TP 1 : Démarrer une Application Web avec Angular CLI

I. Objectif :

Familiariser les étudiants avec l'installation d'Angular et la création d'un premier projet.

Objectifs Spécifiques :

1. Comprendre les outils de développement essentiels (Visual Studio Code, Node.js, Angular CLI).
2. Installer ces outils.
3. Appréhender l'univers d'Angular et son positionnement par rapport à d'autres frameworks front-end.
4. Réviser et approfondir les connaissances de base et avancées de TypeScript pour le développement Angular.

II. Les outils

Les outils que nous utiliserons pour réaliser ce didacticiel sont

1. *Typescript*

TypeScript est un sur-ensemble de JavaScript qui ajoute un typage statique et des fonctionnalités avancées au langage, permettant d'écrire un code plus fiable et maintenable tout en gardant toute la flexibilité de JavaScript.

La version majeur actuelle est 5.X.

2. *Node.js* version LTS (Long Term Support):

Node.js est un environnement d'exécution pour JavaScript côté serveur qui permet de créer des applications performantes et évolutives, en utilisant JavaScript non seulement pour le front-end mais aussi pour le back-end, offrant ainsi une solution complète pour le développement web.

3. *Visual Studio Code*

Visual Studio est un environnement de développement intégré (IDE) performant qui centralise toutes les étapes du cycle de développement en un seul endroit. Il offre une suite complète d'outils pour écrire, modifier, déboguer et compiler du code, ce qui rend l'ensemble du processus de développement plus fluide.

Cet IDE est particulièrement adapté à plusieurs langages de programmation étudiés, notamment ceux utilisés avec Angular, ce qui en fait un outil intéressant et polyvalent pour ce TP et pour les travaux pratiques d'autres matières.

4. *npm* (node package manager) : npm est un gestionnaire de packages pour JavaScript géré par la société npm. **npm** est le gestionnaire de packages par défaut pour l'environnement d'exécution JavaScript Node.js.
5. *Angular* & *Angular CLI*

Les dernières versions de ces outils sont disponibles ci-dessous

- https://github.com/Microsoft/TypeScript/releases
- https://nodejs.org/fr/download/
- https://code.visualstudio.com/
- https://github.com/angular/angular/releases
- https://github.com/angular/angular-cli/releases

Avant de commencer à utiliser Angular il nous faudra au préalable installer Node.js et npm. Le TP suivant nous explique comment faire.

III. Préparation des outils

1. Installer Visual Studio Code.
2. Installer Node.Js, de préférence utiliser une version LTS et pas la dernière version pour des raisons de stabilité et de compatibilité avec Angular. N'est pas choisir l'option « installer automatiquement les outils… ».

Il est à noter que ce TP a été réalisé avec la version 16.x et npm 8.x.

3. Pour vérifier la version de Node.Js installée, sous la ligne de commande ou dans PowerShell, taper

```
# Vérification version nodejs
node -v
# Vérification version npm
npm -v
#noter que la commande suivante est plus détaillée.
npm version
si vous voulez mettre Mise à jour de npm
npm install npm -g            ou      npm install npm@latest -g

Pour installer une version spécifique de npm il faut exécuter.
npm install -g npm@8.19.2
```

Attention :
npm est un script Windows. Pour pouvoir l'exécuter sans soucis, il faut exécuter au préalable la ligne de commande en mode administrateur et qu'il faut autorise l'exécution de script dans Windows possible
taper la commande suivante pour résoudre ce problème
Set-ExecutionPolicy Unrestricted -Scope CurrentUser

4. Installer Angular CLI. Pour ce TP on utilise la version 18.x

```
Pour installer la distribution CLI de Angular
npm install -g @angular/cli
```

```
# Installation d'angular-cli version spécifique
npm install -g @angular/cli@16.0.0
```

Si une version précédente était installée sur votre poste vous pouvez la désinstaller avec la commande suivante

```
# Désinstallation d'angular-cli
npm uninstall -g @angular/cli
# vider le cache
npm cache clean –force
# Supprimer les dossiers associés
```

5. Vérifier la version installée.

```
# Test de version installée
ng version                    ou          ng v
```

IV. Création d'un projet

Pour créer une nouvelle application (site) Angular on utilise la commande :

```
ng new <nom application> <options>    OU
ng n <nom application> <options>
```

Pour ce TP on va opter pour les valeurs par défaut.

