

**INSTITUTO FEDERAL**

São Paulo

Câmpus São Carlos

Ednilson Geraldo Rossi – SC1702718  
[ednilsonrossi@gmail.com](mailto:ednilsonrossi@gmail.com)

# **Desenvolvimento de Aplicações WEB para Dispositivos Móveis**

Projeto Final: Diário do Turista

# Descrição da Atividade

- Desenvolver um aplicativo que deverá necessariamente conter, pelo menos:
  - Uma ação de formulário (FormBuilder);
  - Uma ação de recuperação de dados (BD || arquivo || web);
  - Um componente Cordova;

# Descrição do Aplicativo

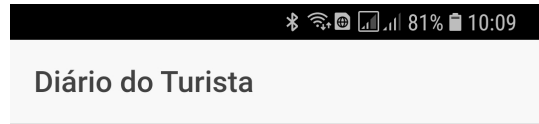
- **Descrição Geral:**

- O APP Diário de Turista permite que seu usuário registre informações sobre pontos turísticos que visitou. Seja em uma viagem ou mesmo os pontos turísticos de sua própria cidade.
- O usuário fará o cadastro do ponto turístico visitado com informações gerais e também com os comentários relevantes para ele.
- Além disso, o usuário registrará a localização do ponto turístico por meio de localização geográfica e também poderá fazer o registro por fotos de cada ponto turístico.

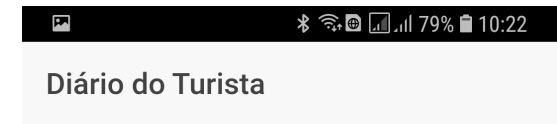
- **Funcionalidades:**

- Após acessar o sistema o usuário poderá registrar um novo ponto turístico, para isso preencherá as informações: ponto turístico (nome popular); tipo de atração (espaço de lazer, monumento, espaço de alimentação, etc); descrição do ponto turístico; seus comentários pessoais. Dados de localização serão obtidos pelo próprio sistema por geolocalização. Após registrar um ponto turístico o usuário poderá adicionar registros fotográficos do ponto turístico.
- Visualizar registros de pontos turísticos. A consulta será filtrada por data de registro, nome do ponto turístico, tipo de atração ou localização (cidade).

# O Aplicativo



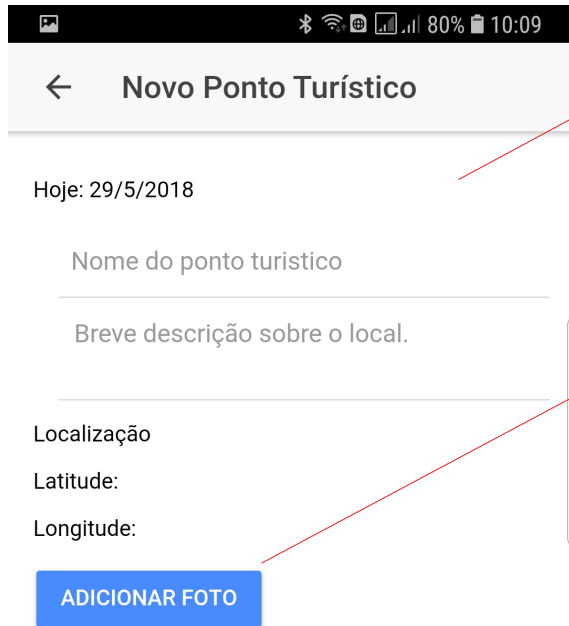
Com objetivo de manter a simplicidade a tela principal do APP apresenta a lista de pontos turísticos cadastrados e um Float Button para inclusão de novos pontos.



Fila do posto de combustíve  
29/5/2018  
Durante a greve do setor de transporte os motoi



# O Aplicativo



Hoje: 29/5/2018

Nome do ponto turístico

Breve descrição sobre o local.

