Rapport n°2 **Groupe 6** **05/03/2018**

Projet TER

Mathilde Pechdimaldjian

Brian Lenormand

Corentin BOuchaudon

TABLE DES MATIERES

INTRODUCTION (MATHILDE)

Dans le cadre de notre Master M1 MIAGE, nous avions à réaliser un site web. Notre client fictif souhaitait développer un site pour sa société Netflox, il nous a donc fourni un cahier des charges avec les détails des fonctionnalités attendues, codées avec Java J2EE et en utilisant un pattern MVC.

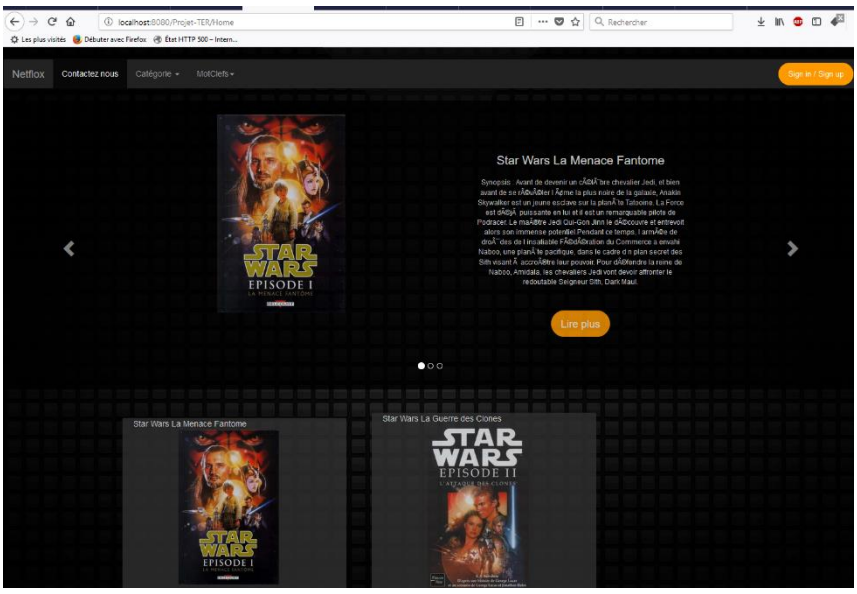
Le but de ce site est d'accéder à une base de données de vidéos visionnables en streaming ou en téléchargement ainsi qu'une gestion des comptes clients avec quatre types : non inscrit, inscrit, premium, administrateur.

Dans ce document nous allons vous faire une présentation rapide du site, pour vous présenter les fonctionnalités développées et comment y accéder.

Ce document est là pour illustrer la réalisation du projet, comment nous l'avons géré, les difficultés rencontrées et nos choix d'implémentation.

PRESENTATION DU PROJET (BRIAN)

PAGE D'ACCUEIL



L'accès au site se fait dans un premier temps par /Home, cette page est accessible par tous sur le site via le lien netflox à gauche de la barre de navigation.

De là, vous avez plusieurs choix :

accéder à la liste de films qui sont accessibles sur la plateforme
accéder au carrousel, qui affiche les dernières sorties de films sur le site

vous connecter, ou vous inscrire sur le site, ce qui vous permettra d'accéder à d'autres fonctionnalités

CONNEXION/INSCRIPTION

En appuyant sur « Connexion/Inscription », vous allez pouvoir vous connecter sur votre compte netflox, ou bien vous y inscrire. Lorsque cette inscription sera effectuée, vous aurez accès aux fonctionnalités d'achat, de location et de devenir membre premium mais également à votre compte qui regroupe toutes vos opérations bancaires sur le site. Pour cela, vous pouvez cliquez sur "Bienvenu xxx", qui est un lien vers votre compte

DETAILS VIDEO

Vous avez accès aux détails de chaque vidéo : nom du film, résumé, nombre de vues. Cela permettra d'avoir plus d'informations sur le film demandé, mais aussi la possibilité de louer ou acheter le film.

LOCATION-ACHAT

Le paiement d'une location vous permet d'avoir un accès illimité à la vidéo en streaming pendant une durée de 7 jours, c'est donc une durée déterminée. L'achat d'une vidéo vous permet de la télécharger à tout moment sans contraintes de temps.

