GANATAN

Démarrer une Application Web avec Angular 20

Guide Complet

Mise à jour du 30 juillet 2025

Nous allons réaliser une **Application Web.**

Dans ce tutoriel, nous utiliserons **Angular version 20.1.3**

Pour commencer notre application nous partirons de zéro (**from scratch**) en nous efforçant de suivre les meilleures pratiques (**best practices)** d'Angular.

- Angular a été créé par **Google**.
- Angular est **open source**, son utilisation est donc gratuite.
- Angular utilise **Typescript**.
- Angular est un framework **Javascript Frontend**.

www.ganatan.com

Comment le faire ?

Pour débuter notre projet voici un résumé de ce que nous allons faire.

- **Installation des outils nécessaires**

  Node.js sera notre plateforme de développement javascript.
  Pas le choix, sans Node.js ça ne marchera pas.

  Visual studio Code sera notre éditeur de code.
  Le choix est totalement arbitraire mais pour un outil Microsoft c'est une petite merveille

  Git sera notre gestionnaire de logiciel.
  Grâce à lui vous pourrez utiliser le code source de ce tutoriel.

  Angular CLI sera notre homme à tout faire.
  Sans doute l'outil le plus connu et le plus utilisé du Framework Angular.

- **Initialisation du projet**
  Nous utiliserons Angular CLI pour la mise en place de l'architecture du projet, en utilisant les meilleures pratiques (best practices) préconisées par Google.

- **Mise à jour du projet**
  Vérifier les dépendances utilisées et les mettre à jour.

- **Effectuer les Tests**
  Les tests Unitaires et les outils qui leur sont dédiés Karma et Jasmine.
  Le linting et l' amélioration du code avec ESLint.

- **Environment**
  Depuis la version 15 l'équipe d'Angular n'intègre plus les paramètres d'environnement (environment en anglais).
  Très utiles nous verrons comment les déclarer et les utiliser.

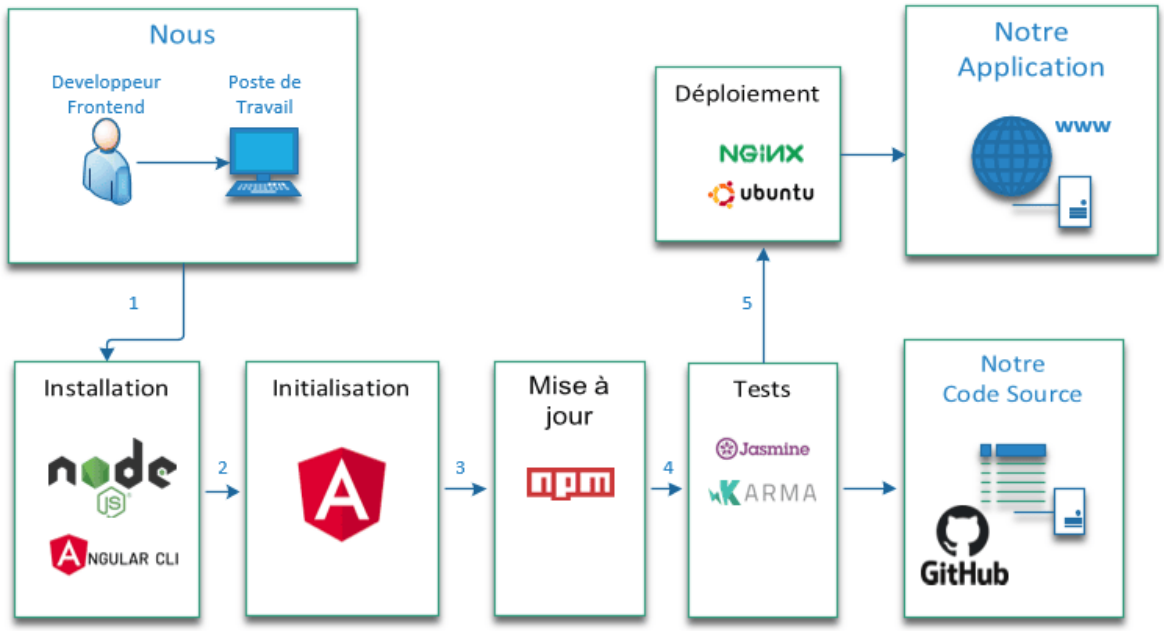
- **Déploiement**

Comment déployer votre application sur internet.

- **Code source**
  Le code complet du projet est disponible sur Github.

Une image vaut mille mots

Le résumé en image de ce que nous allons faire



Si vous êtes du genre très pressé ci-dessous un résumé en commandes sinon passez à l'étape suivante.

Commandes à éxécuter

```
# Désinstallez Angular CLI (au cas ou une ancienne version d'Angular aurait été installée)
npm uninstall -g @angular/cli

# Installez Angular CLI version spécifique (la dernière si possible)
npm install -g @angular/cli@20.1.3

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

# Allez dans ce répertoire
cd demo

# Générez un projet appelé angular-starter avec choix manuel des options
# Sélectionnez les options par défaut
ng new angular-starter

# Positionnez vous dans le projet
cd angular-starter

# Exécutez l'application
npm run start

# Testez l'application dans votre navigateur
http://localhost:4200
```

Et si vous êtes moins pressé ça va commencer

On va faire les choses sérieusement mais on ne va pas se prendre au sérieux. Donc c'est parti pour **un peu d'humour** et **beaucoup de technique.**

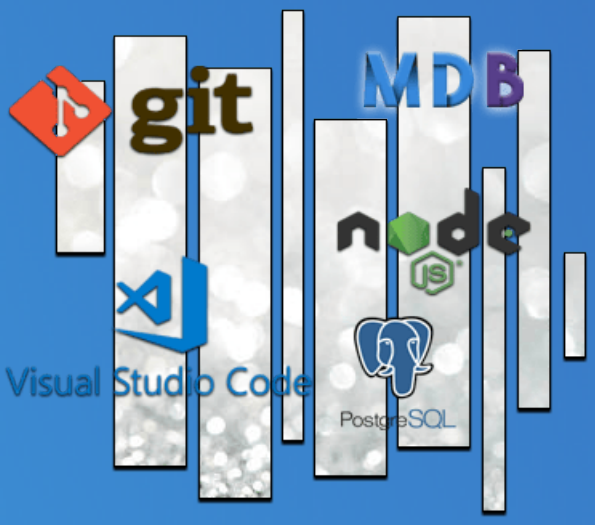
sans IA

avec IA

Etape 1

Installation des outils nécessaires

Avant d'utiliser Angular il nous faut installer un certain nombre de logiciels



- **Node.js**
  Impossible de faire fonctionner Angular sans lui.
- **Visual Studio Code**
  Ce choix est arbitraire.
- **Git**
  Très utile mais pas essentiel
- **Angular CLI**
  C'est l'homme à tout faire d'Angular

Installation de Node.js

Si vous ne l'installez pas, Angular ne fonctionnera pas.

A ce propos Angular, React et Vuejs ont besoin tous trois de Node.js.

Le site officiel c'est ici https://nodejs.org/en

Voilà ce qu'il nous dit:

Node.js est un environnement d’exécution JavaScript construit sur le moteur JavaScript V8 de Chrome.

