteção de Dados. Neste momento ainda estamos em fase de transição mas a partir do dia 28 de Maio de 2018 será aplicado obrigatoriamente em todos os países da União Europeia. Para os mais curiosos, no caso de Portugal, a lei substituída será a Lei 67/98 (que transpõe para a ordem jurídica portuguesa a anterior directiva 95/46/CE).

Um dos direitos consagrados na Constituição Europeia (artigo 35º) é o direito à privacidade. Um dos grandes desafios desta nova regulamentação é de facto garantir o controlo sobre a privacidade dos dados pessoais assim como a integridade dos mesmos.

É importante referirmos que o conceito de dados pessoais passa a incluir quaisquer dados que sejam susceptíveis de identificar mesmo que de forma indirecta um determinado individuo. Assim, este regulamento europeu reforça os direitos de todos os indivíduos e torna as empresas responsáveis pelos dados pessoais que processam. Por “indivíduos”, entende-se não só clientes, mas também fornecedores e funcionários.

Sem querer entrar em detalhes aprofundados dos artigos que fazem parte desta regulamentação, o objectivo é reforçar os direitos dos titulares desses dados pessoais, o reconhecimento da importância da dimensão da protecção de dados na manutenção desse mesmo direito individual que cada um de nós tem à sua privacidade.

Caso algum destes pontos seja comprometido, as empresas passam a ter a obrigatoriedade de reportar à Autoridade de Controlo qualquer incidente relativo ao comprometimento desses mesmos dados. A nível de sanções as coimas aplicadas podem ir até um máximo de 20.000.000 Euros ou a 4% do volume de negócios global do exercício financeiro

anterior. Além das sanções financeiras, as empresas passam a ter 72horas para avisar os seus clientes de que os seus dados foram comprometidos.

De todos os artigos, os que me pareceram mais importantes, e isto claro, entrando num aspecto mais pessoal, foram os artigos 6, 29 e 32.

O artigo 6 espelha e enaltece a importância da encriptação de dados, estejam ou não esses dados em repouso. Além da encriptação refere que todos os dados devem ser pseudo-anonimizados, isto é, a capacidade de ligar os dados ao seu utilizador /dono deve ser reduzida ao máximo. Trocando por miúdos, os dados devem sofrer algoritmos de "*data reduction*" e "*pseudo minimization*", isto em ambientes não produtivos. Em ambientes de teste e produção, havendo redução de dados e anonimização dos mesmos, o mesmo sai do scope do regulamento.

O artigo 29 refere que pessoas com acesso aos dados não devem processar os mesmos, ou seja a encriptação dos mesmos deve ser transparente, existindo uma arquitectura de segurança máxima (*MSA – Maximum Security Architecture*).

O artigo 32 traz-nos várias alíneas importantes. O tema principal deste artigo é a garantia da não destruição da base de dados, uma vez que a mesma contem transacções que são dados pessoais dos clientes. Assim, este artigo mais uma vez refere a necessidade de pseudoanonimização (alínea A), a necessidade da disponibilidade de sistema em caso de recuperação do mesmo (alínea B), o restauro dos dados (neste ponto há a referência concreta a que o utilizador tem o direito de ser conhecido e o direito a ser esquecido (*right to be know and right to be forgotten*), isto é, se um utilizador quiser ser "apagado" de um sistema deve ser totalmente eliminado, passando a não existir um referencial em relação aos dados do individuo nem na base de dados principal desse mesmo sistema, nem nos dados de backup.

Um outro ponto importante é a referência de que as empresas que devem ter uma pessoa responsável pela implementação de todas estas normas e boas práticas a nível de segurança, o DPO (*Data Protection Officer*) - esta pessoa deve ter conhecimentos jurídicos, conhecimentos tecnológicos e ter conhecimento de gestão de negócios e do sector de actividade.

Não querendo alargar-me muito mais, devo chamar-vos a atenção para os 3 pilares que esta regulamentação quer implementar em cada uma das empresas que detém algum tipo de dados pessoais de clientes, fornecedores ou funcionários…

1. **Avaliar** – Ter conhecimento dos dados, importância dos mesmos e implementação de processos
2. **Prevenir** – Garantir a integridade e segurança dos dados
3. **Detectar** – Detectar possíveis falhas e corrigir as mesmas

Neste contexto, assume-se a extrema importância em definir uma linha de protecção de dados, clarificar a responsabilidade pela protecção desses mesmos dados, elaborar os princípios de protecção dos mesmos e actualizar a tecnologia necessária à implementação e garantia dessas boas práticas.

