

Cycle Ingénieur Polytech Lyon
Par Rémi DA COSTA

TD3 : Initiation à Angular

Contexte

Vous travaillez pour Air France, une compagnie aérienne qui fait voler des milliers d'avions dans le monde par an. Air France cherche à lister tous les vols qui décollent ou atterrissent dans les aéroports français. Cela dans le but d'optimiser le nombre d'avions à disponibiliser et de vérifier dans quels aéroports ils ont la plus grande concentration d'avions.

Le but de cette application sera dans un premier temps de lister tous les avions qui décollent d'un aéroport donné en paramètre entre une date de début et une date de fin. Ces 3 critères de recherche seront présents dans une section « Filtre ».

Nous utilisons un jeu de données en Open Data afin de récupérer le champ icao, un identifiant unique qui identifie le matricule d'un avion. Pour se faire, nous définissons un service pour venir faire une requête http sur ce jeu de données. Nous allons récupérer aussi l'aéroport de destination si celui-ci est connu pour l'afficher dans l'application.

Il sera possible de sélectionner un vol dans la liste, ainsi la liste des passagers de ce vol devra s'afficher dans la partie droite de l'écran. On pourra alors visualiser les passagers, la classe de leur vol (business, premium ou standard) et le nombre de bagages qu'ils emportent en soute.

Ensuite, il faudra faire la même chose mais pour l'atterrissage dans un aéroport et dans une période donnée.

Le développeur en charge de ce projet est tombé malade et est en arrêt maladie. Pour que le projet ne prenne de retard, on vous demande d'avancer sur les points indiqués dans ce document.

Le précédent développeur a quand même mis en place la structure du code et documenté son code pour qu'il soit moins compliqué pour vous de continuer.

Consignes

Le TD peut être fait seul ou en groupe (maximum 3 personnes). Aucun document n'est à rendre, seul le projet devra être livré avant le **16/05/2022 23h59**. Vous devrez le mettre à disposition sur un gestionnaire de versions au choix : Gitlab, Github, etc.

Le barème est indiqué sur chaque partie. Le rendu du TD comptera pour 60% de la note du cours, le questionnaire de fin de cours comptera pour 40%.

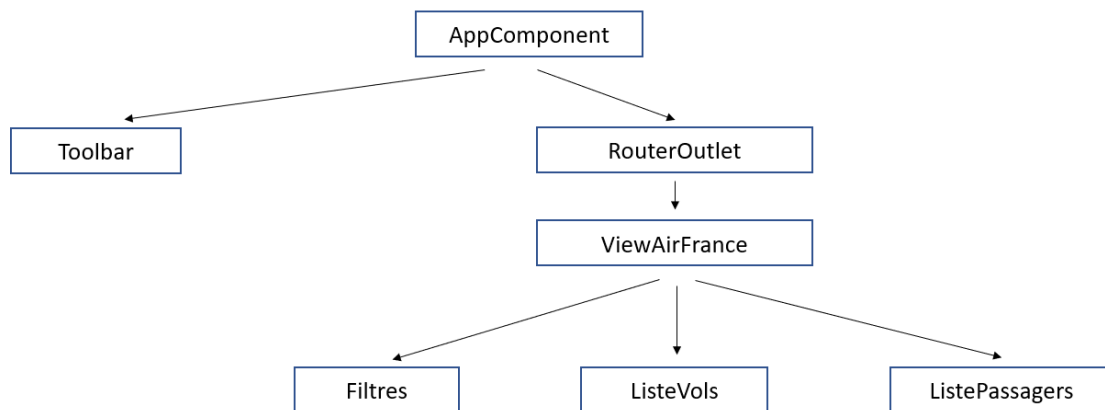
Mise en place du projet

Allez sur gitlab et faites un `git clone` du projet du TD3 à un endroit sur votre PC.

