Notes

1. **app.component.ts** est le fichier source qui décrit le composant racine de l'application, c'est-à-dire le composant Angular de premier niveau de l'application. **Un composant** : est le bloc de construction de base d'une application Angular. La description du composant comprend le code du composant, le modèle HTML et les styles, qui peuvent être décrits dans ce fichier ou dans des fichiers séparés.
2. Le répertoire **(/assets)** contient les images utilisées par l’application
3. Création d’un nouveau composant HomeComponent :

* **ng generate component Home --standalone --inline-template --skip-tests** ou
* **ng g component Home --standalone --inline-template --skip-tests**
* **standlone = autonome, la commande ci-dessus crée un composant autonome**

1. Création du composant HousingLocation :

**ng g component HousingLocation --standalone --inline-template --skip-tests**

1. Les interfaces sont des types de données personnalisées pour l’application Angular.

Pour créer une interface dans le dossier interfaces : **ng g interaface interfaces/housinglocation**

1. **@Input() test!: Test** -> le **!** implique que l'entrée s'attend à ce que la valeur soit transmise. Dans ce cas, il n'y a pas de valeur par défaut. Dans notre exemple d'application, nous savons que la valeur sera transmise - c'est une question de conception. **Le point d'exclamation est appelé l'opérateur d'assertion non-null et il indique au compilateur TypeScript que la valeur de cette propriété ne sera pas nulle ou indéfinie**.
2. Le modèle de liaison dans Angualr est synchronisation antre le modèle et la vue (template). Il existe plusieurs types de liaisons de données dans Angular :

* Interpolations de textes (pour l’affichage des données) **{{ expression }}**
* Liaison de propriété **[attribute] = "value"**
* Liaison événementielle
* Liaison bidirectionnelle

1. **ngFor** : est un type spécifique de [directive](https://angular.io/guide/built-in-directives) utilisé pour répéter dynamiquement des données dans un modèle. En JavaScript, on utilise une boucle **for** - **ngFor** fournit une fonctionnalité similaire pour les modèles Angular. On peut utiliser [ngFor](https://angular.io/api/common/NgFor) pour itérer sur des tableaux et même sur des valeurs asynchrones.
2. **Les services Angular** : permettent de séparer les données et les fonctions de l'application Angular qui peuvent être utilisées par plusieurs composants de votre application. Pour être utilisé par plusieurs composants, un service doit être rendu injectable. Les services qui sont injectables et utilisés par un composant deviennent des dépendances de ce composant. Le composant dépend de ces services et ne peut pas fonctionner sans eux.

**L'injection de dépendances** : est le mécanisme qui gère les dépendances entre les composants d'une application et les services que les autres composants peuvent utiliser.

Pour créer un nouveau service nommé Housing : **ng g service housing --skip-tests**

1. Routage

Le routage est la capacité de naviguer d'un composant de l'application à un autre. Dans les [applications à page unique (SPA)](https://angular.io/guide/router-tutorial#using-angular-routes-in-a-single-page-application), seules certaines parties de la page sont mises à jour pour représenter la vue demandée par l'utilisateur.

* Pour créer un composant pour voir les détails de la maison : **ng g component housing-location/housing-details/details --standalone --inline-template --skip-tests**
* La directive [routerLink](https://angular.io/api/router/RouterLink) permet au routeur Angular de créer des liens dynamiques dans l'application. La valeur attribuée à [routerLink](https://angular.io/api/router/RouterLink) est un tableau à deux entrées : la partie statique du chemin et les données dynamiques.

1. Intégration de http et d’une API dans une application Angular

* JSON Server est un outil open source utilisé pour créer des API REST fictives.

**npm install -g json-server**

* Créer un fichier nommé dbname.json et remplir ce fichier par des données au format json

Pour tester la configuration, exécuter la commande : **json-server --watch db.json**

* Vérifier que les données insérer dans le fichier dbname.json safficher sur le navigateur à l’adresse : <http://localhost:3000/locations>
* Liens documentation de JSON server : <https://www.npmjs.com/package/json-server>