PROGRAMAR

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

EDIÇÃO #52 - MARÇO 2016

ISSN 1647-0710

TESTAR APLICAÇÕES MÓVEIS COM XAMARIN TEST CLOUD

ANDROID MONKEY TEST
Um "Macaco" ao Serviço dos Programadores Android

PROGRAMAÇÃO (IN)SEGURA
Transbordo de Memória

ASP.NET CORE 1.0
Ligação a base de dados SQL Azure

EM ANÁLISE
JAVASCRIPT 6

NO CODE
O QUE ESCONDE O CQRS

e muito mais!

EDITORIAL

EQUIPA PROGRAMAR

Coordenador
António Pedro Cunha Santos

Editor
António Pedro Cunha Santos

Design
Sérgio Alves
Twitter: @scorpion_blood

Ilustração
Sara Freixo

Redacção
Ana Rita Queiroz
António Pedro Cunha Santos
Guilherme Ferreira
João Sousa
Nuno Santos
Patrício Domingues
Ricardo Cabral
Ricardo Miranda
Ricardo Peres
Rita Peres
Sara Silva
Tânia Valente
Vítor Távora

Staff
António Pedro Cunha Santos
Rita Peres
Rui Gonçalves

Contacto
revistaprogramar@portugal-a-programar.org

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

ISSN
1 647-071 0

gcc ed52.c -o ed52

É com orgulho que vos trazemos mais uma edição da PROGRAMAR.

A edição 52. A primeira de 2016.

E a título de homenagem a todos vós escolhemos o dia do pai para mais um lançamento. Porquê? Porque em cada programador há um pai. (e em cada programadora uma mãe, claro!…)

Quantos de vocês, não se sentiram já um pouco "pai" do vosso programa? Creio que há sempre algum projeto que nos ficou, que nos foi mais especial. Que ainda recordamos com saudade. Ou aquele projeto que não conseguimos deixar para trás.

Creio que em cada programador há um pouco de pai. Em cada compilação, em cada build, criamos expectativas e traçamos metas. Em cada linha de código "ensinamos-lhe" o comportamento a adotar. Tentamos prever todos os crash e segmentation fault que "ele" possa sofrer. Queremos o melhor para o nosso programa. Queremos que os utilizadores falem e "gostem" dele. Queremos que "ele" seja o melhor. E queremos sempre melhorá-lo.

Sim, é justo dizer que em cada programador há um pouco de pai!

Como esta é a primeira edição do ano, e apesar de ser Março, não queremos deixar de vos desejar um ano fabuloso (pronto… que sejam pelo menos os últimos três trimestres de 2016 fabulosos)!

Esta edição foi mais uma vez, feita a pensar em todos os nossos leitores, aos amadores e aos profissionais, sem nunca esquecer os aspirantes e simpatizantes de tecnologia.

Queremos ainda dizer-vos que a próxima edição será em Maio. No dia do aniversário da PROGRAMAR!

Até lá… aproveitem, criem, programem, busquem a perfeição e fiquem orgulhosos dos vossos projetos! Porque em cada um deles há uma parte de nós. E em pelo menos um… há o melhor de nós!

Rita Peres

A revista PROGRAMAR é um projecto voluntário sem fins lucrativos. Todos os artigos são da responsabilidade dos autores, não podendo a revista ou a comunidade ser responsável por alguma imprecisão ou erro.

Para qualquer dúvida ou esclarecimento poderá sempre contactar-nos.

ÍNDICE

TEMA DE CAPA

6 Testar aplicações móveis com Xamarin Test Cloud - **Sara Silva**

A PROGRAMAR

18 Extrair dados do cartão de cidadão - **Ricardo Cabral**

30 Android Monkey Test: Um "Macaco" ao Serviço dos Programadores Android - **Nuno Santos**

33 Programação (in)Segura - Transbordo de Memória - **Patrício Domingues, Vítor Távora**

41 ELM - **Ricardo Miranda**

44 ASP.NET Core 1.0 – Ligação a base de dados SQL Azure - **João Sousa**

ELECTRÓNICA

49 Automação com GENUINO- **António C. Santos, Rita Peres**

COLUNAS

54 O papel do profissional de Marketing & Comunicação no sector das TI's em Portugal - **Ana Rita Queiroz**

ANÁLISES

58 JavaSscript 6 - **Tânia Valente**

60 Modelação de Dados em UML – Uma Abordagem por Problemas - **Ricardo Peres**

NO CODE

62 O que esconde o CQRS - **Guilherme Ferreira**

64 Comunidades e Voluntariado Presencial - **Ricardo Cabral**

66 SHIFT APPENS 2016 - COIMBRA- **Rita Peres**

68 RASPBERRY PI 3 MODEL B - **Rita Peres**

70 Windows 10: Gestão de redes Wireless através da Linha de Comandos - **Nuno Silva**

75 Projecto em Destaque P@P - MATRAQUILHOS

EVENTOS

Porto Codes - http://www.meetup.com/portocodes/events/227739316/ - 23 de Março
9° Evento da AzurePT - https://www.facebook.com/events/1431223563559109/ - 26 de Março
Main Hub Talks - Intro a RavenDB 3.0, dynamic version—https://www.facebook.com/events/243953265942298/ - 29 de Março
2º Encontro da Comunidade IT Pro Portugal - https://www.facebook.com/events/1567318656918711/ - 31 de Março
Porto Wordpress meetup - http://www.meetup.com/WP-Porto/- 31 de Março
[0x32] - The meet - http://www.meetup.com/0xOPOSEC/ - 31 de Março
Genuino Day Lisbon http://genuinoday.iot.org.pt - 2 de Abril
Hack for Good http://www.hackforgood.pt - 23 e 24 de Abril de 2016
IV Electrónica & Informática - https://www.facebook.com/events/152497711796483/ - 10 de Junho

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

#PGGD30 - 30º Portugal Girl Geek Dinner – Lisboa

#PGGD31 - 31º Portugal Girl Geek Dinner – Porto

Como sabem os nossos leitores, aqui na PROGRAMAR temos acompanhado o PGGD! E nesta edição é com prazer que vos falamos novamente neste projecto. No passado mês de Fevereiro, tiveram lugar dois encontros da comunidade Portugal Girl Geek Dinner.

O **#PGGD30 - 30º Portugal Girl Geek Dinner – Lisboa**, aconteceu no dia 23 nas instalações da Affinity, que patrocinou mais uma edição lisboeta.

Todo o grupo foi bastante bem recebido e foi num ambiente descontraído e bem-disposto que ocorreu toda a sessão, e agradecemos à Rita Queiroz/Affinity a disponibilidade ao longo de todo o evento, no qual estivemos presentes!

As apresentações deste encontro ficaram a cargo da *Liliana Carvalho - Atitude, Processos e Valor - CI/CD* - e do *Nuno Martins - You, Me and Git*.

Mais uma vez a PluralSight não deixou de apoiar o PGGD, oferecendo uma licença e-learning de um mês na sua plataforma.



Por sua vez, o **#PGGD31 - 31º Portugal Girl Geek Dinner – Porto**, realizou-se no dia 27 de Fevereiro, na Porto Design Factory com o patrocínio da Ambidata, sendo que agradecemos também desde já toda a disponibilidade para apoiar o evento.

Desta vez, as apresentações ficaram a cargo da *Joana Carvalho - Cultures of Performance: making it through the cat door*, e da *Joana Fillol – Jornalíssimo*, e da *Isabel Portugal - Big Data: When the size matter.*



Porto Design Factory
ambidata
Digital Innovation Solutions & Consulting
GIRL GEEK DINNER
definitely does compute

Queremos relembrar ao leitor que os encontros desta comunidade são abertos a todos os que queiram participar! Se forem uma *geek girl*, inscrevam-se e se quiserem tragam um acompanhante (*geek* ou *não geek*), se forem *geek*... bem... peçam a uma *geek girl* (ou a uma *girl "com queda" para geek*) que vos leve ao encontro! Vão encontrar gente bem-disposta, e acima de tudo um ambiente de partilha de informação e ideias. Mais uma vez desafiámos-vos a frequentarem as sessões!

Certamente será um momento bem passado e a repetir!

Mais informações em: http://portugalgirlgeekdinners.com

«Escrito por: Rita Peres»

Ataques DDOS Afectam uma em cada 5 Empresas

A Karspersky conduziu no ano passado um estudo que revela que cerca de vinte por cento das empresas foram já alvo de ataques DDoS. Este estudo foi realizado em parceria com a B2B International, analisando também os setores mais afetados, destacando as telecomunicações. Houve reconhecimento por parte de algumas das empresas inquiridas nesta área de atividade, de já terem sido vítimas de ataques deste tipo.

Estes ataques são caracterizados pelo envio de um elevado número de pedidos para um determinado site ou serviço, de tal forma que este acaba por não ser capaz de gerir um número de solicitações tão elevado, ficando então indisponível. Este estudo teve como alvo apenas empresas com mais de 50 funcionários, tendo sido inquiridos 5.500 em 26 países.

TEMA DE CAPA

Testar aplicações móveis com Xamarin Test Cloud

Testar aplicações móveis com Xamarin Test Cloud

Âmbito

Este artigo tem como objetivo mostrar como podemos testar as aplicações móveis usando Xamarin Test Cloud, sejam estas aplicações Xamarin ou aplicações nativas.

Introdução

O desenvolvimento móvel tem tido um crescimento exponencial, trazendo com isso um "*booom*" de aplicações para as diversas lojas. Claro está, que muitas das vezes quantidade não significa qualidade e quantas vezes nós próprios nos deparamos com potencias aplicações que apenas precisam de melhorias, e em alguns casos a aplicação funciona bem numa plataforma e noutra não (por razões diversas). Com este crescimento e com base nas diversas necessidades muitas soluções têm surgido no mercado para que os programadores possam acompanhar o desempenho da aplicação durante a sua utilização.

Independentemente da forma como os programadores acompanham as suas aplicações para fazer manutenção corretiva, coloca-se a seguinte questão: será que isto é uma boa solução? Isto é, será que é bom serem os nossos utilizadores a encontrar os bugs e comportamentos não esperados? Porque isto é ou tem sido uma realidade,… Ou será que os programadores devem-se preocupar com estratégias de testes antes de esta chegar ao utilizador final?

Seguindo boas práticas podemos dizer que cada aplicação deve ser testada antes de chegar ao utilizador final, e deve-se definir pelo menos os testes unitários e os testes de aceitação (em alguns casos pode haver necessidade de definir os testes de integração, com serviços externos, como por exemplo o BackEnd). Com a criação de testes vamos conseguir provar que o comportamento desenvolvido é o que está a ser testado, e se alguma alteração quebrar os testes ou o bom funcionamento da aplicação, é possível colocar em causa se houve um erro inicial, um erro no momento da alteração ou se o problema em causa não foi tido em conta, e claro está, proceder à correção.

Tipos de testes

A fase de testes é o processo que permite garantir que o desenvolvimento realizado faz efetivamente o que foi definido inicialmente. Esta fase pode incluir vários tipos de teses, tendo cada um o seu propósito, resumidamente podemos dizer que:

- Os testes unitários são testes escritos normalmente pelo(s) programador(es) que visa validar as várias funcionalidades de forma isolada, ie, o programador cria vários testes para verificar que um determinado método, classe, módulo ou componente é válido de acordo com os requisitos e faz o que é expetável.

- Os testes de integração são testes criados com o objetivo de validar a integração de classes, módulo ou componentes umas com as outras. No caso de aplicações móveis podemos dar como exemplo para testes de integração a comunicação entre o backend e a aplicação móvel.

- Os testes de aceitação são testes criados maioritariamente pela equipa de QA, e que visa testar a aplicação como sendo uma caixa negra, ie, é sabido à partida que é possível realizar um conjunto de operações e ao efetuar essas operações é expetável um resultado final.

Na figura 1, podemos ver outros tipos de testes, e a relação custo/esforço e nº de testes, claro está que o mais comum é a criação de testes unitários, e nem sempre é feita a cobertura de todos os tipos de testes.

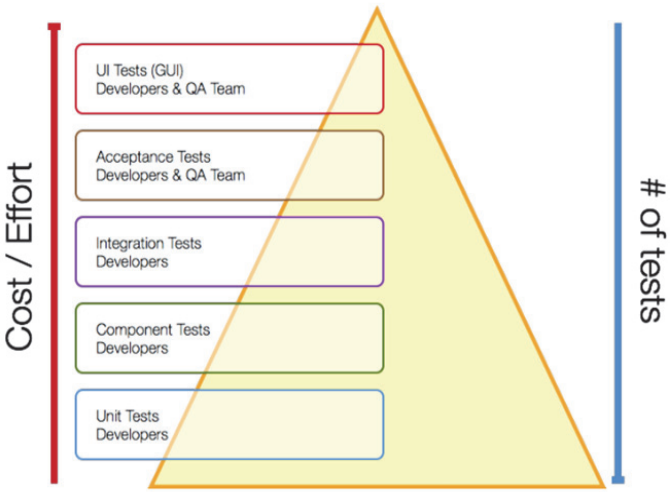


Figura 1: Vários tipos de testes

Execução automática de testes

Outra questão extremamente relevante, é a execução dos diversos testes ser um processo automático, ie, à medida que as alterações se vão refletindo se possa confirmar que tudo o que estava a funcionar está efetivamente a funcionar, sem que se tenha que recorrer a processos manuais. Por esta razão, muitas vezes é criando um sistema de integração continua (usando por exemplo Team City ou TFS Online), em que na operação de *checkin* / *push* para o gestor de controlo de versões são executados todos os testes, refe-

rente à aplicação, e só após a confirmação de que todos os testes estão OK é que são aceites as novas funcionalidades ou alterações por parte do servidor.

Um programador mais atento vai questionar-se: é verdade que posso executar os testes unitários e de integração em processo automático, usando as ferramentas de desenvolvimento e de testes (seja NUnit ou MSTest), mas então, e os testes de aceitação? Uma vez que para verificar os diversos testes de aceitação é preciso executar a aplicação!? Vou ter constantemente uma equipa de QA a fazer o processo manualmente?

Neste artigo vamos ver como podemos escrever e executar os testes de aceitação de forma automática recorrendo à solução da Xamarin, o Test Cloud.

Descrição

Para dar suporte ao artigo, vamos considerar a aplicação **TODO**, desenvolvida com base na plataforma Xamarin, cujo código fonte pode ser obtido no seguinte endereço:

https://github.com/xamarin/xamarin-forms-samples/tree/master/Todo

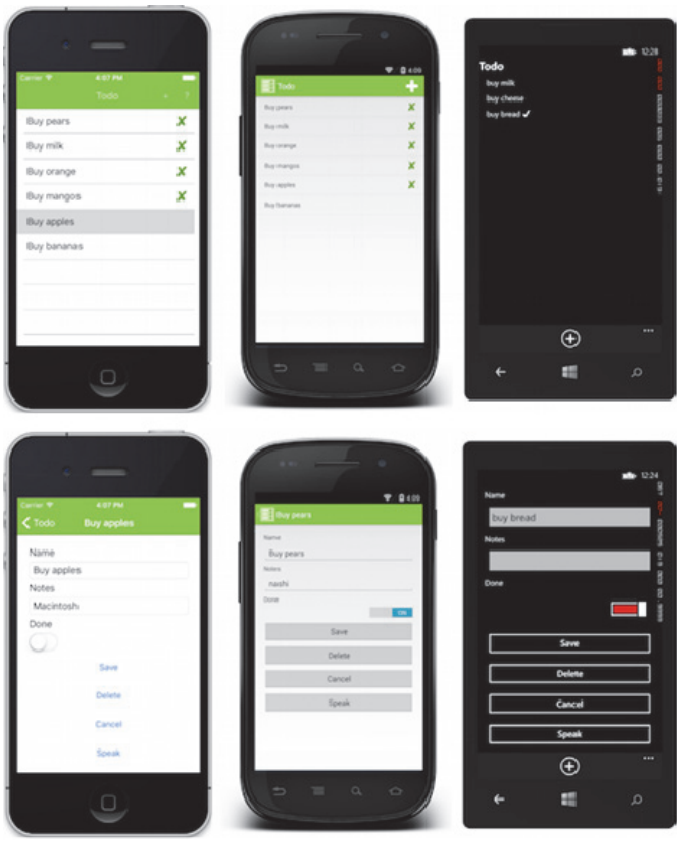


Figura 2: Aplicação TODO

Consideremos agora que foi terminada uma das fases de desenvolvimento, e que existe um conjunto de funcionalidades testadas recorrendo a testes unitários e de integração. Visto isto, a equipa de desenvolvimento e/ou a equipa de QA pretende proceder à criação de testes de aceitação, para validar todo o fluxo da aplicação e as diversas funcionalidades. Desta forma, vamos proceder à criação dos testes de aceitação, recorrendo à biblioteca *Xamarin.UITest.*

Xamarin.UITest

A *Xamarin.UITest* é uma biblioteca desenvolvida pela Xamarin que permite escrever os testes automáticos de aceitação, ao nível da UI, de aplicações desenvolvidas com base na plataforma Xamarin. Esta biblioteca é baseada em Calabash – uma biblioteca que permite escrever testes automáticos de aceitação, ao nível UI, para aplicações nativas iOS e Android.

Na figura 3, podemos ver o diagrama de decisão relativo às *frameworks* que podemos utilizar nas aplicações móveis.

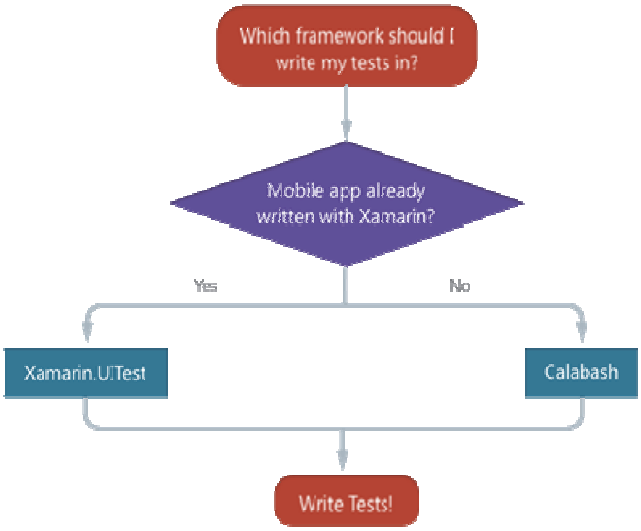


Figura 3 Diagrama de decisão

Esta biblioteca:

- É baseada em C# e usa a biblioteca NUnit (versão 2.6.4)

- Está apenas disponível para Xamarin.iOS e Xamarin.Android

- É suportada pelos diversos IDE (Xamarin Studio e Visual Studio)

- O Xamarin Studio disponibiliza projetos *template* para Xamarin.iOS, Xamarin.Android e Cross-Platform

- O Visual Studio apenas disponibiliza projetos *template* para projetos Cross-Platform

- Está disponível através de pacote Nuget – cuja versão atual é Xamarin.UITest 1.2.0

- É executado sobre um agente, *Xamarin Test Cloud Agent*, que vai permitir executar os testes sobre a aplicação (quer estejamos perante o simulador ou um dispositivo)

Criação de projeto

Comecemos por abrir o projeto TODO no Xamarin Studio, como podemos visualizar nas imagens seguintes.

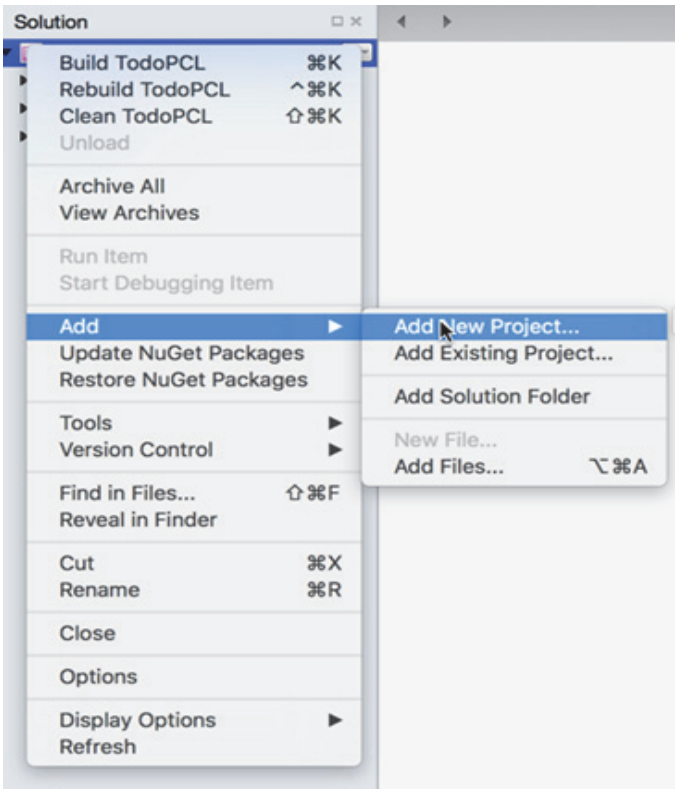


Figura 4 Adicionar novo projecto

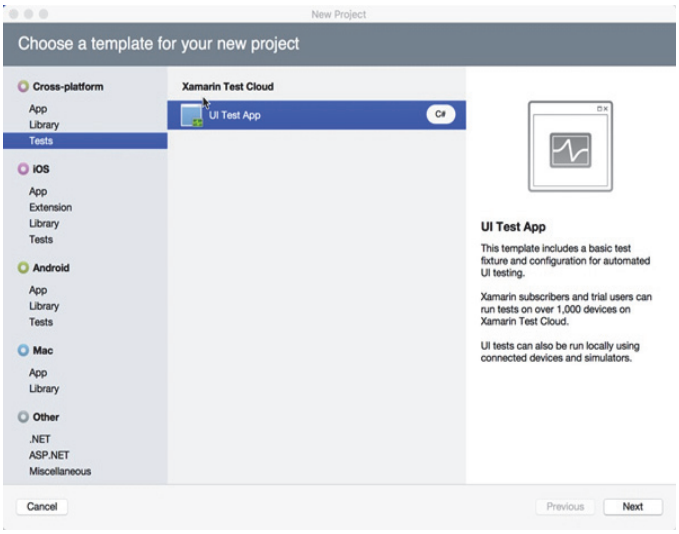


Figura 5 Seleção do projecto

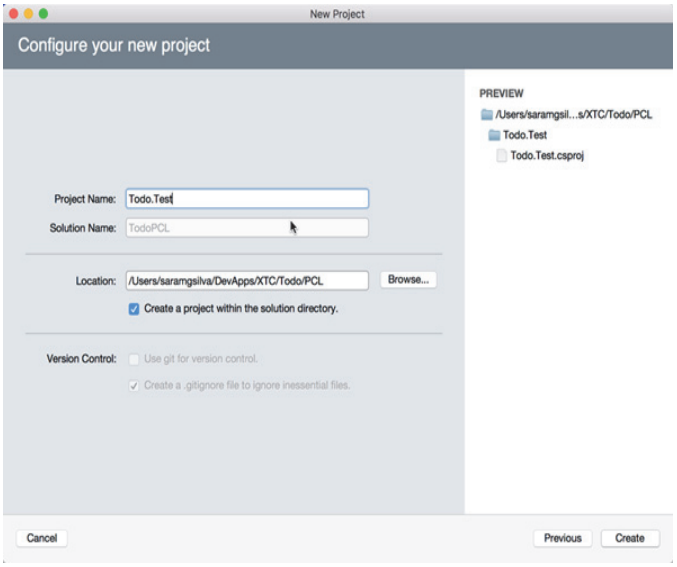


Figura 6 Definição do nome do projecto

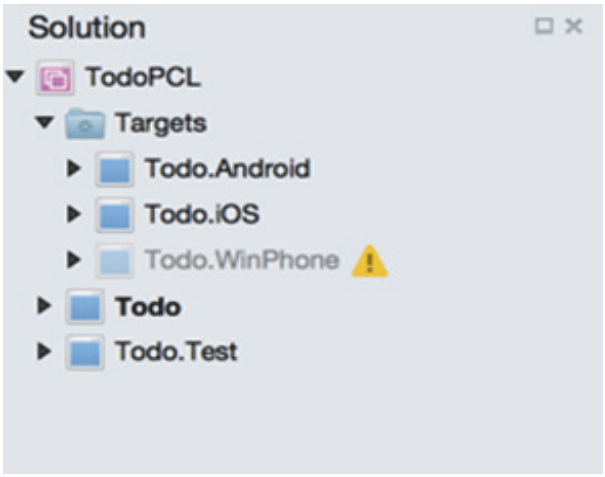


Figura 7 Solução com o projeto de testes

O *template* do projeto de testes, que acabamos de criar, inclui dois ficheiros: *AppInitializer*.cs e *Test.cs*.

AppInitializer

A classe *AppInitializer*, com apenas um método (*StartApp*) irá:

- Definir a aplicação sobre qual correm os testes

  - Com base no pacote APK, IPA ou bundle (.app)

```
app = ConfigureApp.iOS.AppBundle("../../path/todo.app");

app = ConfigureApp.Android.ApkFile("../../path/todo.apk");
```

- Com base no nome da aplicação instalada no dispositivo

app = ConfigureApp.iOS.InstalledApp ("com.xamarin.todo");

app = ConfigureApp.Android.InstalledApp ("com.xamarin.todo");

- Com base no projeto definido na solução

- Definir o dispositivo que irá executar os testes (por omissão é usado o simulador)

- Iniciar a aplicação, quando o método *StartApp* é invocado

Para saber mais detalhes, consulte a documentação fornecida pela Xamarin:

- Xamarin.UITest.Configuration.iOSAppConfigurator Class
- Xamarin.UITest.Configuration.AndroidAppConfigurator Class

Para o caso da aplicação **TODO** iremos ter o seguinte *code snippet*:

```
public class AppInitializer
    {
        public static IApp StartApp (Platform
                                        platform)
        {
            // TODO: If the iOS or Android app
            //being tested is included
            //in the solution
            // then open the Unit Tests window,
            //right click Test Apps,
            //select Add App Project
            // and select the app projects that
            //should be tested.
            // The iOS project should have the
            //Xamarin.TestCloud.Agent
            //NuGet package
            // installed. To start the Test Cloud
            // Agent the following code should be
            // added to the FinishedLaunching
            //method of the AppDelegate:
            //    #if ENABLE_TEST_CLOUD
            //    Xamarin.Calabash.Start();
            //    #endif
            if (platform == Platform.Android) {
                return ConfigureApp
                    .Android
                    .ApkFile ("../../../
             Todo.Droid/bin/Debug/TodoDroid.apk")
                    .StartApp ();
            }

            return ConfigureApp
                .iOS
                .AppBundle ("../../../Todo.iOS/
bin/iPhoneSimulator/Debug/TodoiOS.app")
                .StartApp ();
        }
    }
```

De realçar que na configuração inicial do pacote da aplicação para iOS, é possível surgir o erro

SetUp: System.Exception: The app bundle in XTC/Todo/PCL/Todo.iOS/bin/iPhoneSimulator/Debug/TodoiOS.app does not seem to be properly linked with Calabash. Please verify that it includes the Calabash component.

que significa que a aplicação iOS não contém a referência para o *Xamarin Test Cloud Agent* ou não contém a chamada ao método *Xamarin.Calabash.Start()* no event handler *AppDelegate.FinishedLaunching*. Para o caso em que este erro continue a surgir e esteja tudo bem configurado, é recomendável que se apague a pasta *bin* do projeto de iOS e se faça *build* novamente.

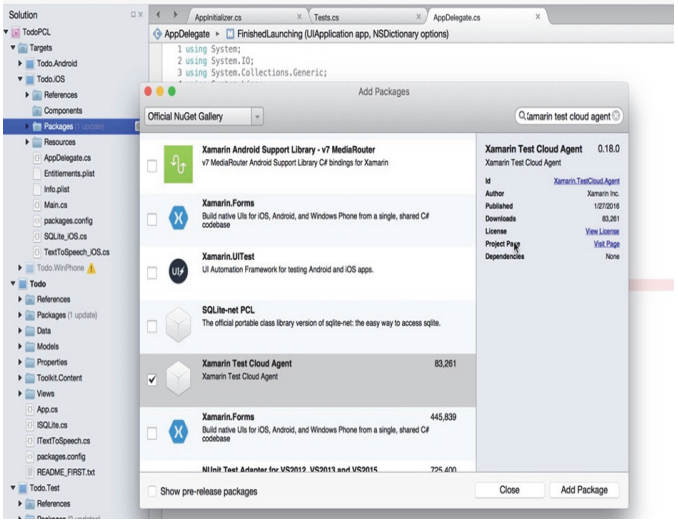


Figura 8 Adiciona o pacote Xamarin Test Cloud Agent ao projeto de iOS

Tests

A classe *Tests* define um método que é designado por *setup* que permite iniciar os testes sobre a UI da aplicação, e contém os vários testes para as diversas plataformas (na prática podem existir mais do que uma classe com testes, dependendo da organização destes).

Por omissão, esta classe está definida para lançar os testes nas duas plataformas, isto porque a classe está decorada com os seguintes atributos:

```
[TestFixture (Platform.Android)]
    [TestFixture (Platform.iOS)]
    public class Tests { … }
```

Isto significa, se os testes definidos estiverem a suportar mais do que uma plataforma, a sua execução originará que vários simuladores iniciem a aplicação ao mesmo tempo (em alguns casos poderá ser interesse analisar em

simultâneo o comportamento da aplicação nas diversas plataformas, não sendo, no entanto, um cenário comum).

O IDE irá lançar a aplicação no simulador ou no dispositivo, consoante o que estiver configurado. No caso da plataforma Android é requisito haver pelo menos um dispositivo conectado ao ADB, para que assim seja possível executar a aplicação e consequentemente os testes. É também requisito que a aplicação tenha permissões de internet, independentemente se esta requer o envio/receção de dados, uma vez que a anatomia da biblioteca requer esta permissão.

Nota: Sendo a biblioteca Xamarin.UI.Tests baseada em NUnit, é possível usar o atributo [Category] para categorizar os testes e desta forma organizar os diversos testes.

Interface IAPP

A classe *Tests* pode ser alterada, sendo possível definir os testes que se pretender. A interface IApp acessível por esta classe, permitirá a interação com a UI da aplicação, e portanto, é possível localizar elementos gráficos (textbox, botões,…), lançar eventos (como por exemplo "Tapping", "Typing"), entre outras operações como podemos ver na figura 9, e iremos descrever mais à frente neste artigo.

Commonly used methods		
Query	Tap	WaitFor
WaitForElement	WaitForNoElement	Screenshot
SwipeLeft	SwipeRight	ScrollUp
ScrollDown	Flash	

Figura 9 Operações mais comuns

Vejamos alguns code snippets

- Tirar screenshot à aplicação e atribuição do nome da imagem

```
app.Screenshot ("First screen.");
```

- Pressionar o botão com o texto ou id "Add"

```
app.Tap (c => c.Marked ("Add"));
```

- Obtenção do botão (visível) com o texto ou id "Save"

```
AppResult[] matchedItems = app.Query( c => c.Button().Marked("Save"));
```

- Obtenção de controlos baseado no tipo por plataforma

Para iOS

```
var matches = app.Query(c => c.Class ("UILabel"));
```

```
var matches = app.Query(c => c.Class ("TextView"));
```

Para Android

```
var matches = app.Query(c => c.Button());
```

```
var matches = app.Query(c => c.TextField());
```

Execução de testes localmente

Atualmente o melhor cenário para correr os testes será usando o Xamarin Studio num Mac, uma vez que é a única forma de executar os testes para iOS. No Xamarin Studio é possível escolher a vista "Unit Testing", como podemos ver na figura 10.

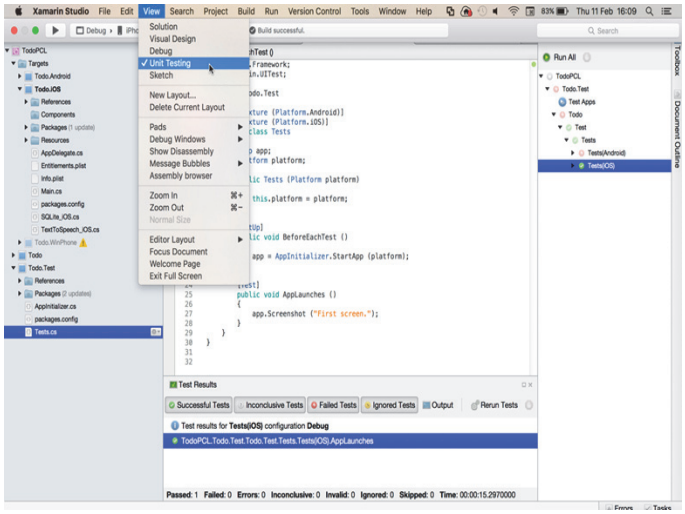


Figura 10 Vista Unit Testing

No menu "Unit Testing", do lado direito, é possível escolher o(s) teste(s) que pretendemos executar, como podemos ver na figura 11.

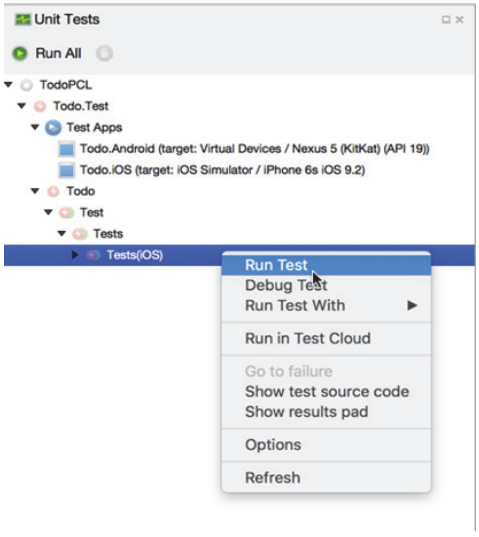


Figura 11 Execução de testes

Portanto, para o código fornecido pela classe Tests

```
[TestFixture (Platform.Android)]
  [TestFixture (Platform.iOS)]
  public class Tests
  {
      IApp app;
      Platform platform;

      public Tests (Platform platform)
      {
          this.platform = platform;
      }

      [SetUp]
      public void BeforeEachTest ()
      {
       app = AppInitializer.StartApp (platform);
      }

      [Test]
      public void AppLaunches ()
      {
          app.Screenshot ("First screen.");
      }
}
```

O constructor da classe é invocado, e de seguida é invocado o método Setup que irá conhecer a plataforma para a qual os testes vão ser executados, e por fim o método AppLaunches é invocado e um screenshot é tirado. Considerando que vamos executar os testes apenas para iOS, a figura 12, mostra a aplicação no simulador de iOS.

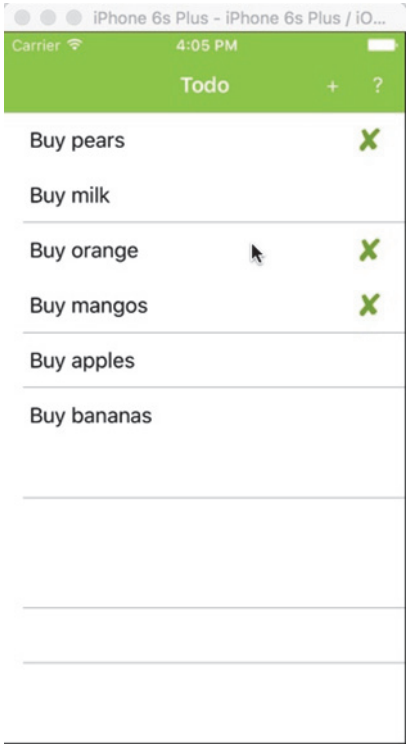


Figura 12 Aplicação lançada pelo teste

O resultado da execução irá ser apresentado no painel "*Test Results*", como podemos ver na figura 13.

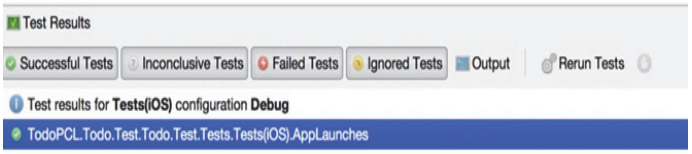


Figura 13 Resultado da execução do(s) teste(s)

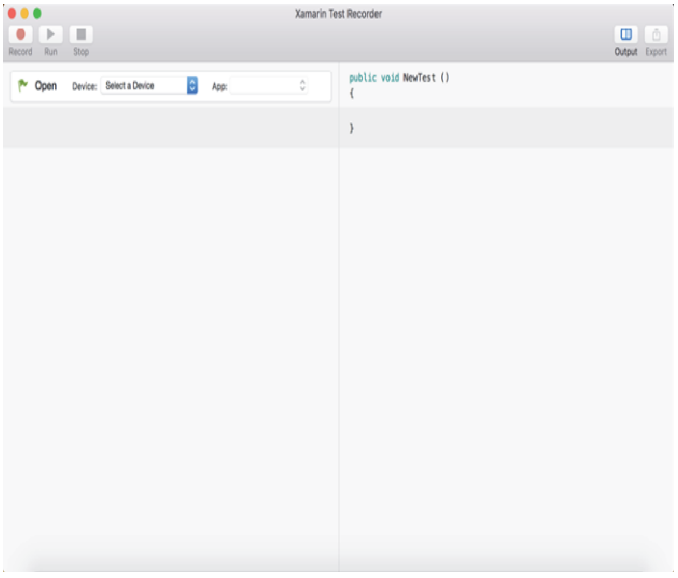
Xamarin Test Record

Os testes de aceitação podem ser definidos manualmente, no entanto, a sua definição manual pode não ser um processo tão simples quanto isso. No lançamento do Xamarin 4, a Xamarin lançou a versão *preview* do *Xamarin Test Record*, que é uma aplicação que permite gravar as operações efetuadas na aplicação, ie, o *Test Record* irá escrever os testes, que irão verificar o comportamento da aplicação, que são obtidos pela execução manual da aplicação. Desta forma é possível replicar automáticamente o comportamento efetuado pelo utilizador.

