Apresentação CGII

Luis Filipe de Sá Estrella DRE: 107390627
Glauco Barbosa Primo DRE: 106030096

Programando para iOS no Xcode 3.2.5 com SDK 4.2

Máquinas Virtuais

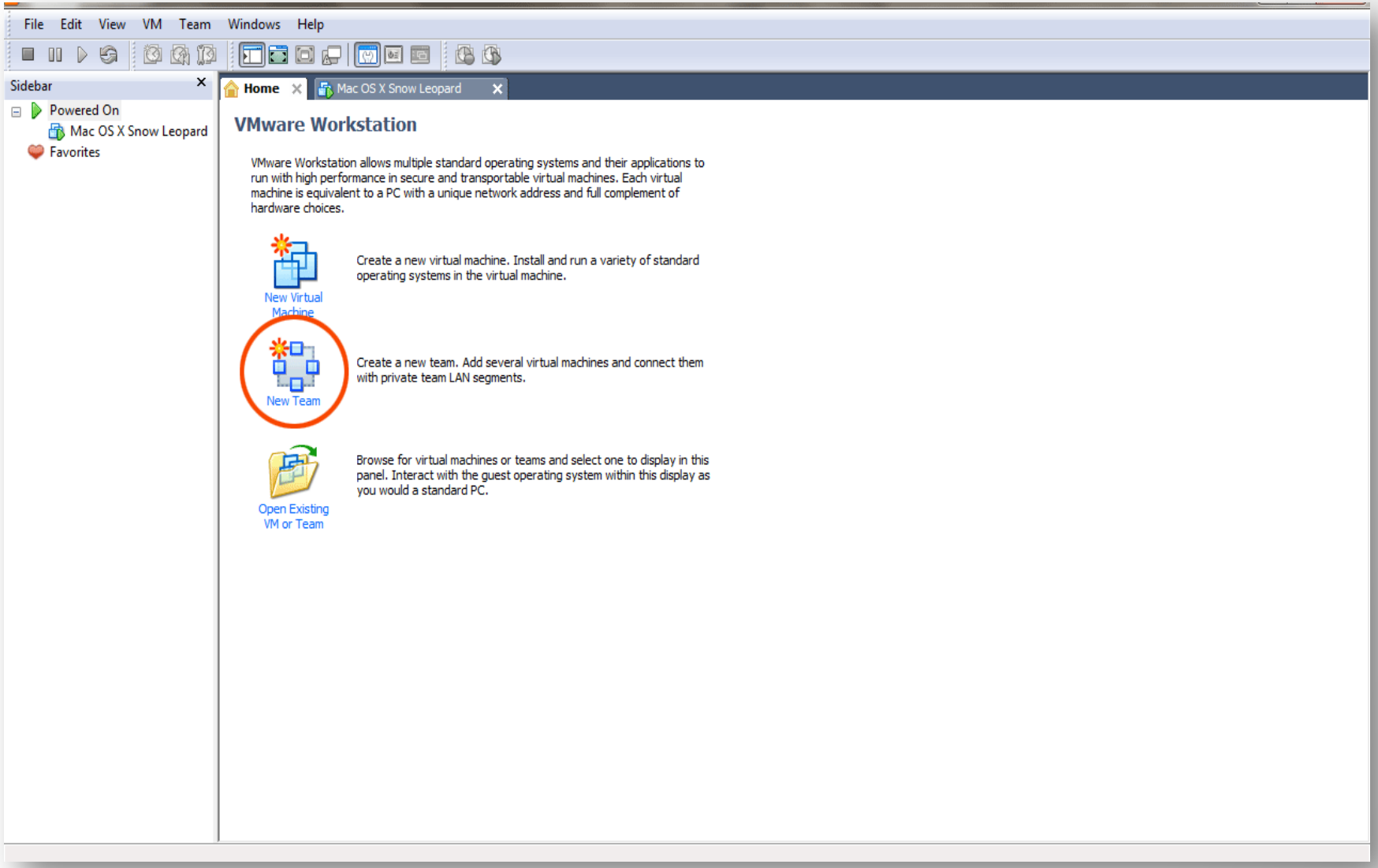
- **Vantagens:**
  - Não precisa de Novas Partições, ou formatar o Disco.
  - Não influencia no funcionamento do seu SO
  - Fácil remoção

