Ionic 2-mobile app Framework

Ionic 2 depende de Angular 2 y de Corodova para funcionar, es un framework completo para crear vistas específicas para dispositivos móviles. Desde la página oficial se consultar toda la información con códigos de ejemplo. https://ionicframework.com/docs/

Instalación

Lo primero es instalar a través del NPM los paquetes de ionic y de cordova.

```
npm install -g ionic cordova
```

Para poder crear las apliaciones en Android u otras plataformas se necesita tener instalado en el equipo el SDK de Android, recomendada el API 24,

Una vez completada la instalación con éxito, se pueden crear proyectos en Ionic 2 con el comando: el atributo –V2 es para indicar que use Ionic 2

```
ionic start <nombre_app> --v2
```

Esto genera una estructura de archivos con todas las dependencias de Ionic 2, todo el desarrollo de la APP se crea dentro de la carpeta src. Para ejecutar la aplicación en modo desarrollador en el navegador, entrar en la carpeta del proyecto y ejecutar:

```
Ionic serve
```

Plataformas Ionic 2

Ionic 2 dispone de varias plataformas para generar la compilación, Android, IOS (solo desde Mac), Blackberry, Windows,… para ver el listado de plataformas, usar el comando *ionic platform list.* Cada plataforma suele tener sus dependencias, por ejemplo, para poder añadir (platform add nombre_plataforma)/construir (ionic build nombre_plataforma) la plataforma de Android, previamente hay que tener instalado el SDK de Android.

Para añadir plataforma Android:

```
ionic platform add android
```

Para compilar y obtener archivo .apk para ser instalado en dispositivo:

```
Ionic build android
```

Navegación en Ionic 2, componente NavController

Usando este componente en cada página se disponen para usar las características de este controlador, que a diferencia del routing de angular 2, en Ionic 2 tiene su propio sistema, copiando la funcionalidad de IOS, basado en capas o también llamada “navegación natural”.

- *En cada selección, se hace push de la vista siguiente*
    - En la práctica del restaurante, al hacer clic sobre uno del listado se llama a la función itemTapped que hace un this.nvCtrl.push con parámetros llamando a ItemDetailsPage. **Para poder hacer push de una página esta debe ser importada en el componente que la llama**:

```
itemTapped(event, restaurante) {
    this.navCtrl.push(ItemDetailsPage, {
      item: restaurante
    });
  }
```

- *Si en algún momento, seleccionas el botón atrás, se hace un pop de la vista actual.*

```
this.navCtrl.pop();
```

Declaración del uso del componente:

```
import { NavController } from 'ionic-angular';

class MyComponent {
  constructor(public navCtrl: NavController) {

  }
}
```

LifeCycles

Un aspecto muy importante a conocer de Ionic 2 son los ciclos de vida de una página, el caso más claro es cuando queremos recargar la página cada vez que se muestre por algún tipo de necesidad, como reflejar cambios que afectan a la página que se han hecho en otras páginas/secciones de la app. Por ejemplo, si eliminamos un restaurante, se debe recargar de nuevo el listado cuando se muestre de nuevo, entonces, en el componente del listado, que es al que le afecta, se debe declarar el método *ionViewWillEnter*.

```
ionViewWillEnter (){
            console.log('Pagina principal activa');
            this.getRestaurantes();
      }
```

En la página oficial comenta a fondo todos los tipos.

Personalizar tema

Ionic 2 por defecto, tiene un archivo SASS "variables.scss" dentro de la carpeta *theme* con la declaración de los colores por defecto, color del texto, fondo, etc. En este archivo se pueden declarar más variables, por ejemplo, un color para el fondo del header sería añadir una nueva variable dentro de $colors, por ejemplo, navbarColor:

```
$colors: (
  primary:    #387ef5,
  secondary:  #32db64,
  danger:     #f53d3d,
  light:      #f4f4f4,
  dark:       #222,
  navbarColor: #CC3F10
);
```

Componentes Ionic2

Ionic 2 dispone de una gran variedad de componentes que nos ayudan a construir una App 100% mobile, a destacar las cards, actionsheets, lists, buttons, icons, toasts, modals, alerts, entre otras.

