Routing et Navigation avec Angular

27/06/2020

ganatan.com

Nous allons **intégrer** le routing dans notre **Application Web**.

Nous utiliserons pour cela le framework javascript **Angular version 10.0.1**

Il dispose de la librairie **angular@routeur** qui nous permettra de gérer la mise en place du routing.

Comment le faire ?

Pour réaliser ce routage voici un résumé de ce que nous allons faire

- **Avant de Commencer**
  Qu'est ce que le routing ou routage.
- **Création de notre projet Angular**
  Nous utiliserons un projet existant contenant les fonctionnalités essentielles.
  Le projet a été généré avec Angular CLI.
- **Structure du projet**
  Avant d'intégrer nos modifications il nous faut choisir une structure.
- **Initialisation du projet**
  En utilisant angular Cli
- **Effectuer les Tests**
  Unitaires et end to end.
- **Code source**
  Le code complet du projet sur github.

Avant de commencer

Un site web est constitué d'une multitude de pages.
Ces pages sont écrites grâce notamment au langage **HTML**.
HTML signifie **HyperText Markup Language**.

L'**hypertexte** est la technologie qui permettra de relier une page à d'autres pages via des **hyperliens**.

Le **Routing** est le mécanisme qui permet de naviguer d'une page à une autre sur un site web.

Par exemple si vous tapez ces deux url dans votre navigateur.

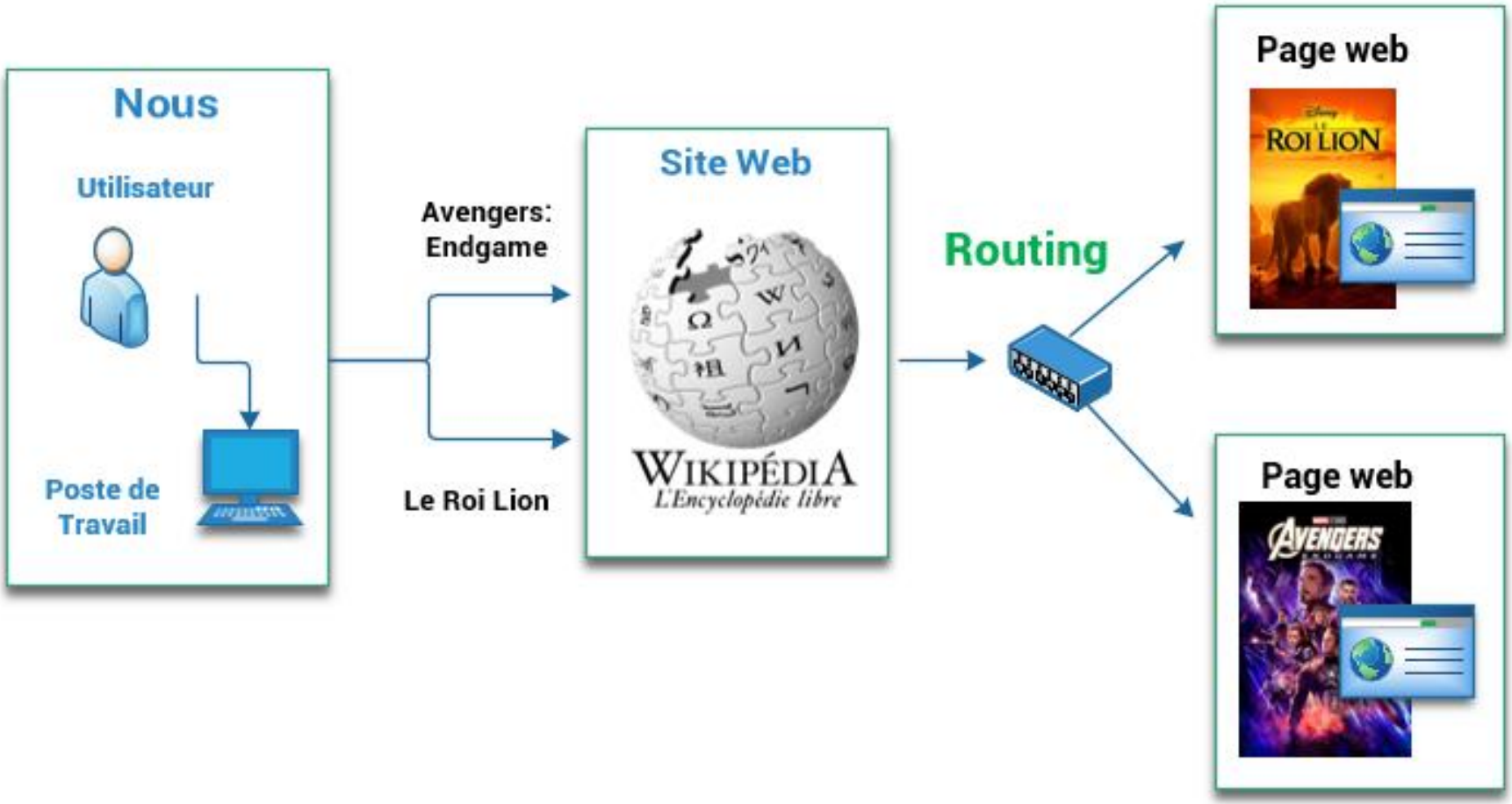
- https://fr.wikipedia.org/wiki/Le_Roi_lion
- https://fr.wikipedia.org/wiki/Avengers:_Endgame

