PROGRAMAR

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

EDIÇÃO #53 - AGOSTO 2016

Edição Comemorativa

10 ANOS

.NET

HMTL

C#

MVC

ENTITY FRAMEWORK

A PROGRAMAR

FUNÇÕES DISTRIBUIÇÃO DE PROBABILIDADES E O PYTHON

PHP SISTEMA DE CHAT PÚBLICO

MONOGAME

OBRIGADO, CORE 1

REVIEWS

ANDROID DESENVOLVIMENTO DE APLICAÇÕES COM ANDROID STUDIO

BOMBEIROS 1

SQL SERVER 2014

A VIDA NA CLOUD

INDUSTRIA ALIMENTAR ALIADA ÀS TI

ALGORITMO DIJKSTRA

NO-CODE

INTRODUÇÃO AO ARDUINO

PODER DE UMA SPA

ENGENHARIA DE SOFTWARE

PROJECTO P@P DIRTY BIKE EXTREME

ENTREVISTA PROGRAMAR

EDITORIAL

EQUIPA PROGRAMAR

Contacto
revistaprogramar@portugal-a-programar.org

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

ISSN
1 647-071 0

TC BANKCALL #TEMPORARY I HOPE HOPE HOPE

Todos os percalços fazem parte da "evolução", de versão para versão! E desta feita o atraso nesta edição, foi fruto de um dos maiores percalços até agora enfrentado! Mas como sempre sobrevivemos, superamos, evoluímos! E passados 10 anos, cá estamos!

Para esta edição, estava com imensas dificuldades em escolher um titulo para o editorial, até que me lembrei daquilo que nos acontece, a todos nós que desenvolvemos e criamos tecnologia! Aquelas situações em que escrevemos algo, ou ligamos algo, e acreditamos com todas as forças, que vai funcionar, ainda que seja algo temporário! E nessa perene memória lembrei-me de um comentário que li, no recentemente tornado publico, código fonte desenvolvido para o modulo lunar da missão Apollo 11, que colocou o primeiro Homem na lua! Um singelo comentário quase que humorístico, onde se pode ler *"Temporary, I hope hope hope".*

De facto a ideia do autor da linha de código em questão, bem poderia ser que aquele "truque", fosse apenas temporário, mas facto é que levou um Humano à lua e trouxe-o de volta!

Esta edição comemora os 10 anos da revista, e relembra-nos que a cada iteração, continuamos a evoluir, a melhorar a aprender, apesar de algumas vezes também nós termos vontade de fazer como um outro developer de uma plataforma que tanto usamos escreveu "this is a hack for this release"! Sim, algumas coisas são mesmo "hacks for this release" ou "temporary we hope, hope, hope", mas a verdade é que continuamos cá! A trazer-vos novos conteúdos a cada edição! Foram 10 anos, venham mais 10! Venham mais 2^{10}-1 e nós cá estaremos!

Até lá, despedimo-nos com votos de umas excelentes férias, recheadas de muito código e sem nenhum overflow!

António Santos

Nota: *A partir desta edição ficará sem efeito o concurso habitual de entrega de t-shirts aos três artigos mais votados pelos leitores. Contudo, de forma a continuarmos a premiar a contribuição dos nossos redactores, no final do ano, será atribuída uma lembrança aos autores com mais contribuições para a revista , na forma de artigos para a revista Programar. Agradecemos desde já a vossa a atenção e compreensão. Em breve daremos novidades sobre este tema. Fiquem atentos!*

ÍNDICE

TEMA DE CAPA

A PROGRAMAR

ELECTRÓNICA

COLUNAS

ANÁLISES

NO CODE

EVENTOS

7º Encontro da comunidade IT Pro Portugal - 30 de Agosto de 2016 http://itproportugal07.eventbrite.pt/
Pixels Camp 6 - 8 Outubro de 2016 https://pixels.camp/
Lisbon Game Conference 14-15 de Outubro de 2016 http://lisbongameconf.iscte-iul.pt/

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

Microsoft PowerShell torna-se open-source e ganha versões para GNU/Linux e Mac OS

Na passada semana, a Microsoft tornou open-source a Shell PowerShell. Inicialmente apenas disponível para sistemas Windows, está agora disponível para Linux e Mac . Estão a ser lançadas versões alfa do PowerShell para Linux e Mac OS X , podendo os utilizadores aceder ao GitHub para descarregar os packages pré-compilados do software e ter acesso ao código fonte da peça do mesmo. Esta ferramenta irá permitir a utilização da linguagem de script da Microsoft em plataformas Linux e Mac. À semelhança do que foi feito com a .NET framework em 2014, a iniciativa da Microsoft abre mais um produto, anteriormente exclusivo à plataforma Windows, ao universo MultiPlataforma.

Até ao momento mais de 3500 pessoas marcaram o repositório e mais de 300 fizeram forks do mesmo. Foram detectadas mais de 200 questões e submetidos mais de 800 pedidos de alteração e acrescento ao código original. Foram feitas mais de 2000 adições ao código original só na última semana.

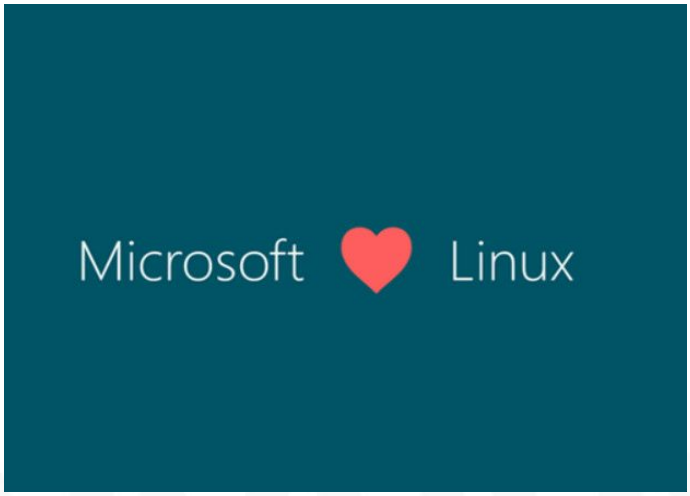
Tal como aconteceu aquando da disponibilização do código fonte da framework .NET, foram encontrados alguns comentários engraçados ao longo do código, a exemplo: *AssemblyInfo.cs Linha 16 "// These attributes aren't every used, it's just a hack to get VS to not complain" , VariableAnalysis.cs linha 631: "(this is an ugly hack, but it works.)"*

Estão neste momento disponíveis instruções de instalação para as plataformas Windows 10, Windows 8.1, Windows Server 2016 e 2012 R2, distribuições de GNU/ Linux Ubuntu 16.04 e 14.04, CentOS 7, Mac OS X 10.11 e o popular Docker.

Para mais detalhes aqui fica o link:

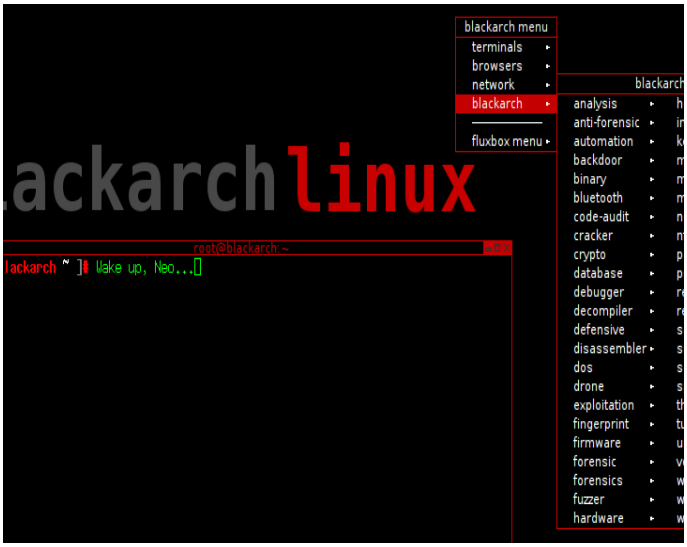
https://github.com/PowerShell/PowerShell

Fonte: «Revista PROGRAMAR»



BlackArch Linux: disponibilizados cerca de 1900 pacotes e mais de 1500 ferramentas de testes de segurança.

Desde o lançamento da ISO BlackArch Linux há três meses atrás, a equipa de desenvolvimento adicionou mais de 100 ferramentas novas para uso em testes de segurança. Baseado no sistema operativo Arch Linux, leve e altamente personalizável, a nova ISO contém todas as actualizações de segurança e software lançadas desde o anúncio em Maio da versão anterior.



A equipa BlackArch tem mais de 1500 ferramentas na imagem ISO mais recente, bem como actualizações ao BlackArch Linux installer.

A imagem ISO está disponível para download no site oficial, bem como os respectivos mirrors.

Fonte: «Revista PROGRAMAR»

TEMA DE CAPA

Entity Framework Core 1

Entity Framework Core 1

Introdução

A Entity Framework é a tecnologia para acesso a dados recomendada pela Microsoft. A sua próxima versão (disponível a partir de Junho de 2016) será a quinta e promete ser revolucionária em vários sentidos. Este artigo irá revelar o que precisa de saber sobre ela.

História

A Entity Framework foi lançada em conjunto com o Service Pack 1 da *framework* .NET versão 3.5, em meados de 2009. No início, dividiu atenções com o **LINQ to SQL**, outra *framework* de acesso a dados lançada ao mesmo tempo; por esta, no entanto, ser mais simples e limitada unicamente a SQL Server, cedo foi deixada para trás, passando a Microsoft a recomendar a utilização da Entity Framework para todos os cenários de acesso a bases de dados relacionais.

Com a primeira versão da Entity Framework, a Microsoft iniciou uma aventura num mercado onde já existiam outros participantes, sendo o mais notório o NHibernate. Como *framework* **ORM** – **Object-Relational Mapper** -, a Entity Framework encontrava-se então muito longe do que estes outros participantes ofereciam, beneficiando, no entanto, de bom suporte a LINQ, tecnologia então nos seus primórdios, e que se viria a revelar essencial.

A segunda encarnação foi introduzida com o .NET 4.0 e trouxe consigo o suporte opcional a **Plain Old CLR Objects** (**POCOs**) – simples classes .NET sem necessidade de herdar de uma classe específica da *framework*, carregamento implícito (*lazy loading*) e a possibilidade de personalizar os modelos T4 usados para gerar as classes, quer quando o modelo era importado da base de dados, quer quando este era definido no editor integrado no Visual Studio.

Com a terceira versão lançada, 4.1, a Microsoft passou a propor um novo fluxo para o desenvolvimento: começar com o código (modelo **POCO**) e a partir daí gerar a base de dados. Esta abordagem adequa-se bem à metodologia de desenvolvimento conhecida por **Domain-Driven Design** (DDD), a qual advoga em primeiro lugar a representação por meio de classes de todos os conceitos do domínio de negócio, sendo a persistência um aspecto que, embora claramente importante, deve ser o mais transparente possível, não devendo limitar as opções de desenho do modelo de domínio. Passaram a existir convenções para definir grande parte da tradução de e para o modelo relacional, sendo desnecessário configurar todos os detalhes explicitamente. O novo API **DbContext**, grandemente simplificador relativamente ao **ObjectContext**, revelou-se extremamente popular, apesar de algumas ausências, totalmente aceitáveis numa primeira versão, e o mesmo se passou com a nova funcionalidade de migrações, poderosa, que trouxe para o mundo das bases de dados as funcionalidades de controlo de versões, tão bem conhecidas dos programadores. Passou a falar-se de **Entity Framework Code First**.

Brevemente a Microsoft disponibilizou uma actualização – 5 - que introduziu tipos de dados enumerados e geoespaciais ao modelo **Code First**, bem como a possibilidade de chamar funções SQL transparentemente em pesquisas LINQ através de métodos estáticos em classes .NET.

Finalmente, a versão 6 veio culminar o processo de maturação da Entity Framework como ferramenta ORM de pleno direito, com suporte a intercepção e *logging* de chamadas, uso de *stored procedures* para operações **Create-Update-Delete** (**CUD**), possibilidade de definição de convenções personalizadas, e, uma das funcionalidades mais apreciadas, a possibilidade de efectuar a maior parte das operações de forma assíncrona. Muitas das funcionalidades previamente introduzidas foram também alvo de melhoramentos, com destaque para as migrações. Parecíamos estar no bom caminho.

Eis senão quando a Microsoft veio baralhar completamente o jogo, com o anúncio da sua nova versão, designada inicialmente por 7, e, mais tardiamente, por **Core 1**, nome que ficou.

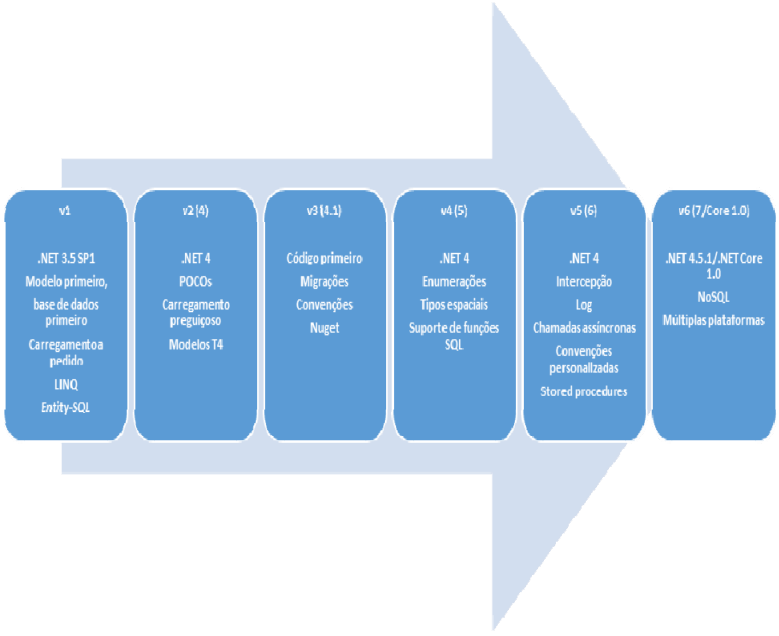


Figura 1 - Evolução da Entity Framework

TBD

Funcionalidades Adicionadas ou Alteradas

Novo Código

A EF Core foi totalmente escrita de raiz. Esta decisão teve a ver com o desejo da Microsoft de se ver livre de código "legado", muito do qual praticamente não era usado – alguém usa Entity-SQL ou o **ObjectContext** desde a saída do **Code First**? – mas também com a necessidade de uma profunda alteração que permitisse a implementação de novas funcionalidades radicalmente incompatíveis com a anterior arquitectura. Esta abordagem, com méritos, veio, no entanto, causar algumas limitações à versão inicial, 1.0, como veremos a seguir.

Instalação

À semelhança das anteriores versões, a EF Core irá ser disponibilizada exclusivamente através do Nuget. A diferença é que, ao invés de um único pacote Nuget, teremos de instalar vários, consoante as necessidades:

Pacote	Descrição
Microsoft.EntityFrameworkCore	Classes base da Entity Framework
Microsoft.EntityFrameworkCore.Tools	Outras funcionalidades (migrações, p.ex.)

Tabela 1 - Pacotes Nuget

NoSQL

O mundo em 2015 e 2016, no que respeita às Tecnologias de Informação, é diferente do que era em 2009. Por um lado, o movimento **No SQL** ou **Not-Only SQL** (**NoSQL**), ganhou grande preponderância – passou a ser normal, numa empresa, a utilização de vários tipos de tecnologias para armazenamento de dados, mesmo concorrentemente, para responder a diferentes necessidades. Se a Entity Framework no início correspondia ao que as pessoas esperavam, pode argumentar-se que já não é assim. Revelando atenção às novas tendências, a Microsoft tornou a nova versão da Entity Framework agnóstica em relação às fontes de dados de uma forma inédita: passou a suportar não apenas bases de dados relacionais mas potencialmente qualquer fonte de dados, incluindo NoSQL, através de um mecanismo de *providers*. Foram anunciadas duas provas de conceito, na forma de *providers* para **Azure Table Storage** e para o muito popular serviço de *cache* distribuída **Redis**, mas este anúncio fica, para já, pela intenção, já que a versão inicial, Core 1.0, não irá incluir esta funcionalidade. A ideia é promissora – usar a mesma API para efectuar consultas a potencialmente qualquer fonte de dados – mas teremos de esperar para ver.

Múltiplas Plataformas

Sendo construída de raiz com suporte ao .NET Core, a versão de .NET multi-plataforma, a Entity Framework Core irá, assim, correr nos sistemas operativos Linux, MacOS, FreeBSD e Windows, bem como em dispositivos móveis usando Windows 10 e aplicações desenvolvidas para a Windows Store. Tal não exclui, no entanto, a utilização em aplicações .NET 4.6 ou superior, a qual continuará a ser, por agora e durante muito tempo, largamente dominante.

Configuração de Fornecedores de Dados

É substancialmente mais fácil a configuração de fornecedores de dados, passando a existir um método virtual na classe **DbContext** exclusivamente para esse efeito: trata-se do método **OnConfiguring**:

```
protected override void OnConfiguring
(DbContextOptionsBuilder optionsBuilder)
{
    optionsBuilder.UseSqlServer(@"Data
     Source=.\SQLEXPRESS; Integrated Security=SSPI;
                          Initial Catalog=Blogs;")
                base.OnConfiguring(optionsBuilder);
}
```

Código 2 – Configuração do fornecedor SQL Server

Este exemplo ilustra a utilização do fornecedor SQL Server. Outro exemplo, agora para o fornecedor em memória (requer a instalação do pacote **Microsoft.EntityFrameworkCore.InMemory**):

```
optionsBuilder.UseInMemoryDatabase();
```

Código 3 – Configuração do fornecedor In-Memory

Para que possamos usar um fornecedor, será necessário instalar o seu pacote Nuget. Os pacotes incluídos na versão 1 da Entity Framework Core são:

Nome	Descrição
Microsoft.EntityFrameworkCore.SqlServer	Fornecedor de acesso para SQL Server
Microsoft.EntityFrameworkCore.SQLite	Fornecedor de acesso para SQLite
Microsoft.EntityFrameworkCore.InMemory	Fornecedor de acesso *in-memory* (para testes)
Npgsql.EntityFrameworkCore.PostgreSQL	Fornecedor de acesso para PostgreSQL
EntityFrameworkCore.SqlServerCompact35 EntityFrameworkCore.SqlServerCompact40	Fornecedores de acesso para SQL Server Compact 3.5 e 4.0
*.Design	Fornecedores para geração de entidades a partir de uma base de dados relacional (p.ex., Microsoft.EntityFrameworkCore.SqlServer.Design)

Tabela 2 - Fornecedores de acesso

Para cada um destes existe um método de extensão específico para a classe **DbContextOptionsBuilder**, o qual aceita diferentes parâmetros, tais como a *connection string* a

usar (o *in-memory* não aceita nenhum).

Podemos ver a lista de fornecedores sempre actualizada em https://docs.efproject.net/en/latest/providers/index.html.

Geração de Identificadores

Historicamente, a EF apenas suportava duas estratégias para geração de chaves primárias:

- **IDENTITY**, do SQL Server;
- Chaves fornecidas manualmente (p.ex., GUIDs geradas por código, ou quaisquer outras chaves de tipos de dados básicos que fossem unívocas).

A nova versão vem finalmente trazer novidades a este respeito, nomeadamente, pela introdução do algoritmo **High-Low**. Este algoritmo, bem conhecido dos utilizadores de NHibernate, permite a geração de chaves primárias pela Entity Framework e não pela base de dados. Como não há bela sem senão, a Microsoft implementou este algoritmo com recurso às sequências do SQL Server 2012, pelo que não será suportado em versões anteriores, e o mecanismo no qual assenta não é verdadeiramente modular. Cai assim por terra o sonho de um mecanismo de geração de chaves primárias independente da base de dados.

A forma de configurar o uso de sequências para geração de identificadores globalmente é a seguinte (no método **OnModelCreating**):

```
modelBuilder.ForSqlServerUseSequenceHiLo();
```

Código 4 – Configuração global para uso de sequências para gerar identificadores

Mas é também possível defini-lo entidade a entidade:

```
modelBuilder
.Entity<Blog>()
.Property(b => b.BlogId)
.ForSqlServerUseSequenceHiLo();
```

Código 5 – Configuração de sequências para gerar identificadores para uma entidade

Geração de SQL Optimizada

A Microsoft gaba-se de ter optimizado a geração de SQL, no caso da utilização de bases de dados relacionais. Este era, de facto, um dos calcanhares de Aquiles das versões anteriores. Adicionalmente, passou a ser possível enviar ao mesmo tempo múltiplos comandos **INSERT**, **UPDATE** e **DELETE**, ao invés de um por cada, respectivamente, inserção, actualização ou apagamento, o que, só por si, deve melhorar o desempenho sensivelmente. Além disso, a reescrita do gerador de SQL introduz outras melhorias, no sentido de produzir SQL mais optimizado.

Um exemplo de SQL a inserir três registos e a retornar as chaves geradas (**IDENTITY**) pode ser:

```
INSERT INTO [Blog] ([Name], [Url], [Description])
OUTPUT INSERTED.[BlogId]
VALUES (@p0, @p1, @p2), (@p3, @p4, @p5), (@p6,
                                          @p7, @p8);
```

Código 6 – SQL para inserção de dados em bloco

Execução de Funções no Lado do Cliente

O LINQ introduz validação de *queries* no momento da compilação; no entanto, esta validação não inclui chamadas de métodos de extensão: estas aparentarão ser válidas até que o código corra, altura em que a EF tentará executá-los na base de dados, caso estes métodos tenham sido marcados para tal. Se não, com as versões anteriores da EF, isto resulta numa excepção, mas tal não é o caso da EF Core: pura e simplesmente, as chamadas serão efectuadas no lado do cliente, após os resultados terem sido materializados, de forma transparente:

```
var formattedEntities = (from blog in ctx.Blogs
     select blog.Format())
     .ToList();
```

Código 7 – Execução de funções no lado do cliente

Neste exemplo, o método **Format** irá ser executado após os registos terem sido retornados da base de dados e os objectos materializados. Qualquer método de extensão pode ser usado, apenas não na cláusula **where**, como é facilmente compreensível.

Outro exemplo, usar uma função *client-side* para transformar uma coluna de forma a que possa ser ordenada:

```
var sortedEntities = (from blog in ctx.Blogs
    select blog orderby blog.Name.Soundex())
   .ToList();
```

Código 8 – Execução de funções no lado do cliente

A ordenação, neste caso, apenas será aplicada no lado do cliente, já que a Entity Framework não sabe nada sobre o método de extensão **Soundex** (mero exemplo, este método não existe).

Mistura de SQL com LINQ

Passa a ser possível combinar *queries* LINQ com SQL. Um exemplo ilustrativo:

```
var itemsContainingTerm = ctx.Blogs
.FromSql("SELECT * FROM dbo.SearchBlogs (@p0)",
".net")
.OrderBy(b => b.Name)
.ToList();
```

Código 9 – Mistura de LINQ com SQL

Parece magia, mas a Entity Framework é capaz de combinar o resultado produzido pelo *stored procedure* **Sear-**

chBlogs (poderia ser um **SELECT**, claro está) e aplicar ordenação pelo campo indicado (**Name**), resultando no SQL seguinte (simplificado):

```
EXEC sp_executesql 'SELECT [b].BlogId, [b].Name,
[b].Url, [b].Description FROM (SELECT * FROM
dbo.SearchBlogs (@p0)) AS [b] ORDER BY [b].Name'
```

Repare-se que o SQL passado ao método **FromSql** suporta parâmetros. O único cuidado a ter com este SQL é que esteve deverá retornar colunas compatíveis com a entidade que se pretende devolver.

Propriedades Sombra

Passou a ser possível configurar propriedades de uma entidade que, tendo correspondência em colunas na base de dados (ou numa fonte de dados NoSQL), não possuem equivalência na classe .NET. Ou seja, são possivelmente incluídas em **SELECTs**, **UPDATEs** e **INSERTs**, apesar de não serem "visíveis" nas classes POCO. Um exemplo óbvio seria a adição de colunas de "auditoria" – "criado por", "criado em", "modificado por", "modificado em". Por não serem visíveis no código, serão, em teoria, mais difíceis de interferir com. O código para configurar tal seria algo como o seguinte:

```
public interface IShadowAuditable { }

protected override void OnModelCreating
(ModelBuilder modelBuilder)
{
    foreach (var e in modelBuilder.
          Model.GetEntityTypes().Where(e => typeof
  (IShadowAuditable).GetTypeInfo().IsAssignableFrom
                        (e.ClrType.GetTypeInfo()))
{
        modelBuilder
        .Entity(e.ClrType)
        .Property<string>("CreatedBy")
        .IsRequired();
        modelBuilder
        .Entity(e.ClrType)
        .Property<DateTime>("CreatedOn")
        .IsRequired();
        modelBuilder
        .Entity(e.ClrType)
        .Property<string>("UpdatedBy")
        .IsRequired();
        modelBuilder
        .Entity(e.ClrType)
        .Property<DateTime>("UpdatedOn")
        .IsRequired();
    }

    base.OnModelCreating(modelBuilder);
}
```

Código 10 – Configuração de propriedades sombra

Já a utilização seria:

```
public override int SaveChanges(bool acceptAllChan-
gesOnSuccess)
{
  foreach (var e in this.ChangeTracker.Entries()
.Where(e => typeof(IShadowAuditable).GetTypeInfo()
.IsAssignableFrom(e.Entity.GetType().GetTypeInfo
())))
    {
        if (e.State == EntityState.Added)
        {
            e.Property("CreatedBy")
     .CurrentValue = WindowsIdentity.GetCurrent()
     .Name;
            e.Property("CreatedOn")
             .CurrentValue = DateTime.UtcNow;
        }

        e.Property("UpdatedBy")
        .CurrentValue =
               WindowsIdentity.GetCurrent().Name;
        e.Property("UpdatedOn")
        .CurrentValue = DateTime.UtcNow;
    }
    return base.SaveChanges
                    (acceptAllChangesOnSuccess);
}
```

Código 11 – Actualização de propriedades sombra

Finalmente, é também possível efectuar queries LINQ sobre propriedades sombra, recorrendo a uma sintaxe algo bizarra:

```
var blogsCreatedToday = (from blog in ctx.Blogs
               where EF.Property<DateTime>(blog,
                  "CreatedOn") == DateTime.Today)
            .ToList();
```

Código 12 – Queries sobre propriedades sombra

Anexar Entidades em Cascata

Sempre foi possível "anexar" entidades a um contexto Entity Framework diferente daquele que as carregou. Por exemplo, entidades devolvidas por um *web service* (saber se um *web service* deve ou não devolver entidades é uma questão que não será discutida aqui) podem ser posteriormente anexadas a um contexto a residir num servidor diferente e serem tratadas como originarias desse contexto. A novidade introduzida na nova versão consiste em poder definir, para cada entidade relacionada com a entidade a anexar, qual o seu estado, do ponto de vista do novo contexto. Vejamos o que acontecia se anexássemos uma entidade com uma colecção de outras entidades associadas:

```
BlogContext ctx = /* … */;
Blog blog = /* … */;
ctx.Entry(blog).State = EntityState.Unchanged;
```

Código 13 – Alteração do estado de uma entidade e todos os seus descendentes

Todos os elementos **Post** da colecção **Posts** da instância da classe **Blog** seriam, assim, marcados como não modificados. Imaginemos agora que possuíamos um grafo muito complexo onde, para cada entidade nele presente pretendíamos especificar o seu estado. Para esse cenário, foi criado o API **TrackGraph**:

```
BlogContext ctx = /* … */;
Blog blog = /* … */;
ctx.ChangeTracker.TrackGraph(blog, (node) =>
      {
            if (node.Entry.Entity is Post)
```

```
        {
              //marcar o post como inalterado
        node.Entry.State = EntityState.Unchanged;
          //se a data do post for anterior a hoje
              if (((Post) node
    .Entry.Entity).Timestamp.Date<DateTime.Today)
              {
                    //colocar uma indicação que
          //será partilhada por todos os filhos
                    node.NodeState = true;
              }
        }
        else if (node.Entry.Entity is Tag)
        {
              //usa a indicação colocada
              //anteriormente
              if (node.NodeState == true)
              {
                    node.Entry.State =
                          EntityState.Deleted;
              }
              else
              {
                    node.Entry.State =
                          EntityState.Unchanged;
              }
        }
  });
```

Código 14 – Utilização do TrackGraph

Ou seja, o **TrackGraph** sabe como percorrer todas as relações a partir de uma entidade raiz e permite que adicionemos a nossa lógica para definição dos estados encontrados.

Geração de Entidades a Partir da Base de Dados

A Entity Framework possibilita a geração de entidades a partir da base de dados (claro está, apenas para bases de dados relacionais), sem necessidade do Visual Studio. Para tal, é necessário ter presente o fornecedor de acesso pretendido (p.ex., **Microsoft.EntityFrameworkCore.SqlServer**) e também a sua versão "design", a qual contém a funcionalidade para geração de entidades (**Microsoft.EntityFrameworkCore.SqlServer.Design**). A sua utilização é simples:

```
dotnet ef dbcontext scaffold "Data
        Source=.\SQLEXPRESS; Initial Catalog=Blog;
                        Integrated Security=SSPI"
         "Microsoft.EntityFrameworkCore.SqlServer"
```

Código 15 – Geração de entidades a partir da base de dados

Claro está, este exemplo assume um servidor de base de dados SQL Server Express local com o nome **SQLEXPRESS**, uma base de dados chamada **Blog** e autenticação integrada do Windows. O leitor deverá alterar estes dados de acordo com o seu cenário concreto.