Toda la información y códigos de ejemplo está disponible en https://ionicframework.com/docs/v2/components/

Grupoloyal

Geo-localización y Google Maps

Mapas con Ionic2

Se puede usar tanto en ionic 2 como en Angular 2, es el método standard para usar google maps sin necesidad de instalar nada previamente llamando a la librería directamente en el index.html de la App, aquí veremos un ejemplo que sirve para cualquier otro APIi que se quiera usar en TypeScript usando una librería externa declarada en el index.html principal.

En index.html se declara el uso del API, para entornos de producción es necesario declarar un API añadiendo ?key=<key_api>, es recomendable también ponerla en entorno de desarrollo ya que pueda saltar el warning en la consola si se usa muchas veces seguidas.

```
<script src="http://maps.google.com/maps/api/js"></script>
```

Usando de base una plantilla en blanco de ionic 2, el home.ts quedaría.

```
import { Component ,ViewChild, ElementRef} from '@angular/core';

import { NavController } from 'ionic-angular';

declare var google;

@Component({
  selector: 'page-home',
  templateUrl: 'home.html'
})
export class HomePage {
       @ViewChild('map') mapElement: ElementRef;
  map: any;

  constructor(public navCtrl: NavController) {

  }

   ionViewDidLoad(){
    this.loadMap();
  }

  loadMap(){

    let latLng = new google.maps.LatLng(-34.9290, 138.6010);

    let mapOptions = {
      center: latLng,
      zoom: 15,
      mapTypeId: google.maps.MapTypeId.ROADMAP
    }

    this.map = new google.maps.Map(this.mapElement.nativeElement,
mapOptions);

  }

}
```

Grupoloyal

Mapas con geo-localización

Primero, añadir el plug-in.

```
ionic plugin add cordova-plugin-geolocation
```

```
import { Component ,ViewChild, ElementRef} from '@angular/core';

import { NavController } from 'ionic-angular';
import { Geolocation } from 'ionic-native';

//Necesario para poder crear mapas, es la forma de declara una variable de
uso javascript dentro de typeScript
declare var google;

@Component({
  selector: 'page-home',
  templateUrl: 'home.html'
})
export class HomePage {
       @ViewChild('map') mapElement: ElementRef;
  map: any;

  constructor(public navCtrl: NavController) {

  }

   ionViewDidLoad(){
    this.loadMap();
  }

  loadMap(){

    Geolocation.getCurrentPosition().then((position) => {

      let latLng = new google.maps.LatLng(position.coords.latitude,
position.coords.longitude);

      let mapOptions = {
        center: latLng,
        zoom: 15,
        mapTypeId: google.maps.MapTypeId.ROADMAP
      }

      this.map = new google.maps.Map(this.mapElement.nativeElement,
mapOptions);

    }, (err) => {
      console.log(err);
    });
  }
}
```

Plantilla

#map es el atributo que se declara para el @ViewChild('map') mapElement: ElementRef;

```
<div #map id="map" style="height: 300px;width: 300px"></div>
```

Añadir Marcadores

Añadir las funciones

```
addMarker(){

        let marker = new google.maps.Marker({
          map: this.map,
          animation: google.maps.Animation.DROP,
          position: this.map.getCenter()
        });

        let content = "<h4>Information!</h4>";

        this.addInfoWindow(marker, content);

            }
      addInfoWindow(marker, content){

        let infoWindow = new google.maps.InfoWindow({
          content: content
        });

        google.maps.event.addListener(marker, 'click', () => {
          infoWindow.open(this.map, marker);
        });

      }
```

En el template

```
<button ion-button block (click)="addMarker()">Añadir Marker</button>
```