À data da publicação deste artigo, o *Xamarin Teste Record* está apenas disponível para Mac, sendo possível obtê-lo em https://xamarin.com/test-cloud/recorder.

Criação automática de testes

O Xamarin Test Record permitirá gravar e executar testes de aceitação para aplicações desenvolvidas usando a plataforma Xamarin, apresentando uma interface minimalista com as opções básicas: gravar, editar, executar e exportar testes (como podemos constatar na figura 14).



Antes de iniciar a gravação dos testes, é requisito definir:

- Qual o dispositivo que pretendemos usar para executar a aplicação

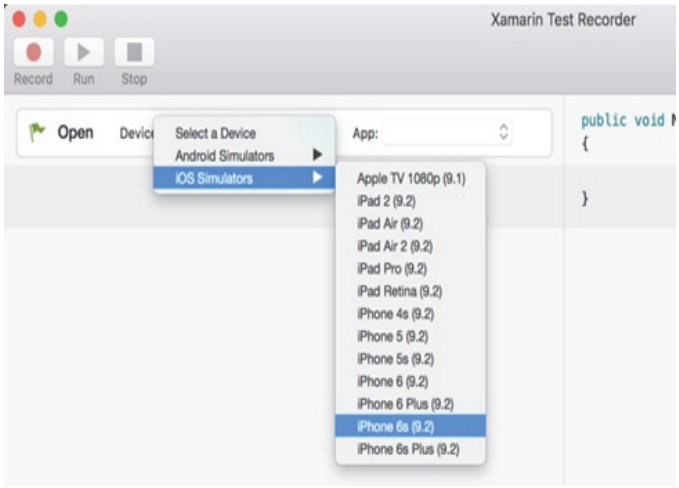


Figura 15 Seleção do dispositivo

- Qual a aplicação que vamos executar



Figura 16 Seleção da aplicação

Após esta seleção a aplicação é executada no dispositivo selecionado e em seguida é possível gravar as diversas funcionalidades da aplicação. Vejamos a figura 17, que nos mostra o Test Record a executar a aplicação TODO.

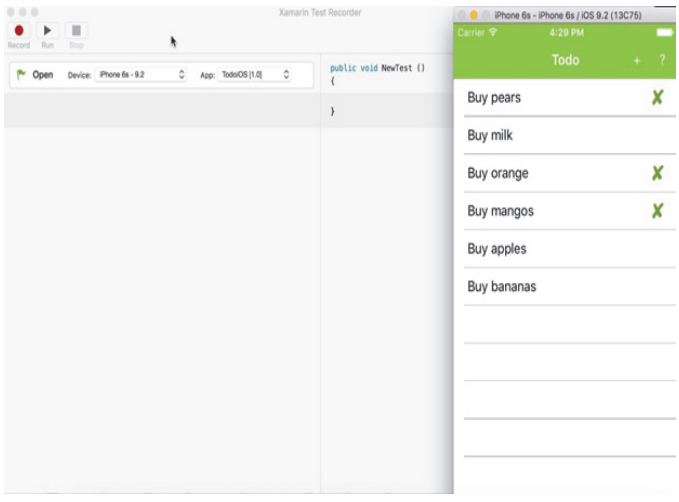


Figura 17 Xamarin Test Record a executar a aplicação TODO

Como se pode constatar, a aplicação apresenta alguns items já comprados, suponhamos agora que o nosso objetivo é selecionar o item “Buy milk” e defini-lo como “Done”, para que seja considerado como comprado. A operação descrita irá ser gravada pelo *Test Record*, que por sua vez irá escrever o testes de aceitação para este comportamento. Apartir da obtenção deste teste, é possível replicar este comportamento automáticamente e sem que seja necessário recorrer a processos manuais. Sendo assim, iniciemos a gravação, como podemos ver na figura 18.

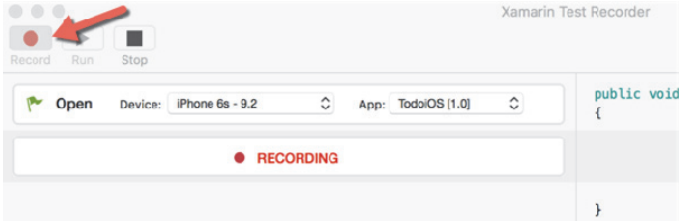


Figura 18 Inicio da gravação

Após isto, clicando no item “Buy milk” iremos ver no Test Record o código referente ao “Tap” no “Buy milk”, como se pode ver na figura 19.

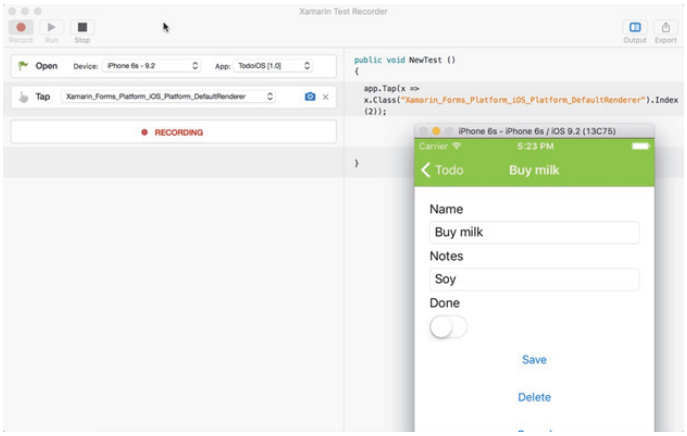


Figura 19 Resultado da seleção do item "Buy milk"

O resultado final pode ser analisado na figura 20, e é possível concluir que o teste é definido por três operações:

- Tap sobre o elemento “Buy milk”;
- Swipe no elemento “Done”;
- Tap no botão “Save”;

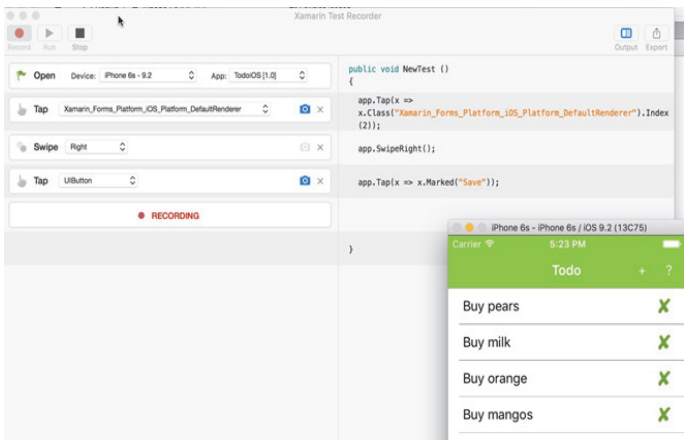


Figura 20 "Buy milk" selecionado como comprado

Edição manual de testes

À data, não é possível editar manualmente o código gerado no Test Record (que podemos vizualizar do lado direito), a sua edição tem que ser feita no menu do lado esquerdo através das opções disponíveis, que permitem editar o controlo selecionado, remover operações realizadas e adicionar screenshots em cada passo. Na figura 21, podemos ver em mais detalhe as várias opções.

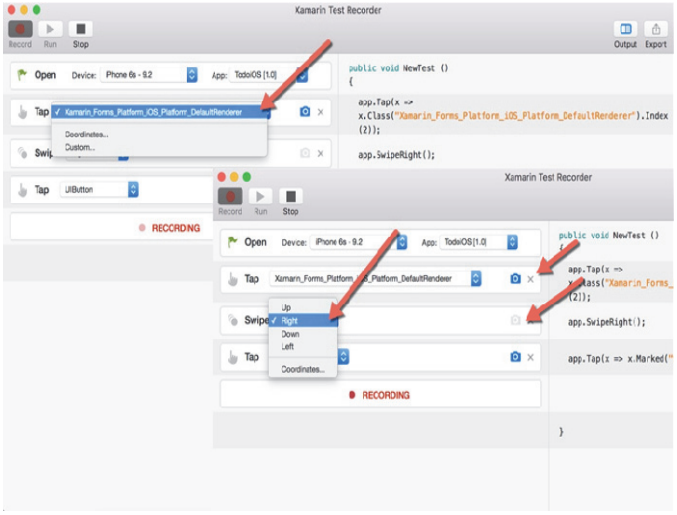


Figura 21 Edição do teste

Exportação de testes

No final do teste ser gravado é possível exportá-lo para o clipboard, como ficheiro *.cs ou enviar o teste para o Test Cloud, como podemos ver na figura 22.

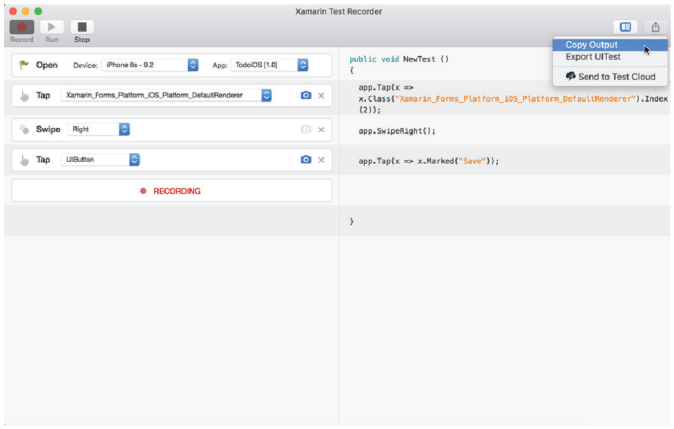


Figura 22 Exportar teste

O output, neste caso, será então o seguinte code snippet:

```
[Test]
    public void NewTest ()
    {
        app.Tap(x => x.Class
                ("Xamarin_Forms_Platform_iOS_
         Platform_DefaultRenderer").Index(2));
        app.Screenshot
 ("Tapped on view Xamarin_Forms_Platform_iOS_
                        Platform_DefaultRenderer");
        app.SwipeRight();
        app.Screenshot("Swiped right");
        app.Tap(x => x.Marked("Save"));
    }
```

Nota: Em *Xamarin.iOS* ou *Xamarin.Android* normalmente os elementos gráficos são identificados pelo valor dos atributos "*Text*" ou do "*Id*". Em *Xamarin Forms* são identificados pelo atribuito "StyleId" (que tem apenas este propósito), no entanto é preciso:

- Em iOS, no método *AppDelegate.FinishedLaunching* depois da inicialização do *Forms.Init* definir a atribuição do *StyleId* ao atributo *AccessibilityIdentifier*.

```
Forms.ViewInitialized += (object sender, ViewInitializedEven-
    tArgs e) =>

{

  if (null != e.View.StyleId) {

    e.NativeView.AccessibilityIdentifier = e.View.StyleId;

  }

};
```

- Em Android, no método *MainActivity.OnCreate* depois da inicialização do *Forms.Init* definir a atribuição do *StyleId* ao atributo *ContentDescription*.

```
Xamarin.Forms.Forms.ViewInitialized += (object sender,
    Xamarin.Forms.ViewInitializedEventArgs e) =>

{

  if (!string.IsNullOrWhiteSpace(e.View.StyleId)) {

    e.NativeView.ContentDescription = e.View.StyleId;

  }

};
```

Xamarin Test Cloud

O *Xamarin Test Cloud, disponível em testcloud.xamarin.com*, é um "serviço na nuvem" disponibilizado pela Xamarin que permite testar automáticamente as aplicações móveis com base nos *UI Acceptance Testing*, em centenas de dispositivos, fornecendo informação sobre a performance e eficiência da aplicação. Claro está, o resultado final da execução dos testes nos diversos dispositivos vai permitir concluir se a aplicação executa corretamente e se esta está apta para ser publicada nas diversas lojas. Os dispositivos disponíveis pelo serviço podem ser consultados na seguinte referência: https://testcloud.xamarin.com/devices.

O custo do serviço é baseado no tempo de execução dos testes nos dispositivos selecionados, sendo excluido o

custo de manutenção e provisionamento. Para consultar o preço dos planos disponíveis consultar a referência: https://xamarin.com/test-cloud.

De uma forma geral, o fluxo do processo de execução dos testes no serviço Test Cloud, é definido por (ver figura 23):

- O programador publica o(s) teste(s) e o(s) pacote(s) da aplicação no Test Cloud
- Seleciona os dispositivos onde pretende executar a aplicação
- A aplicação é instalada nos dispositivos e os testes são executados nos dispositivos
- Para cada dispositivo é recolhido os screenshots e resultados da execução do teste
- Os resultados são fornecidos ao programador

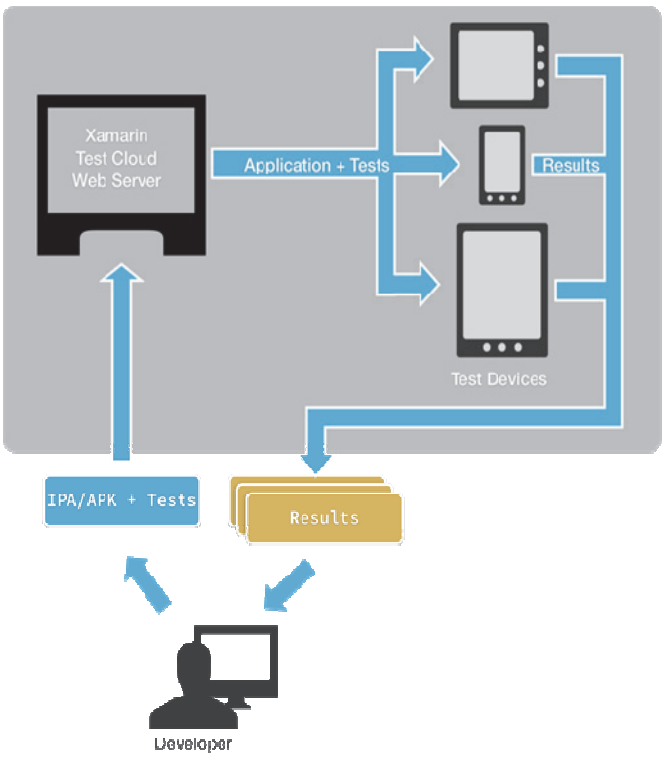


Figura 23 Workflow do processo de testes usando o Test Cloud

Anatomia da framework do Test Cloud

O Test Cloud suporta duas framework Calabash e Xamarin.UITest que já referimos anteriormente. Esta bibliotecas requerem o Xamarin Test Cloud Agent que irá fazer com que os testes sejam executados nos dispositivos disponibilizados pelo serviço, e tem como base um modelo cliente-servidor cuja comunicação é feita via json através pedidos HTTP.

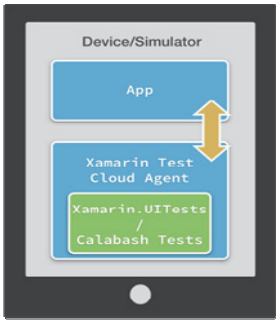


Figure 1 Xamarin Test Cloud Agent & App

O modo de funcionamento deste agente nas plataformas suportadas, iOS e Android, é diferente:

- Em iOS o agente é instalado na própria aplicação, via pacote Nuget
- Em Android o agente é uma aplicação à parte (dai o requisito das permissões de acesso à internet) que comunica via http

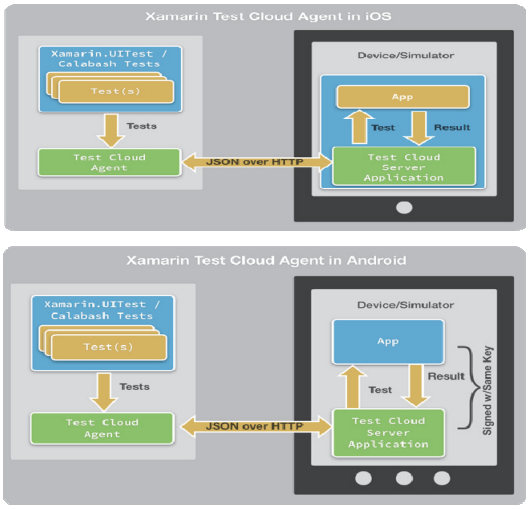


Figura 24 Xamarin Test Cloud Agent em iOS e Android

Aplicações que estejam integradas num sistema de integração contínua, podem-se conetar ao Test Cloud para que seja feita a verificação dos testes de aceitação sobre as alterações realizadas. Sendo o fluxo do processo descrito pela figura 25.

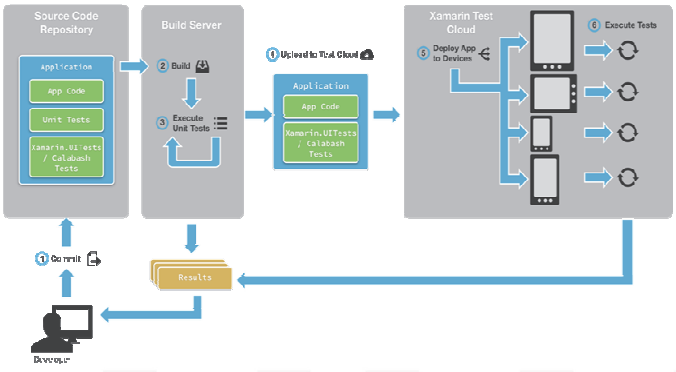


Figura 25 Workflow usando integração continua

Publicação para o Xamarin Test Cloud

A publicação da aplicação e respetivos testes no Test Cloud pode ser feita:

- Usando o Xamarin Studio, na vista "Unit Testing", escolhendo a opção "Run in Test Cloud"
- Usando o site testcloud.xamarin.com e a linha de comandos

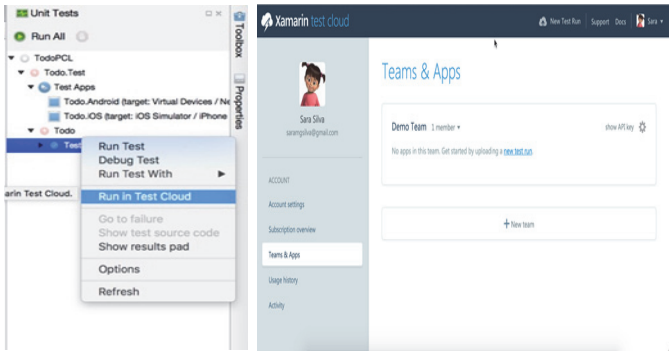


Figura 25 Publicação dos testes via Xamarin Studio e via testcould.xamarin.com

Xamarin Studio

No cenário em que usamos o Xamarin Studio, o pacote da aplicação e os testes são enviados para o Test Cloud:

```
/Library/Frameworks/Mono.framework/Versions/4.2.2/bin/mono "/Users/saramgsilva/DevApps/XTC/Todo/PCL/packages/Xamarin.UITest.1.1.1/tools/test-cloud.exe" submit "/Users/saramgsilva/DevApps/XTC/Todo/PCL/Todo.Android/bin/Debug/Todo.Android.apk" --async --fixture Todo.Test.Tests(Android) --app-name Todo.Android --assembly-dir "/Users/saramgsilva/DevApps/XTC/Todo/PCL/Todo.Test/bin/Debug"

Negotiating file upload to Xamarin Test Cloud.

Posting to https://testcloud.xamarin.com/ci/anonymous_uploads


Uploading Xamarin.UITest.dll ... Already uploaded.

Uploading nunit.framework.dll ... Already uploaded.

Uploading AndroidTestServer.apk ... Already uploaded.

Uploading Todo.Android_resigned.apk ... Already uploaded.


Uploading Todo.Test.dll... 100%

Uploading Todo.Test.dll... 100%

Upload complete.


Skipping wait for cloud execution. Upload Id: c043fdb4-8bcb-4ff8-29a8-45e7fb32298e
```

Sendo de seguida lançado no browser, o testcloud.xamarin.com, que irá permitir

- Escolher os dispositivos

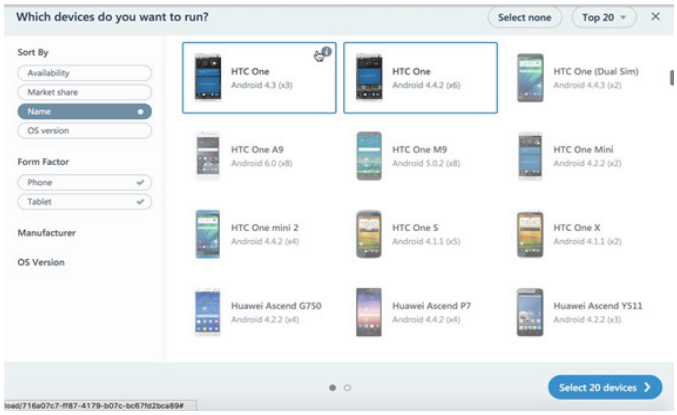


Figura 26 Seleção dos dispositivos

- Configurar o ambiente de testes

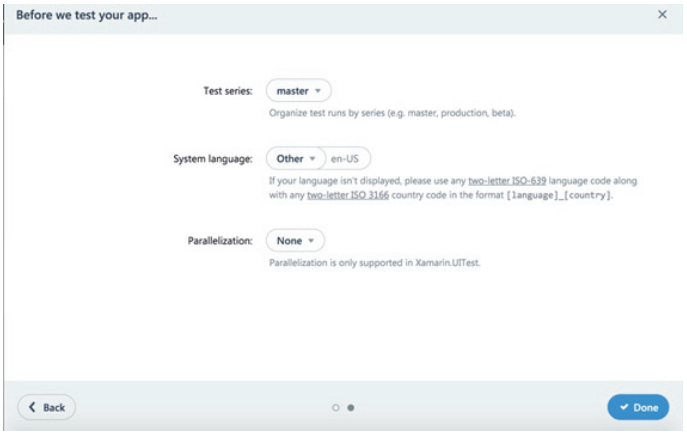


Figura 27 Configurações



Figura 28 Execução dos testes em progresso

TESTAR APLICAÇÕES MÓVEIS COM XAMARIN TEST CLOUD

Linha de comandos

No cenário em que usamos a linha de comandos, vamos executar algo do género:

- Em MAC

```
mono packages/Xamarin.UITest.[version]/tools/test-cloud.exe submit yourAppFile.apk 4b84c09bc46e247c35cad0f0dad5ae8e --devices 7492d185 --series "master" --locale "en_US" --app-name "Todo" --user nome@mail.com --assembly-dir path-ToTestDllFolder
```

- Em Windows

```
packages\Xamarin.UITest.[version]\tools\test-cloud.exe submit yourAppFile.apk 4b84c09bc46e247c35cad0f0dad5ae8e --devices d1a15df1 --series "master" --locale "en_US" --app-name "Todo" --user nome@mail.com --assembly-dir path-ToTestDllFolder
```

Resultado dos testes

Como podemos ver no tópico anterior, a execução de testes no Test Cloud é um processo muito simples. No fim da execução de todos os testes, é gerado um relatório como podemos ver na figura 29 e 30.

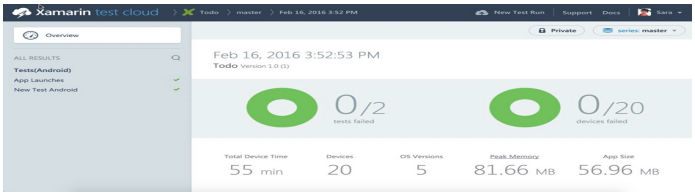


Figura 29 Painel com resumo da execução dos testes

De salientar, que neste painel é possivel analisar o tempo de execução, o número de dispositivos para o qual os testes passaram com sucesso, a versão da plataforma, o total de memória usado e o tamanho da aplicação. Métricas estas que podem depois ser usadas para análise estatística entre as sucessivas execuções da aplicação.

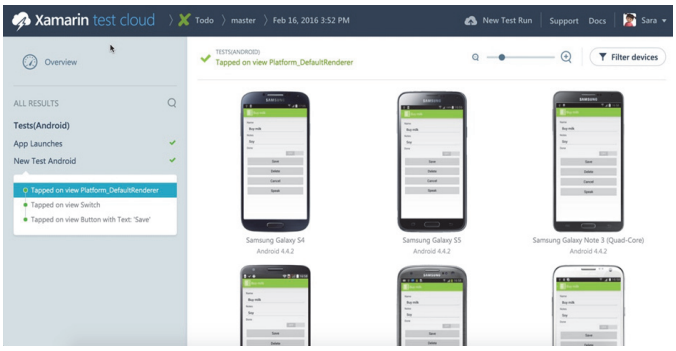


Figura 30 Relatório final por dispositivo

Para terminar, é de referir que o relatório final apresenta as diversas operações que o teste executa, e para cada uma das operações diz se executou com sucesso ou não. Neste caso simples, da aplicação TODO, todos os passos foram efetuados com sucesso sendo possível constatar isso mesmo nos screenshots apresentados na figura 30. No entanto, podemos consultar a demo online, disponibilizada pela Xamarin, na referência https://xamarin.com/test-cloud/demo, que mostra as metricas, screenshoots e erros da aplicação *Account Verification*.

Conclusão

Em conclusão, podemos concluir que a Xamarin está a apostar nas ferramentas, APIs e serviços para melhorar as aplicações móveis, sendo o Xamarin Test Cloud uma solução que nos vai permitir identificar os diversos problemas que muitas vezes é impossível detetar em processos manuais, e claro está a framework *Xamarin.UITest* em conjunto com o *Xamarin Test Record* facilita todo o processo automático referente à criação dos testes de aceitação. A diversidade de dispositivos, os filtros disponíveis, o detalhe do resultado dos testes e o ambiente oferecido é sem dúvidas as mais valias para quem usa o serviço, que tem apresentado uma evolução constante de acordo com as necessidades do mercado.

Referências

- Introduction to Xamarin.UITest
- Xamarin.UITest Quickstarts
- Working with Xamarin.UITest
- Adding Xamarin.UITest to a Solution
- Xamarin.UITest Cheat Sheet
- Introduction to Xamarin Test Cloud
- Xamarin Test Recorder
- Submitting UITests to Xamarin Test Cloud

AUTOR



Escrito Por Sara Silva

Licenciada em Matemática pelo DMUC, e o seu foco de desenvolvimento está direccionado para a área Mobile. Atualmente desenvolve na área do Windows, Xamarin, Azure, e é Microsoft MVP Mentor. A Sara foi condecorada com vários prémios com especial destaque: Microsoft MVP, Xamarin MVP, Telerik Developer Especialista, C# Corner MVP, TechNet Wiki - Technical Guru. O trabalho que vai sendo desenvolvido pela Sara pode ser seguido através do seu blog www.saramgsilva.com e do twitter é @saramgsilva.

A PROGRAMAR

Extrair dados do cartão de cidadão e validação dos números de identificação Português Europeu com JAVA Swing

Android Monkey Test: Um “Macaco” ao Serviço dos Programadores Android

Programação (in)Segura - Transbordo de Memória

ASP.NET Core 1.0 – Ligação a base de dados SQL Azure

Extrair dados do cartão de cidadão

e validação dos números de identificação Português Europeu com JAVA Swing

Neste artigo vou demostrar como se podem obter dados do cartão de cidadão utilizando a linguagem de programação JAVA e validar os dados de identificação.

Porquê Java? Não vejo muitos artigos a utilizar Java e é uma linguagem que continua a ser utilizada nas universidades, nas empresas de desenvolvimento de software e não só. Já vi em fóruns de comunidades pedidos de ajuda para obter os dados do cartão de cidadão, mas o pedido que mais encontro é como obter a fotografia. Este artigo foi feito a pensar nessas questões. Também inclui funções de validação do número de bilhete de identidade e cartão de cidadão, número de identificação fiscal, número de identificação da segurança social e número de identificação bancária.

Sobre o cartão de cidadão

O cartão de cidadão veio substituir vários cartões de identificação Portugueses para um único cartão de identificação mas este cartão não vem apenas reduzir a quantidade de cartões que o cidadão tem que transportar mas permite também fazer algumas operações utilizando os serviços público do estado português através da internet e assinar documentos eletrónicos.

O cartão de cidadão tem incluído um chip ISO/IEC 7816 é neste chip que está guardado toda a informação.

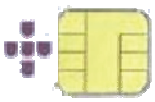


Ilustração 1 Chip ISO/IEC 7816

Esta informação está bloqueada? Podemos utilizar?

Informação existente no cartão não está toda disponível. Existem casos como a morada e certificados privados que apenas podem ser acedidos com pins que foram fornecidos juntamente com o cartão.

A informação a que se tem acesso é:

Área	Sobre o cartão	Sobre o Cidadão
Dados	Data de entrega, entidade que efetuou a entrega, local de pedido, número de série, número do cartão, país, tipo de documento, validade, versão, zona de leitura óptica.	Altura, apelido, apelido da mãe, apelido do pai, certificados digitais (autenticação), data de nascimento, fotografia, indicações eventuais, nacionalidade, número de identificação civil, número de identificação fiscal, número da segurança social, número do sistema nacional de saúde, primeiro nome, primeiro nome da mãe, primeiro nome do pai, sexo.

Mas que podemos nós fazer alguma coisa com esta informação?

Com os dados existentes podemos criar vários projetos como por exemplo autenticação de utilizador no Windows Active Directory (https://www.cartaodecidadao.pt/documentos/CC_MW_Windows_Logon_V1.0.pdf) , assinar documentos, recrutamento e outros.

Material necessário para um projeto do cartão de cidadão

Antes de iniciar um projeto é recomendado que se tenha disponível um leitor de cartões e um cartão de cidadão. Podemos fazê-lo de duas formas.

- Comprar um Kit de desenvolvimento que está disponível na página de internet do Kit de desenvolvimento do cartão de cidadão em http://www.kitcc.pt/ccidadao/kits.
- Comprar um leitor de cartão de cidadão e usar o nosso próprio cartão de cidadão.

A diferença entre os dois é que o kit de desenvolvimento já inclui cartões com estado válido, revogado, suspenso, expirado, não ativado. O nosso próprio cartão está sempre válido e se se enganarem no pin mais do que três vezes o cartão fica bloqueado e para algumas operações é necessário deslocar-se á sua junta de freguesia ou loja do cidadão para solicitar um novo pin que é enviado por via postal.

Neste artigo pode utilizar o seu próprio cartão de cidadão. Mas se criar um projeto de autenticação recomendo comprar um kit de desenvolvimento.

Middleware do cartão de cidadão

É sempre necessária a instalação da aplicação do Cartão de Cidadão disponível em https://www.cartaodecidadao.pt. Esta aplicação permite ver os dados existentes do cartão de cidadão que está a ser utilizado mas também atua como um middleware.

Para o desenvolvimento vai ser utilizado uma biblioteca com o nome de pteidlibj. Esta biblioteca está disponível para desenvolvimento de aplicações .NET e Java, em duas formas:

- Na diretoria de instalação existe uma diretoria com o nome “sdk”.
- Middleware que está disponível na área de downlo-

ads na página de internet do Kit de desenvolvimento do cartão de cidadão.

Tanto um com o outro já inclui um projeto de exemplo para .NET e Java. Existe o manual técnico de middleware em: https://www.cartaodecidadao.pt/documentos/CC_Technical_Reference_1.60.0_PT.pdf e também está disponível na área de downloads na página de internet do Kit de desenvolvimento do cartão de cidadão.

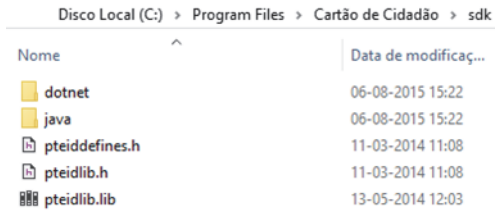


Ilustração 2 pasta do cartão de cidadão com SDK

Obtenção da fotografia

As questões que mais encontro sobre a obtenção de dados do cartão de cidadão são "Como é possível obter a fotografia do cartão de cidadão?" e "Qual é o formato do cartão de cidadão?". O formato de imagem do cartão de cidadão é JPEG2000 (http://jpeg.org/jpeg2000/) que é um padrão de alta definição e que permite comprimir a imagem. Para visualizar a foto é necessário ler a imagem com este formato que depois pode converter em memória para outro formato se pretende guardar. No projeto de Java vai ser utilizado o Java Advanced Imagin (Jai-imageio) versão 1.1. As versões superiores foram removidas o suporte para JPEG2000.

Para criar a miniatura da fotografia vai ser utilizado o Java Image Scaling Library(imgscalr).

Nota: Se não carregarem a livraria Jai-imageio no projeto quando obterem a fotografia não vão obter nenhum erro de falha de carregamento da livraria apenas vão obter um valor nulo.

Vamos programar

Neste artigo vou utilizar o IDE Netbeans mas pode utilizar outro IDE a sua escolha.

Vão ser criadas:

- Uma classe com as caixas de diálogo estáticas para as mensagens de erro e de informação;
- Uma classe para obtenção dos dados do cartão de cidadão e controlo de erros;
- Um formulário JFrame que já inclui o main e todos os elementos gráficos;
- Uma classe para validação de números de identificação.

Crie um novo projeto "Java Application" com o nome á sua escolha sem a pasta de ficheiros para guardar as bibliotecas e classe main definidas. Depois crie um "Java Package" com o nome "PaPCC" é neste Package que vão estar todas as Java class e JFrame.

Classe com as caixas de diálogo para as mensagens de erro e informação

Esta classe inclui as funções estáticas para mostrar as mensagens de informação e erro.

Crie uma nova Java classe com o nome "JMessageDialog" dentro do package "PaPCC".

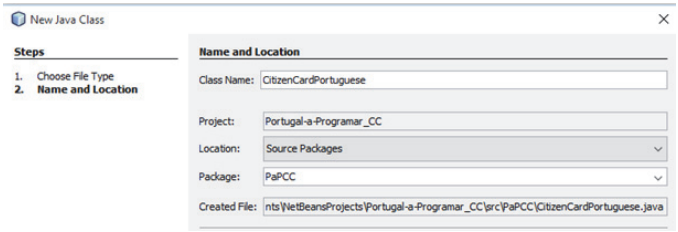


Ilustração 3 Nova Java Class JMessageDialog.java

Incluía as seguintes importações uma vez que vão ser necessárias para as funções.

```
import javax.swing.JFrame;
import javax.swing.JOptionPane;
```

E depois as seguintes funções estáticas para apresentar as mensagens.

```
/**
    * Mostra uma janela com informação do erro
    * @param title título do erro
    * @param message mensagem do erro
    */
public static void erroMsg(String title, String
                                        message) {
    JOptionPane.showMessageDialog(new JFrame(),
      message, title, JOptionPane.ERROR_MESSAGE);
}

/**
    * Mostra uma janela com informação
    * @param title título da informação
    * @param message mensagem da informação
    */
    public static void infoMsg(String title,
                                    String message) {
    JOptionPane.showMessageDialog(new JFrame(),
message, title, JOptionPane.INFORMATION_MESSAGE);
}
```

A classe está terminada.

Classe Obtenção dos dados do cartão de cidadão e controlo de erros

Esta classe inclui todas as variáveis e funções de leitura e dados do cartão de cidadão.

Crie uma nova Java classe com o nome "CitizenCardPortuguese" no package "PaPCC" e inclua os

EXTRAIR DADOS DO CARTÃO DE CIDADÃO

seguintes importações que vão ser necessárias para as funções.

```
import static PaPCC.JMessageDialog.erroMsg;
import java.awt.image.BufferedImage;
import java.io.ByteArrayInputStream;
import java.security.AccessController;
import java.security.PrivilegedAction;
import java.util.Iterator;
import javax.imageio.ImageIO;
import javax.imageio.ImageReader;
import javax.imageio.stream.ImageInputStream;
import javax.swing.JOptionPane;
import pteidlib.PTEID_ADDR;
import pteidlib.PTEID_ID;
import pteidlib.PTEID_PIC;
import pteidlib.PTEID_Pin;
import pteidlib.PteidException;
import pteidlib.pteid;
```

Primeiro vão ser definidos todas as variáveis dos dados do cartão, pins e tentativas

```
private String birthDate;
private String deliveryEntity;
private String cardNumber;
private String cardNumberPAN;
private String cardVersion;
private String country;
private String documentType;
private String firstname;
private String firstnameFather;
private String firstnameMother;
private String height;
private String locale;
private String mrz1;
private String mrz2;
private String mrz3;
private String name;
private String nameFather;
private String nameMother;
private String nationality;
private String notes;
private String numBI;
private String numNIF;
private String numSNS;
private String sex;
private String validityDate;
private String numSS;
private String deliveryDate;
private String addrType;
private String street;
private String municipality;
private String addressF;
private String building;
private String buildingAbbr;
private String countryM;
private String countryDescF;
private String cp3;
private String cp4;
private String district;
private String districtDesc;
private String door;
private String floor;
private String freguesia;
private String freguesiaDesc;
private String locality;
private String localityF;
private String municipalityDesc;
private String numMor;
private String numMorF;
private String postal;
private String postalF;
private String place;
private String regioF;
private String side;
private String streettype;
private String streettypeAbbr;
private String cityF;
private int authPinTriesLeft;
private int sigPinTriesLeft;
private int addrPinTriesLeft;
private String addr = null;
```

A seguir inclua os getters das variáveis definidas. No Netbeans pode fazer esta operação automaticamente clicando no botão direito do rato selecionado a opção “Insert Code…” a seguir a opção “Getter” selecione todas as variáveis e por fim clique no botão “Generate”.

