Projet JEE : Faire une application web avec *Spring MVC*

GSI2 – Groupe 2

- KAMGAING Rodrigue Ulrich
- DELPRAT Florian

Description du projet

AIR Nomads est une application à destination des voyageurs et des compagnies aériennes. Pour les uns, elle permet de trouver le trajet idéal en avion. Pour les autres, elle permet de gérer sa compagnie aérienne.

Ce qui était présent à la soutenance

1. Backend REST complète (toutes les entités pouvaient être créées, mises à jour, supprimées, récupérées à travers une API REST)
2. Sécurité (connexion, déconnexion, gestion des rôles)
3. Liste des employés
4. Toutes les entités avec les bonnes relations (OneToMany, ManyToMany...)

Ce qui est nouveau

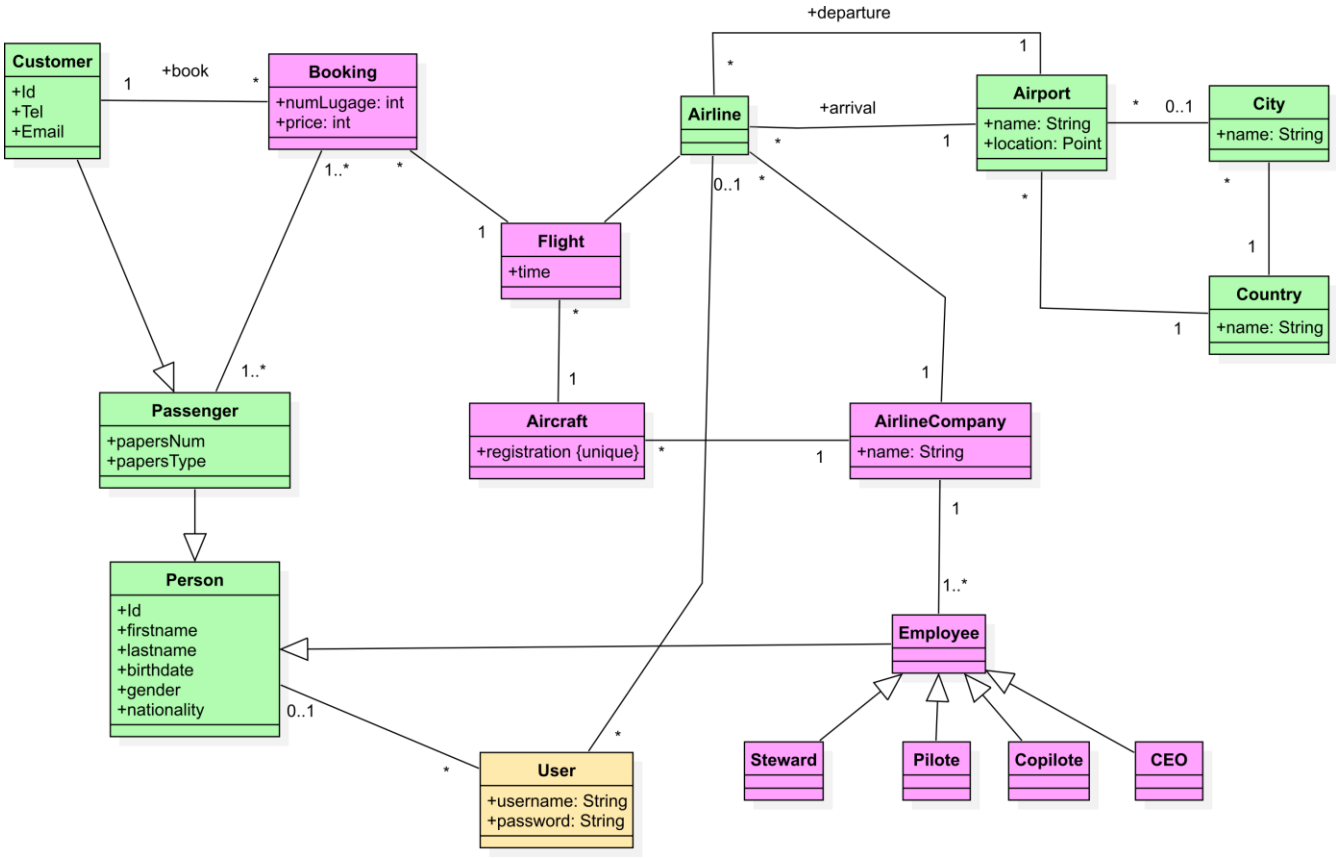
1. MVC pour trouver le trajet idéal, créer une réservation et créer un compte client
2. MVC pour consulter / supprimer / mettre à jour ses réservations
3. MVC pour réaliser les opérations d'une compagnie aérienne (Création de vol, achat d'avion...)
4. Page admin qui permet d'afficher les listes d'entités et de parcourir leurs relations en AJAX
5. La connexion renvoie sur différentes pages en fonction du rôle

Voyons ça plus en détail...