- **Desvantagens:**
  - Depende de Hardware
  - Processador precisa de Virtualização Habilitada
  - Necessita de hardware potente para um bom desempenho

Vmware

- Para o projeto que será mostrado a seguir, utilizamos a máquina virtual Vmware Workstation 7.7.1 e a imagem do MacOs 10.6.8.
- Para instalação do sistema operacional você não terá grandes problemas. Basta Clicar em Open Existing VM or Team e Selecionar o '.vmx' de sua imagem.

Vmware


File Edit View VM Team Windows Help
Sidebar
Powered On
Mac OS X Snow Leopard
Favorites
Home
Mac OS X Snow Leopard
VMware Workstation
VMware Workstation allows multiple standard operating systems and their applications to run with high performance in secure and transportable virtual machines. Each virtual machine is equivalent to a PC with a unique network address and full complement of hardware choices.
New Virtual Machine
Create a new virtual machine. Install and run a variety of standard operating systems in the virtual machine.
New Team
Create a new team. Add several virtual machines and connect them with private team LAN segments.
Open Existing VM or Team
Browse for virtual machines or teams and select one to display in this panel. Interact with the guest operating system within this display as you would a standard PC.

Configurações

- Neste tutorial foi utilizado o seguinte Hardware:
  - Dell Vostro 1320 Centrino Inside
  - Windows 7 64 Bits
  - 4 GB Memoria Ram DDR2
  - Intel Core 2 Duo T6670 2.20 GHZ
  - Video Intel Integrado
  - Rede Intel Integrado
  - Ipod Touch 2ª geração Firmware 4.2

Xcode

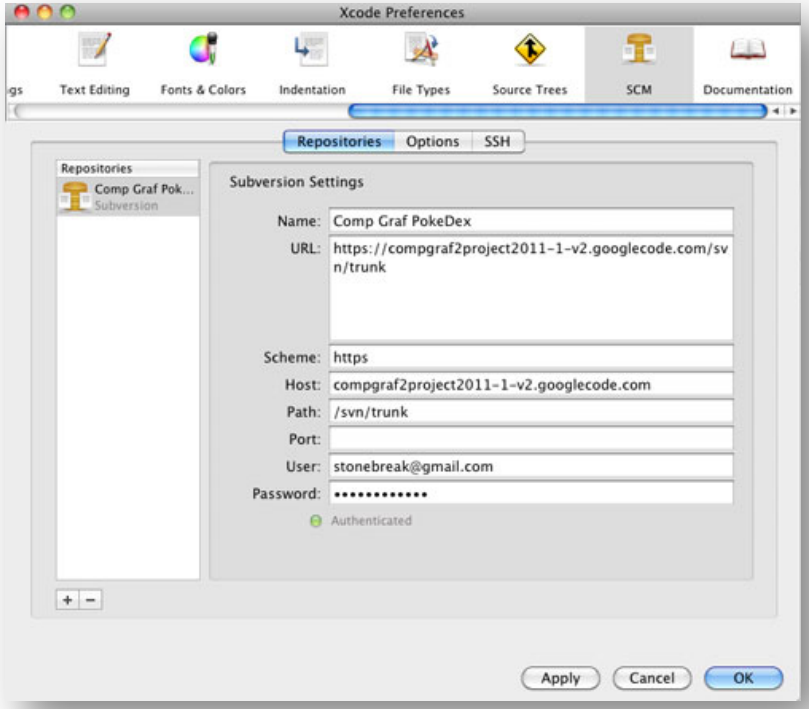
- Para programar para iOS você deve ter uma IDE, no caso utilizamos o Xcode 3.2.5 com iOS SDK 4.2
- Com isso você conseguirá programar para iPhone, iTouch e iPad.
- **Advertências**: Não baixe o Xcode 4 pois é pago. Não baixe também a SDK 3 pois você não conseguirá executar a aplicação no iPhone/iTouch a partir do Xcode.