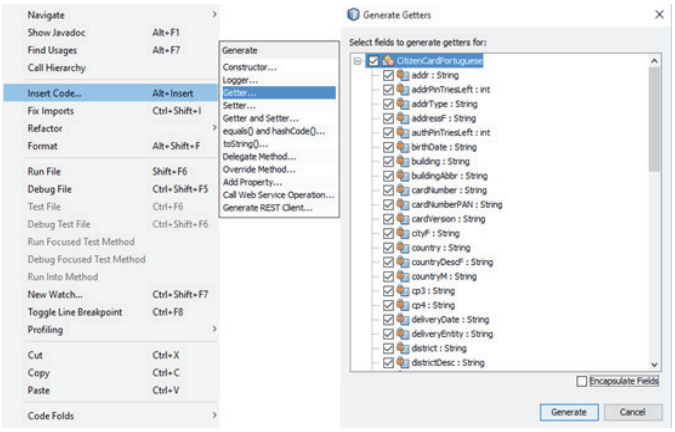


Ilustração 4 Getter das varáveis

A seguinte função é usada para carregar o middleware e que permite saber se a biblioteca foi carregada com sucesso ou se existe algum problema. Esse resultado é dado ou fornecido através de um valor booleano.

```
/**
* Carrega a biblioteca pteidlibj
*
* @return um valor booleano se o middleware foi
carregado com sucesso
* @throws UnsatisfiedLinkError erro do middleware
*/

public static boolean loadPteidLib() throws
                          UnsatisfiedLinkError {
    return ((Boolean)
                  AccessController.doPrivileged(
                        new PrivilegedAction() {
        @Override
        public Boolean run() {
            try {
                System.loadLibrary("pteidlibj");
                return true;
            } catch (UnsatisfiedLinkError t) {
                if (t.getMessage().contains
                              ("already loaded")) {
                    JOptionPane.showMessageDialog
          (null, "Biblioteca do Cartão de Cidadão
             bloqueada.", "Biblioteca bloqueada",
                        JOptionPane.ERROR_MESSAGE);
                } else {
                    JOptionPane.showMessageDialog
          (null, "Middleware do Cartão de Cidadão
             não está instalado.", "Aplicação não
      está instalada", JOptionPane.ERROR_MESSAGE);
                }
```

EXTRAIR DADOS DO CARTÃO DE CIDADÃO

```
                return false;
            }
        }
    }));
}
```

A obtenção dos dados do cartão de cidadão vai ser efetuado por partes, primeiro vão ser obtidos os dados sem necessitar de pin de acesso, depois os dados da morada que requer o pin e por fim a fotografia.

A seguinte função destina-se a obter os dados do cartão de cidadão sem pin de acesso e com um valor booleano vamos saber se foi obtido com sucesso.

```
/**
* Obtenção dos dados do cidadão
*
* @return um valor booleano se os dados foram obti-
dos com sucesso
* @throws PteidException erro do middleware
*/
public boolean getData()
    throws PteidException {
try {
    pteid.Init("");

    pteid.SetSODChecking(false);

    PTEID_ID idData = pteid.GetID();
    if (null != idData) {
        this.deliveryEntity =
                            idData.deliveryEntity;
        this.birthDate = idData.birthDate;
        this.cardNumber = idData.cardNumber;
        this.cardNumberPAN = idData.cardNumberPAN;
        this.cardVersion = idData.cardVersion;
        this.country = idData.country;
        this.documentType = idData.documentType;
        this.firstname = idData.firstname;
        this.firstnameFather =
                           idData.firstnameFather;
        this.firstnameMother =
                           idData.firstnameMother;
        this.height = idData.height;
        this.locale = idData.locale;
        this.mrz1 = idData.mrz1;
        this.mrz2 = idData.mrz2;
        this.mrz3 = idData.mrz3;
        this.name = idData.name;
        this.nameFather = idData.nameFather;
        this.nameMother = idData.nameMother;
        this.nationality = idData.nationality;
        this.notes = idData.notes;
        this.numBI = idData.numBI;
        this.numNIF = idData.numNIF;
        this.numSNS = idData.numSNS;
        this.sex = idData.sex;
        this.validityDate = idData.validityDate;
        this.numSS = idData.numSS;
        this.deliveryDate = idData.deliveryDate;

        PTEID_Pin[] pin = pteid.GetPINs();
        this.authPinTriesLeft = pin[0].triesLeft;
        this.sigPinTriesLeft = pin[1].triesLeft;
        this.addrPinTriesLeft = pin[2].triesLeft;
        }

    } catch (PteidException ex) {
        int errorNumber = Integer.parseInt
    (ex.getMessage().split("Error code : -")[1]);
        errorCC(errorNumber, ex.getMessage());
            return false;
        } finally {
            pteid.Exit(0);
        }
        return true;
}
```

A seguinte função permite obter a morada do cartão de cidadão. Se experimentar com o cartão de cidadão com versão superior ou igual a 004.003 o PIN público é “0000” este pin apenas permite consultar a morada não permite alterar os dados da morada.

```
/**
* Obtenção da morada
*
* @return um valor booleano se os dados foram obti-
dos com sucesso
* @throws PteidException erro do middleware
*/
public boolean getAddress() throws PteidException {
try {
    pteid.Init("");
    pteid.SetSODChecking(false);
    PTEID_ADDR adData = pteid.GetAddr();
    if (null != adData) {
        if ("N".equals(adData.addrType)) {
            this.addrType = "Nacional";
            this.district = adData.district;
            this.districtDesc =
                                adData.districtDesc;
            this.municipality =
                                adData.municipality;
            this.municipalityDesc =
                             adData.municipalityDesc;
            this.freguesia = adData.freguesia;
            this.freguesiaDesc =
                               adData.freguesiaDesc;
            this.streettypeAbbr =
                              adData.streettypeAbbr;
            this.streettype = adData.streettype;
            this.street = adData.street;
            this.buildingAbbr =
                                adData.buildingAbbr;
            this.building = adData.building;
            this.door = adData.door;
            this.floor = adData.floor;
            this.side = adData.side;
            this.place = adData.place;
            this.locality = adData.locality;
            this.cp3 = adData.cp3;
            this.cp4 = adData.cp4;
            this.postal = adData.postal;
            this.country = adData.country;
            this.numMor = adData.numMor;
        } else {
            this.addrType = "Internacional";
            this.district = adData.district;
            this.districtDesc =
                                adData.districtDesc;
            this.municipality =
                                adData.municipality;
            this.municipalityDesc =
                             adData.municipalityDesc;
            this.freguesia = adData.freguesia;
            this.freguesiaDesc =
                               adData.freguesiaDesc;
            this.streettypeAbbr =
                              adData.streettypeAbbr;
            this.streettype = adData.streettype;
            this.street = adData.street;
            this.buildingAbbr =
                                adData.buildingAbbr;
```

EXTRAIR DADOS DO CARTÃO DE CIDADÃO

```
            this.building = adData.building;
            this.door = adData.door;
            this.floor = adData.floor;
            this.side = adData.side;
            this.place = adData.place;
            this.locality = adData.localityF;
            this.cp3 = adData.cp3;
            this.cp4 = adData.cp4;
            this.postal = adData.postalF;
            this.country = adData.countryDescF;
            this.countryDescF =
                                adData.countryDescF;

            this.addressF = adData.addressF;
            this.cityF = adData.cityF;
            this.numMor = adData.numMorF;
            this.regioF = adData.regioF;
        }
    }
    } catch (PteidException ex) {
        int errorNumber = Integer.parseInt
      (ex.getMessage().split("Error code : -")[1]);
        errorCC(errorNumber, ex.getMessage());
        return false;
    } finally {
        pteid.Exit(0);
    }
    return true;
}
```

A seguinte função é para obter a fotografia.

```
/**
* Obtenção da fotografia
*
* @return um valor booleano se os dados foram obti-
dos com sucesso
* @throws PteidException erro do middleware
* @throws Exception erro do InputStream
*/
public BufferedImage getPhoto() throws
                        PteidException, Exception {
    BufferedImage bimg = null;
    try {
        pteid.Init("");
        pteid.SetSODChecking(false);
        PTEID_PIC picData = pteid.GetPic();
        if (null != picData) {
            byte[] byteImg = picData.picture;
            Iterator iterator =
   ImageIO.getImageReadersByFormatName("jpeg2000");
            while (iterator.hasNext()) {
                try (ImageInputStream iis =
                 ImageIO.createImageInputStream(new
                  ByteArrayInputStream(byteImg))) {
                    ImageReader reader =
                     (ImageReader) iterator.next();
                    reader.setInput(iis);
                    bimg = reader.read(0,
                     reader.getDefaultReadParam());
                }
            }
        }
    } catch (PteidException ex) {
        int errorNumber = Integer.parseInt
(ex.getMessage().split("Error code : -")[1]);
        errorCC(errorNumber, ex.getMessage());
    } finally {
        pteid.Exit(0);
    }
    return bimg;
}
```

As seguintes funções são para identificar o código de erros que o middleware retorna. Estes erros estão documentados no manual técnico com na secção “Códigos de Erro”.

```
/**
* Lista de códigos de erro do middleware
*
* @param errorNumber código de erro
* @param ex mensagem de erro auxiliar
*/
public void errorCC(int errorNumber, String ex) {
    String message;
    switch (errorNumber) {
        case 1101:
            message = ("Erro desconhecido -
     Problemas com o serviço de leitor de cartões
                                \n Message: " + ex);
            erroMsg(message);
            break;
        case 1104:
            message = ("Não foi possível aceder
      ao cartão.\n Verifique se está corretamente
                               inserido no leitor");
            erroMsg(message);
            break;
        case 1109:
            message = ("Acão cancelada pelo
                                       utilizador");
            erroMsg(message);
            break;
        case 12109:
            message = ("Não é permitido.");
            erroMsg(message);
            break;
        case 1210:
            message = ("O cartão inserido não
    corresponde a um cartão de cidadão válido.");
            erroMsg(message);
            break;
        case 1212:
            message = ("Pin de morada bloqueado.
               Resta(m) 0 tentativa(s).\n" + ex);
            erroMsg(message);
            break;
        case 1214:
            message = ("Pin inválido, não tente
                                novamente.\n" + ex);
            erroMsg(message);
            break;
        case 1304:
            message = ("Pin inválido, não tente
                                novamente.\n" + ex);
            erroMsg(message);
            break;
        default:
            message = ("Erro desconhecido: " +
                                               ex);
            erroMsg(message);
            break;
    }
}
```

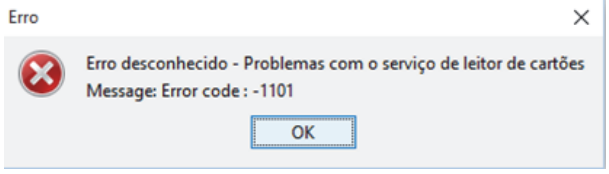


Ilustração 5 Exemplo o leitor de cartões não se encontra ligado

E temos a nossa classe terminada.

Formulário da interface gráfica

Crie um novo JFRAME no package "PaPCC" e incluía os seguintes importações que vão ser necessárias para as funções.

```
import static PaPCC.JMessageDialog.erroMsg;
import static PaPCC.JMessageDialog.infoMsg;
import java.awt.Image;
import java.awt.image.BufferedImage;
import javax.swing.ImageIcon;
import javax.swing.UIManager;
import javax.swing.UnsupportedLookAndFeelException;
import org.imgscalr.Scalr;
```

Insira a seguinte variável que representa um objecto da classe CitizenCardPortuguese.

```
private final CitizenCardPortuguese ccp = new
                                CitizenCardPortuguese();
```

No formulário insira um botão que vai correr a instrução de ler os dados e mostrar nos campos. Todos os campos são caixa de texto (JText Field) não editável. O campo notas é um JText Área e a fotografia é um JLabel. Pode dividir a informação a mostrar por secções utilizando separadores (Tabbed Pane) por exemplo Identificação fiscal, Outros dados e Morada. Se pretender pode separar por JPanel num inclui os separadores e noutro a fotografia.

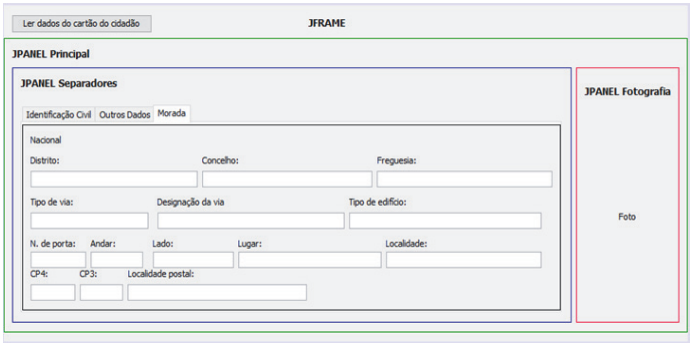


Ilustração 6 JFrame com os JPanel identificados

Todos os campos com o botão, separador identificação fiscal e fotografia.

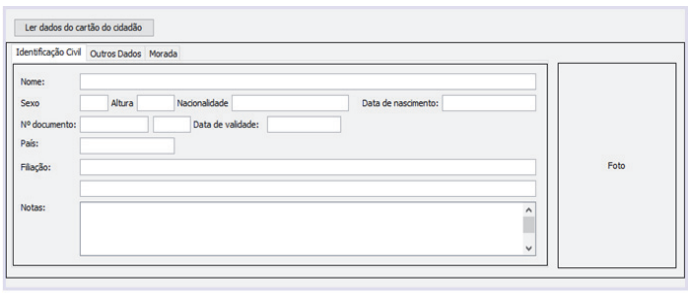


Ilustração 7 Formulário com os campos Identificação da fiscal e fotografia

Todos os campos do separador outros dados.

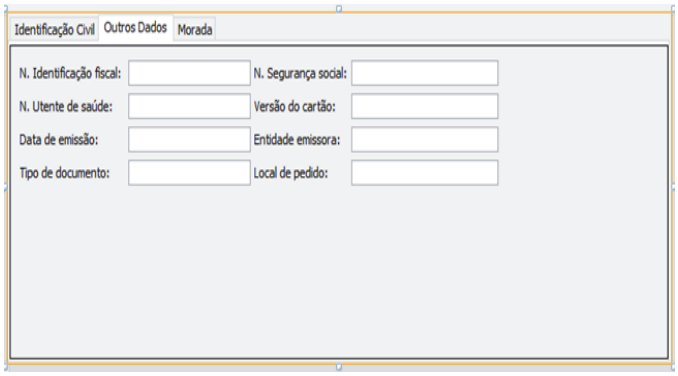


Ilustração 8 Formulário cm os campos de Outros Dados

Todos os campos do separador morada.

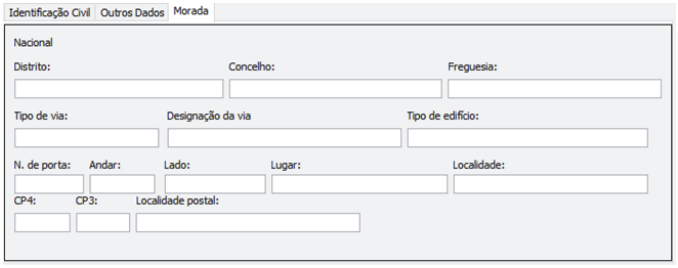


Ilustração 9 Formulário com os campos da Morada

A seguinte função é para obter os dados identificação fiscal e outros dados do objeto e mostrar nas caixas de texto.

```
/**
* Leitura dos dados de identificação fiscal e ou-
tros dados
* @return um valor booleano se foi lido com su-
cesso
*/
private Boolean readCCData() {
    try {
        if (ccp.getData() != false) {
            jTname.setText(ccp.getFirstname() + "
                                " + ccp.getName());
            jTsex.setText(ccp.getSex());
            jTheight.setText(ccp.getHeight());
            jTnationality.setText
                            (ccp.getNationality());
            jTbirthDate.setText(ccp.getBirthDate
                        ().replaceAll("\\s", "-"));
            jTnumBI.setText(ccp.getNumBI
                            ().substring(0, 8));
            jTnumBI2.setText(ccp.getCardNumber
      ().substring(8, 14).replaceAll("\\s", ""));
            jTvalidityDate.setText
  (ccp.getValidityDate().replaceAll("\\s", "-"));
            jTcountry.setText(ccp.getCountry());
            jTnameFather.setText
                (ccp.getFirstnameFather() + " " +
                            ccp.getNameFather());
            jTnameMother.setText
                (ccp.getFirstnameMother() + " " +
                            ccp.getNameMother());
            jTnotes.setText(ccp.getNotes());
            jTnumNIF.setText(ccp.getNumNIF());
            jTnumSS.setText(ccp.getNumSS());
            jTnumSNS.setText(ccp.getNumSNS());
            jTcardVersion.setText
```

EXTRAIR DADOS DO CARTÃO DE CIDADÃO

```
                                (ccp.getCardVersion());
            jTdeliveryDate.setText
    (ccp.getDeliveryDate().replaceAll("\\s", "-"));
            jTdeliveryEntity.setText
                           (ccp.getDeliveryEntity());
            jTdocumentType.setText
                            (ccp.getDocumentType());
            jTlocale.setText(ccp.getLocale());
            return true;
        }
    } catch (Exception ex) {
      erroMsg("Erro dados civil", ex.getMessage());
    }
    return false;
}
```

A seguinte função é para obter a morada do objeto e mostrar nas caixas de texto referente a morada.

```
/**
* Leitura da morada
* @return um valor booleano se foi lido com sucesso
*/
private boolean readCCAddress() {
    try {
        if (ccp.getAddrPinTriesLeft() != 0) {

            infoMsg("Obter dados da morada", "Vai
 ser solicitado o pin da morada,\n0 pin por defeito
 é:\n" + " - cartão com a versão infeiror a 004.003
 é : é desconhecido \n" + " - cartão com a versão
                    superior a 004.003 é : 0000.");
            if (ccp.getAddress() != false) {
             jLNacional.setText(ccp.getAddrType());
                jTdistrictDesc.setText
                            (ccp.getDistrictDesc());
                jTmunicipalityDesc.setText
                       (ccp.getMunicipalityDesc());
                jTfreguesiaDesc.setText
                      (ccp.getFreguesiaDesc());
                jTstreettype.setText
                        (ccp.getStreettype());
                jTstreet.setText(ccp.getStreet());
             jTbuilding.setText(ccp.getBuilding());
                jTdoor.setText(ccp.getDoor());
                jTfloor.setText(ccp.getFloor());
                jTside.setText(ccp.getSide());
                jTplace.setText(ccp.getPlace());
             jTlocality.setText(ccp.getLocality());
                jTcp4.setText(ccp.getCp4());
                jTcp3.setText(ccp.getCp3());
                jTpostal.setText(ccp.getPostal());
            } else {
                return false;
            }
            return true;
        } else {
            erroMsg("Erro", "Não é possível obter a
    morada.\nPin de morada está bloqueado. Resta(m)
          0 tentativa(s).\nNão foi possível obter a
                                        morada.\n");
            return false;
        }
    } catch (Exception ex) {
        erroMsg("Erro morada", ex.getMessage());
        return false;
    }
}
```

Existem duas maneiras de obter a fotografia e convertê-la para um ícone de JLabel à escala. Uma é a utilização de escala com The Abstract Window Toolkit (AWT) a outra é criação de uma miniatura com Java Image Scaling Library e com a conversão da imagem para Buffered image.

Função apenas com AWT.

```
/**
* Função da leitura da fotografia
*
* @return um valor booleano se foi lido com su-
cesso
*/
private boolean readCCPhoto() {
    try {
        jLccPhoto.setText("");
        BufferedImage xpto = ccp.getPhoto();
        Image img = xpto;
        Image newimg = img.getScaledInstance(130,
               180, java.awt.Image.SCALE_SMOOTH);
        ImageIcon newIcon = new ImageIcon
                                          (newimg);
        jLccPhoto.setIcon(newIcon);
        return true;
    } catch (Exception ex) {
     erroMsg("Erro obter foto", ex.getMessage());
        return false;
    }
}
```

Função com Java Image Scaling Library.

```
/**
* Função da leitura da fotografia
*
* @return um valor booleano se foi lido com su-
cesso
*/
private boolean readCCPhoto() {
      try {
            jLccPhoto.setText("");
            BufferedImage bi = ccp.getPhoto();
            Image img = bi;
            if (img != null) {
                  Image newimg =
                   img.getScaledInstance(130, 180,
                     java.awt.Image.SCALE_SMOOTH);
                  jLccPhoto.setText(null);
                  jLccPhoto.setIcon(new ImageIcon
      (Scalr.resize(imageToBufferedImage(newimg),
                                          140)));
                  return true;
            } else {
                  jLccPhoto.setText("Foto");
                  erroMsg("Erro obter foto",
       "Fotografia não foi obtida com sucesso.");
                  return false;
            }
      } catch (Exception ex) {
            erroMsg("Erro obter foto",
                                 ex.getMessage());
            return false;
      }
}
```

Crie a seguinte função para converter a imagem para Buffered image.

```
public static BufferedImage imageToBufferedImage
(Image im) {
      BufferedImage bi = new BufferedImage
          (im.getWidth(null), im.getHeight(null),
                     BufferedImage.TYPE_INT_RGB);
      Graphics bg = bi.getGraphics();
```

```
        bg.drawImage(im, 0, 0, null);
        bg.dispose();
        return bi;
}
```

A seguinte função limpa todos os campos. Substitua os nomes dos campos pelos nomes que definiu.

```
/**
* Limpa todos os campos
*/
private void fullClean() {
        jTname.setText(null);
        jTsex.setText(null);
        jTheight.setText(null);
        jTnationality.setText(null);
        jTbirthDate.setText(null);
        jTnumBI.setText(null);
        jTnumBI2.setText(null);
        jTvalidityDate.setText(null);
        jTcountry.setText(null);
        jTnameFather.setText(null);
        jTnameMother.setText(null);
        jTnotes.setText(null);
        jTnumNIF.setText(null);
        jTnumSS.setText(null);
        jTnumSNS.setText(null);
        jTcardVersion.setText(null);
        jTdeliveryDate.setText(null);
        jTdeliveryEntity.setText(null);
        jTdocumentType.setText(null);
        jTlocale.setText(null);
        jTdistrictDesc.setText(null);
        jTmunicipalityDesc.setText(null);
        jTfreguesiaDesc.setText(null);
        jTstreettype.setText(null);
        jTstreet.setText(null);
        jTbuilding.setText(null);
        jTdoor.setText(null);
        jTfloor.setText(null);
        jTside.setText(null);
        jTplace.setText(null);
        jTlocality.setText(null);
        jTcp3.setText(null);
        jTcp4.setText(null);
        jTpostal.setText(null);
        jLccPhoto.setIcon(null);
        jLccPhoto.setText("Foto");
}
```

Para criar um evento para o botão selecione o botão e clique uma vez com o botão direito do rato e selecione events. A seguir selecione a subopção Action depois ActionPerformed. O Netbeans vai gerar automaticamente um evento para a função.

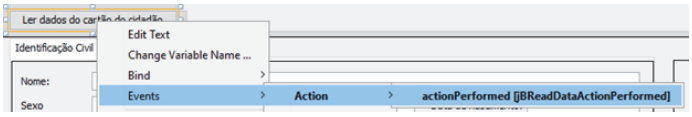


Ilustração 10 Evento leitura do cartão de cidadão

Depois de gerado o evento complete com o seguinte código que vai limpar todos os campos e chamar funções para obter e mostrar os dados do cartão de cidadão.

```
fullClean();
if (CitizenCardPortuguese.loadPteidLib() != false)
{
    if (readCCData() != false) {
        readCCAddress();
        readCCPhoto();
    }
}
```

No construtor do formulário vai ser definido o estilo de UI do Windows substitua o código do construtor pelo seguinte. (Atenção o estilo tem de ser definido antes de criar os componentes.)

```
/**
* Creates new form PaPCC
*/
public PaPCC() {
    try {           UIManager.setLookAndFeel
("com.sun.java.swing.plaf.windows.
                        WindowsLookAndFeel");
        initComponents();
    } catch (ClassNotFoundException |
InstantiationException | IllegalAccessException |
            UnsupportedLookAndFeelException ex)
{
      erroMsg("Erro ao criar aplicação gráfica.",
ex.getMessage());
    }
}
```

E terminamos a obtenção e visualização dos dados do cartão de cidadão. Mas antes de arrancar com o projeto é necessário definir a classe main. Nas propriedades do projeto, na opção Run das categorias selecione o JFRAME e clique em ok.

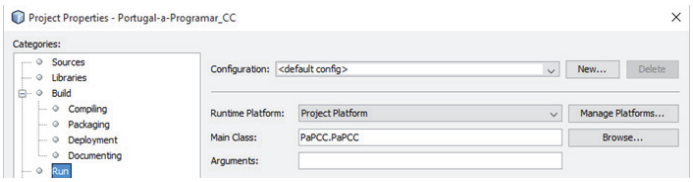


Ilustração 11 Main Class do projecto

Agora pode correr a aplicação e experimentar.

Já podemos assim incluir funções de validação dos números de identificação. Crie uma nova classe com o nome "PortugueseID" para a verificação de identificação feitas através de cálculos matemáticos, incluindo então a seguinte importação que vai ser necessária para as funções.

```
import static java.lang.Math.abs;
```

A seguinte função permite validar o número de identificação do bilhete de identidade.

```
/**
* Validation number of European PortugueseID ID
* @param idNumber number id
* @param checkDigit number id checkDigit
* @return true if is valid
*/
private boolean validatorCitizenID(String idNum-
ber, String checkDigit) {
    String BIformated;
    if ((idNumber.length() < 6) ||
(idNumber.length() > 8)) {
        return false;
    }
```

EXTRAIR DADOS DO CARTÃO DE CIDADÃO

```
    if (checkDigit.length() != 1) {
        return false;
    }
    if (idNumber.length() == 8) {
        BIformated = idNumber + checkDigit;
    } else {
        BIformated = '0' + idNumber + checkDigit;
    }
    int val = 0;
    for (int pos = 0; pos < 8; ++pos) {
        val += Integer.parseInt(BIformated.charAt
                        (pos) + "", 10) * (9 - pos);
    }
    int ctl = 0;
    if ((val % 11) != 0) {
        ctl = (11 - val % 11) % 10;
    }
    return ctl == Integer.parseInt
                    (BIformated.charAt(8) + "", 10);
}
```

A seguinte função permite validar o número identificação fiscal (NIF).

```
/**
* Validation number of European PortugueseID
*  contributors
*
* @param taxNumber
* @return true if is valid
* @throws Exception if the received variable con-
tains a non-numeric character
*/
public static boolean validatorTaxNumber(String
                        taxNumber) throws Exception {
    int checkDigit = 0;
    if (taxNumber.length() != 9) {
        return false;
    } else {
        char c1 = taxNumber.charAt(0);
        if (c1 == '1' || c1 == '2' || c1 == '5' ||
  c1 == '6' || c1 == '7' || c1 == '8' || c1 == '9')
        {
            checkDigit = Integer.parseInt("" + c1,
                                            10) * 9;
            for (int i = 2; i < 9; i++) {
                checkDigit += Integer.parseInt("" +
           taxNumber.charAt(i - 1), 10) * (10 - i);
            }
            checkDigit = 11 - (checkDigit % 11);
            if (checkDigit >= 10) {
                checkDigit = 0;
            }
            return checkDigit == Integer.parseInt
                        ("" + taxNumber.charAt(8));
        } else {
            return false;
        }
    }
}
```

As seguintes funções permitem validar um número identificação do cartão de cidadão.

```
/**
* Validation number of European PortugueseID
*  Citizen Card Source: https://
* www.cartaodecidadao.pt/images/stories/
* Algoritmo_Num_Documento_CC.pdf
*
* @param documentNumber
* @param documentCheckDigit
* @return true if is valid
* @throws Exception
*/
public static boolean validatorCitizenCardNumber
                    (String documentNumber, String
            documentCheckDigit) throws Exception {
               if (documentNumber.length() == 8 &&
               documentCheckDigit.length() == 1) {
        return new PortugueseID
            ().validatorCitizenID(documentNumber,
                                documentCheckDigit);
    } else if (documentNumber.length() == 8 &&
               documentCheckDigit.length() == 4) {
                 documentNumber = documentNumber +
                                documentCheckDigit;
        return new PortugueseID
            ().vCitizenCardNumber(documentNumber);
    } else {
        return false;
    }

}

/**
* Validation number of European PortugueseID
* Citizen Card
* @param documentNumber Citizen card number
* @return true if is valid
* @throws Exception if the received variable con-
tains a non-numeric character
*/
public boolean vCitizenCardNumber(String
                documentNumber) throws Exception {
    int sum = 0;
    boolean secondDigit = false;

    if (documentNumber.length() != 12) {
        throw new Exception("Invalid size for
                                document number.");
    }
    for (int i = documentNumber.length() - 1; i
                                        >= 0; --i) {
        int valor = getCitizenCardNumberFromChar
                    (documentNumber.charAt(i));
        if (secondDigit) {
            valor *= 2;
            if (valor > 9) {
                valor -= 9;
            }
        }
        sum += valor;
        secondDigit = !secondDigit;
    }
    return (sum % 10) == 0;
}

/**
* Validation number of European PortugueseID Ci-
tizen Card Check Digit
* @param Citizen card number check digit
* @return number
*/
private int getCitizenCardNumberFromChar(char
letter) {
    switch (letter) {
        case '0':
            return 0;
        case '1':
            return 1;
        case '2':
            return 2;
        case '3':
            return 3;
        case '4':
            return 4;
```

```
        case '5':
            return 5;
        case '6':
            return 6;
        case '7':
            return 7;
        case '8':
            return 8;
        case '9':
            return 9;
        case 'A':
            return 10;
        case 'B':
            return 11;
        case 'C':
            return 12;
        case 'D':
            return 13;
        case 'E':
            return 14;
        case 'F':
            return 15;
        case 'G':
            return 16;
        case 'H':
            return 17;
        case 'I':
            return 18;
        case 'J':
            return 19;
        case 'K':
            return 20;
        case 'L':
            return 21;
        case 'M':
            return 22;
        case 'N':
            return 23;
        case 'O':
            return 24;
        case 'P':
            return 25;
        case 'Q':
            return 26;
        case 'R':
            return 27;
        case 'S':
            return 28;
        case 'T':
            return 29;
        case 'U':
            return 30;
        case 'V':
            return 31;
        case 'W':
            return 32;
        case 'X':
            return 33;
        case 'Y':
            return 34;
        case 'Z':
            return 35;
        default:
            return 0;
    }
}
```

A seguinte função permite verificar o número de identificação da segurança social.

```
/**
* Validate identification number of European Portu-
gueseID Social Security number Source: http://
www.galhano.com/blog/wp-content/2007/06/NISS.pdf
*
* @param ssID Social Security number
* @return true if is valid
* @throws Exception if the received variable con-
tains a non-numeric character
*/
public static boolean validateSocialSecurityID
                    (String ssID) throws Exception
{
    int val = 0;
 int[] array = {29, 23, 19, 17, 13, 11, 7, 5, 3,
                                                2};
    if (ssID.length() != 11) {
        return false;
    } else {
        for (int i = 0; i < 10; i++) {
           val += Integer.parseInt(ssID.charAt(i)
                            + "", 10) * array[i];
        }
    }
    val = val % 10;
    return (9 - val) == Integer.parseInt
                      (ssID.charAt(10) + "", 10);

}
```

A seguinte função permite converter o antigo número da segurança social para o novo número.

```
/**
* Converts the old European PortugueseID Social
* Security number to the new number.
*
* @param ssID Social Security number
* @return true if is valid
* @throws Exception if the received variable
* contains a non-numeric character
*/
public static String newSocialSecurityID(String
                        ssID) throws Exception {
    int val = 0;
    int[] array = {29, 23, 19, 17, 13, 11, 7, 5,
                                            3, 2};
    ssID = '1' + ssID;
    for (int i = 0; i < 10; i++) {
        val += Integer.parseInt(ssID.charAt(i) +
                            "", 10) * array[i];
    }
    val = val % 10;
    val = abs(9 - val);
    return ssID + val;
}
```

A seguinte função permite verificar o número de identificação bancária (NIB). O número internacional de conta Bancária (IBAN) apenas acresce “PT50” no início do NIB.



Ilustração 12 número IBAN fonte Banco de Portugal

```
/**
* Portuguese validation procedures (Portuguese)
* http://www.bportugal.pt/pt-PT/pagamentos/
* InstrumentosdePagamento/IBAN/Documents/
* International%20Bank%20Account%20Number_pt.pdf
*
* @param nib
* @return true if is valid
```

EXTRAIR DADOS DO CARTÃO DE CIDADÃO

```
*/
public static Boolean verefyNIB(String nib) {
    try {
        if (nib.length() != 21) {
            return false;
        } else {
            char[] toValidate = nib.substring(0,
                                19).toCharArray();
            Integer checkDigit = Integer.valueOf
                                (nib.substring(19));
            Integer[] wi = {73, 17, 89, 38, 62, 45,
  53, 15, 50, 5, 49, 34, 81, 76, 27, 90, 9, 30, 3};
            Integer sum = 0;
            for (int i = 0; i < 19; i++) {
                sum += Character.digit(toValidate
                                [i], 10) * wi[i];
            }
            return checkDigit.equals(98 - (sum %
                                            97));
        }
    } catch (Exception ex) {
        throw new NumberFormatException("Invalid
                                            NIB");
    }
}
```

E terminamos a classe. Podemos experimentar com os campos identificação fiscal e segurança social.

No formulário insira as seguintes importações:

```
import java.awt.Color;
import static
            PaPCC.PortugueseID.validatorTaxNumber;
import static
       PaPCC.PortugueseID.validateSocialSecurityID;
```

No formulário defina os campos como editáveis e em cada um crie o evento Caret Update. Este evento vai chamar a função cada vez que um caracter é inserido ou apagado na caixa de texto.

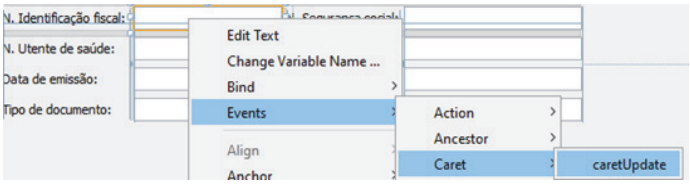


Ilustração 13 Evento caret update

Em cada evento que vamos chamar vai ser enviado o número de caracteres na caixa de texto para a função de validação.

Evento do NIF

```
validateCitizenTaxNumber(jTnumNIF.getText().
                                        length());
```

Evento número da segurança de social

```
validateCitizenSSecurity(jTnumSS.getText().length
                                                ());
```

A função do NIF recebe o número de caracteres. Se for 9 chamam a função para validar. Se o número de caracteres for diferente de 0 ou 9 o campo fica preenchido com a cor encarnada.

```
/**
* *
* Validação do NIF
*
* @param value número de caracteres do NIF
*/
private void validateCitizenTaxNumber(int value)
{
    try {
        switch (value) {
            case 0:
                jTnumNIF.setBackground
                                (Color.white);
                jTnumNIF.setForeground
                                (Color.black);
                break;
            case 9:
                if (validatorTaxNumber
                (jTnumNIF.getText()) != false) {
                    jTnumNIF.setBackground
                                (Color.green);
                    jTnumNIF.setForeground
                                (Color.black);
                } else {
                    jTnumNIF.setBackground
                                 (Color.red);
                    jTnumNIF.setForeground
                                (Color.white);
                }
                break;
            default:
                jTnumNIF.setBackground
                                 (Color.red);
                jTnumNIF.setForeground
                                (Color.white);
                break;
        }
    } catch (Exception ex) {
        erroMsg("Erro ao validar o número de
       identificação fisacal", ex.getMessage());
    }
}
```

A função da segurança social é idêntica ao NIF, apenas mudando a função de validação e número de caracteres que são 11. Se ao utilizar introduzisse o número antigo podia adicionar mais um caso para converter o número e depois validar.

```
/**
* Validação do número da segurança social
*
* @param value número de caracteres do número da
segurança social
*/
private void validateCitizenSSecurity(Integer
value) {
    try {
        switch (value) {
            case 0:
                jTnumSS.setBackground
                                (Color.white);
                jTnumSS.setForeground
                                (Color.black);
                break;
            case 11:
                if (validateSocialSecurityID
                (jTnumSNS.getText()) != false) {
                    jTnumSS.setBackground
                                (Color.green);
                    jTnumSS.setForeground
                                (Color.black);
                } else {
                    jTnumSS.setBackground
```

```
                                        (Color.red);
                    jTnumSS.setForeground
                                        (Color.white);
                }
                break;
            default:
                jTnumSS.setBackground(Color.red);
                jTnumSS.setForeground(Color.white);
                break;
        }
    } catch (Exception ex) {
        erroMsg("Erro ao validar o número de
                beneficiário", ex.getMessage());
    }
}
```

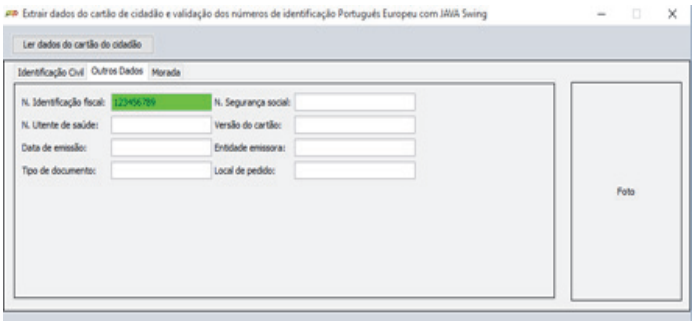


Ilustração 14 NIF correto

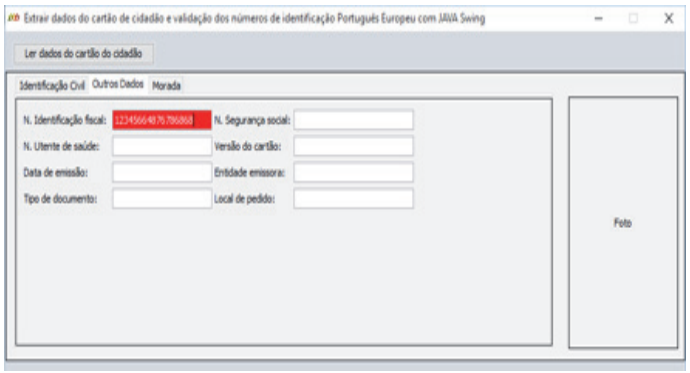


Ilustração 15 NIF incorrecto



E está finalizada a aplicação.

