Tutorial para criação de aplicativo com Cordova/Ionic e AngularJs

Sumário

1 Setup das ferramentas

Instalar o Node js: https://nodejs.org/en/ (utilizei a versão 6.10.3 – estável)

Instalar o npm (Node Package Manager): https://www.npmjs.com/package/npm3 - Você pode instalar a última versão, porém se encontrar problemas na criação de projetos ou adição de módulos, faça um downgrade (Utilizei a versão 3).

Instalar o cordova > Após a instalação do Node, abra o prompt de comando e digite:

```
npm install -g cordova
```

Instalar o Ionic > Ainda no prompt, digite:

```
npm install -g cordova ionic
```

Para desenvolvermos um app para a plataforma Android, será necessário seguir esse tutorial:
https://cordova.apache.org/docs/en/latest/guide/platforms/android/

Mas com a instalação do Node, Cordova e Ionic, já podemos testar nosso app no browser.

Neste tutorial, estou utilizando a IDE Visual Studio Code para o desenvolvimento do projeto (https://code.visualstudio.com/), lembrando que não é necessário o uso de nenhuma IDE para a codificação do app, pode-se alterar e criar os arquivos direto no diretório do projeto, porém uma IDE facilia (muito) nosso trabalho. Ao final do artigo deixarei os links para outras IDEs e ferramentas alternativas para todos os gostos.

2 Desenvolvendo o aplicativo

Agora que estamos com todas as ferramentas 'em mãos', vamos ao trabalho!

2.1 Criando o projeto

Abra o prompt de comando e insira o comando a seguir:

```
ionic start NOME_DO_PROJETO sidemenu --type ionic1 --appname "NOME_DO_APP"
```

- ionic start: cria o projeto – diretório e todas as dependências
- sidemenu: template do ionic para um app com menu lateral (existem os templates "tabs" e "blank")
- type: tipo do projeto – estamos utilizando a versão 1 do ionic (discutiremos mais adiante os motivos)
- appname: nome "amigável" do app

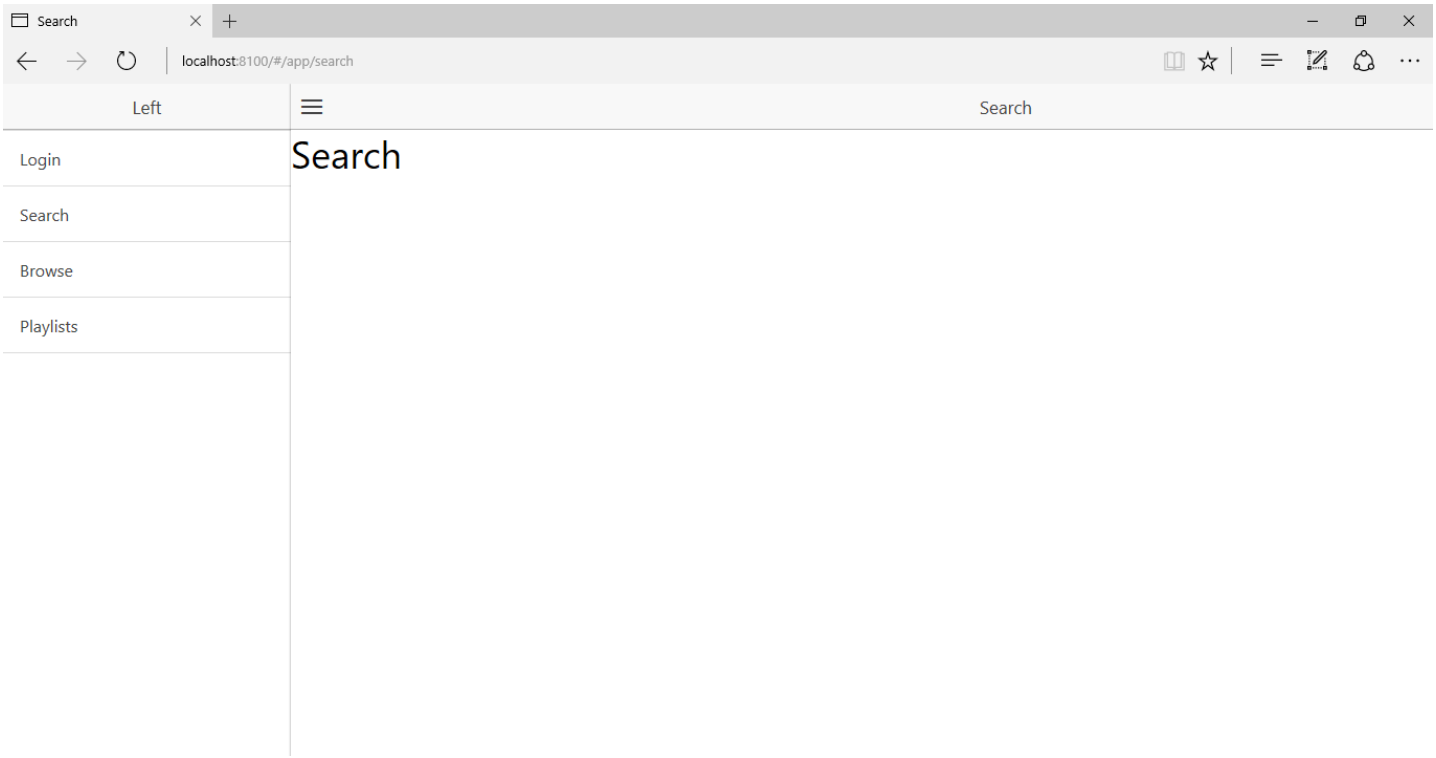
Agora, vamos entrar no diretório do nosso app batizado de "Fake.News", com o comando:

```
cd .\Fake.News
```