Liste des entités

- Aircraft
- Airport
- Airline : ligne aérienne / trajet entre deux aéroports
- AirlineCompany
- Booking : réservation
- City
- Country
- Flight
- Person
  - → Passenger
    - → Customer
  - → Employee
- User : compte utilisateur utilisé par la sécurité

Diagramme de classes



Les couleurs vert / violet indiquent comment le travail a été réparti entre les membres du groupe.

Outils principaux

- webpack
  - optimisation et configuration de la compilation du JS et du CSS
- stimulus
  - implémentation des fonctionnalités frontend
- sass
  - génération procédurale de CSS (notamment pour configurer des variables *bootstrap*)
- bootstrap, tailwind
  - librairies CSS pour l'affichage
- typescript
  - javascript avec des types pour implémenter des design pattern (notamment Command)
- google places api
  - recherche de lieux afin d'obtenir latitude et longitude
- thymeleaf
  - rendu des vues du MVC
- font-awesome
  - belles icones
- jquery
  - manipulation du DOM

Jeu de données : test.LoadDataSampleStartupRunner

Un jeu de données est chargé lors du premier lancement de l'application.

- tous les aéroports de taille moyenne et grande du monde
- tous les pays du monde
- toutes les villes dans lesquelles il y a un aéroport de taille moyenne ou grande
- 3 utilisateurs
    - user 1234 | ROLE_USER
    - airline 1234 | ROLE_AIRLINE
    - admin 1234 |ROLE_ADMIN
- 100 employés générés aléatoirement
- des routes aériennes et des vols entre plusieurs grosses villes de partout dans le monde
- et toutes les autres entités possibles avec tous leurs liens et données renseignées

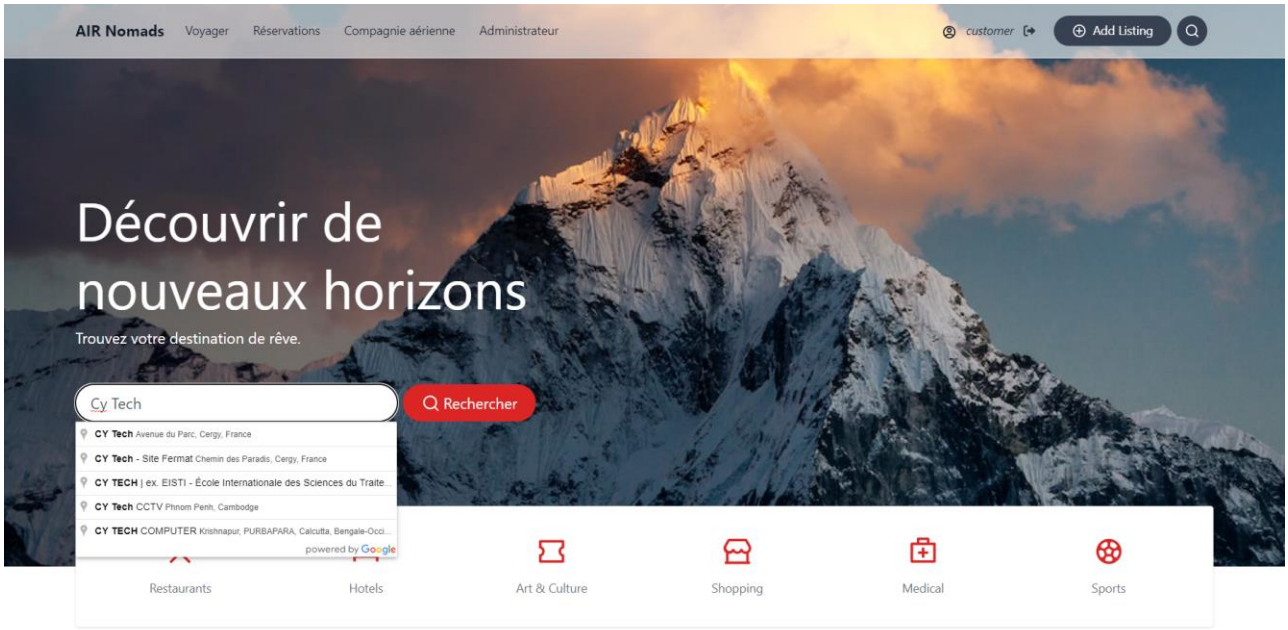
Les utilisateurs airline et admin pourront servir à voir les côtés compagnie aérienne et administration.

