# PROGRAMAR

REVISTA PORTUGUESA DE PROGRAMAÇÃO • WWW.PORTUGAL-A-PROGRAMAR.ORG

EDIÇÃO #35 - JUNHO 2012

ISSN 1647-0710

# RIA

## DESENVOLVIMENTO DE APLICAÇÕES WEB RICAS (RIA) COM EXT JS 4 E RAILS 3

NO CODE

A ENTREVISTA
PEDRO ANICETO

ANÁLISE
DO LIVRO C# 4.0

COLUNAS

VISUAL (NOT) BASIC
XNA
ATAQUE AO QUINTAL (PARTE 2/2)

ENIGMAS DO C#
O ESTRANHO CASO DOS ENUMERADOS

A PROGRAMAR

GERENCIAMENTO DE HORAS

ARDUINO E O CÁLCULO DA FFT

PASCAL REGISTOS VARIANTES

O ECOSISTEMA UMBRACO (4.7.X)

COMUNIDADES

PTCORESEC
SEGURANÇA NA WEB PARTE 2

AZUREPT
WINDOWS AZURE SECURITY

# EDITORIAL

## EQUIPA PROGRAMAR



ISSN
1 647-071 0

## TTL 255

Desde março de 2006 que a chegamos junto de vós leitores, com a coragem, brio, humildade, profissionalismo, vontade, que compõem e editam esta Revista.

O tempo passa e como tudo, a tecnologia amadurece. Nós também temos feito o nosso esforço nesse sentido, dando o nosso melhor, voluntariamente para vos trazer esta publicação.

Ao contrário de tanta outra coisa em tecnologia, não temos o "garbadge colector", nem um TTL com duração definida. Não precisamos de "destruir objectos", não temos essa preocupação, pois as referências não são apagadas, são guardadas como lições para as edições seguintes.

Além poderá pensar porquê este titulo! Mais uma mensagem de erro ? Não, é bem isso, é uma mensagem de certeza, de confiança de vontade, da certeza de que iremos continuar cá, que faremos melhor, a cada edição, que apesar das mudanças, e dos tempos de mudança, nós não vamos "desistir", não vamos "deixar de existir", não vamos parar.

Em março de 2006 um grupo de pessoas entusiasmadas pelo projecto, lançou a primeira edição da Revista PROGRAMAR. Numa altura em que não se encontravam boas publicações de programação, escritas em português, para quem lê português! Em Junho de 2012, seis anos depois, a revista continua, não parou, não esmoreceu, não morreu, não "baixou a guarda". Bimestralmente voltamos com uma nova edição! Por vezes com esforço e noitadas, mas muito boa vontade! A boa vontade, que nos pauta, que nos compõe, a boa vontade de que esta revista é feita!

Por brincadeira, esta semana que passou, enquanto trabalhava na revista, pediram-me que a comparasse com alguma coisa do mundo da tecnologia… "Estranho pedido", pensei! Mas sem grandes pensamentos, disse que não a poderia definir em nenhuma linguagem de programação, sem ser "tendencioso", por esse motivo aquilo que eu via na tecnologia que se assemelha à revista, seria um pacote IPv4, pois tal como os pacotes IPv4 temos um "Header", que contem a versão (no nosso caso a edição), um "internet header lenght" (no nosso caso o tamanho do header), o "Differentiated Services Code Point ", o "Explicit Congestion Notification", (quando por algum motivo atrasamos uns dias a edição), o Total Length (o tamanho total da edição), a "Identification" , os restantes 16 bits referentes às flags e fragment off-set, seguidos daqueles oito bits que para mim são os mais importantes, o Time-To-Live, que para nós será 255 (tempo indefinido), pois não prevemos parar de trabalhar. Prevemos crescer e continuar a fazer um trabalho cada vez melhor, feito por vós e por nós, para vós leitores da revista. Os 32 bits que se seguem, somos nós equipe e comunidade que edita a revista, os outros 32bits são todos os nosso leitores (vocês). Os últimos 32bits são tudo aquilo que a revista é.

Face a esta resposta, quem me perguntou, ficou sem argumentação, dizendo apenas "bem definido". É a primeira vez que escrevo um editorial sem pensar duas vezes, idealizado num momento, no meio de uma conversa, mas com muito sentido, para mim.

Até à próxima edição!

António Santos
<antonio.santos@revista-programar.info>

# ÍNDICE

## TEMA DE CAPA



## A PROGRAMAR



## COLUNAS



## EVENTOS



## Analises



## COMUNIDADES



## No Code



## EVENTOS

| | |
|---|---|
| 09 de Junho 2012 | V Reunião Presencial da Comunidade NetPonto em Coimbra |
| 09 de Junho 2012 | SharePointPT - 21ª Reunião da SPUGPT |
| 20 de Junho 2012 | Windows Azure [R]Evolution - Lisboa |
| 29 e 30 de Setembro 2012 | WordCamp Lisboa 2012 |

Para mais informações/eventos: http://bit.ly/PAP_Eventos

## Google melhora motor de pesquisa com o Knowledge Graph

A Google introduziu um novo mecanismo de pesquisa no seu motor de busca. Chama-se "Knowledge Graph" e o que faz é dividir a informação que é relevante da que pouco interessa. O Google deixa de ver as palavras como meros algoritmos e passa a reconhecê-las tal como elas são. Veja o vídeo explicativo.

"Quando pesquisas não pretendes encontrar uma página na Internet, pretendes alcançar respostas, perceber conceitos e aprofundar conhecimento".

É com este conceito que a Google lança o "Knowledge Graph", o novo e apurado motor de pesquisa que vai deixar de mostrar apenas informações e simples resultados dos termos pesquisados para passar a mostrar um resultado que se adapta melhor à procura que o utilizador faz.

O novo mecanismo da Google vai funcionar como uma grande teia de informação que consegue estabelecer relações históricas, sociais e temporais entre diferentes termos.

No "Knowledge Graph" as palavras deixam de ser vistas como meros algoritmos de pesquisa e passam a ser reconhecidas como elas são, seja uma pessoa, uma cidade ou uma empresa.

A título de exemplo, se o utilizador fizer uma pesquisa sobre a física Marie Curie, vários podem ser os resultados, desde a biografia da cientista, até ao filme feito sobre ela.

No Gráfico do Conhecimento (em tradução livre), ao escrever o termo que procura vai aparecer, do lado direito da página de resultados, uma descrição da cientista com nome completo, ano de nascimento e morte, nome do marido, dos filhos, onde viveu, entre outras informações.

Por baixo dessa informação vão surgir outros dados relativos que podem vir a interessar o utilizador, neste caso, podem aparecer nomes de mais cientistas ou dados sobre as descobertas de Marie Curie.

O sistema de relacionamento de informação não é novidade no mercado, sendo que o Facebook já faz sugestão de amigos baseado nos gostos em comum que ambos possam ter, contudo o "Knowledge Graph" marca a diferença no aspeto em que tem "consciência" ao apresentar resultados.

O novo mecanismo percebe o que é procurado, não se limitando a fazer uma rede de palavras semelhantes.

"O "Knowledge Graph" não é infalível, mas aprende com os erros e com os resultados positivos acumulados.

Se for sugerido ao utilizador um determinado conteúdo que não tem nada a ver com o verdadeiro motivo da pesquisa, assim que o erro seja reportado, o motor de busca regista e da próxima vez que alguém fizer a mesma pesquisa vai ter em conta o sucedido.

Para além disso, o Google Search vai recolher dados das utilizações corretas para tornar as sugestões ainda mais precisas.

Esta nova função é o resultado de um trabalho realizado nos últimos dois anos e levou a que a Google analisasse, recolhesse e catalogasse não apenas as páginas web, mas também os seus conteúdos.

Ao fim deste tempo foram recolhidos mais de 500 mil objectos e mais de 3,5 mil milhões de referências e relações entre estes objectos.

A atualização no modelo de busca ainda só existe nos Estados Unidos da América e prevê-se que nos próximos dias fique disponível por todo o mundo.

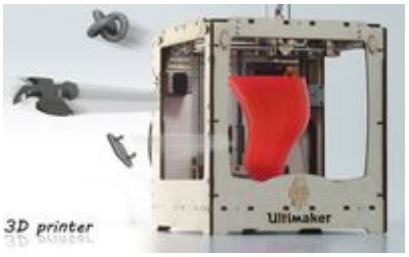
Se quiser ter acesso ao vídeo de apresentação do Knowledge Graph poderá pesquisar em www.youtube.com por "ntroducing the Knowledge Graph".

*Fonte: Jornal de Notícias online*

## Vêm aí as impressoras 3D de baixo custo

Inicialmente reservadas à indústria, pelos seus custos de aquisição e manutenção, e maioritariamente dedicadas à criação de protótipos ou peças industriais, as impressoras 3D são uma realidade cada vez mais próxima do consumidor.

São várias as notícias que ultimamente dão conta do lançamento de modelos de preços mais reduzidos, que deixam antever que o dispositivo pode em breve começar a fazer parte do nosso quotidiano e, provavelmente vir a transformar-se num gadget "apetecido".




A Ultimaker é uma das máquinas que integra o grupo das chamadas impressoras 3D domésticas e já pode ser adquirida há algum tempo. Com 9kg de peso, a impressora vende-se num kit que depois será montado em casa pelo utilizador e custa 1.194 dólares, segundo informação na loja do site.

Usa recargas de plástico PLA (embora também suporte plástico ABS), que custam a partir de 31,5 dólares cada uma, havendo várias cores disponíveis. As especificações técnicas e as características são explicadas a partir de um wiki.

Mas há modelos mais recentes. No mercado norte-americano está prestes a ser lançada outra impressora 3D destinada ao mercado doméstico, com preço ligeiramente superior (1.299 dólares), que contudo poupa os seus compradores do trabalho de montagem ( e apresenta uma imagem mais "aprumada").

A Cubify, que vai ser colocada à venda a partir de 25 de maio, permite imprimir objetos em 3D com cerca de 14 cm de diâmetro, de diversos formatos e cores.

A impressora tanto pode usar projetos em 3D criados pelo utilizador como ficheiros disponíveis na Internet, tendo ligação Wi-Fi. Na "caixa", além da máquina (prateada), vem um pequeno cartucho de plástico (verde neon), e o direito a fazer download de 25 criações. As recargas de plástico, disponíveis num conjunto alargado de cores, custam 50 dólares cada uma quando compradas individualmente, 139 dólares um pacote de três e 219 dólares um pacote de cinco.

Mas há pouco mais de um mês foi lançado um modelo que pode vir realmente a impulsionar o mercado doméstico das impressoras 3D, dada a diferença de preço relativamente aos seus rivais.
A responsabilidade é da Soliddodle que acaba de estrear a sua segunda geração de impressoras 3D "acessíveis e fáceis de utilizar", segundo a frase de promoção usada pela marca.

A impressora tem um preço base de 499 dólares, sendo que existem mais duas versões, o modelo Pro, que custa mais 50 dólares, e o modelo Expert, 100 dólares mais caro, com pequenas diferenças entre elas. Como matéria-prima, o equipamento aconselha a utilização de recargas de plástico ABS. A máquina já vem montada e é capaz de moldar itens com até 15cm de diâmetro. Veja o vídeo pesquisando em www.youtube.com por “Meet Solidoodle”.

E depois de quebrada a barreira (abaixo) dos 500 dólares, é muito provável que surjam mais modelos próximos deste valor, contribuindo para que o formato ganhe popularidade.

*Escrito ao abrigo do novo Acordo Ortográfico*
Patrícia Calé

Fonte: Sapo Tek

# TEMA DA CAPA

Desenvolvimento de Aplicações Web Ricas (RIA) com Ext JS 4 e Rails 3

# Desenvolvimento de Aplicações Web Ricas (RIA) com Ext JS 4 e Rails 3

**Introdução**

O Ext JS é um *framework* escrito com a linguagem de scripts JavaScript focada para o desenvolvimento de RIA (*Rich Internet Application*) ou Aplicações de Internet com interfaces Ricas.

Originalmente foi criado por Jack Slocum e começou como uma extensão do Yahoo! User Interface. Teve o apoio da comunidade no desenvolvimento e na versão 2.0 começou a ter duas licenças: a paga e a *open source*.

A licença *open source* é distribuída através da licença GPLv3 (*General Public Licence*) ou também conhecida como licença GNU, ou seja, a licença *open source* é válida apenas se a sua aplicação também for licenciada sobre os termos da GNU.

Atualmente o *framework* está na versão 4.1 e é distribuído pela Sencha Inc, uma companhia localizada na Califórnia, Estados Unidos. Segundo dados da própria Sencha, a comunidade de desenvolvedores ao redor do mundo do Ext JS é superior a um milhão de pessoas!

O Ext JS é um *framework* "cross-browser" (roda até no Internet Explorer 6!) exclusivo para o desenvolvimento do *client-side* (lado cliente), ou seja, o lado servidor (*server-side*) pode ser desenvolvido em qualquer linguagem de programação para a internet. Isso inclui por exemplo, PHP, Python, Java, ASP.Net, etc. Para o desenvolvimento neste artigo, irei optar pelo *framework* de desenvolvimento web Ruby on Rails 3. No entanto, se desejar, você pode portar o código do lado do servidor que utilizaremos para a sua linguagem preferida livremente, já que o foco é introduzirmos ao Ext JS 4.

**Instalação do Ruby e do Rails**

Na plataforma Windows, a forma mais simples de instalar o Ruby, é baixar o pacote *one-click installer*, disponível no link: http://rubyforge.org/frs/download.php/72075/rubyinstaller-1.9.1-p430.exe. A instalação é simplesmente o "famoso" *Next, Next, Finish*.

Se você estiver utilizando o Linux, nas distribuições Debian ou Ubuntu, basta utilizar o gerenciador de pacotes:

```
sudo apt-get install ruby irb rdoc
```

Para instalar o framework Rails no Windows, abra o prompt de comando e digite:

```
gem install rails
```

No Linux, abra o terminal e informe:

```
sudo gem install rails
```

Para certificar-se que tudo está rodando corretamente, ainda no prompt de comando ou no terminal, digite:

```
rails -v
```

Será exibida a versão instalada do Rails. Dessa forma, seja bem vindo ao Rails!

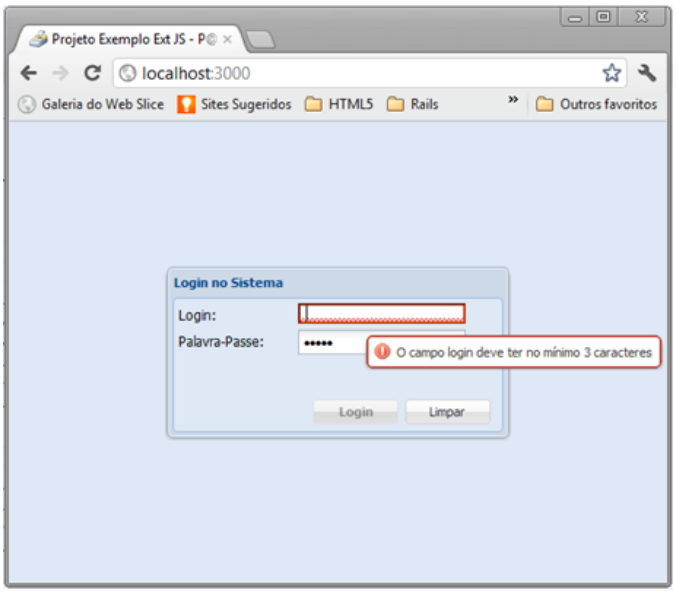
**Download do Ext JS**

O download do Ext JS 4 deve ser efetuado a partir do endereço http://cdn.sencha.io/ext-4.1.0-gpl.zip

Observe que será salvo um arquivo compactado chamado ext-4.1.0-gpl.zip, que é a versão que utilizaremos para a confecção do nosso exemplo.

**Interface de Login**

A aplicação que será desenvolvida neste artigo, será uma interface de login, conforme imagem a seguir:




Nela, entre as diversas configurações, iremos definir por exemplo, a obrigatoriedade dos atributos, o tamanho mínimo de cada e habilitaremos o botão login somente quando os dois campos forem corretamente informados.

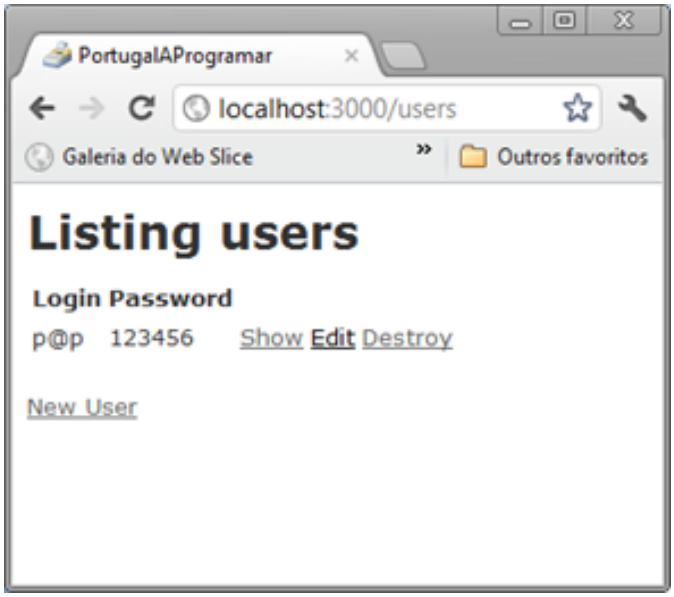
Abra o seu terminal ou prompt de comando, crie uma pasta para armazenar o nosso projeto, (eu sugiro projetos) e digite:

```
rails new portugal-a-programar
```

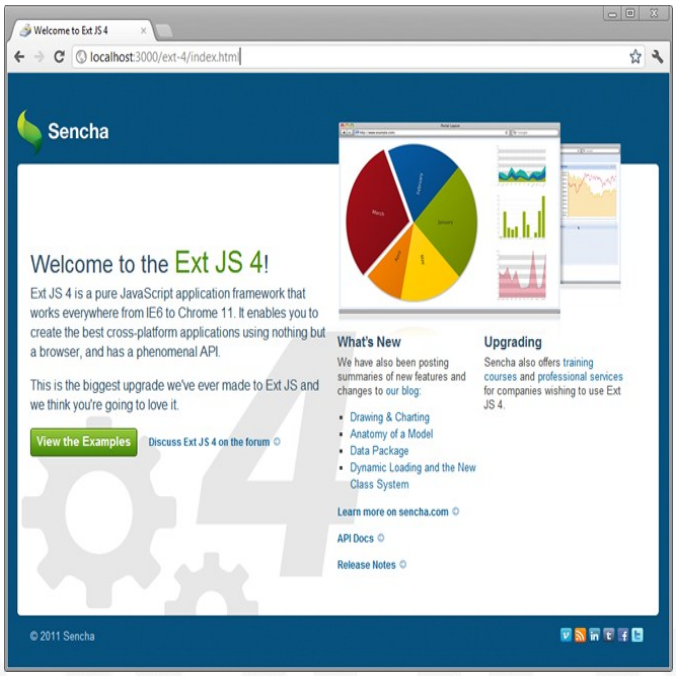
```
cd portugal-a-programar

rails g scaffold user login:string
                    password:string

rake db:migrate

rails s
```

Abra o seu navegador preferido e aponte para:

http://localhost:3000/users. Observe que em apenas cinco linhas, já criamos um pequeno registo de utilizadores com o CRUD (*Create, Read, Update e Delete*) completo!




Agora que o nosso registo de utilizadores já está correndo, vamos criar a interface de login no Ext JS. Para isso, descompacte o arquivo *ext-4.1.0-gpl.zip* no diretório *portugal-a-programar\public\ext-4*.




Aponte o navegador para http://localhost:3000/ext-4/index.html e terá acesso à tela inicial do Ext JS, com acesso a ótima documentação e exemplos.

O primeiro passo será substituir todo o código da página *index.html* do nosso projeto dentro do diretório *portugal-a-programar\public*, pelo código a seguir:

```
<html>
 <head>
  <meta http-equiv="Content-Type"
        content="text/html; charset=utf-8" />
  <title>Projeto Exemplo Ext JS - P@P </title>
  <link rel="stylesheet" type="text/css"
        href="ext-4/resources/css/ext-all.css">
  <style type="text/css">
    body {background-color: #DFE8F6;}
  </style>
  <script type="text/javascript"
          src ="ext-4/bootstrap.js"></script>
  <script type="text/javascript"
          src ="login.js"></script>
 </head>
 <body>
 </body>
</html>
```

Na criação dessa página HTML, observe que estamos a efetuar uma chamada à folha de estilos CSS padrão do framework chamada *ext-all.css*, estamos também a chamar o *framework* através do arquivo bootstrap.js e na linha relativa ao *login.js* iremos chamar o arquivo JavaScript que será criado por nós.

O código do arquivo login.js que deverá ser salvo em *portugal-a-programar\public* está reproduzido a seguir:

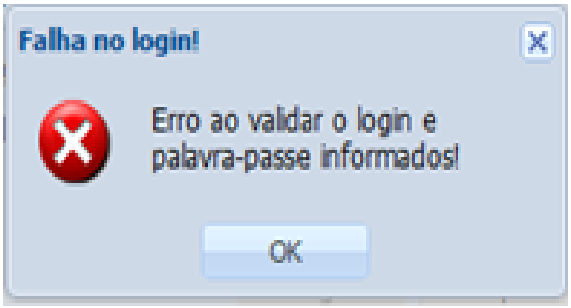
```
Ext.onReady(function () {
 Ext.QuickTips.init();
  function doLogin()
   {
    if (camposlogin.getForm().isValid())
     {
      camposlogin.getForm().submit(
       {
        method: 'POST',
        waitTitle: 'Aguarde enquanto o login é
                                  efetuado...',
        waitMsg: 'Verificando informações...',
        success: function ()
         {
          camposlogin.getForm().reset();
          var redirect = "principal.html"
          window.location = redirect;
         },
          failure: function (form, action)
          {
           if (action.failureType == 'server')
           {
            obj = Ext.decode
                  (action.response.responseText);
            Ext.Msg.show(
             {
```

```
   title: 'Falha no login!',
   msg: obj.errors.reason,
   buttons: Ext.Msg.OK,
   icon: Ext.MessageBox.ERROR,
   scope: this,
          width: 150
   });
  } else {
  Ext.Msg.alert('Atenção', 'Não foi possível
                validar o login: ' +
                action.response.responseText);
  }
  camposlogin.getForm().reset();
 }
 });
}
};
var camposlogin = Ext.create('Ext.form.Panel',
{
 labelWidth: 50,
 url: '/login',
 frame: true,
 defaultType: 'textfield',
 monitorValid: true,
 items: [
  {
   fieldLabel: 'Login',
   name: 'login',
   emptyText: 'Informe o seu login',
   blankText: 'Informe o seu login',
   minLength: 3,
   minLengthText: 'O campo login deve ter no
                        mínimo 3 caracteres',
   inputType: 'textfield',
   allowBlank: false,
   anchor: '90%'
   }, {
      fieldLabel: 'Palavra-Passe',
      name: 'password',
      blankText: 'Informe a sua palavra-passe',
      inputType: 'password',
      allowBlank: false,
      anchor: '90%'
       }
      ],
      buttons: [
   {
      text: '<b>Login</b>',
      formBind: true,
      handler: doLogin
      }, {
       text: 'Limpar',
       scope: this,
       handler: function () { camposlogin.getForm
                                    ().reset(); }
      }
                ]
    });

 var win = Ext.create('Ext.window.Window', {
        layout: 'fit',
        title: 'Login no Sistema',
        width: 300,
        height: 150,
        closable: false,
        resizable: false,
        draggable: false,
        plain: true,
        border: false,
        items: [camposlogin]
    });
    win.show();
});
```

Na criação da janela de login, possuímos algumas opções de como se ela pode ser fechada (closable: false), se pode ser redimensionada (resizable: false), ou ainda se pode ser arrastada. (draggable: false)

Observe também que estamos verificando se o formulário foi validado, ou seja se os campos **login** e **palavra-passe** foram informados. No campo login iremos obrigar também o tamanho mínimo de 3 caracteres através da propriedade minLength. Caso o formulário esteja validado, (isValid()) encaminharemos ao servidor a requisição via o método POST do protocolo HTTP para uma url chamada /login. O servidor web retornará um arquivo no formato JSON (*JavaScript Object Notation*) com a propriedade :success como true ou false.

Caso o utilizador informado seja válido (“success”:true), a aplicação tentará carregar uma página chamada principal.html, onde poderia estar, por exemplo, o menu principal da nossa aplicação. Caso o utilizador não exista na base de dados (“success”:false), será exibida uma mensagem de erro, conforme imagem a seguir:




Para que o Rails reconheça a rota /login, necessitaremos editar o arquivo portugal-a-programar\config\routes.rb, inserindo a linha a seguir, logo após a linha PortugalAProgramar::Application.routes.draw do:

match 'login' => "users#login"

Nessa linha, estamos a redirecionar a aplicação para o método login do controller **users**. (O Rails utiliza o padrão de desenvolvimento MVC-*Model View Controller*).

Para implementar o método login no controller User, edite o arquivo portugal-a-programar\app\controllers\users_controller.rb, adicionando o

método login, logo após a definição da Classe, conforme o código a seguir:

```
class UsersController < ApplicationController
def login
    if   params[:login].nil?   ||   params
[:password].nil?
       render  :json  =>  {  :success  =>  false,
:errors =>
          {:reason => "Informe o login e a pala-
vra-passe do utilizador!" }}
     else
      dados = User.find(:first,:conditions =>
           [" login = ? and password = ?",
           params[:login],params[:password]])
     if dados
        #definimos um cookie
        cookies[:utilizador_id] = dados.id
       render :json => { :success => true}
     else
       render :json => { :success => false,
      :errors => {:reason =>
      "Erro  ao  validar  o  login  e  palavra-passe
informados!" }}
     end
   end
end
#mantenha o código já existente...
```

O método login basicamente irá receber dois parâmetros chamados login e password que foram definidos através do name no arquivo login.js. Caso um dos parâmetros esteja vazio (nil), será retornado um JSON com success:false e a respectiva mensagem de erro.

Se os dois parâmetros forem passados, iremos verificar se o utilizador existe no banco de dados através do método find na classe User e armazenaremos em um array chamado dados, armazenando também um cookie com o ID do utilizador logado, retornando logo em seguida a propriedade true para a chave success.

Para realizar os testes, aponte o seu navegador para http://localhost:3000/users, cadastre um ou mais utilizadores e em seguida aponte o navegador para http://localhost:3000 e informe um login e palavra-chave válidos e um login e palavra-chave inválidos.

Caso o utilizador exista na base de dados, a aplicação emitirá um erro, informando que não foi possível localizar o arquivo principal.html, no qual poderemos abordar a criação de um menu num próximo artigo.

Note que por questões didáticas não foi implementado o método de criptografia da palavra-chave.

**Conclusão**

O *framework client-side* Ext JS 4 integrado ao *framework server-side* Ruby on Rails mostram uma alternativa bastante produtiva para o desenvolvimento de aplicações ricas. Validações como campos obrigatórios, tamanho mínimo do campo ou se a janela pode ser fechada ou arrastada são facilmente configuradas através de propriedades no código JavaScript.

Para saber o que cada propriedade faz, sugiro consultar a ótima documentação disponível em http://docs.sencha.com/ext-js/4-1/

**Bibliografia**

GRONER, Loiane. Ext JS 4 First Look. Birmingham, Uk: Packt Publishing, 2012. 340 p.

LEME, Ricardo. Desenvolvendo aplicações web com Ruby on Rails 2.3 e Postgresql. Rio de Janeiro: Brasport, 2009. 198 p.