Mais uma vez chamo a atenção a que o regulamento se aplica não só a todas as organizações da União Europeia, mas a todas as organizações que tenham dados pessoais de cidadãos europeus.

Para quem estiver interessado em saber mais acerca desta temática, fica aqui a página oficial: http://www.eugdpr.org/

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.

SOPHIA, A HUMANOIDE

Nos próximos dias vai decorrer em Lisboa mais uma edição do WebSummit. Espera-se que mais uma vez este evento seja um sucesso. Contudo este artigo não é sobre sobre o WebSummit, mas sim sobre uma participante especial da cimeira que nos despertou a atenção. Dia 7 de Novembro sobe ao palco, Sophia um robot humanoide. E o que é que este humanoide tem de especial pode perguntar o caro leitor…? Sophia foi a primeira humanoide a obter o estatuto de cidadã. Este facto ocorreu no final do mês de Outubro sendo que o Reino da Arábia Saudita concedeu a Sophia oficialmente o estatuto de cidadã do país. E sim, esta foi uma decisão polémica. A verdade é que um país com costumes muito próprios como a Arabia Saudita concedeu a cidadania à Sophia, logo a mesma é uma cidadã desse país. Muitos dizem que a humanoide tem mais direito do que as mulheres desse mesmo país. E isto sim, é algo controverso.

Voltando um pouco atras, Sophia foi criada pela Hanson Robotics, pela mão do CEO da empresa, o Dr. David Hanson. Este senhor tem já reputação de criar robôs que parecem e atuam de forma humana. Sophia foi criada para ser um robot social, o seu design foi projetado para poder ser uma mais-valia para ajudar nos cuidados de saúde, terapia, educação e aplicações de atendimento ao cliente. É um facto de que os robots são projetados para serem parecidos com os humanos mas a verdade é que a Sophia é talvez o robot mais avançado do mundo neste aspeto das parecenças humanas.

Esta humanoide tem duas câmaras nos olhos e graças aos avançados algoritmos de inteligência artificial do qual é dotada é capaz de processar dados visuais, reconhece pessoas, assim como dados da conversa, conseguindo manter uma conversa fluida, fazendo contato visual, criando laços e relações com as pessoas. É também capaz de interpretar expressões faciais e reproduzi-las. Quanto mais interage com as pessoas, mais aprende, conseguindo aumentar a sua inteligência e conhecimento com o passar do tempo.

A Hanson Robotics admite que um dos seus objetivos principais é que a Sophia seja consciente, criativa e capaz como um ser humano. Em algumas entrevistas o CEO da empresa admitiu que crê que em algumas décadas os humanoides andarão de facto entre nós, ajudando-nos e tornando-se nossos amigos.

Os leitores mais apaixonados pela inteligência artificial, certamente se lembram da ELIZA. Um programa criado na década de 60 por Joseph Weizenbaum no laboratório de inteligência artificial do MIT. Claro que a Eliza apesar de conseguir manter uma conversa com os mais distraídos, ao fim de algum tempo torna-se falível. Contudo o conceito por trás deste projeto creio ser o mesmo. Estou a lembrar-me também de um outro projeto amplamente conhecido como a Alexa. Com recurso a um pequeno dispositivo que interage connosco (Amazon Echo Dot) é capaz de aprender, responder e executar pequenas tarefas. A diferença principal da Sophia aos olhos das pessoas é o seu aspecto.

A verdade é que a Sophia é uma versão mais elaborada e muito melhor, é um facto. E é continuamente aprimorada sendo a coqueluche da Hanson Robotics. Acima de tudo a sua forma humanoide faz com que seja facilmente interagivel com a mesma. Do pouco que se sabe sobre o projeto, sabemos que foi pensada para se parecer com a atriz Audrey Hepburn, uma das mulheres consideradas das mais bonitas do ultimo seculo.

