



ÉCOLE NATIONALE
DES SCIENCES
GÉOGRAPHIQUES

Ecole Nationale des
Sciences Géographiques



Université Gustave Eiffel

Rapport de projet JavaEE

Cycle : Ingénieur 3ème année - TSI

Rapport du site Tropik Airlines

Omran Edoo, Lina Saba

20 Janvier - 13 février 2023

ÉCOLE NATIONALE DES SCIENCES GÉOGRAPHIQUES

Résumé

Dans le cadre du cours d'apprentissage de JavaEE, nous avons été invités à prendre part à un projet de développement de site web intitulé *Tropik Airlines*. L'objectif est de créer une application web de gestion des participants à des colloques.

Ces événements sont en général organisés par des groupes de personnes, autour de la réservation des billets d'avion en ligne. Le site va permettre de gérer les événements et l'inscription des participants. Les administrateurs vont alors créer des événements avec des paramètres prédéfinis et les participants vont pouvoir s'y inscrire.

L'un des principaux objectifs de ce projet est de se familiariser avec JavaEE, Spring et Hibernate. Le but de ce rapport est de faire la synthèse de notre expérience.

Table des matières

Introduction	3
1 Présentation des outils utilisés	5
1.1 Technologies	5
1.2 Modèle MVC	7
2 Fonctionnalités de l'application	9
2.1 Application	9
2.2 Contexte	9
2.3 CRUD	10
2.4 Spring security	11
2.5 Utilisation de Session Factory :	12
2.6 ManyToMany méthode pour la création de la table de liaison entre trip et user . . .	12
2.7 Autres fonctionnalités	13
2.8 Frontend	13
Conclusion	14

Introduction

Le projet de JEE consiste dans la réalisation d'une application web en utilisant JavaEE. L'application représente un site de réservation de billets d'avion, un site web qui va permettre aux utilisateurs de trouver et de réserver des billets d'avion en ligne. Les utilisateurs peuvent généralement effectuer une recherche en saisissant leurs dates de voyage, leur point de départ et leur destination. Le site affiche alors une liste de vols disponibles et leurs tarifs correspondants, permettant aux utilisateurs de comparer les options et de choisir le vol qui convient le mieux à leurs besoins et à leur budget.

Une fois qu'ils ont choisi leur vol, les utilisateurs peuvent généralement effectuer leur réservation en ligne en fournissant des informations personnelles telles que leur nom, leur adresse électronique et leur moyen de paiement. Le site proposera également de s'identifier pour pouvoir gérer les réservations.

Nous nous sommes intéressés au développement de site web de réservation de billets d'avion pour de multiples raisons :

- Curiosité personnelle pour savoir comment les sites de réservation de billets fonctionnent,
- Simplification du processus de réservation,
- Sujet d'actualité,
- Convenance pour les clients, éviter le déplacement dans une agence de voyage,
- Élargissement du marché

Pour ce faire, nous utiliserons Spring et Hibernate.

1.1 Technologies

Notre site va utiliser Spring et Hibernate pour la persistance des données. Pour le contrôleur, nous utiliserons SpringMVC. Nous avons utilisé SpringBoot pour la configuration de Spring. Pour la sécurité, nous avons également utilisé Spring Security et l'utilisation de Spring Data pour la persistance est possible. Notre projet a utilisé Maven comme outil de build.

1.1.1 JavaEE

Java Enterprise Edition est une extension du framework Java SE qui ajoutent des bibliothèques logicielles réservées à des applications professionnelles.

Le but étant de faciliter le développement d'applications distribuées ; pour des applications évoluant à long terme, et éviter un code beaucoup plus compliqué.

Les applications sont alors déployées et exécutées sur un serveur d'applications.

En effet, JavaEE privilégie une architecture d'applications basée sur les composants, en découpant l'application et séparant les rôles. Les avantages de JEE sont l'interfaçage avec de nombreuses API comme JDBC, JPA, JSP et le découpage de l'application en client/métier/données en suivant le modèle d'architecture MVC.

1.1.2 Spring

Framework Java libre qui définit l'infrastructure d'une application Java. Représente un conteneur "léger" qui implémente les spécifications Java EE ainsi que d'autres modules et pas d'interface à implémenter.

Spring est conçu pour simplifier le développement d'applications Java en fournissant un conteneur léger pour gérer les Java beans et offrir un large éventail de services, tels que la gestion des transactions, la sécurité et l'accès aux données. Il offre également une approche modulaire du développement d'applications, permettant aux développeurs d'utiliser uniquement les composants dont ils ont besoin et d'ajouter des composants supplémentaires si nécessaire.

