

Documentation Technique : Intégration Backend/Frontend

Projet : Moteur de Recherche de Vols
Version API : 1.0

1. Informations Générales

- **Base URL (Local)** : `http://localhost:XXXX/api` (Vérifiez le port sur la console au lancement).
 - **Format de réponse** : JSON uniquement.
 - **Authentification** : Gérée intégralement par le backend. Le Frontend n'a pas à gérer de tokens.
-

2. Workflow & Endpoints

Voici l'ordre logique d'intégration des APIs dans l'interface utilisateur.

Étape A : Initialisation (Remplissage des listes)

Au chargement de la page, utilisez ces endpoints pour peupler les menus déroulants dynamiquement.

1. Récupérer les Classes de Voyage

- **GET** `/flights/classes`
- **Usage** : Remplir le dropdown "Classe" (Economy, Business...).
- **Note** : Utilisez le code comme valeur (value) et le name pour l'affichage.

2. Récupérer les Options de Tri

- **GET** `/flights/sort-options`
 - **Usage** : Remplir le dropdown "Trier par".
 - **Note** : Utilisez le code comme valeur (ex: price, duration).
-

Étape B : Autocomplétion (Barre de recherche)

Lorsque l'utilisateur tape dans les champs "Départ" ou "Arrivée".

- **GET** `/flights/airports?keyword={votre_mot}`
- **Exemple** : `/flights/airports?keyword=Par`

- **Comportement attendu du Frontend :**
 1. Afficher le champ name à l'utilisateur (ex: "Paris, France").
 2. **Stocker le champ iataCode** (ex: "PAR") dans une variable cachée ou un attribut data.
 3. C'est ce code **IATA** qui doit être envoyé lors de la recherche.
-

Étape C : Recherche de Vols

Exécutée au clic sur le bouton "Rechercher".

- **GET /flights/search**
- **Paramètres obligatoires :**
 - origin (Code IATA, ex: "PAR")
 - destination (Code IATA, ex: "JFK")
 - departureDate (YYYY-MM-DD)
 - adults (Entier)
- **Paramètres Optionnels (Filtres & Tri) :**
 - sortBy (Valeur du select Tri)
 - directOnly (booléen)
 - travelClass (Valeur du select Classe)

Exemple d'URL :

http://localhost:5000/api/flights/search?origin=PAR&destination=JFK&departureDate=2024-12-25&adults=1&sortBy=price

3. Fonctionnalité UX : Persistance des données (Modification)

Afin de permettre à l'utilisateur de modifier sa recherche sans perdre ses saisies (User Experience), le backend renvoie systématiquement le contexte de la recherche.

Logique d'implémentation :

1. Chaque réponse JSON contient un objet **originalRequest**.
 2. Le Frontend doit stocker cet objet en mémoire après une recherche.
 3. Si l'utilisateur souhaite modifier sa recherche, utilisez les données de originalRequest pour **ré-injecter les valeurs** dans les champs du formulaire.
-

4. Gestion du Tri et des Filtres (Server-Side)

Architecture : Le tri et le filtrage sont gérés côté serveur pour garantir la performance et la

cohérence des données.

- **Ne pas trier en JavaScript** (Array.sort()).
- **Procédure** : Si l'utilisateur change une option de tri (ex: "Prix croissant"), il faut relancer une requête API /search avec le nouveau paramètre &sortBy=.... Le backend renverra la liste réorganisée.

5. Modèles de Données (Exemples JSON)

Objet Aéroport (Autocomplétion)

JSON

```
{
  "iataCode": "LHR",      // <-- À utiliser pour la requête de recherche
  "name": "London Heathrow", // <-- À afficher dans la liste
  "cityName": "London",
  "countryName": "UNITED KINGDOM"
}
```

Objet Réponse Recherche (Principal)

```
{
  "success": true,
  "totalResults": 5,

  // Objet pour la persistance du formulaire (Re-remplissage)
  "originalRequest": {
    "origin": "LHR",
    "destination": "JFK",
    "departureDate": "2024-12-25T00:00:00",
    "adults": 1,
    "travelClass": "ECONOMY"
  },

  // Liste des résultats à afficher
  "flights": [
    {
      "carrierName": "British Airways",
```

```
"departureAirportCode": "LHR",
"arrivalAirportCode": "JFK",
"departureTime": "2024-12-25T08:25:00",
"arrivalTime": "2024-12-25T11:20:00",
"totalDuration": "PT7H55M", // Format ISO 8601
"totalPrice": 650.00,
"currency": "EUR",
"isDirect": true
}
]
}
```