Se pretenderem fazer uma Java applet é recomendado que incluam instruções da instalação do módulo de segurança pkcs11 e o certificado raiz da entidade de certificação do cartão de cidadão para os vários navegadores de Internet.

> ***(...) O cartão de cidadão tem incluído um chip ISO/IEC 7816 é neste chip que está guardado toda a informação. (...)***

O projeto está disponível no GitHub em: https://github.com/rramoscabral/pap-cartao_de_cidadao .



CARTÃO DE CIDADÃO

AUTOR



Escrito por Ricardo Cabral

Licenciado em Engenharia Informática pela Universidade Autónoma de Lisboa e registado na Ordem dos Engenheiros Técnicos. Participante regular nas comunidades de Informática. O seu Twitter é @rramoscabral

Android Monkey Test: Um "Macaco" ao Serviço dos Programadores Android

O processo de criar e desenvolver uma aplicação não passa apenas pela programação propriamente dita. Tudo começa pela análise de requisitos, planeamento, programação e por fim o teste ao que foi implementado para garantir que tudo funciona como o previsto.

Todos os programadores fazem testes ("debug") às suas aplicações em busca de erros execução ou situações inesperadas que possam ocorrer no decorrer da execução da aplicação desenvolvida.É neste processo que a ferramenta "Android Monkey Test" é extremamente interessante para quem desenvolve para a plataforma móvel Android.

O Android Monkey Test é uma ferramenta incorporada no SDK Android que tem como objetivo fazer testes aleatórios e sequenciais a uma aplicação, coisas como a ação de touch, introdução de texto, etc… Basicamente e como o nome da ferramenta indica é como se a nossa aplicação estivesse a ser usada por um Macaco ou então por um utilizador que não tem a mínima noção do que está a fazer.

Algumas pessoas perguntar-se-ão o porque de esta ferramenta ser útil. É útil no sentido em que permite encontrar alguns tipos específicos de bugs, tais como bugs relacionados com a sequência do touch, ou seja, o clique num botão antes de algum componente ser inicializado provocando um crash. Estes tipos de erro são muito difíceis de ser testados por Humanos mas como existe uma grande diversidade de equipamentos podem ser notados em equipamentos mais lentos ou com conexões mais lentas.

Criar um Projecto Exemplo Android:

Para o desenvolvimento deste exemplo utilizei o Android Studio como Ambiente de Desenvolvimento.

O exemplo usado para demonstrar a utilização do Android Monkey Test é muito simples e tem o seguinte código fonte:

MainActivity.java

```
package
com.androidmonkeytest.nuno.androidmonkeytest;

import android.content.Intent;
import android.os.Bundle;
import android.support.v7.app.ActionBarActivity;
import android.view.View;
import android.widget.Button;

public class MainActivity extends ActionBarActivity
{
    Button b1;
    @Override

    protected void onCreate(Bundle
                                savedInstanceState)
{
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        //button ir para activity2
        b1=(Button)findViewById(R.id.button);
        b1.setOnClickListener(new
                            View.OnClickListener() {
            @Override
            public void onClick(View v) {
                //Iniciar nova Activity
                Intent in = new Intent
                (MainActivity.this, second.class);
                startActivity(in);
            }
        });
    }
}
```

SecondActivity.java

```
package com.androidmonkeytest.nuno.androidmonkey
                                                test;

import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.Toast;
public class second extends Activity{
    @Override

    protected void onCreate(Bundle
                              savedInstanceState) {
        super.onCreate(savedInstanceState);

        setContentView(R.layout.activity_second);
        Button b2=(Button)findViewById
                                  (R.id.button2);

        b2.setOnClickListener(new
                          View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Toast.makeText
    (second.this,"Obrigado por Ter Lido a Revista
        Programar =)",Toast.LENGTH_SHORT).show();
            }
        });
    }
}
```

main_activity.xml

```
<RelativeLayout xmlns:android="http://
            schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/
       tools" android:layout_width="match_parent"
    android:layout_height="match_parent"
                      android:paddingLeft="@dimen/
                       activity_horizontal_margin"
    android:paddingRight="@dimen/
                       activity_horizontal_margin"
    android:paddingTop="@dimen/
                         activity_vertical_margin"
    android:paddingBottom="@dimen/
```

```
                        activity_vertical_margin"
tools:context=".MainActivity">

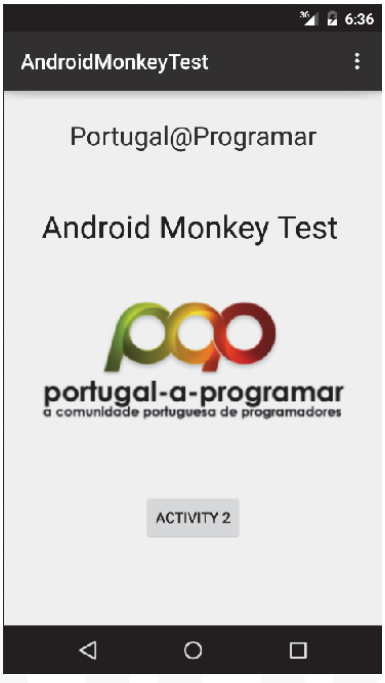
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Portugal@Programar"
        android:id="@+id/textView"
        android:textSize="25sp"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="10dp" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Android Monkey Test"
        android:id="@+id/textView2"
        android:layout_below="@+id/textView"
        android:textColor="#ff000000"
        android:textIsSelectable="false"
        android:textSize="30dp"
        android:layout_marginTop="50dp"
        android:layout_alignLeft="@+id/imageView"
        android:layout_alignStart="@+id/imageView"
        android:layout_alignRight="@+id/imageView"
        android:layout_alignEnd="@+id/imageView" />

    <ImageView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/imageView"
        android:src="@drawable/pap_2011"
        android:layout_below="@+id/textView2"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="50dp" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Ir Para Activity 2"
        android:onClick="button"
        android:id="@+id/button"
        android:layout_alignParentBottom="true"
        android:layout_centerHorizontal="true"
        android:layout_marginBottom="63dp" />

</RelativeLayout>
```



SecondActivity.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://
            schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
            android:layout_height="match_parent">

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Portugal@Programar "
        android:id="@+id/button2"
        android:layout_below="@+id/imageView"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="84dp" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Android Monkey Test"
        android:id="@+id/textView3"
        android:textColor="#ff000000"
        android:textSize="30dp"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="52dp" />


    <ImageView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/imageView"
        android:src="@drawable/pap_2011"
        android:layout_marginTop="58dp"
        android:layout_below="@+id/textView3"
        android:layout_centerHorizontal="true" />

</RelativeLayout>
```



Como Executar o Android Monkey Test:

Após a implementação do projeto teremos de executar o projeto onde a aplicação ficará funcional no AVD. Só posteriormente poderemos executar o Android Monkey Test uma vez que este este irá basear a sua implementação na app a correr no AVD.

1º Passo:

Abrir a consola e navegar até à pasta "android\sdk\platform-tools"

2º Passo:

Executar o seguinte comando:

```
adb shell monkey -p package.nome --throttle 50 -v
500
```

No "package.nome" colocaremos o package referente ao nosso projecto.



Legenda:

adb – Android Debug Bridge é uma ferramenta que permite conectar e enviar comandos para um dispositivo Android, neste caso o AVD (emulador).

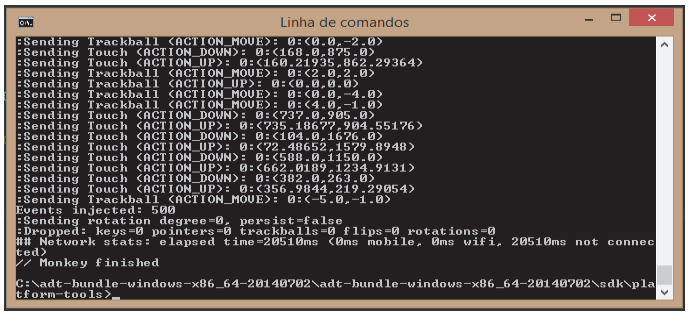
shell – Shell é uma interface que traduz os nossos comandos em comandos para o sistema.

Monkey – é a ferramenta de teste

Throttle – Tempo de espera entre cada evento lançado para a aplicação (milissegundos)

V – é referente ao método verbose que não é mais do que quantificar o "nível" permitindo obter mais detalhe sobre o teste

500 – É o número de eventos que serão enviados para o dispositivo, este valor pode ser alterado pelo utilizador.



Após executarmos o comando a ferramenta de teste irá então "bombardear" a aplicação que está a correr no emulador ou dispositivo com eventos aleatórios.

Desta forma poderemos detetar alguns erros ou situações que poderão fazer a aplicação falhar, permitindo resolver a situação e assim tornar a aplicação mais robusta.

Vantagens e Desvantagens:

A grande vantagem da ferramenta "Android Monkey Test" é a facilidade de uso, uma vez que não é necessário efectuar alterações no código-fonte da aplicação ou instalar ferramentas externas. É apenas necessário a execução de um comando na consola para que a ferramenta seja executada no AVD (emulador) ou num dispositivo físico conectado.

Mas nem tudo são vantagens para a ferramenta "Android Monkey Test" pois por vezes são gerados uma quantidade assinalável de crashs inválidos. Estes crashs acontecem devido á frequência dos eventos despoletados ser tão elevada que nunca conseguiria ser reproduzida por um humano.

Para evitar estes "falsos" crash's é aconselhado usar o parâmetro "- -throttle" que permite regular o intervalo de tempo entre eventos despoletados pela ferramenta.

Conclusão:

O "Android Monkey Test" não é a única ferramenta existente que permite efectuar testes automáticos a aplicações, no entanto a sua maior vantagem é a facilidade de uso, simplicidade e utilidade.

Esta simplicidade e facilidade de uso terão sempre importância na escolha da ferramenta de teste "Android Monkey Test" pois será sempre uma forma simples e rápida de efetuar testes automáticos a uma aplicação Android.

AUTOR

Escrito por Nuno Santos

Curioso e autodidacta com uma grande paixão pela programação e robótica, frequênta o curso de Engenharia Informática na UTAD alimentando o sonho de ainda vir a ser um bom Engenheiro Informático. Estudante, Blogger, e moderador no fórum Lusorobótica são algumas das suas actividades. Os seus projectos podem ser encontrados em: http://omundodaprogramacao.com

Programação (in)Segura - Transbordo de Memória

Introdução

O software assume cada vez mais uma importância primordial no nosso dia-a-dia. De facto, é crescente o número de dispositivos com o qual interagimos quotidianamente e cujo funcionamento está dependente de software. Exemplos incluem, obviamente, computadores e tablets, bem como dispositivos ditos inteligentes, como telemóveis, relógios e televisões. Outros exemplos abarcam sistemas de transportes como automóveis, aeronaves e barcos, e sistemas de domótica, para citar apenas alguns dos mais conhecidos. Dado a complexidade associada não só à criação e manutenção de programas informáticos como ainda dos sistemas que pretendem controlar, o software está sujeito a ocorrência de erros. Alguns desses erros podem ser aproveitados por indivíduos ou entidades com intenções maliciosas para subverter os dispositivos controlados, comprometendo deste modo, parcial ou totalmente, a segurança dos sistemas.

Este artigo analisa os erros do tipo transbordo de memória, em particular os que poderão ocorrer no segmento de pilha. O artigo foca alguns dos problemas de segurança que estão associados a situações de transbordo de memória afeta ao segmento de pilha. Os exemplos de código apresentados foram testados num sistema Linux - Lubuntu 14.04 / 32 bits, com kernel versão 3.13.04. Os exemplos foram compilados com a versão 4.8.2 do compilador de linguagem C GNU Collection Compiler (GCC).

Transbordo de zona de memória

O termo *transbordo de zona de memória* é mais conhecido pela designação Anglo-Saxónica de *buffer overflow* (Chris Anley, 2007). Trata-se de um dos erros de programação que mais frequentemente está associada a falhas de segurança como se pode constatar numa qualquer lista de falhas de segurança informática, e em particular na lista CVE (Common Vulnerabilities and Exposures -- https://cve.mitre.org/). A falha CVE-2015-7547, publicamente divulgada em fevereiro de 2016, afeta a importante GNU C Library (glibc), sendo mais um exemplo de vulnerabilidade devido a um transbordo de memória. Assim, nos sistemas afetados, e em certas circunstâncias, o uso da função **getaddrinfo** pode levar a um transbordo de memória que deixa o sistema vulnerável à execução remota de código. O facto da *glibc* estar presente na maioria dos sistemas Linux, da função **getaddrinfo** ser empregue por um número significativo de aplicações e da falha existir desde 2008 tornaram a vulnerabilidade particularmente preocupante (O'Donell, 2016).

Um transbordo de zona de memória ocorre quando uma operação escreve fora da zona de memória que lhe foi previamente atribuída, corrompendo a informação armazenada na zona indevidamente acedida. Por não possuir suporte nativo para verificação de fronteiras de zonas de memória (*bound checking*), a linguagem C pode originar programas que efetuem transbordos de zonas de memória. Tal ocorre, por exemplo, quando se utiliza a função **strcpy** para efetuar a cópia de uma *string* para um vetor de caracteres sem o espaço suficiente para armazenar todos os caracteres da string. Em particular, é relativamente comum o esquecimento de que uma *string* na linguagem C reque sempre o terminador **'\0'**, o que acresce em uma unidade o número de octetos necessários para o armazenamento da *string*. Este tipo de erro – transbordo em uma unidade de uma zona de memória – é tão comum que é identificado na terminologia anglo-saxónica como *off-by-one error (Chris Anley, 2007)*.

O que sucede quando ocorre um transbordo de uma zona de memória numa operação de cópia num programa escrito em linguagem C? Simplesmente a operação de cópia é efetuada, existindo escrita na memória para além da zona de memória alvo. Deste modo, é corrompida a informação armazenada na zona de memória indevidamente acedida, o que pode levar à alteração do conteúdo de variáveis que estejam guardadas nessa zona de memória. O programa mostrado na Listagem 1 exemplifica um transbordo de zona de memória. Concretamente, a cadeia de caracteres "**programar**" acessível através da variável **source_S** é copiada pela função **strcpy** para a zona de memória reservada para a variável **destination_S**. O problema é que esta zona de memória apenas tem espaço para armazenar 8 octetos, pois foi declarada como **char destination_S[8]**. Deste modo, a zona de memória apontada por **destination_S** apenas pode receber *string* com 7 caracteres mais o caracter de terminação '\0'. Dado que a cadeia de caracteres apontada por **source_S** ocupa 10 octetos (9 caracteres mais a terminação **'\0'**), a operação de cópia realizada pela função **strcpy** leva ao transbordo em dois octetos da zona de memória **destination_S**. Da análise à saída produzida pela execução do programa (Listagem 2) constata-se que após a chamada à função **strcpy**, a string **source_S** contém somente "**I**" em lugar do original "**ABCDEFGHI**". Tal deve-se ao transbordo da zona de memória apontada por **destination_S** aquando da operação de cópia. De facto, ao copiarem-se 10 octetos ("**ABCDEFGHI**" mais o marcador de fim de *string* '**\0**') para uma zona de memória com apenas oito octetos, os últimos dois octetos (caracter "**I**" e '**\0**') são copiados para a zona de memória adjacente a **destination_S,** que neste caso corresponde ao início da zona de memória reservada para **source_S**. Deste modo, os dois primeiros octetos da zona de memória **source_S** passam a ser, respetivamente, "**I**" e o marcador de fim de string '**\0**'. Quando a função **printf** acede à *string* **source_S**, apenas mostra "**I**", pois encontra logo na segunda posição o marcador '**\0**' que interpreta forçosamente como o fim da cadeia de caracteres. A Figura 1 ilustra o

transbordo de memória provocado pelo código da Listagem 1, com a) a corresponder à situação pré-transbordo e b) à situação pós-transbordo.

```
#include <stdio.h>
#include <string.h>
int main(void){
      char destination_S[8];
      char source_S[] = "ABCDEFGHI";
      printf("[INFO]A preparar cópia\n");
      printf("[INFO]'%s' (%zu octetos) para "
                  "zona com %zu bytes\n",
                  source_S,
                  strlen(source_S)+1,
                  sizeof(destination_S));
    strcpy(destination_S,source_S);
    printf("[INFO]Cópia efetuada\n");
    printf("[INFO]string origem='%s'\n", source_S);
    return 0;
}
```

Listagem 1: transbordo de zona de memória em operação de cópia de string (strcpy_overflow_ex1.c)

```
[INFO]A preparar cópia
[INFO]'ABCDEFGHI' (10 octetos)
                              para zona com 8 bytes
[INFO]Cópia efetuada
[INFO]string origem='I'
```

Listagem 2: saída da execução do programa strcpy_overflow_ex1.c

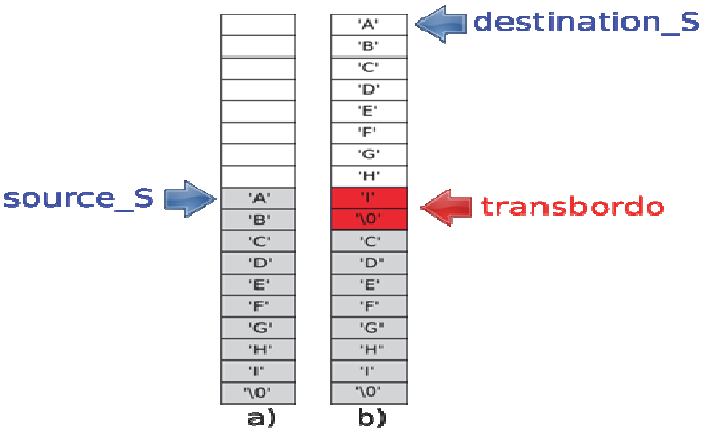


Figura 1: representação da situação de transbordo que ocorre com a execução do código da Listagem 1 (a) antes da execução de strcpy; b) depois da execução de strcpy)

Segmento de Pilha

No exemplo da Listagem 1, as variáveis **source_S** e **destination_S** são locais, pelo que os respetivos conteúdos são armazenados na zona de memória do processo que é designada como segmento de pilha. A designação de pilha deve-se ao facto de se tratar de uma estrutura de armazenamento de dados do tipo LIFO (*Last In First Out*) (Knuth, 1997). Como o nome indica, neste tipo de estrutura apenas é possível retirar elementos pela ordem inversa em que foram colocados. No contexto da memória afeta a um processo, uma dessas zonas é designada por segmento de pilha, seguindo a metodologia LIFO. Em termos práticos, o sistema implementa uma pilha mantendo dois endereços: o endereço inicial e o endereço corrente. O endereço inicial corresponde ao endereço zero da pilha, ao passo que o endereço corrente corresponde ao topo da pilha (endereço do último elemento inserido). O endereço corrente é alterado sempre que são inseridos ou retirados elementos da pilha.

Para além das variáveis locais, o segmento de pilha de um processo armazena **i)** os parâmetros passados nas chamadas a funções e ainda **ii)** o endereço de retorno para cada chamada de função efetuada durante a execução do processo. O endereço de retorno corresponde ao endereço de memória onde se encontra o código a executar quando terminar a chamada da função. A zona de memória da pilha associada a cada chamada de uma função designa-se por *stack frame*. Esta zona é criada dinamicamente sempre que ocorrer uma chamada à função, sendo aí armazenados os parâmetros passados na chamada à função, o endereço de retorno, e as variáveis locais da função. É criado um novo *stack frame* sempre que é efetuada a chamada de uma função. Por sua vez, quando termina a execução de uma função, o endereço de retorno é lido do *stack frame* associado à função, e empregue para posicionar o fluxo de execução do programa na instrução imediamente a seguir à da chamada da função terminada. Adicionalmente, o *stack frame* da função que terminou é marcado como indisponível. A Figura 2 ilustra o segmento de pilha considerando os *stack frames* das funções **main** e **F1** (figura da esquerda) e as alterações provocadas pela chamada à função **F2** efetuada pela função **F1**, nomeadamente, a criação de um terceiro **stack frame** e a correspondente deslocação do ponteiro de pilha (figura da direita). A Figura 2 assume que o segmento de pilha cresce dos endereços maiores para os endereços menores de memória, como normalmente sucede nos sistemas informáticos.

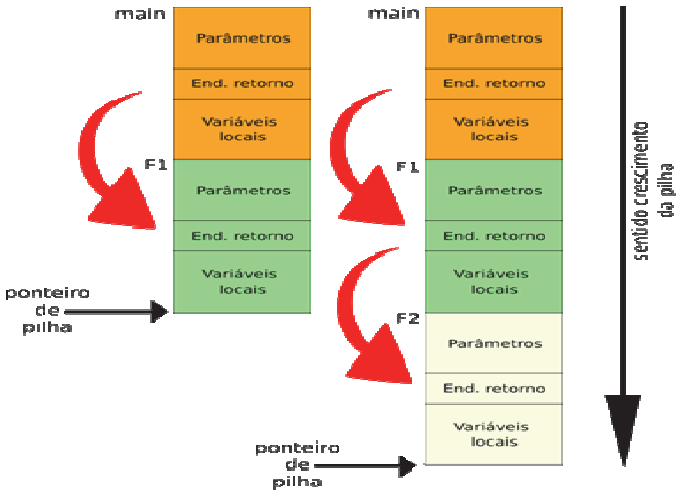


Figura 2: Segmento de pilha com os stacks frames das funções main e F1 (esquerda) e depois, com três stack frames (direita) após a chamada da função F2 por parte da função F1

O segmento de pilha é um dos vários segmentos que existem naquilo que é designado por **imagem de processo** em memória. Os restantes segmentos são: **i) segmento de texto** que contém o código do processo; **ii) segmento de dados** onde são armazenadas as variáveis globais e as variáveis declaradas com o modificador *static*; **iii) segmento de *heap*** que mantém as zonas de memória dinâmica alocada ao processo e, finalmente, **iv)** o **segmento de pilha**. A Figura 3 representa a imagem de um processo em memória. O **segmento de dados** não está explicitado na figura, pois é composto pelas secções **.BSS** e **.DATA**. A secção **.DATA** guarda as variáveis globais/*static* cujo valor inicial é diferente de zero, ao passo que a secção **.BSS** está reservada às variáveis cujo valor inicial é zero ou que foram definidas sem valor inicial. A distinção entre .BSS e DATA tem origem no ficheiro executável: os valores iniciais das variáveis inicializadas têm forçosamente que existir no ficheiro executável, sendo empregues aquando do lançamento do programa.

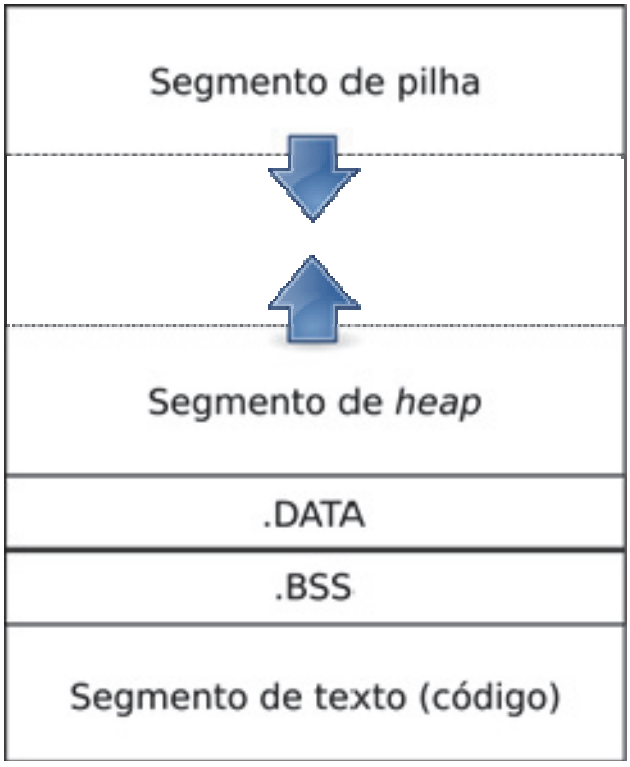


Figura 3: representação esquemática da imagem de um processo

Transbordo do segmento de pilha

O segmento de pilha tem uma capacidade máxima, correspondente ao espaço de memória existente entre o endereço inicial da pilha e o endereço final da ***heap***. Caso seja ultrapassada essa capacidade máxima, ocorrerá um transbordo da pilha, situação que é detetada pelo sistema operativo que consequentemente termina o processo faltoso. A ocorrência do transbordo da pilha pode dever-se **i)** a uma recursividade mal finalizada, em que o espaço ocupado por todos os ***stack frames*** ultrapassa a capacidade do segmento de pilha ou **ii)** à utilização de variáveis locais de grandes dimensões (por exemplo vetores). Interessantemente, a designação anglo-saxónica de transbordo da pilha – *stack overflow* – é sobejamente conhecida dos programadores pois trata-se do nome da bem conhecida plataforma de esclarecimento de dúvidas e partilha de código na área da programação: http://www.stackoverflow.com.

Endereços do Segmento de Pilha

A Listagem 3 recorre ao operador & ("endereço de") e à formatação %p da função printf para mostrar os endereços onde estão armazenadas as variáveis locais PI_value, E_value e f1, esta última da função F1. Da saída produzida pela execução do programa (Listagem 4), verifica-se que os endereços das variáveis locais PI_value (0xbfa2fef0) e E_value (0xbfa2fef8) estão separados por oito octetos, o que corresponde ao tamanho de uma variável do tipo double na norma de vírgula flutuante IEEE 754 (IEEE Standards Committee, 2008). Isto significa que as duas variáveis estão uma a seguir à outra no segmento de pilha. Por sua vez, a variável local f1 encontra-se um pouco mais afastada (0xbfa2fecc), o que se explica pelo facto de se encontrar noutro *stack frame*, concretamente no *stack frame* da função F1.

```
#include <stdio.h>
#include <string.h>
void F1(void){
      int f1 = 100;
      printf("função F1: f1=%d, endereço f1=%p\n",
                                    f1,&f1);
}
int main(void){
      double PI_value = 3.1415;
      double E_value = 2.7182;
      printf("PI=%f, endereço de PI=%p\n",
                  PI_value, &PI_value);
      printf("e=%f, endereço de E=%p\n",
                  E_value, &E_value);
      F1();
      return 0;
}
```

Listagem 3: endereços de variáveis locais (stack_addresses_ex2.c)

```
PI=3.141500, endereço de PI=0xbfa2fef0
e=2.718200, endereço de E=0xbfa2fef8
função F1: f1=100, endereço f1=0xbfa2fecc
```

Listagem 4: saída da execução do programa stack_addresses.c

O programa da Listagem 5 mostra algum do conteúdo do *segmento de pilha*, apresentando, em formato hexadecimal, o conteúdo e respetivo endereço de memória, dos endereços de memória adjacentes à variável local **f1**. Para o efeito, o código faz uso de um ciclo ***for***, cuja variável de controlo (variável **i**) regula o deslocamento relativo ao endereço onde está a variável **f1**. O deslocamento relativo vai de -1 a +8, correspondendo cada unidade ao deslocamento do tamanho de um inteiro (**int**) no sistema de desenvolvimento e

testes considerado, isto é, a quatro octetos. Note-se que no sistema considerado, como em muitos outros, o segmento de pilha cresce "de cima para baixo", isto é, do maior para o menor endereço de memória. Deste modo, o ***stack frame*** da função **main** encontra-se em endereços de memória maiores do que o ***stack frame*** da função **F1**. Assim, quando a variável de controlo **i** do ciclo ***for*** assume valores positivos, os endereços mostrados correspondem a conteúdos armazenados antes do conteúdo da variável local **f1**.

```
#include <stdio.h>
#include <string.h>
void F1(void){
      int f1 = 100;
      int i=0;
      printf("[F1]: f1=%d, endereço f1=%p\n",
                              f1,&f1);
      printf("[F1]: i=%d, endereço i=%p\n", i,&i);
      printf("---------------------------------
\n");
      for(i=-1;i<8;i++){
            printf("[F1]: f1+%d=%x, "
                  "endereço f1+%d=%p\n",
                  i,*(&f1+i),i,&f1+i);
      }
}
int main(void){
      double PI_value = 3.1415;
      double E_value = 2.7182;
      F1();
      printf("---------------------------------
\n");
      printf("[MAIN] PI=%f, endereço de PI=%p\n",
                  PI_value, &PI_value);
      printf("[MAIN] e=%f, endereço de E=%p\n",
                  E_value, &E_value);
      printf("endereço da função main: %p\n",
main);
      printf("endereço da função F1: %p\n", F1);
      return 0;
}
```

Listagem 5: Programa stack_addresses_ex3.c

```
[F1]: f1=100, endereço f1=0xbfc0bd78
[F1]: i=0, endereço i=0xbfc0bd7c
[F1]: f1+-1=ca0000, endereço f1+-1=0xbfc0bd74
[F1]: f1+0=64, endereço f1+0=0xbfc0bd78
[F1]: f1+1=1, endereço f1+1=0xbfc0bd7c
[F1]: f1+2=bfc0d43d, endereço f1+2=0xbfc0bd80
[F1]: f1+3=b7760000, endereço f1+3=0xbfc0bd84
[F1]: f1+4=bfc0bdb8, endereço f1+4=0xbfc0bd88
[F1]: f1+5=8048527, endereço f1+5=0xbfc0bd8c
[F1]: f1+6=1, endereço f1+6=0xbfc0bd90
[F1]: f1+7=bfc0be54, endereço f1+7=0xbfc0bd94

[MAIN] PI=3.141500, endereço de PI=0xbfc0bda0
[MAIN] e=2.718200, endereço de E=0xbfc0bda8
[MAIN] endereço da função main: 0x8048505
[MAIN] endereço da função F1: 0x804844d
```

Listagem 6: resultado da execução do programa stack_address_ex3.c

Os resultados da execução são apresentados na Listagem 6. Verifica-se que a variável **f1** está no endereço **0xbfc0bd78** e que a variável **i** está no endereço adjacente **0xbfc0bd7c**, ou seja quatro octetos acima, sendo esse quatro octetos precisamente o espaço ocupado pela variável **i**. Dos resultados produzidos pelo ciclo **for** observa-se que a iteração com **i=1** corresponde ao endereço da variável de controlo **i**. Por sua vez, observa-se que o valor inteiro guardado no endereço de memória **0xbfc0bd8c**, obtido com **i** igual a 5, é **0x8048527**. Este valor está na gama de valores dos endereços de memória das funções **F1** (**0x804844d**) e **main** (**0x8048505**), indicando que se trata pois de um endereço do segmento de código. Dado a sua localização no **stack frame**, é expectável que se trate do endereço de retorno a ser empregue aquando do término da função F1. Para validar esta hipótese, acrescentou-se a seguinte linha no final da função F1:

(&f1+5) = (int) F1;

A linha modifica o conteúdo da zona de memória correspondente a **f1 + 5 x 4** octetos (ou seja endereço **f1 + 20** bytes), atribuindo-lhe o endereço da função **F1**. Assim, se a hipótese estiver certa – **&f1 + 5** corresponde ao endereço de retorno –, a atribuição irá alterar o endereço de retorno para a própria função **F1**, criando efetivamente um ciclo infinito, pois quando termina a execução da função **F1**, o código retorna, por via do endereço de retorno alterado, para o início da função **F1** e assim sucessivamente. Importa referir que o mesmo código compilado e executado noutros ambientes (por exemplo, num sistema Linux de 64 bits com outro compilador), poderá apresentar uma organização diferente, e consequentemente o endereço de retorno estar localizado numa posição diferente (por exemplo, **&f1+4** ou **&f1+6**). O código da nova versão é apresentado na Listagem 7, e a saída resultante da respetiva execução é mostrada na Listagem 8. Importa notar que a substituição do endereço de retorno pelo endereço da função F1 não respeita o protocolo de chamada de funções empregue pelo GCC, resultando no desequilíbrio da **stack**. De facto, observa-se nas sucessivas execuções da função F1 que a variável local **f1** fica localizada em endereços da stack crescentes (diferença de 4 bytes entre chamadas – 0xbffd81b8, 0xbffd81bc, ...). O desequilíbrio leva a que a aplicação seja terminada pelo sistema operativo com a notificação **segmentation fault** ao fim de umas centenas de execuções da função F1.

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

void F1(void){
 int f1 = 100;
 int i=0;
 printf("[F1]: f1=%d, endereço f1=%
                                  p\n",f1,&f1);
      printf("[F1]: i=%d, endereço i=%p\n",i,&i);
      printf("---------------------------------
                                        \n");
      for(i=-1;i<8;i++){
            printf("[F1]: f1+%d=%x, "
                  "endereço f1+%d=%p\n",
                  i,*(&f1+i),i,&f1+i);
      }
      *(&f1+5) = ((int)F1);
      printf("[F1]:FIM\n");
}
```

```
int main(void){
      if( sizeof(int) != 4 ){
            fprintf(stderr,"[OUT] "
                "Codigo requer sizeof(int)=4\n");
            exit(1);
      }
      if( sizeof(int*) != 4 ){
            fprintf(stderr,"[OUT] "
           "Código requer endereços de 32 bits\n");
            exit(2);
      }
      double PI_value = 3.1415;
      double E_value = 2.7182;
      F1();
      Printf
         ("================================\n");
      printf("[MAIN] PI=%f, endereço de PI=%p\n",
                        PI_value, &PI_value);
      printf("[MAIN] e=%f, endereço de E=%p\n",
                        E_value, &E_value);
      printf("endereço da função main: %p\n",
                                              main);
      printf("endereço da função F1: %p\n", F1);
      return 0;
}
```

Listagem 7: Programa stack_address_argv_ex4.c

```
[F1]: f1=100, endereço f1=0xbffd81b8
[F1]: i=0, endereço i=0xbffd81bc
--------------------------------------
[F1]: f1+-1=ca0000, endereço f1+-1=0xbffd81b4
[F1]: f1+0=64, endereço f1+0=0xbffd81b8
[F1]: f1+1=1, endereço f1+1=0xbffd81bc
[F1]: f1+2=bffd8458, endereço f1+2=0xbffd81c0
[F1]: f1+3=b772e000, endereço f1+3=0xbffd81c4
[F1]: f1+4=bffd81f8, endereço f1+4=0xbffd81c8
[F1]: f1+5=8048540, endereço f1+5=0xbffd81cc
[F1]: f1+6=1, endereço f1+6=0xbffd81d0
[F1]: f1+7=bffd8294, endereço f1+7=0xbffd81d4
[F1]:FIM
[F1]: f1=100, endereço f1=0xbffd81bc
[F1]: i=0, endereço i=0xbffd81c0
--------------------------------------
[F1]: f1+-1=64, endereço f1+-1=0xbffd81b8
[F1]: f1+0=64, endereço f1+0=0xbffd81bc
[F1]: f1+1=1, endereço f1+1=0xbffd81c0
[F1]: f1+2=b772e000, endereço f1+2=0xbffd81c4
[F1]: f1+3=b772e000, endereço f1+3=0xbffd81c8
[F1]: f1+4=bffd81f8, endereço f1+4=0xbffd81cc
[F1]: f1+5=1, endereço f1+5=0xbffd81d0
[F1]: f1+6=bffd8294, endereço f1+6=0xbffd81d4
[F1]: f1+7=bffd829c, endereço f1+7=0xbffd81d8
[F1]:FIM
(...)
Segmentation fault (core dumped)
```

Listagem 8: resultado da execução do programa stack_address_argv_ex4.c

Note-se que embora seja relativamente fácil encontrar a localização do endereço de retorno da função corrente no segmento de pilha, é muito mais complexo alterá-lo por forma a que se possa tomar o controlo do processo. Ao leitor interessado recomendam-se a leitura de (One, 2007) e (Chris Anley, 2007).

Proteção ao Segmento de Pilha

Porventura o maior perigo associado a um transbordo de zona de memória no segmento de pilha é a possibilidade de corrupção maliciosa do endereço de retorno da função corrente. Deste modo, após terminar a função corrente, em vez da execução do programa retornar para o código da função chamante, a execução será desviada para outra zona do código, o que poderá permitir a um utilizador malicioso controlar a restante execução do processo.

Por forma a detetar a corrupção de variáveis do segmento de pilha, alguns compiladores recorrem a valores especiais (*marcadores*) que são escritos no segmento de pilha, por exemplo, logo a seguir ao espaço de memória afeto a uma variável local do tipo *string*. Caso ocorra a alteração do valor dos marcadores, isso significa que a realização de uma operação de escrita numa zona do segmento de pilha que não corresponde a nenhuma variável do processo, indiciando uma anomalia na execução. Os marcadores são designados por *canários*, pois servem de alertas, um pouco à semelhança dos canários outrora empregues nas minas para a deteção da perigosa degradação da qualidade do ar (C. Cowan, 1998).