## AUTOR

**Escrito por Ricardo Roberto Leme**

É professor universitário e líder do grupo de pesquisa para Desenvolvimento de Aplicações Web Ricas do Núcleo de Informática Aplicada da FATEC - Faculdade de Tecnologia de Itu, localizada no interior do estado de São Paulo, Brasil.

# A PROGRAMAR

Gerenciamento de Horas

Arduino e o cálculo da FFT

Pascal – Registos variantes

O ecosistema Umbraco (4.7.x)

# Gerenciamento de Horas

**Introdução**

Uma das necessidades computacionais é a utilização de cálculos relacionados ao gerenciamento de horas (relógio). Algumas aplicações como folhas de cálculo e ambientes integrados de programação oferecem funções para este tipo de operação. No entanto, nem sempre essa funcionalidade atende as necessidades. Este artigo tem por objetivo apresentar os algoritmos para manipulação de adições e subtrações com horas.

**A Métrica do Tempo**

O conceito de tempo pode ser considerado como a sucessão de anos, de dias ou de horas que envolvem, para o ser humano, a noção do presente, do passado e do futuro. Um dos instrumentos mais importantes para se medir o tempo, criado pelo ser humano, é o relógio. Este instrumento em média apresenta o tempo dividido em três componentes básicos: horas, minutos e segundos.

O componente dos segundos e também dos minutos fazem a medida do tempo fundamentando-se na aplicação da base numérica 60 (valores de 0 a 59). A base numérica 60 faz a contagem de 0 a 59 de 1 em 1 e quando é feita a soma de 1 ao valor 59 este se torna zero e se faz o transporte do valor 1 para à esquerda sobre os minutos se estiver em operação os segundos ou sobre as horas se estiver em operação os minutos.

O componente horas faz a medida do tempo fundamentando -se na aplicação da base numérica 24 (valores de 0 a 23). Quando é feita a soma de 1 sobre o valor 23 este valor se torna zero e inicia-se nova contagem de tempo pelo relógio. No entanto, se quiser fazer a contagem de dias basta fazer o transporte de 1 para a esquerda.

A partir do exposto, baseando-se num relógio pode-se considerar que as horas 23:59:59 acrescida de 1 segundo se tornará 00:00:00 e isto indica que um dia foi transcorrido.

**Algoritmo**

Os algoritmos a seguir podem ser utilizados nas operações de acréscimo e decréscimo de tempo baseado na estrutura operacional de um relógio.

O tratamento de cálculo de horas é efetuado separando-se os componentes de tempo. Assim sendo, é necessário a partir do formato HH:MM:SS (HH = horas, MM = minutos e SS = segundos) fazer a separação desses componentes para que cada um seja tratado separadamente iniciando-se no componente SS, seguido do componente MM e por fim o componente HH.

Para proceder com as operações de soma e subtração de horas é necessário converter as horas e minutos em segundos e os segundos ficam sem nenhum tratamento. Para tanto, siga as seguintes instruções:

Reservar o valor dos segundos:

```
RSG ← SS
```

Multiplicar o valor dos minutos por 60:

```
VMT ← MM*60
```

Multiplicar o valor das horas por 3600:

```
VHR ← HH*3600
```

Somar os valores dos segundos, minutos e horas:

```
SVS ← VHR+VMT+RSG
```

Para voltar o valor calculado da soma ou subtração ao formato horas, minutos e segundos siga os seguintes algoritmos:

```
HH ← INT(SVS/3.600)
MM ← INT((SVS/3.600-HH)*60)
SS ← SVS-HH*3.600-MM*60
```

No sentido de exemplificar a utilização desse algoritmo observe o quadro seguinte:

| | HH | MM | SS |
|---|---|---|---|
| **Horário 1** | 02 | 15 | 44 |
| **Horário 2** | 23 | 18 | 56 |

Considere os horários de entrada (Horário 1) 02:15:44 e saída (Horário 2) 23:18:56 a serem subtraídos para saber quanto é o intervalo de tempo decorrido entre os dois horários.

**Horário 1 - 02:15:44**

```
RSG ←4 4          =    44
VMT ←15*60    =   900
VHR ←02*3.600  = 7.200
SVS ←            8.144
```

**Horário 2 - 23:18:56**

```
RSG ← 44          =     56
VMT ← 18*60       =     1.080
VHR ← 23*3.600    =     82.800
SVS ←                   83.936
```

A partir da conversão em segundos basta subtrair o valor 83.936 de 8.144 que resultará no valor 75.792 (SVS) que será convertido em horas, minutos e segundos, apresentando o valor da diferença do tempo decorrido.

Assim que tiver o valor SVS basta convertê-lo como se segue:

```
HH ← INT(75.792/3.600) = 21
MM ← INT((75.792/3.600-21)*60) = 03
SS ← 75.792-21*3.600-03*60  = 12
```

A diferença de tempo decorrido entre as duas datas é de 21:03:12.

## AUTOR

**Escrito por Augusto Manzano**
Natural da Cidade de São Paulo, tem experiência em ensino e desenvolvimento de programação de software desde 1 986. É professor da rede federal de ensino no Brasil, no Instituto Federal de Educação, Ciência e Tecnologia. É também autor, possuindo na sua carreira várias obras publicadas na área da computação.

# Arduino e o cálculo da FFT

**Introdução**

É objectivo deste artigo mostrar quais são as capacidades e limitações do uso da plataforma de desenvolvimento Arduino, num caso concreto, o cálculo da FFT (Fast Fourier Transform) para diferentes frequências do sinal de entrada. Para tal vão ser efectuadas algumas medições de carácter prático utilizando o modelo Arduino Duemilinove.

Como segundo objectivo, e não menos importante, espera-se fazer uma comparação entre os microcontroladores utilizados nos modelos Arduino Mega com ATmega1280, Arduino Duemilinove com ATmega168 e Arduino Duemilinove com ATmega328. Tentando assim dar uma perspectiva das suas capacidades e limitações neste campo.

**Variáveis vs Memória disponível**

Como primeiro passo para o início de uma análise pormenorizada do tema, torna-se necessário ter em consideração a quantidade de memória SRAM (Static Random Access Memory) disponível nos diversos modelos de microcontroladores utilizados.

| | ATmega168 | ATmega328 | ATmega1280 |
|---|---|---|---|
| **Flash** | 16 KBytes (2 KBytes Bootloader) | 32 KBytes (2 KBytes Bootloader) | 128 KBytes (4 KBytes Bootloader) |
| **SRAM** | 1024 Bytes | 2048 Bytes | 8192 Bytes |
| **EEPROM** | 512 Bytes | 1024 Bytes | 4096 Bytes |

Tabela 1 – Quantidades de memória disponíveis

O *Arduino Duemilinove* está equipado com o ATMega168, podendo ser "actualizado" facilmente substituindo o microcontrolador actual (ATmega168), por um ATMega328 pré-programado (com o respectivo Bootloader).

Fazendo uma breve análise à tabela 1 (valores retirados do datasheet de cada microcontrolador), é possível verificar que a versão Arduino Mega com o ATmega1280 leva clara "vantagem" em termos de capacidade, mas resta analisar até que ponto nesta aplicação concreta será suficiente essa "vantagem".

Ao ser declarada (utilizada) uma variável do tipo float estão a ser reservados 4 bytes (32 bits) na memória SRAM. Se considerarmos que ao efectuar a FFT vamos obter resultados com parte real e imaginária, vamos ter de reservar dois "espaços" de um array (vector)do tipo float para cada número. O que nos leva a que ao ser declarado p.ex. um array de f[i] com i=1 a N com N=128 , teremos um array que corresponderá a 64 números do tipo "a+jb)". Em que p.ex. corresponderá à parte real ("a"), e f[2] corresponderá a ("b") do primeiro número. Esta conclusão vai ser importante nos cálculos efectuados, para achar a capacidade máxima de armazenamento em SRAM dos microcontroladores referidos na Tabela 1.

Numa situação ideal, em que mais nenhuma variável necessitaria de ser usada, a capacidade de armazenamento (para este caso concreto) seria a seguinte:

- Arduino Duemilinove (ATMega168)

$$ATMega168 = \frac{1024}{8} = 128$$

- Arduino Duemilinove (ATMega328)

$$ATMega328 = \frac{2048}{8} = 256$$

- Arduino Mega (ATMega1280)

$$ATMega1280 = \frac{8192}{8} = 1024$$

Pela análise dos valores acima obtidos, podemos constatar que poderíamos armazenar 128 números do tipo *"a+jb"* num ATMega168, 256 num ATMega328 e 1000 recorrendo ao uso de um ATMega1280.

Mas como é óbvio é necessário recorrer a variáveis para o próprio cálculo da FFT, bem como a variáveis "intermédias" para obter e guardar os valores obtidos nas diferentes entradas analógicas presentes no Arduino (qualquer que seja a versão a considerar). O que nos leva desde já a tomar os valores acima descritos como um exercício meramente teórico.

Após uma abordagem sobre os valores teóricos de armazenamento em SRAM, dos diversos modelos de microcontrolador utilizado na plataforma de desenvolvimento Arduino, vai ser tomado em consideração o tempo de conversão A/D levado pelo ADC (Analog-to-digital-converter ou conversor analógico-digital). Vai também ser tomado em consideração o tempo necessário para a aplicação do algoritmo de cálculo da FFT utilizado.

**Arduino e o ADC**

Para esta aplicação específica torna-se necessária uma análise do tempo de conversão A/D, com vista a saber qual o tempo que se consegue obter entre amostras. Para tal foram lidos os valores de tempo de conversão obtidos experimentalmente, modificando os valores do factor de divisão. Este factor de divisão (Prescaler) define o input clock do conversor A/D, ou seja, se tivermos um factor de divisão de 32 e um clock de sistema de 16 MHz (comum a todos os modelos de Arduino) teremos um input clock no conversor A/D de:

$$Input_{A/D} = \frac{16}{32} = 0.5\ MHz$$

Cada conversão A/D (de acordo com o especificado no datasheet do microcontrolador) demora cerca de 13 ciclos de clock, à excepção da primeira conversão que demorará 25 ciclos de clock (efectua a inicialização do conversor A/D). Ao considerar o valor obtido no cálculo efectuado acima, iríamos obter uma taxa de amostragem de:

$$Taxa\ de\ amostragem = \frac{0.5 * 10^6}{13} \cong 38\ KHz$$

O que não corresponderá aos valores obtidos experimentalmente, correspondendo o valor acima a um valor teoricamente esperado. Devido a este facto vão ser efectuadas medições experimentais, tentando assim obter estimativas mais reais dos tempos necessários para efectuar a conversão A/D.

Para obter os valores de tempo de conversão, execução da instrução "analogRead", vai ser utilizada a função "micros()". Esta função retorna o valor de tempo, em *µs*, que passou desde que se iniciou a execução do programa. Ao fazer a diferença entre os valores de retorno da função, em sítios distintos da execução do nosso programa, conseguimos obter o tempo dispendido na sua execução. O código a utilizar será o seguinte:

```
#ifndef cbi
#define cbi(sfr, bit) (_SFR_BYTE(sfr) &= ~_BV(bit))
#endif
#ifndef sbi
#define sbi(sfr, bit) (_SFR_BYTE(sfr) |= _BV(bit))
#endif
void setup(void)
{
      Serial.begin(9600); ); //Inicializa o envio
//por Série, e define a Baud Rate
      sbi(ADCSRA, ADPS2); //Neste caso concreto o
//bit ADPS2 será definido com o valor 1
      cbi(ADCSRA, ADPS1); //Neste caso concreto o
//bit ADPS1 será definido com o valor 0
      cbi(ADCSRA, ADPS0); //Neste caso concreto o
//bit ADPS0 será definido com o valor 0
}
int a,b,c;
void loop()
{
      b = micros();); //Atribui à variável "b" o
//tempo actual de execução do programa actual
      a = analogRead(0);
      c = micros(); // Atribui à variável "c" o
//tempo actual de execução do programa actual
      Serial.print(c-b); //Envia (série) o valor
//necessário para executar a instrução
//analogRead" (conversão A/D)
}
```

Ao fazer variar o conteúdo do registo ADCSRA, mais precisamente dos bits designados por ADPS2, ADPS1 e ADPS0, é possível fazer variar o valor do factor de divisão referido anteriormente. As combinações possíveis encontram-se descritas na seguinte tabela:

| ADPS2 | ADPS1 | ADPS0 | Factor de divisão obtido |
|---|---|---|---|
| 0 | 0 | 0 | 2 |
| 0 | 0 | 1 | 2 |
| 0 | 1 | 0 | 4 |
| 0 | 1 | 1 | 8 |
| 1 | 0 | 0 | 16 |
| 1 | 0 | 1 | 32 |
| 1 | 1 | 0 | 64 |
| 1 | 1 | 1 | 128 |

Tabela 2 – Combinações possíveis (Registo ADCSRA)

Os valores obtidos, utilizando um Arduino Duemilinove com ATMega168 (modelo testado), foram os seguintes:

| Factor de divisão | Conversão A/D (*µs*) |
|---|---|
| 128 | ≅ 116 |
| 64 | ≅ 64 |
| 32 | ≅ 32 |
| 16 | ≅ 20 |
| 8 | ≅ 16 |
| 4 | ≅ 12 |
| 2 | ≅ 8 |

Tabela 3 – Tempo de conversão A/D obtido

Os valores obtidos na Tabela 3 correspondem a uma média de 1000 valores obtidos para cada caso (Recolha e cálculo da média efectuado recorrendo ao Software MatLab), ou seja, aos valores de tempo obtidos ao fazer variar o respectivo factor de divisão.

## ARDUINO E O CÁLCULO DA FFT

Outra conclusão a retirar das medições efectuadas, e que foi possível verificar através da experiência, é o facto de apenas se deverem considerar valores do factor de divisão (Prescaler) acima de 2. Verificando-se que para factores de divisão de valor 2 são obtidos alguns erros na conversão A/D (Obtiveram-se alguns valores de leitura nulos).

Pela análise da Tabela 3, e pelo descrito acima, é óbvio que o tempo de conversão mínimo que se pode obter será de ≅12$\mu s$. Este valor é apenas referente a uma entrada analógica, se quisermos considerar as 6 entradas analógicas disponíveis teremos de multiplicar esse valor por 6. Obtendo assim:

$$T_{conversão\ A/D\ \ com\ seis\ entradas} \cong 12 * 6 \cong 72\mu s$$

Podendo assim considerar que obtemos uma "conversão muito mais rápida" ao efectuar a conversão A/D de cada canal individualmente.

Os valores da Tabela 3 podem ser descritos em número de amostras por segundo (taxa de amostragem). Sendo os valores obtidos descritos na tabela seguinte:

| Factor de divisão | Taxa de amostragem (*KHz*) |
|---|---|
| 128 | ≅ 8 |
| 64 | ≅ 16 |
| 32 | ≅ 31 |
| 16 | ≅ 50 |
| 8 | ≅ 63 |
| 4 | ≅ 83 |
| 2 | ≅ 125 |

A taxa de amostragem que se consegue obter tem influência directa, segundo o teorema de Nyquist, na frequência máxima do sinal de entrada que se consegue reconstruir sem perda de informação. Ou seja, se conseguirmos uma taxa de amostragem de 125KHz, a frequência máxima (sinal de entrada) que poderíamos reconstruir estaria situada nos 62.5KHz.

**Arduino e o Cálculo da FFT**

O algoritmo utilizado não foi "feito de raiz", mas sim adaptado de um já existente. Tentando assim neste artigo, apenas encontrar e adaptar uma função existente na linguagem de programação C que permitisse efectuar o cálculo da FFT. A adaptação consistiu em alterar certas instruções utilizadas pelo algoritmo original, que recorria a bibliotecas próprias da linguagem C, para instruções equivalentes (que permitissem obter o mesmo resultado) existentes actualmente nas bibliotecas disponíveis para o Arduino.

A principal dificuldade prendeu-se com a utilização, por parte dos algoritmos existentes em C, de bibliotecas não existentes no ambiente de desenvolvimento do Arduino. Tornando a adaptação de certos algoritmos encontrados uma tarefa morosa e algo complexa.

O código utilizado neste estudo baseia-se no algoritmo redescoberto por "Danielson and Lanczos" em 1942 (Press, Teukolsky et al. 1992). O array de entrada (do tipo float) deverá ter uma dimensão correspondente a uma potência de dois, ou seja, neste caso concreto analisado f[i] deverá estar compreendido entre 0 e $2^n$ - 1 ( n ϵ N). O que limita a utilização da memória SRAM disponível a vectores que possuam essas dimensões. Num caso concreto em que se queira aplicar o algoritmo a um conjunto de números do tipo "a+jb", que não seja exactamente uma potência de dois, deverá ser reservado espaço de memória para um array f[i] com um valor máximo de i que corresponda ao valor da potência de dois imediatamente acima do número pretendido menos um ($2^n$–1). Ao ser efectuada a passagem do array para a função de cálculo da FFT deverá ser efectuado Zero padding (Chu 2008), ou seja, todos os restantes valores do array deverão ser preenchidos com o valor zero.

Ao utilizar a versão Arduino Duemilinove com ATmega168, e como foi referido anteriormente, podemos armazenar na memória SRAM disponível um array (do tipo float) com a dimensão máxima de 128. Mas devido às razões enumeradas no início deste artigo o número de pontos máximo que se consegue calcular actualmente será de 64.

Tornando-se agora possível, e recorrendo a um raciocínio semelhante ao utilizado no cálculo experimental da conversão A/D, calcular qual o tempo necessário para executar o algoritmo redescoberto por "Danielson and Lanczos". Estimando assim o tempo total levado pelo Arduino a adquirir e a aplicar a função de cálculo da FFT.

Os tempos de cálculo obtidos, utilizando o Arduino Duemilinove com ATMega168, foram os seguintes:

| Factor de divisão | Número de pontos utilizados (FFT) | Conversão A/D ($\mu s$) | Cálculo da FFT ($\mu s$) | Tempo total ($\mu s$) |
|---|---|---|---|---|
| 128 | 64 | ≅ 7424 | ≅ 21944 | ≅ 29368 |
| 128 | 32 | ≅ 3712 | ≅ 10236 | ≅ 13948 |
| 128 | 16 | ≅ 1856 | ≅ 4880 | ≅ 6736 |
| 64 | 64 | ≅ 4096 | ≅ 21944 | ≅ 26040 |
| 64 | 32 | ≅ 2048 | ≅ 10236 | ≅ 12284 |
| 64 | 16 | ≅ 1024 | ≅ 4880 | ≅ 5904 |
| 32 | 64 | ≅ 2048 | ≅ 21944 | ≅ 23992 |
| 32 | 32 | ≅ 1024 | ≅ 10236 | ≅ 11260 |
| 32 | 16 | ≅ 512 | ≅ 4880 | ≅ 5392 |
| 16 | 64 | ≅ 1280 | ≅ 21944 | ≅ 23224 |
| 16 | 32 | ≅ 640 | ≅ 10236 | ≅ 10876 |
| 16 | 16 | ≅ 320 | ≅ 4880 | ≅ 5200 |
| 8 | 64 | ≅ 1024 | ≅ 21944 | ≅ 22968 |
| 8 | 32 | ≅ 512 | ≅ 10236 | ≅ 10748 |
| 8 | 16 | ≅ 256 | ≅ 4880 | ≅ 5136 |
| 4 | 64 | ≅ 768 | ≅ 21944 | ≅ 22712 |
| 4 | 32 | ≅ 384 | ≅ 10236 | ≅ 10620 |
| 4 | 16 | ≅ 192 | ≅ 4880 | ≅ 5072 |
| 2 | 64 | ≅ 512 | ≅ 21944 | ≅ 22456 |
| 2 | 32 | ≅ 256 | ≅ 10236 | ≅ 10492 |
| 2 | 16 | ≅ 128 | ≅ 4880 | ≅ 5008 |

Tabela 5 – Tempo total despendido (Cálculo FFT+ADC)

Os valores obtidos na coluna 4 (Cálculo da FFT) da Tabela 5 foram obtidos efectuando a média de 1000 valores obtidos experimentalmente (Recolha e cálculo da média efectuado recorrendo ao Software MatLab). O algoritmo utilizado no cálculo da FFT foi aplicado variando o número de pontos a calcular, mas utilizando o mesmo array f[i] para todos os testes efectuados. (Todos os pontos a calcular foram considerados como sendo *1+j*).

A coluna 3 corresponde ao produto do tempo de conversão obtido na Tabela 3 pelo número de pontos a serem adquiridos, obtendo assim uma estimativa para o tempo de conversão A/D necessário em cada caso.

O algoritmo utilizado poderá ser melhorado e a sua execução poderá ser optimizada, porém esse não será factor mais importante mas sim a velocidade com que se efectua a amostragem do sinal de entrada.

A coluna 5 (Tempo total) corresponde à soma do tempo necessário para efectuar a conversão A/D dos pontos referidos na coluna 2 (Número de pontos utilizados) com o tempo necessário para aplicar o algoritmo de cálculo da FFT a esses pontos. Se considerarmos que temos seis entradas analógicas, os tempos totais necessários serão o produto dos valores obtidos na coluna 5 por um factor de 6. Resultando os seguintes valores finais:

| Factor de divisão | Número de pontos adquiridos por cada entrada analógica (FFT) | Tempo final ($\mu s$) |
|---|---|---|
| 128 | 64 | ≅ 176208 |
| 128 | 32 | ≅ 83688 |
| 128 | 16 | ≅ 40416 |
| 64 | 64 | ≅ 156240 |
| 64 | 32 | ≅ 73704 |
| 64 | 16 | ≅ 35424 |
| 32 | 64 | ≅ 143952 |
| 32 | 32 | ≅ 67560 |
| 32 | 16 | ≅ 32352 |
| 16 | 64 | ≅ 139344 |
| 16 | 32 | ≅ 65256 |
| 16 | 16 | ≅ 31200 |
| 8 | 64 | ≅ 137808 |
| 8 | 32 | ≅ 64488 |
| 8 | 16 | ≅ 30816 |
| 4 | 64 | ≅ 136272 |
| 4 | 32 | ≅ 63720 |
| 4 | 16 | ≅ 30432 |
| 2 | 64 | ≅ 134736 |
| 2 | 32 | ≅ 62952 |
| 2 | 16 | ≅ 30048 |

Tabela 6 – Tempo total despendido para 6 entradas

A Tabela 6 apenas serve como indicação da gama de valores de tempo (em µs ) de execução, necessários para converter o número de pontos indicado na coluna 2 de analógico para digital (por cada entrada analógica) e aplicar o respectivo algoritmo de cálculo da FFT. É de salientar que o valor obtido para o cálculo final (coluna 3 da Tabela 6) utilizando um factor de divisão de 2 e adquirindo 64 pontos (de cada entrada analógica) corresponde apenas a ≅ 0.13s.

**Conclusões**

No modelo testado (Arduino Duemilinove com ATmega168), e numa situação ideal, existiria a possibilidade de efectuar o armazenamento em SRAM de cerca de 128 pontos distintos. O que não se pode comparar quase aos 1000 pontos que se conseguiriam armazenar utilizando o Arduino Mega com ATmega1280. O que nos leva a considerar, e se quisermos continuar a apostar no cálculo da FFT recorrendo directamente ao Arduino (e não ao processamento recorrendo a computador), a considerar a sua aquisição (custo de aquisição ≅45€ por unidade). Recorrendo ao uso do Arduino Mega teríamos a possibilidade de efectuar a aquisição e o cálculo da FFT para 512 pontos. Este valor surge pois $2^{11}$ corresponde a um valor de 2048, ou seja, 1024 números do tipo "*a + jb*" o que excede a capacidade máxima de armazenamento disponível. A solução é considerar a potência de dois imediatamente abaixo $2^{10}$ que corresponde a um valor de 1024, ou seja, a 512 pontos do tipo "a+jb".

Como se sabe, o que irá definir se o desempenho obtido é suficiente será o destino para o qual o nosso sistema é desenvolvido. Em caso de necessidade de maiores capacidades de processamento, não se deve descartar a hipótese do cálculo da FFT recorrendo ao uso de p.ex. um computador (e uso do Arduino apenas como datalogger).

Caso se queira melhorar o desempenho do Arduino no cálculo da FFT, será sem dúvida um bom desafio pois "O saber não ocupa espaço de memória".

**Bibliografia**

Chu, E. (2008). Discrete AND Continuous Fourier Transform. Boca Raton, USA, A CHAPMAN & HALL BOOK.

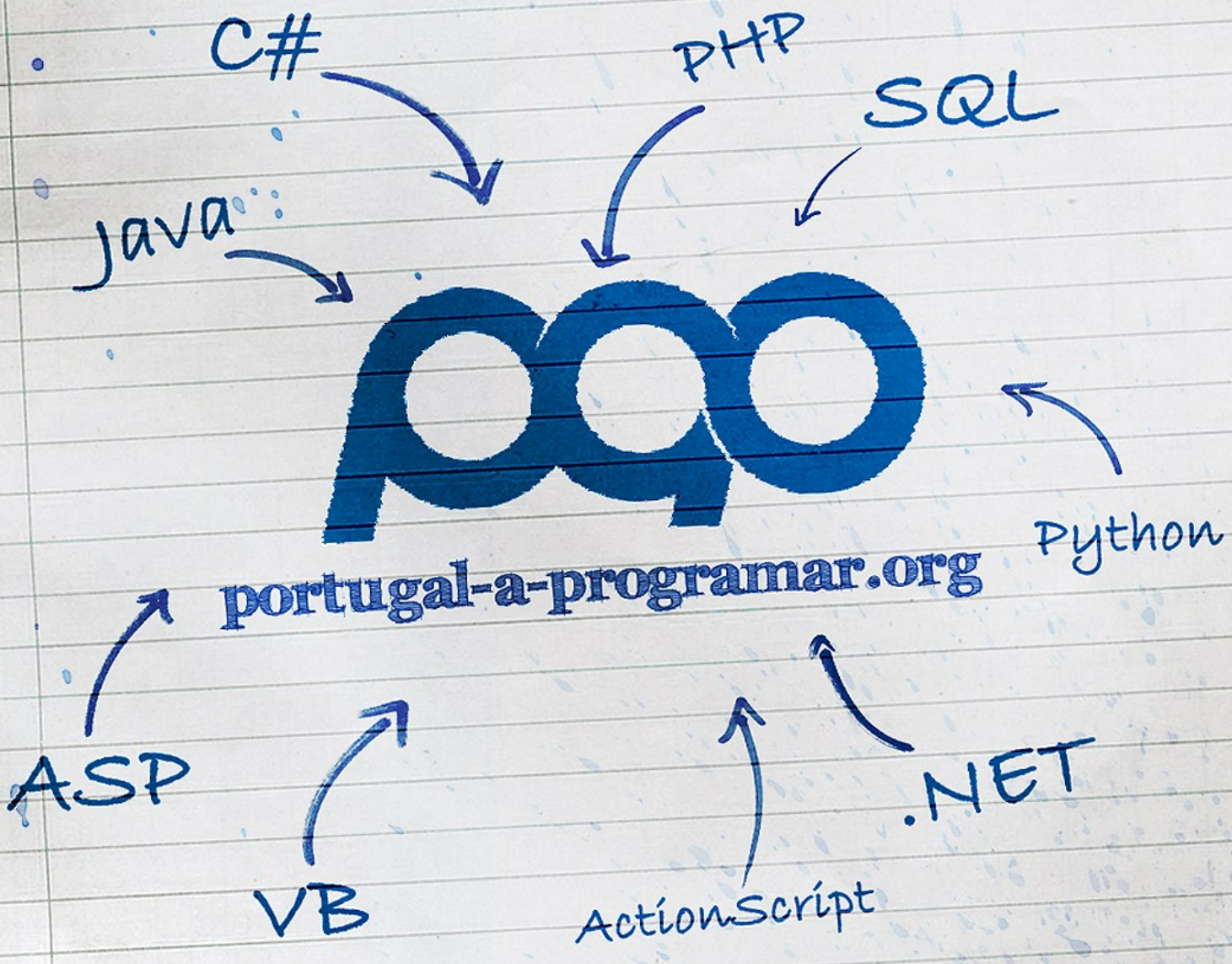
Press, W. H., S. A. Teukolsky, et al. (1992). Numerical Recipes in C. Melborne, Australia, Cambridge University Press.

