



Billetterie



Découverte du la CLI

Angular CLI

- ▷ Pour ce TP, plus d'IDE en ligne, vous développerez en local avec votre IDE
- ▷ Il est devenu très simple de générer un projet Angular, pour ce faire, télécharger la Angular CLI qui vous fournira la commande « ng » en ligne de commande.
 - › Angular CLI comme « Angular Command Line », une aide en ligne de commande pour angular

```
> npm install -g "@angular/cli"
```

Angular CLI – Générer vos projet et composants

- ▷ Une fois la « Angular CLI » installé, générez votre premier projet :

```
> ng new NOM_DE_VOTRE_PROJET
```

- ▷ Au moment de la création, la CLI vous posera des questions
 - › Répondre « N » pour les routing, nous verrons ce chapitre plus tard ;
 - › Pour le reste, répondez en fonction de vos préférences 😊

- ▷ Une fois votre projet généré, il vous sera également possible de générer composants / services / pipes etc.

```
> ng generate component NOM_DU_COMPOSANT
```

Quelques astuces pour Angular CLI

- ▷ En générant une nouvelle entité, vous pouvez spécifier dans quel dossier vous souhaitez que celle-ci soit générée

```
> ng generate component home/shared/comps/mon-composant
```

- ▷ On peut écrire ça encore plus rapidement

```
> ng g c home/shared/comps/mon-composant
```

Macademia

Lancer votre site

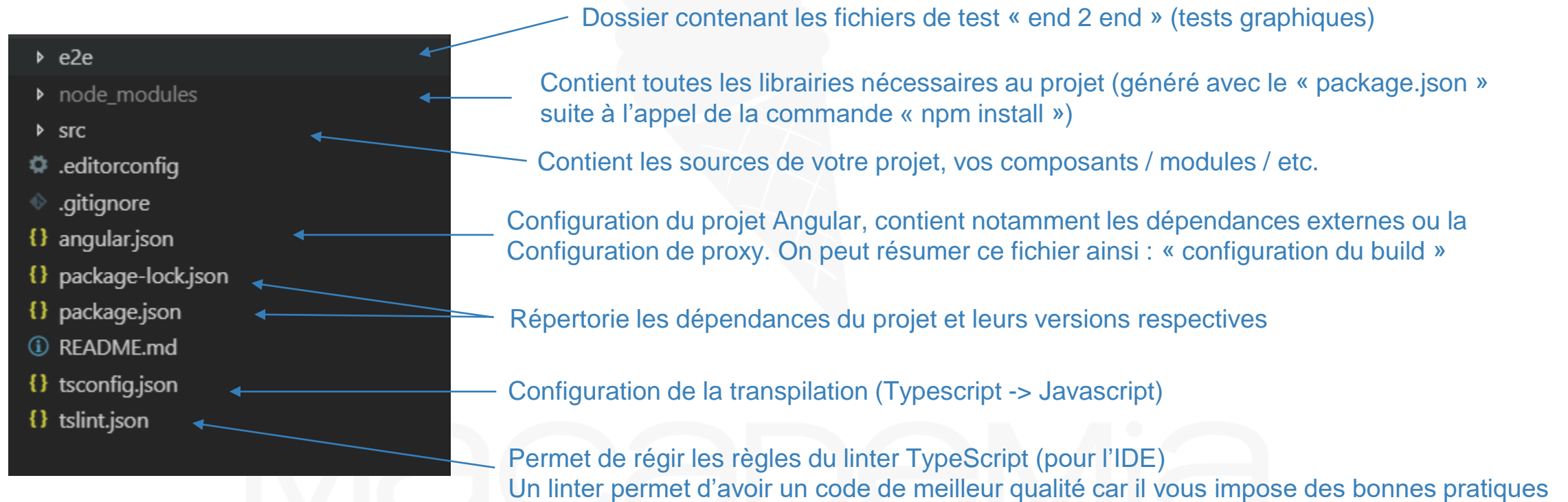
- ▷ Une fois le projet créé, la Angular CLI vous donne la possibilité de lancer un serveur local :

```
> ng serve
```

- ▷ Cette simple commande, quand elle est lancée à la racine de votre projet, permettra :
 - › De lancer un serveur local qui fera tourner votre application Angular ;
 - › D'appliquer un watcher sur TOUS vos fichiers qui permettra de relancer automatiquement le serveur à chaque modification que vous réalisez ;
 - › De transpiler automatiquement vos fichiers en javascript !