Son inventeur Ryan Lienhart Dahl l'a créé le 27 mai 2009.

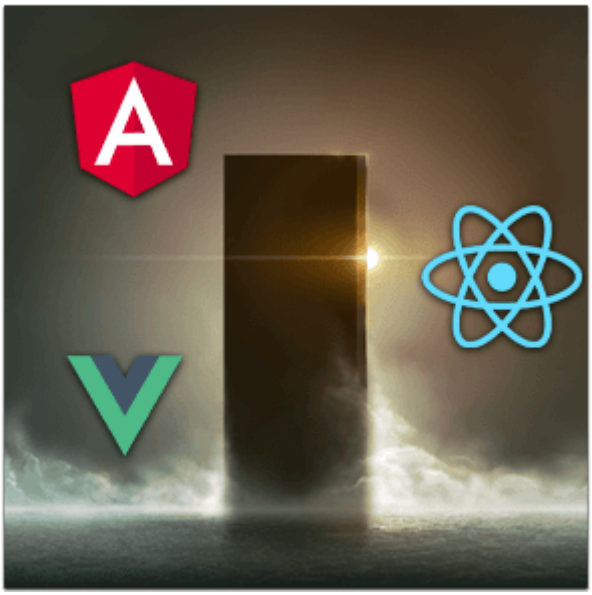
Il avait une idée précise derrière la tête : *la simplicité et la rapidité d'exécution de programmes écrits en javascript*.

Le choix du nom n'est donc pas anodin.

- **Node** signifie **noeud**
- **JS** signifie **javascript**

Node.js est ainsi le **point central** qui va permettre d'**exécuter des programmes écrits** en javascript **côté serveur.**

Mon Dieu ! C'est plein d'étoiles !



Node.js utilise un outil **npm** (Node Package Manager)

Npm simplifie la vie du développeur en permettant de publier et de partager des librairies Node.js.

Npm permet notamment de simplifier l'installation, la mise à jour ou la désinstallation de ces librairies.

On pourra parler de librairies, de paquets ou de dépendances (en anglais packages ou dependencies).

Comment l'installer ?

Sur le site officiel le téléchargement est accessible à l'adresse

https://nodejs.org/en

Nous allons utiliser la version **LTS** (**Long Term Support** ou Support à long terme).

LTS signifie que l'éditeur nous garantit en général une période de maintenance d'au moins deux ans,

- **Node.js** version 22.16.0 LTS
- **npm** (node package manager) version 11.4.1

Il s'agit d'une installation classique.

- Sélectionnez **Download Node.js (LTS)**.
- Téléchargez le programme et éxécutez le.

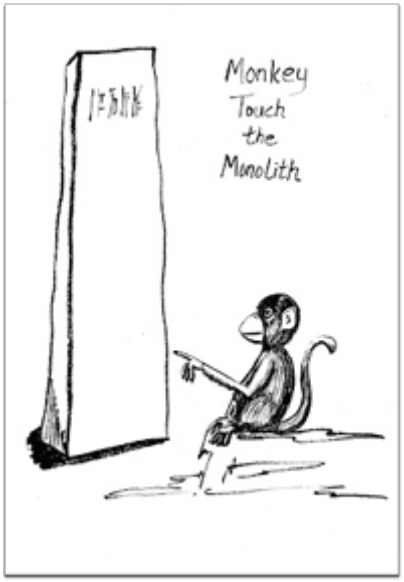
Une fois l'installation effectuée on peut vérifier que Node.js est installé sur notre poste de travail.

Commandes à éxécuter

```
# Vérification de la version de Node.js et de npm (méthode 1)
node --version
npm --version

# Vérification de la version de Node.js et de npm (méthode 2)
node -v
npm -v

# Mise à jour de npm
npm install npm -g

# Vérification de la mise à jour de npm
npm -v
```

Comment savoir que Node.js fonctionne ?

On va vérifier que Node.js fonctionne et qu'il permet d'exécuter un programme javascript.



Allons sur Wikipedia https://fr.wikipedia.org/wiki/Node.js

Testons le programme d'exemple *"un Hello Word"* qu'il nous propose.

Créez un fichier index.js avec un éditeur de code (bloc-notes fera l'affaire).

Copiez le code exemple suivant

hello.js

```
const  { createServer } = require('http');

// Création du serveur
const  server = createServer((request, response) => {
    response.writeHead(200, {'Content-Type': 'text/plain'});
    response.end('Hello World\n');
});

server.listen(3000, () => console.log(`Adresse du serveur : http://localhost:3000`));
```

Il ne reste plus qu'à procéder aux tests

Test à éxécuter

```
# Exécution du programme javascript
node index.js

# Vérification dans le navigateur
http://localhost:3000
```

Installation de Visual studio code

Visual Studio Code est l'éditeur utilisé dans la plupart des conférences sur Angular.

Il est notamment utilisé par **John Papa** l'un des meilleurs conférenciers Angular et auteur des guides Angular

https://github.com/johnpapa/angular-styleguide

Dans la suite du tutoriel nous utiliserons donc **Visual Studio Code**.

VS code est un éditeur de code développé par Microsoft pour Windows, Linux et OS X.

Procédons à l'installation.

Le site officiel est là https://code.visualstudio.com/

Nous utiliserons la dernière version **1.100.2** à télécharger ici

https://code.visualstudio.com/updates/v1_100

L'installation est simple comme celle de Node.js.

Cliquez sur **Download for Windows**
Téléchargez et exécutez

Installation de Git

Ecrire une Application Web c'est un peu comme écrire un livre.
Plus le temps passe plus le **nombre de pages** augmentent.
De quelques centaines vous pouvez passer à des milliers de pages.
Le **nombre de modifications** devient considérable et pour s'y retrouver ce n'est pas une mince affaire.

Des questions, des questions, trop de questions

Pour gérer ce problème, des outils ont été développés.
Ce sont les **logiciels de gestion de versions** (ou **VCS** en anglais, pour version control system).

Le plus connu est **Git**. Il a été créé par **Linus Torval** le créateur de Linux.

Il va nous permettre de gérer notre code source et ses différentes versions.
Et surtout de pouvoir partager ce code source, permettant ainsi de travailler à plusieurs.

Git vous permettra aussi d'utiliser et de tester le code source de ce tutoriel.

Passons à l'installation.

Le site officiel est à l'adresse suivante https://git-scm.com/

L'installation est accessible ici https://git-scm.com/download/win

Téléchargez l'application puis exécutez la.

Pour vérifier que Git est installé sur votre poste de travail il suffit de lancer une ligne de commande.

```
# Test de la version
git --version
```

Installation d' Angular CLI

Angular CLI ça veut dire **Angular Command Line Interface.**
Mais c'est surtout ça.

L' homme à tout faire d'Angular

Et autant utiliser la version la plus récente.

Angular version **20.1.3**
Angular CLI version **20.1.3**

Les dernières versions de ces outils sont disponibles ci-dessous
https://github.com/angular/angular/releases
https://github.com/angular/angular-cli/releases

La procédure d'installation est détaillée sur le site officiel d'Angular

https://angular.dev/tools/cli/setup-local#dependencies

La méthode est décrite sur la page du site officiel.

Je vais détailler celle-ci.