Routes principales (1/3)

Côté client : customer.CustomerController

@GetMapping(“/”)
welcomePage()

- page d’accueil
- permet de chercher un lieu (renvoie sur */booking* pour trouver un vol et réserver)
- basé sur un template trouvé sur internet (seule page dans ce cas, les autres ont été crées à partir de rien en utilisant un mix de *bootstrap* et *tailwind*)



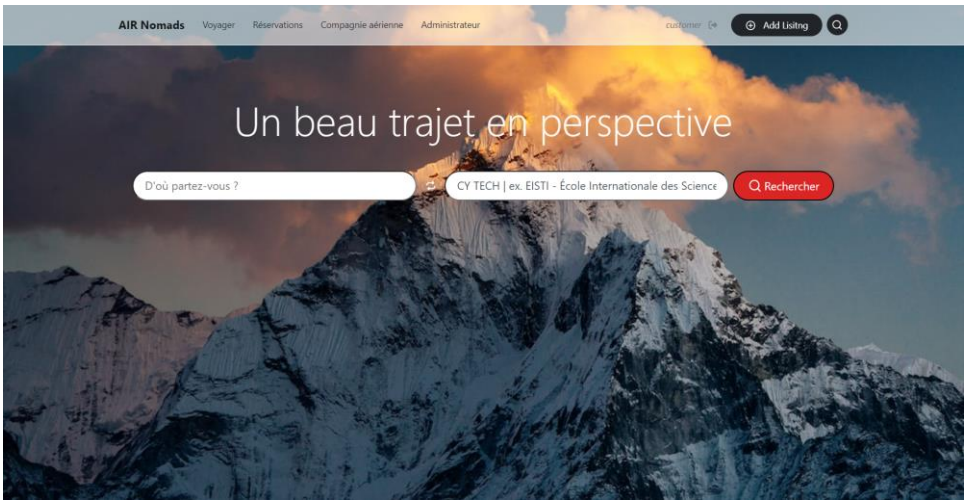

```
@GetMapping("/booking")
public String bookingForm(
        Model model,
        @RequestParam(value = "lat", required = false) Double lat,
        @RequestParam(value = "lon", required = false) Double lon,
        @RequestParam(value = "q", required = false) String q,
        @RequestParam(value = "tlat", required = false) Double tlat,
        @RequestParam(value = "tlon", required = false) Double tlon,
        @RequestParam(value = "tq", required = false) String tq,
        Authentication authentication, HttpServletRequest request

)
```

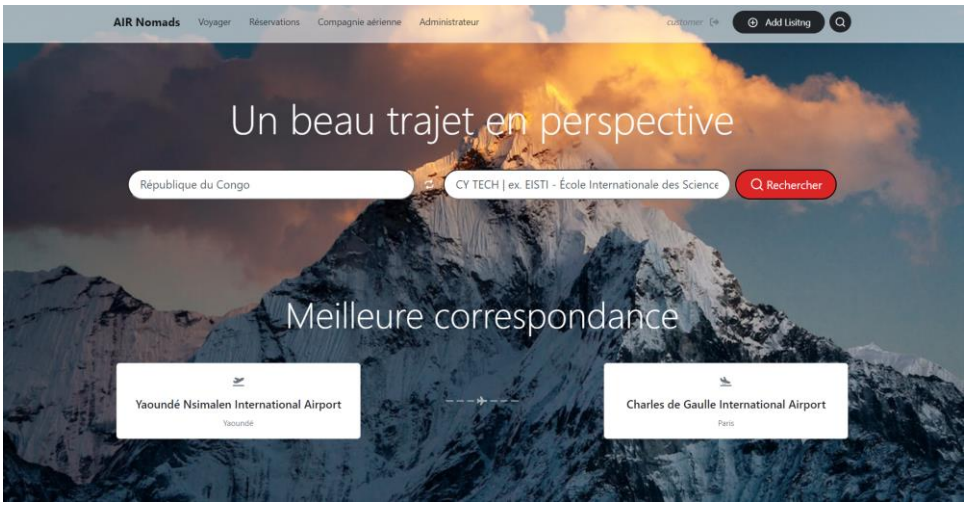
- page multitâche autour de la réservation d’un vol
    - rechercher un vol
    - créer un compte client
    - créer une réservation

