

# REFLIGHTS

## Contexte




Le projet REFLIGHT a pour but de permettre aux utilisateurs de laisser des avis sur des vols, de consulter les avis, et de fournir une interface d'administration (back-office) pour la modération des avis.

Le backend, développé en Java / Spring Boot, expose une API RESTful consommée par un frontend Angular.

L'objectif principal est de concevoir une architecture simple, maintenable et extensible pour une application de gestion de données de vol et de ses avis.

## Architecture logicielle

Bien que l'application ait vocation à évoluer vers des microservices (Flight, Review, Client), le choix d'une architecture monolithique pour cette première version est volontaire :

-  Simplicité de déploiement (un seul projet Spring Boot)
-  Cohérence transactionnelle entre les entités Flight, Airline, Review et User
-  Gain de temps de développement et de configuration

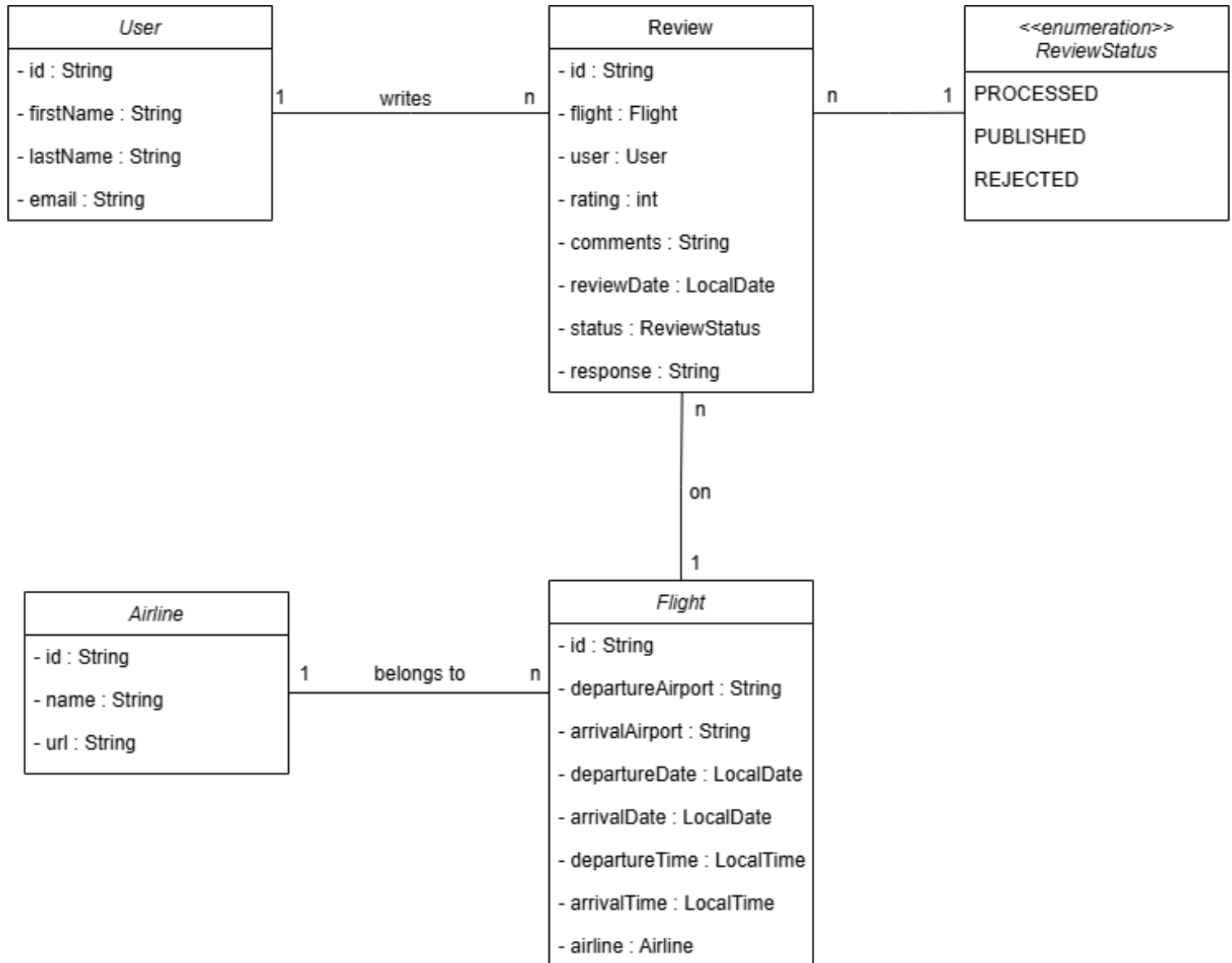
⚙️ Structure en couches clairement séparées :

