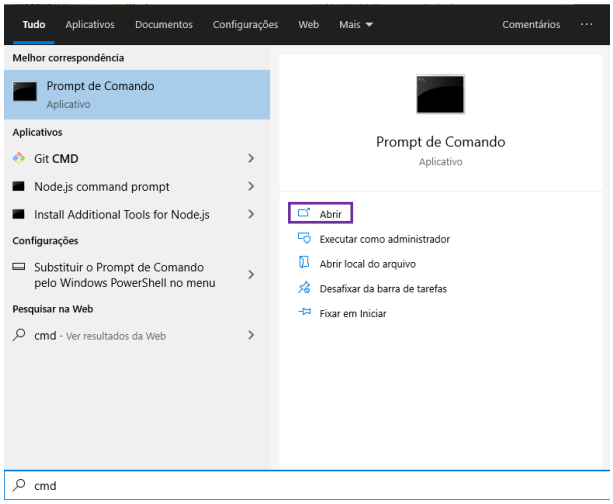
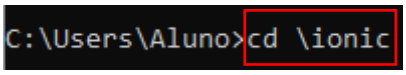
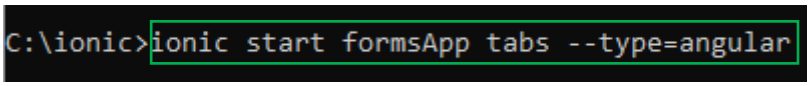
https://github.com/gabrielravanhan/formsApp

Passo 01 – Abra o Prompt de Comando e navegue até a pasta ionic com o comando cd \ionic.

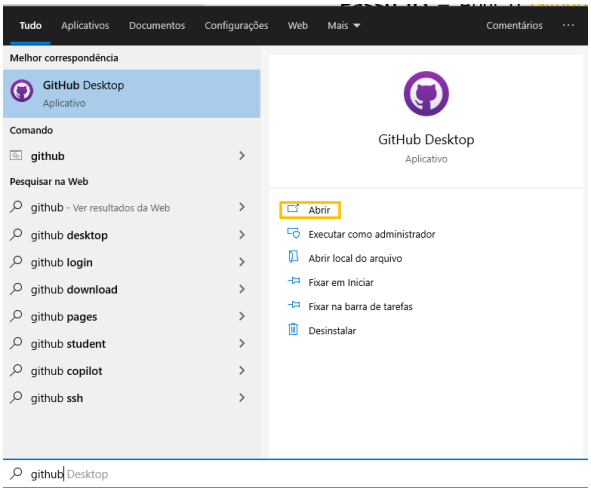


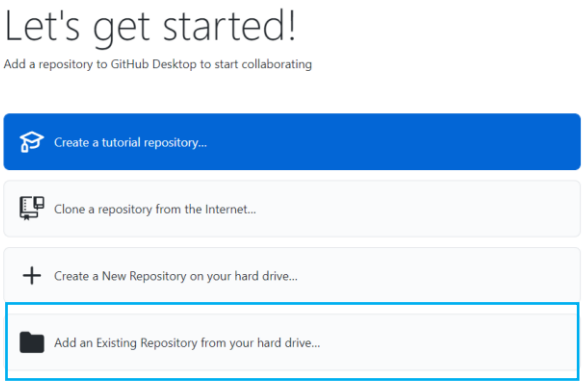


Passo 02 – Crie o projeto com o comando ionic start formsApp tabs --type=angular.

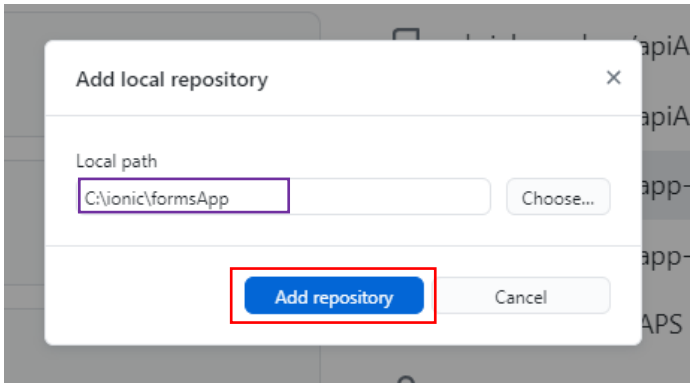


Passo 03 – Abra o Github Desktop e selecione o botão "Add an Existing Repository from your hard drive...".