- 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
```

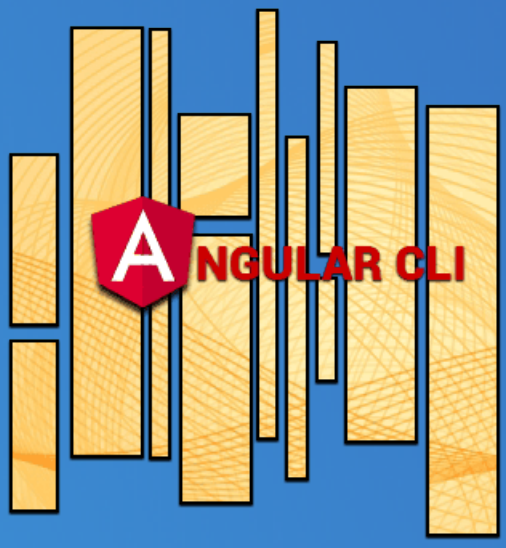
Angular CLI est une librairie (ou package).
Nous allons l'installer avec npm le gestionnaire de node.js

Vous pouvez installer une version spécifique d'angular ou installer par défaut la dernière disponible.

```
# Installation d'angular-cli dernière version disponible
npm install -g @angular/cli

# Installation d'angular-cli version spécifique
npm install -g @angular/cli@20.1.3

# Test de version installée
ng version
```

Etape 2

Initialisation du projet avec Angular CLI



Angular est un framework complet qui couvre un grand nombre de fonctionnalités.

La documentation est particulièrement fournie et détaillée.

Nous essaierons le plus souvent possible de respecter les meilleures pratiques (**best practices**) préconisées par l'équipe d'angular.

Nous pouvons créer manuellement chaque élément de notre application mais le plus simple est d'utiliser **Angular CLI**

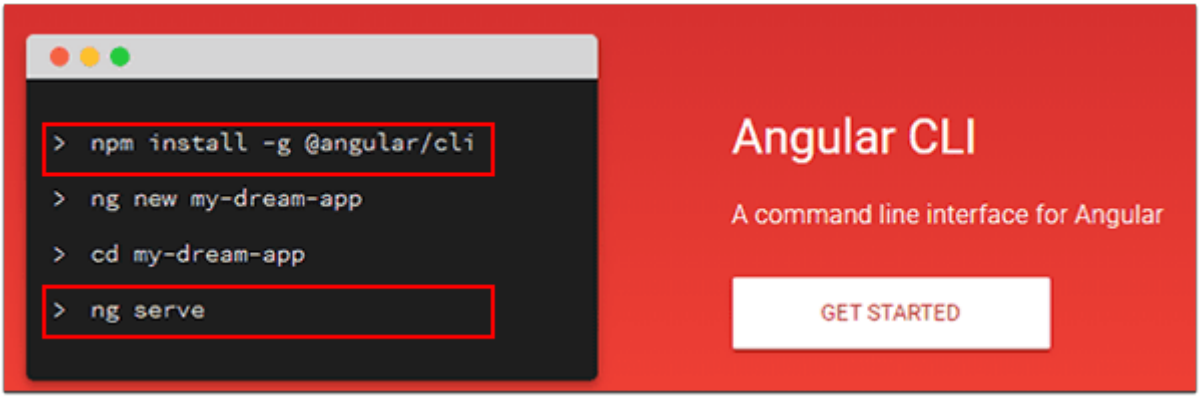
Créer notre application avec Angular CLI

Angular CLI est un outil pour initialiser, développer et maintenir des applications Angular.

Le site officiel est ici https://angular.dev/cli

Et si vous voulez avoir la liste des commandes Angular CLI

https://github.com/angular/angular-cli/wiki



Pour aller plus vite je vous fais un résumé essentiel.

Angular CLI nous offre un certain nombre de commandes.

Ces commandes nous évitent d'effectuer des tâches répétitives.

La première commande que nous allons utiliser est **ng new** ou **ng n**

- Elle va créer notre application.
- Elle va générer tous les fichiers nécessaires à cette application.
- Elle va évidemment suivre les best practices préconisées par l'équipe de Google.

On choisit le nom de notre application (arbitrairement ce sera ici **angular-starter**)

On tape la commande ng new avec les paramètres correspondants

- On génère le projet (*cette partie prend quelques minutes*)
- *Pour une compréhension plus simple nous gérerons le routing et le sass*

dans un autre tutoriel.

- **Choisir le type CSS** (valeur par défaut Yes)
- **Désactiver le Server side Rendering** (valeur par défaut Non)

- On se positionne dans le projet

- On exécute le projet

Ce qui donne

```
# Générer un projet appelé angular-starter avec choix manuel des options
ng new angular-starter

# Générer un projet appelé angular-starter avec options par défaut
ng new angular-starter --defaults

# Se positionner dans le projet
cd angular-starter

# Exécuter
ng serve

# Exécuter et lancer automatiquement l'application dans le navigateur
ng serve -o
```

Angular CLI via la commande **ng serve** execute le projet sur un port par défaut (4200).

Il ne reste qu'à tester le fonctionnement dans un navigateur en lançant l'url suivante.

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

Utiliser notre application avec Visual Studio Code

Lancez **VS Code**.

Ouvrez un dossier dans le répertoire **angular-starter** que nous avons créé lors de l'initialisation.

Ouvrez ensuite le fichier **package.json**.

Celui-ci contient un certain nombre de commandes (ou scripts) que nous utiliserons tout au long de ce tutoriel.

Ouvrez une console VS code (sélectionnez Afficher/Terminal) pour exécuter les scripts suivants

- **npm run start** : Exécute l'application en mode développement.
- **npm run build** : Compile l'application dans le répertoire **dist.**
- **npm run test** : Exécute les tests unitaires en utilisant le framework **Karma.**

Remarque pour les nostalgiques des versions précédentes
La commande ng eject (permettant de générer la configuration webpack) a été désactivée.
Elle a été supprimée depuis la version 8
Projet exemple de gestion du format de configuration
https://github.com/manfredsteyer/ngx-build-plus

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 tutoriel.

package.json