- config/
- controllers/
- dtos/
- exceptions/
- models/
- services/
  - impl/
- repositories/

Cette séparation assure une isolation des responsabilités et une meilleure maintenabilité.

# Modélisation des données

La modélisation suit une logique relationnelle hiérarchique, alignée sur les concepts du domaine aérien :



Reflight - Diagramme de classes

# Base de données – PostgreSQL (NeonSQL)

Le choix de PostgreSQL hébergé sur NeonSQL (cloud serverless) repose sur plusieurs critères :

- Compatibilité native avec Spring Data JPA
- Utilisation dans anciens projets
- Création automatique du schéma à partir des entités JPA lors du démarrage de l'application

📄 Fonctions SQL :

Pour les requêtes de recherche avec plusieurs critères (Flight et Review), des fonctions SQL natives (COALESCE) sont utilisées afin d'éviter les valeurs nulles.

```
@Query("""
    SELECT f FROM Flight f
    JOIN f.airline a
    WHERE f.departureDate = COALESCE(:date, f.departureDate)
    AND a.name             = COALESCE(:airline, a.name)
    AND f.id               = COALESCE(:number, f.id)
    """)
```

## Structure logique du code

Couche et rôle :

- Controller : expose les endpoints REST (/v1/flights, /v1/reviews, etc.) et reçoit les requêtes du frontend Angular
- Service : contient la logique métier : création, filtrage, recherche, gestion des états
- Repository : communication avec la base PostgreSQL via Spring Data JPA (JpaRepository)
- DTO/Model : sépare les objets d'échange (ReviewDTO, FlightDTO, etc.) des entités persistées
- Model (Entity) : définit le schéma de données et les relations ORM

# Front-end - Technologies utilisées

Le front-end de l'application **REFLIGHTS** a été développé avec les technologies modernes du web, garantissant performance, modularité et maintenabilité :

- Angular : framework SPA robuste basé sur TypeScript, gestion de composants, routing, services et injection de dépendances
- Bootstrap 5 : framework CSS pour assurer un design responsive et harmonieux sans surcharger le CSS natif
- Font Awesome : utilisation d'icônes vectorielles pour enrichir l'interface utilisateur (ex. étoiles de notation, avions, utilisateurs)
- Google Fonts : Rubik (700) pour les titres et éléments importants
- SVG & Data URLs : pour des icônes dans les inputs/select
- REST API : Communication fluide avec le backend Spring Boot via HttpClient

## Front-end - Architecture

L'application Angular repose sur une structure modulaire et organisée, conforme aux bonnes pratiques du framework :

- src/
  - app/
    - components/ **#Composants visuels et logiques d'affichage**
      - add-flight-form/
      - flight-list/
      - flight-search/
      - review-list/
      - review-search/
      - top-bar/
    - data/ **#Modèles de données (interfaces TypeScript)**
      - airline.ts
      - flight.ts
      - review.ts
      - user.ts
    - services/ **#Appel des APIs**
      - airlineService.ts
      - flightService.ts
      - reviewService.ts
      - userService.ts
    - validators/ **#Validation des formulaires personnalisés**
    - environnements/ **#Variables d'environnement (URL API)**
    - utils/ **#Fonctions utilitaires et types globaux**

# Communication avec le backend (REST API Spring Boot)

La communication entre le front-end Angular et le back-end Spring Boot se fait exclusivement via HTTP REST, grâce au module HttpClient d'Angular. Les services Angular assurent la centralisation des requêtes HTTP. Les composants consomment ces services via injection de dépendances, ce qui facilite la maintenance. Les appels API utilisent RxJS pour gérer les données de manière asynchrone et réactive (via les Observables).

