

Rapport de développement d'application réticulaire



BookExchange

Plateforme dédiée à l'échange de livres

Réalisée par :

- Alexandra TUDOR
- Sylia RIGHI

Table des matières

Liste des figures.....	iv
Présentation de l'application	5
• Origine de l'idée.....	5
• Fonctionnalités générales de BookExchange	5
• Les différents cas d'utilisation.....	6
1. Créer un compte utilisateur sur la plateforme BookExchange.....	6
2. Se connecter à la plateforme BookExchange	7
3. Modifier mon profil utilisateur	8
4. Proposer un livre à échanger	9
5. Consulter les livres proposés sur la bibliothèque BookExchange	10
6. Trouver sur la bibliothèque BookExchange un livre qui m'intéresse	11
7. Consulter la fiche d'un livre	12
8. Contacter un Book Exchanger pour un livre qui m'intéresse	13
9. Consulter le profil d'un Book Exchanger	14
10. Supprimer un livre	14
11. Participer à un évènement proposé sur la plateforme BookExchange	15
Conception de l'application	16
• Conception architecturale.....	16
• Conception détaillée.....	16
1. Client :	16
2. Serveur :	16
3. Base de données interne NoSQL	17
4. Les APIs utilisés	18
• Sécurité de l'application	21
1. Authentification.....	21
2. Sessions	21
3. Sécurité de la base de données	22
Réalisation de l'application.....	23
• APIs est technologies utilisées.....	23

1. Choix persistance :	23
2. Choix des Framework et librairies Frontend.....	23
3. Implémentation des servlets.....	23
• Tests de l'application	25
Déploiement de l'application	27
• Choix de l'hébergement.....	27
1. Application web.....	27
2. Base de données	27
Compléments du projet	28
• Organisation et méthodologie de travail suivis.....	28
• Extensions possibles du système	28
1. Extension du système par le système du troc à point	28
2. Evaluation des utilisateurs	29
3. Chats et messagerie interne	29
• Comment monétiser notre application ?	29
1. Via le système à points :	29
2. Affiliation Amazon :	29
Conclusion générale	29

Liste des figures

Figure 1: Adresse postale	6
Figure 2: Géolocalisation d'un utilisateur	6
Figure 3: Page inscription	7
Figure 4: Erreur (Validation du formulaire d'inscription)	7
Figure 5: Page de connexion	8
Figure 6: Modification du profil utilisateur	8
Figure 7: Page de modification du profil utilisateur	9
Figure 8: Proposer un nouveau livre	9
Figure 9: Trouver le livre à proposer	10
Figure 10: Confirmer l'ajout d'un nouveau livre	10
Figure 11: Appliquer des filtres à une recherche de livres	11
Figure 12: Consulter les résultats sur Google Maps	11
Figure 13: Consulter la fiche d'un livre	12
Figure 14: Book Exchangers	12
Figure 15: Consulter la liste des Books Exchangers qui possède un livre	13
Figure 16: Formulaire de contact d'un bookExchanger	13
Figure 17: Consulter le profil d'un autre BookExchanger	14
Figure 18: Profil d'un autre BookExchanger	14
Figure 19: Supprimer un livre de sa liste de livre proposés	15
Figure 20: Participation à un évènement	15
Figure 21 : Architecture globale de l'application "BookExchange"	16
Figure 22: Document d'un livre sur MongoDB	17
Figure 23: Document d'un utilisateur sur MongoDB	18
Figure 24: Cluster Google Maps	19
Figure 25: Application de la variation sur les positions des livres d'un même utilisateur sur Google Maps	20
Figure 26: Exemple de test Junit du servlet de login	25
Figure 27: Résolution de l'application sur Google Chrome	25
Figure 28: Résolution de l'application sur smartphone	25
Figure 29: Exemple du résultat du test de vitesse de chargement sur un navigateur web	26
Figure 30: Exemple du résultat du test de vitesse de chargement sur un navigateur web sur smartphone	26
Figure 31: Déploiement de l'application sur Heroku	27
Figure 32: Hébergement de la base de données MongoDB sur mLab	27
Figure 33: Tableau kanban du suivi du projet sur Trello	28