Acerca do software e do hardware envolvido no desenvolvimento deste robot pouco se sabe. Foi inicialmente concebido na primeira década do seculo XXI, no seio do projeto "Loving AI" que consiste no desenvolvimento de software que permite aos robôs humanoides interagir com as pessoas socialmente e de forma capaz. Este projeto que centrou-se na Sophia e fornece à mesma uma personalidade e conteúdo cognitivo, linguístico, preceptivo e comportamental. E é este facto que permite que existam as interações humanas e que as mesmas sejam aprofundadas.

Do pouco que se sabe, sabe-se que esta humanoide além de usar tecnologia de processamento distribuido está dotada de ferramentas PNL (Programação neurolinguística). A PNL é um subdomínio da inteligência artificial que é centrado na palavra. Varia desde a entrada da voz à saída da fala, isto é,

a análise da palavra varia desde a determinação da frequência do uso de palavras e da sua colocação até à correspondência de padrões, derivação de palavras e etiquetagem e análise de parte de fala. Uma outra tecnologia presente no projeto é o chatscript que é um também um sistema industrial com uma ampla gama de tarefas de PNL em inglês tendo sido arquitetado em consideração à expressividade, velocidade, tamanho, capacidade e capacidade de manutenção de dados.

Em setembro de 2017, realizou-se um pequeno estudo piloto, envolvendo este humanoide, que liderou os assuntos humanos através de diálogos e exercícios centrados na meditação, visualização e relaxamento. Aparentemente este estudo foi um sucesso, e permitiu demonstrar qualitativamente a viabilidade da abordagem e a capacidade de uma interação apropriada humano-robô para aumentar o bem-estar humano e promover a consciência humana.

Sophia é uma máquina inteligente em evolução, é um facto e a sua semelhança humana é inegável e invariavelmente vai conseguir criar laços com humanos, e conseguir que tenhamos gosto e curiosidade em fazê-lo. Mas numa realidade em que muitos se queixam de que os computadores estão por exemplos a tomar o lugar do emprego humanos, até que ponto estamos preparados para os humanoides viverem entre nós?

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.

Projecto em destaque na PROGRAMAR

<Destaque>Hydriney</Destaque>

Hydriney – A app que o vai ajudar a controlar os cálculos renais

Os cáculos renais, popularmente conhecidos por "pedras nos rins" são um dos problemas mais comuns na área clínica de Urologia. Os cálculos renais formam-se por meio de cristais que se separam da urina e que ao unirem-se formam pedras.

Os doentes com cálculos de ácido úrico tomam o medicamento Uralyt-U. O calendário de controlo é um folheto associado ao medicamento e que permite o apoio fulcral ao tratamento da doença. Este artefacto permite ao doente registar valores para que o médico possa verificar se a situação do paciente se encontra em ordem.

Estando a tecnologia constantemente na ponta dos nossos dedos, é possível através da aplicação Hydriney haver a monitorização constante de cada paciente, de forma individual e personalizada. "Hydriney" é uma aplicação inovadora e simples que funciona como "agenda pessoal" da doença. A aplicação pressupõe uma interação diária com o utilizador, dadas as características desta doença, que requerem que haja a gestão e registo dos valores em base diária. A aplicação permite gerir o tratamento de cálculos renais através de vários indicadores: o registo do consumo de água, o registo da medicação e medição do pH da urina.

TECNOLOGIAS UTILIZADAS

A aplicação foi desenvolvida na plataforma Android Studio na linguagem de programação Java. Os dados foram armazenados numa Base de Dados SQLite, pois houve a necessidade de uma base de dados eficiente, com um bom desempenho, flexível e que pudesse ser facilmente copiada e restaurada.