Localização

Latitude:

Longitude:

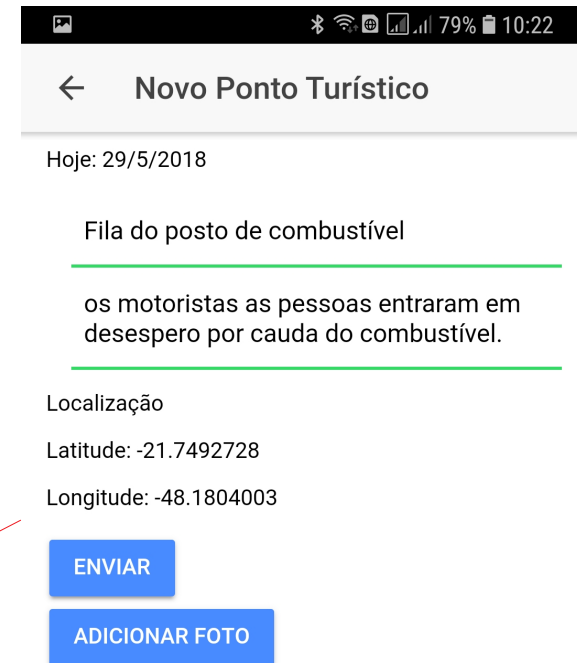
ADICIONAR FOTO

A foto:

Alguns dados são obtidos automaticamente como data e localização.

O registro fotográfico é obrigatório, por isso só é possível salvar os dados após obter uma foto.

Com os campos preenchidos é possível efetivar o registro do local visitado.



Hoje: 29/5/2018

Fila do posto de combustível

os motoristas as pessoas entraram em desespero por causa do combustível.

Localização

Latitude: -21.7492728

Longitude: -48.1804003

ENVIAR

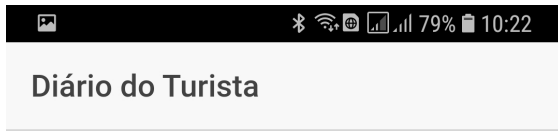
ADICIONAR FOTO

A foto:



# O Aplicativo

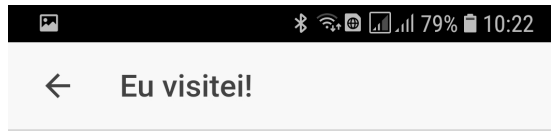
Com os dados salvos é possível visualizar as informações e, inclusive, visualizar mapa do local e traçar rota.



## Fila do posto de combustíve

29/5/2018

Durante a greve do setor de transporte os motori



## Local: Fila do posto de combustível

### Descrição:

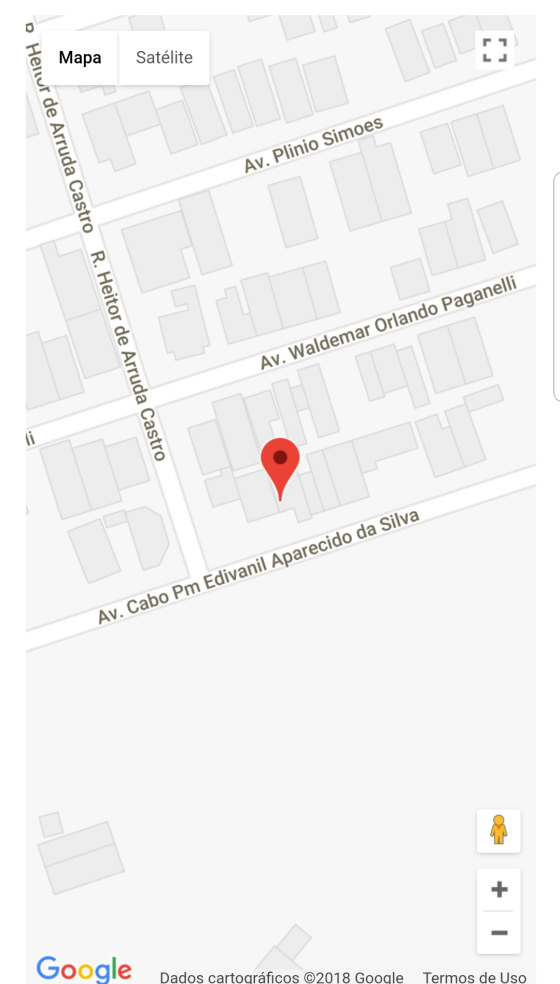
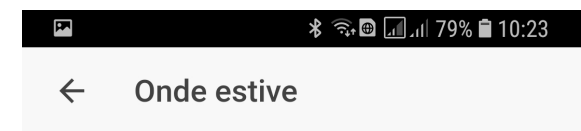
Durante a greve do setor de transporte os motoristas as pessoas entraram em desespero por cauda do combustível.