O compilador GCC implementa um mecanismo de canário para variáveis locais. O seu funcionamento será verificado durante a execução do código da Listagem 9 que efetua a cópia da *string* recebida através do primeiro (e único) argumento da linha de comando aquando da execução do programa. Esta *string* é copiada para a zona de memória reservada para a variável **char string[4]**, que apenas tem capacidade para armazenar três caracteres e o marcador '\0'. Deste modo, sempre que o programa for lançado com um argumento que tenha mais do que três caracteres (e.g., "12345") ocorrerá o transbordo da variável **string**. Por sua vez, a variável **int counter** ao ser declarada imediatamente antes da variável **string** é colocada no segmento de pilha em na zona de memória seguinte à reservada para a variável **string.** Deste modo, um transbordo da variável **string** deve originar a corrupção da memória reservada para a variável **counter** e consequentemente alteração do valor guardado na variável. Com vista à deteção dessa alteração, o programa mostra o valor inicial da variável **counter**, efetua a operação de cópia do argumento da linha de comando (**argv[1]**) para a variável **string** e, por fim, volta a mostrar o valor da variável counter.

A Listagem 10 apresenta o resultado da execução do programa, tendo a aplicação sido compilada com a seguinte linha de comando:

```
gcc -w -Wall -g ex3.c -o ex3.exe
```

Na execução do programa com o argumento "12345" (Listagem 10) observa-se que é detetado um ***stack smash***, sendo produzido um ficheiro do tipo *core* que pode ser empregue com um depurador (e.g., GDB) para análise mais detalhada à situação que levou ao ***stack smash***. Para observar os efeitos do transbordo de memória da variável **string**, é necessário desativar a proteção de pilha, acrescen-

tando-se para o efeito a opção -fno-stack-protector na compilação com o GCC.

Os resultados da execução (Listagem 11) mostram que a corrupção da variável **counter** ocorre somente quando o transbordo da variável **string** for de 10 ou mais octetos. De facto, com um argumento de 13 octetos (12 caracteres mais o terminador '\0') o valor da variável **counter** mantém-se inalterado. Contudo, quando é passado um parâmetro com 14 octetos (transbordo de 10 octetos), o valor da variável **counter** é alterado de -1 para -256, o que significa que o octeto menos significativo passou para zero. Concretamente, tal se deve à escrita do marcador '\0' na zona de memória onde é guardado o octeto menos significativo da variável **counter**.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
int main(int argc, char *argv[]){
      int counter = -1;
      char string[4];
      char *argv_ptr;
      if( argc != 2 ){
            printf("[usage]%s <string>\n",
                              argv[0]);
            exit(EXIT_FAILURE);
      }
      argv_ptr = argv[1];
      size_t argv_len = strlen(argv_ptr);
      printf("[INFO]:'%s' "
            "(%zu caracteres, %zu bytes)\n",
            argv_ptr, argv_len, argv_len+1);
      printf("[ANTES] Counter=%d\n", counter);
      strcpy(string, argv_ptr);
      printf("[DEPOIS] Counter=%d\n", counter);
      return 0;
}
```

Listagem 9: programa vulnerável à corrupção do segmento de pilha (strcpy_argv_ex5.c)

```
./ex5.exe 12345
[INFO]:'12345' (5 caracteres, 6 bytes)
[ANTES] Counter=-1
[DEPOIS] Counter=-1
*** stack smashing detected ***: ./ex5.exe termi-
nated
Aborted (core dumped)
```

Listagem 10: resultado da execução de strcpy_argv_ex5.c

```
./ex5_no.exe 12345789012
[INFO]:'12345789012' (11 caracteres, 12 bytes)
[ANTES] Counter=-1
[DEPOIS] Counter=-1

./ex5_no.exe 123457890123
[INFO]:'123457890123' (12 caracteres, 13 bytes)
[ANTES] Counter=-1
[DEPOIS] Counter=-256

./ex5_no.exe 1234578901234
[INFO]:'1234578901234' (13 caracteres, 14 bytes)
[ANTES] Counter=-1
[DEPOIS] Counter=-65484
```

Listagem 11: resultado da execução de strcpy_argv_ex5.c compilado sem proteção de segmento de pilha (-fno-stack-protector)

Cuidados a Observar

Os cuidados a observar para minimizar a possibilidade de vulnerabilidade de uma aplicação a situações de transbordo de memória passam por um cuidado tratamento das entradas fornecidas direta ou indiretamente pelo utilizador. Este tratamento é frequentemente designado por *sanatização* dos dados de entrada. O objetivo é filtrar conteúdo que possa prejudicar a aplicação. Por exemplo, o código apresentado na Listagem 9 está (intencionalmente) vulnerável a um transbordo de memória do segmento de pilha, dado que copia para uma variável local com capacidade para armazenar três caracteres (mais o marcador '\0'), os dados fornecidos pelo utilizador, sem previamente verificar o tamanho dos dados a copiar. Uma abordagem correta passa pela validação do tamanho dos dados introduzidos pelo utilizador, assinalando como erro dados com tamanho superior a três caracteres. Outra abordagem, mais flexível passa pelo uso de memória dinâmica. Concretamente, determina-se, por exemplo, através da função **strlen** o número de caracteres da string passada pelo utilizador através do parâmetro da linha de comando e aloca-se um bloco de memória dinâmica com a capacidade necessária para armazenar a *string*, isto é, com um número de octetos igual ao número de caracteres da *string* acrescido em uma unidade para o marcador '\0'.

Outro cuidado passa pelo uso de funções que permitam estabelecer limites no número de elementos a copiar e/ou a acrescentar. Por exemplo, a função **char *strcpy(char *dest, const char *src)** efetua a cópia de uma cadeia de caracteres que se inicia no endereço apontado por **src** para o endereço de memória apontado por **dest**, não sendo possível limitar a cópia a um número máximo de caracteres. Como alternativa poderá utilizar-se a função **char *strncpy (char *dest, const char *dest, size_t n)**, que permite indicar, através do parâmetro **n**, o número máximo de octetos que podem ser copiados para o endereço de destino. Deste modo, se o parâmetro **n** for especificado com o tamanho máximo da zona de memória de destino, o uso da função **strncpy** garante que não existe transbordo da zona de memória. Contudo, no caso em que a string a copiar ocupa um número de octetos superior a **n**, a função **strncpy** não termina a string de destino com o marcardor '\0', pelo que a *string* de destino fica efetivamente aberta. Para evitar esta última situação, o programador deve garantir que a *string* de destino é devidamente fechada, escrevendo o marcador '\0' na posição de índice **n-1** da zona de memória de destino. A Listagem 12 exemplifica o uso da função **strncpy** de modo a garantir que a *string* de destino é sempre terminada com o marcador '\0'.

```
#include <stdio.h>
#include <string.h>
#define TAM_DEST (4)   /* tamanho buffer destino
*/
int main(void){
      char src[] = "0123456789";
      char dest[TAM_DEST];
      strncpy(dest,src,TAM_DEST);
```

```
    /* Garante que dest é NULL-terminated */
    dest[TAM_DEST-1] = '\0';
    printf("dest='%s'\n", dest);
    return 0;
}
```

Listagem 12: uso da função strncpy com terminação da string de destino (strncpy_ex6.c)

Para além da função **strcpy**, outras funções da biblioteca da linguagem C expõem os programas a situações de transbordo de memória, pelo que o seu uso deve ser evitado. São exemplos, entre muitas outras, as funções **strcat** (concatenação de strings) e **sprintf** (escrita com formatação para uma zona de memória). Sempre que existam, deverão ser utilizadas as versões das funções que permitam limitar o número de caracteres escritos no destino. Para o caso da função **strcat**, a versão com limitador designa-se por **strncat**, apresentando o seguinte protótipo:

char *strncat(char *dest, const char *src, size_t n);

Apesar da função **strncat** assegurar a escrita do terminador '\0' na zona de memória da string de destino, o programador terá de garantir que a zona de memória de destino possui o espaço suficiente para acrescentar ao seu conteúdo **n** caracteres da string **src** e ainda o terminador '\0'. Caso a capacidade da zona de memória de destino não seja suficiente, poderá ocorrer um transbordo de memória.

> ***"Este tipo de erro – transbordo em uma unidade de uma zona de memória – é tão comum que é identificado na terminologia anglo-saxónica como off-by-one error (Chris Anley, 2007)."***

A função **sprintf** poderá ser substituída pela variante **int snprintf(char *dest, size_t n, const char *format,...)**.. Esta função garante que não são escritos mais do que **n** octetos a zona de memória de destino, acrescentando sempre o terminador '\0' dentro do limite dos **n** octetos. Contudo a função **snprintf** não faz parte das funções definidas pela norma da linguagem C, pelo que a sua existência não é garantida, o que pode trazer problemas com a portabilidade do código.

A norma C11

A revisão à linguagem C ocorrida em 2011, deu origem à norma C11 (ISO, 2011). A norma C11 teve em conta o problema do transbordo de memória que pode ser causado por várias funções da linguagem. Para o efeito foram criadas versões ditas seguras das funções mais frequentemente associadas a situações de transbordo de memória. Contudo, essas versões seguras estão marcadas como opcionais, pertencendo ao anexo K da norma C11, pelo que não existem em todos os ambientes de desenvolvimento C11. Uma omissão relevante é a da glibc (GNU C library), empregue pelo sistema operativo Linux, pelo que essas versões seguras não se encontram disponibilizadas nessa plataforma. A norma C11 indica que o suporte para as versões seguras pode ser verificado através da existência da constante do pré-processador __STDC_LIB_EXT1__. Caso esta constante esteja definida, o ambiente de desenvolvimento implementa as versões seguras definidas no anexo K da norma C11 (Peter Prinz, 2015).

As versões seguras definidas na norma C11 são facilmente reconhecíveis, pois o nome é composto pelo nome original da função acrescido do sufixo **_s** (Peter Prinz, 2015). Assim **strcpy_s** corresponde à versão segura da função **strcpy** e **strcat_s** corresponde à versão segura de **strcat**. Importa notar que as versões seguras diferem substancialmente das versões originais. Por exemplo, a função **strcpy_s**, cujo protótipo se encontra na Listagem 13, devolve um valor do tipo **errno_t**, recebendo um ponteiro para a zona de escrita (parâmetro **dest**), um limitador de tamanho (parâmetro **destmax**) do tipo **rsize_t** e o ponteiro para a string que se pretende copiar (parâmetro **src**). Antes de efetuar a operação de cópia, a função **strcpy_s** efetua as seguintes verificações:

I. Confirma se os ponteiros **dest** e **src** não são **NULL**;

II. Confirma se o valor de **destmax** se encontra compreendido entre zero e **RSIZE_MAX**;

III. Confirma se o tamanho da string a copiar (**src**) é inferior a **RSIZE_MAX**;

IV. Confirma se os endereços de memória indicados por **src** e **dest** não tem nenhuma sobreposição. Este requisito é expresso através da palavra chave '**restrict**' que, em certas circunstâncias, permite ao compilador aplicar certas otimizações.

Caso não se verifique uma das condições, a função **strcpy_s** devolve um valor diferente de zero, escrevendo o terminador '\0' no endereço apontado por **dest**, isto caso **dest** não seja **NULL** e **destmax** se encontre compreendido

entre zero e **RSIZE_MAX**. Caso a execução da função decorra normalmente é devolvido o valor zero. Relativamente à constante **RSIZE_MAX**, o anexo K da norma C11 indica que deve ter um valor correspondente a metade do valor máximo que pode ser representado pelo tipo de dado **size_t**.

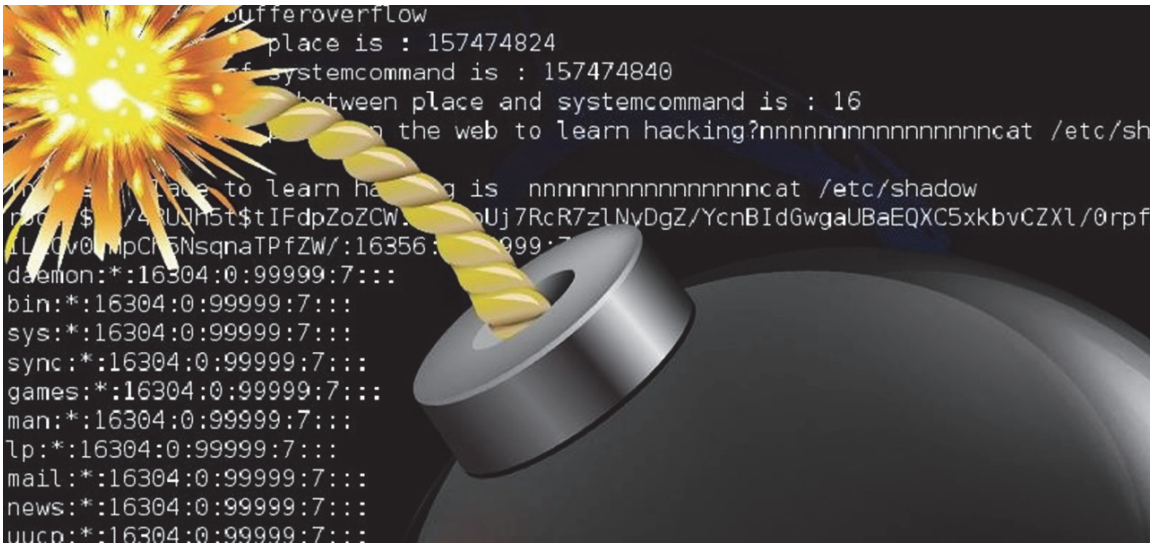
Listagem 13: protótipos das funções strcpy_s e strcat_s (Norma C11)

Nota final

Este artigo abordou a vasta temática do transbordo de memória em programas escritos na linguagem C. Tendo em consideração as nefastas consequências que poderão existir ao nível da segurança das aplicações, é fundamental que o programador esteja sensível e atento à problemática do transbordo de memória.

Bibliografia

- C. Cowan, C. P. (1998). Automatic Detection and Prevention of Buffer-Overflow Attacks. *7th USENIX Security Symposium.* San Antonio, Texas, USA.: USENIX.
- Chris Anley, J. H. (2007). *The Shellcoder's Handbook: Discovering and Exploiting Security Holes* (2nd ed.). Wiley Publishing, Inc.
- IEEE Standards Committee. (2008). 754-2008 IEEE standard for floating-point arithmetic.
- ISO. (2011). *ISO/IEC 9899:2011 - Information technology - Programming languages - C.*
- Knuth, D. (1997). *The Art of Computer Programming, Volume 1: Fundamental Algorithms* (3rd ed.).
- O'Donell, C. (02 de 2016). *CVE-2015-7547 --- glibc getaddrinfo() stack-based buffer overflow.* Obtido de https://sourceware.org/ml/libc-alpha/2016-02/msg00416.html
- One, A. (2007). Smashing the stack for fun and profit. *Phrack, 7*(49), 32.
- Peter Prinz, T. C. (2015). *C in a Nutshell: The Definitive Reference* (2nd ed.). USA: O'Reilly.



AUTOR



Escrito Por Patrício Domingues

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

AUTOR

Escrito Por Vítor Távora

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

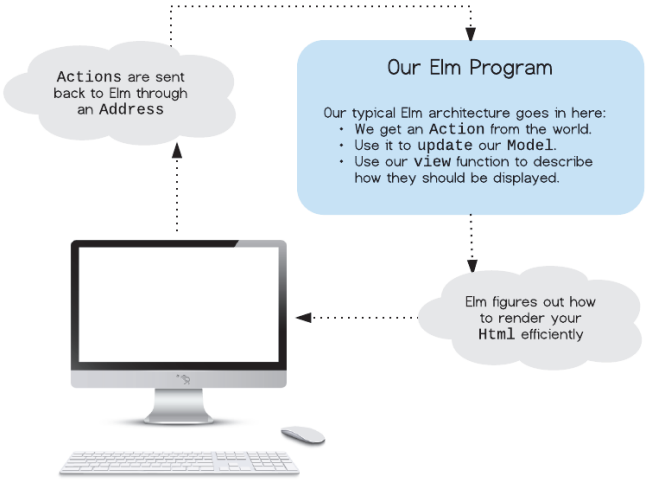
ELM

Resumo

Como vamos ver Elm é bastante pequena e simples, o que torna fácil criar interfaces gráficas para a Internet. Elm quando compilada tem como alvos JavaScript, HTML e CSS. Elm é uma linguagem ainda muito jovem, foi criada em 2012 e está na versão 0.16 (https://en.wikipedia.org/wiki/Elm_%28programming_language%29).

Introdução

Elm utiliza o paradigma de programação FRP (Functional Reactive Programming), ou seja é uma linguagem funcional que está alerta ao tempo. Por outro lado desenvolvimento em Elm diverge da maioria das alternativas porque não utiliza a arquitetura MVC (Model, View & Controller). O fluxo de informação num programa Elm, que corre dentro do navegador de Internet, segue apenas uma direção tal como se pode ver na imagem abaixo.



Fluxo de informação num programa Elm (retirado da documentação oficial).

Instalação

Para instalar a linguagem Elm é necessário o Node.Js para proceder à sua compilação. A infraestrutura Elm também disponibiliza um gestor de pacotes, ferramenta de testes unitários, REPL (Read, Evaluate, Print, Loop), debugger e instaladores para Windows e Mac que podem ser descarregados em http://elm-lang.org. A instalação em Ubuntu é feita com os seguintes comandos, começando por instalar o Node.Js seguida da instalação de Elm:

```
$ curl -sL https://deb.nodesource.com/setup_5.x |
sudo -E bash -
$ sudo apt-get install -y nodejs
$ npm install -g elm
```

Agora estamos prontos para criar o nosso primeiro programa em Elm. A forma mais simples é partir de uma infraestrutura e preencher apenas algumas funções. Esta infraestrutura será responsável pelo fluxo de informação e pela receção de sinais a que o nosso programa responderá (o tempo, o clicar e mover do rato e o teclado são exemplos de sinais). Os sinais registados no programa levarão a uma mudança no estado do programa e consequentemente a uma atualização da interface gráfica. É mais fácil do que parece, vamos a isto.

Programa que conta cliques do rato

A imagem abaixo mostra-nos o programa Click numa janela do navegador.



Este programa reage a dois eventos:

- se se clicar no botão esquerdo do rato em cima do número a azul, o mesmo aumenta em uma unidade; e
- se se carregar no botão a contagem fica a zero.

O valor inicial é 100.

O código

Para começar o programa criar a diretoria "Click" e de seguida ir para a mesma. Escrever o comando:

$ elm-package install evancz/start-app

E responder "y" às perguntas para autorizar a instalação. Start-app é a infraestrutura de que falámos atrás. O conteúdo da diretoria é:

$ elm-package.json

$ elm-stuff/

A diretoria elm-stuff não contém nada com que seja necessário preocuparmo-nos agora. O ficheiro elm-

package.json contém informação que deve ser editada manualmente, por exemplo a licença, a versão, e uma explicação sobre a aplicação. Para a nossa aplicação temos que acrescentar a informação sobre a biblioteca elm-http, ficando o ficheiro elm-package.json com a seguinte informação:

```
{
  "version": "1.0.0",
  "summary": "Programa para contar cliques.",
  "repository": "https://github.com/user/
project.git",
  "license": "BSD3",
  "source-directories": ["."],
  "exposed-modules": [],
  "dependencies": {
    "elm-lang/core": "3.0.0 <= v < 4.0.0",
    "evancz/elm-html": "4.0.1 <= v < 5.0.0",
    "evancz/start-app": "2.0.2 <= v < 3.0.0"
  },
  "elm-version": "0.16.0 <= v < 0.17.0"
}
```

Aqui são declaradas as bibliotecas externas utilizadas pelo programa. Um programa em Elm começa pelo ficheiro Main.elm e, no nosso caso criamos o ficheiro Click.elm.

O ficheiro Click.elm

No ficheiro Click.elm temos as funcionalidades do programa conforme, começando pela declaração do tipo de variável Model que guarda o estado do programa. Uma nova variável deste tipo é criada quando o programa reage a um sinal externo. Também é declarado o tipo de variável Action que corresponde às possíveis mudanças de estado do programa.

module Click where

```
import Html exposing (..)
import Html.Attributes exposing (style)
import Html.Events exposing (onClick)

type alias Model = Int
type Action = Increment | Reset
```

No programa Click o nosso modelo corresponde a uma contagem de cliques no botão esquerdo do rato, que é guardado num tipo inteiro. O tipo de ações que iremos utilizar no nosso programa serão incrementar a contagem e colocá-la a zero.

Segue-se a função update que cria variáveis do tipo Model sempre que é invocada.

```
update : Action -> Model -> Model
update action model = case action of
  Increment ->
    model + 1
  Reset ->
    0
```

A declaração da função update mostra-nos que entram duas variáveis, a primeira do tipo Action e a segunda do tipo Model. A função update devolve uma variável do tipo Model. Se a ação invocada for Increment, à variável com o valor do modelo atual é somado 1. Se a ação for Reset, é criada uma variável do tipo Model com o valor 0.

A função view que atualiza a interface gráfica.

```
view : Signal.Address Action -> Model -> Html
view address model = div []
  [ div [ textStyle ]
          [ text "Clique no numero para
                                  incrementar: " ]
  , div [ countStyle
        , onClick address Increment ]
          [ text (toString model) ]
  , button [ onClick address Reset ]
          [ text "Colocar a 0" ]
  ]
```

A função view tem como argumentos de entrada uma variável do tipo Action e outra variável do tipo Model. O resultado da função é um DOM (Html).

Os atributos dos elementos criados são dados em textStyle e countStyle, conforme podemos ver abaixo.