ASPETOS TÉCNICOS

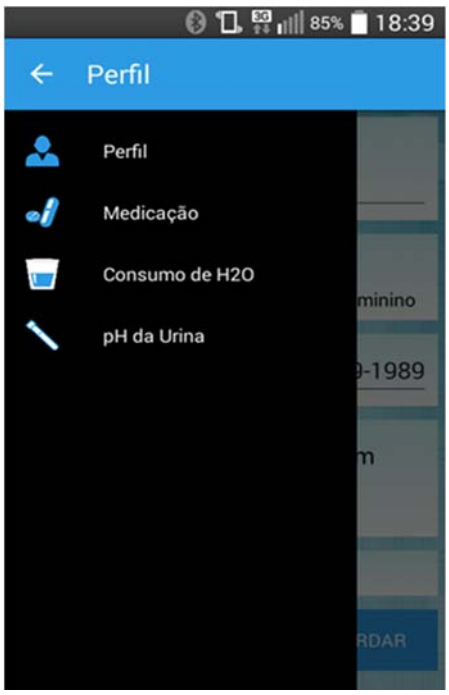
"Hydriney" é uma aplicação que consegue adaptar-se aos diferentes fusos-horários e às principais línguas dos hipotéticos utilizadores (português, inglês, francês e espanhol). O utilizador define os seu objetivos de uso, transcreve a sua prescrição e a partir disso a aplicação vai ajuda-lo a automatizar os registos para a melhor tomada de decisão no tratamento. Para os módulos de medicação e medição do pH da urina a aplicação apresenta um ecrã de registo personalizado que permite visualizar, adicionar, alterar ou remover. Para cada módulo a aplicação existe um gráfico intuitivo e com um aspecto limpo para que o utilizador visualize de forma rápida e simples a variação dos seus níveis de consumo de água e pH da urina. Os gráficos apresentam 3 escalas: últimas 24 horas, últimos 30 dias dias do mês e últimos 12 meses do ano.

OBJECTIVOS

Esta aplicação é apenas uma ferramenta de registo de informações e de auto-monitorização e não se destina a diagnosticar, prevenir ou recomendar tratamento para a doença. As consultas presenciais continuarão a existir e a ser importantes na relação humana profissional de médico-paciente, no entanto, com o uso desta tecnologia móvel, estas consultas ganham um apoio extra e tornar-se-ão até mais eficazes e produtivas pois podem prolongar-se para lá do espaço físico do consultório/centro de Saúde/hospital. Nas consultas periódicas os pacientes deverão facultar os registos efetuados na aplicação para que o médico os possa aconselhar de modo mais eficiente. A aplicação mostrará ao doente mensagens e gráficos, que por exemplo, permitem identificar padrões altos ou baixos e relacioná-los com os hábitos de vida errados que lhe deram origem e assim ajudar a melhorar o controlo dos cálculos renais no seu percurso. Desta forma o médico analisará a informação mais facilmente e poderá ajustar a medicação ou regular os intervalos entre medições.

FUNCIONAMENTO

O menu principal da aplicação surge lateralmente e apresenta uma listagem das funcionalidades existentes. As funcionalidades existentes são: Perfil; Medicação; Consumo de H2O; pH da Urina. Há uma indicação de três listras horizontais para demonstrar a existência do menu lateral à esquerda, seguindo o padrão de design da Google (Menu Navigation Drawer).



Módulo de perfil:

Quando a aplicação é iniciada pela primeira vez, o utilizador é direcionado para o módulo de "Perfil". O "Perfil" apresenta um formulário que está dividido em 5 secções: Nome, Data de Nascimento, Género, IMC e Objetivo diário (de consumo de água). Na secção do IMC, o utilizador deve introduzir o seu peso e altura para depois ser calculado o IMC (Índice de Massa Corporal). O IMC resulta da divisão do peso pelo quadrado da altura, sendo que o peso é dado em quilogramas (Kg) e a altura em metros (m). O utilizador deve também estabelecer algumas metas, como definir o objetivo diário de ingestão de água, em litros (L).

Módulo de consumo de água:

O módulo do Consumo de H2O divide-se em dois separadores: "Actual" e "Histórico". O separador "Actual" permite ao utilizador registar, ao longo do dia, os consumos de água. Este layout permite ao utilizador ir registando a ingestão diária de água. Para o efeito, deve pressionar a(s) garrafa (s) correspondente(s) ao volume de água que acabou de beber, existindo 5 volumes de garrafa ajustáveis às necessidades padrão do utilizador: 1.5 L, 1 L, 0.5L, 0.33L e 0.10 L. Á medida que o utilizador vai registando o que bebe, o gráfico circular atualiza, em tempo real, a percentagem de água bebida em relação ao objetivo (definido no layout Perfil). Os valores registados vão surgindo à direita do gráfico, podendo o utilizador anular um registo ou mesmo a totalidade (através da garrafa "VAZIO", com posição invertida).

Após a introdução dos valores, a aplicação tem a capacidade de mostrar no separador “Histórico” o histórico detalhado e gráfico do consumo de água de hoje, do último mês e também do último ano.

