

Nom de naissance

▸ NESIC

Prénom

▸ Alexandre

Adresse

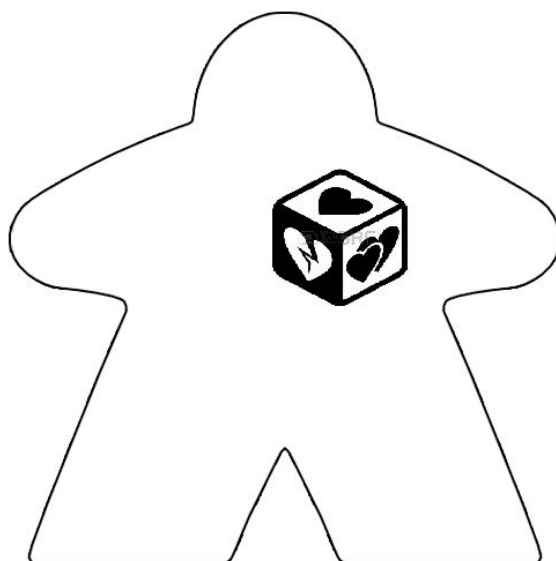
▸ 10 rue de Guébriant
75020 PARIS

Titre professionnel visé

Développeur Logiciel - Niveau III

PROJET RÉALISÉ PAR :

NESIC Alexandre



SOMMAIRE

Titre professionnel visé	1
INTRODUCTION	4
RÉSUMÉ DU PROJET EN ANGLAIS	6
LISTE DES COMPÉTENCES	7
Maquetter une application / Concevoir une base de données	7
METTRE EN PLACE UNE BASE DE DONNÉES	7
DÉVELOPPER DES COMPOSANTS D'ACCÈS AUX DONNÉES	7
DÉVELOPPER UNE INTERFACE UTILISATEUR	7
DÉVELOPPER UNE APPLICATION MOBILE	7
DÉVELOPPER DES PAGES WEB EN LIEN AVEC UNE BASE DE DONNÉES	8
UTILISER L'ANGLAIS DANS SON ACTIVITÉ PROFESSIONNELLE EN INFORMATIQUE	8
CAHIER DES CHARGES	9
GESTION DE PROJET	10
1. Git/GitHub	10
2. Trello	11
SPÉCIFICATIONS FONCTIONNELLES	12
1. Diagrammes des cas d'utilisation	12
2. Diagrammes d'activité	14
3. Wireframes	16
SPECIFICATIONS TECHNIQUES	22
1. Outils et technologies utilisés	22
Diagramme des classes (MOO)	26
Diagramme de la base de données (MPD)	27
RÉALISATION	28
Mise en place du Back end	29
1.1. La base de données	29
1.2. Le codage	30
1.2.1 Le POJO (Plain Old Java Object)	31
1.2.2. Le Repository	33
1.2.3. Le Controller	34

1.2.4. Le Service	36
Mise en place du Front end	37
Présentation du template layout.html	38
Présentation du template catalogue.html	39
Présentation du template gestionCatalogue.html	40
ÉVOLUTION À VENIR	41
CONCLUSION	42
REMERCIEMENTS	43
LEXIQUE	44

INTRODUCTION

Le groupe La Poste ayant pour objectif de digitaliser l'ensemble de ses services, son besoin en développeurs web et mobile s'est accrue. Elle a donc mis en place, par l'intermédiaire de l'école Simplon, des formations de reconversion à l'adresse de l'ensemble de ses employés, aboutissant au passage du titre professionnel de développeur logiciel. Dans ce contexte, nous sommes dix-neuf à avoir réussi les épreuves de sélection, pour la deuxième itération de ces formations développeur logiciel, basées sur le langage Java.

La formation se déroule sur neuf mois, en alternance avec un service informatique du groupe La Poste (maison mère ou filiale). L'alternance est un bon complément à la formation, dans la mesure où de nombreux concepts vus en formation restent assez abstraits sans pratique. L'application, ou la visualisation, de ces concepts en entreprise permettent une meilleure compréhension.

Depuis début mai, je consacre mon temps en formation à l'élaboration du projet dont je vais vous parler. Veuillez noter que les extraits de code présentés sont susceptibles d'évoluer, car je suis toujours en phase de développement.

Le projet présenté pour le titre professionnel « développeur logiciel » est une application web permettant la gestion du catalogue d'une ludothèque, des comptes de ses adhérents, le suivi des réservations et la publication d'articles concernant la vie de l'établissement. L'application permettra aussi d'avoir une alerte lors des nouvelles demandes de réservation et de suivre ces dernières en générant une alerte, également, lors des retards de restitution.

J'ai réalisé le cahier des charges, en tâchant de définir quels seraient les besoins d'un ludothécaire et des utilisateurs du service. J'ai toutefois consulté mon entourage, des collègues, mes co-apprenants et formateurs, afin de recueillir leurs besoins s'ils étaient client et/ou utilisateur.

Les objectifs de cette application sont :

1. Tenir à jour la liste des jeux disponibles au catalogue (module accessible au Ludothécaire).
2. Publier des articles concernant la vie de l'établissement (module accessible au Ludothécaire)
3. Consulter les alertes de demande de réservation et de retard de restitution des jeux (module accessible au Ludothécaire)
4. Tenir à jour les comptes "abonné" (module accessible à l'Administrateur)

5. Modifier certaines informations personnelles du profil (module accessible aux Abonnés)
6. Réserver un jeu (module accessible aux Abonnés)
7. Consulter le catalogue de jeux (module accessible à tous les utilisateurs)
8. Consulter les publications d'articles informatifs (module accessible à tous les utilisateurs)

Pour le développement de l'application, j'ai décidé d'utiliser les technologies suivantes :

1. Pour la partie Back-end :
 - 1.1. Le langage JAVA et son Framework Spring Boot
 - 1.2. L'API de persistance JPA et son implémentation Hibernate
2. Pour la partie Front-end :
 - 2.1. Html, CSS et JavaScript
 - 2.2. Les Frameworks Thymeleaf et Bootstrap
3. Pour la gestion de la base de données, PostgreSQL et Git/GitHub pour la gestion des versions du projet.

ABSTRACT

I decided to develop a web application aiming to manage a board game library.

First, the tool is designed to help the librarian to manage the games catalog, by creating, updating and deleting the games available for subscribers. He'll be able to consult the subscriber's renting story. And he'll be able to post articles concerning the institution's life (informations, events, etc...).

Secondly, the app will give the opportunity to users to be informed of the institution's life, to consult the catalog and the games' availability. If the user is a subscriber, he'll be able to login via his account, and will be able to book a game he wishes to rent. He'll have the possibility to update some of his personal datas like email, phone number, password and alias. And he'll be able to consult his rental history.

Finally, there will be an Admin role aiming to manage the users account. He'll be in charge to create or delete the accounts, he'll still be able to update them too, if some subscribers encounter some difficulties with the updating function.

To develop this app, I used the following technologies :

* Front-end part :

- HTML, CSS and JavaScript languages
- Bootstrap and Thymeleaf frameworks

* Back-end part :

- The Java language with the JPA interface (and the Hibernate implementation)
- The Spring Boot framework.

For the database management, I chose PostgreSQL et Git/GitHub the project versionning purpose.

LISTE DES COMPÉTENCES

MAQUETTER UNE APPLICATION / CONCEVOIR UNE BASE DE DONNÉES

Afin de réaliser la maquette de l'application, j'ai utilisé :

- Draw.io, qui offrait l'avantage d'être complet (réalisation des différents diagrammes et wireframes) et de permettre la sauvegarde de mon travail sur mon Google Drive.
- Pour la gestion de la base de données, j'ai utilisé PostgreSQL comme logiciel de base.

METTRE EN PLACE UNE BASE DE DONNÉES

J'ai utilisé Jpa/Hibernate pour générer les tables automatiquement, après avoir créé manuellement la base de données.

Enfin, j'ai ajouté un compte utilisateur avec les autorisations nécessaires, pour être utilisé lors du déploiement.

DÉVELOPPER DES COMPOSANTS D'ACCÈS AUX DONNÉES

Pour me permettre de développer efficacement des composants d'accès aux données, j'ai utilisé l'interface JPA et son implémentation Hibernate.

DÉVELOPPER UNE INTERFACE UTILISATEUR

J'ai développé l'interface utilisateur avec Thymeleaf, un moteur de template écrit en Java, particulièrement adapté au modèle MVC. Ce framework apporte une meilleure séparation des responsabilités entre l'affichage et le contenu. Les développeurs et les web designers peuvent ainsi travailler sur la même partie du code, avec un impact modéré sur le travail de l'autre.

DÉVELOPPER UNE APPLICATION MOBILE

Lors du développement de l'application, j'ai gardé à l'esprit que celle-ci devait avoir un affichage optimisé, quelque soit la taille de l'écran de l'appareil utilisé.

DÉVELOPPER DES PAGES WEB EN LIEN AVEC UNE BASE DE DONNÉES

Avec la mise en place d'une API Rest dans le back-end et en utilisant les requêtes HTTP depuis le front-end, les utilisateurs peuvent interagir avec la base de données.

UTILISER L'ANGLAIS DANS SON ACTIVITÉ PROFESSIONNELLE EN INFORMATIQUE

Même si j'ai décidé de développer principalement en français, j'ai eu recours à de nombreuses ressources en anglais et je suis capable de lire l'anglais comme du français car je le pratique depuis longtemps dans le cadre de mes hobbies.

CAHIER DES CHARGES

Qui ?

Pour le ludothécaire afin de gérer son catalogue de jeux et la liste de ses abonnés.

Pour les curieux afin de découvrir la ludothèque et son fonctionnement.

Pour les abonnés, afin de gagner du temps et s'assurer de la disponibilité des jeux à disposition et/ou qu'ils souhaiteraient réserver.

Quoi ?

L'objectif de cette application est donc de satisfaire aux besoins des différents acteurs susceptibles de l'utiliser.

Pourquoi ?

Afin de créer une application simple d'utilisation pour les utilisateurs.

Comment ?

Via la création d'une interface simple où chaque page aura une fonction dédiée.

