PROGRAMAR

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

EDIÇÃO #50 - SETEMBRO 2015

ISSN 1647-0710

EDIÇÃO 50

O FUTURO DOS BACKENDS?

E AINDA, NINJECT, OPENSHIFT, BITS EM C, WINDOWS 10 IOT E OUTROS MAIS ARTIGOS

A PROGRAMAR

COMO USAR BASE DE DADOS SQLITE EM WINDOWS 10 UNIVERSAL APPS

WINDOWS HELLO A AUTENTICAÇÃO BIOMÉTRICA NO WINDOWS 10

OFFICE GRAPH A INTELIGÊNCIA DO OFFICE 365

RECONHECIMENTO DE VOZ COM JAVASCRIPT

COLUNAS

AS NOVIDADES DO C# 6 **C#**

EDITORIAL

EQUIPA PROGRAMAR

Coordenador
António Pedro Cunha Santos

Editor
António Pedro Cunha Santos

Design
Sérgio Alves
Twitter: @scorpion_blood

Ilustração
Sara Freixo

Redacção
André Vala
António Pedro Cunha Santos
António Pereira
João Pedro Martins
Nuno Caneco
Nuno Santos
Nuno Silva
Patrício Domingues
Paulo Morgado
Pedro Sarmento
Ricardo Cabral
Ricardo Castro
Rita Peres
Sara Silva
Tânia Valente

Staff
António Pedro Cunha Santos
Rita Peres
Rui Gonçalves
Sara Freixo
Tiago Sousa

Contacto
revistaprogramar@portugal-a-programar.org

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

ISSN
1 647-071 0

Conectando os pontos

Eis que chegamos quinquagésima edição da Revista PROGRAMAR. Cinquenta edições, de muito trabalho, muito esforço, muita dedicação e uma história que já se escreve ao longo de nove anos, mais de uma centena de autores, arrisco dizer milhares de litros de café, uma imensidão de linhas de código, de desafios, de esforços de problemas e soluções.

Até aqui, passaram cinquenta edições da revista, passaram nove anos, a tecnologia reinventou-se sucessivamente! Nestes nove anos, cinquenta edições apareceram dispositivos que revolucionaram a maneira como lemos, agora nos tão habituais tablets, que em 2009 viram a sua popularidade entrar num ritmo desenfreado e de certa forma massificaram a leitura em formato digital, num tamanho de ecrã mais "confortável". Foram criadas novas linguagens de programação, novas ferramentas, novos ide's, várias versões de sistemas operativos, acompanhamos os "pequenos" tornarem-se "grandes" como o já muitas vezes falado Raspberry Pi, que tem vindo a tornar-se mais popular.

Fugindo à tentação das frases feitas, esta edição foi propositadamente lançada no ducentésimo quinquagésimo sexto dia do ano, o dia do programador. Assim escolhido por como todos sabemos 255 ser 2^8, o maior número inteiro representável com 8 bits, oito pequenos zeros e uns, que são os "blocos de construção" de todo um mundo, escrito pedaço a pedaço, bit a bit, por todos quantos programam. Os que constroem o mundo de bits e bytes, que nos rodeia a todos e que por todos é construído, bit a bit, linha a linha, a "brincar às escondidas com o ponto e virgula (;), o mestre do "jogo do esconde", desde 1972", de tantas vezes que consome horas à procura da linha onde por algum acaso falhou ao digitar o ponto e vírgula cuja falta provoca um erro.

Nesta edição não poderia deixar de dar os parabéns a todos aqueles que lêem a revista, a todos aqueles que nela participam, mas também e com entusiasmo, a todos aqueles que programam! Esta é a quinquagésima edição, mas hoje, estamos todos de parabéns pois é dia do Programador! Daquele para quem todos nós que tornarmos esta revista realidade, nos esforçamos! A nós, pois de uma forma ou de outra todos somos programadores e aos leitores da revista! Programadores, futuros programadores, mestres e aprendizes, entusiastas e menos entusiasmados! A todos os que lêem a revista, e quando a lerem mesmo depois do 256 dia do ano, se vão rir ao ler este editorial e perceber que desta vez, não foi um atraso na publicação, não foi um "packet-loss", foi antes uma "brincadeira" de programador e uma forma "talvez original" de festejar o dia do programador, com todos aqueles que lêem a revista, que programam, que se dedicam a esta magia, que é PROGRAMAR.

Até à próxima, agradeço-vos a todos por lerem a revista, que esta pequena-grande equipe vos traz.

António Santos

ÍNDICE

TEMA DE CAPA

A PROGRAMAR

ELECTRÓNICA

COLUNAS

ANÁLISES

NO CODE

EVENTOS

TechDays Aveiro 17-18 Setembro 2015
7ª Reunião Presencial da Comunidade NetPonto no Porto @ 26 Setembro 2015
Lisbon Makers Fair 16-18 de Setembro 2015

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

O mundo da inteligência artificial debate-se em Coimbra

Entre os dias 8 e 11 de Setembro, no Departamento de Matemática da Universidade de Coimbra o Encontro Português de Inteligência Artificial (EPIA). Conta com a participação do biólogo e físico Ricard Solé, autor de "Vidas Sintéticas", obra que aborda um vasto conjunto de questões desencadeadasa pelo desenvolvimento no campo da I. A., nomeadamente biologia sintética e vida artificial. Também contará com as participações de **Helder Coelho**, um dos pais da inteligência artificial em Portugal, e **François Pachet**, diretor do Laboratório de Informática da SONY em Paris.

Este encontro reúne mais de uma centena de investigadores e académicos de todo o mundo.

Nestes quatro dias debater-se-á a **evolução e a consolidação do conhecimento na área da inteligência artificial**, bem como **as implicações geradas por essa evolução e os limites que se devem impor à aplicação da inteligência artificial**.

Em destaque estarão as novas tendências da investigação neste campo do conhecimento e a discussão de **áreas emergentes como Aprendizagem Profunda**, **uma das mais revolucionárias e promissoras técnicas para "dar mais inteligência" aos computadores**.

Segundo Francisco Baptista Pereira e Penousal Machado, investigadores do Centro de Informática e Sistemas da Universidade de Coimbra (CISUC) e responsáveis pela organização do encontro, «**apesar dos desafios que ainda tem pela frente, a distância entre a realidade e a ficção poderá ser ultrapassada a médio prazo**».

Os investigadores acreditam que «**no futuro, a inteligência artificial terá um papel muito importante na resolução de grandes problemas do mundo. A ciência tem permitido avanços muito significativos na construção de mundos artificiais em áreas muito distintas como, por exemplo, medicina, robótica, ambiente e inteligência de negócio.**»

O programa do Encontro Português de Inteligência Artificial está disponível em: http://epia2015.dei.uc.pt/program/

Fonte: «Press Release da UC remetido à redacção da Revista Programar».

Ferramenta desenvolvida na UC diminui os custos do processo de estampagem na indústria automóvel

Diogo Neto, do Centro de Engenharia Mecânica da Universidade de Coimbra, venceu o "Prémio Melhor Tese de Doutoramento em Mecânica Aplicada e Computacional de 2014" atribuído pela Associação Portuguesa de Mecânica Teórica, Aplicada e Computacional, com um estudo que **permite diminuir os custos do processo de estampagem na indústria automóvel.**

Este trabalho teve como resultado uma ferramenta computacional de apoio ao projecto de conformação de chapas metálicas, processo largamente utilizado na industria automóvel.

Esta solução optimiza esse processo, reduzindo o tempo e custos de produção.

Segundo Diogo Neto, **«Devido à crescente competitividade internacional, o projeto e conceção virtual deste tipo de processos recorre cada vez mais a ferramentas computacionais, onde as técnicas avançadas de modelação em computador substituem os dispendiosos testes experimentais. Deste modo, os modelos numéricos desenvolvidos e implementados num programa de elementos finitos (DD3IMP) permitem fazer a simulação completa deste processo tecnológico com grande precisão e rigor, permitindo acelerar a fase de projeto e consequentemente diminuir os seus custos**».

Com várias parcerias com a industria automóvel, indústria metalomecânica e empresas de software em mecânica computacional, entre outras, o CEMUC participa em várias redes de investigação nacionais e internacionais.

A «**transferência do conhecimento para a sociedade pauta a atividade científica do CEMUC e, por isso, o reconhecimento público do trabalho desenvolvido neste centro de investigação é a confirmação de que se encontra no bom caminho e um incentivo para continuar numa rota sustentada de criação e transferência de conhecimento, contribuindo deste modo para o cumprimento da missão da Universidade de Coimbra**», sublinha Luís Menezes, que em conjunto com Marta Oliveira, orientou a investigação.

Fonte: «Press Release da UC remetido à redacção da Revista Programar».

TEMA DE CAPA

Azure Logic Apps: o futuro dos backends?

Azure Logic Apps: o futuro dos backends?

Introdução

Em Março de 2015 a Microsoft anunciou o **Azure App Service**, uma evolução da oferta aplicacional Azure que funde dois serviços existentes - **Web Sites** e **Mobile Services** - com dois novos serviços: as **Logic Apps**, destinadas ao desenvolvimento rápido e visual de processos de negócio, e as **API Apps** que encapsulam funcionalidades autónomas reutilizáveis, consistindo ambos numa concretização tecnológica de uma arquitectura baseada em **Microserviços**.

O objectivo deste artigo é apresentar as Logic Apps, tanto em termos do seu contexto e arquitectura, como na utilização concreta e simbiose com as API Apps.

Contexto

O caminho que nos traz às **Logic Apps** começou muito antes de Março de 2015.

Com o surgimento da chamada **Web 2.0**, um termo popularizado em 2004, surgiu igualmente o conceito de ***mashup***. Baseado na utilização de padrões e tecnologias *web* (como HTTP, REST, XML, RSS/Atom e Ajax), um *mashup* é uma aplicação *web* que combina conteúdo de várias fontes e APIs para criar um novo serviço, com essa combinação a ser feita tipicamente num *web browser*. Exemplos clássicos deste tipo de aplicações envolvem enriquecimento de mapas geográficos como novas camadas de informação: fotos tiradas em cada local, localização de farmácias ou hotéis, local onde foram feitas as últimas compras em *sites* de comércio electrónico. Um cenário de *Presentation Integration*, portanto.

Um termo relacionado surgido pouco depois foi o de **Service-Oriented Architecture (SOA)**. Aplicável principalmente a cenários empresariais, é um padrão de arquitectura - ou filosofia de desenho de aplicações, para alguns - segundo o qual uma aplicação é montada pela composição de serviços autónomos que encapsulam lógica e que podem ser acedidos usando mecanismos e formatos padrão (o SOAP e o WSDL são normas intimamente ligadas à materialização de arquitecturas SOA). Uma das ideias geralmente associada a SOA é a de que os serviços têm granularidade elevada, ou seja, os serviços devem encapsular muita funcionalidade e não ser chamadas atómicas do tipo *Create-Read-Update-Delete*, dado que são operações pesadas (desde a serialização de/para XML ao transporte pela rede). Este tipo de arquitecturas tem como dois dos grandes atractivos a criação de aplicações *loosely-coupled* e em que existe um ganho pela reutilização de serviços existentes.

A noção de SOA tem evoluído tanto pelas lições da sua utilização em contextos empresariais - em que o sucesso das implementações nem sempre foi o melhor (vejam-se por exemplo as promessas não cumpridas de padrões como o BPEL ou o UDDI, e a complexidade de normas como WS-Trust ou WS-Federation), como com a própria evolução natural da Web e a adopção de formatos mais simples e abertos como REST ou JSON.

Nos últimos dois anos esta História tem mais um capítulo, a noção de **Microserviços**. Construindo sobre as ideias anteriores e popularizada mais recentemente por pessoas como Martin Fowler, as principais diferenças face ao SOA talvez sejam a granularidade de cada serviço, que agora se entendem frequentemente como devendo ser mais atómicos e simples, como a utilização de padrões "CRUD" sobre HTTP/REST. Os serviços continuam a ser independentes entre si (podendo ser desenvolvidos sobre qualquer tecnologia), e a dever estar organizados em torno de conceitos ou operações relevantes para o negócio.

Uma materialização desta abordagem de Microserviços no espaço de consumidor final é o IFTTT (*If this then that*, https://ifttt.com/). Este sistema, usado tipicamente via *apps* para telemóvel, permite a criação de pequenos *workflows* ("receitas"), em que caso se verifique uma determinada condição monitorizada pelo IFTTT, é despoletado um evento/acção de resposta. Alguns exemplos seriam: sempre que um determinado utilizador colocar uma foto nova para o Instagram, gravá-la para o OneDrive; sempre que receber um *email* marcado como Urgente, enviar um SMS para um determinado número, etc. O número de acções disponíveis é muito elevado, e cada uma delas pode ser vista como um microserviço.

No mundo Microsoft, e depois de uma História que inclui actores como o BizTalk Server, Azure Service Bus, Windows Workflow Foundation, e todo um foco em evoluir as tecnologias para abordagens *cloud*, foram pré-apresentados no Integrate 2014 em Redmond os **BizTalk Microservices**, componentes atómicos *cloud-based* utilizáveis para integração de sistemas, a que estava associado um motor de *Workflow* e uma ferramenta de modelação de processos *web -based*.

Finalmente, com o anúncio de Março de 2015 tornou-se finalmente público o quadro todo: as **API Apps** (apresentadas no artigo "*Criar uma API no Azure App Sevice*" de Guilherme Ferreira na revista PROGRAMAR nº 49 de Junho 2015) são a implementação da Microsoft do conceito de Microserviços, e as **Logic Apps** a forma de os orquestrar para a criação de aplicações de negócio.

Azure Logic Apps

Como referido, as Logic Apps são uma das componentes do Azure App Service, e o seu objectivo é permitir a automação de processos de negócio ou *workflows*, realizando uma modelação gráfica dos mesmos. Ao contrário de abordagens como o IFTTT, os destinatários principais são tanto o mercado empresarial/*start-ups* como utilizadores com *know-how* técnico ou programadores.

Os processos de negócio são modelados usando peças de uma paleta onde se encontram as API Apps já mencionadas, e que incluem conectores para sistemas como bases de dados ou ofertas SaaS variadas como o Salesforce ou Office 365, suportando desta forma cenários de integração de sistemas *Cloud-based*.

As componentes das Logic Apps são os seguintes:

Workflow – modelação gráfica dos vários passos de um processo de negócio, e respectivo motor de execução. Esta modelação é feita (por agora) exclusivamente em ambiente *web browser* no portal Azure, e internamente ao sistema os processos são representados como JSON.

Triggers – um *trigger* é um caso particular de uma acção, e permite desencadear a execução de uma Logic App. Alguns exemplos de *triggers* são: execução recorrente (ex: a cada 5 minutos), resposta a um evento (ex: aparece um registo novo numa determinada tabela SQL) ou a pedido (ex: invocação de uma Logic App via pedido HTTP, como se fosse um Web Service).

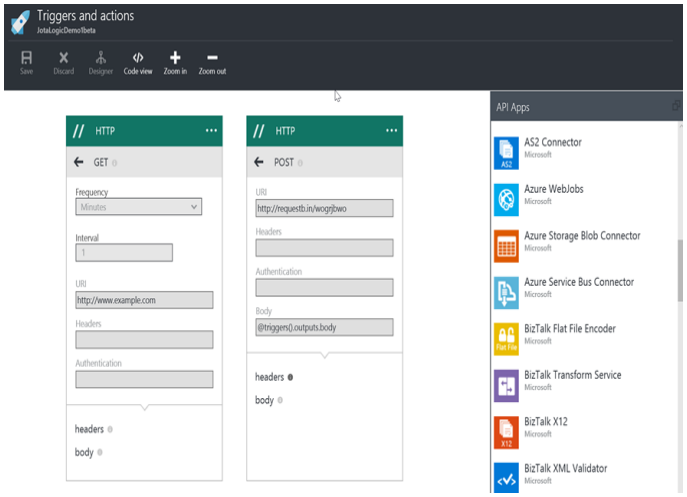
Actions – de uma forma genérica, uma Action é um passo do *workflow*, executada numa sequência depois de uma Logic App ser iniciada via *trigger*. As *actions* são essencialmente API Apps, sendo que se podem tanto usar as disponibilizadas pela Microsoft como simplesmente ser desenvolvidas à medida em Net/C#. De entre as primeiras (ver lista completa em https://azure.microsoft.com/en-us/documentation/articles/app-service-logic-connectors-list/), por vezes chamadas "*Connectors*" na documentação, existem dois tipos: as '**Standard'** que oferecem funcionalidades como integração com Facebook, Office 365, SQL, FTP, SharePoint/Yammer, Salesforce ou Twitter, e as '**Premium'** que se destinam a cenários de integração de sistemas e são provenientes da oferta de BizTalk Services. Estas últimas incluem potencialidades de processamento de EDI/AS2, EDIFACT, mapeamentos JSON-XML, validação de formatos de mensagem, ou integração com Informix, WebSphere MQ, Oracle ou SAP, entre outras. A utilização de *actions* Premium tem requisitos adicionais em termos de subscrição Azure e respectivos custos.

Internamente, as Logic Apps dependem de módulos já existentes em Azure como sejam as WebApps e o Resource Manager, que ficam fora do âmbito deste artigo.

Exemplo

Apresentam-se de seguida dois exemplos de Logic Apps criadas com *actions* disponibilizadas nativamente. Para exemplos passo-a-passo recomenda-se a consulta da documentação em https://azure.microsoft.com/en-us/documentation/services/app-service/logic/ .

A figura abaixo apresenta provavelmente o que é o exemplo mais simples possível de uma Logic App.



Esta Logic App é composta apenas por duas acções, ambas utilizações de uma API App que permite fazer pedidos HTTP.

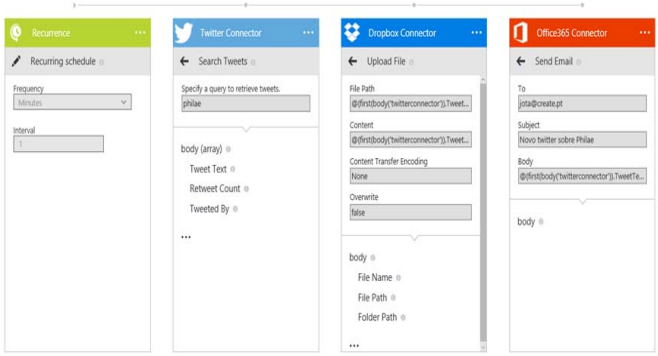
A primeira é uma acção de **HTTP/Get**, que a cada minuto faz um pedido HTTP a -um URL configurado (no caso, "http://www.example.com"), e obtém tanto o conteúdo HTML da página ("body") da resposta como os cabeçalhos HTTP ("headers"). A segunda acção é um **HTTP/Post**, em que se envia a informação obtida no **HTTP/Get** anterior como corpo do pedido para um serviço gratuito (http://requestb.in) que por sua vez permite consultar o que está a receber via HTTP Post.

De notar que a configuração de que o resultado da 1ª acção deve ser usado como *input* da 2ª é feito via uma *drop-down* que apresenta todos valores possíveis de utilizar, não sendo necessário regra geral a introdução de texto. Quando se faz esta escolha, a representação textual interna substitui a *dropdown* (no caso acima, "@triggers().output.body").

A imagem acima é um *screenshot* retirado do novo Portal de Azure (http://portal.azure.com), onde são efectuadas as operações realizadas tanto com Logic Apps como com API Apps. Além do *designer* visual na parte central, são visíveis as opções de topo do editor (onde se inclui uma opção "*Code View*" que permite consultar a especificação do processo em formato JSON passível de armazenamento em *source control*), como a paleta de API Apps do lado direito,

onde se vêm algumas das dezenas de opções já disponíveis.

A figura abaixo apresenta um exemplo um pouco mais complexo que o anterior.



No exemplo acima são usadas 4 acções:

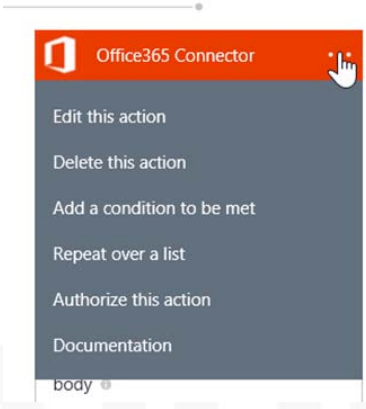
Recurrence – acção de *trigger* da Logic App, indica que a mesma vai ser executada a cada minuto.

Twitter Connector – procura no Twitter por algum texto (no caso, "Philae"). Quando se adiciona esta *action* é necessário realizar autenticação no Twitter, que fica guardada internamente à Logic App.

Dropbox Connector – *action* que pega no resultado da acção de Twitter e cria um ficheiro numa conta Dropbox com o conteúdo do *tweet* encontrado. Tal como no caso do HTTP/Post no exemplo anterior, esta acção tem uma dependência criada implicitamente pelo facto de usar dados de outra acção, sendo também necessário – como no caso da *action* de Twitter - fazer autenticação no Dropbox.

Office365 Connector – *action* que pega no resultado da acção de Twitter e envia um *email* para o endereço configurado, com o conteúdo do *tweet*. Mais uma vez, é necessário fazer autenticação no serviço (agora o Office 365) quando se usa esta acção.

Cada uma das acções anterior tem opções de refinamento que permitem especificar comportamento mais complexo, que são acessíveis através do botão de reticências no seu canto superior direito.



No caso anterior, é possível por exemplo indicar que esta acção apenas é executada caso se verifique uma determinada condição, ou que se pretende iterar a sua utilização sobre uma lista de itens de informação.

A figura abaixo apresenta um excerto da *Code View*, mostrando o JSON que representa a Logic App descrita acima. Neste JSON podemos ver uma secção de *triggers* que desencadeiam a execução, e uma sequência de *actions*, cada uma com o seu tipo e configuração específica. Como se pode reparar também, cada *action* tem um atributo *type* que no caso do *twitterconnector* é uma ApiApp.

```
"triggers": {
    "recurrence": {
        "recurrence": {
            "frequency": "Minute",
            "interval": 1
        },
        "type": "Recurrence"
    }
},
"actions": {
    "twitterconnector": {
        "type": "ApiApp",
        "inputs": {
            "apiVersion": "2015-01-14",
            "host": {
                "id": "/subscriptions/00000000-
      0000-0000-0000-1234567890ab/resourceGroups/
              DevTechRefreshLisboa2015/providers/
                    Microsoft.AppService/apiApps/
                                TwitterConnector",
                              "gateway": "https://
                             Devtechrefreshlisboa
            012345678909834b42198fd4b593463ced8b.
                                azurewebsites.net"
            },
            "operation": "SearchTweets",
            "parameters": {
                "Query": "philae",
                "MaxResults": 20
            },
            "authentication": {
                "type": "Raw",
                "scheme": "Zumo",
                "parameter": "@parameters('/
            subscriptions/00000000-0000-0000-0000-
                      1234567890ab/resourceGroups/
                DevTechRefreshLisboa2015/providers/
    Microsoft.AppService/apiApps/TwitterConnector/
                                          token')"
            }
        },
        "conditions": []
    },
    "dropboxconnector": {
        "type": "ApiApp",
        "inputs": {
```

Um dos pontos mais curiosos das Logic Apps é a forma como lidam com **controlo de fluxo**. Por omissão, todas as acções executam em paralelo umas com as outras, não existindo - como sucede nos fluxogramas ou *workflows* tradicionais – um controlo de fluxo explícito. Este fluxo é criado implicitamente com base em dependências de dados.

Assim, se a acção B depende da acção A, B vai ser

executado depois de A. Se não depende, vai ser executado em paralelo.

Na implementação actual, as formas de controlo de fluxo que existem são as 4 já referidas: a) recorrência; b) dependência de dados; c) execução condicionada a uma condição; d) repetição sobre lista. Uma outra *action* disponibilizada, o "Wait", pode igualmente ser utilizada neste contexto, mas o mecanismo como um todo é muito diferente daquilo a que estamos habituados, lembrando a forma de funcionamento altamente paralelizável de motores de regras (como o do BizTalk Server ou do Windows Workflow Foudation v2).

Conclusão

Neste artigo procurou-se dar uma introdução às Logic Apps, no seu contexto histórico e arquitectural, sem entrar em detalhe de implementação, e omitindo aspectos como monitoria no Portal Azure, custos, tratamento de erros, escalabilidade, bem como a integração profunda com as Web Apps e vários outros temas. Existem no entanto três aspectos que vale a pena referir como conclusão:

Primeiro, as Logic Apps são uma **oferta Empresarial/ Premium**, tendo origem no mesmo universo empresarial de onde vêm produtos *business-critical* como o BizTalk Server. São portanto de esperar potencialidades como escalabilidade, tolerância a falhas, tratamento de erros ou robustez, entre outros. Apesar de estarem disponíveis actualmente muitas *actions* orientadas para o consumidor final (como Facebook ou Twitter), sendo estas sempre usadas nas demonstrações públicas, o foco arquitectural é empresarial.

Segundo, As Logic Apps são materializações de arquitecturas baseada em **microserviços**, e dependem de API Apps, em que se podem tanto usar as disponibilizadas nativamente pela Microsoft, mas também desenvolver novas, sendo este um processo que se reveste de uma **enorme simplicidade** (uma API App é essencialmente uma WebApi alojada em Azure, com um contracto de utilização descrito em formato Swagger/JSON). Apesar de ainda não disponível completamente, está previsto um Marketplace de API Apps, onde ISVs poderão disponibilizar as suas próprias API's e monetizar a sua utilização, quer sejam usadas directamente quer via Logic Apps. Alguns exemplos poderão ser serviços de detecção de rostos, acesso a informação como bolsa ou meteorologia, motores de recomendações, entre muitos outros.

> ***(...)Azure App Service, uma evolução da oferta aplicacional Azure (...)***

Finalmente, deve salientar-se o facto de as Logic Apps ainda estarem em **Preview** e com evoluções frequentes (ver por exemplo as alterações mais recentes em http://azure.microsoft.com/en-us/updates/schedule-logic-apps-to-run-in-the-future/), existindo limitações como sejam a impossibilidade de criar *workflows* em Visual Studio, e de o processo de sincronização com repositórios de código como o TFS ser manual, entre outras. Ainda assim, as Logic Apps são **provavelmente a novidade mais interessante de entre as últimas que têm surgido no universo Microsoft Azure**, e vale a pena tanto perceber que cenários suportam, como acompanhar o rumo e evolução dos próximos meses.

AUTOR

Escrito Por João Pedro Martins

CTO e Arquitecto de Soluções na |create|it|. MVP de BizTalk Server entre 2006 e 2011. Focos tecnológicos na área de Arquitectura, Azure e Integração. Orador frequente em conferências e eventos em temas como Arquitectura de Projectos, dinâmicas de equipa, estimativas de software, e vários temas de Azure. Membro dos Azure Insiders.

Gosto em fazer bem, resolver problemas criativamente, e nunca parar de aprender.

A PROGRAMAR

Ninject – O Ninja das dependências

Como usar base de dados SQLite em Windows 10 Universal Apps

Como fazer o deploy de uma aplicação web com PrimeFaces no OpenShift

Manipulação ao nível do bit na Linguagem C

Reconhecimento de voz com JavaScript

Cria o teu cliente de 9GAG em 15 minutos, com OutSystems

Ninject – O Ninja das dependências

O que é o Ninject?

O Ninject é uma biblioteca de software aberto que providencia uma *framework* de injecção de dependências (*Dependency Injection* ou DI) leve, fácil de integrar e de utilizar.

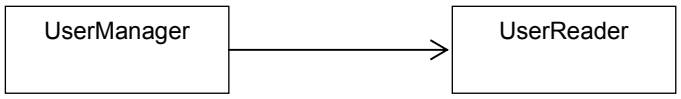
O padrão *Dependency Injection* é bastante utilizado em *software* empresarial porque, entre outras razões, permite manter o controlo do ciclo de vida dos objectos e facilita a implementação de testes automáticos.

O que devo saber antes de começar a utilizar o Ninject?

O Ninject pode ser utilizado em todos os tipos de aplicações desenvolvidas em .NET. No caso das *Windows Applications* ou *Console Applications* basta incluir a biblioteca no projeto e começar a utilizar.

Já se se pretender integrar o Ninject numa aplicação Web MVC ou numa biblioteca de serviços WCF, será necessário utilizar alguns plugins que permitam tratar da "canalização" necessária para garantir que as *factories* das *frameworks* MVC e WCF utilizam o Ninject como "fornecedor" de instâncias.

Vejamos os princípios: Suponhamos que temos uma classe **UserManager** que faz uso de uma classe **UserReader**.



Num cenário clássico, o código que representa este cenário seria parecido com:

```
class UserReader
{
    // User reader methods
}

class UserManager
{
    private UserReader userReader;
    public UserManager()
    {
        // A classe UserManager está a criar uma
        //instância da classe UserReader
        this.userReader = new UserReader();
    }
}
```

Esta implementação num projecto de larga escala levanta duas questões essenciais:

Se o construtor da classe **UserReader** for alterado, todas as classes que utilizam a classe **UserReader** terão que ser também alteradas. Tal pode levar a que os próprios construtores dessas classes seja também alterados, o que vai gerar uma onda de alterações em várias classes do projecto.

A proliferação de chamadas a construtores poderá escalar de tal forma que se perde o controlo de quantas instâncias da classes estão de facto a ser criadas e utilizadas pela aplicação

Ao utilizar uma *framework* de injecção de dependências, a gestão das instâncias das classes é efectuada pelo *container* que "injecta" a instância de uma classe sempre que está é necessária.

Neste cenário, teríamos uma implementação parecida com:

```
class UserReader
{
    // User reader methods
}

class UserManager
{
    private UserReader userReader;
    public UserManager(UserReader userReader)
    {
        // A classe UserManager recebe uma
        //instância da classe UserReader
        this.userReader = userReader;
    }
}
```

Neste caso, o *container* saberia construir uma instância da classe **UserReader** e injectá-la-ia no construtor da classe **UserManager**, tornando esta classe completamente agnóstica à forma como a classe **UserReader** foi criada.

NOTA: Seria também possível utilizar uma *factory class* para este efeito, o que consistiria uma alternativa à utilização de uma *framework* de *dependency injection* (no entanto menos poderosa).

Como começo?

O ponto central do Ninject é a interface **IKernel** cuja implementação por omissão é a classe **StandardKernel.** Esta classe implementa um *container* que armazena a configuração dos mapeamentos de interfaces em classes bem como as instâncias de alguns objectos disponibilizados pelo *container.*

Toda a interacção com o *container* é efectuada por código usando uma API fluente e bastante completa. Para

quem prefere usar XML, o plugin Ninject.Extensions.Xml permite efectuar a configuração do Ninject usando XML.

Para começar a utilizar o Ninject, é necessário importar o namespace **Ninject** e declarar uma variável do tipo **StandardKernel**.

```
using Ninject;

class Program
{
    static void Main(string[] args)
    {
        IKernel kernel = new StandardKernel();
    }
}
```

O objecto kernel será o *container* da aplicação e deverá ser declarado no ponto de entrada da aplicação, tão cedo quanto possível.

Seguindo o exemplo anterior, podemos agora solicitar uma instância de uma variável do tipo **UserManager** ao Ninject sem necessitar de qualquer outra configuração adicional. Para tal, utiliza-se o extension method **Get<>** para obter a instância:

```
using Ninject;

class Program
{
    static void Main(string[] args)
    {
        IKernel kernel = new StandardKernel();
        UserManager manager = ker-
nel.Get<UserManager>();
    }
}

class UserReader
{
    // User reader methods
}

class UserManager
{
    private UserReader userReader;
    public UserManager(UserReader userReader)
    {
        this.userReader = userReader;
    }
}
```

Após a instrução **Get<>**, a variável **manager** do método **Main** fica populada com uma instância de **UserManager** que foi construída usando o construtor que recebe um **UserReader**. O Ninject percebeu que podia construir uma instância da classe **UserReader** para injectar no construtor da classe **UserManager**.

Neste caso, o Ninject aplicou o mecanismo de resolução automática de dependências que constroi instâncias de acordo com as seguintes regras: caso o construtor não tenha argumentos este é invocado para criar a instância. Caso o construtor tenha argumentos, o Ninject tenta criar uma instância de cada uma das classes de argumento do construtor (aplicando as regras de resolução automática).

NOTA: Caso a classe possua mais que um construtor o Ninject tentará utilizar o construtor sem argumentos (construtor *default*) para construir a classe. Caso esta não tenha nenhum construtor *default*, será necessário indicar ao Ninject qual o construtor que este deve utilizar para construir a classe.

Mecanismos de injecção

O exemplo acima utiliza o mecanismo *constructor injection* que faz com que o *container* inspeccione o construtor da classe, tente resolver as dependências deste (i.e. os argumentos do construtor) e chame o construtor passando as dependências identificadas.

O Ninject suporta ainda outro tipo de injecção: a *property injection*. Neste caso, o Ninject encara uma propriedade da classe como uma dependência a ser injectada na classe e tenta resolvê-la e injectá-la como se um argumento de um construtor se tratasse.

Para declarar que uma propriedade deve ser injectada, será necessário decorar a propriedade com o atributo **[Inject]**.

O mecanismo de *constructor injection* é tipicamente mais utilizado que o *property injection* por este garantir que todas as dependências estão no construtor e que o objecto é inicializado correctamente no construtor. Outro motivo a favor da utilização de *constructor injection* é o facto de evitar que as classes "conheçam" o Ninject, já que desta forma raramente será necessário as classes injectadas utilizarem o Ninject.

E quando a resolução automática não consegue resolver?

Há casos em que a resolução automática não consegue resolver a dependência de forma inequivoca. Nestes casos, o Ninject lançará uma excepção e caberá ao programador dar indicação sobre como resolver a dependência.

Tipicamente, existem três casos em que a resolução automática não consegue resolver:

i. Quando a classe tem mais que um construtor em que nenhum deles é o construtor *default*

ii. Quando o construtor tem um argumento do tipo Interface

iii. Quando o construtor tem um argumento com um tipo básico

Nestas situações, o *container* disponibiliza um conjunto de métodos de configuração para instruir o Ninject sobre como proceder. O método **Bind<>()** permite indicar ao Ninject como resolver uma determinada classe ou interface.

Na primeira situação, é possível instruir o Ninject sobre qual o construtor a usar usando o método **Bind<>().ToConstructor().** A utilização do método **WithParameter()**

permite especificar valores a atribuir a certos parâmetros do construtor.

```
class UserReader
{
      public UserReader(string connectionString)
{ ... }
      public UserReader(string connectionString,
int max) { ... }
}
      // Após declarar o kernel
kernel.Bind<UserReader>()
.ToConstructor<UserReader>((ctx) => new UserReader
                                    (string.Empty))
.WithParameter(new Parameter("connectionString",
                        "myConnectionString", true));
```

Neste caso, foi necessário indicar qual o construtor a usar, bem como o valor do parâmetro a utilizar no tipo básico do construtor que foi escolhido. O método **ToConstructor()** instrui o Ninject para utilizar aquele construtor. O método **WithParameter()** indica ao Ninject qual o valor a atribuir ao parâmetro **connectionString**.

No caso em que o construtor necessita de uma interface, utiliza-se o método **Bind<>().To<>()**:

```
class FacebookUserReader : IUserReader
{
      // Get user information from Facebook
}

class UserManager
{
      public UserManager(IUserReader userReader)
{ ... }
}

// Instrução que indica ao NInject que a classe que
//implemementa a interface IuserReader é a classe
//FacebookUserReader
kernel.Bind<IUserReader>().
                        To<FacebookUserReader>();
```

Top tip: Um caso de injecção é utilizar o *container* para injectar um *logger* que, tipicamente, é inicializado em função da classe onde o *logger* é utilizado. Neste caso, é possível utilizar o método **Bind<>().ToMethod()** para garantir a inicialização por classe:

```
kernel.Bind<ILog>().ToMethod(ctx =>
       LogManager.GetLogger(ctx.Request.Service));

class UserManager
{
   public UserManager(IUserReader userReader,
                              ILog logger) { ... }
}
class UserReader
{
public UserReader(string connectionString,
                              ILog logger) { ... }
}
```

Neste caso, a variável **ctx.Request.Service** contém o nome da classe para a qual foi solicitada a injecção. Quando uma classe requisitar uma instância de **ILog,** o Ninject irá executar a Func<> indicada no argumento do método **ToMethod()** para obter uma instância especifica para a classe que a solicitou. No exemplo anterior, seriam injectadas duas instâncias diferentes de **Logger**, uma para cada classe e inicializadas com o nome da respectiva classe onde foi injectada.