Módulo de medição do pH da urina:

A aplicação apresenta também um módulo para registo do pH da urina. Doentes que tomam o medicamento Uralyt -U necessitam de medir e registar o pH da urina 3 vezes ao dia: antes do pequeno-almoço, antes do almoço e antes do jantar. A interface deste módulo é limpa e uma caixa com quatro divisões divide as medições por turno – manhã, tarde, noite e madrugada.

Esta funcionalidade facilita a realização dos registos de pH, substituindo o papel, para que seja mais fácil não só o ato de registar em si mas também o agrupamento e organização por fases do dia. Esta análise dos dados por fases do dia ajuda a traçar o perfil do doente e o registo em papel dificulta esta tarefa. Ao inserir as medições de pH, a aplicação mostra no separador “Histórico” um gráfico intuitivo para que fique mais claro sobre a evolução do pH da urina. Os gráficos melhoram a compreensão das oscilações dos dados registados e promovem um melhor autocontrolo de fatores específicos da doença, tornando mais intuitiva a apreensão da condição do doente por parte do médico.

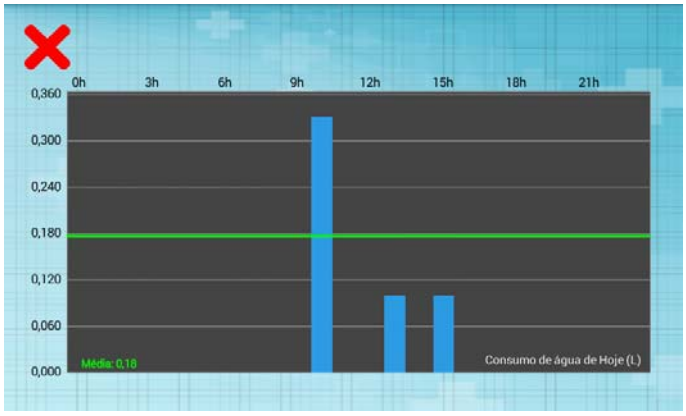
Módulo de medicação:

Nem sempre é fácil guardar na memória a hora certa de tomar um medicamento ou administrar a quantidade de doses necessária até o fim do tratamento. A aplicação Hydriney permite detalhar a programação dos medicamentos com diversas opções de dosagem (comprimidos, caixas etc.) e intervalos de medicação. À semelhança do módulo anterior, este módulo apresenta também uma caixa com quatro divisões divide os medicamentos por turno – manhã, tarde, noite e madrugada. Cada caixa apresenta o nome do medicamento a tomar, a hora a tomar e a flag se já foi tomado ou não.

Planos para futuro

Tendo consciência que haverá muitos aspetos a serem melhorados, continua a haver necessidade de trabalho futuro e necessidades a suprir. Pretende-se dar continuidade ao desenvolvimento da aplicação e torná-la mais confiável e completa no mundo das aplicações móveis. Pretende-se estender as funcionalidades da aplicação de gestão do dia-a-

dia do doente adicionando a capacidade de proporcionar alarmes ou avisos através de regras que analisem os registos efetuados pelo paciente. Os dados que o utilizador insere diariamente na aplicação poderão ser utilizados para detetar padrões nos registos efetuados, tendo em conta o que é o conhecimento médico inscrito na aplicação. Isto poderá refletir-se em avisos ao paciente sobre padrões que indicam desvios do controlo da doença ou mesmo em novos conselhos que são criados baseados nas regras médicas. Quando as funcionalidades anteriores forem implementadas, a aplicação será disponibilizada no Google Play por forma a avaliar a sua popularidade e aceitação.



AUTOR

Escrito por Tânia Valente

Natural de Coimbra, licenciou-se em Engenharia Informática pelo Instituto Superior de Engenharia de Coimbra e tem mestrado em Human Computer Interaction. É entusiasta nas áreas de Business Intelligence, UI/UX Design e Desenvolvimento (Web e Mobile).

Veja também as edições anteriores da Revista PROGRAMAR

57ª Edição - Julho 2017

56ª Edição - Maio 2017

55ª Edição - Março 2017

54ª Edição - Janeiro 2017

53ª Edição - Agosto 2016

52ª Edição - Março 2016

e muito mais em ...

DUVIDAS?

IDEIAS?

AJUDAS?

PROJECTOS?

portugal-a-programar
.org