Outras Funcionalidades

Importa também referir que a nova Entity Framework possibilita também, na senda das versões anteriores, uma elevada extensibilidade: grande parte dos seus componentes pode ser trocada por outros. É possível, nomeadamente, injectar fornecedores de *log*, de geração de SQL, e muitos outros. Este tópico é vasto, pelo que fica apenas um exemplo de configuração de *log* para a consola:

```
public class ConsoleLoggerFactory : ILoggerFactory
{
 public void AddProvider(ILoggerProvider provider)
 { }
  public void Dispose() { }
  public ILogger CreateLogger(string categoryName)
    {
        //fazer log de tudo para a consola
        return new ConsoleLogger(categoryName,
                  (facility, level) => true, true);
    }
}

protected override void OnConfiguring
          (DbContextOptionsBuilder optionsBuilder)
{
    optionsBuilder.UseLoggerFactory(new
                          ConsoleLoggerFactory());
    optionsBuilder.UseSqlServer(@"Data
     Source=.\SQLEXPRESS; Integrated Security=SSPI;
                          Initial Catalog=Blogs;")
     base.OnConfiguring(optionsBuilder);
}
```

Código 16 – Configuração do fornecedor de log

Funcionalidades Ainda Não Suportadas

Nem tudo, no entanto, é positivo. Para não atrasar ainda mais a data de lançamento (já bastante atrasada), a versão 1.0 não inclui um número de funcionalidades que se encontravam nas versões anteriores:

- Estratégias de mapeamento de heranças: apenas **Table Per Type Hierarchy** / **Single Table Inheritance** é suportada, não **Table Per Type** / **Class Table Inheritance** nem **Table Per Concrete Type** / **Concrete Table Inheritance**;
- Carregamento implícito (*lazy loading*) e explícito (*explicit loading*) não são suportados; é necessário indicar, aquando da execução de uma *query* LINQ as entidades a trazer, por meio de chamadas a **Include**;
- Relações muitos-para-muitos: podem ser emulados recorrendo a duas relações um-para-muitos;
- Tipos complexos (*complex types*) também não existem na nova versão;
- Mapeamento de inserções, alterações e apagamentos por procedimentos armazenados (*stored procedures*);
- Agrupamentos (**GROUP BY**) na base de dados, ao invés de em memória;
- A capacidade de associar entidades a vistas;
- Estratégias para tentar re-executar comandos aquando de perda de ligação também não existirão;

- A possibilidade de adicionar convenções personalizadas;
- Intercepção de *queries* e comandos SQL;
- Filtrar colecções de entidades a carregar;
- Conversões de tipos de dados, por exemplo, de **String** para **XML**;
- Como foi dito no início, todo o suporte a NoSQL não foi ainda incluído nesta primeira versão.

Funcionalidades Removidas

Foram removidas as seguintes funcionalidades, a título definitivo:

- Todos os APIs dependentes de **ObjectContext** (introduzido na versão 1.0 da Entity Framework) foram removidos. Isto significa que não existe mais **Entity SQL** ou os eventos **SavingChanges** e **ObjectMaterialized**, mas também entidades com estado auto-gerido (*self-tracking entities*);
- O interface **IDatabaseInitializer<T>** e todas as suas implementações (**CreateDatabaseIfNotExists**, **DropCreateDatabaseAlways**, **DropCreateDatabaseIfModelChanges**); a alternativa passa por usar exclusivamente migrações; também deixa de ser possível a alimentação de registos iniciais (*seed*);
- Deixam de existir migrações automáticas;
- A validação de dados de acordo com o API **System.ComponentModel.DataAnnotations** deixou de ser efectuada;
- A abordagem *model first* deixa de ser suportada, apenas *database first* ou *code first*.

Conclusão

A nova versão da Entity Framework não deverá ficar para a história por si, mas pelo que possibilita. Na verdade, é a própria Microsoft a admiti-lo: esta nova versão não é a recomendada para uso profissional, devendo essa escolha recair sobre a Entity Framework 6.x. De facto, são tantas as lacunas que quase parece impossível que o gigante do *software* tenha optado por a tornar disponível, com tanta pompa e circunstância. No entanto, se pensarmos no suporte a múltiplas plataformas e na possibilidade (futura…) de usar o mesmo API quer para fontes de dados relacionais quer para não relacionais (Azure Table Storage, Redis, outras), a coisa muda de figura. Para sermos honestos, há algumas funcionalidades que merecem a nossa atenção desde já, mas, para uma utilização "a sério", resta-nos esperar pela próxima versão. A curiosidade já aperta!

Referências

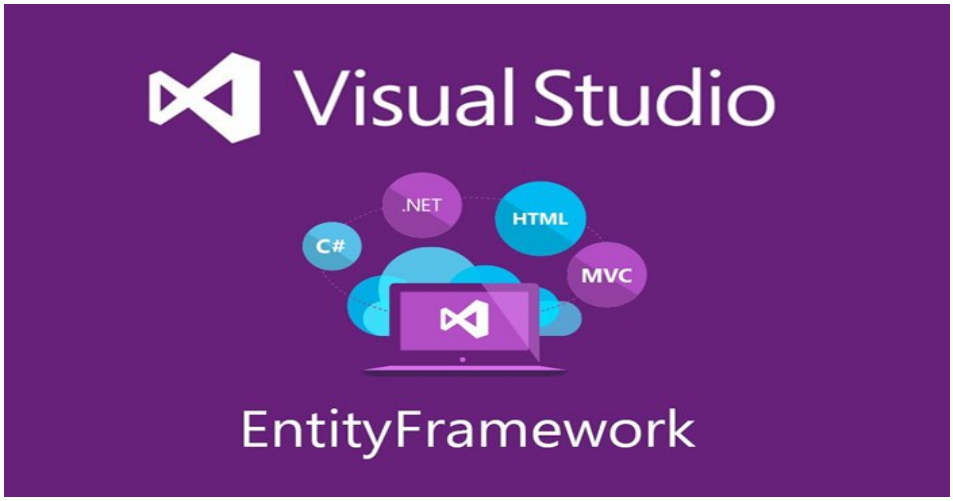
https://github.com/aspnet/EntityFramework

https://github.com/aspnet/EntityFramework/wiki/Roadmap

http://ef.readthedocs.io/en/latest

https://data.uservoice.com/forums/72025-entity-framework-feature-suggestions

https://blogs.msdn.microsoft.com/dotnet/



AUTOR

Escrito por Ricardo Peres

Evangelista Tecnológico na Simplifydigital. Microsoft MVP.

A PROGRAMAR

Algoritmo de Dijkstra

Funções de distribuição de probabilidade e o Python

Sistema de chat público em PHP

O meu primeiro jogo em MonoGame

Algoritmo de Dijkstra

Nesta edição da Programar, não quisemos deixar de lado uma das linguagens mais usadas de todos os tempos.

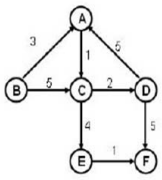
A famosa linguagem C.

E nesta edição comemorativa dos 10 anos da nossa revista, achamos que faria todo o sentido recordar um algoritmo, que em algum dia das nossas vidas, todos nós, programadores ouvimos falar… o não menos famoso que a própria linguagem C, o algoritmo de Dijkstra… e porque este algoritmo? Porquê este refere, o caminho do custo mínimo. E todos nós sabemos que a nossa revista já percorreu muitos caminhos até chegamos à edição 53.

Ora para os mais distraídos, e para os menos recordados, este algoritmo data do ano de 1956, tendo tido a sua primeira publicação em 1959. Foi criado por um matemático computacional holandês, Edsger Dijkstra. E trouxe uma solução que vários procuravam na altura, a solução para o problema do caminho mais curto num grafo dirigido.

Este algoritmo é muito útil para minimizar custos em várias áreas como por exemplo, na implementação de redes (por exemplos redes OSPF), ou no conhecido sistema GPS.

:Tracing Dijkstra's algorithm starting at vertex B



Pass:	initially	1	2	3	4	5	6	Shortest distance	Predecessor
Active vertex:		B	A	C	D	E	F		
A	∞	3						3	B
B	0							0	-
C	∞	5	4					4	A
D	∞	∞	∞	6				6	C
E	∞	∞	∞	8	8			8	C
F	∞	∞	∞	∞	11	9		9	E

:The resulting vertex-weighted graph is

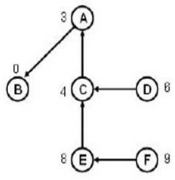


Ilustração 1 – Exemplo 1 Algoritmo Dijkstra

Em termos teóricos, o algoritmo é simples…

Ora vejamos…

Num dado vértice (nó) no grafo, o algoritmo localiza o caminho com a menor custo (isto é, o caminho mais curto) entre esse vértice e todos os outros vértices, recorrendo ao peso/custo da aresta. Este sistema, pode também ser usado para encontrar custos de caminhos mais curtos a partir de um único vértice para um vértice de destino parando o algoritmo uma vez que o caminho mais curto para o vértice destino tiver sido determinado.

Voltando ao exemplo do GPS, se os vértices do gráfico representarem cidades e os custos de caminho representarem distâncias entre as cidades ligadas por uma estrada directa, o algoritmo de Dijkstra pode ser usado para encontrar o caminho mais curto entre uma cidade e todas as outras cidades.

Vertex	Known	Cost	Path	
0	true	0	-1	Path: 0
1	true	1	0	Path: 0 1
2	true	5	4	Path: 0 4 2
3	true	1	0	Path: 0 3
4	true	1	0	Path: 0 4
5	true	6	1	Path: 0 1 5

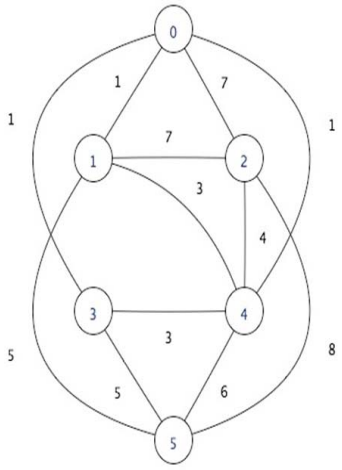


Ilustração 2 - Exemplo 2 Algoritmo Dijkstra

A nível de algoritmo temos o seguinte:

Seja G(V,A) um grafo orientado e **s** um vértice de G:

1. Atribuir valor zero à estimativa do custo mínimo do vértice **s** (a raiz da busca) e infinito às restantes estimativas;

2. Atribuir um valor qualquer aos precedentes (o precedente de um vértice **t** é o vértice que precede **t** no caminho de custo mínimo de **s** para **t**);

3. Enquanto houver vértice aberto:

   - seja **k** um vértice ainda aberto cuja estimativa seja a menor dentre todos os vértices abertos;
   - feche o vértice **k**
   - Para todo vértice **j** ainda aberto que seja sucessor de **k** faça:
     - some a estimativa do vértice **k** com o custo do arco que une **k** a **j**;

ALGORITMO DE DIJKSTRA

caso esta soma seja melhor que a estimativa anterior para o vértice **j**, substituir e anotar **k** como precedente de **j**.

Passemos agora à parte da implementação em C.

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

#define FLSH gets(l)

/* definir variavies */
int destino, origem, vertices = 0;
int custo, *custos = NULL;


void dijkstra(int vertices,int origem,int destino,int *custos)
{
      int i,v, cont = 0;
      int *ant, *tmp;
      int *z;     /* vertices para os quais se
                     //conhece o caminho minimo */
      double min;
      double dist[vertices]; /* vetor com os custos
                                //dos caminhos */


      /* aloca as linhas da matriz */
      ant = calloc (vertices, sizeof(int *));
      tmp = calloc (vertices, sizeof(int *));
      if (ant == NULL) {
       printf ("** Erro: Memoria Insuficiente **");
            exit(-1);
      }

      z = calloc (vertices, sizeof(int *));
      if (z == NULL) {
       printf ("** Erro: Memoria Insuficiente **");
            exit(-1);
      }

      for (i = 0; i < vertices; i++) {
         if (custos[(origem - 1) * vertices + i] !=
                                                - 1)
{
                  ant[i] = origem - 1;
                  dist[i] = custos[(origem-1)
                                   *vertices+i];
            }
            else {
                  ant[i]= -1;
                  dist[i] = HUGE_VAL;
            }
            z[i]=0;
      }
      z[origem-1] = 1;
      dist[origem-1] = 0;

      /* Ciclo principal */
      do {

      /* Encontra o vertice que deve entrar em z */
            min = HUGE_VAL;
            for (i=0;i<vertices;i++)
                  if (!z[i])
                   if (dist[i]>=0 && dist[i]<min) {
                              min=dist[i];v=i;
                        }

            /* Calcula distancias dos novos
//vizinhos de z */
            if (min != HUGE_VAL && v != destino - 1) {
                  z[v] = 1;
                  for (i = 0; i < vertices; i++)
                        if (!z[i]) {
                              if (custos
              [v*vertices+i] != -1 && dist[v] + custos
                              [v*vertices+i] < dist[i]) {
                                    dist[i] = dist
                              [v] + custos[v*vertices+i];
                                    ant[i] =v;
                              }
                  }
            }
      } while (v != destino - 1 && min !=
                                          HUGE_VAL);
      /* Mostra o Resultado da procura */
      printf("\tDe %d para %d: \t", origem,
                                          destino);
      if (min == HUGE_VAL) {
            printf("Nao Existe\n");
            printf("\tCusto: \t- \n");
      }
      else {
            i = destino;
            i = ant[i-1];
            while (i != -1) {
            //    printf("<-%d",i+1);
                  tmp[cont] = i+1;
                  cont++;
                  i = ant[i];
            }

            for (i = cont; i > 0 ; i--) {
                  printf("%d -> ", tmp[i-1]);
            }
            printf("%d", destino);

            printf("\n\tCusto:  %d\n",(int) dist
                                          [destino-1]);
      }
}


void cabecalho(void)
{

      printf("Implementacao do Algoritmo de
                                    Dijasktra\n");
      printf("Comandos:\n");
      printf("\t d - Adicionar Grafo\n"
            "\t r - Procura Os Menores Caminhos
                                    no Grafo\n"
            "\t CTRL+C para Sair do
                                    programa\n");
      printf("-------- ");
}

void add(void)
{
      int i, j;

      do {
            printf("\nQual o numero de vertices
                        (numero minimo = 2 ): ");
            scanf("%d",&vertices);
      } while (vertices < 2 );

      if (!custos)
            free(custos);
      custos = (int *) malloc(sizeof(int)
                              *vertices*vertices);
      for (i = 0; i <= vertices * vertices; i++)
            custos[i] = -1;
            printf("Insira as arestas:\n");
```

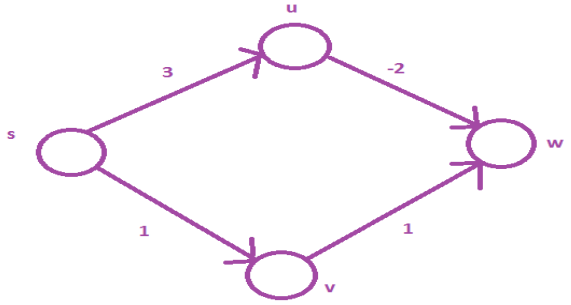
```
      do {
            do {
                  printf("Origem da aresta (entre 1
                    e %d ou '0' para sair): ", vertices);
                  scanf("%d",&origem);
            } while (origem < 0 || origem >
                                          vertices);

            if (origem) {
                  do {
                        printf("Destino da aresta
    (entre 1 e %d, menos %d): ", vertices, origem);
                        scanf("%d", &destino);
                  } while (destino < 1 || destino >
                     vertices || destino == origem);

                  do {
                        printf("Custo (positivo) da
         aresta do vertice %d para o vertice %d: ",
                                    origem, destino);
                        scanf("%d",&custo);
                  } while (custo < 0);

                  custos[(origem-1) * vertices +
                              destino - 1] = custo;
            }

      } while (origem);
}

void procurar(void)
{
      int i, j;

      /* Azul */

      printf("Lista dos Menores Caminhos no Grafo
                                          Dado: \n");

      for (i = 1; i <= vertices; i++) {
            for (j = 1; j <= vertices; j++)
                  dijkstra(vertices, i,j, custos);
            printf("\n");
      }

      printf("ENTER para voltar ao menu
                                          inici-
al>\n");
      /* Volta cor normal */
      printf("\033[m");
}

int main(int argc, char **argv) {
      int i, j;
      char opcao[3], l[50];

      do {

            cabecalho();
            scanf("%s", &opcao);

            if ((strcmp(opcao, "d")) == 0) {
                  add();
            }
            FLSH;

            if ((strcmp(opcao, "r") == 0) &&
                                    (vertices > 0) )
{
                  procurar();
                  FLSH;
            }

      } while (opcao != "x");

      printf("\nOver & Out\n\n");

      return 0;
}
```

Quero recordar ao leitor que esta é apenas uma das várias maneiras e formas de implementar este algoritmo.

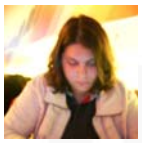
Assim que percebemos o algoritmo, a implementação fica a cargo da imaginação de cada um.

Aqui na PROGRAMAR desejamos que todos os nossos leitores encontrem o vosso melhor caminho sempre!

Até breve!



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.

Funções de distribuição de probabilidade e o Python

Resumo

Este artigo aborda a construção de um programa escrito em Python capaz de calcular funções de distribuição de probabilidade (PDF) a partir de arquivos *.txt ou *.csv contendo uma ou duas colunas de dados. Essas informações estatísticas são importantes para compreensão de diversos problemas complexos que usualmente são representados, de forma equivocada, através de uma abordagem Gaussiana simples

Introdução

As funções de distribuição de probabilidade (PDF) são usadas para descrever a dispersão de dados experimentais em torno de uma média. As distribuições de probabilidade são geralmente representadas em termos de integrais como uma função de densidade de probabilidade e podem ser interpretadas como um histograma construído com intervalos infinitesimais. A PDF mais comum é a distribuição Gaussiana ou normal, geralmente utilizada para o cálculo dos limites de confiança de dados experimentais1.

A construção de PDFs deve satisfazer as seguintes condições:

1 - PDFs são sempre positivas;

2 - A área delimitada pela PDF f(x) é sempre igual a 1.0 2;

3 - A probabilidade de um valor aleatório para estar dentro do intervalo (α, Ω) é igual à área delimitada pela curva dentro deste intervalo, como mostrado na Figura 1.

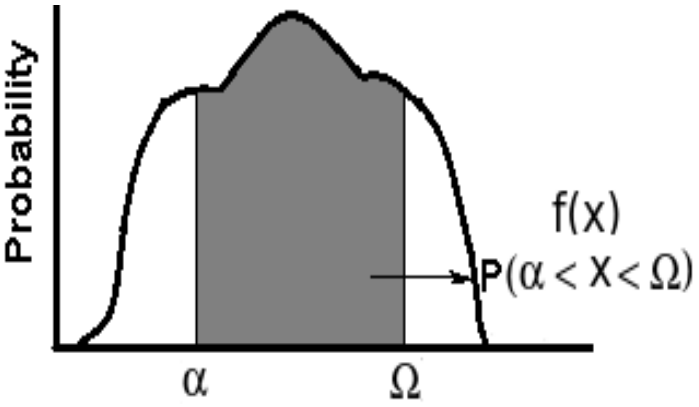


Figura 1 – PDF bimodal hipotética

A maioria das investigações científicas assume que as flutuações de dados experimentais podem ser representadas por uma função de densidade normal. Bard [3] já provou que essa abordagem é muito útil, principalmente devido à sua simplicidade. No entanto, as flutuações experimentais podem seguir diferentes funções de distribuição, que podem ser úteis para uma interpretação mais apurada dos dados experimentais.

Assim, procurando acelerar a análise de dados, nosso laboratório decidiu construir uma ferramenta em Python, capaz de gerar PDFs de forma rápida e confiável. O programa desenvolvido, denominado *PDF&Freq,* é capaz de gerar melhores informações estatísticas, as quais foram usadas em diversos trabalhos do grupo onde foi usado para gerar PDFs de tamanho de poros [4], diâmetro de partículas [5] e mesmo a resistividade volumétrica [6] dos nossos materiais.

Método

Desenvolvimento e descrição do programa:

O programa foi escrito usando a linguagem de programação Python. O Python foi lançado em 1991 pelo programador Guido Van Rossum e é uma linguagem de alto nível [7]. Uma característica em destaque desta linguagem é a priorização da legibilidade do código, tornando-a mais concisa e clara. O corpo do programa foi construído usando um ambiente de desenvolvimento integrado denominado PYTHON-IDLE, o qual permitiu a rápida importação de módulos, além do fácil uso de algumas funções básicas. O programa foi dividido em vários trechos, procurando facilitar a sua compreensão.

Identificação do programa e listagem de atualizações - A primeira etapa do programa visa a identificação do mesmo para o utilizador, fornecendo dados sobre a versão usada e a data de criação. Assim, foi utilizada a função **print**, que apresenta os seus objetos na tela do computador. Vários comentários foram acrescidos ao longo das linhas do programa usando o símbolo *hash* (**#**), o que permitiu manter um registo das atualizações:

```
#!/usr/bin/env python
# -*- coding: cp850 -*-

print "Programa para calculo de PDF's"
print "e para silulacao de valores mais pro-
vaveis"
print "Fernando Gomes - UFRJ"
print "V0.4 - 19/05/16"

#lista de atualizacoes:
#19/05/16 - Programa versao 0.4
#16/03/15 - Modifiquei a forma de verificar se
#ha uma ou duas linhas
#18/03/12 - Inseri rotina para detectar se o
#arquivo de entrada possui uma ou duas colunas
#16/06/10 - Programa versao 0.3
#21/05/10 - Programa versao 0.2
#18/03/10 - Programa versao 0.1
```

Importação dos módulos necessários - Os módulos ***numpy***, ***matplotlib***, ***os***, ***pylab***, ***random***, ***scipy***, ***statistics*** e

string, necessários para o funcionamento do programa, foram invocados conforme descrito abaixo:

```
#Importacao dos modulos
import string
#from rpy import *
from statistics import pdf
from numpy import *
from statistics import mean
from statistics import variance #as desvpad
from string import upper
from pylab import *
from scipy import integrate
from matplotlib.patches import Polygon
from numpy import log
from os import getcwd, listdir
from random import random as rnd
```

Escolha do kernel - Os histogramas são muito úteis para o tratamento de dados discretos e remetem intuitivamente, a alguma forma de estimativa de densidade. A heurística diz que é possível construir distribuições contínuas, desde que estejam disponíveis mais de 20 experiências 8. Para isso, a escolha de um kernel capaz de descrever as nuances amostrais da amostra é fundamental. Existem várias opções disponíveis para o cálculo de estimativas de densidade usando o Python. Cada uma delas depende de quais são os objetivos particulares da análise 9. Vários tipos de funções do kernel são disponíveis. Os mais comuns, como o Epanechnikov, Uniform, Triangle, Gaussian, Quartic/biweight , Triweight e Cosine estão disponíveis no programa.

```
print
print
print "Escolha o kernel:"
print
print "-'E' or 'Epanechnikov' : Epanechnikov kernel
                                        (padrao)"
print "-'U' or 'Uniform'      : Uniform kernel"
print "-'T' or 'Triangle'     : Triangle kernel"
print "-'G' or 'Gaussian'     : Gaussian kernel"
print "-'B' or 'Biweight'     : Quartic/biweight
                                        kernel"
print "-'3' or 'Triweight'    : Triweight kernel"
print "-'C' or 'Cosine'       : Cosine kernel"
print
k = raw_input("Escolha o kernel (E, U, T, G, B, 3
ou C)")
print
if k == "":
    k = 'G'
    print "Vc escolheu a opcao padrao (G)"
```

Recolha de informações gerais - Escolhido o kernel, o utilizador deve inserir a legenda desejada na abscissa do gráfico PDF, conforme descrito abaixo:

```
print
print
print "Escolha o kernel:"
print
print "-'E' or 'Epanechnikov' : Epanechnikov kernel
                                        (padrao)"
print "-'U' or 'Uniform'      : Uniform kernel"
print "-'T' or 'Triangle'     : Triangle kernel"
print "-'G' or 'Gaussian'     : Gaussian kernel"
print "-'B' or 'Biweight'     : Quartic/biweight
                                        kernel"
print "-'3' or 'Triweight'    : Triweight kernel"
print "-'C' or 'Cosine'       : Cosine kernel"
print
k = raw_input("Escolha o kernel (E, U, T, G, B, 3
                                        ou C)")
print
if k == "":
    k = 'G'
    print "Vc escolheu a opcao padrao (G)"

print
print
print
print "Voce deve escolher a legenda da abscissa"
print "Ex.: Diameter (nm)"
abscissa=raw_input("Qual eh a legenda da
                                    abscissa? ")
print
print
print
print "A legenda da abscissa eh:",abscissa
print
print
correto=raw_input("A legenda esta correta? (S/N)
")
while correto=="n" or correto=="N":
   abscissa=raw_input("Qual eh a legenda
                                        correta?")
    print
    print abscissa
    correto=raw_input("A legenda esta correta?
                                        (S/N) ")
    print
```

Além disso, o utilizador pode optar por gerar diferentes quantidades de valores simulados mais prováveis. A opção padrão é de 5 números:

```
print
print "Simulação dos valores mais provaveis"
print "Padrao = 5"
print
qp = raw_input("Quantos pontos? ")
print
if qp=="":
    qp=5
else:
    qp=int(qp)

gravar_log=open('log.dat','w') #Cria arquivo de
                                        log
gravar_log.write("kernel= "+k+'\n')
string_log = 'Arquivo ; LI(95%) ; MAXPROB; LS
                (95%) ; Media ; DesvPad ; Area'
gravar_log.write(string_log+'\n')
gravar_log.close()
```

Como observação, destacamos que o número de pontos para a construção da PDF, por padrão, é igual a 500:

```
npdf = 500 #Numero de pontos para a contrucao da
                                        PDF
```

Criação do arquivo de log - Um arquivo de log é criado para armazenar todas as informações relevantes como, o nome da amostra (Arquivo), o limite de confiança inferior com 95% de probabilidade (LI95%), a probabilidade máxima (MAXPROB), o limite de confiança superior com 95% de probabilidade (LS95%), o valor médio (Media), o desvio padrão (DesvPad) e a área sob a curva PDF (Area).

FUNÇÕES DE DISTRIBUIÇÃO DE PROBABILIDADE E O PYTHON

```
gravar_log=open('log.dat','w') #Cria arquivo de log
gravar_log.write("kernel= "+k+'\n')
string_log = 'Arquivo ; LI(95%) ; MAXPROB; LS
                    (95%) ; Media ; DesvPad ; Area'
gravar_log.write(string_log+'\n')
gravar_log.close()
```

Identificação dos arquivos para análise - Em seguida, são selecionados todos os arquivos *.txt ou *.csv presentes na pasta. Esses arquivos são listados e depois abertos, um a um, para a análise PDF, conforme descrito abaixo:

```
Conj_arquivos=listdir(getcwd())
arquivos=[]
for i in range(0,len(Conj_arquivos)):
    if Conj_arquivos[i][-3]=='t' and Conj_arquivos
        [i][-2]=='x' and Conj_arquivos[i][-3]=='t':
        arquivos.append(Conj_arquivos[i][0:-4])
    if Conj_arquivos[i][-3]=='c' and Conj_arquivos
        [i][-2]=='s' and Conj_arquivos[i][-3]=='v':
        arquivos.append(Conj_arquivos[i][0:-4])

print type(arquivos)
print arquivos
for i in range(0, len(arquivos)):
    print str(arquivos[i])
    nomearquivodados = string.rstrip(str(arquivos
                                        [i]),'\n')
    nomearquivodados = string.rstrip(str(arquivos
                                       [i]),'.txt')
    nomearquivodados = string.rstrip(str(arquivos
                                       [i]),'.csv')
    nomefig = nomearquivodados
    print nomearquivodados
    try:
        arquivo=open(nomearquivodados+".csv", "r")
#Abre arquivo para leitura
    except:
        arquivo=open(nomearquivodados+".txt", "r")
#Abre arquivo para leitura
    gravar=open(nomearquivodados+".rep", "w")
#Report
    gravar_pdf=open(nomearquivodados+".fdp", "w")
#Distribuição de probabilidade
    gravar_freq=open(nomearquivodados+".frq", "w")
#Arquivo com a frequencia das repeticoes
    gravar_log=open('log.dat','a') #Arquivo de log
    dados=[]
    for line in arquivo:
        linha_extra=line.split(',')
        if len(linha_extra)>1: #Verifica se ha uma
                                        coluna ou duas
            linha_extra[1].rstrip('\r\n')
            dados.append(float(linha_extra[1]))
#usar dados da segunda coluna
        else:
            dados.append(float(line)) #para o caso
```

Cálculos estatísticos - Em seguida é calculada a média, o desvio padrão e a PDF da amostra. Nesta etapa também é determinado onde ocorre o máximo de probabilidade na PDF.