```
"scripts": {
  "ng": "ng",
  "start": "ng serve --port 4200",
  "build": "ng build",
  "watch": "ng build --watch --configuration development",
  "test": "ng test"
}
```

Etape 3

Mise à jour du package.json

Node.js est la **plateforme** pour développer notre application.

Node.js est basée sur l'utilisation de librairies ou dépendances.

Npm est le gestionnaire des **librairies** (packages en anglais)

La mise à jour d'une application et donc de ses librairies est une question périlleuse.

Je vais vous montrer qu'avec Angular la notion de **mise à jour** des versions est essentielle.

Elle doit être menée avec prudence.

Update or not update

Les **librairies javascript** sont constamment **modifiées** et **mises à jour** par leur concepteur.

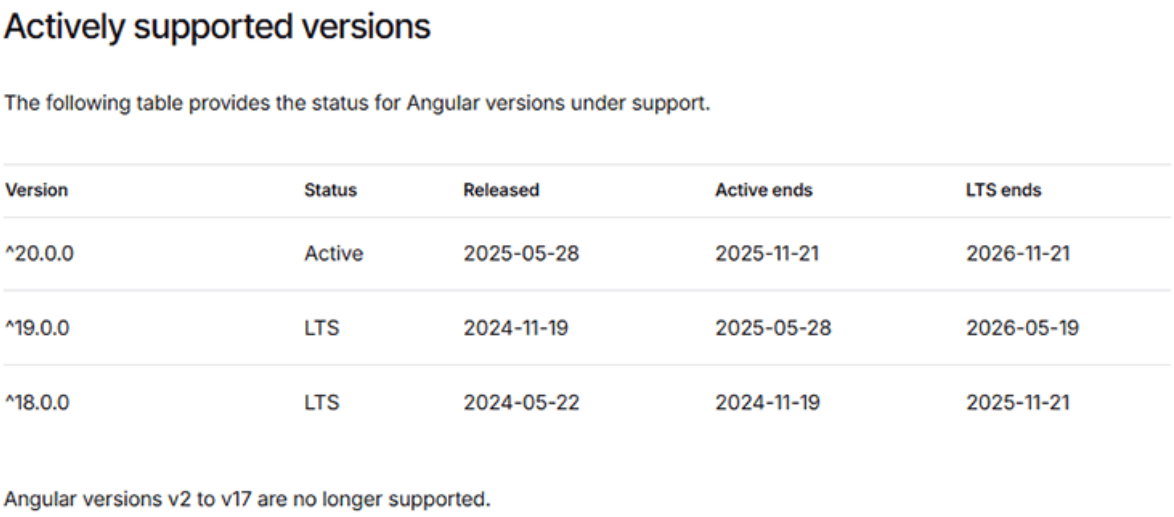
Lorsqu'une nouvelle version est disponible elle porte le nom de **release** (sortie en anglais) et dispose d'un numéro spécifique.

Si la librairie est open-source vous pouvez voir les dernières versions disponibles en allant sur le dépôt correspondant sur Github puis allez sur Releases.

Par exemple les différentes versions d'Angular sont accessibles ici
https://github.com/angular/angular/releases

Le calendrier des mises à jour est ici
https://angular.dev/reference/releases#versioning

Votre passé, votre présent et votre futur.



Actively supported versions

The following table provides the status for Angular versions under support.

Version	Status	Released	Active ends	LTS ends
^20.0.0	Active	2025-05-28	2025-11-21	2026-11-21
^19.0.0	LTS	2024-11-19	2025-05-28	2026-05-19
^18.0.0	LTS	2024-05-22	2024-11-19	2025-11-21

Angular versions v2 to v17 are no longer supported.

Remarquez que les versions 2 à 17 ne sont plus supportées.

Et le danger est évidemment que toutes ces mises à jour altèrent le fonctionnement de notre application.

Dans tous les cas on ne pourra y échapper il faudra un jour ou l'autre essayer de les intégrer dans nos projets.

Je vais vous expliquer comment je procède personnellement.

Je n'ai pas pensé aussi loin

Comment fait-on ?

Utilisons npm (Node Package Manager) le gestionnaire de librairies de Node.js.
La documentation complète est
ici https://docs.npmjs.com/cli/outdated.html

Deux commandes nous seront utiles

Vérifier les versions effectivement installées

- npm list --depht=0

  Cette commande vérifie les dépendances effectivement installées dans le répertoire node_modules.

  Elle nous fournit une liste que l'on peut renseigner dans package.json (voir Remarque)

Vérifier les versions de nos librairies via la commande

- npm outdated

  Cette commande vérifie le registre des dépendances pour vérifier si les packages installés sont à jour.

  Elle nous fournit ainsi une liste que l'on peut contrôler.

Remarque
Avant de vérifier les dépendances mettons à jour le fichier package.json en fonction des versions fournies
par la commande npm list
Pour chaque dépendance indiquée supprimez le caractère ~ ou ^
Par exemple remplacez
- "rxjs": "~7.8.0",
- "tslib": "^2.3.0",

par
- "rxjs": "7.8.2",
- "tslib": "2.8.1",

Pour éviter les erreurs éventuelles supprimer au préalable

le fichier **package-lock.json** et le répertoire **node_modules**.

Puis réinstallez les dépendances avec npm install (package-lock.json et node_modules sont alors recréés automatiquement).

```
# Vérification des versions des dépendances installées dans node_modules
 npm list --depth=0

# Vérification des dépendances disponibles
npm outdated
```

Résultats de la mise à jour !

Si je mets à jour le fichier package.json je me retrouve confronté à 3 cas de figure

- 1/ Ca marche

  C'est pas tous les jours la fête mais depuis Angular 8 c'est de plus en plus souvent

- 2/ Ca ne marche pas on essaie de debugger sans y passer trop de temps.

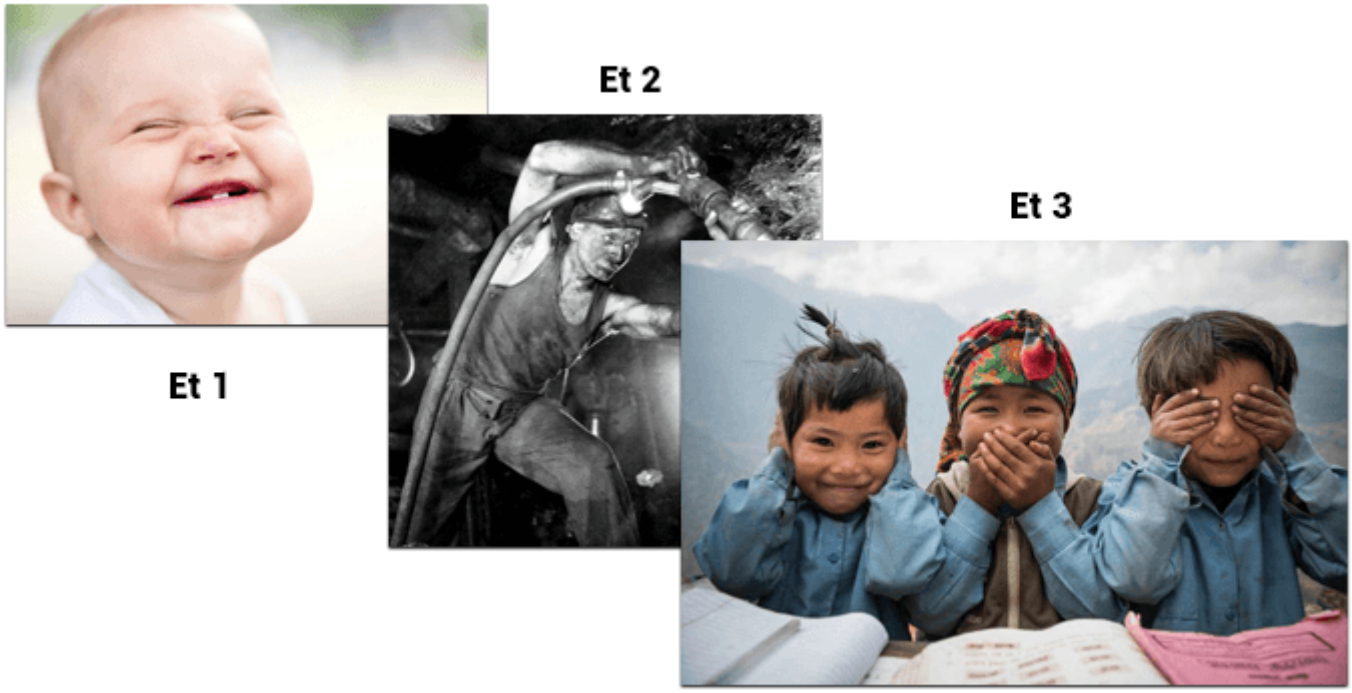
  Ca dépend de votre patience et du temps que vous avez devant vous

- 3/ Ca marche pas et on attend.

  Souvent (mais pas toujours) Angular résout votre problème avec la mise à jour suivante.

  De toute façon ce n'est la peine d'attendre indéfiniment, il faudra bien trouver une solution.

  Ou alors on se retrouve avec AngularJS en 2022 et là on n'est pas dans le pétrin.



Notre prototype Angular

L'idéal est d'avoir un prototype d'application qui contienne suffisamment de fonctionnalités.

Vous pouvez être à-peu-près sûr que la mise à jour pourra s'effectuer sur la plupart de vos applications.

Bien sûr ça ne vous épargnera pas d'optimiser votre CI/CD et de veiller à vos tests.

En tout cas voici une liste des fonctionnalités essentielles à une application selon moi.

- **Routing**
- **Lazy Loading**
- **Bootstrap**
- **Httpclient**
- **SSR**
- **PWA**
- **SEO**
- **Components**
- **Services**
- **Observables**
- **Directives**
- **Pagination**
- **ScrollBox**
- **Charts**
- **Authentification (authentication/Route guard/Role guard/Jwt)**
- **Ngrx**
- **Reactiveform / Tempate Driven form**
- **Form Modal**
- **Internationalization**
- **Tests (unit et end-to-end)**

Le dépôt qui me sert pour l'instant de prototype est le suivant.

https://github.com/ganatan/angular-app

Donc c'est parti pour la mise à jour

Pour l'exemple nous allons utiliser cette méthode sur notre application angular-starter.

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**.

Concernant les dépendances et leur version la documentation npm est la suivante

https://docs.npmjs.com/files/package.json#dependencies

Les spécificateurs de version sont nombreux.

Nous pouvons utiliser par exemple

- **version** Doit correspondre à la version exactement
- **~version** "Approximativement équivalente à la version"
- **^version** “Compatible avec la version”

Nous opterons quant à nous pour le premier spécificateur "**version**", qui est le plus simple, le plus explicite mais aussi le plus restrictif.

Nous allons mettre à jour le fichier package.json avec les dernières dépendances

- Pour contrôler les dépendances à mettre à jour lancez la commande
npm outdated
- Dans certains cas toutes les dépendances peuvent être mises à jour à l'exception de typescript
Angular 20.1.3 accepte par exemple TypeScript supérieur ou égal à 5.8.3
Vous pouvez le vérifier après la mise à jour en exécutant le script npm run build
- Dans le cas donc d'Angular 20.1.3 toutes les dépendances peuvent être mises.
- Supprimez le fichier package-lock.json et le répertoire node_modules
Modifiez le fichier package.json comme suit puis executez le script
npm install

package.json