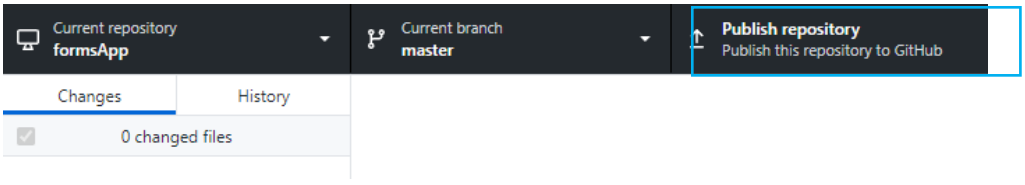


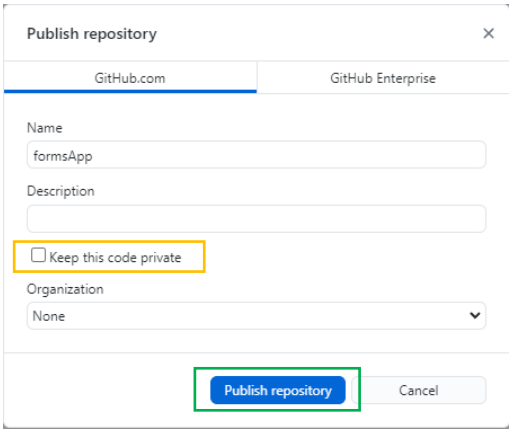


Passo 04 – Digite “C:\ionic\formsApp” e aperte o botão “Add Repository”.



Passo 05 – Selecione “Publish repository”, desmarque a caixa “Keep this code private” e, novamente, aperte “Publish repository”.





Passo 06 – Após publicar o projeto no Github, crie as páginas de login, registro e cadastro de produto com os comandos ionic g page login, ionic g page registro e ionic g page cadastro-registro.

```
C:\ionic\formsApp>ionic g page login
```

```
C:\ionic\formsApp>ionic g page registro
```

```
C:\formsApp>ionic g page cadastro-produto
```

Passo 07 – No o arquivo *app-routing.module.ts* e adicione o seguinte código:

```typescript
import { NgModule } from '@angular/core';
import { PreloadAllModules, RouterModule, Routes } from '@angular/router';

const routes: Routes = [
  {
    path: '',
    redirectTo: 'login',
    pathMatch: 'full'
  },
  {
    path: '',
    loadChildren: () => import('./tabs/tabs.module').then(m => m.TabsPageModule)
  },
  {
    path: 'login',
    loadChildren: () => import('./login/login.module').then(m => m.LoginPageModule)
  },
  {
    path: 'registro',
    loadChildren: () => import('./registro/registro.module').then(m => m.RegistroPageModule)
  },
  {
    path: 'cadastro-produto',
    loadChildren: () => import('./cadastro-produto/cadastro-produto.module').then( m => m.CadastroProdutoPageModule)
  }
];
```

Passo 8 – Adicione e importe o *ReactiveFormsModule* nos arquivos *module.ts* das páginas de login, registro e cadastro de produto, como no exemplo abaixo:

```typescript
import { NgModule } from '@angular/core';
import { CommonModule } from '@angular/common';
import { FormsModule, ReactiveFormsModule } from '@angular/forms';

import { IonicModule } from '@ionic/angular';

import { CadastroProdutoPageRoutingModule } from './cadastro-produto-routing.module';

import { CadastroProdutoPage } from './cadastro-produto.page';

@NgModule({
  imports: [
    CommonModule,
    FormsModule,
    IonicModule,
    CadastroProdutoPageRoutingModule,
    ReactiveFormsModule
  ],
  declarations: [CadastroProdutoPage]
})
export class CadastroProdutoPageModule { }
```

Passo 9 – No *login.page.ts*:

- Crie as variáveis *formLogin* e *mensagens*;
- Crie outra chamada *formBuilder* dentro dos parênteses do *constructor;*
- Insira o código que fará a validação dentro das chaves do *constructor*.
- Realize as importações necessárias.

```ts
import { Component, OnInit } from '@angular/core';
import { FormBuilder, FormGroup, Validators } from '@angular/forms';

@Component({
  selector: 'app-login',
  templateUrl: './login.page.html',
  styleUrls: ['./login.page.scss'],
})
export class LoginPage implements OnInit {

  mostrarSenha = false;
  iconeSenha = 'eye';

  formLogin: FormGroup;

  mensagens = {
    email: [
      { tipo: 'required', mensagem: 'O campo E-mail é obrigatório.' },
      { tipo: 'email', mensagem: 'E-mail Inválido.' },
    ],
    senha: [
      { tipo: 'required', mensagem: 'É obrigatório confirmar senha.' },
      { tipo: 'minlength', mensagem: 'A senha deve ter pelo menos 6 caracteres.', },
      { tipo: 'maxlength', mensagem: 'A senha deve ter no máximo 8 caracteres.' },
    ]
  };

  constructor(private formBuilder: FormBuilder) {
    this.formLogin = this.formBuilder.group({
      email: ['', Validators.compose([Validators.required, Validators.email])],
      senha: ['', Validators.compose([Validators.required, Validators.minLength(6), Validators.maxLength(8)])]
    });
  }
```

Passo 10 – Adicione o seguinte código no arquivo *global.scss*:

```scss
.msg-erro {
  color: #ff0000;
  font-size: 12px;
  clear: both;
  margin-left: 5px;
}
```

Passo 11 – No *login.page.html*:

- Adicione um *[formGroup]* na tag *form*.
- Coloque um *formControlName* em todos os *ion-input*'s.

- Adicione um *ng-container* após cada um deles.

```
<form [formGroup]="formLogin">
```

```
<ion-item>
  <ion-input type="email" placeholder="E-mail" clear-input formControlName="email"></ion-input>
</ion-item>
<ng-container *ngFor="let erro of mensagens.email">
  <span class="msg-erro"
    *ngIf="formLogin.get('email').hasError(erro.tipo) && (formLogin.get('email').dirty || formLogin.get('email').touched)">
    {{ erro.mensagem }}
  </span>
</ng-container>
```

Passo 12 – No *registro.page.ts*:

- Crie as variáveis *formRegistro* e *mensagens*;
- Crie outra chamada *formBuilder* dentro dos parênteses do *constructor;*
- Insira o código que fará a validação dentro das chaves do *constructor*.
- Realize as importações necessárias.

```
import { FormBuilder, FormGroup, Validators } from '@angular/forms';

@Component({ ···
})
export class RegistroPage implements OnInit {

  mostrarSenha = false;
  mostrarConfirmarSenha = false;
  iconeSenha = 'eye';
  iconeConfirmarSenha = 'eye';

  formRegistro: FormGroup;

  mensagens = {
    nome: [
      { tipo: 'required', mensagem: 'O campo Nome é obrigatório.' },
      { tipo: 'minlength', mensagem: 'O nome deve ter pelo menos 3 caracteres.' }
    ],
    cpf: [
      { tipo: 'required', mensagem: 'O campo CPF é obrigatório.' }
    ],
    email: [
      { tipo: 'required', mensagem: 'O campo E-mail é obrigatório.' },
      { tipo: 'email', mensagem: 'E-mail Inválido.' }
    ],
    senha: [
      { tipo: 'required', mensagem: 'A senha é obrigatória.' },
      { tipo: 'minlength', mensagem: 'A senha deve ter pelo menos 6 caracteres.', },
      { tipo: 'maxlength', mensagem: 'A senha deve ter no máximo 8 caractéres.' }
    ],
    confirmaSenha: [
      { tipo: 'required', mensagem: 'É obrigatório confirmar senha.' },
      { tipo: 'minlength', mensagem: 'A senha deve ter pelo menos 6 caracteres.', },
      { tipo: 'maxlength', mensagem: 'A senha deve ter no máximo 8 caractéres.' }
    ]
  };

  constructor(private formBuilder: FormBuilder) {
    this.formRegistro = this.formBuilder.group({
      nome: ['', Validators.compose([Validators.required, Validators.minLength(3)])],
      cpf: ['', Validators.compose([Validators.required])],
      email: ['', Validators.compose([Validators.required, Validators.email])],
      senha: ['', Validators.compose([Validators.required, Validators.minLength(6), Validators.maxLength(8)])],
      confirmaSenha: ['', Validators.compose([Validators.required, Validators.minLength(6), Validators.maxLength(8)])]
    });
  }
```