Geolocalización por dirección postal

Basados en el ejemplo que ofrece el propio google todo se basa en la declaración del objeto de google *let geocoder = new google.maps.Geocoder();* Se le manda la dirección por variable address, geolocaliza y añade marcador en el mapa. En el ejemplo la variable address está definida estática pero podría ser un dataBinding de un elemento de formulario (ngModel)

```
      geocodeAddress() {
              let geocoder = new google.maps.Geocoder();
              var address = 'Carrer dels madrazo, 27, barcelona';
              geocoder.geocode({'address': address}, (results, status) => {
                if (status === google.maps.GeocoderStatus.OK) {
                  this.map.setCenter(results[0].geometry.location);
                  var marker = new google.maps.Marker({
                    map: this.map,
                    position: results[0].geometry.location
                  });
                } else {
                  alert('Geocode was not successful for the following
reason: ' + status);
                }
              });
      }
```

Grupoloyal

Ionic 2, acceder a cámara, parte 1

Toda la info official de Ionic sobre el acceso a la cámara.

```
ionic platform add android

ionic plugin add cordova-plugin-camera

ionic state save
```

Después de este último paso, hay que verificar que ionic ha añadido el permiso en el archivo AndroidManifest.xml, es vital para que la App pueda acceder al recurso.

<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />

Llamada a cámara desde un component.

Vista

```
<ion-content padding>
<button ion-button block (click)="getPicture()">Toma una foto</button>
  <img [src]="image" *ngIf="image" />
</ion-content>
```

Component

```
import { Component } from '@angular/core';
import {Camera} from 'ionic-native';

@Component({
  templateUrl: 'copy-hello-ionic.html'
})
export class CopyHelloIonicPage {
        image: string;
  constructor() {}
  getPicture(){
    let options = {
      destinationType: Camera.DestinationType.DATA_URL,
      targetWidth: 1000,
      targetHeight: 1000,
      quality: 100
    }
    Camera.getPicture( options )
    .then(imageData => {
      this.image = `${imageData}`;
    })
    .catch(error =>{
      alert( error );
    });
  }
}
```

Obtener imagen desde la galería

Para obtener la imagen desde la galería de fotos del dispositivo, basta con añadir el parámetro sourceType : Camera.PictureSourceType.PHOTOLIBRARY en las options.

Ionic 2, acceder a cámara, parte 2

Para poder ejecutar correctamente la App, se necesita un servidor web real para poder acceder a los PHP (API Rest y upload) desde el propio dispositivo y con una base de datos con los campos:

	#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado
☐	1	**id** 🔑	int(255)			No	*Ninguna*
☐	2	**titulo**	varchar(255)			Sí	*NULL*
☐	3	**descripcion**	text			Sí	*NULL*
☐	4	**imagen**	varchar(255)			Sí	*NULL*
☐	5	**latitud**	float(10,7)			No	*Ninguna*
☐	6	**longitud**	float(10,7)			No	*Ninguna*

Ahora, el objetivo es subir archivo de imagen a un servidor, el usuario podrá escoger elegir la imagen desde la galería de fotos o hacer la foto directamente. También podrá añadir la geo-localización.

Primero se prepara la parte de programación PHP que recibe el archivo y lo guardará en carpeta.

Upload.php

```
<?php
//Allow Headers
header('Access-Control-Allow-Origin: *');
//print_r(json_encode($_FILES));
$new_image_name = urldecode($_FILES["file"]["name"]);
//Move your files into upload folder
move_uploaded_file($_FILES["file"]["tmp_name"], "uploads/".$new_image_name);
?>
```

Para poder subir archivos, se necesitan instalar dos plug-ins:

```
ionic plugin add cordova-plugin-file

ionic plugin add cordova-plugin-file-transfer
```