En fonction du nom de film indiqué dans l'url, l'application Web de Wikipedia va déterminer le traitement à effectuer.
Ce traitement permettra d'afficher la page web correspondante au film demandé (ici **Le_Roi_lion** ou **Avengers:_Endgame**).

C'est ce que l'on appelle le **Routage** (traduit littéralement en anglais par **Routing**).

Si nous devions faire une comparaison c'est un peu comme un **aiguilleur** ou un **chef de gare**.
Sans Routage on ne sait plus on va.

Le schéma suivant illustre notre exemple de wikipedia.



Vous l'aurez compris sans routage pas d'application Web.

Ce principe de fonctionnement étant omniprésent dans l'utilisation d'un site web, il nous faut l'appliquer au plus vite dans tout projet web.

Nous allons donc implémenter ce mécanisme dans notre projet Angular.

Création du projet Angular

Pour pouvoir continuer ce tutoriel nous devons bien evidemment disposer de certains éléments

- **Node.js** : La plateforme javascript
- **Git** : Le logiciel de gestion de versions.
- **Angular CLI** : L'outil fourni par Angular.
- **Visual Studio code** : Un éditeur de code.

Vous pouvez consulter le tutoriel suivant qui vous explique en détails comment faire

- https://www.ganatan.com/tutorials/demarrer-avec-angular

Dans ce tutoriel nous allons utiliser un projet existant dont les caractéristiques sont

- Genéré avec Angular CLI

```
# Créez un répertoire demo (le nom est ici arbitraire)
mkdir demo

# Allez dans ce répertoire
cd demo

# Récupérez le code source sur votre poste de travail
git clone https://github.com/ganatan/angular-example-starter

# Allez dans le répertoire qui a été créé
cd angular-example-starter

# Exécutez l'installation des dépendances (ou librairies)
npm install

# Exécutez le programme
npm run start

# Vérifiez son fonctionnement en lançant dans votre navigateur la commande
http://localhost:4200/
```

Structure du projet

La question de la structure de notre projet se pose très rapidement.
Cette structure conditionnera les modifications que nous ferons au fur et à mesure du développement du projet.