```
countStyle: Attribute
countStyle = style
   [("font-size", "20px")
  , ("color", "blue")
  , ("font-family", "monospace")
  , ("display", "inline-block")
  , ("width", "50px")
  , ("text-align", "center")
  ]
textStyle: Attribute
textStyle = style
  [("font-size", "20px")
  , ("font-family", "fantasy")
  , ("display", "inline-block")
  , ("text-align", "center")
  ]
```

O ficheiro Main.elm

O ficheiro Main.elm declara o estado inicial do programa, note-se que iniciamos o modelo a 100.

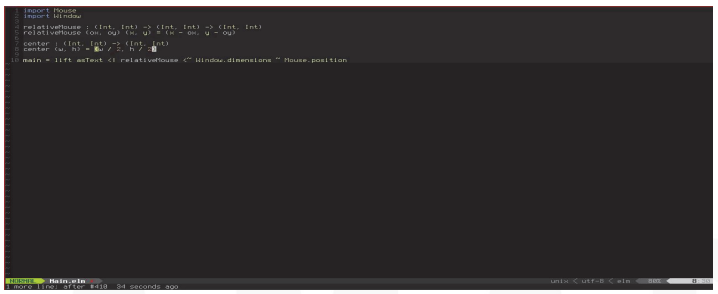
```
import Click exposing (update, view)
import StartApp.Simple exposing (start)
main = start
  { model = 100
  , update = update
  , view = view
  }
```

Compilação

Um programa Elm é compilado para JavaScript. O comando para o fazer é:

$ elm-make Main.elm

Para correr o programa abrir o ficheiro index.html na diretoria em que escreveu o programa a partir do seu navegador favorito.



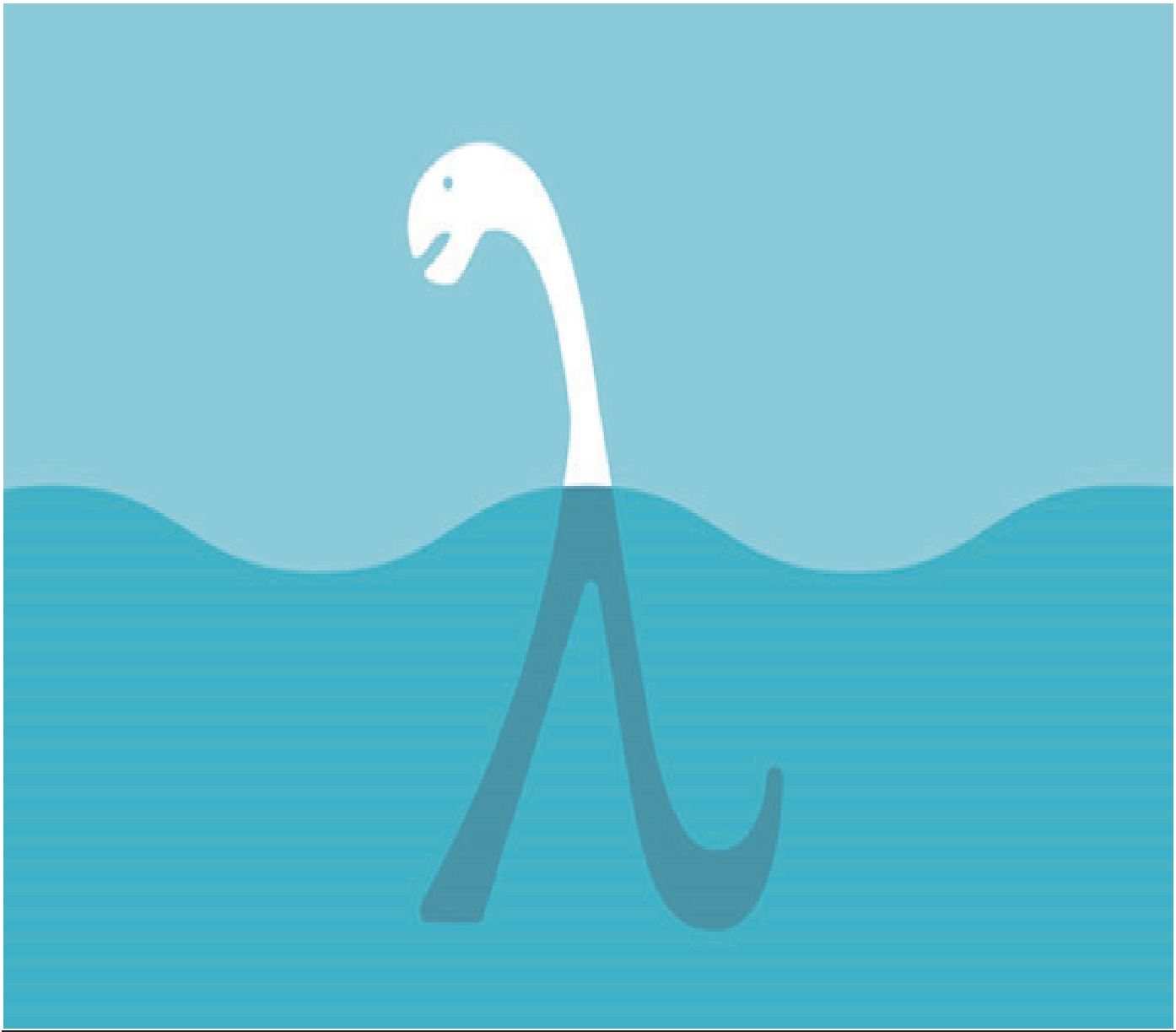
Conclusões

A linguagem Elm corresponde aos anseios de quem pretende fugir à arquitetura MVC e a quem deseja fazer programação funcional nos navegadores.

A linguagem Elm insere-se na família ML, tem variáveis dinâmicas (a declaração dos tipos das funções, tal como foi apresentada neste artigo, é opcional).

A linguagem Elm é pequena e muito pragmática pelo que é acessível a programadores que não conheçam programação funcional. A documentação, principalmente em Inglês, é clara e a comunidade é muito amigável.

Refira-se ainda que as mensagens de erro são muito claras, tentando guiar o programador diretamente para a origem do problema de uma forma direta e sugerindo soluções.



AUTOR

Escrito por Ricardo Miranda

ASP.NET Core 1.0 – Ligação a base de dados SQL Azure

Neste artigo vamos criar uma simples aplicação web utilizando ASP.NET 5. Esta aplicação irá guardar os dados numa base de dados SQL Azure utilizando Entity Framework (EF) Scaffolding e ASP.NET MVC para suportar as operações básicas (criação, leitura, atualização e remoção de dados).

Antes de mais é necessário ter acesso a uma conta Microsoft Azure. Qualquer pessoa pode ter acesso a uma conta gratuita, bastando para isso aceder ao endereço em baixo. Aceda ao link para obter mais informações de como criar a sua conta http://www.windowsazure.com/en-us/pricing/free-trial/

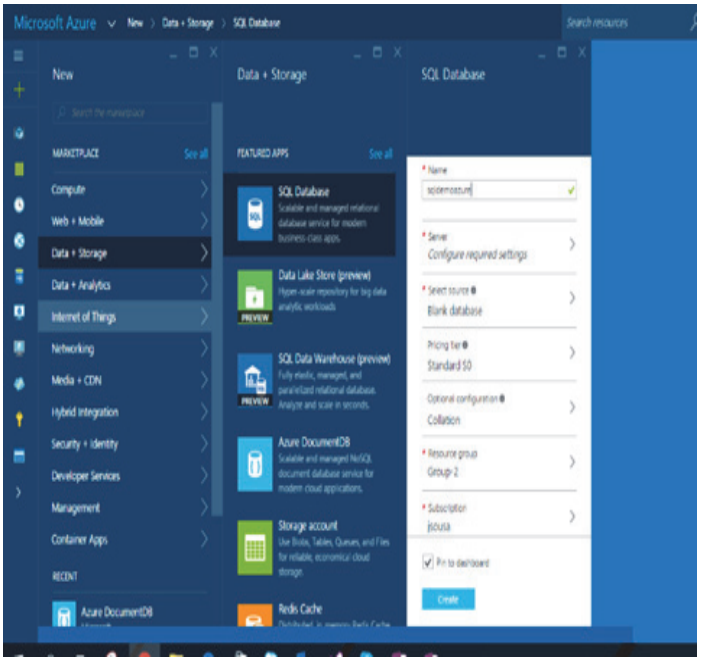
Criar a base de dados em Microsoft Azure

Após ter acesso a conta Microsoft Azure, é necessário criar uma instância de SQL que aloje a informação da nossa aplicação.

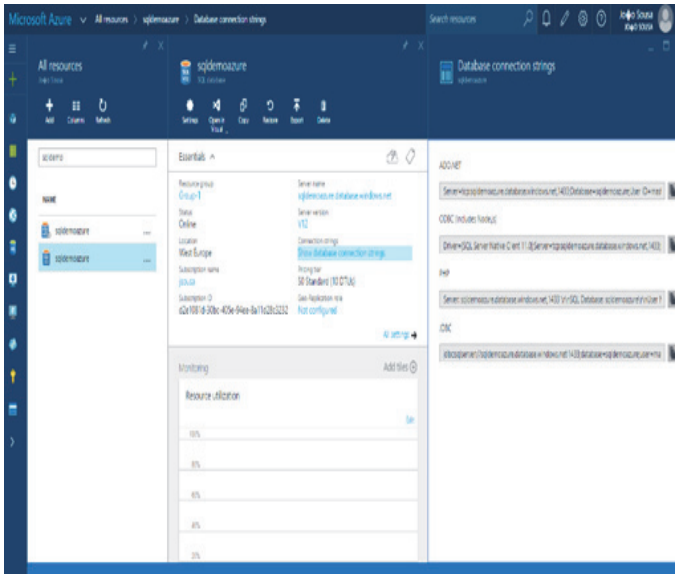
Para tal deve-se selecionar a opção de novo (+) no menu lateral esquerdo e nas opções que aparecem na primeira coluna, selecionar a opção “Data + Storage”.

Depois de selecionada esta opção uma segunda coluna irá aparecer, onde nos são dados a conhecer os diferentes fornecedores de dados disponíveis no Microsoft Azure.

Selecionemos então a opção “SQL Database”, dado que é o fornecedor, que iremos utilizar neste artigo.



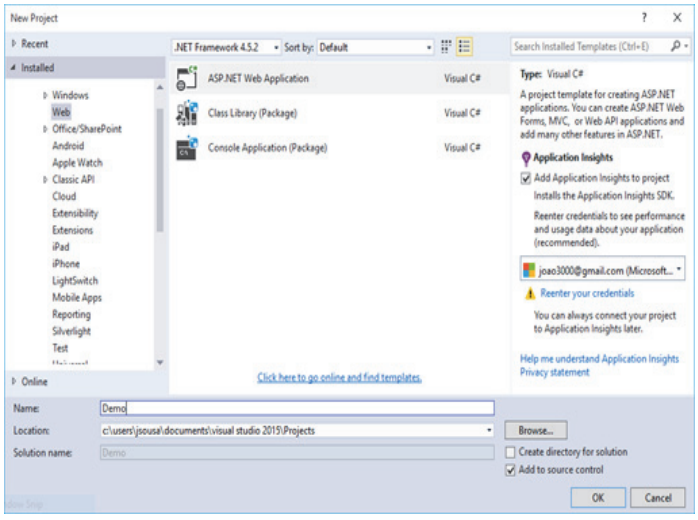
Depois de selecionada a opção, é necessário configurar a base de dados selecionada. No caso deste artigo, daremos o nome de "SQLDemoAzure" à nossa base de dados.



Após a criação da base de dados, necessitamos de obter a conexão à base de dados, a qual será utilizada na nossa aplicação web. Para isso, se selecionarmos a base de dados que criamos, tal como mostrado na imagem acima, existe uma opção “Show Connection String”, a qual nos dá acesso à conexão que necessitamos.

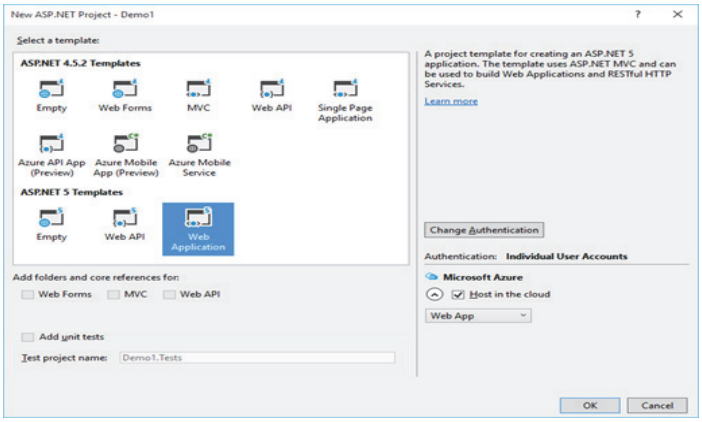
Criar a Aplicação Web

Para começarmos a criar a nossa aplicação web, vamos ter de selecionar a opção “ASP.NET Web Application” do template Web.

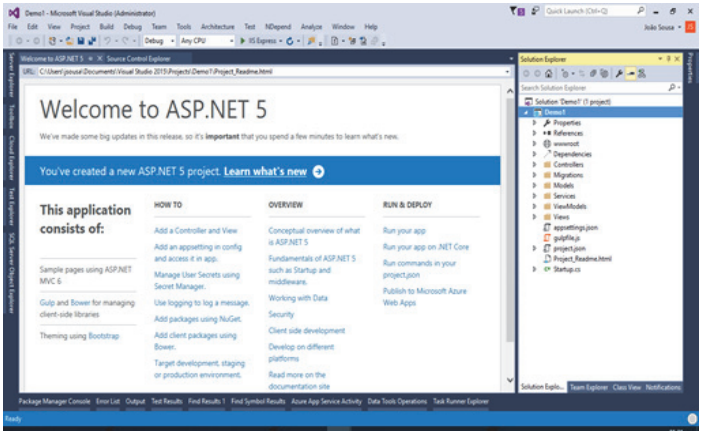


Ao darmos um nome á nossa aplicação web e carregando na opção OK, vamos ser reencaminhados para um novo ecrã onde nos é possível selecionar qual o template a ser aplicado. Como vamos querer desenvolver a aplicação em ASP.NET 5, temos três opções ao nosso dispor. Em

branco, WebAPI ou Web Application. Será esta terceira a opção a qual devemos selecionar.



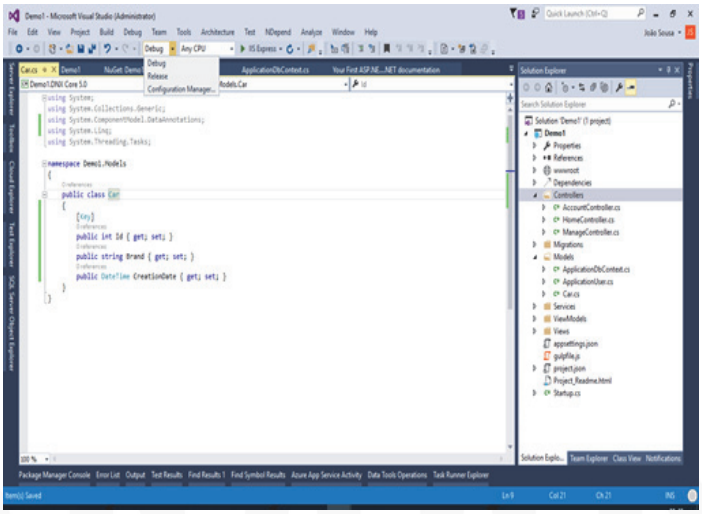
Depois de selecionado o nosso template, a aplicação começará a ser criada e configurada.



Criar modelo de dados

Depois da criação da nossa aplicação web, o passo seguinte será a definição do modelo de dados a ser utilizado. Para isso, no projeto web, define-se uma nova classe tal como definida na imagem abaixo.

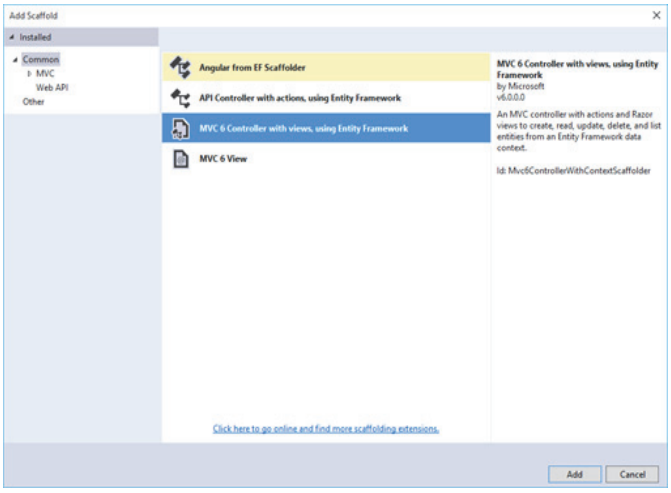
Esta classe é uma classe simples que permite definir propriedades relacionadas com um veículo.



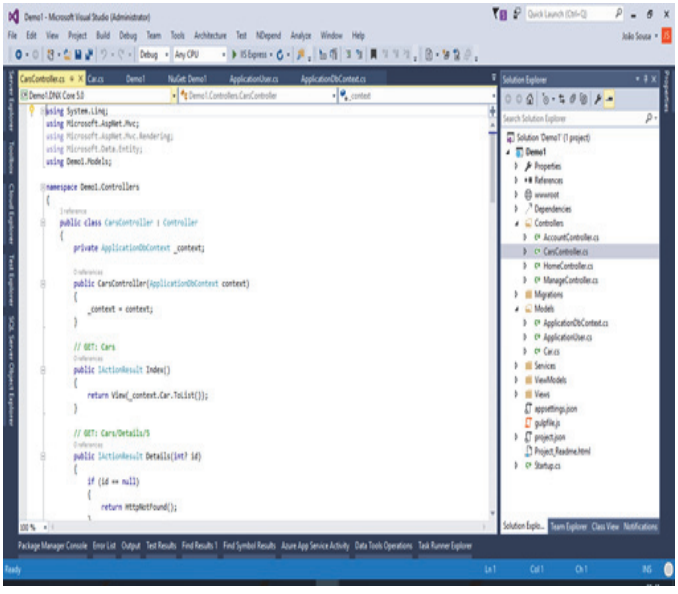
Adicionar Scaffolding

Scaffolding vai poupar tempo e esforço, proporcionando a geração automática de código que nos vai dar um ponto inicial de desenvolvimento das operações básicas (CRUD). Partindo de uma classe base e sem escrever uma única linha de código, conseguimos gerar o controller com as operações CRUD bem como as views associadas.

Na solução selecionando a diretoria dos controllers, basta selecionar a opção “Add New Scaffold Item”. Será aberto um novo ecrã tal como indicado na imagem abaixo, no qual devemos selecionar o “MVC6 Controller With Views Using Entity Framework”.



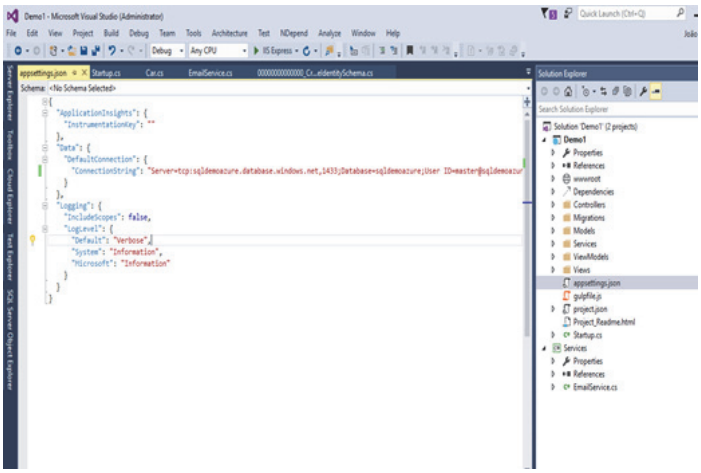
Desta forma, os controllers e views associadas a entidade criada, serão criados de forma automática e com sucesso.

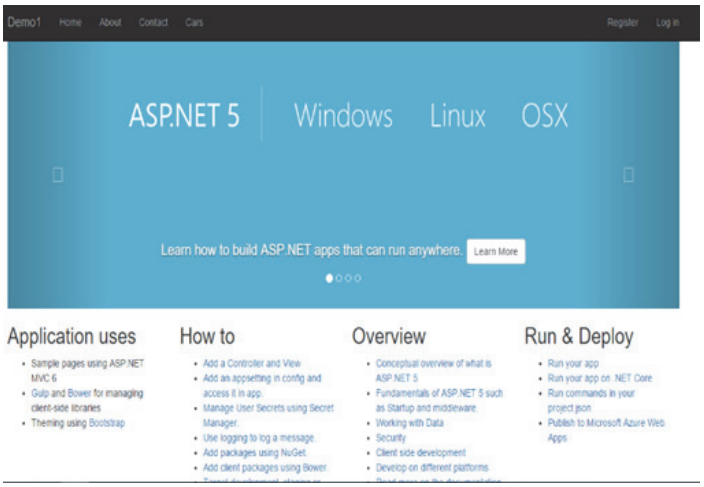


Alterar a Connection String

Para alterar a connection string, basta copiar a connection string definida no Segundo passo e colocá-la no ficheiro appsettings.json.

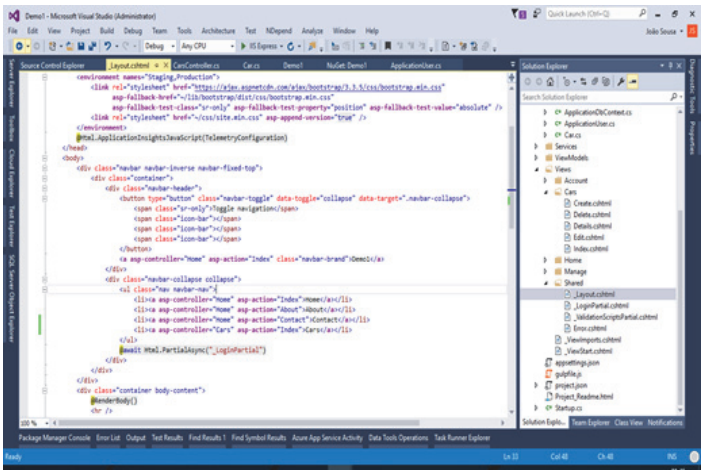
ASP.NET CORE 1.0 – LIGAÇÃO A BASE DE DADOS SQL AZURE





Alterar o layout do menu

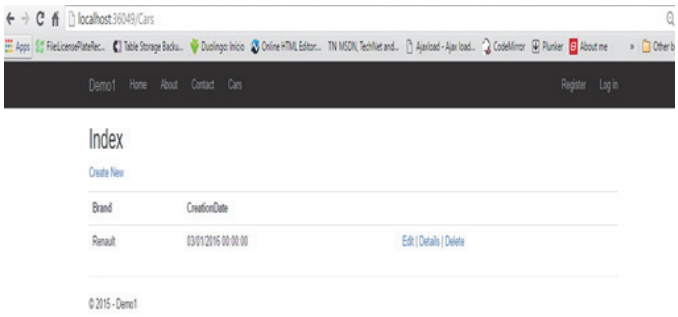
Para conseguirmos testar a nossa entidade, vamos adicionar uma nova entrada ao menu existente de forma a que quando clicarmos nessa opção seja dado acesso à listagem da entidade criada.



Correr a aplicação

Pressionado a Tecla F5 ou selecionando a opção Debug -> Start Debugging no Visual Studio, a aplicação irá ser aberta no browser. As novas entidades criadas irão aparecer no menu de topo, tal como mostra a imagem seguinte.

Selecionando a opção Cars, temos então acesso ao ecrã que geramos automaticamente utilizando o Scaffolding.



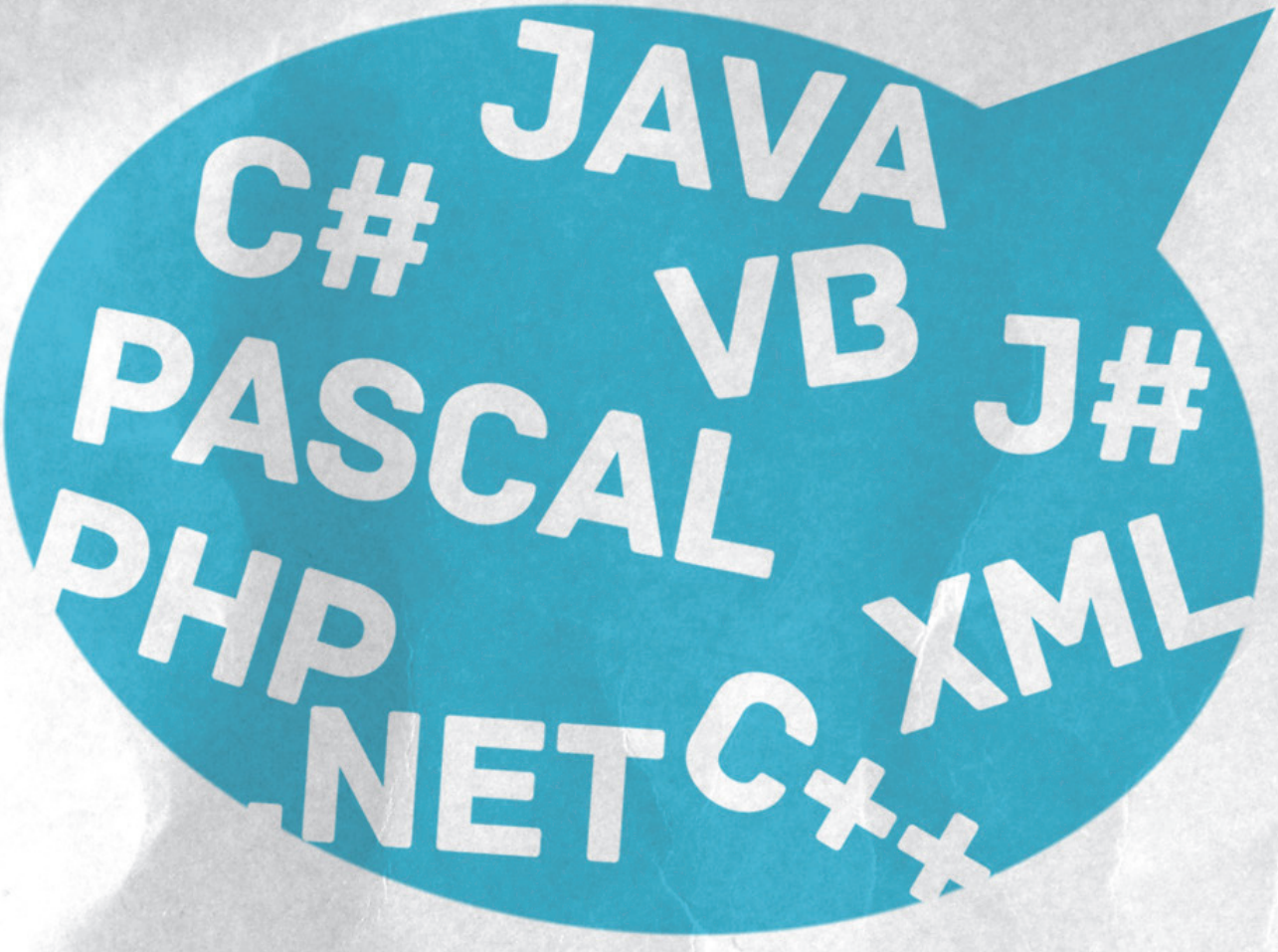
AUTOR



Escrito por Escrito por João Sousa [Microsoft MVP | MSDN Author | TechNet Wiki Technical Guru | MCPD | MCTS | Product Manager]

Actualmente Product Manager na empresa DevScope. É Microsoft Most Valuable Professional (MVP) em Visual Studio and Development Technologies. O seu principal foco de interesse são as tecnologias .NET com foco na parte Web e Azure.

É um participante bastante activo nos fóruns da Microsoft (MSDN Forums), contribuidor no MSDN Code Gallery e autor do Blog: https://joaoeduardosousa.wordpress.com/ - Twitter: @joaoedusousa



ENTÃO, SÓ FALAS EM CÓDIGO?

TEMOS O REMÉDIO CERTO PARA TI!

portugal-a-programar.pt

A MAIOR COMUNIDADE PORTUGUESA DE PROGRAMAÇÃO, APARECE!

ELECTRÓNICA

Automação com Genuino

AUTOMAÇÃO COM GENUINO

Introdução

A plataforma Genuino, tornou-se bastante popular ao longo dos anos, sendo uma das plataformas mais usadas em projectos de IoT e automação, tanto pelo seu baixo custo como pela sua capacidade de processamento e de I/O (input/output). Para os leitores que estão menos familiarizados com este tema, o Genuino é a análogo da marca Arduino.cc, tendo sido criado pelos co-fundadores do Arduino Massimo Banzi , David Cuartielles , Tom Igoe e David Mellis. Na prática, o Genuino é usado para as boards vendidas fora dos EUA.

Isto é, a marca Genuino certifica também a sua autenticidade com a filosofia de hardware aberto e de código aberto que sempre caracterizou os valores do Arduino. As placas Genuino e Arduino partilham os mesmos componentes e as mesmas características. São as mesmas boards, mas a "marca" do circuito é determinada pelo país. O Arduino é comercializado dentro dos EUA e o Genuino fora dos EUA.

Uma das tarefas mais simples que podemos realizar com o Genuino, consiste no "sentir" o mundo exterior! Sentir no sentido figurado, obviamente, pois as máquinas não sentem, "apenas" executam leituras, processamento e escrita. Sentir, neste caso, quer dizer "obter informação sobre o meio ambiente". Ora como podemos fazê-lo? De diversas formas, usando para este fim sensores que comunicam recorrendo a diferentes protocolos de comunicação que executam leituras de diferentes parâmetros. Mais adiante no artigo faremos essa abordagem. Em suma, este artigo foca-se na comunicação entre o genuíno e sensores, tendo como projecto final um circuito que executa algumas tarefas simples de interacção com o mundo exterior.

Sensores

Como referido na introdução, utilizam-se sensores para obter dados sobre o mundo "exterior", recorrendo a dispositivos que efectuam leituras. Para ligar estes sensores ao Genuíno, recorremos ao GPIO (General Purpose Input Output). Dependendo do tipo de sensores, podemos usar entradas analógicas (que recebem informação em formato de voltagem), entradas digitais (que recebem informação em formato binário) e entradas dedicadas a determinados protocolos, como por exemplo I2C, PWM, etc…

Nota: No caso de se querer ligar um sensor analógico a uma porta analógica, pode ser usado um circuito ADC (Conversor Analógico Digital) tipo MCP3008.

Aqui surge o primeiro desafio: quando o sensor trabalha com uma voltagem diferente da do circuito. Neste caso recorremos a resistências pull-up e/ou pull-down ou a reguladores de tensão, conforme o que o leitor desejar executar. A resistência deve sempre ser calculada de acordo com a especificação do sensor que estamos a usar. Neste artigo serão usados sensores de Humidade e Temperatura DHT-11, ligados a portas digitais, bem como sensores de humidade ligados a portas analógicas.

É importante que o leitor tenha presente que existem diversos tipos de sensores, para as mais diversas finalidades, propósitos, dados, valores, leituras, etc… No caso específico deste artigo, serão usados sensores de humidade e temperatura do ar, humidade do solo, luminosidade e pressão barométrica de forma a ler dados do mundo "exterior".

Actuadores

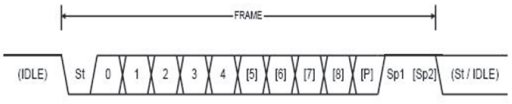
Contrariamente aos sensores, estes dispositivos não efectuam leituras de informação do mundo real. Os actuadores interagem com o mundo real, executando alguma tarefa. Um exemplo deste tipo de circuito é o tradicional relé, que tem dois estados possíveis (ligado / desligado), permitindo por exemplo, ligar ou desligar um equipamento eléctrico (Até mesmo uma maquina de café! Fica o desafio ao critério da imaginação do leitor, fazer um circuito que tire café sozinho).

O protocolo

Uma grande parte dos sensores disponíveis comunica com o circuito por I2C e UART (Universal Asynchronous Receiver-Transmitter). O protocolo UART permite a transmissão de dados em stream de bits, de forma sequencial de maneira a que no destino os conjuntos sejam reunidos em bytes completos. Cada pacote UART usa um shift register que é fundamental para a conversão entre a forma série e a paralela. A transmissão em série por um único condutor é mais fácil que a transmissão paralela recorrendo a múltiplos condutores. Neste caso concreto iremos apenas usar UART para comunicar com o computador, apesar de o podermos usar por exemplo, para comunicar com um circuito wifi 802.11b/g.

Na figura seguinte, encontra-se representada uma frame UART, para facilitar a compreensão do leitor.

Figure 22-4. Frame Formats



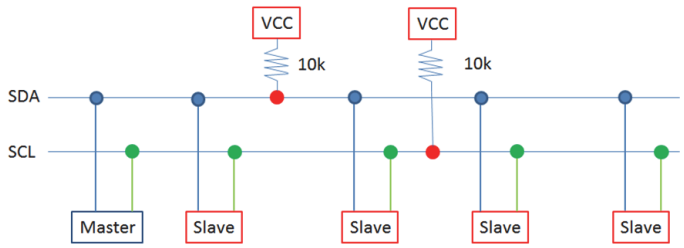
St	Start bit, always low.
(n)	Data bits (0 to 8).
P	Parity bit. Can be odd or even.
Sp	Stop bit, always high.
IDLE	No transfers on the communication line (RxDn or TxDn). An IDLE line must be high.

A comunicação entre os sensores e o Genuino será feita com recurso a outros protocolos, nomeadamente I2C (Inter-Integrated Circuit).

Este protocolo caracteriza-se por ser um protocolo de comunicação em série síncrono, por partilhar o clock e proporcionar a utilização de múltiplos Mestres (masters) e escravos (slaves). O comprimento de bits é fixo, independentemente do número de escravos (slaves) utilizados.

O I2C recorre apenas a dois fios condutores (SDA Serial Data e SCL Serial Clock). Se ambas as linhas de comunicação estão abertas, o estado do barramento é sempre conhecido do software e encontra-se em estado "high", por padrão. Neste protocolo o mestre (master) inicia e termina a transmissão, bem como gera o sinal SCL (clock), os escravos "slaves", são endereçados pelo mestre "master", o transmissor é quem coloca os dados na linha e o receptor recebe os dados da linha.

Quer o SDA, como o SCL são bidireccionais.



O circuito

Chegados à parte prática do nosso artigo, e como o objectivo deste não é um circuito de grande complexidade, vamos usar um sensor de humidade e temperatura, DHT-11, um sensor de luminosidade e um sensor de humidade do solo. Os sensores estarão ligados ao genuino de forma simples. No caso do DHT-11 usaremos uma resistência pull-up e todo o circuito agirá de forma autónoma, mesmo quando desligado do computador.

Sempre que a humidade do solo desça abaixo de um valor de 40%, é accionado um relé, que em teoria estaria ligado a uma bomba de água que procederia à irrigação dos solos, até que o valor da humidade do solo retomasse os 40%.

As ligações são, todas elas, relativamente simples. No caso do DHT-11 é ligado à porta digital D2 do Genuíno, os relés, ficarão nas portas D4 e D5 do Genuíno e o sensor de luminosidade nas portas destinadas a I^2C.

Posto isto, no código iremos fazer todas as operações de comunicação e processamento, bem como para efeitos de depuração de erros, colocar mensagens a serem transmitidas via série para um computador. Estas mensagens podem ser removidas ou utilizadas para por exemplo gerar gráficos, como referido no artigo da edição passada da PROGRAMAR.

```
#include "DHT.h"
#include <SparkFunTSL2561.h>
#include <Wire.h>

#define DHTPIN1 2     // define o pino de dados
do sensor, neste caso 2
#define DHTPIN2 7     // define o pino de dados
do sensor, neste caso 7
#define DHTTYPE DHT11   // DHT 11 (basta uma
constante para dois sensores)
//main time delay before first run
#define mainTimeDelay 2000

//light sensor initial code
SFE_TSL2561 light;
boolean gain;     // Gain setting, 0 = X1, 1 =
                  //X16;
unsigned int ms;  // Integration ("shutter") time
                  //in milliseconds
//alterar conforme o sensor que se use (DHT 11,
//DHT 22 (AM2302), DHT 21 (AM2301) )
// Initializa DHT.
DHT dht1(DHTPIN1, DHTTYPE);
DHT dht2(DHTPIN2, DHTTYPE);
//define relé
int Relay1 = 4;
int Relay2 = 5;

int humidade;

//general setup ops
void setup()
{
  Serial.begin(9600);
  Serial.println("Serial Port OKay");
  dht1.begin();
  dht2.begin();


//light setup ops
  light.begin();
  unsigned char ID;
  if (light.getID(ID))
  {
    Serial.print("Got factory ID: 0X");
    Serial.print(ID,HEX);
    Serial.println(", should be 0X5X");
  }
  else
  {
    byte error = light.getError();
    printError(error);
  }
  gain = 0;
  unsigned char time = 2;
  Serial.println("Definindo intervalo...");
  light.setTiming(gain,time,ms);
  Serial.println("Power up...");
  light.setPowerUp();
//end of ligh sensor ops

//relay ops
  pinMode(Relay1, OUTPUT);      //Set Pin4 as
                                //output
  pinMode(Relay2, OUTPUT);    //Set Pin5 as
                              //output
  digitalWrite(Relay1, HIGH); //inicia o relay1
                              //como desligado
  digitalWrite(Relay2, HIGH); //inicia o relay2
                              //como desligado
}

void loop()
{
  //delay(mainTimeDelay);
  dhtTemp1();
  dhtTemp2();
  lightSaber();
```

```
  moisture();
}

void dhtTemp1()
{
  // Aguarda um pouco antes de iniciar medicoes
  delay(2000); //pode ser um delay inicial

  //as leituras demoram cerca de 250milisegundos,
   //logo as leuituras serão 2 segundos e meio
   //"atrazadas"
  float h = dht1.readHumidity();
  // le a temperatura em graus celcius
  float t = dht1.readTemperature();
  // verifica se alguma leitura falhou e sai para
  //recomecar
  if (isnan(h) || isnan(t)) {
    Serial.println("Falha na aquisição de dados do
sensor DHT11-1!");
    return;
  }
  float hic = dht1.computeHeatIndex(t, h, false);
  Serial.print("Humidade do ar (sensor1): ");
  Serial.print(h);
  Serial.print(" %\t");
  Serial.print("Temperatura do ar (sensor1): ");
  Serial.print(t);
  Serial.print(" *C \n");
  //Serial.print("Indice de calor: ");
  /* HI =
c1+c2T+c3R+C4TR+C5T^2+c6R^2+c7T^2R+c8TR^2+c9T^2R^2
  ver wikipédia https://en.wikipedia.org/wiki/
  Heat_index*/
    Serial.println();
}

void dhtTemp2()
{
  // Aguarda um pouco antes de iniciar medicoes
  delay(2000); //pode ser acomulado no delay
               //inicial
  //as leituras demoram cerca de 250milisegundos,
  //logo as leuituras serão 2 segundos e meio
  //"atrazadas"
  float h = dht2.readHumidity();
  // le a temperatura em graus celcius
  float t = dht2.readTemperature();
  // verifica se alguma leitura falhou e sai para
  //recomecar
  if (isnan(h) || isnan(t)) {
    Serial.println("Falha na aquisição de dados do
                                sensor DHT11-2!");
    return;
  }
  float hic = dht2.computeHeatIndex(t, h, false);
  Serial.print("Humidade do ar (sensor 2): ");
  Serial.print(h);
  Serial.print(" %\t");
  Serial.print("Temperatura do ar (sensor 2): ");
  Serial.print(t);
  Serial.print(" *C \n");
  //Serial.print("Indice de calor: ");
  /*HI =
c1+c2T+c3R+C4TR+C5T^2+c6R^2+c7T^2R+c8TR^2+c9T^2R^2
  ver wikipédia https://en.wikipedia.org/wiki/
Heat_index */
  Serial.println();
}

void moisture()
{
  humidade = analogRead(A0); //pino analógico A0
  int Porcento = map(humidade, 1023, 0, 0, 100);
  Serial.print(Porcento);
  Serial.println("% humidade do solo");
  if (Porcento <= 10)
  {
    Serial.println("Humidade do solo abaixo dos
                                          70%");
    digitalWrite (Relay1, !digitalRead(Relay1));
    digitalWrite(Relay1, HIGH);    //Turn on relay
  }
  else
  {
    digitalWrite(Relay1, LOW); //desactuiva o
                 //sinal no pino 13 (desliga relé)
    digitalWrite (Relay1, !digitalRead
(Relay1));  //Turn off relay
  }
//  delay(1000); //pode e deve ser acomulado no
//main time delay
}