```
    dados_med = 0
    pdf1 = 0
    pdf2 = 0
    pdf3 = 0
    dados_med = mean(dados) #Calcula a média
    dados_desvpad = (variance(dados))**.5 #Calcula
                                    o desvio padrao
    y0, x0 = pdf(dados, kernel = k, n = npdf)
    y_t0, x_t0 = pdf(dados, kernel = k, n = npdf)
    y_int0, x_int0 = pdf(dados, kernel = k, n =
                                            npdf)
    y = array(y0)
    x = array(x0)
    y_t = array(y_t0)
    x_t = array(x_t0)
    y_int = array(y_int0)
    x_int = array(x_int0)
    gravar.write(nomearquivodados+'\n')
    testey = sort(y_t)
    probMax = float(testey[len(y_t)-1:len(y_t)])
    for i in range(0,len(x)):
        if y_t[i] == probMax:
            ponto_x = i #Posição da lista onde a
                        probabilidade é máxima
            centro_x = x_t[i] #Valor da
                    propriedade em analise onde a
                        probabilidade é máxima
            string_MaxProb = "MaxProb = "+ str
                  (probMax) + " @ " + str(x_t[i])
            print string_MaxProb
            gravar.write(string_MaxProb+'\n')
    integral = 0
    cont = 0
    somatorio_Prob=0.0
    lim_cont = 500 - ponto_x
    for i in range(0,len(x)):
        somatorio_Prob=somatorio_Prob+float(y[i])
                    #Calculo do somatorio para
                    #transformar a escala
                    #de probabilidade
```

Posteriormente é calculada a probabilidade acumulada, necessária para a delimitação da região correspondente a 95% de probabilidade a partir da máxima probabilidade calculada:

```
PAC=[] #probabilidade acumulada
    xs=[] #Valor da propriedade para ser usado na
            simulacao
    for i in range(0,len(x_int)):
        xs.append(x[i])
        string_PDF = str(x[i]) + ',' +str(y[i])
        gravar_pdf.write(string_PDF+'\n')
        integral=integrate.trapz(y_int[0:i],x_int
                                            [0:i])
        PAC.append(integral)
        if integral/.025 < 1.0:# > 0.023 and
                                integral < 0.025:
            ponto025 = i
        if integral/.975 <1.0:# > 0.975 and
                                integral < 0.976:
            ponto975 = i
```

A penúltima etapa dos cálculos estatísticos consiste em determinar, a partir da PDF, cinco valores mais prováveis:

```
    dadosimulado=0.0
    prob=0.0
    nps=0 #número pontos simulados
    PAC.sort(reverse=True)
    xs.sort(reverse=True)
    #qp=5 #quantos pontos
    while nps<qp:
        for i in range(1,len(xs)/10):
            prob=rnd()
```

```
            if float(PAC[i])>prob:
                dadosimulado=float(xs[i-1])+(float
                (xs[i])-float(xs[i-1]))*(prob-PAC[i
                            -1])/(PAC[i]-PAC[i-1])
                if dadosimulado>0:
                    gravar_freq.write(str
                                (dadosimulado)+'\n')
                    nps=nps+1
                if nps>=qp:
```

Então, os principais resultados são salvos

```
    string_LInf = "Limite inferior = " + str(x_t
                                        [ponto025])
    print string_LInf
    gravar.write(string_LInf+'\n')
    string_LSup = "Limite superior = " + str(x_t
                                        [ponto975])
    gravar.write(string_LSup+'\n')
    print string_LSup
    string_Resist = "Propriedade (95%LC) = (-" +
    str(float(centro_x)-float(x_t[ponto025])) + ";"
     + str(float(centro_x)) + "; +" + str(float(x_t
                [ponto975])-float(centro_x)) +") "
    gravar.write(string_Resist+'\n')
    print string_Resist
    string_Area = "Área integrada: "+str(integral)
    gravar.write(string_Area+'\n')
    print string_Area
    string_log = nomearquivodados+';'+str(x_t
    [ponto025])+';'+str(float(centro_x))+';'+str
      (x_t[ponto975])+';'+ str(dados_med) +';'+ str
                 (dados_desvpad)+';'+str(integral)
    gravar_log.write(string_log+'\n')
    print somatorio_Prob
```

e, por último, o gráfico contendo o PDF é gerado:

```
#################Gera gráfico
    ax = subplot(111)
    quem = str(nomearquivodados)
    plot(x,y)
    grid()
    dados_cientifico= "%.5g" %(dados_med)
    titulo = "Sample: "+upper(quem) +"; Average
                      value. = "+ dados_cientifico

    title(titulo)
    nome_graf1 = nomefig
    ylabel('Probability density',size=18)
    xlabel(abscissa, size = 18)
    savefig(nome_graf1)
    a = x_int[ponto025]
    b = x_int[ponto975]

    # Gera região sombreada
    ix = x_int[ponto025:ponto975]
    iy = y_int[ponto025:ponto975]
    verts = [(a,0)] + zip(ix,iy) + [(b,0)]
    poly = Polygon(verts, facecolor='0.8',
                                    edgecolor='k')
    ax.add_patch(poly)
    a=float('%.2f'%a)
    centro_x=float('%.2f'%centro_x)
    b=float('%.2f'%b)
    ax.set_xticks((a,centro_x,b))
    nome_graf2 = nomefig+"-P95"
    savefig(nome_graf2)
    close('all')

    gravar.close
    gravar_pdf.close
```

```
    gravar_log.close
    print"Processo concluído!"
    #show()
```

Resultados:

Foram selecionados 20 valores entre 11 e 19, mostrados na Tabela 1, capazes de produzir uma distribuição bimodal.

Tabela 1 – Vinte valores entre 11 e 19 selecionados para o teste

Contagem	Valor
1	11
2	12
3	13
4	14
5	13
6	12
7	11
8	13
9	18
10	19
11	17
12	18
13	17
14	16
15	11
16	12
17	13
18	11
19	12
20	13

Os valores mostrados na Tabela 1 fora usados para construir um histograma, com o auxílio do software QtiPlot®, mostrado na Figura 2.

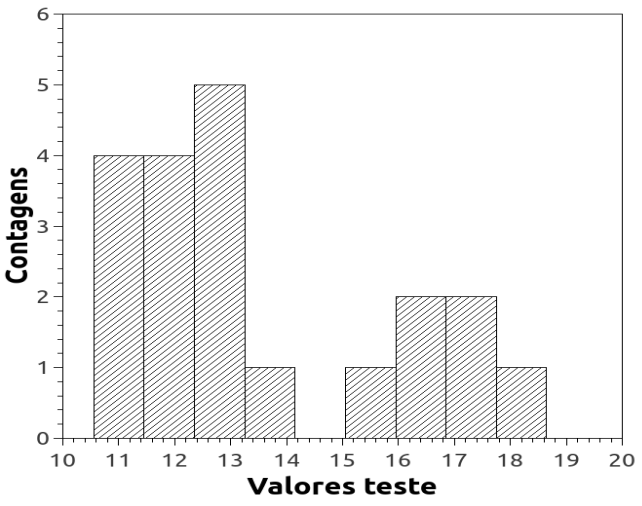


Figura 2 – Histograma gerado a partir dos dados da Tabela 1.

Em seguida, os dados foram salvos num arquivo denominado teste.txt e fornecidos ao programa PDF&Freq. Como primeira prova, foram geradas PDFs usando diferentes kernels. Os resultados são mostrados na Figura 3.

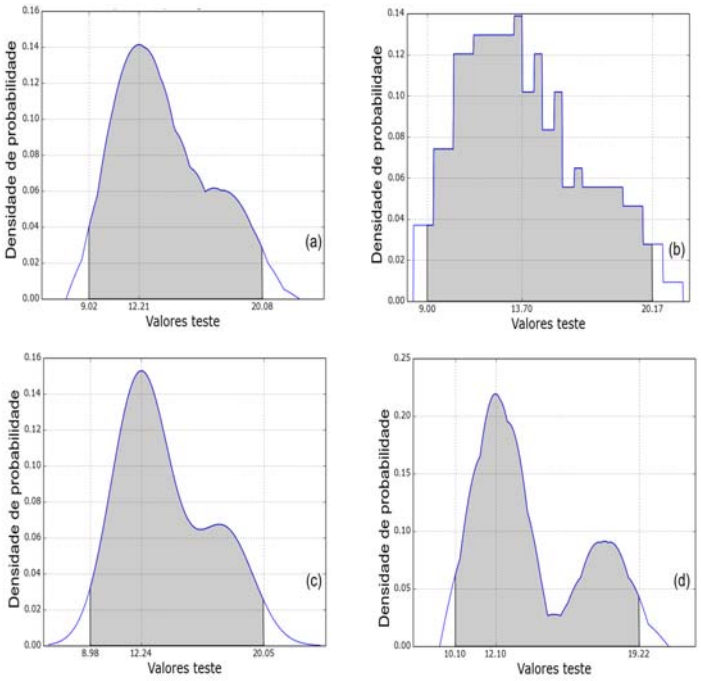


Figura 3 – PDFs obtidos usando os kernels Epanechnikov (a), Triangle (b), Gaussian (c) e Cosine (d)

Comprovada a influência da escolha do kernel sobre os resultados, os cálculos seguintes foram feitos usando o kernel Gaussiano. Os arquivos gerados pelo programa *PDF&Freq* são mostrados na Figura 4.

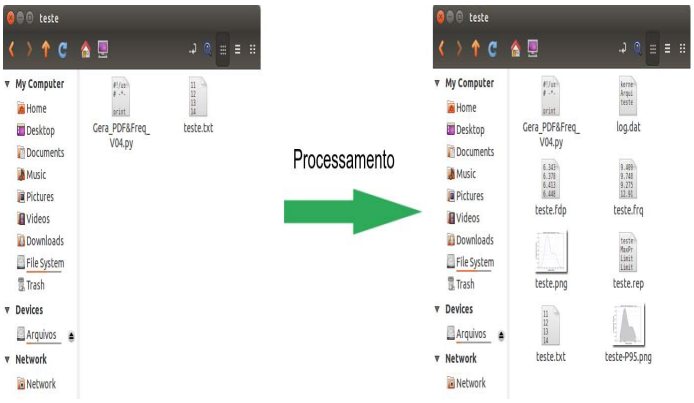


Figura 4 – Arquivos gerados pelo programa PDF&Freq.

Entre estes arquivos, os que contêm as principais respostas geradas pelo programa são três:

(i) Arquivo log.dat, onde são listados o kernel escolhido e as demais informações previamente descritas.

log.dat

kernel= g

Arquivo ; LI(95%) ; MAXPROB; LS(95%) ; Media ; DesvPad ; Area

teste;8.980685;12.2418703;20.0478976;13.8;2.66754371;0.999612524

(ii) Arquivo *.frq, onde estão listados os cinco valores mais prováveis tomados para os valores aleatório da variável prob quando esta mais se aproxima de 1.

Arquivo teste.frq

9.40974719

9.74840852

9.27565556

12.9108177

19.9949159

e por último, (iii) o arquivo *.rep, onde são listadas a ordenada e a abscissa referente à máxima probabilidade, bem como os limites de confiança, com 95% de probabilidade;

Arquivo teste.rep

MaxProb = 0.152862861298 @ 12.2418703658

Propriedade (95%LC) = (-3.26118472476;12.2418703658; +7.80602726672)

Assim, as principais informações estatísticas referentes à construção das PDFs são armazenadas numa única pasta, de maneira muito prática e rápida.

Conclusões

O maior impacto deste trabalho consiste na construção de uma ferramenta computacional que permite a rápida construção de distribuições de probabilidade a partir de diferentes kernels. Isso só foi possível com o uso da linguagem de programação Python, a qual permitiu desenvolver um rapidamente um programa capaz de lidar com diversos arquivos de dados, produzindo informações estatísticas valiosas na forma de tabelas e gráficos, os quais são fundamentais para o nosso grupo de pesquisa e para os demais grupos lidem com esse tipo de informação ou problema.

Agradecimentos

Os autores agradecem ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), à Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), à Financiadora de Estudos e Projetos (FINEP PRESAL Ref.1889/10) e à Fundação Carlos Chagas Filho de Amparo à Pesquisa do Estado do Rio de Janeiro (FAPERJ) por toda a ajuda financeira e pelas bolsas concedidas.

Bibliografia

1. Souza Jr., F. G., Pinto, J. C. & Soares, B. G. SBS/Pani · DBSA mixture plasticized with DOP and NCLS – Effect of the plasticizers on the probability density of volume resistivity measurements. *Eur. Polym. J.* **43,** 2007–2016 (2007).

2. Luce, B. & Anthony, O. *A Primer on Bayesian Statistics in Health Economics and Outcomes Research*. (MEDTAP International, Incorporated, 2003).

3. Bard, Y. *Nonlinear Parameter Estimation*. (Academic Press, 1974).

4. Grance, E. G. O. *et al.* New petroleum absorbers based on lignin□CNSL□formol magnetic nanocomposites. *J. Appl. Polym. Sci.* (2012). doi:10.1002/app.36998

5. Araújo, A., Botelho, G., Oliveira, M. & Machado, A. V. Influence of clay organic modifier on the thermal-stability of PLA based nanocomposites. *Appl. Clay Sci.* **88–89,** 144–150 (2014).

6. de Souza Junior, F. G. *et al.* Conducting and magnetic mango fibers. *Ind. Crops Prod.* **68,** 97–104 (2015).

7. Souza Jr., F. G. & Varela, A. Construção de ferramenta de aquisição e inspeção de dados eletromecânicos usando Python. *Programar - Rev. Port. Programação* **34,** 32–38 (2012).

8. SCHWAAB, M. *Análise de Dados Experimentais: I. Fundamentos de Estatística e Estimação de Parâmetros*. (Editora E-papers).

9. Kernel Density Estimation in Python. Available at: http://jakevdp.github.io/blog/2013/12/01/kernel-density-estimation/. (Accessed: 19th May 2016)

AUTOR

Escrito por Fernando Gomes de Souza Júnior

Bolsista de Produtividade em Pesquisa do CNPq - Nível 2 - CA EQ - Engenharia Química. Possui graduação em Química pela Universidade Federal do Espírito Santo (1999), mestrado em Engenharia e Ciência dos Materiais pela Universidade Estadual do Norte Fluminense Darcy Ribeiro (2002), doutorado em Ciência e Tecnologia de Polímeros pela Universidade Federal do Rio de Janeiro (2006) e Pós-Doutorado no Programa de Engenharia Química da COPPE / UFRJ. Atualmente é Professor Adjunto IV do Instituto de Macromoléculas da UFRJ, Professor Colaborador do Programa de Engenharia Civil da COPPE/UFRJ e Jovem Cientista do Estado do Rio de Janeiro (FAPERJ-2015). Atua principalmente com nanocompósitos poliméricos obtidos a partir de recursos renováveis em três principais linhas: (I) no campo de recuperação ambiental, coordenando projetos de pesquisa focados no uso de recursos renováveis para a remoção de petróleo em derramamentos; (II) no campo de saúde humana, coordenando projetos que buscam o controle cinético e espacial do processo de liberação de fármacos; e (III) no campo de sensores, onde coordena projetos que buscam a obtenção de fibras vegetais condutoras de eletricidade e o seu uso em sensores para dispositivos inteligentes.

http://lattes.cnpq.br/3049721573449880

Sistema de chat público em PHP

INTRODUÇÃO

Embora a base deste sistema seja o PHP, também serão utilizadas outras tecnologias. A nossa "caixa de ferramentas" tem, então, o seguinte conteúdo e a respetiva utilização:

- PHP: Linguagem de programação base;
- HTML: Estrutura das páginas;
- CSS: Design das páginas;
- jQuery/JavaScript: Utilização do AJAX;
- MySQL/MariaDB: Base de dados;
- Apache: Servidor web;

OBJETIVO

\No fim deste artigo o leitor terá uma visão abrangente do que é essencial para criar aplicações web e a ajuda necessária para começar a desbravar este mundo com o PHP.

FUNCIONALIDADES

O nosso sistema de chat terá as seguintes funcionalidades:

- Sala de chat única e pública;
- Escolha de um *nickname* exclusivo;
- Envio de mensagens;
- Consulta de mensagens;

"CAIXA DE FERRAMENTAS"

Para implementar este chat vamos utilizar várias ferramentas que se complementam umas às outras. Algumas destas ferramentas foram instaladas com o XAMPP, permitindo, assim, instalar facilmente um servidor web na nossa máquina local.

PHP

PHP, acrónimo recursivo para "PHP: Hypertext Preprocessor" (originalmente Personal Home Page) é uma linguagem de programação *server-side* de utilização livre e de código aberto. O PHP é, por exemplo, utilizado pelo Facebook e WordPress. Em 2014 foi a linguagem de eleição de 82% dos sites (onde a linguagem de programação é conhecida).

No sistema de chat, que vamos construir, o PHP tem como função a comunicação com a base de dados para armazenamento e consulta das mensagens.

MySQL

O MySQL é um sistema de gestão de base de dados (SGBD) que utiliza a linguagem SQL (Structured Query Language). A sua licença é livre para desenvolvimento, mas, caso seja utilizada em aplicações comerciais, passa a ser paga. Utilizada pela NASA, Friendster, HP, Google, entre outras empresas, é, atualmente, uma das bases de dados mais populares, com mais de 10 milhões de instalações em todo o mundo.

A função do MySQL, no nosso sistema de chat, será o armazenamento das mensagens dos utilizadores (também se pode utilizar o MariaDB). Serão utilizadas duas tabelas, uma para armazenar os *nicknames* e outra para armazenar as mensagens.

HTML

O HTML (HyperText Markup Language) é uma linguagem de marcação utilizada na construção de páginas web. Quando o PHP é interpretado no servidor produz o HTML que será lido pelo Browser (Os browsers não conhecem o PHP).

O HTML será, então, utilizado para criar a estrutura das nossas páginas.

CSS

O CSS (Cascading Style Sheets) é uma linguagem de folhas de estilo utilizada para definir a apresentação de páginas web.

No nosso sistema de chat, todas as cores, tamanhos, tipos de letra, etc. serão definidos no CSS.

jQuery / JavaScript

O jQuery é uma framework de JavaScript, de código aberto, desenvolvida para facilitar a programação no lado do cliente (Browser) e é utilizado por cerca de 77% dos sites mais visitados do mundo. O seu slogan é revelador: "Write Less, Do More" (por exemplo, 20 linhas de código em JavaScript podem ser substituídas por uma em jQuery).

O jQuery será usado no nosso chat para facilitar o uso de outra "tecnologia" denominada de AJAX (Asynchronous Javascript and XML) que, na verdade, é um conjunto de tecnologias que torna as páginas mais interativas.

Apache

O Apache é o servidor web livre melhor sucedido. Estima-se que foi utilizado por metade do sites em 2015.

É através do Apache que o nosso computador conse-

guirá interpretar o PHP.

Procedimento

Passo 1 – Instalação e configuração das ferramentas necessárias

1. Instalar o XAMPP

Fazer o download em https://www.apachefriends.org e executar a instalação.

1.1. Configurar o XAMPP

Abrir o *Control Panel* e iniciar o Apache. Depois de iniciado clique no botão "Admin" que deverá abrir uma página no Browser (página de apresentação do XAMPP). A configuração está concluída.

1.2 Testar o PHP

Abrir o editor de texto, por exemplo, o Notepad++ e escrever o seguinte script:

```
<?php
      echo 'Olá PHP';
?>
```

Guardar o script com o nome teste.php na pasta c:\xampp\htdocs\chat.

NOTA: é necessário criar a pasta chat

Depois de guardar o ficheiro no disco, digite o seguinte URL no browser: http://localhost/chat/teste.php

Apareceu o texto "Olá PHP"? Então o PHP está a funcionar.

2. Base de dados

Vamos agora colocar a base de dados em funcionamento. Obviamente, será aqui que serão armazenadas todas as conversas.

2.1. Criação da base de dados

Com o *Control Panel* do XAMPP aberto, clique no botão "Admin" referente ao MySQL que abrirá o *browser* no *phpMyAdmin*, o que nos permitirá criar uma base de dados e o respetivo utilizador. Para tal, deverá clicar na aba "User accounts" e clicar no link "Add user account". Vamos então escrever os seguintes dados onde diz "Informação de login":

- Nome de usuário: chat
- Host name: localhost
- Senha: programar
- Re-digite: programar

Onde diz "Database for user account" selecionar a *checkbox* que diz:

- Criar banco de dados com o mesmo nome e conceder todos os privilégios.

No fim da página, clique no botão "Executar". Neste momento, a base de dados e o respetivo utilizador estão criados.

2.2 Criação das tabelas

No lado esquerdo selecione a tabela "chat" e no lado direito selecione a aba "SQL". Onde diz "Rodar consulta(s) SQL no banco de dados chat" fazer copiar/colar das seguintes *queries*:

```
CREATE TABLE `messages` (
  `IdMessage` int(8) NOT NULL COMMENT '(PK) ID da
                                        mensagem',
  `FromNickname` varchar(32) NOT NULL COMMENT
                        'Nickname do remetente',
  `Message` text COMMENT 'Mensagem',
  `MessageDate` timestamp NOT NULL DEFAULT CUR
      RENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP
                      COMMENT 'Data da mensagem'
            ) ENGINE=MyISAM DEFAULT CHARSET=utf8
                    COMMENT='Mensagens do chat';

CREATE TABLE `users` (
  `Nickname` varchar(32) NOT NULL COMMENT '(PK)
                        Nickname do utilizador',
           `LoginDate` timestamp NOT NULL DEFAULT
    CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP
                        COMMENT 'Data do login'
            ) ENGINE=MyISAM DEFAULT CHARSET=utf8
                COMMENT='Utilizadores do chat';

ALTER TABLE `messages`
  ADD PRIMARY KEY (`IdMessage`);

ALTER TABLE `users`
  ADD PRIMARY KEY (`Nickname`);

ALTER TABLE `messages`
  MODIFY `IdMessage` int(8) NOT NULL AU-
TO_INCREMENT
                    COMMENT '(PK) ID da mensagem';
```

Depois de carregar no botão "Executar" será criada a tabela users que armazenará os *nicknames* dos utilizadores e a tabela messages que armazenará as mensagens.

3. Script das constantes

Crie um ficheiro com o nome constants.inc.php e copie o seguinte conteúdo:

```
<?php
# Base de dados
define('DB_HOSTNAME', 'localhost');
define('DB_USERNAME', 'chat');
define('DB_PASSWORD', 'programar');
define('DB_NAME', 'chat');

# Tabelas
// Utilizadores do chat
define('USERS', 'users');
// Mensagens do chat
define('MESSAGES', 'messages');
```

Este ficheiro deve ser guardado na pasta chat. Ele contém os dados necessários para a comunicação com a

base de dados.

Mais tarde, vamos testar se a ligação à base de dados está a funcionar, mas antes disso vamos precisar de mais alguns scripts.

Passo 2 – Criação das classes

Optou-se pela programação orientada a objetos, modelando, assim, o sistema de chat através das classes Users e Chat.

Vamos utilizar a extensão PDO (PHP Data Objects) para comunicar com a base de dados. Para o efeito será criada a classe DbConnPDO, que é uma extensão da classe PDO, que já se encontra instalada no PHP.

1. Classe DbConnPDO

A classe DbConnPDO representa a ligação à base de dados. Vamos utilizar o PDO através da classe DbConnPDO (extensão da classe PDO, que já se encontra instalada no PHP) para comunicar com a base de dados.

A classe DbConnPDO tem como única função a ligação à base de dados. Esta ligação é efetuada no seu construtor utilizando as constantes definidas, anteriormente, no script constants.inc.php.

Vamos, então, criar um ficheiro na pasta chat com o nome DbConnPDO.class.php e escrever o seguinte código:

```php
<?php
/**
 * PDO Database connection
 * Representa a ligação global à base de dados
 *
 * @package Chat\PDO
 * @author Sandro Miguel Marques
<sandromiguel@produlogia.com>
 * @version v.1.0 (13/01/2016)
 */

class DbConnPDO extends PDO {
 /** @var string|null Mensagem de erro */
 private $_error = null;

 /**
 * Construtor
 * Abrir a ligação à base de dados
 */
 public function __construct() {
 $connection_string = sprintf('mysql:host=%s;
    dbname=%s;charset=UTF8', DB_HOSTNAME, DB_NAME);
 try {
 parent::__construct($connection_string,
                          DB_USERNAME, DB_PASSWORD);
 $this->setAttribute(PDO::ATTR_ERRMODE,
                            PDO::ERRMODE_EXCEPTION);
 } catch (PDOException $e) {
 $this->error = $e->getMessage();
 throw $e;
 }
 }
 /**
 * Mostra o erro.
 *
 * @return string Devolve o erro.
 */
 public function getError() {
 return $this->_error;
 }

 /**
 * Fecha a ligação à base de dados quando o
                                  objeto é destruido
 */
 public function __destruct() {
 $this->conn = null;
 }
}
```

2. Classe User

A classe User representa os utilizadores do chat. Uma vez que esta classe faz ligação à base de dados, estende a classe DbConnPDO. O método checkNicknameExists () verifica se um determinado *nickname* existe. O método insert() insere o *nickname* na base de dados, nomeadamente, na tabela users.

Cria-se agora o ficheiro na pasta chat com o nome User.class.php e escreve-se o seguinte código:

```php
<?php
/**
 * Utilizador do chat
 *
 * @package Chat\User
 * @author Sandro Miguel Marques
<sandromiguel@produlogia.com>
 * @version v.1.0 (06/04/2016)
 * @copyright Copyright (c) 2016, Sandro
 */

class User extends DbConnPDO {
 /** @var string|null Query SQL */
 private $_sql;

 /**
 * Construtor
 */
 public function __construct() {
 parent::__construct();
 }

 /**
 * Verifica se um determinado nickname existe.
 *
 * @param string $nickname Nickname do utilizador
 *
 * @return boolean Devolve 'true' se o nickname
já existir.
 */
 public function checkNicknameExists($nickname) {
 try {
 $this->_sql = '
 SELECT
 NIckname
 FROM
 `'. USERS .'`
 WHERE
 `Nickname`=:nickname
 LIMIT 1
 ';
 $stmt = $this->prepare($this->_sql);
 $stmt->bindParam(':nickname', $nickname,
                                    PDO::PARAM_STR);
 $stmt->execute();
 $num_rows = $stmt->rowCount();
```

```
 if ($num_rows == 0) {
 return false;
 }
 return true;
 } catch (Exception $e) {
 throw $e;
 }
 }

 /**
 * Insere um utilizador da base de dados.
 *
 * @param string $nickname Nickname
 *
 * @return boolean Devolve 'true' em caso de
                sucesso ou 'false' em caso de erro.
 */
 public function insert($nickname) {
 $this->_sql = 'INSERT INTO `'.USERS.'`
                (Nickname)  VALUES (:nickname)  ';
 try {
 $stmt = $this->prepare($this->_sql);
 $stmt->bindParam(':nickname', $nickname,
PDO::PARAM_STR);
 if ($stmt->execute()) {
 $stmt->closeCursor();
 return true;
 }
 $stmt->closeCursor();
 return false;
 } catch (Exception $e) {
 throw $e;
 }
 }

 /**
 * @return string Devolve a query SQL.
 */
 public function __toString() {
 if (is_null($this->_sql)) {
 return 'NULL';
 }
 return $this->_sql;
 }
}
```

3. Classe Chat

A classe Chat representa o chat propriamente dito. Tal classe estende a classe DbConnPDO pois, também, faz ligação à base de dados. Para inserir as mensagens na tabela messages utiliza-se o método insert() e para as ler utiliza-se o método getMessages().

Para esta classe criamos um ficheiro na pasta chat com o nome Chat.class.php, com o seguinte código:

```
<?php
/**
 * Utilizador do chat
 *
 * @package Chat\User
 * @author Sandro Miguel Marques
<sandromiguel@produlogia.com>
 * @version v.1.0 (06/04/2016)
 * @copyright Copyright (c) 2016, Sandro
 */

class User extends DbConnPDO {
 /** @var string|null Query SQL */
 private $_sql;


 /**
 * Construtor
 */
 public function __construct() {
 parent::__construct();
 }

 /**
 * Verifica se um determinado nickname existe.
 *
 * @param string $nickname Nickname do utilizador
 *
 * @return boolean Devolve 'true' se o nickname
                                        já existir.
 */
 public function checkNicknameExists($nickname) {
 try {
 $this->_sql = '
 SELECT
 NIckname
 FROM
 `'. USERS .'`
 WHERE
 `Nickname`=:nickname
 LIMIT 1
 ';
 $stmt = $this->prepare($this->_sql);
 $stmt->bindParam(':nickname', $nickname,
PDO::PARAM_STR);
 $stmt->execute();
 $num_rows = $stmt->rowCount();
 if ($num_rows == 0) {
 return false;
 }
 return true;
 } catch (Exception $e) {
 throw $e;
 }
 }

 /**
 * Insere um utilizador da base de dados.
 *
 * @param string $nickname Nickname
 *
 * @return boolean Devolve 'true' em caso de
                sucesso ou 'false' em caso de erro.
 */
 public function insert($nickname) {
 $this->_sql = 'INSERT INTO `'.USERS.'`
                (Nickname)  VALUES  (:nickname) ';
 try {
 $stmt = $this->prepare($this->_sql);
 $stmt->bindParam(':nickname', $nickname,
                                        PDO::PARAM_STR);
 if ($stmt->execute()) {
 $stmt->closeCursor();
 return true;
 }
 $stmt->closeCursor();
 return false;
 } catch (Exception $e) {
 throw $e;
 }
 }

 /**
 * @return string Devolve a query SQL.
 */
 public function __toString() {
 if (is_null($this->_sql)) {
 return 'NULL';
 }
 return $this->_sql;
 }}
```

4. Script para testar a ligação à base de dados

Já temos os scripts necessários para fazer o teste de ligação à base de dados. Crie um ficheiro com o nome teste2.php e escreva o seguinte código:

```
<?php
error_reporting(E_ALL);
ini_set('display_errors','On');

require 'constants.inc.php';
require 'DbConnPDO.class.php';
require 'Chat.class.php';