Angular nous donne certains conseils dans sa documentation officielle

- Notamment concernant le choix de l'architecture
  https://angular.io/guide/architecture

- Et la structure des fichiers
  https://angular.io/guide/file-structure

Lors de la création du projet avec Angular CLI la structure créée par défaut est la suivante

```
|-- e2e/
|-- src/
    |-- app
    |-- assets
    |-- environnement
package.json
```

Nous allons suivre cette façon de procéder en imaginant les éléments qui pourraient constituer notre projet.

```
|-- e2e/
|-- src/
    |-- app
        |-- components
        |-- directives
        |-- mocks
        |-- modules
        |-- pipes
        |-- services
    |-- assets
    |-- environnement
package.json
```

Comme nous le verrons dans le tutoriel concernant le **lazy loading**, un projet Angular fonctionnera sur la base de **modules.**
Cette partie importante sera construite de la façon suivante.

Le choix que nous avons fait est arbitraire, votre structure pourra prendre n'importe quelle autre forme.

Ce choix sera néanmoins suivi dans les tutoriels suivants.

- **modules/general** regroupera les éléments que l'on retrouve dans tous les projets.

- **modules/application** regroupera les éléments spécifiques à une application.

  Pour créer un nouveau projet il suffira de copier le projet de base et d'adapter modules/application à nos besoins.

```
|-- app
    |-- components
    |-- directives
    |-- mocks
    |-- modules
        |-- general   (contient les éléments que l'on retrouve dans tous les types de projet)
            |-- not-found
            |-- home
            |-- contact
            |-- login
            |-- signin
        |-- application  (contient les éléments spécifiques à notre projet)
            |-- item01
            |-- item02
            |-- ....
            |-- itemXX
    |-- pipes
    |-- services
|-- assets
|-- environnement
```

Initialisation

On va vérifier que Node.js fonctionne et qu'il permet d'exécuter un programme javascript.
Un rapide tour d'horizon avant de voir les détails techniques

Angular propose une librairie **angular@routeur** qui permet de gérer le routing.

Nous importerons cette librairie dans un **module dédié** au routage.
Ce module portera le nom suivant **app-routing.module.ts**

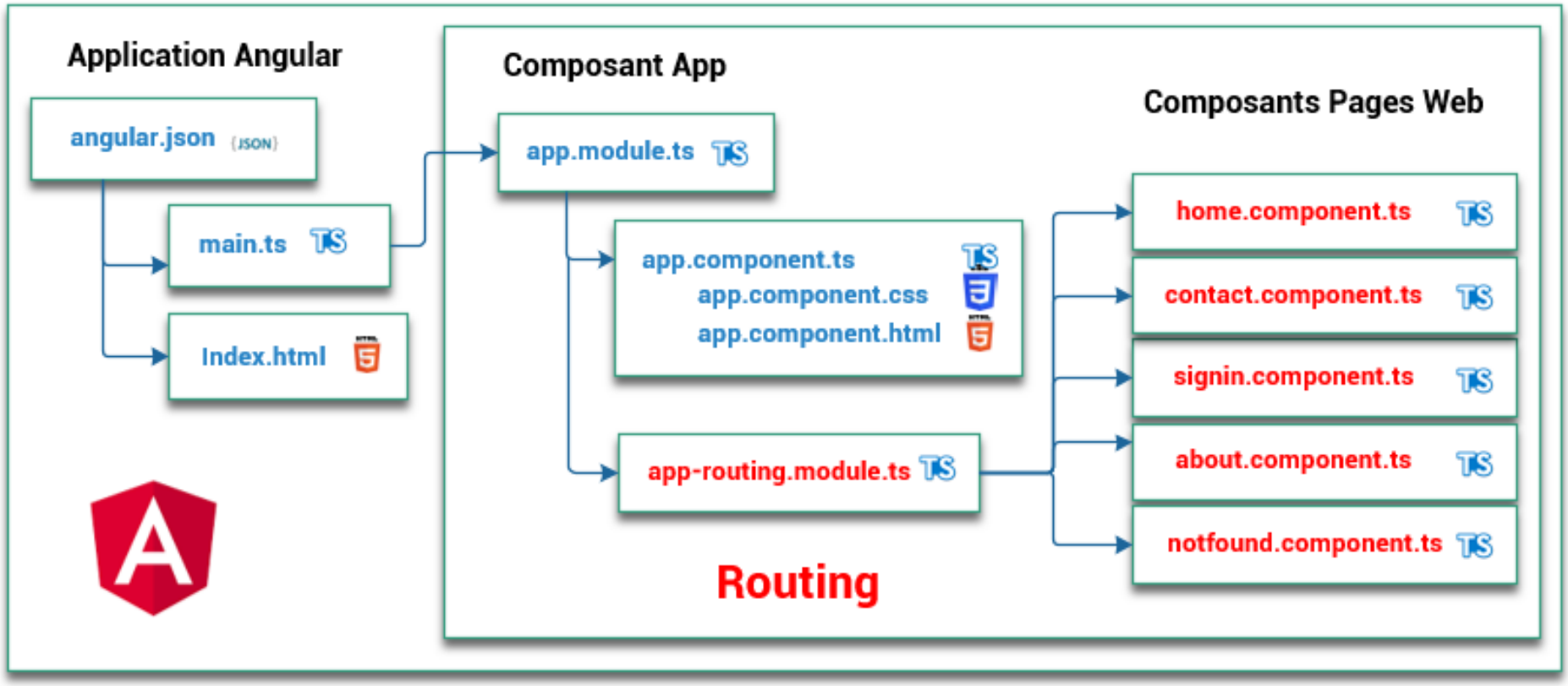
La **configuration du "Routing"** sera transmise au module **RouterModule**

