TP IONIC

MovieDB

Introduction

Pourquoi choisir Ionic?Le développement d'applications mobiles natives est un casse-tête: plusieurs langages de programmation, plusieurs plate-formes et plusieurs stores.Se lancer dans le développement d'une application mobile native sur plusieurs plate-forme demande donc une quantité de ressource et de temps considérable difficile d'accès pour une start-up ou une PME.Une alternative est donc Ionic.

-Étape 1 : initialisation du projet

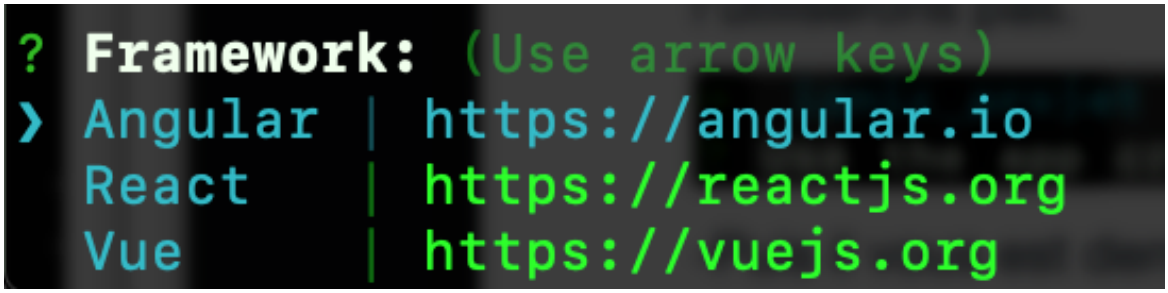
Pour configurer un projet vide, vous pouvez utiliser l'interface de ligne de **commande Ionic** afin que nous nous retrouvions avec un nouveau projet Ionic avec prise en charge d'Angular (Nous pouvons également utiliser React ou Vue).
Nous créerons de nouvelles pages et un service pour notre application que nous voulons utiliser plus tard.

```
npm install -g @ionic/cli
```

L'initialisation d'une premier application est très simple, il suffit d'exécuter la commande suivante dans votre terminal, bien sûr dans le dossier choisis à cette effet au préalable.

```
ionic start
```

Puis il vous est demandé de choisir entre different Framework JS, nous choisirons Angular par exemple.



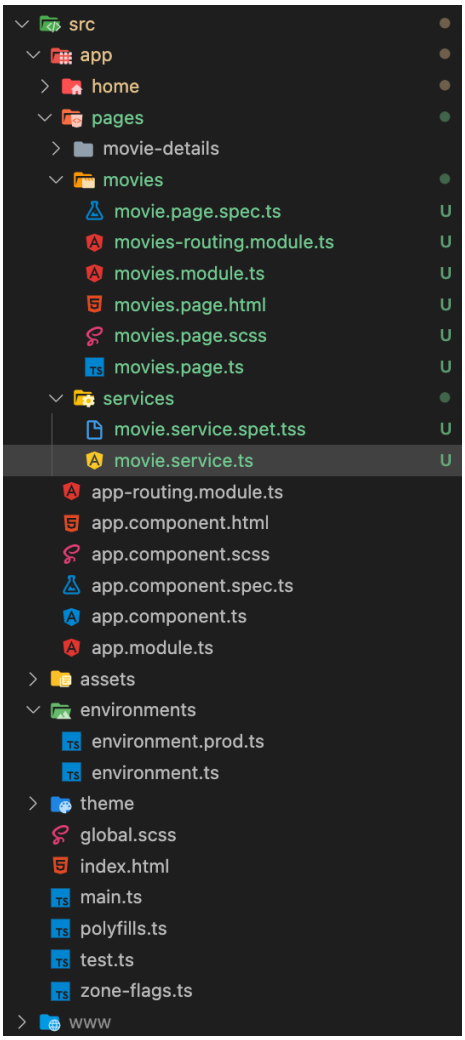
Ce projet utilise désormais automatiquement **Capacitor** et non plus Cordova, mais plus d'informations à ce sujet dans le dernier chapitre.