1. De préférence créer un dossier pour vos projets Angular. Par exemple « d:\AngularProjects » ou « D:\AngularProjects »

Passer au dossier de vos Projets Angular.

2. Créer un nouveau projet Angular.

```
# Générer un projet «project0» avec choix des options
ng new project0
# Générer un projet appelé angular-project0 avec options par défaut
ng new project0 --defaults
```

Selon la ligne de commande que vous avez tapée, et l'environnement que vous utilisez, il est possible que vous soyez appelé à répondre aux questions :

- Routing : Répondre non
- Choix de type CSS : choisir le premier

Par défaut, cette commande crée un dossier d'espace de travail nommé « **project0**» et génère un nouveau squelette d'application dans un dossier ***src/*** au niveau supérieur de l'espace de travail. Une application nouvellement générée contient des fichiers source pour un module racine, avec un composant racine et un modèle.

3. Parcourir ces différents dossiers et fichiers et compléter cette description.

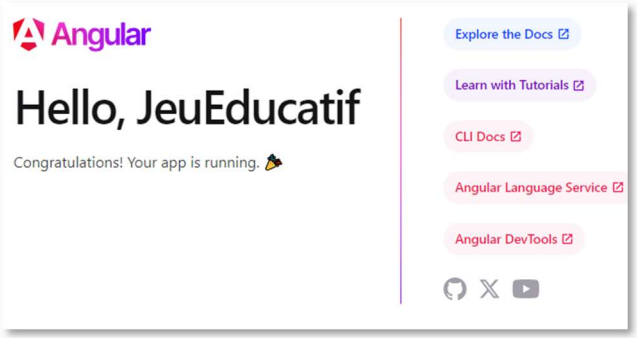
- Le dossier src/ contient ……
- Le dossier node_modules/ contient ……
- Le fichier angular.json contient ……
- Le fichier package.json ……

- Le fichier package-lock.json Fournit ……
- Le fichier tsconfig.json contient ……

4. Se positionner dans le dossier du projet et l'exécuter.

```
# Se positionner dans le projet
cd project0
# Exécuter le projet (sans lancer le navigateur)
ng serve
# Exécuter et lancer automatiquement l'application dans le navigateur
ng serve -o
```

```
# Tester
http://localhost:4200
```



5. Autres commandes utiles :

```
#Exécute de l'application en mode développement.
npm run  start
# Compile l'application dans le répertoire dist.
npm run  build
#Exécute les tests unitaires en utilisant le framework Karma.
npm run  test
#Exécute l'analyse de code avec TSLint.
npm run  lint
#Execute les tests end-to-end avec Protractor.
npm run  e2e
```

En mode développement si nous voulons personnaliser le port il suffit de modifier le script start dans le fichier package.json.

Par exemple pour utiliser le port 4201 le script serait le suivant "start" : "**ng serve --port 4201**"

Nous laisserons le port 4200 modifiable à volonté pour la suite du tp.

V. Visual Studio Code

A cette étape nous utilisons Visual Studio Code. Avec VSC, ouvrir le dossier de votre projet. Répondez par Yes à la question "Do you trust the authors of the files in this folder?"

Les fichiers et dossier du dossier **src** sont comme suit :

