Curso: Spring Boot com Ionic - Estudo de Caso Completo

https://www.udemy.com/user/nelio-alves

Prof. Dr. Nelio Alves

Capítulo: Aplicação Ionic - Parte 1/2

Objetivo geral:

- Construir a primeira etapa do aplicativo de pedidos usando Ionic. Nesta primeira etapa será realizado o seguinte escopo:
  - Tela de login
  - Cadastro de cliente (signup)
  - Tela de perfil
  - Fluxo básico de seleção de item para carrinho:
    - Listagem de categorias
    - Listagem de produtos de uma categoria selecionada
    - Exibição dos detalhes do produto
  - Tela de carrinho de compras com funcionalidades de incrementar, decrementar e remover itens

Instalação das ferramentas

Checklist:

- Instalar o NodeJS (https://nodejs.org)
- Testar o NodeJS:

```
node -v
npm -v
```

- Instalar o Ionic CLI e Cordova (https://ionicframework.com/docs/intro/installation):

```
npm install -g ionic@3.19.0 cordova@7.1.0
```

ATENÇÃO: O curso foi feito para Ionic versão 3.

Nós poderemos dar suporte:
1) Somente para a versão 3
2) Somente para o projeto do curso (e não para outros projetos do aluno)

Assim, caso queira suporte a eventuais dúvidas sobre o conteúdo do curso, quando for instalar o Ionic e Cordova, favor executar o comando como mostrado acima.

- Testar a instalação do Ionic e Cordova:

```
ionic -v
cordova -v
```

- Instalar seu editor preferido (sugestão: VS Code: https://code.visualstudio.com)

Criando a aplicação (Initial commit)

ionic start CursoSpringIonic sidemenu
Integrar Cordova? SIM
Ionic Pro SDK? NÃO

Apagando página List

Checklist:

- Apagar pasta
- Em app.module.ts, retirar as referências
- Em app.component.ts, retirar as referências

Lazy loading

http://blog.ionicframework.com/ionic-and-lazy-loading-pt-1/

Atenção:
- Toda página/componente deve estar declarado em um módulo da aplicação
- Por padrão, criaremos um módulo para cada página

Checklist:

- Incluir um módulo HomeModule (arquivo home.module.ts) para a página Home:

```
import { IonicPageModule } from 'ionic-angular/module';
import { NgModule } from '@angular/core';

import { HomePage } from './home';

@NgModule({
    declarations: [HomePage],
    imports: [IonicPageModule.forChild(HomePage)]
})
export class HomeModule {
}
```

- Em home.ts, incluir @IonicPage()
- Em app.module.ts, retirar as referências para HomePage
- Em app.componentes.ts, declarar a RootPage como um string, retirar import de HomePage

Tela inicial

Checklist:

- Obter uma imagem e salvá-la em src/assets/imgs
- Se desejar, aplicar estilo
- Escrever o HTML

Navegação

Checklist (navegação):

- Criar página Categorias: ionic generate page Categorias
- Incluir botão menuToggle na barra de navegação
- Em home.ts, criar método login()
- Em home.html, no botão de login, criar binding para método login()

Desabilitar menu na tela inicial

Checklist

- Em home.ts:

```
constructor(public navCtrl: NavController, public menu: MenuController) {
}

ionViewWillEnter() {
  this.menu.swipeEnable(false);
}

ionViewDidLeave() {
  this.menu.swipeEnable(true);
}
```

Listando categorias

Referências:

https://angular.io/guide/architecture

> *"Component classes should be lean. They don't fetch data from the server, validate user input, or log directly to the console. They delegate such tasks to services.*

A component's job is to enable the user experience and nothing more. It mediates between the view (rendered by the template) and the application logic (which often includes some notion of a model). A good component presents properties and methods for data binding. It delegates everything nontrivial to services."

https://angular.io/guide/http

HttpClient - versão 4.3.1+

Checklist:

- Importar **HttpClientModule** no módulo principal (depois de BrowserModule)
- Criar arquivo com configurações da API
- Criar CategoriaDTO
- Criar CategoriaService com método findAll
- Em app.module.ts, registrar CategoriaService nos providers
- Em CategoriasPage, chamar findAll

Mostrando as categorias

Referências:

https://ionicframework.com/docs/components/#thumbnail-list

Checklist:

- Em API_CONFIG, incluir a URL básica do bucket
- Atualizar o component
- Atualizar o template

Criando um interceptor para tratamento de erros

Referências:

https://angular.io/guide/http
https://theinfogrid.com/tech/developers/angular/building-http-interceptor-angular-5/
https://stackoverflow.com/questions/46019771/catching-errors-in-angular-httpclient
https://juristr.com/blog/2017/08/intercept-http-requests-in-angular/