L'une des principales caractéristiques de Spring est sa capacité à s'intégrer à d'autres frameworks et technologies, tels que Hibernate pour la persistance des bases de données, JPA pour l'accès aux données.

1.1.3 Hibernate

Hibernate est un cadre de persistance Java open-source populaire qui fournit une solution de map-page objet-relationnel (ORM) de haut niveau pour les applications Java. Il nous a permis d'interagir avec les bases de données à l'aide d'objets Java plutôt que d'écrire du code SQL, ce qui facilite l'écriture, la maintenance et le test des applications orientées base de données.

Il nous a offert un moyen souple et efficace de mapper des objets Java sur des tables de base de données et de générer automatiquement du code SQL pour les opérations de base de données courantes, telles que la création, la lecture, la mise à jour et la suppression (CRUD).

Configuration de la base de données

Pour créer la base de données il faut suivre les instructions suivante :

1.1 Procédure d'installation (sous Linux)

1. Sur votre machine, installez le paquet `postgresql`
2. L'installation ajoute un nouvel utilisateur linux et un utilisateur de la base de données nommé `postgres`. Il est seul autorisé, pour l'instant, à se connecter à la base de données. Il ne possède pas de mot de passe, et par mesure de sécurité nous ne lui en attribuerons pas. Nous allons créer un nouvel utilisateur. Connectez avec l'utilisateur `postgres` en utilisant la commande suivante :
`sudo -i -u postgres`
3. Lancez l'invite de commande Postgresql à l'aide de la commande `psql`.
4. Vous êtes désormais connecté à Postgresql en mode administrateur.
5. Créer un utilisateur portant le même nom que votre utilisateur linux (on supposera « test » dans la suite)
`CREATE USER test ;`
6. Donnez-lui le droit de créer une base de données.
`ALTER ROLE test WITH CREATEDB ;`
7. Créez une base portant le même nom que l'utilisateur :
`CREATE DATABASE test OWNER test ;`
8. Ajoutez un mot de passe à votre utilisateur
`ALTER USER test WITH ENCRYPTED PASSWORD 'test' ;`
Personnalisez ici votre mot de passe pour sécuriser votre base.
9. Quittez Postgresql avec `\q`
Quittez l'utilisateur `postgres` avec `exit`
Lancez Postgresql avec la commande `psql` (normalement vous êtes automatiquement connecté avec la base de données `test`).

La limitation d'accès par les rôles ne fonctionnant pas, pas besoin de peupler les tables "roles" et "users_roles". On aurait dans le cas contraire rempli la base de données avec les requêtes suivantes :

```
INSERT INTO role (name) VALUES ('ADMIN');  
INSERT INTO users_roles (username, roles_id) VALUES ('test', 0);
```

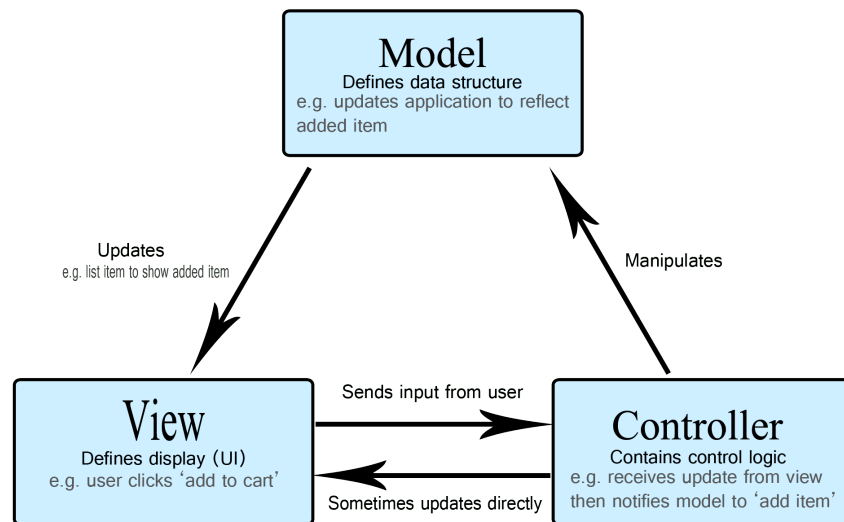
Il faudrait au préalable s'inscrire avec 'test' pour username.

1.1.4 Maven

Maven fournit une structure de projet standard et utilise des conventions pour gérer les dépendances et le processus de construction. Il nous a permis de déclarer les dépendances du projet en un seul endroit et les télécharge automatiquement depuis des dépôts distants.