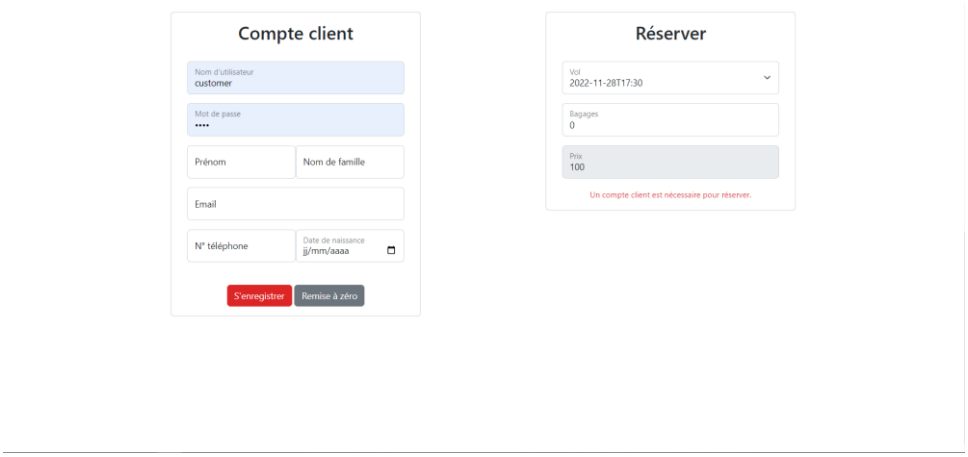
- créée à partir de rien en utilisant *bootstrap* / *tailwind*
- **affichage variable** dépendant des paramètres et de l'utilisateur

    - **Si paramètres incomplets**

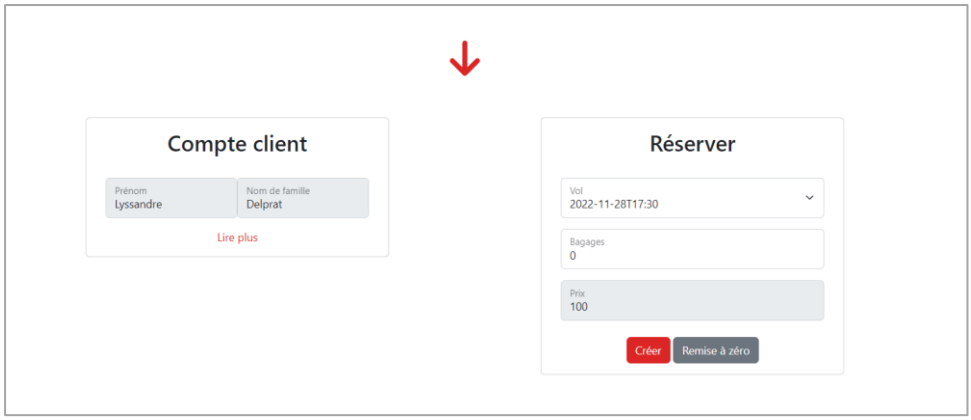


    - **Si paramètres complets mais l'utilisateur n'est pas un client**
        - il peut / doit créer un compte client pour réserver





- **Si paramètres complets et utilisateur connecté avec un compte client**
  → il peut faire une réservation parmis les vols proposés



- **beaucoup d'interactivité**
  - les éléments apparaissent avec une animation
  - les champs de recherches peuvent être intervertis avec un bouton
  - si une ligne aérienne est affichée, la page scroll automatiquement dessus
  - la flèche rouge scroll la page vers le bas
  - si le client est connecté, il peut voir ses informations simplifiées ou non avec un bouton "Lire plus" / "Lire moins"

```
@PostMapping("/booking/create-customer")
public String createCustomer(
        @ModelAttribute("redirectUrl") String redirectUrl,
        @ModelAttribute Customer customer,
        @ModelAttribute User user,
        HttpServletRequest req)
```

- crée un utilisateur et un compte client associé à cet utilisateur, puis redirige sur *redirectUrl*

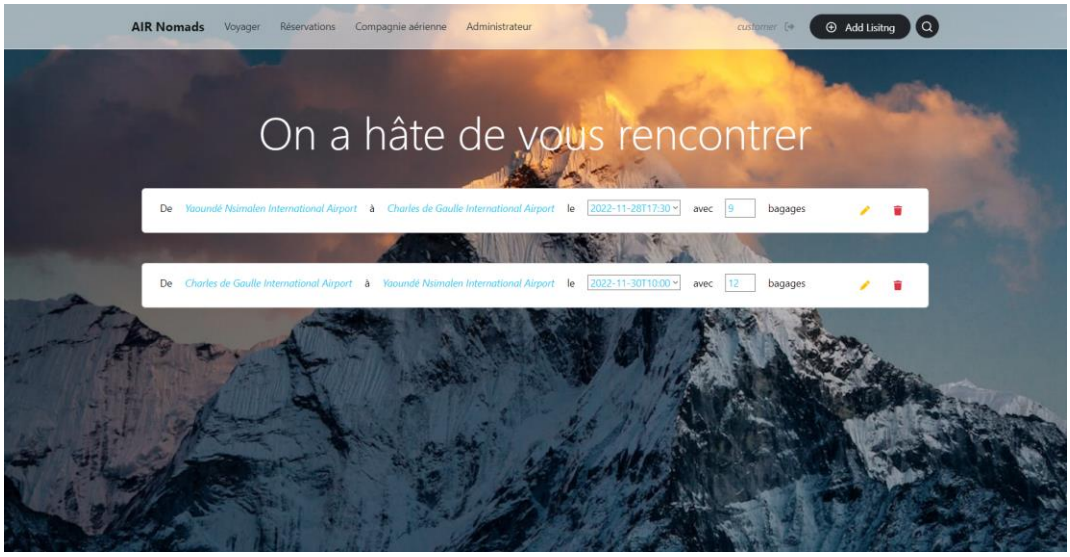
- *redirectUrl* est notamment utile pour garder les paramètres de /booking comme les lieux recherchés

```
@PostMapping("/booking")
public String bookingSubmit(
       @ModelAttribute Booking booking,
       @ModelAttribute("redirectUrl") String redirectUrl,
       Authentication auth, RedirectAttributes redirectAttributes)
```

