sãojudas universidade	FTCE Faculdade de Tecnologia e Ciências Exatas

USJT - 2019 – Programação para Web e Dispositivos Móveis

Professores: Anderson e Bossini

Aula: 09
Assunto: Ionic – CRUD em memória volátil

Neste material iremos construir uma aplicação que permite ao usuário fazer o registro de notas de seu interesse.

Passo 1 (Criando uma pasta de trabalho) Crie uma pasta para abrigar seus projetos Ionic.

Passo 2 (Usando o Visual Studio Code (VSC)) Iremos utilizar o Visual Studio Code (VSC) para criar projetos Ionic. Qualquer outro editor de texto poderia ser utilizado.

2.1 Abra um terminal e navegue até a pasta criada. Digite o comando da Listagem 2.1.1. Isso irá abrir o VSC com a pasta vinculada. Assim ela será tratada como um projeto.

Listagem 2.1.1

```
code .
```

2.2 A seguir, no VSC, clique em View >> Terminal para que ele mostre um terminal integrado, o que irá facilitar a digitação dos comandos para criação do projeto Ionic e seus componentes.

Passo 3 (Criando um novo projeto Ionic) Projetos e componentes Ionic são criados, em geral, por meio da linha de comando.

3.1 Para criar um projeto chamado **NotableNotes**, vamos usar o comando da Listagem 3.1.1.

Nota: Ao longo da criação do projeto, o Ionic irá perguntar se deseja instalar alguns itens. Diga que não quer instalar nenhum deles.

Listagem 3.1.1

```
ionic start NotableNotes blank
```

Note que o nome do projeto é NotableNotes e que ele usa um template (coleção de arquivos com código pronto, para facilitar o início do projeto) chamado **blank**. O template blank inclui pouquíssimo código inicial. O Ionic inclui outros templates, como mostra a Tabela 3.1.1.

Tabela 3.1.1

Template	Funcionalidade
tabs	Aplicação que inclui navegação por abas.
sidemenu	Aplicação que inclui um menu lateral.
blank	Aplicação muito simples, com uma única página.
super	Projeto com 14 páginas de exemplo.
tutorial	Projeto que inclui uma navegação inicial, tipo tutorial para explicar como uma aplicação funciona.

Passo 4 (Testando a aplicação pela primeira vez) Note que o Ionic criou uma pasta com o nome do projeto dentro da sua pasta de trabalho.

4.1 Use o terminal interno do VSC para navegar até essa pasta usando o comando da Listagem 4.1.1.

Listagem 4.1.1

```
cd NotableNotes
```

4.2 No Terminal, digite o comando da Listagem 4.2.1 para iniciar um servidor que irá simular o funcionamento da aplicação. Note que seu navegador padrão deverá ser aberto e exibir uma amostra da aplicação em diferentes plataformas. O servidor se encarrega também de atualizar o projeto sempre que um arquivo for atualizado.

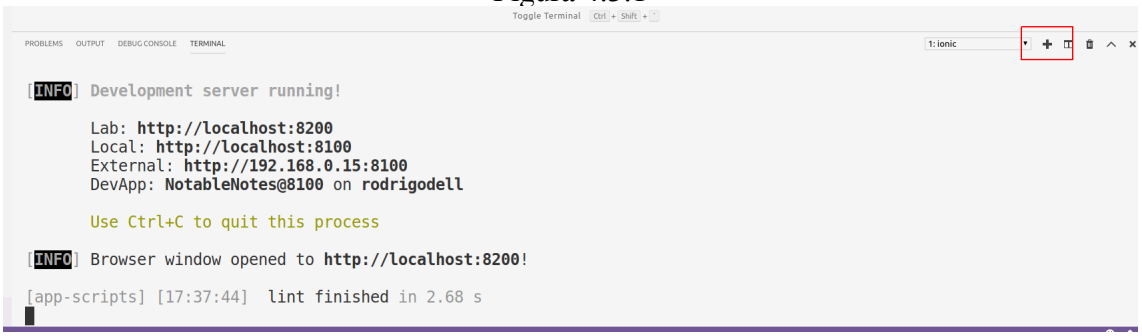
Nota: De vez em quando será necessário reiniciar esse servidor, pois ele poderá falhar quando tentar detectar novas alterações em arquivos.