GESTION DE PROJET

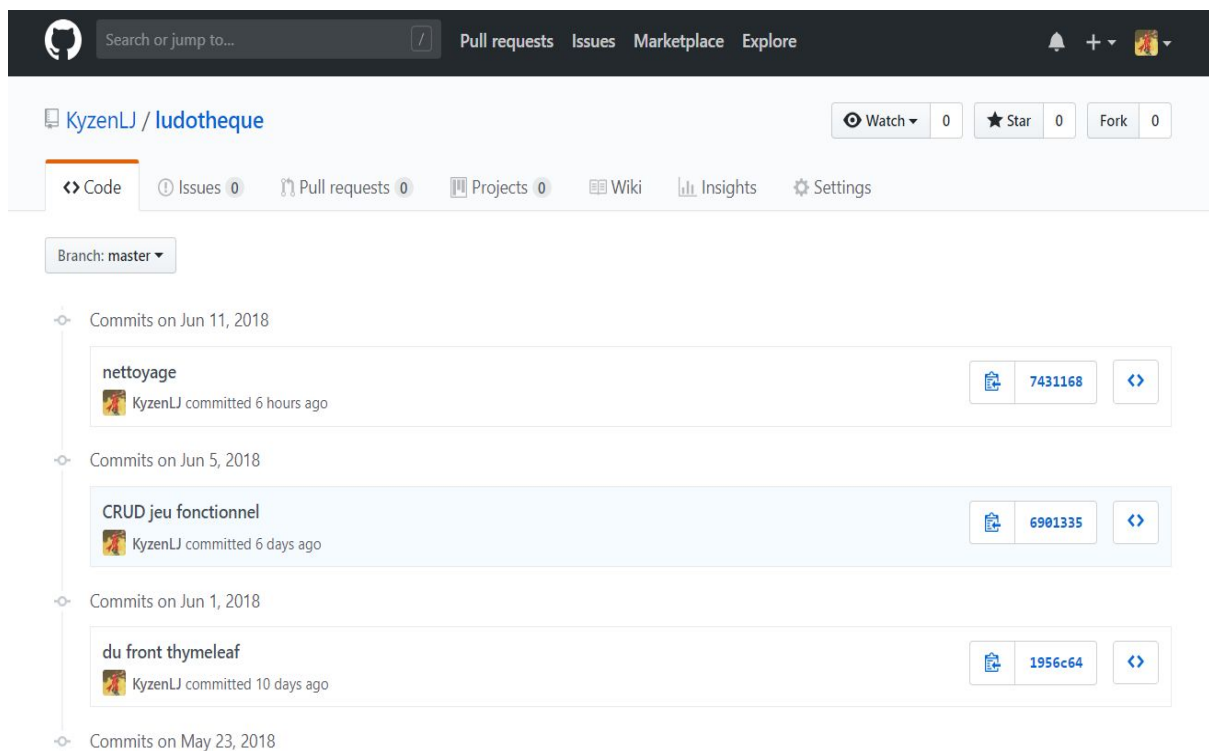
Pour m'aider à suivre l'avancement des tâches et les versions du code j'ai utilisé plusieurs outils :

1. Git/GitHub

GitHub est un service web d'hébergement et de gestion de développement de logiciels, utilisant le logiciel de gestion de version : Git.

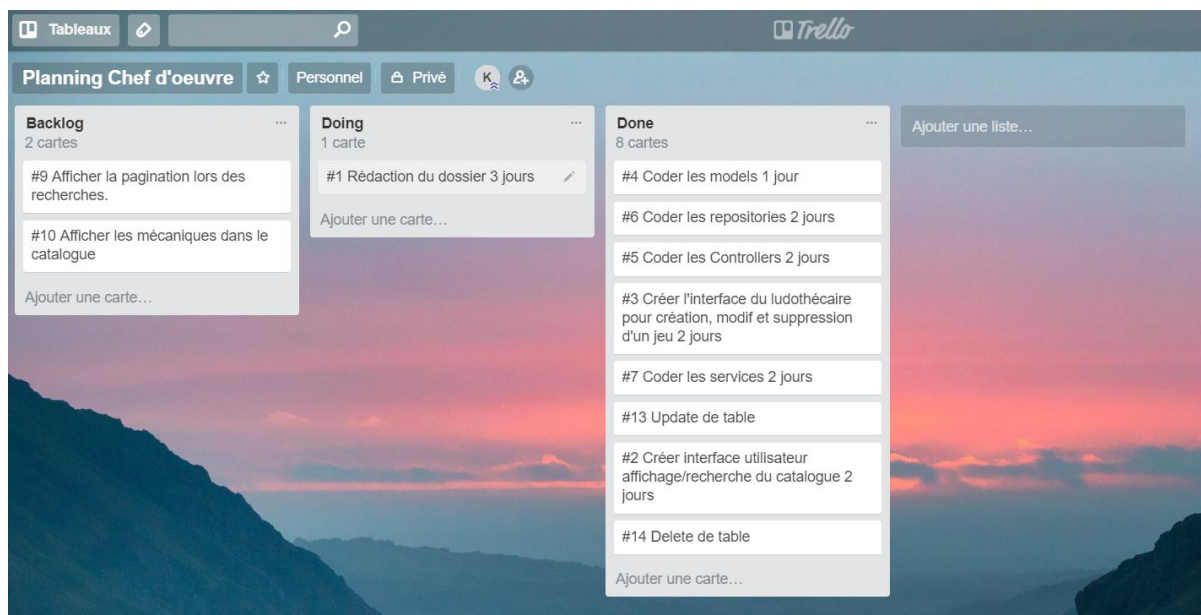
Git est un logiciel qui permet de stocker un ensemble de fichiers en conservant la chronologie de toutes les modifications qui ont été effectuées dessus. Il permet notamment de retrouver les différentes versions d'un lot de fichiers connexes.

GitHub propose des comptes professionnels payants, ainsi que des comptes gratuits pour les projets de logiciels libres. Le site assure également un contrôle d'accès et des fonctionnalités destinées à la collaboration comme le suivi des bugs, la gestion de tâches et un wiki pour chaque projet.



2. Trello

Trello est un outil de gestion de projet en ligne inspiré de la méthode Kanban. Il est basé sur une organisation des projets sous forme de planches listant des cartes, chacune représentant des tâches. Les cartes sont assignables à des utilisateurs et sont mobiles d'une planche à l'autre, traduisant leur avancement. La version de base est gratuite, tandis qu'une formule payante permet d'obtenir des services supplémentaires.



Trello m'a permis de suivre de manière continue l'avancement de mon projet, de voir les tâches restantes à produire, et de noter des informations essentielles de chaque module.

Grâce à ces outils, j'ai pu travailler efficacement sur plusieurs machines sans crainte de perdre l'avancement de mon travail ou d'oublier une partie des fonctionnalités.

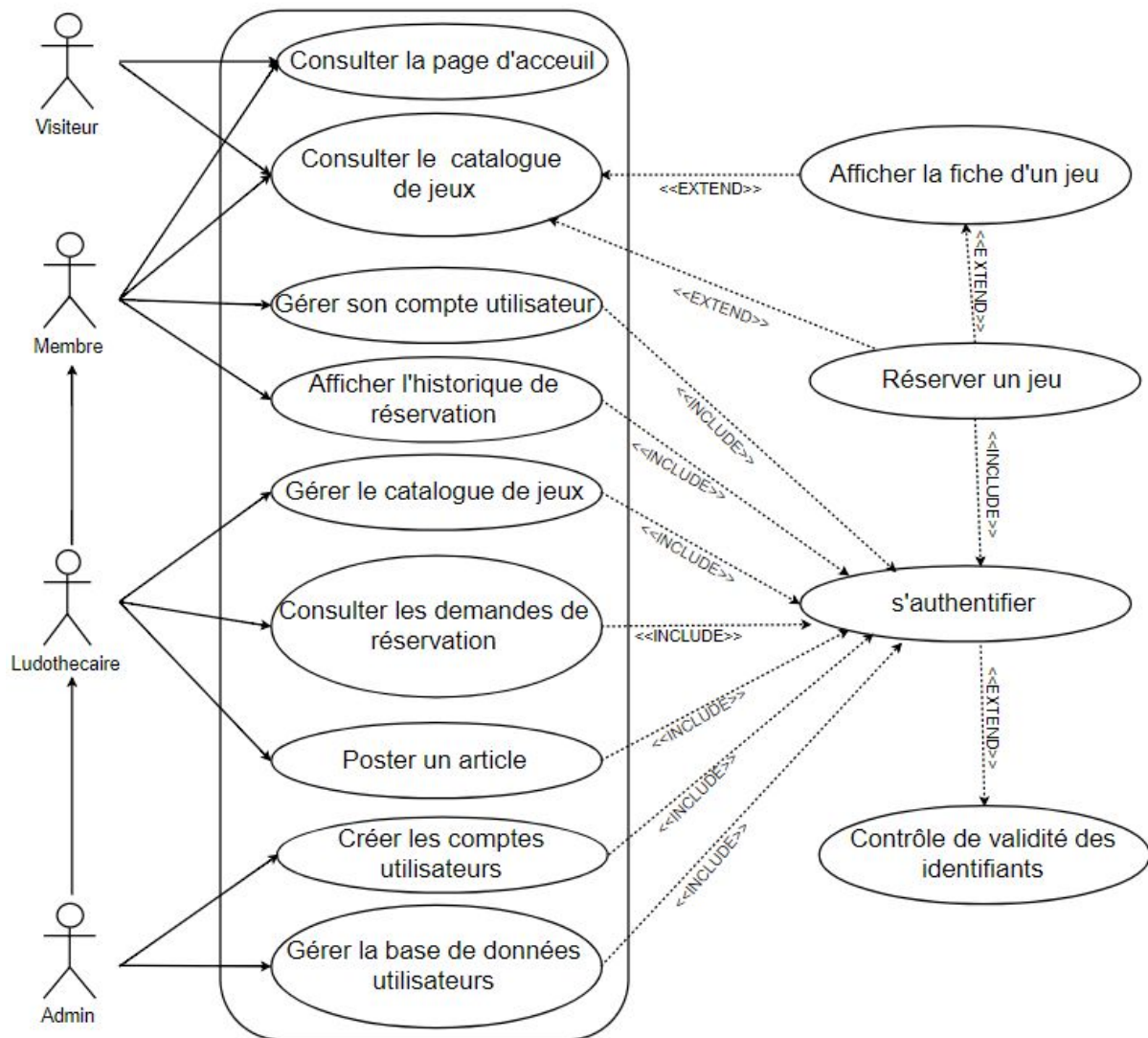
SPÉCIFICATIONS FONCTIONNELLES

Pour concevoir l'application, j'ai utilisé le Langage de Modélisation Unifié, de l'anglais Unified Modeling Language (UML), qui est un langage de modélisation graphique à base de pictogrammes conçu pour fournir une méthode normalisée afin de visualiser la conception d'un système. Il permet d'utiliser un langage commun facilitant la conception d'un projet en développement logiciel et en conception orientée objet.

1. Diagrammes des cas d'utilisation

Le rôle des diagrammes de cas d'utilisation est de recueillir, d'analyser et d'organiser les besoins, et de recenser les grandes fonctionnalités d'un système. Un diagramme de cas d'utilisation capture le comportement d'un système, d'un sous-système, d'une classe ou d'un composant tel qu'un utilisateur extérieur le voit. Il scinde la fonctionnalité du système en unités cohérentes, les cas d'utilisation, ayant un sens pour les acteurs.

Ce diagramme nous permet d'exprimer clairement le besoin des utilisateurs de l'application. Il est une vision orientée utilisateur au contraire d'une vision informatique.



La mention "include" indique que A --include-->B : ainsi pour effectuer l'action A, l'action B doit être réalisée avant.

La mention "extend" indique que A --extends-->B : ainsi la réalisation de l'action B peut induire l'action A.

2. Diagrammes d'activité

Un diagramme d'activité permet de modéliser un processus interactif, global ou partiel pour un système donné (logiciel, système d'information). Il permet d'exprimer une dimension temporelle sur une partie du modèle, à partir de diagrammes de classes ou de cas d'utilisation, par exemple.

Le diagramme d'activité est une représentation proche de l'organigramme ; la description d'un cas d'utilisation par un diagramme d'activité correspond à sa traduction algorithmique. Une activité est l'exécution d'une partie du cas d'utilisation.