Tout au long de ce didacticiel nous essaierons d'utiliser les commandes que propose angular-cli pour genérer automatiquement le code source nécessaire.

Tout d'abord nous allons créer cinq composants qui seront utilisés dans le routing.
Un composant correspondra à une page Web.
Ce sont cinq pages Web classiques que l'on retrouve dans tout site web.

- **Home**
- **Contact**
- **About**
- **Signin**
- **NotFound**

Un schéma pour récapituler ce que nous allons faire.



Nous utiliserons angular CLI pour cela.

Le tutoriel suivant vous donnera des informations sur les commandes Angular CLI
https://www.ganatan.com/tutorials/demarrer-avec-angular-cli

Exécutons les commandes suivantes.

ng g** correspond à la commande **ng generate

```
# Création des nouveaux composants
ng generate component modules/general/home --module=app
ng generate component modules/general/contact --module=app
ng generate component modules/general/about --module=app
ng generate component modules/general/signin --module=app
ng generate component modules/general/not-found --module=app

# Création des nouveaux composants (méthode 2)
ng g c modules/general/home --module=app
ng g c modules/general/contact --module=app
ng g c modules/general/about --module=app
ng g c modules/general/signin --module=app
ng g c modules/general/not-found --module=app
```

Les répertoires **modules** et **modules/general** sont créés par la commande éxécutée.
Tous les fichiers relatifs à chaque composant sont créés automatiquement par angular CLI.

Par exemple pour le composant **Home** 4 fichiers sont créés

- **home.component.css** (code CSS dédié au design)
- **home.component.html** (code HTML)
- **home.component.spec.ts** (code de test unitaire du composant)
- **home.component.ts** (partie logique en typescript de notre composant)

Le fichier **app.module.ts** est automatiquement modifié comme suit.