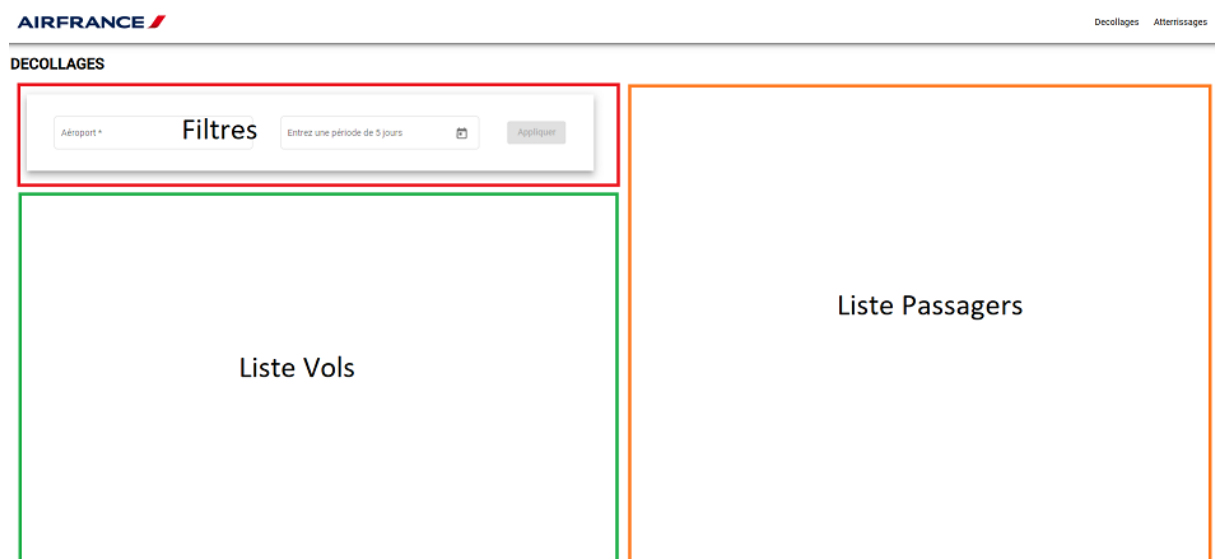
Lancez la commande `npm install` pour télécharger les dépendances nécessaires au projet.

Lancez ensuite le script `npm start` et connectez-vous sur **localhost:4200**.

Les composants disponibles sont hiérarchisés de la manière suivante :



Voici le rendu de l'application :



Gérez les décollages (6 points)

Par défaut nous arrivons sur la page des décollages. Il sera possible plus tard de choisir décollage ou atterrissage depuis le menu en haut de page.

App-routing.module.ts

```
const routes: Routes = [
  { path: 'decollages', component: ViewDecollagesComponent },
  { path: '**', redirectTo: 'decollages' }
];
```

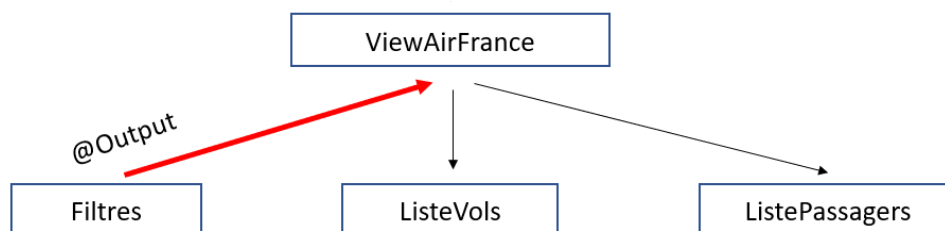
La section des filtres est déjà fonctionnelle. On peut choisir un aéroport parmi une liste fixe des principaux aéroports français. Ensuite nous pouvons choisir une période, le choix de la période ne doit pas excéder 7 jours, sinon l'API de récupération des informations de vol renverra une erreur. Nous avons donc choisi arbitrairement une période de 3 jours.

Interrogez l'API open data

Le composant de Filtres possède un attribut d'**Event binding** : `filtresEvent`.

```
export class FiltresComponent {
  @Output() filtersEvent = new EventEmitter<IFiltres>();
}
```

Nous allons pouvoir récupérer les filtres après chaque clic sur le bouton « Appliquer ». Encore faut-il récupérer le changement de filtres dans le composant parent : **ViewAirFrance**.



La fonction **onFiltresEvent** doit être complétée pour appeler le service **VolService**, nous utiliserons les 3 critères récupérés pour requêter l'API d'informations des vols.

