Rapport Projet

Flight Tracker

2022, ESCHENBRUMER Brice, GRANDJANIN Tom,
KESTELOOT Kevin

16/01/2023

Sommaire

I - Introduction

Lors de ce projet, nous devions réaliser un service web. Nous avons donc décidé de faire une application web permettant de traquer des vols d'avion dans le monde entier, c'est-à-dire qu'elle recense environ 5 000 vols dans le monde pour lesquels nous avons accès à leur pays de départ ainsi qu'à leur position actuelle à l'instant T. De plus, ce site permet d'avoir différentes informations supplémentaires sur les vols recensés.

II - Description détaillée du cas d'application

Notre application est basée sur l'architecture MVC, c'est-à-dire "modèle vue contrôleur". Nous avons donc des modèles qui vont permettre de contenir les données à afficher, des vues qui contiennent la représentation de l'interface graphique, c'est-à-dire nos différentes pages sur notre site, et des contrôleurs qui contiennent la logique concernant les différentes actions effectuées par l'utilisateur.

Nous avons donc créé les modèles suivants :

Un modèle utilisateur qui va nous permettre de créer et de stocker les différentes données de l'utilisateur de notre site. Nous avons choisi de créer un utilisateur avec seulement un pseudo et un mot de passe, car nous estimons qu'il n'était pas nécessaire de demander plus d'informations pour l'utilisation de notre site.

Nous avons ensuite créé un deuxième modèle qui est le modèle principal de notre application, c'est-à-dire le modèle vol, celui-ci comporte toutes les informations que l'API nous renvoie concernant un vol que nous pourrons exploiter par la suite.

En ce qui concerne la création des vues nous avons créé autant de vues que nous avions de page, c'est-à-dire une vue pour la page d'accueil, une pour la page de connexion et une pour la page d'inscription. Ensuite, deux pages concernant les recherches sur les vols que nous avons stocké, une première qui permet de rechercher les différents vols en fonction de leur pays d'origine et de s'ils sont au sol ou non, avec un affichage des différents avions en fonction du résultat de la recherche. La deuxième page elle, permet de rechercher un vol en fonction de son callsign avec un affichage sur une carte de l'endroit où il est précisément. Puis une troisième page qui permet d'afficher tous les avions que nous possédons dans notre base de données sur une carte.

Pour ce qui est des contrôleurs, nous avons un contrôleur qui permet de gérer les actions de connexion et d'inscription. Un deuxième contrôleur qui nous permet de gérer l'affichage des images sur notre site. Puis un troisième contrôleur qui nous permet de gérer toutes les actions concernant les vols, c'est-à-dire les recherches de vols et le chargement de la base de données.

Notre site fonctionne avec le framework Spring boot et une base de données. Cette base de données est composée de 2 tables :

La première table contient les utilisateurs recensés sur le site, car celui-ci requiert de s'être inscrit au préalable. Nous avons veillé à ce que les mots de passe des utilisateurs soient cryptés dans la base de données par souci de confidentialité et de sécurité. En ce qui concerne les noms d'utilisateurs, nous les avons rendus uniques afin qu'une personne s'étant déjà inscrite ne puisse pas créer un nouveau compte et donc surcharger notre base de données de comptes inutiles. Pour ce qui est de l'encryptage des mots de passe, nous avons utilisé la fonction de hachage Bcrypt qui est basée sur un algorithme de chiffrement nommé Blowfish, cette fonction de hachage utilise aussi l'ajout d'un sel ce qui permet de rendre unique chaque hachage et ce qui permet d'éviter certains types d'attaque comme une attaque de type rainbow table par exemple. Pour ce qui est de l'implémentations de cette fonction de hachage, nous avons fait appel à une dépendance de spring qui est la dépendance suivante : " spring security crypto", cette dépendance nous a permis d'avoir accès aux différentes méthodes associées à la fonction de hachage Bcrypt afin de comparer nos mots de passe et ainsi valider la connexion ou non de l'utilisateur.