Após entrar na pasta do app, vamos emular o template no browser para verificar que tudo está ok, com o comando:

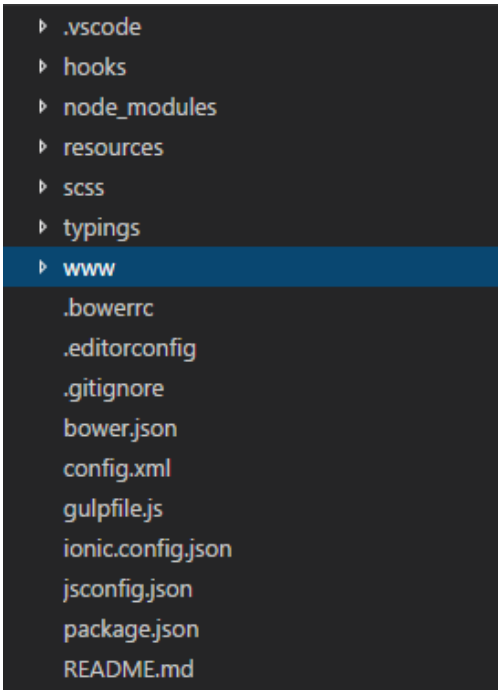
```
ionic serve
```

Nosso app aparecerá assim no browser:



2.2 A estrutura do projeto

Após criado o nosso app, vamos conhecer a estrutura do projeto montada pelo Ionic. Você pode navegar pelo explorer ou abrir o diretório na sua IDE de preferência. No Visual Studio Code, a estrutura é apresentada conforme a imagem a seguir:



O foco deste artigo está no desenvolvimento do app com tecnologias web, portanto não serão discutidos com profundidade os diversos arquivos de configuração e diretórios criados pela estrutura do Ionic.
Algumas considerações importantes sobre a estrutura:

- O diretório 'node_modules' contém todas as dependências utilizadas pelo projeto, como as bibliotecas para operações com arrays, arquivos, strings, entre outros. Este é o nosso ".NET Framework" ☺, todas as bibliotecas externas necessárias para as mais diversas funcionalidades estão ai.
- No diretório 'scss' estão os arquivos utilizados pelo pré-processador CSS, o SASS. Não vamos utilizar esses arquivos neste tutorial, vamos focar nas tecnologias web mais 'tradicionais'
- A pasta 'resources' contém os ícones e splash screens para cada plataforma
- O arquivo 'gulpfile.js' contém a tarefas a serem executadas no processo de build da nossa aplicação, e a ordem em que devem ser executadas, como o processo de 'compilação' dos arquivos scss, por exemplo.
- No arquivo 'config.xml' estão as demais configurações do app para as diferentes plataformas e informações sobre o nosso app
- Por fim, o diretório 'www' contém o nosso web app. Segue a estrutura de uma aplicação SPA, com os arquivos html, Javascript e CSS. Basicamente um web site, no qual vamos concentrar nosso desenvolvimento.

2.3 Botando a mão na massa!

Antes de iniciar o desenvolvimento, sugiro que navegue pelo projeto, para conhecer sua estrutura, verifique os arquivos contidos em 'www' para se familiarizar com a estrutura. O projeto foi criado com a versão 1.X.X do Angular JS, neste caso a versão 1.5.4. Note que não é obrigatório utilizar o framework Angular, você pode utilizar javascript 'puro' ou jQuery, por exemplo (lembrando que o Ionic trabalha com todo o seu potencial utilizando o Angular). Utilize as feramentas que te deixem confortável para o desenvolvimento.

2.3.1 Criando o módulo principal

Para quem está familiarizado com o Angular JS, temos um módulo principal para nosso aplicativo, no qual serão adicionadas nossas controllers, services, etc. Não vamos criar um módulo, pois o projeto já criou um, vamos apenas alterá-lo. Abra o arquivo www/js/app.js e faça as seguintes alterações:

Altere o nome do módulo para o nome do seu app, e deixe somente a dependência do Ionic:

```
var app = angular.module('fakeNews', ['ionic']);
```

Remova todo o código da seção ‘config’, e adicione o seguinte trecho:

```
app.config(function($stateProvider, $urlRouterProvider) {
  $stateProvider
  .state('dashboard', {
    url: '/dashboard',
    views: {
      'menuContent': {
        templateUrl: 'templates/dashboard.html'
      }
    }
  });
  // if none of the above states are matched, use this as the fallback
  $urlRouterProvider.otherwise('/dashboard');
});
```

Nosso módulo completo fica assim:

```
var app = angular.module('fakeNews', ['ionic']);

app.run(function($ionicPlatform) {
  $ionicPlatform.ready(function() {

    if (window.cordova && window.cordova.plugins.Keyboard) {
      cordova.plugins.Keyboard.hideKeyboardAccessoryBar(true);
      cordova.plugins.Keyboard.disableScroll(true);

    }
    if (window.StatusBar) {
      // org.apache.cordova.statusbar required
      StatusBar.styleDefault();
    }
  });
});

app.config(function($stateProvider, $urlRouterProvider) {
  $stateProvider
  .state('dashboard', {
    url: '/dashboard',
    views: {
      'menuContent': {
        templateUrl: 'templates/dashboard.html'
      }
    }
  });
  // if none of the above states are matched, use this as the fallback
  $urlRouterProvider.otherwise('/dashboard');
});
```

Até agora, criamos nosso módulo geral da aplicação, o ‘fakeNews’ e configuramos a rota para uma página dashboard.

Vamos criar nossa página dashboard e sua respectiva controller.

2.3.2 Criando uma página e sua controller

No diretório www/js, exclua o arquivo ‘controllers.js’ gerado pelo template. Particularmente gosto de separar uma controller para cada arquivo, para melhor organização de código. Dentro de www/js, crie um diretório ‘controllers’, e em seguida adicione o arquivo ‘dashboardController.js’. Essa será nossa primeira controller Angular. Dentro do arquivo criado no passo anterior, digite o seguinte trecho de código:

```
var app = angular.module('fakeNews'); //adicionar controller ao módulo já existente

app.controller('DashboardController', DashboardController) //nome da controller

function DashboardController(){ //código  da controller
    this.description = 'Esta é a dashboard'; //variável para a descrição da página (apenas para teste)
}
```