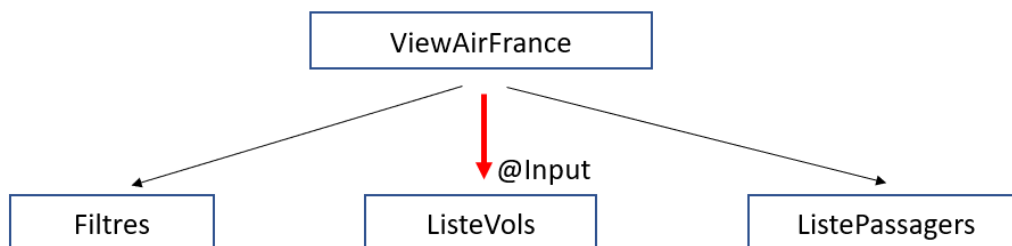
Dans le service **VolService** vous retrouverez la fonction `getVolsDepart`, utilisez-la en y renseignant les bons attributs. La fonction `getVolsDepart` retourne un observable, il va falloir souscrire à cet observable pour récupérer la liste des vols.

Attention : Les attributs début et fin, sont au format `Date`, cependant l'API open source attend un temps exprimé en seconde.

Note : Dans le cours nous avons vu que toute souscription devait être stockée pour être détruite convenablement et donc éviter les fuites mémoires, **mettez en place un moyen de se désabonner d'une souscription lorsque le composant est détruit.**

Affichez la liste des vols

Nous avons maintenant récupéré la liste des vols lorsque nous utilisons des filtres. Nous allons mettre en place un **Property Binding** sur le composant **ListeVols** pour lui passer cette liste à afficher.



Une fois que c'est fait, nous allons appeler le composant **Vol** autant de fois qu'il y a de vols dans notre liste. Le composant Vol sera un *Dumb Component*, il se contentera de l'affichage des données qu'on lui donne.

Complétez le composant Vol pour qu'il reçoive un objet Vol du composant parent et qu'il l'affiche. Les attributs qui doivent être affichés sont : le numéro icao, le matricule de l'avion, sa compagnie aérienne, l'aéroport de départ (qui est forcément connu), et l'aéroport de destination (qui n'est pas forcément connu), si celui-ci est null, on affichera « Inconnu ».

Air France est un grand groupe il possède des filiales comme : Air France Hop et Transavia France. Il faut donc pouvoir afficher ces compagnies aussi. Les différentes valeurs de l'attribut compagnie sont : « Air France », « Air France Hop » et « Transavia France », vous pouvez retrouver ces libellés dans le fichier **compagnie.constants.ts** qui est utilisé pour formater les données.

Vous avez accès à des logos dans le dossier assets pour afficher les bons logos en fonction des différentes compagnies.

Le résultat attendu peut ressembler à cela :

Aéroport *		Entrez une période de 3 jours				Appliquer
Marseille - Marseille Provence Airport		04/05/2022 - 06/05/2022				
✈	3950d0	AFR45KD	LFML → LFPO	AIRFRANCE		
✈	39d305	TVF68HM	LFML → Inconnu	transavia		
✈	3946e5	AFR55VY	LFML → LFPG	AIRFRANCE		
✈	3950c9	AFR94NV	LFML → LFPO	AIRFRANCE		
✈	398572	AFR27BD	LFML → EHAM	AIRFRANCE		

J'utilise l'icône `flight_takeoff` pour représenter un avion en train de décoller.

```
<div class="icon">
  <mat-icon>flight_takeoff</mat-icon>
</div>
```

Enfin j'utilise l'icône `arrow_right_alt` pour représenter la flèche entre les 2 aéroports

```
<div class="fleche">
  <mat-icon>arrow_right_alt</mat-icon>
</div>
```

Vous pouvez retrouver pleins d'icônes compatibles avec Angular Material ici : <https://www.angularjswiki.com/fr/angular/angular-material-icons-list-mat-icon-list/>

Affichez la liste des passagers pour un vol (5 points)

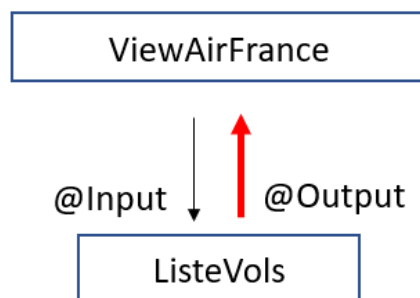
Nous arrivons maintenant à afficher les vols qui décollent d'un aéroport. Mais on souhaite pouvoir afficher les passagers relatifs à un vol.