Nous avons ensuite une seconde table dans laquelle nous stockons les vols. Ces vols sont récupérés à partir de l'API open source Opensky recensant plusieurs informations sur des vols dans le monde entier. Nous avons essayé d'utiliser l'API java sauf qu'au moment de l'importation dans notre projet par l'intermédiaire de maven nous avons eu une erreur stipulant qu'il était impossible de trouver les fichiers concernant cette API dans le dépôt Maven. Nous sommes donc passés par l'API Rest. Pour cela, nous avons créé une page permettant de charger la base de données en envoyant une requête GET sur l'end point de l'API. Le retour de cette requête était un format JSON un peu particulier ce qui nous à empêché d'utiliser les outils de traitement JSON intégrés à spring. Afin de stocker toutes les données dans notre base de données, nous avons donc fait un parsing à la main de la requête afin de récupérer toutes les données contenues dans la requête. Cela implique que le remplissage de la base de données est très long car nous avons environ 5 000 vols à chaque requête. Nous sommes alors obligés de passer par le remplissage de notre base de données, car les différentes options que propose l'API de base ne correspondaient pas à ce que nous voulions faire, nous avons donc dû dupliquer la base de données de l'API dans la nôtre afin de pouvoir l'utiliser. Une fois que nous avons chargé notre base de données, nous avons la possibilité de consulter les vols en fonction de leur pays d'origine et en fonction de leur position au sol ou non comme expliqué précédemment.

Pour cela, nous sommes simplement passés par une requête SQL sur notre base de données afin de récupérer le résultat que nous avons ensuite stocké dans une liste afin de le passer à la vue et de l'afficher. Afin de créer cette requête SQL nous nous sommes basés sur un repository, c'est-à-dire une classe JAVA héritant de JPA repository qui nous permet de faire des interactions avec notre base de données. Cette requête SQL est donc générée par JPA repository en créant juste une classe avec le nom de ce que l'on recherche, par exemple ici nous avons créé une méthode se nommant "findVolByGroundAndOrigin" avec en paramètre le pays d'origine et le booléen nous permettant de savoir si l'utilisateur voulait que l'avion soit au sol ou non.

Cette requête SQL est effectuée par l'intermédiaire d'une requête http POST qui est donc gérée dans le contrôleur comme expliqué précédemment. Le résultat est donc stocké dans une liste que nous ajoutons à la vue par l'intermédiaire d'un Service et d'un controller qui nous permet de faire un lien entre le backend et le frontend avec Spring. Afin de récupérer le contenu de cette liste et de

l'afficher sur la vue, nous avons utilisé thymeleaf qui est un moteur de template permettant de gérer l'affichage sur la vue de différentes données stockées en back end.

En ce qui concerne l'affichage, nous avons décidé d'afficher les avions résultant de la recherche de l'utilisateur sur une carte du monde. Pour cela, nous avons utilisé leaflet qui est une bibliothèque JavaScript de cartographie. Nous avons donc créé une carte avec cette bibliothèque sur laquelle nous avons rajouté la position de chaque avion correspondant au résultat de la recherche effectuée. Ainsi, nous obtenons une page composée d'un tableau avec toutes les informations sur les vols de la recherche et une carte sur laquelle nous pouvons observer à quels endroits sont ces avions.

Les autres recherches disponibles sur notre site fonctionnent de la même manière, la seule différence est la requête SQL effectuée.

Par l'intermédiaire de thymeleaf nous avons pu gérer les affichages des différentes options en fonction de si l'utilisateur était connecté ou pas avec l'usage du stockage de Session. C'est à dire qu'une fois que l'utilisateur s'est connecté sur notre site nous stockons sont nom d'utilisateur dans la session afin qu'il puisse avoir accès aux différentes options que propose notre site, sans cette connexion l'utilisateur n'a pas accès aux différentes recherches.

.