Présentation de l'application

Vous êtes amoureux de la lecture et vous souhaitez partager votre passion avec d'autres personnes ? Vous voulez échanger vos vieux livres contre de nouveaux pour de nouvelles histoires et aventures ? Rendez-vous sur notre plateforme web [BookExchange](#), une plateforme qui vous permet de manière intuitive de renseigner d'une part les livres que vous êtes prêts à échanger, prêter, ou céder et de les transmettre aux personnes qui sont intéressées, et d'emprunter ou d'acquérir d'une autre part de nouveaux livres proposés par les autres membres inscrits sur la plateforme.

Origine de l'idée

Comme pour la majorité des personnes au monde, vous avez certainement accumulé des centaines de vieux bouquins, pour plusieurs raisons généralement communes : parce qu'ils vous ont coûtés cher, parce que ce sont des cadeaux, parce qu'il « faut » les avoir lus dans sa vie. Vous les lisez, et placez dans un coin de votre maison, ils prennent de la place, et de la poussière !

Une bonne idée pour éviter cela, tout en bénéficiant de nouveaux livres sans avoir à les payer, et de faire le tri, pour ne garder que ceux qui vous touchent le plus, ou à relire un jour, et de relâcher le reste dans la nature pour les laisser vivre leur destinée de livres. C'est de là que vient notre idée de plateforme en ligne qui servira de boîte à livres en regroupant une large communauté de personnes, pour échanger leurs vieux livres.

Certes, plusieurs applications ont été développées dans ce cadre, telle que [Bibliotroc](#), ou [BookMooch](#), mais qui se restreignent soit à la vente, le don, ou l'échange d'un livre par un autre, et nous voulons justement rester plus larges et ouverts aux différents modes d'échanges à travers notre plateforme BookExchange. Nous apportons également notre valeur ajoutée à travers l'organisation d'événements entre un ensemble de membres de la plateforme pour se retrouver dans un endroit d'échange de main à main et de discussion autour des livres.

Fonctionnalités générales de BookExchange

De manière générale, notre application « BookExchange » propose les fonctionnalités suivantes :

- Créer un compte utilisateur pour rejoindre la communauté des Book Exchangers
- Ajouter les vieux livres que vous êtes prêts à échanger, donner ou prêter
- Consulter la liste et les fiches descriptives de tous les livres proposés par les Book Exchangers
- Chercher sur la plateforme par filtrage des livres qui vous intéressent (par catégorie, langue, zone géographique)
- Contacter un BookExchanger pour l'acquisition d'un livre qui vous intéresse
- Participer à des événements livres et de discussions autour de la lecture entre les Book Exchangers.

🐼 Les différents cas d'utilisation

Dans ce qui suit, nous allons détailler l'ensemble des cas d'utilisation les plus importants de notre application web BookExchange :

1. Créer un compte utilisateur sur la plateforme BookExchange

Préconditions :

- Alice est sur la [page de connexion](#) de la plateforme BookExchange.

Description :

Alice souhaite rejoindre la communauté des Book Exchangers, elle clique sur le bouton « Register here » de la page de connexion, elle est redirigée vers la [page d'inscription](#) de la plateforme et un formulaire d'inscription s'affiche à l'écran. Alice saisie les informations indispensables à son enregistrement sur la plateforme tel que l'adresse mail et le mot de passe, et les informations facultatives si elle le souhaite. Elle peut également saisir son adresse manuellement, et la page propose une fonction autocomplete, ou elle peut utiliser la géolocalisation.