Visitado em: 29/5/2018



VER LOCAL

APAGAR





# Desenvolvimento do app

# Banco de dados

## Instalação

```
$ ionic cordova plugin add cordova-sqlite-storage  
$ npm install --save @ionic-native/sqlite
```

No terminal

app.modules.ts

```
import { SQLite } from '@ionic-native/sqlite';  
import { DatabaseProvider } from '../providers/database/database';  
  
...  
  
providers: [  
  ...  
  SQLite,  
  DatabaseProvider,  
  PontoTuristicoProvider,  
  ...  
]
```



# Banco de dados Provider

```
import { Injectable } from '@angular/core';
import { SQLite, SQLiteObject } from '@ionic-native/sqlite';

@Injectable()
export class DatabaseProvider {
  constructor(private sqlite: SQLite) { }

  public getDB() {
    return this.sqlite.create({
      name: 'diario_turista.db',
      location: 'default'
    });
  }

  public createDatabase() {
    return this.getDB()
      .then((db: SQLiteObject) => {
        this.createTables(db);
      })
      .catch(e => console.log(e));
  }

  private createTables(db: SQLiteObject) {
    db.executeSql('CREATE TABLE IF NOT EXISTS PontoTuristico (
      id integer primary key AUTOINCREMENT NOT NULL,
      ponto_turistico TEXT,
      descricao TEXT,
      data_visita TEXT, latitude TEXT, longitude TEXT, foto TEXT)', {})
      .then(() => console.log('Tabelas criadas'))
      .catch(e => console.error('Erro ao criar as tabelas', e));
  }
}
```

Quando o BD não existe  
esse é criado, caso  
contrário é aberto.

Script para a criação da  
base de dados do app.

# PontoTuristico Provider

```
export class PontoTuristicoProvider {  
  
  constructor(private dbProvider: DatabaseProvider) {  
    console.log('Hello PontoTuristicoProvider Provider');  
  }  
  
  public insert(ponto : PontoTuristico) {  
  
    return this.dbProvider.getDB()  
      .then((db: SQLiteObject) => {  
        let sql = 'INSERT INTO PontoTuristico (ponto_turistico, descricao, data_visita,  
          latitude, longitude, foto) values (?, ?, ?, ?, ?, ?)';  
        let data = [ponto.ponto_turistico,  
          ponto.descricao,  
          ponto.data_visita,  
          ponto.latitude,  
          ponto.longitude,  
          ponto.foto];  
  
        return db.executeSql(sql, data)  
          .catch((e) => console.error(e));  
      })  
      .catch((e) => console.error(e));  
  }  
}
```

Operação SQL para  
inserção de dados na  
base.

