sãojudas universidade	FTCE Faculdade de Tecnologia e Ciências Exatas

USJT - 2019 – Programação para Web e Dispositivos Móveis

Professores: Anderson e Bossini

Aula: 10
Assunto: Ionic – CRUD em memória volátil – parte 2

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

Passo 17 (Apagando uma nota) A fim de apagar uma nota, iremos adicionar um botão na página de exibição de detalhes.

17.1 A alteração deve ser feita no arquivo detail.html. Note que o botão irá aparecer na barra superior. Usamos uma tag ion-buttons e a propriedade end para posicionar o botão à direita (ou à esquerda, depende do idioma). Além disso, note que usamos um nome para o ícone. O ícone a ser usado será diferente dependendo da plataforma. O Ionic faz essa escolha automaticamente. Para conhecer mais ícones e seus nomes, veja o Link 17.1.1.

Link 17.1.1
https://ionicframework.com/docs/v3/ionicons/

A Listagem 17.1.1 mostra as alterações. O método onTrash ainda não existe, iremos implementá-lo a seguir.

Listagem 17.1.1

```
<ion-header>
 <ion-navbar>
  <ion-title>{{note.title}}</ion-title>
  <ion-buttons end>
    <button ion-button icon-only (click)="onTrash()">
      <ion-icon name="trash"></ion-icon>
    </button>
  </ion-buttons>
 </ion-navbar>
</ion-header>

<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>
```

17.2 O método onTrash solicitará ao serviço que apague a nota desejada. O método da Listagem 17.2.1, que deve ser definido na classe **NoteService**, recebe uma nota e remove da coleção, caso exista. “splice” significa algo como emendar. Neste caso, estamos removendo o objeto na posição encontrada. O número indica que, a partir daquela posição (incluindo ela), desejamos apagar somente 1 elemento.

Listagem 17.2.1

```
removeNote (note){
     let index = this.notes.indexOf (note);
     if (index > -1){
        this.notes.splice(index, 1);
     }
  }
```

17.3 A seguir, na classe DetailPage, definimos o método onTrash. Os seguintes passos são necessários:

- importar o NoteService com uma instrução import
- injetar a dependência de NoteService no construtor
-implementar o método onTrash. Ele pede ao serviço que remova a nota e remove a página atual do topo da pilha, voltando à página anterior.

Veja a Listagem 17.3.1.

Listagem 17.3.1

```
import { Component } from '@angular/core';
import { IonicPage, NavController, NavParams } from 'ionic-angular';
import { NoteService } from '../../app/note.service'
@IonicPage()
@Component({
 selector: 'page-detail',
 templateUrl: 'detail.html',
})
export class DetailPage {
 note;
 constructor(public navCtrl: NavController,
             public navParams: NavParams,
                private noteService: NoteService) {
  this.note = this.navParams.get("noteParam");
  console.log('nav-param', this.note);
 }
```

```
 onTrash (){
  this.noteService.removeNote(this.note);
  this.navCtrl.pop();
 }
 ionViewDidLoad() {
  console.log('ionViewDidLoad DetailPage');
 }
}
```

17.4 No momento um usuário pode clicar sem querer no ícone e apagar uma nota. A aplicação pode ser mais amigável caso exiba um alerta para confirmar se o usuário deseja mesmo apagar a nota. Os passos necessários para exibir um alerta são os seguintes:

- Importar a classe AlertController com uma instrução import (ela vem do pacote ionic-angular)
- Injetar um AlertController no construtor
- No método onTrash, construir um objeto que representa o alerta, usando o objeto injetado.

O objeto alerta é representado como um JSON e algumas de suas propriedades são:

- title (Um título)
- message (Uma mensagem de confirmação)
- buttons (Um vetor de botões)

Cada botão é representado por um objeto JSON e algumas propriedades possíveis são:

- text (texto a ser exibido)
- handler (uma arrow function que será executada caso o botão seja tocado)

Um botão que não especifica um handler simplesmente faz com que o alerta suma, sem executar nenhum procedimento.

Depois do o objeto ter sido criado com o método create, ele precisa ser exibido com o método present, ambos de AlertController.

Veja a nova definição do método onTrash (e os ajutes necessários) na Listagem 17.4.1.