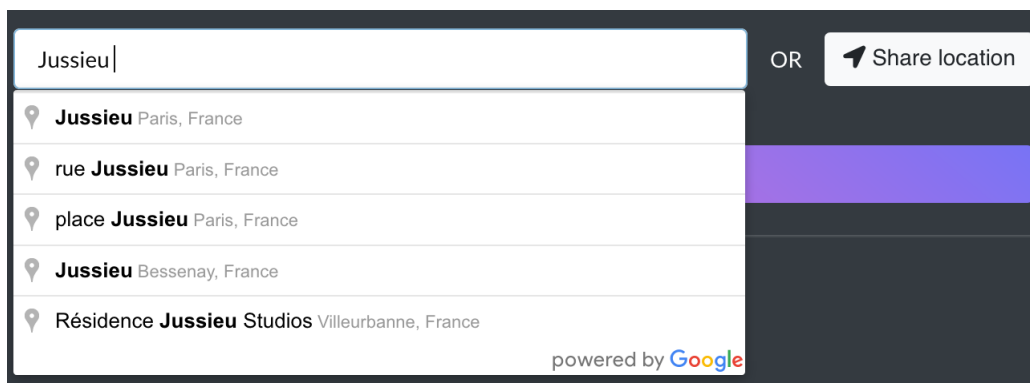


Figure 1: Adresse postale

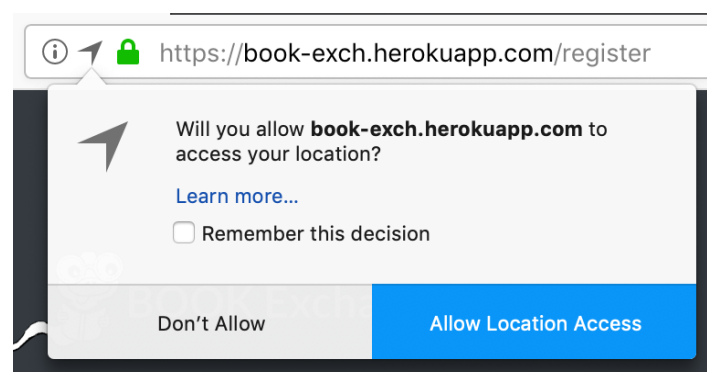


Figure 2: Géolocalisation d'un utilisateur

Alice valide la création de son compte en cliquant sur le bouton « Register ».

Figure 3: Page inscription

Résultat attendu :

Si le formulaire à bien été remplis (les champs obligatoires sont saisis, l'adresse mail renseignée n'a pas déjà été utilisée et le mot de passe saisi est fort), Alice est redirigée vers la page d'accueil. Sinon, un message d'erreur est affiché et Alice doit réessayer.

Figure 4: Erreur (Validation du formulaire d'inscription)

2. Se connecter à la plateforme BookExchange

Préconditions :

- Alice est inscrite
- Alice est sur la [page de connexion](#) de la plateforme BookExchange

Description :

Alice souhaite se connecter à son compte sur la plateforme BookExchange. Sur la page de connexion de la plateforme BookExchange, Alice saisit son adresse mail et son mot de passe avec lesquels elle s'est inscrite, et clique ensuite sur le bouton « Sign in ».

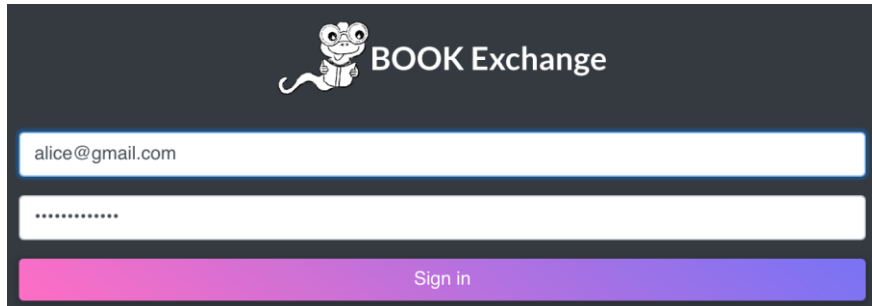


Figure 5: Page de connexion

Résultat attendu :

Si l'adresse mail et le mot de passe sont correctes, Alice est redirigée vers la page d'accueil de la plateforme, sinon un message d'erreur s'affiche pour lui demander de saisir à nouveau les bons identifiants de connexion.

3. Modifier mon profil utilisateur

Préconditions :

- Alice doit être connectée à la plateforme BookExchange.