```
  "dependencies": {
    "@angular/common": "20.1.3",
    "@angular/compiler": "20.1.3",
    "@angular/core": "20.1.3",
    "@angular/forms": "20.1.3",
    "@angular/platform-browser": "20.1.3",
    "@angular/router": "20.1.3",
    "express": "5.1.0",
    "rxjs": "7.8.2",
    "tslib": "2.8.1",
    "zone.js": "0.15.1"
  },
  "devDependencies": {
    "@angular/build": "20.1.3",
    "@angular/cli": "20.1.3",
    "@angular/compiler-cli": "20.1.3",
    "@types/jasmine": "5.1.8",
    "angular-eslint": "20.1.1",
    "eslint": "9.32.0",
    "jasmine-core": "5.9.0",
    "karma": "6.4.4",
    "karma-chrome-launcher": "3.2.0",
    "karma-coverage": "2.2.1",
    "karma-jasmine": "5.1.0",
    "karma-jasmine-html-reporter": "2.1.0",
    "typescript": "5.8.3",
    "typescript-eslint": "8.38.0"
  }
```

Il suffit alors de tester tous les scripts pour vérifier que les mises à jour ont fonctionné.

Etape 4

Tests et déploiement

Le développement est entré dans sa phase d'industrialisation.

Au même titre que les autres industries la qualité et la quantité doivent être au rendez-vous.

Des méthodes **Agile** ont été inventées pour cela.

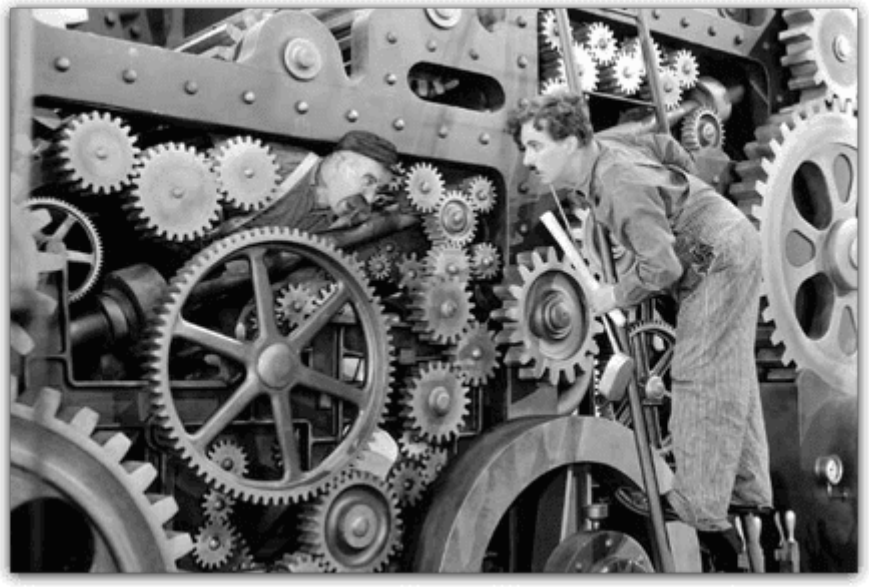
Les **tests** en font partie intégrante.

Nous allons voir que les concepteurs d'Angular ont tout prévu.
Enfin nous déploierons notre application via plusieurs méthodes.

Les Tests : Le secret de mon succès

Créer une application Web c'est comme créer n'importe quel objet.
On crée une voiture, une télévision ou un avion par exemple.
Et avant de le donner à quelqu'un on va tester son fonctionnement.

Les informaticiens se sont dit autant que ce soit simple et automatique.
Comme toujours c'est plus facile à dire qu'à faire.