Les 5 composants sont importés dans le module app.module.ts
Puis sont utilisés dans le décorateur (decorator) **@ngModule** au niveau de son objet metadata **declarations.**

src/app/app.module.ts

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';

import { AppComponent } from './app.component';
import { HomeComponent } from './modules/general/home/home.component';
import { ContactComponent } from './modules/general/contact/contact.component';
import { AboutComponent } from './modules/general/about/about.component';
import { SigninComponent } from './modules/general/signin/signin.component';
import { NotFoundComponent } from './modules/general/not-found/not-found.component';

@NgModule({
  declarations: [
    AppComponent,
    HomeComponent,
    ContactComponent,
    AboutComponent,
    SigninComponent,
    NotFoundComponent
  ],
  imports: [
    BrowserModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

Nous suivrons maintenant les conseils suivants
d'Angular https://angular.io/tutorial/toh-pt5

La meilleure pratique consiste à charger et à configurer le router dans un module dédié au routing.
Ce module sera ensuite importé par dans le module App (appModule).
Par convention le module sera nommé **AppRoutingModule** dans le fichier app-routing.module.ts et situé dans le répertoire **src/app**.

Nous créerons donc

- **app-routing.module.ts**
- **app-routing.module.spec.ts**

Pour cela nous utiliserons la commande angular-cli suivante

```
#  Création du routing
ng generate module app-routing --flat --module=app

#  Création du routing (méthode 2)
ng g m app-routing --flat --module=app
```

Il faut ensuite modifier les fichiers suivants

- **app-routing.module.ts**
- **styles.css**
- **app.component.html**
- **app.component.css**
- **app.component.ts**
- **app.component.spec.ts**
- **app.e2e-spec.ts**
- **app.po.ts**

Ce qui permettra de traiter le routage désiré et les composants appelés.

Dans le fichier app-routing.module.ts

- Importer les modules Routes et RouterModule
- configurer les Routes
- appeler la méthode forRoot de RouterModule

src/app/app-routing.module.ts

```
import { NgModule } from '@angular/core';
import { Routes, RouterModule } from '@angular/router';

import { AboutComponent } from './modules/general/about/about.component';
import { ContactComponent } from './modules/general/contact/contact.component';
import { HomeComponent } from './modules/general/home/home.component';
import { SigninComponent } from './modules/general/signin/signin.component';
import { NotFoundComponent } from './modules/general/not-found/not-found.component';

const routes: Routes = [
  { path: '', component: HomeComponent, },
  { path: 'contact', component: ContactComponent },
  { path: 'about', component: AboutComponent },
  { path: 'signin', component: SigninComponent },
  { path: '**', component: NotFoundComponent }
];

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule],
  declarations: []
})
export class AppRoutingModule { }
```

src/styles.css

```
body {
  color: black;
  font-weight: 400;
}
```

Dans le composant AppComponent nous prendrons soin de rajouter l'élément **<router-outlet>** au niveau du fichier app.component.html.
Il indiquera au router ou afficher les éléments graphiques routés.

L'élément **routerLink** permettra de créer le lien vers les pages souhaitées.

src/app/app.component.html

```
<div class="app">
  <header>
    <section>
      <h1>{{ title }}</h1>
      <p> {{ version }}</p>
    </section>
    <nav>
      <ul>
        <li><a routerLink="/">Home</a></li>
        <li><a routerLink="/about">About</a></li>
        <li><a routerLink="/contact">Contact</a></li>
        <li><a routerLink="/signin">Signin</a></li>
      </ul>
    </nav>
  </header>

  <main>
    <router-outlet></router-outlet>
  </main>

  <footer>
    <a href="https://www.ganatan.com/">&copy; 2020 - www.ganatan.com</a>
  </footer>

</div>
```

src/app.component.css

```
h1 {
  color: blue;
}

.app {
  font-family: Arial, Helvetica, sans-serif;
  max-width: 500px;
  margin: auto;
}
```

src/app.component.ts

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

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  title = 'angular-starter';
  version = 'Angular version 10.0.1';
}
```

Il nous faut rajouter le module **RouterTestingModule** au niveau des test unitaires du composant App.
Ce rajout se fait au niveau du fichier de test correspondant **app.component.spec.ts**.

Nous modifierons les tests end to end pour s'adapter aux modifications du fichier app.component.html
Modifications qui s'effectueront dans le fichier **app.e2e-spec.ts**.

src/app/app.component.spec.ts