Checklist:

- Criar um interceptor que retorne apenas o objeto de erro devolvido pelo backend
- Declarar o provider do interceptor (pode ser no mesmo arquivo do interceptor)
- Em app.module.ts, registrar o provider do interceptor

Obtendo os dados do formulário de login

Checklist:

- Criar um model CredenciaisDTO
- Em home.ts, declarar um objeto do tipo CredenciaisDTO com valores vazios
- Em home.html, fazer nos inputs o binding com os dados do objeto declarado
- Em home.ts, testar o envio dos dados

Começando a implementar a autenticação

Checklist:

- Criar um novo serviço AuthService com um método authenticate
  - observe : 'response'
  - responseType: 'text'
- Em home.ts, chamar authenticate no método login
- Em app.module.ts, declarar AuthService nos providers

Salvando os dados do usuário logado no localStorage

`localStorage.getItem("chave")`
: Retorna o valor (string) correspondente à chave, ou null caso a chave não exista

`localStorage.removeItem("chave")`
: Remove o item do localStorage, caso ele exista

`localStorage.setItem("chave", "valor")`
: Define um item no localStorage com a chave e valor dados

Checklist:

- Definir um nome de chave para o usuário logado ser armazenado no localStorage
- Criar um tipo LocalUser correspondente aos dados do usuário logado
- Criar um serviço StorageService para salvar e obter o usuário logado
- Em app.module.ts, registrar o StorageService nos providers
- Em AuthService, criar os métodos successfulLogin e logout
- Em home.ts, chamar successfulLogin se a autenticação ocorrer com sucesso

Extraindo o email do token e armazenando-o em localStorage

Checklist:

- Instalar biblioteca para manipular token jwt:
  - npm install --save angular2-jwt
- Em LocalUser, incluir o atributo email
- Em AuthService, atualizar successfulLogin, incluindo email extraído do token

Criando uma página de profile

Checklist:

- Criar página Profile
- Em profile.ts, criar um atributo email e carregá-lo do storage
- Em profile.html, mostrar o valor de email por meio de uma interpolação
- Em app.component.ts, atualizar os itens do menu: Categorias e Profile

Mostrando dados e imagem do cliente para página de profile

Checklist:

- Incluir uma imagem de avatar na pasta assets/imgs
- Criar ClienteDTO
- Criar o serviço ClienteService com o método findByEmail
    - Provisoriamente: inclua o header Authorization
- Em app.module.ts, registrar ClienteService nos providers
- Criar página Profile
- Registrar a página Profile nos itens do menu da aplicação
- Em profile.ts, chamar o método findByEmail
- Atualizar HTML
- Incluir lógica para obter a URL da imagem no bucket S3, se ela existir

Interceptor para incluir token nas requisições

Referências:

https://angular.io/guide/http
http://blog.ionicframework.com/handling-cors-issues-in-ionic/
https://stackoverflow.com/questions/11315872/allow-ajax-gets-from-amazon-s3-access-control-allow-origin
http://docs.aws.amazon.com/pt_br/AmazonS3/latest/dev/cors.html#how-do-i-enable-cors
http://docs.aws.amazon.com/AmazonS3/latest/user-guide/add-cors-configuration.html

Checklist:

- Criar AuthInterceptor
    - Atenção: getLocalUser() pode retornar null
- Declarar o provider do interceptor (pode ser no mesmo arquivo do interceptor)
- Em ClienteService, atualizar o código
- Em app.module.ts, registrar o AuthInterceptorProvider ANTES do ErrorInterceptorProvider
- Em AuthInterceptor, incluir código para não enviar header Authorization em caso de requisição para o bucket do S3

Tratando erros 403

Checklist:

- Em ErrorInterceptor, acrescentar um tratamento específico para 403
- Em profile.ts, realizar o redirecionamento para HomePage em caso de erro 403

Caso queira usar para teste de invalidação de token:
localStorage.setItem('localUser',
'{"token":"eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJuZWxpby5jdXJzb3NAZ21haWwuY29tIiwiZXhwIjoxNTEzNjkxMTE1fQ.Bg8nyUf5Hsw2CC3dQffZrip822eFB18jNLrsySe51Eb-SioUH-uq7CQ4dWoBixZmzT-PWdE1iZZ1uRhu-aaaaa","email":"nelio.cursos@gmail.com"}')

Tratamento padrão para outros erros com Alert

- Atualizar ErrorInterceptor
- Tratar erros 401
- Tratar outros erros

Fazendo o app aproveitar o usuário logado na tela inicial

Checklist:

- Em AuthService, criar o método refreshToken
- Em home.ts, implementar o evento ionViewDidEnter para chamar o refresh token

