Ionic Angular

Cross Device Framework

Requisiti

Per poter lavorare con Ionic Angular, è necessario procedere con l'installazione tramite npm di alcuni moduli.

npm install -g @ionic/cli native-run cordova-res

PWA elements

Le **PWA** sono applicazioni web che offrono un'esperienza simile a quella di un'app nativa, con funzionalità avanzate come:

- Installazione (possono essere aggiunte alla home screen)
- Funzionamento offline (grazie ai Service Worker)
- Notifiche push
- Accesso a funzionalità hardware (come fotocamera, geolocalizzazione)
- Caricamento veloce

Cosa sono i PWA Elements?

- Sono una raccolta di librerie, componenti UI e strumenti che semplificano lo sviluppo di PWA, spesso integrati in framework come:
- @ionic/pwa-elements (per Ionic Framework)
- Componenti pronti all'uso per aggiungere funzionalità PWA (es. modali per la fotocamera, notifiche).
- Facilita l'uso di API come Camera, File System, Geolocation in modo cross-platform.

Cordova e Capacitor

Cordova e **Capacitor** sono i framework più famosi per riuscire a creare un wrapper di una web application all'interno di un container nativo.

Questo permette l'utilizzo delle funzionalità e delle periferiche del device (fotocamera, GPS, file system) richiamate attraverso l'uso di API native.

Cordova è un po' più datato ma comunque ancora utilizzato, mentre Capacitor è stato sviluppato dal team Ionic ed è compatibile con TypeScript. Inoltre, la grande differenza è che il primo deve avere una build per ogni piattaforma il secondo no.

Creazione di un nuovo progetto

Per la creazione di un nuovo progetto è necessario inserire il comando da terminale:

ionic start nome_progetto tabs --type=angular –capacitor

In fase di creazione viene inoltre richiesto se il Progetto che stiamo andando a creare deve utilizzare **NgModules** oppure **Standalone**.

Possiamo inoltre decider la struttura base del Progetto, in questo caso “tabs”, indica che avremo i pulsanti in basso per la navigazione.

Creazione di una nuova pagina

Ionic ha una gestione molto simile a quella di Angular, tuttavia la principale differenza è che si possono usare **page** che vengono visualizzate e occupano tutto lo spazio, inoltre possono essere raggiunte tramite le routes. Essendo elementi creati per essere completamente compatibili con Ionic implementano i life cycle hook (ionViewWillEnter, ionViewDidLeave,ecc…).

Per creare una pagina si usa il comando nel terminale:

ionic generate page nome_page

Creazione di un Componente in Ionic

Vengono utilizzati con una visione di codice modulare e riutilizzabile, ma non essendo built-in ma derivati da Angular, non possono essere raggiunti tramite routes e usare i life cycle hooks di Ionic ma quelli Angular.

In questo contesto vengono utilizzati per creare parte delle interfacce riutilizzabili, infatti vengono usate per generare Modal, Card,ecc…

Per creare un componente, si usa il comando:

ionic generate component nome_componente

Invocare API

Come visto in Angular, la buona prassi richiede l'implementazione di un service in cui vengono implementate le varie chiamate dei servizi.

LoginService.ts

```
constructor(private http: HttpClient) {}

apiUrl = 'http://localhost:5000/api/login';

login(email: string, password: string): Observable<any> {
  return this.http.post(this.apiUrl, { email, password });
}
```

LoginPage.ts

```
login() {
  this.authService.login(this.username, this.password).subscribe({
    next: () => {
      console.log('Login effettuato con successo!');
      this.router.navigateByUrl('/home');
    },
    error: async (err:any) => {
      console.error('Errore durante il login:', err);
      this.router.navigateByUrl('/login');
    }
}
```

Installare Capacitor

Fondamentale è l'installazione di capacitor e dei plugin specifici per l'utilizzo delle varie funzionalità.