Subversion (SVN)

- Depois que seu Xcode estiver devidamente funcionando você fatalmente precisará de um controle de versão. E tal qual no Eclipse utilizaremos o Subversion (SVN).
- **Primeiros Passos:**
  - Se seu projeto já foi criado você deve abrir o terminal
  - Abra o terminal em /Applications/Utilities/
  - Digite svn import **pathlocal subversionpath** –m “initial import”,
  - Substitua pathlocal pelo diretorio onde está o seu projeto e substitua subversionpath pelo endereço do seu repositório.
  - Ele vai pedir a senha do Admin do OS, depois pedirá o login no SVN e depois a senha.

Subversion (SVN)

- Para o Checkout do Repositório:
  - Felizmente podemos fazê-lo pelo Xcode.
  - Vá em SCM no menu superior do Xcode.
    - Clique em Configure SCM For This Project e preencha os dados

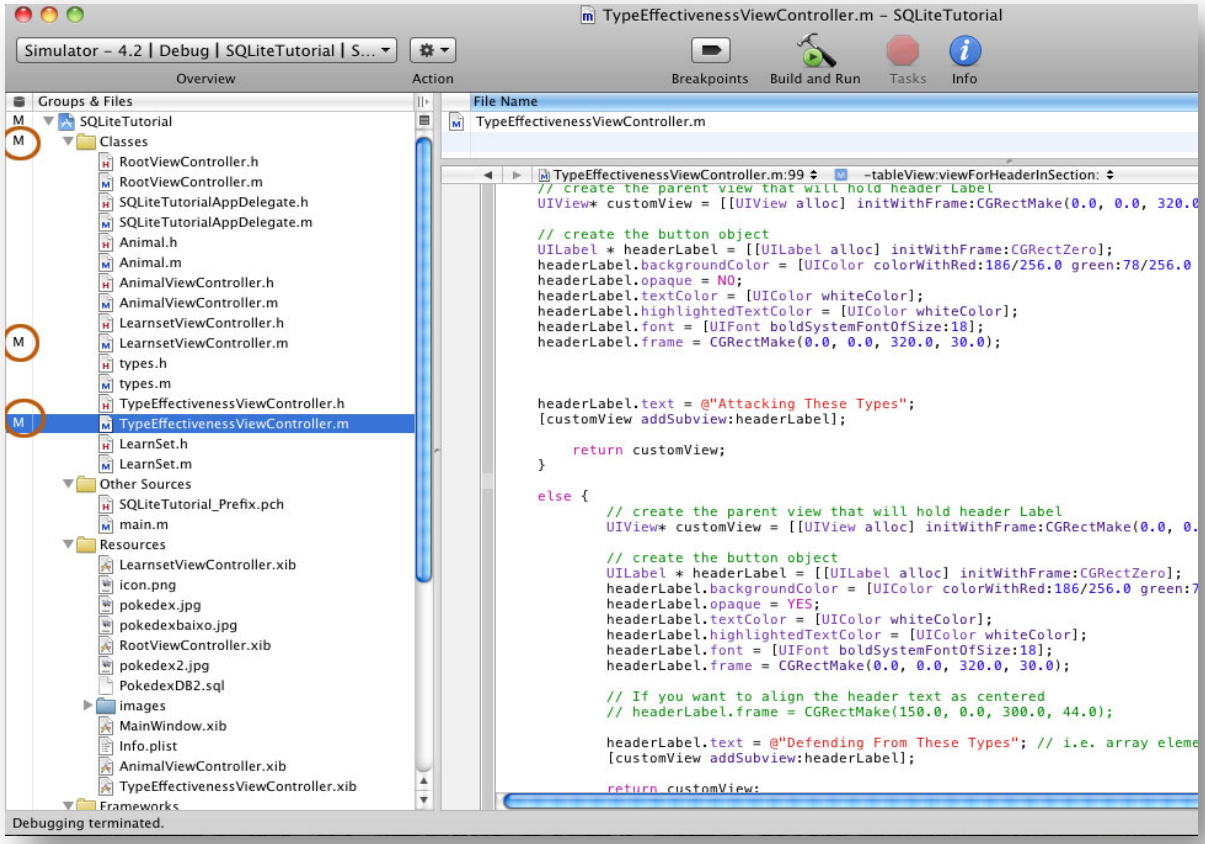


Subversion (SVN)