```
import { TestBed, async } from '@angular/core/testing';
import { AppComponent } from './app.component';
import { RouterTestingModule } from '@angular/router/testing';

describe('AppComponent', () => {
  beforeEach(async(() => {
    TestBed.configureTestingModule({
      imports: [
        RouterTestingModule
      ],
      declarations: [
        AppComponent
      ],
    }).compileComponents();
  }));

  it('should create the app', () => {
    const fixture = TestBed.createComponent(AppComponent);
    const app = fixture.componentInstance;
    expect(app).toBeTruthy();
```

```
  });

  it(`should have as title 'angular-starter'`, () => {
    const fixture = TestBed.createComponent(AppComponent);
    const app = fixture.componentInstance;
    expect(app.title).toEqual('angular-starter');
  });

  it('should render title', () => {
    const fixture = TestBed.createComponent(AppComponent);
    fixture.detectChanges();
    const compiled = fixture.nativeElement;
    expect(compiled.querySelector('h1').textContent).toContain('angular-starter');
  });
});
```

e2e/src/app.e2e-spec.ts

```
import { AppPage } from './app.po';
import { browser, logging } from 'protractor';

describe('workspace-project App', () => {
  let page: AppPage;

  beforeEach(() => {
    page = new AppPage();
  });

  it('should display welcome message', () => {
    page.navigateTo();
    expect(page.getTitleText()).toEqual('angular-starter');
```

```
  });

  afterEach(async () => {
    // Assert that there are no errors emitted from the browser
    const logs = await browser.manage().logs().get(logging.Type.BROWSER);
    expect(logs).not.toContain(jasmine.objectContaining({
      level: logging.Level.SEVERE,
    } as logging.Entry));
  });

});
```

e2e/src/app.po.ts

```
import { browser, by, element } from 'protractor';

export class AppPage {
  navigateTo(): Promise<unknown> {
    return browser.get(browser.baseUrl) as Promise<unknown>;
  }

  getTitleText(): Promise<string> {
    return element(by.css('app-root h1')).getText() as Promise<string>;
  }

}
```

Tests

Il ne reste plus qu'à tester les scripts suivants.

Remarque
Les tests du code source (npm run lint) peuvent parfois indiquer des erreurs.
Lorsque q'un fichier ts doit se terminer par une ligne vide par exemple.
Il suffit de cliquer sur le lien de l'erreur et modifier le fichier incriminé et recommencer le test.

```
# Développement
npm run start
http://localhost:4200/

# Test de la page not-found
http://localhost:4200/lien-inconnu

# Tests
npm run lint
npm run test
npm run e2e

# Production
npm run build
```

Code source

Le code source utilisé **en début de tutoriel** est disponible sur github
https://github.com/ganatan/angular-example-starter

Le code source obtenu à **la fin de ce tutoriel** est disponible sur github
https://github.com/ganatan/angular-example-routing

Les étapes suivantes vous permettront **d'obtenir une application prototype**

- Etape 1 : Démarrer avec Angular
- Etape 2 : Routing avec Angular
- Etape 3 : Lazy loading avec Angular
- Etape 4 : Bootstrap avec Angular
- Etape 5 : Architecture avec Angular
- Etape 6 : Components avec Angular
- Etape 7 : Services avec Angular
- Etape 8 : Template Driven Forms avec Angular
- Etape 9 : Charts avec Angular
- Etape 10 : Server Side Rendering avec angular
- Etape 11 : HttpClient avec Angular
- Etape 12 : Transfer State avec Angular
- Etape 13 : Progressive Web App avec Angular
- Etape 14 : Search Engine Optimization avec Angular

*Le **code source de cette application finale** est disponible sur GitHub*
https://github.com/ganatan/angular10-app

La dernière étape permet **d'obtenir un exemple d'application**
Etape 15 : Créer une application Web complète avec Angular