

Judith Gutiérrez Campillo

Build a Mobile App with Angular 2 and Ionic 2

Index

Perquè Ionic?.....	3
Què construirem?.....	3
Instal·lar Ionic 2.....	3
Què es TypeScript?.....	3
Hello Ionic 2.....	4
Estructura d'Exemple de la nostra aplicació Ionic.....	7
Components:.....	8
Aplicació amb GitHub.....	13
Visualitzant Usuaris de Github.....	16

Perquè Ionic?

- Utilitza HTML,CSS,JS
- Aprofitar el Progrés d'Aplicacions Web
- Es pot dirigir a totes les plataformes mòbils importants
- La funcionalitat nativa és fàcil d'implementar

Què construirem?

Construirem una aplicació senzilla amb l'API de GitHub. La nostra aplicació mostrarà una llista d'usuaris GitHub, ofereixen un quadre de cerca per als usuaris que busquen, i ser capaç de veure el nombre de seguidors, repos, i els GIST que té l'usuari.

Instal·lar Ionic 2

1. Instal·lem cordova → `npm install -g cordova`
2. Instal·lem Ionic → `npm install -g ionic`
3. Instal·lem Typescript → `npm install -g typescript`

Què es TypeScript?

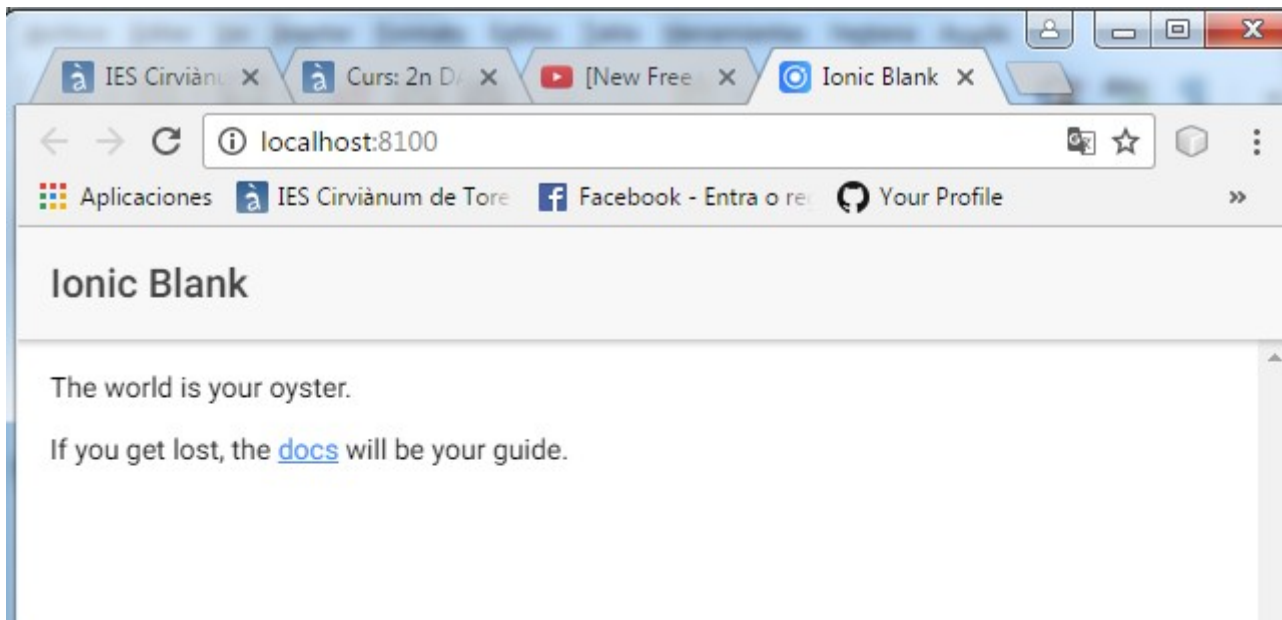
És un superconjunt de JavaScript, i funciona molt bé amb angular 2 .