- crée une réservation pour le client connecté, puis redirige sur *redirectUrl*

```
@GetMapping("/bookings")
public String displayBookings(Authentication auth, Model model)
```

- affiche la liste des réservations du client connecté




```
@DeleteMapping("/bookings/{id}")
public String deleteBooking(
        @PathVariable("id") Long id,
        RedirectAttributes redirectAttributes
)
```

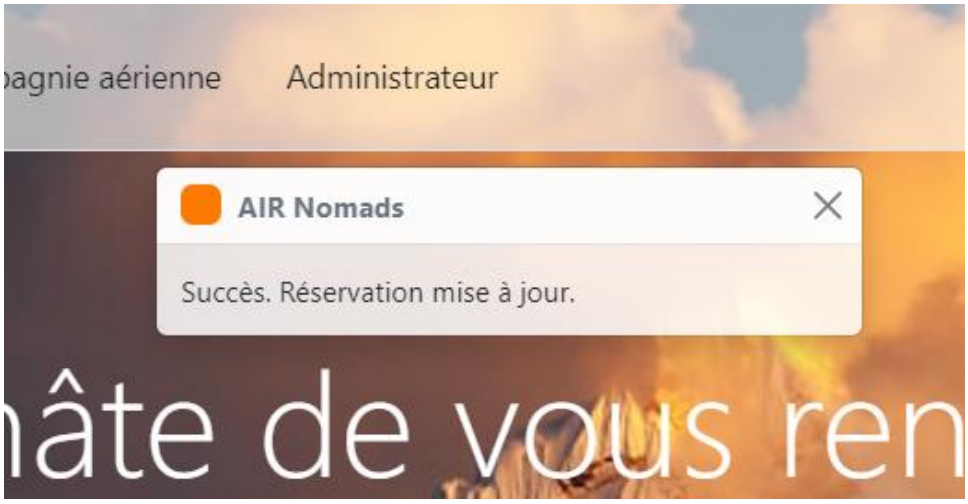
- Supprime une réservation et redirige sur /bookings

```
@PutMapping("/bookings/{id}")
public String putBooking(
        @PathVariable("id") Long id,
        @ModelAttribute("numLuggages") int numLuggages,
```

```
        @ModelAttribute("flightId") Long flightId,
        RedirectAttributes redirectAttributes
)
```

- Met à jour une réservation et redirige sur /bookings

Enfin, toutes les actions entraînent l'affichage d'un message de succès ou d'erreur.