Componente captura

A través de un actionSheet el usuario escoge cómo obtener la imagen con el método chooseImage(), el guardar implica subir la imagen uploadPhoto() y si da el OK al subir guardar la información en la BBDD con addCaptura(). La geolocalización está declarada con el método loadMap() que actúa al hacer clic en el icono declarado en el navbar de la vista.

```
import { Component,ViewChild, ElementRef } from '@angular/core';
import { NavController, LoadingController, ActionSheetController } from
'ionic-angular';
//se importan los módulos de los plug-ins que se acaban de añadir
import { File, Camera, Transfer, Geolocation } from 'ionic-native';

import { HelloIonicPage } from '../hello-ionic/hello-ionic';
import {Captura} from "../../model/captura";
import { PostService } from '../../providers/post-service';

declare var cordova: any;
declare var google: any;

@Component({
  selector: 'page-capturas',
  templateUrl: 'capturas.html',
  providers:[PostService]
})
export class CapturasPage {

  postTitle: any;
  desc: any;
  imageChosen: any = 0;
  imagePath: any;
  imageNewPath: any;
  public buttonDisabled:boolean;
  public captura:Captura;
  public errorMessage: string;
  public status: string;
  public mostrarMapa:boolean;
  @ViewChild('map') mapElement: ElementRef;
  map: any;


  constructor(
    public navCtrl: NavController,
    public actionSheet: ActionSheetController,
    private loadingCtrl: LoadingController,
    private postService: PostService
    )
  {
    this.buttonDisabled=false;
    this.captura = new Captura(0,"","","null",0,0);
    this.mostrarMapa=false;
  }

  ionViewDidLoad() {

  }
```

```
uploadPhoto() {
    let loader = this.loadingCtrl.create({
      content: "Subiendo captura..."
    });
    loader.present();

    let filename = this.imagePath.split('/').pop();
    let options = {
      fileKey: "file",
      fileName: filename,
      chunkedMode: false,
      mimeType: "image/jpg",
      params: { 'title': this.postTitle, 'description': this.desc }
    };

    const fileTransfer = new Transfer();

    fileTransfer.upload(this.imageNewPath,
'http://url_servidor/slim/upload.php',
      options).then((entry) => {
        this.imagePath = '';
        this.imageChosen = 0;
        this.addCaptura();
        loader.dismiss();

      }, (err) => {
        alert(JSON.stringify(err));
      });
  }

  chooseImage() {
    let actionSheet = this.actionSheet.create({
      title: 'Elige origen',
      buttons: [
        {
          text: 'Galería',
          icon: 'albums',
          handler: () => {
            this.actionHandler(1);
          }
        },
        {
          text: 'Cámara',
          icon: 'camera',
          handler: () => {
            this.actionHandler(2);
          }
        },
        {
          text: 'Cancelar',
          role: 'cancel',
          handler: () => {
            console.log('Cancel clicked');
          }
        }
      ]
    });

    actionSheet.present();
```

```
  }


  }

  actionHandler(selection: any) {
    var options: any;

    if (selection == 1) {
      options = {
        quality: 75,
        destinationType: Camera.DestinationType.FILE_URI,
        sourceType: Camera.PictureSourceType.PHOTOLIBRARY,
        allowEdit: true,
        encodingType: Camera.EncodingType.JPEG,
        targetWidth: 500,
        targetHeight: 500,
        saveToPhotoAlbum: false
      };
    } else {
      options = {
        quality: 75,
        destinationType: Camera.DestinationType.FILE_URI,
        sourceType: Camera.PictureSourceType.CAMERA,
        allowEdit: true,
        encodingType: Camera.EncodingType.JPEG,
        targetWidth: 500,
        targetHeight: 500,
        saveToPhotoAlbum: true
      };
    }

    Camera.getPicture(options).then((imgUrl) => {
      var sourceDirectory = imgUrl.substring(0, imgUrl.lastIndexOf('/') + 1);
      var sourceFileName = imgUrl.substring(imgUrl.lastIndexOf('/') + 1,
imgUrl.length);
      sourceFileName = sourceFileName.split('?').shift();
      File.copyFile(sourceDirectory, sourceFileName,
cordova.file.externalApplicationStorageDirectory,
sourceFileName).then((result: any) => {
        this.imagePath = imgUrl;
        this.imageChosen = 1;
        this.imageNewPath = result.nativeURL;
        this.captura.imagen=sourceFileName;
      },
        (err) => {
        alert(JSON.stringify(err));
      })

    }, (err) => {
      alert(JSON.stringify(err))
    });

  }

  addCaptura(){
    this.postService.addCaptura(this.captura).subscribe(
        response => {
          this.status = response.status;
```

```
          if(this.status !== "success"){
            alert("Error en el servidor");
          }
          console.log(this.captura);
          this.navCtrl.setRoot(HelloIonicPage);
        },
        error => {
          this.errorMessage = <any>error;
          if(this.errorMessage !== null){
            alert("Error en la petición");
          }
          this.navCtrl.setRoot(HelloIonicPage);
        }
      );

  }
  loadMap(){
    this.mostrarMapa=true;

    Geolocation.getCurrentPosition().then((position) => {

      let latLng = new google.maps.LatLng(position.coords.latitude,
position.coords.longitude);
      this.captura.latitud=position.coords.latitude;
      this.captura.longitud=position.coords.longitude;
      console.warn(this.captura.latitud);

      let mapOptions = {
        center: latLng,
        zoom: 15,
        mapTypeId: google.maps.MapTypeId.ROADMAP
      }

      this.map = new google.maps.Map(this.mapElement.nativeElement,
mapOptions);
      let marker = new google.maps.Marker({
      map: this.map,
      animation: google.maps.Animation.DROP,
      position: this.map.getCenter()
        });

    }, (err) => {
      console.log(err);
    });
  }
}
```