Passo 13 – No *login.page.html*:

- Adicione um *[formGroup]* na tag *form*.
- Coloque um *formControlName* em todos os *ion-input*'s.
- Adicione um *ng-container* após cada um deles.

```
<form [formGroup]="formRegistro">
```

```
<ion-item>
  <ion-input type="text" placeholder="Nome" clear-input formControlName="nome"></ion-input>
</ion-item>
<ng-container *ngFor="let erro of mensagens.nome">
  <span class="msg-erro"
    *ngIf="formRegistro.get('nome').hasError(erro.tipo) && (formRegistro.get('nome').dirty || formRegistro.get('nome').touched)">
    {{ erro.mensagem }}
  </span>
</ng-container>
```

Passo 14 – No *cadastro-produto.page.ts*:

- Crie as variáveis *formCadastro* e *mensagens*;
- Crie outra chamada *formBuilder* dentro dos parênteses do *constructor;*
- Insira o código que fará a validação dentro das chaves do *constructor*.
- Realize as importações necessárias.

```
import { Component, OnInit } from '@angular/core';
import { FormBuilder, FormGroup, Validators } from '@angular/forms';

@Component({
  selector: 'app-cadastro-produto',
  templateUrl: './cadastro-produto.page.html',
  styleUrls: ['./cadastro-produto.page.scss'],
})
export class CadastroProdutoPage implements OnInit {

  formCadastro: FormGroup;

  mensagens = {
    nome: [
      { tipo: 'required', mensagem: 'O nome do produto é obrigatório.' },
      { tipo: 'minlength', mensagem: 'O nome do produto deve ter pelo menos 3 caracteres.' }
    ],
    descricao: [
      { tipo: 'required', mensagem: 'A descrição é obrigatória.' },
      { tipo: 'minlength', mensagem: 'A descrição deve ter pelo menos 5 caracteres.' }
    ],
    validade: [
      { tipo: 'required', mensagem: 'A validade é obrigatória.' }
    ],
    preco: [
      { tipo: 'required', mensagem: 'O preço é obrigatório.' }
    ]
  };

  constructor(private formBuilder: FormBuilder) {
    this.formCadastro = this.formBuilder.group({
      nome: ['', Validators.compose([Validators.required, Validators.minLength(3)])],
      descricao: ['', Validators.compose([Validators.required, Validators.minLength(5)])],
      validade: ['', Validators.compose([Validators.required])],
      preco: ['', Validators.compose([Validators.required])]
    });
  }
```

Passo 15 – No *cadastro-produto.page.html*:

- Adicione um *[formGroup]* na tag *form*.
- Coloque um *formControlName* em todos os *ion-input*'s.
- Adicione um *ng-container* após cada um deles.

```
<form [formGroup]="formCadastro">
```

```
<ion-item>
  <ion-input type="text" placeholder="Nome" clear-input formControlName="nome"></ion-input>
</ion-item>
<ng-container *ngFor="let erro of mensagens.nome">
  <span class="msg-erro"
    *ngIf="formCadastro.get('nome').hasError(erro.tipo) && (formCadastro.get('nome').dirty || formCadastro.get('nome').touched)">
    {{ erro.mensagem }}
  </span>
</ng-container>
```

Após esses 15 passos, as páginas de registro, login e cadastro de produto realizarão a validação dos dados inseridos em seus formulários.

Passo 16 – Instale o *Ionic Storage* utilizando os comandos *npm install @ionic/storage* e *npm install @ionic/storage-angular*.

```
C:\ionic\formsApp>npm install @ionic/storage
```

```
C:\ionic\formsApp>npm install @ionic/storage-angular
```

Passo 17 – Adicione e importe o *IonicStorageModule* no arquivo *app.module.ts*.

```
import { Drivers } from '@ionic/storage';
import { IonicStorageModule } from '@ionic/storage-angular';


@NgModule({
  declarations: [AppComponent],
  imports: [
    BrowserModule,
    IonicModule.forRoot(),
    AppRoutingModule,
    IonicStorageModule.forRoot({
      driverOrder: [Drivers.IndexedDB, Drivers.LocalStorage]
    })
  ],
  providers: [{ provide: RouteReuseStrategy, useClass: IonicRouteStrategy }],
  bootstrap: [AppComponent],
})
export class AppModule { }
```