Description :

Alice clique sur le sous menu « Account » du menu principal, et clique sur le bouton « Edit Profile », ou elle se met sur la page « My Books » et clique sur le bouton « Edit » de son profil ; Alice est redirigée vers un [formulaire de modification](#), elle saisit les nouvelles informations de son profil, et clique sur le bouton « Save modifications ».

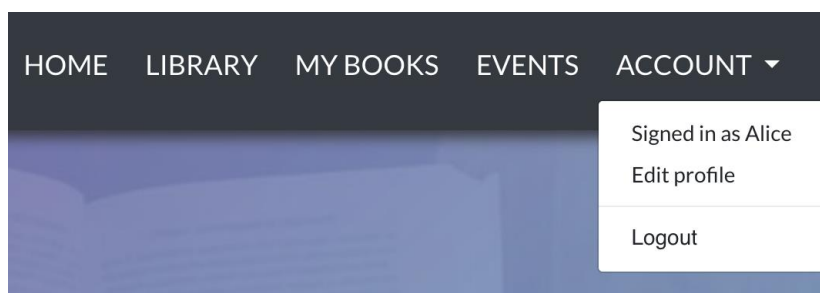


Figure 6: Modification du profil utilisateur



OR

Figure 7: Page de modification du profil utilisateur

Résultats attendus :

Les informations relatives au profil de Alice sont mises à jour.

4. Proposer un livre à échanger

Préconditions :

- Alice doit être connectée à la plateforme BookExchange.

Description :

Alice se met sur la page « My Books » à partir du menu général, et clique sur bouton « [Add book](#) » :

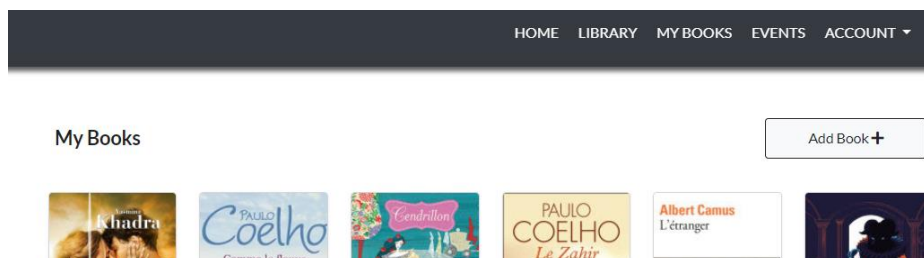


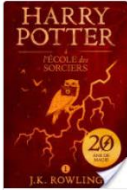
Figure 8: Proposer un nouveau livre

Dans une première étape, Alice cherche le livre qu'elle souhaite proposer, elle peut effectuer la recherche par titre, auteur, isbn, ou par n'importe quel autre critère. Une liste de livres satisfaisant les critères de recherche s'affiche à l'écran, si son livre s'y trouve, elle le sélectionne en cliquant dessus. Sinon, elle clique sur le bouton « Load Next » pour afficher les résultats suivants.

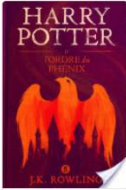
Si Alice Souhaite ajouter manuellement son livre, elle clique sur le bouton « Skip ».

Step 1: Search for your book Skip >


Q



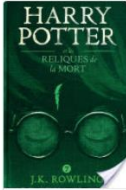
Harry Potter à




Harry Potter et




Harry Potter et



Harry Potter et



Harry Potter et

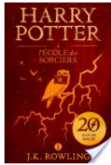


Meet J.K.
Rowling
S. Ward

Figure 9: Trouver le livre à proposer

Dans une deuxième étape, si Alice a sélectionné un livre, ou cliqué sur le bouton « Skip », un deuxième écran de confirmation s'affiche à l'écran, pour remplir ou modifier les informations chargées sur le livre à rajouter et valider son ajout. Pour valider l'ajout, Alice clique sur le bouton « Add ».

Step 2: Confirm



Title

Author(s)

Description

Categories Language Industry Identifiers

Figure 10: Confirmer l'ajout d'un nouveau livre

Résultats attendus :

Le livre s'ajoute à la liste de ses livres ajoutés sur l'écran « My Books » du menu.

5. Consulter les livres proposés sur la bibliothèque BookExchange

Préconditions :

- Alice doit être connectée à la plateforme BookExchange.