Listagem 4.2.1

```
ionic serve --lab
```

4.3 Note que o terminal do VSC ficou preso executando o processo do servidor criado pelo Ionic. O VSC permite que múltiplos terminais fiquem abertos, o que facilita o desenvolvimento. Clique no símbolo **+** como mostra a Figura 4.3.1. Assim o primeiro terminal ficará dedicado para o servidor enquanto o segundo ficará livre para executarmos comandos do Ionic.

Figura 4.3.1



Passo 5 (Inspecionando (alguns) arquivos do projeto) Um projeto Ionic é composto por diversos componentes. Vejamos alguns.

5.1 Uma pasta importante é chamada **src**. Expanda-a e veja que há uma subpasta chamada **pages**. Expanda-a também e veja que há uma subpasta chamada **home**. Expandindo essa pasta você encontrará três arquivos:

- **home.ts** (ts é abreviação de typescript. esse arquivo especifica a parte dinâmica do funcionamento desse componente)
- **home.html** (uma página html, é o template da página, utilizado em home.ts)
- **home.scss** (para decorar a página)

5.2 Abra o arquivo home.html e altere o título da página. Use o texto “Notable Notes”. Salve o arquivo e verifique o resultado no navegador.

5.3 Uma outra pasta importante se chama **app**. Ela possui arquivos que definem o componente principal da aplicação:

- app.component.ts (Define o componente raiz, inicial da aplicação. Observe que ele define a página HomePage como a página inicial. Seu construtor desempenha atividades necessárias assim que a aplicação fica pronta).

- app.html (Esse arquivo é especificado na propriedade templateUrl do anterior. Veja que ele define HomePage como a página inicial, usando uma tag ion-nav, análoga ao funcionamento do Angular router.)

- app.module.ts (Neste arquivos declaramos diversos componentes, como componentes para os quais a aplicação pode navegar, componentes provedores de conteúdo, as páginas etc.

Passo 6 (Definindo algumas notas fixas no código) Para começar o desenvolvimento, iremos definir algumas notas fixas no código, representadas como um vetor JSON.

6.1 Abra o arquivo home.ts e adicione o vetor JSON da Listagem 6.1.1 diretamente ao corpo (fora de qualquer método e fora do construtor) da classe.

Listagem 6.1.1

```
notes = [
   {
    id: '1',
    date: '2016-02-01',
    title: 'Firebase',
    content: 'Que tal programar Serverless?'
   },
   {
    id: '2',
    date: '2016-01-01',
    title: 'Ionic',
    content: 'Aprenda o básico de Ionic'
   },
   {
    id: '3',
    date: '2016-03-01',
    title: 'Angular',
    content: 'Importante para desenvolver com Ionic'
   }
 ]
```

Passo 7 (Exibindo as notas em cartões) Iremos utilizar um cartão (ion-card) para exibir cada nota. Veja a documentação deles no Link 7.1.

Link 7.1
https://ionicframework.com/docs/api/card

7.1 Um card pode ter algumas seções, como header, title e content. Para gerar uma coleção deles, usamos uma diretiva ngFor do Angular. Veja a Listagem 7.1.1. A tag **ion-card** deve ser filha direta de **ion-content** no arquivo home.html. Antes de mais nada, apague o conteúdo da tag ion-content.

Listagem 7.1.1

```
<ion-content padding>
 <ion-card *ngFor="let note of notes">
  <ion-card-header>
   {{note.date}}
  </ion-card-header>
  <ion-card-content>
   <ion-card-title>
    {{note.title}}
   </ion-card-title>
   <p>
    {{note.content}}
   </p>
  </ion-card-content>
 </ion-card>
</ion-content>
```

Nota: O operador {{}} chama-se operador de **interpolação**. Ele permite avaliar uma expressão a fim de obter-se seu resultado.

Nota: É fundamental usar os componentes prontos (aqueles cuja tag começa com "ion") do Ionic sempre que possível. Cada componente desse se renderiza adequadamente conforme a plataforma em que está. Por exemplo, a exibição de um card no Android pode ser diferente da exibição de um card no IOS. Usando a tag ion-card, escrevemos código uma única vez que se adapta à plataforma subjacente.