Nous allons tout d'abord ajouter la possibilité de sélectionner un vol.

Dans le composant `ListeVols`, nous allons ajouter une méthode qui va réagir à un clic souris sur la ligne d'un vol.

Cette méthode va alors émettre le vol sélectionné au composant parent.

Complétez le composant `ListeVols` pour émettre le vol sélectionné au parent.

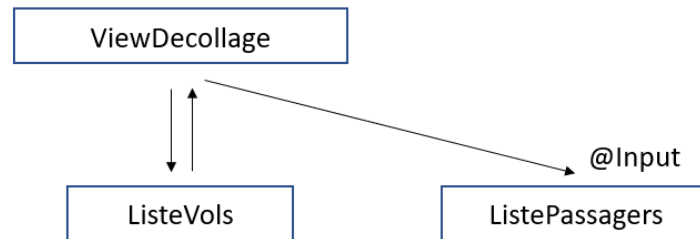


Dans le composant parent, faites le nécessaire pour récupérer le vol sélectionné.

Une fois le vol récupéré nous allons devoir passer par une étape importante avant d'afficher ses passagers. En effet le vol sélectionné ne contient aucun passager... Puisque l'API que nous avons utilisée pour récupérer les informations de vol ne possède pas ces informations.

Nous allons alors appeler un autre service dans notre composant `ViewAirFrance` : **PassagerService**, et utiliser la fonction `getPassagers`. Cette méthode prend en paramètre le numéro icao du vol sélectionné et va retourner un observable qui contient la liste des 20 passagers du vol.

L'appel à l'API de passagers nous retourne maintenant une liste de passagers que nous allons pouvoir envoyer à notre composant enfant ListePassagers.



De manière analogue à l'affichage des vols, nous allons afficher tous les attributs d'un passager : sa photo, son nom, la classe de son vol, ainsi que le nombre de bagage qu'il a en soute.

Un passager peut être en classe Standard, Business ou Premium, ces valeurs se trouvent dans l'énumération ClasseVol.

```

export enum ClasseVol {
  STANDARD,
  BUSINESS,
  PREMIUM
}
  
```

Vous devriez vous retrouver avec un affichage similaire :

	Rasmus Møller	STANDARD	 x1
	Dominic Zahl	PREMIUM	 x0
	Elsa Brun	BUSINESS	 x2
	Craig Hill	STANDARD	 x1

De manière plus générale votre écran devrait ressembler à ça :

AIRFRANCE

Decollages Atterrissages

DECOLLAGES

















Aéroport *

Marseille - Marseille Provence Airport

Entrez une période de 3 jours

04/05/2022 - 06/05/2022

Appliquer

✈	3950d0	AFR45KD	LFML → LFPO	AIRFRANCE		Rasmus Møller	STANDARD	 x1
✈	39d305	TVF68HM	LFML → Inconnu	transavia		Dominic Zahl	PREMIUM	 x0
✈	3946e5	AFR55VY	LFML → LFPG	AIRFRANCE		Elsa Brun	BUSINESS	 x2
✈	3950c9	AFR94NV	LFML → LFPO	AIRFRANCE		Craig Hill	STANDARD	 x1
✈	398572	AFR27BD	LFML → EHAM	AIRFRANCE		Malte Kröger	STANDARD	 x1
✈	3950cc	AFR37DC	LFML → LFPO	AIRFRANCE		Amelia Walker	BUSINESS	 x2
✈	392aeb	AFR78XM	LFML → LFPG	AIRFRANCE		Walterus Glastra	BUSINESS	 x0
✈	393320	AFR25CA	LFML → LFPG	AIRFRANCE		Kaïs Gautier	STANDARD	 x0
✈	3944f7	AFR99XW	LFML → LFPO	AIRFRANCE				
✈	392aed	AFR41JL	LFML → LFPG	AIRFRANCE				
✈	398572	AFR43HC	LFML → EHAM	AIRFRANCE				
✈	3950cd	AFR12NZ	LFML → LFPO	AIRFRANCE				

Félicitations, vous venez de gérer les décollages ! Maintenant vous allez faire tout pareil mais pour les atterrissages.