Vous pouvez maintenant faire apparaître directement votre application en exécutant la commande suivante dans votre projet :

```
ionic serve
```

Cela ouvrira le navigateur avec l'aperçu de votre application qui se rechargera automatiquement une fois que vous aurez modifié quoi que ce soit à l'intérieur de votre projet.

```
android             karma.conf.js     src                www
angular.json        node_modules      tsconfig.app.json
capacitor.config.ts package-lock.json tsconfig.json
ionic.config.json   package.json      tsconfig.spec.json
```



Base du projet

-Dossier WWW

Le dossier Src contient tout le code non compilé de votre application. Le dossier www contient la version compilée de votre code. Chaque fois que vous construisez votre projet, le contenu www est reconstruit. Si vous souhaitez déployer votre

application sur le Web, le code à l'intérieur du dossier www peut être utilisé.

-Dossier Src

Ce dossier est l'endroit où les développeurs font la plupart de leur travail. Lorsque vous naviguez dans le dossier src, vous pourrez voir 4 autres dossiers à l'intérieur du dossier src. Il s'agit **d'un dossier d'application, d'un dossier d'actifs, d'un dossier de pages, d'un dossier de thème.(app folder, assets folder, pages folder, theme folder).**

-Routage

Dans les applications Ionic Angular, nous utilisons le routage Angular pour naviguer vers différentes pages de notre application.

Dans notre application, nous avons besoin de 2 itinéraires :

- **/Listmovie** - Accédez à notre première page qui devrait afficher une liste de films tendance
- **/movies/:id** - Nous montrons les détails d'un film avec la référence id du film.

Pour configurer nos informations de routage, ouvrez notre **src/app/app-routing.module.ts**

```
const routes: Routes = [
  {
    path: '',
    redirectTo: 'listmovie',
    pathMatch: 'full',
  },
  {
    path: 'listmovie',
    loadChildren: () =>
      import('./pages/movies/movies.module').then((m) =>
m.MoviesPageModule),
  },
  {
    path: 'movies/:id',
    loadChildren: () =>
      import('./pages/movie-details/movie-
details.module').then(
        (m) => m.MovieDetailsPageModule
      ),
  },
];
```

Nous devons également appliquer un autre changement à notre application car nous voulons effectuer **des appels HTTP** et devons donc importer un autre module Angular pour faire ces demandes.

Nous devons ajouter un module HttpClientModule dans le fichier **src/app/app.module.ts.**

```
import { HttpClientModule } from '@angular/common/http';

@NgModule({
  declarations: [AppComponent],
  imports: [
BrowserModule,
IonicModule.forRoot(),
AppRoutingModule
HttpClientModule],
  providers: [{ provide: RouteReuseStrategy, useClass:
IonicRouteStrategy }],
  bootstrap: [AppComponent],
})
```

Apres avoir créé un compte sur le site web MovieDB
-Puis dans **/environments/environment.ts**
on insère les éléments d'authentification et les endpoints(url pour l'exploitation d'api):

```
export const environment = {
  production: false,
  apiKey: '', // <-- Entrez votre clé API'
  baseUrl: 'https://api.themoviedb.org/3',
  images: 'https://image.tmdb.org/t/p',
};
```

-API Request

Un service Angular est simplement une classe que vous pouvez **injecter dans d'autres composants** qui donne accès à certaines méthodes que vous définissez.

Dans notre cas, nous voulons préparer deux fonctions pour appeler l'API.
Dans votre fichier **src/app/services/movie.service.ts** :

```
export interface ApiResult {
  page: number;
  results: any[];
  total_pages: number;
  total_results: number;
}

@Injectable({
  providedIn: 'root',
})
export class MovieService {
  constructor(private http: HttpClient) {}

  getTopRatedMovies(page = 1): Observable<ApiResult> {
    return this.http.get<ApiResult>(
      `${environment.baseUrl}/movie/popular?page=${page}
&api_key=${environment.apiKey}`
    );
  }

  getMovieDetails(id: string): Observable<any> {
    return this.http.get<ApiResult>(
      `${environment.baseUrl}/movie/${id}?api_key=$
{environment.apiKey}`
    );
  }
}
```

-Liste des films tendance

Pour cela nous devrons intervenir dans src/app/pages/movies/movies.page.ts pour charger la liste, et les film à chaque fin de scroll.

```
@Component({
  selector: 'app-movies',
  templateUrl: './movies.page.html',
  styleUrls: ['./movies.page.scss'],
})
export class MoviesPage implements OnInit {
  movies = [];
  currentPage = 1;
  imageBaseUrl = environment.images;

  constructor(
    private movieService: MovieService,
    private loadingCtrl: LoadingController
  ) {}

  ngOnInit() {
    this.loadMovies();
  }

  async loadMovies(event?: InfiniteScrollCustomEvent) {
    const loading = await this.loadingCtrl.create({
      message: 'Loading..',
      spinner: 'bubbles',
    });
    await loading.present();

    this.movieService.getTopRatedMovies(this.currentPage).su
bscribe(
      (res) => {
        loading.dismiss();
        this.movies.push(...res.results);

        event?.target.complete();
        if (event) {
```

```
          event.target.disabled = res.total_pages ===
this.currentPage;
        }
      },
      (err) => {
        console.log(err);
        loading.dismiss();
      }
    );
  }

  loadMore(event: InfiniteScrollCustomEvent) {
    this.currentPage++;
    this.loadMovies(event);
  }
}
```