Diagramme représentant la consultation du catalogue de jeux :

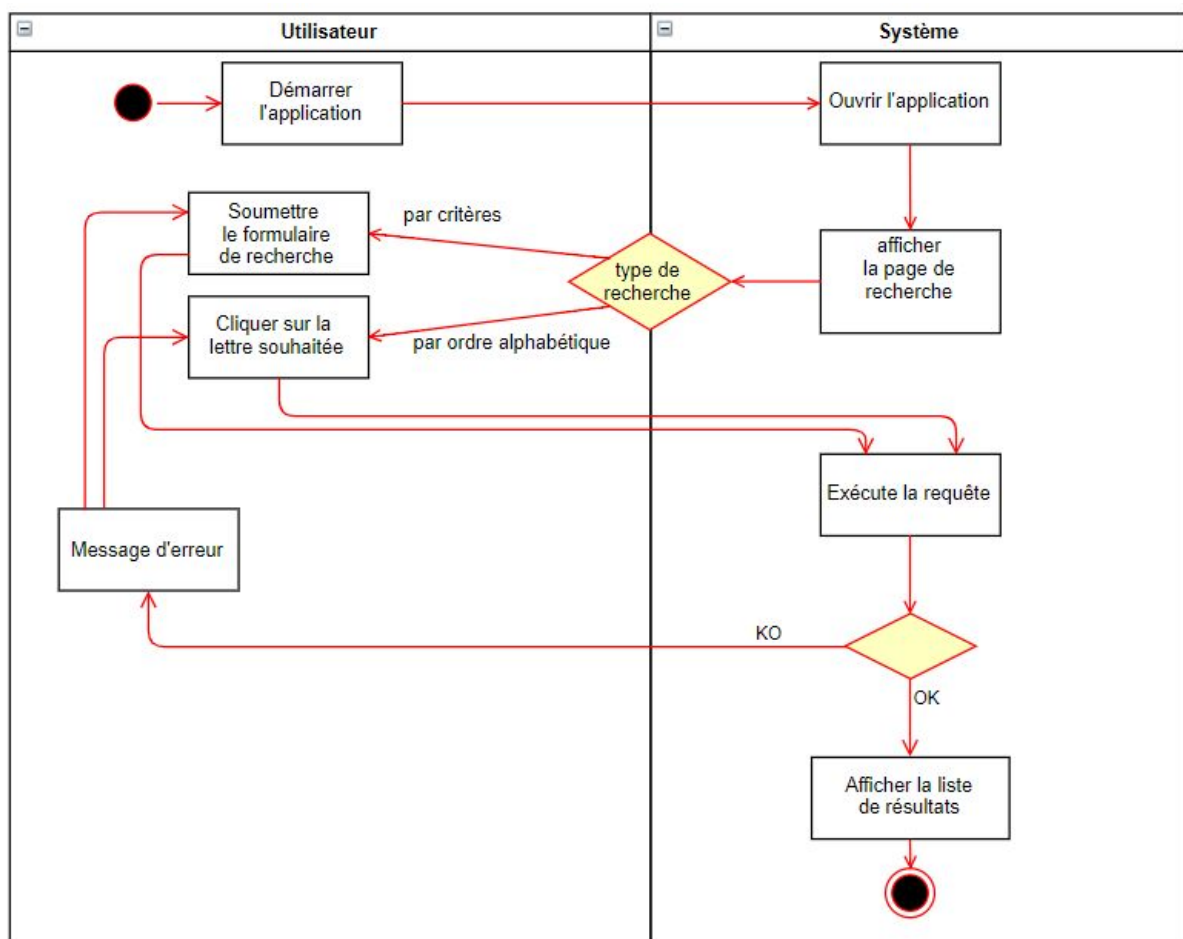
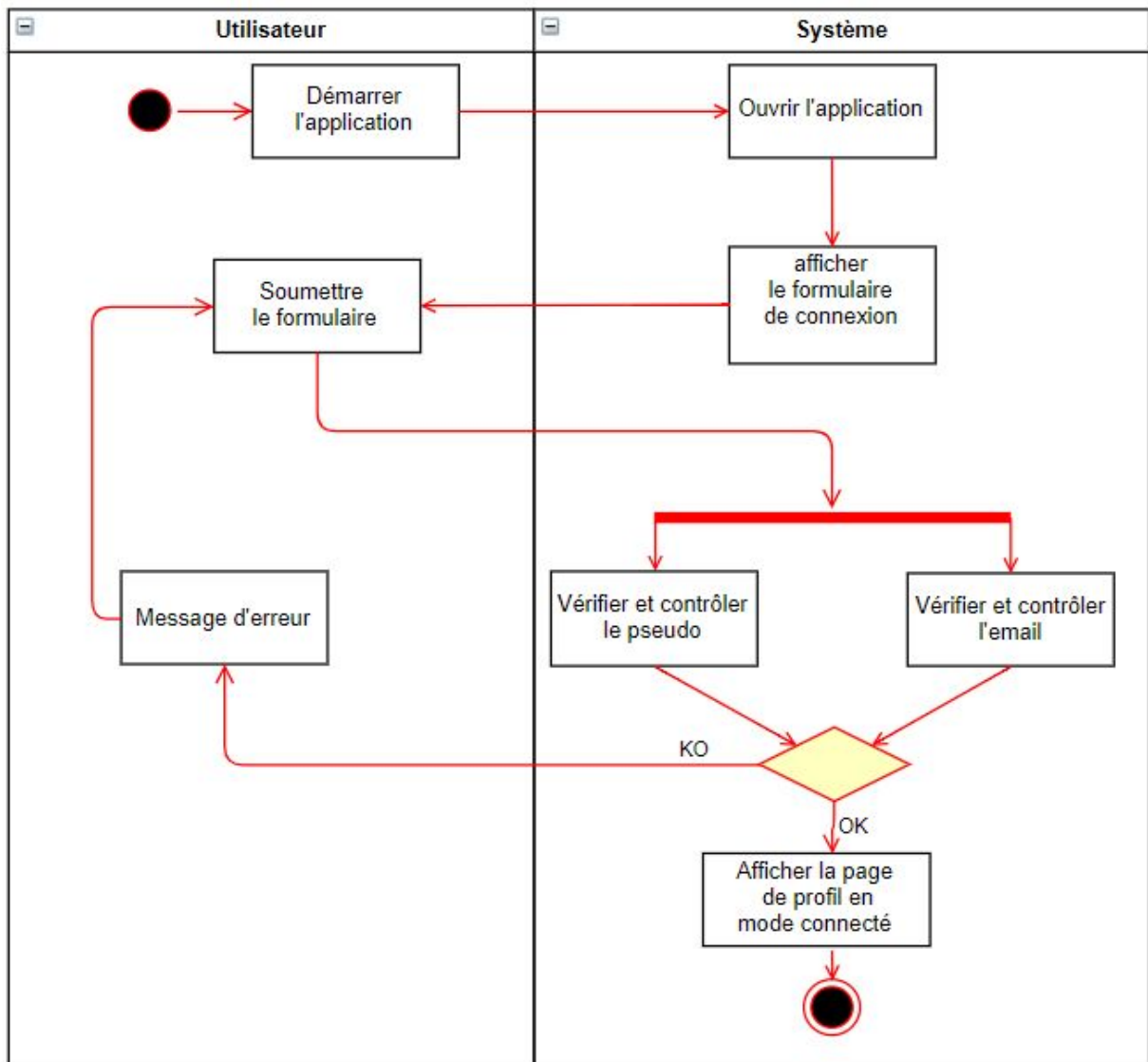


Diagramme représentant la connexion :