- Click em Ok
- Sua lista de diretórios aparecerá em seguida
- Simplesmente dê checkout (mas não sobrescreva a cópia anterior; crie um novo diretório para receber o checkout), esse diretório passará a ser seu workspace e tudo que você mudar ficará out-of-date com a letra M antes do arquivo na lista de diretórios (e conseqüentemente poderá dar commit). Se estiver com a letra U signfica que o seu arquivo está out-of-date em relação ao do repositório e você precisará dar Update.

Subversion (SVN)

- O M abaixo significa que você alterou o arquivo e ele precisa ser commitado.



Xcode iPhone 4 Build

- Para fazer os passos abaixos você deve ter o Iphone com Jailbreak, e AppSync versão 4.2
- Para instalar o AppSync Adicione o source http://cydia.hackulo.us no cydia e instale o app. Após instalado **reboot o iPhone.**
- Para que possamos executar o programa no iPhone diretamente do Xcode (Build and Run) faremos os seguintes passos:
- Abra o Terminal (Utilitarios/Terminal) e execute os comandos:

```
cd  /Developer/Platforms/iPhoneOS.platform/Developer/SDKs/iPhoneOS4.2.sdk   <ENTER>
sudo cp SDKSettings.plist SDKSettings.plist.orig                            <ENTER>
sudo vi SDKSettings.plist                                                   <ENTER>
```

Xcode Iphone 4 Build

- Procure a seguinte linha no arquivo aberto e altere de Yes para NO:

  / <key>CODE_SIGNING_REQUIRED</key>

  <string>YES</string>

- Agora procure esta outra linha e altere o YES para NO:

  / <key>ENTITLEMENTS_REQUIRED</key>

  <string>YES</string>

Xcode Iphone 4 Build

- Agora precisamos alterar o arquivo Info.plist

```
cd /Developer/Platforms/iPhoneOS.platform/      <ENTER>
sudo cp Info.plist Info.plist.orig              <ENTER>
sudo vi Info.plist                              <ENTER>
```

- Neste arquivo irá aparecer duas vezes o código:

```
<key>CODE_SIGN_CONTEXT_CLASS</key>
<string>XCiPhoneOSCodeSignContext</string>
```

  - Alterar as linhas <string>XCiPhoneOSCodeSignContext</string>

Para : <string>XCCodeSignContext</string>

Xcode Iphone 4 Build

- Agora preciamos fazer as alterações no Xcode:

```
cd ~/Desktop        <ENTER>
vi script           <ENTER>
```

Pressione a tecla “i” para entrar no modo de inserção e depois cole o texto abaixo:

```
#!/bin/bash
cd /Developer/Platforms/iPhoneOS.platform/Developer/Library/Xcode/Plug-ins/iPhoneOS\ Build\ System\ Support.xcplugin/Contents/MacOS/
dd if=iPhoneOS\ Build\ System\ Support of=working bs=500 count=255
printf "\xc3\x26\x00\x00" >> working
/bin/mv -n iPhoneOS\ Build\ System\ Support iPhoneOS\ Build\ System\ Support.original
/bin/mv working iPhoneOS\ Build\ System\ Support
chmod a+x iPhoneOS\ Build\ System\ Support
```

Xcode Iphone 4 Build

- Pressione a tecla esc e em seguida as teclas “:” “x” “enter”. Então ainda no terminal digite:

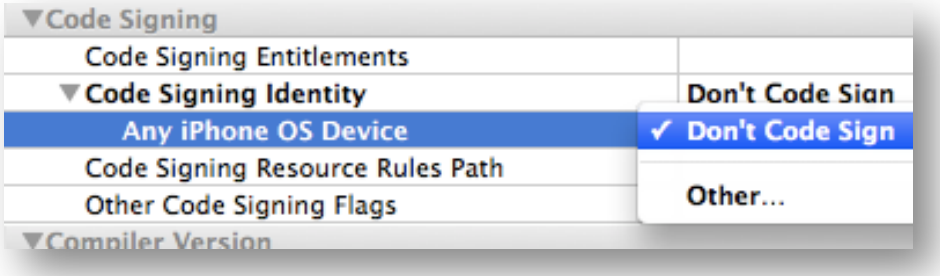
  chmod 777 script <ENTER>
  ./script <ENTER>