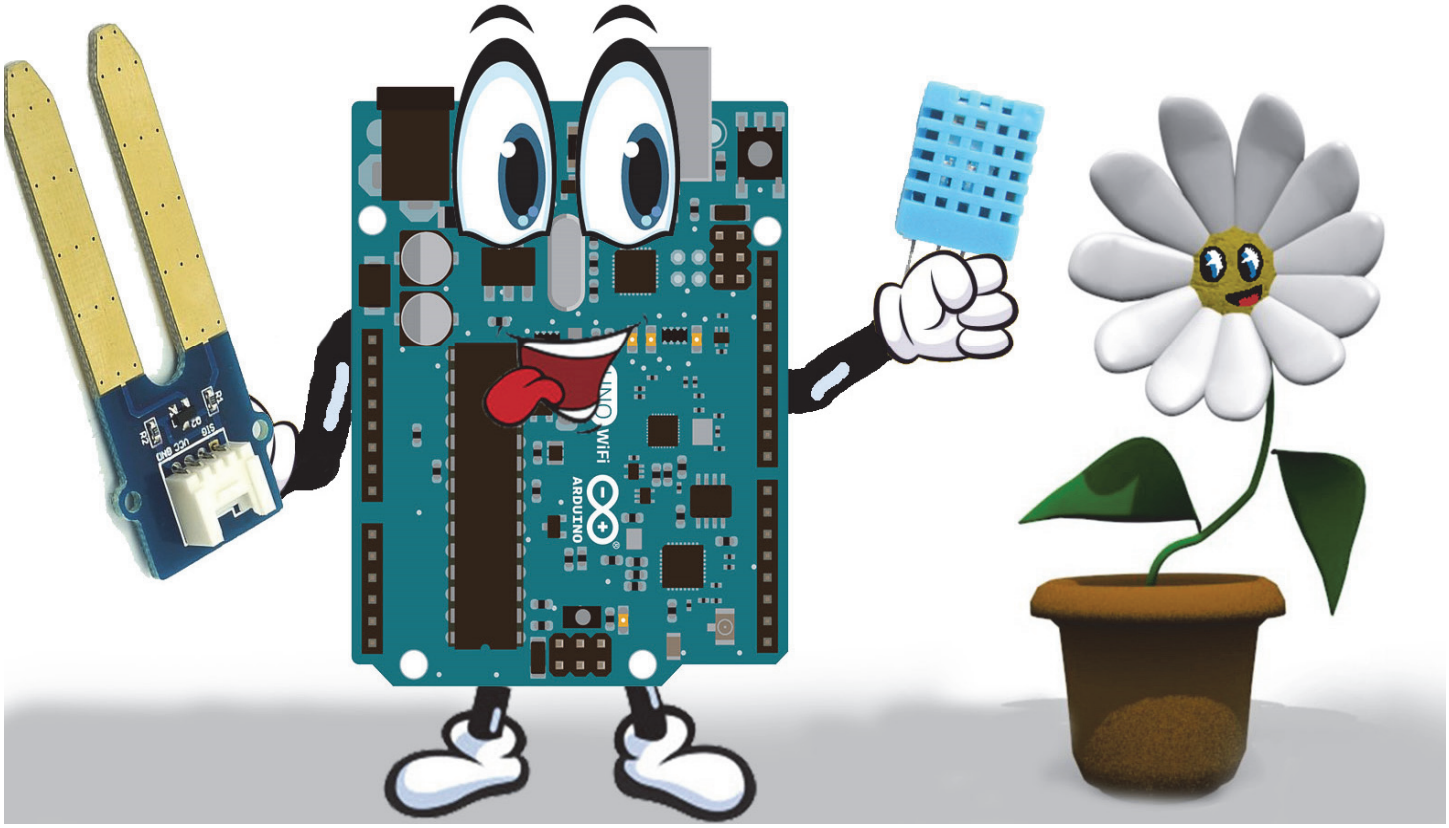
//light sensor functions:
void printError(byte error)
{
  Serial.print("I2C error: ");
  Serial.print(error,DEC);
  Serial.print(", ");
  switch(error)
  {
    case 0:
      Serial.println("success");
      break;
    case 1:
      Serial.println("data too long for transmit
buffer");
      break;
    case 2:
      Serial.println("received NACK on address
(disconnected?)");
      break;
    case 3:
      Serial.println("received NACK on data");
      break;
    case 4:
      Serial.println("other error");
      break;
    default:
      Serial.println("unknown error");
  }
}
//funcao principal do sensor de luz
void lightSaber()
{
  delay(ms);
  unsigned int data0, data1;
  if (light.getData(data0,data1))
  {
    /*Serial.print("Luminosidade data0:  ");
    Serial.print(data0);
    Serial.print("\n Luminosidade data1: ");
    Serial.print(data1);
    */
    double lux;
    boolean good;
    //calculo de Lux
    good = light.getLux(gain,ms,data0,data1,lux);
    Serial.print(" lux: ");
    Serial.print(lux);
    if (good) Serial.println(" (good) \n"); else
Serial.println(" (BAD) \n");
  }
  else
  {
    byte error = light.getError();
    printError(error);
  }
```

AQUISIÇÃO DE DADOS VIA TCP/IP COM GENUINO (ARDUINO)

Conclusão

Este artigo pretendeu apresentar uma breve introdução à automação com o Genuino, realizando tarefas bastante básicas e primitivas, explicando os dois protocolos mais utilizados bem como uma breve explicação sobre as ligações feitas ao Genuino e o código que será carregado no mesmo.

As potencialidades são imensas, pelo que deixamos ao critério do leitor, explorar mais um pouco este tema. Para a próxima edição, apresentaremos um projecto um pouco mais complexo, usando conceitos dos artigos da edição 51 e desta edição! Até lá!!!



AUTOR



Escrito por Rita Peres

Natural de Castelo Branco, licenciou-se em Engenharia Informática pela Universidade da Beira Interior. Membro do P@P desde Janeiro de 2010.



AUTOR



Escrito por António C. Santos

Com uma enorme paixão por tecnologia, autodidacta desde tenra idade, cresceu com o ZX Spectrum. Tem vasta experiência em implementação e integração de sistemas ERP, CRM, ERM, BI e desenvolvimento de software por medida nas mais diversas linguagens. Diplomado do Curso de Especialização Tecnológica em Tecnologias e Programação de Sistemas de Informação pela ESTG-IPVC. Membro da Comunidade Portugal-a-Programar desde Agosto de 2007, é também membro da Sahana Software Foundation, onde é Programador Voluntário. Neste momento é aluno no Instituto Politécnico de Viana do Castelo, na Escola Superior de Tecnologia e Gestão no curso de Licenciatura em Engenharia Informática e Xamarin Student Partner nesta mesma escola Twitter:@apocsantos



COLUNAS

Kernel Panic - Cross-Platform - “O papel do profissional de Marketing & Comunicação no sector das TI’s em Portugal”

O papel do profissional de Marketing & Comunicação no sector das TI's em Portugal

No 29.º Encontro das Geek Girls Portugal, lançaram-me o desafio de "transformar" a apresentação e escrever um artigo para a Programar, o que me traz hoje até vos, caros leitores.

Gostaria de vos apresentar e sensibilizar para debati alguns pontos sobre o contributo de um profissional de Marketing & Comunicação no sector das TI's. Ao longo deste artigo foco-me em alguns pontos debatidos ao longo do encontro, que se destinou essencialmente aos profissionais de Marketing que estejam a abraçar o desafio de desenvolver um gabinete de Marketing & Comunicação numa empresa do sector das TI's. De acordo com a minha experiência pessoal e profissional, deverão ter em mente os seguintes aspectos:

1 - **Promover a empresa do mercado nacional e internacional**. Portugal é um país de excelência para investidores, por apresentar custos reduzidos de investimento, vantagens associadas ao fuso horário, talento proveniente de um sistema de ensino reconhecido, e um excelente clima. Aproveitando essas vantagens, é necessário ter em mente a promoção da empresa a nível nacional, e também fora no nosso país. Essa presença passa por inicialmente estabelecer parcerias estratégicas com entidades internacionais, participar em feiras internacionais, colaborar com a imprensa local, participar em *ranking* e estudos, sendo que há muitas outras formas de o fazer, mas estas são um excelente ponto de partida.

2 – **Criação de uma forte cultura interna.** Antes de partir para uma exposição externa, é preciso alicerçar-nos com uma forte cultura interna. Neste sector das TI's a tendência é o informalismo, ou seja, o trato por "tu", a adopção de um *dress code* casual, contudo não dispensa o rigor e as boas regras. Criar rotinas a nível da comunicação é fundamental para todos os colaboradores estarem cientes da cultura da empresa para o qual desenvolvem a sua actividade profissional. Para criar uma cultura interna é necessário garantir que todos os colaboradores conheçam a cultura da empresa, ou seja, é preciso inicialmente dota-los de uma formação inicial, desenvolver uma calendarização de eventos, dinamizar academias internas, munir de parcerias estratégicas, e garantir que as mesmas acontecem de forma sistemática.

3 – **Comunicar a cultura da organização.** A comunicação deve ser transversal a todos os *stakeholders* da organização, é necessário adoptar canais de comunicação adequadas para cada um deles. É fundamental a criação de conteúdos criativos - artigos de opinião, partilha de experiências, através do uso de vídeo, criação de um tour virtual, uso do GIF, entre outros que estão em "voga". E partilhar junto dos canais da empresa – site, redes sociais, email, newsletter – e também em canais de comunicação externo – imprensa, rádio, televisão.

Para além de divulgar a cultura junto de outros públicos, todos os *stakeholders* gostam de sentir que a empresa está presente e contribui com artigos interessantes, com participações que acrescentam valor também para outras pessoas.

Alguns dos conteúdos deverão ser pensados em bilíngue, se poder alcançar outros públicos e fazer jus ao primeiro ponto que abordei.

Exemplos portais:

- Computerworld
- Sapo Tek
- IT Channel
- Portal Ntech
- Bit Magazine
- StartUPMag
- Portugal a Programar
- Outsourcing Magazine

4 – **Estar de mãos dadas com o Ensino**. Os estabelecimentos de ensino são uma fonte de inspiração e inovação. Estabelecer contacto irá trazer muitas vantagens para as organizações, que irão estar junto dos profissionais de amanhã. As acções com o ensino poderão passar por sessões de *mentoring;* colaboração com projectos académicos; acolher os alunos em sistema *on the job*, onde os alunos vêm conhecer de perto a empresa; e, a realização de *workshops* temáticos, poderão ser exemplos de actividades a desenvolver junto das entidades de ensino.

Exemplos de parcerias académicas:

- Pitch Bootcamp (Spark Agency)
- AIESEC
- Associação de Estudantes

- Participação no Dia Aberto nas Empresas (promovido pela IPAV)

5 – **Escuta activa e seguir tendências.** Ser sensível as necessidades dos diferentes públicos são fundamentais para conseguir ter sucesso nas acções que são desenvolvidas. Como tal, é necessário a escuta activa, conhecer as tendências do mercado. Actualmente existem inúmeras ferramentas que nos permitem rapidamente prosseguir com essa recolha, por exemplo, a recolha interna de *feedback* poderá ser efectuada através de um formulário online (ou através de um *software* próprio da empresa, ou através dos formulários do Google ou do Survmetrics). Outra plataforma para se confirmar tendências é o Google Trends, que permite confirmarmos as "palavras" mais pesquisas, possibilitando-nos da redacção dos conteúdos adequados. Ainda quando queremos validar potenciais parceiros a apostar na colocação da nossa marca, o Alexa, ajuda-nos e validarmos os sites com maior número de visualizações.

Exemplos de plataformas de recolha de informação:

- Survmetrics (https://survmetrics.com/)
- Alexa (http://www.alexa.com/topsites)
- Google Trends (www.google.pt/trends)



Um profissional de Marketing deve ter sempre presente a **criação de valor** em todas as suas acções, de forma a servir de facilitador entre todos os departamentos de uma organização, por exemplo, a realização de materiais de comunicação, com um design *clean* e coeso, e que facilite o dia-a-dia, de quem os usa. Em todas as acções e parcerias é importante **inovar** e não cair na "rotina". A inovação deve estar relacionada com todos os aspectos referidos anteriormente, ou seja, nos materiais, nos canais de comunicação, fornecedores, entre outros. Outra premissa passa pela **criação estratégica de parceiros/protocolos**. A criação de protocolos permite que outras organizações possam contribuir também para a área das TI's. É importante que sejam criadas parcerias que venham contribuir e sustentar a cultura da organização. O acompanhamento dessas parcerias deve ser sustentado pelo desenvolvimento de actividades em comum.

Todos os aspectos que referi carecem da realização de **plano de comunicação anual**, que é fundamental para que haja planeamento prévio e um alinhamento com a estratégia da empresa, para que seja dado espaço para a criatividade actuar. É importante estar ciente que qualquer profissional, nomeadamente o de Marketing & Comunicação depende da colaboração de todos os departamentos, pois só essa sinergia irá garantir o sucesso e cumprimento de todos os objectivos.

O sector das TI's apresenta uma grande dinâmica e competividade, pelo que é importante que o os profissionais da área do Marketing actuem no sentido de permitir que todos os que lidam com a organização possam ter uma **experiência única** e uma história positiva para partilhar sobre a mesma.



AUTOR



Escrito por Ana Rita Queiroz

Marketing Manager da AFFINITY

Linked in: https://pt.linkedin.com/in/anaritaqueiroz

Media Partners da Revista PROGRAMAR



























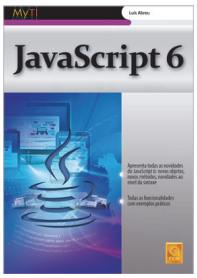


Análises

JavaScript 6

Modelação de Dados em UML – Uma Abordagem por Problemas

JavaScript 6



Título: JavaScript 6

Autores: Luís Abreu

Editora: FCA - Editora de Informática

Páginas: 192

ISBN: 978-972-722-815-7

Âmbito

O *JavaScript* já não é só a linguagem de script da *Web*. Com o passar dos anos, cada vez mais pessoas recorrem a esta linguagem para automatizar a realização de várias tarefas, uma vez que tem um número infindável de utilizações práticas, que vão desde a criação de páginas *Web* até à administração de máquinas com diferentes sistemas operativos. Por conseguinte, o conhecimento das funcionalidades de *JavaScript* é uma mais-valia no currículo de qualquer programador.

Introdução

Neste artigo são apresentadas as principais novidades, características e funcionalidades da versão 6 da linguagem *JavaScript*, com recurso a exemplos práticos. Para além de novos objetos e métodos que foram adicionados aos objetos existentes, esta nova versão introduz também várias novidades ao nível da sintaxe (por exemplo, a definição de classes) que contribuem para simplificar a escrita de código e aumentar a produtividade do programador.

Conteúdos

O livro começa por apresentar uma breve história da linguagem *JavaScript* que culminou com a introdução da última versão da especificação *ECMAScript* (*Harmony*). De seguida são abordados alguns problemas que podem preocupar o programador que deseja iniciar a escrita de código *JavaScript 6*.

No capítulo 2 é apresentado o conceito de bloco associado ao uso dos novos termos *let* e *const* e, de seguida, são ilustradas algumas das vantagens inerentes ao uso de novas instruções de desestruturação. O capítulo apresenta ainda as várias novidades introduzidas nas *APIs* dos objetos básicos e culmina com a introdução dos novos tipos de coleções que foram adicionados à nova versão (*Set*, *Map*, *WeakSet* e *WeakMap*).

O capítulo 3 apresenta uma análise detalhada da criação e utilização de funções, que contribuem para uma melhoria de produtividade do programador. Neste contexto, são abordados tópicos como a inicialização predefinida de parâmetros, a utilização de parâmetros *rest*, a utilização do operador *spread* e uma nova propriedade (*name*). O capítulo refere ainda a nova sintaxe de funções anónimas, nomeadamente o uso de funções *arrow*.

O capítulo 4 apresenta os conceitos de objetos e classes, nomeadamente algumas alterações introduzidas na definição de novos métodos adicionados ao objeto *object*, e ainda algumas novidades relacionadas com a definição de classes. Estas alterações simplificam a definição de propriedades e métodos, e introduzem novos termos que simplificam a criação de classes em *JavaScript*. Foi também introduzido o conceito de *proxy* que é bastante útil em vários cenários onde é necessário intercetar o acesso aos membros de um objeto.



O capítulo 5 apresenta uma visão sobre os *iterators*. Na nova versão o *JavaScript* passa a suportar os conceitos de *iterator* e *iterable*. São apresentadas as vantagens associadas ao uso de *iterators* e outros detalhes relacionados com a criação e utilização deste tipo de elementos em *JavaScript*. A criação de *iterators* é simples, especialmente quando recorremos às novas funções geradoras. Não obstante, a introdução de ciclos *for of* permite simplificar o código usado para enumerar os items armazenados em todas as coleções classificadas como *iterable*.

O capítulo 6 aborda as funcionalidades associadas ao uso de símbolos em *JavaScript*. As propriedades dos objetos no *JavaScript* são *strings* e esta liberdade de definir qualquer nome acabou por criar alguns padrões. O *JavaScript 6* introduziu o conceito de símbolos que podem ser usados como chave de propriedades, sendo possível criar chaves únicas para as nossas propriedades. A especificação atual permite

a criação de novos símbolos locais e globais, e a introdução de um conjunto de símbolos predefinidos, que permitiram tornar públicos muitos dos mecanismos que, até ao momento, eram internos.

> ***"O JavaScript já não é só a linguagem de script da Web. Com o passar dos anos, cada vez mais pessoas recorrem a esta linguagem para automatizar a realização de várias tarefas, uma vez que tem um número infindável de utilizações práticas (...)"***

O capítulo 7 aborda o conceito de programação assíncrona. Com o lançamento de *JavaScript 6*, o ambiente de *runtime* passa a disponibilizar novas funcionalidades que facilitam o programador na área de programação assíncrona. Neste contexto, são analisadas as principais características associadas ao uso de *promises*. A especificação *Harmony* introduziu o conceito e define uma *API* que nos permite simplificar a forma como escrevemos código assíncrono em *JavaScript*. A título de exemplo, podemos recorrer a funções geradoras para realizar tarefas assíncronas.

O oitavo e último capítulo apresenta o conceito de módulos. Os módulos permitem encapsular porções de código que podem ser consumidas por outras aplicações. Com a especificação do conceito de módulos, passamos a ter um mecanismo comum para a definição destes elementos em todos os ambientes de *runtime* existentes.

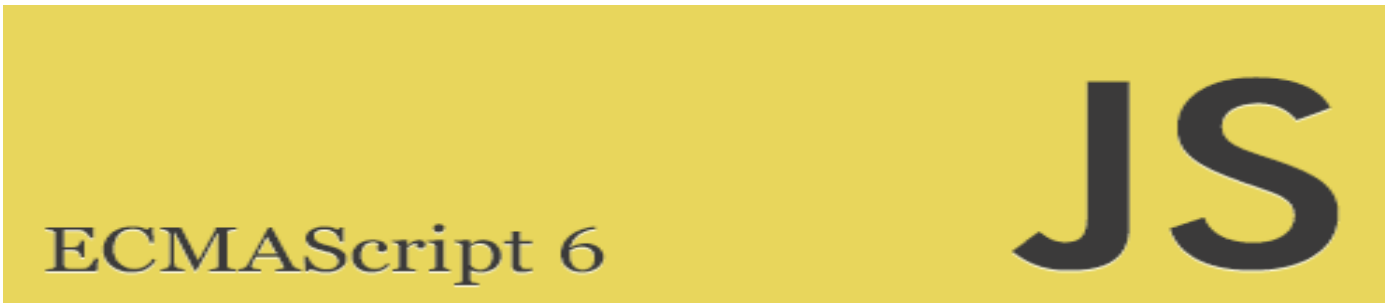
Conclusão

Nos dias que correm não há dúvida quanto à importância da linguagem *JavaScript*, pelo que o conhecimento das suas funcionalidades é uma mais-valia no currículo de qualquer programador.

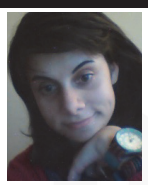
O *JavaScript* é uma linguagem dinâmica, cuja correta utilização pressupõe o uso de determinados padrões de programação, que já não são muito diferentes daqueles usados pelas linguagens tradicionais de desenvolvimento orientado a objetos.

Para além de introduzir as novidades da nova versão *JavasScript 6*, este livro apresenta várias recomendações que contribuem para a escrita de bom código *JavaScript*.

De realçar que todas as funcionalidades e caraterísticas apresentadas não dependem de nenhum ambiente de *runtime* específico, fazendo assim com que esta obra seja adequada não apenas para programadores *Web*, mas também para todas as pessoas interessadas em usar *JavaScript*.



AUTOR



Escrito por Tânia Valente

Natural de Coimbra, licenciou-se em Engenharia Informática pelo Instituto Superior de Engenharia de Coimbra e, actualmente, frequenta o mestrado em Human Computer Interaction. É entusiasta na área de Desenvolvimento Web, no que concerne às Tecnologias Web, Design de Interface (UI) e User Experience (UX). Curiosa e motivada por novos desafios, acredita que a criatividade pode transformar a maneira como as pessoas pensam, sentem e agem.

Modelação de Dados em UML – Uma Abordagem por Problemas

Título: Modelação de Dados em UML – Uma Abordagem por Problemas

Autores: José Luís Moura Borges | João Falcão e Cunha | Teresa Galvão Dias

Editora: FCA - Editora de Informática

Páginas: 192

ISBN: 978-972-722-812-6



A utilização de linguagens formais para representação de aspectos de sistemas informáticos tem vários anos, tendo o UML – Unified Modeling Language – sido adoptado como um standard pelo Object Management Group (OMG) em 1997. No entanto, creio poder dizer que essa formalização, nos últimos anos, tem caído algo em desuso, em detrimento de modelos menos formais mas mais perceptíveis: ou seja, os developers têm-se preocupado mais em desenhar, quando o fazem, diagramas que sejam perceptíveis por outros, sem se preocuparem excessivamente no grau de adequação desses diagramas a standards existentes, dos quais o UML sobressai.

E, no entanto, esses standards existem, e é precisamente do standard UML – ou, mais precisamente, de uma parte dele – que os autores nos vêm falar neste Modelação de Dados em UML, editado pela FCA. Sendo académicos, é normal que assim seja, e é, talvez até, necessário: quantas vezes não vimos já, diagramas feitos por diferentes pessoas, pretendendo ser UML, apresentarem diferenças substanciais na abordagem e na simbologia usada? No caso em apreço, o livro foca-se nos diagramas de classes, um dos diagramas estruturais mais comummente utilizado, e que se presta igualmente bem quer à representação de classes de linguagens orientadas a objectos, quer à representação de entidades ou tabelas de um modelo de dados.

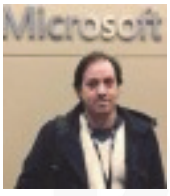
O livro é de fácil leitura, com cerca de 160 páginas, e encontra-se dividido em duas partes: na primeira é-nos apresentado um problema concreto – a gestão de uma empresa de produção de vinhos -, e, na segunda, um conjunto de problemas também realistas, mas já não relacionados com o exemplo da primeira parte – recrutamento, agências de modelos, gestão de cirurgias, entre outros. A discussão é muito centrada no problema em apreciação, sem grandes divagações teóricas. Ainda assim, é desejável que o leitor possua algum conhecimento de modelação de dados.

Na primeira parte, após uma descrição do cenário, a tal empresa vitivinícola, passamos ao enumerar dos requisitos do sistema que pretendemos construir. Com base nestes requisitos, passamos então às entidades do sistema, extraídas a partir destes, as quais deverão permitir a adequada representação dos conceitos do negócio. Para tal, são enumerados os vários tipos de relacionamento possíveis entre as entidades, e é explicado o que usar em cada caso, entre as várias entidades. São dados alguns conselhos práticos, tais como, quando promover um atributo a uma classe. Vamos progredindo nos vários aspectos do sistema, incluindo a apresentação de esboços da interface de utilizador que este poderia ter, e terminamos com as técnicas para conversão do modelo de classes nas tabelas "físicas" correspondentes numa qualquer base de dados. Aqui, creio que teria sido útil abordar o conceito de normalização, embora, é verdade, os exemplos apresentados estejam já num nível adequado de normalização, e também alguns conceitos algo mais avançados, como a herança e a composição.

Na segunda parte do livro, são-nos apresentados vários problemas, não relacionados com o exemplo da primeira. Para cada um, existe uma descrição do problema, e também uma resolução possível, comentada. Sendo certo que os exemplos apresentados excedem em muito o âmbito do problema original, fornecendo mais matéria para pensar, penso que teria sido útil a manutenção do mesmo cenário, ainda que, claro está, complementado com novos problemas, conducentes a novos desenvolvimentos, e novas necessidades de especificação.

Em resumo, creio que o livro faz uma apresentação interessante de uma pequena parte do standard UML, mas penso que poderia ter feito um bocado mais. Certos tópicos ou não foram de todo abordados – caso da normalização – ou foram-no de forma ligeira – chaves e geração do modelo físico. Para uma introdução ligeira, poderá ser útil, mas quem procure algo mais abrangente, deverá consultar outras referências.

AUTOR



Escrito Por Ricardo Peres

Evangelista Tecnológico na Simplifydigital. Microsoft MVP.

No Code

O que esconde o CQRS

Comunidades e voluntariado presencial

SHIFT APPENS 2016 - COIMBRA

RASPBERRY PI 3 MODEL B

Windows 10: Gestão de redes Wireless através da Linha de Comandos

Projecto em Destaque na Comunidade p@p: Matraquilhos

O QUE ESCONDE O CQRS

Introdução

O que é o CQRS (Command Query Responsibility Segregation)? Quais os seus benefícios? Que desafios esconde?

Estas são algumas das questões que este artigo pretende responder, demonstrando que o CQRS parte de um princípio simples e que é capaz de evoluir para dar resposta a problemas mais complexos.

O que é o CQRS?

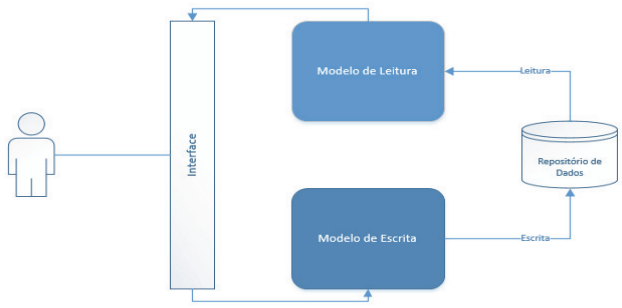
As origens do CQRS são baseadas no CQS (Command-Query separation), um princípio criado por Bertrand Meyer durante o desenvolvimento da linguagem de programação Eiffel.

Meyer afirma que fazer uma pergunta não deve alterar a sua resposta. Esta é a ideia basilar do CQRS.

Analisando esta afirmação podemos concluir que um comando corresponde a qualquer método que cria, atualiza ou remove dados, e que uma consulta é qualquer método que retorne dados.

É evidente que o CQRS é uma evolução do CQS, elevando o conceito inicial a um novo nível, afirmando que o modelo de leitura e de escrita devem ser distintos. Em suma, **CQRS é simplesmente isso, uma separação das duas responsabilidades**.

Se olharmos friamente para este princípio, vemos o reflexo de algumas das preocupações que tem surgido no desenvolvimento de software. Desnormalizar a base de dados, utilizar sistemas de cache e utilizar bases de dados otimizadas para consultas, são alguns dos exemplos de que a indústria já se encontra a trabalhar para resolver alguns dos problemas a que o CQRS pretende dar resposta.



Muitas vezes o CQRS é apresentado em conjunto com outras técnicas ou peças tecnológicas, como o Event Sourcing ou o NServiceBus, mas nada no CQRS diz que devemos utilizá-los. É importante salientar que estes elementos adicionam um nível adicional de complexidade que deve ser evitado caso não seja necessário.

Benefícios do CQRS

Todos os benefícios aqui apresentados são capacidades que o CQRS fornece, no entanto, estes não são requisitos para a implementação do CQRS.

Consultas mais simples

Com a separação do modelo de leitura, do modelo de escrita, é possível criar consultas mais simples e com melhor desempenho. Quando o modelo de leitura e escrita são partilhados, somos levados a aplicar condicionantes às consultas, devido aos requisitos presentes no modelo de escrita e vice-versa.

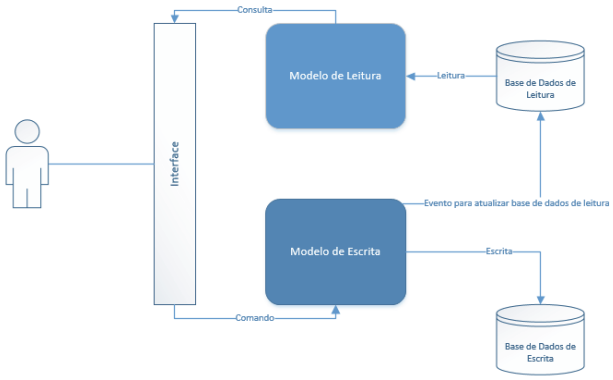
O sistema irá assim beneficiar de ter uma camada de leitura mais fina e otimizada. Citando Udi Dahan, Código simples é Código rápido.

Bases de dados separadas

Uma vez que o modelo de leitura está separado, é possível dividir a base de dados, possibilitando consultas a uma base de dados otimizada para leitura.

Esta possibilidade leva a conceitos como Eventual Consitency e Event Sourcing.

Nota: É perfeitamente possível implementar CQRS partilhando a base de dados entre os modelos.



Fila de trabalho

Ao separar os dois modelos, permitimos que os comandos possam ser processados utilizando uma queue,

habilitando um processamento assíncrono do trabalho a executar. Isto pode ser útil para lidar com processamentos complexos ou para gerir a carga no sistema.

Separação da carga

Tipicamente, o número de consultas num sistema é muito superior ao número de comandos realizados. Este facto levanta a questão de porque é que tipicamente se utiliza as mesmas tecnologias e padrões para responder a consultas e a escritas?!

O CQRS permite a separação da carga, com possibilidade de utilizar servidores diferentes para cada parte do sistema. Por exemplo, podemos perfeitamente utilizar tecnologias completamente distintas para leitura e escrita.

Desafios

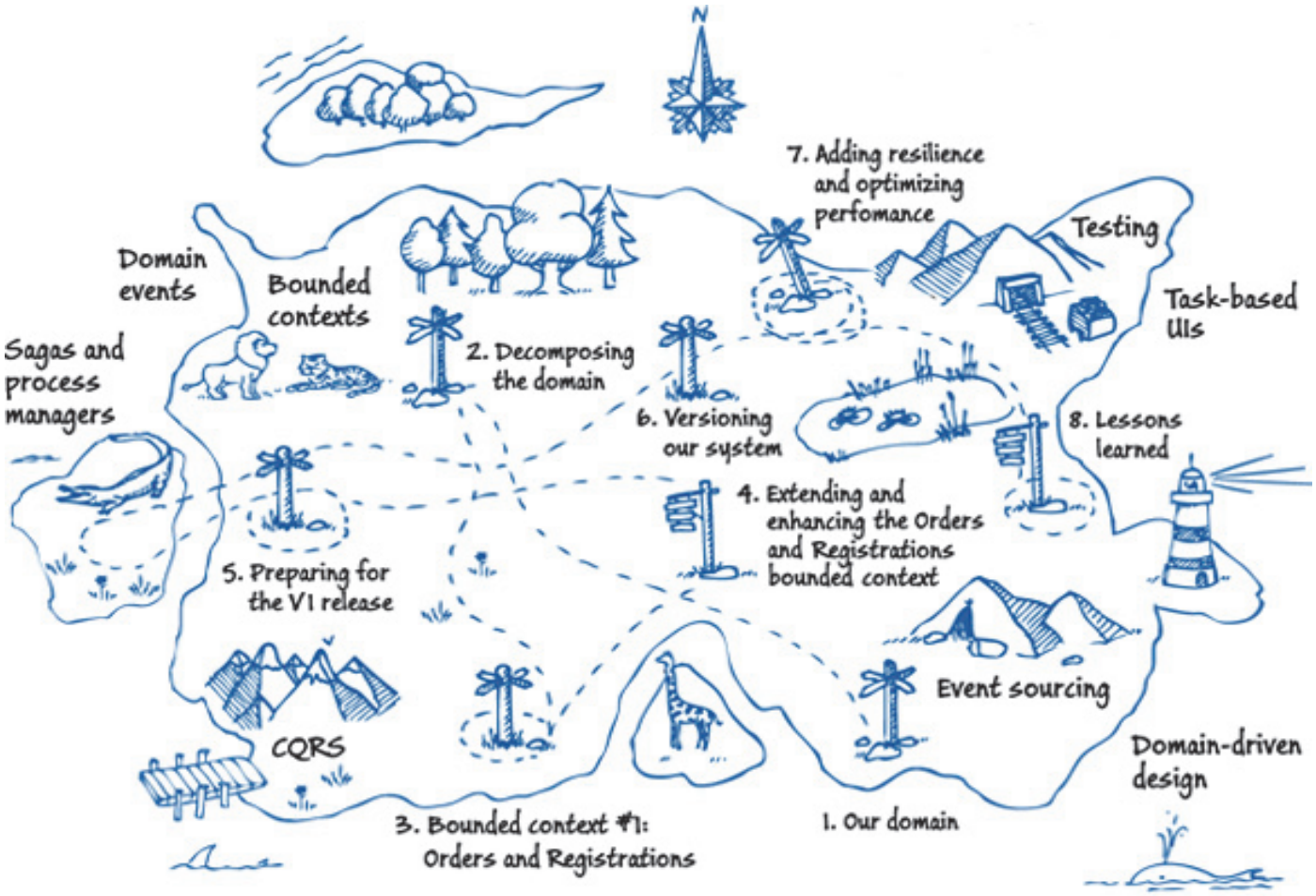
O CQRS não é apenas uma lista de benefícios. Diria que o seu maior desafio reside na complexidade inerente à sua implementação.

É também importante salientar que a sua implementação implica uma visão/abordagem diferente. Habitualmente estamos formatados para pensar em operações CRUD, mas com o CQRS somos levados a pensar em tarefas/comandos, o que tem implicações não só a nível das operações bem como no desenho da interface da aplicação.

Conclusão

CQRS não se aplica a todos os casos e não é uma receita que se deva seguir cegamente. No entanto é uma ferramenta que pode ser extremamente útil no momento de lidar com alguns problemas de elevada complexidade.

É importante relembrar que se deve começar simples e que aplicando o CQRS será sempre possível evoluir de modo a dar respostas a problemas mais complexos.



AUTOR



Escrito por Guilherme Ferreira

O Guilherme Ferreira é um apaixonado pelo desenvolvimento de produtos, que ao longo dos anos, e muitos cafés depois, especializou-se no desenvolvimento de soluções para a cloud, com especial foco nas tecnologias Microsoft. Podes saber mais sobre ele em http://gsferreira.com ou no seu twitter @gsferreira.

COMUNIDADES E VOLUNTARIADO PRESENCIAL

Com o aparecimento de páginas de internet de eventos comunitários como o Eventbrite (https://www.eventbrite.pt/) e Meetup (http://www.meetup.com/pt/) possibilitou um aumento de comunidades gratuitas ligadas a startup, informática, promoções e outros.

Sou assistente assíduo de várias comunidades gratuitas sem fins lucrativos como Iot Portugal, NetPonto, O365PT, PTJug, SQLPort, e outros. Também faço voluntariado em algumas comunidade como o SQL Sartuday Lisboa e Porto.

Já convidei várias pessoas a assistirem a estas comunidades porque não tem nenhum custo, é totalmente gratuito e a reação positiva é de 95%. As pessoas mostram interesse mas quando falo no horário que é normalmente em pós-laboral ou aos sábados de manhã a reação positiva muda para média de 25% e os motivos são vários “Não tenho tempo.”, “Não me vou levantar cedo só para ir a um evento.”, “Não tenho mais nada que fazer do que ir a eventos a essa hora.”. Quando falo em voluntariado, a reação positiva é quase nula. Já me disseram na cara que “devia ser pago só por estar a ajudar” e neste tipo de situações já não tento explicar o significado da palavra “voluntário” devido a discussões sobre este tema.