Exemple d'une requête dans un service Angular :

```
getAll(): Observable<ENTITY[]> {  
    return this.http.get<ENTITY[]>  
    (`${this.baseUrl}${this.getEndpointUrl()}`)  
    .pipe(  
        catchError(this.handleError<ENTITY[]>('getAll', []))  
    );  
}
```

Exemple d'une consommation côté composant :

```
export class FlightListComponent {  
    flights:Flight[] = [];  
  
    constructor(private flightService: FlightService) {}  
    ngOnInit(): void {  
        this.flightService.getAll().subscribe(flights => {  
            this.flights = flights;  
        })  
    }  
}
```

Ce modèle favorise une architecture réactive : les composants s'abonnent (subscribe) aux flux de données, garantissant une interface dynamique et réactive.

# Utilisation de l'IA

L'intelligence artificielle a été utilisée comme outil d'aide, pas comme moyen de "faire le travail à ma place". L'idée était surtout de gagner du temps sur certaines tâches répétitives et d'améliorer la qualité du code, tout en restant totalement impliquée dans le développement et en gardant le contrôle sur ce que je produisais.

- Front-end (Angular / HTML / CSS) : j'ai principalement utilisé **Claude** pour la partie front-end : génération de structures HTML et mise en forme CSS. L'IA m'a permis d'obtenir des bases claires et cohérentes pour les composants visuels, que j'ai ensuite ajustées, refactorisées et harmonisées selon mes jugements personnels.
- Back-end (Spring Boot) : pour le back-end, j'ai utilisé **ChatGPT** uniquement dans un cadre ciblé : la rédaction de requêtes complexes dans les repositories JPA, notamment pour les fonctionnalités de recherche et de tri dynamiques. Les propositions ont systématiquement été relues, testées et optimisées pour s'intégrer au modèle de données.

En résumé, l'IA a été utilisée comme co-pilote intelligent :

- Claude pour accélérer la partie front-end et la conception visuelle,
- ChatGPT pour optimiser les requêtes avancées côté base de données.

## Axes d'amélioration

Bien que l'application réponde aux principaux objectifs pédagogiques du projet, plusieurs évolutions peuvent être envisagées afin d'enrichir les fonctionnalités, d'améliorer l'expérience utilisateur ainsi que la qualité technique de la solution.

### Ajout d'une table Airport

Actuellement, l'application utilise des chaînes de caractères statiques pour définir les aéroports de départ et d'arrivée lors de la création des vols.

Une amélioration consisterait à introduire une entité Airport dédiée, stockée en base de données avec des informations telles que : code IATA, nom de l'aéroport, ville/pays...

Cela permettrait :

- Une meilleure intégrité des données
- La réutilisation de données standardisées
- Une évolution possible vers des fonctionnalités avancées (ex. autosuggestion, carte interactive, calcul de distance, statistiques par aéroport...)

### Mise en place d'un système d'authentification

L'application permet actuellement de laisser des avis et gérer les retours administrateur sans authenticité de l'utilisateur.

# Axes d'amélioration

Un système d'authentification (ex : Spring Security + JWT ou OAuth2) offrirait :

- Véritable espace utilisateur
- Protection des fonctionnalités administrateur
- Gestion des droits (user/admin)
- Historique personnalisé des avis postés

Cela renforcerait la sécurité, la cohérence du flux utilisateur ainsi que le réalisme de l'application dans un contexte professionnel.