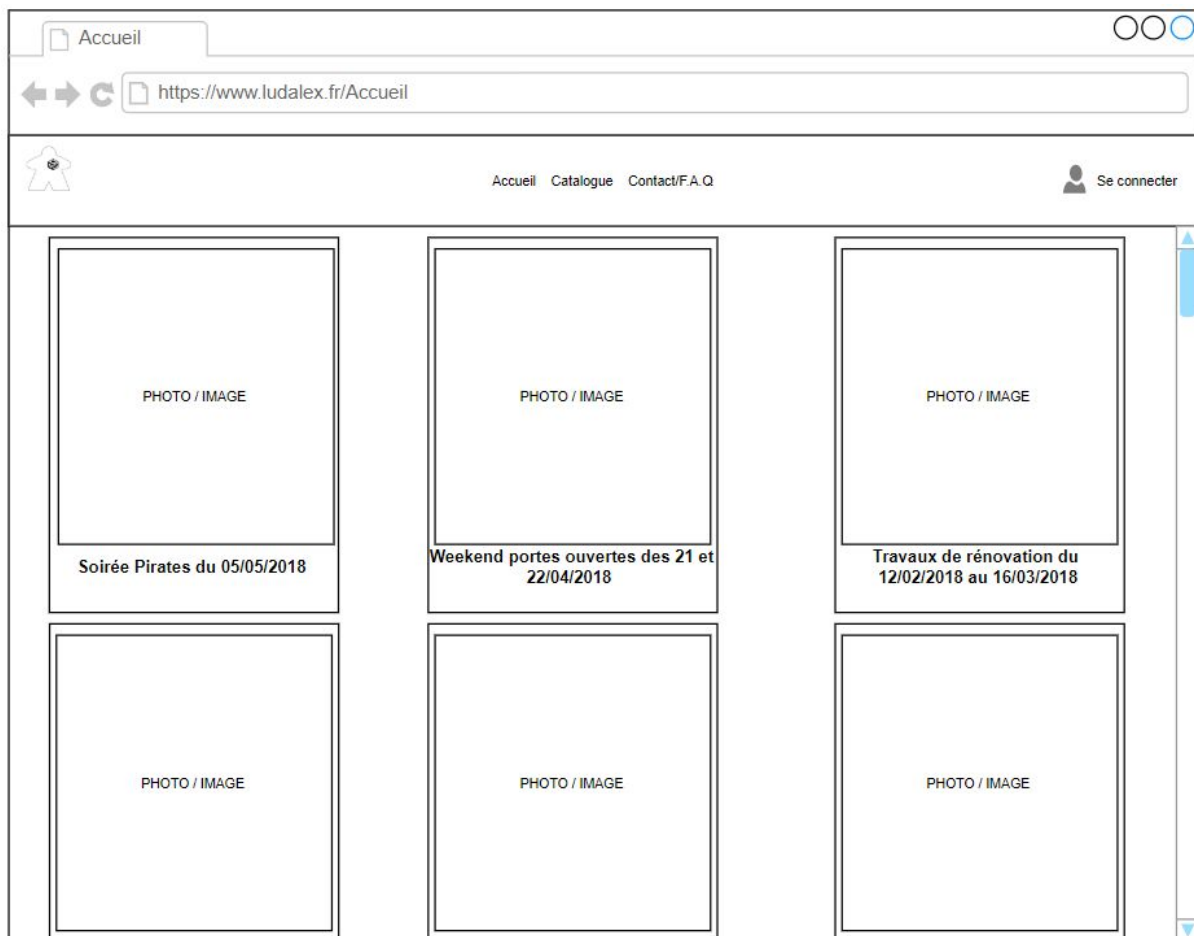


3. Wireframes

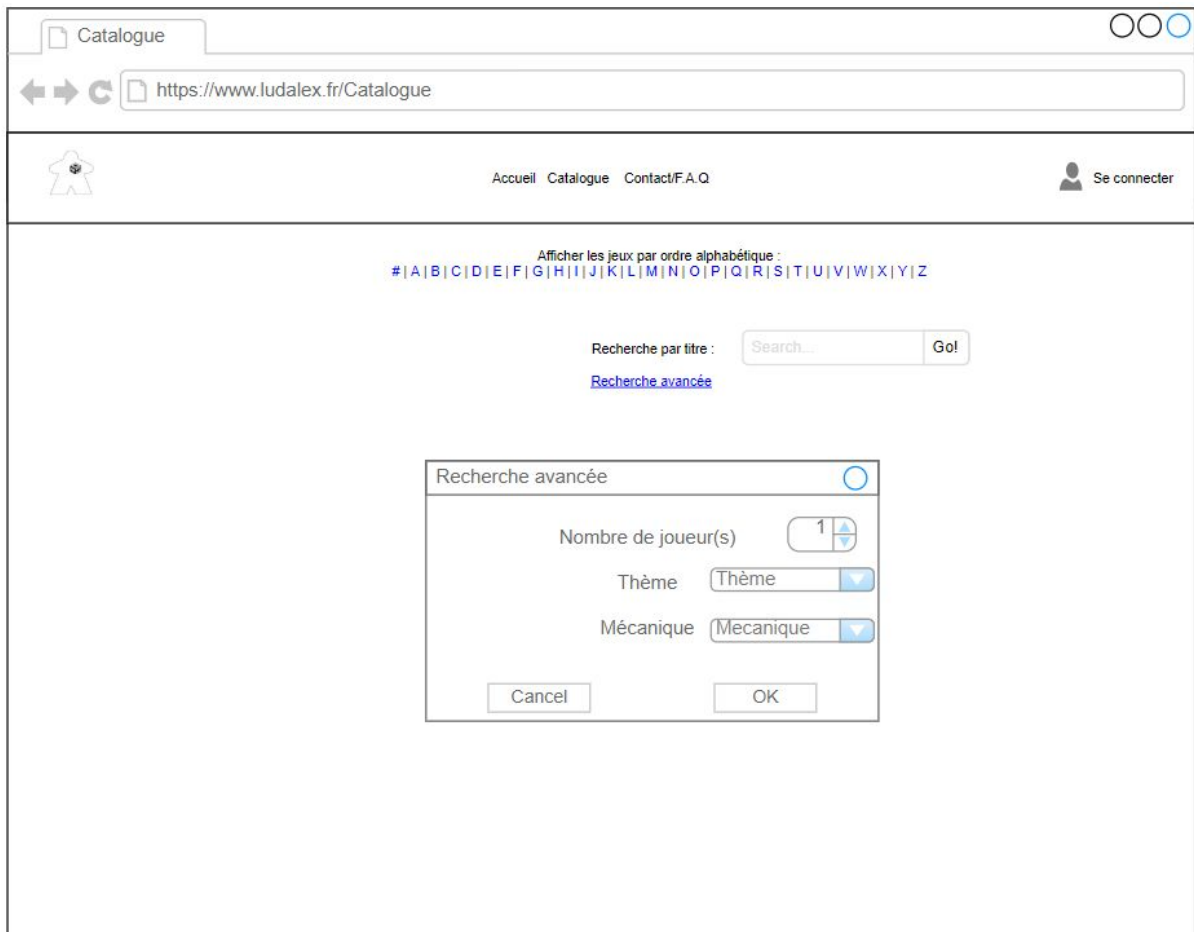
Pour avoir un premier aperçu de l'interface utilisateur, j'ai créé des wireframes.

Le wireframe est l'étape de la conception d'une interface homme-machine qui consiste à réaliser une maquette des pages web. Durant la conception, ce prototype permet de confirmer les exigences spécifiées dans le cahier des charges.

La page d'accueil :



La page de recherche de jeux :



The screenshot shows a web browser window with the address bar displaying `https://www.ludalex.fr/Catalogue`. The page has a header with a logo, navigation links (Accueil, Catalogue, Contact/F.A.Q.), and a 'Se connecter' button. The main content area features a link to 'Afficher les jeux par ordre alphabétique' followed by a list of letters from A to Z. Below this is a search bar with the text 'Recherche par titre :', a 'Search...' input field, and a 'Go!' button. A link for 'Recherche avancée' is also present. A modal window titled 'Recherche avancée' is open, showing options for 'Nombre de joueur(s)' (set to 1), 'Thème' (set to 'Thème'), and 'Mécanique' (set to 'Mécanique'). The modal has 'Cancel' and 'OK' buttons.

Catalogue

https://www.ludalex.fr/Catalogue

Accueil Catalogue Contact/F.A.Q. Se connecter

Afficher les jeux par ordre alphabétique :
#|A|B|C|D|E|F|G|H|I|J|K|L|M|N|O|P|Q|R|S|T|U|V|W|X|Y|Z

Recherche par titre : Search... Go!

[Recherche avancée](#)

Recherche avancée

Nombre de joueur(s) 1

Thème Thème

Mécanique Mécanique

Cancel OK

La page d'affichage de résultat de la recherche :

Catalogue

https://www.ludalex.fr/Catalogue

Accueil Catalogue Contact/F.A.Q.

Se connecter

Afficher les jeux par ordre alphabétique :
| A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z

Recherche par titre : Go!

[Recherche avancée](#)

Titre	Thème	Mécanique	Joueur Min	Joueurs Max	Disponible	Réserver
B-sieged	Med-fan	Coopératif	1	6	Non	<input type="checkbox"/>
Castles of Mad King Ludwig	Baroque	Pose de tuile Enchères	1	4	Non	<input type="checkbox"/>
Keyflower	Moyen-age	Pose de tuile Enchères Pose d'ouvriers	2	6	Oui	<input type="checkbox"/>
Lewis & Clark	Historique	Deck building Course	1	5	Oui	<input type="checkbox"/>
Patchwork	Couture	Pose de tuile Gestion du temps	2	2	Oui	<input type="checkbox"/>
Viceroy	Med-fan	Pose de cartes Enchères	1	4	Oui	<input type="checkbox"/>



Ceci
colonne
n'est
visible
qu'en
mode
connecté


<< 1 2 3 4 5 6 7 8 9 >>

La page de Contact / F.A.Q. :

Aide et cont...

https://www.ludalex.fr/faq

 [Accueil](#) [Catalogue](#) [Contact/F.A.Q](#)  [Se connecter](#)

 **Contactez-nous**

Nom Prénom *

Email *

Téléphone ▾

Motif de la demande

Demande (300 caractères max.)

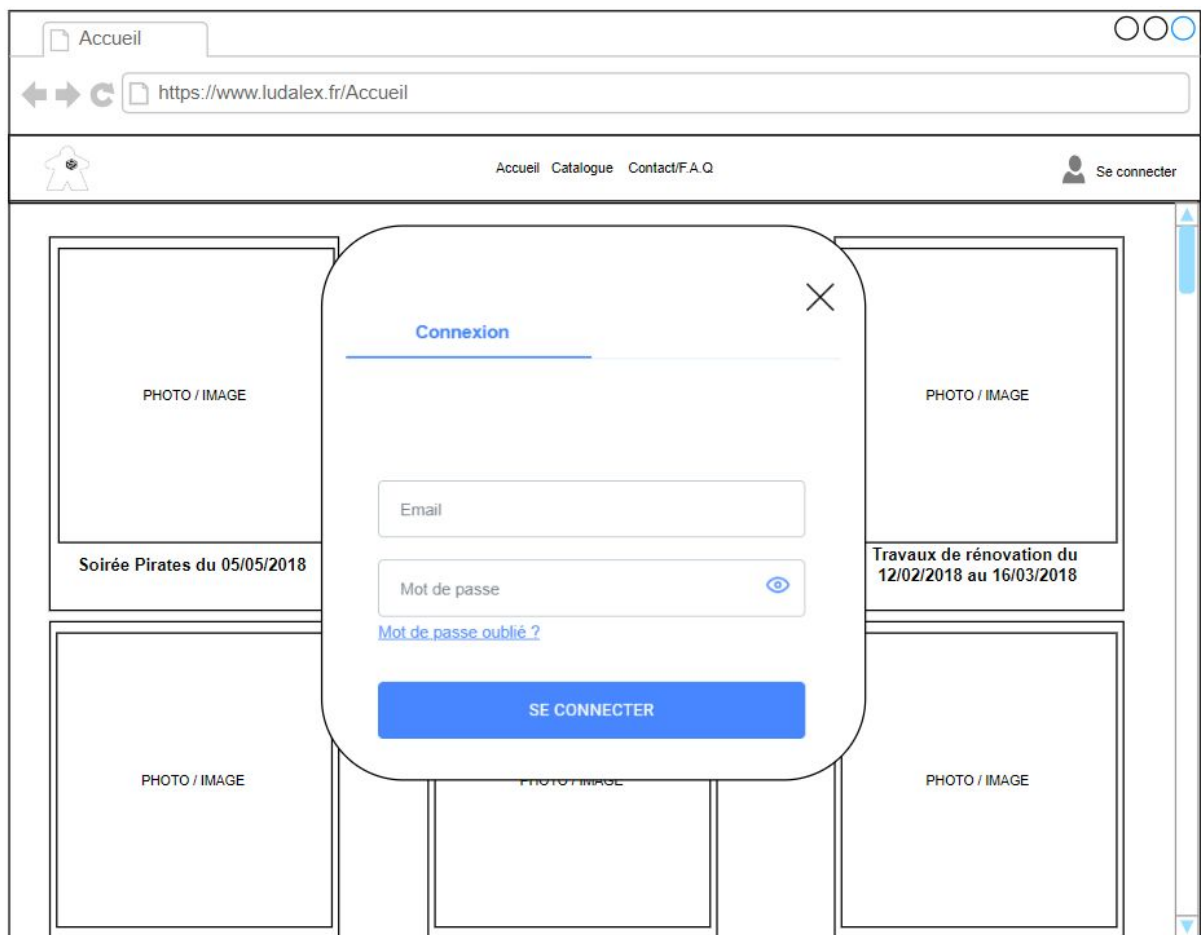
Envoyer

Questions fréquentes

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed varius eu metus consectetur ornare. Mauris posuere, sem a accumsan laoreet, odio nisi molestie dolor, ac ullamcorper eros massa vitae est. Integer sollicitudin congue diam sit amet mattis. Vestibulum lacinia augue quis enim molestie gravida. Phasellus vitae urna portitor, porta orci vel, aliquam nulla. Vivamus nec eros dui. Integer iaculis semper tellus id rutrum. Nam ut est quis arcu vehicula tincidunt vitae in ex.

Sed dapibus maximus magna, id condimentum risus elementum at. Nunc pretium risus vitae ante placerat, in faucibus felis ornare. Integer tortor eros, condimentum pharetra est pharetra, egestas euismod leo. Fusce efficitur rutrum nisi at accumsan. Vestibulum accumsan at lacus vitae convallis. Cras ornare ante vel odio hendrerit, a fermentum nisi sodales. Aliquam facilisis pellentesque ante, eget tempus tellus. Phasellus pretium arcu eros, quis sodales i


La modale de connexion :



La page de Profil :

Profil

https://www.ludalex.fr/Profil



Accueil Profil Gestion Compte Catalogue Gestion catalogue Contact/F.A.Q

Bonjour, utilisateur Déconnexion

Nom : TRUCPrénom : MACHIN
E-mail : machin.truc@yopmail.comMot de passe : *****
[Modifier les informations personnelles](#)

Réervations en cours

Titre	Date de réservation	Date de retrait	Date de retour prévu	Retard	Annuler la réservation
B-sieged	05/05/2018	-	-	non	<input type="checkbox"/>
Castles of Mad King Ludwig	04/05/2018	04/05/2018	11/05/2018	non	<input type="checkbox"/>

<< 1 2 3 4 5 >>

Historique de réservations

Titre	Date de réservation	Date de retrait	Date de rendu	Retard
Vendredi	24/04/2018	25/04/2018	03/05/2018	oui
Dawn of the Zed	08/03/2018	08/03/2018	12/03/2018	non

SPECIFICATIONS TECHNIQUES

1. Outils et technologies utilisés

1.1.1. Back end

J'ai décidé d'utiliser le langage Java, car c'est celui qui nous a été enseigné lors de cette formation. Il s'agit également du langage utilisé par ma future équipe chez Docapost.