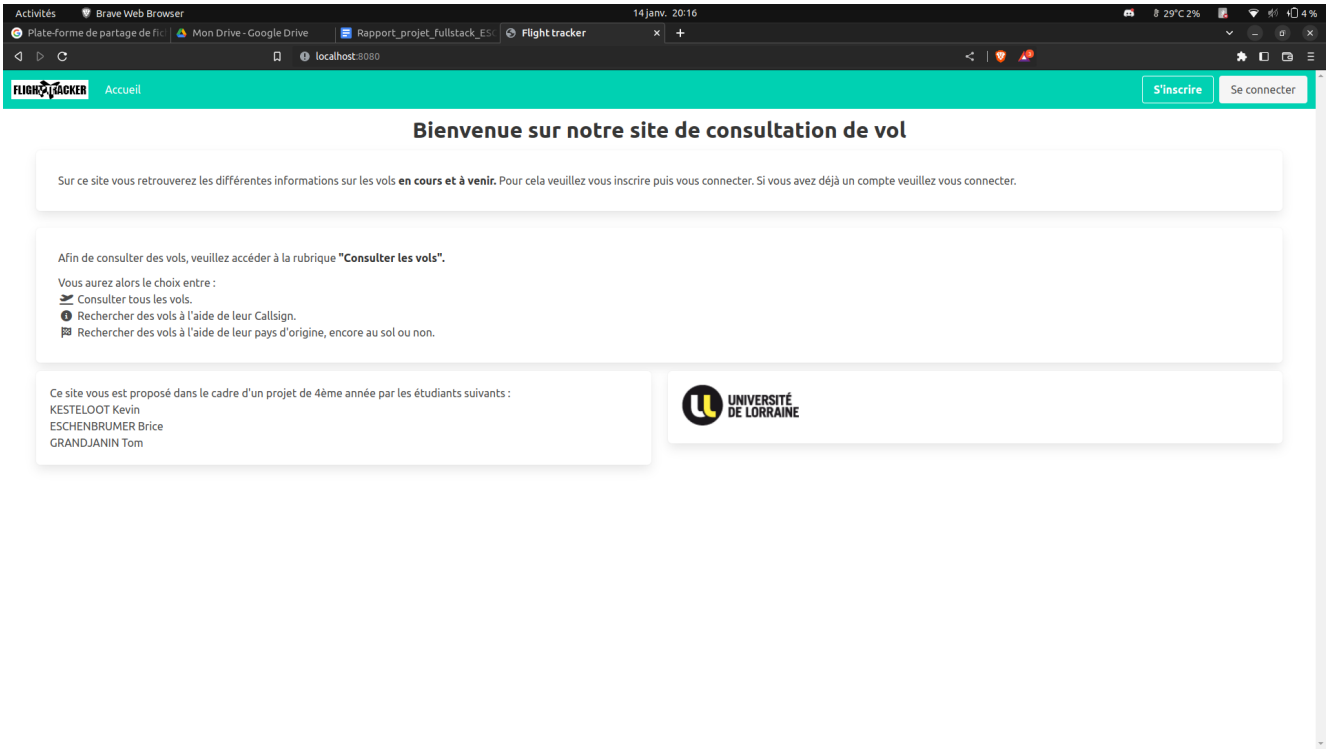


Figure 1: page d'accueil du site