Gérez les atterrissages (4 points)

Heureusement le plus gros du travail est fait, nous allons modifier nos composants intelligemment pour pouvoir les réutiliser.

Réutiliser les composants existants

Tout d'abord, dans le composant Toolbar, complétez la fonction **toAtterrissages** en vous inspirant de ce qui a été fait pour la fonction **toDecollages**. Appelez cette fonction dans le fichier html au clic souris.

Ensuite, modifiez le fichier **app-routing.module.ts** pour ajouter une route qui mène à l'écran des atterrissages. Le composant cible sera le même que pour les décollages : **ViewAirFrance**.

Nous allons passer un paramètre dans la route qui mène au composant ViewAirFrance. Ce paramètre permettra d'indiquer au composant s'il doit afficher les décollages ou les atterrissages.

```
const routes: Routes = [
  {
    path: 'decollages', component: ViewAirFranceComponent,
    data: { type: 'decollages' },
  },
  {
    path: 'atterrissages', component: ViewAirFranceComponent,
    data: { type: 'atterrissages' }
  },
  {
    path: '**', redirectTo: 'decollages'
  }
];
```

Nous allons maintenant récupérer ce paramètre, grâce au service **ActivatedRoute**.

Dans le composant ViewAirFrance, définissez une variable de ce type

```
constructor(
  private _volService: VolService,
  private _passagerService: PassagerService,
  private _activatedRoute: ActivatedRoute) { }
```

Implémentez l'interface OnInit, pour qu'à l'initialisation le composant récupère le paramètre et sache s'il doit être de type **décollages** ou **atterrissages**.

```
ngOnInit(): void {
  this._activatedRoute.data.subscribe((data$) => {
    this.type = data$['type'] ? data$['type'] : 'decollages';
  })
}
```

Info : N'oubliez pas de sauvegarder cet observable auquel vous venez de souscrire pour le détruire une fois qu'il ne vous servira plus.

Dans le fichier html du composant ViewAirFrance, conditionner l’affichage du titre en fonction du type pour afficher DECOLLAGES ou ATTERISSAGES.

Interrogez l’API open data

Cet attribut **type** qui se trouve dans le composant **ViewAirFrance**, va nous permettre de conditionner la récupération des vols. Souvenez-vous, l’API de récupération des vols, ne récupère que les vols qui décollent d’un aéroport.

Créez une fonction getVolsArrivee dans le service et copiez/collez le contenu de la fonction getVolsDepart, vous ne changerez que l’url en remplaçant **departure** par **arrival**.

Voici la documentation de l’API si besoin : <https://openskynetwork.github.io/opensky-api/rest.html#arrivals-by-airport>

Modifiez la fonction qui réagit aux changements des filtres dans le composant ViewAirFrance, pour qu’en fonction du type de recherche, elle appelle le bon service.

Si vous avez correctement implémenté le service, vous devriez avoir des vols qui s’affichent lorsque vous êtes dans l’onglet des atterrissages.

Aéroport *
Marseille - Marseille Provence Airport

Entrez une période de 3 jours
04/05/2022 – 06/05/2022

Appliquer

✈	3950cc	AFR53SM	LFPO → LFML	AIRFRANCE
✈	3950cd	AFR64DL	LFPO → LFML	AIRFRANCE
✈	3950d0	AFR25RC	LFPO → LFML	AIRFRANCE
✈	39d305	TVF4281	OLBA → LFML	transavia

On constate d’ailleurs que l’aéroport de destination est bien celui sélectionné : ici Marseille (LFML).

A noter : Pour pousser plus loin la personnalisation nous pourrions modifier l’icône d’avion qui décolle, en avion qui atterrit, en utilisant l’icône flight_land au lieu de flight_takeoff.

Modifiez les passagers (5 points)

Les différentes classes de vol

Les passagers appartiennent tous à une classe. Nous allons mettre en évidence ces classes en les affichant d'une couleur différente.