REGARDER EN STREAMING/PREMIUM

En souscrivant à l'abonnement premium de Netflox via une page, vous aurez un accès illimité à toutes les vidéos pendant la durée de votre abonnement. Vous aurez donc accès directement aux streaming sans passer par le paiement à chaque fois.

RECHERCHE

Le moyen d'accéder à cette fonctionnalité est via la barre de navigation qui vous enverra vers une autre page avec la liste des films en fonction de vos choix.

ADMINISTRATION

Accessible uniquement pour les membres "administrateur" via la barre de navigation. il est possible de supprimer, ajouter ou modifier un client ou un film mais également de voir l'évolution du site et d'obtenir son audit.

CHOIX D'IMPLEMENTATION / PROBLEME(S) RENCONTRE(S)

MODELE (CORENTIN)

Dans cette partie du projet il y avait trois sous parties majeures, la classe DataBase.java, le script d'insertion de données et les classes.java restantes.

Concernant les classes.java, il n'y avait aucune difficulté à les coder car il s'agissait simplement de créer des classes ayant pour attributs les mêmes que ceux de la Base de Données. Il ne restait ensuite plus qu'à créer les constructeurs et les accesseurs nécessaire à l'utilisation des données.

Nous avons ensuite le script d'insertion de données, c'est ici que j'ai rencontré ma première difficulté. En effet il m'a fallu plus de temps que prévu pour la réaliser. Ce retard est expliqué par des problèmes de cohérence de données, les données devant rester cohérentes quel que soit la quantité. Le retard peut aussi être expliqué par le fait que rentrer les valeurs tests a pris plus de temps que prévu (entrer 100 résumés de films/séries/documentaires est une tâche qui est assez long).

Pour finir vient la classe DataBase.java. La première difficulté a été de réussir à créer la base de données. En effet, habituellement nous utilisons les Bases de Données de la faculté (BD local). Or nous n'y avions pas accès depuis nos domiciles, c'est ainsi que nous avons décidé d'installer postgresql sur nos machines respectives et de créer une BD cbouch3_a qui a pour user cbouch3_a (avec mdp cbocuh3_a). Ce qui nous a permis d'économiser les allers-retours à la faculté pour tester les fonctions.

La seconde difficulté fut pour l'implémentation des fonctions sur la BD. Nous avions deux choix qui se présentaient à nous, le premier était d'écrire directement dans le code java les insertions ou sélections , et donc permet un accès rapide en cas de changement ou d'erreur mais cela peut facilement rendre le code illisible.

La deuxième solution est de créer des fonction psql qui seront appelées dans les contrôleurs, cette façon permet de simplifier la partie de développement de mes camarades mais ne maîtrisant pas assez la création de fonction psql j'ai préféré choisir la première solution car elle est plus simple d'utilisation pour mes camarades

Une autre difficulté résidait dans la réalisation de certaines fonctions incluant un nombre important de boucles imbriquées, ce qui peut être perturbant et demande du temps pour ne pas se perdre dans les valeurs (cf. recherche de MC ou suggestions).

Chaque fonction était testée indépendamment afin de vérifier le bon fonctionnement de celle-ci, ce qui demande beaucoup plus de temps. Dans le code que l'on vous rend aujourd'hui toutes les fonctions retournent les bonnes valeurs mais, elles ne sont pas optimisées (cf. fonctions énoncées ci-dessus)'

La dernière des difficultés pour ma part, était dans la génération de pdf via java. En effet, n'ayant jamais fait ce genre d'implémentation, j'ai du effectuer quelques recherches, qui m'ont permis de trouver la librairie "iTest.jar". Il m'a fallu également me l'approprier assez rapidement.

CONTROLEUR (BRIAN)

Dans notre pattern MVC, le contrôleur permet de “contrôler” les données envoyées à la vue en récupérant certaines fonctions du modèle (représenté par la Base de données). Chaque fonction de notre application web devait comporter son contrôleur, permettant ainsi de diviser chaque partie pour une meilleure gestion de ces dernières. Nous avons rencontrée plusieurs difficultés, tant sur le code java que sur la notion de contrôleur.