Hello Ionic 2

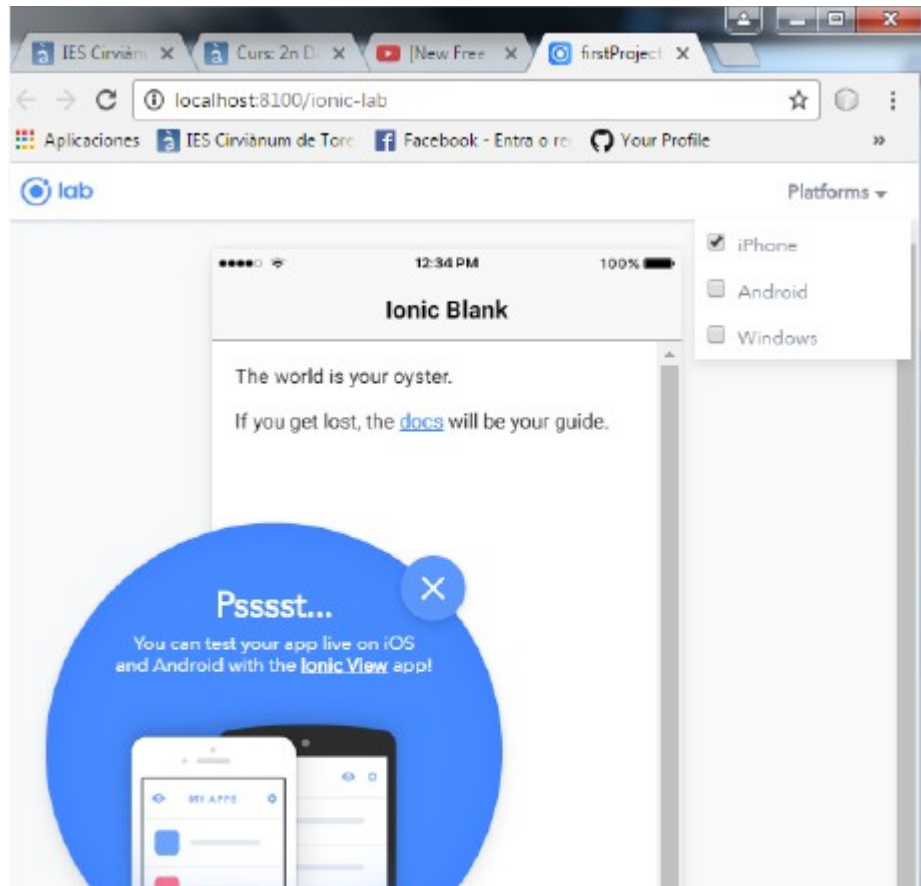
Crear un projecte → Per a crear un projecte obrim un terminal i executem la següent comanda: `'ionic start nomprojecte blank -v2'`

Executar el projecte → Un cop creat el projecte i fet el `'cd nomprojecte'` executem la comanda `'ionic serve'` per a que el projecte s'executi al nostre navegador.

Url → La url del projecte normalment sera `'http://localhost:8100'`



Plataforma → Depenent de la plataforma en que el vulguis obrir utilitzaràs ionic serve (per plataforma de navegador web) o ionic serve -l (per plataformes mòbils).



Estructura → La estructura d'un projecte normalment serà algú similar això:

```
├─ config.xml
├─ hooks
├─ ionic.config.json
├─ node_modules
├─ package.json
├─ platforms
├─ plugins
├─ resources
├─ src
├─ tsconfig.json
├─ tslint.json
```

Src → Nosaltres gairebé sempre passen el 90% de les vegades a la carpeta src, ja que és on està la lògica de l'aplicació. Es carrega des d'un simple fitxer src / index.html, que durant la construcció es copia en una nova carpeta www .

<ion-app> → aquest és el punt d'entrada per a la nostra aplicació iònic. Es diu el component arrel de la nostra aplicació.

<root-component> → tipus d'estructura a la pàgina index.html. Les altres parts d'aquest arxiu s'acaba carregant dependències de construcció iònics.

Config.xml → Aquest conté configuracions com el nom de l'aplicació, i el nom del paquet, que serà utilitzat per a instal·lar la nostra aplicació en un dispositiu real.

node_modules → Conté els paquets del npm package dins el arxius package.json. Aquest són els paquets necessaris per a la construcció de l'aplicació d'ionic.

Platforms → Aquí és on es construeix i emmagatzema la plataforma , les eines i paquets / biblioteques .

Plugins → Aquí és on s'emmagatzemaran els connectors de cordova quan s'instal·lin

resources → Això també conté recursos específics de la plataforma, com ara icones i pantalles d'inici.

src/app/app.component.ts → és el component de l'arrel de la nostra aplicació. Es carrega/declara en **src/app/app.module.ts** que simplement representa tota la nostra aplicació com un mòdul, que és la càrrega en **app/main.dev.ts** o en **app/main.prod.ts** depenent de on ho construeixis. Aquesta estructura suporta Angular 2.

src/app/app.html → punt de vista de l'aplicació en forma d'arrel.

Src/pages → Ionic 2 es sol dividir en components de pàgina que estan continguts en el directori src/pages. Es pot veure la pantalla en el mateix moment que es programa.