Architecture du projet

▷ Après la génération du projet, c'est une multitude de fichiers / dossiers qui apparaissent sous nos yeux ébahis



The diagram shows a file explorer view of an Angular project. The files and folders are listed on the left, and blue arrows point from descriptive text on the right to the corresponding items in the file explorer.

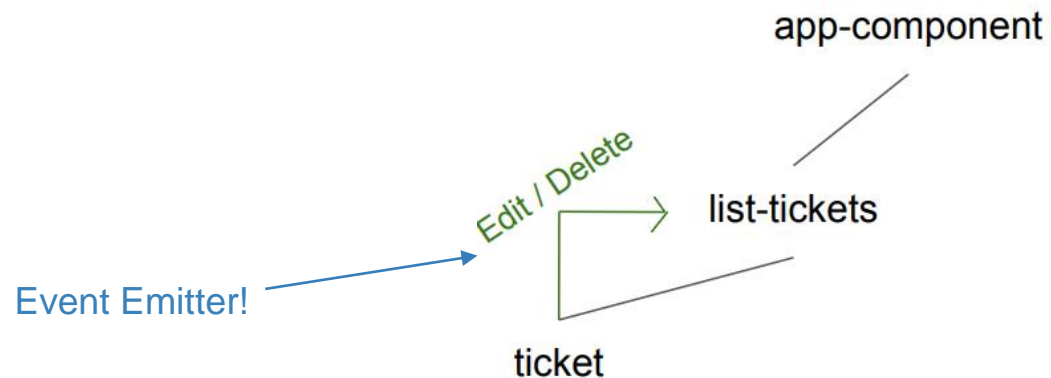
- e2e**: Dossier contenant les fichiers de test « end 2 end » (tests graphiques)
- node_modules**: Contient toutes les librairies nécessaires au projet (généré avec le « package.json » suite à l'appel de la commande « npm install »)
- src**: Contient les sources de votre projet, vos composants / modules / etc.
- .editorconfig**: Configuration du projet Angular, contient notamment les dépendances externes ou la Configuration de proxy. On peut résumer ce fichier ainsi : « configuration du build »
- .gitignore**: Répertorie les dépendances du projet et leurs versions respectives
- angular.json**: Configuration de la transpilation (Typescript -> Javascript)
- package-lock.json**: Permet de régir les règles du linter TypeScript (pour l'IDE)
- package.json**: Un linter permet d'avoir un code de meilleure qualité car il vous impose des bonnes pratiques
- README.md**
- tsconfig.json**
- tslint.json**