## AUTOR

**Escrito por Nuno Pessanha Santos**

Mestrado em Engenharia Naval ramo de Armas e Electrónica pela Escola Naval. É um apaixonado pela área da electrónica e das telecomunicações.

# A maior comunidade portuguesa de programação!




# Visite-nos!

Elege o melhor artigo desta edição

Revista PROGRAMAR

http://tiny.cc/ProgramarED35_V

# Pascal – Registos variantes

Os registos são um dos tipos de dados compostos mais úteis em qualquer linguagem de programação que os suporte, como o C ou o Pascal. Permitem-nos criar formulários com vários campos em que cada campo é de um tipo de dado distinto.

E, em conjunto com matrizes (regra geral, as unidimensionais), ficamos com uma autêntica lista de registos, o que nos dá um poder tremendo de manipular informação em poucas linhas de código.

Contudo, há certos registos cujos campos que mudam são poucos, e torna-se algo frustrante criar mais um registo só para ter um campo ou dois diferentes.

Então, o Pascal oferece-nos mais uma poderosa ferramenta para simplificar. Se só alguns campos mudam, para quê criar um registo totalmente novo? No mesmo registo, fazemos campos que alteram consoante determinadas condições. E temos, então, os chamados **registos variantes**.

Assim, os objectivos para o presente artigo são:

- Rever os registos (tipo de dado composto Record);
- Dar a conhecer os registos variantes;
- Comparar os registos estáticos com os variantes e conhecer as suas vantagens;

**Breve revisão dos registos**

Um registo é um tipo de dado composto constituído por vários campos, os quais podem ser, igualmente, outros registos. Permitem juntar uma série de dados relacionados e permitem manipulá-los de uma forma tão simples quanto a manipulação de dados simples.

Um exemplo de um registo é de um pequeno formulário para

# ENIGMAS DO C#: O ESTRANHO CASO DOS ENUMERADOS

por Paulo Morgado

**Enigma**

Dado o seguinte enumerado:

```
enum SomeEnum : int
{
    Zero
}
```

E os seguintes métodos:

```
static void Do(
    string prompt,
    SomeEnum value)
{
    Console.WriteLine(
      "{0,2}: value = {1,4}, type = {2,8}",
      prompt,
      value,
      value.GetType().Name);
}

static void Do(
    string prompt,
    object value)
{
    Console.WriteLine(
      "{0,2}: value = {1,4}, type = {2,8}",
      prompt,
      value,
      value.GetType().Name);
}
```

Qual é o resultado do seguinte código ?

```
Do("1", 0);
Do("2", 0L);
Do("3", 0D);
Do("4", 0f);
Do("5", 0M);

Do("6", (int)0);

Do("7", 5 * (1 - (2 - 1)));

int i = 0; Do("8", i);
long l = 0L; Do("9", l);
double d = 0D; Do("10", d);
float f = 0F; Do("11", f);
decimal m = 0M; Do("12", m);
object o = 0; Do("13", o);
```

**Veja a resposta e explicação na página 48**

registar os dados de uma pessoa:

```
Formulario:REGISTO
  Nome :String;
  Idade:Byte;
  Sexo :Char;
  BI   :String[8];
  EstadoCivil:
   (Solteiro,Casado,Outro);
FIM REGISTO
```

O campo **Nome** pode ser acedido fazendo **Formulario.Nome**, ou seja, refere-se qual o registo a que queremos aceder e com um ponto separamos o nome do registo do seu respectivo campo que pretendemos manipular. Isto em Pascal será o seguinte:

```
TYPE
(* Estado Civil *)
  TEstCiv =
  (Solteiro,Casado,Outro);

(* Registo *)
TFormulario=RECORD
  Nome:String;
  Idade:Byte;
  Sexo:Char;
  BI:String[8];
  EstadoCivil:TEstCiv;
  END;

VAR
  Pessoa:TFormulario;

(* Manipular "Nome" *)
Pessoa.Nome:='Jose';
```

Para facilitar a manipulação dos campos do registo **Pessoa** podemos recorrer ao bloco **with**:

```
with Pessoa do begin
  Nome:='Jose';
  Idade:=63;
  Sexo:='M';
  BI:='12345678';
  EstadoCivil:=Casado;
end;
```

Isto simplifica muito, pois evita escrevermos sempre o nome do registo. Ou seja, este bloco infra-apresentado substitui o seguinte:

```
Pessoa.Nome:='Jose';
Pessoa.Idade:=63;
Pessoa.Sexo:='M';
Pessoa.BI:='12345678';
Pessoa.EstadoCivil:=Casado;
```

**Os registos variantes**

Os registos são, regra geral, de campos fixos. Podemos-lhes chamar de **registos estáticos**. Ou seja, os seus campos são bem definidos e não se alteram em qualquer circunstância. Contudo, torna-se útil variar certos campos. Vejamos um exemplo prático para entender porquê e como.

Necessitamos de registar, ou catalogar, determinados produtos electrónicos, mais especificamente computadores (PC), impressoras e ratos. Cada um destes produtos tem descrições próprias:

- **PC** – CPU (processador) e HDD (disco rígido);
- **Rato** – de bola ou óptico (laser);
- **Impressora** – impressão a laser ou de jacto de tinta.

Contudo, há dados que lhes são comuns:

- Nome (ou descrição, conforme);
- Marca;
- Modelo;
- Preço.

A questão que se coloca é: vale a pena três registos distintos para cada um dos produtos? A resposta é: não. Num só registo vamos variar os campos conforme o tipo de produto. Como?

Um determinado campo estático determinará quais os campos variantes que irão surgir. Por exemplo, ao indicarmos a um campo estático, destinado para o efeito, que estamos a registar uam impressora, surgirá um campo variante que indicará se a impressora é a laser ou a jacto de tinta.

A estrutura que nos vai criar campos variáveis a partir de campos estáticos será a **Case Of.**

```
TYPE
TConjuntoProdutos =
  (PC, Impressora, Rato);

// ...

case TipoProduto :
TConjuntoProdutos of
  PC : ;
  Impressora : ;
  Rato : ;
end;
```

Todavia, ao invés de se implementar qualquer código (acção) como é habitual nesta estrutura, vamos declarar os campos que correspondem a cada tipo de produto, tal como

listado anteriormente:

```
case TipoProduto :
TConjuntoProdutos of
  PC : (CPU:string[30];
        HDD:string[50]);
  Impressora :
    (Tinta:TTipoImpressao);
  Rato : (Optico:Boolean);
end;
```

Note-se então que **TipoProduto** será um **campo estático** do registo, campo este que vai indicar quais são os campos variantes – ou seja, consoante o valor do campo estático **TipoProduto**, os campos variantes mudam.

Neste caso, se **TipoProduto** for **PC**, ou seja, estamos a registar um computador, então são os campos **CPU** e **HDD** que podem ser atribuídos, sendo que os outros ficam "invisíveis". Quando pretendemos registar uma impressora, o campo que ficará disponível será **Tinta** que nos indica se a impressora é a *laser* ou a jacto de tinta. O mesmo se passa para quando queremos registar um rato, onde o campo que é criado indicar-nos-á se é óptico ou dos antigos, de bola.

Consolidando isto, o nosso registo, incluindo os tipos de dados novos necessários, será o seguinte:

```
TYPE
(*TIPOS necessários*)
TConjuntoProdutos =
  (PC, Impressora, Rato);
TTipoImpressao =
  (JactoTinta, Laser);

(*REGISTO de produto*)
TRegProduto=RECORD
  Nome:string[50];
  Preco:Real;
  Marca:string[20];
  Modelo:string[30];

  //Campos variantes:
  case TipoProduto :
  TConjuntoProdutos of
    PC : (CPU:string[30];
          HDD:string[50]);
    Impressora :
      (Tinta:TTipoImpressao);
    Rato : (Optico:Boolean);
END;
```

Vamos, então, esquematizar os campos estáticos e os variantes deste registo:

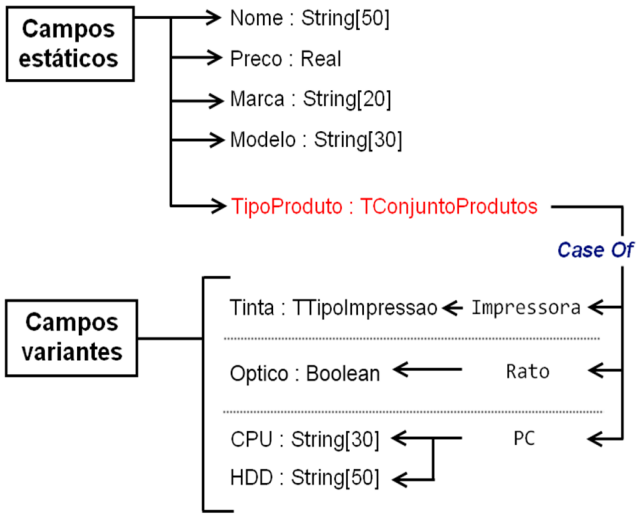



**Figura 1** – Esquematização dos campos do registo TRegProduto.

Assim compreendemos melhor que **TipoProduto** é um **campo estático** que vai influenciar os **campos variantes**.

É de referir que um registo pode ter mais do que um grupo de campos variantes, e que dentro de cada campo variante pode ser criado um outro registo que também contenha campos variantes. As possibilidades são, então, imensas.

### *Input* e *output* de registos variantes

Regra geral, os compiladores não detectam se estão a ser atribuídos valores a um campo que, nesse ponto do programa, não estão "visíveis". O *debug* será a única forma de confirmar se a operação realizada é ilegal naquele ponto ou não. Tenha em atenção quais os campos variantes que estão em voga aquando operações de atribuição nestes.

Portanto, como se deve prevenir e não remediar, convém controlar em código que *input* ou *output* é para ser feito em cada ponto do programa. Mais uma vez, a estrutura de decisão **Case Of** ir-nos-á resolver esta questão.

Seguindo os exemplos infra-apresentados, fazemos o **Case Of** ao campo estático **TipoProduto** e, consoante o valor deste, variamos o *input*/*output* pretendido. Por exemplo:

```
case Reg.TipoProduto of
  Impressora:begin
    repeat
      write('Impressao? ');
      write('T=Tinta; L=Laser ');
      c:=ReadKey;
      c:=UpCase(c);
    until (c in ['T','L']);
    case c of
      'L':Tinta:=Laser;
```

```
      'T':Tinta:=JactoTinta;
    end;
    writeln;
  end;
  // outros casos...
end;
```

Para este trecho de código, no caso do produto (**TipoProduto**) ser uma impressora vamos pedir ao utilizador que nos informe qual o tipo de impressão: se é a *laser* ou é a jacto de tinta. Ou seja, o *input* de dados relativos à impressora só ocorre caso tenhamos definido que o tipo de produto de **Reg** (variável hipotética do tipo **TRegProduto**) é, especificamente, uma impressora. Se for um rato ou um PC outros comandos serão executados.

**Matrizes de registos variantes**

Após toda a teoria, segue o caso da vida real. O que nos dá mais jeito é, claro está, criar uma matriz unidimensional de registos. Por exemplo:

```
RegProdutos:array[1..10]
        of TRegProduto;
```

Temos aqui dez registos de produtos, registos estes que têm campos variantes. O que vai acontecer?

Comecemos no índice 1 da matriz. Estamos a registar um PC, pelo que **TipoProduto** será **PC** e os campos variantes serão **HDD** e **CPU**. Seguindo para o índice 2 definimos uma impressora, e o campo variante será **Tinta**. Já no índice 3 definimos um rato e o campo variante será **Optico**.

Constatamos que, para uma mesma matriz, cada índice tem um conjunto de campos diferente – os campos em comum são os estáticos e aqueles que diferem são, obviamente, os variantes.

Verificamos, portanto, mais um poder dos registos variantes. Uma matriz destes registos vai diferir nos campos em cada índice. Estamos habituados a ver uma matriz de um tipo de dado certo e que, para cada índice, nada muda (a não ser os valores atribuídos, claro). Com os registos variantes isto não acontece.

A seguinte tabela mostra o exemplo de um excerto dos primeiros cinco índices do Array unidimensional **RegProduto**, declarado anteriormente. A tabela só faz referência aos campos variantes e ao campo estático que os varia.

| Índice do Array | TipoProduto | Campos variantes | | | |
|---|---|---|---|---|---|
| | | HDD | CPU | Tinta | Optico |
| 1 | PC | SATA 1TB | i5 2.80GHz | | |
| 2 | Impressora | | | JactoTinta | |
| 3 | Rato | | | | *True* |
| 4 | Rato | | | | *False* |
| 5 | Impressora | | | JactoTinta | |

**Conclusão**

Ficámos a conhecer, por fim, mais uma das capacidades poderosas do Pascal. Para quê criar uma série de registos semelhantes se lhes podemos variar certos campos consoante parâmetros muito específicos?

Os registos ganham, assim, uma nova dimensão aos nossos olhos. Simplificam o nosso trabalho em larga escala e demonstram capacidades que não muitas linguagens têm neste aspecto. E esta é uma capacidade que já vem desde os primórdios da existência do Pascal, ou seja, desde os anos 70.

**Links úteis**

Uma lista de documentos úteis da Wiki P@P, relacionados com o presente artigo.

- Tutorial de Pascal (2011) – http://tinyurl.com/6wjejqo
  - Parte I – Procedimentos e funções – http://tinyurl.com/845gwjo
  - Parte IV – A variável Record – http://tinyurl.com/7q4efd3
  - Parte V – Estruturação de um programa em Pascal – http://tinyurl.com/79lfx4k
- Indentação – http://tinyurl.com/6p3w63y

## AUTOR

**Escrito por Igor Nunes**

Estudante universitário, entrou no mundo da programação aos 14 anos com TI-Basic. Dois anos depois descobriu o Pascal, que ainda hoje é a sua Linguagem de Programação de eleição. Mais recentemente introduziu-se ao VB.NET e ao Delphi.

Membro do P@P desde Abril de 2010 (@thoga31), é actualmente membro da **Wiki Team** e **Moderador Local** dos quadros de Pascal e Delphi/Lazarus. Escreveu o novo Tutorial de Pascal da Wiki P@P, bem como os Tutoriais de TI-Basiz Z80 e de Introdução à Lógica e Algoritmia.

# O ecossistema Umbraco (4.7.x)

Antes de mais, visto este artigo ser relativamente grande em quantidade de conceitos e informação, aqui fica um género de índice sobre o que vai ser abordado:

1. Introdução;
2. Instalação/Como começar;
3. Árvore de documentos;
4. Media;
5. Tipos de documento;
6. Tipos de campo;
7. Templates;
8. Macros;
9. Membros e autenticação;
10. Packages;
11. Umbraco 5 e conclusão.

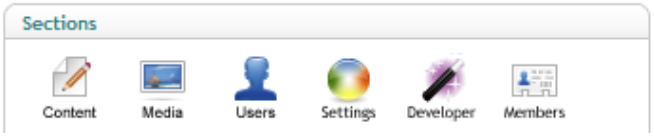
**1. Introdução**

O Umbraco tem por objectivo facilitar o desenvolvimento de *websites* geríveis e flexíveis, que possam dar resposta à mudança de uma forma ágil e que proporcionem o mínimo esforço possível aos gestores de informação. Por outras palavras, ser o *Content Management System* perfeito!

Para possibilitar isto, o Umbraco é estruturado de um modo simples, para minimizar a curva de aprendizagem e possibilitar uma gestão fácil.

Qualquer programador proficiente em XSLT e .NET, mais especificamente ASP.NET, e se já tiver conhecimentos de RAZOR, consegue facilmente usar o Umbraco como a plataforma para as suas soluções web. Aproveito para adiantar uma novidade, o Umbraco 5, retirou o suporte para XSLT, ou seja, tudo o que era feito em XSLT é agora feito em RAZOR, acompanhando a evolução da própria framework .NET e o seu ecossistema web.

O Umbraco está dividido em várias áreas que estão representadas como ícones no canto inferior direito do backoffice:




Cada uma das áreas vai ser focada neste artigo:

- Content
  - Onde reside o nosso conteúdo
- Media
- Armazenamento de ficheiros (imagens,etc)
- Users
  - Gestão de utilizadores do backoffice
- Settings
  - Onde podemos gerir coisas como templates, tipos de documento e tipos de campos
- Developer
  - Área mais para programadores avançados onde podemos desenvolver/instalar customizações
- Members
  - Aqui gerimos os utilizadores do nosso site

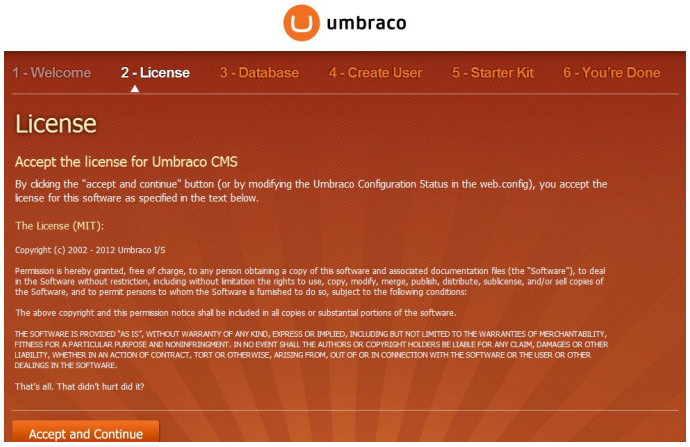
**2. Instalação/Como começar**

O Umbraco tem vários modos de instalação tais como Web Platform Installer, uma plataforma da Microsoft com pacotes de software auto instaláveis, ou podemos simplesmente fazer download do Umbraco no codeplex e instalar manualmente.

Tanto numa opção como noutra, depois de instalar via Web Platform Installer ou de configurar o site no IIS, na opção da instalação manual, vamos ter uma página de instalação:

**2.1 Página inicial de instalação**

### 2.2 Termos e condições

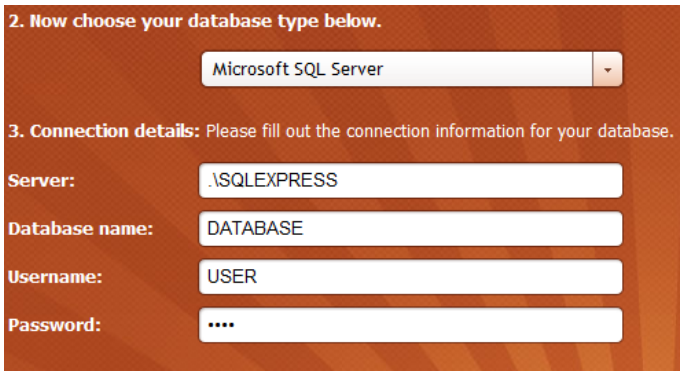



### 2.3 Configuração Base de Dados


Nesta parte da instalação vamos ter de optar onde queremos alojar a base de dados que o Umbraco vai utilizar, opções:

- Já temos uma base de dados SQL Server ou MySQL
    - Ao clicar aí temos 2 formulários:

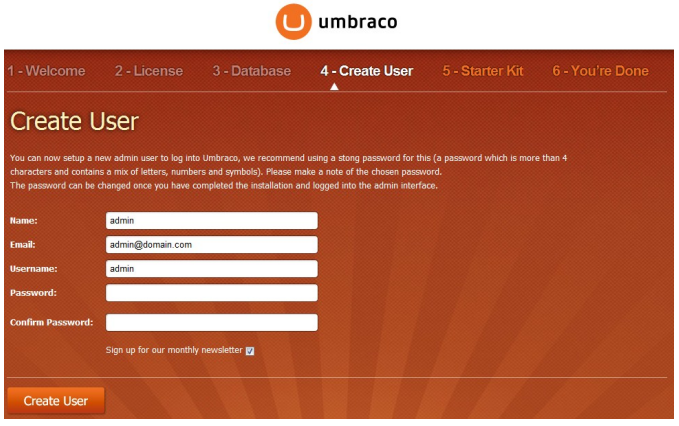
      Podemos seleccionar na caixa de selecção se queremos SQL Server ou MySQL




- Queremos utilizar o SQL CE 4, que basicamente é uma base de dados que vai residir na pasta App_Data da nossa aplicação e que é um simples ficheiro, embora não tenha tantas funcionalidades como o SQL Server, é o suficiente e não nos obriga a ter uma instalação de nenhum SGBD.
    - Ao optar por esta opção, temos de instalar o motor do SQL CE 4 antes, que basicamente é fazer download de algumas *assemblies* to website apresentado como link.
- Já sabemos a nossa *connection string* e queremos apenas escrevê-la e passar ao próximo passo.
    - Aqui é-nos mostrada uma caixa de texto, e podemos introduzir a connection string directamente.
- Precisamos de ajuda, obtemos acesso a vídeos de ajuda para esta configuração.
    - Esta opção será se tivermos dúvidas quanto à opção que queremos escolher, dúvidas que quero precisamente esclarecer com este artigo.

### 2.4 Neste passo o Umbraco vai tratar de se instalar conforme a opção selecciona anteriormente


### 2.5 Após o Umbraco ter configurado a base de dados, configuramos um utilizador



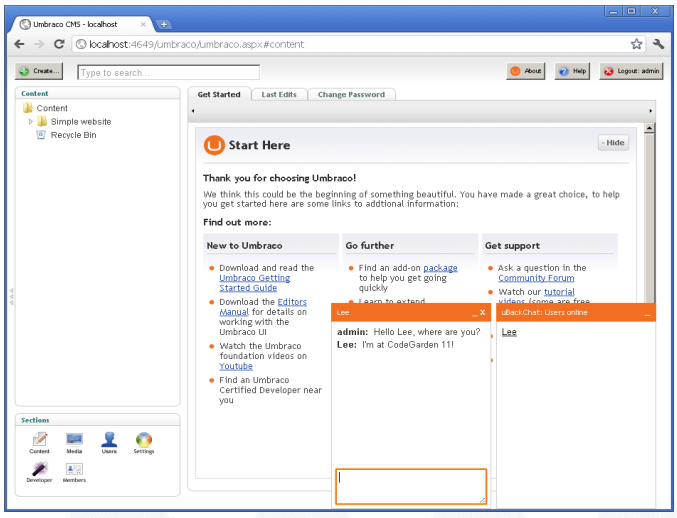
### 2.6 Instalar um kit

Neste passo temos a opção de instalar um dos kit sugeridos para termos um website pronto a funcionar. Penso que para utilizadores iniciados é sempre uma boa ideia, visto que podemos ver como tudo está "montado" entre si, e perceber melhor como o Umbraco funciona.


### 2.7 Instalação finalizada

Aqui já temos o Umbraco instalado, pronto a utilizar.

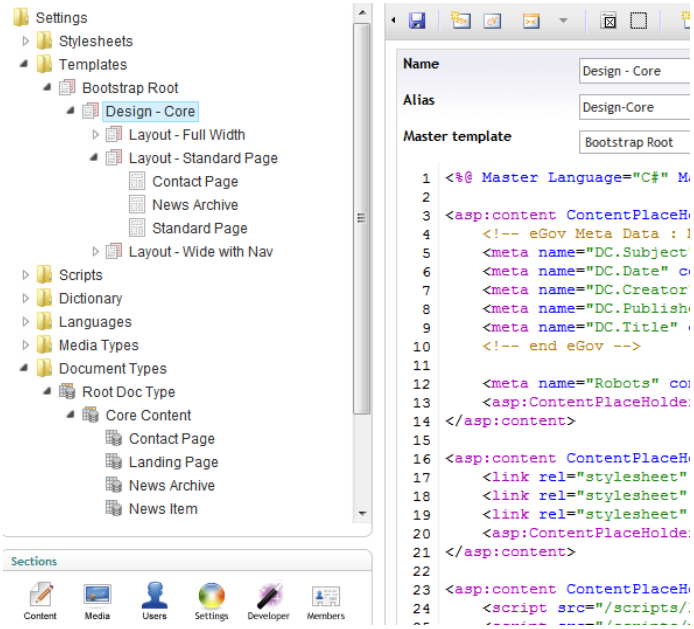

### 3. Árvore de documentos




### 5. Tipos de documento

Na área de settings do Umbraco, entre outras coisas, podemos criar tipos de documento.

Tipos de documento são a estrutura dos nossos dados alojados no Umbraco. Se na instalação seleccionámos algum kit, teremos já alguns tipos de documento criados inicialmente, caso contrário, este será o sítio ideal para começar:

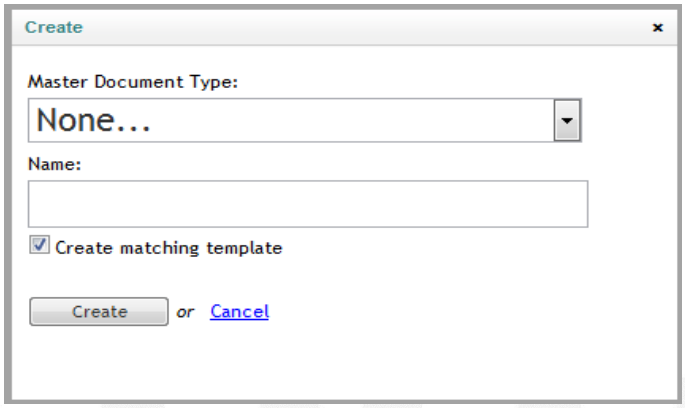



Para criar novos tipos de documento, é bastante simples, ao clicar com o botão direito no nó "Document Types", esta *layer* aparece e temos algumas opções:

- Escolher um tipo de documento já existente para extender, o que significa que este novo tipo vai conter todos os campos que o outro já contém, além dos seus próprios;

- Escrever um nome para o tipo de documento, isto é logicamente obrigatório;

A terceira opção, basicamente, é escolher se queremos criar uma "visualização" para o nosso novo tipo de documento, para que quando criemos um documento desse tipo, na área de conteúdos, a template se encarregar de mostrar o conteúdo visualmente.



Depois de clicarmos em Create, do lado direito vai-nos abrir o detalhe do tipo de documento que acabámos de criar. Aqui temos várias tabs:

- **Info:**

  Nesta tab podemos escolher o name e o alias do tipo de documento. O name é na realidade o nome que queremos dar ao tipo do documento, o nome legível, ao passo que o alias é que se usa do ponto de vista de código, que não pode ter espaços.

  Também se pode escolher que ícone queremos que apareça para este tipo de documento na árvore de documentos e colocar uma descrição.