# 1. Ler as mensagens da base de dados
try {
 $chat = new Chat();
 $messages = $chat->getMessages(10);
 echo json_encode($messages);
} catch (Exception $e) {
 $message = 'Ocorreu um erro.';
 echo json_encode(
 array(
 "action" => "insert",
 "notification" => "error",
 "message" => $message
 )
 );

 exit;
}
```

Ao executar este script, caso não existam erros, deverão aparecer dois parêntesis retos [].

Passo 3 – Comunicação via AJAX

Para comunicar com a base de dados vamos utilizar o AJAX com JSON através do jQuery.

1. Definir o nickname

O script nickname.ajax.php é executado quando o utilizador introduz o seu *nickname*. Este script verifica, então, se o *nickname* contém pelo menos três caracteres, e se já existe. Caso o *nickname* não exista, o script insere-o na base de dados.

O script nickname.ajax.php contém o seguinte código:

```
<?php
session_start();

require 'constants.inc.php';
require 'DbConnPDO.class.php';
require 'User.class.php';

// Definir uma variável com o nickname recebido
pelo método POST
$nickname = filter_input(INPUT_POST, 'nickname',
                         FILTER_SANITIZE_STRING);

# 1. Verificar se o nickname contém o número mínimo
                                        de carateres
if (strlen($nickname) < 3) {
 $message = 'O nickname deve ter no mínimo 3
                                        carateres';
 echo json_encode(
 array(
 'action' => 'insert',
 'notification' => 'error',
 'message' => $message
 )
 );
 exit;
}

# 2. Verificar se o nickname já existe na base de
                                                dados
try {
 $user = new User();
 $nickname_exists = $user->checkNicknameExists
                                        ($nickname);
 if ($nickname_exists) {
 $message = 'Este nickname já existe.';
 echo json_encode(
 array(
 'action' => 'insert',
 'notification' => 'error',
 'message' => $message
 )
 );
 exit;
 }
} catch (Exception $e) {
 $message = 'Ocorreu um erro.';
 echo json_encode(
 array(
 "action" => "insert",
 "notification" => "error",
 "message" => $message
 )
 );
 exit;
}

# 3. Inserir o utilizador na base de dados
try {
 $user->insert($nickname);
 $_SESSION['nickname'] = $nickname;
 echo json_encode(
 array(
 'action' => 'replace',
 'notification' => 'success',
 'message' => 'Olá '.$nickname.', aguarde por
                                        favor...'
 )
 );
} catch (Exception $e) {
 $message = 'Ocorreu um erro.';
 echo json_encode(
 array(
 "action" => "insert",
 "notification" => "error",
 "message" => $message
 )
 );
 exit;
}
```

2. Inserir mensagem

Sempre que é enviada uma mensagem é executado o script set_message.ajax.php que insere a mensagem e o respetivo *nickname* na base de dados.

O código deste script é o seguinte:

```
<?php
require 'constants.inc.php';
require 'DbConnPDO.class.php';
require 'Chat.class.php';

$nickname = filter_input(INPUT_POST, 'nickname',
FILTER_SANITIZE_STRING);
```

```
$message = filter_input(INPUT_POST, 'message',
                        FILTER_SANITIZE_STRING);

# 1. Inserir a mensagen na base de dados
try {
 $chat = new Chat();
 $chat->insert($nickname, $message);
} catch (Exception $e) {
 echo 'ocorreu um erro ao fazer o INSERT na
BD<br>';
 echo $e->getMessage().'<br>';
 echo $chat->__toString();
}
```

3. Ler as mensagens

Para ler as mensagens do chat é executado o script get_messages.ajax.php que devolve as mensagens no formato JSON através da função json_encode() do PHP. No caso deste *chat* são devolvidas as últimas 10 mensagens, mas este número pode ser definido, por exemplo, no script constants.inc.php bastando adicionar mais uma constante.

Aqui está o código para este script:

```
<?php
error_reporting(E_ALL);
ini_set('display_errors','On');

require 'constants.inc.php';
require 'DbConnPDO.class.php';
require 'Chat.class.php';

# 1. Ler as mensagens da base de dados
try {
 $chat = new Chat();
 $messages = $chat->getMessages(10);
 echo json_encode($messages);
} catch (Exception $e) {
 $message = 'Ocorreu um erro.';
 echo json_encode(
 array(
 "action" => "insert",
 "notification" => "error",
 "message" => $message
 )
 );
 exit;
}
```

Passo 4 – Interface

A nossa interface será provida de uma página inicial onde se introduz o *nickname* e a página do chat.

Na página inicial o utilizador é convidado a introduzir o seu *nickname*. A submissão do formulário é feita pelo jQuery através da função ajax(), nativa do jQuery, que envia o *nickname* introduzido para o script nickname.ajax.php. Caso o *nickname* não esteja a ser utilizado o utilizador é redirecionado para o script chat.php.

Segue-se o código do ficheiro index.php:

```
<?php
session_start();

error_reporting(E_ALL);
ini_set('display_errors','On');
?>
<!doctype html>
<html lang="pt">
<head>
 <meta charset="utf-8">
 <title>Chat público</title>
 <meta name="description" content="Chat públi-
co" />
 <meta name="author" content="Sandro Marques">
 <meta name="viewport" content="width=device-
width, initial-scale=1">
 <link href="styles.css" rel="stylesheet" />
 <script src="https://ajax.googleapis.com/ajax/
libs/jquery/1.12.2/jquery.min.js"></script>
</head>

<body>
 <div class="outer">
 <div class="middle">
 <div class="inner">
 <div id="logo"></div>
 <div id="container_nickname">
 <div id="content_insert"></div>
 <div id="content_replace">
 <form name="form_nickname" id="form_nickname"
method="post" action="nickname.ajax.php">
 <p>Nickname</p>
 <input name="nickname" type="text" autofocus re-
quired class="general_textbox" id="nickname"
maxlength="32">
 <button>Entrar</button>
 </form>
 </div>
 </div>
 </div>
 </div>
 </div>
</body>

<script>
    'use strict';
    function resetContentInsert() {
        if ($('#content_insert').children
().length > 0) {
            // existe uma mensagem > remover con-
teúdo com animação
            $('#content_insert').animate({
                height: 0
            }, "fast", function () {
                $(this).empty();
                $('#content_insert').removeAttr
('style');
            });
        }
    }
    $(document).ready(function () {
        // submeter formulário pela função
                                        sendForm()
```

```
    $('#form_nickname').on('submit', function (e)
{
        e.preventDefault();
        sendForm();
    });

    function sendForm() {
        var msg_error = 'Ocorreu um erro';
        var msg_timeout = 'O servidor não está
                                    a responder';
        var message = '';
        var form = $('#form_nickname');
        resetContentInsert();
        $.ajax({
            data: form.serialize(),
            url: form.attr('action'),
            type: form.attr('method'),
            dataType: "json",
          error: function (xhr, status, error)
{
                if (status === "timeout") {
                    message = msg_timeout;
                    message = '<div class=
              "bg-error">' + message + '</div>';
                    $('#content_insert').empty()
          .html(message).hide().fadeIn('slow');
                } else {
                    message = msg_error;
                    message = '<div class=
              "bg-error">' + message + '</div>';
                    $('#content_insert').empty()
          .html(message).hide().fadeIn('slow');
                }
            },
            success: function (response) {
                var action = response.action;
                var notification =
                        response.notification;
                var bg_notification = null;
                switch (notification) {
                    case 'success':
                        bg_notification =
                                'bg-success';
                        break;
                    case 'error':
                        bg_notification =
                                'bg-error';
                        break;
                }
                message = '<div class="' +
      bg_notification + '">' + response.message +
                                '</div>';
                if (action === 'insert') {
                 $('#content_insert').finish();
                    $('#content_insert')
                        .removeAttr('style');
                  $('#content_insert').empty()
          .html(message).hide().fadeIn('slow');
                } else {
                  $('#content_replace').empty()
          .html(message).hide().fadeIn('slow');
                    setTimeout(function ()
{ window.location = "chat.php" }, 1000);
                }
            },
            timeout: 7000
        });

    }

});

</script>

</html>
```

2. Página do chat

Na página do chat o utilizador pode, efetivamente, ver e enviar mensagens. A submissão do formulário, ou seja, a submissão do texto da mensagem, é realizada para o script set_message.ajax.php, através da função ajax() (nativa do jQuery), invocada pela função sendForm(). Neste script o jQuery é, também, utilizado para ler as últimas mensagens do chat através da função getMessages(). Esta função é recursiva, chamando-se a si própria de dois em dois segundos.

NOTA: para simular vários utilizadores no chat, o leitor pode abrir a página em diferentes browsers.

Apresenta-se de seguida o código do script chat.php:

```
<?php
session_start();

error_reporting(E_ALL);
ini_set('display_errors','On');

require 'constants.inc.php';
require 'DbConnPDO.class.php';
?>
<!doctype html>
<html lang="pt">
<head>
 <meta charset="utf-8">
 <title>Chat público</title>
 <meta name="description" content="Chat públi-
co" />
 <meta name="author" content="Sandro Marques">
 <meta name="viewport" content="width=
                device-width, initial-scale=1">
 <link href="styles.css" rel="stylesheet" />
 <script src="https://ajax.googleapis.com/ajax/
      libs/jquery/1.12.2/jquery.min.js"></script>
</head>

<body>
<div id="sidebar"></div>
<div id="primary">
 <div id="log">
 <span class="long-content"> </span>
 </div>
 <div id="composer">
 <form name="form_message" id="form_message"
```

```
method="post" action="set_message.ajax.php">
 <input name="nickname" type="hidden" id="nickname"
      value="<?php echo $_SESSION['nickname']; ?>">
 <input name="message" type="text" autofocus
     required class="textbox_message" id="message">
 <button id="btn_send">Enviar</button>
 </form>
 </div>
</div>
</body>
<script>
    'use strict';
    $(document).ready(function () {
        getMessages();

        function getMessages() {
            $('.long-content').empty();
            var msg_error = 'Ocorreu um erro...';
            var msg_timeout = 'O servidor não está
                                    a responder';
            var message = '';
            $.ajax({
                url: 'get_messages.ajax.php',
                dataType: "json",
                error: function (xhr, status,
                                        error) {
                    if (status === "timeout") {
                        message = msg_timeout;
                        alert(message);
                    } else {
                        message = msg_error;
                        alert(message + ': ' +
                                        error);
                    }
                },
                success: function (response) {
                    $.each(response, function
                                        (i, item) {
                        $('.long-content').prepend
         ('<p><b>' + item.FromNickname + '</b>: ' +
                            item.Message + '</p>');
                    });
                    setTimeout(getMessages, 2000);
                },
                timeout: 7000
            });
        }


      // submeter formulário pela função sendForm()
      $('#form_message').on('submit', function (e)
{
            e.preventDefault();
            sendForm();
        });

        function sendForm() {
            var msg_error = 'Ocorreu um erro..';
            var msg_timeout = 'O servidor não está
                                    a responder';
            var message = '';
            var form = $('#form_message');
            $.ajax({
                data: form.serialize(),
                url: form.attr('action'),
                type: form.attr('method')
            })
            $('#message').val('');
        }

    });

</script>

</html>
```

3. Estilos CSS

Para estilizar as páginas no chat foi utilizado o ficheiro CSS styles.css. Este ficheiro está disponível para download na íntegra e importa salientar as seguintes partes:

CSS Reset

O CSS Reset é uma técnica que "normaliza" o CSS de forma a manter o visual do site em qualquer browser.

CSS responsivo

Foram utilizadas as *Media Queries* para adaptar o layout do chat às dimensões dos vários dispositivos. O seguinte exemplo ilustra a técnica utilizada para alterar a largura do "formulário para a introdução do *nickname*", consoante a largura do *viewport* do *browser*:

```
/* Formulário nickname centrado */
.outer {
 display: table;
 position: absolute;
 height: 100%;
 width: 100%;
}
.middle {
 display: table-cell;
 vertical-align: middle;
}
.inner {
 margin-left: auto;
 margin-right: auto;
 width: 90%;
}
@media (min-width: 768px) {
 /* Small devices (tablets, 768px and up) */
 .inner {
 width: 40%;
 }
}
@media (min-width: 992px) {
 /* Medium devices (desktops, 992px and up) */
 .inner {
 width: 35%;
 }
}
@media (min-width: 1200px) {
 /* Large devices (large desktops, 1200px and up)
*/

 .inner {

 width: 30%;

 }

}
```

O que ficou por fazer?

Todos os sistemas têm sempre algo a melhorar e este não é exceção. Então, o que ficou por fazer?

No que diz respeito à segurança, a aplicação está protegida contra *SQL injection* mas não foram tidos em conta outros potenciais problemas tais como *XSS*, *CSRF*, etc.

A nível de performance este chat poderá ser melhorado de forma a que o servidor notifique cada cliente (push)

sempre que há mensagens novas. Isto pode ser feito através de *websockets*, ou outras técnicas.

Também poderão ser implementadas novas funcionalidades (mostrar os utilizadores ativos, criar outras salas de chat, públicas ou privadas, registar utilizadores, definir uma imagem ou *avatar*, verificar se o utilizador ainda está ativo, implementar *emoticons*, enviar mensagens privadas, etc.).

Conclusão

Neste artigo apresentou-se o desenvolvimento de um pequeno sistema de chat. Tratou-se de um sistema simples, mas que abordou técnicas avançadas, tais como POO, AJAX e JSON.

Código fonte da aplicação: https://github.com/SandroMiguel/public-chat

Bibliografia

Benedetti, R., & Cranley, R. (2011). *Head First jQuery.* Sebastopol: O'Reilly Media, Inc.

Curioso, A., Bradford, R., & Galbraith, P. (2010). *Expert PHP and MySQL®.* Indianapolis: Wiley Publishing, Inc.

Hansen, T., & Lengstorf, J. (2014). *PHP for Absolute Beginners.* New York: Apress.

Kennedy, A., & León, I. d. (2011). *Pro CSS for High Traffic Websites.* New York: Apress.

MacIntyre, P. B. (2010). *PHP: The Good Parts.* Sebastopol: O'Reilly Media, Inc.

MacIntyre, P., Danchilla, B., & Gogala, M. (2011). *Pro PHP Programming.* New York: Apress.

Powers, D. (2014). *PHP Solutions: Dynamic Web Design Made Easy* (3rd ed.). New York: Apress.

Powers, S. (2015). *JavaScript Cookbook* (2nd ed.). Sebastopol: O'Reilly Media, Inc.

Prettyman, S. (2016). *Learn PHP 7: Object-Oriented Modular Programming using HTML5, CSS3, JavaScript, XML, JSON, and MySQL.* Georgia: Apress.

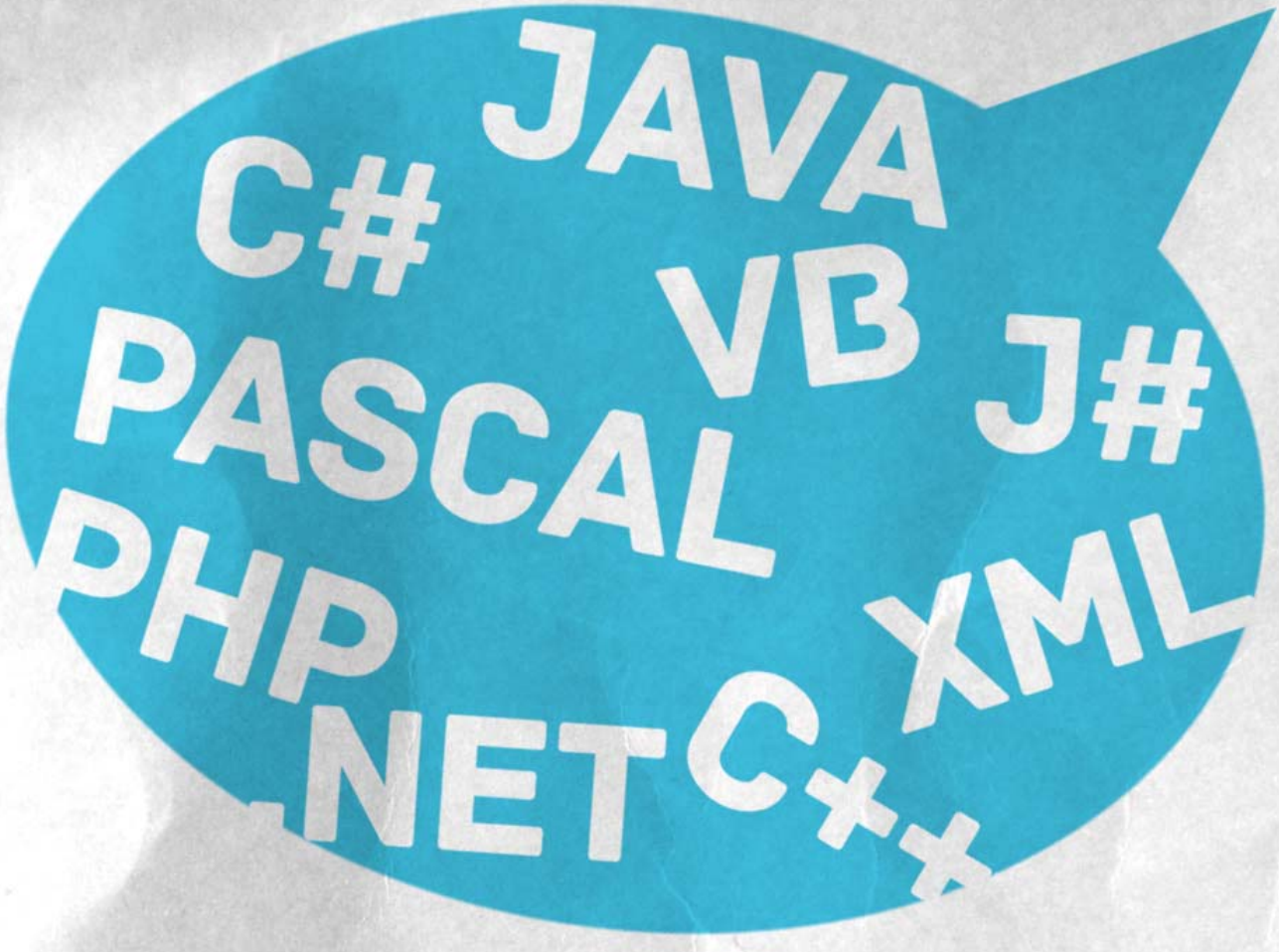
Tatroe, K., MacIntyre, P., & Lerdorf, R. (2013). *Programming PHP* (3ª ed.). Sebastopol: O'Reilly Media, Inc.

Ullman, L. (2012). *PHP and MySQL for Dynamic Web Sites* (4th ed.). Berkeley: Peachpit Press.

AUTOR

Escrito por Sandro Miguel Marques

Apaixonado pela programação e web developer há mais de 10 anos, é finalista da licenciatura em engenharia informática no Instituto Politécnico da Guarda. É colaborador ativo dos sites Stack Overflow e GitHub.



ENTÃO, SÓ FALAS EM CÓDIGO?

TEMOS O REMÉDIO CERTO PARA TI!

portugal-a-programar.pt

A MAIOR COMUNIDADE PORTUGUESA DE PROGRAMAÇÃO, APARECE!

O meu primeiro Jogo em MonoGame

Muitos programadores chegaram ao mundo da programação através do fascínio do desenvolvimento de Jogos. Desde as cassetes de ZX Spectrum que demoravam eternidades a carregar e a criação de jogos era uma tarefa muitas vezes hercúlea até aos dias de hoje, a criação de jogos percorreu um longo caminho e hoje podemos encontrar várias plataformas dedicadas ao seu desenvolvimento.

Para facilitar a criação de jogos para múltiplas plataformas foi criada a framework MonoGame, baseada na framework XNA da Microsoft, que apresenta uma grande facilidade de aprendizagem. Seguindo o princípio “Escreve uma vez, corre em todo o lado”, ao desenvolvermos um jogo com Monogame, ele irá correr em OS, Android, Mac OS X, tvOS, Windows, Linux, Playstation4 e mais.

Neste artigo vamos criar um jogo do princípio ao fim, passo a passo, desde a criação do interface de utilizador até ao adicionar da lógica de jogo.

À semelhança do Jogo da Toupeira (Whack-a-Mole), o objectivo do jogo é tocar nos bonecos à medida que vão aparecendo, evitando que fujam com as revistas. Se um boneco ficar sem ser tocado por mais de cinco segundos, ele leva as revistas e o jogo acaba.

Ficheiro -> Novo Jogo

Antes de dar início ao desenvolvimento do jogo, devemos verificar se temos o Xamarin instalado bem como a última build de desenvolvimento do MonoGame. Os passos que vamos utilizar aplicam-se tanto a Xamarin Studio como a MonoGame no Visual Studio.

Vamos primeiro cirar um projecto partilhado onde iremos guardar toda a lógica de jogo, podendo este ser também partilhado com todas as plataformas alvo em que queremos correr o jogo. Dentro do Xamarin Studio, escolhemos File-> New Solution -> MonoGame -> Library-> MonoGame Shared Library, e damos nome ao projecto PapTap. Não nos podemos esquecer de adicionar projectos para todas as plataformas que queremos para o jogo.

Adicionamos um projecto “MonoGame for Android Application” e damos-lhe o nome “PapTap.Android”. Uma vez que o nosso jogo não irá tirar partido das configurações nativas do Android, como o NFC e outros componentes, podemos apagar a classe Game1 criada no projecto Android e adicionar uma referência ao projecto partilhado que criamos antes.

Para correr a aplicação no emulador seleccionado ou no Xamarin Android Player, pressionamos F5 ou Cmd+Enter. Deverá surgir um lindo ecrã azul.

Agora que configurámos a nossa solução devidamente, vamos começar a escrever o nosso jogo.

Anatomia de um Jogo

Para criar um jogo é necessário que vários componentes trabalhem em conjunto. A classe Game1 contém toda a lógica do jogo e é constituída por cinco métodos principais:

- Constructor
- Initialize
- LoadContent
- Update
- Draw

Cada um deles tem o objectivo de garantir que o jogo funciona devidamente, desde os efeitos sonoros à resposta ao input do utilizador e à execução da lógica do jogo.

Os métodos Constructor e Initialize são utilizados para desempenhar qualquer inicialização que o jogo necessite antes de começar a correr. O LoadContent tem a função de carregar qualquer conteúdo do jogo tal como texturas, sons, sombras, e outros componentes gráficos ou sonoros. O Update é utilizado para actualizar qualquer lógica de jogo enquanto o jogo é executado (recolher o input do utilizador ou actualizar o mundo), enquanto que o Draw deve ser utilizado exclusivamente para desenhar quaisquer gráficos que precisem ser exibidos.

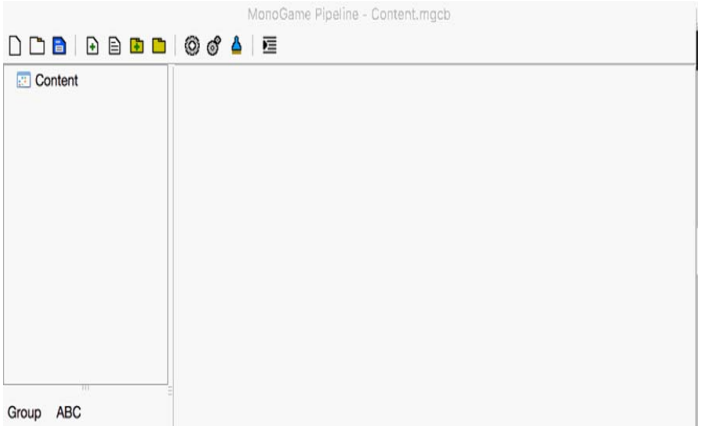
Adicionar Items de Jogo

Para um jogo estar completo, necessita de texturas e efeitos sonoros. Estes são conhecidos no desenvolvimento de jogos como “contente” ou “assets”. A maior parte das frameworks de jogos possui uma pipeline de conteúdo, a qual é utilizada apenas para pegar em items que estão em bruto e transformá-los num formato optimizado para o jogo. Na pasta Content, o ficheiro Content.mgcb é a pipeline de conteúdo do MonoGame. Tudo o que for adicionado a este ficheiro será optimizado e incluído no pacote final da aplicação.

Mais uma vez, todos os items do jogo podem ser partilhados entre as plataformas alvo. Assim, arrastamos o Content Directory do projecto Android para o Projeto Partilhado. Devemos certificar-nos que a Build Action do ficheiro Content.mcgb está definida para MonoGameContentReference. Se a Build Action não aparecer nas opções clicamos com o botao direito do rato no ficheiro Content.mcgb, selecionamos Propriedades e inserimos manualmente o texto toMonoGameContentReference. Em seguinda descarregamso os itmes PapTap e fazemos a extração para a pasta Content do Projeto Partilhado.

No MonoGame encontramos um Editor de Pipeline especial que facilita imenso o trabalho com os items do jogo.

Fazemos duplo clique no ficheiro Content.mcgb para abrir o editor de Pipeline e adicionamos os ficherios de items que acabámos de descarregar.



Agora que o conteúdo do jogo está optimizado para uso na nossa aplicação, podemos usar os items no nosso jogo.

Criando o Interface de Utilizador do PapTap

Para carregar os itmes de jogo recorremos a um ContentManager que é exposto por default através da propriedade "Content" da classe Game. Para começar vamos importar alguns namespaces necessários e declarar campos para armazenar os items de jogo:

```
using System.Collections.Generic;
using Microsoft.Xna.Framework.Audio;
using Microsoft.Xna.Framework.Input.Touch;
using Microsoft.Xna.Framework.Media;
public class Game1 : Game
{

    GraphicsDeviceManager graphics;
    SpriteBatch spriteBatch;
    Texture2D monkey;
    Texture2D background;
    Texture2D logo;
    SpriteFont font;
    SoundEffect hit;
    Song title;
}
```

De seguida é necessário carregar os items. Para isso adicionamos o seguinte código ao método LoadContent, pois é onde todos os items devem ser carregados em MonoGame:

```
monkey = Content.Load ("monkey");
background = Content.Load  ("background");
logo = Content.Load ("logo");
font = Content.Load ("font");
hit = Content.Load ("hit");
title = Content.Load ("title");
MediaPlayer.IsRepeating = true;
MediaPlayer.Play (title);
```

Agora que carregámos o nosso conteúdo, desde as texturas ao áudio, é hora de desenhar a interface do utilizador no ecrã. Para desenhar imagens 2D e texto usamos a classe SpriteBatch. Para fazer o rendering de forma eficiente, o desenho é compactado junto e as sprites devem ser desenhadas entre os métodos da SpriteBatch, Begin e End. De seguida atualizamos o método Draw para desenhar a textura do boneco no ecrã, utilizando o campo SpriteBatch que acabámos de criar:

```
protected override void Draw(GameTime gameTime)
{
    graphics.GraphicsDevice.Clear
(Color.CornflowerBlue);

    spriteBatch.Begin();
    spriteBatch.Draw(monkey, Vector2.Zero);
    spriteBatch.End();

    base.Draw(gameTime);
}
```

Para correr a aplicação carregamos Ctrl+Enter ou na tecla F5. Deveremos ver o boneco com a revista e ouvir a música que definimos no LoadContent. Agora que já temos os items de jogo a carregar, vamos construir o restante do interface do utilizador para jogar PapTap.

Construinndo o Interface do Utilizador do PapTap

Habitualmente nos jogos tradicionais do estilo Whack-a-Mole aparecem toupeiras aleatoriamente no ecrã e devem ser clicadas para que desapareçam. Em vez de fazermos o rendering aleatório de bonecos no ecrã, podemos utilizar uma grelha para ajudar e garantir que os bonecos não se sobrepõem de forma a que a experiência do utilizador seja consistente. A grelha é constituída por várias células. Cada célula contém um rectângulo, cor, temporizador decrescente, e valor de transição que será utilizado para fazer o fade in do boneco. Vamos então copiar e colar para o ficheiro Game1.cs a seguinte classe GridCell:

```
public class GridCell
{
    public Rectangle DisplayRectangle;
    public Color Color;
    public TimeSpan CountDown;
    public float Transition;

    public GridCell()
    {
        Reset();
    }

    public bool Update(GameTime gameTime)
    {
        if (Color == Color.White)
        {
            Transition += (float)
     gameTime.ElapsedGameTime.TotalMilliseconds /
                                              100f;
            CountDown -=
                        gameTime.ElapsedGameTime;
            if (CountDown.TotalMilliseconds <= 0)
            {
                return true;
            }
        }
        return false;
    }

    public void Reset()
    {
        Color = Color.TransparentBlack;
        CountDown = TimeSpan.FromSeconds(5);
```

```
        Transition = 0f;
    }

    public void Show()
    {
        Color = Color.White;
        CountDown = TimeSpan.FromSeconds(5);
    }
}
```

Para reiniciar a célula ao seu estado default onde o boneco está escondido utilizamos o método Reset, o qual é chamado pelo utilizador ao clicar num boneco. Este método é utilizado para definir o temporizador para cinco segundos quando o boneco aparece no ecrã. O método Update é chamado em cada frame para atualizar o temporizador decrescente, ajudando a perceber se o utilizador não clicou no boneco dentro da janela de tempo dos cinco segundos. Uma vez definidas as células, vamos definir a grelha. Começamos por criar um novo campo List chamado grelha(grid):

```
List grid = new List();
```

Para calcular os rectângulos de exibição para cada célula, adicionamos o seguinte código ao método LoadContent:

```
var viewport = graphics.GraphicsDevice.Viewport;
var padding = (viewport.Width / 100);
var gridWidth =
             (viewport.Width - (padding * 5)) / 4;
var gridHeight = gridWidth;

for (int y = padding; y<gridHeight*5;
                          y+=gridHeight+padding) {
    for (int x = padding;
         x<viewport.Width-gridWidth;
         x+=gridWidth+padding) {
              grid.Add (new GridCell()
        {
          DisplayRectangle = new Rectangle(x, y,
                           gridWidth, gridHeight)
        });
    }
}
```

Se quisermos que o jogo pareça bem em todos os fatores de forma, devemos ter em conta o tamanho do ecrã em vez dos valores posicionais definidos no código. O GraphicsDevice.ViewPort proporciona-nos uma forma dinâmica de trabalhar com diferentes factores de forma. No código acima, adicionámos 10% da largura do ecrã como espaçamento entre as células e calculámos uma largura e altura para a grelha utilizando a mesma propriedade. Podemos fazer loop através das coordenadas (X,Y) para cada linha e coluna e calcular o rectângulo de exibição.

Substituímos o método Draw com o código para desenhar os nossos bonecos:

```
protected override void Draw(GameTime gameTime)
{
    graphics.GraphicsDevice.Clear
                              (Color.SaddleBrown);
    spriteBatch.Begin();
    foreach (var square in grid)
        spriteBatch.Draw(monkey,
   destinationRectangle: square.DisplayRectangle,
                                  color: Color.White);
    spriteBatch.End();
    base.Draw(gameTime);
}
```

Por fim, queremos que o nosso jogo corra apenas em Retrato, por isso adicionamos o seguinte código ao construtor:

```
graphics.SupportedOrientations =
                      DisplayOrientation.Portrait;
```

Corremos a aplicação e deveremos ver uma grelha cheia de bonecos!

Posto isto, o interface de utilizador para o nosso jogo construído com MonoGame está completo.

Adicionando Lógica de Jogo

Assim que tenhamos uma interface de utilizador construída para o nosso jogo, o próximo passo será adicionar a lógica necessária para que possa ser jogado.

Precisamos de actualizar a grelha de bonecos para permitir aos utilizadores interagirem com ela para jogar o

jogo. Primeiro definimos um enumarationGameState no Projeto Partilhado PapTap com os seguintes valores:

```
enum GameState
{
    Start,
    Playing,
    GameOver
}
```

Copiamos e colamos o seguintes campos de class-level na classe Game1:

```
// Define o estado inicial do jogo
GameState currentState = GameState.Start;
Random rnd = new Random();

// Texto a ser apresentado ao utilizador
string gameOverText = "Game Over";
string tapToStartText = "Toque para iniciar";
string scoreText = "Pontuação : {0}";

TimeSpan gameTimer = TimeSpan.FromMilliseconds(0);
TimeSpan increaseLevelTimer = TimeS-
pan.FromMilliseconds(0);
TimeSpan tapToRestartTimer = TimeSpan.FromSeconds
(2);

int cellsToChange = 0;
int maxCells = 1;
int maxCellsToChange = 14;
int score = 0;
```

Estes campos têm como utilidade seguir os vários estados em que o jogo pode estar. Ao continuar a jogar o PapTap, mais e mais células serão alteradas durante um dado nível, tornando o jogo mais difícil. Agora que o conjunto de configurações necessárias para seguir o estado do jogo está fora do caminho, é tempo de começar a implementar a nossa lógica de jogo!

Processar o Touch do Utilizador

A maior parte da mecânica de jogo está no tratamento do input do utilizador e por isso é importante selecionar um motor de jogo que consiga tratar todos os tipos de input. No Monogame podemos encontrar um grande conjunto de controlos de input, sendo um deles TouchPanel.

O método GetState do TouchPanel desenvolve uma colecção de localizações touch e o seu estado: Pressed, Moved e Released, Isto permite o acompanhamento dos toques do utilizador no ecrã. Se o utilizado tocar no ecrã, iremos percorrer todas as células da grelha e verificar se essa localização se interseta com o retângulo de exibição da célula. No caso de isso acontecer e o boneco estiver a ser exibido nessa altura, tocamos um som, reiniciamos a célula e incrementamos a pontuação do utilizador, afinal ele impediu que o boneco fugisse com a revista. Adicionamos o método PressTouches abaixo à classe Game1:

```
void ProcessTouches(TouchCollection touchState)
{
    foreach (var touch in touchState)
    {
        if (touch.State != TouchLocationSta-
te.Released)
            continue;
        for (int i = 0; i < grid.Count; i++)
        {
            if (grid[i].DisplayRectangle.Contains
(touch.Position) && grid[i].Color == Color.White)
            {
                hit.Play();
                grid[i].Reset();
                score += 1;
            }
        }
    }
}
```

Verificar se o Jogo acabou

No PapTap os bonecos aparecem por cinco segundos de cada vez e caso não sejam clicados durante esse tempo o jogo acaba. Para verificar se o jogo realmente termina, podemos percorrer todos os items na grelha dos bonecos e chamar o método Update que devolve true no caso de um boneco estar a ser mostrado por cinco segundos. Nesse caso significa que o utlizador não clicou no boneco na janela de tempo permitida, por isso devemos alterar o GameStato para GameOver e começar o tamToRestartTimer, o que impede o jogo de reiniciar imediatamente(e o jogador não ver a sua pontuação) com um click aleatório após o jogo ter terminado. Adicionamos assim o método CheckForGameOver à classe Game1:

```
void CheckForGameOver(GameTime gameTime)
{
    for (int i = 0; i < grid.Count; i++)
    {
        if (grid[i].Update(gameTime))
        {
            currentState = GameState.GameOver;
            tapToRestartTimer = TimeS-
pan.FromSeconds(2);
            break;
        }
    }
}
```

Calcular os Bonecos a mostrar por nível

A maioria dos jogos, o PapTap aumenta em dificuldade conforme o jogo continua. Cada nível irá mostrar mais e mais bonecos, portanto precisamos de calcular exactamente quantas células precisamos de exibir. O gameTimer que criámos anteriormente é utilizado para acompanhar quanto tempo decorreu desde que este nível começou. Podemos incrementar o temporizador acedendo a gameTime.ElapseGameTime que tem a quantidade de tempo desde que o último Update foi chamado. Uma vez que este temporizador passa os dois segundos, redefinimo-lo para zero e depois calculamos o número de células a alterar até um número máximo. Adicionamos o método CalculateCellsToChange á classe Game1:

```
void CalculateCellsToChange(GameTime gameTime)
{
```

```
    gameTimer += gameTime.ElapsedGameTime;
    if (gameTimer.TotalSeconds > 2)
    {
        gameTimer = TimeSpan.FromMilliseconds(0);
        cellsToChange = Math.Min(maxCells,
                                 maxCellsToChange);
    }
}
```

Calcular o nível de dificuldade

Como já vimos em CalculateCellsToChange, maxCells define o número máximo de macacos a mostrar num nível particular; por defeito, este valor está definido para 1. À medida que o jogo progride, queremos que o valor aumente com o tempo. Para fazer isto, vamos usar um temporizador para seguir quanto tempo passou no nível, e pós 10 segundos, avançar para outro nível e incrementar o número máximo de bonecos exibidos. Como resultado, o PapTap mostrará um boneco extra a cada 10 segundos. Adicionamos o método IncreaseLevel à classe Game1:

```
void IncreaseLevel(GameTime gameTime)
{
    increaseLevelTimer +=
                        gameTime.ElapsedGameTime;
    if (increaseLevelTimer.TotalSeconds > 10)
    {
        increaseLevelTimer =
                    TimeSpan.FromMilliseconds(0);
        maxCells++;
    }
}
```

Mostrar Bonecos

Finalmente, temos que tornar visíveis os bonecos que estão invisíveis por defeito. Para evitar que o PapTap seja previsível, podemos usar a classe Random para seleccionar um boneco aleatório na grelha. Se o boneco não está já a aparecer, podemos exibi-lo e decrementar o número de células necessário para mudar para esse nível.

```
void MakeMonkeysVisible()
{
    if (cellsToChange > 0)
    {
        var idx = rnd.Next(grid.Count);
        if (grid[idx].Color ==
                            Color.TransparentBlack)
        {
            grid[idx].Show();
            cellsToChange--;
        }
    }
}
```

Juntando as peças

Agora que temos todas as peças individuais completas, vamos juntá-las num método chamado PlayGame:

```
void PlayGame(GameTime gameTime,
                        TouchCollection touchState)
{
    ProcessTouches(touchState);
    CheckForGameOver(gameTime);
    CalculateCellsToChange(gameTime);
    MakeMonkeysVisible();
    IncreaseLevel(gameTime);
}
```

Geralmente a ordem não tem importância. Contudo, normalmente queremos verificar o input do utilizador primeiro para tornar o jogo mais responsivo e garantir que o boneco esteve no ecrã por cerca de cinco segundos completos.

Executando a Lógica de Jogo

Uma vez completa a maior parte da lógica de jogo para o PapTap, precisamos de uma forma de atualizar continuamente o jogo à medida que ele é executado. O Update é utilizado para atualizar qualquer lógica de jogo que tenhamos enquanto o jogo executa, portanto é aí que devemos colocar o método PlayGame. Vamos substituir o código atual do método Update pelo código abaixo:

```
protected override void Update(GameTime gameTime)
{
    #if !__IOS__ && !__TVOS__
    if (GamePad.GetState
(PlayerIndex.One).Buttons.Back == ButtonSta-
te.Pressed ||
        Keyboard.GetState().IsKeyDown
(Keys.Escape))
    {
        Exit();
    }
#endif

    var touchState = TouchPanel.GetState();
    switch (currentState)
    {
        case GameState.Start:
            if (touchState.Count > 0)
            {
                currentState = GameState.Playing;
            }
            break;
        case GameState.Playing:
            PlayGame(gameTime, touchState);
            break;
        case GameState.GameOver:
            tapToRestartTimer -=
                        gameTime.ElapsedGameTime;
            if (touchState.Count > 0 &&
         tapToRestartTimer.TotalMilliseconds < 0)
            {
                currentState = GameState.Start;
                score = 0;
                increaseLevelTimer =
                    TimeSpan.FromMilliseconds(0);
                gameTimer =
                    TimeSpan.FromMilliseconds(0);
                cellsToChange = 1;
                maxCells = 1;
                for (int i = 0; i < grid.Count;
                                                i++)
                {
                    grid[i].Reset();
                }
            }
            break;
    }
```

```
    base.Update(gameTime);
}
```

Primerio, chamamos o método TouchPanel.GetState para agarrar o actual estado do touch. Em seguida vamos passá-lo para o método PlayGame, que irá usá-lo para tratar o input do utilizador. A declaração de troca controla o estado do jogo. Como já vimos, vamos por default para GameState.Start. O caso GameState.Playing simplesmente chama a lógica de jogo que escrevemos anteriormente, enquanto o caso GameState.GameOver verifica para ver se o utilizador clicou para reiniciar o jogo e, se sim, reiniciar todos os nosso campos ao valor inicial, limpando a grelha, e transitar de volta para o estado GameState.Start.

Desenhando o PapTap

Se correrrmos o PapTap agora, não vemos muito mais do que a grelha de macacos que vimos quando começámos. Apesar de a lógica de jogo estar a correr nos bastidores, a interface do utilizador não está a ser actualizada após o jogo começar a correr. O lugar certo para toda esta lógica é o método Draw, de deve ser usado exclusivamente para desenhar quaisquer gráficos que precisamos de exibir.

```
protected override void Draw(GameTime gameTime)
{
    graphics.GraphicsDevice.Clear
                                (Color.SaddleBrown);
    var center = graphics.GraphicsDevice.Viewport.
                          Bounds.Center.ToVector2();
    var half = graphics.GraphicsDevice.
                                Viewport.Width / 2;
    var aspect = (float)logo.Height / logo.Width;

    var rect = new Rectangle((int)center.X -
        (half / 2), 0, half, (int)(half * aspect));

    spriteBatch.Begin();

    spriteBatch.Draw(background,
     destinationRectangle: graphics.GraphicsDevice.
              Viewport.Bounds, color: Color.White);

    spriteBatch.Draw(logo, destinationRectangle:
                          rect, color: Color.White);
    foreach (var square in grid)
    {
        spriteBatch.Draw(monkey,
     destinationRectangle: square.DisplayRectangle,
          color: Color.Lerp(Color.TransparentBlack,
                 square.Color, square.Transition));
    }

    if (currentState == GameState.GameOver)
    {
        var v = new Vector2(font.MeasureString
                           (gameOverText).X / 2, 0);
        spriteBatch.DrawString(font, gameOverText,
                       center - v, Color.OrangeRed);

        var t = string.Format(scoreText, score);
   v = new Vector2(font.MeasureString(t).X / 2, 0);
        spriteBatch.DrawString(font, t, center +
               new Vector2(-v.X, font.LineSpacing),
                                      Color.White);
    }
    if (currentState == GameState.Start)
    {
        var v = new Vector2(font.MeasureString
                          (tapToStartText).X / 2, 0);
        spriteBatch.DrawString(font,
        tapToStartText, center - v, Color.White);
    }

    spriteBatch.End();
    base.Draw(gameTime);
}
```

Algo importante a lembrar quando construímos aplicações e jogos é que devemos ter em conta vários tamanhos de ecrã e factores de forma dos equipamentos, desde telefones móveis a ecrãs de TV. Ao contrário do desenvolvimento de aplicações, não nos são fornecidos motores de layout dinâmicos como AutoLayout ou contentores de layout como LinearLayout do Android para exibir dinamicamente items de jogo. A propriedade ViewPort expõe propriedades para nos ajudar a descobrir quão grande uma célula deve ser no ecrã.

O texto deve estar centrado no ecrã para que possamos calcular o centro usando o seguinte:

```
var center = graphics.GraphicsDevice.Viewport.
                          Bounds.Center.ToVector2();
```

A propriedade ViewPort.Bounds é um tipo de rectângulo que contém uma propriedade Center que pode ser usada para desenhar texto no centro do ecrã:

```
var v = new Vector2(font.MeasureString
                          (tapToStartText).X / 2, 0);
spriteBatch.DrawString (font, tapToStartText,
                           center - v, Color.White);
```

Podemos utilizar SpriteFont.MeasureString para calcular o tamanho do texto, o que devolve um vetor com a largura (X) e altura (y). Podemos depois tirar esse valor do centro, para que quando desenharmos a string ela termine no lugar errado. SpriteFont também expõe uma propriedade útil chamada LineSpacing, que podemos utilizar para garantir que quando desenhamos o texto verticalmente, este está devidamente espaçado; Utilizamos isto para exibir a pontuação do utilizador abaixo do texto “Game Over”.

Vamos agora correr o jogo e deveremos ter um jogo PapTap completamente funcional para Android. Se correrrmos o código, deveremos ser capazes de jogar o jogo até ao fim:

O MEU PRIMEIRO JOGO EM MONOGAME

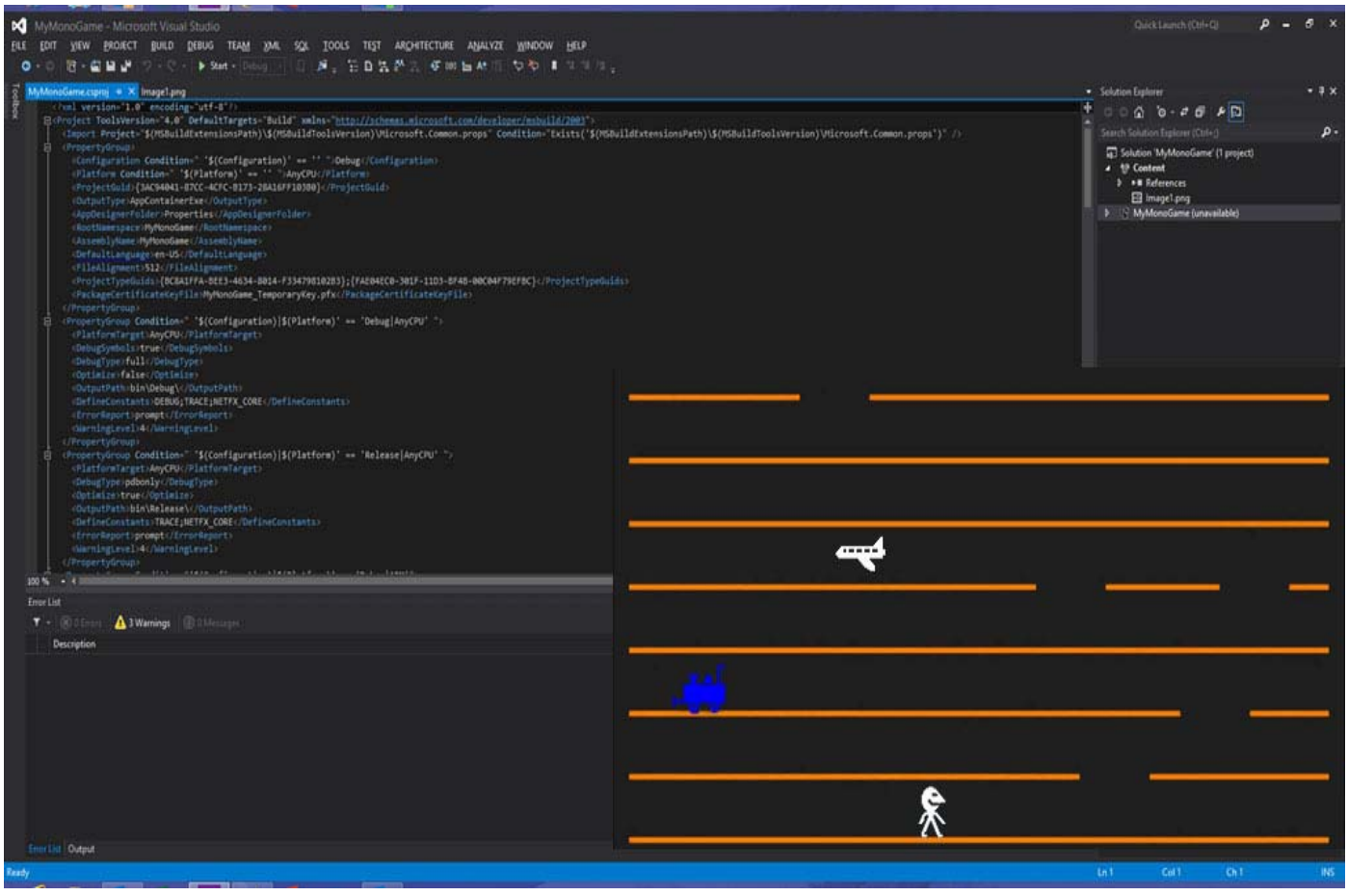
> *Muitos programadores chegaram ao mundo da programação através do fascínio do desenvolvimento de Jogos.*

Terminando

Acabámos de construir o nosso primeiro jogo para Android utilizando MonoGame! Algumas partes podem ser um pouco traiçoeiras, mas uma vez que compreendamos as peças maiores em jogo, tudo começa a fazer sentido. Se quiséssemos poderíamos adicionar uma versão iOS do Pap-Tap simplesmente adicionando um projecto novo à solução PapTap que referencia o Projeto Partilhado que criámos e apagar a classe Game1 que está autocriada.

Boom – num piscar de olhos, adicionámos suporte iOS ao nosso jogo numa questão de segundos!

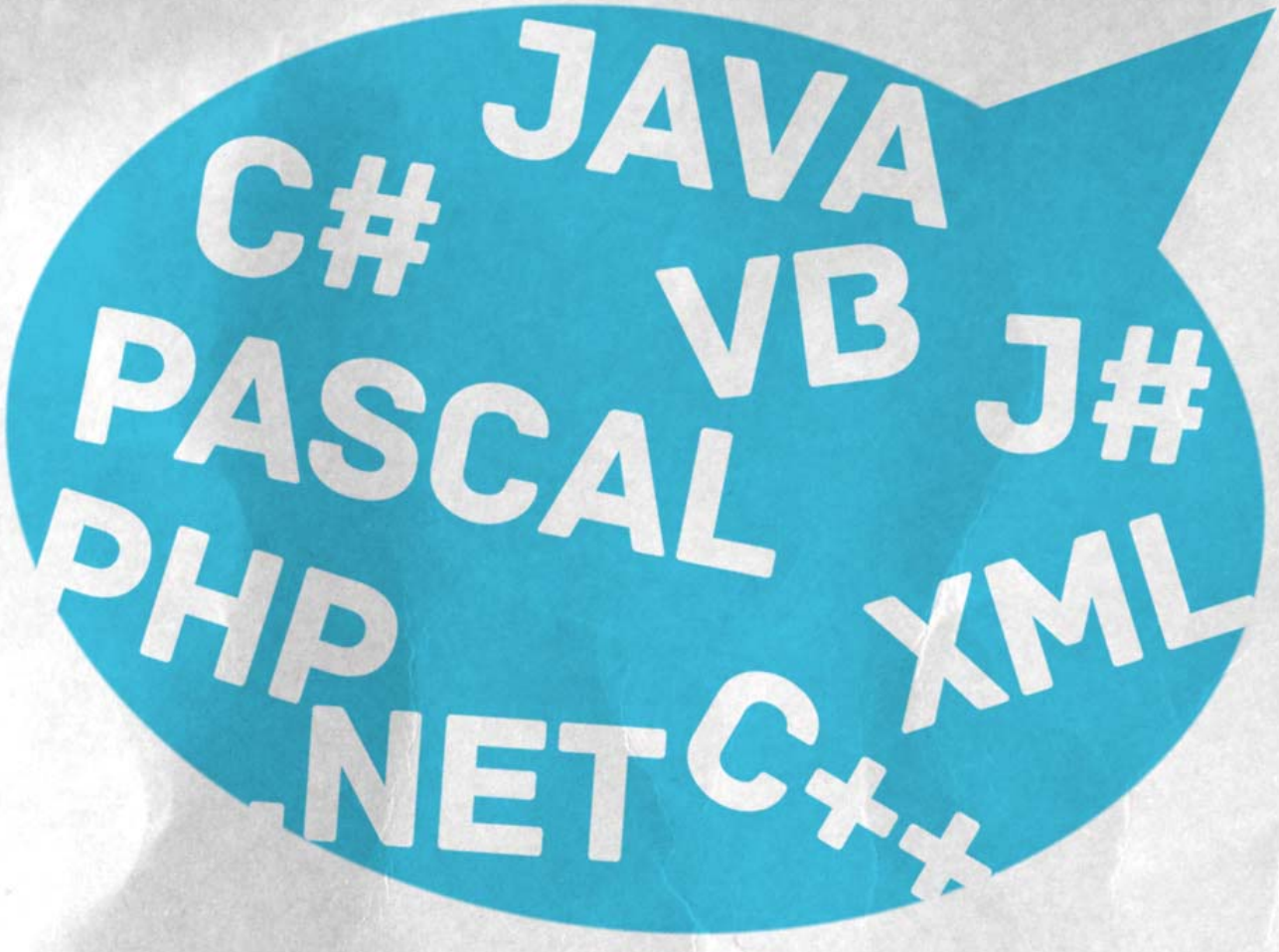
Nestas breves séries, vimos como criar um interface de utilizador para o nosso jogo, bem como adicionar lógica para criar um jogo estilo Whack-a-Mole chamado PapTap.



AUTOR

Escrito por António C. Santos

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



ENTÃO, SÓ FALAS EM CÓDIGO?

TEMOS O REMÉDIO CERTO PARA TI!

portugal-a-programar.pt

A MAIOR COMUNIDADE PORTUGUESA DE PROGRAMAÇÃO, APARECE!

ELECTRÓNICA

Introdução ao Arduino

INTRODUÇÃO AO ARDUINO

Está na moda o conceito *Internet of Things*, que se refere à capacidade de interagir com dispositivos físicos, obtendo informação/métricas (e.g., temperatura, humidade, etc.) e enviando comandos/acções (e.g., abrir porta, ligar ar condicionado, etc.).

O conceito, que não é novo, implica colaboração entre profissionais de electrónica, programadores e até DBAs. Este workshop é adequado para programadores e DBAs que têm poucos conhecimentos de electrónica, dando-lhes uma introdução à utilização de Arduino (actualmente referido como Genuino na Europa), uma das mais conhecidas plataformas de electrónica utilizada nesta área.

Após termos apresentado este workshop no **Genuino Day Lisbon 2016** (em 2016-04-02, na Microsoft Lisbon Experience), e também no IoT Summit 2016 (em 2016-05-05, também na Microsoft Lisbon Experience), decidimos publicar o guião utilizado. Queremos também deixar um agradecimento especial ao João Antunes (do CERN) e ao António Lourenço, pelo apoio pedagógico durante os workshops.

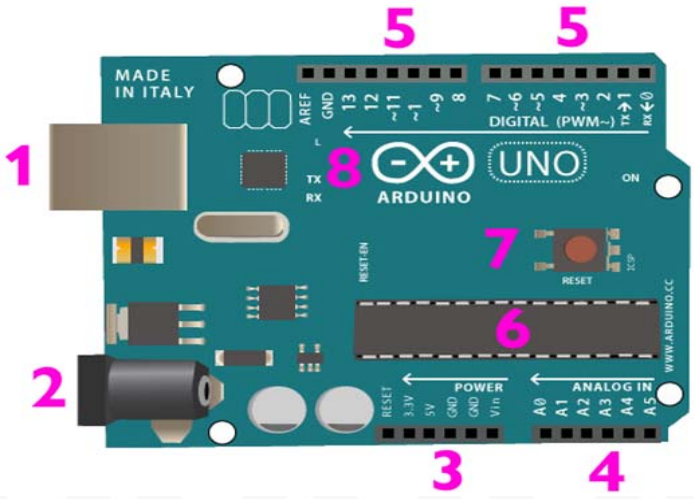
Conhecer o hardware

Existem vários modelos de Arduino/Genuino. Aqui vamos apenas trabalhar com os modelo Uno e Mega.

As maiores diferenças entre eles são a velocidade, a quantidade de memória e o número e tipo de pinos. Alguns são conectados via USB por ligação USB do tipo B, e outros por uma ligação Micro-USB do tipo B. As posições do botão de Reset e dos LEDs também variam conforme o modelo e versão.

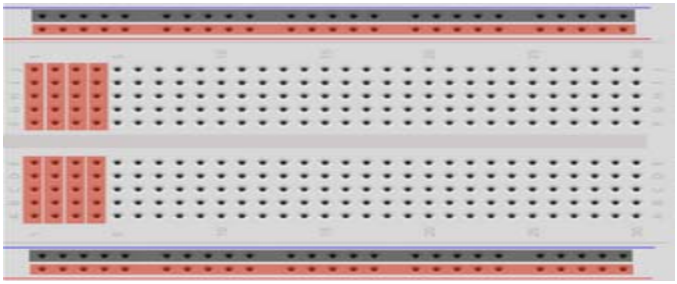
Para efeitos deste workshop, utilizaremos pinos e funcionalidades que são comuns a todos.

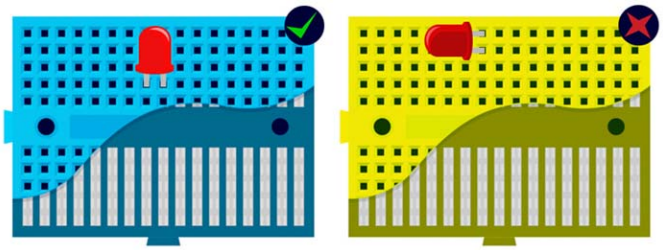
Arduino Uno:



1. USB para alimentação e comunicação;
2. Alimentação alternativa (de 9V a 12V, pino central é **+**);
3. Pinos de alimentação (saídas e entradas);
4. Pinos de entrada analógica (podem ser usados como digitais);
5. Pinos de entrada/saída digital (**~** significa saída digital PWM)
6. Processador;
7. Botão de Reset;
8. LEDs: **RX** e **TX** indicam comunicação série (porta USB), **L** apresenta o valor do pino 13;

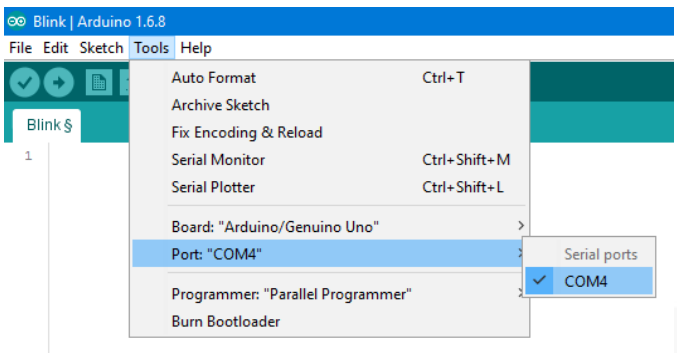
Breadboards:





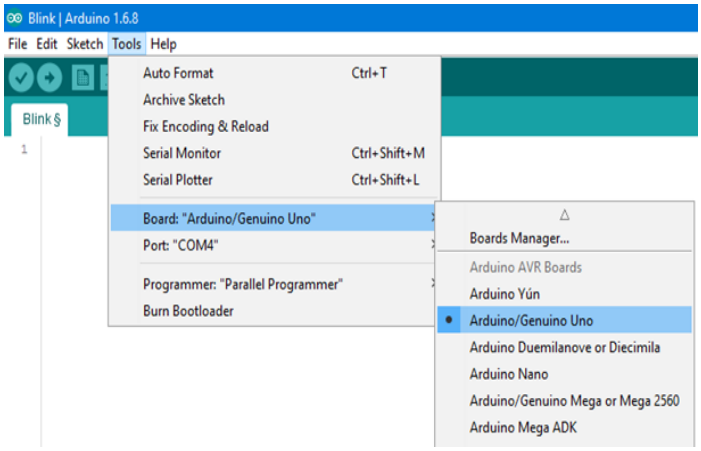
Hello World com o LED integrado

Após instalar o IDE do Arduino (https://Arduino.CC/en/Main/Software) e conectá-lo através do cabo USB, é necessário configurá-lo. Começamos por escolher a porta série onde está ligado (normalmente só haverá uma):

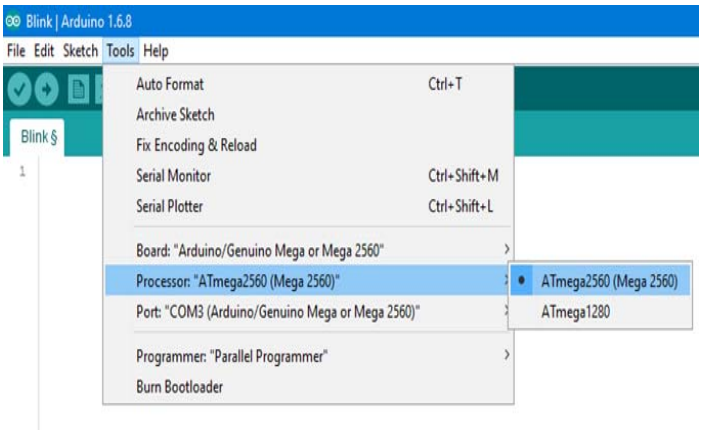


INTRODUÇÃO AO ARDUINO

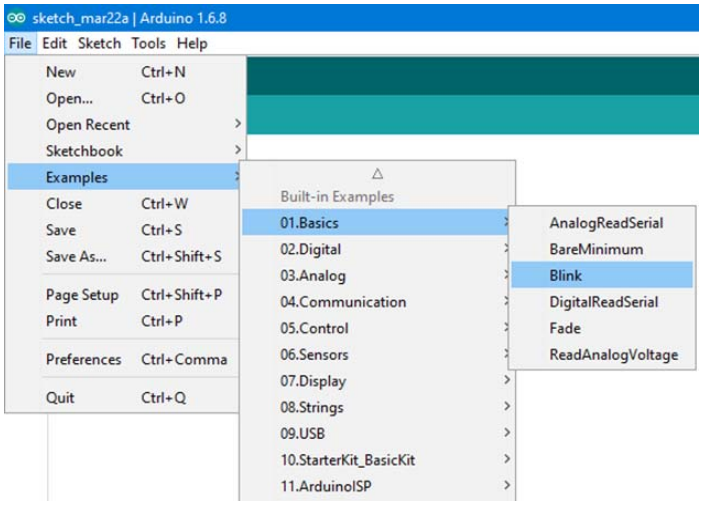
De seguida, é necessário escolher o modelo do Arduino:



Em alguns casos (como o Arduino Mega) é também necessário escolher a versão:



O IDE do Arduino já vem com alguns exemplos. Vamos começar por utilizar o exemplo **Blink**, para piscar o LED integrado na board (**L**), que está ligado ao pino 13.



O exemplo mostra duas funções:

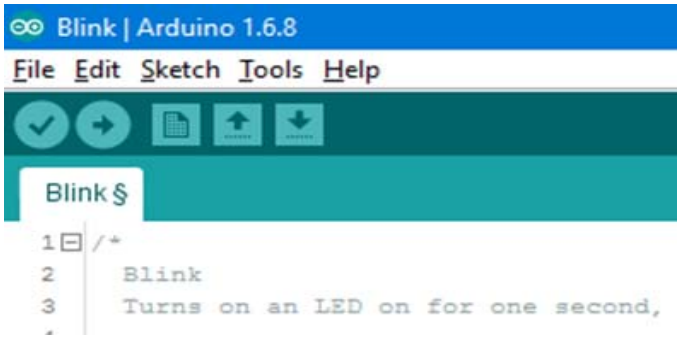
- setup()

  Nesta função inicializamos o hardware que estamos a utilizar. Esta função corre uma única vez quando o Arduino é alimentado.

- loop()

  Nesta função colocamos o programa que realiza a função pretendida. Chegando ao fim desta função, é novamente chamada (repete-se indefinidamente).

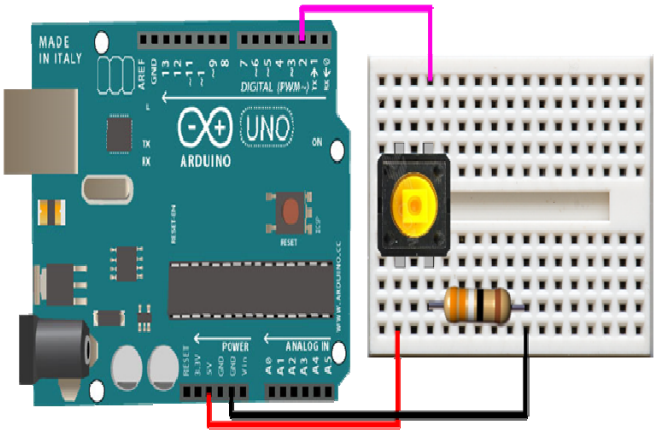
Após criar o programa, devemos validá-lo. Para isso, utilizamos o primeiro botão da barra de ferramentas (√).



Para enviar o programa para o Arduino, utilizamos o segundo botão (→). Durante o upload, veremos várias mensagens na parte inferior do IDE, assim como os LEDs **RX** e **TX** da placa a piscar. Uma vez concluído o upload, o programa inicia-se automaticamente.

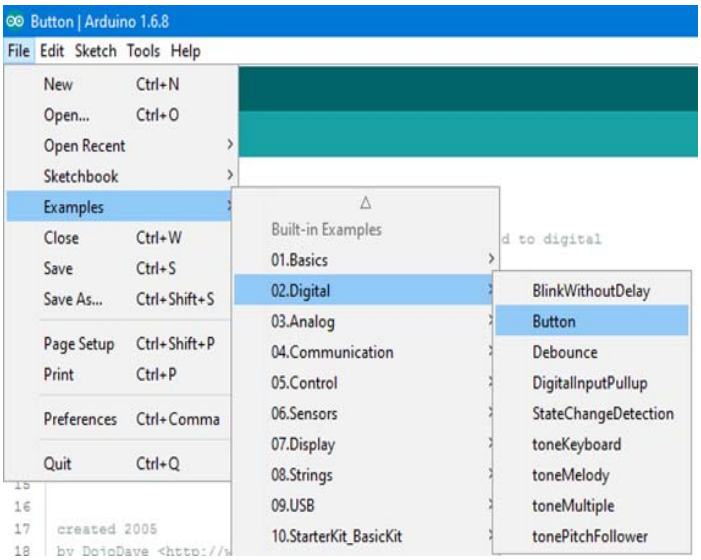
Neste caso, piscará o LED **L** integrado na placa. Poderá ajustar o delay() para alterar a duração de cada estado (ligado ou desligado) do LED. O delay() utiliza milissegundos.

Começamos por implementar o seguinte esquema, com um botão e uma resistência elevada (e.g. 10 kΩ):



Este esquema, que enquanto o botão não é carregado está a enviar 0V para a entrada, chama-se "pull-down". Existe também o "pull-up", mas aí por omissão a entrada recebe +5V (ou +3,3V, conforme o tipo de placa).

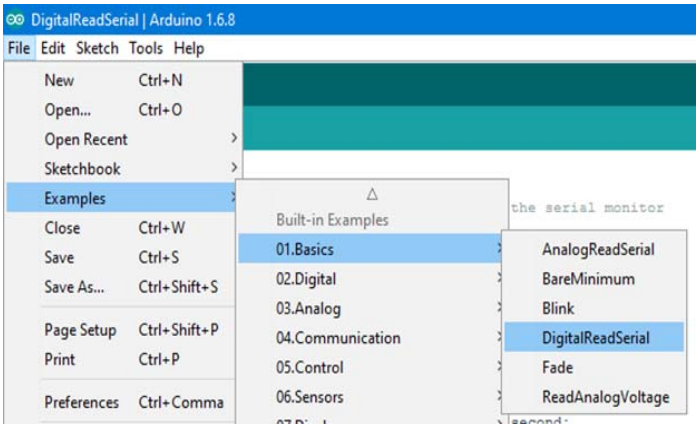
De seguida, carregamos o exemplo **Button**:



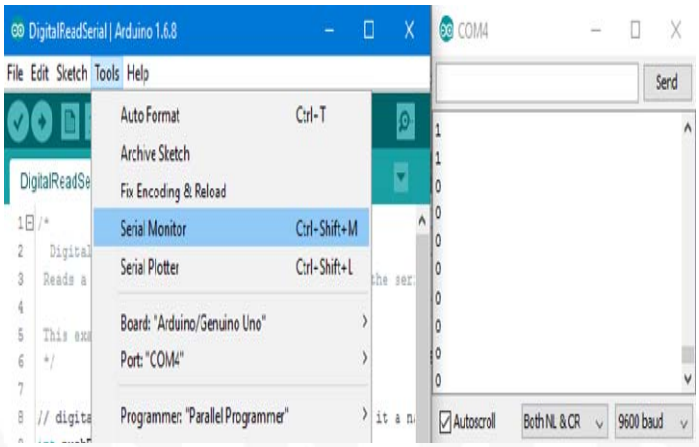
Após o upload, o botão funcionará para activar o LED.

Utilizando o esquema do laboratório anterior, vamos enviar o estado do botão pela ligação USB/série para o computador.

Vamos abrir o exemplo **DigitalReadSerial:**



Após enviar o programa para o Arduino, vamos ver o que é enviado para a porta USB/série, usando o **Serial Monitor:**

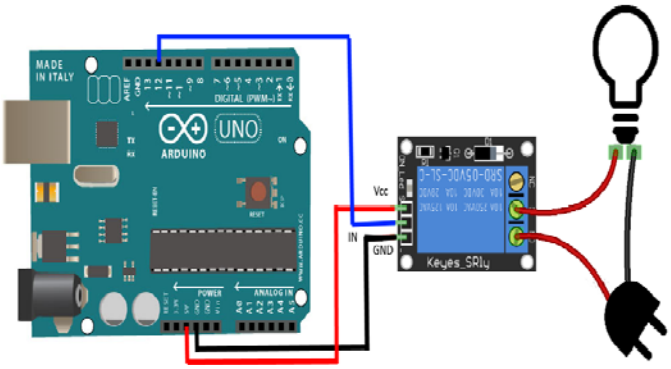


Nota: Antes de ligar o **Serial Monitor**, em algunso buffer da saída série enche rapidamente, e aí LED começa a piscar aleatoriamente e o programa fica suspenso na linha de Serial.println().

Podemos também enviar texto personalizado de debug. Por exemplo, acrescentamos a linha (respeitar sempre maiúsculas/minúsculas):