Nous n'allons pas conditionner le css mais créer une directive qui modifiera la couleur du texte en fonction de la valeur de l'attribut classeVol.

Pour créer une directive, ouvrez un terminal sur VSCode à l'endroit où vous voulez la créer. Et exécutez la commande suivante : `ng g d classe-vol`

La directive a bien été créée :

```
@Directive({
  selector: '[appClasseVol]'
})
export class ClasseVolDirective {
  constructor() { }
}
```

Nous allons ajouter un Input qui permettra de passer la valeur de la classe en paramètre.

Attention : Pour que ça soit plus simple ensuite, le nom de la variable Input doit avoir le même nom que le sélecteur de la directive : `appClasseVol`.

Nous allons ajouter dans le constructeur un service qui va nous permettre de modifier les propriétés de la balise html sur laquelle nous allons placer notre directive.

```
constructor(private el: ElementRef) { }
```

Nous n'avons plus qu'à créer une fonction qui en fonction de la valeur de l'Input mette à jour l'attribut **color** de l'élément html. La syntaxe est la suivante :

```
this.el.nativeElement.style.color = 'blue';
```

Il y a donc 3 couleurs à utiliser, il n'y a aucune contrainte sur le choix des couleurs.

Il faut ensuite appeler notre directive et lui donner un paramètre

```
<div class="attribute" [appClasseVol]="passager.classeVol">
|   {{ passager.classeVol }}
</div>
```

Voici le rendu :



Rasmus Møller

BUSINESS

 x0



Dominic Zahl

BUSINESS

 x0



Elsa Brun

STANDARD

 x0



Craig Hill

PREMIUM

 x0



Malte Kröger

PREMIUM

 x2

Le poids des bagages

Le poids des bagages en soute est un sujet délicat pour les compagnies aériennes. Si l'avion est trop lourd il consommera beaucoup plus et cela va donc entraîner des coûts pour les compagnies. De plus, si tout le monde emporte 10 bagages en soute, l'avion n'aura jamais assez de place pour tout emporter, sans parler de la logistique qui serait terrible.

Air France a donc mis en place une règle de gestion en fonction des classes :

Classe	Nombre maximum en soute
Standard	1
Business	2
Premium	3

Nous allons mettre en place une directive qui mette le fond de la zone d'affichage du nombre de bagages en soute en rouge.

 x2

Cette directive prendra en paramètre la classe, ainsi que le nombre de bagages, et affichera en rouge le fond de la zone si la règle de gestion décrite ci-dessus n'est pas respectée.

L'affichage des photos

A cause des problématiques RGPD, l'entreprise préfère qu'une option soit mise en place pour ne pas afficher les photos des passagers.

L'affichage des photos doit rester possible pour contrôler à posteriori la bonne identité des passagers. Mais pour une autre utilisation, les photos n'ont pas d'intérêt.

Vous allez mettre en place un slide-toggle d'Angular Material, pour conditionner l'affichage des photos dans le composant ListePassagers. **Par défaut le slide-toggle doit être désactivé et les photos ne doivent pas être affichées.**

Lien de la doc : <https://material.angular.io/components/slide-toggle/overview>

L'affichage de l'adresse mail

L'API de récupération des passagers ne récupère pas l'attribut d'adresse mail. Pourtant l'API met à disposition cet attribut. Dans un futur proche, il sera sans doute possible d'envoyer des mails directement depuis l'application aux passagers.

Nous allons donc ajouter cet attribut.

Tout d'abord ajoutons-le dans la méthode getPassagers du service.

Suivez la documentation de la section **Including/Excluding fields** : <https://randomuser.me/documentation>

Pensez à ajouter ensuite l'attribut dans le model, et notamment dans le constructeur de la classe Passager.

Enfin, nous allons afficher cet attribut sous forme d'infobulle lorsque l'utilisateur survole le nom du passager.

Pour ajouter une infobulle, vous allez devoir ajouter le module MatTooltipModule dans votre app.module, comme spécifié dans la doc : <https://material.angular.io/components/tooltip/api>. Si nécessaire relancez votre application pour que le module soit pris en compte.

Vous trouverez dans la documentation des exemples simples pour afficher une infobulle.