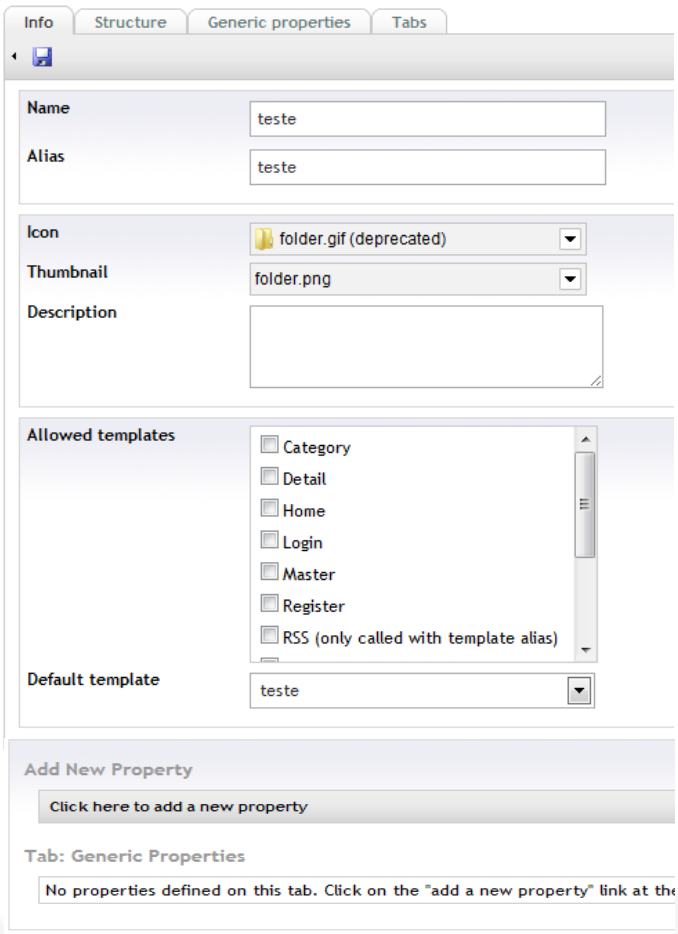
  Mais abaixo temos as templates que autorizamos a apresentar este documento e qual das templates é a que mostra este documento por omissão.

- **Structure:**

  Nesta tab simplesmente podemos escolher que tipos de documento podem ser filhos deste tipo, na árvore de documentos.

- **Generic Properties:**

  Aqui é a zona onde se criam os campos do tipo de documento, existem os mais variados tipos de campos, que iremos focar mais tarde.




- **Tabs:**

Os campos de um tipo de documento podem ser agrupados, cada grupo é mostrado numa tab diferente. Isto normalmente é útil para separarmos campos que dizem respeito a assuntos diferentes, mas existem dentro do mesmo tipo de documento.

### 6. Tipos de campo

Existem vários tipos de campo no Umbraco, e também é possível criar novos tipos de campos. Existem vários pacotes (extensões em forma de pacote que trazem novas funcionalidades para o Umbraco) que instalam novos tipos de campo.

Os tipos de campo mais comuns são:

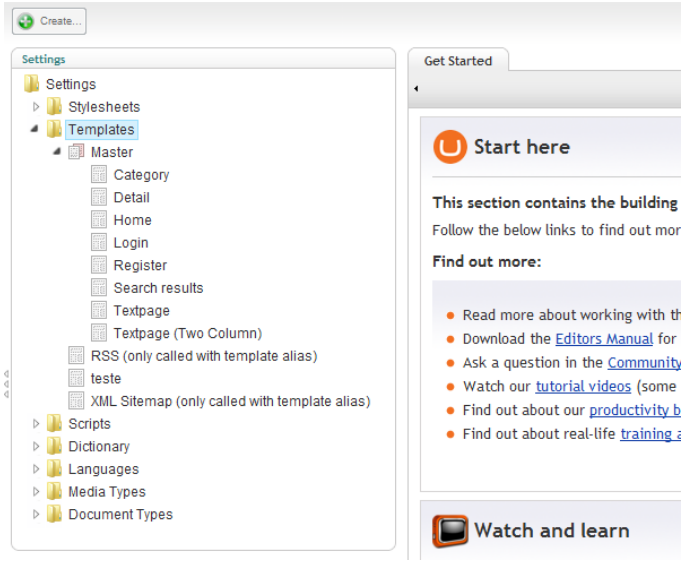
- **Approved color:**
  - Mostra uma lista de cores que podem ser seleccionadas;
- **Checkbox list:**
  - Mostra uma lista de *checkboxes* cujos valores são configuráveis na área *Developer* do *backoffice*;
  - Para cada lista de valores que queremos, criamos um novo tipo de campo baseado neste, com os valores desejados;
- **Content picker:**
  - Este tipo de campo permite ao utilizador escolher outro conteúdo do *backoffice* para ligar ao conteúdo actual;
  - O que fica guardado é o identificador do conteúdo;
- **Date picker with time:**
  - Mostra um calendário que permite também seleccionar a hora;
- **Date picker:**
  - Mostra um calendário que guarda apenas informação de data, sem hora;
- **Dropdown multiple**:
  - Este campo mostra uma lista de opções no formato lista, que permite múltipla selecção;
  - Para cada lista de valores que queremos, criamos um novo tipo de campo baseado neste, com os valores desejados, à semelhança do *checkbox list;*

- **Dropdown**:
  - Este campo tem também uma lista de opções, mas em forma de dropdown e apenas permite seleccionar uma opção;
  - Para cada lista de valores que queremos, criamos um novo tipo de campo baseado neste, com os valores desejados, à semelhança do checkbox list e dropdown multiple;
- **Media picker**:
  - Permite seleccionar um item da área media;
  - Guarda o identificador desse item;
- **Member** picker:
  - Mostra uma *dropdown* que permite seleccionar um membro da área membros do *backoffice*;
- **Numeric**:
  - Uma caixa de texto que apenas aceita números;
- **Radiobox**:
  - Apresenta uma lista de itens em forma de caixas radio, só permite seleccionar um;
  - Para cada lista de valores que queremos, criamos um novo tipo de campo baseado neste, com os valores desejados;
- **Related links**:
  - Neste campo, podemos adicionar links para tanto páginas do Umbraco, como endereços externos;
- **Richtext editor**:
  - Caixa de texto com editor de HTML;
- **Simple editor**:
  - Semelhante ao anterior, mas tem apenas uma pequena parte das opções da versão anterior;
- **Tags:**
  - Permite adicionar palavras a este campo, na forma de uma lista de *tags*;
- **Textbox multiple:**
  - Caixa de texto de múltiplas linhas;
- **Textstring**:
  - Campo para introduzir texto, apenas numa linha;
- **True/false:**
  - Checkbox que guarda um valor booleano;
- **Upload:**
  - Permite fazer o *upload* de um ficheiro directamente para o documento, não indo para à área media.

Além destes tipos, como já foi dito, os pacotes instalado podem introduzir muitos mais, e podem também ser criados mais manualmente na área developer do *backoffice*.

**7. Templates**

As templates, são um dos componentes do Umbraco que serve para apresentar conteúdos. Quando se cria um novo tipo de documento, como mostrado anteriormente, é-nos dada a opção de imediatamente criar uma template para esse mesmo tipo, que servirá para apresentar este tipo de documento quando se criar um nó na árvore de documentos deste tipo. Podemos ter várias templates a apresentar o mesmo tipo de documento, e a mesma template a apresentar vários tipo de documento também.




As templates aparecem na área settings do backoffice sob a forma de nós na árvore, por baixo do nó Templates.

Em termos de código, uma template não é, nada mais nada menos, do que uma masterpage, no entanto, as templates podem ser hierarquizadas. Ao criar uma template podemos, depois de a criar, seleccionar uma outra template como "mãe". O que na prática resulta disto, é a template "mãe" mostrar a "filha" dentro de uma ou várias áreas asp:Content, tal como numa masterpage ASP.NET.

Dentro de uma template podemos incluir código c#, dentro

das usuais tags <% %>. Também podemos incluir macros, que iremos focar mais à frente.

Ao incluirmos código c# dentro das tags, podemos aceder à API do Umbraco, que nos disponibiliza algo como:

```
<% if (Node.GetCurrent().MeuCampo != "algum valor") { %>
<span>...</span>
<% } %>
```

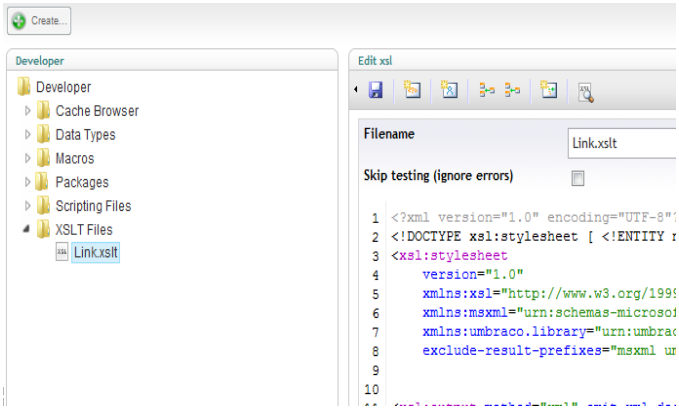
Ao utilizar o Node.GetCurrent(), acedemos ao nó da árvore de documento que estamos a mostrar, a partir daí podemos ter alguma lógica dentro da template, embora aconselhe que a lógica deverá estar encapsulada dentro de macros preferencialmente.

### 8. Macros

As macros no Umbraco têm como objectivo encapsular lógica e visualização e podem ter vários tipos:

- Usercontrol .NET;
- Script Razor ;

XSLT.




As Macros aparecem na área developer do backoffice e estão no nó Macros. Depois conforme o tipo podem também aparecer debaixo dos nós Scripting files (Macros Razor) e XSLT Files (Macros XSLT).

As Macros têm a particularidade de poderem receber parâmetros, que podem ser utilizadas para alterar o seu comportamento.

Numa template em XSLT podemos aceder sempre ao parâmetro $currentPage que contém o XML do documento da página que estamos a ver, e a partir daí utilizar quaisquer campos para a lógica que acharmos necessária.

Numa template Razor, o documento está acessível a partir do objecto Model, para nos manter familiarizados com os ficheiros Razor do MVC3. Existe uma extensa API para se produzirem templates Razor no Umbraco, extensa demais para cobrir toda neste artigo, o ideal será consultar na página oficial do Umbraco a área Wiki que contém uma referência completa.

Um exemplo de uma template Razor seria:

Nesta template acedemos ao objecto Parameter, que contém todos os parâmetros passados para a Macro, neste caso o parâmetro tem o nome tagList.

```
@{
  var tagList = Parameter.tagList.Split(',');
}

<div class="block">
  <ul class="tag-list">
  @foreach (var tag in tagList)
  {
    <li><a href="#">@tag</a></li>
  }
  </ul>
</div>
```

### 9. Members e autenticação

Members é a área do Umbraco que serve para alojar os utilizadores do site. Um membro é um utilizador do site, um user é um utilizador de backoffice. Os membros podem ter vários grupos e tipos. Os dados que constam num membro são editáveis, tal como de um tipo de documento se tratasse. Os campos que existem num membro, pertencem a um tipo de membro, depois cada membro pode pertencer a um determinado tipo.

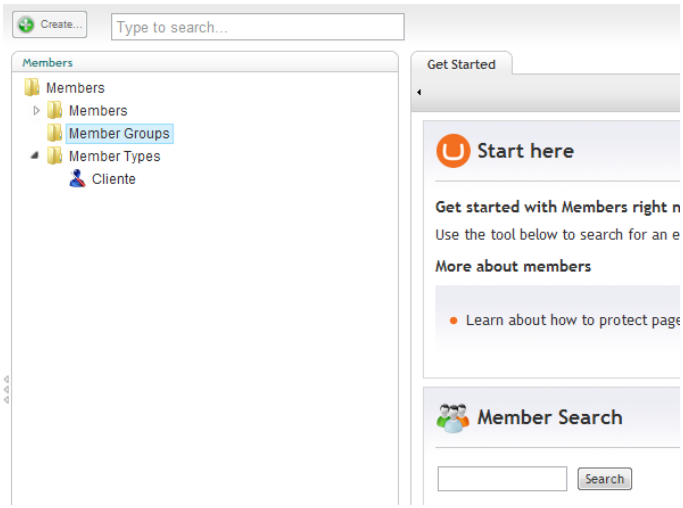
Em termos de API, o Umbraco utiliza o standart Membership Provider, daí ser bastante fácil criar código dentro de qualquer template ou Macro para aceder à API e também de criar um sistema de autenticação.

Para aceder aos campos adicionados, que não constam no MembershipUser, tem que se adicionar informação ao web.config (dentro do system.web): (name é o nome do campo adicionado ao membro)

```
<profile defaultProvider="UmbracoMemberProfileProvider"
enabled="true">
      <providers>
        <clear/>
        <add name="UmbracoMemberProfileProvider"
type="umbraco.providers.members.UmbracoProfileProvider, umbraco.providers"/>
      </providers>
      <properties>
        <clear/>
        <add name="name" allowAnonymous="false"
provider="UmbracoMemberProfileProvider"
type="System.String"/>
      </properties>
    </profile>
```

Na criação de um utilizador via Membership, para aceder a estes campos extra utiliza-se o ProfileBase:

```
Membership.CreateUser(email, password, email);
var profile = ProfileBase.Create(email);
profile["name"] = name;
profile.Save();
```



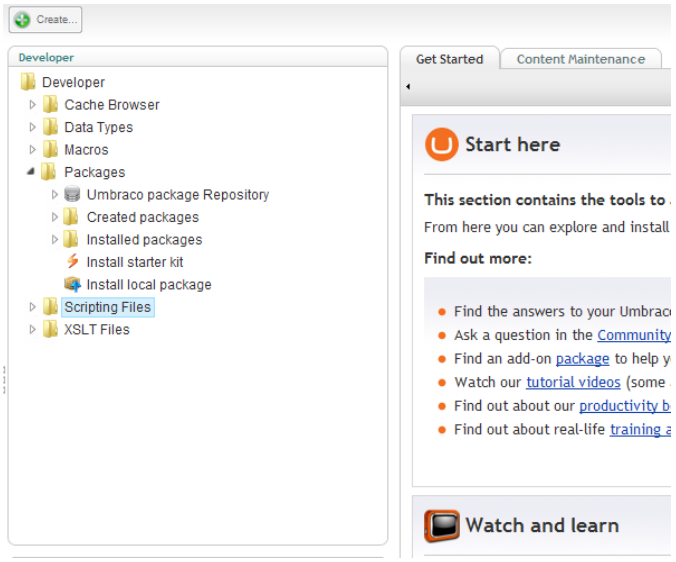

**10. Packages**

Por último, foco algo relevante, que tem directamente a ver com a comunidade (ecossistema) Umbraco: as packages.

As packages, são conjuntos de funcionalidades que são adicionados ao backoffice, ao site, tipos de campos, áreas novas de backoffice, funcionalidades novas para áreas já existentes, entre outras.

Na área developer do backoffice do Umbraco existe um repositório de packages, um género de appstore de pacotes para o Umbraco. Também podemos fazer download de um pacote directamente e escolher a opção install local package e fornecer o ficheiro zip que fizemos o download.

Deixo a sugestão do pacote DAMP que extende as funcionalidades de media do Umbraco adicionando várias funcionalidades bastante úteis.




**11. Umbraco 5 e conclusão**

A mais recente versão do Umbraco é o 5 (nome de código Jupiter).

Algumas das novidades do Umbraco 5 é a quebra de suporte de XSLT, ou seja, já não é possível construir Macros utilizando essa linguagem, optaram por dar ênfase ao Razor, que na minha opinião foi uma boa jogada, visto Razor ser o futuro da apresentação no ecossistema ASP.NET.

Outra das novidades é o site do Umbraco ter uma estrutura muito menor e permitir termos o nosso site MVC3 dentro do próprio Umbraco, aproveitado o CMS mas permitindo termos o nosso projecto com os nossos componentes, que nada têm a ver com o CMS.

Espero que este artigo tenha servido para esclarecer dúvidas a já utilizadores de Umbraco e que tenha incentivado os curiosos a experimentar este fantástico produto.

## AUTOR

**Escrito por Ricardo Rodrigues**

Técnico Nível III em Informática/Gestão pela Fundação Escola Profissional de Setúbal, tendo ingressado após na FCT da Universidade Nova de Lisboa.
Posteriormente frequentou vários cursos da Microsoft em diversas áreas como Windows Forms, ASP.NET, Securing .NET Applications, WCF, WWF, Web Services e COM+ tendo obtido as certificações MCP .NET 2.0, MCAD .NET 1.1, MCSD .NET 1.1, MCPD Windows, Web e Distributed Applications e MCPD - Enterprise Applications Developer. (MCP Profile) Contribui activamente em comunidades como StackOverflow e também possui um blog/twitter com temática relacionada: Blog / @ricmrodrigues

# COLUNAS

Visual (NOT) Basic - XNA: Ataque no quintal (parte 2/2)

Enigmas de C# - O Estranho Caso Dos Enumerados

# XNA: ATAQUE NO QUINTAL (PARTE 2/2)

Lembram-se do jogo que começámos a construír na edição passada?

Onde é que ficámos? Vamos retomar a série com uma recapitulação do que deixamos feito na primeira parte.

Temos o ciclo de vida montado, ecrãs de derrota e victória, o coreógrafo da entrada de enimigos e toda uma estrutura montada, pronta para ser preenchida.

Conseguimos ver o menu e entrar no jogo, os inimigos entram nas filas e quando passamos o rato por cima das extremidades verticais do ecrã, aparecem e desaparecem os selectores do terreno.

A melhor maneira de começar a segunda parte é apresentando as decisões que foram feitas antes, relativamente às torres e suas características, bem como os inimigos.

**As Torres**

Ainda que o ataque seja no quintal, no final do artigo podem contar com um jogo completo que contempla o que melhor a arte da guerra tem para oferecer no campo da defesa baseada em torres (esta informação não está confirmada por o Ministério da Defesa J).

**BÁSICO**

| | |
|---|---|
| DANO | 10 |
| ALCANCE | CURTO |
| RAPIDEZ | 1 SEG. |
| PREÇO | 100 |

**DUPLO**

| | |
|---|---|
| DANO | 20 |
| ALCANCE | CURTO |
| RAPIDEZ | 1 SEG. |
| PREÇO | 150 |

**TRIPLO**

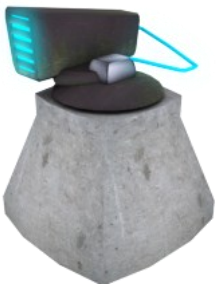
| | |
|---|---|
| DANO | 30 |
| ALCANCE | CURTO |
| RAPIDEZ | 1 SEG. |
| PREÇO | 200 |

**VULCÃO**

| | |
|---|---|
| DANO | 10 |
| ALCANCE | CURTO |
| RAPIDEZ | 500 MS. |
| PREÇO | 300 |

**MORTEIRO**

| | |
|---|---|
| DANO | 350 |
| ALCANCE | LONGO |
| RAPIDEZ | 3 SEG. |
| PREÇO | 325 |



**CONGELADOR**

| | |
|---|---|
| DANO | 25 + GELO |
| ALCANCE | CURTO |
| RAPIDEZ | 5 SEG. |
| PREÇO | 400 |

**INFERNO**

| | |
|---|---|
| DANO | 120 |
| ALCANCE | MÉDIO |
| RAPIDEZ | 250 MS. |
| PREÇO | 500 |

**APOCALIPSE**

| | |
|---|---|
| DANO | 50 |
| ALCANCE | LONGO |
| RAPIDEZ | 80 MS. |
| PREÇO | 650 |

**Os inimigos**

A toque de XML, as vossas torres serão bombardeadas com o pavoneio de 5 ostentosos inimigos.

**JIPE**

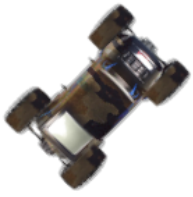
| | |
|---|---|
| VELOCIDADE | 50 |
| ARMADURA | 50 |
| RECOMPENSA | 20 |
| DANO | 10 |

**APCARRIER**

| | |
|---|---|
| VELOCIDADE | 45 |
| ARMADURA | 150 |
| RECOMPENSA | 40 |
| DANO | 15 |

**CARRO-BOMBA**

| | |
|---|---|
| VELOCIDADE | 65 |
| ARMADURA | 20 |
| RECOMPENSA | 30 |
| DANO | 50 |



**BUGGY**

| | |
|---|---|
| VELOCIDADE | 95 |
| ARMADURA | 25 |
| RECOMPENSA | 55 |
| DANO | 60 |

**TANQUE**

| | |
|---|---|
| VELOCIDADE | 15 |
| ARMADURA | 1000 |
| RECOMPENSA | 150 |
| DANO | 70 |

**As regras**

Como qualquer outro jogo, existe um conjunto de regras por as quais a mecânica se rege. O jogador possuí, como já foi anteriormente descrito, duas filas para colocação de torres: uma em cima, outra em baixo. Os inimigos surgem da esquerda para a direita, cada um em um dos três corredores possíveis.

Cada torre, foca-se num qualquer inimigo, desde que ao seu alcance, e fica com ele até sair de alcance ou até o destruir. A cada inimigo está associado um valor recompensatório, a somar aos recursos disponíveis para a acquisição de mais torres, assim que este for destruído. Não é possível, nesta implementação, nem vender torres nem destruír torres.

Sempre que um inimigo consiga atravessar o corredor, o jogador perde "vida". As condições de vitória implicam a destruição de todos os inimigos de todas as ondas e todos os níveis antes da "vida" chegar a zero. Por sua vez, as condições de derrota implicam a chegada da "vida" a zero.

Apresentações feitas, está na altura de voltar ao trabalho. Volto a lembrar que não existe uma forma rígida de implementar lógica.

A que foi implementada pode não ser a mais eficaz nem a mais correcta. É simplesmente uma das mais simples. Isto facilita o processo de aprendizagem e análise. Deixo as implementações padronizadas para os mestres de XNA que nascerem dos artigos ;)

Comecemos por escrever tudo o que são dados e utilidades que são transversais a todo o projecto.

**Dados e utilidades transversais**

Existem várias abordagens, em termos de organização, mas vou optar por declarar todos os objectos que irão ser utilizados logo de início, e carregá-los na troca de fase do menu para o jogo.
Assim, fazemos uma visita ao ficheiro dados.vb para colocar tudo o que são objectos do jogo, para além dos que já lá temos:

```
'Variáveis do jogador
    Public Dinheiro As Integer
    Public VidaRestante As Integer

    'Fundos
    Public fundoJogo As Texture2D
    Public fundoEntrada As Texture2D
    Public fundoEscolha As Texture2D
    Public gameover As Texture2D
    Public ganhou As Texture2D

    'UI
    Public seta As Texture2D
```

## XNA: ATAQUE NO QUINTAL (PARTE 2/2)

```
    Public titulo As Texture2D
    Public botao_defender As Texture2D
    Public botao_sair As Texture2D
    Public botao_defender_p As Texture2D
    Public botao_sair_p As Texture2D
    Public botao_ora_bolas As Texture2D
    Public botao_ora_bolas_p As Texture2D
    Public torreui_selector As Texture2D
    Public terreno_selector As Texture2D
    Public barra_vida As Texture2D
    Public sacodinheiro As Texture2D
    Public semdinheiro As Texture2D
    Public torre1_UI As Texture2D
    Public torre2_UI As Texture2D
    Public torre3_UI As Texture2D
    Public torre4_UI As Texture2D
    Public torre5_UI As Texture2D
    Public torre6_UI As Texture2D
    Public torre7_UI As Texture2D
    Public torre8_UI As Texture2D
    Public Rect_Defender As Rectangle
    Public Rect_Sair As Rectangle
    Public Rect_OraBolas As Rectangle

    'Dados dos XML
    Public baseDados_Tempos As XDocument
    Public baseDados_Inimigos As XDocument
    Public baseDados_Torres As XDocument

    'Partículas e efeitos
    Public poeira As Texture2D
    Public impacto As Texture2D
    Public tiro As Texture2D
    Public impacto_azul As Texture2D
    Public explosao As Texture2D
    Public chamusca As Texture2D

    'Fontes
    Public FonteGeral As SpriteFont
    Public FonteGrande As SpriteFont

    'Torres
    Public todos_base As Texture2D
    Public basico_rotativo As Texture2D
    Public duplo_rotativo As Texture2D
    Public triplo_rotativo As Texture2D
    Public vulcao_rotativo As Texture2D
    Public morteiro_rotativo As Texture2D
    Public congelador_rotativo As Texture2D
    Public inferno_rotativo As Texture2D
    Public apocalipse_rotativo As Texture2D

    'Sons
    Public tiro_basico As SoundEffect
    Public tiro_duplo As SoundEffect
    Public tiro_triplo As SoundEffect
    Public tiro_vulcao As SoundEffect
    Public tiro_morteiro As SoundEffect
    Public tiro_congelador As SoundEffect
    Public tiro_inferno As SoundEffect
    Public tiro_apocalipse As SoundEffect
    Public explosao_inimigo As SoundEffect

    'Inimigos
    Public inimigo_jipe As Texture2D
    Public inimigo_apc As Texture2D
    Public inimigo_tanque As Texture2D
    Public inimigo_carrobomba As Texture2D
    Public inimigo_buggy As Texture2D

    'Posições de torres
    Public PosicoesPossiveis As New List(Of _
PosicaoTorre)

    'Colecções do jogo
    Public objectosJogo As New List(Of Objecto)
    Public particulas_sobrepostas As New List(Of
Particula)
    Public particulas As New List(Of Particula)
    Public particulasAnimadas As New List(Of
SpriteAnimado)
    Public informacoes As New List(Of Splash)
```

Comparando com o que ficou feito na primeira parte, tendo em conta que são todos os elementos do jogo, podemos observar um novo tipo de dados: SoundEffect Cada instância de SoundEffect representa, neste caso, um ficheiro WAV que pode ser utilizado em qualquer altura. Dada a semelhança entre algumas torres, e ainda que se reserve memória para os sons de todas as torres, vamos utilizar o mesmo conteúdo para algumas delas, por exemplo para a Inferno e Apocalipse que partilham o mesmo som. Algures para o meio, avistamos também outro tipo de dados: SpriteFont.

Em suma, uma SpriteFont permite a utilização de qualquer fonte do sistema, mediante parametização do seu tamanho bem como algumas outras características, através de XML. O IDE gera tudo o que é necessário apenas por seleccionar AddàNew ItemàSpriteFont em qualquer local do content pipeline. As declarações das coleções, ao fundo, denunciam 3 novas classes que são necessárias para o jogo, mas não foram introduzidas na primeira parte:

**Splash**

A classe Splash representa um texto ou um sprite (imagem 2d) cuja visibilidade é limitada por um valor de duração. A partir do momento em que é colocado um objecto deste tipo nas colecções do jogo, este fica visível e o tempo de visualização é contabilizado até que atinja a duração. Quando isso acontecer, o objecto é removido da colecção e desaparece.

```
Public Class Splash

    Public Posicao As Vector2
    Public Grafico As Texture2D
    Public Duracao As Integer
    Public Destruir As Boolean = False
    Public Texto As String

    Private FrameInterno As Integer

    Sub New(Posicao As Vector2, Duracao As Integer,
Grafico As Texture2D)
        Me.Posicao = Posicao
        Me.Grafico = Grafico
        Me.Texto = String.Empty
        Me.Duracao = Duracao
    End Sub

    Sub New(Posicao As Vector2, Duracao As Integer,
Texto As String)
        Me.Posicao = Posicao
        Me.Grafico = Nothing
        Me.Texto = Texto
        Me.Duracao = Duracao
    End Sub
```

```
    Public Sub ComunicarFrame()
        If Destruir Then Exit Sub
        FrameInterno += 1
        If FrameInterno Mod Duracao = 0 Then
            Destruir = True
        End If
    End Sub

    Public ReadOnly Property Rectangulo
                                As Rectangle
        Get
            Return New Rectangle(CInt(Posicao.X),
           CInt(Posicao.Y), CInt(Grafico.Width),
           CInt(Grafico.Height))
        End Get
    End Property

    Public Sub Desenhar(SB As SpriteBatch)
        If Me.Grafico IsNot Nothing Then
            SB.Draw(Me.Grafico, Me.Rectangulo,
                            Color.White)
        End If
        If Me.Texto <> "" Then
            SB.DrawString(FonteGeral, Me.Texto,
         Me.Posicao, Color.White, 0,
         Vector2.Zero, 1.5, SpriteEffects.None, 0)
        End If
    End Sub
End Class
```

À semelhança dos restantes objectos, e diferente do método para desenhar inimigos da primeira parte, vamos passar a lógica de desenho para dentro do objecto. Para além de fazer mais sentido estruturalmente, descomplica o código na classe principal.