```
[...]
Serial.println(buttonState);

if (buttonState) Serial.println("LIGADO"); else
Serial.println("DESLIGADO");

delay(1); // delay in between reads for stability
[...]
```

Aproveitando o esquema dos laboratórios anteriores, vamos acrescentar um relé (que age como um interruptor que pode controlar equipamentos externos):



Acrescentamos ao final da função setup():

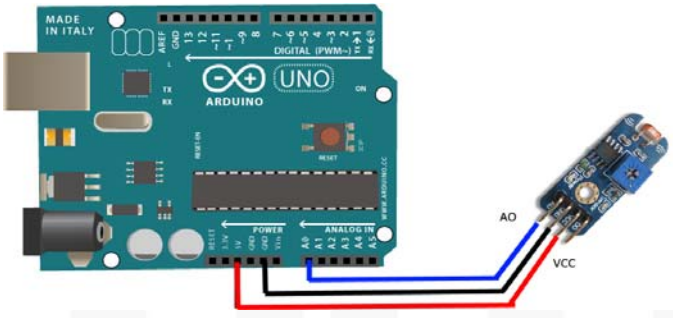
```
pinMode(13, OUTPUT);
pinMode(12, OUTPUT);
```

Acrescentamos ao final da função loop():

```
digitalWrite(13, buttonState);
digitalWrite(12, !buttonState);
```

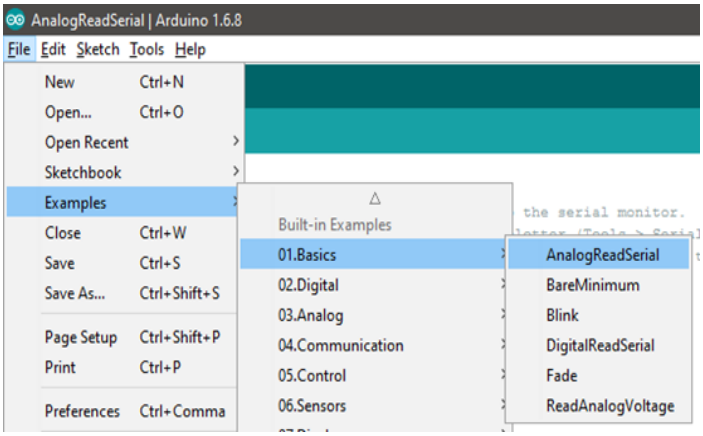
Porque é que o valor do pino 12 (relé) está invertido? Os relés que vamos utilizar ficam activos se no seu pino de sinal receberem 0V, e ficam inactivos se receberem 5V (ou não forem ligados). Nem todos os relés são assim, convém testar antes de utilizar.

Utilizando o esquema dos laboratórios anteriores, vamos acrescentar um sensor de temperatura ou luminosidade analógico ao pino A0:



INTRODUÇÃO AO ARDUINO

De seguida, carregamos o exemplo **AnalogReadSerial:**



Abrimos novamente o **Serial Monitor** para ver os valores do sensor. A leitura de pinos analógicos (A0, A1, etc.) retorna valores entre 0 e 1023 (ou seja, linearmente entre 0V e 5V). No entanto, é provável que o sensor escolhido só apresente valores em parte desta gama. Registe os valores mínimos e máximos dos sensor para utilizar no laboratório seguinte.

Utilizando o esquema dos laboratórios anteriores, vamos definir regras para activar o relé. Por exemplo, se usarmos um sensor de luminosidade, quando deixarmos de ter luminosidade, queremos activar o relé para ligar uma lâmpada.

Acrescentamos ao final da função setup():

```
pinMode(13, OUTPUT);
pinMode(12, OUTPUT);
pinMode(2, INPUT);
```

Acrescentamos ao final da função loop():