## **Application full-stack lancée automatiquement**

Actuellement, le lancement du projet nécessite d'exécuter séparément :

- Le serveur Spring Boot
- L'application Angular

Une amélioration serait d'intégrer un composant d'orchestration, permettant de lancer automatiquement l'ensemble de l'application (exemple : Docker). Cela rendrait l'application plus ergonomique à lancer, plus simple à déployer et plus proche des bonnes pratiques DevOps modernes.

## **Externalisation des variables sensibles (.env)**

Actuellement, les informations sensibles telles que :

- URL de la base de données
- Nom d'utilisateur
- Mot de passe

sont présentes directement dans le fichier de configuration de l'application (application.yml).

Cette pratique fonctionne en environnement pédagogique, mais présente des risques de sécurité élevés dans un contexte réel, notamment lorsque le code est publié sur un dépôt Git public.

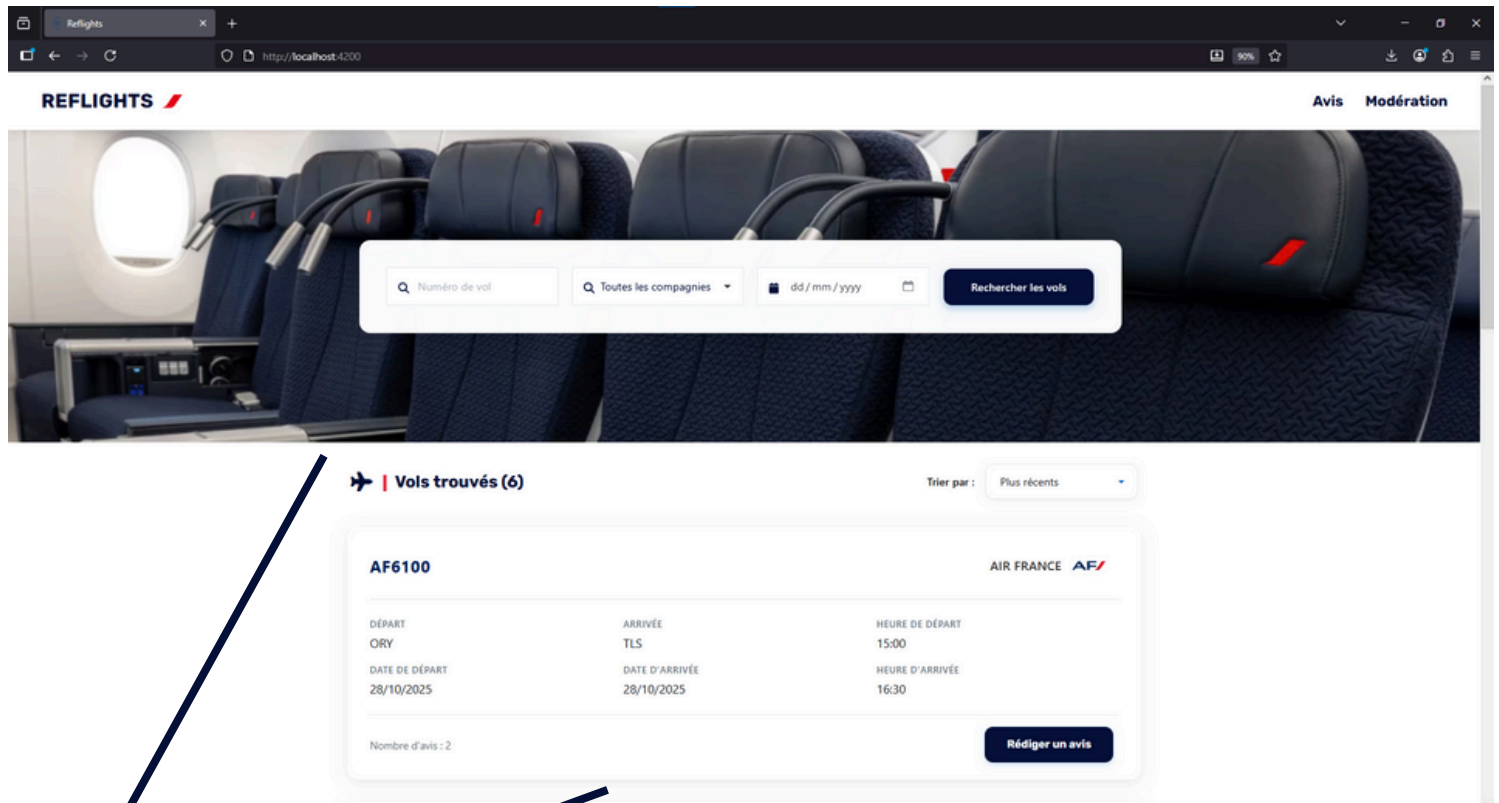
Une amélioration importante consiste à externaliser ces informations dans des fichiers .env ou variables d'environnement, non versionnés dans Git.