Para tal é apenas necessário passar o SpriteBatch da classe principal para que cada objecto se possa nele representar. Isto é feito através de passagem no método Desenhar.

Para que o Splash possa determinar se o objecto é para destruír ou não, o método Update da classe principal deve chamar o método ComunicarFrame, a cada frame.

Se o frame interno chegar à duração do objecto, é levantada a flag de destruição que fará com que este seja eliminado do próximo ciclo.

O Splash desenha texto ou um sprite, dependendo do construtor usado.

**Particula**

A classe Particula representa um sprite orientado que pode, ou não, sofrer algumas transformações ou deformações ao longo do tempo.

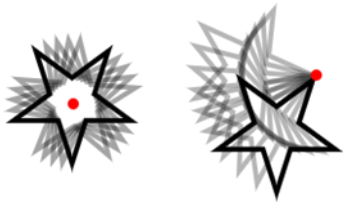
Existem 3 modificadores: Velocidade angular, que determina a velocidade com que o sprite gira em torno do seu centro; Crescimento, que determina a velocidade do seu crescimento, positivo ou negativo; Desvanecimento, que determina a velocidade com que desvanece.

```
Public Class Particula

    Public Posicao As Vector2
    Public Grafico As Texture2D
    Public Opacidade As Double = 255
    Public EscalaX As Double
    Public EscalaY As Double
    Public Rotacao As Double
    Public Desvanecimento As Double
    Public Crescimento As Double
    Public VelocidadeAngular As Double

    Public Sub Deformar()
        Opacidade -= Desvanecimento
        EscalaX += Crescimento
        EscalaY += Crescimento
        Rotacao += VelocidadeAngular
    End Sub

    Sub New(Grafico As Texture2D, Escala As Double,
            Desvanecimento As Double, Crescimento
            As Double, Vang As Double)
        Me.Grafico = Grafico
        Me.EscalaX = Escala
        Me.EscalaY = Escala
        Me.Desvanecimento = Desvanecimento
        Me.Crescimento = Crescimento
        Me.VelocidadeAngular = Vang
    End Sub

    Public ReadOnly Property Rectangulo
                As Rectangle
        Get
            Return New Rectangle(CInt(Posicao.X),
CInt(Posicao.Y), CInt(Grafico.Width * EscalaX),
CInt(Grafico.Height * EscalaY))
        End Get
    End Property

    Public Sub Desenhar(SB As SpriteBatch)
       Dim tmp0 As Byte = CByte(Me.Opacidade)
       Dim C As Color =
            New Color(tmp0, tmp0, _ tmp0, tmp0)
      SB.Draw(Me.Grafico, Me.Posicao, Nothing, C,_
         Rads(CSng(Me.Rotacao)), New Vector2 (CSng
         (Grafico.Width / 2), CSng(Grafico.Height /
         2)), New  Vector2(CSng(Me.EscalaX), CSng
         (Me.EscalaY)),SpriteEffects.None, 0)
    End Sub

End Class
```

O Draw da classe principal deverá chamar o método Desenhar e o Update deverá chamar, a cada frame, o método Deformar, para que os valores configurados possam ser aplicados às suas propriedades.

Este tipo de objecto é usado, por exemplo, para criar a poeira levantada por o movimento dos inimigos.

Para orientar o sprite, a Particula serve-se de um overload do método Draw, classe SpriteBatch, que especifica um parametro para o centro de rotação e o ângulo.



O centro é relativo ao sprite original. Pode ser encarado co-

## XNA: ATAQUE NO QUINTAL (PARTE 2/2)

mo um eixo que trespassa o sprite.

O ponto vermelho na imagem representa o centro de rotação, que quando aplicada em pontos distintos na estrela, causa diferentes transformações.

*SpriteAnimado*

A classe SpriteAnimado representa um sprite capaz de reproduzir uma animação.

A animação acontece quando a partir de uma imagem maior, se desenha apenas uma fracção ponderada de cada vez, sequencialmente e em padrão.

Quando a animação chega ao fim, é levantada uma flag para que o sprite seja eliminado.

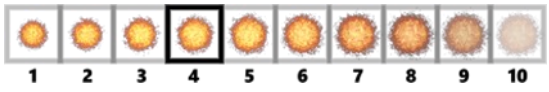
Se a imagem for como a da figura (sem as guias a preto), desenhar apenas a fracção correspondente a cada quadrado a cada frame, criará a animação da explosão.

O número de frames da animação é determinado por a divisão do comprimento do sprite original por a largura de cada frame, fornecida no constructor.

Agarrando no exemplo abaixo, e imaginando que cada frame tem 100px, o comprimento total da imagem principal seria de 1000px com a altura de um frame.

A frequência é também passada no constructor. Esta representa a velocidade com que os frames da animação são trocados, o que representa uma animação mais ou menos rápida.

Quanto mais frames existirem, mais fluída será a animação, mas maior fica a imagem principal. O ideal é procurar um equilíbrio entre a fluídez e o tamanho da imagem, dependendo dos orçamentos em causa.




```
Public Class SpriteAnimado
    Public Posicao As Vector2
    Public Grafico As Texture2D
    Private Largura As Integer
    Private Frequencia As Short

    Public ReadOnly Property RectanguloEfectivo As_
Rectangle
        Get
          Return New Rectangle(FrameActual * _
Largura, 0, Largura, Grafico.Height)
        End Get
    End Property

    Private FrameInterno As Integer = 0
    Private FrameActual As Integer
    Public Destruir As Boolean = False

    Sub New(Grafico As Texture2D, Largura As _
                Integer, Frequencia As Short)
        Me.Largura = Largura
        Me.Grafico = Grafico
        Me.Frequencia = Frequencia
        FrameActual = 0
    End Sub

    Public Sub ComunicarFrame()
        If Me.Destruir Then Exit Sub
        FrameInterno += 1
        If FrameInterno Mod Frequencia = 0 Then
            If FrameActual * Largura = Grafi-
co.Width - Largura Then
                Me.Destruir = True
            Else
                FrameActual += 1
            End If

        End If
    End Sub

    Public Sub Desenhar(SB As SpriteBatch)
        SB.Draw(Me.Grafico, Me.Posicao,
Me.RectanguloEfectivo, Color.White, 0, Vec-
tor2.Zero, Vector2.One, SpriteEffects.None, 0)
    End Sub

End Class
```

O Draw da classe principal deverá chamar o método Desenhar e o Update deverá chamar, a cada frame, o método ComunicarFrame.

É no método ComunicarFrame que o frame interno é comparado com a frequência da animação, e sempre que atingir o mesmo valor, significa que tem de avançar um frame.

O Draw principal, no nosso caso, é disparado 60 vezes por segundo. Se a frequência do SpriteAnimado for de 30, o frame vai avançar uma vez a cada meio segundo, duas vezes por segundo.

*Utilidades*

Ao longo de todo o jogo, são utilizadas com alguma frequência determinos cálculos que resolvi generalizar.

Vamos criar um módulo chamado Utils:

```
Module Utils

    Public Function Rads(Graus As Single) As Single
        Return CSng(Graus * (Math.PI / 180))
    End Function

    Public Function Graus(Rad As Single) As Single
        Return CSng(Rad * (180 / Math.PI))
    End Function

    Public Function RndNum(Min As Double, Max As
                                Double) As Double
        Dim R As New Random(Now.Hour + Now.Minute +
                    Now.Second + Now.Millisecond)
        Return ((Max - Min) * R.NextDouble) + Min
    End Function

    Public Function RndNum(Min As Integer, Max As
                              Integer) As Integer
        Dim R As New Random(Now.Hour + Now.Minute +
                    Now.Second + Now.Millisecond)
        Return R.Next(Min, Max)
    End Function

End Module
```

Os métodos Rads e Graus, tratam da conversão graußàradianos para utilizar em cálculos de rotação e o método RndNum fornece-nos uma forma de obtermos um número aleatório a partir de um determinado alcance.

Para manter coerência, existe um overload deste método que devolve exactamente o que o método Next da classe Random devolve.

**Terminar os objectos chave**

Na primeira parte, cada fase estava acompanhada de uma listagem de ficheiros envolvidos.

Nesta parte, não vou discriminar os ficheiros pois estão implícitos nos trechos de código e/ou na breve descrição de cada elemento… e são vários.

Todos os recursos que detectarem no load têm de ser adicionados ao content pipeline, basta analisar os caminhos de load, que fornecerei mais adiante.

Como já foi explicado na primeira parte, até pode ser encarado como um treino na segunda parte.

**Inimigo**

De visita à classe Inimigo, as alterações para a completar são muito poucas.

Em primeiro lugar, agora que sabemos que existe uma torre capaz de congelar unidades, vamos colocar mais um membro (não vamos ligar ao encapsulamento, ou à falta dele):

```
Public Congelado As Boolean = False
```

Este membro permite-nos determinar se aquele inimigo está ou não congelado. Isto é importante para o método de desenho pois permite-lhe decidir se a unidade deverá ser pintada de azul ou não.

Para além deste novo membro, e como já foi referido, a lógica de desenho foi passada para dentro da classe:

```
    Public Sub Desenhar(SB As SpriteBatch)
        Dim temp_Rect As New Rectangle(CInt
          (Me.Posicao.X), CInt(Me.Posicao.Y),
          Me.Grafico.Width, 120)
        Dim corInimigo As Color = Color.White
        If Me.Congelado Then corInimigo = _
                              Color.Blue
        SB.Draw(Me.Grafico, temp_Rect, corInimigo)
        Dim tmpV As Integer = CInt((Me.Vida *
            Me.Grafico.Width) / Me.VidaMaxima)
        Dim vida_rect = New Rectangle(temp_Rect.X,
                              temp_Rect.Y, tmpV, 5)
        SB.Draw(barra_vida, vida_rect, Color.White)
    End Sub
```

Assim, na classe principal basta passar o SpriteBatch actual na chamada do Desenhar de cada inimigo.
Para além do inimigo em si, é também desenhada uma barra de vida por cima.
A barra de vida é apenas um pixel vermelho que é forçado a esticar por o comprimento e largura que desejarmos. Através de uma regra de três simples conseguimos determinar o comprimento da barra adequado à vida restante do inimigo. Como o aspecto que procuramos é linear, “esticar” 1 pixel é suficiente.

**Torre**

De visita à classe Torre, existem muitas alterações a efectuar. São as torres que detêm grande parte da mecânica do jogo pois foi nelas que decidi implementar a lógica da escolha de inimigo e disparo.
A primeira alteração é a adição de dois novos membros:

```
Public SomTiro As SoundEffect
    Private FrameInterno As Integer
```

SomTiro detém o efeito sonoro que o disparo daquela torre produz e o FrameInterno é usado para calcular as frequências do disparo.

As alterações implicam a adição de dois métodos importantes:

```
Public Sub ValidarAlvo()
 If Alvo IsNot Nothing Then
  Dim DistanciaTorreInimigo As Double = Math.Sqrt
     ((Math.Pow(Alvo.Centro.X - Me.Centro.X, 2)) +
       (Math.Pow(Alvo.Centro.Y - Me.Centro.Y, 2)))
            If Me.Alcance < DistanciaTorreInimigo
 Then Alvo = Nothing : Exit Sub
   If Alvo.Vida <= 0 Then
     Dim novaParticula As New SpriteAnimado _
        (explosao, 200, 2)
         novaParticula.Posicao = Alvo.Centro -
         New Vector2(100, 100)
         particulasAnimadas.Add(novaParticula)
         Alvo = Nothing
    End If
 End If
End Sub
```

O método ValidarAlvo é responsável por determinar se o inimigo está ao alcance e se já foi ou não destruído. Caso a torre tenha um alvo designado, começa-se por determinar a distância matemática entre os dois objectos. Se esta distância não estiver ao alcance da torre, o alvo designado é removido. Caso contrário, verificamos se o inimigo ainda tem vida. Se não for o caso, é adicionada uma explosão à colecção de partículas animadas e o alvo designado é removido. Este método deve ser chamado a cada ciclo do Update.

O outro método importante é o DispararAAlvo:

```
Public Sub DispararAAlvo()
        FrameInterno += 1
        If Alvo IsNot Nothing Then
            Me.AnguloArma = Graus(CSng(Math.Atan2
((Me.Centro.Y - Alvo.Centro.Y), (Me.Centro.X - Al-
vo.Centro.X)))) - 90
            If FrameInterno Mod Frequencia = 0 Then
                For i As Integer = 1 To _
```

## XNA: ATAQUE NO QUINTAL (PARTE 2/2)

```
                        Me.NumeroCanos
                    Dim tiro_som As _
SoundEffectInstance = Me.SomTiro.CreateInstance
                    tiro_som.Pitch = 1
                    tiro_som.Play()
                Next
                Alvo.Vida -= Dano
                Dim novaParticula As Particula = _
                 Nothing
            Dim novaParticulaFogo As Particula = _
                 Nothing
 Select Case Me.Nome.ToUpper
   Case "CONGELADOR"
    Alvo.Congelado = True
    If Alvo.Velocidade > 1 Then
       Alvo.Velocidade -= 10
    End If
    If Alvo.Velocidade < 1 Then Alvo.Velocidade = 1
     novaParticula = New Particula
           (impacto_azul,0.6, 25, 0.008, 0.1)
     novaParticulaFogo = New Particula _
           (impacto_azul, 0.6, 25, 0.008, 0.1)
    Case "MORTEIRO"
     novaParticula = New Particula
           (impacto, 0.8, 25, 0.008, 0)
     novaParticulaFogo = New Particula
           (impacto, 0.8, 25, 0.008, 0)
    Case Else
     novaParticula = New Particula
           (impacto, 0.3, 25, 0.008, 0)
     novaParticulaFogo = New Particula
           (impacto, 0.3, 25, 0.008, 0)
 End Select

 novaParticula.Posicao = Alvo.Centro +
        New Vector2 (RndNum(-5, 5), RndNum(-5, 5))
 particulas_sobrepostas.Add(novaParticula)

 For Each PosFogo As Vector2 In PosicaoFogo
  Dim tmpVec As New Vector2
  Dim copiaParticulaFogo As New Particula _
      (novaParticulaFogo.Grafico, _
       novaParticulaFogo.EscalaX,_
       novaParticulaFogo.Desvanecimento, _
       novaParticulaFogo.Crescimento, 0)
  If Me.Nome.ToUpper <> "CONGELADOR" Then _
     copiaParticulaFogo.Grafico = tiro
     tmpVec.X = CSng((Posicao.X +  _
     Me.Grafico.Width / 2) - (Math.Cos(Rads _
       (Me.AnguloArma+PosFogo.X+90)) * PosFogo.Y))
    tmpVec.Y = CSng((Posicao.Y + _
    Me.Grafico.Height / 2) - (Math.Sin(Rads _
    (Me.AnguloArma + PosFogo.X + 90)) * PosFogo.Y))
    copiaParticulaFogo.Posicao = tmpVec _
    copiaParticulaFogo.Rotacao = Me.AnguloArma - 90
    particulas_sobrepostas.Add(copiaParticulaFogo)
                Next
             End If
          End If
End Sub
```

O método DispararAAlvo é importante e tem de ser chamado a cada ciclo do Update, pois implica um controlo de frame interno. Para além disso, é este método faz com que a torre dispare à frequência para ela designada, e com as suas características.

A primeira coisa que o método trata, é o ângulo da parte rotativa da torre. Isto faz com que os canos se voltem para o alvo antes de disparar. De seguida o frame interno é comparado com a frequência para determinar se estamos num ciclo onde a torre é suposto disparar.

Quando se trata do caso de estar a disparar, soa o som do disparo tantas vezes quantos canos a torre possuir. Isto apesar de parecer redundante, porque o som sai ao mesmo tempo, intensifica o volume e acrescenta algum realismo. Depois disso, é descontado o valor de dano ao inimigo alvo, e por fim são desenhados os sprites de disparo e impacto, tanto na ponta dos canos da torre, como sobre o inimigo, dando a ilusão de um disparo entre os dois objectos. Todas as torres disparam da mesma forma, excepto a torre Congelador, que tem um disparo azul, e o Morteiro que tem um disparo maior. Para o artigo, resolvi utilizar o nome da torre, até porque são apenas duas. De qualquer das formas, seria uma boa ideia utilizar identificadores para cada torre.

Por fim, como já referido, passamos a lógica de desenho para a classe:

```
Public Sub Desenhar(SB As SpriteBatch)
  Dim temp_Rect As New Rectangle(CInt _
(Me.Posicao.X), CInt(Me.Posicao.Y), 86, 86)
        SB.Draw(Me.Grafico, temp_Rect, Color.White)
        temp_Rect.X += CInt(Me.CentroRotativo.X)
        temp_Rect.Y += CInt(Me.CentroRotativo.Y)
        Dim rot_rect As New Rectangle(CInt _
        (Me.Posicao.X) + 43, CInt(Me.Posicao.Y) + _
        43, Me.GraficoMovel.Width, _
              Me.GraficoMovel.Height)
      SB.Draw(Me.GraficoMovel, rot_rect, Nothing, _
      Color.White, Rads(Me.AnguloArma), _
      Me.CentroRotativo, SpriteEffects.None, 1)
End Sub
```

Com as versões finais dos inimigos e torres, temos de nos focar na colocação em jogo das torres e completar a colocação dos inimigos, iniciada na parte anterior.

De visita ao ficheiro Principal.Carregador.vb, que descreve uma parte da classe Principal, vamos escrever a versão final do método ColocarInimigos, que sofreu algumas alterações para a original:

```
Private Sub ColocarInimigos(frame As Integer)
 Dim Nivel As XElement = baseDados_Tempos.
  <tempos>.<nivel>.Where(Function(n) n.@id = _
  Me.nivel.ToString)(0)
 Dim Onda As XElement = Nivel.<onda>.Where _
    (Function(o) o.@id = Me.onda.ToString)(0)
 Dim Tempo As XElement = Onda.<tempo>.Where _
   (Function(t) t.@frame = frame.ToString)(0)

 If Tempo Is Nothing Then Exit Sub

  ondaADecorrer = True

  For Each Entrada As XElement In Tempo.<inimigo>
    Dim novoInimigo As New Inimigo
    Dim pY As Integer
    Select Case Entrada.@corredor.ToLower
      Case "superior" : pY = 140
      Case "central" : pY = 260
      Case "inferior" : pY = 380
     End Select

   Dim tmpTipo As String = Entrada.@tipo
    Select Case tmpTipo
     Case "jipe"
           novoInimigo.Grafico = inimigo_jipe
     Case "apc"
           novoInimigo.Grafico = inimigo_apc
     Case "tanque"
```

```
            novoInimigo.Grafico = inimigo_tanque
        Case "buggy"
            novoInimigo.Grafico = inimigo_buggy
        Case "carro-bomba"
            novoInimigo.Grafico = inimigo_carrobomba
        End Select

Dim pX As Integer = novoInimigo.Grafico.Width * -1

Dim dados_inimigo As XElement = _
    baseDados_Inimigos.<inimigos>.<inimigo>.Where _
                (Function(i) i.@tipo = tmpTipo)(0)

    novoInimigo.Posicao = New Vector2(pX, pY)
    novoInimigo.Vida = Integer.Parse _
           (dados_inimigo.<vida>.Value)
    novoInimigo.VidaMaxima = Integer.Parse _
           (dados_inimigo.<vida>.Value)
    novoInimigo.Velocidade = Integer.Parse _
           (dados_inimigo.<velocidade>.Value)
    novoInimigo.Recompensa = Integer.Parse _
           (dados_inimigo.<recompensa>.Value)
    novoInimigo.Dano = Integer.Parse _
           (dados_inimigo.<dano>.Value)

  For Each Roda As XElement In dados_inimigo.<roda>
   novoInimigo.PosicaoRodas.Add(New Vector2 _
   (Single.Parse(Roda.@x), Single.Parse(Roda.@y)))
  Next
        objectosJogo.Add(novoInimigo)
  Next
End Sub
```

Para além disto, vamos acrescentar também o método.

```
Private Sub DeterminarLimitesNiveis() _
        DadosDeNivel.Clear()
        For Each xNivel As XElement In _
            baseDados_Tempos.<tempos>.<nivel>
            DadosDeNivel.Add(Integer.Parse _
            (xNivel.@id), xNivel.<onda>.Count())
        Next
End Sub
```

Este método vai ajudar a determinar os limites do jogo, na inicialização.

A mecânica do método não sofreu alterações, e foi explicada anteriormente. Apenas estamos a contemplar os restantes inimigos e a adaptar o código para aceitar o comprimento do próprio inimigo, ao invés de um valor fixo.

**Colocação de torres**

A colocação de torres compreende duas partes: a torre em si em uma das posições para as torres, e a selecção da torre no interface. De acordo com a selecção de torre no interface, ao clique sobre uma das posições para as torres no terreno, é colocada uma torre desse tipo, descontando o custo do jogador.

```
Public Sub ColocarTorre(X As Integer, Y As Integer)
Dim id_seleccionado = DirectCast
   Me.seleccao_torreui, TorreUI).ID

Dim Torre As XElement = _
 baseDados_Torres.<torres>.<torre>.Where _
 (Function(t) t.@id = id_seleccionado.ToString)(0)

 If Dinheiro < Integer.Parse(Torre.<preco>.Value)
  Then
   Dim S As New Splash(New Vector2(868, 620), 120,_
semdinheiro)
  informacoes.Add(S)
  seleccao_torreui = Nothing
 Exit Sub
End If

Dim novaTorre As New Torre
 novaTorre.Posicao = New Vector2(X, Y)
 novaTorre.Nome = Torre.<nome>.Value
 novaTorre.Alcance = Integer.Parse _
                (Torre.<alcance>.Value)
 novaTorre.Dano = Integer.Parse(Torre.<dano>.Value)
 novaTorre.Frequencia = Integer.Parse _
                (Torre.<frequencia>.Value)
 novaTorre.AnguloArma = 45

Dim tmpX, tmpY As Integer

 tmpX = Integer.Parse(Torre.<centro_rotativo>.@x)
 tmpY = Integer.Parse(Torre.<centro_rotativo>.@y)

 novaTorre.CentroRotativo = New Vector2(tmpX, tmpY)

For Each PFogo As XElement In Torre.<centro_fogo>
       novaTorre.PosicaoFogo.Add(New Vector _
            (Single.Parse(PFogo.@ang),
                  Single.Parse(PFogo.@dist)))
Next

 novaTorre.NumeroCanos=novaTorre.PosicaoFogo.Count
 novaTorre.Grafico = todos_base

 Select Case id_seleccionado
  Case 1
   novaTorre.GraficoMovel = basico_rotativo
   novaTorre.SomTiro = tiro_basico
  Case 2
   novaTorre.GraficoMovel = duplo_rotativo
   novaTorre.SomTiro = tiro_duplo
  Case 3
   novaTorre.GraficoMovel = triplo_rotativo
   novaTorre.SomTiro = tiro_triplo
  Case 4
   novaTorre.GraficoMovel = vulcao_rotativo
   novaTorre.SomTiro = tiro_vulcao
  Case 5
   novaTorre.GraficoMovel = morteiro_rotativo
   novaTorre.SomTiro = tiro_morteiro
  Case 6
   novaTorre.GraficoMovel = congelador_rotativo
   novaTorre.SomTiro = tiro_congelador
  Case 7
   novaTorre.GraficoMovel = inferno_rotativo
   novaTorre.SomTiro = tiro_inferno
  Case 8
   novaTorre.GraficoMovel = apocalipse_rotativo
   novaTorre.SomTiro = tiro_apocalipse
  End Select
   Dinheiro = Integer.Parse(Torre.<preco>.Value) _
                        objectosJogo.Add(novaTorre)
    seleccao_torreui = Nothing
End Sub
```

A primeira verificação a fazer é se o jogador possuí recursos suficientes para colocar a torre. Se não possuir, será apresentado um símbolo no canto inferior direito, que sugere que não existem recursos suficientes. Se possuir, é preparada uma instância da torre para colocar na posição: determinam-se as características da torre, os sprites e os sons envolvidos, desconta-se o custo e coloca-se a instância na colecção

de objectos do jogo. A partir do momento em que a instância é colocada na colecção de objectos do jogo, passa a ser utilizada no ciclo imediatamente a seguir.

Neste ficheiro vamos também colocar o seguinte método:

```
Public Sub AdicionarTorresUI()
 For Each Torre As XElement In _
      baseDados_Torres.<torres>.<torre>
  Dim TUI As New TorreUI

  Select Case Integer.Parse(Torre.@id)
   Case 1 : TUI.Grafico = torre1_UI : _
     TUI.Posicao = New Vector2(370, 707)
   Case 2 : TUI.Grafico = torre2_UI : _
     TUI.Posicao = New Vector2(432, 707)
   Case 3 : TUI.Grafico = torre3_UI : _
     TUI.Posicao = New Vector2(494, 707)
   Case 4 : TUI.Grafico = torre4_UI : _
     TUI.Posicao = New Vector2(556, 707)
   Case 5 : TUI.Grafico = torre5_UI : _
     TUI.Posicao = New Vector2(618, 707)
   Case 6 : TUI.Grafico = torre6_UI : _
     TUI.Posicao = New Vector2(680, 707)
   Case 7 : TUI.Grafico = torre7_UI : _
     TUI.Posicao = New Vector2(742, 707)
   Case 8 : TUI.Grafico = torre8_UI : _
     TUI.Posicao = New Vector2(804, 707)
  End Select

   TUI.Preco = Integer.Parse(Torre.<preco>.Value)
   TUI.ID = Integer.Parse(Torre.@id)

   objectosJogo.Add(TUI)
 Next
End Sub
```

Este método é utilizado na inicialização da fase do nível, e deve ser chamado sempre que as colecções de jogo são limpas. É responsável por adicionar os selectores das 8 torres na barra inferior, que permitem ao jogador colocar torres. A leitura é feita directamente do XML a partir da definição das torres, daí a constar deste ficheiro. As posições dos selectores no interface são pré-calculadas para 1024x768 apenas.

**Classe principal – Desenho e actualização**

Toda a lógica que não foi implementada nas classes descritas, está implementada aqui, e é aqui que reside a maior parte das alterações. Tudo o que foi implementado na primeira parte estava focado nas outras fases, que não sofreram qualquer alteração. Agora é necessário focar na fase do nível.

Antes de chegar ao desenho e actualização, são necessários alguns membros adicionais na classe:

```
Private frame As Integer
    Private nivel As Integer = 1
    Private onda As Integer = 1
    Private ondaADecorrer As Boolean = False
    Private framesAtraso As Integer = 0
    Private seleccao_torreui As Objecto = Nothing
```

Frame determina o frame global, para utilizar sempre que for necessário basear acção em tempo.

O frame é aumentado a cada ciclo. Nivel e onda são usados para controlar o nível e onda actuais, e ondaADecorrer determina se o jogo iniciou ou se já existem ondas a decorrer. FramesAtraso controla o tempo que se aguarda depois de não existirem inimigos, para dar a onda como concluída. Seleccao_torreui determina qual é a torre que se encontra seleccionada no interface.