De plus, la particularité et l'objectif central de Java sont que les logiciels écrits dans ce langage doivent être très facilement portables sur plusieurs systèmes d'exploitation avec peu ou pas de modifications. En effet, un programme Java s'exécute dans une machine virtuelle (JVM), il est donc indépendant du système d'exploitation et de l'architecture matérielle utilisée.

C'est également un langage typé statiquement, ce qui permet de repérer dès la compilation les erreurs de codage.

J'ai utilisé **Eclipse**, un **EDI** (Environnement de développement intégré) complet et extensible via des plugins, pour développer mon application. Argument de poids en sa faveur est sa gratuité. Il bénéficie également d'une grosse communauté d'utilisateurs, garantissant des mises à jour fréquentes et l'implémentation de nombreuses fonctionnalités optimisant la productivité et limitant les risques d'erreur.

1.1.2. REST

REST (REpresentational State Transfer) ou **RESTful** est un style d'architecture permettant de construire des applications (Web, Intranet, Web Service). Il s'agit d'un ensemble de conventions et de bonnes pratiques à respecter et non d'une technologie à part entière. Il est largement appliqué au sein de l'ensemble des applications Web.

Nous avons appliqué certaines des contraintes requises par l'architecture REST :

- Une URI comme identifiant des ressources

REST se base sur les **URI** (Uniform Resource Identifier) afin d'identifier une ressource. Ainsi une application se doit de construire ses URIs (et donc ses URLs) de manière précise, en tenant compte des contraintes REST. Il est nécessaire de prendre en compte la hiérarchie des ressources et la sémantique des URLs pour les éditer. En construisant correctement les URIs, il est possible de les trier, de les hiérarchiser et donc d'améliorer la compréhension du système.

- Les verbes HTTP comme identifiant des opérations

Une autre règle d'une architecture REST est d'utiliser les verbes HTTP existants plutôt que d'inclure l'opération dans l'URI de la ressource.

- Les réponses HTTP comme représentation des ressources

Il est important d'avoir à l'esprit que la réponse envoyée n'est pas une ressource, c'est la représentation d'une ressource. Ainsi, une ressource peut avoir plusieurs représentations dans des formats divers : HTML, XML, CSV, JSON, etc.

1.1.3. Maven

Maven est un outil de gestion de dépendances automatisé et de compilation. Il met à disposition un dépôt en ligne de bibliothèques Java, sur lequel nous pouvons automatiquement télécharger les éventuelles dépendances requises. Il suffit de les décrire dans un fichier XML, appelé pom.xml (pour Project Object Model), et elles sont directement récupérées et ajoutées au projet.

Il contient également un outil de packaging, permettant rapidement de produire une archive JAR, exécutable par une JVM, pour le déploiement.

1.1.4. Spring Boot

Spring Boot est un projet qui a été créé par Pivotal, les développeurs du framework Spring, pour permettre aux développeurs de produire plus facilement, et donc plus rapidement, des applications Spring.

Spring est un framework dit léger, car il est composé de multiples modules, qui sont, à l'exception de quelques modules clés, utilisables séparément. Un des principaux avantages est l'injection de dépendances. Un des inconvénients majeurs est l'obligation de rédiger des fichiers XML de configuration pour permettre à Spring de fonctionner correctement. Spring Boot applique le principe de Convention Over Configuration. Il simplifie l'utilisation de Spring, en pré configurant automatiquement celui-ci, permettant aux développeurs de se concentrer sur la couche métier de leur application.

Spring Boot gère entre autres les injections de dépendances, facilite le déploiement de l'application sur un serveur, et permet d'éviter la rédaction de fichiers XML, du moment que l'on respecte certaines consignes.

1.1.5. Spring Initializr

Spring Initializr est un outil pour générer rapidement les fondations d'un nouveau projet Spring Boot. Via un site web (<https://start.spring.io>), en choisissant un outil de gestion de dépendances (Maven dans notre cas), nous pouvons décrire les dépendances dont nous avons besoin dans notre projet, pour qu'il génère un projet avec toute la structure conventionnelle d'un projet Spring Boot pour commencer le développement sur de bonnes fondations.

Il génère, entre autres, un fichier `application.properties`, qui nous servira notamment à décrire l'accès à notre base de données, à préciser à Hibernate certains comportements, et à configurer notre logger.

1.1.6. JPA/Hibernate

JPA (Java Persistence API)/**Hibernate** permet de sauvegarder rapidement un objet Java dans une base de données. Il permet de s'affranchir d'un langage de requête, le plus souvent difficile à maintenir dans le temps.

La JPA est une interface définissant un certain nombre de mots-clés et de normes à respecter. Elle repose essentiellement sur l'utilisation des annotations, introduites dans Java 5. Ces annotations permettent de définir facilement des objets métier, qui pourront servir d'interface entre la base de données et l'application.

Hibernate est une des implémentations les plus abouties du standard **JPA**.

1.2. Front end

Au cours de ma formation, j'ai pu manipuler Angular, mais j'ai préféré m'orienter vers le framework Thymeleaf, car c'est celui utilisé par ma future équipe et j'ai souhaité monter en compétence sur cette technologie..

1.2.1. Thymeleaf

Thymeleaf est écrit en Java. C'est un moteur de templates pouvant générer du XML/XHTML/HTML5. Son but principal est d'être utilisé dans un environnement web pour la génération de vue permettant l'affichage d'information ou de texte produit par votre application.

Il est plus adapté pour la génération de XHTML/HTML5 dans les applications web, mais peut gérer des fichiers XML, que ce soit pour des applications web ou standalone.

L'objectif principal de Thymeleaf est de proposer une méthode élégante pour créer des templates. Pour cela, il est basé sur des balises XML et des attributs qui définissent l'exécution d'une logique prédéfinie dans le DOM (Document Object Model), au lieu d'écrire explicitement ces logiques via du code inséré directement dans le template.

Son architecture permet d'afficher rapidement les templates, en se reposant sur un principe de cache qui parcourt les fichiers afin de limiter au maximum les appels au serveur lors de l'exécution.

Enfin, Thymeleaf a été conçu dès le départ avec les standards du web et XML en tête, permettant de créer des templates de validation si besoin.

1.2.2. Bootstrap

Bootstrap est un framework CSS, mais pas seulement, puisqu'il embarque également des composants HTML et JavaScript. Il comporte un système de grille simple et efficace pour mettre en ordre l'aspect visuel d'une page web. Il apporte du style pour les boutons, les formulaires, la navigation... Il permet ainsi de concevoir un site web rapidement et avec peu de lignes de code ajoutées.

1.2.3. Postman/Google Chrome

Postman est un client servant à appeler/tester une API Web. Ils disposent de nombreuses fonctionnalités et permet notamment de construire, d'exécuter et de stocker des requêtes HTTP.

Google Chrome est un navigateur web. Il permet de vérifier le rendu final de l'interface utilisateur.

Diagramme des classes (MOO)

(Modèle Orienté Objet)

Le diagramme de classes est considéré comme le plus important de la modélisation orientée objet, il est le seul obligatoire lors d'une telle modélisation. Il permet de fournir une représentation abstraite des objets du système qui vont interagir pour réaliser les cas d'utilisation. Il s'agit d'une vue statique, car on ne tient pas compte du facteur temporel dans le comportement du système. Il permet de modéliser les classes du système et leurs relations indépendamment d'un langage de programmation particulier.

Grâce à lui, j'ai une vision claire de mes futures classes, déjà décrites, pour faciliter la compréhension de l'application, et sa réalisation.

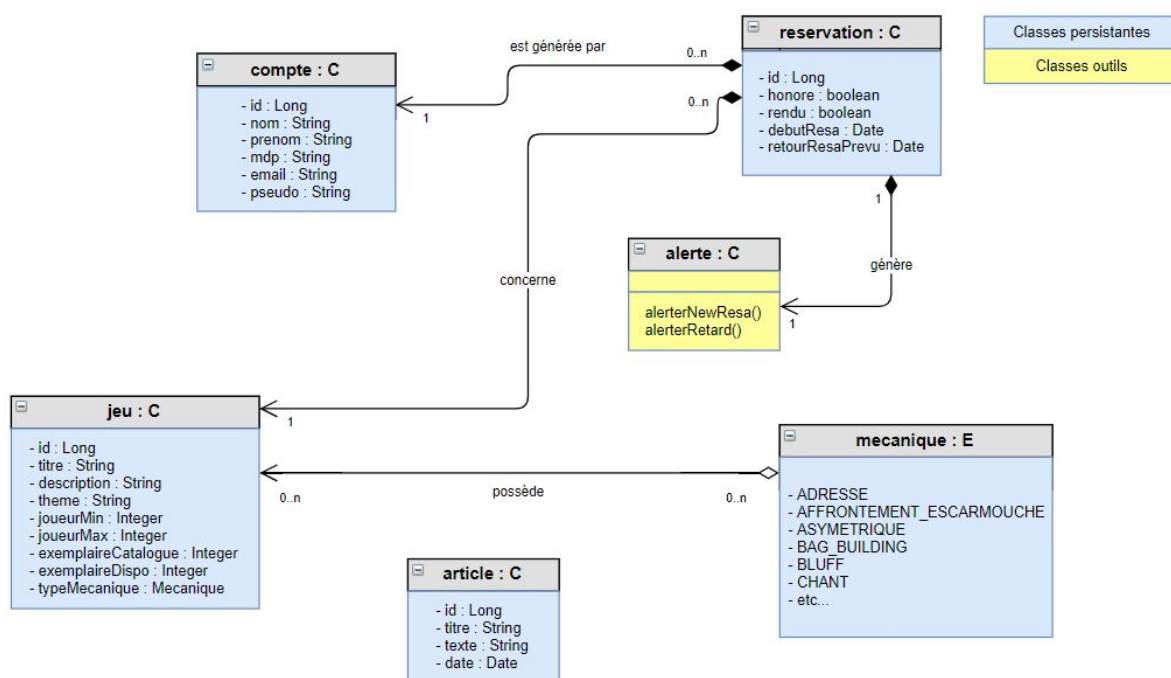
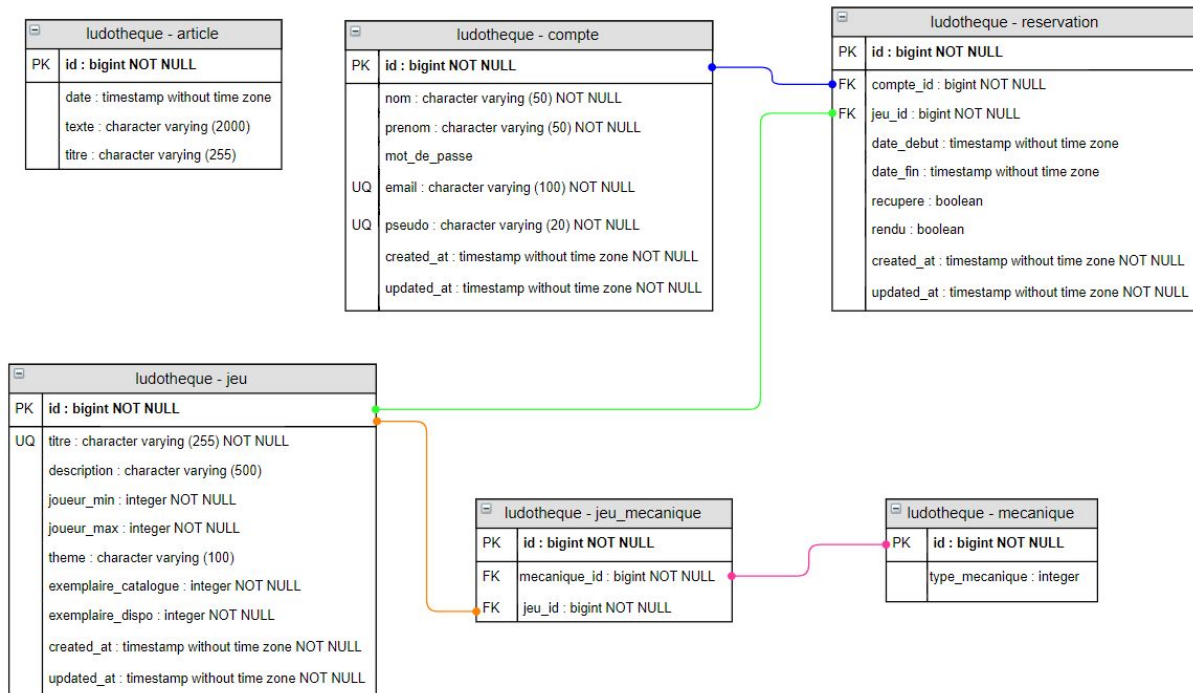


Diagramme de la base de données (MPD)

(Modèle Physique de Données)

Ce diagramme est une “traduction” du diagramme des classes permettant de décrire les entités et leurs relations pour que le SGBD puisse l'interpréter.

