Iniciando tudo de novo. Começarei de novo pois irei fazer o aplicativo em Java e usarei o Dropbox, o CodeAnywhere e BitBucket.

# Enviando repositório do computador para o Bitbucket

Para iniciar um repositório novo no Bitbucket, a forma mais fácil parece ser criar o repositório no computador e depois enviar o repositório para o bitbucket. Para fazer isso, seguir os seguintes passos:

* 1. Criar pasta do projeto
  2. Dar um gitbash na pasta
  3. Dar um “git init”
  4. Entrar no SourceTree e dar o primeiro commit no projeto (ainda local); Usar a aba de “adicionar cópia de trabalaho” e não a opção de “clonar” repositório.
  5. Voltar a pasta e dar os seguintes comandos para enviar para o Bitbucket:
     1. git remote add origin https://fabiomolinar@bitbucket.org/fabiomolinar/photo2me-android.git
     2. git push -u origin –all
     3. git push origin --tags # pushes up any tags

# Criando splash screen

Passos:

1. Criar a activity;
2. Setar a activity como LAUNCH no manifesto
3. Editar o layout dela
4. Para que o fundo ocupe a tela inteira, é necessário criar um novo style.

# Fundo bicolor e radial

Para aplicar alguns efeitos no background, temos que usar um @drawable do tipo shape em um arquivo xml.

# Usando HTML em textviews

Ótima forma de usar html dentro de textviews (se forem bem simples) é converter o html para um resource do tipo string e depois setá-lo ao textview. Para fazer um HTML ser definido dentro de um resource string basta usar: <![CDATA[ html aqui dentro ]]>. Exemplo abaixo:

<string name="termos">  
 <![CDATA[  
 <h1>Título</h1>  
 <p>Aqui irá o parágrafo falando dos termos de uso e etc.</p>  
 <h2>Sub Título</h2>  
 <p>Aqui vai mais um parágrafo falando de um monte de coisa.</p>  
 ]]>  
</string>

# Depois, basta setar essa string ao textview da seguinte forma:

textView.setText(Html.*fromHtml*(getString(R.string.*termos*)));

# Image and svg sizes

Tomar cuidado com os tamanhos das imagens. Precisava usar uma imagem vetorizada no projeto e, como o importador de .svg do Android Studio não funciona muito bem, tive que converter a imagem de .svg para .xml. Porém, durante a conversão, o arquivo .xml mantem os tamanhos do arquivo .svg, que eram enormes! Por causa do arquivo gigante, meu projeto não estava sendo compilado. Sempre dando erro por causa de problemas com o heap memory. Então, tive que diminuir o tamanho da imagem .svg usando o software Inkscape (não esquecer também de reduzir o tamanho da página; no Inkscape ir em “Arquivos -> Propriedades do desenho”) e depois a converti para xml usando o site <http://inloop.github.io/svg2android/> (as vezes é necessário marcar a opção “bake transforms into paths”).

# Adicionando SDKs

* 1. Facebook

Para adicionar o SDK do Facebook, bastou adicionar a linha “mavenCentral()” a lista de repositórios dentro do gradle.build do projeto e a linha “compile 'com.facebook.android:facebook-android-sdk:4.14.0'” dentro da lista de dependências dentro do gradle.build do app.

* 1. Google signin

Primeiro, no SDK manager, verificar se o SDK “Google Play Services” está instalado e selecionado. Depois, no gradle do projeto, adicionar o seguinte classpath: “com.google.gms:google-services:3.0.0”. Depois, no gradle do projeto, adicionar a seguinte linha no começo do arquivo “apply plugin: 'com.google.gms.google-services'”. Ainda no gradle do projeto, adicionar a dependência “compile 'com.google.android.gms:play-services-auth:9.2.1'”.

# Certificate fingerprint

Para pegar o certificate fingerprint (para habilitar serviços de login do Facebook e Google), usar o seguinte comando (exemplo para Google):

keytool -exportcert -list -v -alias androiddebugkey -keystore ~/.android/debug.keystore | grep SHA1

Além disso, é preciso também instalar o openssl. No meu pc está instalado em "C:\Program Files (x86)\GnuWin32\bin\openssl.exe"

Existem dois tipos de chave. Uma para desenvolvimento e outra para produção. Para pegar a minha chave de produção, usar o comando: “keytool -exportcert -alias androiddebugkey -keystore C:\Users\Fabio\.android\debug.keystore | "C:\Program Files (x86)\GnuWin32\bin\openssl.exe" sha1 -binary | "C:\Program Files (x86)\GnuWin32\bin\openssl.exe" base64”. A senha é a padrão: “android”. Ou, melhor ainda, usar o seguinte comando: “keytool -exportcert -list -v -alias androiddebugkey -keystore %USERPROFILE%\.android\debug.keystore”

# Login e cadastro social

<https://auth0.com/blog/how-to-authenticate-on-android-using-social-logins/>

# Criando folder e projeto inicial

Para criar um projeto do phonegap, primeiro vá até o folder que irá conter O FOLDER do projeto. Ou seja, o folder do projeto não deve existir ainda. No folder que irá conter o folder do projeto, abra o git bash e use o seguinte comando:

phonegap create aplicativo --id "me.photo2.app" --name "Photo2MeApp" --template hello-world

phonegap create aplicativo --id "me.photo2.app" --name "Photo2MeApp" --template blank

Onde:

* aplicativo – é o nome do folder que irá conter o aplicativo;
* “me.photo2.app” é o id do projeto;
* “Photo2MeApp” é o nome do projeto;
* hello-world – é o template utilizado para gerar o aplicativo inicial.
* O template “blank” é o template em branco

O código que iremos escrever deve estar SEMPRE dentro da pasta www que fica dentro da pasta do projeto.

