**Résumé du Projet : Gestion d'aéroports et vols en Java, concept appris :**

1. **Gestion des threads et parallélisme**

* J'ai appris à utiliser les threads pour éviter de bloquer l'interface utilisateur (éviter le "freeze"). En particulier, j'ai utilisé la classe Runnable (implémentée dans ApiRequestTask) pour exécuter les appels HTTP dans un thread séparé.
* J'ai compris l'importance de la synchronisation et de l'utilisation de Platform.runLater pour mettre à jour l'interface utilisateur (JavaFX) depuis le thread principal.

1. **Requêtes HTTP avec Java**

* En utilisant les classes HttpClient, HttpRequest et HttpResponse [61†source], j'ai pu interroger une API distante (AviationStack) pour récupérer des données sur les vols.
* L'intégration d'une clé API comme paramètre de la requête et la gestion des réponses ont été des étapes cruciales pour comprendre les appels réseaux.

1. **Analyse et manipulation de JSON**

* J'ai utilisé JsonReader pour analyser les réponses JSON de l'API et créer des objets Flight à partir des données [62†source].
* J'ai appris à gérer des formats complexes comme les dates et heures avec LocalDateTime pour convertir les informations des vols.

1. **JavaFX : Création et manipulation de scènes 3D**

* J'ai compris comment utiliser JavaFX pour créer une scène 3D interactive (à travers la classe Earth), incluant des animations avec AnimationTimer.
* L'affichage dynamique d'éléments graphiques (ex : sphères rouges et jaunes pour représenter les aéroports) m'a permis de relier les données aux interactions utilisateur.

**Problèmes rencontrés et solutions**

1. **Freeze de l'interface utilisateur**

* Problème : Lors de l'appel à l'API, l'animation de la Terre (à l'aide d'AnimationTimer) se bloquait.
* Solution : Utilisation de la classe ApiRequestTask pour exécuter l'appel HTTP dans un thread séparé et de Platform.runLater pour synchroniser les mises à jour graphiques.

1. **Accès concurrent aux ressources**

* Problème : Risque de modification concurrente des éléments de l'interface graphique depuis des threads différents.
* Solution : Encapsulation des mises à jour graphiques dans Platform.runLater, garantissant qu'elles s'exécutent sur le thread principal JavaFX.

1. **Gestion des dates et fuseaux horaires**

* Problème : Les horaires de l'API étaient au format ISO 8601 avec des fuseaux horaires, alors que LocalDateTime ne supporte pas directement les fuseaux.
* Solution : Suppression du fuseau horaire des dates avant conversion avec DateTimeFormatter.

**Conclusion**

Ce projet m'a permis de maîtriser des concepts fondamentaux de la programmation en Java, tels que l'utilisation des threads, les requêtes HTTP, la manipulation de JSON, et la création d'interfaces graphiques interactives avec JavaFX. J'ai également appris à gérer les problèmes de performance et de concurrence tout en garantissant une interface utilisateur fluide et réactive.