No diretório www/templates, adicione o arquivo ‘dashboard.html’, e insira o seguinte código:

```
<ion-view view-title="Dashboard" ng-controller="DashboardController as dashboardCtrl">
  <ion-content>
    <h1>{{dashboardCtrl.description}}</h1>
    <p>Dashboard de DSN-News</p>
  </ion-content>
</ion-view>
```

Agora, vamos até a página www/templates/menu.html. Vamos remover o menu do template, e adicioná-lo diretamente na página ‘index.html’, no diretório www. Como é um template comum e que sempre deverá ser renderizado na tela, fica mais prático deixá-lo na index, para facilitar nossa configuração de rotas que não carrega mais um template padrão para o menu e um secundário com as views. Não se preocupe com o código que ficou do template do menu. Essa é a nossa index (fragmento <body>):

```
<body ng-app="fakeNews">

    <ion-side-menus enable-menu-with-back-views="false">
      <ion-side-menu-content>
        <ion-nav-bar class="bar-stable">
          <ion-nav-back-button>
          </ion-nav-back-button>

          <ion-nav-buttons side="left">
            <button class="button button-icon button-clear ion-navicon" menu-toggle="left">
            </button>
          </ion-nav-buttons>
        </ion-nav-bar>
        <ion-nav-view name="menuContent"></ion-nav-view>
      </ion-side-menu-content>

      <ion-side-menu side="left">
        <ion-header-bar class="bar-stable">
```

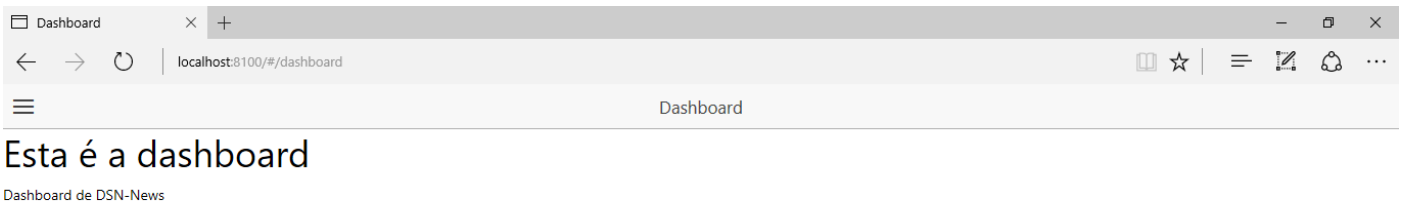
```
        <h1 class="title">Left</h1>
      </ion-header-bar>
      <ion-content>
        <ion-list>
          <ion-item menu-close ng-click="login()">
            Login
          </ion-item>
          <ion-item menu-close href="#/search">
            Search
          </ion-item>
          <ion-item menu-close href="#/browse">
            Browse
          </ion-item>
          <ion-item menu-close href="#/playlists">
            Playlists
          </ion-item>
        </ion-list>
      </ion-content>
    </ion-side-menu>
  </ion-side-menus>

  <ion-nav-view></ion-nav-view>

</body>
```

Na tag body, lembre-se de alterar a diretiva ng-app para o módulo que você criou. No nosso caso, o módulo é 'fakeNews'. A tag <ion-nav-view> é a responsável por renderizar os templates de acordo com as rotas.