7.2 Use o operador pipe (|) do Angular para formatar a data como desejar, como mostra a Listagem 7.2.1.

Listagem 7.2.1

```
{{note.date | date: 'dd/MM/yyyy'}}
```

7.3 Salve todas as alterações e verifique seu navegador. Neste instante os cards já devem ser exibidos. Caso tenha algum problema, para o processo do servidor e execute-o novamente.

Passo 8 (Tornando os itens clicáveis) A fim de tornar um item clicável, associamos a ele um método, por meio de um "event binding" do Angular. O método pode especificar um parâmetro que pode ser utilizado para receber o item clicado.

8.1 A Listagem 8.1.1 mostra o ajuste necessário no arquivo home.html.

Listagem 8.1.1

```
<ion-card *ngFor="let note of notes" (click)="onItemClick(note)">
```

8.2 A Listagem 8.2.1 mostra a definição do método onItemClick, que deve ser feita na classe HomePage, definida no arquivo home.ts.

Listagem 8.2.1

```
onItemClick (note){
  console.log("item-click", note)
}
```

8.3 No momento estamos apenas exibindo o objeto recebido no console do navegador. Para ver o log, abra o console (no Chrome aperte CTRL + SHIFT + I e clique em console).

Nota: No console do navegador você deverá ver uma mensagem como "**Cordova is not available"**. Isso é esperado pois estamos fazendo os testes no navegador, plataforma em que o Cordova não está disponível. O Cordova é responsável por, entre outras coisas, empacotar a aplicação web como uma aplicação nativa, além de fornecer diferentes plugins nativos. Ele só estará disponível quando executarmos a aplicação em um dispositivo.

Passo 9 (Criando uma nova página) Criar uma nova página significa criar um novo componente Angular. Podemos realizar o procedimento manualmente e, para manter o padrão, criar os arquivos .ts, .html e .scss como a página Home já possui. Porém, o Ionic possui um comando que facilita essa tarefa.

9.1 No terminal aberto no VSC, use o comando da Listagem 9.1.1 para criar uma nova página. Note que a letra **g** significa **generate**. Como a página servirá para exibir os detalhes de uma nota, o seu nome será **detail**. Note que nossa convenção para nomes de páginas é especificar o nome da página e colocar "Page" como sufixo. Note também que essa convenção já está implícita no comando de geração de páginas.

Não se esqueça de que seu terminal deve estar no diretório **NotableNotes**.

Listagem 9.1.1

```
ionic g page detail
```

9.2 Inspecione a pasta chamada pages/detail para verificar o resultado. Quatro arquivos foram criados:

- detail.html (análogo ao home.html)
- detail.ts (análogo ao home.ts)
- detail.scss (análogo ao home.scss)
- detail.module.ts (A existência desse arquivo permite que a página seja carregada de modo "lazy", ou seja, somente quando requisitada. Isso é importante para aplicações que possuam muitos componentes. Por exemplo, uma aplicação com 50 páginas poderia carregá-las todas de uma vez logo na inicialização, causando possivelmente uma má experiência para o usuário. E possivelmente nem todas serão utilizadas. Fazer o carregamento lazy permite que uma página seja carregada sob demanda, somente quando necessário).

Passo 10 (Navegando de Home para Detail) A fim de navegar da página home para a página de detalhes uma vez que uma nota for clicada, precisamos utilizar o objeto **NavController** que foi injetado pelo Ionic no construtor da classe HomePage.

10.1 A Listagem 10.1.1 mostra a nova definição do método onItemClick. O método utilizado se chama push pois o Ionic mantém uma pilha de páginas. É natural que a estrutura de dados usada seja uma pilha pois trata-se de uma estrutura de dados conveniente para a semântica do botão voltar de uma aplicação: a última página aberta é a primeira a ser fechada.

Listagem 10.1.1

```
onItemClick (note){
  //console.log("item-click", note)
  this.navCtrl.push('DetailPage')
 }
```

10.2 Verifique no navegador se já é possível navegar para a página de detalhes clicando em um item da lista. Caso encontre algum erro, esse pode ser um bom momento para para o servidor, fechar as abas do Chrome e iniciar novamente.

Passo 11 (Passando a nota selecionada para a página de detalhes) No momento a página de detalhes não mostra coisa alguma. A página Home precisa enviar a nota selecionada para ela. Isso pode ser feito por meio do método push: a nota pode ser enviada como um objeto JSON no segundo argumento do push.