```
if (sensorValue > 500 || digitalRead(2))
{
      // Rele ligado
      digitalWrite(13, HIGH);
      digitalWrite(12, LOW);
}
else
{
      // Rele desligado
      digitalWrite(13, LOW);
      digitalWrite(12, HIGH);
}
```

O valor exacto (aqui 500) e o operador (">" ou "<") podem variar conforme o sensor. Utilize os valores obtidos no laboratório anterior para calibrar.

AUTOR

Escrito por Adrian Pearce

Licenciado e Mestre em Engenharia Electrotécnica, responsável por vários projectos de investigação e desenvolvimento financiados pela União Europeia, nas áreas de electrónica analógica e digital, engenharia de sistema e de produto, gestão e execução financeira.

Contacto: aenpearce@gmail.com

AUTOR

Escrito por André Melancia

Microsoft Certified Trainer (MCT), incidindo maioritamente em SQL Server, assim como Programador/DBA há 17 anos, desenvolvendo sistemas de informação e multimédia. Participa activamente em comunidades e eventos: IoT Portugal, NetPonto, SQLPort/SQLSaturdays, PTXug (Xamarin), PHPLx, ISOC.PT, etc. É fundador das comunidades IT Pro Portugal, IPv6 Portugal and DNSSEC Portugal.

Vai a http://Andy.PT e ficas a saber o mesmo que a NSA...

COLUNAS

Kernel Panic - Cross-Platform - “O silêncio e os interrupt’s”

"O silêncio e os interrupt's"

Ainda que possa parecer o título de um "filme de terceira categoria", qualquer semelhança é apenas mera coincidência fruto de um qualquer infortúnio das palavras! Passando as brincadeiras, e mudando para o verdadeiro assunto do artigo, todos ouvimos falar de *interrupt's* (sinal emitido pelo hardware ou software enviado ao processador, indicando que um evento necessita de atenção imediata), para os mais "vintage" da tecnologia que passaram pelos "tormentos" de configurar os *interrupts* nas bios cada vez que se acrescentava uma placa num pc, o conceito será certamente mais familiar, mas não se trata de *interrupts* de hardware ou software que escrevo! Trata-se antes das "interrupções" no trabalho de um programador e na relação das interrupções com a produtividade.

Todos nós certamente já passamos por situações em que as tarefas que estamos a realizar necessitam de atenção focada e exclusiva, sem distracções ou interrupções, sem ruídos nem perturbações da concentração, para podermos manter o ritmo e a concentração no que estamos a fazer. Ao contrário do ficcionado tantas vezes em televisão ou cinema, programar não é digitar código freneticamente, é desenvolver raciocínios que permitam digitar código eficiente, seguro, estável, e não digitar a velocidades estonteantes, como quem escreve um qualquer texto numa rede social, enquanto toma café, escuta música e fala ao telefone.

Um programador, quando está a trabalhar, numa grande parte das vezes está num estado de uma quasi-meditação, numa espécie de "*bolha protectora*", onde a "velocidade do tempo é definida por ele", onde o próprio som da musica que possa estar a ouvir na realidade acaba sendo uma espécie de "silêncio", na medida em que reflecte uma ausência de sons que possam retirar a sua atenção do seu raciocínio e da tarefa em curso.

Vejo vezes por demais, programadores a quem é pedido que não usem head-fones, durante as horas de trabalho, a quem é exigido que respondam a todo o tipo de solicitações enquanto realizam o seu trabalho, interrompendo-o de forma quase constante, quer seja para responder a um simples e-mail, quer seja para algo tão questionavelmente desnecessário como responder verbalmente a uma qualquer questão à qual já respondeu seja por e-mail, seja por qualquer outra via, mas que a pessoa inquiridora, não quer despender tempo, a ler a resposta, optando por um "atalho", sem se preocupar com as consequências que esse "atalho" possa ter para quem está efectivamente concentrado a realizar uma tarefa complexa. Para quem essa interrupção poderá significar horas de esforço para retomar o trabalho no ponto em que estava quando foi interrompido.

Honestamente não entendo esta "*dislexia*" de atitude, numa espécie de "contra-senso" de pessoas que apesar de muitas vezes serem altamente instruídas, não parecem entender ou ter conhecimento de nenhum dos diversos estudos que comprovam que uma tarefa interrompida, leva o dobro do tempo a ser terminada.

Segundo alguma documentação e estudos realizados um programador leva em média 15 minutos a entrar no seu estado de concentração para trabalhar. Ou seja cada interrupção custará pelo menos mais 15 minutos de tempo, na execução das tarefas! Considerando como média de interrupções num dia um valor de 4, pode-se considerar que 1h do dia é completamente desperdiçada! Essa mesma hora pode ser motivo de tensão e stress, para o programador, reduzindo ainda mais a produtividade.

Num estudo realizado por Chris Parnin, onde foram analisadas 10.000 sessões de programação de 86 programadores, e um questionário a 414 programadores, concluíram o seguinte:

- Um programador leva em média 10 a 15 minutos de tempo, a retomar uma tarefa, após ter sido interrompido
- Quando interrompido durante a edição de um método, apenas 10% das vezes o programador retomou o seu trabalho em menos de 1 minuto.
- Um programador é provável que apenas tenha duas horas por dia de trabalho sem interrupções
- A maioria dos programadores navega para diversos locais para reconstruir o contexto, antes de resumir a tarefa
- Muitas vezes forçam erros de compilação para relembrarem o raciocínio.

Ou seja, cada interrupção, causa não só stress, e perda de tempo, como se traduz em ineficiência, perda do "equilíbrio", e perda da qualidade do trabalho. Não obstante de tudo isto, numa esmagadora maioria das vezes, constato que quem trabalha com programadores, nem sempre tem a percepção do mal que fazem as interrupções e as distracções cada vez maiores.

Se a toda esta situação acrescentarmos a "aparente fixação patológica" pelos "ing's", que parece assolar a cultura portuguesa, (press**ing**, forc**ing**, etc…"**ing**"), denote-se bem a sátira, é quase impossível um programador não ter algumas vezes a necessidade soberana de férias, tão pouco de cometer erros de programação, ou de escrever código que nem sempre faz grande sentido! Tão pouco e ainda que não apenas português, de deixar comentários quase humorísticos no

código como um que se tornou conhecido a quando da divulgação do código fonte da framework .net "*//this is a hack for this release*". Existem diversas "piadas privadas" noutros exemplos de software, mas não valerá a pena citar mais, pois o objectivo é apenas ilustrar a situação.

É complicado explicar a um colega ou um gestor de projecto ou mesmo a qualquer pessoa que não seja programador, o quão complexo é estar no meio de um raciocínio complexo e ter de o suspender!

Não havendo uma formula "mágica", para explicar isso a uma pessoa, existem algumas formas mais ou menos lúdicas, de o ilustrar de forma perceptível! Uma delas que até parece trivial é pedir a quem interrompe, depois de devidamente atendido, para fazer um favor. No caso somar uma lista de números, tipo "0+987+765+543+321+012+234+456+678+9", sem recorrer a papel e caneta. Enquanto o interlocutor executa a operação, interrompe-lo em lapsos de tempo irregulares! Até se pode oferecer café, ou mesmo o almoço ao interlocutor, de forma a tentar tornar o desafio mais interessante, no entanto não pode escrever nada, nem usar papel ou qualquer instrumento para escrever. Mas em qualquer dos casos nunca deixar que o nosso interlocutor execute a tarefa, sem se distrair ou ser interrompido! Passados cerca de 2 minutos, que tal dizer que apenas restam 15 segundos para terminar a operação, enquanto e poucos segundos depois começar a "contagem decrescente". Como é obvio, o interlocutor não irá conseguir resolver a operação, acabando por desistir ou manifestar-se incomodado. Assim que pare e perceba que não existe forma de conseguir realizar a operação, será mais fácil explicar porque ser interrompido vezes sem conta, bem como sentir-se pressionado, apenas dificulta mais o trabalho e faz aumentar o lapso de tempo despendido a realizar o mesmo! Sim, é apenas uma brincadeira, mas como o leitor deve entender, é uma forma lúdica de ilustrar a dificuldade que um programador tem, quando é constantemente interrompido ou pressionado.

A titulo de conclusão deste artigo de opinião, recomendo vivamente a leitura do estudo de Chris Parnin, sobre este mesmo tema, pois creio ser deveras interessante.

AUTOR

Escrito por António C. Santos

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

Media Partners da Revista PROGRAMAR

Análises

Android Desenvolvimento de Aplicações com Android Studio

SQL Server 2014: Curso Completo

Android Desenvolvimento de Aplicações com Android Studio

Título: Android Desenvolvimento de Aplicações com Android Studio

Autores: Ricardo Queirós

Editora: FCA - Editora de Informática

Páginas: 304

ISBN: 978-972-722-819-5

Introdução

Hoje em dia os dispositivos inteligentes estão cada vez mais enraizados nas nossas vidas. A evolução da tecnologia é cada vez mais crescente e os utilizadores tornaram-se mais exigentes, contactando cada vez mais com estes dispositivos. O *Android* é um dos sistemas operativos mais utilizados nos dispositivos móveis e foi desenvolvido pela empresa Google.

Este livro que revemos nesta edição é constituído por 11 capítulos e explica como desenvolver aplicações *Android* usando o *Android Studio*, sendo o público-alvo os programadores de *software*.

Conteúdos

O livro começa por apresentar uma breve história e evolução do sistema operativo *Android* até aos dias de hoje. Por fim, enumera as características da plataforma *Android* desde a sua arquitectura, a Google Play e o conceito de multiplataforma também detalhado em capítulos posteriores, como por exemplo os relógios inteligentes, *Android Wear*, TV e automóveis.

O capítulo 2, o autor descreve o que é necessário para começar a construir uma aplicação *Android* desde a configuração do ambiente de desenvolvimento ao detalhe dos ficheiros constituintes de um projeto *Android* desde o ficheiro manifesto, localização dos recursos, referência aos mesmos e os ficheiros *Gradle*.

O capítulo 3 apresenta o ambiente de desenvolvimento de aplicações *Android*, neste caso o *Android Studio*. É efectuada uma visita guiada às diferentes janelas e ferramentas que o constituem. Gostaria de chamar à atenção do leitor a algumas como as Janelas de Ferramentas de edição (Código e GUI) e as barras presentes. Por fim são apresentados os emuladores no *Android Studio* nomeadamente o *Android Studio Emulator 2.0* e o *Genymotion*.

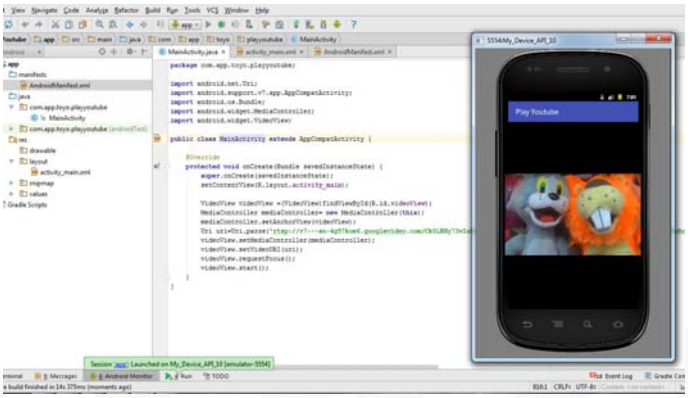
No capítulo 4 são apresentados as principais componentes de uma aplicação *Android* desde actividades (responsáveis pela interface com o utilizador), serviços (responsáveis pelas operações que executam em *background*), receptores de *broadcast* (respondem a eventos da aplicação ou sistema) e fornecedores de conteúdos (armazenam e fornecem dados para a aplicação). Por fim, é efectuada uma visita guiada ao ciclo de vida de uma aplicação *Android*.

No capítulo 5 dá-se maior enfase à criação de interfaces gráficas e os seus principais componentes desde *layouts* (**LinearLayout**, **TableLayout** entre outros), fragmentos (ciclo de vida e comunicação entre fragmentos) e *views*. Detalha-se o conceito de **View** e **ViewGroup** e os atributos constituintes ao alinhamento de *views*.

No capítulo 6 é abordado como aplicar efeitos de profundidade, tais como iluminação e sombras, animações, transições responsivas, navegação e respostas ao toque. Este tipo de padrão tenta criar experiências de utilização mais próximas aos objetos reais ao qual se dá o nome de *Material Design*.

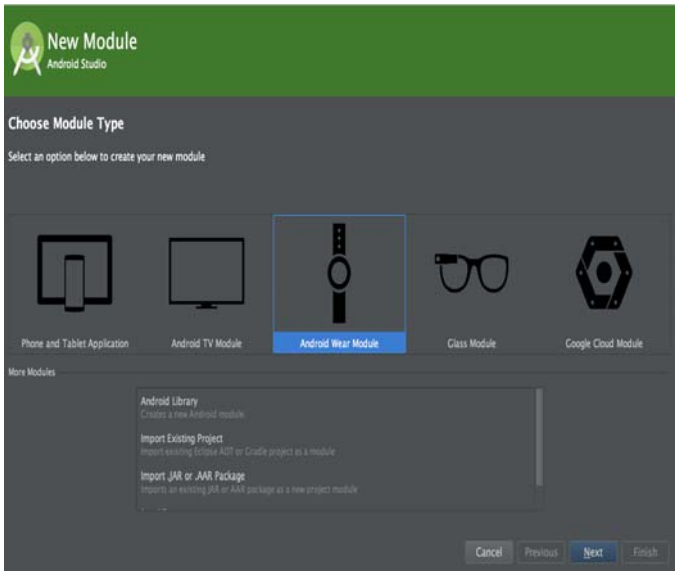
No capítulo 7 o autor destaca a gestão de dados. O *Android* oferece várias opções para a persistência de dados nomeadamente gravação de ficheiros em memória interna/externa, via preferências partilhadas e bases de dados SQLite. Neste capítulo é abordado como funciona cada uma destas técnicas.

O capítulo 8 descreve como aplicar boas práticas no desenvolvimento para reprodução e gravação de vídeos e áudio, captura de imagens e sua manipulação com base em novas classes multimédia disponibilizadas pelo SDK do *Android*. Destaco algumas dessas classes como: **VideoView**, **MediaController**, **MediaPlayer,** entre outras.



No capítulo 9, o tema abordado é o *Networking*. Como é habitual noutras aplicações, hoje em dia a comunicação com serviços externos é cada vez mais usual, e no *Android* não é excepção. O leitor irá aprender boas práticas na codificação de aplicações que operam na rede. É dado a conhecer os recursos disponíveis para gerir dados na web, desde **intents** (para visualização de páginas web) ou com base no uso do *widget* **WebView** e manipulação de HTML através da biblioteca **Jsoup**. Outros aspectos abordados são como gerir pedidos HTTP usando o **HTTPUrlConnection** e **OkHttp** e através de bibliotecas como o ***Volley*** e ***Picasso***. Por fim, será mostrado como integrar com *web services* destacando o uso da API do *Facebook* para *Android*.

O capítulo 10 é respeitante a mapas e localização. Neste capítulo é descrito o *Google Play Services* (internamente denominado por *Google Mobile Services* - **GMS**) como a biblioteca de serviços do *Android*. Esta biblioteca integra com várias API's destacando-se a API Maps (para gestão de mapas as aplicações baseado em Google Maps) e Location (para obtenção periódica da localização).



Por fim, no último capítulo o leitor adquire conhecimento no ***Android Wear***, uma versão de sistema operativo *Android* projectado para *smartwatches* e outros *wearables*. É abordado como desenvolver aplicações para o ***Android Wear*** com base em dois exemplos de aplicações: Uma aplicação que ensina a criar aplicações básicas para a plataforma *Wear* e outra que explica o processo de emparelhamento de um dispositivo *wearable* com um telefone, sendo abordado o sistema de notificações entre dispositivos.

> ***(...) os dispositivos inteligentes estão cada vez mais enraizados nas nossas vidas. A evolução da tecnologia é cada vez mais crescente e os utilizadores tornaram-se mais exigentes, contactando cada vez mais com estes dispositivos.***

Conclusão

Um bom livro da série de livros sobre desenvolvimento para a plataforma *Android*, do autor Ricardo Queirós com foco na plataforma *Android Studio*. Uma leitura muito agradável mas mais adequada para quem já tem conhecimentos sólidos de programação orientada a objectos.

Este livro torna-se também um bom complemento para quem já leu o livro "Android – Introdução ao Desenvolvimento de Aplicações", do mesmo autor, ou para quem deseja iniciar-se no desenvolvimento Android usando como IDE o Android Studio e já tem conhecimento base sólidos sobre esta tecnologia.

AUTOR

Escrito por Mónica Rodrigues

Licenciada em Engenharia Informática e de computadores pelo ISEL. Software engineer com vasta experiência em desenvolvimento web nas mais variadas tecnologias, desde HTML5, AngularJs, Asp.Net Web API, Asp.Net MVC, WCF, Entity Framework e tantas outras. Gosto igualmente de desenhar soluções de arquitectura aplicando padrões de desenho. Gosto de participar, entre outros, nos eventos da Microsoft, das comunidades Netponto e outras de forma a estar atenda às tecnologias emergentes. Linkedin: https://pt.linkedin.com/in/monicascrodrigues

SQL Server 2014: Curso Completo

Título: SQL Server 2014: Curso Completo

Autores: Alberto Magalhães

Editora: FCA - Editora de Informática

Páginas: 606

ISBN: 978-972-722-811-9

Tempo é dinheiro, portanto fica já aqui a minha opinião: Gostei do livro e recomendo.

Queres saber porquê? Continua a ler!

Quando me pediram para fazer a *review* /análise/crítica deste livro, decidi considerar os dois cenários típicos:

1. Alguém que nunca usou SQL Server (iniciante) e gostava de aprender; e

2. Alguém que já é profissionalmente experiente (médio/avançado) com SQL Server numa versão anterior e pretende aprender mais sobre a versão 2014.

Este livro responde às necessidades de ambos, focando a administração e manutenção.

É um livro de leitura fácil, onde não faltam *screenshots* e tabelas comparativas, explicado de forma simples. Para cada funcionalidade, há uma grande preocupação em detalhar cada uma das opções de configuração. Talvez do mais completo que vi, excedendo muito o detalhe dos livros de cursos oficiais da Microsoft (MOC). Por esse motivo, é também uma boa referência para utilizadores mais experientes.

Quem pretende obter certificação Microsoft em SQL Server, este livro é o ideal para o exame https://www.microsoft.com/en-us/learning/exam-70-462.aspx 70-462 (Administração) e um bom complemento para o respectivo curso https://www.microsoft.com/en-us/learning/course.aspx?cid=20462#overview MOC 20-462. Tem também informação introdutória para os cursos/exames seguintes, mas não foca a matéria do curso anterior, https://www.microsoft.com/en-us/learning/course.aspx?cid=20461#overview MOC 20-461 (Queries). Ou seja, o foco é maioritamente em tarefas de administração, não de desenvolvimento ou consulta de dados. Como tal, sinto a falta de informação sobre criação/alteração de tabelas, views, stored procedures, etc. Mas enfim, essa matéria daria outras 600 páginas.

Adicionalmente, este tipo de livros omite referências a cloud (algo imprescindível hoje), mas aqui não faltam (e bem) referências ao http://AZURE.COM Azure (o serviço cloud da Microsoft), nomeadamente a https://azure.microsoft.com/services/sql-database Azure SQL Database. Nota: indico aqui o nome actual do serviço, já que à data de lançamento deste livro tinha ainda a designação *SQL Azure*, alterada recentemente.

A primeira parte do livro está bem estruturada e corresponde ao que é habitual em cursos de formação em SQL Server para iniciantes:

- Componentes e funcionalidades do SQL Server (que são detalhados ao longo do livro);
- Comparativo de edições do SQL Server (Standard, Enterprise, etc.) e licenciamento;

Diferenças entre versões, incluindo obviamente as novidades do SQL Server 2014 (também detalhadas ao longo do livro); etc.

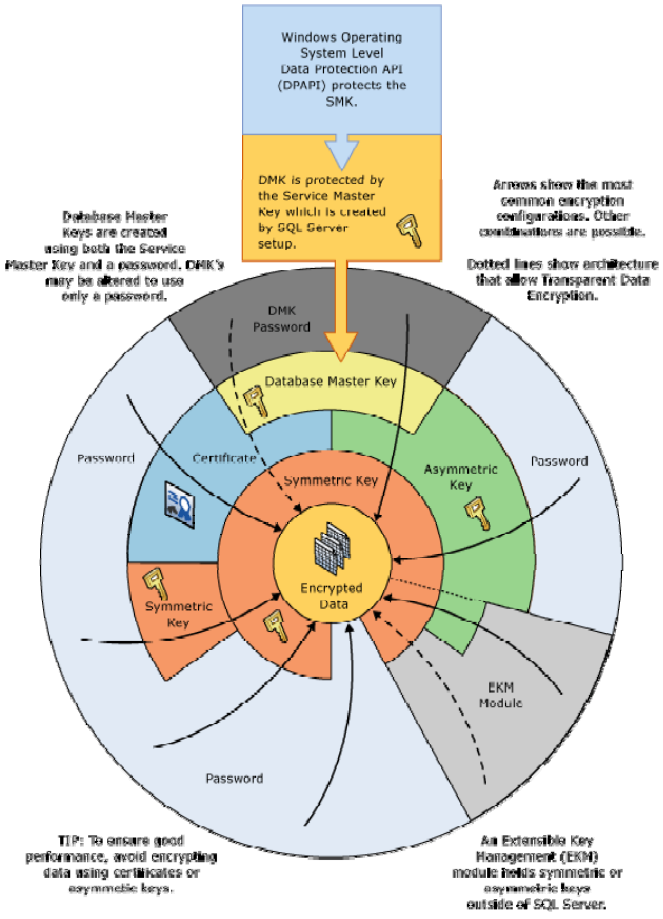
Segue-se o planeamento, instalação e rede:

- Considerações sobre armazenamento (boa explicação, embora falte a referência a discos SSD e as respectivas alterações conceptuais que trazem);
- Instalação (que inclui instalação em linha de comandos). Quem conhece SQL Server e a sua facilidade de instalação, compreende imediatamente porque este capítulo é dos mais pequenos;
- Considerações sobre rede (com um bom detalhe para programadores) e uma secção de troubleshooting que inclui o famoso problema de activar ligações remotas (que é muitas vezes omitido nestes livros).

É feita uma referência detalhada ao Management Studio (bastante acessível para iniciantes), com uma lista

exaustiva de opções de gestão do servidor (incluindo o Resource Governor), manutenção (SQL Server Agent, *maintenance plans*, comando DBCC, etc.), ferramentas de monitorização (Profiler, Extended Events, etc.). Fica a faltar uma referência geral às Dynamic Management Views (só referidas brevemente para replicação).

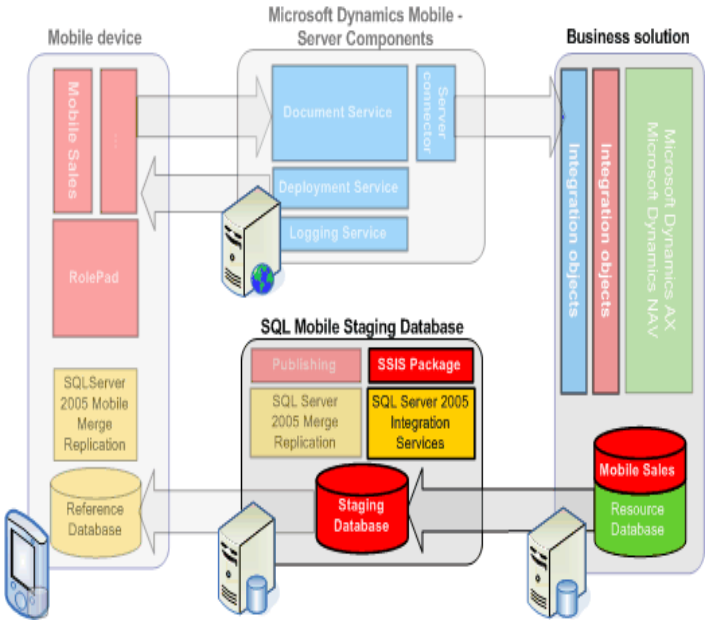
A Segurança não é esquecida, há um capítulo inteiro dedicado ao tema, detalhando inclusivamente opções de encriptação (entre elas *Transparent Data Encryption*, e mais tarde também encriptação de backups), assim como auditorias.



Seria grave não existir um capítulo dedicado a backups (e nunca esquecer, a restores!), e nisto o livro não falha. Tem bom detalhe e inclui opção de backup para Azure.

Entretanto, menos interessante para iniciantes mas muito útil para utilizadores avançados, são os capítulos dedicados a Alta Disponibilidade. Incluem *Always On Failover Cluster*, *Availability Groups*, *Mirroring* e *Replication*, incluindo detalhadamente a implementação de cada caso.

Finalmente, há uma introdução muito útil a *Integration Services*> (a forma recomendada de automatizar transferências e transformações de dados, ou ETL, entre servidores e/ou bases de dados), assim como ao *Reporting Services* (uma das formas de consumir dados relacionais ou de BI em formato de relatórios, páginas web, etc.). Sobre estes dois serviços há obviamente muito mais para dizer, e quem queira seguir para a área de Business Intelligence vai certamente aprofundar estes assuntos.



O que falta no livro? Para além das poucas coisas que mencionei em cima, faltam as melhorias da futura versão 2016, a ser lançada daqui a uns dias (que obviamente não é culpa do autor). Fica a sugestão de expandir este livro para a versão 2016: Descrição dos novos serviços como o R *Services*, novas opções de integração com o Azure como a *Stretch Database*, opções de segurança como *Always Encrypted*, *Row Level Security*, *Dynamic Data Masking*, etc.

Finalmente, termino como comecei: Gostei do livro e recomendo.

AUTOR

Escrito por André Melancia

Microsoft Certified Trainer (MCT), incidindo maioritamente em SQL Server, assim como Programador/DBA há 17 anos, desenvolvendo sistemas de informação e multimédia. Participa activamente em comunidades e eventos: IoT Portugal, NetPonto, SQLPort/SQLSaturdays, PTXug (Xamarin), PHPLx, ISOC.PT, etc. É fundador das comunidades IT Pro Portugal, IPv6 Portugal and DNSSEC Portugal.

Vai a http://Andy.PT e ficas a saber o mesmo que a NSA...

No Code

O Poder de uma SPA

A vida na Cloud

Indústria Alimentar Aliada às TI's

A Engenharia de software, a qualidade final do software e o papel do profissional de Desenvolvimento

Entrevista a: Vânia Gonçalves

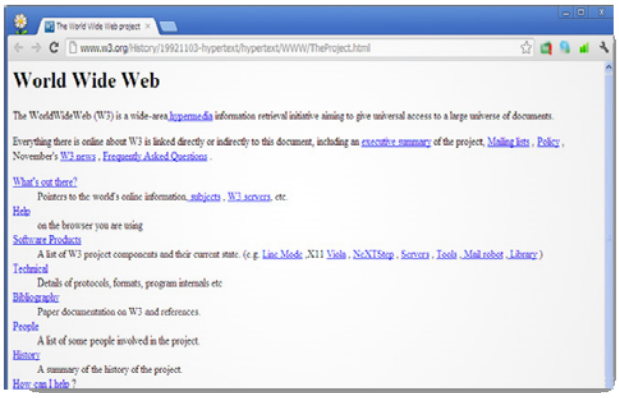
Projecto em Destaque na Comunidade p@p: Matraquilhos

O Poder de uma SPA

Introdução

Nesta edição, trago até vós um artigo sobre o conceito das SPA's. Para os leitores que não estão tão familiarizados com este conceito e quais as vantagens e desvantagens no uso de uma SPA, assim como as diferenças entre ter aplicações de múltiplas páginas e de página única, permitam-me que vós conduza pelas próximas páginas.

A primeira página web surgiu em 1994, uma simples página HTML sem grandes estilos ou animações associados.



Com o passar dos anos, as pessoas foram-se tornando cada vez mais exigentes, com um aumento significativo na evolução tecnológica.

A tecnologia evoluiu, os novos dispositivos (*smartphones*, *tablets*) começaram a aparecer, obrigando as aplicações a serem suportáveis, responsivas e visíveis em cada um desses dispositivos. Cada vez mais, os utilizadores pretendem aplicações mais rápidas, fluídas, com tempos de resposta rápidos, que sejam suportadas em todos os dispositivos, que corram numa única página como se de uma aplicação desktop se tratasse. Ora tudo isto exige aos programadores que repensem a forma como constroem a arquitetura da sua aplicação. É aqui que entra o conceito das Single Page Applications também conhecido pelo acrónimo de SPA, termo que usaremos durante o artigo.

SPA é o conceito usado para aplicações web que carregam uma página HTML simples e em seguida atualiza a página dinamicamente em vez de carregar novas páginas. Após o carregamento da página inicial a aplicação SPA comunica com o servidor por meio de solicitações AJAX (*Asynchronous JavaScript* and XML).

Uma vez que grande parte da aplicação se centra do lado do cliente, exigiu aos programadores um maior domínio da linguagem Javascript. Esta também tem vindo a evoluir exponencialmente ao longo dos anos.

Motivações para o uso de SPA's

Existem várias motivações no uso de uma aplicação como esta, sendo algumas delas as que enumero de seguida:

1. **Reach**
   As aplicações SPA podem ser acessíveis em diferentes plataformas e dispositivos desde que tenha um browser.

2. **Responsive**
   As aplicações necessitam de ser responsivas para serem facilmente visíveis nas diferentes resoluções de ecrãs.

3. **Round Trip**
   Grande redução do número de pedidos efetuados entre cliente e servidor. Uma vez que a aplicação se centra maioritariamente do lado do cliente, o programador só necessita de fazer os pedidos necessários para obter, gravar e listar os dados propriamente ditos, e não páginas HTML.

4. **Melhor experiência de utilização**
   A experiência de utilização numa aplicação deste tipo é mais imersiva.

5. **Separação de responsabilidades entre cliente e servidor**
   Numa SPA a divisão de responsabilidades entre aquilo que é cliente e o que é servidor torna-se evidente. Facilmente se torna possível ter pessoas dedicadas a trabalhar do lado do cliente, focadas no HTML5, CSS3 e Javascript e outras pessoas dedicadas ao lado do servidor, implementando uma Web Api, por exemplo, expondo os recursos necessários em resposta ao cliente.

6. **Offline Applications**
   O HTML5 permite mecanismos de Application Cache, o que significa que a aplicação pode ficar em cache e ser acessível sem ter acesso à internet. A vantagem deste mecanismo é o facto do utilizador conseguir aceder aos dados sem estar conectado à internet. Assim o carregamento dos dados torna-se mais rápido e reduz o carregamento ao servidor uma vez que este só é recorrido para efectuar uma actualização ou alteração. Uma outra forma pode passar por usar os

mecanismos de local storage e session storage fornecidos pelo HTML5.

Considerações

Apesar das inúmeras vantagens existentes na construção e uso de SPA's existem algumas considerações em ter em conta, sendo elas:

1. O primeiro carregamento da página pode ser lento
   A primeira vez que a aplicação é carregada poderá levar algum tempo uma vez que grande parte dos dados têm que ser carregados de inicio para que toda a interação possa ser efetuada maioritariamente do lado do cliente.
2. Deep-linking
   A navegação em aplicações como estas também pode tornar-se complicada, exigindo ao programador que implemente a funcionalidade de *back* por exemplo, algo que nos é diretamente fornecido pelo *browser*.
3. SEO
   Uma vez que não existem âncoras tende-se a que ferramentas de *Searching* como o Google não consigam encontrar facilmente este tipo de aplicações.

Agora que enumeramos algumas das desvantagens do uso de SPA's , passo a descrever as diferentes formas de funcionamento das aplicações de múltiplas páginas (aplicação web tradicional) e as SPA's.

Posteriormente mostramos uma junção destes dois tipos de aplicações que pode ser efectuado.

Comparação entre os diferentes tipos de aplicações

Aplicações web tradicionais

Estas aplicações são baseadas em navegação via browser não necessitando de instalação no cliente.

Uma aplicação web tradicional renderiza uma página web quando recebe um pedido **HTTP** vindo do browser. Após renderizadas são enviados de volta como uma resposta **HTTP** provocando uma actualização da página no browser, sendo esta substituída. A figura 1 mostra o cliclo de vida de uma aplicação web tradicional.

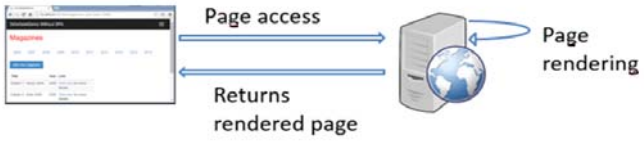


Figure 1 – Ciclo de vida de uma aplicação de múltiplas páginas

Como podemos constatar o peso sobre o desenvolvimento do lado do cliente é pequeno e o Javascript é usado para realizar alterações simples seja a nível da interface ou das animações.

Aplicações Nativas

As aplicações nativas são aplicações stand-alone que necessitam de ser instaladas em primeiro lugar. Sendo que estas só podem correr nos dispositivos para as quais foram criadas. Estas aplicações são desenhadas para tirarem maior proveito do sistema operativo e hardware para as quais foram concebidas, no entanto torna o seu desenvolvimento mais complexo. Não se esqueça que ao desenvolver nativamente para dispositivos tem que ter em conta as versões dos drivers e até mesmo do sistema operativo.

Single Page Application

Como mencionado no início do artigo, uma Single Page Application é uma aplicação web que corre numa única página HTML, não necessitando de ser refrescada no *browser* para mudar para uma outra página.

Um outro aspecto destas aplicações é o facto de não necessitar de ser instalada do lado do cliente uma vez que corre no *browser*, é cross-platform e não necessita de gerir sessão do lado do servidor.

Numa SPA são efectuados pedidos cujo retorno se trata de dados no formato JSON que serão posteriormente processados e manipulados do lado do cliente. A figura 2 mostra o ciclo de vida de uma SPA.

Figure 2 - Ciclo de vida de uma SPA

Tabela comparativa dos diferentes tipos de aplicações

Funcionalidades	Aplicação Web tradicional	Aplicação Nativa	SPA
Cross-plataform	☑	☒	☑
Client-state management	☒	☑	☑
No instalation required	☑	☒	☑

Junção entre Aplicações web tradicionais e SPA's

Um outro tipo de aplicação que se pode ter é a junção entre uma aplicação de múltiplas páginas com aquilo a que eu chamo de Mini-SPA.

Exemplo: Se estivermos a desenvolver uma aplicação de gestão de produtos e categorias do mesmo, podemos ter

diferentes páginas, uma correspondendo à página HTML dos produtos e outra à página HTML das categorias e dentro destas, uma SPA que gere a listagem, inserção, alteração e remoção (operações CRUD, por exemplo) de produtos e/ou categorias, respetivamente.

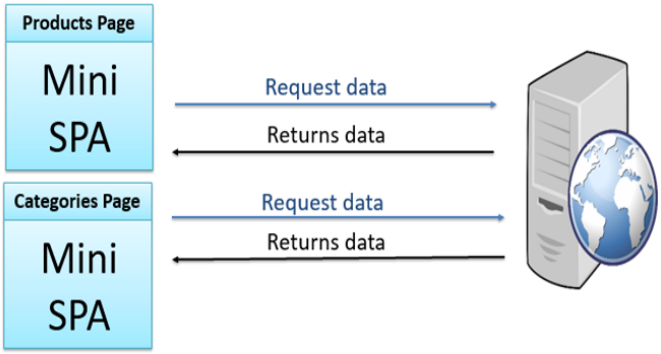
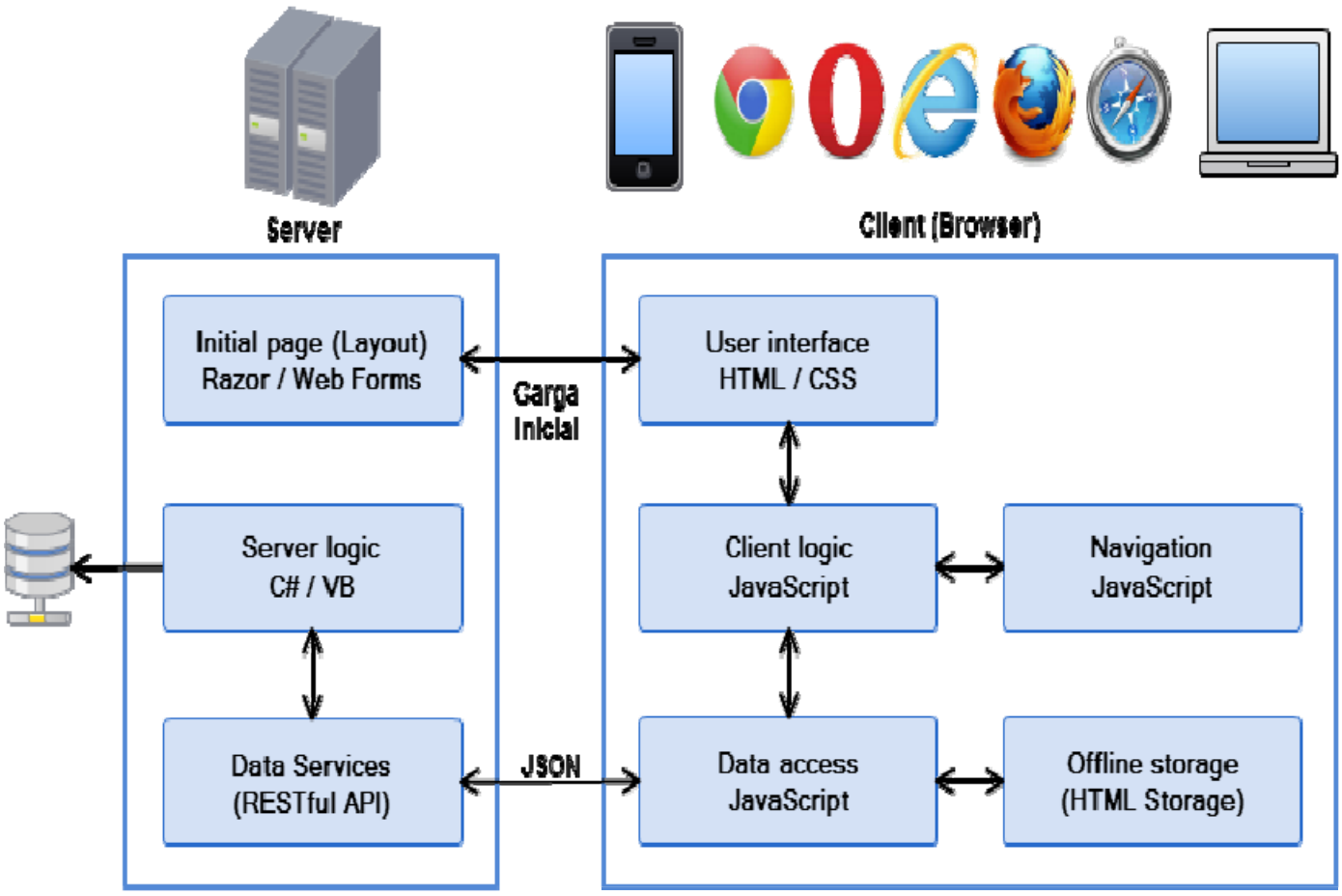


Figure 3 - Junção aplicações web tradicionais com SPA's

Conclusão

Para concluir deixo-vos algumas frameworks que existem actualmente e que nos auxiliam na construção de SPA's sendo elas o Angular, Ember e Aurelia. Existem também algumas bibliotecas como o Knockout, Backbone entre outras que nos auxiliam na construção das chamadas Mini SPA's que referi acima.

Para começarem a desenvolver uma SPA, pensem primeiramente na estrutura da vossa aplicação em termos de conteúdos, HTML, CSS e aconselho-vos piamente a escolher uma destas frameworks e a estudar o seu funcionamento de forma a entenderem se responde às vossas necessidades.



AUTOR

Escrito por Mónica Rodrigues

Licenciada em Engenharia Informática e de computadores pelo ISEL. Software engineer com vasta experiência em desenvolvimento web nas mais variadas tecnologias, desde HTML5, AngularJs, Asp.Net Web API, Asp.Net MVC, WCF, Entity Framework e tantas outras. Gosto igualmente de desenhar soluções de arquitectura aplicando padrões de desenho. Gosto de participar, entre outros, nos eventos da Microsoft, das comunidades Netponto e outras de forma a estar atenda às tecnologias emergentes. Linkedin: https://pt.linkedin.com/in/monicascrodrigues

A VIDA NA CLOUD

O que é a Cloud? Onde está? Todos já ouvimos falar dela e muitos utilizam-na sem sequer saberem. Em termos simplistas a Cloud consiste em armazenar e ter acesso a dados e aplicações na internet, em vez das mesmas estarem no nosso próprio computador. A Cloud, ou a nuvem, foi uma metáfora para a internet desde sempre, já que representa os diferentes cenários nos quais recursos computacionais são disponibilizados por uma rede. Mais do que disponibilizar recursos de hardware e/ou software, a Cloud permite disponibilizar uma gama de recursos virtuais ou físicos remotamente, por oposição ao acesso a esses recursos locais em discos rígidos, o que se denomina normalmente por armazenamento local.

Uma das principais vantagens da Cloud é a Escalabilidade, ou seja, a capacidade de uma aplicação em se adaptar a um novo contexto em que sejam necessários mais ou menos recursos. Imaginemos um portal web que a partir de certa altura devido ao seu crescimento precisa de mais memória ou CPUs para dar resposta aos pedidos, ou então a situação inversa em que se chega à conclusão de que não são necessários tantos recursos para a aplicação funcionar. Com a Cloud é possível aumentar e diminuir esses recursos, sendo que o utilizador apenas paga o que está a usar ao invés de uma situação em que tem de manter a infraestrutura física e virtual, independentemente se esta está ou não a ser utilizada.

Modelos de Cloud

- **Público**

Uma Cloud pública tem os seus serviços e infraestrutura alojados num fornecedor (cloud provider), é partilhado pelos seus clientes e acessível a estes através de redes públicas como a internet. Este modelo de Cloud oferece uma economia de escala e redundância, pecando, no entanto, pela vulnerabilidade no que diz respeito à segurança, principalmente devido à sua acessibilidade.

- **Privado**

Uma Cloud privada, por seu lado utiliza serviços e infra-estruturas alojadas em redes privadas acessíveis apenas a um cliente. A segurança e o controlo são assim os maiores benefícios deste modelo, ao passo que o aumento dos custos passa a ser a sua maior desvantagem.