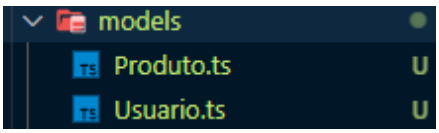
Passo 18 – Crie um serviço utilizando o comando *ionic g s services/storage*.

```
C:\ionic\formsApp>ionic g s services/storage
```

Passo 19 – Adicione o código abaixo no *storage.service.ts*.

```typescript
/* eslint-disable no-underscore-dangle */
import { Injectable } from '@angular/core';
import { Storage } from '@ionic/storage-angular';

@Injectable({...
})
export class StorageService {

  private _storage: Storage | null = null;

  constructor(private storage: Storage) {
    this.init();
  }

  async init() {
    const storage = await this.storage.create();
    this._storage = storage;
  }

  public set(key: string, value: any) {
    this._storage?.set(key, value);
  }

  public get(key: string) {
    this._storage?.get(key);
  }

  public remove(key: string) {
    this._storage?.remove(key);
  }

  public getAll() {
    const lista = [];
    this._storage.forEach((value, key, index) => {
      lista.push(value);
    });
    return lista;
  }
}
```

Passo 20 – Crie uma pasta chamada *models* com os arquivos *Usuario.ts* e *Produto.ts*.



Passo 21 – No *Usuario.ts* e *Produto.ts*, insira os respectivos códigos.

```ts
export class Usuario {
  nome: string;
  cpf: string;
  email: string;
  senha: string;
}
```

```ts
export class Produto {
  nome: string;
  descricao: string;
  validade: Date;
  preco: number;
}
```

Passo 22 – Adicione os seguintes *códigos* no *registro.page.ts*.

```ts
import { Usuario } from '../models/Usuario';
import { StorageService } from './../services/storage.service';
import { Router } from '@angular/router';

@Component({ ···
})
export class RegistroPage implements OnInit {


  formRegistro: FormGroup;
  usuario: Usuario = new Usuario();

  mensagens = { ···
  };

  constructor(private formBuilder: FormBuilder, private storageService: StorageService, private route: Router) { ···
  }


  async salvarRegistro() {
    if (this.formRegistro.valid) {
      this.usuario.nome = this.formRegistro.value.nome;
      this.usuario.cpf = this.formRegistro.value.cpf;
      this.usuario.email = this.formRegistro.value.email;
      this.usuario.senha = this.formRegistro.value.senha;
      await this.storageService.set(this.usuario.email, this.usuario);
      this.route.navigateByUrl('/tabs');
    } else {
      alert('Formulário Inválido');
    }
  }
```

Passo 23 – Coloque o método que salva o registro no botão de salvar do *registro.page.html*.

```html
<ion-button expand="block" color="success" (click)="salvarRegistro()">
  Salvar
</ion-button>
```

Passo 24 – Insira no *tab1.page.ts* o seguinte código.

```typescript
import { Component } from '@angular/core';
import { Usuario } from '../models/Usuario';
import { StorageService } from '../services/storage.service';

@Component({
  selector: 'app-tab1',
  templateUrl: 'tab1.page.html',
  styleUrls: ['tab1.page.scss']
})
export class Tab1Page {

  listaUsuarios: Usuario[] = [];

  constructor(private storageService: StorageService) { }

  async buscarUsuarios() {
    this.listaUsuarios = await this.storageService.getAll();
  }

  ionViewDidEnter() {
    this.buscarUsuarios();
  }

  async excluirCadastro(email: string) {
    await this.storageService.remove(email);
    this.buscarUsuarios();
  }
}
```

Passo 25 – No *tab1.page.html*, faça as seguintes inserções.

```html
<ion-item-sliding *ngFor="let user of listaUsuarios">
  <ion-item>
    <ion-label>
      <div>
        <h1>{{ user.nome }}</h1>
        <span style="float: right;"> {{ user.cpf }}</span>
        <a href="mailto:{{ user.email }}">{{ user.email }}</a>
      </div>
    </ion-label>
  </ion-item>

  <ion-item-options side="end">
    <ion-item-option color="danger" (click)="excluirCadastro(user.email)">
      <ion-icon slot="icon-only" name="trash"></ion-icon>
    </ion-item-option>
  </ion-item-options>
</ion-item-sliding>
```

Assim conseguiremos cadastrar e editar usuários no app.