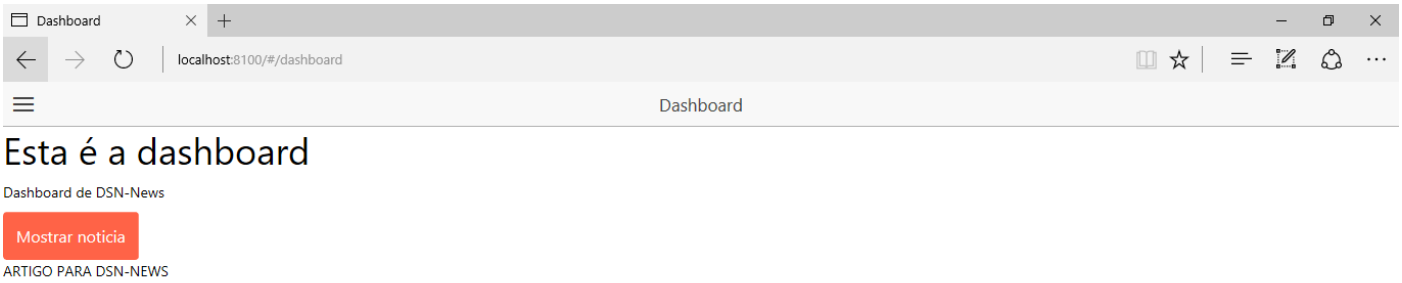
Temos então nossa primeira página da aplicação. Vamos recapitular o que fizemos até agora:

- Criamos nosso módulo 'fakeNews'
- Configuramos uma rota para a tela 'dashboard'
- Criamos a controller para a tela dashboard
- Criamos a tela dashboard
- Passamos o menu para a index, visando facilitar o trabalho de roteamento

Após todas essas ações realizadas, abra o prompt, vá até o diretório do seu projeto e digite o comando para visualizarmos nossa aplicação no browser (ionic serve). A este ponto, deverá se parecer com essa tela:



Agora, 'gaste' um tempo com sua criatividade, antes de prosseguirmos no tutorial, adicione alguns elementos na tela, como um botão que mostra um texto, adicione variáveis na sua controller para serem exibidas na dashboard:



Ainda não exclua os demais arquivos, utilize-os para treinar ou ter idéias.

Agora, vamos incrementar nosso app. Para isso, vamos utilizar os componentes de interface do Ionic, e alterá-los de acordo com nossa necessidade. Acesse o link abaixo para adicionarmos o componente 'card' à nossa dashboard: http://ionicframework.com/docs/v1/components/#content

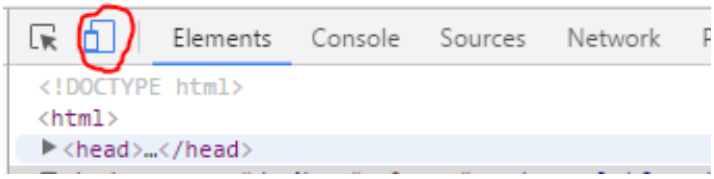
Vamos montar um acesso rápido para as publicações, a conta de usuário e artigos favoritos. Note que alterei o componente 'card' do Ionic para adicioná-lo ao nosso app. Nosso 'card' em dashboard.html fica assim:

```
<div class="list card">
      <a href="#/publications">
        <div class="item item-avatar item-icon-left">
          <i class="icon ion-document-text"></i>
          <h2>Publicações</h2>
        </div>
        <div class="card-cover">
          <img class="full-image" src="img/editions-cover.jpg">
        </div>
      </a>
    </div>
```

A classe 'card-cover' é um estilo interno do nosso app, para ajuste do container da imagem. No arquivo 'style.css', adicione as seguintes classes:

```
.card > a{
    text-decoration: none !important;
}

.card-cover{
    margin-bottom: -5%;
    height: 200px;
    line-height: 100px;
}

.card-cover > img{
    height: 100%;
}
```

Na dashboard, adicionei três 'cards' para os atalhos citados previamente. Lembre-se de pegar algumas imagens avulsas só pra decorar nosso app. Ao executar o aplicativo, abra-o no Google Chrome dessa vez, pressione F12, e clique no botão abaixo, para simular uma tela de celular:



Dentro da dashboard, adicionei um cabeçalho com um subtítulo, só porque achei bonito. Caso queira utilizá-lo também, adicione abaixo de 'ion-content' o trecho abaixo:

```
<ion-view view-title="Dashboard" ng-controller="DashboardController as dashboardCtrl">
  <ion-content>
    <div class="bar bar-header bar-calm">
        <h1 class="title">Dashboard de DSN-News</h1>
    </div>
```