Fichier/Dossier	Description
`app/`	Contient les fichiers de composants dans lesquels la logique et les données de votre application sont définies.
`assets/`	Contient des images et d'autres fichiers d'assets à copier tels quels lorsque vous créez votre application.
`environnements/`	Contient des options de configuration de build pour des environnements cibles particuliers. Par défaut, il existe un environnement de développement standard sans nom et un environnement de production ("prod"). Vous pouvez définir des configurations d'environnement cible supplémentaires.
`favicon.ico`	Une icône à utiliser pour cette application dans la barre de favoris.
`index.html`	La page HTML principale qui est servie lorsqu'un internaute visite votre site. La CLI ajoute automatiquement tous les fichiers JavaScript et CSS lors de la création de votre application
`main.ts`	Le point d'entrée principal de votre application. Compile l'application avec le compilateur JIT et amorce le module racine de l'application (AppModule) pour qu'il s'exécute dans le navigateur.
`polyfills.ts`	Fournit des scripts polyfill pour la prise en charge du navigateur.
`styles.sass`	Répertorie les fichiers CSS qui fournissent des styles pour un projet. L'extension reflète le préprocesseur de style que vous avez configuré pour le projet.
`test.ts`	Le point d'entrée principal pour vos tests unitaires, avec une configuration spécifique à Angular. Vous n'avez généralement pas besoin de modifier ce fichier.

1. Ouvrez le fichier package.json (du dossier projetc0). Parcourir ce fichier et remarquer ce qu'il contient.

Le fichier package.json contient les différentes dépendances de votre projet. Les dépendances sont en quelque sorte toutes les librairies que vous avez décidé d'utiliser dans votre projet. Elles sont gérées par npm (node package manager) le gestionnaire de dépendances de Node.js.

2. Vous pouvez mettre à jour le fichier *package.json* avec les dernières dépendances. Ouvrez dans VSC, une nouvelle fenetre Terminal, et lancez la commande, pour contrôler les dépendances à mettre à jour.

```
# vérifier les mises à jour possibles pour votre environnement\
npm outdated
```

Cette commande vérifiera le registre pour voir si des packages installés (ou spécifiques) ont une mise à jour possible.

```
Toutes les dépendances peuvent être mises à jour à l'exception de typescript.
```

Pour ce faire, et indiquer Typescript version 4.3.5, Modifier le fichier package.json comme suit puis exécuter le script `npm install .`

3. Consulter le fichier ***index.html*** du dossier ***src***. C'est le fichier principal du projet. Son contenu est semblable à ceci.

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>Project0</title>
  <base href="/">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="icon" type="image/x-icon" href="favicon.ico">
</head>
<body>
  <app-root></app-root>
</body>
</html>
```

Vous remarquez que Angular ajoute automatiquement tous les fichiers JavaScript et CSS lors de la création de l'application, vous n'avez donc généralement pas besoin d'ajouter manuellement les balises <script> ou <link> ici.

4. Effectuez des recherches sur la balise ***app-root***. Ajouter un contenu avant la balise app-root et après et remarquez le résultat.

Quand Angular rencontre la balise `<app-root>` dans le document HTML, il sait qu'il doit en remplacer le contenu par celui du template `app.component.html`, en appliquant les styles `app.component. scss`, le tout géré par la logique du fichier `app.component.ts`.

5. Il est possible de faire le débogage, les tests et le passage en production.

Toute modification entraine une recompilation du code. Par exemple Modifier le fichier `app. component.html`

** ICI c est L2DSI **

La compilation est alors exécutée automatiquement et le navigateur se réactualise.

6. ESLint est un outil permettant d'identifier et de signaler les modèles trouvés dans le code ECMAScript/JavaScript, dans le but de rendre le code plus cohérent et d'éviter les bogues. Effectuer des tests suivants.

La commande `ng lint` exécute l'analyse statique du code source TypeScript. Angular utilise l'outil TSLint.

L'analyse syntaxique génère un warning (*no-use-before-declare is deprecated*). Il suffit de passer à `false` l'option `no-use-before-declare` dans le fichier `tslint.json`.

```
# Analyse de code
npm run lint
# Tests unitaires
npm run test
```

7. Compilation et déploiement