- npm i @capacitor/core
- npm i -D @capacitor/cli
- npm install @capacitor/filesystem @capacitor/preferences
- npm install @capacitor/camera

Leaflet

npm install leaflet @types/leaflet

Possiamo installare e utilizzare all'interno del nostro progetto, una mappa gratuitamente. I servizi più noti come Google Maps, richiedono l'utilizzo di specifiche API, sbloccabili solo con l'attivazione di una API KEY che ha un costo per ogni chiamata effettutata.

Gestione della grid come bootstrap

Ionic nativamente permette di gestire gli elementi con una grid così come definito da Bootstrap.

Breakpoint	Sigla	Larghezza minima (px)	Esempio d'uso
Extra Small	xs	< **576px** (default)	Smartphone piccoli
Small	sm	≥ **576px**	Smartphone grandi
Medium	md	≥ **768px**	Tablet (portrait)
Large	lg	≥ **992px**	Tablet (landscape)
Extra Large	xl	≥ **1200px**	Desktop piccoli
Super Large	xxl	≥ **1400px**	Desktop grandi

Gestione delle grid come bootstrap

Ionic offre la possibilità di gestire anche le dimensioni per i singoli dispositivi esplicitando la dimensione per i singoli device:

```
<ion-col size="12" size-md="6">
```

Size definisce il valore predefinito per xs, mentre esplicitiamo per md.

Ovviamente possiamo gestire la visualizzazione per tutti le dimensioni:

```
<ion-col size="12" size-sm="6" size-md="4" size-lg="3" size-xl="2">
```

Gestione della grid come bootstrap

Inoltre è possibile definire uno spostamento verso destra utilizzando l'attributo offset.

offset-<dimensione>="<numero>"

In questo esempio:

<ion-col size="12" size-md="6" offset-md="3">

Solo su tablet il contenuto occupa 6 colonne ma verrà "spostato" di 3 colonne.

Gestione della grid come bootstrap

Possiamo controllare l’allineamento verticale e orizzontale come visto nelle sezione CSS.

Allineamento verticale (ion-align-items-*)

Allineamento orizzontale (ion-justify-content-*)

Esempio:

```
<ion-row class="ion-justify-content-start"><!-- Allinea a sinistra -->
<ion-col size="4">Colonna 1</ion-col>
</ion-row>
```

Build dell’app su Android

- Bisogna eseguire la *ionic build* che crea la cartella *./www*
- Il comando *npx cap init* per inizializzare Capacitor
- ionic cap add android serve per aggiungere la piattaforma Android.
- Usare il comando npx cap sync per aggiornare Plugin o configurazioni.
- npx cap open android apre Android Studio per il test o la build.

Build su iOS

- **macOS**
- **Xcode** installato (da App Store)
- Node.js + npm, Capacitor + Ionic CLI
- **Account Apple** (per test su dispositivo o pubblicazione su App Store)
- **CocoaPods** (per gestire dipendenze native)

Il resto dei comandi resta uguale a prima, con la differenza di:

- *npx cap add ios*
- *npx cap open ios*

Build Desktop

Bisogna installare il pacchetto Electron:

npm install @capacitor/electron –save

Aggiungere la piattaforma electron

npx cap add @capacitor/electron

Utilizzo gli stessi comandi, poi apro con *npx cap open @capacitor/electron*

Per avviarlo:

cd electron

npm install

npm run electron:start

Build per Desktop

È necessario installare

npm install electron-packager --save-dev

Per la build su Windows o MacOs:

npx electron-packager . myApp --platform=win32 --arch=x64

npx electron-packager . myApp --platform=darwin --arch=x64

Permessi per la geolocalizzazione

Permessi (Android/iOS)

Android: Aggiungi in android/app/src/main/AndroidManifest.xml:

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

iOS: Configura in ios/App/App/info.plist:

- <key>NSLocationWhenInUseUsageDescription</key>
- <string>Per mostrare la tua posizione sulla mappa</string>