Âmbito das dependências

Por omissão, o Ninject cria uma nova instância da classe sempre que há uma solicitação de injecção. Ou seja, sempre que uma classe é necessária o construtor é sempre invocado para satisfazer a dependência.

Ao declarar o *binding* com **Bind<>().To<>().InSingletonScope()** o Ninject garante que aquela classe será um *singleton*, ou seja é construída uma única vez e essa instância é utilizada em todas as dependências.

O *binding* **Bind<>().To<>().InThreadScope()** garante que é criada um instância por cada *Thread* da aplicação.

Utilizando o *binding* **Bind<>().To<>().When()** é também possível instruir o Ninject para injectar diferentes tipos em função de determinada condição ditada pelo método **When()**.

> ***O Ninject é uma biblioteca de software aberto que providencia uma framework de injecção de dependências (...) O padrão Dependency Injection é bastante utilizado em software empresarial o controlo do ciclo de vida dos objectos e facilita a implementação de testes automáticos***

Outras funcionalidades

O Ninject disponibiliza também mecanismos de carregamento de módulos e *plugins* que permitem conferir bastante dinamismo às aplicações.

Existem também extensões que oferecem funcionalidades de interceção de métodos que são particulamente úteis para controlo de segurança, *auditing* ou *troubleshooting*.

A extensão Ninject.Mvc oferece a integração plena do Ninject com o ASP.NET MVC, permitindo que os *Controllers* sejam inicializados pelo *container* do Ninject.

> ***"Ao utilizar uma framework de injecção de dependências, a gestão das instâncias das classes é efectuada pelo container que "injecta" a instância de uma classe sempre que esta é necessária."***

Conclusão

O Ninject é um motor de injecção de dependências bastante completo e versátil que garante ao programador um excelente controlo sobre a forma como a aplicação é construida bem como o ciclo de vida dos objectos de suporte à aplicação.

A sua sintaxe fluída oferece um excelente mecanismo de configuração, evitando os longos ficheiros em formato xml típicos de outras *frameworks* de injecção de dependências.

Nos cenários em que a aplicação está sujeita a testes unitários ou de integração, uma *framework* de injecção de dependências é uma ferramenta importante para garantir a fácil utilização de *mock objects*. O Ninject possui um excelente suporte para alterar os bindings em *runtime*, facilitando em muito a execução dos testes.

> ***"O Ninject é um motor de injecção de dependências bastante completo e versátil que garante ao programador um excelente controlo sobre a forma como a aplicação é construída bem como o ciclo de vida dos objectos de suporte à aplicação."***

Pela forma como está construído, apenas as classes de ponto de entrada das aplicações terão que "conhecer" o Ninject, sendo o resto da aplicação completamente agnóstica ao motor de injecção de dependências.

A utilização de módulos permite distribuir a inicialização dos vários componentes aplicacionais, garantido uma melhor organização do código e evitando ficheiros de configuração demasiado longos.

AUTOR

Escrito por Nuno Caneco

Software Development Lead @ Siemens

nuno.caneco@gmail.com https://pt.linkedin.com/in/nunocaneco

Como usar base de dados SQLite em Windows 10 Universal Apps

Âmbito

Este artigo tem como objetivo mostrar como usar uma base de dados SQLite em Windows 10 Universal Apps, mais especificamente ligação à base de dados, obtenção e persistência de dados.

Introdução

Hoje em dia qualquer aplicação tem como requisito o uso de base de dados para armazenar os dados, desta forma existem várias soluções relacionais ou não, para o efeito. Uma solução muito usada nas aplicações móveis é o uso de base de dados SQLite, uma vez que é uma solução simples e fácil de usar. Em Windows 10 Universal Apps é possível usar base de dados SQLite, que atualmente se encontra numa versão *preview*, e que pode sofrer alterações até sair a versão final.

É sabido que a Entity Framework 7 irá suportar o Windows 10 Universal apps, no entanto este artigo não irá usar esta solução e será usada a biblioteca SQLite.Net-PCL, uma vez que é atualmente uma das bibliotecas mais simples de usar, e cujo código fonte está disponível no GitHub.

Nota: Este artigo foi escrito usando a versão Windows 10 Technical Preview Build 10240 e a Technical Preview Tools version 10.0.10069.

Figura 1: Versão do Windows 10

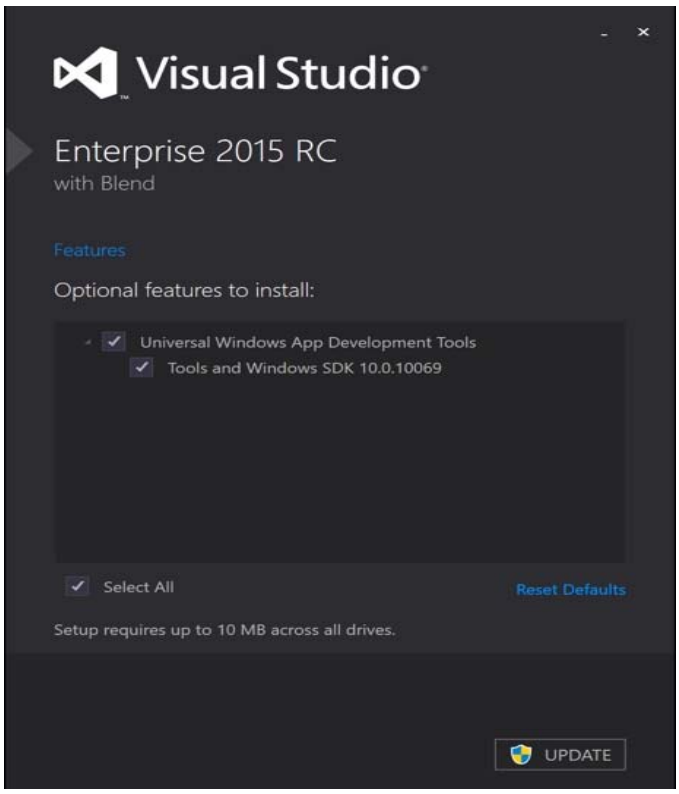


Figura 2: Versão do Technical Preview Tools

Descrição

Comecemos por criar uma Windows 10 Universal app, como podemos ver na figura 3:

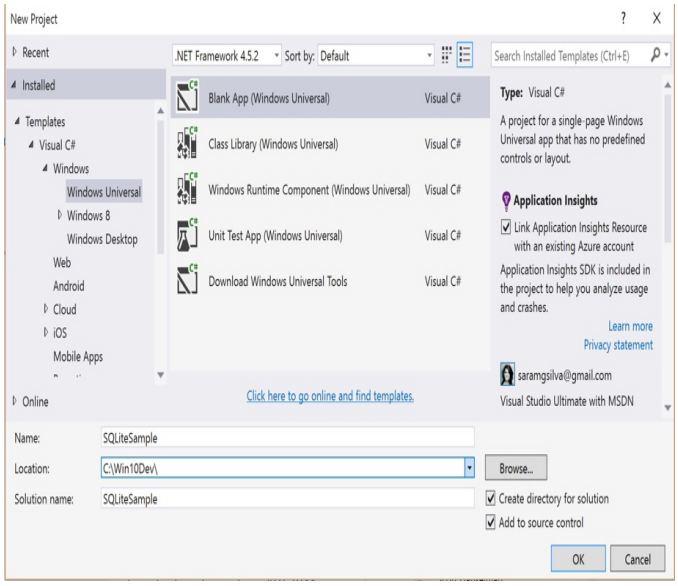


Figura 3: Criação de Windows 10 Universal App

Cujo resultado será :

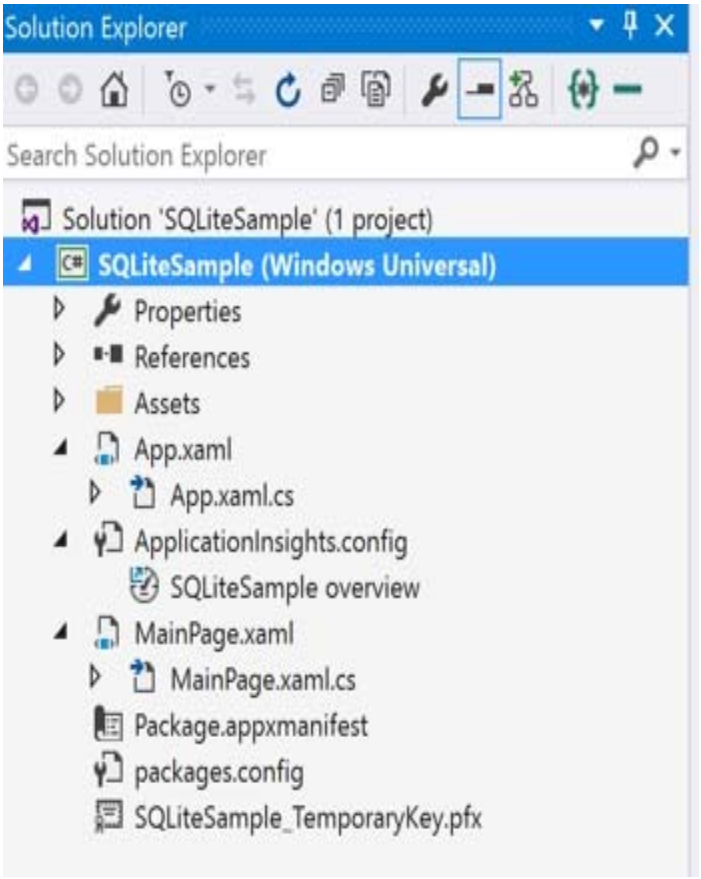


Figura 4: Solução de uma aplicação Windows 10 Universal

Para os mais novatos em Windows 10 Universal apps é preciso ter atenção que deixamos de desenvolver aplicações para cada plataforma, e passamos a ter uma única aplicação que irá correr nas várias plataformas, por esta razão passamos a ter apenas um projeto na solução. Para dar suporte a funcionalidades específicas de cada plataforma passamos a ter disponíveis as extensões, que só irão estar disponível na respetiva plataforma para o qual foram fornecida.

Modelo de dados

Uma vez que temos o projeto criado, agora devemos criar o modelo de dados que define os dados a persistir pela aplicação na base de dados SQLite. Desta forma, para ajudar a construir o exemplo iremos usar um dos datasets fornecido pelo http://www.cityofboston.gov/, mais especificamente o dataset https://data.cityofboston.gov/resource/4swk-wcg8.json.

Sendo assim, iremos ter uma classe chamada BostonEmployee, que é definida por:

```
public class BostonEmployee
    {
        public string Name { get; set }

        public string Title { get; set; }

        [JsonProperty(PropertyName =
                                "department_name")]
        public string DepartmentName { get;
                                            set; }

        public double Regular { get; set; }

        [JsonProperty(PropertyName =
                                "total_earnings")]
        public string TotalEarnings { get; set; }

        public double Value { get; set; }

        public string Zip { get; set; }

        public string Detail { get; set; }

        public string Injured { get; set; }

        public string Other { get; set; }

        public string Retro { get; set; }

        public string Overtime { get; set; }

        public string Quinn { get; set; }
    }
```

Nota: É requisito instalar o NuGet Json.net (Figura 5) uma vez que é alterado o nome das propriedades em relação ao json recebido, de forma a se ter uma leitura mais legível.

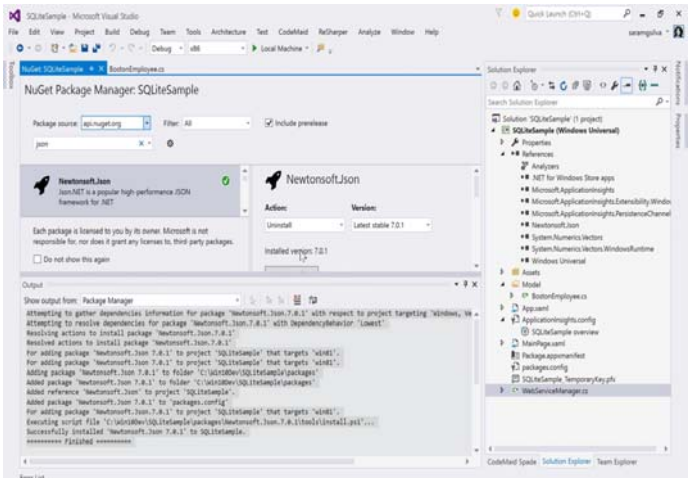


Figura 5: Instalação do NuGet Json.Net

O consumo dos dados será um pedido HTTP, muito simples, como podemos ver de seguida:

```
public class WebServiceManager
{
   public async Task GetBostonEmployeesEarnings()
     {
       string url = "https://
   data.cityofboston.gov/resource/4swk-wcg8.json";

            var client = new HttpClient();
            var request = new HttpRequestMessage
                  (HttpMethod.Get, new Uri(url));

            // Send the registration request.
            var response = await client.SendAsync
                                          (request);
            var message = await
```

```
                response.Content.ReadAsStringAsync();
            var employees = JsonCon
                    vert.DeserializeObject(message);
            return employees;
        }
}
```

Usando SQLite

Instalação

Para podermos usar base de dados SQLite é necessário instalar o vsix fornecido no site oficial da SQLite: http://sqlite.org/download.html, atualmente em versão preview (Figura 6) e adicionar as respetivas referências (Figura 7 e 8):

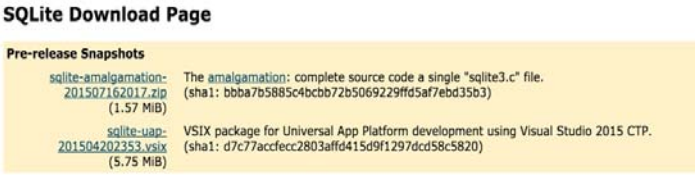


Figura 6: Obtenção do SQLIte para UAP (sqlite-uap-201504202353.vsix)

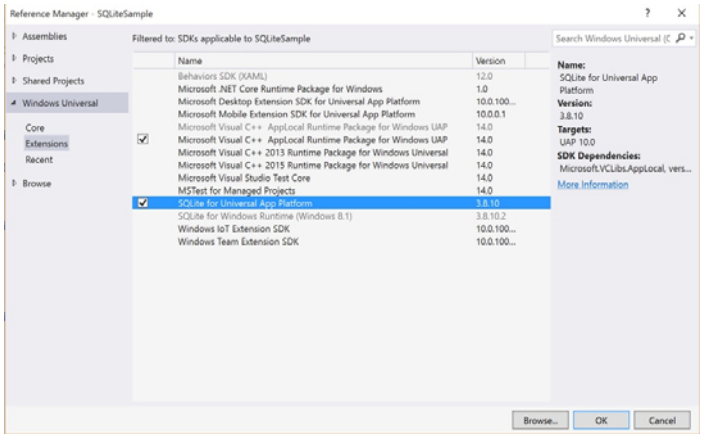


Figura 7: Adicionar referências

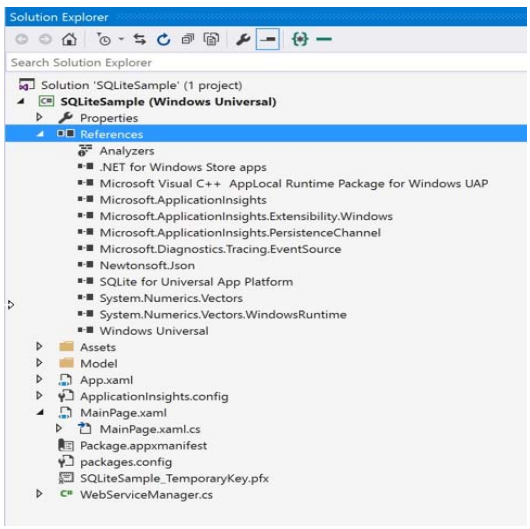


Figura 8: Referências da aplicação

Portanto, neste momento temos o SQLite instalado e as referências foram adicionadas ao projeto, falta agora instalar o NuGet SQLite.Net-PCL (Figura 9), que é uma biblioteca que irá facilitar a criação e conexão à base de dados, assim como na gestão de dados. Este NuGet irá adicionar as seguintes referências:

- Net
- Net.Platform.WinRT

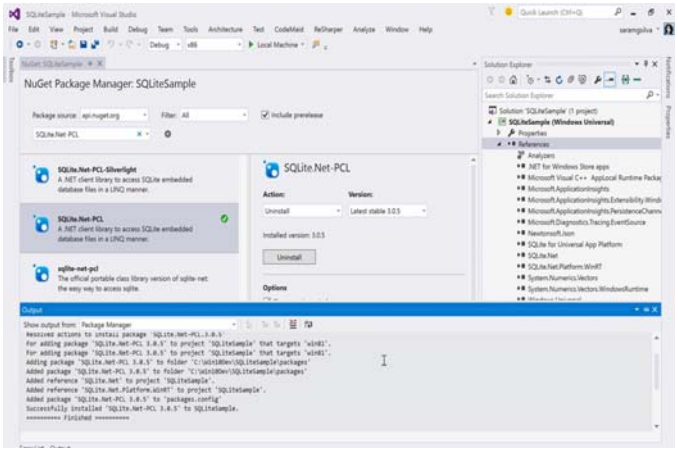


Figura 9: Instalação do NuGet SQLite.Net-PCL

Criação da base de dados

Antes de criarmos a base de dados é necessário definir o percurso da base de dados, isto é, o local onde a base de dados vai ser guardada. Este percurso irá ser utilizados em futuros acessos à base de dados, desta forma podemos definir o *databasePath* no constructor:

```
private string _dataBasePath;

public MainPage()
{
    InitializeComponent();
    _dataBasePath = Path.Combine
        (Windows.Storage.ApplicationData.Current.
                LocalFolder.Path, "db.sqlite");
}
```

De seguida devemos alterar a classe BostonEmployee de forma a definir a PrimaryKey:

```
[PrimaryKey]
public string Name { get; set; }
```

E depois podemos então aceder à base de dados, para podermos criar a tabela referente ao BostonEmployee:

```
public void CreateDatabase()
{
    using (SQLite.Net.SQLiteConnection conn = new
                SQLite.Net.SQLiteConnection(new
 SQLite.Net.Platform.WinRT.SQLitePlatformWinRT(),
                                    _dataBasePath))
    {
        conn.CreateTable();
    }
}
```

Gestão de dados

Uma vez que já criamos a base de dados, agora podemos inserir, apagar e alterar os dados da mesma, para isso devemos usar o SQLiteConnection que terá todos os métodos necessários para efetuar as operações CRUD:

Inserir

```
public void SaveData(BostonEmployee bostonEmployee)
{
    using (SQLite.Net.SQLiteConnection conn = new
            SQLite.Net.SQLiteConnection(new SQLite.
        Net.Platform.WinRT.SQLitePlatformWinRT(),
                                        _dataBasePath))
    {
        conn.Insert(bostonEmployee);
        conn.Commit();
    }
}

public void SaveData(IEnumerable bostonEmployees)
{
    using (SQLite.Net.SQLiteConnection conn = new
                        SQLite.Net.SQLiteConnection(
                    new SQLite.Net.Platform.WinRT.
                            SQLitePlatformWinRT(),
                                        _dataBasePath))
    {
        conn.InsertAll(bostonEmployees);
        conn.Commit();
    }
}
```

Obter

Para obter o BostonEmployee usando o *Name*, podemos efectuar:

```
public BostonEmployee GetByName(string name)
{
    using (SQLite.Net.SQLiteConnection conn = new
            SQLite.Net.SQLiteConnection(new SQLite.
        Net.Platform.WinRT.SQLitePlatformWinRT(),
                                        _dataBasePath))
    {
        return conn.Get(name);
    }

}
```

Para obter BostonEmployees baseadas numa pesquisa ao Title, podemos efectuar:

```
public BostonEmployee GetBostonEmployee(string
                                                title)
{
    using (SQLite.Net.SQLiteConnection conn = new
        SQLite.Net.SQLiteConnection(new SQLite.Net.
              Platform.WinRT.SQLitePlatformWinRT(),
                                        _dataBasePath))
    {
        return conn.Get(i=>i.Title.Equals(title));
    }
}
```

Para fazer paginação sobre os dados, podemos efectuar:

```
public IEnumerable Pagination(int numToSkip, int
                                            numToTake)
{
    using (SQLite.Net.SQLiteConnection conn = new
                        SQLite.Net.SQLiteConnection(
                    new SQLite.Net.Platform.WinRT.
            SQLitePlatformWinRT(), _dataBasePath))
    {
        return conn.Table().Skip(numToSkip).Take
                                        (numToTake);
    }
}
```

Para fazer queries SQL, podemos efectuar:

```
private string sqlquery = "SELECT * FROM
            BostonEmployee] WHERE [Value] = 0.0";

public IEnumerable  SQLQuery(string sqlquery)
{
    using (SQLite.Net.SQLiteConnection conn =
       new SQLite.Net.SQLiteConnection(new SQLite.
         Net.Platform.WinRT.SQLitePlatformWinRT(),
                                        _dataBasePath))
    {
        return conn.Query(sqlquery);
    }
}
```

Nota: É de notar que é possível usar os vários métodos fornecidos pelo *Linq*, que permitirá efetuar diferentes queries à tabela em causa, assim como usar queries em SQL.

Apagar

```
public void DeleteData(BostonEmployee
                                        bostonEmployee)
{
    using (SQLite.Net.SQLiteConnection conn = new
                        SQLite.Net.SQLiteConnection(new
 SQLite.Net.Platform.WinRT.SQLitePlatformWinRT(),
                                        _dataBasePath))
    {
        conn.Delete(bostonEmployee);
        conn.Commit();
    }
}
```

Alterar

```
public void UpdateData(BostonEmployee
                                        bostonEmployee)
{
    using (SQLite.Net.SQLiteConnection conn =
              new SQLite.Net.SQLiteConnection(new
                                SQLite.Net.Platform.
                    WinRT.SQLitePlatformWinRT(),
                                        _dataBasePath))
    {
        conn.Update(bostonEmployee);
        conn.Commit();
    }
}
```

Conclusão

Em conclusão podemos verificar que o uso de base

de dados em Windows 10 Universal Apps é simples, e o uso de base de dados SQLite é atualmente uma das soluções disponível (em versão *preview*), cuja implementação não difere em muito do uso em aplicações WinRT. Sendo muito simples criar e efetuar as operações CRUD usando a biblioteca SQLite.Net-PCL.

> ***Hoje em dia qualquer aplicação tem como requisito o uso de base de dados para armazenar os dados***

> ***Windows 10 Universal apps é preciso ter atenção que deixamos de desenvolver aplicações para cada plataforma, e passamos a ter uma única aplicação que irá correr nas várias plataformas***

AUTOR

Escrito por Sara Silva

Licenciada em Matemática pelo DMUC, e o seu foco de desenvolvimento está direccionado para a área Mobile, sendo a sua principal especialidade aplicações para Windows. Atualmente desenvolve na área do Windows, Xamarin, Azure, e é Microsoft MVP Mentor. É autora de vários artigos técnicos e tutoriais. 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.

Como fazer o deploy de uma aplicação web com PrimeFaces no OpenShift

Neste artigo vou mostrar como importar a User interface (UI) Framework PrimeFaces (http://primefaces.org/) para o desenvolvimento de uma aplicação web de JavaServer Faces (JSF) utilizando NetBeans (https://www.netbeans.org) e alojar a mesma aplicação na plataforma OpenShift por SFTP.

O OpenShift Online (https://www.openshift.com) é um serviço em nuvem da Red Hat (http://www.redhat.com) para desenvolvimento de aplicações e alojamento ou seja é um Platform as a service (PaaS) que permite otimizar a aplicação web e não necessitamos de preocupar com a infraestrutura necessária para distribuir a aplicação web.

Uma das vantagens de utilizarmos o OpenShift é a utilização de um plano gratuito sem necessitarmos de cartão de crédito.

Principais diferenças nos planos disponíveis:

Plano	Gratuito	Bronze	Prateado
Preço base	Gratuito	Gratuito	€15/Mês
Inatividade da aplicação	24 Horas	Nunca	Nunca
Escalabilidade	Sim (3 min / 3 max)	Sim (3 min / 16 max)	Sim (3 min / 16 max)
Alojamento	1 GB	1 GB	6 GB
Suporte	Comunidade	Comunidade	Red Hat e Comunidade

No plano gratuito se a aplicação não receber nenhum pedido a mesma é desligada e apenas volta a ser ligada quando receber um novo pedido. Para que a aplicação se mantenha ligada normalmente utilizam-se técnicas para fazer pedidos de X em X tempos.

Um dos principais concorrentes do OpenShift é a Microsoft Azure que permite também um plano gratuito para alojamento de aplicações Java mas este apenas permite 1 GB de tráfego. No OpenShift nunca fui avisado por utilização de tráfego e a aplicação web continua a funcionar.

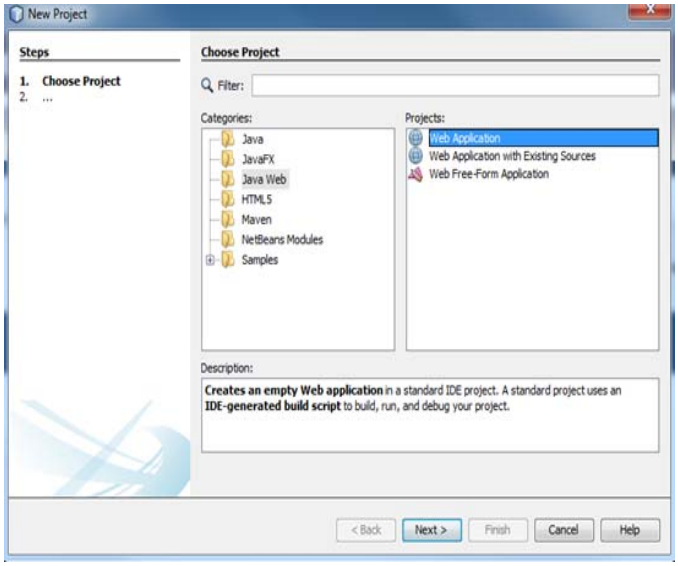
O PrimeFaces é uma UI Framework para Websites em JSF. Esta Framework contém um grande conjunto de componentes UI em JSF Ajax APIs, e pode consultar os componentes no showcase que estão disponíveis na página de internet do PrimeFaces. O desenvolvedor tem à sua disponibilidade quatro versões PRO, CASE, Elite e Comunidade.

	PRO	CASE	Elite	Comunidade
Custo	Subscrição anual por desenvolvedor			Gratuito (incluído comercial)
Suporte	PrimeFaces (resposta no máximo em um dia útil)	PrimeFaces	PrimeTek www.primetek.com.tr	Comunidade PrimeFaces

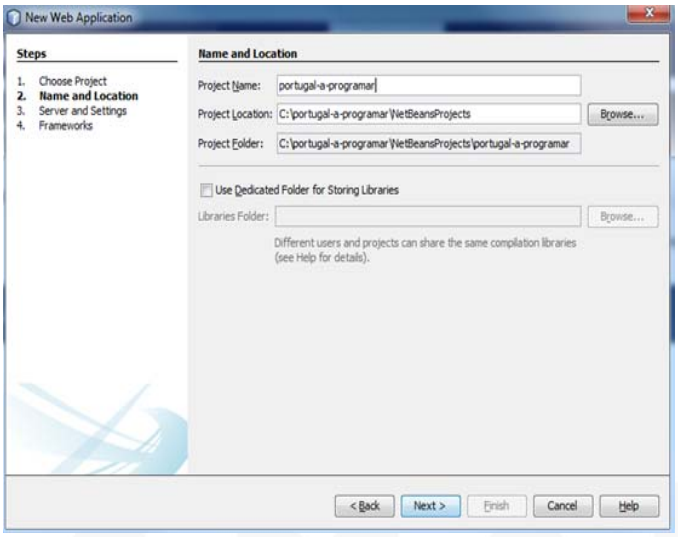
O criador do PrimeFaces também disponibiliza o Mobile UI kit para aplicações mobile, Layouts e temas para as mesmas.

Para criar a aplicação deste projeto web em JSF vai ser utilizado IDE Netbeans esta aplicação é patrocinada pela Oracle e permite o desenvolvimento de projetos Java, HTML5, PHP e C/C++ não tem nenhum custo para o utilizador nem para empresas.

O idioma utilizado neste Netbeans é o Inglês. Selecione a opção “New Project “ para iniciar um novo projeto depois é selecionada a categoria “Java Web” em que é selecionado o projeto “Web Application” sem código sem código incluído, após a seleção avança para fase seguinte.

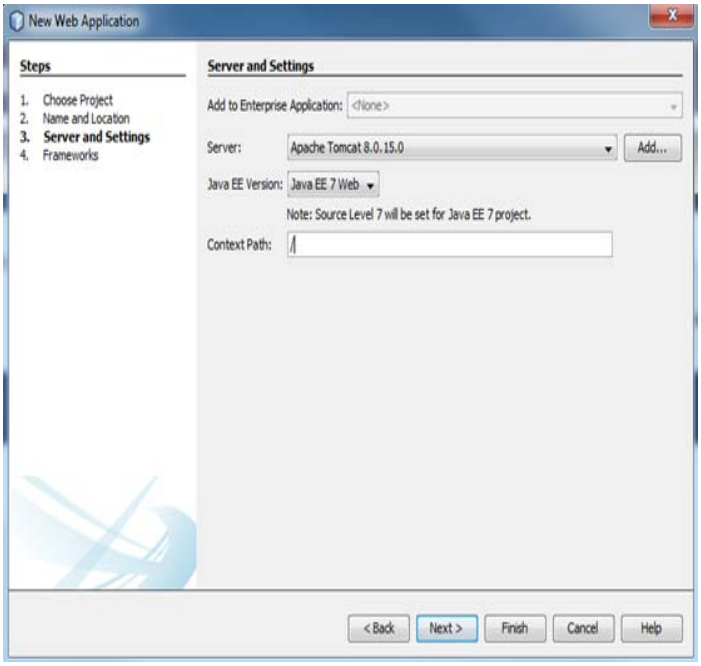


É definido o nome do projeto e a localização do mesmo mas não é definida a localização da pasta das bibliotecas e avança para fase seguinte.

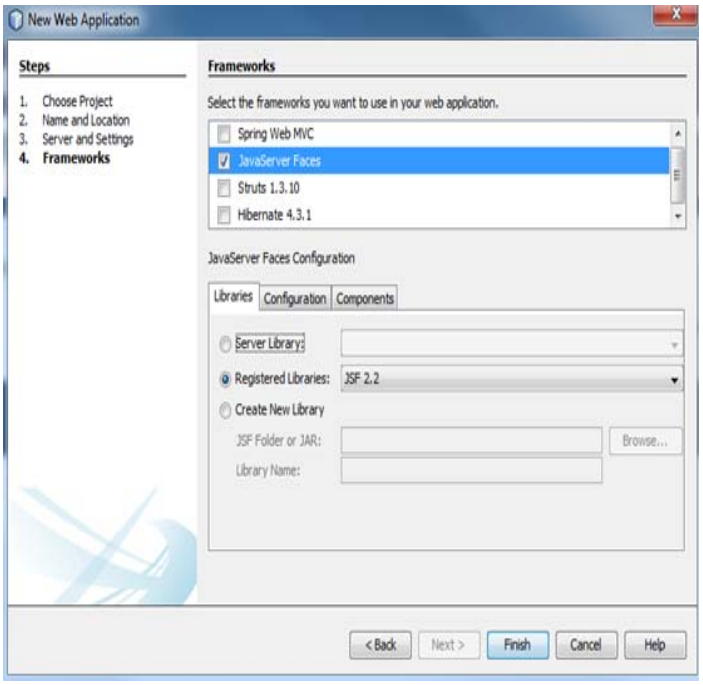


COMO FAZER O DEPLOY DE UMA APLICAÇÃO WEB COM PRIMEFACES NO OPENSHIFT

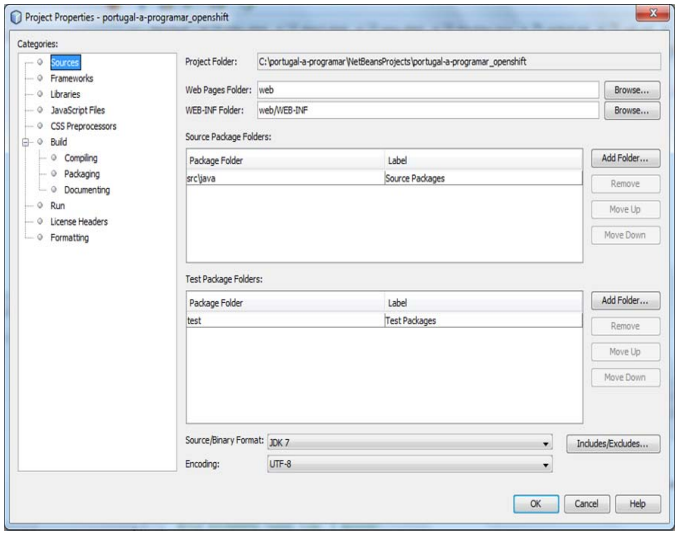
O servidor a ser utilizado é o Apache Tomcat e é o mesmo que vai ser utilizado no OpenShift após a seleção avança para fase seguinte.



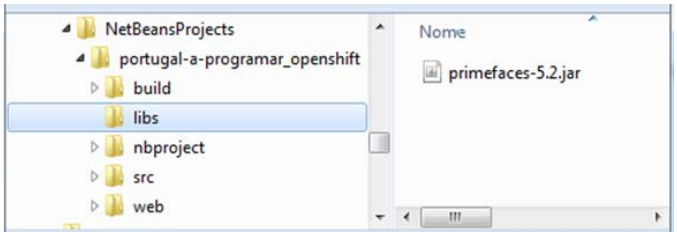
O projeto vai trabalhar com JavaServer Faces o NetBeans já inclui as bibliotecas para o PrimeFaces mas não vai ser utilizada esta funcionalidade vais ser descarregado a última versão e importada para o projeto. Após a seleção avança para fase seguinte.



Com o projeto criado a primeira coisa fazer é definir o "Source/Binary Format" do projeto para JDK 7, porque o OpenShift ainda não trabalha com o Java 8. Para definir clica-se com o botão direito do rato em cima no nome do projeto, seleciona-se "Properties", seleciona-se a opção "Sources" e define-se "Source/Binary Format" para JDK 7.



Agora pode importar-se o PrimeFaces. Por norma eu crio uma pasta com nome "libs" na raiz do projeto onde coloco todas as bibliotecas que utilizo e que não estão integradas no IDE ou que são mais recentes já que assim consigo abrir o projeto noutros computadores sem problemas. Também pode se utilizar o Maven.



Após a importação da biblioteca é necessário que a mesma seja reconhecida nas páginas XHTML que utilizem os componentes UI assim é necessário adicionar a seguinte linha xmlns:p="http://primefaces.org/ui" no tag <html> .

O PrimeFaces tem uma ótima compilação de documentação disponível na sua página de Internet http://primefaces.org/documentation como aplicar os seus componentes e também estão disponíveis demostrações dos seus componentes em http://www.primefaces.org/showcase/ .

Vão ser utilizadas três demonstrações . A primeira um input com validação, a segunda é idêntica à primeira mas com mensagens sobrepostas e a terceira é um menu para navegarmos entre as duas demonstrações anteriores.

Antes de iniciar a criação das demonstrações O tag url-pattern no ficheiro web.xml na pasta WEB-INF será alterado e adicionado um context-param para o PrimeFaces.