Un peu d'histoire avant de commencer

Depuis ces débuts le monde informatique a cherché à s'améliorer.
Plusieurs méthodes de travail ont été adoptées.
En simplifiant on pourrait dire qu'on en est là

Méthode **Cycle en V** contre méthode **Agile**.

La plus rapide est-elle vraiment celle qu'on croit ?

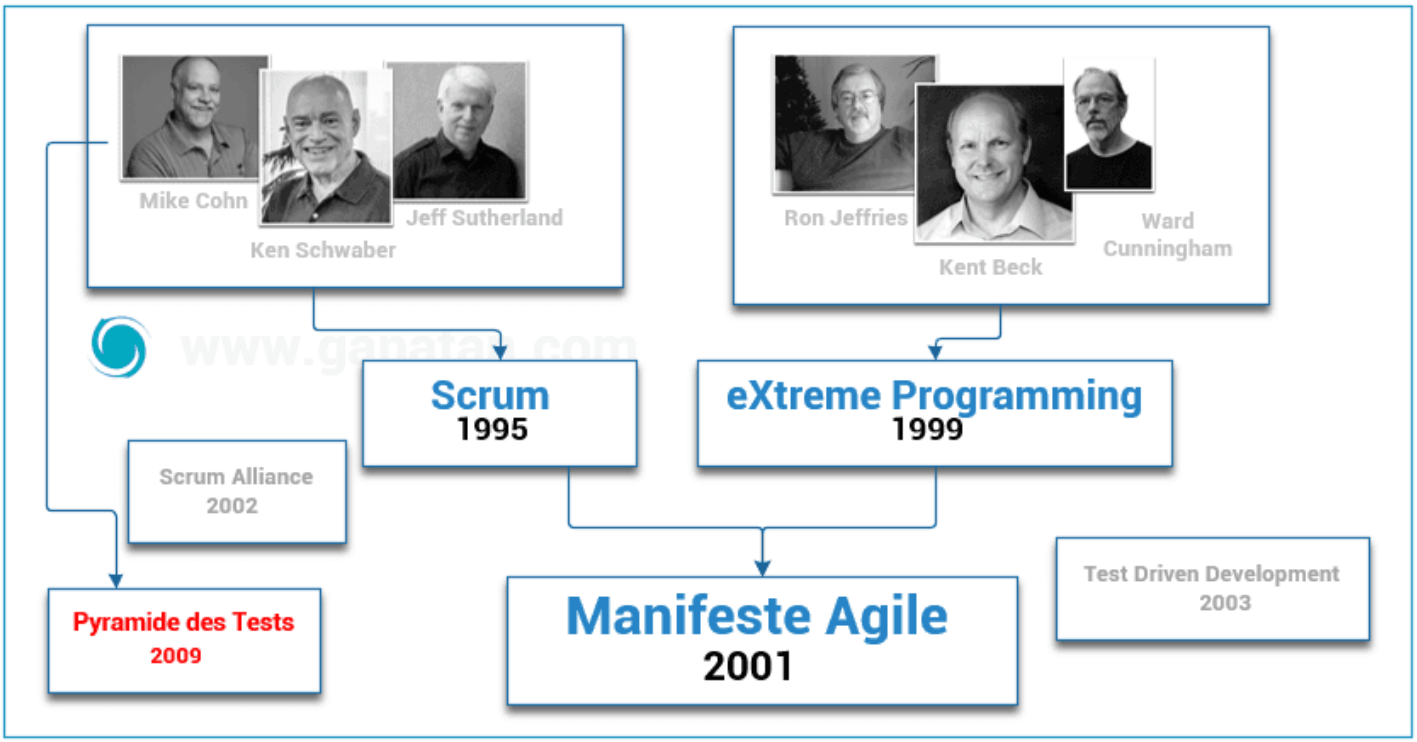
Cycle en V
1991

Agile
2001

Mais qui a fait quoi ?

Si vous voulez travailler dans la programmation c'est sûr il va falloir **se montrer agile**.
La méthode Agile la plus utilisée actuellement est la **méthode Scrum**.

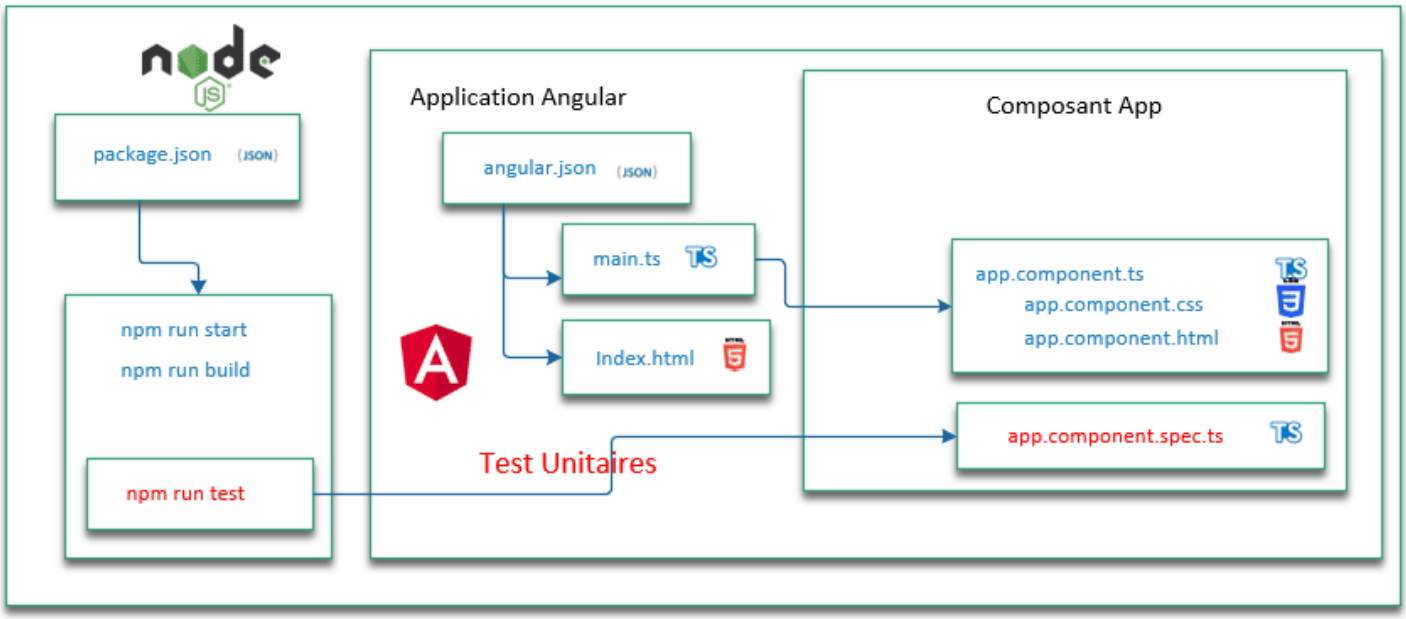
Ci dessous un petit historique des vingt dernières années.



En Novembre 2009 **Mike Cohn** décrit la **pyramide des tests** (**The test pyramide** en anglais) dans son livre
Succeeding with Agile: Software Development Using Scrum

Avec Angular nous nous intéresserons à une catégorie.
- Les test unitaires

Faisons un survol en image de l'architecture d'Angular pour visualiser les tests.



Les tests avec Angular

Sans rentrer dans les détails Angular nous simplifie la vie avec les outils suivants.

Les tests unitaires utilisent
- **Karma**
- **Jasmine**

Les tests end-to-end utilisaient
- **Protractor**

Remarque : Depuis angular 12 les tests end-to-end ont été désactivés.

La documentation Angular concernant la notion de couverture est accessible à cette adresse https://angular.dev/guide/testing/code-coverage

Je rajoute dans **package.json** un script supplémentaire pour tester la couverture

"**coverage**": "ng test --no-watch --code-coverage",

Pour les lancer on utilise les scripts correspondants contenus dans le fichier Package.json