De seguida, no LoadContent, decidi carregar todos os recursos. Trata-se de um pequeno jogo e o impacto não é grande. O método completo fica assim:

```
Protected Overrides Sub LoadContent()
 spriteBatch = New SpriteBatch(GraphicsDevice)
 titulo = Me.Content.Load(Of Texture2D) _
   ("UI/titulo")
 seta = Me.Content.Load(Of Texture2D)("UI/seta")
 botao_defender = Me.Content.Load(Of Texture2D) _
    ("UI/defender")
 botao_sair = Me.Content.Load(Of Texture2D) _
    ("UI/sair")
 botao_defender_p = Me.Content.Load(Of Texture2D) _
    ("UI/defender_p")
 botao_sair_p = Me.Content.Load(Of Texture2D) _
    ("UI/sair_p")
 baseDados_Tempos = XDocument.Load _
    ("Dados/tempos.xml")
 baseDados_Inimigos = XDocument.Load _
    ("Dados/inimigos.xml")
 baseDados_Torres = XDocument.Load _
    ("Dados/torres.xml")
 fundoJogo = Me.Content.Load(Of Texture2D) _
    ("Jogo/chao")
 fundoEscolha = Me.Content.Load(Of Texture2D) _
    ("UI/Nivel/fundoescolha")
 inimigo_jipe = Me.Content.Load(Of Texture2D) _
    ("Jogo/Objectos/Inimigos/jipe")
 inimigo_apc = Me.Content.Load(Of Texture2D) _
    ("Jogo/Objectos/Inimigos/apc")
 inimigo_tanque = Me.Content.Load(Of Texture2D) _
    ("Jogo/Objectos/Inimigos/tanque")
 inimigo_carrobomba = Me.Content.Load _
 (Of Texture2D)_
  ("Jogo/Objectos/Inimigos/carrobomba")
 inimigo_buggy = Me.Content.Load(Of Texture2D) _
  ("Jogo/Objectos/Inimigos/buggy")
 botao_ora_bolas = Me.Content.Load(Of Texture2D) _
  ("UI/orabolas")
 botao_ora_bolas_p = Me.Content.Load(Of Texture2D)
  ("UI/orabolas_p")
 gameover = Me.Content.Load(Of Texture2D) _
  ("UI/gameover")
 ganhou = Me.Content.Load(Of Texture2D) _
  ("UI/ganhou")
 torreui_selector = Me.Content.Load _
  (Of Texture2D)("UI/Nivel/torreui_selector")
 terreno_selector = Me.Content.Load(Of Texture2D)_
  ("UI/Nivel/terreno_selector")
  sacodinheiro = Me.Content.Load(Of Texture2D)_
  ("UI/Nivel/sacodinheiro")
  semdinheiro = Me.Content.Load(Of Texture2D)_
   ("UI/Nivel/semdinheiro")
  tiro_basico = Me.Content.Load(Of SoundEffect)_
   ("Jogo/Objectos/Torres/Sons/vulcao")
  tiro_duplo = Me.Content.Load(Of SoundEffect) _
   ("Jogo/Objectos/Torres/Sons/vulcao")
  tiro_triplo = Me.Content.Load(Of SoundEffect)_
   ("Jogo/Objectos/Torres/Sons/vulcao")
  tiro_vulcao = Me.Content.Load(Of SoundEffect) _
   ("Jogo/Objectos/Torres/Sons/vulcao")
  tiro_morteiro = Me.Content.Load(Of SoundEffect)_
   ("Jogo/Objectos/Torres/Sons/morteiro")
```

```
 tiro_congelador = Me.Content.Load(Of SoundEffect)_
  ("Jogo/Objectos/Torres/Sons/congelador")
  tiro_inferno = Me.Content.Load(Of SoundEffect)_
   ("Jogo/Objectos/Torres/Sons/inferno")
tiro_apocalipse = Me.Content.Load(Of SoundEffect) _
 ("Jogo/Objectos/Torres/Sons/inferno")
explosao_inimigo = Me.Content.Load(Of SoundEffect)_
 ("Jogo/SFX/explosao")
torre1_UI = Me.Content.Load(Of Texture2D)_
  ("UI/Nivel/Torres/basico")
torre2_UI = Me.Content.Load(Of Texture2D)_
   ("UI/Nivel/Torres/duplo")
torre3_UI = Me.Content.Load(Of Texture2D)_
   ("UI/Nivel/Torres/triplo")
torre4_UI = Me.Content.Load(Of Texture2D)_
   ("UI/Nivel/Torres/vulcao")
torre5_UI = Me.Content.Load(Of Texture2D)_
   ("UI/Nivel/Torres/morteiro")
torre6_UI = Me.Content.Load(Of Texture2D)_
   ("UI/Nivel/Torres/congelador")
torre7_UI = Me.Content.Load(Of Texture2D)_
   ("UI/Nivel/Torres/inferno")
torre8_UI = Me.Content.Load(Of Texture2D)_
    ("UI/Nivel/Torres/apocalipse")
todos_base = Me.Content.Load(Of Texture2D)_
    ("Jogo/Objectos/Torres/base")
basico_rotativo = Me.Content.Load(Of Texture2D)_
     ("Jogo/Objectos/Torres/Rotativos/basico")
duplo_rotativo = Me.Content.Load(Of Texture2D)_
     ("Jogo/Objectos/Torres/Rotativos/duplo")
triplo_rotativo = Me.Content.Load(Of Texture2D)_
     ("Jogo/Objectos/Torres/Rotativos/triplo")
vulcao_rotativo = Me.Content.Load(Of Texture2D) _
     ("Jogo/Objectos/Torres/Rotativos/vulcao")
morteiro_rotativo = Me.Content.Load(Of Texture2D)_
     ("Jogo/Objectos/Torres/Rotativos/morteiro")
congelador_rotativo = _
     Me.Content.Load(Of Texture2D)_
     ("Jogo/Objectos/Torres/Rotativos/congelador")
inferno_rotativo = Me.Content.Load(Of Texture2D)_
     ("Jogo/Objectos/Torres/Rotativos/inferno")
apocalipse_rotativo = _
     Me.Content.Load(Of Texture2D) _
     ("Jogo/Objectos/Torres/Rotativos/apocalipse")
barra_vida = Me.Content.Load(Of Texture2D)_
     ("UI/Nivel/barra_vida")
poeira = Me.Content.Load(Of Texture2D)_
     ("Jogo/Particulas/poeira")
impacto = Me.Content.Load(Of Texture2D)_
     ("Jogo/Particulas/impacto")
tiro = Me.Content.Load(Of Texture2D)_
     ("Jogo/Particulas/tiro")
impacto_azul = Me.Content.Load(Of Texture2D)_
     ("Jogo/Particulas/impacto_azul")
explosao = Me.Content.Load(Of Texture2D)_
     ("Jogo/Particulas/explosao")
chamusca = Me.Content.Load(Of Texture2D)_
     ("Jogo/Particulas/chamusca")

PosicoesPossiveis.Add(New PosicaoTorre _
    (New Rectangle(15, 600, 86, 86)))
PosicoesPossiveis.Add(New PosicaoTorre _
    (New Rectangle(115, 600, 86, 86)))
PosicoesPossiveis.Add(New PosicaoTorre _
    (New Rectangle(215, 600, 86, 86)))
PosicoesPossiveis.Add(New PosicaoTorre_
    (New Rectangle(315, 600, 86, 86)))
PosicoesPossiveis.Add(New PosicaoTorre_
     (New Rectangle(415, 600, 86, 86)))
PosicoesPossiveis.Add(New PosicaoTorre_
     (New Rectangle(515, 600, 86, 86)))
PosicoesPossiveis.Add(New PosicaoTorre_
     (New Rectangle(615, 600, 86, 86)))
PosicoesPossiveis.Add(New PosicaoTorre
     (New Rectangle(715, 600, 86, 86)))
PosicoesPossiveis.Add(New PosicaoTorre_
     (New Rectangle(815, 600, 86, 86)))
PosicoesPossiveis.Add(New PosicaoTorre_
  (New Rectangle(915, 600, 86, 86)))
PosicoesPossiveis.Add(New PosicaoTorre_
  (New Rectangle(15, 15, 86, 86)))
PosicoesPossiveis.Add(New PosicaoTorre_
  (New Rectangle(115, 15, 86, 86)))
PosicoesPossiveis.Add(New PosicaoTorre_
  (New Rectangle(215, 15, 86, 86)))
PosicoesPossiveis.Add(New PosicaoTorre_
  (New Rectangle(315, 15, 86, 86)))
PosicoesPossiveis.Add(New PosicaoTorre_
  (New Rectangle(415, 15, 86, 86)))
PosicoesPossiveis.Add(New PosicaoTorre_
  (New Rectangle(515, 15, 86, 86)))
PosicoesPossiveis.Add(New PosicaoTorre_
  (New Rectangle(615, 15, 86, 86)))
PosicoesPossiveis.Add(New PosicaoTorre_
  (New Rectangle(715, 15, 86, 86)))
PosicoesPossiveis.Add(New PosicaoTorre_
  (New Rectangle(815, 15, 86, 86)))
PosicoesPossiveis.Add(New PosicaoTorre_
   (New Rectangle(915, 15, 86, 86)))
FonteGeral = Content.Load(Of SpriteFont)_
   ("UI/Verdana")
FonteGrande = Content.Load(Of SpriteFont)_
   ("UI/grandes")
Rect_Defender = New Rectangle(CInt _
   (512 - 301 / 2), 450, 301, 71)
Rect_Sair = New Rectangle(CInt _
   (512 - 301 / 2), 528, 301, 71)
Rect_OraBolas = New Rectangle(5, 680, 301, 71)
DeterminarLimitesNiveis()
End Sub
```

No bloco que detecta o clique no botão Defender, onde anteriormente tínhamos apenas a troca de fase para o Nível, teremos de colocar alguma inicialização para que possamos limpar todos os valores ao recomeçar o jogo:

```
fase = FasesJogo.Nivel
nivel = 1
onda = 1
Dinheiro = 50000
VidaRestante = 500
ondaADecorrer = False
framesAtraso = 0
frame = 0
seleccao_torreui = Nothing

objectosJogo.Clear()
particulas_sobrepostas.Clear()
particulas.Clear()
particulasAnimadas.Clear()
informacoes.Clear()

AdicionarTorresUI()

Dim texto As String = "Nivel " & nivel & " - " &
"Onda " & onda
Dim tmpTamanho As Vector2 = FonteGran-
de.MeasureString(texto)
Dim S As New Splash(New Vector2(CSng(512 -
((tmpTamanho.X * 1.5) / 2)), 350), 120, texto)
informacoes.Add(S)
Exit Select
```

Por fim, vamos implementar toda a lógica em falta na fase do nível. Comecemos por o Update:

## XNA: ATAQUE NO QUINTAL (PARTE 2/2)

```
For Each tmpI As Inimigo In objectosJogo.Where _
(Function(o) TypeOf o Is Inimigo)
 tmpI.Posicao.X += CSng(tmpI.Velocidade * _
  gameTime.ElapsedGameTime.TotalSeconds)
    If frame Mod 5 = 0 Then
     For Each Roda As Vector2 In tmpI.PosicaoRodas
      Dim novaParticula As New Particula(poeira, _
        0.01, 1.2, RndNum(0.001D, 0.002D), 0.1)
      novaParticula.Posicao = tmpI.Posicao
      novaParticula.Posicao.X += Roda.X
      novaParticula.Posicao.Y += Roda.Y
      particulas.Add(novaParticula)
   Next
  End If

 Next
```

Do que foi escrito na primeira parte, o ciclo For para os inimigos pode ser eliminado e substituído por este.

Como agora temos mais objectos do que na primeira parte, e estes objectos interagem uns com os outros, é importante separar as iterações para que se possa definir uma ordem de desenho. Por exemplo, com o exemplo anterior se um inimigo fosse criado depois de uma cratera criada por um anterior, este passaria por baixo da cratera, o que não está correcto. Assim, separamos os objectos com várias iterações filtradas, à mesma colecção.

Por cada inimigo, actualizamos a sua posição horizontal de acordo com a sua velocidade, o que o faz descolar para a direita, neste caso, e para além disso, com uma frequência de 5 ciclos, criamos uma instância de Partícula, por cada roda, com o sprite poeira. Os parâmetros para esta partícula fazem com que esta cresça e desvaneça, tal como uma nuvem de poeira, e como o veículo está sempre em movimento, criamos uma a cada 5 ciclos por debaixo de cada roda. Isto cria a ilusão de que as rodas estão a levantar poeira, mantendo um rasto aparentemente contínuo ao ritmo de 12 nuvens de poeira em cada roda, ao segundo.

O próximo bloco, logo no seguimento, trata das torres.

```
For Each tmpT As Torre In objectosJogo.Where _
    (Function(o) TypeOf o Is Torre)
  tmpT.DispararAAlvo()
  If tmpT.Alvo Is Nothing Then
   For Each tInimigo As Objecto In _
    objectosJogo.Where(Function(o) TypeOf o _
                                  Is Inimigo)
      Dim tIn As Inimigo = DirectCast(tInimigo, _
                                      Inimigo)
      Dim X1 As Double = tmpT.Centro.X
      Dim X2 As Double = tIn.Centro.X
      Dim Y1 As Double = tmpT.Centro.Y
      Dim Y2 As Double = tIn.Centro.Y
      Dim DistanciaTorreInimigo As Double = _
      Math.Sqrt((Math.Pow(X2 - X1, 2)) + _
      (Math.Pow(Y2 - Y1, 2)))
      If tmpT.Alcance > DistanciaTorreInimigo _
       ThentmpT.Alvo = DirectCast(tInimigo, _
                                  Inimigo)
      End If
     Next
    Else
     tmpT.ValidarAlvo()
    End If
Next
```

A cada ciclo chamamos o método DispararAAlvo para proceder aos disparos ao alvo, se alguns.

De seguida, e caso não exista um alvo designado, calculamos a distância entre a torre e cada um dos inimigos na colecção do jogo. Se for encontrado um que esteja dentro do alcance, torna-se o alvo designado para aquela torre. Se existir um alvo designado são apenas efectuadas validações sobre o mesmo através do método ValidarAlvo.

O próximo bloco, logo no seguimento, trata dos selectores das torres.

```
For Each tmpTUI As TorreUI In objectosJogo.Where _
  (Function(o) TypeOf o Is TorreUI)
 Dim M As MouseState = Mouse.GetState()
 If M.LeftButton = ButtonState.Pressed Then
  Dim rato_rect As New Rectangle(M.X, M.Y, 1, 1)
  Dim torreui_rect As New Rectangle(CInt _
   (tmpTUI.Posicao.X), CInt(tmpTUI.Posicao.Y), 57,_
      57)
   If rato_rect.Intersects(torreui_rect) Then
     seleccao_torreui = tmpTUI
   End If
  End If
Next
```

O intuito deste bloco é apenas determinar se o jogador clicou sobre algum dos selectores. Se for detectado um clique, a selecção da torre é alterada.

De forma semelhante, criamos outro bloco para detectar cliques sobre as áreas de colocação de torres.

```
For Each PosT As PosicaoTorre In PosicoesPossiveis
  Dim M As MouseState = Mouse.GetState()
  If M.LeftButton = ButtonState.Pressed Then
  Dim rato_rect As New Rectangle(M.X, M.Y, 1, 1)
  If PosT.Rectangulo.Intersects(rato_rect) Then
   If PosT.NoTerreno Is Nothing And Not _
    seleccao_torreui Is Nothing Then
    ColocarTorre(PosT.Rectangulo.X, _
    PosT.Rectangulo.Y)
   End If
  End If
 End If
Next
```

Se for detectado um clique sobre uma área de colocação, e existir uma selecção de torre no interface, coloca-se a torre.

Para actualizar todas as particulas, splashes e animações, colocamos os seguintes blocos:

```
For p As Integer = particulas.Count - 1 To 0 _
 Step -1
 Dim Part As Particula = particulas(p)
 Part.Deformar()
 If Part.Opacidade <= 0 Then
    particulas.Remove(Part)
 End If
Next
```

```
For p As Integer = _
    particulas_sobrepostas.Count - 1 To 0 Step -1
 Dim Part As Particula = particulas_sobrepostas(p)
 Part.Deformar()
 If Part.Opacidade <= 0 Then
  particulas_sobrepostas.Remove(Part)
 End If
Next

For p As Integer = particulasAnimadas.Count - 1 _
 To 0 Step -1
 Dim Part As SpriteAnimado = particulasAnimadas(p)
 Part.ComunicarFrame()
 If Part.Destruir Then
  particulasAnimadas.Remove(Part)
 End If
Next
For i As Integer = informacoes.Count - 1 _
   To 0 Step -1
   Dim Info As Splash = informacoes(i)
   Info.ComunicarFrame()
   If Info.Destruir Then
      informacoes.Remove(Info)
   End If
Next
```

Não só comunicam o frame para as respectivas deformações ou acções específicas, mas também percorrem inversamente a colecção para remover todos os objectos que estejam em condições de ser destruídos.

De forma semelhante, a actualização dos inimigos também é feita em iteração inversa.

```
Dim inimigos As IEnumerable(Of Objecto) = _
 objectosJogo.Where(Function(i) TypeOf i Is _
 Inimigo)

 For i As Integer = inimigos.Count - 1 To 0 Step -1
 Dim tmpInimigo As Objecto = inimigos(i)

 If DirectCast(tmpInimigo, Inimigo).Vida <= 0 Then
   Dinheiro += DirectCast(tmpInimigo, _
   Inimigo).Recompensa
  Dim S As New Splash(New Vector2(947, 620), 60, _
   sacodinheiro)
  informacoes.Add(S)

  Dim Chamus As New Inamovivel()
  Chamus.Grafico = chamusca
  Chamus.Posicao = DirectCast(tmpInimigo,  _
   Inimigo).Posicao
  Chamus.Angulo = RndNum(0, 360)
  objectosJogo.Add(Chamus)
  Dim explosao_som As SoundEffectInstance = _
    explosao_inimigo.CreateInstance
  explosao_som.Pitch = 0.4
  explosao_som.Play()
  objectosJogo.Remove(tmpInimigo)
  Continue For
  End If

  If DirectCast(tmpInimigo, Inimigo).Posicao.X > _
    1120 Then
   objectosJogo.Remove(tmpInimigo)
   VidaRestante -= DirectCast(tmpInimigo, _
   Inimigo).Dano
  Continue For
  End If

Next
```

O intuito deste bloco é validar as condições para a remoção dos inimigos. Se for detectado que o inimigo tem a vida a zero, é necessário contabilizar a recompensa, fazer soar a explosão e deixar o chão queimado. A recompensa é directamente contabilizada no dinheiro do jogador. A explosão faz-se soar com uma instância de SoundEffect e uma pequena alteração ao Pitch, para soar mais grave. Por fim, o chão queimado é representado por um objecto semelhante a um splash, mas com orientação.

Para criar este objecto, de nome Inamovivel, basta criar uma classe que herde do objecto e que acrescente apenas um ângulo e o respectivo método de desenho.

```
Public Class Inamovivel
    Inherits Objecto

    Public Angulo As Integer

   Public Sub Desenhar(SB As SpriteBatch)
    Dim temp_Rect As New Rectangle(CInt _
    (Me.Posicao.X) + 60, CInt(Me.Posicao.Y) + 60,_
     120, 120)
     SB.Draw(Me.Grafico, temp_Rect, Nothing, _
     Color.White, Rads(Me.Angulo), New Vector2(60,_
     60), SpriteEffects.None, 0)
    End Sub
End Class
```

Utilizam-se alguns valores fixos porque neste caso o objecto Inamovivel representa apenas o chão queimado.

Poderia ser implementado de outras formas, mas esta é menos confusa.

No final, o inimigo é removido.

Também se pode dar o caso do inimigo não ser destruído e desaparecer por o lado direito. Nesse caso, é contabilizado como dano para o jogador, e o inimigo é removido imediatamente da colecção.

Para terminar o Update, resta controlar se o jogador ainda tem vida e as informações que damos ao jogador sempre que entra um novo nível ou uma nova onda:

```
If inimigos.Count = 0 And ondaADecorrer Then
  framesAtraso += 1
   If framesAtraso >= 300 Then
     ondaADecorrer = False
     If onda = DadosDeNivel(nivel) Then
      If DadosDeNivel.Count = nivel Then
       fase = FasesJogo.Ganhou
      End If
      onda = 1
      nivel += 1
      frame = 0
      framesAtraso = 0
      Else
      onda += 1
      frame = 0
      framesAtraso = 0
     End If
    Dim texto As String = "Nivel " & nivel & " _
      - " & "Onda " & onda
    Dim tmpTamanho As Vector2 = _
     FonteGrande.MeasureString(texto)
    Dim S As New Splash(New Vector2(CSng _
```

## XNA: ATAQUE NO QUINTAL (PARTE 2/2)

```
    (512 - ((tmpTamanho.X * 1.5) / 2)), 350),_
     120, texto)
    informacoes.Add(S)
    End If
   End If
  If VidaRestante <= 0 Then
   fase = FasesJogo.GameOver
  End If
```

São efectuadas verificações sobre a colecção DadosDeNivel, previamente preenchida na inicialização, com o método DeterminarLimitesNiveis. As verificações são simples e têm como objectivo ir percorrendo a sequência de níveis e ondas por nível, informando o jogador de cada troca. Por fim, se a vida do jogador for igual ou inferior a zero…game over!

Por o lado do Draw as coisas são mais simples. É necessário apenas percorrer os objectos e chamar a lógica de desenho na ordem óptima.

Do que foi feito na primeira parte, eliminamos o bloco para iterar os objectos, e antes do bloco das posições possíveis das torres colocamos:

```
For Each P As Inamovivel In objectosJogo.Where _
 (Function(o) TypeOf o Is Inamovivel)
  P.desenhar(spriteBatch)
Next

For Each P As Particula In particulas
  P.Desenhar(spriteBatch)
Next

For Each tmpI As Inimigo In objectosJogo.Where _
  (Function(o) TypeOf o Is Inimigo)
 tmpI.Desenhar(spriteBatch)
Next

For Each tmpT As Torre In objectosJogo.Where _
 (Function(o) TypeOf o Is Torre)
 tmpT.Desenhar(spriteBatch)
Next

For Each tmpTUI As TorreUI In objectosJogo.Where
   (Function(o) TypeOf o Is TorreUI)
 Dim temp_Rect As New Rectangle(CInt _
 (tmpTUI.Posicao.X), CInt(tmpTUI.Posicao.Y), _
   57, 57)
 spriteBatch.Draw(tmpTUI.Grafico, temp_Rect, _
  Color.White)
 If seleccao_torreui Is tmpTUI Then
  spriteBatch.Draw(torreui_selector, temp_Rect, _
    Color.White)
 End If
Next

For Each P As Particula In particulas_sobrepostas
 P.Desenhar(spriteBatch)
Next
```

Iteram-se as colecções, por ordem, para todos os objectos e chama-se o método Desenhar de cada objecto. Ao transportar o spritebatch por os métodos de desenho, faz com que sejam todos adicionados ao mesmo. No caso dos selectores das torres, o desenho é feito directamente, e no caso de estar seleccionado, acrescenta-se uma imagem por cima, semi-transparente, que sugere a selecção.

Depois do bloco das posições possíveis coloca-se o seguinte:

```
spriteBatch.DrawString(FonteGeral, _
   Dinheiro.ToString & " $",
 New Vector2(1019 - FonteGeral.MeasureString _
  (Dinheiro.ToString & " $").X, 712), Color.Gold)
  spriteBatch.DrawString(FonteGeral, _
  VidaRestante.ToString, _
 New Vector2(1019 - FonteGeral.MeasureString _
  (VidaRestante.ToString).X, 732), Color.Gold)

 For Each I As Splash In informacoes
  I.Desenhar(spriteBatch)
 Next
```

Isto fará com que seja apresentado a qualquer altura o dinheiro e a vida disponíveis, do jogador. No final, itera-se a colecção de informações, que se sobrepõe a tudo.

Depois de fechar o spritebatch, e se correrem o jogo agora, vão reparar que as explosões não acontecem quando se destroí um inimigo.

Para as explosões, e também para manter um exemplo do maior número de situações, queremos que sejam desenhadas em outro spritebatch, com características diferentes do normal:

```
spriteBatch.Begin(SpriteSortMode.Deferred, _
  BlendState.Additive)
 For Each P As SpriteAnimado In particulasAnimadas
  P.Desenhar(spriteBatch)
 Next
spriteBatch.End()
```

Este spritebatch onde as particulas animadas são desenhadas tem de diferente o blend. Um blend (mistura) aditivo fará com que os canais se somem com os já existentes no buffer.

Sem o blend aditivo, sobrepor um sprite a outro fazia com que apenas se visse o último. Com blend aditivo, a sobreposição soma a cor, o que faz com que se torne mais brilhante. O resultado máximo de uma sobreposição aditiva é o branco.

No caso da explosão, para que seja brilhante, temos de a desenhar de forma aditiva.

À esquerda, o blend normal, à direita o aditivo, do mesmo sprite.

Com isto, terminamos a parte funcional do jogo.

## XNA: ATAQUE NO QUINTAL (PARTE 2/2)

Se correrem o jogo agora, já é possível jogar normalmente e tudo decorre como esperado… mas não termina por aqui.

Ainda existem duas grandes fases interlaçadas, que não vou cobrir no artigo.

Têm por a frente as fases de balanceamento e testes.

Estas fases são de extrema importância pois representam uma larga fatia no bolo do sucesso de um jogo, ao determinar a sua jogabilidade e a durabilidade.

Para o Ataque no Quintal, os ficheiros de dados do XML exteriorizam todas as variáveis necessárias para o balanceamento, e a cada novo balanço deverá suceder um teste.

O projecto que acompanhou o artigo, que podem e devem descarregar, já incluí alguns valores que permitem jogar o jogo, mas não garanto que tenham um bom balanço de preços/dano!

Espero que o artigo sirva de boa base para produzirem os jogos espectaculares que queremos ver.

Não deixem de os apresentar na comunidade!

**Download:**

http://download.sergioribeiro.com/permaloads/Canyon.zip


## AUTOR

**Escrito por Sérgio Ribeiro**

Curioso e autodidacta com uma enorme paixão por tecnologias de informação e uma saudável relação com a .NET framework. Moderador global na comunidade Portugal@Programar desde Setembro de 2009.
Alguns frutos do seu trabalho podem ser encontrados em http://www.sergioribeiro.com

# Alterações na comunidade Portugal-a-Programar

Conforme demos a entender há algum tempo atrás, o P@P decidiu mudar o software que suporta o fórum. Depois de um longo período de testes, esta mudança foi efectuada! Esta mudança coincide também com uma mudança de domínio principal do P@P, que **passa agora a ser www.portugal-a-programar.pt!**

Depois de vários anos a usar o SMF, o P@P **passa agora a usar o IPBoard**, que para além do fórum, também inclui o blog e uma nova plataforma de downloads, que esperamos que venha a ser bastante útil aos utilizadores da nossa comunidade, promovendo a partilha de recursos.

Para além de várias melhorias ao nível de administração e moderação, o novo software trará também algumas **melhorias para os utilizadores**, das quais destacamos:

- Tema para dispositivos móveis;
- Sistema de tags bastante funcional;
- Integração com redes sociais;
- Melhor sistema de pesquisa e de visualização de novos conteúdos;
- Sistema de notificações bastante mais funcional;
- Novas funcionalidades ao nível de mensagens privadas;
- Sistema de avaliação de tópicos e de mensagens.

Temos consciência que nem todos utilizadores vão gostar da mudança, e que muito irão estranhar esta nova plataforma, mas acreditamos que a longo prazo a generalidade dos utilizadores irá reconhecer as vantagens desta nova opção.

Sublinhamos também que uma mudança destas é um processo complexo, sujeito a erros. Apesar de todos os cuidados na preparação desta migração, é natural que notem alguns problemas, sobretudo nestes primeiros dias, até porque há ainda algumas arestas a limar nos próximos dia/semanas. Adicionalmente, os utilizadores devem rever as suas configurações, através do Painel de Controlo, de modo a garantir que tiram partido de todas as funcionalidades oferecidas pelo IPBoard, ou para corrigir alguma configuração que não tenha sido correctamente migrada.