Src/themes → carpeta conté fitxers Sass que ajuden en la tematització de l'aplicació.

src/app/app.module.ts → lloc on importem els nostres components arrel que es defineixen a src/app/app.component.ts

Estructura d'Exemple de la nostra aplicació Ionic

Ionic serve: Iniciem l'aplicació.

La nostra aplicació iònic tindrà 3 pàgines principals. Una per a usuaris GitHub, un altre per a les organitzacions de GitHub i un altre per als repositoris de GitHub. Les dues últimes pàgines són només seran vistes per a navegar amb la barra lateral, no tenen cap contingut. La pàgina de GitHub usuaris, però, li permetrà veure els detalls d'usuari.

Per afegir pàgines : 'ionic g page nompagina'

```
PS C:\Users\JudithGC\firstProject> ionic g page users
```

```
PS C:\Users\JudithGC\firstProject> ionic g page repos
```

```
PS C:\Users\JudithGC\firstProject> ionic g page organisations
```

Es suposa que fent això veurem 3 hml nous, semblant això:

src/pages/users/users.html

```
HTML

<ion-header>
  <ion-navbar>
    <button ion-button icon-only menuToggle>
      <ion-icon name="menu"></ion-icon>
    </button>
    <ion-title>
      Users
    </ion-title>
  </ion-navbar>
</ion-header>

<ion-content padding>
  <h3>Users view page</h3>
</ion-content>
```

Components:

<ion-navbar> → és responsable de la barra de navegació.

ion-button → per fer botons.

icon-only → indica que el boto de tipus icona.

menuToggle → és un sistema incorporat a la directiva per ajudar a canviar el menú lateral.

<ion-icon> → és un component responsable del maneig de les icones.

<ion-title> → El nom de la pagina.

<ion-content> → Manté el contingut de la pàgina.

IonViewDidLoad() → mètode que t'indica el cicle de vida hook.

Haurem de afegir les pàgines al nostre navegador, anem a `src/app/app.components.ts` i farem uns canvis.

```
<ion-menu [content]="content">

  <ion-header>
    <ion-toolbar>
      <ion-title>Pages</ion-title>
    </ion-toolbar>
  </ion-header>

  <ion-content>
    <ion-list>
      <button ion-item *ngFor="let p of pages" (click)="openPage(p)">
        {{p.title}}
      </button>
    </ion-list>
  </ion-content>

</ion-menu>

<ion-nav [root]="rootPage" #content swipeBackEnabled="false"></ion-nav>
```

NgFor= 'let p of pages' → bucle a través de la col·lecció de pàgines i generar una plantilla per a cada element de la col·lecció.

<ion-nav> → és on es mostra la pàgina de navegació.

Per afegir les pàgines correctes en el sidenav, farem alguns canvis a l'arxiu `src / app / app.component.ts`. He suprimit dues declaracions d'importació (HelloIonicPage i ListPage) a la part superior de les pàgines hem esborrat, i va afegir declaracions d'importació per a les pàgines que hem creat.

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

import { Platform, MenuController, Nav } from 'ionic-angular';

import { StatusBar } from 'ionic-native';

import { UsersPage } from '../pages/users/users';
import { ReposPage } from '../pages/repos/repos';
import { OrganisationsPage } from '../pages/organisations/organisations';

@Component({
  templateUrl: 'app.html'
})
export class MyApp {
  // Commented Out for Brevity
}
```

El UsersPage, ReposPage i OrganisationsPage són classes de components que representen les pàgines . Llavors anem a editar la propietat pàgines classe perquè coincideixi amb les nostres noves pàgines.

```
export class MyApp {
  @ViewChild(Nav) nav: Nav;

  // make UsersPage the root (or first) page
  rootPage: any = UsersPage;
  pages: Array<{title: string, component: any}>;

  constructor(public platform: Platform, public menu: MenuController) {
    this.initializeApp();

    // set our app's pages
    this.pages = [
      { title: 'Users', component: UsersPage },
      { title: 'Repos', component: ReposPage },
      { title: 'Organisations', component: OrganisationsPage },
    ];
  }

  initializeApp() {
    this.platform.ready().then(() => {
      // Okay, so the platform is ready and our plugins are available.
      // Here you can do any higher level native things you might need.
      StatusBar.styleDefault();
    });
  }