J'ai relié mes tables en fonction de leurs relations, par le biais de clés étrangères (Foreign Keys). J'ai choisi également le type de données de chaque colonne (Character varying, integer, timestamp without time zone, etc...) pour qu'ils correspondent aux données attendues.



RÉALISATION

Avant de commencer, je me suis posé la question pour décider de bonnes pratiques à employer sur l'ensemble du projet. En effet, pour avoir une homogénéité au sein du projet, j'ai décidé d'utiliser le camelCase comme règle de nommage, et de respecter l'auto-indentation des éditeurs de texte utilisés.

Pour mon application, j'ai utilisé une architecture MVC. MVC (Modèle-Vue-Contrôleur) est un motif d'architecture logicielle très populaire pour les applications web. Le motif est composé de trois types de modules ayant trois responsabilités différentes : les Modèles, les Vues et les Contrôleurs.

- Le Modèle est l'élément qui contient les données ainsi que de la logique en rapport avec les données. Il représente l'univers dans lequel s'inscrit l'application. Il contient toutes les entités de notre application, tous les services qui traitent les données, ainsi que les interfaces d'accès à la base de données. Le modèle est indépendant de la Vue et du Contrôleur et ne s'en sert pas.
- La Vue est la partie visible de l'application : l'interface utilisateur. Elle contient des éléments visuels ainsi que la logique nécessaire pour afficher les données provenant de l'application. Elle est indépendante du reste, et ne communique avec les Contrôleurs que lorsqu'elle en a besoin.
- Le Contrôleur est le module qui traite les requêtes de la Vue, fait appel à la partie Modèle pour un éventuel traitement, et retourne la réponse à la Vue. C'est la liaison entre les différents modules.

En résumé, lorsqu'un client envoie une requête à l'application :

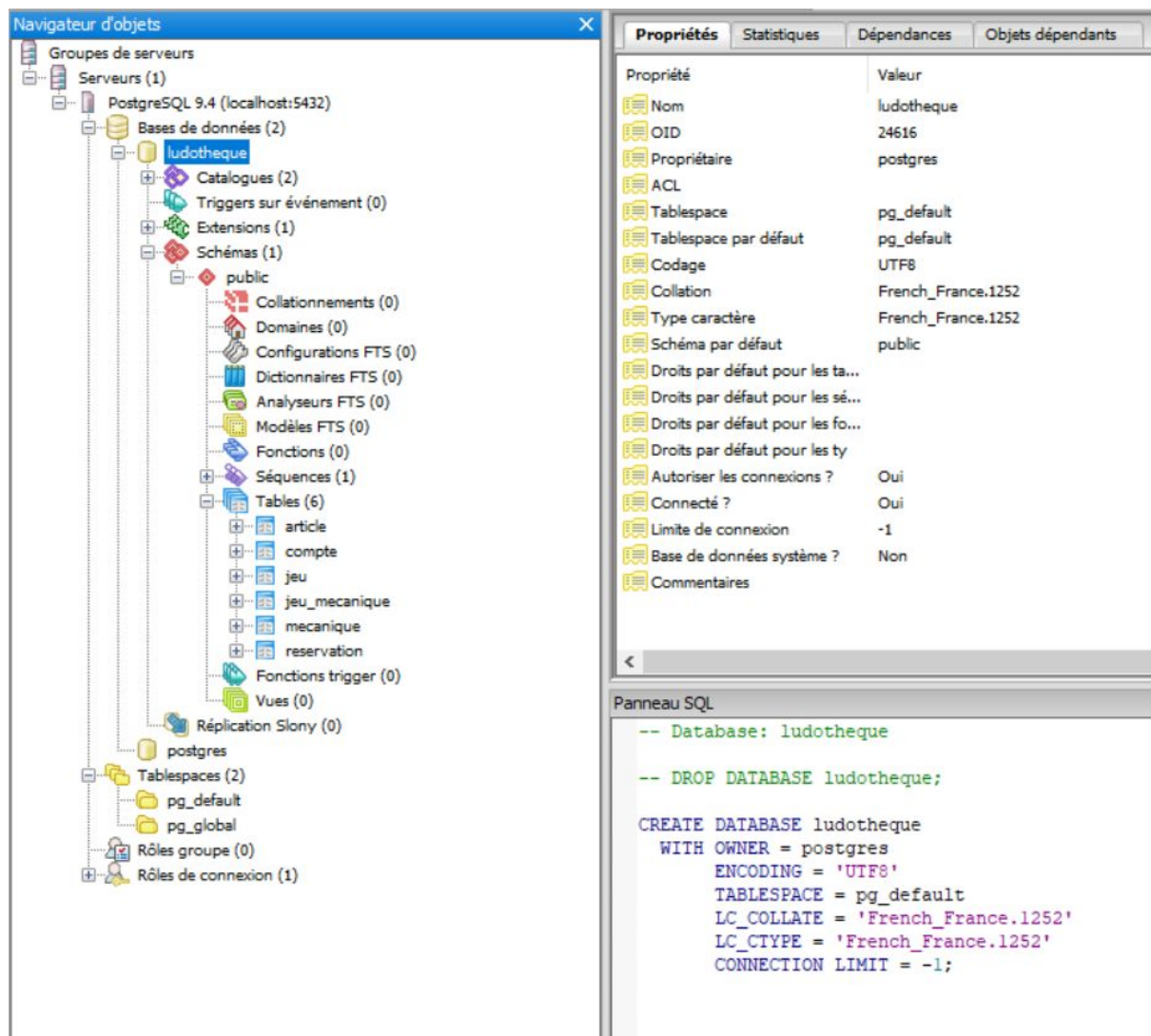
- La requête envoyée depuis la Vue est analysée par le Contrôleur (par exemple un clic de souris pour lancer une récupération de tous les jeux)
- Le Contrôleur demande au Modèle approprié d'effectuer les traitements et renvoie une réponse à la Vue avec, par exemple, une représentation de la liste des jeux sous format JSON.
- La Vue met en forme les données qu'elle a reçues en réponse pour permettre à l'utilisateur de les visionner.

Cette architecture nous permet d'organiser notre application pour diviser les Responsabilités, ce qui facilite la lisibilité, ainsi que la maintenabilité.

Mise en place du back end

1.1. La base de données

En utilisant pgAdmin III, j'ai mis en place un serveur PostGreSQL. Les entités sont créées via JPA/Hibernate.

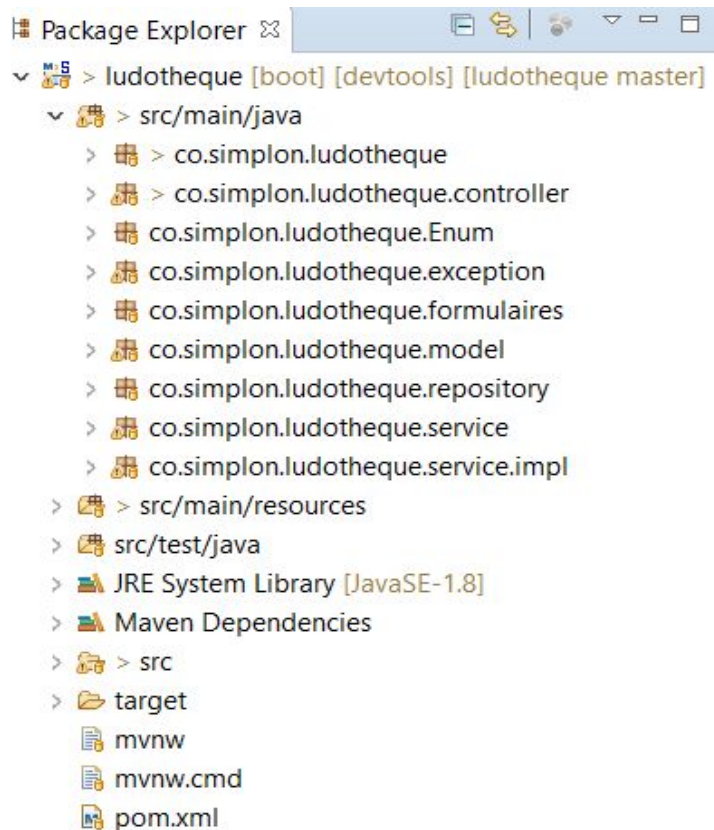


Pour relier l'application à la base de données, j'ai utilisé le fichier ***application.properties***.

```
1# Details de la base de données
2spring.datasource.url = jdbc:postgresql://localhost:5432/ludothèque
3spring.datasource.username = postgres
4spring.datasource.password = admin
5
6# Hibernate
7
8# The SQL dialect makes Hibernate generate better SQL for the chosen database
9spring.jpa.properties.hibernate.dialect = org.hibernate.dialect.PostgreSQLDialect
10
11# Hibernate ddl auto (create, create-drop, validate, update)
12spring.jpa.hibernate.ddl-auto = update
13
14logging.level.org.hibernate.SQL=DEBUG
15logging.level.org.hibernate.type=TRACE
16
17spring.thymeleaf.cache = false
```

1.2. Le codage

J'ai d'abord créé l'architecture du projet, dans *Eclipse*, en accord avec l'architecture ***MVC***.



Voilà comment j'ai procédé pour la réalisation du back end :

1.2.1. Le POJO (Plain Old Java Object)

Toutes les entités s'inspirent du concept du POJO. Cette classe sera utilisée pour stocker les données côté serveur afin de les traiter.

```
@Entity
@Table(name="jeu")
public class Jeu extends AuditModel {

    @Id
    @GeneratedValue
    private Long id;

    @Column(unique = true, nullable = false)
    @NotBlank
    private String titre;
    @Size(max=500)
    private String description;
    @Size(max=100)
    private String theme;
    @Column(name="joueur_min")
    private Integer joueurMin;
    @Column(name="joueur_max")
    private Integer joueurMax;
    @Column( name="exemplaire_catalogue", nullable=false)
    private Integer exemplaireCatalogue;
    @Column( name="exemplaire_dispo", nullable=false)
    private Integer exemplaireDispo;
```


Une classe très simple, avec les annotations nécessaire à JPA pour faire la liaison avec la base de données, ainsi que des annotations servant à la validation des champs :

@Entity

Cette annotation précise à JPA que la classe qui suit correspond à une table de notre base de données.