O **Portal de Downloads** da comunidade Portugal-a-Programar destina-se a promover a partilha de conteúdos relativos à programação e áreas similares, que podem incluir materiais usados por estudantes nos seus ciclos de estudos (e.g., apontamentos, exercícios, testes), ou outros materiais desenvolvidos pelos utilizadores que podem ser úteis a outros utilizadores (e.g., tutoriais, scripts, bibliotecas, aplicações). Tem por objectivo criar um repositório de conteúdos, que permite a sua melhor organização e catalogação, facilitando assim o trabalho de pesquisa dos utilizadores.

Qualquer utilizador registado pode submeter conteúdos, desde que sejam da vossa autoria, ou que o autor tenha decidido disponibilizá-los publicamente. É possível submeter ficheiros directamente, ou indicar um link para um ficheiro externo. (Caso seja o autor dos ficheiros, é preferível enviar directamente o ficheiro, para evitar links quebrados. Caso não seja o autor, pode indicar o link onde o autor disponibiliza os conteúdos. Por favor não use links para sites como Rapidshare e afins.)

Para submeter ficheiros, devem escolher uma categoria apropriada, seleccionar os ficheiros a enviar (ou indicar os links para os ficheiros), e fornecer a seguinte informação:

- **Nome do Ficheiro** (obrigatório): deve permitir aos utilizadores ter uma ideia do conteúdo dos ficheiros, nomeadamente do tema a que dizem respeito (e.g., Exercícios de Java, Slides sobre Algoritmos de Ordenação, Apontamentos de Teoria de Números, Exercícios de Criptografia). Se forem materiais de uma cadeira, também têm que identificar pelo menos a instituição de ensino, e opcionalmente a cadeira, curso (podem usar acrónimos, para o nome não ficar demasiado grande).
- **Versão do Ficheiro** (opcional): identifica a versão do ficheiro. No caso de bibliotecas/código pode ser a versão (e.g. 1.0, 2.1) e no caso de ficheiros como tutoriais, exercícios e afins, que não tenham versão, pode ser a data (e.g. 2011/12).
- **Tags do Ficheiro** (obrigatório): palavras-chave que descrevem os conteúdos submetidos (e.g., c, pascal, bases de dados, algoritmos de ordenação).
- **Change Log** (opcional): registo de alterações introduzidas numa nova versão.
- **Descrição** (obrigatório): devem dar os detalhes dos conteúdos aqui. Deve conter um pequeno parágrafo com um resumo dos conteúdos. Adicionalmente, nos

casos de materiais de uma cadeira, também devem indicar a cadeira, curso e instituição. Caso estejam a submeter conteúdos que não são da vossa autoria, é também conveniente que indiquem o autor.

- **Licença** (opcional): licença dos conteúdos. Se os conteúdos forem da vossa autoria, indiquem a licença que quiserem. Se não forem, devem respeitar a licença que o autor escolheu (se não a souberem, deixem em branco).
- **Submetido pelo Autor** (obrigatório): indica se o utilizador que submeteu o ficheiro é o autor dos conteúdos.
- **Website** (opcional): pode ser um website sobre os conteúdos (ou onde estes também possam ser descarregados), ou o website do autor, por exemplo.

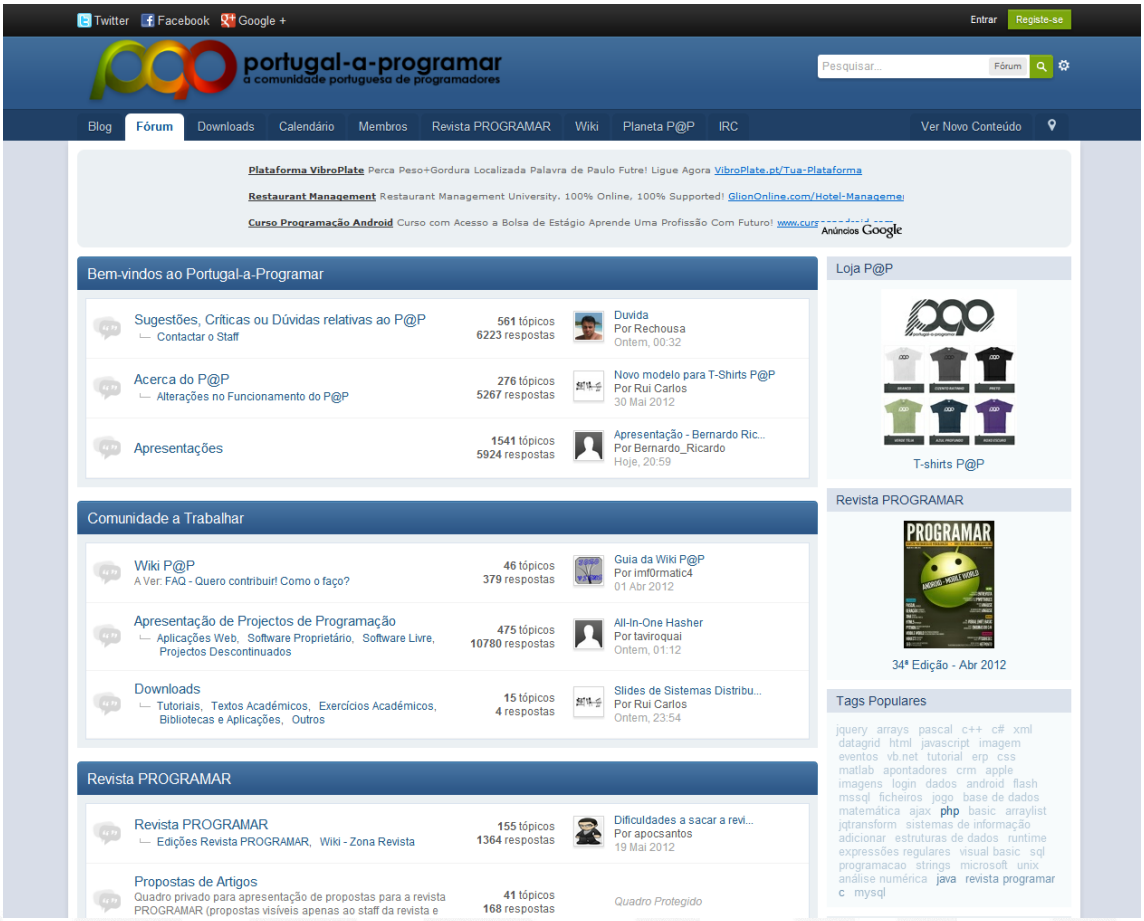
Sempre que possível, comprima os ficheiros antes de os enviar. Em particular no caso de submeter mais do que um ficheiro, deve juntá-los num único arquivo comprimido. Desta forma permite que se poupe espaço e largura de banda. (Caso não o faça, o staff reserva-se no direito de comprimir os ficheiros, sempre que tal pareça adequado.)

Os ficheiros estão sujeitos a moderação, e só ficarão disponíveis publicamente depois de revistos pelo staff. Esta área está sujeita as regras do P@P, que promovem o respeito pelos direitos de autor. Contudo, nem sempre será possível obter informação precisa/correcta sobre os direitos de autor dos conteúdos. Assim, caso encontre no nosso portal conteúdos seus que não autoriza a que sejam disponibilizados, por favor use a link "Denunciar Ficheiro" para notificar o staff de tal situação.

A moderação destina-se também a garantir que os conteúdos correspondem à descrição (embora tal não seja possível garantir totalmente para ficheiros externos), ou que a descrição fornece um nível de informação adequado, e também para normalizar a informação disponibilizada.

Depois de o ficheiro ser submetido, será criado automaticamente um tópico no fórum, onde os conteúdos poderão ser discutidos. Na página do ficheiro irá existir um link para o tópico. (Caso já exista um tópico no fórum a abordar o conteúdo, por favor notifique o staff de tal situação, para evitar a duplicação de discussões.)

O link "Denunciar Ficheiro" pode também ser usado para denunciar links quebrados, ou ficheiros que representem riscos para utilizadores (e.g., malware). O staff procurará analisar as denúncias o mais rapidamente possível.



# ENIGMAS DO C# - O estranho caso dos enumerados

**(continuação da página 20)**

**Resultado**

```
 1: value = Zero, type = SomeEnum
 2: value = Zero, type = SomeEnum
 3: value = Zero, type = SomeEnum
 4: value = Zero, type = SomeEnum
 5: value = Zero, type = SomeEnum
 6: value = Zero, type = SomeEnum
 7: value = Zero, type = SomeEnum
 8: value =    0, type =    Int32
 9: value =    0, type =    Int64
10: value =    0, type =   Double
11: value =    0, type =   Single
12: value =    0, type =  Decimal
13: value =    0, type =    Int32
```

**Explicação**

Segundo a especificação do **C#** (§6.1.3), existe uma conversão implícita de qualquer literal decimal inteiro **0** para qualquer tipo de enumerado ou qualquer tipo *nullable* cujo tipo subjacente seja um enumerado. A especificação define também (§7.5.3) que, na resolução da sobreposição de métodos (*method overloading*), o que tiver o parâmetro com o tipo mais específico é o melhor.

Isto explica os casos 1, 2 e 6.

No entanto, em Março de 2006 foi introduzida uma falha (*bug*) no compilador que faz com que qualquer zero seja implicitamente convertível para qualquer tipo de enumerado ou qualquer tipo *nullable* cujo tipo subjacente seja um enumerado.

Isto explica os casos 3, 4, 5 e 7.

A razão para permitir, por desenho, que qualquer literal zero seja convertível em qualquer enumerado é porque (quer esteja ou não marcado com o atributo Flags) é possível (e até mesmo comum) que uma variável de um tipo enumerado seja zero. Por exemplo:

```
enum DiasDaSemana
{
    Segunda = 1,
    Terça = 2,
    Quarta = 3,
    Quinta = 4,
    Sexta = 5,
    Sábado = 6,
    Doming = 7
}
// ...
class C
{
    public DiasDaSemana Dia;
}
// ...
        switch (c.Day)
        {
            case 0:
                Debug.Fail(
                   "Dia não inicializado.");
                break;
            case DiasDaSemana.Segunda:
            case DiasDaSemana.Terça:
            case DiasDaSemana.Quarta:
            case DiasDaSemana.Quinta:
            case DiasDaSemana.Sexta:
            case DiasDaSemana.Sábado:
            case DiasDaSemana.Doming:
                //...
                break;
            default:
                Debug.Fail("Dia inválido.");
                break;
        }
```

A falha introduzida foi classificada como tal porque o compilador não se está a comportar como especificado. No entanto, foi decidido manter a falha por se considerar benéfica.

**Conclusão**

Verifica-se, então, que é necessário especial cuidado quando se lida com uma API com métodos sobrepostos quando num dos casos o parâmetro é de um tipo enumerado (ou qualquer tipo *nullable* cujo tipo subjacente seja um enumerado).

A partir do C# 4.0, a utilização de argumentos com nome pode ajudar a mitigar eventuais problemas.

**Ligações**

C# Reference
The Root Of All Evil, Part One
The Root of All Evil, Part TwoClasse FlagsAttribute
Argumentos com Nome e Opcionais

## AUTOR

**Escrito por Paulo Morgado**
É licenciado em Engenharia Electrónica e Telecomunicações (Sistemas Digitais) pelo Instituto Superior de Engenharia de Lisboa e Licenciado em Engenharia Informática pela Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa. Pelo seu contributo para a comunidade de desenvolvimento em .NET em língua Portuguesa, a Microsoft premeia-o com o prémio MVP (C#) desde 2003. É ainda co-autor do livro “LINQ Com C#” da FCA.

# Media Partners da Revista PROGRAMAR









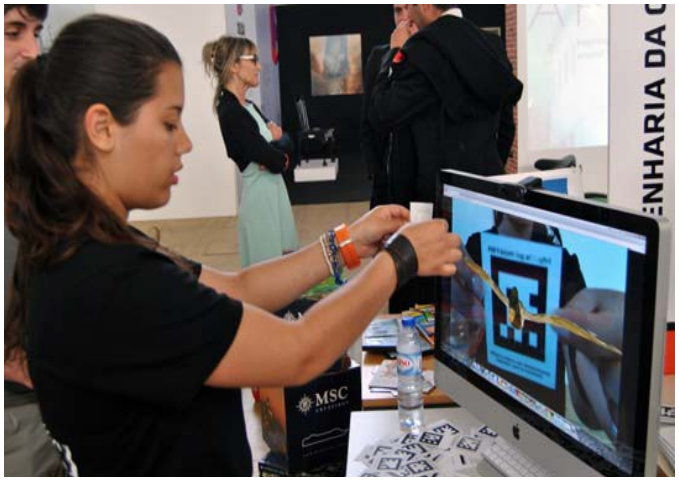



EVENTOS


Nos dias 16 e 17 de Maio teve lugar, no pavilhão AI Minho de Viana do Castelo, a Mostra IPVC 2012. Esta teve como principal objectivo divulgar todos os cursos do Instituto Politécnico de Viana do Castelo, desde CET´s (Cursos de Especialização Tecnológica), licenciaturas e mestrados.

A Mostra contou com a visita de inúmeras escolas secundárias e profissionais, estando também aberta ao público. Ao longo destes dois dias, os visitantes tiveram a oportunidade de conhecer e esclarecer dúvidas sobre os métodos de ingresso e os conteúdos programáticos das várias áreas.

O Instituto Politécnico de Viana do Castelo tem uma vasta oferta formativa na área das informáticas com grande envolvimento de programação como é o caso do CET TPSI (Tecnologias e Programação de Sistemas de Informação), CET DPM (Desenvolvimento de Produtos Multimédia); Engenharia Informática; Engenharia Electrónica e Engenharia-Computação Gráfica. Foram apresentados alguns projectos desenvolvidos pelo curso de Engenharia de Computação Gráfica e Multimédia, dos quais se destacaram, um jogo que utilizava o Kinect da Microsoft, com a SDK, foi desenvolvido em C#, utilizando Windows Presentation Foundation e JavaScript Object Notation e uma aplicação em Flash que utilizava realidade aumentada.


Como não poderia deixar de ser, foi possível assistir à actuação da tuna, referência importante no mundo académico, bem como contactar com algumas experiências dos alunos do programa *Erasmus*, que frequentam actualmente este Instituto Politécnico.

Quando questionados sobre esta iniciativa as opiniões foram consensuais como referiu um dos alunos participantes, que esteve presente neste evento: “esta mostra foi importante para dar a conhecer ao exterior a variedade de cursos do politécnico, incentivando novos alunos a ingressarem nos mesmos”.

Os visitantes mostraram-se interessados e entusiasmados com esta iniciativa, pois “permitiu conhecer melhor as várias valências do politécnico e ajudou na decisão quanto à área a seguir no Ensino Superior”, declarou um dos alunos visitantes.

Esta iniciativa mostrou-se uma mais-valia, permitindo um maior conhecimento do que é desenvolvido no interior do IPVC, aliciando ao ingresso de novos alunos no próximo ano letivo.

Os Cursos de Especialização Tecnológica, de nível superior, possibilitam por um lado uma carreira profissional e por outro o ingresso numa licenciatura, na área. O curso de TPSI (Técnico de Programação de Sistemas Informáticos), tem como objectivo formar profissionais com capacidades para revolucionar o mundo da programação. Fazendo-se representar por alunos e docentes, sempre prontos a esclarecer qualquer questão sobre o curso. Os trabalhos realizados pelos alunos, repletos de qualidade, e com enorme capacidade de inovação demonstram a evolução ao longo das várias edições do CET de TPSI.


A turma de TPSI 2011/2012 agradece ao Prf. Dr. Salvador Lima, Prof. Ricardo Castro, Prof. Bruno Gomes e demais docentes todo o seu apoio e dedicação.


## AUTOR

**Escrito por Vitor Sousa**

Frequenta o CET Técnico de Programação de Sistemas de Informação na Escola Superior de Tecnologia e Gestão do Instituto Politécnico de Viana do Castelo. Membro de uma equipa de Projecto de desenvolvimento de Software, sendo Técnico de Electrónica de Equipamentos, tendo participado no desenvolvimento e concepção de Aranha Robótica desenvolvida de Raiz por um Grupo de Trabalho multidisciplinar. https://www.facebook.com/hugo.viana.169

**Escrito por Ana Luísa Machado**

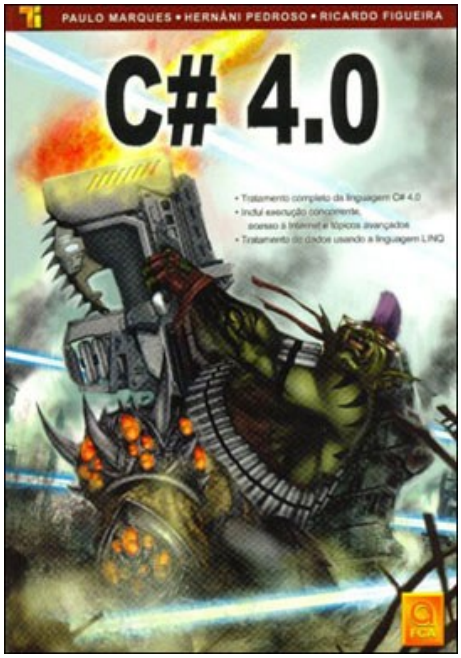
Frequenta o CET Técnico de Programação de Sistemas de Informação na Escola Superior de Tecnologia e Gestão do Instituto Politécnico de Viana do Castelo. É entusiasta das tecnologias de informação, tendo ingressado no curso de Programação no ano lectivo 2011/2012. Neste momento é membro de diversas equipes de trabalho, em projectos de desenvolvimento de software.

# Análises

C# 4.0

# C# 4.0

**Título:** C# 4.0
**Autor:** Paulo Marques / Hernâni Pedroso / Ricardo Figueira
**Editora:** FCA
**Páginas:** 472
**ISBN:** 978-972-722-703-7




Quando vislumbrei o livro pela primeira vez, o meu primeiro pensamento foi que houve um qualquer tipo de engano, tinha-me sido entregue o livro errado!

Uma reacção que pode ser natural a qualquer leitor acostumado a identificar facilmente os livros de caris técnico pela objectividade que o seu conteúdo transcende para a capa, como acontece por exemplo com o livro C# 2.0.

O choque inicial, foi depois transformado numa agradável surpresa, assim que me foi permitido desfolhar as primeiras páginas do livro.

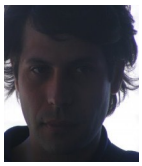
> **“… não se destina apenas a quem se encontra numa fase de aprendizagem … é também um excelente livro para os profissionais de Tecnologias de Informação”**

Concluí que este não se destina apenas a quem se encontra numa fase de aprendizagem e necessita uma base sólida de conhecimento para incrementar a curva de aprendizagem, é também um excelente livro para os profissionais de Tecnologias de Informação, que de quando em vez se deparam com situações pouco ortodoxas face ao seu dia-a-dia e necessitam de um formato físico com informação bem organizada, estruturada e credível para validar as suas decisões, ou pode ser utilizado retirar um qualquer tipo de dúvida.

Considero que cerca de metade dos 12 seus capítulos, entre eles temas como genéricos, tipos dinâmicos ou directivas de pré-processamento, abordam assuntos que são menos convencionais e podem ajudar um estudante a compreender de forma mais aprofundada a linguagem e plataforma, bem como podem motivar um qualquer profissional de IT a actualizar ou refrescar os conhecimentos adquiridos sobre C# e .NET, que muitas vezes não têm oportunidade de colocar em prática pelos mais variados motivos.

Uma excelente evolução face aos livros anteriormente publicados pelos mesmos autores.

## AUTOR



**Escrito por Bruno Pires**
Profissional de TI desde 2008, com experiência nas áreas da banca e televisão digital, onde ganhou competências nas mais várias tecnologias. Membro da Comunidade NetPonto (http://netponto.org) e autor do blog http://blog.blastersystems.com.

Twitter: @brunoacpires
Web: http://www.brunopires.me

# COMUNIDADES

AzurePT- Windows Azure Security

PtCoreSec–Segurança na WEB (Parte 2)

# Windows Azure Security

**Introdução**

Muito se tem falado sobre *Cloud Computing*, que cada vez mais é uma certeza e menos um simples *Hype*, mas mesmo assim muitas questões surgem quando se fala deste assunto. Questões como: Onde estão os meus dados localizados? Qual a segurança do meu fornecedor de serviços de Cloud? Quem tem acesso aos meus dados? Como é que posso ter os meus dados de volta? Posso colocar dados sensíveis na *Cloud*? O que acontece se quiser colocar uma solução na *Cloud* que necessite de estar conforme um dos standards como HIPPA, PCI ou outro?

Muitas outras questões poderão surgir, mas estas são apenas algumas delas e que necessitam de ser respondidas para que possamos avançar no processo de aquisição e desenvolvimento de soluções na *Cloud*.

Um dos aspectos muito importantes na segurança é compreender que "não é suficiente comprar o melhor cofre do mundo é necessário também saber utilizá-lo e mais do que tudo fechá-lo, porque de nada serve um cofre aberto". Tendo em conta todos estes elementos é importantíssimo compreender o facto de que a segurança é multidimensional e essas dimensões são as seguintes:

| | |
|---|---|
| Humana | Como tratar dados sensíveis? |
| Dados | Como encriptar os dados? Como gerir as permissões de acesso aos dados? Como melhorar a segurança da própria base de dados? |
| Aplicação | Boas práticas segurança no desenho e implementação |
| Anfitrião | Como melhorar a segurança do Sistema Operativo? Actualizações regulares do Sistema Operativo. |
| Rede | Firewall, VLANs, Canais Seguros... |
| Físico | Quem pode aceder aos meus servidores? |

Sendo o Windows Azure uma plataforma de *Cloud* é indispensável que analisemos a forma como a Microsoft garante a segurança de todos os dados que se encontram nos seus servidores. Para isso vamos olhar em maior detalhe o Modelo de Segurança da Plataforma *Windows Azure*.

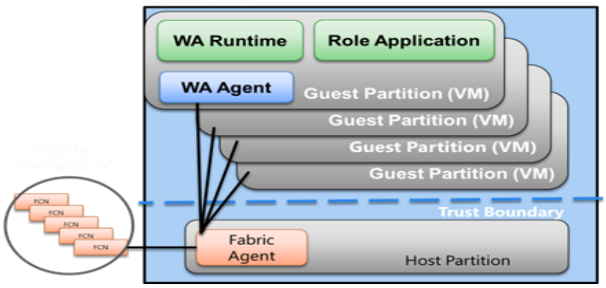
**Modelo de Segurança da Plataforma *Windows Azure***

A Plataforma *Windows Azure* contém desde logo diversos tipos de defesas dependendo da camada que estivermos a analisar.

| Camada | Defesas |
|---|---|
| Dados | Controlo de acessos com base em chaves de segurança de 512 bits<br>SSL em todas as transferências de dados efectuadas em Windows Azure |
| Aplicações | Utilização de "*Partial Trust*" para a execução de código<br>Utilização de contas de Windows com o mínimo de privilégios possíveis |
| Anfitrião | Baseado no sistema operativo Windows Server 2008<br>Segurança reforçada com base no *hypervisor* externo |
| Rede | Limitação de tráfego efectuado pela *Firewall* do sistema anfitrião<br>Utilização de redes virtuais privadas (VLANs) e filtragem de pacotes de comunicação (Packet Filters) nos routers |
| Físico | Segurança física de nível mundial com certificações ISO 27001 e SAS 70 tipo II para todos os processos do data center<br>Sensores de movimento<br>Acesso seguro 24x7<br>Controlo de Acesso com sistemas biométricos<br>Sistemas de Vídeo vigilância<br>Alarmes |

Um dos pontos também deveras importante é a forma como todos os dados "navegam" por diversas *Firewalls* como sendo:

- Controladas pelo Fabric Controller
  - Firewall da máquina física
  - Firewall da máquina virtual local
- Controladas pelo gestor do serviço
  - Firewall da máquina virtual local
  - Firewall do SQL Azure




**(Figura 1: Sistema de Segurança interno do Windows Azure)**

Um dos pontos muito importantes também, uma vez que já analisámos a segurança física, é analisar também a segurança dos vários .

**Segurança do serviço de Storage do Windows Azure**

No Windows Azure os dados que são colocados no Serviço de *Storage* são colocados num hardware completamente separado relativamente aos restantes serviços, sem qualquer conectividade de rede ao Serviço de Computação excepto claro através da Internet. Estes dados são organizados em diversas contas e todos os acessos aos dados é efectuado por HTTP ou opcionalmente com SSL com autenticação de servidor.

O Acesso aos dados de uma conta específica pode apenas ser fornecido a entidades que tenham:

- Chaves de Acesso
  - As chaves de acesso que são geradas quando a conta é criada ou mesmo quando o utilizador do serviço o requisitar.
  - Cada uma destas contas tem 2 chaves de acesso de forma a suportar a rotatividade das mesmas
  - Estas chaves são simétricas de 512 bits
- Shared Access Signatures
  - Estas assinaturas proporcionam uma forma mais granular de controlo de acessos aos dados, uma vez que permitem definir as permissões e também o período de tempo em que cada um dos dados poderá ser acedido, sem que para isso necessitem acesso às chaves “mestras” da conta.

**Serviço de Controlo de Acessos (Windows Azure ACS)**

Uma outra forma de proporcionar um maior nível de segurança aos diversos dados e serviços é mediante a utilização do Serviço de Controlo de Acessos, Windows Azure Access Control Service.

Este serviço proporciona um controlo de acessos utilizando *Claim-based identity*, o que aumenta a granularidade da segurança em é aplicada a cada um dos serviços.

Também um dos pontos fortes deste serviço é o suporte aos protocolos de federação Oauth WRAP, WS-Trust e WS-Federation, bem como o suporte aos formatos de tokens SAML 1.1, 2.0 e também SWT (*Simple Web Token*) .

Para aumentar ainda a integração com outros sistemas já existentes este serviço permite também a integração com a Active Directory.

**Sumário**

Em resumo o Windows Azure proporciona uma plataforma de Cloud com um elevado grau de segurança em que os 9 pontos mais importantes são:

1. Utilização de SSL em todas as comunicações internas
2. Gestão de Certificados e Chaves privadas
3. Utilização de contas com o mínimo de privilégios possíveis
4. Controlo de Acessos ao serviço de Storage do Windows Azure
5. Isolamento do Hypervisor, Sistema Operativo Anfitrião e das máquinas virtuais utilizadas
6. Isolamento de todos os Controladores do Windows Azure
7. *Package Filtering*
8. Isolamento de VLANs
9. Isolamento dos acessos efectuados por clientes

## AUTOR

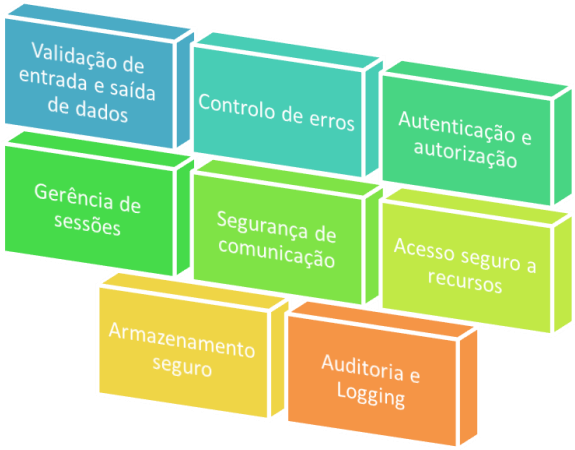
**Escrito por Nuno Filipe Godinho**

Consultor Independente com 10 anos de Experiência e principal responsabilidade de ajudar os clientes a identificar, planear, gerir e desenvolver soluções e produtos de software. Especialista em Tecnologias Microsoft. Orador em alguns dos maiores eventos de desenvolvimento da Microsoft Portugal como MSDN, TechDays, DevDays, além de eventos internacionais como TechEd Europa, TechEd Online Worldwide, MVP Nation e CloudViews.Org. Microsoft MVP há 4 anos, inicialmente em ASP.NET e a partir do início deste ano em Windows Azure com blogs em (Português e Inglês), INETA Country Leader por Portugal, e parte da equipa de gestão de por diversas comunidades Portuguesas como PontoNetPT, XAMLPT e Fundador da AzurePT (Windows Azure em Português).