  openPage(page) {
    // close the menu when clicking a link from the menu
    this.menu.close();
    // navigate to the new page if it is not the current page
    this.nav.setRoot(page.component);
  }
}
```

Observa que la propietat de classe `rootPage` s'estableix a l'`UsersPage`. Això significa que el `UsersPage` es mostrarà primer quan es carrega l'aplicació.

Hem canviat el valor de `this.pages` al constructor perquè coincideixi amb les pàgines que hem afegit.

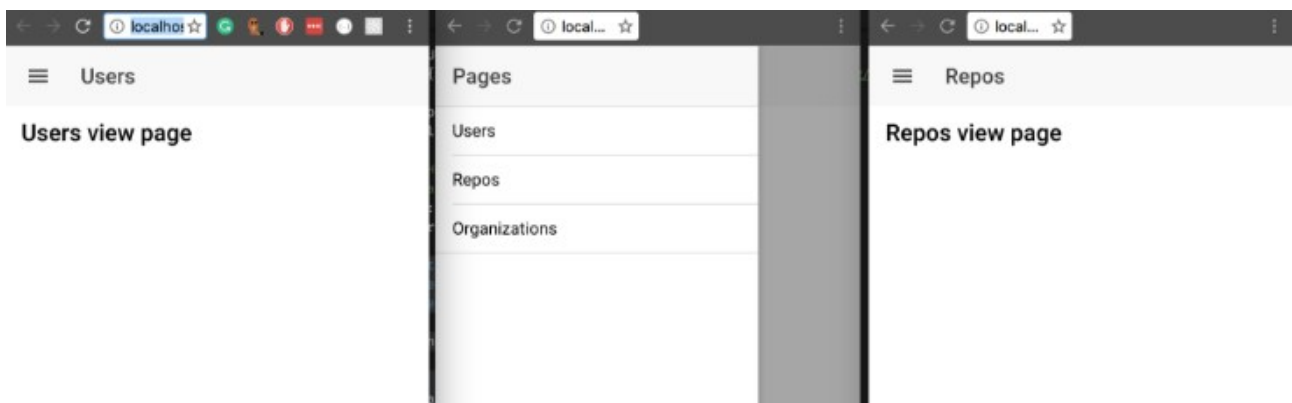
El mètode `OpenPage` és l'encarregat d'obrir les pàgines quan es fa clic.

```
import { NgModule } from '@angular/core';
import { IonicApp, IonicModule } from 'ionic-angular';
import { MyApp } from './app.component';

import { UsersPage } from '../pages/users/users';
import { ReposPage } from '../pages/repos/repos';
import { OrganisationsPage } from '../pages/organisations/organisations';

@NgModule({
  declarations: [
    MyApp,
    UsersPage,
    ReposPage,
    OrganisationsPage
  ],
  imports: [
    IonicModule.forRoot(MyApp)
  ],
  bootstrap: [IonicApp],
  entryComponents: [
    MyApp,
    UsersPage,
    ReposPage,
    OrganisationsPage
  ],
  providers: []
})
export class AppModule {}
```

Per veure com va la aplicació l'executem amb la comanda 'ionic serve -l', hauria de sortir algú semblant això :



Aplicació amb GitHub

Crearem un servei per fer que els usuaris de Github <https://api.github.com/users>. La pàgina mostra prop de 30 dels primers usuaris de GitHub en format JSON.

Hem de crear un model d'usuari Github. Aquesta és una classe que conté els camps rellevants que volem per a un usuari GitHub, des github ofereix una gran quantitat de detalls.

Crearem una carpeta anomenada models en la carpeta src. Aquí és on posarem el nostre model d'usuari i altres models. Dins src / models i afegir un arxiu user.ts.

```
// User model based on the structure of github api at
// https://api.github.com/users/{username}
export interface User {
  login: string;
  avatar_url: string;
  public_repos: number;
  public_gists: number;
  followers: number;
  following: number;
}
```

Per generar un proveïdor cal executar la següent comanda en el terminal :
‘ionic g provider github-users’.

Això crea una carpeta anomenada proveïdors en el directori src, i un arxiu de GitHub-users.ts.

Hem d'afegir un mètode perquè els usuaris GitHub a l'arxiu github-users.ts generat.

Anem a modificar lleugerament el fitxer src / proveïdors generats / github-users.ts

```
import { Injectable } from '@angular/core';
import { Http } from '@angular/http';
import { Observable } from 'rxjs/Rx';
import 'rxjs/add/operator/map';

import { User } from '../models/user';

@Injectable()
export class GithubUsers {
  githubApiUrl = 'https://api.github.com';

  constructor(public http: Http) { }

  // Load all github users
  load(): Observable<User[]> {
    return this.http.get(`${this.githubApiUrl}/users`)
      .map(res => <User[]>res.json());
  }
}
```

@Injectable() → és la forma de angular 2 per declarar els serveis /proveïdors.

Observable es necessari perquè retornem els resultats de la crida de l'API de github com a Observable.