1.2 Modèle MVC

Un MVC (Model-View-Controller) est un modèle de conception de logiciels utilisé pour mettre en œuvre des interfaces utilisateur, des données et une logique de contrôle. Il met l'accent sur la séparation entre la logique métier du logiciel et l'affichage. Cette séparation permet une meilleure division du travail et une maintenance améliorée.



Model définit les données que l'application doit contenir. Si l'état de ces données change, le modèle le notifie généralement à la **View**. Pour revenir à notre application de vente de billets d'avion, le **Model** spécifierait les données que les éléments concernant les voyages tels que la ville de départ, la ville d'arrivée et l'heure de départ.

View définit la manière dont les données de l'application vont être affichées. Dans notre application, la **View** définit la façon dont les voyages sont présentés aux utilisateurs et reçoit du modèle les données à afficher.

Controller contient la logique qui met à jour le modèle et la vue en réponse aux entrées des utilisateurs de l'application. Ainsi, par exemple, nos pages web vont comporter des formulaires de saisie et des boutons permettant d'ajouter ou de supprimer des voyages. Ces actions nécessitent la mise à jour du **Model**. L'entrée est donc envoyée au **Controller**, qui manipule ensuite le **Model** comme il convient, puis envoie les données mises à jour à la **View**.

2.1 Application

Le principe du site est simple : les administrateurs peuvent créer des événements avec un titre, un thème, une date de début, une durée en jours, le nombre maximum de participants, un texte de description, un organisateur, et un type d'événement.

- On doit pouvoir ajouter, lister, modifier, supprimer ces événements dans l'application. Les administrateurs peuvent ajouter des participants avec un nom, un prénom, une adresse email, une date de naissance, une organisation/entreprise, et des observations éventuelles.
- On doit pouvoir ajouter, lister, modifier, supprimer ces participants dans l'application.
- Ce ne sont pas forcément des participants qui pourront se connecter au site, ils pourront être simplement inscrits aux événements via des formulaires.
- On doit disposer d'un formulaire permettant d'inscrire un participant à un événement (cela peut se passer sur la même page que la création d'un participant).

Les administrateurs doivent s'authentifier sur le site (on peut supposer qu'il n'y a qu'un seul administrateur) et les pages doivent être accessibles uniquement pour ceux-ci.

- Si lors de l'inscription d'un participant celui-ci a la même adresse email qu'un participant existant, le site proposera de les fusionner.
- Si le nombre de participants maximum est atteint l'application devra interdire l'inscription.
- Au choix, un des bonus rouge proposé dans la section suivante.

Gérer la persistance de la liste des organisateurs d'événements comme les participants et les événements.

- Gérer les inscriptions multiples, par exemple en proposant un import de fichier CSV pour ajouter des participants en masse à partir d'un fichier de tableur bien formé.
- Créer un système d'authentification complet avec auto-inscription des participants authentifiés sur le site.
- Créer un système de file d'attente pour l'inscription de participants à des événements déjà complets.
- Améliorer l'aspect et l'adaptabilité de votre site en utilisant CSS et/ou Javascript (avec Bootstrap par exemple)
- Toute autre amélioration (que vous pouvez me soumettre si vous avez un doute).

2.2 Contexte

Nous avons choisi de créer un site d'agence de voyage qui propose des vols. L'utilisateur choisit sa destination, sa ville de départ et l'heure de départ selon les vols disponibles. Il peut s'inscrire au site et ensuite se connecter mais il peut également choisir son vol sans être connecté mais il doit alors remplir un formulaire.

2.3 CRUD

CRUD est un acronyme pour les quatre fonctions de base qui sont généralement associées à la persistance dans une base de données via des requêtes HTTP standard tels que POST, GET, PUT, DELETE.

Les opérations de gestion de la base de données par les administrateurs sont fonctionnelles. En effet, ce dernier peut ajouter des voyages, les lire, les mettre à jour et les supprimer de la manière suivante :

Créer : fonction consistant à créer un nouvel enregistrement dans la base de données. Pour l'exemple de la table des trips, ceci est implémenté dans le code avec la méthode *submitTrip*.

Lire : fonction permettant de récupérer les enregistrements existants dans la base de données. Présent dans le code avec la méthode *getTrip*.