@NotBlank

Cette annotation permet de valider les champs correspondants dès l'instanciation de l'objet. Si les paramètres d'entrée d'un service ne sont pas validés, une exception est directement retournée.

@CreatedDate, @LastModifiedDate, @EntityListeners (AuditingEntityListener.class), @EnableJpaAuditing

Invisible dans la classe, mais implémentées grâce à l'héritage de la class AuditModel (cf. extends AuditModel). En utilisant les annotations @CreatedDate et @LastModifiedDate, on peut définir automatiquement la date de création d'un objet et la date de dernière modification de l'objet. En ajoutant l'annotation @EntityListeners (AuditingEntityListener.class) à la classe de l'entité et en ajoutant @EnableJpaAuditing dans la classe Main, les changements sont surveillés et les date sont créées.

1.2.2. Le Repository

Ensuite, je crée les interfaces (Repository) qui auront la responsabilité exclusive d'accéder à la base de données.

```
package co.simplon.ludotheque.repository;

import org.springframework.data.jpa.repository.JpaRepository;

@Repository
public interface JeuRepository extends JpaRepository<Jeu, Long>, PagingAndSortingRepository<Jeu, Long> {
}
```

L'annotation `@Repository` indique à Spring que l'interface qui suit est un repository. Cette interface hérite du `JpaRepository`. `JpaRepository` dispose déjà de toutes les méthodes de base pour gérer des données. On pourrait, néanmoins, ajouter des méthodes spécifiques avec l'annotation `@Query` qui permet de créer des requêtes HQL (Hibernate Query Language) qui est un langage de requête extrêmement puissant qui ressemble (et c'est voulu) au SQL. HQL est totalement orienté objet, cernant des notions comme l'héritage, le polymorphisme et les associations. JPA l'interprètera automatiquement, afin générer de lui-même la requête SQL voulue.

1.2.3. Le Controller

Les contrôleurs interceptent toutes les requêtes concernant nos entités et redirigent, en fonction des paramètres, vers le traitement adéquat.

```
@Controller
public class JeuController {

    @Autowired
    private JeuService jeuService;

    @GetMapping("/catalogue")
    public String getJeuxPage (Model model) {
        model.addAttribute("jeuxFirstPage", jeuService.findAll(PageRequest.of(0,20)));
        return "catalogue";
    }

    @RequestMapping(value={"/gestionCatalogue","/gestionCatalogue/modif/{id}"}, method = RequestMethod.GET)
    public String modifJeuForm(Model model, @PathVariable(required = false, name = "id") Long id) {
        if (null != id) {
            model.addAttribute("jeu", jeuService.findById(id));
        } else {
            model.addAttribute("jeu", new Jeu());
        }
        return "gestionCatalogue";
    }
}
```

Les annotations :

- *@Controller*

Cette annotation est utilisée avec le patron de conception MVC de Spring (le composant utilisé par Spring afin d'implémenter une application web). L'annotation @Controller indique qu'une classe particulière a le rôle de contrôleur. Le dispatcher scanne les classes annotées à la recherche de méthodes "mappées" afin de détecter les annotations @RequestMapping.

- *@Autowired*

L'annotation @Autowired demande à Spring de trouver et instancier la classe fournie en argument de notre constructeur, nous permettant d'utiliser les méthodes de la classe annotée.

- *@RequestMapping*

Cette annotation précise l'URI dont le contrôleur a la responsabilité.

- *@PathVariable*

Cette annotation représente le type d'envoi de données que le contrôleur peut recevoir. *@PathVariable* indique que la donnée est directement dans l'URI.

- *@GetMapping*

Il s'agit d'une version spécialisée de l'annotation *@RequestMapping* qui agit comme un raccourci de *@RequestMapping(method = RequestMethod.GET)*. Les méthodes annotées *@GetMapping* prennent en charge les requêtes HTTP GET requests correspondant à l'URI mentionnée.

1.2.4. Le Service

Le service rassemble toutes les méthodes nécessaires au traitement des données.

```
@Service
public interface JeuService extends JeuRepository {

}
```

- *@Service*

Cette annotation indique à Spring Boot qu'il doit traiter l'interface qui suit comme un Service.

Ceci indique explicitement le rôle de l'interface pour une meilleure compréhension.

Ci-dessus l'initialisation de l'interface qui va gérer les méthodes de l'interface JeuRepository qui elle-même hérite des méthodes de JpaRepository.

Mise en place du Front end

J'ai décidé de développer mon interface utilisateur en Thymeleaf, car c'est la technologie utilisée par ma future équipe et j'ai souhaité monter en compétence dessus.

- **Thymeleaf**

A l'heure où la tendance est aux SPA (Single Page Application) dans le développement front, il est important de ne pas oublier les autres solutions existantes. La structure en plusieurs pages Html est toujours utilisée surtout pour les petites applications et de nombreux sites existant utilise ce format.

A ce moment là, sur une application Java, utiliser Angular, React ou l'une des autres solutions de SPA peut être assez contraignant et risque d'augmenter considérablement le temps de développement. Thymeleaf et son intégration avec Spring MVC est alors la solution parfaite pour allier simplicité, facilité et rapidité ! Thymeleaf ayant été développé pour s'associer idéalement avec Spring MVC. Thymeleaf a également le mérite d'être peu intrusif, car il suffit d'ajouter le dialecte spécifique à Thymeleaf directement dans les balises Html.

Thymeleaf permet de créer ses propres templates et même son propre dialecte, ce qui offre une grande liberté de création.

- **Bootstrap**

Bootstrap est un framework destiné à faciliter la création d'applications Web. Il regroupe une collection d'outils fournis sous la forme de classes CSS et de bibliothèques JavaScript.

- **Présentation du template layout.html**

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head th:fragment="Ludo-head">
  <!-- Compiled CSS -->
  <link rel="stylesheet" href="../css/bootstrap.css" th:href="@{/css/bootstrap.css}">
  <link rel="stylesheet" href="../css/style.css" th:href="@{/css/style.css}">

  <!-- Let browser know website is optimized for mobile -->
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta th:remove="tag">
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
</head>
<body>
  <header th:fragment="Ludo-header">
    <a href="#" th:href="@{/}">
      
    </a>
    <a href="#" th:href="@{/}">Accueil</a>
    <a href="#" th:href="@{/profil}">Profil</a>
    <a href="#" th:href="@{/gestionCompte}">Gestion des comptes</a>
    <a href="#" th:href="@{/catalogue}">Catalogue</a>
    <a href="#" th:href="@{/gestionCatalogue}">Gestion du catalogue</a>
    <a href="#" th:href="@{/aide}">Contact / F.A.Q</a>
    <a href="#" th:href="@{/users/Login}">Se connecter</a>
    <div id="#"> <span>Bonjour, <b>(utilisateur)</b></span>
      <form method="post" th:action="@{/users/Logout}">
        <input type="submit" value="Logout">
      </form>
    </div>
  </header>

  <footer th:fragment="Ludo-footer">
    @author MOI
  </footer>
```

On notera dans la balise `<html>` la présence du dialecte `xmlns:th="http://www.thymeleaf.org"` qui indique au programme que la page est une page comportant du dialecte Thymeleaf. Ainsi, lorsqu'il rencontrera des `"th:"`, il faudra les interpréter et informer l'EDI que la balise est valide bien que différente d'une balise Html classique.

Ce template permet de définir le header et le footer qui seront appelés dans chacune des pages à venir. Cela se remarque au fait que dans les balises, on utilise le dialecte `th:fragment="valeur"`, indiquant au programme que ce sont des composants qui pourront être importés dans les autres pages.

On note également que les href sont doublés (dont un précédé du fameux th:), cela permet à la page de s'afficher, même si le dialecte Thymeleaf n'a pas obtenu de retour de la part du serveur. C'est pour cela que Thymeleaf est considéré comme un template naturel. Toutefois, il s'agira d'une page statique, une sorte de prototype.

- **Présentation du template catalogue.html**

Représente la partie HTML (Template) gérant l’affichage du catalogue de jeux

```
<! DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">

<head th:include="layout :: ludo-head" />
<meta charset="utf-8">
<title>Catalogue</title>

<body>

    <header th:include="layout::Ludo-header" />

    <h1>Liste de jeux</h1>

    <div>
        <table class="table table-bordered">
            <thead>
                <tr>
                    <th scope="col">Titre</th>
                    <th scope="col">Description</th>
                    <th scope="col">Thème</th>
                    <th scope="col">Nombre de joueur min</th>
                    <th scope="col">Nombre de joueur max</th>
                    <th scope="col">Exemplaire catalogue</th>
                    <th scope="col">Exemplaire disponible</th>
                </tr>
            </thead>
            <tbody>
                <tr th:each="jeu : ${jeuxFirstPage}">
                    <td th:utext="${jeu.titre}">...</td>
                    <td th:utext="${jeu.description}">...</td>
                    <td th:utext="${jeu.theme}">...</td>
                    <td th:utext="${jeu.joueurMin}">...</td>
                    <td th:utext="${jeu.joueurMax}">...</td>
                    <td th:utext="${jeu.exemplaireCatalogue}">...</td>
                    <td th:utext="${jeu.exemplaireDispo}">...</td>
                    <td>
                        <a th:href="@{/gestionCatalogue/modif/__$${jeu.id}__}">Modifier</a>
                        <a th:href="@{/supprimeJeu/__$${jeu.id}__}">Supprimer</a>
                    </td>
                </tr>
            </tbody>
        </table>
    </div>

    <footer th:include="layout::Ludo-footer" />

</body>

</html>
```

Comme annoncé plus tôt, on retrouve l’appel au template “layout” permettant d’afficher les fragments header et footer via les dialectes : `th:include="layout::ludo-header"` et `th:include="layout::ludo-footer"`.

On constate que le dialecte Thymeleaf permet d'itérer, directement, sur des objets java grâce au `th:each="jeu : ${jeuxFirstPage}"` qui demande à la variable `jeuFirstPage` de récupérer tous les attributs de chaque entrée dans la table `jeu`.

Et ensuite, on demande d'afficher les attributs spécifiés de l'objet `jeu`, via les `th:utext="${jeu.attribut}"`. Le `${...}` étant appelé une expression de variable.

En dessous, on remarque la forme `@{...}` qui est une expression de lien, qui indique le chemin à suivre en cas de clic.

- **Présentation du template `gestionCatalogue.html`**

Représente la partie HTML (Template) gérant la création/modification de jeu dans le catalogue

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">

<head th:include="layout :: ludo-head" />
<meta charset="utf-8">
<title>Gestion catalogue</title>

<body>

    <header th:include="layout::ludo-header" />

    <h1>Ajouter un jeu au catalogue</h1>

    <form id="modifJeuForm" action="#" th:action="@{/gestionCatalogue}"
        th:object="${jeu}" method="post">
        <table>
            <tr>
                <td><input id="idJeu" type="hidden" th:field="*{id}" /></td>
            </tr>
            <tr>
                <td>Titre :</td>
                <td><input id="titreJeu" type="text" th:field="*{titre}" /></td>
            </tr>
            <tr>
                <td>Description :</td>
                <td><input type="text" th:field="*{description}" /></td>
            </tr>
        </table>
    </form>
```

Ici, nous trouvons `th:action="@{/gestionCatalogue}"` qui indique l'URI qui traitera les informations saisies dans le formulaire.