La fonction de téléchargement fut l’une des difficultés majeur pour cette partie car l'implémentation de celle-ci ne se contentait pas d’utiliser uniquement les méthodes transmises par le model et de les renvoyer correctement les informations à la vue , elle demandait aussi de spécifier le chemin pour retrouver un fichier mais également de gérer les flux entrants.

Le fait de ne pas travailler sur le même environnement pouvait être une difficulté pour tous car cela pouvait poser des problèmes dans les chemins des dossiers de l’application Web.

VUE (MATHILDE)

La vue est là pour utiliser les données transmises par le contrôleur, afin de les afficher correctement.

La difficulté dans cette partie, c’est de définir le rôle et fonctionnalités du contrôleur et donc d’arriver à les exprimer clairement aux autres membres de l'équipe, pour que ceux-ci puissent créer des méthodes qui renvoient des données cohérentes.

La communication des données entre les pages, et le traitement des actions pour une même vue était pour moi une source de difficultés, car je n’avais pas été aussi loin lorsque je développais en J2EE.

Suite à des problèmes personnels d’un des membres du groupe, j’ai eu la partie du contrôleur à coder. Ceci pouvait paraître handicapant pour l'avancement du projet, c’était le cas dans un premier temps, car cela a provoqué un décalage dans le planning mais une fois que nous nous sommes adaptés, je me suis rendu compte qu’en ayant une vue plus globale de ce qu'attendait le contrôleur et la vue, j’ai avancé plus vite que si nous étions chacun sur une partie.

CALENDRIER EFFECTIF VS CALENDRIER FONCTIONNEL (MATHILDE)

Fonctionnalité prévu	Date de fin prévu	Etat d'avancement*	Date Réel	Commentaires
Choix Interface	22/01/18	100%	22/01/18	
Afficher liste film	24/01/18	50%	23/02	retard pris car le développement de l'interface a pris plus de temps que prévu
Connexion/Inscription	26/01/18	30%	25/02	Brian a eu des soucis, les contrôleurs n'étaient donc pas opérationnels
Suggestion	28/01/18	25%	02/03	La fonction en bdd était plus complexe que prévu et a donc pris plus de temps à être implémenté
Téléchargement	08/02/18	0%	03/03	Le téléchargement de vidéo fonctionne, mais n'ayant pas le même système d'exploitation, il risque d'avoir des conflits de path
Location/Achat	18/02/18	10%	02/03	Le retard sur les autres fonctionnalités
Générer PDF	23/02/18	0%	04/03	Nous avons décidé de faire cette fonctionnalité le plus tard possible pour se concentrer sur d'autres

*État d'avancement représente le pourcentage de la partie faite à la date prévue

POURQUOI CE DECALAGE ET CE RETARD ?

La réalisation devait débuter en mi-janvier mais celle-ci a plutôt commencé vers début février, dû à nos impératifs scolaires.

Concernant la vue, Mathilde a décidé de coder complètement la partie qui concerne le html, en attendant que les contrôleurs soient opérationnels. Il n'y avait donc dans une premier temps que l'interface qui fut implémentée.

Par la suite, Brian a rencontré des problèmes personnels qui ne lui ont pas permis de s'investir dans le projet pendant une période. Nous avions donc un retard important qui s'est accumulé pour les contrôleurs impactant ainsi l'implémentation des vues.

Il a fallu faire une redistribution des tâches, sans que cela ralentisse plus l'avancement des autres partie

La vue et le contrôleur étant liés, Mathilde a décidé de s'occuper de ces deux parties. Le travail étant conséquent, cela explique le retard sur la majorité des fonctionnalités.

Nous avions changé de stratégie par la suite, nous avons choisi d'axer notre travail en fonction de l'autre et de ne pas coder chacun de son côté les fonctionnalités.

C'est à dire que, dans un premier temps nous réfléchissions à l'implémentation souhaitée, puis Corentin s'occupait de la coder dans la base de données et ensuite Mathilde faisait la vue et le contrôleur adéquates.