Mettre à jour : fonction permettant de mettre à jour les enregistrements existants dans la base de données. La méthode *updateForm* permet de mettre à jour les données

Supprimer : fonction permettant de supprimer des enregistrements de la base de données. Avec la méthode *deleteTrip*, cette fonction est exécutée

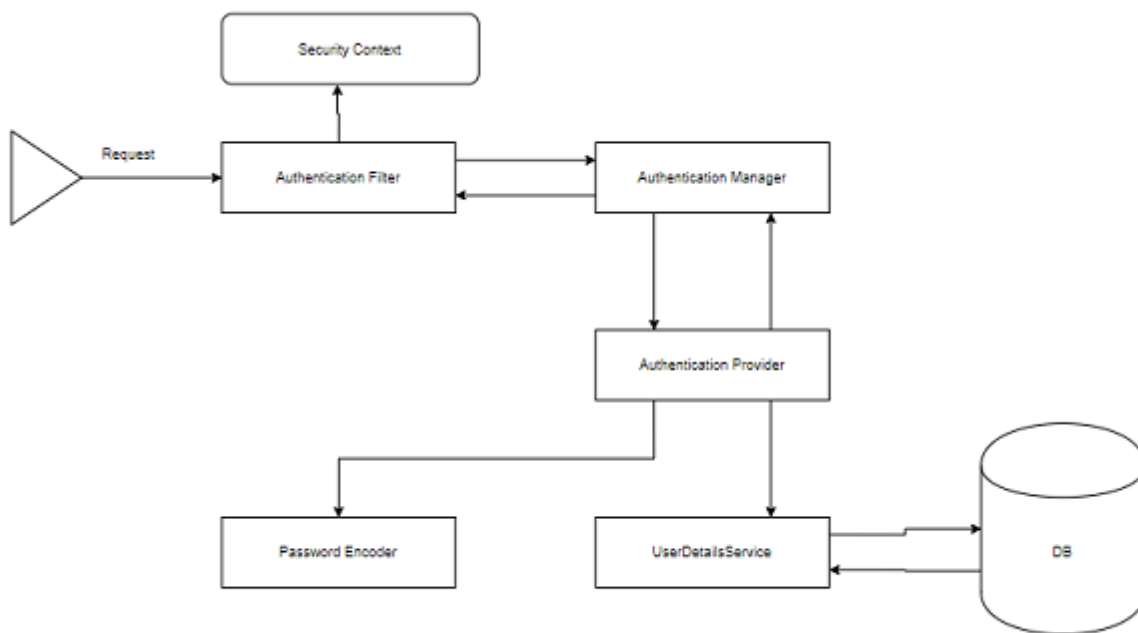
On retrouve ces fonctionnalités dans les pages *users* et *trips* de l'application.

2.4 Spring security

Spring security est un framework d'authentification et de contrôle d'accès qui permet de renforcer la protection des ressources des application Spring.

Le fonctionnement du framework est le suivant, chaque requête est analysée et passe par des filtres. En fonction de si l'utilisateur est trouvé dans la base de données (ou autre système de stockage des données) on gère l'accès aux données ou l'on renvoi des exeptions.

Voici un diagramme décrivant l'architecture de spring security :




Nous avons utilisé spring security afin d'ajouter des options d'authentification à notre site. Les fonctionnalités que nous avons essayé d'implémenter sont :

- la gestion de l'authentification (fonctionnalité principale du framework)
- l'encodage et le décodage des mots de passe
- la gestion d'autorisation d'accès aux pages en fonction du rôle des utilisateurs (administrateur, utilisateur enregistré, visiteur).

La gestion de l'authentification marche parfaitement, elle est gérée par la méthode "authenticate" appartenant à la classe "AuthProvider".

L'encodage des mot de passe se fait grâce à la classe "PasswordEncoder" et son implémentation "BCryptPasswordEncoder" qui sont des classes appartenant à la librairie. Une fois encodés, les mots de passes sont stockés dans la base de données dans la table users.

password	
character varying (255)	
\$2a\$10\$L8jWQtqtBUiie8XgKjg1tu/jOz/g2hEdx6PE1wbc/9q1OrAjsSrS	
\$2a\$10\$.NdnxZ4f2jREq40OMnzhzr.zon8h.Tv15qVQhhlG76hJzZie/z8K	

Les permissions se gère dans la classe "ApplicationConfig" qui hérite de la classe "WebSecurityConfigurerAdapter" où pour chaque page existante on peut indiquer qui peut y accéder. Pour instancier

les rôles notamment d'administrateur, une table "roles" lié à la table "users" par une table de jointure "users_roles" a été créée. Lorsqu'un utilisateur tente d'accéder à une page dont il n'a pas d'autorisation, cela lui renvoi à la page d'authentification s'il y accède via une redirection (en cliquant sur un bouton sur le site par exemple) ou vers une page d'erreur appelée page 403 lui indiquant que l'accès lui est refusé s'il tente de taper l'url correspondant.