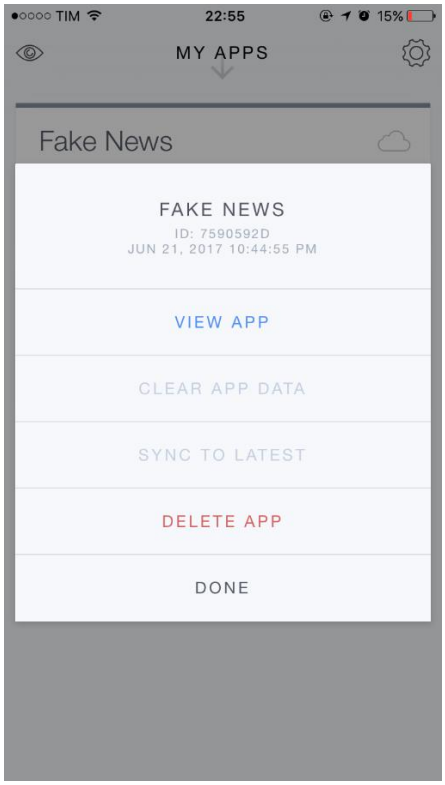
O atributo 'view-title' contém o texto que será exibido na barra de menu. Agora, a dashboard está assim:

2.3.3 Testando o app com o Ionic View

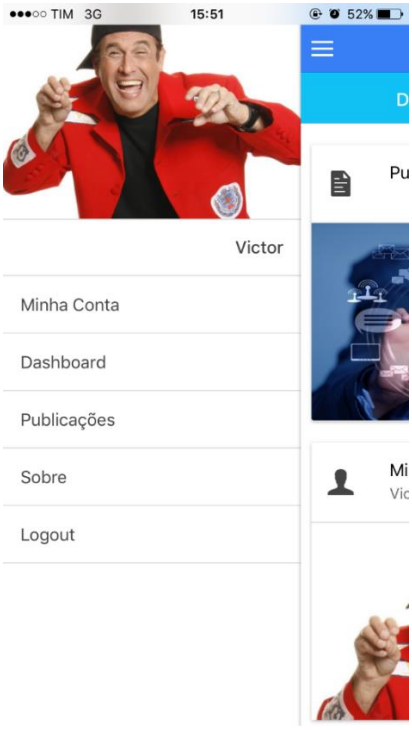
Para testar o app no seu celular, basta baixar o app Ionic View, criar sua conta para o serviço no site do Ionic, e subir seu aplicativo para a nuvem, com os comandos a seguir:

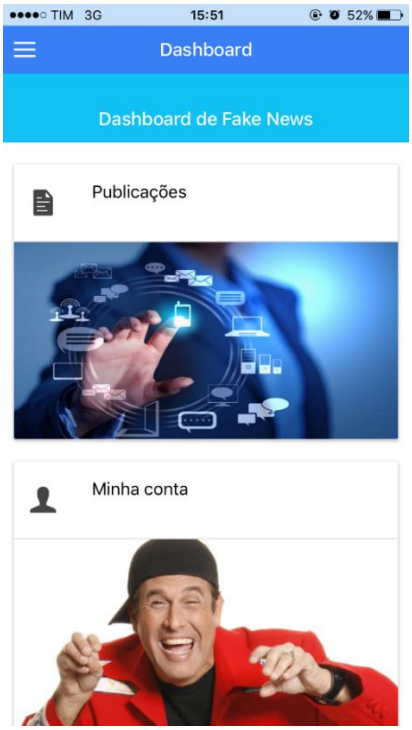
`ionic login` (depois insira usuário e senha)

`ionic upload`



Após subir seu projeto, abra o Ionic View no seu smartphone, clique em View App, e seu aplicativo será iniciado. As telas abaixo são prints do meu celular:





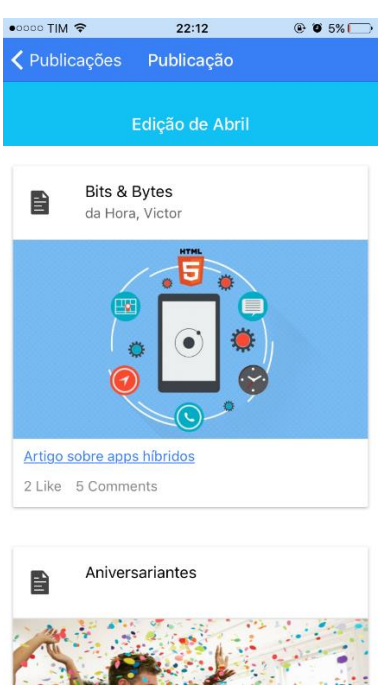
Para acessar o sistema, deixei como obrigatório realizar o login. Não fiz validações, portanto, caso queira, pode clicar em entrar e deixar as informações sem preencher. Ao iniciar o app, a Dashboard está redirecionando para o login. Ao fazer um logout, novamente o login é exibido, e você será redirecionado para a última tela.

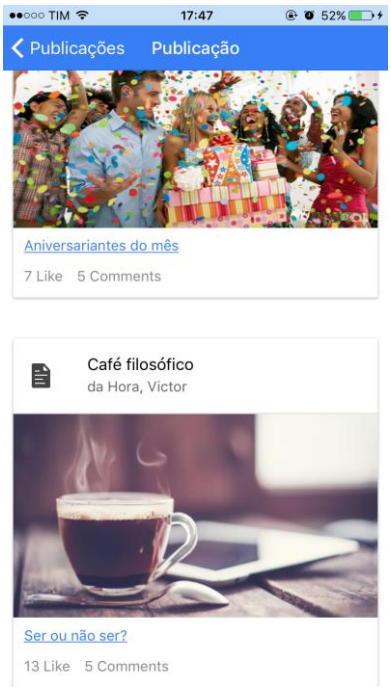
Agora, vamos adicionar mais telas ao nosso app. Para não estender muito esse tutorial, vou somente mostrar as telas e dizer qual o seu funcionamento. Caso queira olhar o código, acompanhe o projeto no ZIP anexo. Se parar por aqui, já tem sua dashboard, e pode abusar de sua criatividade para adicionar funcionalidades ao seu app.

2.3.4 Adicionando mais funcionalidades ao app

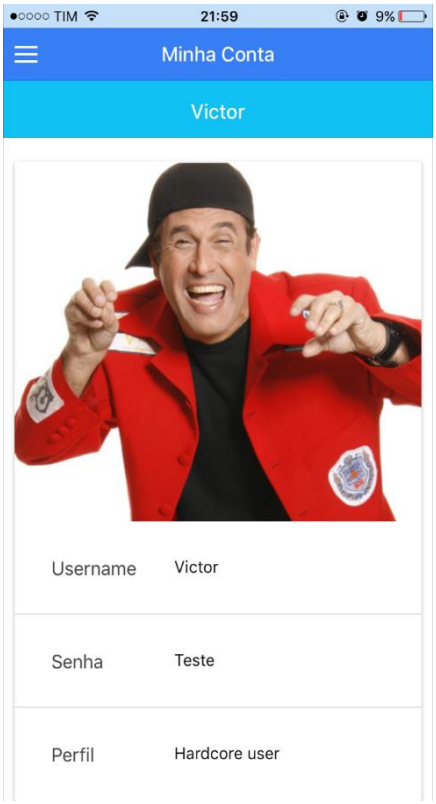
Seguindo com as telas, vamos adicionar uma tela de publicações, sua controller e configurar a rota para a mesma (não se esqueça de adicionar o arquivo da controller na index). A tela de publicações contém uma lista com as edições publicadas.