11.1 A Listagem 11.1.1 mostra a nota selecionada sendo enviada para a página de detalhes. É importante que ela seja enviada como um JSON pois assim terá um rótulo associado que a página de detalhes poderá utiliza para acessá-la.

Listagem 11.1.1

```
onItemClick(note) {
   //console.log("item-click", note);
   this.navCtrl.push('DetailPage', {
    noteParam: note
   });
 }
```

11.2 Por sua vez, a página de detalhes irá fazer o seguinte:

- declarar uma variável para guardar a nota recebida
- obter a nota recebida como parâmetro usando um objeto do tipo **NavParams**
- atribuir a nota recebida à variável declarada
- mostrar a nota recebida no console
- mostrar o título da nota recebida no título da página de detalhes

A Listagem 11.2.1 mostra a implementação do construtor da classe DetailPage.

Listagem 11.2.1

```
note;
 constructor(public navCtrl: NavController, public navParams: NavParams) {
   this.note = this.navParams.get("noteParam");
   console.log("nav-param", this.note);
 }
```

A Listagem 11.2.2 mostra a alteração realizada na página detail.html.

Listagem 11.2.2

```
<ion-header>
 <ion-navbar>
  <ion-title>{{note.title}}</ion-title>
 </ion-navbar>
</ion-header>
<ion-content padding>

</ion-content>
```

Execute novamente a aplicação e verifique se a nota foi exibida no console do navegador. Repare no título da página de detalhes uma vez que você navegue até ela.

Passo 12(Editando uma nota) A página de detalhes permitirá a edição dos dados da nota recebida.

12.1 O primeiro passo é criar uma lista que irá conter um item para cada parte de uma nota. Veja a Listagem 12.1.1. Ela exibe o conteúdo da página detail.html.

Listagem 12.1.1

```
<ion-content padding>
 <ion-list>
  <ion-item>
   <ion-label fixed>Title</ion-label>
   <ion-input type="text" value=""></ion-input>
  </ion-item>
  <ion-item>
   <ion-label fixed>Date</ion-label>
   <ion-input type="text"></ion-input>
  </ion-item>
  <ion-item>
   <ion-textarea placeholder="Content"></ion-textarea>
  </ion-item>
 </ion-list>
</ion-content>
```

12.2 A fim de aumentar o campo para digitação do conteúdo da nota (o textarea) vamos editar o arquivo detail.scss. Note que trata-se de uma edição simples em css. A Listagem 12.2.1 mostra o código.

Listagem 12.2.1

```
page-detail {
  textarea{
    height:80vh;
  }
}
```

Passo 13 (Vinculando os campos do form a campos da nota) É preciso especificar a quais variáveis os campos do form estão vinculados. Assim, conforme o usuário faz atualizações visuais, automaticamente o Ionic (por meio do **two way data binding** do Angular) irá atualizá-las.

13.1 O vínculo entre campos do form e campos da nota será feito usando a diretiva **[(ngModel)]**, como mostra a Listagem 13.1.1.

Listagem 13.1.1

```
<ion-content padding>
 <ion-list>
  <ion-item>
   <ion-label fixed>Title</ion-label>
   <ion-input type="text" [(ngModel)]="note.title"></ion-input>
  </ion-item>
  <ion-item>
   <ion-label fixed>Date</ion-label>
   <ion-input type="text" [(ngModel)]="note.date"></ion-input>
  </ion-item>
  <ion-item>
   <ion-textarea placeholder="Content"
[(ngModel)]="note.content"></ion-textarea>
  </ion-item>
 </ion-list>
</ion-content>
```

Passo 14 (Usando um componente apropriado para seleção de datas) Obter datas do usuário por meio de um campo textual é inconveniente, já que ele pode digitar valores quaisquer e em qualquer formato. O ideal é utilizar um componente que restrinja o que o usuário irá digitar e que, ao mesmo tempo, torne a aplicação mais amigável.