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="3.1" xmlns="http://
                                    xmlns.jcp.org/
xml/ns/javaee" xmlns:xsi="http://www.w3.org/2001/
          XMLSchema-instance" xsi:schemaLocation=
         "http://xmlns.jcp.org/xml/ns/javaee
                http://xmlns.jcp.org/xml/ns/javaee/
                                  web-app_3_1.xsd">
    <context-param>
        <param-name>javax.faces.PROJECT_STAGE
                                        </param-name>
```

```
        <param-value>Development</param-value>
    </context-param>
     <context-param>
        <param-name>javax.faces.CONFIG_FILES
        </param-name>
        <param-value>/WEB-INF/faces-config.xml
        </param-value>
    </context-param>
    <servlet>
        <servlet-name>Faces Servlet</servlet-name>
        <servlet-class>
    javax.faces.webapp.FacesServlet</servlet-class>
        <load-on-startup>1</load-on-startup>
    </servlet>
    <servlet-mapping>
        <servlet-name>Faces Servlet</servlet-name>
        <url-pattern>*.XHTML</url-pattern>
    </servlet-mapping>
    <session-config>
        <session-timeout>
            30
        </session-timeout>
    </session-config>
    <welcome-file-list>
        <welcome-file>index.XHTML</welcome-file>
    </welcome-file-list>
</web-app>
```

É criado um novo ficheiro com nome faces-config.xml na mesma pasta com o seguinte conteúdo.

```
<?xml version='1.0' encoding='UTF-8'?>
<faces-config version="2.2"
         xmlns="http://xmlns.jcp.org/xml/ns/javaee"
          xmlns:xsi="http://www.w3.org/2001/
                                XMLSchema-instance"
         xsi:schemaLocation="http://xmlns.jcp.org/
  xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/
                           web-facesconfig_2_2.xsd">
</faces-config>
```

As alterações anteriores não são obrigatórias mas fica a conhecer os ficheiros onde se definem as configurações a serem utilizadas no PrimeFaces. Com por exemplo a configuração das páginas em que o utilizador consegue aceder sem sessão e as páginas que apenas acede com uma sessão iniciada.

Demo 1 – Validação de entrada de dados executados no servidor (http://www.primefaces.org/showcase/ui/ajax/validation.XHTML)

Na página de índex (index.XHTML) é apagada a linha referente ao Hello World e introduzido o seguinte código que contém o formulário que solicita a introdução de dados.

```
<h:form>
 <p:panel id="panel" header="Demo: http://
              www.primefaces.org/showcase/ui/ajax/
                                  validation.XHTML">
  <p:messages id="msgs" />
  <h:panelGrid columns="3" cellpadding="5">
  <p:outputLabel for="name" value="Insira o seu
                                          nome:" />
  <p:inputText id="name" value="#
  {welcomeView.name}" required="true" label="name">
   <f:validateLength minimum="2" />
  </p:inputText>
  <p:message for="name" display="icon" />
  </h:panelGrid>
  <p:commandButton value="Submeter" update="panel"
                                  actionListener="#
                                  {welcomeView.welcome}"
                                icon="ui-icon-check" />
</p:panel>
</h:form>
```

Para validar dos dados introduzidos é necessário criar uma nova Java Class que contém o código que o servidor vai utilizar para validação dos dados recebidos pelo formulário e entregar a mensagem construída.

```
import javax.faces.application.FacesMessage;
import javax.faces.bean.ManagedBean;
import javax.faces.context.FacesContext;

@ManagedBean
public class WelcomeView {

    private String name;

    public String getName() {
        return name;
    }

    public void setName(String sname) {
        this.name = sname;
    }

    public void welcome() {
       FacesContext.getCurrentInstance()
                                        .addMessage
          (null, new FacesMessage("Olá " + name +
                                  " bem-vindo"));
    }
}
```

Demo 2 – É idêntico ao demo 1 mas com mensagem de informação ao utilizador sobrepostas.

É criado uma nova página XHTML que vai estar na raiz. É necessário introduzir a linha xmlns:p="http://primefaces.org/ui"> no tag “<html>” e alterar os seguintes tags “head” para “h:head” e “body” para “h:body”.

É inserido o código referente ao formulário

```
<h:form>
 <p:growl id="growl" showDetail="true"
                                    sticky="true" />
 <p:panel header="Demo 2 -
   http://www.primefaces.org/showcase/ui/message/
                                      growl.XHTML">
  <h:panelGrid columns="2" cellpadding="5">
   <p:outputLabel for="msg" value="Mensagem:" />
   <p:inputText id="msg" value="#
          {growlView.message}" required="true" />
  </h:panelGrid>
   <p:commandButton value="Guardar"
        actionListener="#{growlView.saveMessage}"
                                  update="growl" />
 </p:panel>
</h:form>
```

É criada uma nova Java Class que contém o código para validação.

```
import javax.faces.application.FacesMessage;
import javax.faces.bean.ManagedBean;
import javax.faces.context.FacesContext;

@ManagedBean
```

```
public class GrowlView {

    private String message;

    public String getMessage() {
        return message;
    }

    public void setMessage(String message) {
        this.message = message;
    }

    public void saveMessage() {
        FacesContext context =
                  FacesContext.getCurrentInstance();
        context.addMessage(null, new FacesMessage
    ("Com sucesso",  "Sua mensagem: " + message) );
        context.addMessage(null, new FacesMessage
        ("Segunda mensagem", "Detalhe adicional"));
    }
}
```

Demo 3 – Menu

A Framework disponibiliza vários tipos de menu onde é utilizada a componente Breadcrumb que fornece informações contextuais sobre a hierarquia da página.

Para o menu é criada uma nova página XHTML e a mesma é importada nas restantes páginas assim cada vez que é necessário atualizar o menu essa alteração é feita apenas numa única página XHTML.

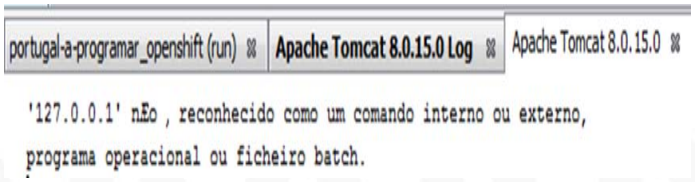
```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/XHTML"
      xmlns:h="http://xmlns.jcp.org/jsf/html"
      xmlns:p="http://primefaces.org/ui">
  <h:body>
    <h:form>
      <p:breadCrumb>
        <p:menuitem value="Demo 1"
                            url="index.XHTML" />
        <p:menuitem value="Demo 1"
                            url="index.XHTML" />
        <p:menuitem value="Demo 2"
                            url="demo2.XHTML" />
      </p:breadCrumb>
    </h:form>
  </h:body>
</html>
```

Nas restantes páginas em que o menu vai ser utilizado é necessário introduzir a linha xmlns:ui="http://xmlns.jcp.org/jsf/facelets"> no tag “<html>”.

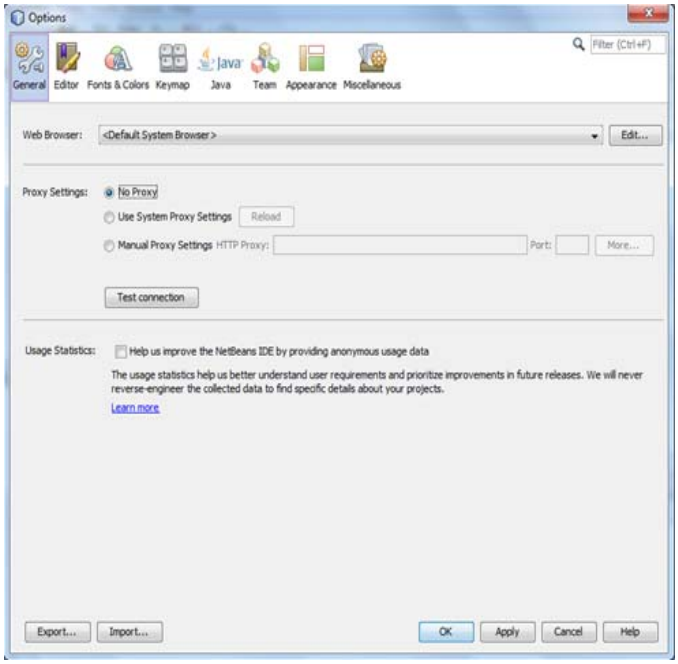
É inserida a importação do menu antes dos formulários.

```
<ui:include src="menu.XHTML"></ui:include>
```

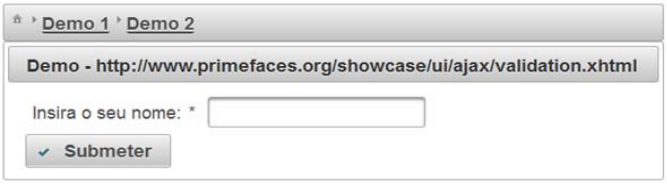
O projeto está terminado. Agora pode ser executado para visualizar o resultado final. Ao iniciar o projeto e deparar-se com o seguinte erro.



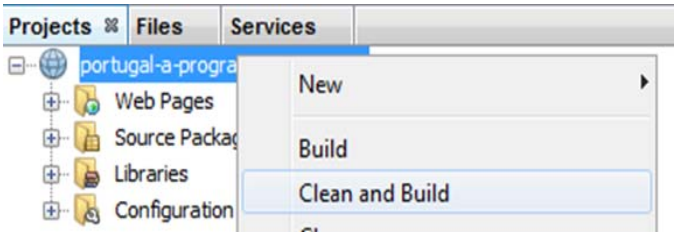
Para ser corrigido tem que definir a não utilização do proxy nas opções gerais do NetBeans.



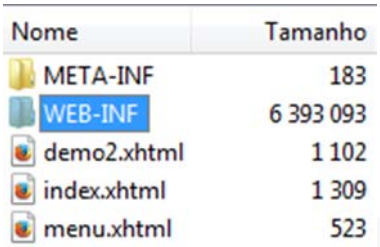
E assim terá o seguinte resultado quando inicia o projeto com o Tomcat.



Com o projeto terminado já se pode criar o ficheiro WAR que contém todos os ficheiros do projeto a serem enviados para o OpenShift através de SFTP. Para criar o ficheiro clique no projeto com o botão direito do rato selecione “Clean and Build”.



O ficheiro WAR é basicamente um ficheiro compactado no formato zip com todos os ficheiros do projeto se o abrir com o compactador de ficheiros com suporte zip pode ver o seu conteúdo.

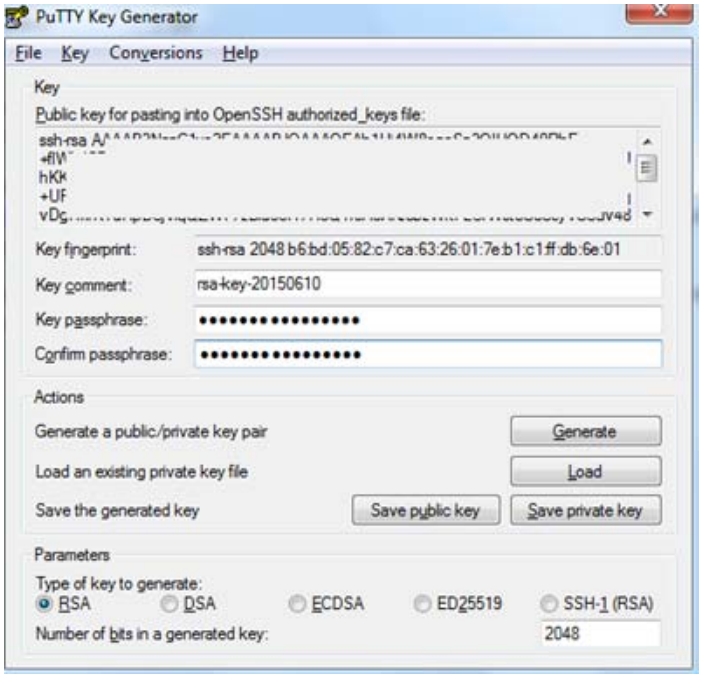


COMO FAZER O DEPLOY DE UMA APLICAÇÃO WEB COM PRIMEFACES NO OPENSHIFT

Antes de enviar o ficheiro o projeto para OpenShift é necessário configurar o SFTP com o nosso certificado digital para que o mesmo aceite a ligação SFTP.

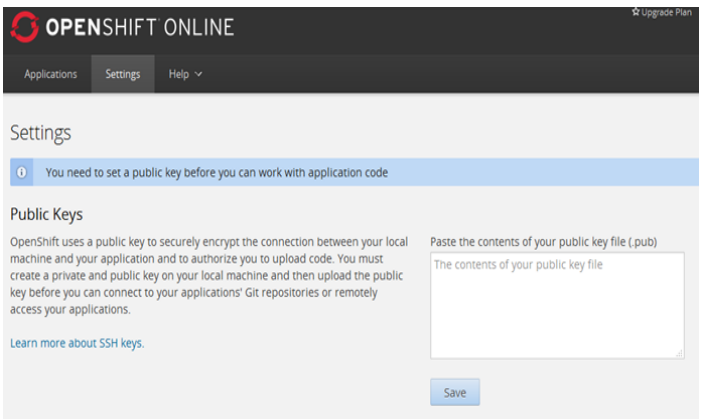
Para criar o certificado é utilizada a aplicação PuTTYgen (http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html). Atenção faça download apenas do site oficial já que existem versões não oficiais que "roubam" as credenciais criadas.

Para criar o certificado inicia-se o PuTTY Key Generator e clica-se no botão "Generate" a seguir move-se o rato aleatoriamente na área em branco até o certificado ser criado e depois define-se a key passphrase.

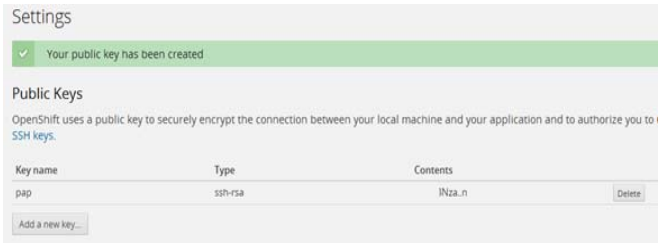


É necessário guardar a chave pública e a chave privada.

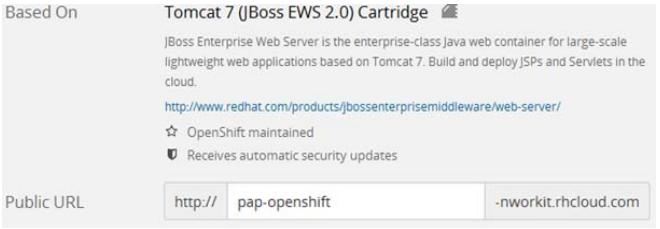
Inicie a sua conta do OpenShift clique em definições e insira a sua chave pública conforme aparece no PuTTY Key Generator.



Se pretender alterar ou apagar a chave pública pode sempre apagar no botão delete e criar uma nova.



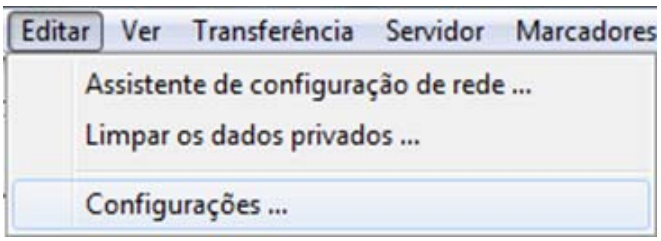
Com a chave inserida podemos criar uma nova aplicação clicando em *Applications* e selecionando "Tomcat 7 (JBoss EWS 2.0)" a seguir define-se o endereço da aplicação.



Nos planos gratuitos por norma não é possível definir-se a região é necessário definir-se "No preference".

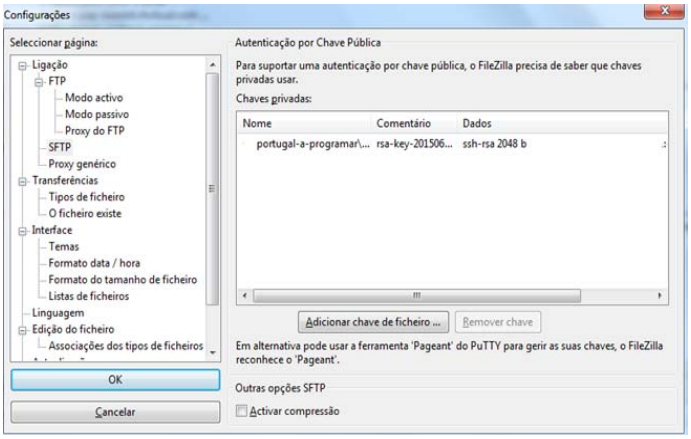
Com a aplicação criada pode consultar-se o endereço e o utilizador para a ligação SSH que está definido no "Soure Code" que tem o seguinte aspeto 854a6dZ40c000d820@xpto-pap.rhcloud.com então o nome de utilizador é 854a6dZ40c000d820 e servidor xpto-pap.rhcloud.com.

Para ligarmos através de SFTP vou demonstrar com aplicação FileZilla. Primeiro é necessário adicionar a chave privada no menu da aplicação selecione a opção "Editar" e depois "configurações".

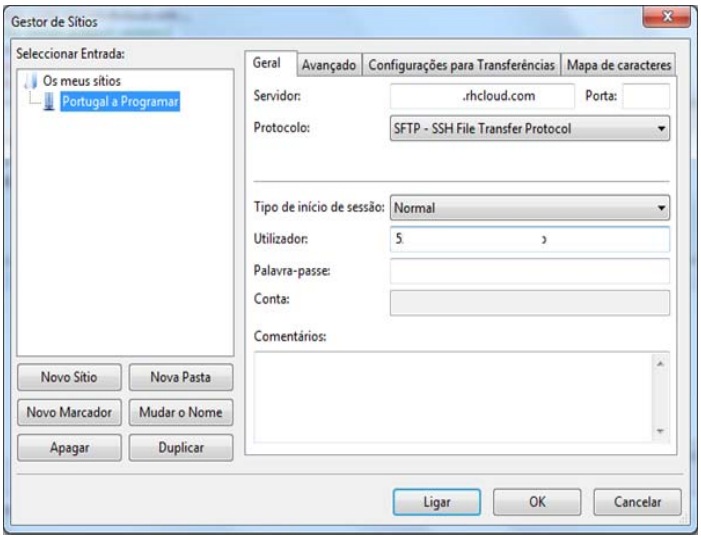


Na janela configurações selecione a página SFTP e clique em "Adicionar chave de ficheiro..", adicione a mesma, que foi criada anteriormente e aplicação questiona se quer

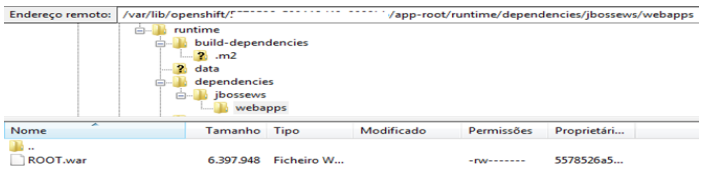
converter clique “Sim” vai ser solicitado a passphrase que foi definida na criação da chave privada.



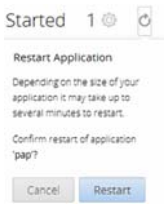
Agora pode criar-se a a ligação ao OpenShift. Com as indicações do “Soure Code” na aplicação do OpenShift que tem o seguinte aspeto 854a6dZ40c000d820@xpto-pap.rhcloud.com o servidor vai ser xpto-pap.rhcloud.com o tipo de sessão é normal o utilizador 854a6dZ40c000d820.Para iniciar a ligação clica-se no botão “Ligar”.



Com a ligação efetuada já pode substituir o ficheiro WAR existente pelo WAR do projeto criado. Para isso selecione a pasta webapps localizado em /var/lib/openshift/[Nome de Utilizador]/app-root/runtime/dependencies/jbossews/webapps” e substitua o ficheiro. Mas antes de fazer é obrigatório mudar o nome do ficheiro WAR do projeto para ROOT.war. Se não mudar o nome do ficheiro e o mesmo se chamar PAP123 a aplicação não fica na raiz mas sim numa subpasta da raiz com o nome PAP123.

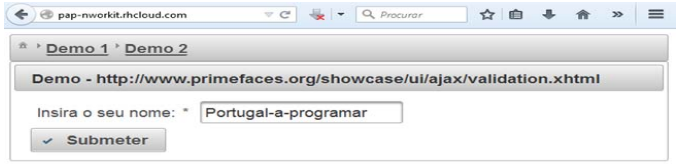


Depois de enviarmos o ficheiro reiniciamos a aplicação no painel do OpenShift.



Atenção: Apagar outras pastas ou ficheiros o SFTP pode danificar a aplicação web.

E ficamos com o seguinte resultado.



A vantagem de se criarem aplicações web em Java no próprio servidor é não obrigar o utilizador a instalar o Java. Mas se o projeto trabalhar com os certificados digitais do utilizador então já é obrigado a instalar o Java para que o mesmo seja validado e identificado. A vantagem de utilizarmos uma UI Framework é de não necessitarmos criar os componentes UI e assim poupar tempo nos projetos. Mesmo utilizando a versão de comunidade existe um grande suporte no fórum do PrimeFace e em grupos do Google ou Facebook. É necessário ter atenção as novas versões em que algumas propriedades são alteradas mas estão identificadas no manual do PrimeFaces. O OpenShift é uma ótima solução de projetos para clientes em SASS /PASS mas também se pode utilizar para experimentar o seu comportamento e/ou para mostrar. Também podemos utilizar o plano gratuito para alojamento de uma página pessoal, blog ou outros.

Os ficheiros do projeto estão disponíveis no Github em https://github.com/rramoscabral/pap-2015-openshift_and_primefaces.

AUTOR

Escrito por Ricardo Cabral

Licenciado em Engenharia Informática pela Universidade Autónoma de Lisboa. O seu twitter é @rramoscabral

Manipulação ao nível do bit na Linguagem C

É sabido que um computador trabalha em modo binário, armazenando e manipulando *bits*, isto é, *zeros* e *uns*. Este artigo procura resumir as metodologias mais comuns para uso e manipulação de *bits* através da linguagem C.

Base binária, octal e hexadecimal

A designação *bit* identifica um valor da base binária. Como o nome sugere, a base binária é composta por dois valores distintos, representados por zero e um, daí também se designar por base *dois*. Assim, um *bit* pode assumir um desses dois valores, sendo muitas vezes empregue para representar um estado ativo (*bit* com o valor a 1) ou inativo (*bit* com valor a 0).

Vários *bits* podem ser agrupados para formar valores com maior amplitude. Concretamente, sempre que se acrescenta um *bit*, está-se a duplicar o número de valores passíveis de serem representados pelo conjunto de bits. Por exemplo, com um bit consegue-se representar dois estados (0 e 1). Com dois bits já é possível representar quatro estados (00, 01, 10 e 11) e com três *bits* oito estados (000, 001, 010, 011, 100, 101, 110 e 111). De uma forma geral, *n* bits permitem a representação de 2^n estados diferentes. Por exemplo, 16 bits permitem a representação de 2^{16} valores distintos, isto é, 65536. É essa a razão porque um inteiro de 16 bits sem sinal (i.e., *unsigned*) pode representar valores entre 0 e 65535. Outro exemplo é o do octeto, que designa um conjunto de oito bits e que pode representar 2^8, i.e., 256 valores inteiros, seja entre -128 e 127 (*octeto com sinal*) ou entre 0 e 255 (*octeto sem sinal*). O termo *byte* é frequentemente empregue para designar um octeto.

A representação binária é usualmente pouco conveniente para o ser humano que facilmente se perde na contagem e localização dos *bits*, especialmente se existir um número elevado de *bits*. Deste modo, os programadores usam frequentemente a base octal e a base hexadecimal como alternativa à representação binária. Estas duas bases são empregues por serem mais compactas e pela facilidade com que se consegue converter de uma representação para a representação binária e vice-versa.

A base octal tem um conjunto de oito símbolos para a representação de uma dada quantidade. Os símbolos são os dígitos compreendidos entre 0 e 7. No caso da linguagem C (e de muitas outras), um valor em base octal é representado com um zero à esquerda. Assim, o valor 0123 numa listagem de código C identifica o valor octal 123 e não o valor decimal 123. Na realidade, o valor 0123 corresponde ao valor 83 em base decimal. A conversão de octal para decimal pode ser feita somando-se as parcelas resultantes da multiplicação de 3 por 8^0, de 2 por 8^1 e de 1 por 8^2 obtendo-se o valor 3x1+2x8+8x8=83. Note-se que o uso do zero à esquerda para indicar que uma constante inteira está em base octal pode confundir o programador menos atento que poderá pensar tratar-se de uma constante em base 10 com um insignificante zero à esquerda.

Conforme já referido, o maior interesse da base octal é a facilidade de conversão para a respetiva representação binária e vice-versa. De facto, por ser composta por 8 símbolos, cada símbolo octal é representado em binário por três bits, pois três bits permitem gerar 8 valores distintos ($2^3 = 8$). Por exemplo, o símbolo octal 3 é representado em binário por 011, o símbolo 6 por 110. A Tabela 1 mostra a conversão entre octal e binário para os oito símbolos da base octal. Voltando ao exemplo anterior, o valor octal 123 (0123 na linguagem C) tem a representação binária 001.010.011, correspondendo à substituição dos símbolos da base octal pelos respetivos valores binários (os pontos empregues na representação binária destinam-se somente a simplificar a leitura). Importa referir que algumas linguagens de programação já suportam representação binária, usando o prefixo 0b antes da quantidade numérica. É o caso da linguagem Java (somente a partir da versão 7), na qual o valor binário correspondendo ao valor octal 0123 poderia ser representado como **0b**001010011.

Octal	Binário
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

Tabela 1: Mapeamento entre base octal e base binária

De modo similar à base octal, uma representação em base hexadecimal é facilmente convertida numa representação binária e vice-versa. A base hexadecimal assenta num conjunto de 16 símbolos, usando os algarismos 0 a 9 para a representação dos 10 primeiros símbolos e as letras de A a F para os restantes seis símbolos. Nas linguagens de programação, as constantes em base hexadecimal são identificadas pelo prefixo *0x*. Assim, retomando o exemplo anterior, 0x123 representa uma quantidade em base 16. A conversão de uma valor hexadecimal para base 10 consiste em somar as parcelas resultantes da multiplicação do símbolo mais à direita por 16^0 (símbolo designado como o *menos significativo*), da multiplicação do segundo símbolo mais à direita por

16^1, da multiplicação do terceiro símbolo mais à direita por 16^2 e assim sucessivamente. Para o caso do valor 0x123, obtém-se $3x16^0 + 2x16^1 + 1x16^2$, isto é, 3+32+256, ou seja o valor decimal 291. Usando uma notação frequentemente empregue, pode dizer-se que $(123)_{16}$ corresponde ao valor $(291)_{10}$.

A conversão de um valor hexadecimal para a representação binária equivalente processa-se de forma similar à conversão de um valor octal para binário, exceto que cada símbolo hexadecimal deve ser mapeado para um valor de 4 bits de acordo com a Tabela 2. O uso de 4 bits por símbolo decorre do facto que são necessário 4 bits para representar todos os 16 símbolos empregues na base hexadecimal (2^4=16). Aplicando-se a metodologia de conversão hexadecimal para binário ao exemplo 0x123, obtém-se a seguinte representação em binário: 0001.0010.0011.

Hexadecimal	Binário
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
A	1010
B	1011
C	1100
D	1101
E	1110
F	1111

Tabela 2: Mapeamento entre hexadecimal e base binária

A maior frequência de uso na programação da representação hexadecimal em relação à representação octal deriva do facto de um valor hexadecimal apresentar um tamanho que é sempre múltiplo de 4 bits. Essa característica possibilita que facilmente possa ser encontrado um valor hexadecimal com o mesmo número de bits de um tipo de dados inteiro. Por exemplo, para o caso de se pretender um valor inteiro com 16 bits, apenas é necessário garantir que a representação hexadecimal tenha 4 símbolos. Similarmente, para um valor de 32 bits, sabe-se que é apropriado um valor hexadecimal com 8 símbolos e assim sucessivamente. Adicionalmente, o formato hexadecimal é empregue para a representação de endereços, dado os endereços terem geralmente um número de bits que é uma potência de dois (8, 16, 32, 64, etc.). Deste modo, não surpreende que a linguagem C disponibilize através da função *printf* e do respetivo operador de formatação *%x*, a representação de um determinado valor em formato hexadecimal. Note-se que em alternativa ao operador %x, pode ser empregue o operador %X (maiúscula) que apresenta o mesmo resultado, exceto que usa maiúsculas para representar os símbolos hexadecimais entre A e F.

Especificação de campos de bits em estruturas

A linguagem C possibilita a declaração de campos binários em estruturas do tipo *struct*. Assim, é possivel declarar um ou mais elementos de uma *struct* como sendo um conjunto de bits. A manipulação do conjunto de bits assim definidos faz-se através do campo da *struct*. Considere-se o exemplo da *struct exemplo1* apresentado na Listagem 1, na qual estão declarados os campos campo01 e campo02, respetivamente com dois e quatro bits. O uso de um campo de bits é efetuado da mesma forma que qualquer outro campo da estrutura, especificando-se o nome do campo. No caso da Listagem 1 são atribuídos os valores 1 (em decimal, correspondendo a 01 em binário) e 0xA (em hexadecimal, correspondendo a 1010 em binário), respetivamente, aos campos *campo01* e *campo02*.

```
/* Exemplo: "bit_fields.c" */
#include <stdio.h>
typedef struct exemplo1{
      int campo_bit01:2;
      unsigned int campo_bit02:4;
      float valor_float;
}exemplo1_t;

exemplo1_t  exemplo;
exemplo1.campo01 = 1;
exemplo1.campo02 = 0xA;
printf("campo01=%d\n", exemplo1.campo01);
printf("campo02=%d\n", exemplo1.campo02);
```

Listagem 1: exemplo bit_fields.c

Importa notar que um elemento especificado como campo de bits deve ser obrigatoriamente declarado como sendo do tipo int (ou equivalentemente do tipo signed int) ou do tipo unsigned int. O número de bits definido para o campo condiciona os valores que lá podem ser armazenado. Assim, para o caso do campo01, os dois bits do campo permitem armazenar um dos conjuntos binários 00, 01, 10 ou 11. Adicionalmente, dado que campo01 é declarado com *int*, isto é, inteiro com sinal, os valores inteiros que o campo pode armazenar são o -2, o -1, 0 e 1. Por sua vez, o elemento campo02 tem espaço para quatro bits, pelo que lhe pode ser atribuído um valor hexadecimal desde que tenha somente um dígito, como é o caso do valor 0xA empregue na Listagem 1.

Bits e variáveis inteiras

Na linguagem C, o acesso ao nível do *bits* não está limitado a campos de bits definidos em *structs*. De facto, é possível efetuar operações envolvendo operadores binários em variáveis do tipo inteiro, sejam elas int, short, long ou mesmo char, independentemente de ser considerado o sinal ou não (signed/unsigned). A principal diferença entre o uso de um campo de bits e o uso de uma variável inteira reside no facto que uma operação binária numa variável inteira envolve todos os bits da variável, ao passo que num campo de

bits, apenas são afetados os bits do campo de *bits*. Assim, quando se efetua uma operação binária envolvendo, por exemplo, uma variável inteira sem sinal com 32 bits, é necessário considerar os efeitos da operação sobre os 32 bits que compõem a variável. É pois importante, quando se faz uso de uma variável inteira, ter em conta o número de bits da variável, algo que pode ser determinado multiplicando o resultado devolvido pelo operador *sizeof* por 8, dado que esse operador devolve o tamanho em octetos (*bytes*) da variável ou do tipo de dados que lhe é passado como parâmetro (Listagem 2). A norma C99 introduziu tipos de dados com tamanho explicitado, como é o caso do tipo int8_t que corresponde a um valor inteiro com sinal de 8 bits (i.e., um octeto) ou o uint16_t que tem 16 bits para guardar valores inteiros sem sinal (Open-STD, 2003). A norma C99 especifica ainda que os tipos inteiros explicitados se encontram definidos no ficheiro <inttypes.h>.

```
int var_a;
printf("nº bits 'int'=%d\n",sizeof(var_a)*8);
printf("nº bits 'short'=%d\n", sizeof(short)*8);
```

Listagem 2: exemplo sizeof.c

Conceito de transbordo

Por terem um número finito de bits, as variáveis do tipo inteiro apenas podem representar um número finito de valores inteiros compreendidos entre um valor mínimo e um valor máximo. Por exemplo, uma variável do tipo `uint16` apenas pode representar os valores inteiros do intervalo [0, 2^{16}-1], isto é, [0, 65535]. Assim, caso se pretenda guardar um valor maior do que aquele suportado pela variável, ocorrerá o que se designa por um transbordo, perdendo-se a parte mais significativa do resultado. A Listagem 3 exemplifica o que sucede quando se soma uma unidade à variável `transbordo_1` do tipo `uint16` (inteiro sem sinal de 16 bits) que foi previamente carregada com o máximo valor que suporta, isto é, 65535: o valor da variável passa para 0. A Listagem 3 mostra ainda o transbordo da variável `transbordo_2` que é do tipo `int16`, isto é, uma variável inteira de 16 bits com sinal, que pode ser empregue para representar os valores do intervalo inteiro [-32768, +32767]. Assim, quando se carrega a variável com o valor máximo (32767) e posteriormente se soma uma unidade à variável, o valor da variável passa a ser o valor mais negativo, isto é, -32768 (ver Listagem 4). A possibilidade de transbordo é algo ao qual o programador deve estar muito atento, pois usualmente provoca comportamentos erráticos da aplicação (Baraniuk, 2015).

```
/*
 * Exemplo: "transbordo.c"
* Compilar:
 * gcc -Wall -W -std=c99 transbordo.c -o transbor-
do.exe
*/
#include <stdio.h>
#include <inttypes.h>

int main(void){
    uint16_t transbordo_1;
    int16_t transbordo_2;
  printf("Nº de bits de 'unsigned short': %u\n",
                          sizeof(transbordo_1)*8);
 transbordo_1 = 65535;/* Carrega valor máximo */
    printf("valor de transbordo_1=%u\n",
                                  transbordo_1);
    transbordo_1++; /* transbordo! */
    printf("(após +1) transbordo_1=%u\n",
                                  transbordo_1);
    transbordo_2 = 32767;
    printf("valor de transbordo_2=%d\n",
                                  transbordo_2);
    transbordo_2++; /* transbordo! */
    printf("(após +1) transbordo_2=%d\n",
                                  transbordo_2);
    return 0;
}
```

Listagem 3: exemplo transbordo.c

```
Nº de bits de 'unsigned short': 16
valor de transbordo_1=65535
(após +1) transbordo_1=0
valor de transbordo_2=32767
(após +1) transbordo_2=-32768
```

Listagem 4: resultados da execução de transbordo.c

Operações binárias acessíveis na linguagem C

Por operação binária entende-se a operação que tem por operando(s) um ou mais valores que são tratados de forma binária, isto é, as operações decorrem bit a bit.

As operações binárias disponibilizadas na linguagem C correspondem às operações habituais de manipulação de bits que são: 1) negação; 2) "e" (conjunção); 3) "ou" (disjunção); 4) "ou exclusivo" (disjunção exclusiva); 5) deslocamento para a esquerda e 6) deslocamento para a direita. As operações binárias são usualmente executadas de forma muito eficiente pelo computador, pois muitos processadores implementam nativamente as operações binárias.

Detalham-se de seguida, as operações binárias anteriormente enumeradas.

Operador de negação

Como o nome sugere, a operação de negação consiste na troca bit a bit, sendo que um bit a 1 é convertido para um bit a 0, e vice-versa. Na linguagem C, a operação de negação (*not* na designação anglo-saxónica) é representada pelo operador ~ (tilde). O operador de negação é dito unário, porque apenas requer um operando. Na Listagem 5, o operador de negação binária é empregue para atribuir à variável out o resultado da negação do conteúdo da variavel in, isto é, a negação de 0x012345678, resultando no valor 0xfedcba98 conforme mostrado na Listagem 6.