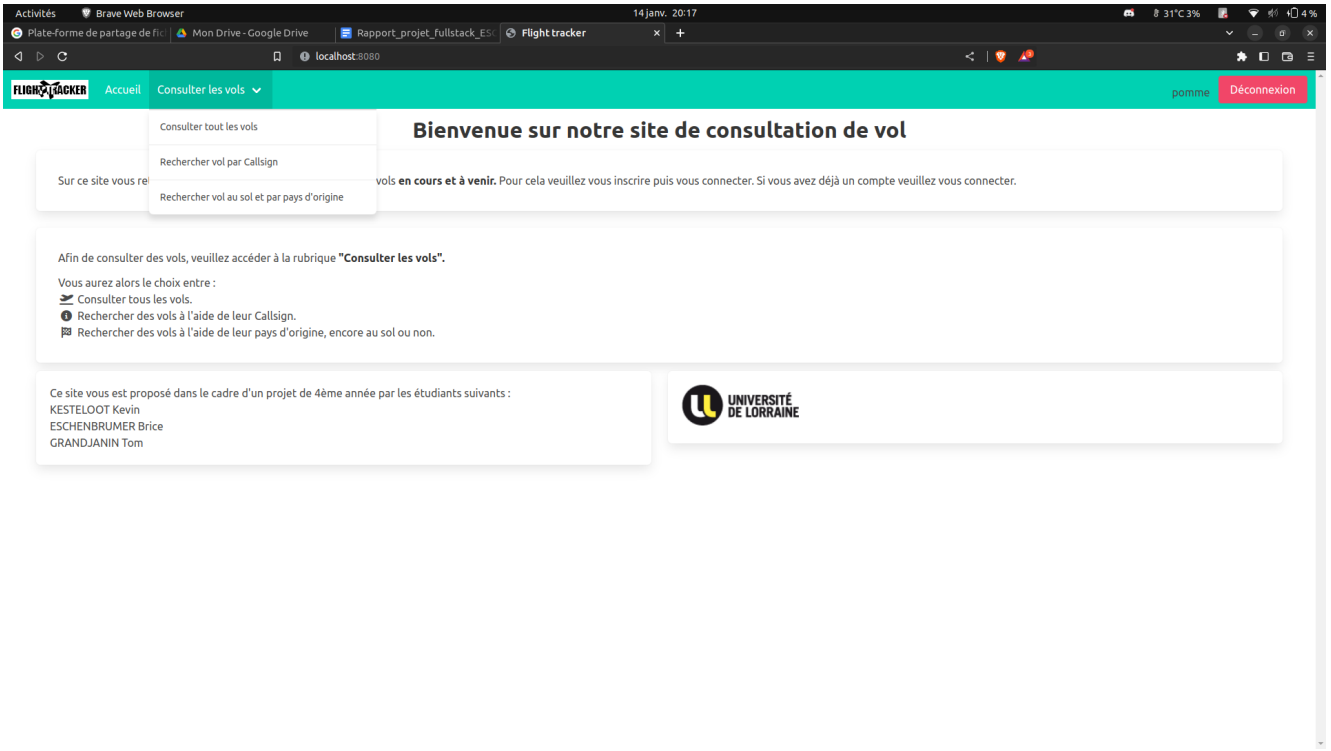
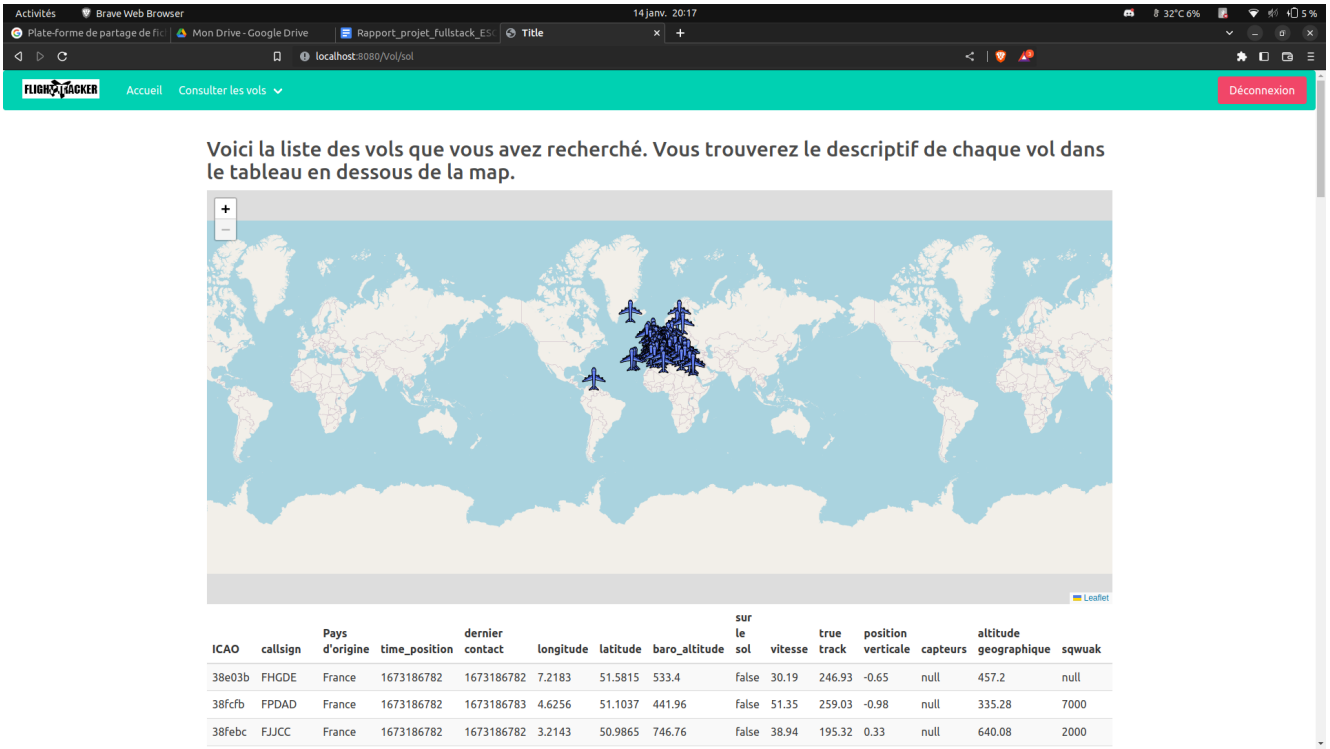