- Se funcionou você verá:

  $ ./script
  223+1 records in
  223+1 records out
  111648 bytes transferred in 0.002678 secs (41692099 bytes/sec)

Xcode Iphone 4 Build

- Com o projeto do Xcode aberto e pronto para se tornar um app, (compilado, sem erros ou warnings), Va no menu ***Project>Edit Project Settings***
- Clique na aba "**Build**". Ache "**Code signing Identity**" e em seguida a baixo "**Any iPhoneOS Device**" e insira a opção "**don't code sign**"



Xcode Iphone 4 Build

- Para finalizar, no Terminal, digite o seguinte:

```
mkdir /Developer/iphoneentitlements401                    <ENTER>
cd /Developer/iphoneentitlements401                       <ENTER>
curl -O http://www.ijust.com.br/iPhone/gen_entitlements.txt   <ENTER>
mv gen_entitlements.txt gen_entitlements.py               <ENTER>
chmod 777 gen_entitlements.py                             <ENTER>
```

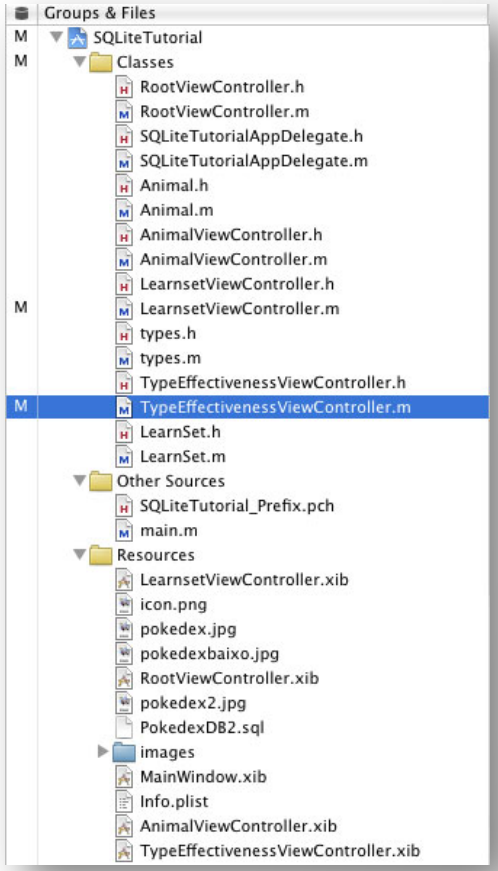
- Agora finalmente você pode plugar seu iPhone ou iTouch ao computador e abrir o Xcode. Va no menu ***Window>Organizer*** Escolha o dispositivo na lista à esquerda e clique no botão “Use for development”. Será aberta uma janela com user e senha, basta clicar em cancel.
- No Xcode você muda a opção de “Simulator” para “Device” e de “Release” para “Debug” e pronto.

A aplicação - Pokédex

- A aplicação é simples, consiste em uma agenda que guarda informações importantes sobre Pokemons™ - criaturas fantásticas criadas por Satoshi Tajiri - marca registrada da Nintendo.
- A idéia não é nova, a pokedex existe no mundo *pokemon* e apresenta as funcionalidades que foram implementadas e outras.
- Resolvemos criá-la para iPhone já que não existe uma pokedex free que seja realmente boa e que contenha informações sobre todos os pokemons catalogados até hoje.
- Muito útil para jogadores do Game que querem informações rápidas que o jogo não nos dá, dispensando então o uso de websites especializados.