```
/* Exemplo: "not_binario.c"*/
#include <stdio.h>

int main(void){
```

```
	unsigned int in = 0x01234567;
	unsigned int out;
	out = ~in;
	printf("in: %x\n", in);
	printf("out: %x\n", out);
	return 0;
}
```

Listagem 5: exemplo not_binario.c

```
	in: 1234567
	out: fedcba98
```

Listagem 6: resultado da execução de not_binario.c

Operador *AND binário*

O operador "e" binário é tambem designado por operador de conjunção. É ainda conhecido pela sua designação anglo□saxónica *and*, sendo identificado na linguagem C através do símbolo "&". O operador requer dois operandos. A tabela de verdade do operador (Tabela 3) mostra que uma operação de AND binário em que pelo menos um dos operandos é o bit 0 resulta sempre no resultado bit 0. Pelo contrário, se um dos operandos for bit a 1, então o resultado corresponderá ao bit do outro operando: será 1 se o bit do outro operando for 1 e 0 se o outro operando for 0. Essas características do *and* binário podem ser empregues para conhecer o valor de um determinado bit (*and* binário com um dos operandos a 1) ou para *zerar* um determinado bit (*and* binário com um dos operandos a 0). A Listagem 7 apresenta código onde é efetuada a operação *and binário* entre os valores numéricos 0x12 (0001.0010 em binário) e 0x0F (0000.1111). A operação produz o resultado binário 0000.0010 (0x02 em hexadecimal), correspondendo ao *and binário* entre cada bit homólogo dos dois operandos 0x12 e 0x0F.

***and* binário (&)**	**0**	**1**
0	0	0
1	0	1

Tabela 3: tabela de verdade do operador and (&)

```
/* Exemplo: "and_binario.c" */
#include <stdio.h>

int main(void){
	int a = 0x12; /* 0001.0010b, 18 base 10 */
	int b = 0x0F; /* 0000.1111b, 15 base 10 */
	int c;
	c = a & b;  /* and binario */
	/* 0001.0010 & 0000.1111 => 0000.0010 */
	printf("c = %d & %d => %x\n", a, b, c);
	return 0;
}
```

Listagem 7: exemplo and_binario.c

É importante distinguir o operador *and binário* do operador *and lógico* no contexto da linguagem C. Em termos de representação, o primeiro é representado pelo símbolo &, ao passo que o operador *and lógico* requer dois símbolos & (&&). No que respeita à funcionalidade, o operador *and lógico* (&&) trata os operandos como entidades lógicas, isto é, tendo um valor verdadeiro ou falso, usualmente designado de *booleano*, e não *bit* a *bit* como sucede com o operador *and binário*. Assim, por exemplo, na expressão if((a==0) && (b==2)){...}, a mesma será considerada verdadeira apenas se o valor da variavel a for 0 **e** se o valor da variável b for 2, isto é, se ambas as operações (a==0) e (b==2) tiverem valor lógico verdadeiro. Se qualquer uma das expressões for falsa, ou ambas, então o resultado do *and lógico* é falso. A Tabela 4 mostra a tabela de verdade do operador *and lógico*.

***and* lógico (&&)**	**Verd.**	**Falso**
Verd.	Verd.	Falso
Falso	Falso	Falso

Tabela 4: tabela de verdade do operador and lógico (&&)

A Listagem 8 efetua a operação de *and lógico* sobre as condições (a==0) e (b==2) atribuíndo o valor resultante da operação à variável inteira *result*. Da análise do resultado da execução do código (Listagem 9), verifica-se que, na linguagem C, o valor lógico verdadeiro é mapeado para o valor inteiro 1 (um), e o valor lógico falso para o valor inteiro 0 (zero). Esse mapeamento mantém-se mesmo com o aparecimento do tipo de dados bool_t com a norma C99.

```
/* Exemplo: "and_logico.c" */
#include <stdio.h>
int main(void){
	int a = 0;
	int b = 2;
	int result;
	/* Condicao verdadeira */
	result = ((a==0) && (b==2));
	printf("verdadeiro => %d\n", result);
	/* Condicao falsa */
	result = ((a==0) && (b==3));
	printf("falso => %d\n", result);
	return 0;
}
```

Listagem 8: exemplo and_logico.c

```
verdadeiro => 1
falso => 0
```

Listagem 9: resultado da execução de and_logico.c

Operador *OR binário*

O operador "ou" binário, corresponde à operação de disjunção. É ainda designado por "OR" binário, sendo representado na linguagem C através do símbolo da barra vertical "|" . A tabela de verdade do operador OR binário (Tabela 5) mostra que sempre que um dos operandos é o bit 1, o resultado final é o bit 1, independentemente do valor do outro operando. Por sua vez, o bit 0 é o elemento neutro do operador OR binário, dado que o resultado de um OR binário com um dos operandos a bit 0 é determinado pelo valor do outro

operando: 0 se o outro operando for bit a 0, e 1 se o outro operando for bit a 1.

Um dos usos do operador OR binário é a ativação de um bit, isto é, colocar a 1 um determinado bit. Por exemplo, o resultado da operação OR binário com o operando 0011, terá sempre os dois bits menos significativos à 1, independentemente do valor do outro operando. De facto, conforme anteriormente observado, sempre que um determinado bit dos operandos do operador OR binário é 1, o resultado do bit correspondente é também ele 1. O código *or_binario.c* (Listagem 10) exemplifica o uso de 0x003, ou seja 0000.0000.0011b, como operando no operador OR binário, originando um resultado cujos dois bits menos significativos têm o valor 1 (Listagem 11).

***or* binário (|)**	**0**	**1**
0	0	1
1	1	1

Tabela 5: tabela de verdade do operador or (|)

```
/* Exemplo: or_binario.c */
#include <stdio.h>

int main(void){
    int a = 0x003;/* 0000.0000.0011b, 3 base10 */
    int b = 0x120;/* 0001.0010.0000b, 288 base10 */
      int c;
      c = a | b;  /* or binario */
      /* 0000.0000.0011 | 0001.0010.0000
      => 0001.0010.0011 */
      printf("c = %d | %d => %d\n", a, b, c);
      return 0;
}
```

Listagem 10: exemplo or_binario.c

```
c = 3 | 288 => 291
```

Listagem 11: resultado da execução de or_binario.c

À semelhança do anteriormente visto para o operador *AND*, existe também na linguagem C um operador *OR lógico*, representado através de dupla barra vertical, isto é, ||. Com exceção da tabela de verdade (Tabela 6) que obviamente difere da tabela do *AND lógico*, tudo o anteriormente mencionado para o operador *AND lógico* se mantém.

***or* lógico (||)**	**Verd.**	**Falso**
Verd.	Verd.	Verd.
Falso	Verd.	Falso

Tabela 6: tabela de verdade do operador or lógico (||)

Operador *ou exclusivo* (xor)

O operador *ou exclusivo, ou xor* na designação anglo-saxónica (contração de *eXclusive OR*) é também conhecido por disjunção exclusiva. Na linguagem C, o operador XOR é representado pelo símbolo **^**. Conforme mostra a tabela de verdade (Tabela 7), a operação de XOR resulta no bit 1 se os dois operandos corresponderem a bits diferentes (i.e., um dos operandos é o bit a 1 e o outro o bit a 0). Caso ambos os operandos representem o mesmo bit, então o resultado da operação de *XOR* é o bit a 0.

***xor* (^)**	**0**	**1**
0	0	1
1	1	0

Tabela 7: tabela de verdade do operador xor (^)

Uma das aplicações do operador XOR binário é o cálculo de paridade de um determinado conjunto de bits. A paridade par de uma sequência de bits diz-se *par* se o número de bits a 1 na sequência é par, e *impar* se o número de bits a 1 na sequência é impar. A Listagem 12 apresenta código em linguagem C que calcula a sequência de paridade de uma sequência de sete inteiros.

```
/* Exemplo: xor_paridade.c */
#include <stdio.h>

int main(void){
      /* Vetor de 7 inteiros sobre os
      quais é calculada a sequencia de paridade */
      int i;
      int paridade; /* Sequência de paridade */
      int vetor_entrada[7];
      vetor_entrada[0] = 0x12;     /* 0001.0010 */
      vetor_entrada[1] = 0x02;     /* 0000.0010 */
      vetor_entrada[2] = 0x22;     /* 0010.0010 */
      vetor_entrada[3] = 0x00;     /* 0000.0000 */
      vetor_entrada[4] = 0xA0;     /* 1010.0000 */
      vetor_entrada[5] = 0xFA;     /* 1111.1010 */
      vetor_entrada[6] = 0x4D;     /* 0100.1101 */
  /* Sequencia de paridade esperada: 0010.0101 */
      paridade = vetor_entrada[0];
      for(i=1; i<7; i++){
           paridade = paridade ^ vetor_entrada[i];
      }
      printf("paridade=0x%x (hex)\n", paridade);
      return 0;
}
```

Listagem 12: exemplo xor_paridade.c

No exemplo apresentado, cada inteiro é representado por dois símbolos hexadecimais, considerando-se assim apenas 8 bits por inteiro (independentemente de cada inteiro ter 32 bits – os bits mais significativos para além do oitavo bit estão a zero). O cálculo da sequência de paridade resume-se a aplicar a operação de XOR de forma iterativa entre os sete elementos da sequência de entrada, usando-se para o efeito a variável paridade para guardar a sequência de paridade. Note-se que a sequência de paridade corresponde à sequência de bits que é necessário acrescentar à sequência de entrada para obter paridade par.

Entrada[i]	Representação binária
[0]	0001.0010
[1]	0000.0010
[2]	0010.0010
[3]	0000.0000
[4]	1010.0000
[5]	1111.1010
[6]	0100.1101
Sequência de paridade	**0010.0101**

Tabela 8: sequência de paridade

É importante observar que no exemplo apresentado se está a considerar as sequências de bits que ocorrem na vertical, calculando-se o respetivo bit de paridade. Por exemplo, uma das sequências é formada pelo bit mais significativo (relembre-se, o bit mais à esquerda) de cada um dos sete valores inteiros, corresponde à sequência 0000.110, sendo o bit de paridade o bit 0 por forma que a sequência de oito bits (sete mais o bit de paridade) tenha paridade par, isto é, um número par de bits a 1. A sequência de paridade é pois 0010.0101, ou equivalentemente, 0x25 em hexadecimal. A Tabela 8 mostra os sete conjuntos de bits e a respetiva sequência de paridade.

Embora a linguagem C não disponibilize o operador lógico XOR, a operação XOR entre valores lógicos pode ser obtida através do recurso aos operadores AND, OR e NOT, conforme mostrado na Equação 1.

a XOR b = (!a && b) || (a && !b) (Eq. 1)

Uma aplicação comum do operador XOR, especialmente em *assembler*, é o de zerar o valor de uma variável. Para o efeito, efetua-se o XOR da variável com ela própria (a = a xor a) levando a que o resultado final seja zero, pois a xor a é forçosamente zero.

Operador *deslocamento para a esquerda*

Como o nome sugere, os operadores de deslocamento efetuam o deslocamento de bits. Na linguagem C, o operador deslocamento para a esquerda tem a seguinte sintaxe: valor << n. O operador de deslocamento à esquerda efetua uma translação em *n* posições dos bits para a esquerda do valor especificado. A Figura 1 ilustra uma operação de deslocamento para a esquerda em um bit do valor 0001.0010, resultando no valor 0010.0100. Note-se que devido ao deslocamento à esquerda em 1 bit, o anterior bit mais significativo (bit mais à esquerda) é perdido, sendo acrescentado um bit a 0 para a posição do bit menos signficativo (bit mais à direita, representado a azul na Figura 1). Caso o deslocamento fosse de *n* bits para a esquerda, perder-se-iam os *n* bits mais significativos, sendo ainda acrescentados *n* bits a 0 como bits menos significativos.

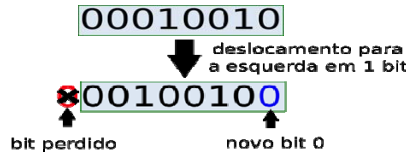


Figura 1: Exemplo de uma operação deslocamento para a esquerda em 1 bit

Quando efetuada sobre a representação binária de um número inteiro, a operação deslocamento para a esquerda em *n* bits produz um resultado final que corresponde à multiplicação por 2^n do valor inteiro original. Por exemplo, na Figura 1, o deslocamento em um bit para a esquerda do valor original 0001.0010 que corresponde ao inteiro 18 em base decimal, é transformado no valor 0010.0100 que corresponde ao valor 36 em base decimal, isto é, ao dobro do valor original. Esta propriedade do operador deslocamento para a esquerda é frequentemente empregue, especialmente em linguagens *assembler*, para efetuar multiplicações de valores inteiros por 2^n, pois é bastante mais rápida do que o algoritmo de multiplicação entre dois números inteiros.

A Listagem 13 exemplifica o uso do operador deslocamento para a esquerda. No exemplo, é aplicado a rotação à esquerda ao valor inteiro 1, usando-se um operando de deslocamento (variável *i*) que incrementa em cada iteração do ciclo *for*. Deste modo, na 1ª iteração do ciclo (i=0), o valor inteiro 1 não é deslocado, não sendo pois alterado. Na iteração seguinte (i=1), o valor 1 é deslocado em 1 bit para a esquerda, passando de 0...001 para 0...010, correspondendo ao valor inteiro 2. Na iteração seguinte (i=2), o valor inteiro 1 é deslocado para a esquerda em dois bits, resultando no valor 0...100, correspondendo ao valor 4. A Listagem 14 apresenta a saída gerada pela execução do programa. Facilmente se depreende que o código da Listagem 13 gera as sucessivas potências do número inteiro 2 (1, 2, 4, 8, 16, 32, 64, 128,...). Acresce-se ainda que os números inteiros potências de dois são frequentemente empregues como operandos dos operadores AND e OR pelo facto da respetiva representação binária comportar apenas um bit a 1, sendo os restantes 0. É frequente a designação de *máscara* para caracterizar um valor inteiro cuja representação binária tenha somente um bit a 1 ou, pelo contrário, somente um bit a 0.

```c
/* Exemplo: shift_left.c */
#include <stdio.h>
int main(void){
      unsigned int valor = 1;
      unsigned int valor_shift;
      size_t size_bits=sizeof(valor)*8;
      unsigned int i;
      for(i=0;i<size_bits;i++){
            valor_shift = valor << i;
            printf("[shift (valor << %02u)]%u\n",
                        i, valor_shift);
      }
      return 0;
}
```

Listagem 13: exemplo shift_left.c

```
[shift (valor << 00)]1
[shift (valor << 01)]2
[shift (valor << 02)]4
[shift (valor << 03)]8
[shift (valor << 04)]16
(…)
[shift (valor << 29)]536870912
[shift (valor << 30)]1073741824
[shift (valor << 31)]2147483648
```

Listagem 14: saída da execução do programa shift_left.c

Operador *deslocamento para a direita*

O operador deslocamento para a direita funciona de forma análoga ao operador de deslocamento para a esquerda, alterando-se somente o sentido do deslocamento. Assim, na operação de deslocamento para a direita em *n* bits, há lugar à deslocação em *n posições dos* bits para a direita. A Figura 2 ilustra uma operação de deslocamento para a direita. Na linguagem C, o operador deslocamento para a direita é representado por **>>**, e à semelhança do operador deslocamento para a esquerda requer dois operandos. Do lado esquerdo do operador fica o operando cujo valor irá ser alvo da operação de deslocamento para a direita. Por sua vez, o operando do lado direito indica de quantos bits deve o valor inicial ser deslocado.

valor_deslocado = valor_inicial >> num_bits;

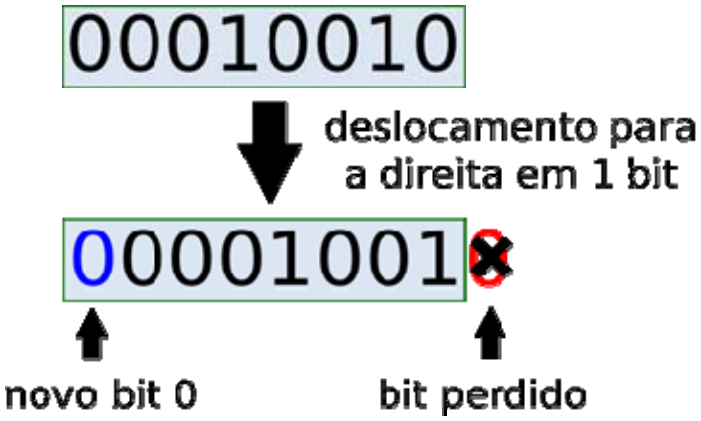


Figura 2: Exemplo de uma operação deslocamento para a direita em 1 bit

```
/* Exemplo: shift_right.c */
int main(void){
      int positive = 998;
      unsigned int sem_sinal = 998;
      int positive_shift_R;
      unsigned int sem_sinal_shift_R;
      int i;
      for(i=0; i < 4; i++){
            positive_shift_R = positive >> i;
            printf("===[i=%d]===\n", i);
            printf("positive_shift_R=%d\n",
                        positive_shift_R);
            sem_sinal_shift_R = sem_sinal >> i;
            printf("sem_sinal_shift_R=%d\n",
                  sem_sinal_shift_R);
      }
      return 0;
}
```

Listagem 15: Exemplo shift_right.c

```
===[i=0]===
  positive_shift_R=998
  sem_sinal_shift_R=998
===[i=1]===
  positive_shift_R=499
  sem_sinal_shift_R=499
===[i=2]===
  positive_shift_R=249
  sem_sinal_shift_R=249
===[i=3]===
  positive_shift_R=124
  sem_sinal_shift_R=124
```

Listagem 16: Saída da execução de right_shift.c

A Listagem 15 exemplifica a operação de deslocamento para a direita em duas variáveis de tipos diferentes. A variável sem_sinal é do tipo unsigned int, isto é, um inteiro sem sinal, ao passo que a variável positive corresponde a um inteiro com sinal (tipo int). Ambas as variáveis são inicializadas com o valor 998, sendo aplicada, sucessivamente, a ambas as variáveis a operação de deslocamento para a direita com 0, 1, 2 e 3 bits de deslocamento. A saída resultante da execução do código é mostrada na Listagem 16. Da análise da saída observa-se que a operação de deslocamento de *n* bits para a direita corresponde à divisão inteira por 2^n do valor inicial. Por exemplo, a operação de deslocamento para a direita em dois bits equivale à divisão inteira por 4 (2^2) do valor inicial. É contudo necessário ter em atenção que se trata de uma divisão inteira, perdendo-se a parte não inteira do resultado e que este comportamente, conforme veremos mais adiante, apenas é válido para operandos do tipo unsigned, isto é, sem sinal. Por exemplo, a divisão de 998 por 8 (2^3) é 124,75, mas quando se procede ao deslocamento em 3 bits para a direita (998 >> 3), obtém-se o valor inteiro 124. Recomenda-se pois cautela no uso do operador deslocamento à direita para efeitos de divisão por 2^n (Steele, 1977).

Uma outra limitação do operador >> envolve o bit mais à esquerda que deve ser acrescentado pelo operador quando ocorre um deslocamento para a direita. De facto, no caso de um valor inteiro representado em complementos de dois, o bit mais à esquerda (bit mais significativo) corresponde ao sinal do número: 0 indica número positivo, ao passo que 1 corresponde a um número negativo. Assim, a operação deslocamento para a direita não pode simplesmente acrescentar um bit zero em lugar do bit mais significativo, pois tal poderá resultar num valor com sinal diferente do valor inicial. No caso da linguagem C, não está definido qual o bit a ser inserido como bit mais significativo pelo operador deslocamento à direita quando lida com inteiros com sinal. Deste modo, o comportamento fica dependente do compilador empregue. No caso do código da Listagem 17, quando compilada com o GCC numa plataforma Linux verifica-se que o operador deslocamento à direita mantém o sinal da valor inicial conforme ilustram os resultados da Listagem 18.

```
/* shift_right_signed.c */
#include <stdio.h>
int main(void){
      int positive = 998;
      int negative = -998;
      int positive_shift, negative_shift;
      int I;
      for(i=0; I < 4; i++){
            printf("===[shift right %d]===\n",i);
            positive_shift = positive >> I;
            negative_shift = negative >> I;
      printf("positive_shift=%
                              d\n",positive_shift);
```

```
      printf("negative_shift=%d\n",negative_shift);
      }
      return 0;
}
```

Listagem 17: Exemplo shift_right_signed.c

```
===[shift right 0]===
positive_shift=998
negative_shift=-998
===[shift right 1]===
positive_shift=499
negative_shift=-499
===[shift right 2]===
positive_shift=249
negative_shift=-250
===[shift right 3]===
positive_shift=124
negative_shift=-125
```

Listagem 18: Saída da execução de right_shift_signed.c

Concretamente, na plataforma considerada, o operador deslocamento para a direita acrescenta um bit a zero se o valor inicial for não negativo, e um bit a um se o valor inicial for negativo. Importa observar, que para números negativos, e considerando que o operador deslocamento à direita aplica a persistência do bit mais significativo, a operação de deslocamento de *n* bits para a direita já não produz uma divisão por 2^n com truncagem. Por exemplo, a operação -998 >> 2 resulta, conforme mostrado na Listagem 18, no valor -250 e não no valor -249 como seria expectável pelo facto da divisão de -998 por 4 resultar em -249,5. Esta particularidade do operador deslocamento à direita tem causado erros em vários sistemas, nomeadamente compiladores conforme discutido por Steele Jr. já em 1977 (Steele, 1977). De modo a evitar da armadilha do operador de deslocamento para a direita, a linguagem Java disponibliza o operador deslocamento para a direita sem sinal, representado pelo símbolo >>>. Esse operador preenche sempre a posição do bit mais significativo com um bit a zero.

Casos de usos

Apresentam-se de seguida alguns dos casos de usos mais frequentes de manipulação binária.

Deteção do estado de um bit

A deteção do estado de um bit consiste em determinar o valor do $i^{ésimo}$ bit de um determinada sequência de bits. Para o efeito, faz-se uso do operador AND binário, tendo como operandos o valor que se pretende analisar e uma máscara binária. A máscara binária é inteiramente composta por bits a zero, exceto para o bit a um na posição do bit cujo valor se pretende detetar.

```
/*
 * Exemplo: mostra representação em
 * bits do valor inteiro da variável valor
 *
*/
#include <stdio.h>
#include <assert.h>

int is_bit_um(int valor, int num_bit){
      int num_bits_int = sizeof(valor) * 8;
      assert( num_bit < num_bits_int );
      int mascara_num_bit = (1 << num_bit);
      return ( valor & mascara_num_bit );
}
int main(void){
      int hex = 0xF0F1F2F3;
      int bit_i, i;
      int total_bits = sizeof(hex) * 8;
      printf("Conversão de 0x%X:\n",hex);
      for(i=total_bits-1;i>=0;i--){
            bit_i = is_bit_um(hex, i) ? 1 : 0;
            printf("%d",bit_i);
            if( (i % 4 == 0) && (i>0)){
                  printf(".");
            }
      }
      printf("\n");
      return 0;
}
```

Listagem 19: Exemplo mostra_em_bin.c

No programa mostra_em_bin.c (Listagem 19), a função is_bit_um devolve zero se o bit num_bit do parâmetro valor é zero e não zero (valor lógico verdadeiro) se o bit num_bit for um. Para o efeito, a função atribui à variável mascara_num_bit um bit a um na posição pretendida através da operação de deslocamento à esquerda, aplicando posteriormente a mascára através da operação de AND binário. No exemplo, a função é chamada sucessivamente para mostrar cada um dos bits da variável hex, disponibilizando assim a representação binária do valor da variável hex como ilustra a Listagem 20.

```
Conversão de 0xF0F1F2F3:
1111.0000.1111.0001.1111.0010.1111.0011
```

Listagem 20: Saída da execução de mostra_em_bin.c

Ativação/desativação seletiva de bits

A ativação seletiva de bits consiste em ativar, num determinado valor inteiro, um ou mais bits a um. Por sua vez, a desativação seletiva de bits corresponde à operação inversa, isto é, colocar um ou mais bits a zero.

A ativação seletiva de bits efetua-se através da operação de OR binário, usando como operandos o valor que se pretende modificar e uma máscara apropriada. A máscara deve ser constituída por bits a zero, exceto para os bits que se pretendem ativar, que devem estar a um. Por exemplo, caso se pretenda ativar os 4 bits menos significativos de um valor inteiro, deve-se empregar como máscara o valor binário que tenha os quatro bits menos significativos a 1111, estando os restantes a zero. Deste modo, e considerando um valor de 16 bits, a máscara corresponderá ao valor 0X000F, ativando-se os 4 bits menos significativos através da operação: novo_valor = valor | 0x000F.

A desativação seletiva de bits é conseguida através da operação de AND binária, usando como operadores, o valor que se pretende alterar e uma apropriada máscara binária. A máscara binária deve ser composta por bits a zero

nas posições que se pretendem desativar e bits a um nas restantes posições. Por exemplo, caso se pretendam desativar os 4 bits menos significativos de um valor de 16 bits, usar-se-á a máscara 0xFFF0, resultando na seguinte operação: novo_valor = valor & 0xFFF0.

Deteção de valores potências de dois

Determinar se o valor de uma determinada variável inteira sem sinal corresponde a uma potência de dois é uma operação trivial quando se recorre a operações binárias. De facto, dado que uma potência de dois tem um e só um bit a um (e.g., 16 que é 0001.0000 em binário), subtraindo-se uma unidade à potência de dois, obtém-se um valor que tem todos os bits à direita do bit ativo da potência de dois a um, e a zero o bit ativo bem como todos os bits à esquerda do bit ativo da potência de dois. Por exemplo, subtraindo uma unidade a 16 obtém-se 15, correspondendo a 0000.1111 em binário. Assim, para determinar se um determinado valor é uma potência de dois, basta efetuar uma operação de AND binário entre o valor e o valor menos uma unidade. Se o resultado for zero, o valor em apreço é uma potência de dois. É importante notar que este algoritmo só é válido para valores positivos. A função is_potencia_dois (Listagem 21) faz uso dessas propriedades das potências de dois para detetar se parâmetro valor corresponde ou não a uma potência de dois. A saída da execução do programa is_potencia_dois é mostrada na Listagem 22.

```
/*
 * Exemplo: operações binária para averiguar
 * se número positivo é potência de dois.
*/
#include <stdio.h>
int is_potencia_dois(unsigned int valor){
      if( valor == 0 || valor == 1 ){
            return 0;
      }
      return ((valor & (valor-1)) == 0? 1:0);
}
int main(void){
      unsigned int i;
      for(i=2;i <= (1<<10); i++){
            if( is_potencia_dois(i) ){
                  printf("%u\n", i);
            }
      }
      return 0;
}
```

Listagem 21: Exemplo is_potencia_dois.c

```
2
4
8
16
32
64
128
256
512
1024
```

Listagem 22: Execução de is_potencia_dois.c

Ativação de opções

Algumas funções da linguagem C requerem o uso da operação de OR binário por forma a que seja possível especificar múltiplas opções através de um parâmetro. Um exemplo é a função open que é empregue para a abertura de um ficheiro. Conforme mostra a Listagem 23, a função apresenta dois parâmetros. O primeiro corresponde ao caminho do ficheiro que se pretende manipular. Mais interessante para o âmbito deste artigo, é o segundo parâmetro, designado de *flags*, pois permite a especificação de vários elementos. De facto, a documentação da função open (e.g., man 2 open num sistema Linux) indica que podem ser especificada, entre outros, constantes para a criação de um ficheiro. Por exemplo, a criação de um ficheiro somente para escrita é especificada através de O_CREAT | O_WRONLY | O_TRUNC, isto é, especificando-se as opções O_CREAT, O_WRONLY e O_TRUNC através do operador OR binário. O valor que é efetivamente recebido pela função open corresponde pois ao resultado da operação de OR binário das três constantes. Na prática, as três constantes são potências de dois, significando que cada uma apenas têm um bit ativo. Tal é confirmado pelo programa open_flag.c (Listagem 24) que mostra o valor numérico das constantes O_CREAT, O_WRONLY e O_TRUNC (Listagem 25). Deste modo, torna-se possível passar, através de um mesmo parâmetro, várias configurações, sendo cada configuração especificada por um ou mais bits. Contudo, é necessário ter em conta que esta metodologia de empacotamento em bits de configurações requer código do lado da função chamada para que essa possa identificar as configurações pretendidas pela função chamante.

```
int open(const char *pathname, int flags);
```

Listagem 23: protótipo da função open

```
/*
 * Mostra o valor numérico de algumas das
 * constantes
 * que podem ser empregues pela função open
 */
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <stdio.h>
int main(void){
      printf("O_WRONLY = %X\n", O_WRONLY);
      printf("O_CREAT = %X\n", O_CREAT);
      printf("O_TRUNC = %X\n", O_TRUNC);
      return 0;
}
```

Listagem 24: Exemplo open_flags.c

```
O_WRONLY = 1
O_CREAT = 40
O_TRUNC = 200
```

Listagem 25: Saída do programa open_flags.c

Notas finais

A manipulação ao nível do bit é algo que os programadores da linguagem C devem conhecer por forma a tirar o melhor

partido da linguagem. Embora o seu uso explícito seja mais comum na programação sistema de baixo nível, o exemplo da função open ilustra que a manipulação ao nível de bit, embora de forma implícita, ocorre frequentemente na linguagem C.

> ***(...) a base binária é composta por dois valores distintos, representados por zero e um, daí também se designar por base dois (...)***

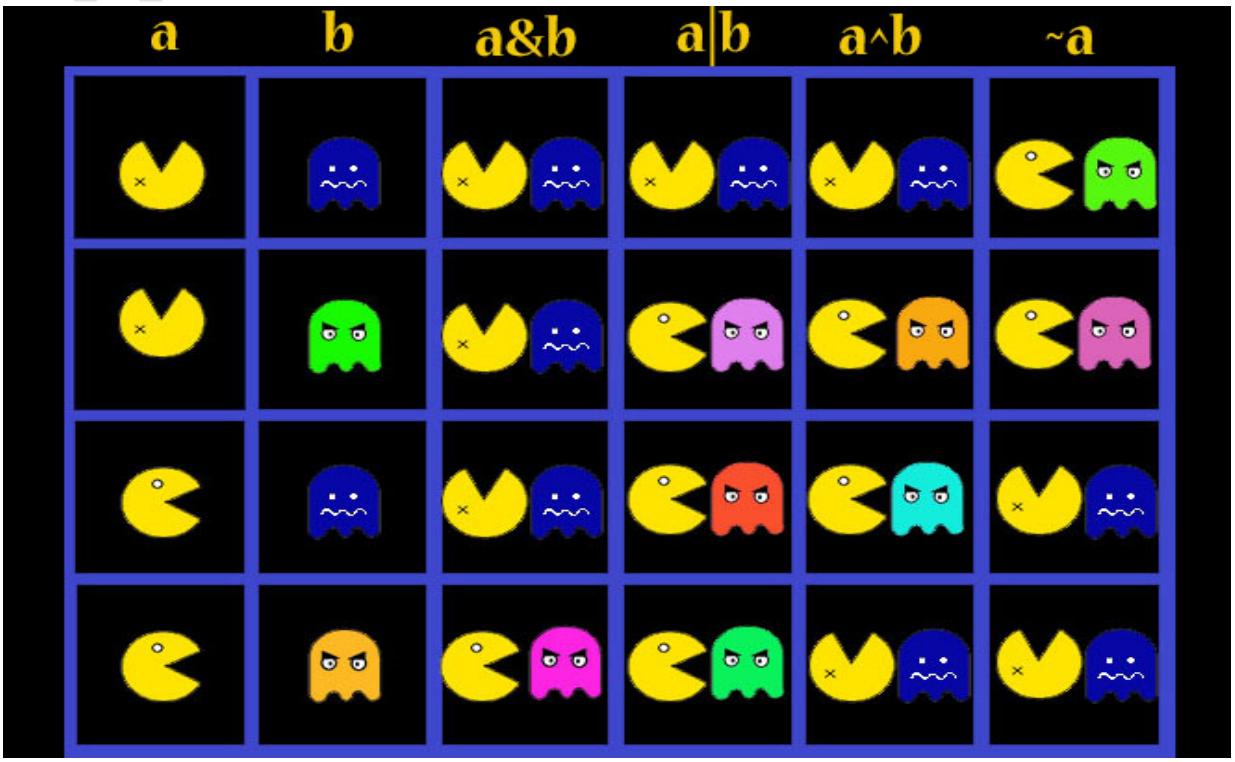
Para quem tem necessidade de recorrer à manipulação ao nível do bit, é ainda importante ter em conta os problemas, uns mais subtis do que outros, que podem ser encontrados. É exemplo disso o uso da operação de deslocamento à direita, cujo comportamento varia consoante o compilador e a plataforma que se está a usar.

Bibliografia

Baraniuk, C. (05 de 05 de 2015). *The number glitch that can lead to catastrophe.* Obtido de BBC: http://www.bbc.com/future/story/20150505-the-numbers-that-lead-to-disaster

Open-STD. (2003). *Rationale for International Standard - Programming Languages - C .* Obtido de http://www.open-std.org/JTC1/SC22/WG14/www/C99RationaleV5.10.pdf

Steele, G. L. (1977). Arithmetic shifting considered harmful. *ACM SIGPLAN Notices, 11*(12), 61-69. Obtido de http://dspace.mit.edu/bitstream/handle/1721.1/6090/AIM-378.pdf



AUTOR

Escrito por Patrício Domingues

doutorado em Engenharia Informática e professor do Departamento de Engª Informática na Escola Superior de Tecnologia e Gestão (ESTG) do Instituto Politécnico de Leiria (IPLeiria). Tem lecionado, entre outras, a disciplina de Programação Avançada da Licenciatura em Engenharia Informática. É ainda responsável pelo GPU Education Center (antigo NVIDIA CUDA Teaching) da ESTG/IPLeiria.

Reconhecimento de voz com JavaScript

Âmbito

Atualmente, o reconhecimento de voz tem várias aplicações no mundo real. O conceito de reconhecimento de voz está subjacente a softwares como o Siri e S-Voice. Esta aplicação pode melhorar drasticamente a usabilidade dos websites, principalmente para deficientes visuais. Por conseguinte, os utilizadores podem navegar pelas páginas ou preencher campos de formulário utilizando a sua voz.

Introdução

A API Web Speech é uma API de reconhecimento de voz que está implementada no Chrome 25 e superiores. A API Web Speech foi lançada no final de 2012 e fornece a entrada de voz e recursos de saída de texto para voz num web browser. Esta API tem em conta a privacidade dos utilizadores, pois antes de ativar a voz através do microfone, o utilizador deve explicitamente conceder a permissão. O pedido de autorização é o mesmo que a API getUserMedia, apesar de não necessitar da webcam. Se a página que executa esta API usa o protocolo HTTPS, o browser solicita a permissão apenas uma vez.

De seguida é apresentado um exemplo básico de como implementar esta API:

Primeira página com reconhecimento de voz

1. Estrutura HTML

A estrutura HTML é muito simples:

```
<p id="olá">Olá mundo!</p>
<div id="transcription"></div>
<button id="rect">Gravar</button>
<span id="unsuported" class="hidden">API
                              not supported</span>
```

O atributo **"transcription"** contém o texto que informa o que utilizador falou.

O botão **"rect"** é o botão utilizado para reconhecer a voz do utilizador.

O atributo **"unsupported"** é utilizado caso a API não seja suportada pelo browser.

2. Teste

A API Web Speech contempla um objeto chamado **"SpeechRecognition"**. Para saber se o browser suporta SpeechRecognition basta verificar se este objeto existe:

```
//Testa se o browser suporta
windows.SpeechRecognition =
                        Window.SpeechRecognition ||
          Window.webkitSpeechRecognition || null;

//Caso não suporte esta API de voz
if (window.SpeechRecognition === null) {
    document.getElementById
('unsuported').classList.remove('hidden');
} else {
    //......
}
```

3. Métodos e propriedades

Depois de testar o suporte e compatibilidade da API é criada uma instância do objeto "SpeechRecognition".

```
var recognizer = new window.SpeechRecognition();
```

Este objeto tem os seguintes métodos:

- **onstart:** Define um callback que é disparado quando o serviço de reconhecimento começou a ouvir o áudio para reconhecimento.
- **onResult:** Define um callback que é disparado quando o reconhecedor de voz devolve um resultado.
- **onerror:** Define um callback que é acionado quando ocorre um erro de reconhecimento de voz.
- **onend:** Define um callback que é disparado sempre que o serviço é desligado (o evento é gerado quando a sessão termina).

Por conseguinte foi criada uma variável com a função de exibir o texto que o utilizador falou. Para a API reconhecer a fala continuamente é necessário colocar a propriedade **"continuous"** como **"true"**. Esta propriedade faz com que o reconhecedor de voz não pare de ouvir, mesmo que tenha pausas do utilizador.