Nous avons besoin d’un pattern pour l’affichage dont voici le code src/app/pages/movies/movies.page.html:

```
<ion-header>
  <ion-toolbar color="primary">
    <ion-title>Trending Movies</ion-title>
  </ion-toolbar>
</ion-header>

<ion-content>
  <ion-list>
    <ion-item button *ngFor="let item of
movies" [routerLink]="[item.id]">
      <ion-avatar slot="start">
        <img [src]="imageBaseUrl + '/w92' +
item.poster_path" />
      </ion-avatar>

      <ion-label class="ion-text-wrap">
```

```
        <h3>{{ item.title }}</h3>
        <p>{{ item.release_date | date:'y' }}</p>
      </ion-label>

      <ion-badge slot="end"> {{ item.vote_average }} </ion-
badge>
    </ion-item>
  </ion-list>

  <ion-infinite-scroll (ionInfinite)="loadMore($event)">
    <ion-infinite-scroll-content loadingSpinner="bubbles"
loadingText="Loading more data..."> </ion-infinite-scroll-
content>
  </ion-infinite-scroll>
</ion-content>
```

-Création d'une page de détails

Après avoir extrait l'ID des paramètres, nous pouvons faire un autre appel à notre service (que nous avons injecté à nouveau par le constructeur) et obtenir les informations détaillées sur le film pour n'importe quel ID que nous avons obtenu.

Dans **src/app/pages/movie-details/movie-details.page.ts** :

```
@Component({
  selector: 'app-movie-details',
  templateUrl: './movie-details.page.html',
  styleUrls: ['./movie-details.page.scss'],
})
export class MovieDetailsPage implements OnInit {
  movie = null;
  imageBaseUrl = environment.images;

  constructor(
```

```
    private route: ActivatedRoute,
    private movieService: MovieService
  ) {}

  ngOnInit() {
    const id = this.route.snapshot.paramMap.get('id');
    this.movieService.getMovieDetails(id).subscribe((res) =>
{
      this.movie = res;
    });
  }

  openHomepage(url) {
    window.open(url, '_blank');
  }
}
```

Dans src/app/pages/movie-details/movie-details.page.html :

```
<ion-header>
  <ion-toolbar color="secondary">
    <ion-buttons slot="start">
      <ion-back-button defaultHref="/movies"></ion-back-
button>
    </ion-buttons>
    <ion-title>{{ movie?.title }}</ion-title>
  </ion-toolbar>
</ion-header>

<ion-content>
  <ion-card *ngIf="movie as movie;">
    <img [src]="imageBaseUrl + '/w400' + movie?.poster_path"
/>

    <ion-card-header>
      <ion-card-title> {{ movie.title }} </ion-card-title>
      <ion-card-subtitle> {{ movie.tagline }} </ion-card-
subtitle>
      <ion-text color="tertiary">
```

```
        <span *ngFor="let g of movie.genres;  let isLast =
last"> {{ g.name }} {{ !isLast ? '·' : '' }}</span>
      </ion-text>
    </ion-card-header>
    <ion-card-content>
      <ion-label color="medium">{{ movie.overview }}</ion-
label>

      <ion-item lines="none">
        <ion-icon name="calendar-outline" slot="start"></
ion-icon>
        <ion-label>{{ movie.release_date | date: 'y'}}</ion-
label>
      </ion-item>

      <ion-item lines="none">
        <ion-icon name="cash-outline" slot="start"></ion-
icon>
        <ion-label>{{ movie.budget| currency: 'USD' }}</ion-
label>
      </ion-item>
    </ion-card-content>
  </ion-card>
</ion-content>

<ion-footer>
  <ion-button
expand="full" (click)="openHomepage(movie.homepage)"
*ngIf="movie?.homepage">
    <ion-icon name="open-outline" slot="start"></ion-icon>
    Open Homepage
  </ion-button>
</ion-footer>
```