- **Hibrido**

Tal como o nome sugere, o modelo hibrido combina as características dos modelos público e privado, permitindo a uma organização maximizar a sua eficiência utilizando a Cloud pública para operações menos sensíveis, enquanto as operações mais críticas são operadas em modo privado.

Tipos de serviço

- **Infrastructure as a service (IaaS)**

As IaaS referem-se à disponibilização de recursos computacionais (maioritariamente de hardware) através da rede. A sua oferta abrange entre outros recursos o espaço em disco, ligações de rede e endereços IPs. Os recursos encontram-se em vários data centers sendo que os clientes podem aceder a estes elementos virtuais e construir assim as suas próprias plataformas.

- **Platform as a Service (PaaS)**

As PaaS são uma extensão das IaaS e consistem em recursos computacionais que proporcionam aos programadores ambientes de desenvolvimento para que desenvolvam as suas aplicações na internet. Exemplos de recursos disponibilizados pelas PaaS são o Sistema Operativo, Sistema de Gestão de Base de Dados, Ferramentas de desenho e desenvolvimento, entre outros.

- **Software as a Service (SaaS)**

Os SaaS são aplicações disponibilizadas pela rede, ou seja, aplicações que o utilizador não precisa de instalar para aceder às mesmas. Exemplos de SaaS são o Gmail, Facebook, Twitter ou Office Online. Ao contrário do modelo tradicional de software em que se compra uma licença, se instala e se utiliza, o SaaS permite ao cliente pagar on demand, ou seja, pagar por mês em função da sua utilização.

Enquanto a diminuição dos custos de acesso à internet faz com que o acesso a serviços Cloud aumente, a privacidade e controlo de dados fazem com que utilizadores hesitem em usar a cloud. **Mas até que ponto?** Por um lado, temos cada vez mais serviços terminais em que TVs, smartphones, computadores, tablets, e outros dispositivos apenas têm capacidade de processamento local e armazenamento temporário de dados que são obtidos através da Cloud. Os utilizadores não hesitam em partilhar os seus conteúdos quer em informações no facebook ou conteúdos multimédia no youtube. Os jogos de vídeo também dão prova de que a comunidade online é por ventura a que mais anima a indústria. Quando há uns anos todos guardavam os seus emails localmente no Outlook, hoje o Gmail guarda a maioria dos e-mails da população mundial. Os grandes fornecedores de conteúdos (música, cinema, séries, etc) adoptam um paradigma de pagamento por subscrição através de serviços como o Netflix ou Spotify. Por outro lado, serviços como o

Cloud Drive da Amazon, o iCloud da Apple, Google Drive ou o Microsoft One Drive, alavancam ainda mais a utilização de serviços Cloud por parte dos utilizadores particulares.

E quanto aos utilizadores empresariais? Apesar do crescente aumento da oferta no que diz respeito às soluções empresariais de serviços na Cloud, onde guardam os seus dados e serviços as grandes empresas, assim como os Bancos, Seguradoras ou Governos? Num mundo em que a confidencialidade predomina, as grandes companhias optam por ter os seus próprios data centers de forma a controlarem os seus dados e serviços. Mas não estarão estas organizações num modelo de Cloud privada, visto que os seus utilizadores acedem a linhas seguras e intranets que circunscrevem Clouds privadas, ou mesmo híbridas? No caso dos bancos, por exemplo, existe o processamento interno e as aplicações externas de ebanking disponíveis para aplicações móveis e internet. Por outro lado as pequenas e médias empresas começam a optar por serviços na Cloud, dada a vasta gama de soluções existentes. Sem dúvida este é um espaço de futuro para as Telcos que apostam fortemente em aumentar os seus data centers o que lhes permite diversificar a sua oferta quer para utilizadores particulares como empresariais.

Numa era em que já não é possível imaginar o mundo sem a internet, **a Cloud não é uma possibilidade ou incerteza, mas sim um facto incontornável na tecnologia do século XXI**. Depois dos conteúdos físicos em que Vinis, Cds, DVDs e livros ocupavam prateleiras e estantes, passou-se para o paradigma de discos rígidos e NAS (Network Attached Storage). Hoje em dia é na Cloud que os utilizadores guardam os seus conteúdos. Já não nos preocupamos em ter aquele filme ou livro, pois a facilidade com que alugamos um filme diretamente na nossa TV, subscrevemos um serviço de música ou guardamos livros na Cloud fazem com que o mercado dos conteúdos online ou na Cloud seja o que mais cresce. Numa era em que as aplicações online são cada vez mais comuns, mesmo os clientes dos bancos optam por soluções como o eBanking às tradicionais agências. **A segurança é o desafio, mas a utilização é já uma certeza.** É impensável conceber um mundo fechado em silos em que a troca de informação e a disponibilização de dados e serviços remotos não existem. A Cloud está aqui, veio para ficar e é apenas uma questão de saber que forma vai assumir no futuro. Um futuro cada vez mais virtual e em que o online e o físico se misturam numa realidade em que os dados e serviços passam a ser ubíquos. Os providers de serviços e conteúdos são cada vez mais diversificados e a promessa de mais e mais gadgets terminais que possam ligar-se a esses serviços não pára de aumentar.

Quando a tecnologia já não é o desafio, passa-se a discussão para o nível ético da privacidade que mais cedo ou mais tarde terá de ser feita ao nível das sociedades. Estaremos dispostos a ter a nossa vida na Cloud? Incluindo dados de saúde, bancários, localização, rendimentos e despesas? Ou haverá uma fronteira a qual a Sociedade vai ter de impor à tecnologia para garantir a liberdade individual? O futuro está aqui e a Cloud é apenas a metáfora para o lugar onde os nossos bens e serviços habitam. Deixaram há muito as nossas casas e cidades, hoje em dia estão em todo o lado. Até onde vamos virtualizar os nossos dados é a questão que nos próximos anos vamos ter de dar resposta.

AUTOR

Escrito por Pedro Pico

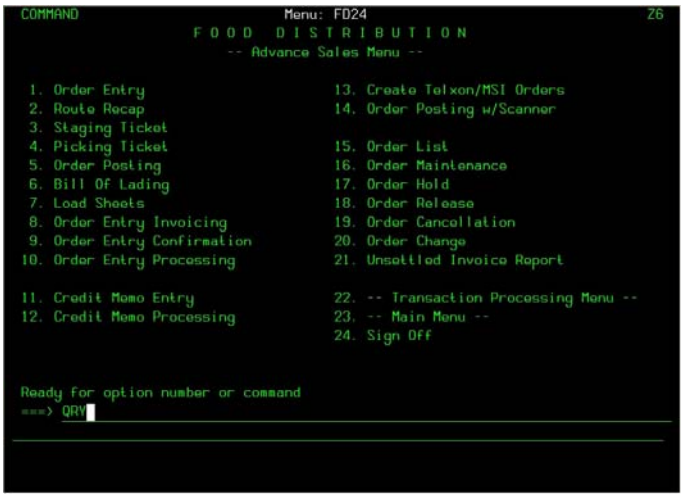
Engenheiro Informático

https://www.linkedin.com/in/ppico

INDÚSTRIA ALIMENTAR ALIADA ÀS TI'S

Para esta edição da Programar, aceitei o desafio de escrever um artigo que pretende relacionar a indústria alimentar com o sector das Tecnologias de Informação. Todos sabemos que por trás de um sistema informático, está pelo menos um programador. E um programador é por si só, um facilitador de processos. O exemplo prático que irei apresentar ao leitor reflete a ascensão que os sistemas informáticos estão a ter na indústria alimentar. Este é um dos temas em que estou a trabalhar atualmente na minha dissertação de Mestrado em Engenharia Alimentar (Instituto Superior de Agronomia). Ao estagiar numa pequena empresa que se encontra na fase de implementação é fundamental perceber o impacto que um sistema informático pode causar no dia-a-dia da produção, pois este permite uma melhor gestão e um acesso praticamente imediato à informação.

Só em Portugal, segundo dados apresentados no *eInforma*, existem atualmente 18.712 empresas da indústria alimentar, o que representa 11,4 % das indústrias transformadoras existentes no país. Ao longo dos anos o sector alimentar tem vindo a ajustar-se às novas tecnologias, tirando partido das mesmas, aliando o controlo da produção e qualidade à vertente tecnológica. Sendo um sector bastante conhecido pelo excesso de registos diários que se transformam em pilhas de papéis, com a evolução destes sistemas informáticos deixa-se de necessitar de papel passando-se a recorrer a um tablet existente no chão de fábrica. De um modo eficaz e em tempo real é possível transformar muitos dos dados em informação de apoio à gestão.



Quer pela via da legislação que foi mudando e impondo novos requisitos em termos de registos e de sistemas de informação, para comunicação de dados às autoridades competentes, quer pela necessidade de sistemas de informação mais simples e genéricos, que não careças de um departamento de informática interno a cada empresa, que desenvolva e mantenha o seu próprio software, como acontecia na década de oitenta do século passado, a evolução "guiou" o uso da tecnologia na industria alimentar, para o que podemos denominar de gestão de *smart manufacturing*. Uma vez que estes sistemas ao funcionarem ligados à internet e a intranets, conseguem ligar-se a vários departamentos do processo produtivo reportando, em tempo real, as suas atividades. A revista *Food Engineering* realizou recentemente um inquérito em que pergunta aos seus leitores, a área em que vêm o *smart manufacturing* a criar mais impacto no processo, e a resposta mais votada foi a organização no trabalho. O que não é de estranhar, uma vez que é possível relacionar num só programa as rotinas administrativas e produtivas:

- Gestão da Produção (ex: agendamento de ordens de fabrico);
- Área Administrativa e Financeira (ex: recursos humanos);
- Gestão de Stock;
- Entre outras funcionalidades.

O uso dos sistemas informáticos já começa a ser uma realidade cada vez mais recorrente de algumas indústrias alimentares, principalmente nas grandes indústrias que são as que possuem capital suficiente para procederem ao investimento tecnológico. Em Portugal já existem inúmeras empresas capazes de fornecerem este tipo de serviços, saliento duas que considero mais relevantes:

- Flow Technology da FoodInTech que é praticamente um veterano na indústria alimentar é capaz de fornecer uma gestão na produção, qualidade e segurança alimentar;
- Prodsmart que está a dar os primeiros passos na indústria alimentar estando a evoluir de forma a responder às necessidades dos seus clientes, fornece atualmente um programa de gestão de produção.

Pegando nestes dois exemplos, podemos afirmar que o grande desafio destas duas empresas é demonstrar as inúmeras vantagens que os seus serviços podem possuir, considerando cada cliente único. Têm assim como principal objetivo oferecer aos seus potenciais clientes um serviço que seja flexível e ajustável às necessidades de cada área, sendo imperativo que não se obriguem as empresas a sujeitarem-se a uma reestruturação incompatível com a sua realidade financeira e com a sua dimensão. Confesso que o primeiro contacto com esta nova forma de gestão tive em conta a expres-

são “primeiro estranha-se e depois entranha-se” de Fernando Pessoa, mas depois ao pensar fora do quadrado conseguimos perceber que os vários departamentos existentes numa empresa ao funcionarem em tempo real, interligados entre si conseguem trabalhar e produzir de forma muito mais eficiente.

Apesar do sistema informático envolver um elevado investimento inicial, acredito que este investimento irá fazer com que a médio prazo todas as pequenas, médias e grandes empresas acabem por aderir. Para este facto, também concorre o aparecimento de software disponibilizado em modelo de retalho, dedicado às industrias alimentares e com grandes mecanismos de implementação que dão às empresas a possibilidade de ajustarem o sistema a si, sem terem de se ajustar ao sistema, promovendo a concorrência pela diferenciação de produto e evitando as chamadas “guerras de preços”, que normalmente resultam numa espiral de descidas até à venda do produto com margens de lucro incomportavelmente baixas e por consequência o “desastre”. Para as empresas continuarem a competir no mercado, necessitam de estar numa permanente atualização e comprometidas com a inovação. O investimento tecnológico é sem dúvida para muitas empresas fulcral, porque conseguem manter-se ativas no mercado e vir a reduzir os seus custos de produção a curto prazo através de uma eficiente gestão de recursos.

Com a massificação de tecnologias de comunicação que permitem ter no limite “tudo” ligado à internet, como o caso do IoT e dos sistemas cloud, é altamente provável que a industria cada vez mais aposte na tecnologia, como forma de se diferenciar e promover a concorrência no mercado, de formas “inteligentes”.

AUTOR

Escrito por Vanessa Faquir dos Santos

Licenciada em Engenharia Alimentar encontra-se actualmente a terminar o mestrado em Engenharia Alimentar com especialização em Processamento de Alimentos no Instituto Superior de Agronomia.

A ENGENHARIA DE SOFTWARE,

A QUALIDADE FINAL DO SOFTWARE E O PAPEL DO PROFISSIONAL DE DESENVOLIMENTO

O termo *Engenharia de Software* como é conhecido foi cunhado e usado pela primeira vez pelo professor Friedrich Ludwig Bauer em 1968 na primeira conferência dedicada ao assunto patrocinado pelo *NATO Science Committee* (NAUR & RANDELL, 1969), seu surgimento decorreu da análise feita na época sobre as condições da indústria de *software* que estava entrando em um período crítico de colapso que ficou conhecido pela alcunha de *crise do software* que teve seu início em meados da década de 1960, quando os programas existentes tornaram-se difíceis de serem mantidos, estendendo-se até o final da década de 1970 (PRESSMAN, 1995, p. 6).

A *crise de software* foi uma decorrência da imaturidade do mercado e dos profissionais da computação da época, pois vinha de um período onde o desenvolvimento do software não exigia requisitos e configurações complexas, sua utilização era, em média, limitado ao ambiente em que era desenvolvido. De fato, em meados de 1965 o termo *crise de software* não havia sido usado, isto ocorreu durante a década de 1970 quando às dificuldades relacionadas ao desenvolvimento do *software* começaram a ser mais graves, principalmente no aumento das demandas e da complexidade que o *software* passava a ter que executar, frente a inexistência de técnicas adequadas para resolver tais desafios (ENGHOLM JR, 2010, p. 32-33). Foi a partir desta lacuna que os princípios da *Engenharia de Software* tomaram forma e passou-se a considerar o que apregoava Bauer, o *software* passou a partir de então a ser visto como um produto e como tal necessita ser desenvolvido a partir de critérios de produção acentuados na busca de sua qualidade, custo adequado e entregue dentro dos prazos prometidos (PRESSMAN, 1995).

A *Engenharia de Software* foca sua preocupação em manter o controle sobre todas as fases do processo de desenvolvimento do *software* por meio de métricas voltadas ao controle produtivo dessas aplicações. Esse controle é calcado sob a ótica de uso de diversos paradigmas (modelos metodológicos) de controle e acompanhamento. Cada paradigma, em particular, possui características relacionadas ao tipo de *software* desenvolvido ou ao tamanho que certa aplicação deverá possuir para atender as necessidades de seus clientes. Essas preocupações começaram a levar os analistas de sistemas de simples gerentes de projetos que pensavam e olhavam apenas a aplicação das regras computacionais em seus projetos para que o software atendesse o mínimo da necessidade dos clientes a um nível superior de ordem onde se soma as preocupações tradicionais outras como o custo da produção do software e a qualidade efetiva, além de ter que se preocupar com os métodos de desenvolvimento e suas ferramentas de aplicação. Frente a isso a definição sobre *Engenharia de Software* apregoada por Bauer toma força como mostra Pressman (1995, p. 31):

> *O estabelecimento e uso de sólidos princípios de engenharia para que se possa obter economicamente um software que seja confiável e que funcione eficientemente em máquina reais.*

Os paradigmas das metodologias de desenvolvimento de *software* são segundo Pressman (2006, p. 37) “propostos para colocar ordem no caos do desenvolvimento do software”, são processos metodológicos que podem acomodar atividades genéricas entre eles, mas guardam a particularidade que cada método necessita para controlar o projeto a que é submetido (PRESSMAN, 2006, p. 38). O objetivo principal do uso de tais métodos é garantir uma melhor qualidade no processo de desenvolvimento do software, devido a esta característica operacional presumisse serem esses métodos ferramentas que auxiliam a obtenção da qualidade final no processo de desenvolvimento, pois é melhor uma ferramenta simples que garanta um mínimo de mensuração a qualidade do que não ter absolutamente nada com que medir e acompanhar.

Pressman (2006, p. 38-51) comenta sobre o uso de alguns métodos de acompanhamento ao desenvolvimento de software podendo-se destacar resumidamente:

- Modelo em cascata – proposta de acompanhamento sequencial e linear no processo de desenvolvimento sendo útil quando os requisitos do sistema são estáveis e bem definidos, usados em projetos de pequeno porte com baixa margem de manutenção;
- Modelo incremental – proposta de produção de software a partir de diversas versões e incrementos, sendo adequado para a produção de projetos maiores em espaço de tempo limitado;
- Modelo evolucionário – proposta usada para acomodar modificações que sejam constantes na fase de desenvolvimento do projeto, sendo esta que melhor usa a natureza iterativa da maioria dos projetos de engenharia de software adequada para projetos maiores;
- Modelo baseado em componentes – proposta

baseada na reutilização de componentes prontos de outros projetos, desde que possam ser conectados ao projeto em desenvolvimento, adequado para projetos maiores;

- Processo unificado – proposta de projeto baseada na orientação de casos de uso centrado na arquitetura incremental e iterativa, adequado para projetos maiores.

Na *Engenharia de Software* há a consideração de que não é possível gerenciar aquilo que não pode ser medido (PRESSMAN, 1995, p. 55-61). Isto somente é possível de ser executado com o uso de ferramentas adequadas ao acompanhamento das etapas do trabalho planejado. Fazer uso de técnicas de planejamento consiste em determinar antecipadamente o que deve ser feito e como será feito, considerando sempre as metas a serem atingidas (SILVA, 1997, p. 24). Além das atividades de planejamento que visualiza o que será realizado, como será realizado e por quem será realizado, o engenheiro de software necessita adotar critérios de gerenciamento do processo de desenvolvimento de software, uma vez que precisa medir constantemente as atividades de desenvolvimento e evolução, não importando o modelo de acompanhamento de projeto escolhido para determinado projeto. Não obstante, as atividades de planejamento e gerenciamento do processo de produção do software estão, na prática, interligadas e não podem ser vistas separadamente, pois é por meio dessas atividades que se iniciou a partir da *crise do software* uma preocupação maior com a qualidade do desenvolvimento do software e dos profissionais dessa área.

Cabe uma observação, mesmo que empírica e crítica, sobre a *Engenharia de Software*, pois a partir do dito por Bauer acaba-se dando foco aos princípios de controle do planejamento e do processo de desenvolvimento apenas sob a ótica da engenharia, deixando-se de lado o fato que não é apenas a área de engenharia a fazer uso dos princípios indicados por Bauer, pois muitos desses princípios são encontrados na área da *administração de empresas*. É necessário considerar que o perfil do profissional de engenharia de software deve ir além das técnicas computacionais e de gerenciamento e controle que ele aprende na escola, que sem sombra de dúvida são necessários aos exercícios de suas funções, mas este profissional necessita de uma visão coadjuvante da área fim que os produtos da computação que vai gerenciar serão direcionados. Se o engenheiro de software direcionar seus esforços em atender empresas do ramo comercial precisará ter treinamento para entender as particularidades do setor e a real necessidade de seus profissionais, o mesmo deve ser considerado para outros setores produtivos da sociedade, não deve apenas ter uma visão técnica do processo de desenvolvimento de software.

Mais recentemente vem surgindo cursos de graduação na área de *Engenharia de Software* reconhecidos pelo Ministério da Educação brasileiro. Mas, parece este esforço estar muito distante do aceite da profissão pelo Conselho Regional de Engenharia e Agronomia (CREA) representante das carreiras de engenharia. No entanto, nem mesmo outras atividades da produção de software executadas por programadores e analistas de sistemas são reconhecidas (TEBALDI, 2013, p. 28), apesar das discussões já efetuadas no Congresso Nacional como mostra o Projeto de Lei 1561 de 2003, proposto pelo Sr. Ronaldo Vasconsellos que teve seu despacho registrado em 21 de outubro de 2004 (CÂMARA DOS DEPUTADOS, 2003) e se encontra arquivado desde então, pelo menos até setembro de 2015.

Outro ponto de conflito em relação ao exercício das atividades como engenheiro de software é o uso das ferramentas computacionais voltadas para o setor, enquanto a engenharia civil possui ferramentas sofisticadas que podem ser usadas em computadores ao seu exercício o setor de produção de software não tem tanta sorte. Parece que foi mais fácil produzir ferramentas computacionais para outras áreas da engenharia do que para o uso da própria produção de software, como diz o dito “em casa de ferreiro o espeto é de pau”. Apesar de existirem ferramentas para uso da *Engenharia de Software* parece que elas são um tanto distantes e não tão bem elaboradas como as usadas pelas outras engenharias.

Pressman (1995) compara a evolução das engenharias civil, mecânica e elétrica que em 1995 utilizavam livros, tabelas, réguas de cálculo e calculadoras mecânicas e começaram a partir de 1965 experimentar uma engenharia baseada em computadores e que a partir de 1975 começaram a usar ferramentas que possuíam tudo o que precisavam embutidos em diversos programas de computador. Atualmente vê-se tudo isso na palma das mãos por meio do uso de *smatphones* e *tablets*. Esse grupo de engenheiros possui atualmente uma interface humano-computador bastante vantajosa aos seus serviços. Pressman aponta que as ferramentas chamadas de CASE (*Computer Aided Software Engineering*), ou seja, a *Engenharia de Software Assistida por Computador* estava por volta de 1995 onde estavam as demais engenharias no uso de ferramentas computacionais em 1975 e parece que não houve grandes avanços neste sentido se olhar as ferramentas existentes na época e as atuais, que são basicamente as mesmas com pequenas ou poucas melhorias.

Se as ferramentas computacionais para a *Engenharia de Software* estão um tanto estagnadas, já os método e técnicas de gerenciamento apresentaram um nível de evolução mais acentuados como o surgimento das técnicas de desenvolvimento ágil como o *Extreme Programming* (XP) que surgiu no final da década de 1990 nos Estados Unidos e foca o processo de desenvolvimento do código do software (TELES[1], 2014) voltado a grupo de desenvolvedores

pequenos e médios (BECK, 2008, p. xi), além da metodologia *Scrum* voltado para gestão e planejamento de projetos (TELES[2], 2014). Esses métodos mais recentes parecem indicar algumas respostas as perguntas apontadas por Pressman (1995, p. 9):

- Por que demora tanto tempo para que os programas sejam concluídos?
- Por que os custos são tão elevados?
- Por que não são descobertos todos os erros antes do software ser entregue ao cliente?
- Por que as dificuldades em medir o progresso do software enquanto está sendo desenvolvido?

As metodologias XP e Scrum podem ajudar a *Engenharia de Software* a desenvolver softwares de melhor qualidade, produzidos em menos tempo e com custo menor (TELES[1], 2014), pois permitem a participação do usuário. Segundo Heuser (2009, p. 29) o "envolvimento do usuário na especificação do software aumenta a qualidade do software produzido" quando menciona a prática da *Engenharia de Software*. Essas técnicas se mostram voltadas a projetos de pequeno e médio portes. Projetos maiores podem enfrentar outros tipos de problemas, necessitando de outras técnicas de gerenciamento, neste sentido, pode-se fazer uso de ferramentas como PERT (*Programa Evaluation and Review Technique*) usado na década de 1970 pelo governo norte-americano para a construção da série de submarinos atômicos Polaris e o CPM (*Critical Payh Method*), da mesma época, desenvolvido pela empresa Du Pont para controlar projetos de engenharia (PRADO, 2004, p. 15). As ferramentas PERT/CPM foram desenvolvidas para o gerenciamento de grandes projetos e podem ser facilmente usados na produção de software e possuem no mercado ferramentas consagradas baseadas nesta dinâmica, sendo o MS-Project da Microsoft (menos robusto com custo baixo de aquisição) e o Primavera Project Management da Oracle (mais robusto com custo mais alto de aquisição, se comparado ao Project da Microsoft).

Cabe considerar que a *Engenharia de Software* propõe levar em consideração o desenvolvimento de software de maneira mais profissional e séria. Nos incentiva a deixar de lado métodos empíricos e pessoais e adotar outras medidas de trabalho, buscando o desenvolvimento de aplicações que venham a atender a necessidade dos clientes e que estejam dentro de um custo e tempo de desenvolvimento aceitáveis.

BIBLIOGRAFIA

BECK, K. **Programação Extrema aplicada**. Porto Alegre: Bookman, 2008.

CÂMARA DOS DEPUTADOS. **Projetos de Leis e Outras Proposições: PL 1561/2003**. Brasília: Palácio do Congresso Nacional, 2003. Disponível em: <http://www.camara.gov.br/proposicoesWeb/fichadetramitacao?idProposicao=126039 >. Acesso em: 23 de fev 2014.

ENGHOLM JR, H. **Engenharia de Software na Prática**. São Paulo: Novatec, 2010.

HEUSER, C. A. **Projeto de banco de dados**. 6. ed. Porto Alegre: Bookman, 2009.

NAUR, p. & RANDELL, B. **Software Engineering: A Reporto n a Conference Sponsored by de NATO Science Committee**. Garmisch, Germany: NATO, 1969. Disponível em: <http://homepages.cs.ncl.ac.uk/brian.randell/NATO/nato1968.PDF>. Acesso em: 22 de fev. 2014.

PRADO, D. **PERT/COM**. 4. ed. Belo Horizonte: INDG, 2004.

PRESSMAN, P. **Engenharia de Software**. 6. ed. São Paulo: McGraw-Hill, 2006.

_____. **Engenharia de Software**. São Paulo: Makron Books, 1995.

SILVA, A. de T. **Administração e Controle**. 10. ed. São Paulo: Atlas, 1997.

TEBALDI, J. Z. F. **Direito e Ética: Legislação Aplicada**. Batatais: Claretiano, 2013.

TELES[1], V. M. **Extreme Programming**. DesenvolvimentoAgil.com .br, 2014. Disponível em: <http://desenvolvimentoagil.com.br/xp/>. Acesso em: 23 de fev. 2014.

TELES[2], V. M. **Scrum**. DesenvolvimentoAgil.com.br, 2014. Disponível em: <http://desenvolvimentoagil.com.br/scrum>. Acesso em: 23 de fev. 2014.

AUTOR

Escrito por Augusto Manzano

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

ENTREVISTA A VÂNIA GONÇALVES

Revista PROGRAMAR - (RP): Fale-me um pouco de si e do seu percurso na área das tecnologias.

Vânia Gonçalves - (VG): Sou licenciada em Eng. Informática e Computação pela FEUP e antes da conclusão do curso comecei a trabalhar nas áreas de desenvolvimento de software e segurança e gestão de redes. Mais tarde concluí um Mestrado em Políticas Tecnológicas pela Universidade de Cambridge e desde então tenho-me focado mais na investigação da gestão de inovação tecnológica, políticas e modelos de negócio associados. Atualmente sou docente convidada do Departamento de Engenharia Informática da FEUP e Coordenadora de projetos de Investigação e Desenvolvimento na NMusic.

RP: Como surgiu a comunidade Portugal Girl Geek Dinners e qual a sua missão?

VG: Conhecia os eventos Girl Geek Dinners em Bruxelas, sempre esgotados e com mais de 80-100 mulheres a participar. Na altura não havia nenhuma comunidade ou eventos semelhantes em Portugal, que unissem mulheres e o gosto por tecnologia. Comecei por organizar os eventos cá, enquanto ainda vivia em Bruxelas e às vezes era difícil motivar mulheres a participarem. A missão da comunidade Portugal Girl Geek Dinners é encorajar e inspirar jovens mulheres a seguirem uma carreira na área da tecnologia e potenciar a partilha de histórias, troca de experiências, e relações em rede entre mulheres com carreiras na área tecnológica.

RP: Quais as dificuldades mais difíceis de contornar para tornar os encontros possíveis?

VG: A disponibilidade de tempo para manter os encontros e motivar as voluntárias que organizam os encontros em várias cidades. É normal as pessoas terem alterações na sua vida profissional e privada e deixam de estar disponíveis. Por isso, é necessário garantir frequentemente que temos pessoas motivadas e a contribuir voluntariamente para esta comunidade.

RP: O primeiro encontro surgiu em Dezembro de 2009 em Coimbra. O que mudou desde então?

VG: Muito mudou! Desbravou-se muito caminho! Temos cada vez mais mulheres a participar nos encontros e, acima de tudo, mais conscientes para as vantagens de participar neste tipo de comunidades e evento e do contributo tanto para o seu desenvolvimento pessoal como profissional. Entretanto, também há agora uma grande escolha de comunidades na área tecnológica e focadas no networking, e também exclusivamente para mulheres. Tudo pode coexistir e o importante é que as pessoas se envolvam e tirem o maior partido disso.

RP: Que preciso fazer para envolver-me com a comunidade PGGD?

VG: Participar num dos nossos encontros o mais depressa possível! :) Há também a possibilidade de nos seguir nas redes sociais Facebook, Linkedin e Twitter. Mas é mesmo na mailing list e na equipa do Slack que as mulheres do nosso grupo comunicam mais.

RP: Em que cidades estão presentes núcleos PGGD? E se uma cidade não tiver núcleo, o que precisam de fazer para passar a ter?

VG: Neste momento há encontros, com mais ou menos regularidade, no Porto, Lisboa, Coimbra, Leiria e vamos tentar regressar a Braga depois do Verão. Para começar em novas cidades é necessário reunir um grupo de 2 a 3 voluntárias e propor a existência de um novo grupo explicando a principal motivação. Por exemplo, em cidades com universidades que oferecem cursos na área tecnológica ou cidades com pólos empresariais com empresas tecnológicas haverá certamente mulheres na área tecnológica. É essas (jovens) mulheres que queremos que comecem a falar entre si e a trocar experiências.

RP: O que é que distingue o vosso movimento de outros encontros similares?

VG: Penso que neste momento, e talvez por ter sido o primeiro, é o movimento que reúne mais mulheres apaixonadas por tecnologia em várias cidades espalhadas pelo país. A nossa mailing list reúne quase 300 mulheres e há uma comunicação constante e um espírito de entre ajuda que não conheço noutras comunidades. As relações que se têm criado nos encontros têm facilitado muitas mulheres a encontrarem novas oportunidades de emprego e a sentirem-se mais confiantes no trabalho.

RP: Em que medida o PGGD contribui para o presente e futuro dos profissionais e apaixonados das TI?

VG: O PGGD contribui essencialmente para criar uma vasta rede de mulheres interessadas em tecnologia e dar-lhes as condições para estabelecerem contatos com outras mulheres e empresas de forma a trazer-lhes mais-valias profissionais. Nos nossos encontros, tentamos sempre abordar temas tecnológicos de vanguarda para enriquecer o conhecimento e fomentar o posterior estudo individual do tema.

GIRL GEEK DINNER
definitely does compute

PROJECTO EM DESTAQUE NA COMUNIDADE P@P: DIRT BIKE EXTREME

Num ambiente industrial, com um belo cenário de fundo encontram-se muitos obstáculos pela frente, cair vezes sem conta é quase natural, mas com tempo o jogador, acaba tornando-se um profissional! Com perseverança tudo se consegue. Ao longo dos 20 níveis que compõem o jogo, uns vão parecer impossíveis outros nem por isso, mas todos contribuiem para que o jogador se torne um profissional de Dirt Bike Xtreme!

Além do modo carreira está também disponível o modo multiplayer. É simples, basta criares uma sala e escolheres o nível que se quer jogar, depois é só convidar amigos para a sala e jogar com eles. Algo bastante simples e que torna o jogo ainda mais interessante.

Um jogo inteiramente português, recheado de diversão, com um grafismo bastante agradável e muitas manobras dignas de um “Eveil Kenivel”, dos tempos modernos. É de facto uma “prova provada” de que em terras lusas, ainda se fazem jogos de qualidade, como nos tempos áureos dos videojogos made in Portugal.

Veja também as edições anteriores da Revista PROGRAMAR

52ª Edição - Março 2016

51ª Edição - Dezembro 2015

50ª Edição - Setembro 2015

49ª Edição - Junho 2015

48ª Edição - Março 2015

47ª Edição - Dezembro 2014

e muito mais em ...

www.revista-programar.info

DUVIDAS?

IDEIAS?

AJUDAS?

PROJECTOS?

portugal-a-programar
.org