Ensuite, nous avons `th:object="${jeu}"` qui informe Thymeleaf où trouver l'objet auquel il va lier les paramètres de réponse.

Enfin, nous avons le `th:field="*{attribut}"` qui définit l'attribut attendu dans le champ de saisie, la forme `*{...}` est une expression de sélection, qui sera exécutée en lien avec un objet défini précédemment, ici `th:object="${jeu}"`.

ÉVOLUTION À VENIR

Pour la suite du développement, j'envisageais d'ajouter les fonctionnalités suivantes :

- Possibilité de commenter les articles pour donner un aspect plus communautaire au site.
- Dans la même idée, donner la possibilité de noter et donner son avis sur les jeux.
- Proposer des suggestions aux membres en fonction de leur historique de location.
- Permettre au ludothécaire de créer des événements auxquels les membres pourraient s'inscrire en ligne via leur compte.
- Permettre d'adhérer à la ludothèque et régler son abonnement en ligne.
- Étendre le catalogue aux jeux vidéo et aux jouets.

CONCLUSION

Grâce à ce projet, j'ai pu voir toutes les étapes du développement d'une application. En effet, lors de mes sessions en entreprise, je n'ai pas eu le choix sur la conception, ou les technologies à utiliser. Ici, j'ai pu réaliser intégralement toutes les étapes.

Cela m'a également permis de mettre en œuvre mes nouvelles compétences. J'ai réalisé l'importance d'une bonne conception, étape indispensable au bon développement d'une application, car elle permet une bonne compréhension du cahier des charges, et facilite énormément la partie réalisation.

Même si le projet est individuel, l'émulation du groupe m'a permis d'échanger sur des idées et de partager compétences et bonnes pratiques afin d'atteindre les objectifs.

J'ai dû apprendre à planifier, à trancher (choix technologique ou technique) et à arrêter des décisions dans le but d'avancer et de fournir un livrable dans les temps, J'ai eu un aperçu de la complexité du développement d'une application.

A l'évidence, j'ai encore énormément de choses à apprendre et à améliorer, mais malgré les difficultés rencontrées, j'ai réussi à mener à bien ce projet.

REMERCIEMENTS

Je remercie tous les membres de la promotion La Poste pour leur partage de connaissances, leur disponibilité, leur bonne humeur et l'émulation née de ces rencontres.

Je tiens à remercier la fabrique SIMPLON, de m'avoir intégré au monde du numérique, ainsi que toute l'équipe pédagogique, nos formateurs, Sébastien MARTIN et Emmanuel LE PEVEDIC et tous les intervenants, pour leur soutien, leur encadrement et leurs directives tout au long de la formation.

Je suis également reconnaissant envers La Poste qui a rendu possible ma reconversion dans le développement logiciel. Et par extension, mon équipe chez Docapost pour leur accueil chaleureux malgré des délais de livraison très rapprochés.

Merci aux membres du jury de prendre le temps de venir évaluer notre travail.

Et enfin, un grand merci à mon épouse pour son infini patience lors de mes crises de doute et son soutien afin de me maintenir dans la bonne direction.

LEXIQUE

nom	définition	commentaire
ADRESSE IP	Internet Protocol	numéro qui identifie chaque ordinateur sur Internet, et plus généralement, l'interface avec le réseau de tout matériel informatique (routeur, imprimante) connecté à un réseau informatique.
ALGORITHME		Un algorithme est un ensemble d'activités logiques à l'accomplissement d'une tâche précise.
API	Application Programming Interface	permet d'établir des connexions entre plusieurs logiciels pour échanger des données. Plus techniquement, il s'agit d'un ensemble de fonctions qui vont permettre à un développeur d'utiliser simplement une application dans son programme.
BACK-END	Il s'agit de la partie arrière d'un site web	Il s'agit de la partie arrière d'un site web, la couche la plus éloignée de l'utilisateur final. Le back-end se réfère à l'administration du site.
BASE DE DONNEES	ensemble structuré et organisé d'informations	Les informations sont placées dans des fichiers, et organisées de manière à pouvoir être facilement triées, classées et modifiées par le biais d'un logiciel spécialisé appelé système de gestion de base de données (SGBD).
BATCH	traitement par lots	Enchaînement automatique d'une suite de commandes (processus) sur un ordinateur sans intervention d'un opérateur. Une fois que ce processus est terminé (quel que soit le résultat), l'ordinateur traite le lot suivant.
CLASSE	modèle de définition	Une classe est un modèle de définition pour des objets ayant le même ensemble d'attributs, et le même ensemble de méthodes. A partir d'une classe on peut créer un ou plusieurs objets par instanciation ; chaque objet est une instance d'une seule classe.
CLASSE ABSTRAITE	Concept POO	En programmation orientée objet (POO), une classe abstraite est une classe dont l'implémentation n'est pas complète et qui ne peut être instanciée . Elle sert de base à d'autres classes dérivées (héritées).
CSS	Cascading Style Sheets	Mise en forme de page HTML
EXTENDS	Concept POO	Permet d'étendre une classe ou une interface. Il ne peut y avoir qu'un seul extends pour une classe. (Héritage simple).

FRAMEWORK	Socle d'application	Un ensemble cohérent de composants logiciels structurels, qui sert à créer les fondations ainsi que les grandes lignes de tout ou d'une partie d'un logiciel (architecture). Un framework se distingue d'une simple bibliothèque logicielle principalement par son caractère générique non-spécialisé.
FRONT-END	Désigne la partie visible	L'objectif du front-end est de rendre le site internet le plus ergonomique possible et de rendre plus accessible la partie de navigation sur le site. Contribue à une bonne UX .
HERITAGE	Concept POO	L'héritage, est l'un des mécanismes les plus puissants de la programmation orientée objet, permet de reprendre des membres d'une classe (appelée superclass ou classe mère) dans une autre classe (appelée sous-classe, classe fille ou encore classe dérivée), qui en hérite.
HTML	Hypertext Markup Language	Structure de page HTML
IDE	Environnement de Développement	Est un logiciel qui rassemble des outils permettant de développer d'autres logiciels tels que des applications mobiles, des logiciels pour ordinateur ou consoles de jeux, des sites web, etc. ainsi que de réaliser des librairies ou des frameworks, c'est-à-dire des morceaux de code qui pourront être sauvegardés et réutilisés dans d'autres programmes.
IMPLEMENTS	Concept POO	Permet de spécifier quelle interface on souhaite utiliser pour une classe. On peut implémenter plusieurs interfaces dans la même classe.
JAVA	Langage programmation orienté objet	Créé par James Gosling et Patrick Naughton, employés de Sun Microsystems. La particularité et l'objectif central de Java est que les logiciels écrits dans ce langage doivent être très facilement portables sur plusieurs systèmes d'exploitation tels que Unix, Windows, Mac OS ou GNU/Linux, avec peu ou pas de modifications. Pour cela, divers plateformes et frameworks associés visent à guider, sinon garantir, cette portabilité des applications développées en Java.
JDK	Java Development Kit	Désigne un ensemble de bibliothèques logicielles de base du langage de programmation Java, ainsi que les outils avec lesquels le code Java peut être compilé, transformé en bytecode destiné à la machine virtuelle Java .
JRE	Java Runtime Environment	Environnement permettant l'exécution de programmes Java lors de leurs diffusions.
JVM	Java Virtual Machine	c'est la Java Virtual Machine, donc la machine virtuelle de Java. Elle est contenu dans la JRE, et c'est plus particulièrement elle qui fait fonctionner les applis java.

LIBRAIRIE	bibliothèque logicielle	Une bibliothèque logicielle est une collection de routines, qui peuvent être déjà compilées et prêtes à être utilisées par des programmes. Elles sont utilisées pour réaliser des interfaces de programmation, des frameworks , des plugins ainsi que des langages de programmation. Les routines contenues dans les bibliothèques sont typiquement en rapport avec des opérations fréquentes en programmation : manipulation des interfaces utilisateur , manipulation des bases de données ou les calculs mathématiques.
MOCKUP	maquette d'une interface utilisateur aboutie	Un mockup est un terme informatique, et même un terme de design, qui désigne une maquette d'une interface utilisateur. ... Attention : selon les personnes, le terme peut être employé pour désigner un wireframe : en fait, il s'agit d'un WIREFRAME amélioré faisant apparaître des liens fonctionnels motif d'architecture logicielle destiné aux interfaces graphiques.
MVC	Pattern: Modèle-vue-contrôleur	Un modèle (Model) contient les données à afficher. Une vue (View) contient la présentation de l'interface graphique. Un contrôleur (Controller) contient la logique concernant les actions effectuées par l'utilisateur.
PATTERN	Modèle de conception	
POLYMORPHISME	Concept POO	Action de modifier " override " les méthodes définies par une interface .
PROTOCOLE		Règles et conventions régissant l'échange de données entre ordinateurs.
REGEX	Expressions régulières	Permet de définir des patterns lors des validations de données ex: Format tel, mail, ...
SERVEUR		Un serveur web est un ordinateur connecté à Internet qui héberge des données et fichiers afin de répondre aux requêtes provenant des navigateurs des internautes.
SGBD	système de gestion de base de données	Logiciel système destiné à stocker et à partager des informations dans une base de données, en garantissant la qualité, la pérennité et la confidentialité des informations, tout en cachant la complexité des opérations
Site dynamique	Site dont les pages sont générées par le serveur	

Site statique	Un site statique est un site dont le contenu n'est modifiable que par le webmaster.	
SSL	Secure Socket Layer	Protocole d'accord de sécurisation.
UI	Interface Utilisateur	L'UI peut se résumer à l'organisation des éléments graphiques et textuels pour proposer une application agréable et attrayante.
URL	Uniform Ressource Locator	L'URL sert à désigner une adresse web. A la différence du « nom de domaine » qui n'est qu'une composante de l'adresse du site web, l'URL se compose des éléments suivants : l'indication du protocole de communication, le plus souvent http:// pour les serveurs web + le sous-domaine cible, par exemple : www. + le nom de domaine et le chemin de la ressource, par exemple : /page01
variables d'environnement	Variables dynamiques utilisées par les différents processus d'un système d'exploitation	Elles servent à communiquer des informations entre programmes qui ne se trouvent pas sur la même ligne hiérarchique, et ont donc besoin d'une convention pour se communiquer mutuellement leurs choix.
WIREFRAME	Amélioration du ZONING Maquette interface utilisateur	Un wireframe dans le domaine du webdesign représente le schéma d'une page web ou d'un site web. Ce schéma est utilisé lors des étapes préliminaires à la conception d'une interface web. Il permet de faciliter la communication sur le début du design.
WORDPRESS		Logiciel principalement connu pour être un outil de création de blog, mais en réalité c'est un système de gestion de contenu CMS (content management system) pour mettre tout le contenu des sites Internet à jour.
XML	eXtensible Markup Language	Désigne un langage informatique qui permet d'enregistrer des données textuelles. Contrairement au langage HTML qui présente un nombre fini de balises, le XML donne la possibilité de créer de nouvelles balises à volonté. Ce langage permet également de séparer le contenu (les données) du contenant (la présentation des données) facilitant ainsi l'élaboration du site.
SQL	Structured Query Language	Llangage informatique normalisé servant à exploiter des bases de données relationnelles.

NOSQL

Not Only SQL

NoSQL est une approche de la conception des bases et de leur administration particulièrement utile pour de très grands ensembles de données distribuées.

BIG DATA

Méga Données

Désigne des ensembles de données devenus si volumineux, qu'ils dépassent l'intuition et les capacités humaines d'analyse et même celles des outils informatiques classiques de gestion de base de données ou de l'information.