Figure 2 : Différentes options proposées par notre site



ICAO	callsign	Pays d'origine	time_position	dernier contact	longitude	latitude	baro_altitude	sur le sol	vitesse	true track	position verticale	capteurs	altitude geographique	sqwuak
38e03b	FHGDE	France	1673186782	1673186782	7.2183	51.5815	533.4	false	30.19	246.93	-0.65	null	457.2	null
38fcfb	FPDAD	France	1673186782	1673186783	4.6256	51.1037	441.96	false	51.35	259.03	-0.98	null	335.28	7000
38febc	FJJCC	France	1673186782	1673186782	3.2143	50.9865	746.76	false	38.94	195.32	0.33	null	640.08	2000

Figure 3 : résultat d'une recherche d'avion provenant de france et pas au sol

III - Frameworks utilisés

Lors de la conception de notre application en langage java, nous avons principalement utilisé le framework Spring. Celui-ci permet de construire et de définir l'infrastructure de notre application. C'est une infrastructure qui est similaire à un serveur d'application J2EE (Jakarta EE), elle permet la prise en charge de création d'objet de leur mise en relation par l'intermédiaire d'un fichier de configuration. Lors du développement, nous avons utilisé plusieurs dépendances spring comme thymeleaf qui nous a permis de faire notre liaison entre nos objets backend et notre frontend. Nous avons aussi utilisé la dépendance JPA qui nous a permis de faire toutes nos interactions avec notre base de données. Lombok nous a aussi permis de laisser Spring gérer la gestion des données sur nos objets. Enfin, nous avons aussi utilisé une dépendance spring sécurité afin d'encrypter certaines données dans notre base de données.

Pour ce qui est du frontend, nous avons utilisé le framework CSS Bulma qui est un framework open source fonctionnant avec un système de classes qui permet de produire des apparences attrayantes tout en faisant gagner beaucoup de temps. Nous avons préféré choisir Bulma et non Bootstrap, car ce premier était bien plus simple d'utilisation et plus instinctif pour des résultats d'une qualité équivalente.

IV - Conclusions

Ce projet nous a permis de découvrir l'utilisation de spring et de mettre en place un service web avec. Cela nous a aussi permis d'utiliser une REST API et de faire différentes manipulations dessus. Nous avons pu mettre en place une application par l'intermédiaire du design pattern MVC qui était un design pattern que nous n'avions jamais mis en place. Ce projet nous à permis de comprendre ce que représentait le développement d'une application fullstack, c'est-à-dire le côté back end comme le côté front end. En ce qui concerne les perspectives il sera nécessaire d'améliorer la vitesse d'actualisation de la base de données si le nombre de requête journalière augmente, de plus l'API actuelle ne gère pas les vols en cours, changer d'API est une voie envisageable si le projet prend de l'ampleur, il est aussi envisageable d'afficher les détails d'un vol directement sur la carte, enfin, pour l'instant l'aspect gestion de compte n'est pas d'une grande utilité mais si nous nous développons vers une API payante, mettre en place un abonnement pourrait permettre de couvrir les frais.