Cela permet de protéger les informations sensibles et éviter les fuites de secrets et faciliter le passage entre plusieurs environnements (dev, prod, recette...).

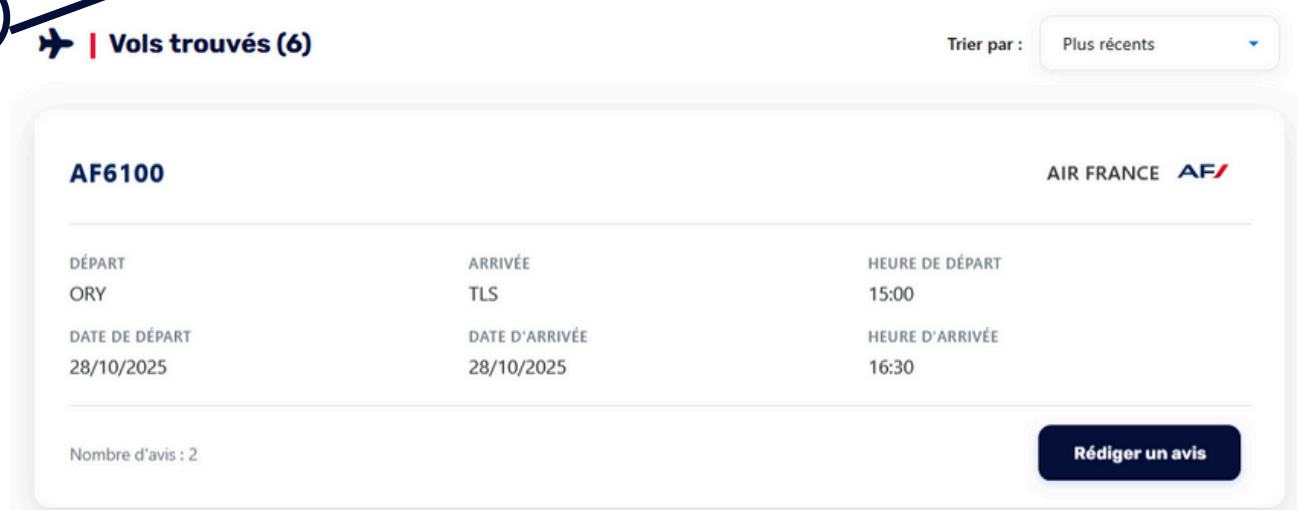


# REFLIGHTS

## ANNEXES





Page d'accueil de Reflights




Zoom sur les données d'un vol



 **Rédiger un avis** ×


 **VOTRE VOL**


**ORY**  
DÉPART