```
# Tests unitaires
npm run test

# Tests unitaires avec couverture
npm run coverage
```

Modifier et vérifier

Effectuons

- Un test simple modification et de débuggage
- Un test de contrôle de code source.
- Débuggage.

Toute modification entraine une recompilation du code.

Par exemple Modifier le fichier app.component.html
Congratulations and Modifications! Your app is running.

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

Remarque :

Le fichier favicon.ico représente l'icône de votre application.

Vous pouvez le personnaliser.

Dans cet exemple vous pouvez récupérer celui de ce dépôt.

```
# Executer
npm run start

# Tester
http://localhost:4200/

# Effectuer des modifications
```

Vérification du code

En tant qu'informaticien on essaiera de se simplifier la vie.
Autant avoir de l'aide pour écrire notre code.

L'un des outils utilisés est le linting qui permet d'améliorer la qualité du code.

Angular utilisait l'outil **TSLint** accessible à
cette adresse https://palantir.github.io/tslint/

Remarque
Cette commande a été désactivée à partir d'Angular 12.
On attend le prochain outil préconisé par angular (probablement ESLint ?).

Comme l'équipe de Google ne s'est pas décidé nous intégrerons ESLint dans notre projet.

Pour cela nous utiliserons Schematics qui est un générateur de code Angular basé sur des templates de base.

La commande ng lint exécute l'analyse statique du code source TypeScript.

```
# Installation via schematics
ng add @angular-eslint/schematics

# Répondre oui à la question
The package @angular-eslint/schematics@19.6.0 will be installed and executed.

# Test du code source
npm run lint
```

Schematics a rajouté un script dans le fichier **package.json (lint)**
et a créé un fichier **eslint.config.js**

Pour vérifier que notre linter fonctionne
Rajoutons des règles spécifiques dans le fichier **eslint.config.js**

Remarque modifiez cette propriété

- "no-var": "**error**"
  une erreur sera signalisée sur l'utilisation de **var**
- "no-var": "**off**"
  Aucune erreur ne sera signalisée sur l'utilisation de **var**

Pour vérifier le comportement du linter.

Modifions un fichier par exemple **app.component.ts**
Et écrivons du code qui ne correspond pas aux règles

On teste avec le script *npm run lint* qui donnera une erreur

Unexpected var, use let or const instead no-var

eslint.config.js

```
    rules: {
      "@angular-eslint/directive-selector": [
        "error",
        {
          type: "attribute",
          prefix: "app",
          style: "camelCase",
        },
      ],
      "@angular-eslint/component-selector": [
        "error",
        {
          type: "element",
          prefix: "app",
          style: "kebab-case",
        },
      ],
      "@angular-eslint/component-class-suffix": [
        "error",
        {
          suffixes: ["", "Component"]
        }
      ],
      "semi": ["error", "always"],
      "comma-dangle": ["error", "always-multiline"],
      "no-undefined": "error",
      "no-var": "error",
      "prefer-const": "error",
      "func-names": "error",
      "id-length": "error",
      "newline-before-return": "error",
      "space-before-blocks": "error",
      "no-alert": "error"
    },
```

app.ts

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

@Component({
  selector: 'app-root',
  imports: [RouterOutlet],
  templateUrl: './app.html',
  styleUrl: './app.css'
})
 export class App {
  protected title = 'angular-starter';


  checkError() {
    var err = 10;

    return err;
  }
}
```

Environment

L'équipe Angular a décidé de ne plus intégrer automatiquement **les paramètres d'environnement (environment en anglais).**
Les nouveaux développeurs moins portés sur les problèmes de configuration n'utilisaient pas ces paramètres.

La documentation Angular est accessible à cette adresse
https://angular.dev/tools/cli/environments#configure-environment-specific-defaults

La commande est la suivante **ng generate environments**

Les éléments nécessaires à son fonctionnement sont créés automatiquement.
- Création d'un répertoire **src/environments**
- Création d'un fichier **environment.development.ts**
- Création d'un fichier un fichier **environment.ts**
- Modification du fichier **angular.json**

environment.development.ts

```
export  const environment = {};
```

environment.ts

```
export  const environment = {};
```

angular.json

```
      "configurations": {
        "production": {
          "budgets": [
            {
              "type": "initial",
              "maximumWarning": "500kB",
              "maximumError": "1MB"
            },
            {
              "type": "anyComponentStyle",
              "maximumWarning": "4kB",
              "maximumError": "8kB"
            }
          ],
          "outputHashing": "all"
        },
        "development": {
          "optimization": false,
          "extractLicenses": false,
          "sourceMap": true,
          "fileReplacements": [
            {
              "replace": "src/environments/environment.ts",
              "with": "src/environments/environment.development.ts"
            }
          ]
        }
      },
```

Déploiement

Tout ce que nous avons fait est bien sympathique.

Mais une application Web n'a d'intérêt que si nous la rendons accessible sur le Web.

C'est ce que l'on appelle le **déploiement**.

Nous allons voir comment le faire via deux méthodes de **la plus simple** à **la plus compliquée.**

Mais tout d'abord parlons **compilation**.

Comme nous l'avons vu précédemment le fichier package.json contient un certain nombre de scripts (ou commandes).

Le script qui nous intéresse est **npm run build**

Il permet de compiler notre application.

Ce script exécute la commande d'Angular CLI **ng Build**

Sans rentrer dans les détails voilà comment ça fonctionne.
Via cette commande Angular utilise l'outil Webpack (un module bundler) pour créer le produit final.
L'exécution de cette commande va créer un répertoire dist.
Celui-ci contiendra ce que l'on peut appeler le produit final (ou livrable ou artefact).
C'est cette partie que l'on va déployer.
Les conseils donnés par Angular sont à l'adresse suivante

https://angular.dev/tools/cli/deployment

Déploiement avec lite-server

Le déploiement le plus simple est d'utiliser le serveur Http développé par John Papa.

Comment procéder ?

- On installe la librairie lite-server en global avec npm
- On exécute l'application en mode production

```
# Compilation du projet !!!!!!!! Très important à ne pas oublier
npm run build

# Installation du serveur de développement lite-server
npm install -g lite-server

# Exécution de notre application
lite-server --baseDir="dist/angular-starter/browser"

# Tester l'application dans notre navigateur avec l'url suivante
http://localhost:3000/
```

Deploiement avec nginx

Une solution plus complexe mais plus proche de la réalité.

Il nous faudra disposer d'un serveur virtuel ou VPS (Virtual private server).

Je vous conseille d'en acheter un chez un fournisseur de VPS.

Par exemple **OVH** ou **Digital Ocean** sont parmi les moins chers et les plus efficaces.