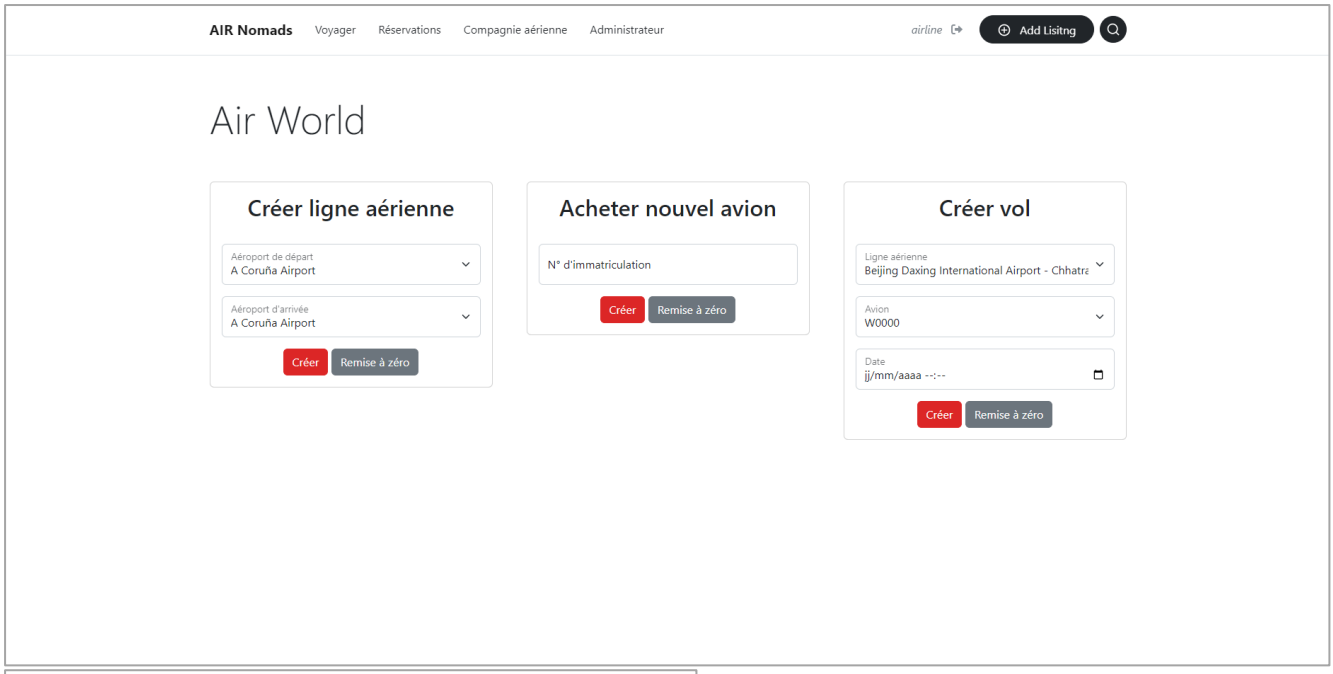


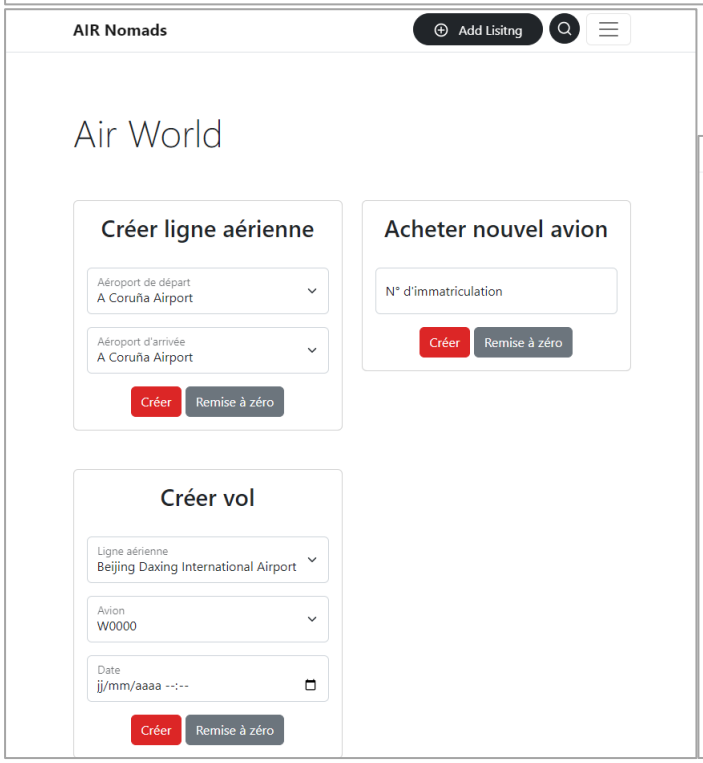
Routes principales (2/3)

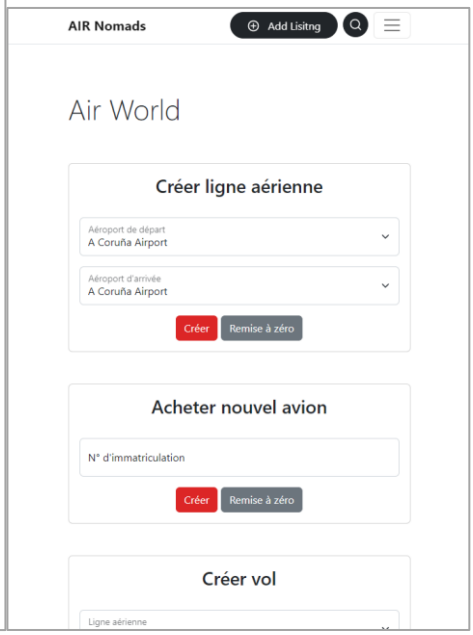
Côté compagnie aérienne : airline.AirlineController

`@GetMapping("/airline")`

- affiche une page contenant plusieurs formulaires
  - création de ligne aérienne
  - achat d'avion
  - création de vol
- responsive


AIR Nomads
Voyager
Réservations
Compagnie aérienne
Administrateur
airline
Add Lisitng
Air World
Créer ligne aérienne
Aéroport de départ
A Coruña Airport
Aéroport d'arrivée
A Coruña Airport
Créer
Remise à zéro
Acheter nouvel avion
N° d'immatriculation
Créer
Remise à zéro
Créer vol
Ligne aérienne
Beijing Daxing International Airport - Chhatra
Avion
W0000
Date
jj/mm/aaaa --:--
Créer
Remise à zéro