**TLS**  
ARRIVÉE

# Vol AF6100

 AIR FRANCE

 28/10/2025

**Vos informations**

Prénom \*

Jean






Nom \*

Dupont

Email \*

jean.dupont@exemple.com

**Votre note**



Cliquez sur les étoiles pour noter


**Votre commentaire**

Partagez votre expérience \*

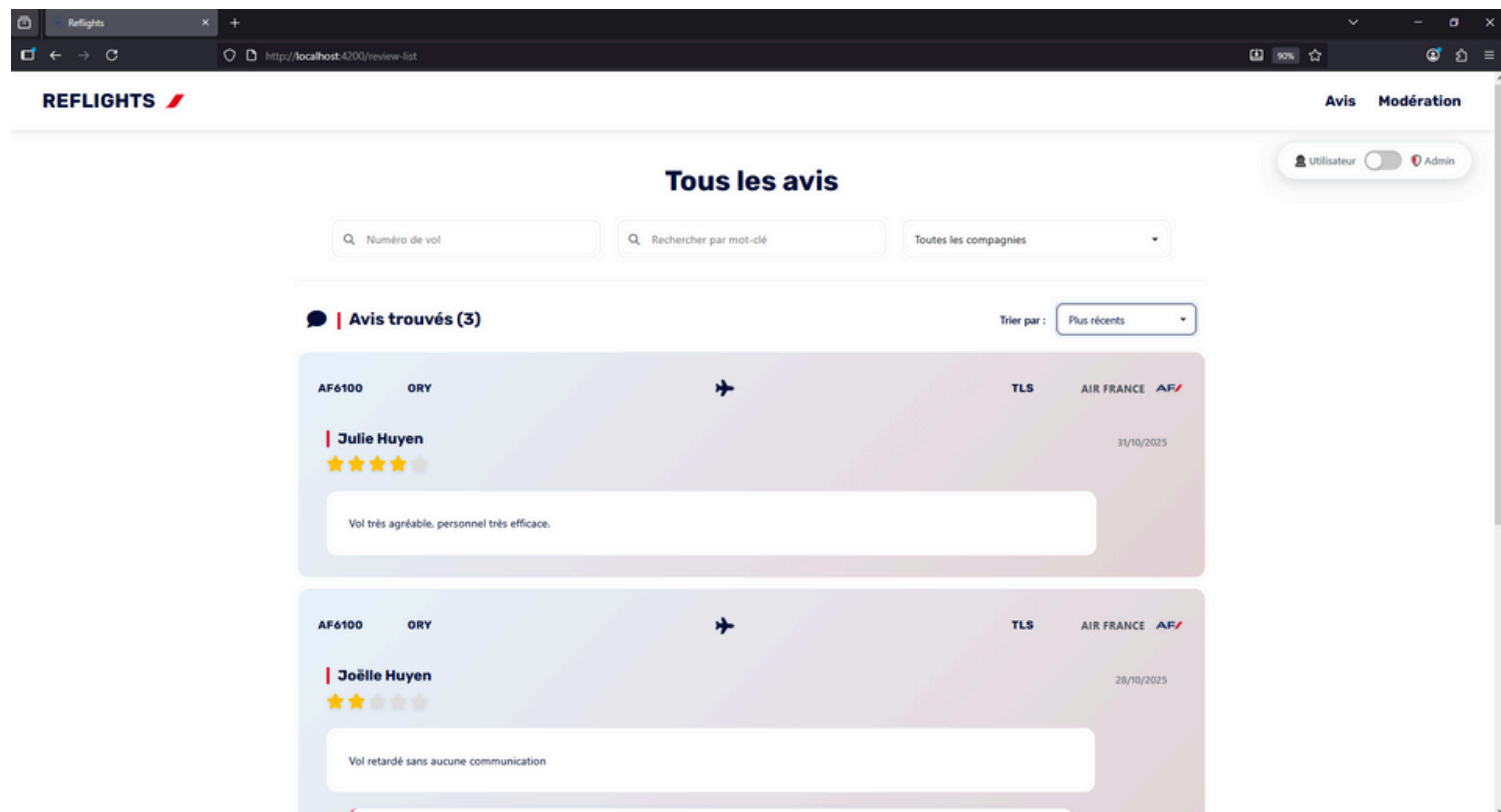
Décrivez votre expérience de vol : confort, service, ponctualité, etc.