# SEGURANÇA NA WEB (PARTE 2)

Neste artigo pretendemos ajudar-vos a continuar a elevar a qualidade do código das vossas aplicações. Na primeira parte falámos sobre os diferentes tipos de vulnerabilidades em aplicações web, como XSS e SQL Injection, entre outros. Neste, e em próximos artigos, iremos focar-nos como criar código seguro, assim como em melhorar o código já existente nas nossas aplicações.

Para começar é importante definir um conjunto de regras que deverão ser seguidas e que, apesar de genéricas (não serem orientadas a uma linguagem de programação específica), nos podem ajudar imenso no desenvolvimento seguro das nossas aplicações. O seguinte diagrama mostra o conjunto de regras que iremos utilizar:




As regras que iremos apresentar foram escritas em grande parte por um analista de segurança aplicacional que já trabalha em análise de código há alguns anos (@securityninja) em conjunto com outros especialistas. Para mais informação sobre este indivíduo e o seu trabalho poderão consultar os links no final do artigo.

**Regras para programação segura**

**Validação de entrada e saída de dados na aplicação**

Este princípio não é definitivamente nenhuma magia e baseia-se na verificação de que todos os dados recebidos e processados pela aplicação são correctamente validados. Isto poderá ajudar a prevenir muitas das vulnerabilidades mais comuns que estão a ser activamente exploradas por utilizadores maliciosos. É importante sabermos detalhadamente quais os tipos de dados que a aplicação deverá aceitar, que sintaxe devemos usar e quais os valores mínimos e máximos para cada variável.

Esta informação permitirá que defina um conjunto de "dados correctos" com valores para cada ponto de entrada com dados vindos do exterior.

Existem duas versões de validação de pontos de entrada, estas são chamadas "lista branca" e "lista negra". É errado tentar sugerir que uma destas versões é sempre a correcta, mas normalmente "lista branca" é a versão mais usada e segura. Uma "lista branca" define que tipos de dados deverão ser aceites pela sua aplicação para um ponto de entrada de dados específicos, resumidamente define um conjunto de "bons dados". A versão "lista negra" tenta fazer o oposto e define um conjunto de "pontos de entrada maus", isto requer que um programador entenda o funcionamento de um largo conjunto de pontos de entrada maus.

Em seguida é mostrada uma expressão regular e simples que é usada para criar listas de aceitação para números de cartões de crédito:

```
^\d{12,16}$
```

Esta expressão garante que quaisquer dados recebidos neste ponto de entrada serão constituídos por um número (\d = 0-9) com um mínimo de tamanho 12 e máximo de 16 (\d = 0-9).

Apesar de se tratar um exemplo simples, demonstra claramente o poder do uso de listas de aceitação como técnicas de validação e este ponto de entrada previne muitos dos ataques que são normalmente utilizados.

Existe também a possibilidade de se usar listas de rejeição - estas são usadas para identificar pontos de entrada de dados maliciosos e depois alterar ou remover esses mesmos dados. O exemplo que se segue irá procurar nos dados recebidos por um ponto de entrada e substituir quaisquer aspas por aspas duplas.

s.replaceAll(Pattern.quote(" ' "),

Matcher.quoteReplacement(" " "));

Normalmente evita-se usar este tipo de técnica de listas de rejeição porque só protege contra ameaças conhecidas pelo programador e que ele se lembrou no momento em que desenvolvia este código. Isto significa que esta técnica não conseguirá proteger contra futuras ameaças e vectores de ataque, e que também terá elevado custos de manutenção, quando comparado com a técnica de listas de aceitação.

Práticas corretas na validação de pontos de entrada de dados:

- Uso de listas de aceitação (valores que se sabe que estão correctos) sempre que possível;
- Normalização de todos os valores. Isto significa que devemos reduzir os dados à sua forma mais simplificada;
- Análise dos conteúdos (exemplo 0-9), tamanho mínimo e máximo e sintaxe correcta de todos os pontos de entrada.

Validação dos pontos de saída de dados:

Para além de se validar todos os pontos de entrada de dados que a aplicação recebe, também se deve criar um processo similar para os dados que a informação retorna. Alguns ataques como Cross Site Scripting tiram vantagem de pontos de saída não validados e usam-nos para atacarem utilizadores através da aplicação.

Existem três pontos principais em que se deverá ter algum cuidado numa aplicação: **codificação de dados, formatos de dados e tamanho dos dados.**

O processo de codificação de dados é ligeiramente diferente dependendo de onde os dados irão sair. Por exemplo, se os dados vão sair num endereço URL, tem de se ter certeza de que se codifica o URL. Em seguida, exemplifica-se como um valor malicioso foi colocado no URL e como a codificação do URL poderia remover esta ameaça.

O site de exemplo tem um parâmetro no URL chamado day. Este parâmetro irá conter o dia actual e irá escrevê-lo para a homepage. Isto permite que a data actual seja sempre mostrada ao utilizador.

```
www.examplesite.com/home.html?day=Monday
```

Se assumirmos que o site de exemplo não implementou nenhuma validação de saída de dados para o parâmetro **day,** um utilizador malicioso poderia substituir Segunda com qualquer valor que ele quisesse. A falta de validação do parâmetro poderia ser abusada da seguinte forma:

```
www.examplesite.com/home.html?day=%3Cscript%
3Ealert%28document.cookie%29%3C/script%3E
```

Se um utilizador não malicioso fosse aceder a este URL, iria aparecer um pop up contendo a cookie do utilizador para o site de exemplo. Isto é apenas um exemplo mas um utilizador malicioso poderia silenciosamente roubar a cookie em vez de a colocar num pop up. Se o site tivesse utilizado alguma técnica de codificação de URL, esta ameaça seria imediatamente anulada como demonstrado no seguinte exemplo:

```
www.examplesite.com/home.html?day=<script>alert
(document.cookie)</script>
```

Um segundo tipo de codificação que deverá ser considerado é codificação HTML. O primeiro tipo de codificação que demonstrámos apenas afectava a codificação de um URL; se os dados vão ser expostos numa página HTML então este tipo de codificação deverá ser aplicado.

Seguem-se dois exemplos de código. O primeiro não contém qualquer tipo de validação na saída de dados e por isso estará vulnerável a vários tipos de ataque como Cross Site Scripting.

```
#!/usr/bin/perl
use CGI;
my $cgi = CGI->new();
my $name = $cgi->param('username');
print $cgi->header();
print 'You entered $name';
```

Este código aceita qualquer texto no parâmetro username e depois mostra esses dados.

```
print 'You entered $name';
```

Facilmente se verifica que não ocorre qualquer validação nestes dados. Os dados do username deveriam ter sido submetidos a validação na entrada e saída antes de serem mostrados no ecrã. Este exemplo usa a linguagem Perl, o que significa que podemos usar o módulo HTML::Entities para codificar estes dados automaticamente por nós. O código seguinte demonstra este exemplo:

```
#!/usr/bin/perl
use CGI;
use HTML::Entities;
my $cgi = CGI->new();
my $name = $cgi->param('username');
print $cgi->header();
print "You entered ", HTML::Entities::encode
                                    ($name);
```

Quaisquer dados inseridos no campo username serão agora codificados com HTML antes de serem mostrados. Se um utilizador malicioso tentasse o exemplo anterior,

```
Content-Type: text/html; charset=utf-8
```

seria alterado para o seguinte:

```
<META   HTTP-EQUIV="CONTENT-TYPE"   CONTENT="text/
html; charset=UTF-8">
```

, o que anularia a ameaça causada pelos dados do utilizador malicioso. Os valores que foram alterados (exemplo: < & >) ainda seriam escritos na página mas como valor literal em vez de usados como caracter especial. Isto permite que se implementem técnicas de validação fortes mas que também se continue a mostrar caracteres como < & > na página.

## SEGURANÇA NA WEB (PARTE 2)

Para além da codificação que explorámos, devemos sempre controlar o método de codificação de conteúdos usado pelo browser.

Podemos configurar isto de duas maneiras: primeira pelo cabeçalho da resposta HTTP:

```
Content-Type: text/html; charset=utf-8
```

E segundo nas Meta tags:

```
<META HTTP-EQUIV="CONTENT-TYPE" CONTENT="text/
html; charset=UTF-8">
```

Isto irá confirmar que o browser irá codificar os dados correctamente.

O ponto final a relembrar é: Quando se implementa validação na saída, é importante não esquecer também a validação de tamanho (definição de um valor mínimo e máximo para os dados), como se viu na validação de entrada.

Em seguida apresenta-se um exemplo maior de código relativo a esta regra:

```
//Lista branca (permite o que se sabe que é
//valido)
<td>
      <input type=text runat=server id=userID>
      <asp:RegularExpressionValidator runat=server
            ControlToValidate= "userID"   //We
//want to validate the User ID value
            ErrorMessage="ID must be 6-10
letters."   //Error message to return if invalid
//characters are found
            ValidationExpression="[a-zA-Z]
{6,10}" />   //Our "known good" characters for the
//User ID value
</td>

Lista negra (Corrige o que se sabe que é mau)

public class ReplaceSingleQuotes
{
 public static void main(String[] args)
 {
  String str = " ' OR 1=1-- ";   //The string we
//want to validate
  String strreplace = " " ";   //We will replace
the single quote with "
  String result = str.replaceAll(" ' ",
strreplace);   //Replacing our "known bad"
characters
  System.out.println(result);
 }
}

HTML Encoding

#!/usr/bin/perl
use CGI;
use HTML::Entities;
my $cgi = CGI->new();
my $name = $cgi->param('username'); //Get the
username value and assign it to $name
print $cgi->header();
print "You entered ", HTML::Entities::encode
($name); //Using HTML Entities to HTML encode the
$name value
```

**Esta regra permite que a nossa aplicação esteja protegida contra:**

Ataques de injecção, Cross Site scripting, erros de configuração de segurança, Redireccionamentos e forwards invalidados, Spoofing de conteúdos, envio de ficheiros de tipo perigoso, falha na preservação da estrutura de uma query SQL, falha de preservação da estrutura de uma página web, falha na preservação na estrutura de um comando de sistema operativo, redireccionamento de um URL para um site não confiável, copy de buffer sem verificar tamanho dos dados na entrada, limitação imprópria de uma directoria para uma directoria restrita, controlo impróprio do nome de ficheiro a ser incluído ou um require statement em um programa php, acesso a um buffer com valores incorrectos, validação imprópria do índice de um array, overflow de inteiros, cálculo incorrecto de tamanho de buffer.

**Lidar com erros**

Todas as aplicações eventualmente terão alguma excepção e é vital que estas sejam tratadas de forma segura. Se um atacante conseguir forçar a ocorrência de excepções e a aplicação não as tratar de forma segura, é possível que estas exponham informação sensível sobre como a aplicação trabalha internamente. Estas mensagens de erro por vezes têm um alto nível de detalhe que ajuda um atacante a construir uma melhor imagem da aplicação e a melhorar os ataques usados.

Um ataque, como por exemplo de injecção SQL, será mais fácil de abusar se um atacante conseguir ver as mensagens de erro do servidor interno. No seguinte exemplo incluímos uma tentativa de ataque e uma mensagem de erro não saneada que é mostrada ao atacante:

```
http://www.examplesite.com/home.html?day=Monday
AND userscolumn= 2
```

Neste exemplo pode ver-se que o atacante incluiu AND userscolumn=2 no URL para testar a injecção SQL. A entrada de dados colocada pelo atacante foi processada no servidor SQL e causou uma excepção, que resultou no seguinte erro, devido à inexistência do campo userscolumn:

```
Microsoft OLE DB Provider for ODBC Drivers
(0x80040E14)

[Microsoft][ODBC SQL Server Driver][SQL Server]
Invalid column name 'userscolumn'.

/examplesite/login.asp, line 10
```

Este é um típico exemplo de erro que se vê na Internet e que irá ajudar um atacante a melhorar o seu ataque contra a aplicação.

Para prevenir que este tipo de erros seja mostrado aos utilizadores da aplicação, temos de ter a certeza que o código consegue processar correctamente excepções. Estes erros quando mostrados ao utilizador, devem apenas mostrar mensagens genéricas como “Erro no servidor- Por favor contacte a equipa de suporte”. Existem múltiplos pontos a relembrar quando se está a tentar implementar um processo de excepções seguro:

- Nunca incluir informação como, por exemplo, a linha em que uma excepção ocorreu, o nome do método ou stack traces;
- Nunca incluir localização ou directórios de ficheiros nas mensagens;
- Assegurar que informação como a versão usada do ASP.net não é incluída nas mensagens de erro.

Em seguida apresentamos um exemplo de código relativo a esta regra:

```
import java.io.IOException;
import java.io.InputStream;
import java.net.MalformedURLException;
import java.net.URL;

public class Test
{
public static void main(String[] args)
 {
  String urlStr = "http://www.ptcoresec.eu/
nao_existe.html "; //URL usado quando página não
existe


  try //inicio de  Try/Catch para retorno de uma
mensagem saneada caso url.openstream falhe
  {
   URL url = newURL(urlStr);
   InputStream is = url.openStream();
   is.close();
  }
  catch(Exception e)
  {
   System.out.println("Error requesting" +
e.getMessage()); //Print out the exception that
occurred
  }
 }
}
```

**Esta segunda regra permite proteger a nossa aplicação contra:**

Saída de informação, exposição de informação em mensagens de erro, verificações impróprias para excepções não casuais. (talvez fosse melhor por como pontos separados)

Autenticação e autorização

Se existir uma falha na construção de um processo forte de autenticação na aplicação, um atacante poderá conseguir acesso a conteúdo sensível sem ter que se autenticar correctamente.

A regra the **“autenticação e autorização”** é usada para tentar remover os seguintes riscos (esta é uma lista simplificada):

- Existência de um timeout apropriado;
- Uso de passwords fracas;
- Uso de perguntas fracas no sistema de pergunta secreta;
- Uso de um sistema CAPTCHA fraco ou com falhas;
- Falha na protecção de credenciais a serem transmitidas;
- Falha na implementação do sistema acesso com privilégios.

Se for necessário o uso de algum sistema de login na aplicação, deverão implementar-se também temporizadores de logout automático e uma regra que obrigue o uso de passwords fortes. Para determinar o tempo usado nos temporizadores deverá estabelecer-se primeiro o tipo de dados que estão a ser acedidos e o seu nível de confidencialidade. O temporizador de um banco será provavelmente mais curto do que o de um site de jogos,  por exemplo.

A mesma questão aplica-se quando se está a tentar descobrir o quão forte deverão ser as passwords dos utilizadores, tudo depende do que estamos a tentar proteger. Normalmente a aplicação deve obrigar  ao uso de passwords complexas, com um tamanho mínimo de 7 caracteres. Passwords complexas têm normalmente um requerimento mínimo de 3 dos 4 elementos seguintes:

- Caracteres em letra maiúscula;
- Caracteres em letra minúscula;
- Números;
- Caracteres especiais.

Dependendo do uso da aplicação poderá haver a necessidade de se implementarem controlos adicionais, como a dura-

ção máxima e prevenção de reúso de passwords. As passwords têm de ser protegidas enquanto estão armazenadas no servidor da aplicação e enquanto estão a ser transmitidas. Existem múltiplos pontos durante o tempo de vida de uma password que necessitam de atenção especial.

As passwords têm que ser armazenadas numa localização segura e encriptada, e nunca deverão ser transmitidas sem algum tipo de codificação(por exemplo: sem protecções como SSL) e nunca deverão ser completamente visíveis em emails de gerência de contas.

Neste ponto já começamos a construir um sistema de autenticação seguro, mas este trabalho árduo pode ser desfeito pelo uso incorrecto de sistemas automáticos, que são feitos para o ajudar a si e aos seus utilizadores. Quase todas as aplicações web irão ter alguma forma de sistema de lembrança de passwords e a grande maioria deles terá falhas de segurança. Estes sistemas são desenhados para terem funções automáticas, mas também podem ajudar os atacantes a roubar as contas dos utilizadores. O ponto no qual estes sistemas normalmente falham é o da pergunta secreta, usado para relembrar passwords. As respostas a estas perguntas são normalmente facilmente adivinháveis com um pequeno uso de engenharia social ou simplesmente por um ataque de força bruta. Se o sistema usa a pergunta “Qual é a sua capital favorita” o atacante sabe que existe um pequeno grupo de respostas a esta questão e poderá facilmente com um ataque de força bruta adivinhar a resposta. Se o sistema de pergunta secreta falha na prevenção deste tipo de ataques, as passwords de utilizadores serão facilmente obtidas. Para prevenir este tipo de ataque normalmente deverá deixar-se o utilizador definir a sua própria pergunta secreta ou responder a múltiplas questões antes de revelar a password.

Para além das falhas de pergunta secreta, muitos sistemas falham quando tentam criar funções de confirmação de informação. Este tipo de funcionalidade falha imensas vezes na requisição de informação suficiente por parte do utilizador final antes de lhe fornecer a password da conta. Isto é um erro comum feito por estes sistemas e deverá evitar-se sempre que possível. Fazer o sistema requisitar informação suficiente do utilizador que não é facilmente obtida, como por exemplo um segundo endereço email ou um número de carta de condução, é um dos métodos que pode ser utilizado.

Um segundo tipo de sistema que também é usado normalmente durante a criação e gerência de contas de utilizadores, são CAPTCHAs (Completely Automated Public Turing test to tell Computers and Humans Apart). Como o nome indica, estes sistemas são usados para validar que realmente é um utilizador que está a providenciar os dados e não um sistema automático; no entanto já existem imensas falhas bem conhecidas de funcionamento destes sistemas. Sistemas CAPTCHA implementados pela Google, Microsoft e Yahoo já foram quebrados, o que demonstra o nível de dificuldade de fazer este tipo de sistema funcionar correctamente. Se se decidir usar um destes sistemas deverá ter-se a certeza que os CAPTCHAS não são facilmente adivinháveis (por exemplo: Quanto é 2 + 3?) ou demasiado limpos que possam ser lidos por software OCR (conversão de imagem para texto).

O ponto final a relembrar é conhecido como “autorização de privilégio mínimo”. Uma simples aplicação teria dois níveis de acesso, administrador e utilizador, e cada uma destas teria o seu próprio nível de acesso e requisitos. Uma diferença óbvia entre dois níveis de acesso é a autorização para usar as funções administrativas da aplicação. As funções administrativas só deverão estar disponíveis para utilizadores no grupo de administração e os utilizadores comuns não deverão conseguir elevar os seus privilégios. As contas de utilizadores comuns devem ter o mínimo de privilégios necessários para executarem as suas funções correctamente. O ponto de início ideal para configurar os controlos de acesso, seria negar todos e gradualmente incluir o acesso até se encontrar o nível correcto para cada tipo de utilizador.

Deverá sempre evitar-se usar valores do lado do cliente para responder a decisões de acesso ou usar-se informação como tokens do lado do cliente, valores de URL ou campos escondidos, pois estes podem ser manipulados para aumentar os privilégios de um utilizador.

Seguem-se múltiplos exemplos de código relativo a esta regra:

```
//Usando uma expressão em PHP para verificar que a
password do utilizador é forte:
$text=$_POST['password'];
$regex='/(?=^.{8,}$)((?=.*\d)|(?=.*\W+))(?![.\n])
(?=.*[A-Z])(?=.*[a-z]).*$/';
//comparação
if ( ! preg_match($regex,$text) )
{
 //ESTA PASSWORD FALHOU
}
else
{
//password correcta, cifrar e gravar
}
//Forçar uma página a funcionar apenas em SSL:
protected void Application_BeginRequest(Object
                            sender, EventArgs e)
{
   if
(HttpContext.Current.Request.IsSecureConnection.
                                    Equals(false))
   {
    Response.Redirect("https://" +
    Request.ServerVariables["HTTP_HOST"] +
    HttpContext.Current.Request.RawUrl); }
   }
//Forçar os utilizadores a autenticarem se para
//uma certa página:
<%@ Page Language="C#" MasterPageFile="~/
```

```
MasterPage.master" AutoEventWireup="true"
CodeFile="Home3.aspx.cs" Inherits="Home2"
Title="Test Page" %>
<script runat="server">
protected void form1_Load(object sender, EventArgs
                                                e)
{
if (User.Identity.IsAuthenticated == false)
   {
    Server.Transfer("login.aspx");
   }
}
</script>
<asp:Content ID="Content1"
ContentPlaceHolderID="ContentPlaceHolder1"
                            Runat="Server">
```

Esta regra protege as vossas aplicações contra:

Falhas na autenticação e gerência de sessões, erros de configuração de segurança, redireccionamentos e forwards invalidados, autorização insuficiente, autenticação insuficiente, abuso de funcionalidade, uso de credenciais no código fonte, delegação incorrecta de permissões para determinados recursos, dependência em entradas de dados não confiáveis e decisões de segurança, falta de autenticação em funções críticas, controlo de acesso impróprio.

Terminamos assim este artigo, com o primeiro conjunto de regras que serão utilizadas para melhorar a segurança das nossas aplicações. Nos próximos artigos iremos continuar a definir o nosso pequeno conjunto de regras e demonstraremos como algum software nos pode ajudar a criar aplicações mais seguras.

Links de referência:

1. http://bit.ly/NmSwJ
2. http://bit.ly/NCgb5
3. http://bit.ly/Jy86VW
4. http://bit.ly/N38b9T
5. http://slidesha.re/LRzBOr
6. http://slidesha.re/gLKFeo

## AUTOR

**Escrito Por Tiago Henriques** @balgan, email: balgan@ptcoresec.eu

Equipa PTCoreSec: http://www.ptcoresec.eu

# No Code

Entrevista a Pedro Aniceto

# ENTREVISTA A PEDRO ANICETO

*Blogger e especialista em produtos Apple. Foi gestor de produto na Interlog, o primeiro representante da Apple em Portugal, e editor da revista iCreate Magazine Portugal. Hoje é Product & Marketing Manager at GMS Store.*

**Como entrou no mundo da Apple?**

A minha entrada no mundo da Apple é quase acidental. Em 1982, um cliente da área militar requereu-me a instalação de um sistema que eu, a bem dizer pouco conhecia. Vivíamos a transição da computação mecânica para a computação electrónica e eu estava encarregue de colocar em funcionamento um Apple II, uma impressora e aquilo que viria a ser o "avô" do Excel. As poucas dúvidas que tive (curiosamente com a única peça daquela equação que não era fabricada pela Apple) foram-me retiradas pelo representante em Portugal à altura. Fiquei tão impressionado com a forma como era feita a formação (os cursos eram fornecidos em vídeos e os exames já feitos remotamente embora de forma analógica), que um ano depois tinha três certificações feitas, sendo que uma delas, ainda é a segunda certificação de Service Apple mais antiga em território nacional. Foi o início de uma viagem absolutamente extraordinária...

> **"O futuro é absolutamente risonho sobretudo porque Steve Jobs soube ter a clarividência de se rodear dos recursos humanos que, esses sim, fazem o futuro das empresas"**

**O que acha do futuro da Apple sem o Steve Jobs?**

É verdade que a mais carismática figura da História da Informática, desapareceu. Mas deixou um legado absolutamente extraordinário: Uma das empresas mais valiosas do mundo, com um potencial de inovação e desenvolvimento fora de série, com uma cultura empresarial a todos os títulos diferente. O futuro é absolutamente risonho sobretudo porque Steve Jobs soube ter a clarividência de se rodear dos recursos humanos que, esses sim, fazem o futuro das empresas. Porque uma corporação não é uma pessoa, por mais representativa e marcante que essa pessoa possa ser. A companhia possui quadros em todas as áreas, quadros esses que marcaram e marcarão décadas de desenvolvimento e negócio.

**Acha que a confiança dos utilizadores poderá ficar afectada com o aparecimento de malwares como o flashback?**

Se perguntarmos a um elefante se fica incomodado com uma formiga…

**OSX/iOS em Português ou em Inglês? Porquê?**

Durante dez anos estive à frente da localização portuguesa do Mac OS. É-me absolutamente indiferente a língua em que trabalho em termos de sistema (o código é exactamente o mesmo e a localização geográfica de menus e strings permanece imutável de versão para versão) e acontece-me com frequência terminar uma demonstração com uma mudança de língua e essa língua permanecer activa no sistema até à próxima apresentação/demo. Neste momento o meu sistema fala francês, mas podia falar castelhano, português ou coreano. Na verdade, podiam os menus não ter sequer texto, uma vez que o Mac OS X é um sistema com acessibilidade completa, aspecto que se reflecte no iOS, neste momento um sistema capaz de ser entendido por utilizadores com qualquer espécie de deficiência ou dificuldade…

**No que respeita a inovação, há quem diga que a Apple deixou de liderar para seguir, qual a sua opinião?**

> **Neste momento o meu sistema fala francês, mas podia falar castelhano, português ou coreano. Na verdade, podiam os menus não ter sequer texto**

"E pur si muove", foi o que Galileu disse quando se viu condenado pela Inquisição, não foi? Dizer que a Apple deixou de liderar é um tremendo disparate e basta olhar para os números de mercado em diversas áreas para perguntar em que momento é que a pergunta que me é posta, alguma vez faz sentido.

**Houve rumores que a Apple iria comprar o Twitter, algo que até ao momento não se comprovou, tendo em conta que seria uma aquisição fora do âmbito das anteriores (essencialmente companhias de software), acha que faria sentido uma aposta do género?**

É regra das pessoas que trabalham na área Apple não se manifestar sobre negócios ou produtos que ainda não o são. É o caso. Mas falarmos de eventos anteriores, é fácil perceber-se que sempre que a Apple quis pegar num bom conceito e transformá-lo numa explosão de mercado, fê-lo sem qualquer obstáculo. Basta pensar na génese do iTunes (o melhor bloco de código alguma vez escrito para Windows...) para perceber que se tiver de acontecer, acontecerá. Pessoalmente sou grande fã e utilizador Twitter e não faz para mim qualquer sentido uma aquisição deste género a não ser que o sistema se possa reinventar, o que acho complexo sem lhe abastardar a essência.

**Como é a "realidade" Apple em Portugal?**

Exactamente igual à realidade em qualquer outro território. A Apple é dona de mercados que há poucos anos eram impensáveis serem liderados por ela. Estabelece standards a cada lançamento. As coisas evoluíram muito em termos de Distribuição e a globalização é um facto cada vez mais presente. Ter sistemas e plataformas independentes de barreiras culturais e/ou linguísticas ajuda bastante a uma experiência de utilização cada vez mais satisfatória e que faz com que o marketshare vá reflectindo essa mesma realidade.

**Defenderia uma Apple mais "liberal"?**

A Apple não está no mercado para ser mais ou menos liberal. A Apple é uma corporação e na sua essência está uma qualidade de integração de hardware e software que não tem paralelo em qualquer outro fabricante. Estão na nossa memória colectiva os erros cometidos quando essa "liberalização" foi permitida. A qualidade e a experiência de uso tombaram a níveis nunca vistos e rapidamente a empresa se voltou para o modelo inicial.

Aquilo que conta é a experiência do utilizador. Se for para a piorar, não obrigado.

> **A Apple não está no mercado para ser mais ou menos liberal**

# Veja também as edições anteriores da Revista PROGRAMAR

334 Edição - Abril 2012


33ª Edição - Fevereiro 2012


32ª Edição - Dezembro 2011


31ª Edição - Outubro 2011


30ª Edição - Agosto 2011


29ª Edição - Junho 2011


e muito mais em ...

www.revista-programar.info

DUVIDAS?

IDEIAS?

AJUDAS?

PROJECTOS?

portugal-a-programar.org