Botão de logout no menu

Em app.component.ts, incluir código para fazer logout

Criando página de Signup

Checklist:

- Criar página de Signup com Ionic CLI
- Especificar o formulário HTML
    - Nota: usar (ngSubmit)="signupUser(); $event.preventDefault()" para evitar envio indesejado na página
- Programar a navegação para a página de Signup
- Testar evento de submissão do formulário

Criando um FormGroup para controlar o formulário

Checklist:

- Em signup.ts, definir o formGroup
- Em signup.html, fazer os devidos bindings

Povoando dinamicamente estado e cidade

Checklist:

- Criar DTO para Cidade e Estado
- Criar service para Cidade e Estado
- Em signup.module.ts, registrar os serviços nos providers
- Em signup.ts, implementar ionViewDidLoad para carregamento inicial dos estados e cidades
- Em signup.html, povoar os <ion-option> com *ngFor
- Criar um evento para carregar as cidades quando um estado é selecionado

Mostrando erros de validação

Checklist:

- Incluir tags para validação
- Definir estilo CSS
- Bloquear o formulário se ainda não estiver válido

Salvando novo cliente

Checklist:

- Em ClienteService, criar método para inserir
- Em signup.ts, implementar a lógica para inserir
- Criar um model FieldMessage
- Em ErrorInterceptor, tratar especificamente o erro 422
- Mostrar uma mensagem de confirmação de cadastro

Criando página de produtos

Referências:
https://pt.stackoverflow.com/questions/191940/alterar-máscara-de-real-angular-2

Checklist:

- Criar ProdutoDTO
- Criar página Produtos
    - Definir HTML
    - Definir script básico com dados mockados
    - Incluir a imagem de produto sem imagem na pasta assets/imgs
- Em categorias.ts, acrescentar um método showProdutos() para abrir a página de produtos
- Em categorias.html, acrescentar um binding de evento click para o método showProdutos()

Carregando produtos de uma dada categoria

Checklist:

- Criar ProdutoService com um método findByCategoria para obter os produtos de uma dada categoria
- Em app.module.ts, declarar ProdutoService nos providers
- Em categorias.ts, em showProdutos, incluir o código da categoria como parâmetro da chamada de push
- Em categorias.html, acrescentar o código da categoria como parâmetro na chamada showProdutos
- Em produtos.ts, fazer as alterações necessárias

Carregando imagens dos produtos

Checklist:

- Em ProdutoService, implementar um método getSmallImageFromBucket
- Em produtos.ts, criar um método para setar as URL's das imagens de miniatura dos produtos

Criando página de detalhes do produto

Checklist:

- Criar página ProdutoDetail
    - Definir HTML (usar operador de navegação segura)
    - Definir script básico com dados mockados
- Em produtos.ts, acrescentar um método showDetail() para abrir a página de detalhes
- Em produtos.html, acrescentar um binding de evento click para o método showDetail ()

Carregando dados do produto

Checklist:

- Em ProdutoService, incluir um método findById e o getImageFromBucket
- Em produtos.ts, em showDetail, incluir o código do produto como parâmetro da chamada de push
- Em produtos.html, incluir o código do produto como parâmetro da chamada de showDetail
- Em produto-detail.ts, fazer as alterações necessárias

Criando página de carrinho de compras

Checklist:

- Criar models CartItem e Cart
- Em STORAGE_KEYS, acrescentar uma chave "cart"
- Em StorageService, criar métodos para obter e salvar o carrinho em localStorage
- Criar CartService com operações para criar, limpar, obter e adicionar produto ao carrinho
- Em app.module.ts, registrar CartService nos providers
- Criar página Cart
    - Definir HTML
    - Definir script básico
- Em produto-detail.html, criar binding de click para método addToCart(item)
- Em produto-detail.ts, criar método addToCart(item)

Terminando as funcionalidades do carrinho

Checklist:

- Em CartService, acrescentar funcionalidades para remover item, incrementar e decrementar quantidade, calcular total
- Em cart.ts (componente), criar os métodos para remover, incrementar, decrementar e calcular total
- Atualizar HTML
- Em cart.html, incluir um botão "Continuar comprando"
- Em cart.ts (componente), implementar a função para continuar comprando
- **IMPORTANTE**: Em auth.service.ts, incluir instrução para limpar o carrinho em caso de login

Acesso ao carrinho: botão flutuante e menu

Checklist:

- Acrescentar um item de menu
- Acrescentar o código HTML nas páginas desejadas (abaixo)

```
<ion-fab top right edge>
  <button navPush="CartPage" ion-fab mini><ion-icon name="cart"></ion-icon></button>
</ion-fab>
```