0 / 1000 caractères

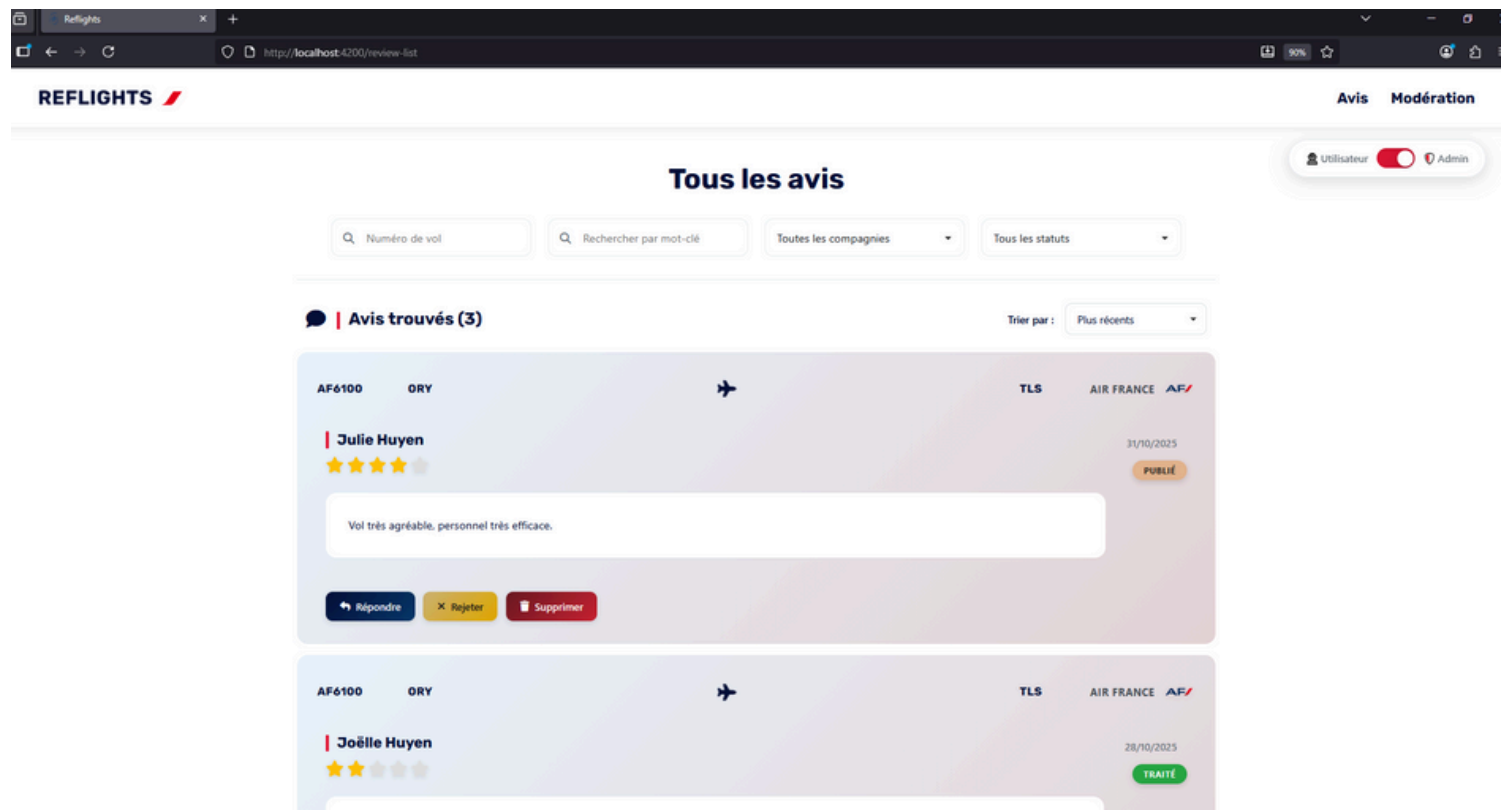
Annuler

 Publier mon avis

Formulaire de rédaction d'un avis (modal)



Page des avis (user mode).



Page des avis (admin mode).

# REFLIGHTS

## MERCI !

Découvrez l'ensemble de **REFLIGHTS** depuis mon repository Github :  
<https://github.com/Joelle13/Reflights>