# PontoTuristico Provider

```
public get(id:number){  
    return this.dbProvider.getDB()  
        .then((db:SQLiteObject) => {  
            let sql = 'select * from PontoTuristico where id = ?';  
            let data = [id];  
  
            return db.executeSql(sql, data)  
                .then((data:any) => {  
                    if(data.rows.length > 0){  
                        let item = data.rows.item(0);  
                        let p = new PontoTuristico();  
                        p.data_visita = item.data_visita;  
                        p.descricao = item.descricao;  
                        p.foto = item.foto;  
                        p.id = item.id;  
                        p.latitude = item.latitude;  
                        p.longitude = item.longitude;  
                        p.ponto_turistico = item.ponto_turistico;  
  
                        return p;  
                    }  
  
                    return null;  
                })  
            .catch((e) => console.error(e));  
        })  
    .catch((e) => console.error(e));  
}
```

Operação SQL recuperar um ponto turístico específico, considerando o id como chave da busca.

# PontoTuristico Provider

```
public getAll() {  
    return this.dbProvider.getDB()  
        .then((db: SQLiteObject) => {  
            let sql = 'SELECT * FROM PontoTuristico order by id  
desc';  
            var data: any[];  
  
            return db.executeSql(sql, data)  
                .then((data: any) => {  
                    if (data.rows.length > 0) {  
                        let pontos: any[] = [];  
                        for (var i = 0; i < data.rows.length; i++) {  
                            var ponto = data.rows.item(i);  
                            pontos.push(ponto);  
                        }  
                        return pontos;  
                    } else {  
                        return [];  
                    }  
                })  
                .catch((e) => console.error(e));  
        })  
        .catch((e) => console.error(e));  
}
```

Operação SQL recuperar  
todos os pontos turísticos  
inseridos na base de  
dados.

# PontoTuristico Provider

Remover ponto turístico.

```
public delete(id : number){  
  return this.dbProvider.getDB()  
    .then((db:SQLiteObject) => {  
      let sql = 'delete from PontoTuristico where id = ?';  
      let data = [id];  
  
      return db.executeSql(sql, data)  
        .catch((e) => console.error(e));  
    })  
    .catch((e) => console.error(e));  
}
```

```
export class PontoTuristico{  
  id : number;  
  ponto_turistico : String;  
  descricao : String;  
  data_visita : String;  
  latitude : String;  
  longitude : String;  
  foto : String;  
}
```

Ainda no Provider foi definida a classe que representa um ponto turístico. Por meio desta classe os objetos são representados em todo app.

# Gerar lista de pontos turísticos

- A lista foi gerada usando um componente ion-list e os dados são recuperados do Banco SQLite.

```
<ion-content padding>
  <ion-list>
    <ion-item-sliding *ngFor="let ponto of pontos">
      <button ion-item (click)="exibe(ponto.id)">
        <h1>{{ponto.ponto_turistico}}</h1>
        <h6 align="right">{{ponto.data_visita}}</h6>
        <h3>{{ponto.descricao}}</h3>
      </button>
    </ion-item-sliding>
  </ion-list>

  <ion-fab right bottom>
    <button ion-fab color="light" (click)="add()"><ion-icon name="add"></ion-icon></button>
  </ion-fab>
</ion-content>
```

```
ionViewDidEnter() {
  this.getAllPontos();
}

getAllPontos() {
  this.pontoProvider.getAll()
    .then((result: any[]) => {
      this.pontos = result;
    });
}
```

# Inserir Ponto Turístico

## Apresentação

```
<ion-header>
  <ion-navbar>
    <ion-title>Novo Ponto Turístico</ion-title>
  </ion-navbar>
</ion-header>

<ion-content padding>
  <form [formGroup]="formulario" (submit)="salvar()">
    <p>Hoje: {{hoje}}</p>
    <ion-item>
      <ion-input type="text" formControlName="titulo"
        placeholder="Nome do ponto turistico"></ion-input>
    </ion-item>
    <ion-item>
      <ion-textarea formControlName="descricao"
        placeholder="Breve descrição sobre o local."></ion-textarea>
    </ion-item>
    <p>Localização</p>
    <p>Latitude: {{ponto.latitude}}</p>
    <p>Longitude: {{ponto.longitude}}</p>
    <button ion-button *ngIf="temFoto">Enviar</button>
  </form>
  <button ion-button (click)="tiraFoto()">Adicionar Foto</button>
  <h2>A foto: </h2>
  <img [src]="imagem" *ngIf="imagem" />
</ion-content>
```

Os dados são inseridos no sistema por meio de um formulário. Esse por sua vez é gerado e validado por um formGroup

Mecanismo para submissão dos dados apenas se houver uma foto do ponto turístico.