Ao clicar em um dos itens, a tela da publicação específica é aberta. Os artigos são exibidos nos 'cards' do Ionic. Ao clicar em um dos artigos, uma tela com um artigo do Reddit é exibido através da API pública NewsAPI:

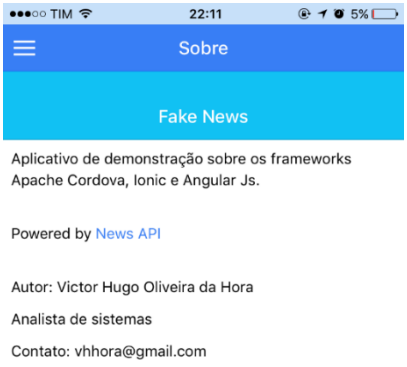




Além dessas telas, criei também uma tela de perfil do usuário, com as informações inseridas na tela de login.



No atalho para favoritos, apenas redirecionei para a tela de publicações do mês de abril.
Por fim, há uma tela 'Sobre', com informações gerais sobre o app:



2.3.5 Factories e Services

Na tela que contém o artigo a ser exibido, criei uma factory para consumir uma API Rest de notícias do Reddit. Não entrarei na discussão entre factory e services, basicamente, nossa 'newsFactory' retorna um objeto que contém o resultado da chamada na API.

```
var app = angular.module('fakeNews');
app.factory('newsFactory', ['$http', newsFactory]);
```

```
function newsFactory($http) {
    return {
        getNews: function(source) {
           return $http.get('https://newsapi.org/v1/articles?source='+ source
+'&sortBy=top&apiKey=fa3a83bd70c54b618119be309e979237');
        }
    };
}
```

A chamada na controller da página de artigos é feita assim:

```
    var self = this;

    this.loadData = function(){
        loadingService.showLoading();

        var response = newsFactory.getNews('reddit-r-all').then(function(response){
            self.article = response.data.articles[id];

            loadingService.hideLoading();
        }, function(response){
            toastr.error('Erro ao acessar o artigo');
            loadingService.hideLoading();
        });
    }
```

Note que nesse caso atribui 'this' a uma variável, pois é a própria controller, e não pertence ao escopo da function dentro do retorno da factory. Ao final deixarei links sobre 'this' e '$scope', e também sobre factories e services.

Para que a página não fique em branco ao ser carregada, criei um serviço que utiliza a dependência do $ionicLoading, para mostrar uma tela de loading. Este serviço não retorna nada, apenas realiza uma ação, conforme o código abaixo:

```
var app = angular.module('fakeNews');

app.service('loadingService', ['$ionicLoading', loadingService]);

function loadingService($ionicLoading) {
    this.showLoading = function(){
        $ionicLoading.show({
        content: 'Loading',
        animation: 'fade-in',
        showBackdrop: true,
        maxWidth: 200,
        showDelay: 0
      });
    };

    this.hideLoading = function(){
        $ionicLoading.hide();
    }
}
```

Na controller, o serviço é utilizado para exibir o loading enquanto a factory não retorna o resultado:

```
    this.loadData = function(){
        loadingService.showLoading();
```

```
        var response = newsFactory.getNews('reddit-r-all').then(function(response){
            self.article = response.data.articles[id];
            loadingService.hideLoading();
```

Utilizei também o serviço de loading no módulo principal, atribuindo ao $rootScope os eventos de carregamento das páginas. Caso uma página demore mais do que o esperado para carregar, o loading aparecerá.

Algumas referências:

Para as notificações, utilizei o componente Angular toastr: https://github.com/Foxandxss/angular-toastr

A API para os artigos pode ser encontrada em: https://newsapi.org/
* Não se esqueça de adicionar a referência para o site de NewsAPI, de acordo com a licença da mesma.

This e $scope: https://johnpapa.net/do-you-like-your-angular-controllers-with-or-without-sugar/

Factory e service: https://blog.thoughtram.io/angular/2015/07/07/service-vs-factory-once-and-for-all.html

2.4 Considerações finais

E aqui termina nosso tutorial. Sinta-se livre para utilizar o template publicado, alterando seus componentes e funcionalidades. Obrigado pela atenção, caso tenha alguma dúvida, sugestão ou reclamação sobre esse tutorial, entre em contato comigo, seu feedback é muito importante.

Um grande abraço,

Victor Hugo Oliveira da Hora
Analista de sistemas
vhhora@gmail.com