```
// Imports commented out for brevity

import { GithubUsers } from '../providers/github-users';

@NgModule({
  declarations: [
    // Declarations commented out for brevity
  ],
  imports: [
    IonicModule.forRoot(MyApp)
  ],
  bootstrap: [IonicApp],
  entryComponents: [
    // Entry Components commented out for brevity
  ],
  providers: [GithubUsers] // Add GithubUsers provider
})
export class AppModule {}
```


Visualitzant Usuaris de Github

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

import { User } from '../../models/user';

import { GithubUsers } from '../../providers/github-users';

@Component({
  selector: 'page-users',
  templateUrl: 'users.html'
})
export class UsersPage {
  users: User[]

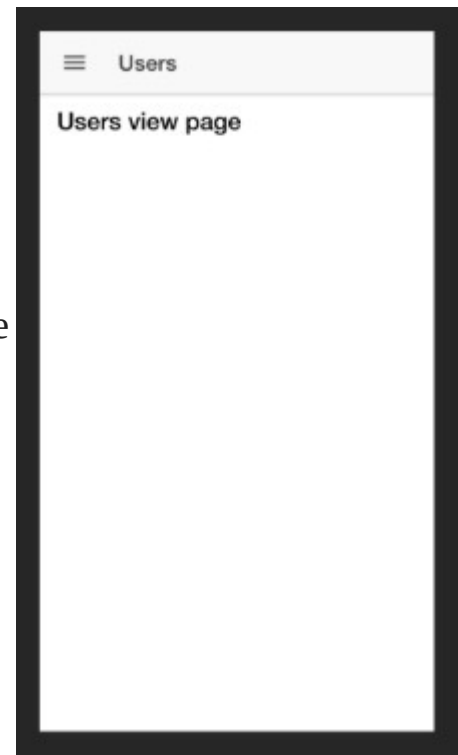
  constructor(public navCtrl: NavController, private githubUsers: GithubUsers) {
    githubUsers.load().subscribe(users => {
      console.log(users)
    })
  }
}
```

En primer lloc, hem importat el proveïdor GithubUsers a la part superior amb la importació `{ GithubUsers }` de `../../providers/github-users` '. També hem importar el model de l'usuari.

En el constructor de la nostra UsersPage, hem afegit `githubUsers: GithubUsers` com un dels paràmetres. Aquesta és la forma en què injectem en dependències angular 2. Després cridem a la funció de càrrega al constructor i registrar el resultat amb `console.log`.

Heu de veure que una matriu d'objectes s'ha connectat a la nostra consola.

Per veure els usuaris a la nostra pàgina, ens cal una variable local a la classe `UserPage` que serà un conjunt d'usuaris. És per això que hem afegit als usuaris: `Usuari []`, just després de la declaració de la classe. Anem a assignar-li el `Reponse` el de l'anomenada de servei `githubUsers`.



src/pages/users/users.ts

```
@Component({
  // Commented out for brevity
})
export class UsersPage {
  users: User[]

  constructor(public navCtrl: NavController, private githubUsers: GithubUsers) {
    githubUsers.load().subscribe(users => {
      this.users = users;
    })
  }
}
```

src/pages/users/users.html











```
<ion-header>
  <!-- ion-header contents commented out for brevity -->
</ion-header>

<ion-content padding>
  <ion-list>
    <button ion-item *ngFor="let user of users">
      <ion-avatar item-left>
        <img [src]="user.avatar_url">
      </ion-avatar>
      <h2>{{ user.login }}</h2>
      <ion-icon name="arrow-forward" item-right></ion-icon>
    </button>
  </ion-list>
</ion-content>
```

ion-list : s'utilitza per representar llistes en ionic. Els elements de la llista tenen una directiva d'element que anem a recórrer amb 2 angular incorporat a la directiva de plantilla * ngFor. Ens recórrera tots els usuaris amb * ngFor = "deixar a l'usuari dels usuaris". usuaris aquí es refereix als usuaris de la propietat de classe UsersPage.

A continuació, fem servir propietat d'enllaç per carregar l'avatar dins ion-avatar, que només afegeix una vora arrodonit a la imatge. Recordeu que el nostre model d'usuari tenia una propietat avatar_url, d'aquí el user.avatar_url.

A continuació, afegir nom d'usuari github l'usuari, que en aquest cas és {{ }} user.login.

Users		
	mojombo	→
	defunkt	→
	pjhyett	→
	wycats	→
	ezmobius	→
	ivey	→
	evanphx	→
	vanpelt	→
	wayneeseguin	→
	brynary	→