La redirection pour les utilisateur non connectée fonctionne, néanmoins nous n'avons pas réussi à limiter les accès en fonction des rôles. En effet, il nous semble avoir implémenté toutes les éléments nécessaires et nous réussissons lorsqu'un utilisateur est connecté à déterminer son rôle (nous pouvons l'afficher), cependant, l'autorisation en elle-même dans le fichier ApplicationConfig ne semble pas fonctionner et refuse l'accès à aux administrateur aux pages dont on leur paramètre l'accès. Nous avons donc au final autorisé l'accès aux utilisateurs connectés à toutes les pages.

Nous aurions pu ajouter une simple colonne administrateur à la table users qui définirait si l'utilisateur est un administrateur et ou non mais cela occulterait l'utilité de spring security.

Spring security propose d'autre fonctionnalités comme par exemple des limites de tentatives de connexion que nous n'avons pas implémenté.

2.5 Utilisation de Session Factory :

L'utilisation de SessionFactory est un élément clé de l'architecture de la bibliothèque Hibernate, qui est un framework populaire pour la persistance des données en Java. La SessionFactory est utilisée pour créer des sessions (Session) qui peuvent être utilisées pour accéder et manipuler les données dans une base de données.

Les avantages de l'utilisation de SessionFactory sont les suivants :

Performances : La SessionFactory est créée une seule fois pour chaque application et peut être partagée par plusieurs sessions. Cela permet de gagner en performance en évitant la création de nouvelles instances de Session pour chaque opération. Connexion à la base de données : La SessionFactory s'occupe de la gestion de la connexion à la base de données. Il peut établir une connexion avec la base de données, la maintenir et la libérer lorsque ce n'est plus nécessaire. Configuration : La SessionFactory peut être configurée pour utiliser des options spécifiques telles que le cache de second niveau, les stratégies de verrouillage et les stratégies de génération de clé primaire. En résumé, l'utilisation de SessionFactory permet de gérer de manière efficace les sessions d'accès à la base de données dans une application Java qui utilise Hibernate pour la persistance des données.

2.6 ManyToMany méthode pour la création de la table de liaison entre trip et user

Pour pouvoir créer une table qui va permettre de faire la liaison entre les table *users* et *trips* et créer une nouvelle table qui quand un voyageur va s'inscrire à un voyage va le mettre à jour dans la table *users_trips*.

ManyToMany est une annotation utilisée en Java pour définir une relation de type "plusieurs à plusieurs" entre deux entités dans le cadre d'une architecture basée sur Hibernate. Il fait toujours une table de jointure.

Nous avons également utilisé les méthodes "findByColumnName" et "findByColumnNameAndColumnName" dans les contrôleurs afin de rechercher des entités dans une base de données en fonction d'une valeur de colonne spécifique. Ceci permet à l'utilisateur de choisir CityA, soit la ville de départ puis ensuite de choisir entre les villes d'arrivée qui s'offrent à lui. Ainsi cela fait éviter que l'utilisateur recherche des voyages qui n'existent pas dans la base de données.

2.7 Autres fonctionnalités

Lorsqu'un utilisateur essaie de s'inscrire avec un email déjà existant, on lui propose de mettre à jour ses informations.

Lorsqu'un utilisateur tente de s'inscrire avec un username existant, on lui informe qu'il doit en choisir un autre.

Pour chaque vol un nombre maximum de participant existe. Si le nombre de participants maximum est atteint le vol n'est plus affiché.

2.8 Frontend

Nous avons tiré profit de la librairie Bootstrap afin de rendre notre site responsive et d'améliorer son esthétique.

Nous l'avons principalement utilisé pour la barre de navigation et les formulaires. Nous souhaitons également mettre en place d'autres éléments comme un carrousel d'images que les administrateurs pourraient actualiser en important des images afin de rendre le site plus attrayant, cependant le temps nous a fait défaut.

Conclusion

Nous avons pu implémenter un site fonctionnel proposant des voyages qui présente les fonctionnalités principales attendus : L'authentification des utilisateurs et le stockage en base de données, la gestion de la persistance de données avec les opérations CRUD. Nous avons également gérer la sécurité avec spring security qui afin de gérer l'authentification et l'encodage, cependant, nous n'avons pas réussi à gérer les permissions d'accès aux pages en fonction des rôles. La compréhension et l'utilisation du framework nous a prit beaucoup de temps et a nécessité une restructuration du code. Nous n'avons ainsi pas pu implémenter d'autres fonctionnalités bonus comme l'import de fichier CSV. Enfin, nous avons pu améliorer l'aspect du site et gérer la responsivité avec bootstrap.