Foi exatamente por isto que resolvi escrever este artigo baseado apenas na minha experiência.

Muitas das pessoas com quem falei têm uma ideia errada das comunidades ou eventos nacionais ligados a informática, sejam eles do tema que forem. A primeira ideia é que estão perante uma apresentação de produtos em que levamos uma injeção da apresentação de um produto ou serviço e pode existir ou não um intervalo e por fim mais umas horas com outra injeção de como o produto ou serviço é melhor que a concorrência e devíamos comprar porque nos é oferecido uma promoção ou um brinde. Não os censuro porque eu também vou a este tipo de apresentações e quando pensei ir pela primeira vez a um evento de comunidade pensei que iria ser a mesma coisa. Mas realmente não o é.

Um evento de comunidade não é apenas para ouvir uma ou várias pessoas a falar sobre um tema ou para comer gratuitamente. É um local de autêntica aprendizagem. Quando conheci a SQLPort a comunidade Portuguesa de SQL Server pensei que não valia a pena perder o meu tempo e a deslocação até porque iria ouvir falar de temas que já tinha visto na Internet e que era uma perda de tempo: enganei-me a 100%. Fui ver duas sessões e senti-me um insignificante, ouvi falar de temas que já tinha ouvido mas não a um nível profissional e prático incluindo casos de situações reais críticas que aconteceram e o que devemos fazer para não nos acontecer a nós. Mas não foi apenas isso, conheci também profissionais portugueses que não são apenas administradores de bases de dados (DBA) encontrei programadores e gestores de informática ou seja, não é exclusivo aos DBA. O mesmo aconteceu na Netponto onde já tive uma conversa com um contabilista que tinha as suas próprias ideias em como melhorar o seu sistema contabilístico. Devido às comunidades conheço grandes profissionais fiquei com os seus contactos e se tiver alguma dúvida posso perguntar-lhes ou á própria comunidade.

Com a minha experiência de voluntariado no SQL Saturday em Lisboa e Porto conheci grandes profissionais estrangeiros sem ter que sair do país, tive longas conversas com eles e não foram apenas sobre tecnologia. Não foi uma perda de tempo, diverti-me, adorei e isto tem um valor incalculável e impagável, quer a nível profissional quer a nível pessoal.

Uma das coisas que são toleradas são as questões e dúvidas estúpidas, porque não há questões ou dúvidas estúpidas pode existir mais que uma pessoa com a mesma dúvida e é o local apropriado para as tirar e com isto perder alguma timidez.

As comunidades são também um local de propostas de emprego e de criação e startups e sim, por acaso já recebi uma proposta de emprego.

Como podem constatar as comunidades não são um ponto de encontro de vendas. São um local de aprendizagem, de conhecimento de novas pessoas, de troca de contactos profissionais, um autêntico local de socialização e divertimento e isso é referido no início de todas as apresentações da Netponto.

Não estou a dizer que as comunidades presenciais sejam melhores que as comunidades ou fóruns eletrónicos ou vice-versa. Os dois possibilitam uma enorme fonte de aprendizagem e convívio mas podemos fugir do mundo virtual para o mundo real. Um autêntico caso de sucesso no mundo eletrónico é a comunidade Portugal-a-Programar que não é apenas a maior comunidade portuguesa de programação mas é também uma revista eletrónica totalmente gratuita e sem fins lucrativos.

Quero alertar para que nem todos os eventos são gratuitos. Alguns eventos são pagos por isso antes de se inscrever confirme se é uma comunidade ou um evento pago.

Podem estar a perguntar porque que uma comunidade gratuita promove eventos pagos. Será para encher o bolso

deles à nossa custa? Não, o dinheiro angariado possibilita a comunidade de continuar a oferecer os seus eventos gratuitamente. Uma comunidade precisa de um local e às vezes de pausa para café quando consegue patrocinadores para ambas é fabuloso mas quando não conseguem um patrocinador para o local ou para pausa para café é a própria organização que tem de pagar e se fizermos as contas a um evento por mês anualmente sem esse patrocínio é um investimento que pode matar uma comunidade.

Fica aqui o meu convite para que experimentem ir a uma comunidade da vossa área ou da vossa curiosidade não se acanhem e divirtam-se! É totalmente gratuito não pagam nada para entrar nem é solicitado qualquer jóia de sócio. As comunidades precisam de nós não só para assistir mas também para fazer apresentações.

Algumas das comunidades fazem um sorteio de brindes no final do evento eu já ganhei autocolantes para portátil, t-shirts e licenças de software. E quem sabe podem ir para casa com um brinde?!?!

Boa sorte para todos os que querem experimentar.

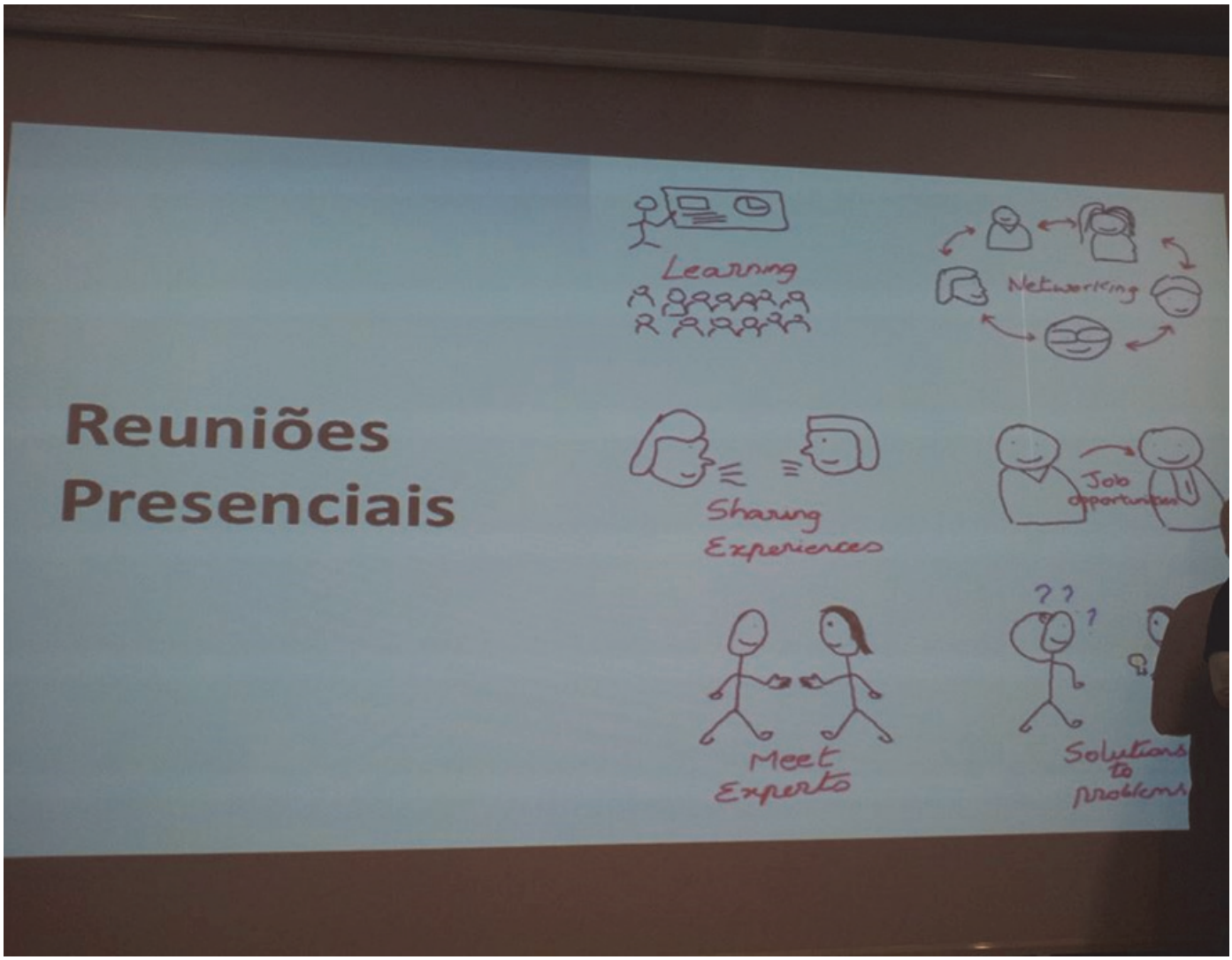


Ilustração 1 Reuniões presenciais da Netponto

AUTOR



Escrito por Ricardo Cabral

Licenciado em Engenharia Informática pela Universidade Autónoma de Lisboa e registado na Ordem dos Engenheiros Técnicos. Participante regular nas comunidades de Informática. O seu Twitter é @rramoscabral

SHIFT APPENS 2016 - COIMBRA



Dia 19, 20 e 21 de Fevereiro de 2016, Coimbra encheu-se de jovens (e alguns menos jovens) dispostos a escrever um novo mundo em código.

O Pavilhão Centro de Portugal, nas margens do Rio Mondego encheu-se de vida pra um dos mais conhecidos hackathons portugueses. Os participantes foram maioritariamente estudantes ligados à informática, contudo desta vez o evento abriu as portas também a quem já deixou de ser estudante. Organizado pelo Núcleo de Estudantes de Informática da Associação Académica de Coimbra e pela Jeknowledge, este evento superou as expectativas dos participantes.





A PROGRAMAR esteve no local e mesmo nas primeiras horas da manhã, após uma longa noite, já os participantes iam chegando sorridentes e cheios de ideias. Uma nota que vos queremos deixar é que com bom grado vimos que havia várias geek girls na sala, contrariamente ao habitual nestes eventos!

Géneros à parte, foram cerca de 110 participantes, divididos em 32 equipas, que se inspiraram e programaram como se não houvesse amanhã. Porque na verdade durante cerca de 60 horas não houve um amanhã. Porque ninguém se lembrou dele.

Apesar da organização providenciar o descanso merecido, muitos foram os que apesar de abandonarem o recinto por algumas horas, levaram as ideias na mente, desenvolvendo algoritmos.

Foram debatidas ideias, feitos novos contactos e acima de tudo, muita partilha de informação e conhecimento.

Durante o tempo em que o hackathon decorreu tiveram lugar várias palestras para todos os que quisessem participar.

A zona de descanso e de descontração também estiveram presentes e claro, como não podia deixar de ser, a organização também não deixou ninguém de barriga vazia.





Queremos parabenizar toda a organização e deixar um agradecimento pelo convite, sendo que agradecemos em especial à Adriana Costa e ao Tiago Botelho que gentilmente nos receberam apesar das poucas horas de descanso!

Que venha o próximo SHIFT APPENS!





AUTOR



Escrito por Rita Peres

Natural de Castelo Branco, licenciou-se em Engenharia Informática pela Universidade da Beira Interior. Membro do P@P desde Janeiro de 2010.

RASPBERRY PI 3 MODEL B

Definitivamente o Raspberry Pi veio para ficar. Na última edição da Programar falamos de um novo membro da família, o Raspberry Pi Zero. Passados 3 meses, estamos a falar-vos de mais um novo membro. O Raspberry Pi 3.

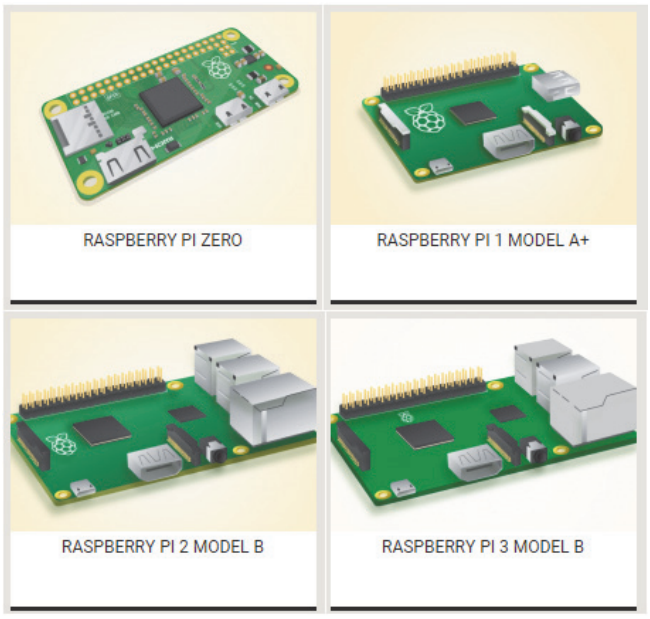


Ilustração 1 – Alguns membros da Família Raspberry

A fundação que ultimamente nos tem habituado a ter novidades em curtos espaços de tempo, voltou a "surpreender" os seguidores mais fieis.

Se o Pi zero nos fazia voltar à simplicidade do circuito, o Pi 3 assume-se como o mais bem "preparado" da família, superando o seu antecessor o Pi 2, como alias, seria esperado.

Lançado em Fevereiro de 2016, a nível de forma, o Pi 3 é idêntico aos modelos anteriores e, como seria esperado, tem total compatibilidade com Raspberry Pi 1 e 2.

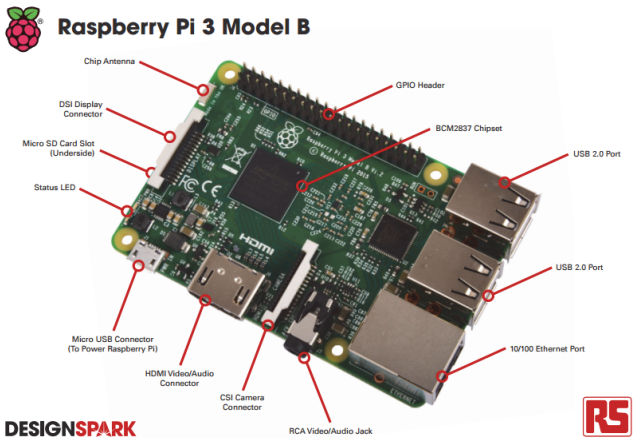
Ora, em termos práticos, o que é que isto significa?

Continuamos a ter as seguintes características:

- 4 portas USB
- 40 pinos GPIO
- Porta HDMI
- Porta Ethernet
- Jack de 3,5 mm de áudio e vídeo composto
- Interface da câmera (CSI)
- Interface de exibição (DSI)
- Slot para cartão MicroSD (*agora em versão push-pull em vez de push-push*)
- Núcleo gráfico VideoCore IV 3D

Então, pergunta o leitor, o que ganhámos nesta nova versão?

- CPU 1.2GHz 64-bit quad-core ARMv8 (o *Raspberry Pi 2 tem CPU 900MHz quad-core ARM Cortex-A7*)
- 802.11n Wireless LAN
- Bluetooth 4.1
- BLE - Bluetooth Low Energy (*isto é, em comparação com o Bluetooth clássico, o Bluetooth Smart destina-se a fornecer um consumo de energia consideravelmente reduzido e de baixo custo, mesmo mantendo um alcance de comunicação similar*).



O pequeno equipamento, ganhou velocidade e "comunicação". O controlador Radio Wireless nesta versão merece o destaque uma vez que suporta Wifi 802.11n e Blue-Tooth Low Energy (BTLE) Broadcom BCM43438. Tenho que admitir que estes dois "upgrades" do Pi me encantaram, isto porque já estava habituada a ter o adaptador de wifi a ocupar uma das portas USB. Com a inclusão deste módulo, o Pi passou a ter mais uma porta livre. Até porque a interface wifi built-in, permite aos mais "entusiastas" criar facilmente um

router baseado em Raspberry, fazendo o routing entre as duas portas (ethernet e wifi). (*Mais um ponto a favor desta versão da fundação Raspberry.*)

O ganho de desempenho é substancial face às versões anteriores, e a GPIO foi melhorada. A mesma, continua a suportar os protocolos das versões anteriores, mas tem melhor suporte e mais pinos.

Acerca da fonte de alimentação desta versão, esta apresenta-se-nos como uma alimentação de 2,5 A/5v (anteriormente tínhamos uma fonte de alimentação 1,8A/5v). Esta nova versão do Raspberry tem ainda uma antena de chip, onde anteriormente estavam os LEDs de estado (*que se note que os LEDs continuam presentes nesta versão mas ao lado da slot microSD*).

Um outro ponto que quero deixar ao leitor, e que pode ser visto na página oficial do fabricante - https://www.element14.com/community/community/raspberry-pi/blog/2016/02/29/the-most-comprehensive-raspberry-pi-comparison-benchmark-ever é um dos testes que foram efectuados à board, o *MemTester*.

Para os que não estão familiarizados com este teste, a própria página da element14 explica-nos que este se destina principalmente a diagnosticar/testar a memória RAM (Random Access Memory).

Para que as conclusões sejam mais claras e mais "justas" para todas as versões do pequeno Pi, o teste foi efectuado com um limite de 256MB. Isto porque o Pi, tem memória RAM partilhada com o processador de vídeo.

Assim, usando um comando Linux simples pode facilmente verificar-se quanto tempo demorou a testar 256MB de RAM em cada Raspberry Pi.

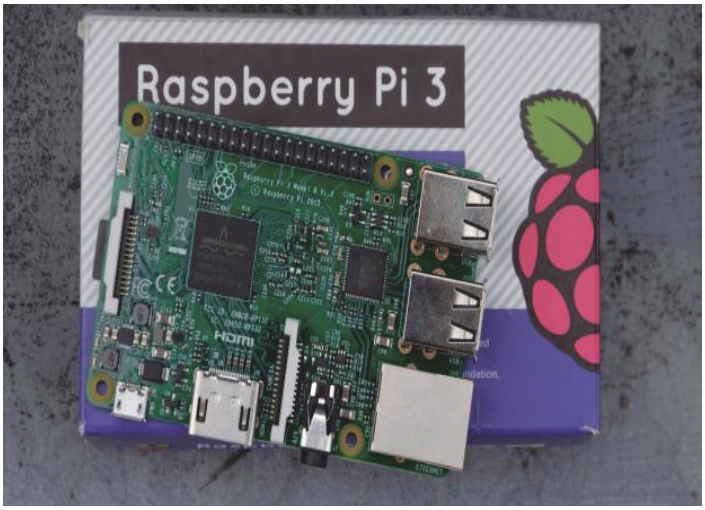
```
sudo time memtester 256M 1
```

Raspberry Pi Board	Time Taken
1 Model B+	76 minutes 23.296 seconds
2 Model B	23 minutes 39.07 seconds
3 Model B	8 minutes 37.078 seconds

Como nos diz o fornecedor, os números falam por si, e embora a velocidade RAM seja quase a mesma, isto é, pelo menos, 400Mhz e evidentemente, o limite é em grande parte devido ao processador, isto significa então que temos um aumento de velocidade significativo. Neste caso, a PI 3 é 50 % mais rápido do que o Pi.

Para os mais curiosos, deixamos ainda uma outra nota… o overclock desta versão permite "puxar" a velocidade até 1.45 GHz e a memória dos 400mhz para os 500mhz sem perdas de estabilidade.

Em suma, o novo PI 3 tem maior poder de processamento e conectividade, o que o torna atractivo ao utilizador, quer seja o utilizador "comum", quer seja programador, uma vez que nos vai permitir poupar tempo e dores de cabeça no desenvolvimento de possíveis aplicações, uma que que também nos permite "ligar-nos" mais rapidamente sem que seja necessário mais algum dispositivo externo. Este Raspberry trata-se de uma máquina com a qual podemos facilmente executar testes de segurança e análise de dados, pois tem um consumo de energia bastante baixo. Saliento ainda o facto de correr o Windows 10 IoT Core, sem dúvida mais uma mais-valia com que podemos contar (*para os mais distraídos, o Windows 10 IoT Core é uma versão do Windows 10 desenvolvida especificamente para uso em dispositivos de baixo custo e dispositivos da Internet das Coisas. É uma nova versão dos sistemas antigos da Microsoft chamados de Windows Embedded*). Assim, sendo este um equipamento de baixo custo, torna-se um aliado de peso, por exemplo, para projectos IoT (Internet das coisas).



AUTOR



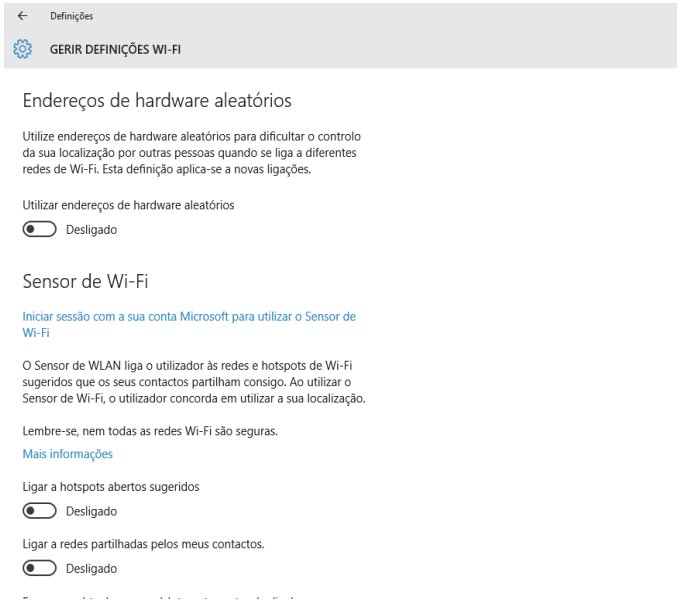
Escrito por Rita Peres

Natural de Castelo Branco, licenciou-se em Engenharia Informática pela Universidade da Beira Interior. Membro do P@P desde Janeiro de 2010.

WINDOWS 10: GESTÃO DE REDES WIRELESS ATRAVÉS DA LINHA DE COMANDOS

Introdução

No **Windows 10**, os utilizadores podem fazer a **gestão** e **configuração** das **redes Wireless** a partir de dois locais distintos: **Painel de Controlo** e **App Definições**. Aqui podemos fazer configurações básicas, como alterar definições da placa, ativar endereços aleatório de hardware, esquecer redes Wireless às quais já não fazemos ligação, etc.



Contudo, existem inúmeras tarefas que estas aplicações não permitem executar, como por exemplo obter informações detalhadas sobre a **placa de rede** e respetivos **drivers**, informações sobre os diversos **perfis** de **redes** armazenados no **PC** e respetivas **passwords**, gerar **relatórios** que permitem despistar **problemas** de **conectividade**, entre outros.

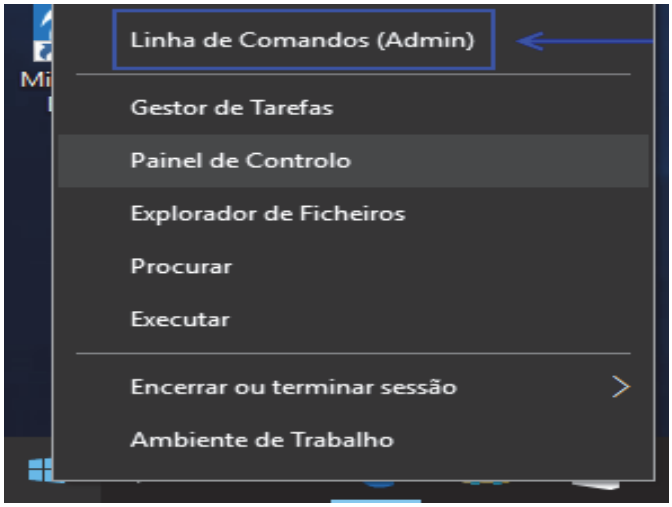
Para os casos referidos anteriormente, normalmente recorremos a um **utilitário** da **Linha de Comandos** disponível desde o **Windows 2000** e que se chama **Netsh**. O **Netsh** é utilizado para **gerir redes com** e **sem fios** e a através de um conjunto de parâmetros, podemos configurar, obter informações e resolver problemas de conectividade em praticamente todos os tipos de adaptadores.

Vejamos então em detalhe, quais os comandos **Netsh WLAN** disponíveis e que nos permitem executar algumas das tarefas referidas anteriormente.

Aceder à Linha de Comandos no Windows 10

Antes de começar a **analisar** cada um dos **comandos**, é necessário **abrir** uma janela da **Linha de Comandos** com **privilégios** de **administrador**.

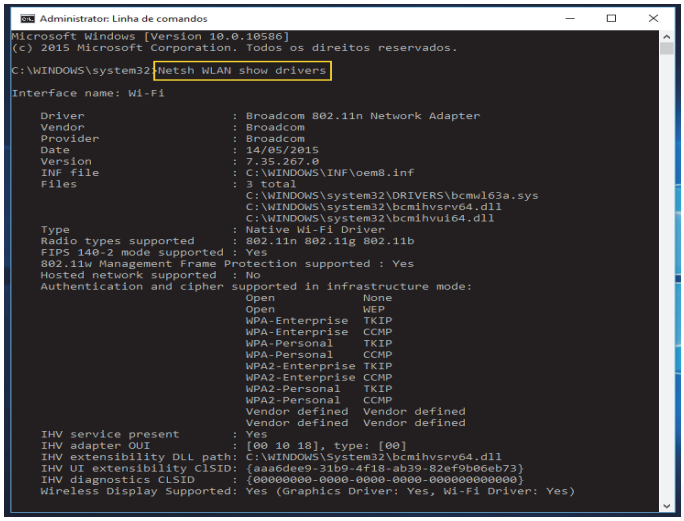
Uma das formas mais rápidas de o fazer no **Windows 10**, é através da combinação de teclas **WIN + X** abrir o **Menu Power User** e em seguida, **selecionar** a opção **Linha de Comandos (admin)**.



Informações sobre os drivers dos adaptadores

Para obter informações pormenorizadas sobres os drivers da placa de rede Wireless, vamos escrever o seguinte comando: Netsh WLAN show drivers

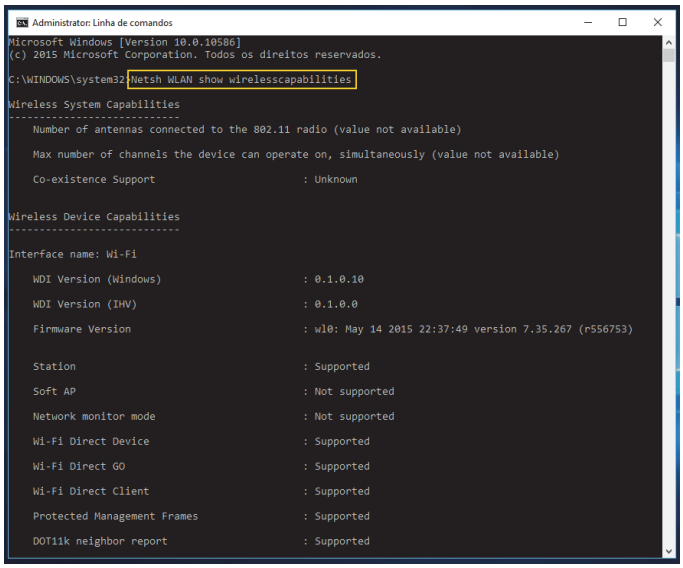
Este comando mostra as principais informações sobre os **drivers** instalados no **PC**, e que normalmente incluem o **nome** do **driver** e **fabricante**, **data** e **versão**, **tipo** e **rádios** suportados, entre outras.



Informações sobre as capacidades do adaptador Wireless

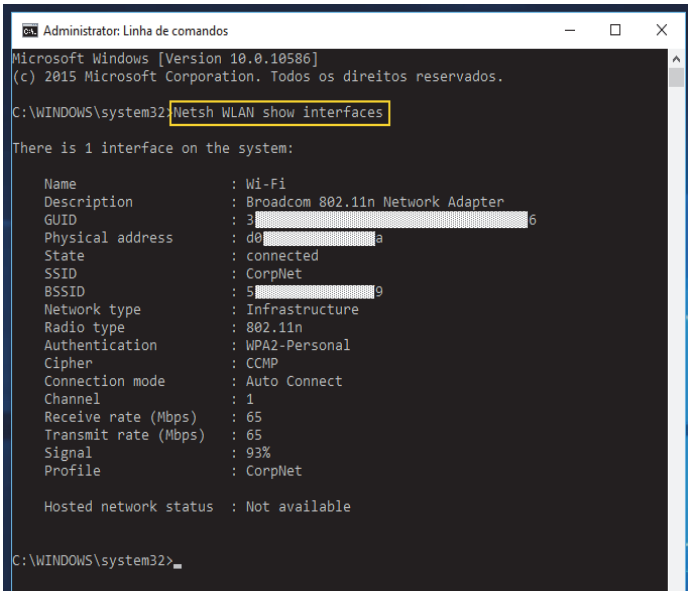
Para além das informações gerais sobre os **drivers**, é possível obter informações sobre outras **capacidades suportadas** pela **placa de rede**.

Para isso utilizamos o comando: Netsh WLAN show wirelesscapabilities

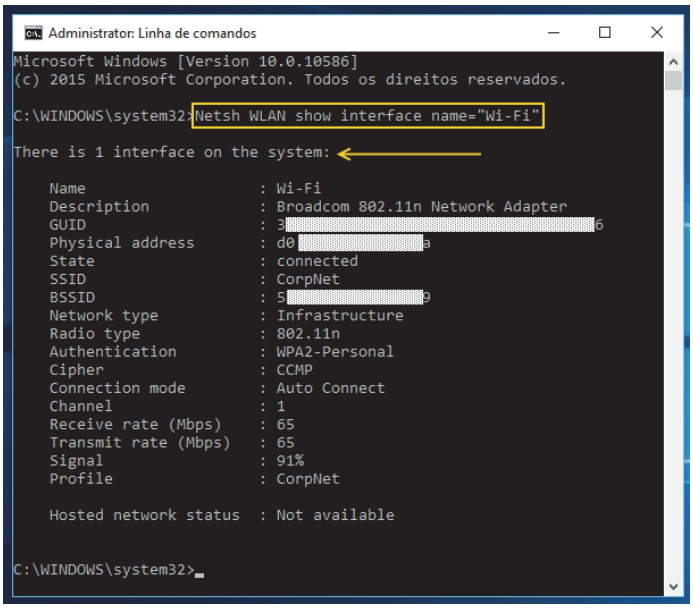


Informações sobre as definições do adaptador

Para obtermos informações especificas de **todas** as **placas de rede** instaladas no **PC**, como por exemplo **Nome**, **Tipo de Rede**, **Tipo de Radio**, **SSID**, **Autenticação**, etc., utilizamos o seguinte comando: Netsh WLAN show interfaces



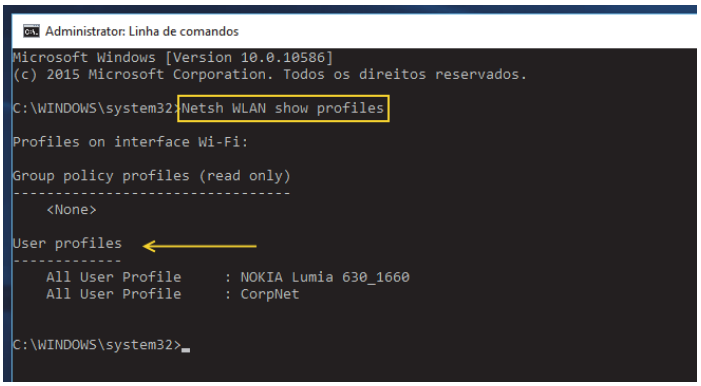
Para obter informações sobre **apenas uma** das **placas de rede** instaladas, podemos usar o seguinte comando: Netsh WLAN show interface name="nome do interface"



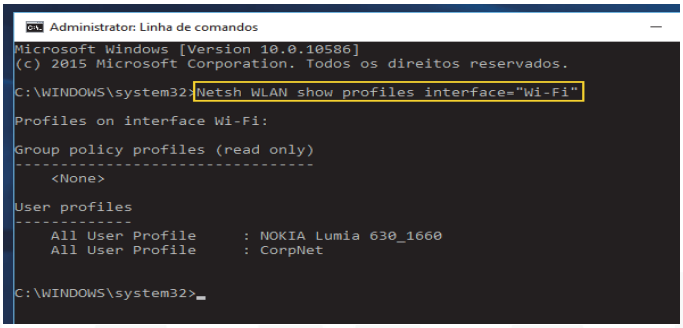
Informações sobre os perfis de redes wireless armazenadas no PC

Quando nos ligamos a determinado **Access Point Wireless**, o **Windows 10** cria automaticamente um novo **perfil** de **rede wireless** que guarda **localmente** no **PC**.

Para **visualizar** os **perfis** que estão associados a determinada **placa de rede** e quais os utilizadores com permissões para fazer ligações através destes, utilizamos o comando: Netsh WLAN show profiles.



Tal como acontece com os **interfaces**, podemos obter estas informações para apenas uma placa de rede. Para isso, utilizamos o comando Netsh WLAN show profiles interface="nome do interface"

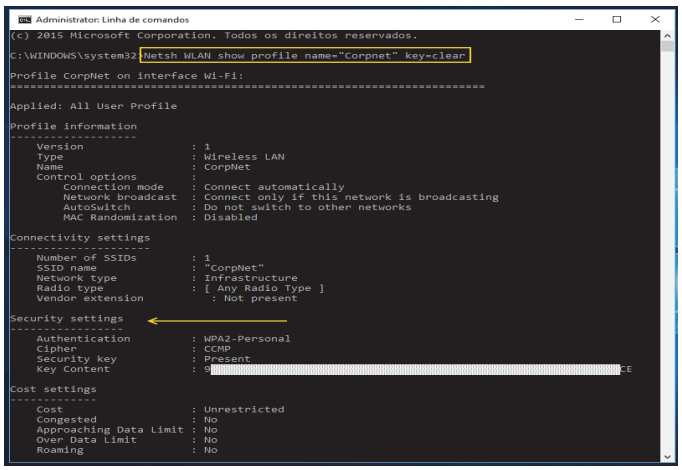


Visualizar as passwords dos perfis das redes wireless

No **Windows 10**, acedendo às propriedades da **placa de rede** no **Painel de Controlo**, conseguimos visualizar a **password** da rede wireless à qual estamos ligados, contudo, existindo mais do que um **perfil** guardado, não conseguimos visualizar as passwords definidas nos restantes perfis.

Para **visualizar** as **passwords** de todos os **perfis** guardados no **PC**, utilizamos o comando: Netsh WLAN show profile name="nome do perfil" key=clear

O parâmetro key=clear, vai mostrar a password associada ao perfil em questão no formato de **texto**.

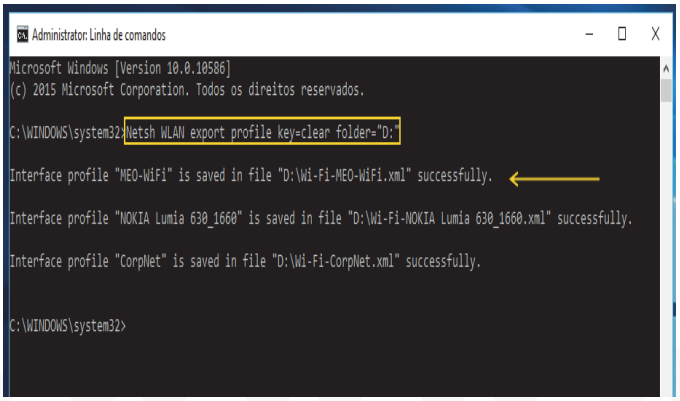


Importar/Exportar perfis de redes wireless

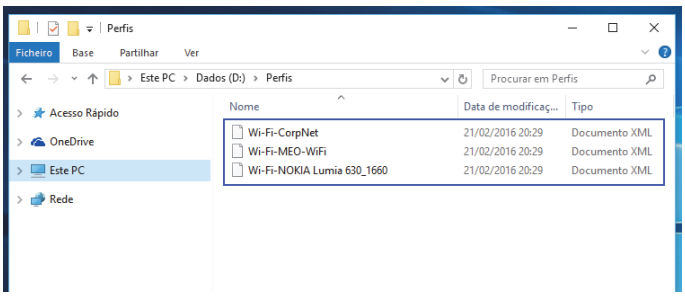
No **Windows 10** e também **Windows 8.x**, a utilização da **Microsoft Account** vem permitir a **sincronização** de determinadas **definições** entre os vários **dispositivos** que usamos, nomeadamente as **definições wireless**. Com este novo cenário, a funcionalidade no **Painel de Controlo** que permitia **exportar/importar** perfis de redes wireless e que ainda se encontra disponível no **Windows 7**, foi descontinuada.

Ainda assim e caso seja necessário, os utilizadores através do comando **Netsh WLAN** podem continuar a **exportar/importar** estes **perfis**.

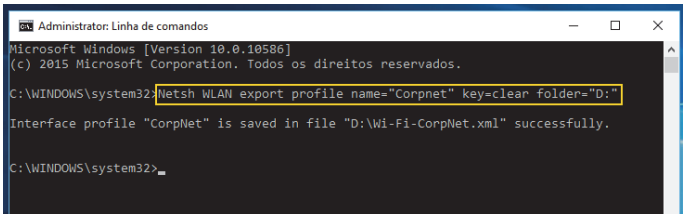
Para isso, utilizamos o seguinte comando: Netsh WLAN export profile key=clear folder="caminho da pasta"



Este comando, cria um ficheiro **.xml** por cada **perfil** guardado no **PC** e contém todas as informações relacionadas com os mesmos, incluindo as **passwords**. Caso não queiramos que esta informação inclua as **passwords** por uma questão de segurança, devemos **excluir** o **parâmetro** key=clear.



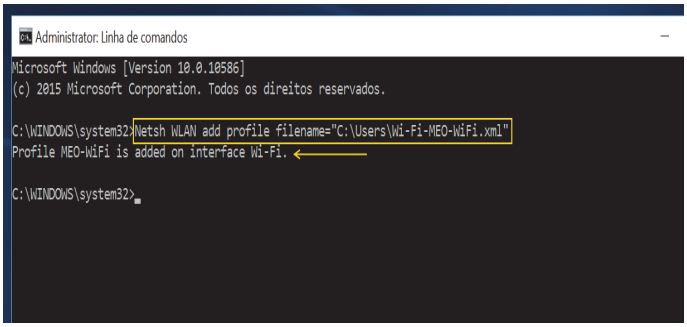
No caso de queremos exportar apenas **um** perfil, vamos acrescentar ao comando anterior o parâmetro name="nome perfil": Netsh WLAN export profile name="nome perfil" key=clear folder="caminho da pasta" .



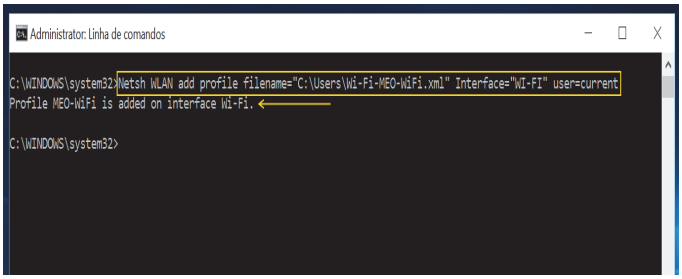
Quando for necessário fazer a importação destes perfis noutro dispositivo, podemos considerar **2 cenários**:

- Importar o **ficheiro .xml** dos vários perfis para que fiquem disponíveis em todos os interfaces e para todos os utilizadores.
- Importar o **ficheiro .xml** dos vários perfis para determinado perfil e apenas para o utilizador com sessão iniciada.

Assim, usamos para o primeiro cenário o comando: Netsh WLAN add profile filename="caminho_ficheiro.XML"



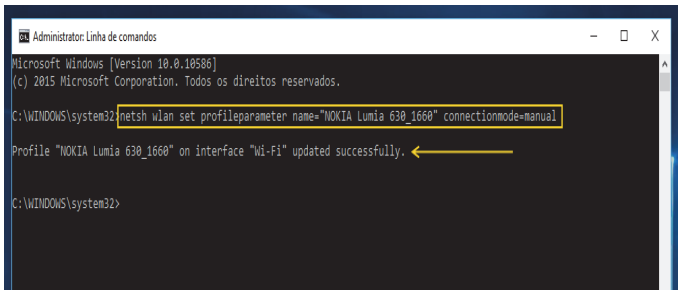
Para o segundo cenário, utilizamos o comando: Netsh WLAN add profile filename=" caminho_ficheiro.xml" Interface="nome do interface" user=current



Desativar a ligação automática a redes wireless indisponíveis

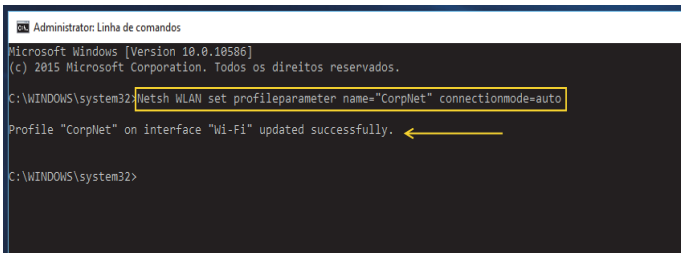
Quando configuramos no **PC** ligações a **redes wireless** que usamos com frequência, normalmente optamos por definir que as próximas ligações sejam feitas de forma **automática**. Contudo, se uma destas redes oferece uma qualidade de ligação inferior ou simplesmente deixamos de a usar com frequência, podemos definir novamente que a ligação seja feita de forma manual sem ter que remover o perfil.

Para isso, vamos usar o comando: Netsh WLAN set profileparameter name="Profile_Name" connectionmode=manual



Nos perfis cuja ligação está definida como automática, podemos ainda indicar qual o **perfil prioritário**, ou seja, qual é o que estabelece uma ligação em primeiro ligar.

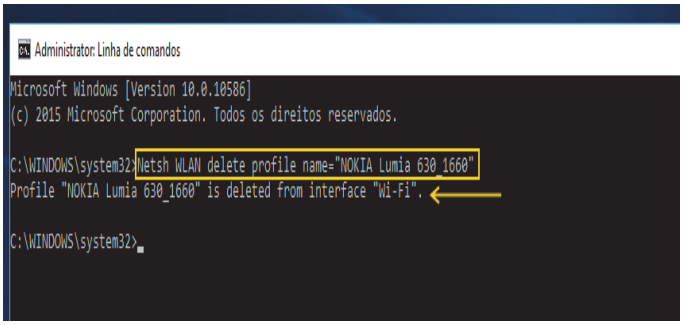
Para tal utilizamos o comando: Netsh WLAN set profileparameter name=" Profile_Name" connectionmode=auto



Eliminar perfis de redes wireless

Ao fim de algum tempo a fazer ligações a diversas **redes wireless**, vamos acumulando perfis que usamos apenas uma ou duas vezes e que não pensamos voltar a utilizar. Nestes casos, podemos apagar permanente estes perfis do PC. Em termos de sincronização e ao contrario do que acontece quando adicionamos novos perfis, quando eliminamos um perfil de rede num **PC** com **Windows 10**, o mesmo não é eliminado nos restantes dispositivos que pos-suímos e onde usamos a mesma **MSA**.

Para **eliminar** um perfil, vamos utilizar o seguinte comando: Netsh WLAN delete profile name="nome perfil"

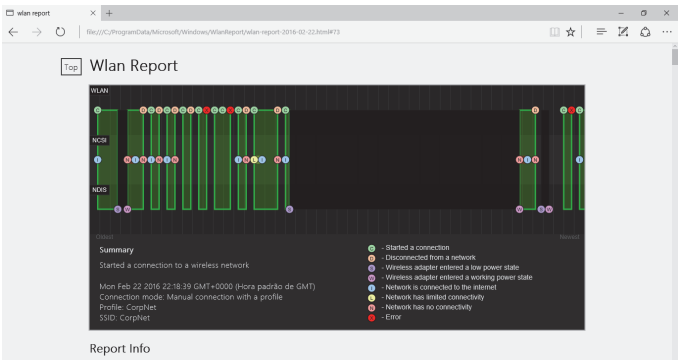


Criar relatórios sobre o adaptador e ligações

Quando estamos perante problemas de conectividade, nem sempre é fácil fazer a despistagem e avançar para a resolução. Para nos ajudar nesta tarefa, o **Windows 10** disponibiliza uma ferramenta que permite gerar um relatório da rede wireless. Este relatório detalhado, apresenta em forma de gráficos inúmeras informações que nos podem ajudar a identificar eventuais problemas.

Para **gerar** um relatório, vamos utilizar o seguinte comando: Netsh WLAN show WLANreport

O relatório é então gerado no formato **.HTML** e guardado na pasta C:\ProgramData\Microsoft\Windows\WlanReport\wlan-report-latest.html



Neste relatório vamos encontrar **informações** gerais do **sistema** e do **utilizador,** informações sobre os **adaptadores** disponíveis, **sumário** de **ligações sucedidas/falhadas**, **duração** das **sessões**, **eventos**, **erros**, entre outras.

Recursos

Para obter informações sobre outros comandos disponíveis, os utilizadores podem escrever na janela da linha de comandos **"Netsh WLAN"**.

Para saber mais sobre o **comando Netsh** e a sua utilização noutros cenários, como por exem-

plo **LAN**, **WINSOCK**, **IPv4** e **IPv6**, entre outros, poderá consultar a **lista** disponível no **site Microsoft TechNet** e acessível através do link: Netsh Command Reference

Para obter ajuda na **resolução de problemas** de **conectividade** no **Windows 10**, poderá consultar a informação disponível no **site do Windows** e acessível através do link: Corrigir problemas de ligação de rede.

Conclusão

Em conclusão, apesar das configurações que os utilizadores podem fazer através do **Painel de Controlo** e **App Definições**, o comando **Netsh WLAN** dá-nos outras opções sem as quais não era possível executar tarefas igualmente importantes na **gestão** das **redes wireless,** como por exemplo, a recuperação das passwords ou a exportação/importação de perfis.

A possibilidade de criar um relatório detalhado sobre o estado das ligações wireless, é uma mais valia no que diz respeito ao despiste e resolução de problemas.



AUTOR

Escrito por Nuno Silva

Microsoft MVP Windows Experience | Microsoft Technical Beta Tester

PROJECTO EM DESTAQUE NA COMUNIDADE P@P: MATRAQUILHOS

Nesta edição o projecto em destaque vai para o jogo Matraquilhos, para Android. É um jogo simples, baseado no tradicional jogo de matraquilhos, tão popular entre os mais jovens.

Este jogo possui diversas funcionalidades interessantes, além da simples "partida de matraquilhos", permitindo participar em campeonatos, escolher equipes e tácticas, participar em competições, desbloquear conteúdos e partilhá-los com os amigos, entre outras curiosidades.

Um jogo gratuito que faz jus à cultura do desenvolvimento de jogos, em Portugal, mostrando que não é necessário ter uma grande equipe para desenvolver um bom jogo! Basta dedicação.

De entre as tácticas disponíveis no jogo destacamos o 4-3-3, tão conhecido dos adeptos de futebol! Este esquema foi popular no final da década de 1960 e início da década de 1970, tendo sido usado pela Holanda na Copa de 1974. Dispõe também de outras táticas como 4-4-2, 3-4-3, 5-4-1, entre outras das mais comuns num jogo de futebol.

O jogo permite também escolher o tipo de campo, sendo que este determina o comportamento da bola durante o jogo.

Os controlos do jogo são simples, podendo ser usado o acelerómetro do dispositivo, para movimentar os jogadores, inclinando o dispositivo para um lado e para o outro. O jogo permite também jogar utilizando os botões no ecrã, caso se prefira esta forma de jogar. No entanto, caso se tenha um teclado ligado ao dispositivo, pode-se jogar usando o teclado.

Durante o jogo, para a defesa, carregamos no guarda redes e no botão para defender ou atacar quando a bola estiver próxima.

Para instalação desta aplicação é necessária uma versão Android 4.0 ou superior. Ao proceder à instalação damos permissão a esta aplicação para ler, alterar ou eliminar o conteúdo da memória USB em fotos, multimédia, ficheiros e armazenamento, bem como ver ligações Wi-Fi e acesso total à rede.

No que diz respeito à Informação de Jogos Pan Europeia (PEGI), está classificado no nível 3, em que o seu conteúdo é considerado apropriado para todos os grupos etários.

Contando já com um número de instalações compreendido entre 100 e 500, tendo recebido já vários comentários de vários utilizadores que classificam este jogo de uma forma muito positiva, estando já num rating de 4,9 estrelas.

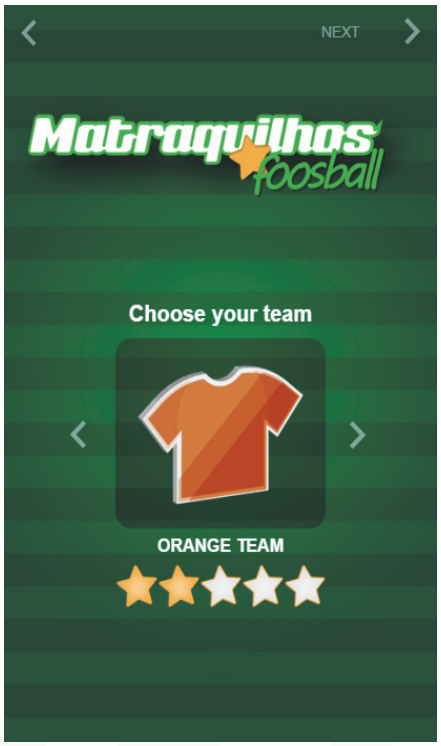
Simples e ao mesmo tempo viciante, o jogo cativa pela simplicidade, jogabilidade e grafismo, simplista.

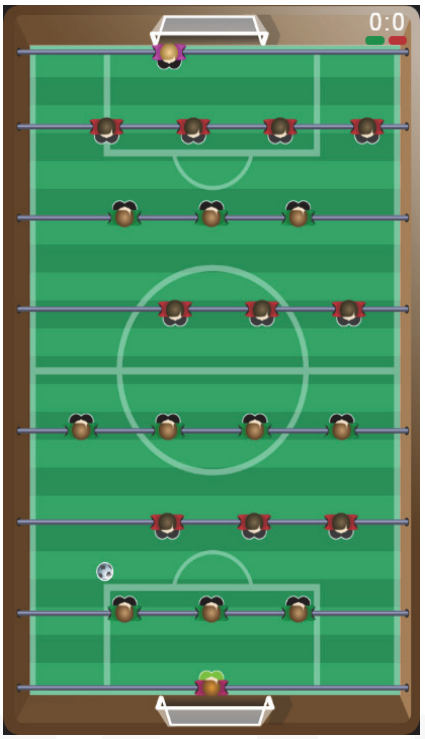
Testamos o jogo num dispositivo móvel e numa set-top box com Android, onde a jogabilidade com o teclado foi excelente e o grafismo acompanhou bem, tal como era espectável.

Para os adeptos de matraquilhos e/ou de futebol, um excelente jogo a não perder!

Parabéns ao programador Bruno Cardoso Aleixo!







Elege o melhor artigo desta edição

Revista PROGRAMAR

http://bit.do/ProgramarED52_V

Os 10 Anos da PROGRAMAR vêm aí!

Participa já com o teu artigo!





Veja também as edições anteriores da Revista PROGRAMAR

51ª Edição - Dezembro 2015



50ª Edição - Setembro 2015



49ª Edição - Junho 2015



48ª Edição - Março 2015



47ª Edição - Dezembro 2014



46ª Edição - Setenmbro 2014



e muito mais em ...

www.revista-programar.info

DUVIDAS?

IDEIAS?

AJUDAS?

PROJECTOS?



portugal-a-programar.org

Esta obra foi licenciada com uma Licença Creative Commons - Atribuição - Uso Não-Comercial - Partilha nos Mesmos Termos 3.0 Não Adaptada.