Description :

Alice se met sur la page « [Library](#) » à partir du menu général, pour consulter la liste de tous les livres proposés par les autres Book Exchangers. Alice peut restreindre sa recherche à certains critères en les activant sur la barre des filtres qui se trouve sur la gauche de l'interface.

Par exemple, Alice souhaite consulter les livres de « Fiction » qui sont écrits en Anglais, Alice clique sur la checkBox à basculer « Fiction » du volet catégorie, et clique également sur la checkBox « English ». Les filtres s'appliquent automatiquement.

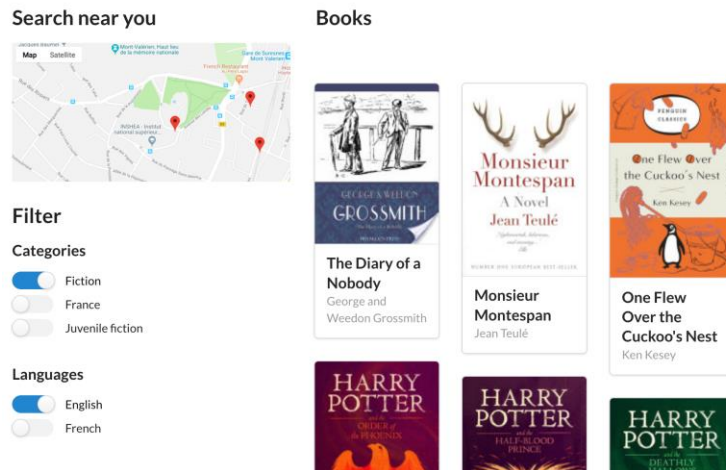


Figure 11: Appliquer des filtres à une recherche de livres

Alice peut aussi consulter les livres sur une carte avec l'option "Search near you":

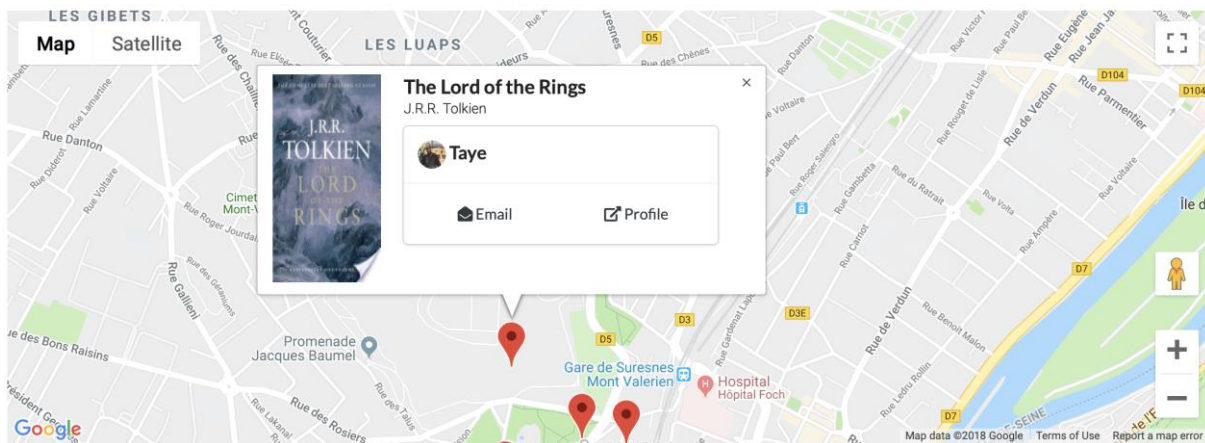


Figure 12: Consulter les résultats sur Google Maps

Résultats attendus :

Les livres proposés par la communauté des Book Exchangers, et qui satisfont les filtres appliqués s'affichent à l'écran, et Alice peut consulter la fiche d'un livre parmi la liste.

6. Trouver sur la bibliothèque BookExchange un livre qui m'intéresse

Préconditions :

- Alice doit être connectée à la plateforme BookExchange.