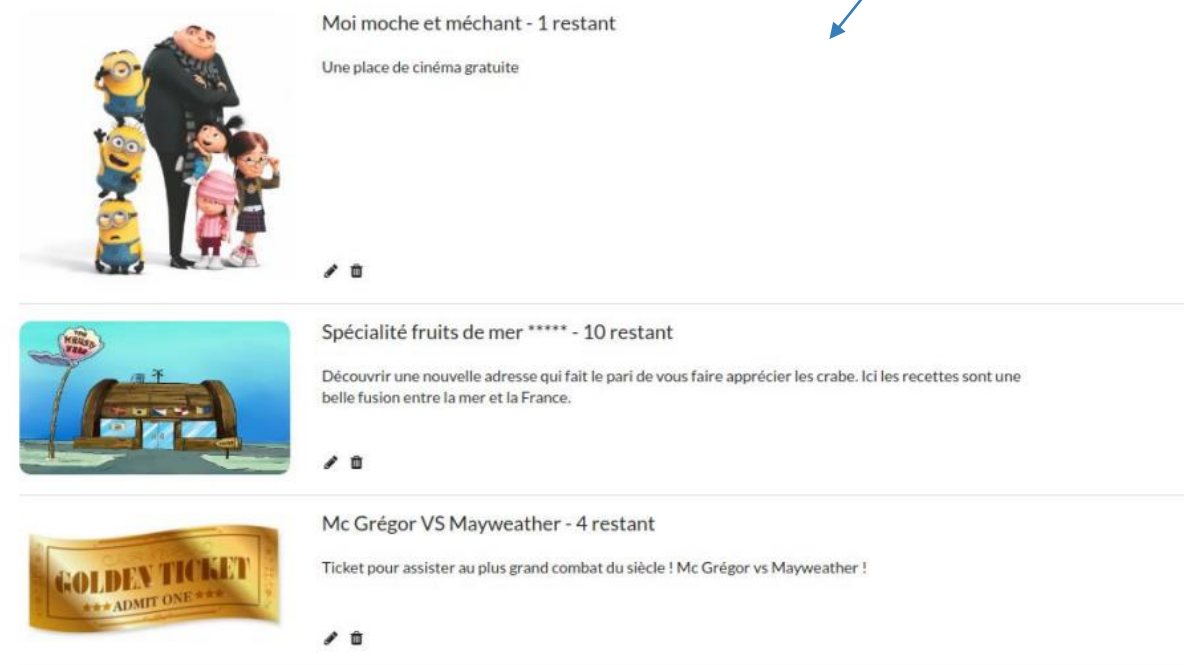
Les composants

Mise en place de la vue




▷ Commencez par créer vos composants de sorte à obtenir cette architecture :



Résultat attendu



Bonus

- ▷ Créer une classe pour la gestion des tickets 
- ▷ Créer un fichier qui exporte un tableau de tickets (mock) 
- ▷ Respecter les bonnes pratiques pour la hiérarchie de fichiers 

Macademia



Gestion de formulaires

Création d'un formulaire de connexion

- ▷ Notre application de billetterie **vaut de l'or** !
- ▷ Il va être nécessaire de **sécuriser** notre application
- ▷ Pour ce faire, nous implémenterons les Model Driven Forms
 - › **Pensez à intégrer le module « ReactiveFormsModule »**
- ▷ Le formulaire protégera l'accès à la page « List Tickets »
 - › Gérée par le composant « ListTickets » développé hier



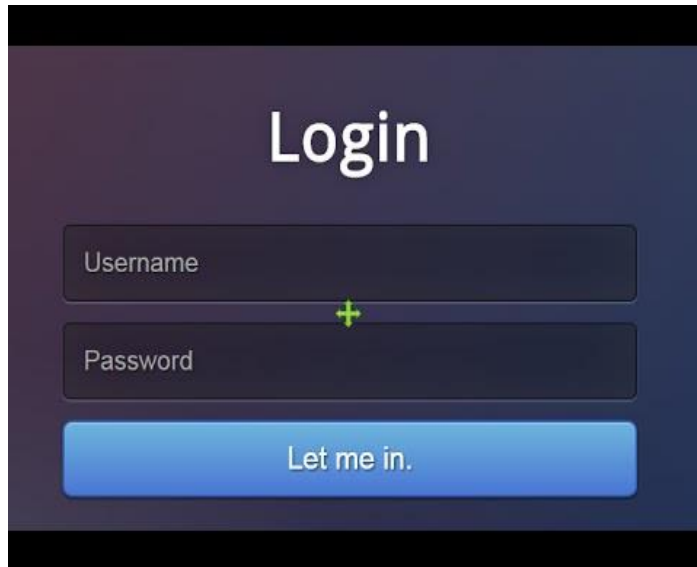
Connexion

admin

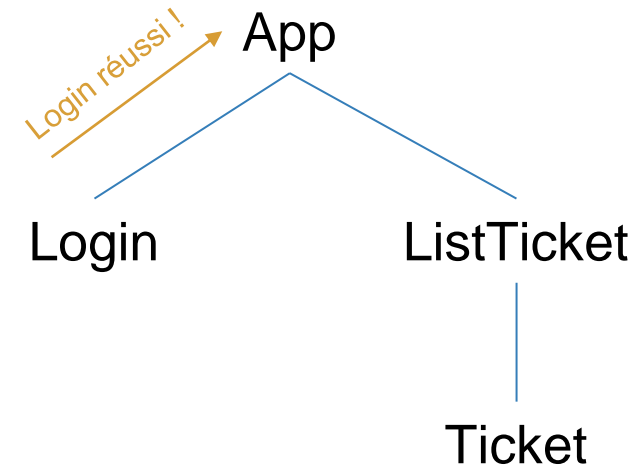
.....

Valider

Formulaire



A screenshot of a login form titled "Login". It features two input fields: "Username" and "Password". A green plus sign is visible between the two fields. Below the fields is a blue button labeled "Let me in.".