```
# Compiler
npm run build
```

VI. Validation des acquis

Partie 1 : Questions à Choix Multiples (QCM) sur les outils de développement

Veuillez répondre aux questions suivantes pour évaluer votre compréhension des outils de développement essentiels pour Angular.

1. Quelle est la principale fonction de Node.js dans le développement Angular ?

a) Un environnement d'exécution pour le code JavaScript côté serveur.

b) Un éditeur de code source pour écrire du code Angular.

c) Un gestionnaire de packages pour installer les dépendances Angular.

d) Un framework front-end pour créer des interfaces utilisateur.

2. Quelle commande permet d'installer Angular CLI globalement sur votre système ?

a) npm install -g @angular/core

b) npm install @angular/cli

c) npm install -g @angular/cli

d) ng new my-app

3. Parmi les éditeurs de code suivants, lequel est recommandé pour le développement Angular en raison de ses extensions et fonctionnalités intégrées ?

a) Notepad++

b) Visual Studio Code

c) Sublime Text

d) Eclipse

4. Quel fichier dans un projet Angular contient les dépendances du projet et est utilisé par npm pour gérer les packages ?

a) angular.json

b) package.json

c) tsconfig.json

d) README.md

5. Quelle est la commande pour créer un nouveau projet Angular nommé "ma-app" en utilisant Angular CLI ?

a) ng create ma-app

b) ng init ma-app

c) ng new ma-app

d) ng generate app ma-app

6. Quel est le rôle de la commande ng serve ?

a) Compiler le projet Angular en un fichier exécutable.

b) Lancer un serveur de développement et recharger l'application lors des modifications.

c) Installer les dépendances du projet.

d) Générer des composants, services et autres éléments Angular.

Partie 2 : QCM sur la création d'un projet Angular

Veuillez répondre aux questions suivantes concernant la création et la structure d'un projet Angular.

1. Quel répertoire contient les fichiers sources de l'application Angular que vous allez développer ?

a) src/

b) dist/

c) node_modules/

d) e2e/

2. Dans quel fichier définissez-vous les modules, composants, services et autres éléments qui font partie de votre application Angular ?

a) main.ts

b) app.module.ts

c) index.html

d) styles.css

3. Quel est le rôle du fichier angular.json dans un projet Angular ?

a) Configurer les dépendances npm du projet.

b) Définir les configurations de compilation, les entrées et sorties du projet.

c) Spécifier les règles de linting pour le code TypeScript.

d) Définir les styles globaux de l'application.

4. Quelle commande Angular CLI utilisez-vous pour générer un nouveau composant nommé header ?

a) ng new component header

b) ng generate header

c) ng generate component header

d) ng create component header

5. Quel est le rôle du fichier tsconfig.json dans un projet Angular ?

a) Configurer le compilateur TypeScript pour le projet.

b) Définir les styles CSS globaux.

c) Spécifier les configurations de test unitaire.

d) Gérer les dépendances des modules Angular

Partie 3 : Exploration de l'univers Angular

1. Rédigez un rapport comparatif entre Angular et deux autres frameworks front-end populaires (par exemple, React et Vue.js), en abordant les points suivants :
   - Les principales caractéristiques et fonctionnalités d'Angular.
   - Les différences clés entre Angular et les autres frameworks choisis.
   - Les avantages et inconvénients de chaque framework.
2. Présentez votre rapport lors d'une discussion de groupe ou soumettez-le pour évaluation.

Partie 4 : Révision des bases de TypeScript

1. Créez un fichier exercices-typescript.ts dans lequel vous allez pratiquer les concepts de base de TypeScript.
2. Effectuez les tâches suivantes :
    - Déclarez des variables en utilisant différents types primitifs (string, number, boolean, etc.).
    - Créez une fonction typée qui calcule la somme de deux nombres.
    - Définissez une interface Etudiant avec des propriétés comme id, nom, prenom, age.
    - Implémentez une classe Etudiant qui implémente l'interface et ajoute une méthode pour afficher les informations de l'étudiant.
3. Compilez le fichier TypeScript en JavaScript et exécutez-le pour vérifier son bon fonctionnement.

Partie 5 : Approfondissement de TypeScript

1. Dans le même fichier exercices-typescript.ts, mettez en pratique les concepts avancés suivants :
    - Utilisez les types génériques pour créer une fonction qui retourne un tableau de valeurs du même type.
    - Implémentez des unions de types et des types optionnels dans une fonction.
    - Expérimentez avec les énumérations (enum) pour représenter des valeurs constantes.
2. Commentez votre code pour expliquer les concepts utilisés et les choix effectués.