14.1 O componente DateTime do Ionic resolve esse problema. Veja seu uso na Listagem 14.1.1.

Listagem 14.1.1

```
<ion-content padding>
 <ion-list>
  <ion-item>
   <ion-label fixed>Title</ion-label>
   <ion-input type="text" [(ngModel)]="note.title"></ion-input>
  </ion-item>
  <ion-item>
   <ion-label fixed>Date</ion-label>
   <ion-datetime type="text" displayFormat="DD/MM/YYYY"
[(ngModel)]="note.date"></ion-datetime>
  </ion-item>
  <ion-item>
   <ion-textarea placeholder="Content" [(ngModel)]="note.content"></ion-textarea>
  </ion-item>
 </ion-list>
</ion-content>
```

Veja o Link 14.1.1 para saber mais sobre o componente DateTime.

Link 14.1.1
https://ionicframework.com/docs/v3/api/components/datetime/DateTime/

Passo 15 (Especificando um serviço para lidar com os dados) No momento os dados de notas estão fixos na classe que representa a tela inicial da aplicação, o que evidentemente não é uma boa ideia. Iremos definir um serviço responsável por dar acesso a eles o que os tornará mais facilmente reutilizáveis e acessíveis. Para utilizar o serviço em outras classes, bastará pedir ao Ionic que faça a injeção de dependência.

15.1 Um serviço no Angular é simplesmente uma classe. Para criar o serviço para lidar com dados de notas, crie um arquivo chamado **note.service.ts** na pasta app da aplicação.

15.2 A classe simplesmente define o vetor JSON com as notas. Veja sua definição na Listagem 15.2.1. Ela deve ser definida no arquivo recém-criado, o note.service.ts. Os dados são somente uma cópia daqueles que já temos na classe home.ts.

Listagem 15.2.1

```
export class NoteService{
  notes = [
    {
      id: '1',
      date: '2016-02-01',
      title: 'Firebase',
      content: 'Que tal programar Serverless?'
    },
    {
      id: '2',
      date: '2016-01-01',
      title: 'Ionic',
      content: 'Aprenda o básico de Ionic'
    },
    {
      id: '3',
      date: '2016-03-01',
      title: 'Angular',
      content: 'Importante para desenvolver com Ionic'
    }
  ]
}
```

15.3 Para poder usar o serviço nos componentes da aplicação, é preciso informar sua existência no módulo principal dela. Abra o arquivo app.module.ts, mantenha todo seu conteúdo existente e faça as alterações da Listagem 15.3.1.

Listagem 15.3.1

```
import {NoteService} from './note.service'

providers: [
   StatusBar,
   SplashScreen,
   NoteService,
   {provide: ErrorHandler, useClass: IonicErrorHandler}
  ]
```

Passo 16 (Usando o serviço na classe home.ts) Para usar o serviço, precisaremos injetá-lo. Ele irá substituir o vetor de notas que temos fixo no código.

16.1 Os seguintes passos serão necessários:

- importar o serviço com uma instrução import
- declarar uma variável para guardar as notas
- pedir ao Ionic que injete a dependência (um objeto do tipo noteService) usando um parâmetro no construtor
- no construtor, obter os dados do serviço e atribuir à variável declarada

Veja a nova versão da classe HomePage na Listagem 16.1.1.

Listagem 16.1.1

```
import { Component } from '@angular/core';
import { NavController } from 'ionic-angular';
import { NoteService } from '../../app/note.service'
@Component({
  selector: 'page-home',
  templateUrl: 'home.html'
})
export class HomePage {
  notes;
  constructor(public navCtrl: NavController, private noteService:
NoteService) {
    this.notes = noteService.notes;
  }
  onItemClick(note) {
    //console.log("item-click", note);
    this.navCtrl.push('DetailPage', {
      noteParam: note
    });
  }
}
```

Exercício para avaliação

Crie uma aplicação Ionic. Ela será um chat com algumas salas de bate papo. A primeira tela deve ter dois campos:

- Um menu (https://ionicframework.com/docs/api/select) para que o usuário possa selecionar em qual sala de bate papo deseja entrar. Devem existir as salas: cinema, curiosidades e esportes.
- Um campo para texto em que o usuário digitará seu nome de usuário.

Além disso, a tela inicial tem um botão que o leva para a página do chat selecionada.

Na página do chat selecionado, há:

- Uma lista, responsável por mostrar as mensagens do chat.
- Um campo de texto em que o usuário pode digitar as suas mensagens.
- Um botão que, quando clicado, faz com que a mensagem digitada sejá adicionada ao textarea.

A mensagem deve aparecer associada ao nome do usuário.

Nesta versão inicial não há outros usuários nas salas de bate papo.