Listagem 17.4.1

```
import { IonicPage, NavController, NavParams, AlertController } from 'ionic-angular';
constructor(public navCtrl: NavController,
                public navParams: NavParams,
                   private noteService: NoteService,
                    private alertCtrl) {
   this.note = this.navParams.get("noteParam");
   console.log('nav-param', this.note);
 }
onTrash (){
   //constrói o alerta
   let confirm = this.alertCtrl.create({
     title: "Delete?",
     message: `Are you sure you want to delete this note: "${this.note.title}"?`,
     buttons:[
       //primeiro botão, sem handler não faz nada
       {
         text: "Cancel"
```

```
    },
    //segundo botão
    {
     text: "Confirm",
     handler: () =>{
      this.noteService.removeNote(this.note);
      this.navCtrl.pop();
     }
    }
   ]
  });
  //exibe
  confirm.present(); }
```

Passo 18 (Adicionando uma nova nota) A fim de adicionar uma nova nota, iremos utilizar um Floating Action Button (FAB), da especificação Material Design do Google.

18.1 Abra o arquivo home.html e faça as alterações da Listagem 18.1.1. Note que usamos um FAB posicionado à direita e na parte inferior da tela. Além disso, ele tem vinculado o método onAddClick (a ser implementado) e seu nome é add, um nome também pré determinado pelo Ionic.

Listagem 18.1.1

```
<ion-content padding>
 <ion-card *ngFor="let note of notes" (click)="onItemClick(note)">
  <ion-card-header>
   {{note.date | date: 'dd/MM/yyyy'}}
  </ion-card-header>
  <ion-card-content>
   <ion-card-title>
    {{note.title}}
   </ion-card-title>
   <p>
    {{note.content}}
   </p>
  </ion-card-content>
 </ion-card>
 <ion-fab end bottom>
   <button ion-fab color="primary" (click)="onAddClick()">
    <ion-icon name="add"></ion-icon>
   </button>
 </ion-fab>
</ion-content>
```

18.2 O método onAddClick irá utilizar a mesma página de detalhes para fazer a adição. Ele simplesmente empilha DetailPage sem passar parâmetros. Veja a Listagem 18.2.1.

Listagem 18.2.1

```
onAddClick (){
  this.navCtrl.push('DetailPage');
 }
```

18.3 Para diferenciar uma edição de uma adição, a classe DetailPage terá um flag (uma variável booleana) que indicará se a página está aberta para fazer uma nova adição ou não. O seu construtor tenta obter o parâmetro. Caso tenha sucesso, a operação é edição. Caso contrário, é uma adição de nova nota. O construtor se encarrega de fazer esse teste. Caso seja a adição de uma nova nota, as propriedades da nota são inicializadas com strings vazias para que a página carregue corretamente. Além disso, o flag é atualizado para true, assim a página não exibirá o botão de remoção caso estejamos lidando com uma adição. Veja a Listagem 18.3.1.

Listagem 18.3.1

```
export class DetailPage {
 note;
 newNoteFlag = false;
 constructor(public navCtrl: NavController,
                public navParams: NavParams,
                   private noteService: NoteService,
                     private alertCtrl: AlertController) {
  this.note = this.navParams.get("noteParam");
  if (!this.note){
    this.note = {
     id: "",
     date: "",
     title: "",
     content: ""
    };
    this.newNoteFlag = true;
  }
 }
```

18.4 A seguir, na classe **NoteService** implemente um método para adição de uma nota, como mostra a Listagem 18.4.1.

Listagem 18.4.1

```
addNote (note){
     this.notes.push(note);
}
```

18.5 A adição de uma nova nota só acontecerá de fato quando o usuário sair da página de adição. O método **ionViewWillLeave** faz parte do ciclo de vida do Ionic e, como o nome sugere, é chamado quando uma tela vai ser desempilhada. Neste momento verificamos se o flag de adição de nova nota é true. Se for, fazemos a adição por meio

do serviço. Caso contrário, só desempilhamos a página. Veja a Listagem 18.5.1. O método deve ser implementado na classe DetailPage.

Listagem 18.5.1

```
ionViewWillLeave (){
   if (this.newNoteFlag)
     this.noteService.addNote(this.note);
 }
```

18.6 Só desejamos exibir o botão de exclusão de notas se a página for aberta para edição. Se ela for aberta para adição de uma nova nota, não faz sentido mostrar o botão. Para exibí-lo de maneira condicional, usamos uma diretiva if do Angular. Veja a Listagem 18.6.1. Essa alteração deve ser feita no arquivo **detail.html.**

Listagem 18.6.1

```
<button ion-button icon-only (click)="onTrash()" *ngIf="!newNoteFlag">
```

Exercício para avaliação

Ajuste o seu chat da seguinte forma:

- Verificar se o nome de usuário escolhido já está em uso. Essa deve ser uma funcionalidade implementada na classe NoteService.

- Na tela de login, permita que o usuário escolha um ícone. Mostre uma lista de ícones com pelo menos 10 ícones. Esse ícone será utilizado a cada mensagem que ele enviar no chat. Assim, cada mensagem enviada por ele terá o ícone escolhido, seu nome e o texto da mensagem.

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

Suba a solução no Github.