# Simulando

Para simular o app, basta usar o comando:

phonegap serve

Options:

* --port, -p <n> port for web server (default: 3000)
* --autoreload enable app refresh on file changes (default: true)
* --no-autoreload disable app refresh on file changes
* --localtunnel enable a local tunnel for public access (default: false)

# Building app

Para fazer o build do app pelo phonegap build service, é só usar o seguinte comando na pasta do projeto:

phonegap remote build android

Obs: podemos usar o comando abaixo para que um QRCode seja gerado na command line para baixarmos o aplicativo.

phonegap remote install

# Adicionando plug ins

No phonegap 3, nenhum plug in vem habilitado no projeto inicial. Para adicioná-los, por exemplo, para ter acesso à câmera ou aos arquivos do celular, é necessário fazer a instalação dos plug-ins.

Lista de core plug-ins pode ser encontrado aqui: <https://github.com/ccoenraets/phonegap-day-eu>

Exemplo, para adicionar o plug in battery status ao projeto, basta usar o seguinte comando:

phonegap local plugin add <https://git-wip-us.apache.org/repos/asf/cordova-plugin-battery-status.git>

Observação que recebi ao usar o comando:

[warning] The command `phonegap local <command>` has been DEPRECATED.

[warning] The command has been delegated to `phonegap <command>`.

[warning] The command `phonegap local <command>` will soon be removed.

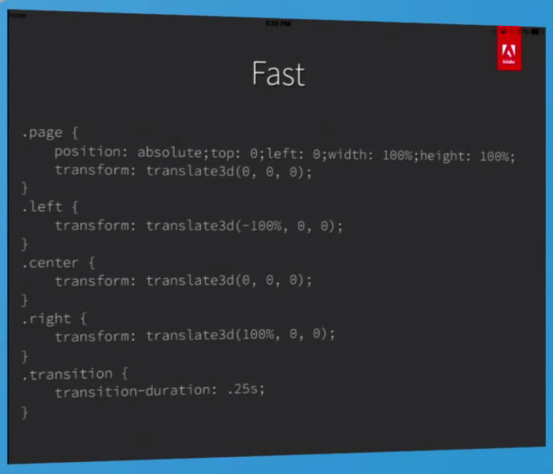
É sempre uma boa idéia adicionar o plug in chamado “device”, que nos dá informações gerais sobre o celular. Para usar logs, temos que usar o “console”. Sempre uma boa idéia ter esses dois instalados.

# Plug ins IMPORTANTES!

* Para celulares, é melhor usar o touch event, pois ele é muito mais rápido e dá a sensação de um aplicativo nativo. O problema é que ao usar o touch event a gente não pode fazer o debug no browser. Para resolver isso, usar github.com/ftlabs/fastclick. Com esse plug in, podemos continuar usando o evento click e, se o evento touch estiver disponível, ele usará o touch, se não, o click.

# UI

* Usar topcoat?
* Usar hardware acceleration para fazer transições. Para fazer as transições, usar CSS. Por exemplo, uma classe é responsável pelo conteúdo que ficará à esquerda, a outra pelo conteúdo que ficará à direita e a última pelo conteúdo que ficará no centro e que será exibido. Exemplo abaixo:



* Usar o atributo CSS “-webkit-touch-callout: none;” para evitar que, no app, quando alguém apertar algum link por um longo tempo, o aplicativo evite de dar a opção de copiar o link.

# Dicas

* USAR CACHE SEMPRE QUE POSSÍVEL!
* Sempre carregar as views primeiro e só depois fazer a chamada para coletar dados no servidor.
* CODEANYWHERE: Como instalar o node.js no codeanywhere
  + sudo apt-get update
  + sudo apt-get install nodejs
  + sudo apt-get install npm

# AngularJS

* 1. Estou usando o mesmo padrão de estrutura pastas e arquivos e construção de views e controllers que é usado no projeto “angular seed” (<https://github.com/angular/angular-seed>).
  2. As routes são definidas dentro dos módulos de cada uma das páginas.
  3. Lembrar de, na página index.html, carregar os módulos logo abaixo o app.css (módulo do app).
  4. Alguns comandos do angular não são executados automaticamente quando dentro de algumas funções especiais do DOM, como, por exemplo, “setTimeout”. Para que a função do angular seja executada, temos que “forçar” a execução delas. Fazemos isso através do comando “**$scope.$apply();**” logo após a função que não foi carregada.
  5. Quando uma variável ou função tiver que ser compartilhada com a view, cria-la da seguinte forma:
     1. $scope.funcao = function(){}
     2. $scope.variavel = true;
  6. Quando a variável ou função não precisar ser compartilhada com a view, cria-la assim:
     1. Var função = function(){}
     2. Var variável = true;
  7. Sempre preferir localizar os módulos. Ou seja, em vez de criar o módulo da seguinte forma: “var app = angular.module(‘app’,[]);”, cria-lo dessa forma: “angular.module(‘app’,[]);”. Dessa forma evitamos que o módulo fique acessível em qualquer lugar do código, evitando a possibilidade de que ele seja sobrescrito.

# Acabando projeto em Phonegap

Irei parar de fazer o projeto em Phonegap para partir para Java e Objective C. Parece que o Phonegap não é capaz de rodar funções no background. Isso por que o Phonegap roda dentro de um tipo de browser no celular e todo o códio escrito em Javascript só é rodado e só funciona quando o app está sendo utilizado. Se for necessário rodar alguma função enquanto o app está no background, não conseguimos fazer isso só através do Javascript e do Phonegap. É necessário criar um background servisse com Java – para Android – para adicionar essa capacidade ao app.

# DESENVOLVENDO APP PARA ANDROID COM JAVA