- ▷ L'affichage entre « Login » et « ListTicket » est **conditionnel**
 - › Pensez au `*ngIf`

Mock

- ▷ Contentez-vous pour le moment de travailler en local
- ▷ Pour que login fonctionne et redirige correctement
- ▷ Nous verrons plus tard comment relier notre formulaire à un serveur !

Macademia



Services

Un service Tickets

- ▷ En Angular, l'algorithmie et la gestion de données doivent être centralisées dans les **services**
 - ▷ Il va donc être nécessaire de factoriser votre application afin de déléguer le traitement du composant ListTicket vers TicketService
1. Commencez par créer le service
 2. Puis injectez-le
 3. Enfin, copiez-collez le code pour déplacer la logique

Un exemple d'implémentation

▷ Voici une proposition :



TicketService

```
tickets: Ticket[];  
loadTickets(void): Promise<Ticket[]>;  
getOne(index: number): Ticket;  
edit(index: number, newT: Ticket): boolean;  
delete(index: number): boolean;  
add(ticket: Ticket): boolean;
```



Pipes

Créez votre premier pipe

- ▷ Le client souhaite mettre en avant les tickets
- ▷ Pour ce faire, il souhaite afficher **une lettre sur 2 en majuscule !**
- ▷ Développez un pipe qui nous permettra de répondre à cette demande loufoque

Macademia



Communication back-end

HttpClientModule

▷ Nous allons relier le formulaire de login à un véritable backend

▷ Pour ce faire, un serveur a été implémenté à cette adresse :

<https://serveur-demo.macademia.fr/>

▷ Pour communiquer avec un serveur avec Angular, il est nécessaire d'importer un nouveau module « HttpClientModule »

› Vous trouverez plus d'informations sur la [doc officielle](#)

HttpClient

- ▷ Il vous sera ensuite possible d'injecter dans vos composants un nouveau service : **HttpClient**

```
this.http.get('https://serveur-demo.macademia.fr').subscribe((value) => {  
  console.log(value); // {response: 'Vide'}  
})
```

- ▷ Utilisez ce service pour communiquer avec le serveur afin de déterminer si les identifiants entrés par le client sont corrects ou non;

Macademia

Les routes fournies

▷ Voici les différentes routes fournies par le serveur :



/login

POST

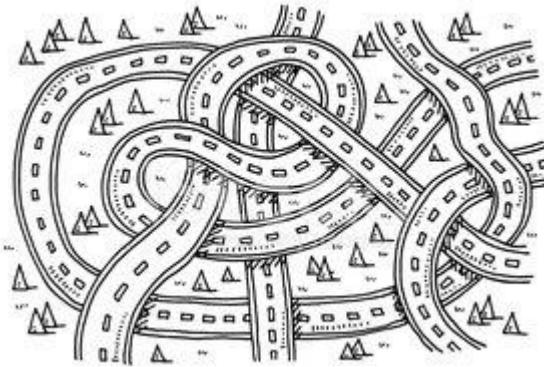
```
1 {  
2   "pseudo": "admin",  
3   "password": "password"  
4 }
```

```
"response": {  
  "messages": "connexion réussie",  
  "time": 1519761382,  
  "token": "montoken",  
}
```

/whoAmI

GET

```
"user": {  
  "pseudo": "admin",  
  "created_at": 1465800700,  
  "gender": "male",  
  "role": "admin",  
}
```



Gestion du routing

Implémentation des routes

- ▷ Il nous faut désormais implémenter le routing !
- ▷ Plusieurs pages doivent être disponibles :
 - › Page **Login** ;
 - › Page **Home** (vue des tickets) ;
 - › Page **Ajout de Ticket** ;
 - › Page **Edition de Ticket** ;
- ▷ Les **routes en bleues** ne doivent être accessibles qu'en étant connecté
- ▷ Les **routes en jaunes** peuvent être accessible tout le temps

Une fois connecté...

► Voici ce que verra un utilisateur connecté :

- Liste tickets - Ajouter un ticket - Déconnexion -

Tous les tickets dispos

	<p>Moi moche et méchant X Edit</p> <p>Il reste 1 ticket(s) Expire le 17/08/2019 Une place de cinéma gratuite</p>
	<p>Voyage en tunisie X Edit</p> <p>Il reste 10 ticket(s) Expire le 17/08/2019 Vous rêvez de vacances sur la plage? Ce ticket vous offre une réduction de 20% sur votre voyage en Tunisie.</p>