A aplicação - Pokédex

- A Aplicação que demonstraremos a seguir se baseia no uso de banco de dados (no Xcode usamos o ***SQLite***) *e no uso de Multiviews.*
- Nossas classes se dividem em 8 tipos:
  - AppDelegate
  - RootViewController
  - Animal
  - AnimalViewController
  - Types e TypeEffectivenessViewController
  - LearnSet e LearnSetViewController

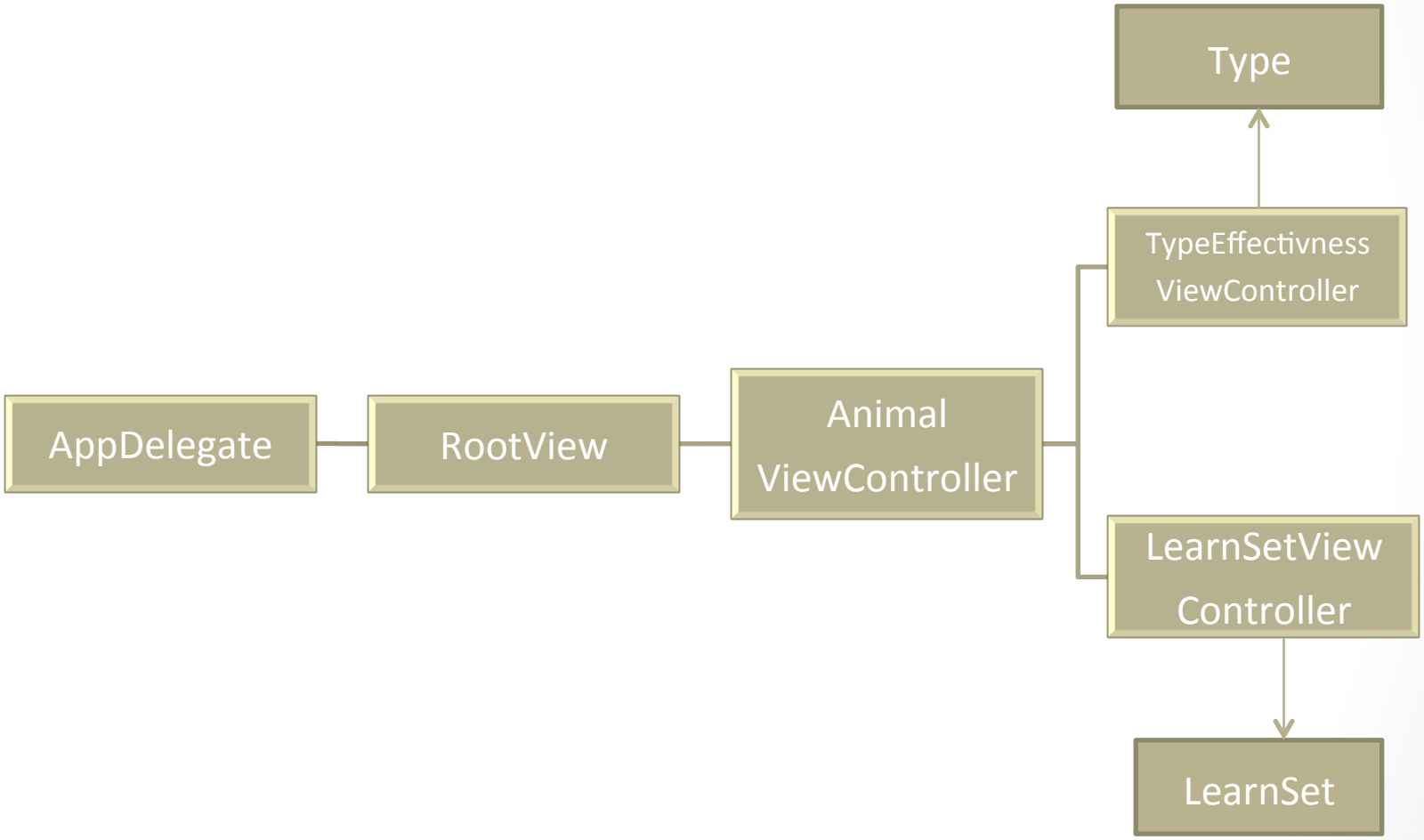


A aplicação - Pokédex

A aplicação tem um fluxo simples:

- O AppDelegate é responsável por construir os pokemons em um mutableArray, chamado “animals”. Esse array será o array que irá conter todas as informações que precisaremos de todos os pokémons.
- O RootViewController é responsável por instanciar o AppDelegate e colocar as informações pertinentes do banco(id e nome) em um tableview.
- No RootViewController também temos instanciado o SearchBar para buscar os pokémons na pokédex.
- Quando uma célula da root da pokédex é clicada chamamos o AnimalViewController, e então tratamos para colocar as informações do banco que queremos (Nome, Tipo1, Tipo2, Descrição, Espécie) a partir do ID da célula selecionada.
- No AnimalViewController temos os botões TypeEffectiveness e LearnSetViewController . O primeiro mostra as vantagens e desvantagens do pokémon em relação a outros tipos de pokémon; o segundo mostra todos os golpes que o pokémon vai aprender ao longo da vida.

A aplicação - Pokédex



A aplicação - Pokédex

- **AppDelegate** – É responsável por criar os pokemons, em nossa aplicação chamamos de animal.
- Esta classe herda da classe Animal.m que é uma struct que contém todas as informações do nossos pokemons
- Carregamos o banco e fazemos as operações nele utilizando o framework libsqlite3.0.dylib
- Para acrescentar o framework basta clicar com botão direito em *frameworks > Add > Existing Frameworks* e escolher o libsqlite3.0.dylib na lista.
- Criamos um mutableArray para armazenar os pokemons e suas informações.

A aplicação - Pokédex

- **RootViewController** – Responsável por pegar todos os objetos armazenados no mutableArray do appDelegate e colocálos ordenados em um table.
- Essa classe é uma view que tem uma instância de uma tableview e outra de um UIsearchBar.
- Esta classe instancia o AppDelegate, lê o seu MutableArray que contém as informações dos pokémons e o coloca na tableView.

A aplicação - Pokédex

- **AnimalViewController** – Contém as informações do pokémon selecionado e os botões TypeEffectivness e LearnSet.
- Esta view tem um objeto UIImageView que irá conter a imagem do pokémon e objetos UILabel que vão descriminar informações como espécie, tipos, peso e altura.
- Os dados nesses objetos serão inseridos por código através da RootViewController ao selecionar uma célula da tableView (os dados são pegos do MutableArray já citado anteriormente), pois a classe AnimalViewController têm variaveis OutLet associadas com os objetos em sua xib.
- Assim temos as informações básicas do pokemon escolhido.

A aplicação - Pokédex

- **TypeEffectiveness:** mostra os danos super ou pouco efetivos que o pokémon causa/recebe em todos os outros tipos de pokémons. Considere que um dano “normal” vale 100.
- A idéia é pegar o valor do dano do pokemon e dividir por 100, daí entraremos nos casos seguintes:
  - Se for **maior** que 1 no ataque, quer dizer que o dano é **super efetivo** contra o outro tipo.
  - Se for **menor** que 1 no ataque, quer dizer que o dano é **pouco efetivo** contra o outro tipo.
  - Se for **maior** que 1 na defesa, quer dizer que ele recebe muito dano do outro tipo, então ele é **pouco efetivo** defendendo.
  - Se for **menor** que 1 na defesa, quer dizer que ele não recebe muito dano do outro tipo, então ele é **super efetivo** defendendo.

A aplicação - Pokédex

- **LearnSet:** Mostra todos os golpes que um pokemon aprende e em qual nível ele aprende
- Pegamos o ID do pokemon selecionado na RootViewController e então fazemos a consulta no Banco, selecionando todos os golpes do pokemon com este ID e o respectivo nível em que eles são aprendidos.
- Os dados são então colocados em uma tableView.

Bibliografia

- http://osxdaily.com/2010/06/03/configuring-xcode-to-use-subversion/
- http://www.ijust.com.br/iPhone/
- http://gigaom.com/apple/using-subversion-with-xcode-30/
- http://pt.wikipedia.org/wiki/Pok%C3%A9mon
- http://maczealots.com/tutorials/xcode-svn/
- http://svnbook.red-bean.com/
- http://dblog.com.au/iphone-development-tutorials/iphone-sdk-tutorial-reading-data-from-a-sqlite-database/