```
var transcription = document.getElementById
                                  ("transcription");

//Para o reconmhecedor de voz não parar de ouvir,
mesmo que tenha pausas do utilizador
recognizer.continuous = true;
```

Posteriormente existe a função **"onresult"** que define um callback que é disparado quando o reconhecedor de voz devolve um resultado.

```
recognizer.onresult = function (event) {
    transcription.textContent = "";
    for (var i = event.resultIndex; i <
                        event.results.length; i++) {
        if (event.results[i].isFinal) {
            transcription.textContent =
```

```
                    event.results[i][0].transcript +
                ' (taxa de acerto [0/1] : ' +
                  event.results[i][0].confidence + ')';
            } else {
                transcription.textContent +=
                        event.results[i][0].transcript;
            }
        }
}
```

Vamos analisar este código com mais detalhe:

A propriedade "**results"** é um array de objetos em que cada item do array contém um possível resultado do reconhecimento de voz. Para verificar se já é um resultado final utiliza-se a propriedade "**isFinal"** que é booleana.

"transcription.textContent" faz com que o texto dentro da **" "** seja limpo.

"for (var i = event.resultIndex; i < event.results.length; i++)" é o ciclo que percorre o evento que contém o texto que o utilizador falou.

Dentro deste ciclo há uma condição que verifica se o evento se encontra na última posição ("**event.results [i].isFinal"**). Caso seja verdadeira, é imprimido todo o texto, junto com a taxa de acerto, que varia entre "0" até "1". Caso seja falsa, é adicionado mais texto à div.

4. Evento de click

Por conseguinte, foi criado um evento de click associado ao botão:

```
document.querySelector("#rect").addEventListener
("click", function () {
    try {
        recognizer.start();
    } catch (ex) {
        alert("error: " + ex.message);
    }
});
```

Onde:

"recognizer.start()" inicia a gravação

e

catch(ex) {
alert("error: "+ex.message);
} faz o tratamento de log, caso exista, algum erro de gravação.

Conclusão

Esta foi uma breve explicação com a implementação da primeira página com reconhecimento de voz. Para o efeito foi utilizada a API Web Speech de JavaScript. Esta API é uma mais-valia porque facilita muito a integração de sistemas de reconhecimento de voz em sites.

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.

Cria o teu cliente de 9GAG em 15 minutos, com OutSystems

A OutSystems Platform é uma plataforma de desenvolvimento made in Portugal que te permite desenvolver aplicações web e mobile. As aplicações são programadas visualmente, e publicadas na cloud. Estes dois factores fazem com que consigas entregar as tuas aplicações aos utilizadores muito rapidamente.

Embora seja uma plataforma desenvolvida em Portugal, a OutSystems Platform está em grande crescimento a nível mundial e já é utilizada por algumas das maiores empresas como a Siemens, a Vodafone ou a Mercedes-Benz.

Neste tutorial vamos desenvolver uma aplicação web para memes que apresentará um meme de cada vez. Para obtermos os memes iremos utilizar o 9GAG através de uma API não oficial.

Antes de começarmos

Para podermos começar a desenvolver a nossa aplicação, precisamos de:

- Instalar o IDE da OutSystems Platform, chamado Service Studio. Este IDE é o que nos permite desenvolver e publicar a aplicação;
- Um ambiente na cloud para onde publicar a aplicação. A OutSystems oferece um ambiente na cloud, basta nos registarmos no site.

Se já tens o Service Studio instalado e um ambiente na cloud, podes saltar para a secção "Cria a aplicação". Caso contrário vamos tratar rapidamente disso.

Obter o IDE e um ambiente na cloud

Primeiro, vamos criar uma conta no site da OutSystems. Para tal acedes a www.outsystems.com/get-started e inseres os dados que são pedidos no formulário. Depois disso carrega em “Get Started”. Irás receber um email para ativares a tua conta.

Ao carregares em “Activate Your Account”, irá abrir uma página no teu browser para preencheres com mais alguma informação sobre ti. Apenas o url e a password para a tua conta são obrigatórios. Uma das informações que te irão ser pedidas será o url que vais querer usar para as tuas aplicações. Para este tutorial vamos usar “portugal-a-programar”.

Agora que já temos um ambiente na cloud, só nos falta instalar o Service Studio, o IDE que nos permite desenvolver as aplicações. Neste momento o Service Studio só está disponível para Windows.

Para obter o instalador, na barra lateral, carrega em “Start” e depois em “Download the Development Environment”. Depois de fazeres download, instala o Service Studio.

Agora temos tudo pronto para começar.

Cria a aplicação

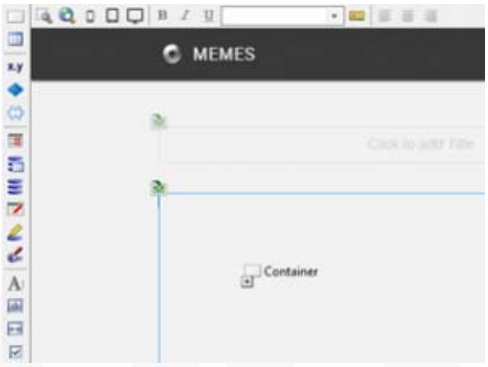
Abre o Service Studio. Cria uma nova aplicação e chama-lhe “Memes”.

Uma aplicação é constituída por módulos. Num módulo podemos definir o modelo de dados, implementar a lógica e desenhar a UI das nossas aplicações. Para simplificar, a nossa aplicação terá um único módulo. Cria o módulo e dá-lhe o mesmo nome da aplicação.

Desenha o ecrã

Agora que já temos a nossa aplicação criada, vamos tratar de desenhar o ecrã. O nosso ecrã terá a imagem do post do 9GAG e um botão para ir buscar um novo post.

Na tab “Interface”, seleciona o elemento HomePage e nas suas propriedades ativa a opção “Anonymous” para que não nos tenhamos que preocupar com a autenticação quando acedes ao ecrã. De seguida faz duplo clique no elemento “HomePage” para abrir o ecrã. Vamos adicionar um Container à página. Um Container é um Div em HTML. Arrasta da barra lateral esquerda o elemento *Container* para dentro da página.



De modo a centrar o elemento Image que colocámos dentro do Container, seleciona o Container e na propriedade "Align" escolhe "Center".

Agora vamos inserir a imagem no ecrã. Arrasta da barra lateral esquerda o elemento *Image* para dentro do container que criámos. Podes fechar a janela que te pede para selecionares a imagem a utilizar, visto que vamos carregar a imagem a partir de um URL.

Nas propriedades da imagem atribui à propriedade "Width" e "Height" o valor 500. Desta forma a imagem do post terá sempre o mesmo tamanho.

Falta-nos agora a opção para carregar um novo post.

Arrasta da barra lateral esquerda o elemento _____ *Web Block* e coloca-o por baixo do elemento da imagem. Os Web Blocks são elementos que podem ser reutilizados em vários ecrãs. Na janela que aparece logo depois de arrastares, procura por Icon e carrega "OK".

Nas propriedades do Icon, atribui à propriedade "Name" o valor "arrow_right" (disponível nas sugestões que te são apresentadas) e para a propriedade "Size" o valor "Size_4x", que permitirá aumentarmos o tamanho do Icon.

Vamos agora centrar o Icon, para tal seleciona o seu Container (carrega no Icon e de seguida no elemento Container que aparece na barra inferior do Service Studio _____) e à propriedade "Align" atribui o valor "Center".

Integra com o 9GAG

Para podermos obter os posts do 9GAG vamos usar uma API REST não oficial que podes encontrar na Internet (Infinigag - http://k3min.github.io/infinigag/).

No Service Studio, vai à tab "Logic", abre a pasta "Integrations" e na opção REST carrega com o botão direito do rato e escolhe "Consume REST API...". Na janela que aparece de seguida, vamos preencher o pedido HTTP que vai ser feito à API para ir buscar os posts. Preenche os campos sob "Method URL" com:

- GET, o verbo HTTP que vamos enviar no pedido;
- http://infinigag.eu01.aws.af.cm/{section}/{id}, o URL do método da AP

Precisamos também de inserir um exemplo da resposta JSON retornada pela API, para que o Service Studio crie as estruturas necessárias para recebermos a resposta. Copia o JSON disponível em http://pastebin.com/F7CNpYMz e cola no campo "Response".

Clica "OK". O Service Studio cria um novo método REST de nome "Get", com os parâmetros de input "section" e "id" que definimos no URL (dentro das chavetas), e com o parâmetro de saída "Response".

Implementa a lógica

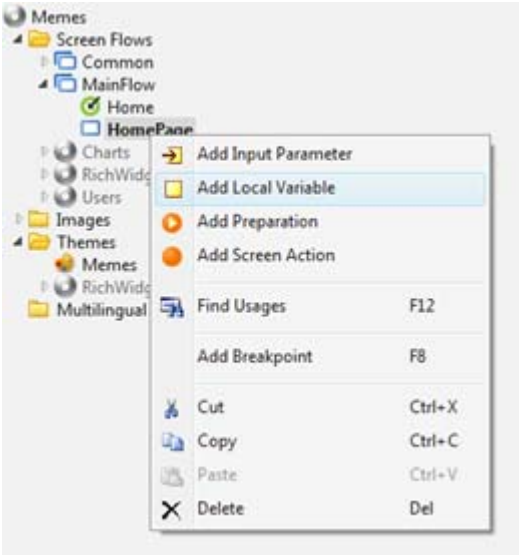
Agora precisamos de implementar a lógica para invocar o método da API que importámos, e usar o valor retornado para os elementos ecrã.

Para tal iremos usar uma _____ *Action*. Uma action é como um método em Java ou .NET. Na tab "Interface", vai a "Screen Flows">"Main Flow" e faz clique com o botão direito no ecrã "Homepage". Seleciona "Add Preparation". A ação Preparation é uma ação específica de um ecrã e que corre sempre que o ecrã é carregado.

De seguida, vai à tab "Logic" e em "Integrations">"REST" e arrasta para o fluxo da ação o método da API (de nome "Get"). Nos parâmetros de entrada do método coloca:

- section - "trending", para obtermos os posts mais falados no 9GAG;
- id – 0, para os posts mais recentes.

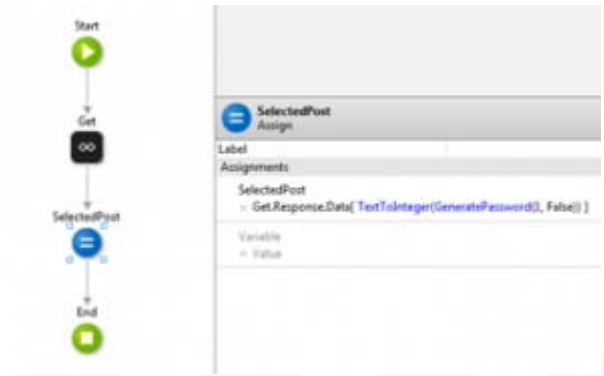
Precisamos agora de uma variável para guardar o post que vamos querer apresentar. Da mesma forma que fizémos para adicionar a ação "Preparation", sobre o ecrã "HomePage", faz right-click e seleciona "Add Local Variable".



Nas propriedades desta variável criada, dá-lhe o nome de "SelectedPost" e o "Data Type" com o tipo "DatumItem".

No fluxo, arrasta um elemento *Assign* e coloca-o por baixo do método da API. Na janela de propriedades do Assign coloca:

- Variable - SelectedPost, a variável local que criámos;
- Value - Response.Data[TextToInteger (GeneratePassword(1, false))], para aleatoriamente obtermos um dos posts da resposta do método da API.

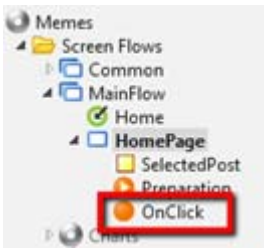


Agora que temos uma ação que retorna um post aleatório, só nos falta associar o dados do post selecionado ao ecrã, mais concretamente associar a imagem do post ao elemento Image.

De volta ao nosso ecrã "HomePage", clica no elemento Image. Define a propriedade "Type" como "External" e a propriedade "URL" com o valor "SelectedPost.Images.Large".

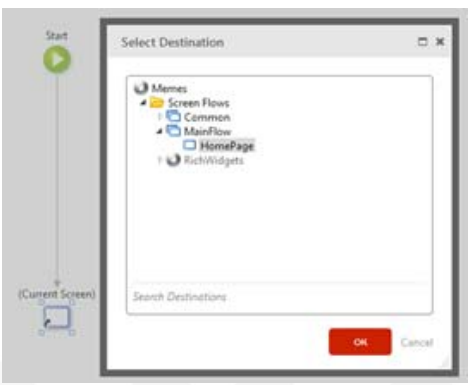
Neste momento, quando abrires a página já poderás ver a imagem do post. Mas antes de irmos experimentar, vamos apenas fazer com que carregar um novo post seja possível ao clicar no Icon que inserimos para o efeito.

Para tal, acede ao Container que tem o Icon (clica no Icon e na barra inferior do Service Studio seleciona o elemento Container) e na propriedade "Destination", escolhe a opção "(New Screen Action)". Esta opção irá criar uma nova ação chamada "OnClick", tal como a outra que criámos anteriormente, mas com o objetivo de ser executada ao carregarmos no Container onde está o Icon.



Vamos agora fazer com esta ação ao ser executada carregue um novo post. No nosso caso bastar-nos-á que este ação recarregue a página. Para tal, faz *double-click* sobre a ação "OnClick" (na ávore de elementos por cima das propriedades) e no seu fluxo adiciona o elemento da barra lateral esquerda *Destination* sobre o último elemento do fluxo.

Na janela que aparecerá de seguida, procura por "HomePage", o nosso ecrã e faz "OK". Desta forma iremos dizer que a última ação do fluxo será abrir um novo ecrã, que no nosso caso é o mesmo.



Feito isto, podemos publicar para ver o resultado no nosso tutorial. Para publicar, carrega no botão  na parte superior do Service Studio. Quando a tua aplicação estiver publicada, o icon muda para azul  . Clica nesse icon para acederes à aplicação que foi publicada.

Podes ver na aplicação web que é aberta no teu browser, a imagem do post e a opção para carregar um novo post.

Conclusão

Existem muitas outras funcionalidades que podes explorar e aplicações que podes desenvolver em OutSystems. Espero que com este tutorial tenhas percebido como é fácil desenvolver com a OutSystems Platform.

Segue o tutorial e alguma dúvida ou informação adicional não hesites em contactar-me. Podes obter o código e experimentar o resultado final deste tutorial (com a funcionalidade de poder fazer like/dislike num meme) em http://www.outsystems.com/forge/component/1025/Memes/.

> ***As aplicações são programadas visualmente, e publicadas na cloud. Estes dois factores fazem com que consigas entregar as tuas aplicações aos utilizadores muito rapidamente***

Referências

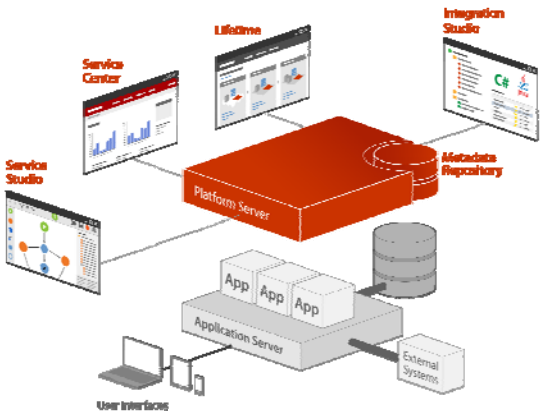
Site oficial da Outsystems: www.outsystems.com

Documentação oficial do IDE da plataforma: www.outsystems.com/help/servicestudio/9.0

API do 9Gag (não oficial): http://k3min.github.io/infinigag/

Site do 9gag: www.9gag.com

> ***Uma aplicação é constituída por módulos. Num módulo podemos definir o modelo de dados, implementar a lógica e desenhar a UI das nossas aplicações (...)***



AUTOR

Escrito por António Pereira

Mestre em Engenharia Informática e de Computadores pelo Instituto Superior Técnico (Lisboa, Portugal) e com certificação em gestão de projectos IPMA Nível-D e Associate Developer em OutSystems. Actualmente engenheiro de software na OutSystems. Curioso por natureza, procura sempre saber mais e adora trabalhar em equipa. Dotado de um conhecimento profundo em ferramentas de produtividade e de uma obsessão por livros, gosta de se dedicar a 100% a todos os desafios que abraça. Email: antonio.pereira@outsystems.com

Office Graph: A inteligência do Office 365

Recentemente apresentei uma sessão sobre "Office Graph" no **Microsoft Developer Tech Refresh**, em Lisboa. Tentando ter uma noção sobre o nível de conhecimento da audiência sobre os temas que ía abordar, perguntei quem conhecia o Office Graph e o Office Delve. Fiquei surpreendido por verificar que é um assunto relativamente desconhecido da maioria das pessoas. Na realidade, grande parte dos espetadores que ali estavam a ouvir-me não fazia a mais pequena ideia do que é o Office Graph e, por essa razão, pensei que seria uma boa ideia escrever um artigo introdutório sobre o tema. Vamos então começar pelo início...

O que é um grafo?

Começar pelo início implica explicar o que é um grafo (ou *graph*, em inglês). Um **grafo** é um conceito matemático que é também utilizado em computação como uma estrutura de dados, composta por **nós** (*nodes*) e **arestas** (*edges*). Cada nó representa algum tipo de entidade, e cada aresta uma relação entre duas dessas entidades. Cada aresta pode ainda ter uma direção e armazenar informação.

As redes sociais, como o Facebook, o LinkedIn ou o Yammer, utilizam este tipo de estrutura de dados para representar as relações entre pessoas e os conteúdos armazenados nas mesmas.

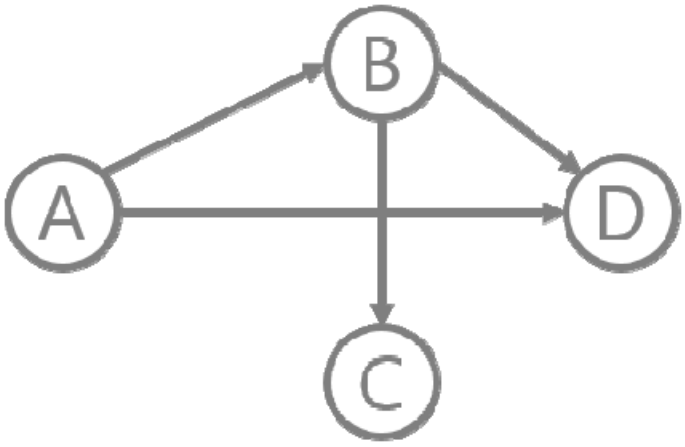


Diagrama de um Grafo

O que é o Office Graph?

É mais fácil de explicar o que é o Office Graph descrevendo o que é que este faz. Vamos focar-nos numa pessoa que utiliza o Office 365 nas suas atividades do dia-a-dia enquanto trabalha na sua empresa. Cada ação que esta pessoa realiza envia um sinal para o Office Graph que, depois de processado, pode originar uma nova relação entre dois dos seus nós.

Um sinal é enviado para o Office Graph sempre que:

- Eu abro um documento no SharePoint Online ou no OneDrive for Business;
- Um colega partilha um documento comigo;
- Um colega me envia um email;
- Um colega me apresenta um slide deck em PowerPoint;
- Eu realizo um "gesto social", ou seja, algo como pressionar o botão "like" num conteúdo do Yammer da empresa.

Mas não é apenas de sinais que se alimenta o Office Graph, este também consegue obter informação sobre a minha organização ligando-se ao Azure Active Directory (AAD) da minha organização e ao meu perfil de utilizador no SharePoint Online. A partir desta informação, o Office Graph constrói um mapa organizacional da minha empresa e passa a saber quem é o meu *manager* e quem é que reporta a mim.

Adicionalmente, o Office Graph utiliza mecanismos de aprendizagem (*machine learning*) para identificar automaticamente as pessoas com quem eu trabalho ativamente, baseado em quem partilha informação comigo e nas mensagens de correio eletrónico que eu troco com outros utilizadores da minha organização. Por exemplo, se eu recebo frequentemente emails do João e respondo imediatamente, o Office Graph vai inferir que eu trabalho proximamente com ele. Por outro lado, se eu recebo emails da Joana e não respondo ou demoro mais tempo a responder, o Office Graph vai inferir que a relação de trabalho não é tão próxima.

Office Graph - nós (entidades) e arestas (relações)

O Office Graph faz isto para mim e para todos os colegas que trabalham na minha organização, assumindo que eles também usam o mesmo *tenant* de Office 365. É por isto que o Office Graph é também apelidado de "cérebro do Office 365".

O que está guardado no Office Graph?

Atualmente, os nós do Office Graph são **Documentos** e **Pessoas** mas brevemente haverá novos tipos de nós. O **Profile** passará a viver no Graph, os **Groups** também terão os seus próprios nós assim como algumas das ações realizadas pelos utilizadores (que neste momento correspondem apenas a arestas).

Nos eventos Build e Ignite deste ano, a Microsoft partilhou algumas estatísticas impressionantes que nos permitem ter uma ideia da escala que o Office Graph atinge:

- O Office 365 armazena mais de **70 PetaBytes** de informação espalhada por todos os seus *tenants*. Isto corresponde a mais de 78.812.993.478.983.680 bytes!
- Foram enviados mais de **60 mil milhões de anexos** de email através do Exchange Online;
- Todos os meses são marcadas **850 milhões de reuniões** através do Exchange Online;
- O Office Graph tem atualmente mais de **4 biliões de nós** e **8 mil milhões de relações** entre nós (arestas);
- 25% de todas as relações são entre pessoas.

Esta escala só é possível na nuvem, onde o poder computacional e a capacidade de armazenamento são praticamente ilimitados. É também por isso que o Office Graph não está disponível *on premises*. No entanto, será possível utilizar uma abordagem híbrida e ligar uma *farm* de SharePoint Server 2016 *on premises* a um *tenant* de Office 365 para tirar partido do Office Graph numa organização. Mas isso terá que ser tema para outro artigo.

É seguro?

Uma questão importante que surge frequentemente quando se fala no Office Graph é a da **privacidade**. Nem todas as ações realizadas pelos utilizadores são públicas, algumas são privadas:

- Visualizar um documento é uma ação privada e não será partilhada com outros utilizadores. O número de vezes que eu abro um determinado documento é utilizado para calcular quão importante esse documento é para mim, e quão forte é a minha relação com o autor desse documento. No entanto, essa contagem nunca é partilhada com ninguém.
- O envio de emails também é uma ação privada, apenas disponível ao remetente e ao destinatário da mensagem de email, assim como qualquer anexo da referida mensagem.

No que respeita à **segurança**, o Office Graph respeita as permissões de acesso que foram definidas sobre cada conteúdo, independentemente de este estar armazenado no SharePoint Online, no OneDrive for Business ou no Exchange Online. Cada utilizador conseguirá ver apenas os conteúdos aos quais tem acesso.

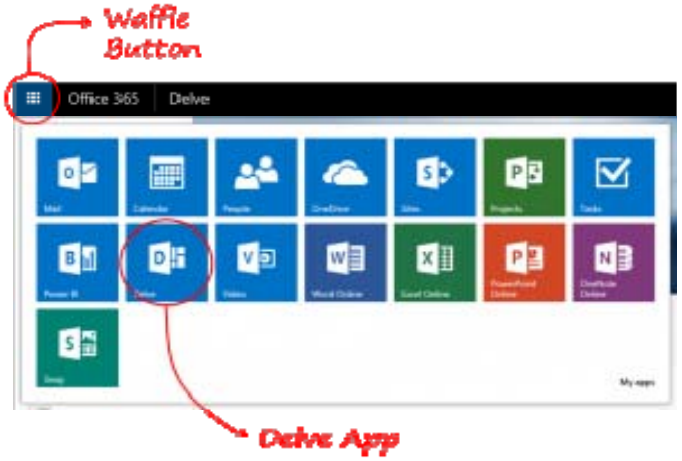
O que é o Office Delve?

O **Office Delve** é uma aplicação web, recentemente adicionada à família de aplicações que constitui o Office 365. Foi anunciada na SPC (SharePoint Conference) em 2014 com o nome de código Oslo (ou **Oslo Experience**). Começou por ser uma demo interna que mostrava como apresentar a informação armazenada no Office Graph e foi construída pela equipa da Microsoft de Oslo, na Noruega, que anteriormente era parte da FAST antes desta ser adquirida pela Microsoft.

O Office Delve é descrito como uma experiência de **Search & Discovery** (ou Pesquisa & Descoberta) porque, por um lado, permite-nos fazer pesquisas sobre pessoas e documentos através da introdução de termos numa caixa de pesquisa (a componente de Pesquisa mais tradicional). Por outro lado, apresenta-nos os conteúdos que nos interessam sem que seja necessário pesquisar por eles (a tal componente de Descoberta). Este é, aliás, o seu principal elemento diferenciador e faz com que o Office Delve funcione especialmente bem como a *homepage* personalizada do Office 365.

A Home Page

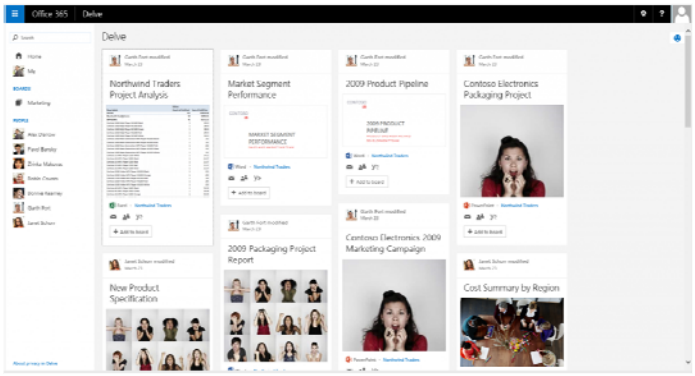
Para quem tem uma subscrição de Office 365, o acesso ao Office Delve é feito através do menu de aplicações do Office 365 (pressionando o botão do *waffle* no canto superior esquerdo) e clicando no ícone do Delve.



Menu do Office 365

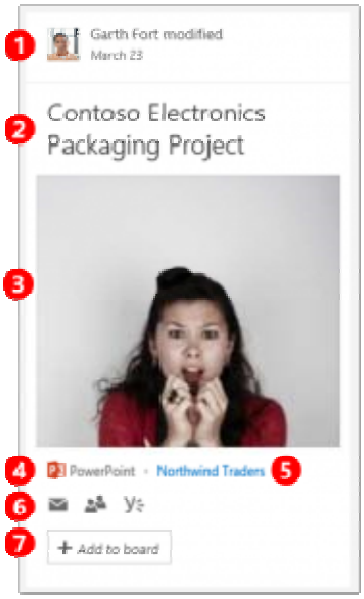
É imediatamente apresentada a **Home Page** que mostra os conteúdos que o Office Graph "pensa" que me interessam. Podem ser conteúdos que eu próprio editei, que

algum colega partilhou comigo ou que foram editados por um colega com o qual tenho interações frequentes.



Office Delve – Homepage

Do lado esquerdo, o Delve apresenta a lista de pessoas com as quais eu trabalho. São sempre utilizadores que pertencem ao mesmo *tenant* de Office 365, o que normalmente quer dizer que trabalham na mesma empresa que eu. Esta informação é automaticamente calculada com base nas trocas de emails, documentos partilhados, visualizados e editados.



Office Delve – Cartão

Cada conteúdo é apresentado sob a forma de um **cartão** contendo a seguinte informação:

1. A razão que justifica que o conteúdo esteja a ser mostrado
2. O título do conteúdo
3. Imagem extraída automaticamente do conteúdo
4. Tipo de conteúdo (ou aplicação associada a este)
5. Localização do conteúdo
6. Ícones para partilhar o conteúdo por email, para gerir permissões de acesso e para iniciar uma conversação no Yammer sobre o conteúdo
7. Botão para adicionar o conteúdo a um *Board*

A Me Page

A **Me Page** apresenta informação que está diretamente relacionada comigo. É composta por duas subpáginas: a Activity Page e a Profile Page.

- A **Activity Page** apresenta os conteúdos que eu editei pessoalmente ou que foram explicitamente partilhados comigo. Nesta página é ainda possível aceder a informação adicional sobre mim através da hiperligação para a **Trending Around Me Page** que, por sua vez, mostra as pessoas e os conteúdos que o Office Graph automaticamente inferiu que estão relacionados comigo, com base na minha utilização da plataforma.
- A **Profile Page** apresenta uma vista agregada do meu perfil de utilizador, com informação obtida a partir da *Active Directory* da minha organização e do perfil de utilizador do Office 365. O diagrama organizacional (organograma) é automaticamente calculado com base no valor da propriedade *Manager* de cada perfil de utilizador.

Boards

No Delve, um **Board** é um grupo de cartões de conteúdos, agregados através de uma **etiqueta** (*tag*). É um conceito com algumas semelhanças ao utilizado pelo Pinterest mas com algumas pequenas diferenças.

Os Boards são sempre públicos, o que quer dizer que qualquer pessoa consegue encontrar um Board que eu criei apenas pesquisando pela etiqueta que lhe está associada e poderá até adicionar-lhe os seus próprios conteúdos. No entanto, porque tudo no Delve se baseia na pesquisa, cada utilizador conseguirá ver apenas os conteúdos aos quais tem acesso. Isto significa que dois utilizadores que acedem a um determinado Board verão, provavelmente, conjuntos diferentes de cartões de conteúdos.

Interrogar o Office Graph

Dado todo o conteúdo armazenado no Office Graph, é natural que os *developers* tenham interesse em interrogá-lo e queiram usar todo este conhecimento nas suas próprias aplicações de negócio. Atualmente há duas formas de interrogar o Office Graph:

- Através de Graph Query Language (GQL)
- Através da Office 365 Unified API

Graph Query Language

A **Graph Query Language** (GQL) foi desenvolvida para ser utilizada pelo Office Delve e funciona sobre a API

REST da pesquisa. Utiliza uma sintaxe semelhante à FQL (FAST Query Language, a linguagem utilizada pela plataforma de pesquisa FAST e suportada também no SharePoint 2013) ou não tivessem ambas sido criadas pelas mesmas pessoas.

Antes de mergulhar no GQL, é necessário discutir alguns conceitos importantes:

- No Office Graph, todos os **nós** (*nodes*) representam uma entidade, como um documento ou uma pessoa, e cada nó é identificado por um número inteiro;
- Cada **aresta** (*edge*) representa uma ação entre dois nós e tem uma direção, ou seja, um nó de origem (chamado **actor**) e um nó de destino (chamado **object**);
- Uma aresta pode ainda ter informação adicional, como um *timestamp* e um **peso** (*weight*).

Em GQL existe apenas um operador - **ACTOR** - que é usado da seguinte forma:

```
ACTOR(<ActorId> [, filter])
```

O primeiro parâmetro é sempre o identificador do nó sobre o qual queremos fazer a *query*. Opcionalmente, pode ser adicionada uma expressão para filtrar os resultados.

Por exemplo, para obter todos os itens modificados por um utilizador específico, utilizaria a seguinte *query* GQL:

```
ACTOR(1234, action:1003)
```

Em que:

- 1234 é o identificador do utilizador (ou seja, do seu nó)
- 1003 é o código da ação "modified by"

Para interrogar o Office Graph é necessário injetar a *query* GQL num pedido à API REST da pesquisa, utilizando o parâmetro Properties tal como apresentado abaixo:

```
https://[URL  tenant]/_api/search/query?
QueryText='*'  &Properties='GraphQuery:actor(1234
\,action\:1033)' &SelectProperties='DocId,Title'
```

Para obter os itens relacionados com o utilizador autenticado, pode ser utilizada a *query* abaixo, em que a palavra **ME** é automaticamente substituída pelo identificador deste.

```
ACTOR(ME)
```

É possível ainda combinar mais do que uma expressão de filtro através de operadores lógicos. O exemplo abaixo permite obter todos os itens modificados ou visualizados pelo utilizador 1234.

```
ACTOR(1234, OR(action:1001, action:1003))
```

Como indicado acima, as arestas do grafo são caracterizadas por vários atributos. Um deles é o código da ação (action) mas existem outros como o time que, quando aplicável, representa a data e hora em que decorreu determinada ação representada pela aresta.

Para filtrar o grafo por este atributo, utiliza-se a *query* abaixo que retorna todos os itens modificados pelo utilizador autenticado no dia 15-08-2015.