AIR Nomads
Add Lisitng
Air World
Créer ligne aérienne
Aéroport de départ
A Coruña Airport
Aéroport d'arrivée
A Coruña Airport
Créer
Remise à zéro
Acheter nouvel avion
N° d'immatriculation
Créer
Remise à zéro
Créer vol
Ligne aérienne
Beijing Daxing International Airport
Avion
W0000
Date
jj/mm/aaaa --:--
Créer
Remise à zéro


AIR Nomads
Add Lisitng
Air World
Créer ligne aérienne
Aéroport de départ
A Coruña Airport
Aéroport d'arrivée
A Coruña Airport
Créer
Remise à zéro
Acheter nouvel avion
N° d'immatriculation
Créer
Remise à zéro
Créer vol
Ligne aérienne

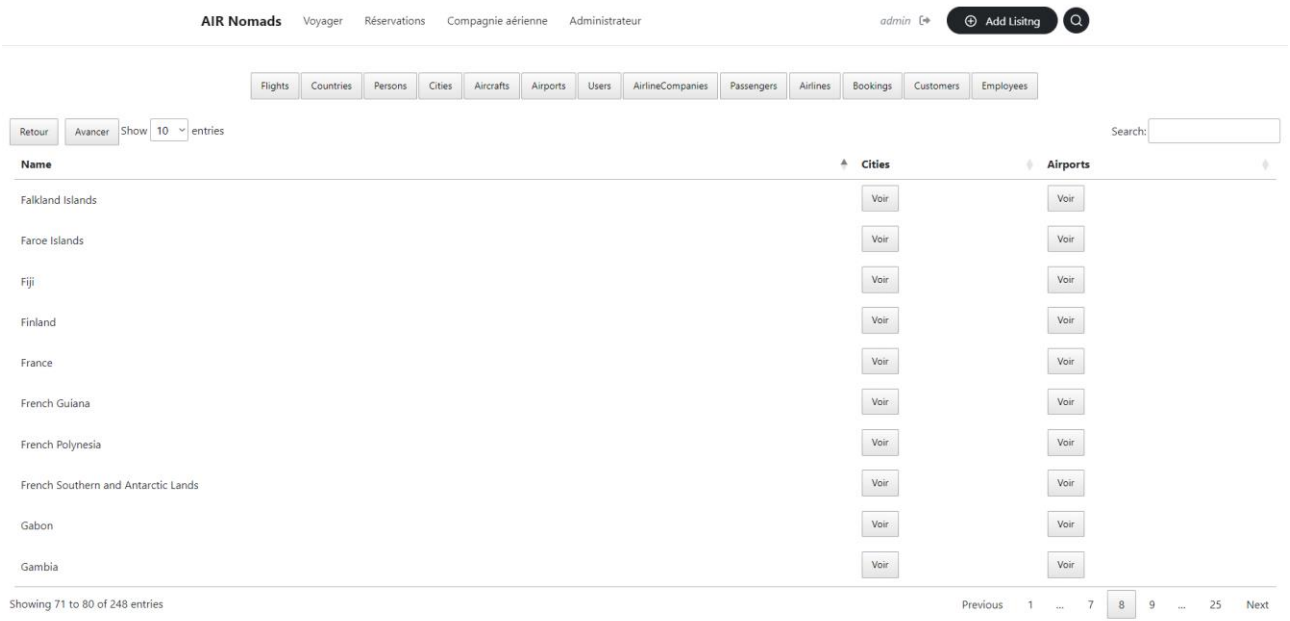
Routes principales – Suite (3/3)