# Inserir Ponto Turístico

## Recuperar localização

```
public getLocation():void{
    this.geolocation.getCurrentPosition().then((resp) => {
        this.ponto.latitude = resp.coords.latitude.toString();
        this.ponto.longitude = resp.coords.longitude.toString();

    }).catch((error) => {
        console.log('Error getting location', error);
    });
}
```

A localização é recuperada ao carregar a página com o formulário.

Os dados são exibidos tão logo sejam recuperados.



# Inserir Ponto Turístico

## Tirar a foto

```
public tiraFoto(){
  const options: CameraOptions = {
    quality: 100,
    destinationType: this.camera.DestinationType.DATA_URL,
    encodingType: this.camera.EncodingType.JPEG,
    mediaType: this.camera.MediaType.PICTURE
  }

  this.camera.getPicture(options).then((imageData) => {
    // imageData is either a base64 encoded string or a file URI
    // If it's base64:
    let base64Image = 'data:image/jpeg;base64,' + imageData;
    this.imagem = base64Image;
    this.ponto.foto = this.imagem;
    this.temFoto = true;
  }, (err) => {
    // Handle error
    console.log("Erro ao tirar foto!");
  });
}
```

Imagem salva em base 64 para armazenamento na base de dados em formato texto.

# Inserir Ponto Turístico

## Salvar dados

```
public salvar(){  
  
    let {titulo, descricao} = this.formulario.controls;  
  
    let str1 = descricao.value.toString();  
    let str2 = titulo.value.toString();  
  
    this.ponto.descricao = str1;  
    this.ponto.ponto_turistico = str2;  
  
    this.pontoTuristicoProvider.insert(this.ponto)  
        .then(() => {  
        this.toast.create({ message: 'Ponto turístico salvo.', duration: 3000,  
            position: 'bottom' }).present();  
        this.navCtrl.pop();  
    })  
        .catch(() => {  
        this.toast.create({ message: 'Erro ao salvar o ponto turístico.',  
            duration: 3000, position: 'bottom' }).present();  
    });  
  
}
```

Dados são extraídos do formulário e o objeto ponto tem seus dados setados.

Com o objeto ponto insere-se o mesmo na base de dados utilizando o provider.

# Visualização Mapa

Utilizando a API da Google o Mapa é carregado com os dados do ponto turístico visitado.

Importante observar que para utilizar o mapa deve-se solicitar uma chave ao Google.

```
ionViewDidLoad() {  
  console.log('ionViewDidLoad MapaPage');  
  
  const position = new google.maps.LatLng(this.latitude, this.longitude);  
  const mapOptions = {  
    zoom : 18,  
    center : position  
  }  
  
  this.map = new google.maps.Map(document.getElementById('map'), mapOptions);  
  
  const marker = new google.maps.Marker({  
    position: position,  
    map: this.map  
  });  
}
```



# Referências

- <https://ionicframework.com/docs/native/sqlite/>
- <https://ionicframework.com/docs/native/camera/>
- <https://ionicframework.com/docs/native/geolocation/>
- <https://ionicframework.com/docs/developer-resources/forms/>
- <http://www.fabricadecodigo.com/google-maps-e-geolocalizacao-com-ionic/>



# Acessar o projeto

- **Projeto**

- [https://github.com/ednilsonrossi/DWM\\_DiarioDoTurista](https://github.com/ednilsonrossi/DWM_DiarioDoTurista)

- **APK**

- <https://bit.ly/2H065WQ>