Description :

Alice se met sur la page Library de la plateforme « BookExchange », elle saisie sur la barre de recherche le titre de son livre, exemple « Harry Potter », et clique sur le bouton « Search ».

Résultats attendus :

Si le livre « Harry Potter » est proposé par un ou plusieurs Book Exchangers, le livres s'affiche, et Alice peut consulter la fiche du livre.

7. Consulter la fiche d'un livre

Préconditions :

- Alice doit être connectée à la plateforme BookExchange.
- Alice trouve le livre dont elle veut consulter la fiche, soit en passant par le cas d'utilisation (6), ou alors en se mettant simplement sur la page « Library » de tous les livres proposés sur la plateforme « BookExchange ».

Description :

Alice, clique sur le livre dont elle souhaite voir l'affiche.

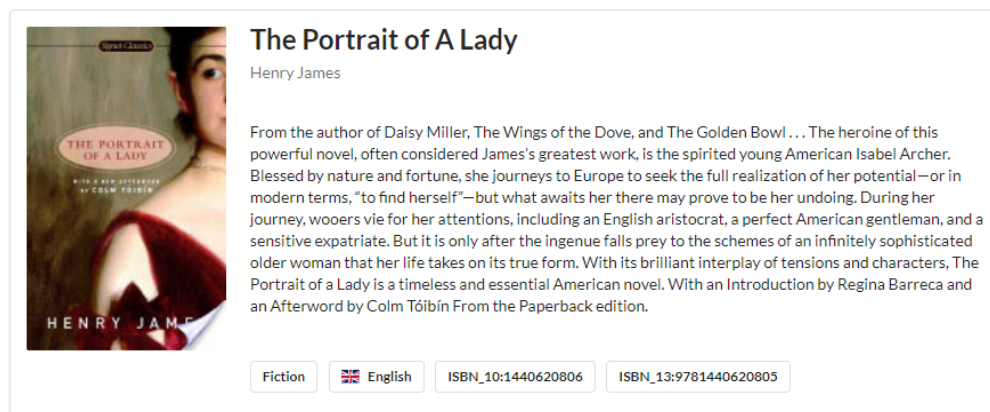


Figure 13: Consulter la fiche d'un livre

Who's trading for it?

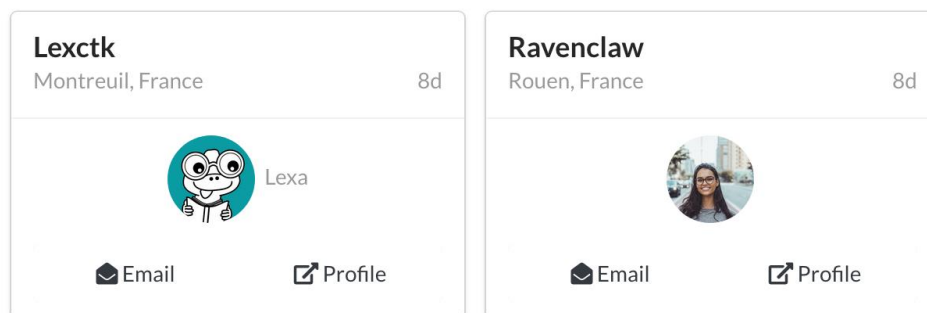


Figure 14: Book Exchangers

Résultats attendus :

Alice est redirigée vers une page avec les détails du livre : titre, auteurs, catégories, description, ISBNs, la langue du livre, ainsi que la liste des Books Exchangers qui proposent le livre.

8. Contacter un Book Exchanger pour un livre qui m'intéresse

Préconditions :

- Alice doit être connectée à la plateforme BookExchange.
- Alice doit être sur la page d'affichage du livre qui l'intéresse (Cas d'utilisation 7), sur la carte (Cas d'utilisation 5), ou sur le profil d'un autre Book Exchanger (Cas d'utilisation 9).

Description :

Alice clique sur le bouton email de la personne qu'elle souhaite contacter.

Who's trading for it?