Le tutoriel suivant peut vous être utiles

[Installer Angular sur un serveur Ubuntu]

Sur notre serveur (exemple d'un serveur avec ubuntu et l'adresse ip 192.168.100.1)

- Installer nginx
- Tester nginx
- Copier notre répertoire dist sur /var/www/html
- Tester le serveur

```
# connection sur le serveur en ssh
ssh root@192.168.100.1

# installation de nginx sur le serveur
sudo apt-get --yes install nginx
sudo apt-get update

# Démarre le service nginx
sudo service nginx start

# Tester l'installation du serveur nginx
http://localhost:192.168.100.1/

# Copier le contenu du répertoire dist/angular-starter/browser
# sur le serveur dans le répertoire /var/www/html/

# Tester l'application
http://localhost:192.168.100.1/
```

Paramètrage de nginx

Je vous rajoute deux fichiers qui vous seront utiles

- un exemple d'un fichier de configuration **nginx.conf**

- un exemple de fichier javascript **server.js** pour lancer en local votre application

  A utiliser avec la commande **node server.js**

  Le script suivant est à rajouter dans package.json

  "serve": "node server.js"

fichier nginx.conf

```
user www-data;
worker_processes auto;
pid  /run/nginx.pid;
error_log /var/log/nginx/error.log;
include /etc/nginx/modules-enabled/*.conf;

events {
  worker_connections 768;
}

 http {
  sendfile on;
  tcp_nopush on;
  types_hash_max_size 2048;

  include /etc/nginx/mime.types;
  default_type application/octet-stream;

  ssl_protocols TLSv1 TLSv1.1 TLSv1.2 TLSv1.3; # Dropping SSLv3, ref: POODLE
  ssl_prefer_server_ciphers on;

  access_log /var/log/nginx/access.log;
  gzip on;
  include /etc/nginx/conf.d/*.conf;
  server {
    listen 80 default_server;
    listen [::]:80 default_server;

    root /var/www/html;
    index index.html index.htm index.nginx-debian.html;
    server_name _;
    location / {
      try_files $uri $uri/ =404;
    }

  }
}
```

server.js

```
const  express = require('express');
const  path = require('path');
const app = express();

app.use(express.static(path.join(__dirname, 'dist/angular-starter/browser')));

app.get('/*', function (req, res) {
  res.sendFile(path.join(__dirname, 'dist/angular-starter/browser', 'index.html'));
});

const port = 4000;
const host = 'localhost';
app.listen(port, () => {
  console.log(`Server running at http://${host}:${port}`);
})
```

Etape 5

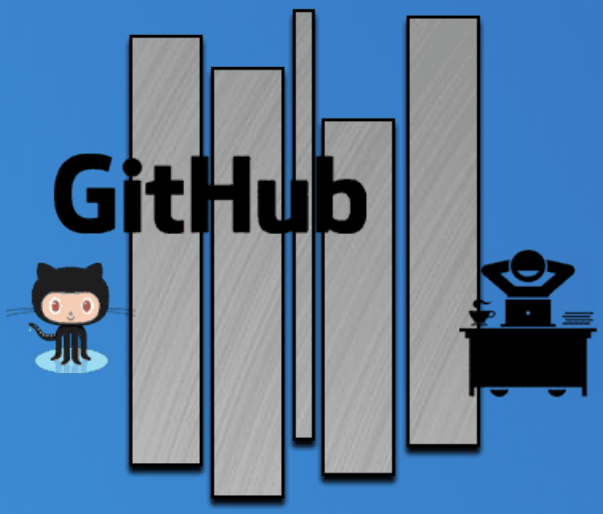
Code Source

Ce guide nous a permis de créer une application Web prête à fonctionner.

Pour vous simplifier la tâche vous pouvez utiliser directement le code source de cette application, pour la tester et vérifier qu'elle fonctionne.

Pour cela il suffit d'utiliser **le logiciel Git**.

Je vais vous montrer comment procéder.



Cette première application reste néanmoins basique.

Je vous proposerai pour terminer un certain nombre d'étapes qui vous permettront de créer une application plus complexe.

Utiliser Git avec le code source

En suivant chacun des conseils que je vous ai donnés dans ce guide vous obtenez au final un code source Angular.

Ce code représente votre travail et doit faire l'objet de toute votre attention.

Comme nous l'avons vu précédemment, **Git** va nous permettre de gérer tous nos codes sources.

Un petit tour sur Wikipedia https://en.wikipedia.org/wiki/GitHub
nous apprend que

GitHub est le plus grand hébergeur de code source au monde.

En janvier 2023 on compte

- 100 millions d'utilisateurs
- Plus de 420 millions de dépôts (ou **repositories** en anglais)

Je vous conseille donc de publier vos sources sur cet hébergeur.

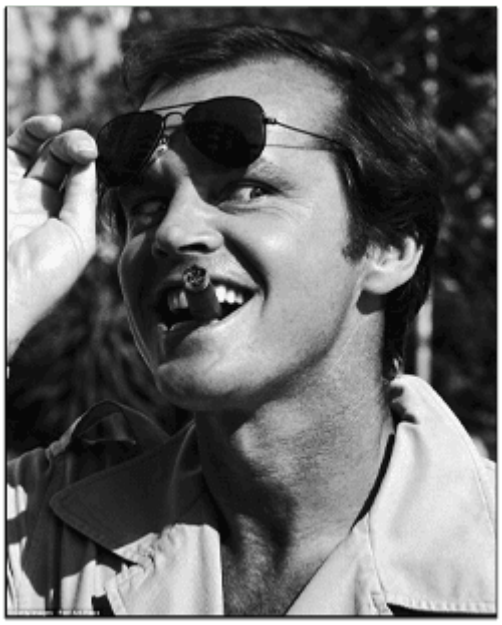
Le **code source** de ce tutoriel est bien évidemment disponible sur GitHub.

Utilisez git pour récupérer ce code et vérifier son fonctionnement.

Il vous suffit de vous rendre à l'adresse suivante

https://github.com/ganatan/angular-react-starter

Si ce guide vous a plu et que vous allez sur GitHub pour consulter le code, n'hésitez pas à cliquer sur STAR .



Sinon pour aller plus vite encore suivez les conseils suivants.

Utilisez la commande classique de **prompt** sous windows (cmd) ou linux.
Puis tapez la liste des commandes

```
# 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-react-starter.git

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

# 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/
```

Pour aller plus loin

Ce tutoriel nous a permis de créer notre première application.

Celle-ci reste relativement simple.

Si vous voulez créer une application plus complète, il vous faudra mettre en œuvre certains principes et fonctionnalités supplémentaires comme

- Le **Routing** (gestion de plusieurs pages)
- Le **Lazy loading** (rapidité de l'application)
- Les **PWA** (fonctionnement sur mobile et desktop)
- Le **Server Side Rendering** (permettre le référencement)

L' étape suivante est logiquement la gestion du Routing.

Elle nécessite un tutoriel complet qui est à l'adresse suivante

- Etape 2 : Routing avec Angular

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

- Etape 3 : Lazy loading avec Angular
- Etape 4 : Bootstrap avec Angular
- Etape 5 : Modules avec Angular
- Etape 6 : Server Side Rendering avec angular
- Etape 7 : Progressive Web App avec Angular
- Etape 8 : Search Engine Optimization avec Angular
- Etape 9 : HttpClient avec Angular

Les étapes suivantes vous permettront d'améliorer ce prototype

- Components avec Angular
- Services avec Angular
- Template Driven Forms avec Angular
- Charts avec Angular

Cette dernière étape permet **d'obtenir un exemple d'application**

- Créer une application Web complète avec Angular

Le **code source de cette application finale** est disponible sur GitHub https://github.com/ganatan/angular-node-java-ai

The end