Vista

A destacar el uso de [hide] de Angular 2 para mostrar el mapa solo cuando se haya producido una geolocalización, que a diferencia de *ngIf, el atributo [hide] oculta la etiqueta en vez de declararla en el código o no como hace el *ngIf, necesario para que la variable #map no de error de “undefined” en el componente.

```
<ion-header>
  <ion-navbar color="navbarColor">
    <ion-title>Nueva Captura</ion-title>
     <ion-buttons end>
      <button ion-button icon-only (click)="loadMap()">
        <ion-icon name="compass"></ion-icon>
      </button>
    </ion-buttons>
  </ion-navbar>
</ion-header>
<ion-content padding>
      <ion-card>
  <ion-card-header>
   <div class="uploadWrap" *ngIf="imageChosen == 0">
<ion-icon (click)="chooseImage()" name="camera"></ion-icon>
    </div>
<div class="img-preview" *ngIf="imageChosen == 1">
<img alt="" src="{{imagePath}}">
    </div>
</ion-card-header>
 <form #formCaptura="ngForm" (ngSubmit)="uploadPhoto()">
  <ion-card-content>
    <ion-list>
  <ion-item>
    <ion-label floating="">Título</ion-label>
    <ion-input #titulo="ngModel" [(ngModel)]="captura.titulo" name="titulo"
type="text" required></ion-input>
  </ion-item>
  <ion-item>
    <ion-label floating="">Descripción</ion-label>
    <ion-input #descripcion="ngModel" [(ngModel)]="captura.descripcion"
name="descripcion" type="text" required></ion-input>
  </ion-item>
  <ion-item>
    <div [hidden]="!mostrarMapa" #map id="map" style="height: 300px;width:
100%"></div>
  </ion-item>
  <ion-item>
   <button [disabled]="imageChosen == 0" type="submit" full ion-
button>Crear</button>
  </ion-item>

</ion-list>
  </ion-card-content>
 </form>
</ion-card>
</ion-content>
```

Servicio

```
import { Injectable } from '@angular/core';
import { Http, Headers } from '@angular/http';
import 'rxjs/add/operator/map';
import {Captura} from "../model/captura";

@Injectable()
export class PostService {
  posts: any;

  constructor(public _http: Http) {

  }

  getPosts() {
   return this._http.get("http://url_servidor/slim/api.php/capturas")
                                                .map(res => res.json());
  }
  addCaptura(captura: Captura) {
    let json = JSON.stringify(captura);
    let params = "json="+json;
    let headers = new Headers({'Content-Type':'application/x-www-form-
urlencoded'});
    return this._http.post("http://url_servidor/slim/api.php/captura-add",
        params, {headers: headers}).map(res => res.json());
  }
}
```

Estilos

```
page-capturas {
ion-card{
    margin: 3px !important;
    width: calc(100% - 6px) !important;
    .uploadWrap{
        ion-icon{
            color:#777;
            font-size: 150px;
            text-align: center !important;
            display: block;
        }
    }
    button{
        height: 40px !important;
    }
}
}
```