Côté administrateur : admin.AdminController

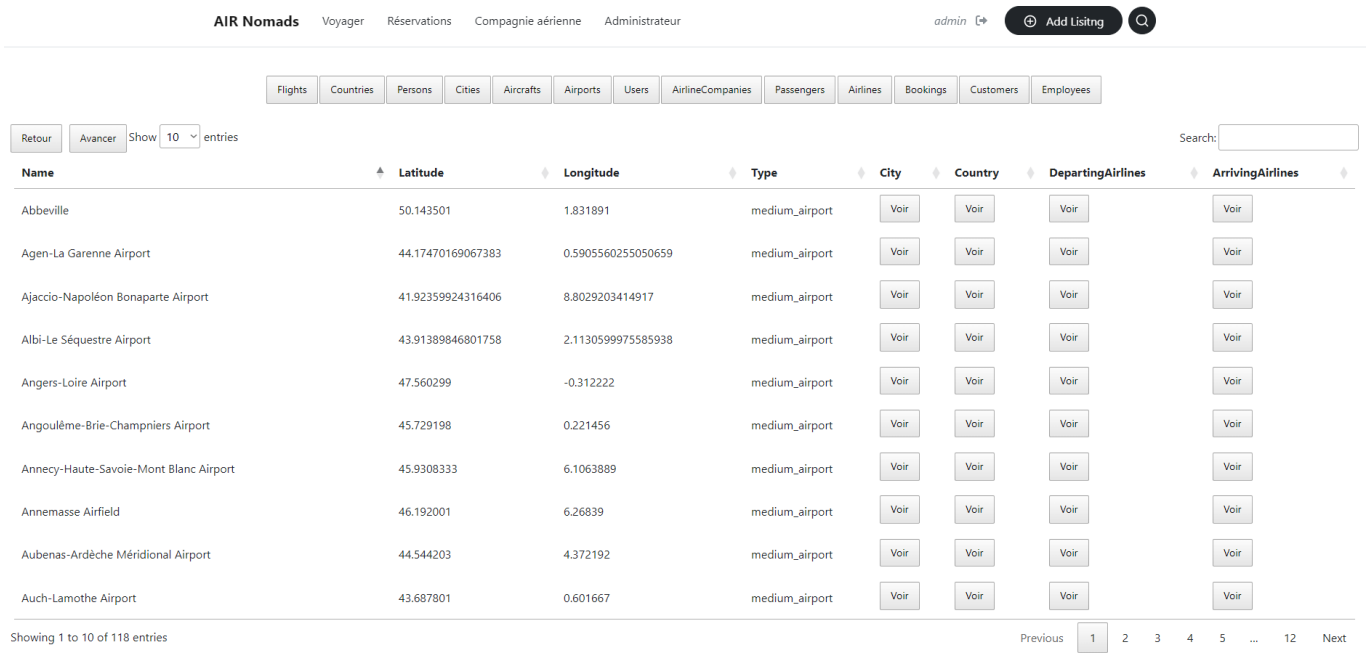
```
@GetMapping(name = "admin-dashboard", value = "/admin")
public String adminDashboard()
```

- affichage de la liste des entités
    - les liens entre entités sont navigables à l'infini
    - navigation dans l'historique des recherches (possibilité de retour en arrière / d'avancer grâce au design pattern **Command**)
    - route sans paramètres car toute la récupération des données et l'affichage sont fait de façon REST et en AJAX
        - l'**affichage** est **entièrement procédural** basé sur une série de requête AJAX à l'api REST de notre application
            1. récupération liste des entités et affichage d'un bouton par entité avec comme valeur le lien vers le profil de l'entité (profil : structure d'une entité)
            2. clic sur un bouton
                i. structure de la table de données récupérées à travers le profil de l'entité
                ii. contenu de la base de données récupérées à travers le lien renvoyant la liste des entités de ce type

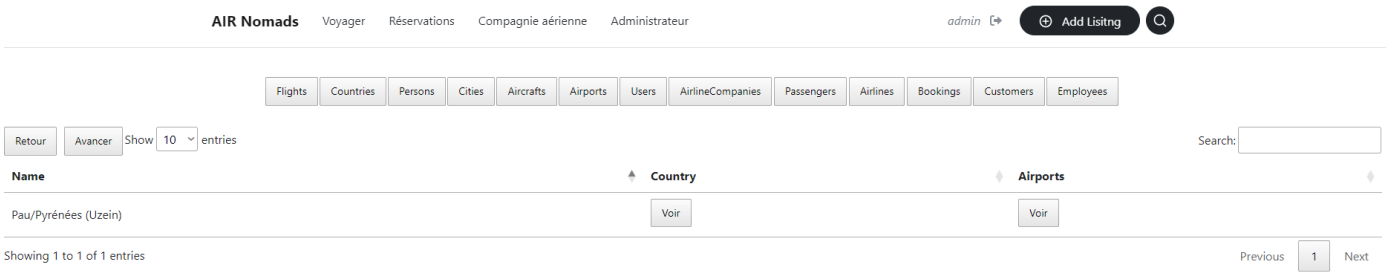
Pays du monde



Aéroports de France



Ville de l'aéroport Pau Pyrénnées



DAO

Toutes les requêtes à la base de données se font à travers des objets dédiés, en accord avec le design pattern DAO.

Exemple :

```
public interface AirlineRepository extends CrudRepository<Airline, Long> {
    @Query("select a from Airline a order by
                    distance(a.departure.location, :pos) +
                distance(a.arrival.location, :dest)")

    Iterable<Airline> findNearestAirline(@Param("pos") Point position, @Param("dest") Point
destination);
}
```

Cette interface générera un objet doté de toutes les méthodes nécessaires pour faire du CRUD, ainsi qu'une méthode importante qui servira à trouver la meilleure ligne aérienne pour les clients, en fonction d'une position et d'une destination.