```
ACTOR(ME, AND(action:1003, time:datetime
                          (2015-08-15)))
```

Como se pode verificar pelos exemplos apresentados, o filtro pela ação é o mais comum, mas requer que se saiba o código da ação pela qual se pretende filtrar o grafo. A tabela abaixo resume as ações e respetivos códigos.

Ação	ID	Descrição
PersonalFeed	1021	Feed do utilizador (actor) tal como mostrado na sua homepage no Delve. Privada.
Modified	1003	Itens modificados pelo utilizador nos últimos 3
OrgColleague	1015	Pessoas que reportem ao mesmo manager que o
OrgDirect	1014	Pessoas que reportam ao utilizador.
OrgManager	1013	A pessoa a quem o utiliza-
WorkingWith	1019	Pessoas com as quais o utilizador comunica ou colabora com frequência. Privada.
TrendingAround	1020	Itens populares junto das pessoas com as quais o utilizador comunica ou colabora com frequência.
Viewed	1001	Itens visualizados pelo utilizador nos últimos 3
WorkingWithPublic	1033	Versão pública da aresta WorkingWith.

Muitas outras *queries* podem ser realizadas com GQL. Para saber mais, visite este endereço: https://msdn.microsoft.com/en-us/office/office365/howto/query-Office-graph-using-gql-with-search-rest-api.

Office 365 Unified API

A nova **Unified API** expõe todas as APIs do Office 365 a partir de um único *endpoint*, oferecendo aos *developers* uma experiência mais robusta e consistente. A utilização de um único fluxo de autenticação também é muito mais simples do que realizar a autenticação separadamente para cada API do Office 365.

A Unified API pode ser utilizada para operações

CRUD (Create, Read, Update and Delete) sobre múltiplas entidades da plataforma Office 365, desde *Users* (utilizadores) e *Groups* (grupos) até *Files* (documentos) e *Mail* (mensagens de email), e até ao *Office Graph*.

Atualmente, a Unified API ainda está em *Preview* e algumas das operações ainda não estão disponíveis. No entanto, já é possível utilizá-la com qualquer *tenant* de Office 365 utilizando o *endpoint* com o endereço:

https://graph.microsoft.com/beta/

Além de ações relacionadas com *Users*, *Groups* ou *Files*, a Unified API fornece o acesso a duas ações muito importantes do Office Graph: **TrendingAround** e **WorkingWith**.

A ação **TrendingAround** retorna todos os nós relacionados com um utilizador específico, tal como inferido pelo Office Graph, e pode ser invocada da seguinte forma:

https://graph.microsoft.com/beta/me/trendingAround

A ação **WorkingWith** retorna a lista dos colegas que colaboram ativamente com um determinado utilizador, tal como inferido pelo Office Graph, e pode ser invocada da seguinte forma:

https://graph.microsoft.com/beta/me/workingWith

Para mais informação sobre como utilizar a Unified API:

- http://dev.office.com/unifiedAPIs
- https://msdn.microsoft.com/en-us/office/office365/howto/office-365-unified-api-overview

O que é que aí vem?

O Office Graph e o Office Delve são ainda tecnologias muito recentes, com muito para evoluir e, por estarem tão intimamente relacionadas, evoluirão juntas.

O roadmap do Office Graph inclui:

- A nova Unified REST API, atualmente em *preview*;
- Client SDKs para simplificar a integração e a autenticação com a nova API;
- Extensibilidade das entidades permitindo aos *developers* definir entidades próprias que são também armazenadas no Graph, tal como entidades de negócio;
- Conetores para permitir a outros sistemas como Dynamics CRM, Salesforce ou Trello, o envio de sinais para o Office Graph;
- Custom Analytics;
- Notificações sempre que determinadas ações são realizadas sobre o Graph;
- Licenciamento para a utilização do Office Graph.

No que respeita ao Office Delve, podem esperar:

- Uma nova Profile Page mais rica;
- Uma nova área para os Office 365 Groups com informação analítica;
- Uma nova área dedicada à organização com informação analítica.

Para saber mais...

Para quem está interessando em aprender mais sobre Office Graph, aqui ficam alguns links interessantes:

- **Aprender sobre Office Graph**

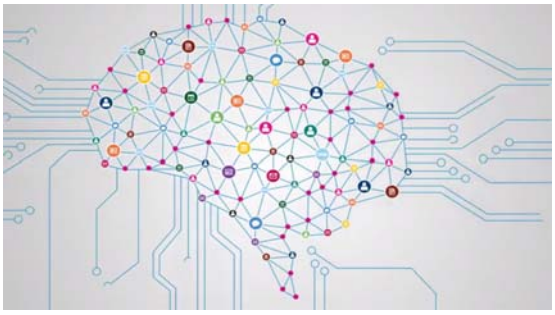
  http://dev.office.com/officegraph

- **Criar apps com Office 365 API e com a Unified API**

  https://msdn.microsoft.com/en-us/office/office365/api/api-catalog

- **Testar a Office Graph Preview API**

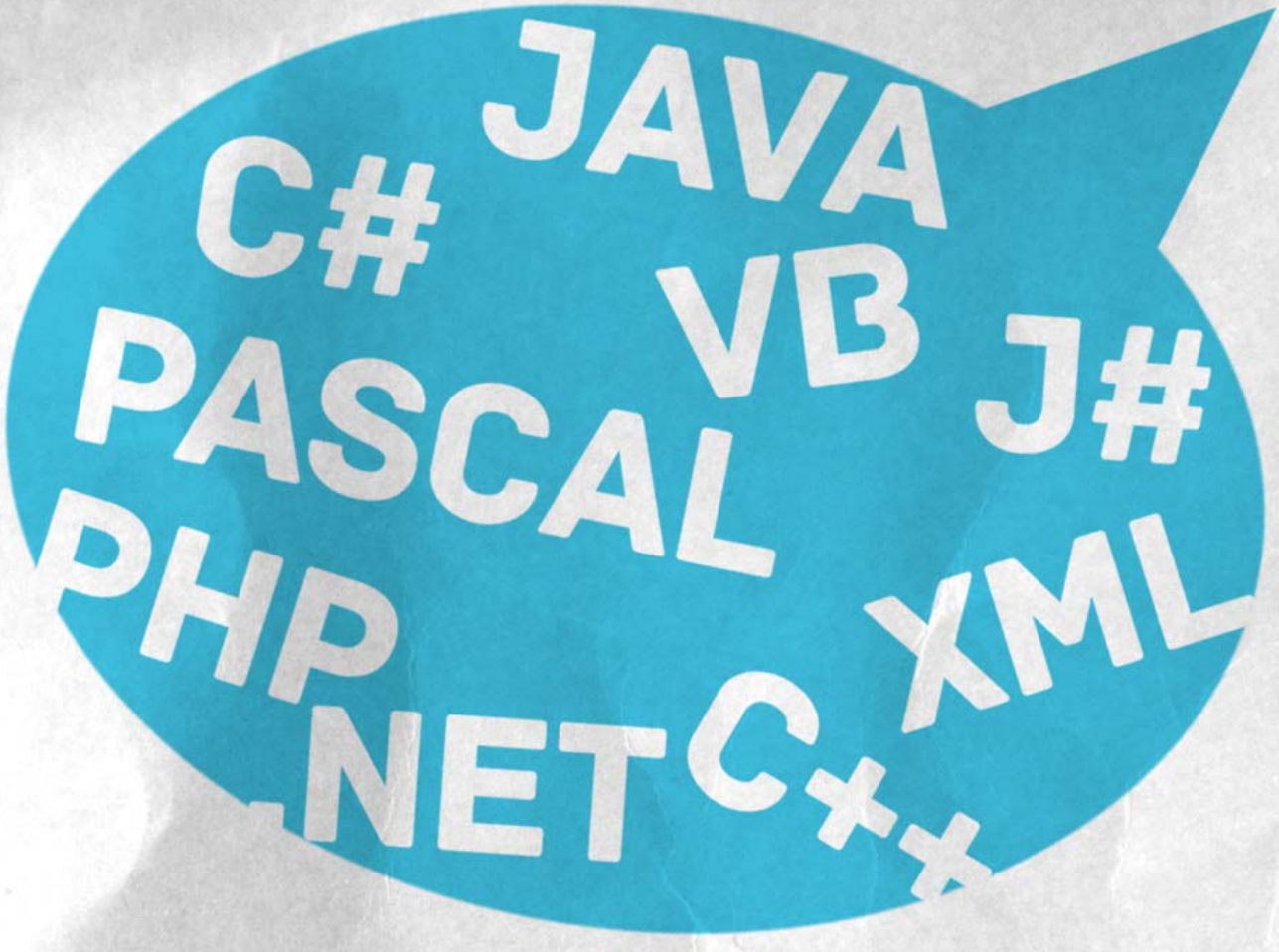
  http://msdn.microsoft.com/en-us/library/office/dn783218(v=office.15).aspx



AUTOR

Escrito por André Vala

Licenciado e Mestre em Engenharia Informática e de Computadores pelo Instituto Superior Técnico, é actualmente Arquiteto de Soluções SharePoint na |create|it| e co-fundador da Comunidade Portuguesa de SharePoint. Autor do blog http://blogit.create.pt/andrevala, trabalha com SharePoint desde 2006, altura em que surgiu a primeira versão beta do SharePoint 2007. Tem participado em vários projectos nacionais e internacionais sobre SharePoint, e participa frequentemente como orador em eventos da Microsoft relacionados com o mesmo tema.



ENTÃO, SÓ FALAS EM CÓDIGO?

TEMOS O REMÉDIO CERTO PARA TI!

portugal-a-programar.pt

A MAIOR COMUNIDADE PORTUGUESA DE PROGRAMAÇÃO, APARECE!

ELECTRÓNICA

Um “cofre” para passwords simples e de baixo custo!

UM "COFRE" PARA PASSWORDS SIMPLES E DE BAIXO CUSTO!

Introdução

Um dos mais comuns e mais falados problemas de segurança de um sistema de informação são as passwords sem "qualidade" muitas vezes motivadas pela dificuldade de memorização das mesmas.

De forma a enquadrar o leitor, cada password deve ter um comprimento adequado e preferencialmente não ser previsível. Por exemplo, uma password como "*1979aMelhorGeracaoDeSempre!*"(27 caracteres), é previsível se considerarmos que o utilizador nasceu em 1979 e possivelmente falará imenso desse facto gabando a sua geração. Neste caso, apesar de ser fácil de memorizar, é relativamente simples de "adivinhar", ou melhor deduzir, por parte de alguém que pretenda obter acesso ao sistema no qual o utilizador em causa usa esta password.

Por outro lado, passwords com qualidade como: "*zb8@g-DMK&7@%pRyhE45DhbbPs$! angSRhHNUenBpu4AZ4+$KLA-gcJFYfdwV=yN$RXw6TmD-YTpBf9?dWRkRAXu35XhwE=d*!vt53-m8dq34fmr? cCAv#k#u*gsSdgg*" (128 caracteres), apesar de serem praticamente impossíveis de deduzir, são demasiado complexas para serem memorizadas, tornando o seu uso difícil e praticamente inviável.

Quanto mais complexa for uma password, mais complexa será a sua memorização e mais tendenciosa será. Por exemplo contrariamente à anteriormente apresentada que tem 128caracteres, uma password grande em tamanho, definida por um metodo que não seja pseudo-aleatório, para memorização tenderá a ser um conjunto de caracteres segundo por exemplo uma cifra César.

Deve sempre existir uma razoabilidade entre o tamanho das passwords e aquilo que elas protegem. Os exemplos dados, são simplesmente para ilustrar a questão da dificuldade de memorizar passwords.

Existem diversos projectos que tentam resolver este problema; entre eles, um dos que me parece mais interessante e que recentemente se tornou comercialmente disponível, o Mooltipass, é baseado em arduino. Este projecto, tem diversas funcionalidades sendo a sua segurança elevada, na minha opinião, perde um pouco pelo elevado custo.

Com base no conceito de "cofre de passwords" e com a ideia de que o factor custo pode em muitos casos ser um problema, pensei em fazer o meu próprio XXVB"cofre de passwords", de baixo custo, que se comportasse como um teclado e fosse capaz de digitar qualquer uma das minhas passwords, sem que eu tivesse de as digitar manualmente nem das saber de cabeça! Passemos então, caro leitor, à prática…

Hardware

Existem centenas de microcontroladores disponíveis no mercado, uns mais caros, outros mais baratos, para todos os gostos e propósitos. Neste caso, usei e, recomendo o uso dos microcontroladores baseados em ATMega U32, como o caso do Arduino Leonardo, Yun, etc… . Esta escolha prende-se principalmente pelo suporte USB de que dispõem, permitindo que sejam tratados pelo sistema operativo como um teclado USB.

Neste caso usei um Arduino Leonardo R3, baseado no Microcontrolador ATmega32u4, com 20 pinos digitais de input/output, 7 canais PWM, 32KB de memória Flash (dos quais apenas 28 estão disponíveis, pois 4 são usados pelo bootloader), 2.5KB de SRAM, 1KB de EEPROM, e um peso relativamente baixo de 20g (o circuito completo pesará cerca de 70g).

Depois de alguma pesquisa e tendo em conta que era desejável um circuito de pequenas dimensões, baixo custo e uma interface de utilizador simples de usar, optei por adicionar ao Arduino, um LCD Keypad Shield, simples com 16 colunas por duas linhas, 6 teclas, com 8 caracteres programáveis pelo utilizador, de forma a ser possível "digitar" uma password e interagir com o dispositivo.

Programa

Escolhidos os componentes de hardware, o circuito é relativamente fácil de programar em C++, usando ou o ambiente de desenvolvimento do Arduino, ou outro, como o Visual Studio. Optei por fazer a reprogramação na linha de comandos usando um editor de texto (no meu caso o nano), e usando o platformio, para compilar e fazer upload do programa via interface de linha de comandos (CLI).

Como as passwords serão armazenadas na memória flash do arduino e considerando que estas devem ser mudadas com regularidade, o que implica compilar novamente o sketch do arduino e fazer o respectivo upload, o platformio é uma opção bastante boa para estas tarefas, sendo de instalação e utilização simples, cross-platform. Acima de tudo, é eficaz.

Instalação do Platformio

Para instalar o platformio deve-se ter instalado o Python 2.7, (atenção que não é compatível com as versões 3.x).

Em GNU/Linux e Mac OS X, a instalação segue os seguintes passos:

```
python -c "$(curl -fsSL https://
   raw.githubusercontent.com/platformio/
                                  platformio/
```

```
                master/scripts/get-
platformio.py)"
cd /path/get-platformio.py/script
python get-platformio.py
pip install https://github.com/platformio/
                platformio/archive/develop.zip
pip install platformio && pip install --egg scons
pip install -U platformio
pip install https://github.com/platformio/
                    platformio/archive/
develop.zip
```

Em Windows a instalação segue passos ligeiramente diferentes:

```
python.exe get-platformio.py
pip search platformio
pip install platformio && pip install --egg scons
pip install -U platformio
pip install https://github.com/platformio/
platformio/archive/develop.zip
```

Uma vez instalado o platfromio, será necessário instalar a plataforma atmelavr. Neste caso os passos são os mesmos quer se esteja a usar Windows, GNU/Linux ou MacOS X.

```
platformio install atmelavr
cd directorioDoProjecto
platformio init
```

Dentro desta directoria estará um ficheiro chamado platformio.ini, que devemos editar e colocar com as configurações correctas, de placa de desenvolvimento, plataforma, framework e porta.

```
#
# Project Configuration File
#
# A detailed documentation with the EXAMPLES is
                                    located here:
# http://docs.platformio.org/en/latest/
projectconf.html
#
# A sign `#` at the beginning of the line
                              indicates a comment
# Comment lines are ignored.
# Simple and base environment
[env:mybaseenv]
platform = atmelavr
framework = arduino
board = leonardo
upload_port = COM10
#
# Automatic targets - enable auto-uploading
targets = upload
```

No meu caso a porta correcta foi a COM10. Convém verificar qual a porta que está a ser utilizada pelo Arduino.

Para compilar e fazer o upload do sketch para o Arduino, utilizando o platformio, via CLI, basta digitar o comando:

```
platformio run --target upload
```

Voltemos ao desenvolvimento da nossa aplicação, que irá armazenar as passwords e "digitá-las" quando precisarmos.

Tal como escrevi no início do artigo, normalmente temos mais do que uma password diferente e essa será, de facto, a política mais segura. Uma vez que as passwords serão armazenadas num "cofre", será necessária uma só senha para acedermos a todas as senhas guardadas nele. No entanto precisamos de aceder a cada senha especificamente, para não inserirmos senhas trocadas, bem como para escolhermos que senha digitar. Para tal precisamos de um interface que nos permita escolher a senha. Neste caso será o LCD a mostrar a senha e o keypad vai permitir-nos escolher a senha.

Explorando um pouco o código:

No início do programa, por brincadeira e até nostalgia, criei alguns caracteres de 8x5 bits, dos quais apenas uso dois no início do programa, mas que não deixam de ter a sua graça. Caso o leitor deseje explorar um pouco mais bastará eliminar um dos existentes no programa e criar um novo, sendo o processo simples. Os bits a zero, significam que nada será exibido no lcd e os bits a 1 exactamente o oposto, como se ilustra no exemplo seguinte:

```
byte smiley[8] = {
  B00000,
  B10001,
  B00000,
  B00000,
  B10001,
  B01110,
  B00000,
};
```

Como o LCD Keypad tem apenas 6 teclas (Select; Left; Up; Down; Right; Rst), e dessas 6 apenas devem ser usadas 5, pois a tecla Rst está pré-definida para fazer o reset ao circuito, temos de programar a intercepção das restantes teclas, para lhes atribuirmos funcionalidades. Para tal começamos por definir os valores delas quando premidas:

```
//mapa de teclas e respectivos valores

/*
+----------+---------+
| Texto    | Pino    |
+----------+---------+
|   UP     |   0     |
|  DOWN    |   1     |
|  SELECT  |   2     |
|  RIGHT   |   3     |
|  LEFT    |   4     |
+----------+---------+
*/
#define UP 0
#define DOWN 1
#define SELECT 2
#define RIGHT 3
#define LEFT 4
```

De seguida criamos as funções destinadas à leitura das teclas se premidas. As teclas estão ligadas a pinos analógicos, logo temos de ler intervalos de valores, usando a função analogread(), para fazer a leitura dos mesmos.

```
int ReadKey()
{
```

```
  int x = 1023;
  do
  {
    x = analogRead (0);
    if (x < 60) return RIGHT;
    else if (x < 200) return UP;
    else if (x < 400) return DOWN;
    else if (x < 600) return LEFT;
    else if (x < 800) return SELECT;
  }
  while (x > 800);
}
```

Com estas funcionalidades básicas implementadas, optei por criar um array de chars com todos os caracteres "A-Z"; "a-z"; "0-9", para permitir que a password seja uma string do comprimento que o utilizador decidir. Também foi opção na programação das teclas usar a tecla "left" para acrescentar o caracter actual à password, permitindo assim escrever a password, normalmente, e a tecla Select, para "introduzir" a password, uma vez toda digitada.

Posto isto, basta apenas fazer um simples menu, como é explicado no próprio código, para se navegar entre as senhas armazenadas.

O código do scratch será o seguinte:

```
/*
*Thx to Mkman for the help and advice!
*In loving memory of
*       Misha II
* 12/12/2004 - 13/07/2015
*    You are missed
*/

#include <LiquidCrystal.h>
#include <String.h>

//cria alguns caracteres engraçados
byte smiley[8] = {
  B00000,
  B10001,
  B00000,
  B00000,
  B10001,
  B01110,
  B00000,
};
byte skull[8] = {
  B00000,
  B01110,
  B10101,
  B11111,
  B11011,
  B01110,
  B01010,
};
byte tulip[8] = {
  B10101,
  B11111,
  B11111,
  B01110,
  B00100,
  B10101,
  B01110,
  B00100
};
byte prompt[8] = {
  B10000,
  B01000,
  B00100,
  B00010,
  B00010,
  B00100,
  B01000,
  B10000
};
byte et[8] = {
  B11111,
  B10101,
  B11111,
  B00100,
  B01110,
  B11111,
  B11111,
  B11111
};
byte sandclock[8] = {
  B11111,
  B01110,
  B01110,
  B00100,
  B01110,
  B01110,
  B11111,
  B00000
};
byte cat[8] = {
  B01010,
  B11111,
  B10101,
  B11111,
  B00100,
  B01110,
  B01110,
  B11111
};

//define a senha para abrir o cofre
static String strkey = "abcddcba";
//Descrições das passwords
//São armazenado na memória flash (não volátil)
do arduino.
static char *desc[] = { "P@P", "Gmail", "MyApp",
                        "Blog", "FacelessBook"  };
//Passwords
static char *keys[] = { "Password1", "Password2",
        "Password3", "Password4", "Password5" };
//array de caracteres ascii com as letras a-z;A-
Z;0-9
static char ascii[] = {'a', 'b', 'c', 'd', 'e',
'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o',
'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y',
'z', 'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I',
'J', 'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S',
'T', 'U', 'V', 'W', 'X', 'Y', 'Z', '0', '1', '2',
        '3', '4', '5', '6', '7', '8', '9', '0'};
//mapa de teclas e respectivos valores

/*
+----------+---------+
| Texto    | Pino    |
+----------+---------+
|   UP     |   0     |
|  DOWN    |   1     |
|  SELECT  |   2     |
|  RIGHT   |   3     |
|  LEFT    |   4     |
+----------+---------+
*/
#define UP 0
#define DOWN 1
#define SELECT 2
#define RIGHT 3
#define LEFT 4
```

```
int index = 0;
#define COUNT 5 //numero de passwords e descrições
                //armazenadas

LiquidCrystal lcd(8, 9, 4, 5, 6, 7); //Inicializa
                                     //O lcd

/*
*função que tem como missão interceptar a leitura
analógica dos pinos das teclas up, down, left,
right e select
*/
int ReadKey()
{
  int x = 1023;
  do
  {
    x = analogRead (0);
    if (x < 60) return RIGHT;
    else if (x < 200) return UP;
    else if (x < 400) return DOWN;
    else if (x < 600) return LEFT;
    else if (x < 800) return SELECT;
  }
  while (x > 800);
}
/*
*funcao setup
*/
void setup()
{
  Keyboard.begin();
  lcd.createChar(0, smiley);
  lcd.createChar(1, skull);
  lcd.createChar(2, tulip);
  lcd.createChar(3, prompt);
  lcd.begin(16, 2); //16 colunas por 2 Linhas
  lcd.setCursor(0, 0);
  lcd.print("Lego Pwd Safe");
  lcd.setCursor(0, 1);
  lcd.print("by apocs");
  lcd.write(byte(0));
  lcd.write(byte(1));
  lcd.write(byte(2));
  delay(5000);
  lcd.clear();
  Unlock();
}

/**
funcao que tem por tarefa interceptar as teclas
primidas e permitir que seja composta a string da
password
*/
int inkeys()
{
  lcd.write(byte(3));
  int i = 0;
  char c;
  String str;
  while (1)
  {
    int key = ReadKey();
    delay(200);
    switch (key)
    {
      case UP:
        if (i >= 62)
        {
          i = 0;
        }
        c = ascii[i];
        lcd.print(c);
        //inc
        i++;
        //lcd.clear();
        break;
      case DOWN:
        if (i <= 0)
        {
          i = 0;
        } else if (i >= 62)
        {
          i = 0;
        }
        else {}
        i--;
        c = ascii[i];
        lcd.print(c);
        break;
      case LEFT:
        lcd.clear();
        break;
      case RIGHT:
        str += c;
        lcd.clear();
        break;
      case SELECT:
        Serial.println(str); //for debug porposes
        Serial.println(strkey); //for debug
                                //porposes
        return str.compareTo(strkey);
    }
  }
}
/**
funcao de bloqueio
*/
void blocked(void) {
  int i = 0;
  for (i = 30; i >= 0; i--) {
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Locked");
    lcd.setCursor(0, 1);
    lcd.print("");
    lcd.print("seg");
    delay(1000);
  }
}
/** funcao de desbloqueio
*/
void Unlock() {
  int count = 0;
  while (1) {
    if (inkeys() != 0)
      count++;
    else
      return;

    if (count == 2) {
      blocked();
      count = 0;
    }
  }
}

void loop()
{
  // put your main code here, to run repeatedly:
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Pass: ");
  lcd.setCursor(0, 1);
  lcd.print(desc[index]);
  int key = ReadKey();
  delay(200);
  switch (key)
```

```
  {
    case UP:
      --index;
      if (index < 0) index = COUNT - 1;
      break;
    case DOWN:
      ++index;
      if (index > (COUNT - 1)) index = 0;
      break;
    case SELECT:
      Keyboard.print(keys[index]);
      lcd.print(" - OK");
      delay(1000);
      break;
  }
}
```

Uma vez carregado o scratch no Arduino, bastará ligá-lo numa porta USB, que fará a dupla funcionalidade de o alimentar e de comunicar com o computador, permitindo assim que se utilize o circuito para introduzir passwords onde precisamos de as usar.

Cada vez que alteremos ou acrescentemos passwords ao "cofre", bastará alterar o valor da constante *COUNT* e/ou acrescentar ou alterar nos vectores de descrição e chaves, as respectivas descrições e chaves.

Conclusão

Como referido no início do artigo, recordar na nossa própria memória dezenas de passwords é extremamente difícil, pelo que o armazenamento externo das mesmas se torna interessante do ponto de vista prático, mas criando um problema de segurança. O armazenamento das mesmas num circuito externo pode ser uma solução, desde que tal se revele prático.

Ao longo deste artigo, expliquei como construir um circuito, baseado em Arduino Leonardo e LCD Keypad de 6 teclas, bem como compilar o código e fazer o upload via CLI. Certamente este código pode ser muito melhorado, como por exemplo a colocação das passwords no 1KB de EEPROM do Arduino ou o armazenamento encriptado das senhas, etc. Fica ao critério do leitor, modificar o código e o projecto caso assim o entenda. Neste caso esta solução não é 100% eficaz em termos de segurança pelo que não se recomenda o seu uso em ambientes mais sensíveis.

Em termos de hardware também podem ser feitas diversas melhorias, como por exemplo a ligação de módulos de memória externos, por exemplo um NXP PCF8570P ou uma EEPROM 24LC256, que são relativamente simples em termos de interface com o Arduino, ou até acrescentar funcionalidades de criptografia por hardware, por exemplo com um Atmel ATSHA204, que pode ser facilmente encontrado online por um preço bastante baixo, mesmo quando já montado numa breakout board, o que facilitará a ligação ao arduino.

Ficou também por explorar a possibilidade de construir uma caixa, para o circuito, possivelmente até feita numa impressora 3D em polímero ABS, ou outro. No entanto essa temática sairia muito do âmbito especifico deste artigo.

Outra possibilidade que não foi explorada neste artigo, foi a escrita de um driver não-standard para o dispositivo, que permitisse algumas funcionalidades extra. Neste caso foi uma opção para evitar usar drivers não padrão, uma vez que o dispositivo funciona nas principais plataformas "Microsoft Windows, GNU/Linux e Apple Mac OS".

Ficou igualmente por explorar o desenvolvimento de uma app com uma interface de utilizador simplificada, para o carregamento de novas passwords e descrições para o circuito! Inicialmente ainda pensei em incluir também esta temática no artigo, mas cedo decidi não o fazer deixando ao critério do leitor desenvolver a sua própria aplicação caso ache interessante. Pessoalmente eu fá-lo-ia usando as ferramentas Xamarin, uma vez que se trata de um sistema cross-platfrom, mas preferi deixar ao critério do leitor e/ou de um segundo artigo sobre este mesmo tema.

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.

Twitter:@apocsantos

COLUNAS

C# - As novidades do C# 6

AS NOVIDADES DO C# 6

Com o recente lançamento do Visual Studio 2015, foi lançada a versão 6 da linguagem de programação para a plataforma .NET **C#**.

Como neste lançamento o enfoque principal foi na nova plataforma de compiladores ("Roslyn"), os melhoramentos e adições à linguagem foram escassos mas, tal como os melhoramentos e adições das versões anteriores, tornarão a vida de quem desenvolve usando a linguagem de programação **C#** muito melhor.

1. Melhoramentos em auto-propriedades

As propriedades implementadas automaticamente (ou, abreviando, **auto-propriedades**) são propriedades não abstratas e não externas com acessores com corpo apenas com ponto e virgula.

Quando uma propriedade é implementada automaticamente, é criado um campo escondido para dar suporte à propriedade e os acessores de leitura e escrita são são implementados para, respetivamente, ler e escrever desse campo.

1.1. Inicializadores para auto-propriedades

Passa a ser possível declarar a inicialização de auto-propriedades da mesma forma que se inicializam os campos:

```
public class Person
{
    public string First { get; set; } = "Jane";
    public string Last { get; set; } = "Doe";
}
```

Com esta sintaxe, o inicializador inicializa diretamente o campo que dá suporte à propriedade sem recorre ao setter da propriedade.

Os inicializadores de propriedades são executados, tal como e juntamente com, os inicializadores de campos.

Tal como acontece com os inicializadores de campos, os inicializadores de propriedades não podem fazer referência a this porque, tal como acontece com os inicializadores dos campos, correm antes dos objetos estarem devidamente inicializados.

A implementação desta nova funcionalidade é feita usando funcionalidades tradicionais da linguagem tornado possível a utilização do código gerado em versões anteriores da plataforma .NET. Na verdade o código anterior é traduzido pelo compilador para o seguinte código C# 1:

```
public class Person
{
    [CompilerGenerated]
    [DebuggerBrowsable
                (DebuggerBrowsableState.Never)]
    public string k__BackingField = "Jane";
    [CompilerGenerated]
    [DebuggerBrowsable
                (DebuggerBrowsableState.Never)]
    public string k__BackingField = "Doe";

    public string First
    {
        [CompilerGenerated]
        get { return k__BackingField }
        [CompilerGenerated]
        set { k__BackingField = value; }
    }
    public string Last
    {
        [CompilerGenerated]
        get { return k__BackingField }
        [CompilerGenerated]
        set { k__BackingField = value; }
    }
}
```

Note-se que os campos k__BackingField e k__BackingField têm nomes que não são válidos em C#. Tal acontece para que não haja qualquer hipótese de colisão entre os atribuídos pelo programador e os nome atribuídos pelo compilador.

1.2. Auto-propriedades apenas de leitura

As auto-propriedades passam a dispensar o acessor de escrita passando, por isso, a poder ser apenas de leitura:

```
public class Person
{
    public string First { get; } = "Jane";
    public string Last { get; } = "Doe";
}
```

Neste caso, o campo gerado é declarado implicitamente como **readonly** (embora isto apenas tenha importância para efeitos de reflexão – *reflection*).

À semelhança do caso anterior, o código gerado será:

```
public class Person
{
    [CompilerGenerated]
    [DebuggerBrowsable
                (DebuggerBrowsableState.Never)]
    private readonly string k__BackingField =
                                        "Jane";
    [CompilerGenerated]
    [DebuggerBrowsable
(DebuggerBrowsableState.Never)]
    private readonly string k__BackingField =
                                        "Doe";
```

```
    public string First
    {
        [CompilerGenerated]
        get { return k__BackingField; }
    }
    public string Last
    {
        [CompilerGenerated]
        get { return k__BackingField; }
    }
}
```

Tal como acontece com os campos apenas de leitura, no caso das auto-propriedades apenas de leitura é possível inicializar o seu valor no construtor:

```
public class Person
{
    // ...

    public Person(string first, string last)
    {
        First = first;
        Last = last;
    }
}
```

E, mais uma vez, o compilador gera código equivalente a C# 1:

```
public class Person
{
    // ...

    public Person(string first, string last)
    {
        k__BackingField = first;
        k__BackingField = last;
    }
}
```

2. Membros função com corpo em formato expressão

Passam a poder ser usadas expressões semelhantes às funções lambda para definir corpos de funções com apenas de uma instrução (*statement*) ou bloco trazendo às funções membros de tipos a mesma clareza e simplicidade.

2.1. Corpos em formato expressão em membros do tipo método

Métodos, assim como operadores definidos pelo utilzador e conversões, podem ter o seu corpo definito por uma expressão usando a "seta das lambdas":

```
public Point Move(int dx, int dy) => new Point(x +
                                     dx, y + dy);
```

```
public static Complex operator +(Complex a,
                          Complex b) => a.Add(b);
```

```
public static implicit operator string (Person p)
                       => p.First + " " + p.Last;
```

O efeito é exatamente o mesmo que se os métodos tivessem apenas uma instrução de **return**. Os exemplos acima são convertidos pelo compilador para:

```
public Point Move(int dx, int dy)
{
    return new Point(x + dx, y + dy);
}
```

```
public static Complex operator +(Complex a,
                                 Complex b)
{
    return a.Add(b);
}
```

```
public static implicit operator string (Person p)
{
    return p.First + " " + p.Last;
}
```

Para métodos cujo tipo de retorno seja **void** (ou **Task** para métodos assíncronos) a sintaxe da seta ainda se aplica, mas a expressão que se segue tem de ser uma instrução (à semelhança do que já acontece com as lambdas):

```
public void Print() => Console.WriteLine(First + "
" + Last);
```

que será traduzido para:

```
public void Print()
{
    Console.WriteLine(First + " " + Last);
}
```

2.2. Corpos em formato expressão em membros do tipo propriedade

Os corpos do tipo expressão também podem ser usados para definir o corpo de propriedades e indexadores apenas de leitura:

```
public string Name => First + " " + Last;
```

```
public Person this[long id] => store.LookupPerson
                                        (id);
```

Note-se a ausência da palavra-chave **get**, que se torna implícita pela sintaxe de expressão.

Os exemplos anteriores são traduzidos pelo compilador para:

```
public string Name
{
    get
    {
        return First + " " + Last;
    }
}
```

```
public Person this[long id]
{
    get
    {
        return store.LookupPerson(id);
    }
}
```

3. Diretiva using static

À semelhança do que acontece com a diretiva **using** para espaços de nomes (*namespaces*), a diretiva **using static** adiciona os membros estáticos da classe ou enumerado dado como argumento ao espaço de nomes global, permitindo a sua utilização sem a necessidade de qualificação com o nome da classe:

```
using static System.Console;
using static System.Math;
using static System.DayOfWeek;
class Program
{
    static void Main()
    {
        WriteLine(Sqrt(3 * 3 + 4 * 4));
        WriteLine(Friday - Monday);
    }
}
```

O código anterior será traduzido pelo compilador para:

```
class Program
{
    static void Main()
    {
        System.Console.WriteLine(System.Math.Sqrt
                                 (3 * 3 + 4 * 4));
        System.Console.WriteLine
                (System.DayOfWeek.Friday - System.
                                 DayOfWeek.Monday);
    }
}
```

Esta funcionalidade é ótima quando se tem um conjunto de funções relacionadas com um determinado domínio que se usa frequentemente, de que System.Math é um bom exemplo. Permite também especificar individualmente os nomes de um enumerado, como os membros de System.DayOfWeek no exemplo acima.

3.1. Métodos de extensão

Os métodos de extensão são métodos estáticos, mas a intenção é de que sejam usados como métodos de instância dos tipos que estendem. Em vez de trazer esses métodos para o âmbito global, a funcionalidade **using static** faz com que esses métodos estejam disponíveis como métodos de extensão:

```
using static System.Linq.Enumerable; // The type,
not the namespace
class Program
{
    static void Main()
    {
       var range = Range(5, 17);// Ok: not
                                //extension
       var odd = Where(range, i => i % 2 == 1);
                            // Error, not in scope
       var even = range.Where(i => i % 2 == 0);
                                    // Ok
    }
}
```

Isto faz com que alterar um método para que passe a ser método de extensão passe a ser uma modificação fraturante, o que não era o caso anteriormente. Mas os métodos de extensão são geralmente chamados como métodos estáticos nos casos raros em que existe uma ambiguidade e, nesses casos, parece legítimo que sejam qualificados com o nome da classe.

4. Operadores condicionados por null

É frequente a necessidade de ter código salpicado de verificação para **null**. Os operadores condicionados por **null** permitem o acesso a membros e elementos apenas quando o recetor não é **null**, retornando um resultado **null** caso contrário:

```
int? length = people?.Length; // null se people é
                              //null
Person first = people?[0];    // null se people é
                              //null
```

O código anterior será traduzido para:

```
int? nullable = (people != null) ? new int?
                            (people.Length) : null;
Person person = (people != null) ? people[0] :
                                            null;
```

Os operadores condicionados por **null** pode ser muito conveniente quando usado com o operador de coalescência de **null** (**??**):

```
int length = people?.Length ?? 0; // 0 se people é
                                  //null
```

Os operadores condicionados pot **null** têm um comportamento de curto-circuito. em que a cadeia de acesso a membros, elementos ou invocações imediatamente a seguir apenas são executados se o recetor original não for **null**:

```
int? first = people?[0].Orders.Count();
```

O exemplo anterior é, na essência, equivalente a:

```
int? first = (people != null) ?
            people[0].Orders.Count() : (int?)null;
```

Com a exceção de que **people** é avaliado apenas uma vez. Nenhum dos acessos a membros ou elementos e invocações que se seguem ao operador **?** são executados se o valor de **people** for **null**.

E nada impede que os operadores condicionados por **null** sejam encadeados, no caso de ser necessária alguma verificação de **null** mais que uma vez na cadeia:

```
int? first = people?[0]?.Orders.Count();
```

A invocação (uma lista de argumentos entre parêntesis) não pode ser precedida imediatamente pelo operador **?** – isso levaria a demasiadas ambiguidades. Assim sendo, a esperada invocação de um *delegate* caso este não seja **null** não funciona. Contudo, o *delegate* pode sempre

ser invocado via o seu método **Invoke**:

```
if (predicate?.Invoke(e) ?? false) { … }
```

Uma utilização muito comum desta funcionalidade é o disparo de eventos:

```
PropertyChanged?.Invoke(this, args);
```

Que é traduzido para:

```
var handler = PropertyChanged;
if (handler != null)
{
    handler.Invoke(this, args);
}
```

Que é uma forma segura para *threads* de verificar se o evento tem subscritores porque apenas avalia o lado esquerdo da invocação uma vez e mantem o seu valor numa variável temporária.

5. Interpolação de *strings*

O método String.Format com as suas variadas versões é muito versátil e útil, mas a sua utilização é um bocado desajeitada e sujeita a erros devido aos marcadores numéricos ({0}) que têm de corresponder à posição dos argumentos fornecidos em separado:

```
var s = string.Format("{0} tem {1} ano{{s}}.",
                                    p.Name, p.Age);
```

A interpolação de *strings* permite substituir diretamente no literal *string* os índices por "buracos" com as expressões que correspondem aos valores:

```
var s = $"{p.Name} tem {p.Age} ano{{s}}.";
```

Tal como acontece com o método String.Format, é possível a especificação de alinhamentos e formatos:

```
var s = $"{p.Name,20} tem {p.Age:D3} ano{{s}}.";
```

O conteúdo dos buracos pode ser qualquer expressão, incluindo *strings*:

```
var s = $"{p.Name} tem {p.Age} ano{(p.Age == 1 ?
                                    "" : "s")}.";
```

Note-se que a expressão condicional está entre parêntesis, para que : "s" não seja confundido com o especificador de formato.

5.1. *strings* formatáveis

Quando não é especificado um provedor de formatação na invocação do método String.Format, é usada a cultura corrente do *thread* corrente e isso nem sempre é o desejado. Por isso. à semelhança do que acontece com as expressões lambda, o compilador traduz a *string* interpolada de forma diferente consoante o tipo do recetor da expressão.

Se o recetor da expressão for do tipol Formattable:

```
IFormattable christmas = $"{new DateTime(2015, 12,
                                        25):f}";
```

o compilador gera o seguinte código:

```
IFormattable christmas =
          FormattableStringFactory.Create("{0:f}",
                        new DateTime(2015, 12, 25));
```

que pode ser usado da seguinte forma:

```
var christamasText = christmas.ToString(new
                                CultureInfo("pt-PT"));
```

5.1.1. FormattableString

O tipo concreto retornado por FormattableStringFactory.Create é derivado de:

```
namespace System
{
    public abstract class FormattableString :
                                        IFormattable
    {
        protected FormattableString();
        public abstract int ArgumentCount { get; }
        public abstract string Format { get; }
        public static string Invariant
                  (FormattableString formattable);
        public abstract object GetArgument(int
                                            index);
        public abstract object[] GetArguments();
        public override string ToString();
        public abstract string ToString
                  (IFormatProvider formatProvider);
    }
}
```

Isto permite, não só acesso a formato mas também aos argumentos da *string* formatável.

5.1.2. Retrocompatibilidade

As funcionalidades introduzidas pelo **C# 6** são compatíveis com as plataformas .NET anterirores. No entanto, esta funcionalidade em particular necessita dos tipos System.Runtime.CompilerServices.FormattableStringFactory e System.FormattableString que só foram introduzidos na versão 4.6 da plataforma. A boa notícia é que o compilador não está preso à localização destes tipos numa determinada *assembly* e, caso se pretenda usar esta funcionalidade numa versão anterior da plataforma, basta adicionar a implementação destes tipos.

6. Expressões nameof

Ocasionalmente é necessário providenciar uma *string* com o nome de alguns elementos do programa:

- Quando se lança uma System.ArgumentNullException
- Quando se dispara um evento PropertyChanged.
- etc.

Usar literais *string* para isto é simples, mas sujeito a erros. Pode haver erros de escrito, ou uma refatorização do código pode ter mudado o nome do artefacto.

As expressões **nameof** são uma espécie de literal do tipo *string* em que o compilador valida a existência de algo com aquele nome. Uma vez que passa a ser uma referência ao artefacto, o Visual Studio sabe a que se refere e navegação e refatorização do código funcionarão.

No essencial, código como o seguinte:

```
if (x == null) throw new ArgumentNullException
                                        (nameof(x));
var s = nameof(person.Address.ZipCode);
```

será convertido em:

```
if (x == null) throw new ArgumentNullException
                                        ("x");
var s = "ZipCode";
```

6.1. Código fonte vs. metadados

Os nomes usados pelo compilador são os nomes do código fonte e não os nomes dos metadados dos artefactos, pelo que, o seguinte código:

```
using S = System.String;
class C
{
    void M<T>(S s)
    {
        var s1 = nameof(T);
        var s2 = nameof(S);
    }
}
```

é convertido em:

```
using S = System.String;
class C
{
    void M<T>(S s)
    {
        var s1 = "T";
        var s2 = "S";
    }
}
```

6.2. Tipos primitivos

Não é permitida a utilização de tipos primitivos (int, long, char, bool, string, etc.) em expressões nameof.

7. Métodos de extensão Add em inicializadores de coleções

Quando os inicializadores de coleções foram introduzidos na linguagem C#, os métods Add chamados não podia ser métodos de extensão. O Visual Basic acertou na sua implementação à primeira ao permitir a sua utilização, mas isso parece ter ficado esquecido para o C#.

Nesta versão a falha foi corrigida e é possível agora usar métodos de extensão Add em inicializadores de coleções.

Não é uma grande funcionalidade, mas é útil e, em termos de implementação do compilador, tratou-se apenas de remover a verificação da condição que o impedia.

8. Inicializadores de índices

A inicialização de objetos e coleções são úteis para inicializar declarativamente os campos e propriedades de objetos ou, no caso das coleções, um conjunto inicial de elementos.

A inicialização de dicionários, por outro lado, não era tão elegante, obrigando à existência de um método Add que recebesse como argumento a chave e o valor correspondente a essa chave. Se uma implementação em particular de dicionário não tivesse um método Add com as caraterísticas mencionadas, não seria possível usar um inicializador.

A partir de agora, passa a ser possível usar inicializadores em que são usados indexadores:

```
var numbers = new Dictionary<int, string>
{
    [7] = "sete",
    [9] = "nove",
    [13] = "treze"
};
```

que serão traduzidos para:

```
var dictionary = new Dictionary<int, string>();
dictionary[7] = "sete";
dictionary[9] = "nove";
dictionary[13] = "treze";
var numbers = dictionary;
```

9. Filtros de exceções

Os filtros de exceção são uma funcionalidade da CLR já disponibilizada pelo **Visual Basic** e pelo **F#** e que passa agora a estar disponível também no **C#**:

```
try
{
    ...
}
catch (Exception ex) when (SomeFilter(ex))
{
    ...
}
```

Se a avaliação da expressão entre parêntesis a seguir à palavra-chave **when** resultar no valor **true**, a exceção é apanhada. Caso contrário, o bloco **catch** é ignorado.

Isto permite que sejam definidos mais que um bloco **catch** para o mesmo tipo de exceção:

```
try
{
    //...
}
catch (SqlException ex) when (ex.Number == 2)
{
    // ...
}
catch (SqlException ex)
```

```
{
    // ...
}
```

No exemplo anterior o primeiro bloco **catch** apenas é executado se ocorrer um exceção do tipo SqlException em que o valor da propriedade Number seja 2. Caso contrário é executado o bloco seguinte.

É considerado aceitável e comum o "abuso" de filtros de exceções com efeitos colaterais, como *logging.*

ATENÇÃO: Os filtros de exceção são executados no contexto do lançamento da exceção (**throw**) e não do contexto do seu tratamento (**catch**).

10. await em blocos catch e finally

No **C#5** não era permitida a utilização da palavra-chave **await** em blocos **catch** e **finally** porque, na altura da implementação da funcionalidade **async-await**, a equipa pensou que isto não seria possível implementar. Mas agora descobriram que afinal não era impossível.

Passa a ser possível escrever código como este:

```
Resource res = null;
try
{
    res = await Resource.OpenAsync();
}
catch (ResourceException e)
{
    await Resource.LogAsync(res, e);
}
finally
{
    if (res != null) await res.CloseAsync();
}
```

11. Melhorias na resolução de sobrecarga de métodos

Foram introduzidas algumas melhorias na resolução de sobrecarga de métodos por forma a tornar mais expetável a forma como o compilador decide qual o método de sobrecarga a usar.

Onde isto se fará notar mais (ou deixar de notar) é na escolha de métodos de sobrecarga que recebam tipos valor *nullable*. Ou quando se passa um grupo de métodos (em vez de uma lambda) para métodos de sobrecarga que recebem *delegate*s.

> ***"(...) os melhoramentos e adições das versões anteriores, tornarão a vida de quem desenvolve usando a linguagem de programação C# muito melhor"***

Recursos

- New Language Features in C# 6
- C# 7 Work List of Features
- nterpolated Strings (C# and Visual Basic Reference)

AUTOR

Escrito por Paulo Morgado

Bacharel em Engenharia Electrónica e Telecomunicações (Sistemas Digitais) pelo Instituto Superior de Engenharia de Lisboa e Licenciado em Engenharia Informática pela Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa exerce variadas funções relacionadas com o desenvolvimento, distribuição e manutenção de software há mais de 10 anos. Participa em diversas comunidades nacionais e internacionais (pontoNETpt, NetPonto, SharePointPT, SQLPort, Portugal-a-Programar, CodeProject, CodePlex, etc.). Pelo seu contributo para com estas comunidades, a Microsoft premeia-o com o prémio MVP (C#) desde 2003. É ainda co-autor do livro "LINQ Com C#" da FCA.

Media Partners da Revista PROGRAMAR

Análises

Introdução ao Cloud Computing

Python – Algoritmia e Programação Web

Introdução ao Desenvolvimento de Jogos em Android

Introdução ao Cloud Computing

Título: Introdução ao Cloud Computing

Autores: António Miguel Ferreira

Editora: FCA - Editora de Informática

Páginas: 200

ISBN: 978-972-722-802-7

O livro *Introdução ao Cloud Computing* é um livro muito útil, não só para programadores, mas essencialmente, para todos aqueles que procuram um maior conhecimento e domínio no que ao tema do Cloud Computing diz respeito. Acredito igualmente, que este é extremamente elucidativo uma vez que, para além de executar uma abordagem histórica da tecnologia, apresenta-se como uma ferramenta de extrema importância para as pretensões tecnológicas das organizações, uma vez que, este livro, não só proporciona uma perspetiva realista e inovadora do que deve ser feito, como ainda, redireciona para a solução que melhor se enquadrará nas organizações.

Desta forma, um dos capítulos com mais interesse, principalmente devido à sua utilidade prática, que se deve muito ao facto de ser direcionado, essencialmente para leigos na temática, foi o capítulo 9 – Cloud para empresas. Considero igualmente útil e cativante, o capitulo 8 – Cloud para particulares. A abordagem e exemplos práticos, referidos ao longo da leitura, permite ao leitor percecionar o quanto o Cloud Computing faz parte do quotidiano de cada um. Aliás, quando olhei para o livro questionei se este seria direcionado para profissionais da área ou para leigos na temática. Após a leitura, reflexiono que este é direcionado e elucidativo para ambos. A leitura revela e demonstra, ao leitor, que afinal o Cloud Computing está "*all over*", tal como suprime os receios vulgarmente conhecidos dos leitores, que acreditam que a informação que guardam na Cloud, fica nas "nuvens", e que, como tal, não acarreta qualquer tipo de segurança. As "cerejas no topo do bolo", na minha perspetiva, são os capítulos 12 e 13, Benchmarking e Aplicações de Sucesso respetivamente. Estes capítulos são fulcrais para rematar quaisquer dúvidas sobre as vantagens na aplicabilidade do Cloud Computing e alcançar uma perspetiva futurista, pois este tema é o presente e certamente, será muito mais o futuro.

Em forma de conclusão a linguagem apresentada permite a todos usufruir da mais valia da informação que este livro nos fornece. Considero que, a introdução que o autor fornece no início de cada capítulo se torna essencial, e é uma ferramenta imprescindível, para que todo o capítulo faça sentido. Nesta pequena leitura, o leitor tem uma perspetiva do que será abordado, sendo esclarecido, "à priori" acerca de conceitos fundamentais para todo o entendimento da problemática que irá ser tratada. Por fim, é um livro muitíssimo completo, que nos fornece de forma clara, concisa e acima de tudo, descomplicada, uma visão da temática, permitindo assim, uma maior compreensão da mesma, a fim de garantir uma gestão e utilização mais rentável. Desta forma, acredito ser uma boa leitura de cabeceira, uma vez que nos trará um aprofundamento da realidade com a qual lidamos, e no fim subiremos à "Cloud" para um sono revitalizante.

> ***"Cloud Computing é um livro muito útil, não só para programadores, mas essencialmente, para todos aqueles que procuram um maior conhecimento e domínio no que ao tema (...)"***

AUTOR

Escrito por Ricardo Castro

Licenciado em Sistemas de Informação para a Gestão, tem na última década dedicado o seu tempo ao ensino superior e profissional. Com o mestrado em Ensino de TIC e doutorando em Educação – Ferramentas à Distância e elearning, dedica maioritariamente o seu tempo ao Instituto de Emprego e Formação Profissional, no entanto, desenvolve funções de consultor de Tecnologias de Informação em instituições públicas e privadas, sendo administrador de 5 plataformas LMS.

Python – Algoritmia e Programação Web

Título: Python - Algoritmia e Programação Web

Autores: José Braga de Vasconcelos

Editora: FCA - Editora de Informática

Páginas: 324

ISBN: 978-972-722-813-3

Para a review desta edição, chegou-me às mãos o livro Python Algoritmia e Programação Web de José Braga Vasconcelos.

Doutorado em Ciências da Computação pela Universidade de York (UK), José Vasconcelos, leva-nos, através deste livro, a novos conhecimentos.

Sendo o Python uma linguagem de programação que cada vez mais se afirma quer no mundo académico, quer no mundo empresarial, este livro pode ser uma boa aposta a todos os que querem iniciar-se nesta linguagem ou aprofundar conhecimentos.

Destinado a profissionais e a alunos das áreas das tecnologias de informação, pode também ser lido por todo o público em geral que se interesse pelo assunto.

A obra tem como principal objectivo apresentar as principais tecnologias e tendências da programação de aplicações Web utilizando a Linguagem Python.

A obra está dividida em 8 capitulos bem estruturados, que nos levam a aprofundar conhecimentos. É um livro de leitura fácil e pode tanto ser lido por aqueles que nunca utilizaram Python ou pelos profissionais que já tenham os seus conhecimentos cimentados no assunto.

Os principais capítulos desenvolvidos no livro:

- Linguagem de programação Python
- Algoritmos e estruturas de dados em Python
- Programação orientada a objectos em Python
- Arquitectura de aplicações web
- Tecnologias de programação web em Python
- Web Frameworks
- Projecto de software web

Após a leitura desta obra, o leitor terá competências para desenvolver as suas próprias aplicações web, utilizando os conhecimentos adquiridos ao longo do livro.

Partindo de princípios simples, como por exemplo, as estruturas de dados, somos ainda levados no terceiro capitulo, a aprender e/ou recordar alguns dos principais algoritmos académicos, como por exemplo como podemos implementar uma árvore de pesquisa (BST – Binary Search Tree), ou algoritmos de grafos (O autor aborda alguns algoritmos de grafos como o DFS – Depth-First-Search ou o BFS – Breath-First-Search).

No quarto capítulo somos levados pelo caminho das classes e herança, sendo que a parte do Python como linguagem de programação orientada a objecto não foi esquecida sendo claramente explicada neste livro.

A partir do quinto capítulo, as coisas “começam a aquecer” e iniciamos a descoberta da arquitectura de aplicações Web, o que permite aos leitores que ainda não estejam familiarizados com este ramo da tecnologia, possam adquirir todos os conhecimentos necessários à implementação das vossas próprias aplicações web.

No sétimo capítulo são abordadas as principais frameworks web desta tecnologia.

É principalmente abordada a framework Django. A Django utiliza o princípio DRY (Don't Repeat Yourself), onde faz com que o desenvolvedor aproveite ao máximo o código já feito, evitando a repetição. É também referenciada a API DOM (Document Object Model), uma biblioteca de software definida pela W3C, que nos permite analisar simultaneamente diferentes elementos num documento XML.

Chamo à atenção do leitor para o ultimo capítulo do livro, em que somos levados passo a passo a implementar uma aplicação web. Apartir daqui, munidos do conhecimento adquirido… o limite é a imaginação do caro leitor.

O autor disponibiliza ainda no site da FCA, todo o código relativo aos exemplos explicados no livro.

Em jeito de conclusão quero dizer-vos que este livro é um excelente material de estudo e uma boa leitura quer para quem está a iniciar na área das tecnologias de informação, quer para todos os profissionais com experiência.

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.

Introdução ao Desenvolvimento de Jogos em Android

Título: Introdução ao Desenvolvimento de Jogos em Android

Autores: Ricardo Queirós / Alberto Simões

Editora: FCA - Editora de Informática

Páginas: 274

ISBN: 978–972–722–807-2

"Móveis" são atualmente a plataforma mais emergente no que toca ao desenvolvimento de aplicações de *software*, no entanto os dispositivos móveis são orientados a duas vertentes. Uma delas orientada ao meio profissional, sendo os dispositivos móveis um excelente complemento ao trabalho profissional, mas por outro lado a área do entretenimento ocupa uma grande fatia da utilização dos dispositivos móveis na atualidade e é exatamente nesta área que se inserem os Jogos.

O livro "Introdução ao Desenvolvimento de Jogos em Android" vem no seguimento de dois outros títulos lançados por um dos autores, Ricardo Queirós, sendo eles "Android: Introdução ao Desenvolvimento de Aplicações" e "Desenvolvimento de Aplicações profissionais em Android", destinando-se aos profissionais da área do desenvolvimento de videojogos para plataformas móveis, assim como professores e alunos de disciplinas de computação móvel e desenvolvimento de jogos que necessitem de algum suporte teórico e prático.

Ao nível físico da edição não tenho nada de errado a apontar, sendo que a tipografia é de qualidade permitindo uma fácil e rápida perceção visual dos conteúdos abordados.

A organização dos conteúdos foi escolhida mediante a sua importância e dependência entre conteúdos, incluindo exemplos dos vários motores de jogos disponíveis e respetivas considerações.

Dividido em 6 grandes capítulos, as primeiras páginas do livro são orientadas à introdução ao desenvolvimento em Android onde são abordados alguns dos conteúdos básicos do desenvolvimento em Android, tal como a criação de um projeto, estrutura de ficheiros, interface gráfica, criação de um AVD (Android Virtual Device), execução da aplicação e seus componentes principais.

Além da introdução ao desenvolvimento em Android o livro aborda temas como API da Google e Motores de jogos, API 2D para Android, libGDX, Unity para Android e Google Play, Serviços e Publicação.

No capítulo 2 é feita uma exposição de vários motores de jogos disponíveis no mercado e que podem ser usadas para o desenvolvimento de jogos para as plataformas móveis, neste caso a plataforma Android. Além de abordada a API da Google, são destacados motores de jogos tais como, Unity, Cry Engine, Cocos 2D, Havok Vision Engine, libGDX, Marmalade, Game Salad, GameMaker Studio, Corona SDK entre outros.

Os capítulos 3, 4 e 5 abordam a API 2D para Android, libGDX e Unity respetivamente, onde aprofundam alguns dos conceitos fundamentais para desenvolver jogos usando estas frameworks, com exemplos teóricos e práticos que ajudam a compreender a mecânica do desenvolvimento de jogos.

Por fim o sexto e último capítulo é centrado no Google Play, Serviços e Publicação, onde é explicado e exemplificada a configuração dos serviços do Google Play, acesso à API, Leaderboards, Achievements e por fim a publicação na Play Store.

O conteúdo é de simples leitura e de fácil perceção, sendo que é usada uma linguagem muito "user-frendly" o que facilita muito a leitura da obra, não se tornando extremamente técnica do ponto de vista da linguagem utilizada.

Como estudante de Engenharia Informática e sobretudo como curioso e se assim posso chamar de "Indie Game Developer" considero o livro "Introdução ao Desenvolvimento de Jogos em Android" um livro de qualidade na apresentação dos conceitos introdutórios para quem quer iniciar-se no desenvolvimento de jogos para a plataforma móvel 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

No Code

Big Data: um conjunto de tecnologias imprescindíveis no futuro

Windows Hello: A autenticação biométrica no Windows 10

Windows 10 IOT Core no Raspberry Pi 2 B

BIG DATA: UM CONJUNTO DE TECNOLOGIAS IMPRESCINDÍVEIS NO FUTURO

Tecnologias como a Internet, computadores, *smartphones*, *tablets* e sensores estão a mudar o mundo em que vivemos. Por um lado a massificação destas tecnologias potencia a digitalização dos consumidores e das máquinas, por outro disponibiliza um maior volume de informação para análise, levando a uma melhor compreensão de hábitos, padrões e anomalias. Esta é a base do Big Data.

Na digitalização dos consumidores observamos que 2 em cada 3 pessoas com idades inferiores a 45 anos estão constantemente ligadas e a desenvolver interações digitais, através de múltiplos dispositivos e em múltiplos locais. O que implica uma alteração de hábitos de consumo de media, com os canais digitais (2.3 horas/dia) a ultrapassarem os antigos canais de media, nomeadamente televisão (2,2 horas/dia) e jornais/revistas (0,1 horas/dia), e um incremento da pegada digital (rasto de informação voluntária e involuntária) nos diversos sistemas que suportam todos estes serviços.

Em 2014 registaram-se valores, ao minuto, de 6 novos artigos no Wikipedia, 1.300 novos utilizadores móveis, 100 novas contas no LinkedIn, 20 milhões de fotos vistas no Flickr, 100 mil novos tweets, 2 milhões de pesquisas no Google, 6 milhões de consultas no Facebook e 1,3 milhões de vídeos colocados no YouTube.

A permanência dos consumidores no mundo digital obriga as organizações a estarem sempre presentes. Neste mundo digital onde os consumidores são bombardeados com inúmeras ações é preciso criar sistemas analíticos que analisem o contexto e personalizem as mensagens para os utilizadores com o objetivo de construir uma relação mais emocional.

O conceito de Big Data reúne as tecnologias, *frameworks* e infraestruturas necessárias para guardar, processar e analisar toda a informação existente no mundo digital (pegada digital). É na Google que surge este conceito que mais tarde evoluiu para várias definições (ver Wikipedia ou Gartner). Para a Accenture, o Big Data consiste na capacidade e no conjunto de competências que utilizamos para mobilizar e gerir um ecossistema com volumes grandes de dados, internos e externos, de forma a extrair informação e criar valor diferenciado.

Com o Big Data pretendemos gerar valor através do incremento de produtividade e eficiência, aumento de conhecimento baseado em dados, melhores retornos de investimento (ROI), e criação de novos e inovadores serviços de negócio. Observamos seis tendências para atingir estes objetivos: Explosão de Dados, Agregação de Dados, Tecnologia, Monetização de Dados, Redes Sociais e Mobilidade.

Na tendência "incremento de produtividade e eficiência" procuramos melhorar a inteligência operacional dos sistemas com grandes volumes de dados, conseguir responder em tempo real e potenciar a transparência através de uma melhor partilha de dados.

Relativamente ao "aumento de conhecimento" potenciamos uma visão abrangente das organizações através da análise de dados estruturados e não estruturados, construímos pontes entre silos e aumentamos as capacidades analíticas através de novos modelos que não são assentes apenas em amostras de dados.

Na vertente de "investimento", e para o otimizar, procuramos uma abordagem *open source* ao *storage* e processamento de forma a torná-los facilmente escaláveis, e pretendemos eliminar infraestrutura redundante e trabalho duplicado.

Por último, apoiamos a criação de novos e inovadores serviços de negócio assentes na recente capacidade de exploração, análise e visualização de dados.

Para suportar a execução destes objetivos, o ecossistema Big Data tem uma proliferação de tecnologias / *frameworks* que dividimos em grupos consoante os desafios que endereçam, nomeadamente processamento e *storage* distribuída, base de dados não relacionais com baixa latência, *streaming* e processamento de eventos complexos, processamento de múltiplos tipos de dados, processamento e base de dados em memória, base de dados analíticas e aplicativos.

No "processamento e *storage* distribuída" englobamos por exemplo fornecedores como Hadoop, Map Reduce, Cloudera, Hortonworks, IBM BigInsight e Amazon Elastic MapReduce Cloud. Promovemos escalabilidade horizontal de processamento e *storage* sobre uma variedade de diferentes infraestruturas, processamento de larga escala muito eficiente com uma arquitetura "*share nothing*" e TCO baixo devido ao *hardware* de baixo custo e *software open source*.

Recorremos a estas tecnologias para armazenar volumes grandes de dados no seu estágio original, evitando os custos de aquisição de licenças comerciais, implementar *sandboxes* de Business Intelligence de baixo custo, e construir sistemas de processamento de dados antes do Data Warehouse.

No segundo grupo que apelidamos de "base de dados não relacionais" contemplamos produtos como Key-Value, Column-Oriented, Document Databases, Cassandra, Riak,

BIG DATA: UM CONJUNTO DE TECNOLOGIAS IMPRESCINDÍVEIS NO FUTURO

MongoDB e Redis. Conseguimos com estas tecnologias o processamento de grandes volumes a velocidades altas, ultrapassando os constrangimentos dos sistemas de gestão de base de dados relacionais, a construção de arquiteturas de alta resiliência e a modelação dinâmica de perfis de dados.

Temos utilizado este tipo de tecnologias para construir aplicações dinâmicas de baixa latência que utilizam dados semiestruturados e aplicações web com personalização através de análise e atualizações em tempo real.

No terceiro grupo de tecnologias que apelidamos de "*streaming* e processamento de eventos complexos" enquadramos os fornecedores GemFire, Espertech, SenseiDB, Sensage, Zoie, IBM InfoStreams, uCIRRUS, Flume, Splunk e Sumologic. Estas tecnologias permitem a ingestão de dados em grande escala para *storage* e análise, consumo contínuo de grandes volumes de dados com pesquisas em tempo real, e consolidação de eventos em tempo real e a sua disseminação para um grande número de sistemas cliente.

Este tipo de tecnologia permite-nos implementar sistemas de criação de anúncios e promoções em tempo real para portais online e mobile, e desenvolver sistemas de resposta em tempo real a eventos com capacidade de adaptação a alterações de tipo e formato de dados.

No grupo a que chamamos "processamento de múltiplos tipos de dados" identificamos como produtos a MarkLogic, Neo4j e FlockDB. Este grupo é composto por tecnologias que se caracterizam por base de dados *Graph* para processamento de grandes volumes de dados em *Extensible Markup Language* (XML). Tecnologias que servem para implementar sistemas de otimização de caminhos e para a pesquisa e análise de relacionamentos complexos como *graph* sociais.

No conjunto de tecnologias de "processamento e de bases de dados em memória" contemplamos fornecedores como VoltDB, Applications and Products in Data Processing - High Performance Analytic Appliance Systems, Applications and Products (SAP HANA), QlikView, SolidDB, Membase, DRUID (Metamarkets), Statistical Analysis System (SAS HPA), and GemFire. Estas tecnologias diferenciam-se por processamento de grandes volumes com velocidade, Online Analytical Processing (OLAP) distribuído e em memória, e processamento analítico distribuído em memória para Message Passing Interface (MPI).

Utilizamos este tipo de tecnologia para implementar sistemas de ingestão e analítica de *feeds* em tempo real, sistema de trocas comerciais em tempo real, e processamento em tempo real de informação de máquinas e sensores.

Por último, no grupo das "bases de dados analíticas e aplicativos" contemplamos fornecedores como Greenplum DB, Teradata Aster, Kognitio, Vertica, ParAccel, Sybase IQ, Netezza, Teradata, Greemplum e Exadata Appliance. Estas tecnologias destacam-se por serem soluções "in-a-box" com baixo esforço de manutenção e interfaces para a escrita de *queries* complexas de Structured Query Language (SQL).

Este tipo de tecnologias são usadas para analisar de forma eficiente enormes volumes de dados estruturados, construção de aplicações complexas para análise de dados estruturados e implementação de grandes Data Warehousing paralelos.

Em conclusão, as tecnologias Big Data permitem às organizações receber, processar, guardar e analisar toda a informação disponibilizada por sistemas e sensores. Esta informação detalha todas as interações entre máquinas e pessoas e é necessária para compreender *quem*, *quando*, *porquê* e *para quê,* perceber o contexto e personalizar a interação de forma a construir uma relação emocional com cada uma das pessoas. Adicionalmente, no contexto de interações de máquinas permite-nos perceber situações anómalas e prever essas mesmas situações.

Sabendo que a explosão de dados já começou e que tem tendência para aumentar exponencialmente com a criação de mais serviços no mundo digital (abertura dos sistemas das organizações através de API), com a penetração de dispositivos e com os novos dispositivos ligados (*Internet of Things*), é fácil antever cada vez mais necessidades de processamento, *storage* e análise de dados.

Este é o futuro do IT e são necessárias pessoas com as competências certas para compreenderem este cenário, conhecerem as tecnologias e que queiram ajudar a resolver os desafios que o futuro apresentará…

AUTOR

Escrito por Pedro Sarmento
Manager Analytics, Accenture Digital

WINDOWS HELLO: A AUTENTICAÇÃO BIOMÉTRICA NO WINDOWS 10

Introdução

A Microsoft iniciou a 29 de julho a disponibilização à escala mundial do **Windows 10**, e nesta altura, são já os milhões de utilizadores que possuem os seus PCs atualizados. Como se tem falado nos últimos meses e tal como já tive oportunidade de escrever na edição nr. 47 desta revista, o Windows 10 traz imensas inovações das quais se destacam o tão aguardado **Menu Iniciar**, o **Microsoft Edge** e também a **Cortana**.

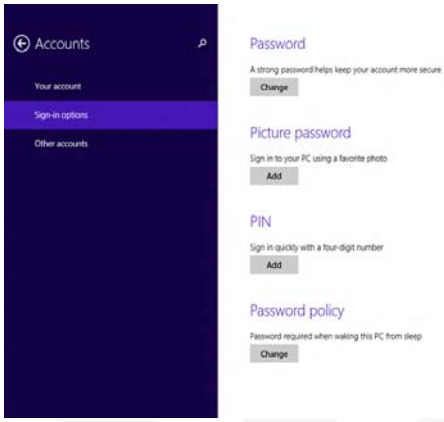
Para além de todas as novas funcionalidades que oferecem uma experiência de utilização mais familiar e pessoal aos utilizadores, a Microsoft continua a apostar fortemente na segurança do novo Windows através de inovações como o **Windows Hello** e o **Microsoft Passport**.

Nos próximos parágrafos deste artigo, vamos ficar a conhecer um pouco mais sobre estas funcionalidades e como são mais seguras que as tradicionais *"passwords"*.

A evolução das passwords no Windows

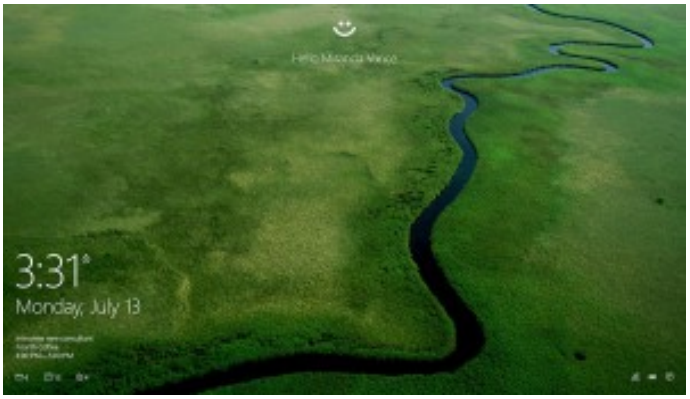
A palavra *"password"* é algo muito familiar nas nossas vidas, e diariamente, usamos diversas para manter os nossos dados seguros, aceder ao e-mail, efetuar compras online, aceder à rede empresarial e a dados sensíveis, etc. Apesar de utilizarmos passwords complexas, não significa necessariamente que estamos seguros e cada vez mais, surgem novas formas de ilicitamente retirar esta informação aos utilizadores.

Considerando então que a tradicional *password* já não é tão segura como o desejado, é inevitável que comecem a surgir novos tipos de passwords. O Windows 8 é exemplo disso e para além de ser possível iniciar sessão da forma tradicional, a Microsoft implementou duas novas alternativas mais seguras – a **Picture password** e o **Pin password**.



No Windows 10 o trabalho neste campo foi alvo de uma maior evolução, recorrendo agora à biometria. O Windows Hello

Em traços gerais, o **Windows Hello** em conjunto com o **Microsoft Passport**, permite o início de sessão no Windows 10 sem recorrer às tradicionais passwords. Através de uma nova geração de credenciais e biometria, aliadas também a novo hardware, passa a ser possível iniciar sessão em qualquer dispositivo, redes empresariais, serviços online, aplicações, etc.



Hardware com características específicas, permite que os utilizadores usem a **face**, a **íris** ou uma **impressão digital** para desbloquear os seus dispositivos. Estes três fatores de autenticação mais a criação de um **PIN** password, são certamente mais seguros que a simples *password*.

Para além de reconhecer o utilizador, o Windows Hello possui as seguintes funcionalidades:

- Permite configurar o desbloqueio do PC automaticamente ou com uma 2ª autenticação.
- Autenticação ao nível empresarial e acesso a conteúdos que suportem as **"Next Gen Credentials (NGC)"**. Exemplo: Acesso a redes empresariais e respetivos recursos, sites de comparas online, entre outros.
- Integra medidas **anti-spoofing** para mitigar ataques físicos, como o acesso a dispositivos e início de sessão não autorizado. Exemplo: Utilização de uma foto impressa ou cartão de funcionário para tentar iniciar sessão num PC.
- Através de infravermelhos, consegue obter uma imagem pormenorizada do utilizador nas mais diversas condições de luminosidade e considera pequenas alterações na aparência como a barba, maquiagem e óculos ou lentes de contato.

WINDOWS HELLO: A AUTENTICAÇÃO BIOMÉTRICA NO WINDOWS 10

Requisitos de Hardware

Para que seja possível configurar o Windows Hello, é necessário possuir algum hardware especial. O mais simples é o leitor de impressão digital e que muitos fabricantes de PCs já incluem nos seus equipamentos. Se possui um destes equipamentos e já fez o upgrade para o Windows 10, poderá desde já começar a utilizar o Windows Hello.

Leitor impressão digital

Para o reconhecimento facial, uma simples Webcam não é o suficiente para que o Windows Hello funcione, pois esta não consegue detetar os detalhes do rosto do utilizador com a precisão necessária. Para isso necessitamos de uma câmara que possua tecnologia de infravermelhos que para além de recolher com detalhe a fisionomia da nossa face, permite a utilização do Windows Hello nas mais variadas condições de luminosidade.

Uma dessas câmaras, é a **Creative Intel RealSense 3D DevKit** que pode ser adquirida no site da **Intel** por cerca de 120€ aproximadamente. Recentemente foi também atualizado o SDK desta câmara que a torna 100% compatível com o Windows Hello e com o desenvolvimento de aplicações que usem esta funcionalidade.

Para além deste DevKit, já estão a ser comercializados alguns equipamentos com esta tecnologia da Intel dos quais se destacam:

- Dell Inspiron 15 5548
- Acer Aspire V 17 Nitro
- Lenovo ThinkPad Yoga 15
- HP Sprout
- Lenovo ThinkPad E550
- Asus N551JQ
- Asus ROG G771JM
- Asus X751LD
- Dell Inspiron 23 7000
- HP Envy 15t Touch RealSense Laptop
- Lenovo B5030

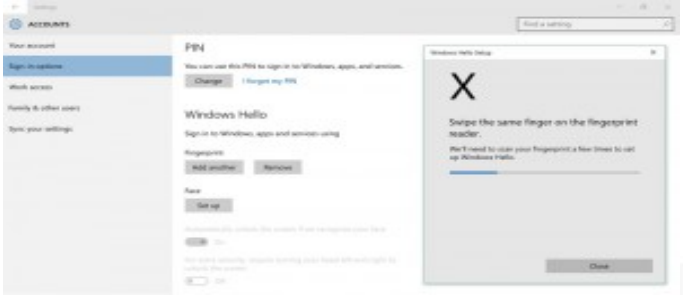
Relativamente a sensores para a íris, ainda não existem muitos detalhes mas com certeza que brevemente começarão a surgir equipamentos para comercialização no mercado.

Configuração

A configuração do Windows Hello é bastante simples e intuitiva. Imaginando que temos um equipamento que possui um leitor de impressão digital, vamos aceder a "**Settings**" e selecionamos **"Accounts"**.

Após este passo, vamos clicar no painel esquerdo em **"Sign-in options"**. No painel do lado direito para além das opções habituais (Password, Pin, Picture Password) vamos encontrar o **Windows Hello** ativo.

Para adicionar uma impressão digital, vamos clicar em **"Add"** e seguir os passos apresentados pelo Wizard. Será necessário repetir o processo algumas vezes para garantir uma leitura correta da impressão digital.



Depois de tudo configurado corretamente, o Windows Hello vai solicitar a nossa impressão digital cada vez que formos iniciar sessão e também nos dará indicações caso a impressão não esteja a ser lida com sucesso.

Segurança dos dados biométricos

Durante a configuração, o Windows através dos dados recolhidos pelos sensores, cria uma representação gráfica da biometria que é automaticamente encriptada e armazenada localmente no PC. A informação originalmente obtida como a **fotografia** da nossa face, impressão digital etc., não é guardadas pelo Windows nem enviada para qualquer local fora do dispositivo.

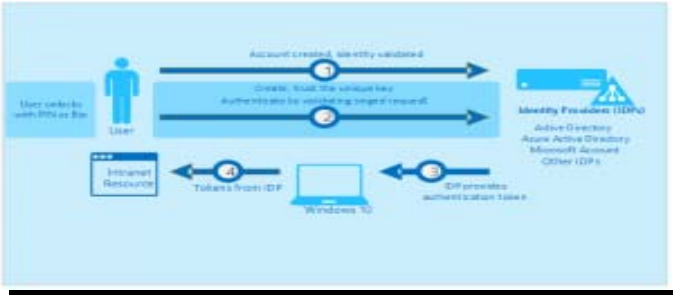
Esta representação gráfica vai impedir por exemplo que imagens reais sejam usadas para tentativas de acesso ilícito.

A Microsoft no site do Windows 10, disponibiliza um conjunto de perguntas e respostas que esclarece estas e outras questões.

Microsoft Passport

Tal como referi no início deste artigo, outra inovação associada à segurança do **Windows 10** e que pode ser usada em conjunto com o Windows Hello, é o nome de código **"Passport"**. A Microsoft descreve o Passport como um "sistema de programação" que gestores TI, programadores web e também de software, podem usar para tornar a nossa autenticação mais segura e sem recurso a passwords quando iniciamos sessão em sites ou aplicações.

O Passport permite a autenticação dos utilizadores em contas **Microsoft**, contas do **Active Directory** e **Microsoft Azure Active Directory**, ou contas de serviços de terceiros que suportem a autenticação Fast ID Online **(FIDO)**. Depois de uma verificação de dois fatores feita durante o processo inicial de inscrição no Passport, o mesmo fica configurado no dispositivo e o utilizador terá que definir um gesto que pode ser uma **autenticação biométrica** configurada no **Windows Hello** ou então um **PIN**.



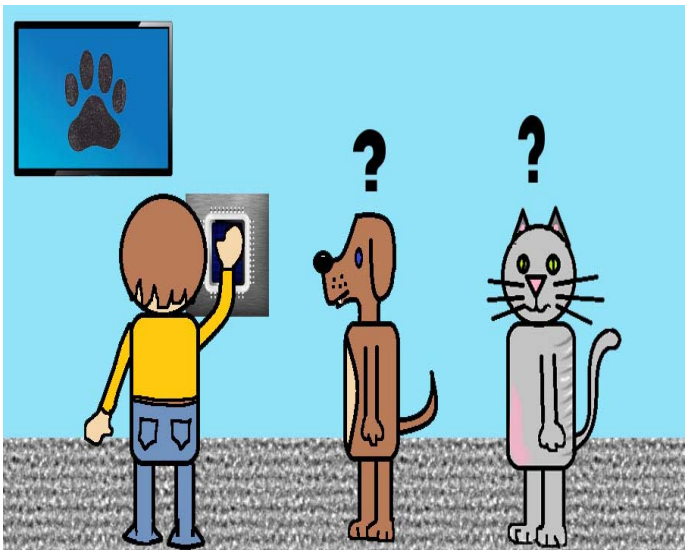
Estando as configurações terminadas, os utilizadores identificam-se através deste gesto e o **Windows 10** solicita ao **Passport** que faça a autenticação dos utilizadores nos serviços ou recursos com acesso protegidos.

A Microsoft no site TechNet, fornece informação detalhada sobre o Passport e as suas aplicações a nível empresarial e que poderão ser consultadas no seguinte link: Password-less Authentication with Microsoft Passport.

Conclusão

Em conclusão, a utilização destes métodos de autenticação traduz-se numa série de benefícios, não só porque simplifica a forma como interagimos com os nossos dispositivos, mas também pelo aumento significativo da segurança das nossas credenciais.

A adoção do Windows Hello poderá ser adiada por alguns utilizadores que queiram usar o reconhecimento facial, considerando o número reduzido de equipamentos no mercado que possuem hardware compatível, ainda assim, poderá desde já ser configurado em PCs que por exemplo possuam um leitor de impressão digital.



AUTOR

Escrito por Nuno Silva

Microsoft MVP Windows Experience | Microsoft Technical Beta Tester

WINDOWS 10 IOT CORE NO RASPBERRY PI 2 B

Quando a Microsoft divulgou que seria disponibilizado o Windows 10 Iot Core para o Raspberry Pi 2 B ficamos a ganhar um minicomputador com o sistema operativo Windows gratuitamente.

Mas esta versão está destinada a Internet-of-Things (IoT), e existem várias versões.

		Visual Studio e UWP	Interfaces de utilizador	Microsoft Azure IoT	Conectividade de dispositivo integrado
Dispositivos da indústria	Desktop Shell, aplicações Win32, 1 GB RAM, 16 GB de armazenamento.				
Dispositivos móveis	Shell Moderno, Mobile Chassis requirement, 512 MB RAM, 4 GB de armazenamento.				
Pequenos dispositivos	Dispositivos dedicados, ecrã opcional, Sem Shell/ armazenamento/ aplicações, 256MB RAM, 2GB de armazenamento, aplicações universais.				

Este não é o primeiro sistema operativo da Microsoft para IoT já existiam outros, como por exemplo o Windows 8.1 para o Intel Galileo que é uma placa concorrente ao Raspberry.

A IoT e a utilização de componentes físicos e eletrónicos de medida, controle e não só que estão ligados a Internet. Por exemplo numa estufa não é necessário ir a um local X em X horas para controlar a temperatura, com a comunicação machine-to-machine (M2M) os dados da estufa são enviados para uma base de dados utilizando a Internet e os mesmos dados podem ser consultados por outras máquinas.

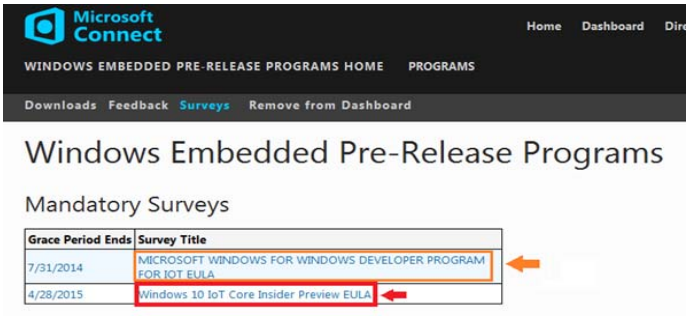
O Raspberry Pi 2 B é exatamente igual a Raspberry Pi B+. Sobre a evolução do Raspberry Pi convido-o a ler o artigo "*Raspberry Pi2 – Evolução ou Revolução?*" da Rita Peres publicado na edição nº 48 de Março de 2015.

Ilustração 1 Raspberry Pi B+ e Raspberry Pi 2 B

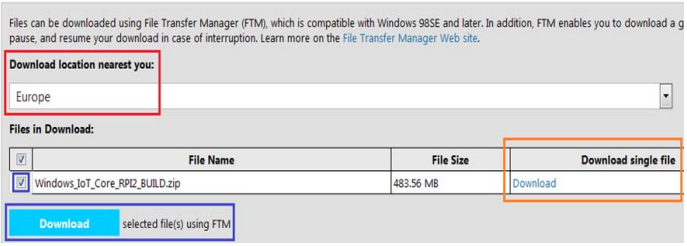
Para iniciar a instalação do Windows 10 no Raspberry Pi é necessário o seguinte equipamento para o mesmo:

- Raspberry Pi 2 B;
- Carregador de 5 Volts micro USB com uma amperagem de 1.0 (Carregador de telemóvel micro USB);
- Cartão micro SD class 10 8GB ou superior;
- Cabo HDMI;
- Cabo de rede Ethernet;
- Não é obrigatório mas recomendo uma caixa para o Raspberry Pi 2 B.

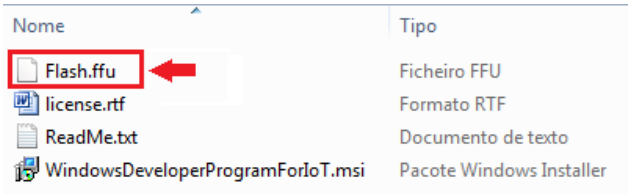
Para descarregar o Windows 10 é necessário aceder ao programa Microsoft Connect (https://connect.microsoft.com/windowsembeddediot/SelfNomination.aspx?ProgramID=8558) com uma conta Microsoft. Depois de aceder clique em "Surveys" e terá de aceitar todos os surveys que estão disponíveis se estiver de acordo com os termos dos mesmos.



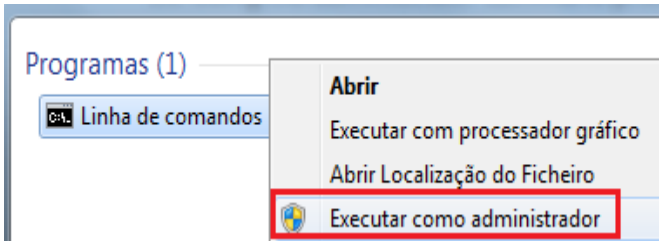
Após aceitar todos os survey clique em "Downloads" selecione "Windows 10 IoT Core Insider Preview Image for Raspberry Pi 2". Antes de clicar no botão "Download" selecione a localização mais próxima para que o mesmo seja mais rápido. É possível fazer o Download de duas formas, direto clicando em Download na coluna "Download single file" a outa é através de um gestor de transferência de ficheiros selecionado os ficheiros que se pretende na caixa de seleção e a seguir clicando no botão azul "Download"



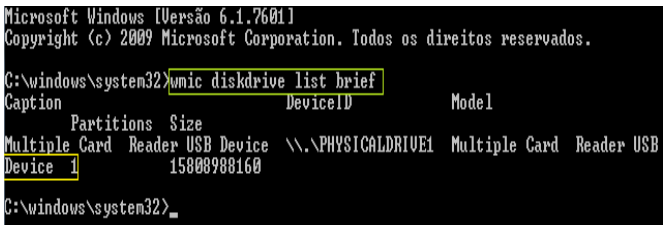
Ao concluir o Download é necessário extrair o seu conteúdo para uma pasta, recomendo a criação de uma pasta temporária na raiz do disco, por exemplo "temp" a mesma contem o ficheiro "Flash.ffu" que é a imagem do Windows 10 a ser transferida para o cartão SD.



Antes de colocar o Windows 10 no cartão SD recomendo a formatação do mesmo e prara tal utilize a aplicação SDFormatter (https://www.sdcard.org/downloads/formatter_4/) esta aplicação é gratuita e permite formatar cartões de memória SA, SDHC e SDXC. Com o cartão formatado inicia-se a Linha de comandos do Windows com opções administrativas, antes de clicar na aplicação "Linha de comandos" no menu iniciar clique uma vez com o botão direito do rato e selecione a opção "Executar como administrador".



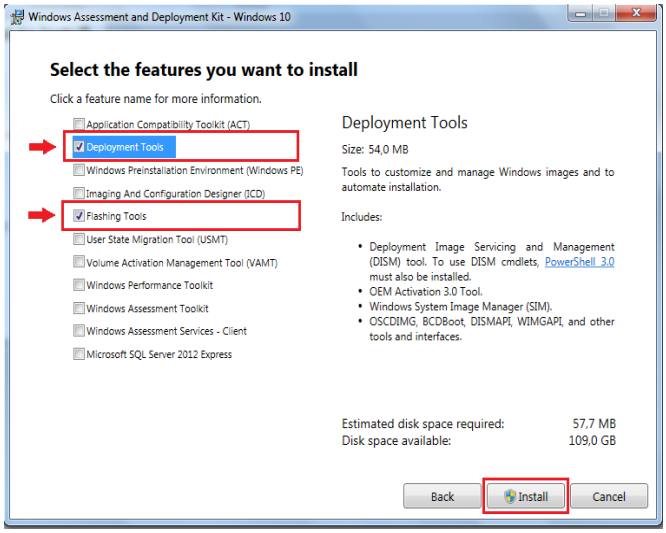
Ligue o cartão de memória SD no computador e execute o seguinte comando "wmic diskdrive list brief" que vai identificar todos os sistemas de armazenamento que estão ligados no computador. A seguinte imagem mostra que o cartão SD está ligado a um leitor de cartões e que o mesmo está identificado como Device 1.



Para transferir a imagem do Windows 10 para o cartão SD é utilizada a aplicação Microsoft Deployment Image Servicing and Management (DISM). Esta aplicação é uma ferramenta de implementação de imagens a mesma é também é utilizada para implementar a instalação do Windows em USB flash drive (Pen drive).

O DISM já se encontra instalada no Windows mas se utilizar o Windows 7, como a versão está desatualizada e assim é necessário instalar o Microsoft Windows Assessment and Deployment Kit (ADK) para Windows 10 (http://go.microsoft.com/fwlink/p/?LinkId=526740).

Na instalação do ADK não são necessários todos os componentes selecione apenas os seguintes: "Deployment Tools" e "Flashing Tools".



No Windows 7 a 64 bits execute o seguinte comando " cd C:\Program Files (x86)\Windows Kits\10\Assessment and Deployment Kit\Deployment Tools\amd64\DISM".

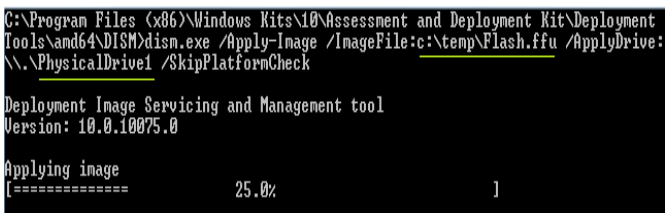
No Window7 a 32 bits execute o seguinte comando "cd C:\Program Files (x86)\Windows Kits\10\Assessment and Deployment Kit\Deployment Tools\x86\DISM"

No Windows 8 ou superior não é necessário qualquer comando.

Para transferir a imagem execute o seguinte comando:

"dism.exe /Apply-Image /ImageFile:[localização da pasta] Flash.ffu /ApplyDrive:\\.\PhysicalDrive[numero da unidade] / SkipPlatformCheck"

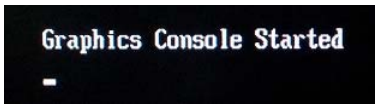
Na seguinte imagem está a ser utilizado o Windows 7 a 64 bits com o ADK a localização do ficheiro está na raiz da unidade C na pasta "temp" e o cartão SD está identificado como unidade 1.



Ao finalizar a transferência da imagem já pode colocar o cartão SD no Raspberry e ligar o mesmo.

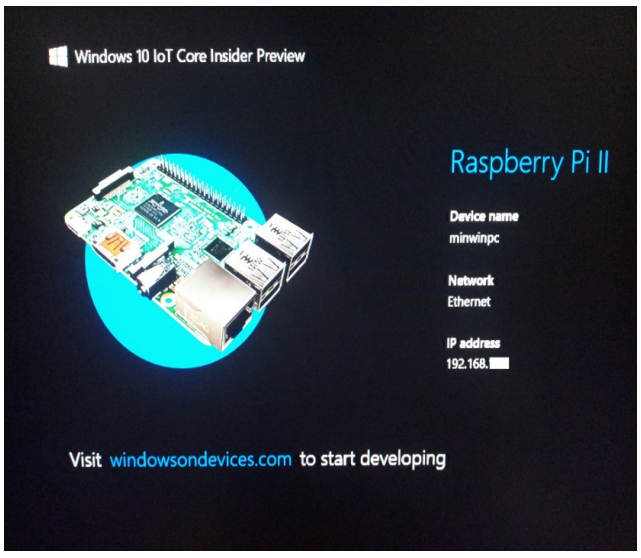
WINDOWS 10 IOT CORE NO RASPBERRY PI 2 B

A primeira informação que o Raspberry vai mostrar é que está a inciar “Graphics Console Started”



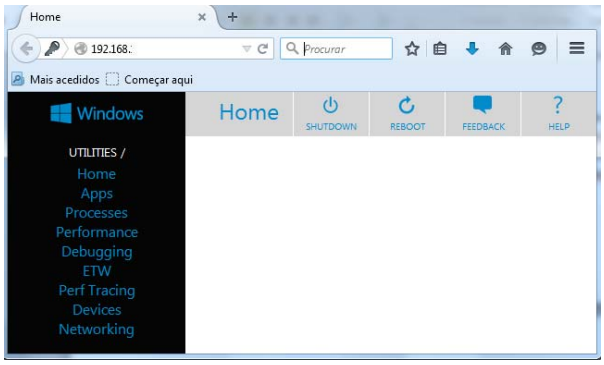
A primeira inicialização do Windows pode demorar alguns minutos e até carregar completamente irá ver o logotipo do Windows 10, só depois um segundo logotipo.

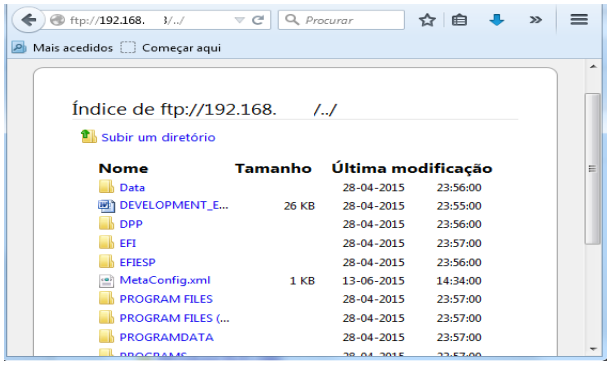
Por fim quando carregar completamente o Windows irá mostrar informação com o nome da máquina e endereço IP em que está configurado.



Com o Raspberry completamente ligado pode verificar que não existe nenhum sistema de janelas como nas versões de trabalho do Windows. Assim é possível reduzir o consumo de recursos utilizados. Isto já acontece com o Windows Server 2012 em que um administrador de sistema pode instalar um novo servidor sem Graphical user interface (GUI) e fazer toda administração por PowerShell conseguindo assim mais performance.

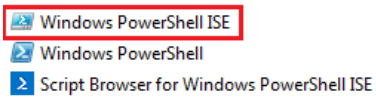
É possível aceder ao Windows no Raspberry de três formas. Por navegador de internet, FTP ou PowerShell através do endereço de IP. O nome de utilizador por defeito é “Administrator” e a palavra passe é “p@ssw0rd”.





Por navegador de Internet podemos desligar ou reiniciar o equipamento, instalar ou retirar aplicações, ver a performance. Por FTP podemos consultar, inserir, modificar e apagar ficheiros. Por PowerShell podemos fazer a gestão.

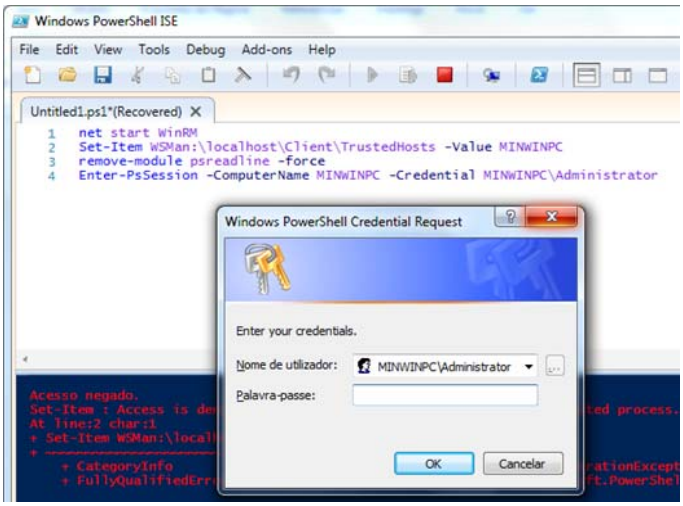
Para fazer a gestão por PowerShell recomendo a utilização da aplicação gráfica PowerShell Integrated Scripting Environment (ISE) que está incluído no Windows Management Framework 3.0 (https://www.microsoft.com/en-us/download/details.aspx?id=34595) após a instalação o PowerShell ISE está disponível no menu iniciar, inicie o mesmo com privilégios administrativos a mesma opção utilizado quando foi executado a Linha de comandos.



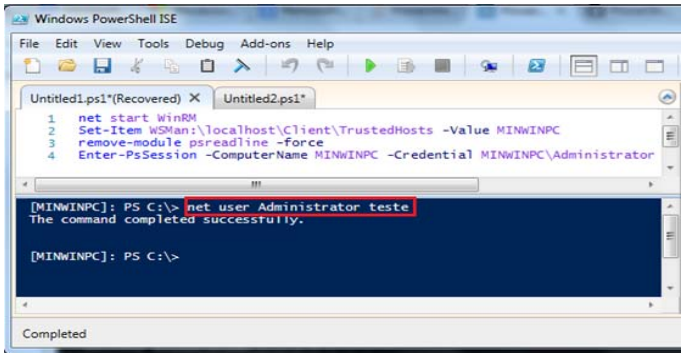
Para executar comandos remotamente é necessário executar o seguinte script