HEURE PASSEES SUR LES PARTIE(MATHILDE)

Ce tableau vous illustre, le temps que chaque membre a passé sur les différentes parties du projet avec le code couleur suivant :

Corentin	
Mathilde	
Brian	

Fonctionnalités	Vue	Contrôleur	Base de donnée
Accueil	2	3	2
inscription/connection	0.5	0.5	6
Premium	1	0.5	0.5
Administration client/film	4	1	1
Générer pdf	1	2	0

Téléchargement	0.5	3	0
Détails Vidéo	2	2	0.5
Recherche	0.5	1	0.5
Afficher Video	2	2	1
Payement	3	1	1
Compte Client	2	1	1
Testes BDD			5
Rapport	3	3	3
TOTALE	21h30	20h	21h30
Totale de Corentin	0h	2h	21h30
Totale de Mathilde	21h30	12h	0h
Totale de Brian	0h	6h	0h

CHOIX D'IMPLEMENTATION(CORENTIN)

RECHERCHER

Pour la recherche , nous avons choisi d'implémenter les films en 3 catégories (film,série et documentaire) et de leur ajouter jusqu'à 3 mots-clefs, ce qui permet donc de faire une recherche via les catégories ou les mot-clefs. La recherche en combinant n'a pas été implémentée car nous manquions de temps et nous avons préféré nous concentrer davantage sur d'autres fonctionnalités.

HOMESERVELT

Dans la page index.jsp, la connection et l'inscription se fait via un modal.

Il y avait donc à gérer les Servlet de la page d'accueil, de connection, inscription et déconnection.

Dans un premier temps, j'ai séparé ces Servlets mais dans chaque fichier il y avait une partie dans le code qui était identique donc trop de redondances inutiles.

C'est pour cela que j'ai décidé de tout concentrer en un seul servlet.

Pour cela, j'ai utilisé une des propriétés de l'annotation @WebServelt qui permet de spécifier différents paternes d'url (ici /Home , /Inscription….)

Ce qui permet à un seul servlet de traiter des requêtes attenantes à diverses url.

Une fois que le servlet a été appelé, je vérifie l'URI (seulement la dernière partie de l'URL, sans le localhost :8080…) afin de savoir quel traitement effectuer.

ADMINISTRATIONTRAITEMENT ET PAYEMENT

Pour ces deux page jsp, en fonction de l'action précédente de l'utilisateur, le traitement varie.Pour le paiement, qui utilise la même page qu'achat, louer, et devenir premium, nous avions besoin d'un moyen pour informer la page du type d'action attendue. Pour administrateur, il y a au total 6 actions possibles sur cette page (ajout, suppression client et film, et la génération de pdf audit ou du CA) qui se déclenchent avec des boutons.

Nous avons donc choisi d'utiliser l'url pour transmettre le type d'action à réaliser

SUGGESTION

Nous avons choisi d'implémenter cette fonctionnalité en prenant dans un premier temps compte de la catégorie de la vidéo. Si celle-ci possède un numéro d'épisode et de saison, elle était considérée comme une série et il y avait donc dans un premier temps une recherche des épisodes suivant. il y a ensuite l'utilisation des mots clefs, en cherchant les mots-clefs commun entre plusieurs films

TEST

Nous avons choisi de réaliser des tests avec JUnit4 pour les différentes classes du modèle. Pour cela nous avons importé la library JUnit4. Les tests sur les classes dites basiques permettent de vérifier les Getteurs/Setteurs ainsi que les différents constructeurs. Concernant la classe DataBaseTest.java, nous avons décidé de mettre en commentaire les tests sur les fonctions qui peuvent modifier la Base de Données. À la suite de cette implémentation, la couverture de test de notre application est d'environ 65% (80-90% si on ajoute les fonctions de la bd qui peuvent la modifier).

CONCLUSION(BRIAN)

Le but de ce projet est de pouvoir reprendre les bases de programmation java que nous avons, ainsi que le pattern MVC qui est très utilisé dans le monde de l'entreprise. De plus, nous avons utilisé le framework bootstrap, pour obtenir un site de meilleure qualité et plus rapidement, ce qui est très communément utilisé maintenant dans le développement web. De plus, nous avons travaillé en groupe, ce qui nous a permis de développer nos compétences de gestion de projet, mais aussi de gestion des risques. En effet, nous ne sommes pas à l'abri d'un retard, mais nous devons réagir à ces retards. Cela nous permet donc une meilleure organisation du temps, et une vision de notre projet plus claire.