```
net start WinRM
Set-Item WSMan:\localhost\Client\TrustedHosts -
                                    Value MINWINPC
remove-module psreadline -force
Enter-PsSession -ComputerName MINWINPC -Credential
                                MINWINPC\Administrator
```

Ao executar o script será solicitado a palavra passe do administrador e por fim é feita a ligação remota.



Com a ligação efetuada agora pode-se inserir qualquer comando diretamente na subjanela abaixo. A primeira recomendação é alterar a palavra passe. Para alterar insira o seguinte comando “net user Administrator [nova palavra passe] ”.



Apos a confirmação do comando anterior é necessário executar o seguinte comando “schtasks /Delete /TN Microsoft\Windows\IoT\Startup /F” este comando apenas é executado uma vez.

Se quiser a identificação do equipamento o comando é “set computername [novo nome] ” para assumir a alteração é necessário reiniciar o comando é “shutdown /r /t 0”.

Apartir agora o Windows 10 no Raspberry Pi 2 B está pronto a ser utlizado pode experimentar as demostrações disponíveis na documentação disponível no Windows Dev Center (http://ms-iot.github.io/content/en-US/win10/StartCoding.htm).

Uma lista de comandos para PowerShell está disponível no Windows Dev Center (https://ms-iot.github.io/content/en-US/win10/tools/CommandLineUtils.htm).

Acho excelente que a Microsoft continue a apostar no IoT fornecendo uma versão completamente gratuita que permita ao programador interagir com o Hardware conseguindo-se assim um modelo de prototipagem em que pode trabalhar diretamente com sensores e outros componentes eletrónicos.

Com convergência de todos os sistemas operativos da Microsoft para uma só plataforma conseguimos criar aplicações universais não só compactáveis com pequenos dispositivos mas também com equipamentos móveis ou industriais.

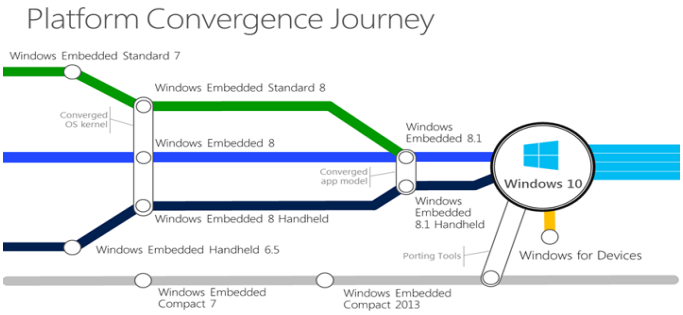


Ilustração 2 IOT201 - Building Devices with Windows 10 IoT

Temos também mais vantagens com o Raspberry Pi 2 B + Windows 10 conseguimos ter uma forma de aprender a fazer script em PowerShell e a Microsoft continua a inovar e melhorar os seus serviços em nuvem com o Microsoft Azure (http://azure.microsoft.com/pt-pt/) a mesma já disponibiliza um grande conjunto de serviços para IOT para poder receber e processar a informação recolhida dos dispositivos.

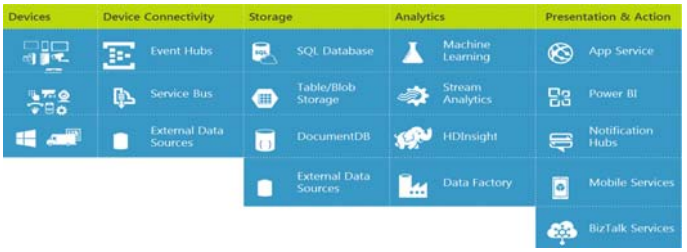


Ilustração 3 Microsoft Azure IoT (Build 2015)

Os ficheiros de PowerShell no do GitHub em: https://github.com/rramoscabral/PowerShell/tree/master/Windows_10/Iot

AUTOR

Escrito por Ricardo Cabral

Licenciado em Engenharia Informática pela Universidade Autónoma de Lisboa. O seu twitter é @rramoscabral

E QUE VENHAM MAIS 50 EDIÇÕES!

E eis caros leitores, que chegamos ao fim de mais uma edição da nossa PROGRAMAR. Esta é uma edição especial para todos nós e esperamos que também o seja para todos vós que nos lêem e seguem edição após edição.

Nunca será demais dizer-vos, e agradecer-vos, a vossa "presença". A cada download. A cada visualização.

Chegados à quinquagésima edição, aqui na PROGRAMAR aproveitamos a ocasião para recordar um pouco o passado. Um passado recente, construído página por página.

A primeira edição, foi lançada em Março de 2006, há nove anos e seis meses atrás. Na altura a equipa estava ainda longe de saber o sucesso que esta "pequena grande publicação" teria.

Artigo após artigo, número após número, foram muitos os que se juntaram a este projecto, participando nele quer como autores, quer como colaboradores da revista. Até à data cerca de 180 pessoas já colaboraram neste projecto. Um projecto que sendo "pequeno", se tornou enorme. Sendo a única revista portuguesa de programação. E disponibilizada de forma gratuita a todos os interessados.

Já ultrapassamos os 500 artigos publicados. Um número que será sempre maior a cada edição lançada.

As estatísticas revelam que em média, cada edição tem cerca de 18.950 downloads.

18950 x 49 edições ≈92.8550 downloads

Claro que secretamente todos nós esperamos que a edição 50 seja a edição com mais downloads de todas. Mas afinal, não é para isso que trabalhamos a cada nova edição?

Voltando às datas, em Setembro de 2008, a revista foi finalmente reconhecida e registada na base de dados do Centro Internacional de ISSN, sendo o número de referência o ISSN 1647-0710. Esta referencia é hoje, reconhecida em todo o mundo.

Como todos os que nos lêem sabem, somos um projecto sem fins lucrativos, todos nós somos voluntários, e cedemos de bom grado, algumas horas da nossa própria vida para dar continuidade a um projecto único em muitas vertentes.

Nunca é demasiado recordar que todos podem ser autores da Programar. A revista é de todos, feita por iguais.

A PROGRAMAR É UM PROJETO DE TODOS E PARA TODOS

Ao longo destes anos, tivemos vários artigos, sendo eles das mais variadas temáticas. Tivemos e temos colunas residentes, que consideramos serem uma mais valia a esta publicação.

Ao longo da história da revista, 548 artigos foram publicados e 183 autores colaboraram connosco, mais concretamente 10 autoras e 173 autores.

Como este é um projecto de todos para todos, e queremos que os leitores interajam, há várias edições que são votados os 3 melhores artigos, sendo o prémio uma t-shirt do P@P. Esta votação é aberta e feita pelos leitores que assim o queiram.

Em Maio de 2014, foi relançado o nosso site, em formato blog-post. Neste momento a revista é disponibilizada nesse formato, e claro, no tradicional formato pdf.

Os leitores mais atentos recordam que apesar de neste momento sermos uma publicação trimestral, já fomos uma publicação bimensal.

Contudo, se tivermos mais artigos, mais qualidade, poderemos voltar a ser uma publicação bimensal. Mais uma vez defendemos, que o poder de tornar este projecto maior, está em cada um de vós. Não pensem que não têm qualidade para escrever, que não sabem de que tema escrever, ou que não têm nada para apresentar. Desenvolvam uma ideia, e criem asas ao vosso artigo.

Num futuro próximo, que passam pelas ideias da equipa:

- Desenvolver app cliente da revista cross-platform
- Formato .iba (iTMS)

Contudo, estamos sempre abertos a novas ideias. Juntos seremos mais e melhores.

Porque nós, PROGRAMADORES, somos “gente de ideias”, e não desistimos!

Por isso e por muito mais… PROGRAMAR… ontem, hoje e amanhã!

Às próximas 50 edições!...

No Code


PROGRAMAR
Network Scanning
Interacção Python/MySQL
REFLECTION E SUAS APLICAÇÕES
Engenharia de Software
Techdays 2008
Festival Nacional da Robótica
PROGRAMAR
Estado de Visualização em ASP.NET
Programas Circulares em Haskell / Lucene / Internet Protocol v6
PROGRAMAR
Microsoft Visual Studio 2008 e .net Framework 3.5
O que há de novo
PROGRAMAR
CONTROLO TOTAL SOBRE SOFTWARE
SUBVERSION
PROGRAMAR
Cloud Computing
Subversion
Controlo Total sobre o Software - 2.ª Parte
Packet Capture em Java
com JPCAP
3º Aniversário da Revista PROGRAMAR!
PROGRAMAR
Google WEB TOOLKIT
AJAX
LINUX GRAPHICS STACK
Processamento de Texto
AWK
INTRODUÇÃO PRÁTICA A POWERSHELL
PROGRAMAR
Metaprogramação em C++
Google Web Toolkit - Parte 2
Arduíno e a Aquisição de Dados
PROGRAMAR
Diagrama de Blocos
Templates T4
LUA: Linguagem de Programação
Acesso a BDs remotas em Windows Mobile 6
PROGRAMAR
REVISTA PORTUGUESA DE PROGRAMAÇÃO • WWW.PORTUGAL-A-PROGRAMAR.COM
COMPUTAÇÃO EM GRID
LUA 2ª PARTE: LINGUAGEM DE PROGRAMAÇÃO
CAKEPHP SCRIPT BAKE
O PRÓXIMO PADRÃO DE C++
PROGRAMAR
REVISTA PORTUGUESA DE PROGRAMAÇÃO • WWW.PORTUGAL-A-PROGRAMAR.ORG
INTRODUÇÃO A PROGRAMAÇÃO DE ANDROID
LUA 3ª PARTE: LINGUAGEM DE PROGRAMAÇÃO
PYTHON PROGRAMAÇÃO FUNCIONAL
PROGRAMAR
REVISTA PORTUGUESA DE PROGRAMAÇÃO • WWW.PORTUGAL-A-PROGRAMAR.ORG
INTRODUÇÃO A BASES DE DADOS PARA OBJECTOS
LUA 4ª PARTE: LINGUAGEM DE PROGRAMAÇÃO
PADRÕES DESENHO CORPORATIVOS
INTRODUÇÃO WINDOWS COMMUNICATION FOUNDATION
MODELO3 ENTREVISTA A CELSO PINTO
PROGRAMAR
REVISTA PORTUGUESA DE PROGRAMAÇÃO • WWW.PORTUGAL-A-PROGRAMAR.ORG
JOGO DO PONG COM SLICK2D (JAVA)
LUA 5ª PARTE: LINGUAGEM DE PROGRAMAÇÃO
FLEX e BYACC
SOFTWARE OPEN SOURCE EM SISTEMAS DE INFORMAÇÃO GEOGRÁFICA
PROGRAMAR
REVISTA PORTUGUESA DE PROGRAMAÇÃO • WWW.PORTUGAL-A-PROGRAMAR.ORG
LINQ
IMPLEMENTAÇÃO DOS OPERADORES TAKELAST, TAKELASTWHILE SKIPLAST E SKIPLASTWHILE
PROGRAMAÇÃO
INTRODUÇÃO
LUA
FLEX e BYACC
OPTIMIZAÇÃO
ASPECTOS
SharePoint EPT
NetPonto
MAC ADDRESS
CORE DUMP
PROGRAMAR
REVISTA PORTUGUESA DE PROGRAMAÇÃO • WWW.PORTUGAL-A-PROGRAMAR.ORG
ASP.net MVC3
SILVERLIGHT 4.0
WINDOWS PHONE 7
LUA
FLEX e BYACC
OPTIMIZAÇÃO
JQUERY
AJAX
CORE DUMP
PROGRAMAR
REVISTA PORTUGUESA DE PROGRAMAÇÃO • WWW.PORTUGAL-A-PROGRAMAR.ORG
BUSINESS CONNECTIVITY SERVICES
ENTITY FRAMEWORK
CORE DUMP
LUA
ORACLE
WPF
NOVIDADES
SMARTY PHP
WINDOWS AZURE
ANDROID
SHAREPOINT
PROGRAMAR
git
CONTROLO DE VERSÕES PARA PEQUENOS E GRANDES PROJECTOS
INTRODUÇÃO
LUA
VIM
VISUAL (NOT) BASIC
CORE DUMP
NETPONTO


PROGRAMAR
RAILS
INTRODUÇÃO AO RUBY ON RAILS
NHIBERNATE
TÉCNICAS PARA SOFTWARE MELHOR E MAIS RÁPIDO
iOS COCOA TOUCH & MVC
KINECT HACK
ANDROID - MOBILE WORLD
RIA
DESENVOLVIMENTO DE APLICAÇÕES WEB RICAS (RIA) COM EXT JS 4 E RAILS 3
NOVIDADES DO VISUAL STUDIO 2012
MAKEFILES
CUDA
Windows 8
OBJECTOS VS ESTRUTURAS
JAVA EE & JAVA WEB
WEB PERSISTENTE LOCAL STORAGE
INTERAÇÃO COM VOZ NO ANDROID
CRIAR EXTENSÕES PARA GOOGLE CHROME
EXTENDENDO UMA APLICAÇÃO
CRIADA NO APP STUDIO DA MICROSOFT

No Code

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.

PROJECTO EM DESTAQUE NA COMUNIDADE P@P: REACH FOR 24

O “Reach for 24” é um jogo matemático que desafia a destreza e rapidez de raciocínio. O seu objectivo consiste em, através da utilização de todos os números mostrados, em apenas uma tentativa, atingir o número 24 utilizando as operações aritméticas de soma, subtracção, divisão e multiplicação.

Uma boa maneira de manter a calculadora cerebral em forma.

Na actualidade, pesquisadores, estudiosos e profissionais da educação que buscam criar situações desafiadoras e significativas para a construção de conhecimentos concebem os jogos como estratégias pedagógicas favoráveis, inclusive para a construção de conceitos matemáticos.

Segundo Kishimoto (2007), os jogos estão já vinculados nos pensamentos humanos desde a infância mesmo que não nos apercebamos, porque a criança cria suas próprias fantasias através de brinquedos ligados ao seu quotidiano familiar.

Segundo Montessori (1965), trabalhar com os jogos nos primeiros anos, é uma técnica que facilita o desenvolvimento do indivíduo. Através da utilização de jogos no ensino da matemática, é criada a possibilidades de oferecer várias opções para desenvolver as capacidades do indivíduo em cada fase em que se encontra. A utilização coerente dos jogos de acordo com objectivos, explorando o lúdico, é uma forma inteligente e criativa de promover a superação de obstáculos na aprendizagem da matemática.

Por exemplo, o ensino da matemática por meio de jogos, pode transformar as actividades matemáticas que, às vezes, causam ansiedade e sofrimento a muitos alunos, em fonte de satisfação, motivação e interacção social.

Da autoria de Tiago Rodrigues, desenvolvido em Delphi , utilizando o FireMonkey , vai já na versão 1.1.0, tendo sido actualizado pela última vez a 23 de Julho de 2015.

Para poder funcionar plenamente requer um sistema operativo Android 2.3 ou superior, ocupando 7,2 MB. Poderá ter experimentado algumas dificuldades de com processadores INTEL devido à tecnologia usada no desenvolvimento, mas tal já não ser verifica a partir da versão Android 4.4.

Ocupa uma classificação PEGI3 de acordo com a Classificação de PEGI (Pan European Game Information, informação pan-europeia sobre jogos).

Elege o melhor artigo desta edição

Revista PROGRAMAR

http://bit.do/ProgramarED50_V

Veja também as edições anteriores da Revista PROGRAMAR

48ª Edição - Março 2015

47ª Edição - Dezembro 2014

46ª Edição - Setenmbro 2014

45ª Edição - Maio 2014

44ª Edição - Fevereiro 2014

43ª Edição - Dezembro 2013

e muito mais em ...

www.revista-programar.info

DUVIDAS?

IDEIAS?

AJUDAS?

PROJECTOS?

portugal-a-programar
.org