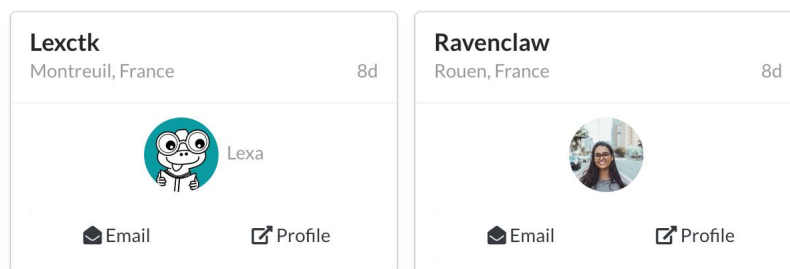


Figure 15: Consulter la liste des Books Exchangers qui possède un livre

Un formulaire de contact pré remplis avec le sujet et informations de contact (ex : Lexctk) s'affiche à l'écran, Alice renseigne le message pour demander le livre, et clique sur le bouton « Send » pour envoyer le message.

Figure 16: Formulaire de contact d'un bookExchanger

Résultats attendus :

Le message envoyé par Alice, et reçu par Lexctk par email.

9. Consulter le profil d'un Book Exchanger

Préconditions :

- Alice doit être connectée à la plateforme BookExchange.
- Alice doit être sur la page d'affichage d'un livre (Cas d'utilisation 7), d'un événement (Cas d'utilisation 13), ou sur la carte (Cas d'utilisation 5).

Description :

Alice clique sur le bouton "Profile" d'un utilisateur :

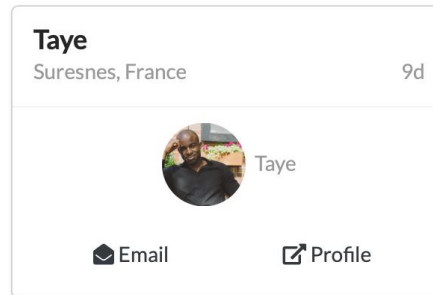


Figure 17: Consulter le profil d'un autre BookExchanger

Résultats attendus :

Alice est redirigée sur une page où elle peut consulter tous les livres proposés par cet utilisateur.

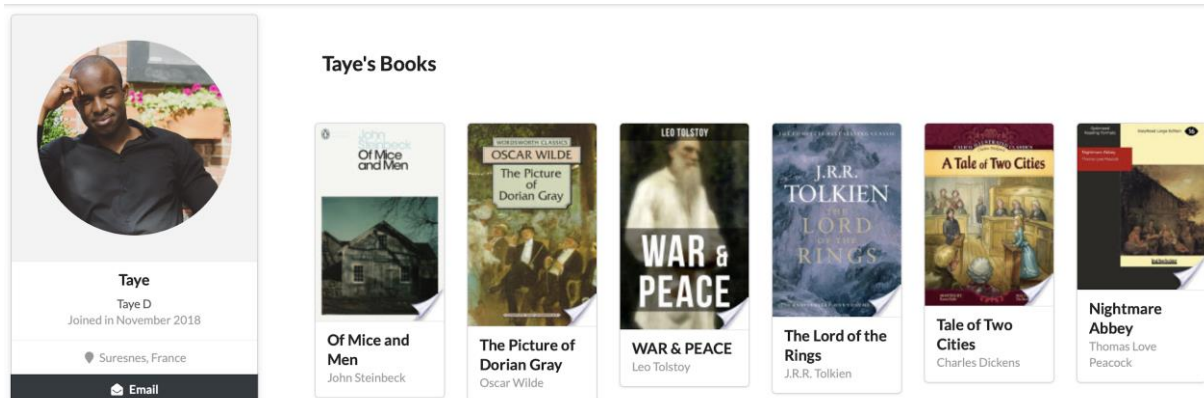


Figure 18: Profil d'un autre BookExchanger

10. Supprimer un livre

Préconditions :

- Alice doit être connectée à la plateforme BookExchange.
- Alice doit être sur la page d'affichage du livre qui l'intéresse (Cas d'utilisation 7), sur la carte (Cas d'utilisation 5), ou sur le profil d'un autre Book Exchanger (Cas d'utilisation 9).

Description :

Alice clique sur le bouton « Delete from my books »



Figure 19: Supprimer un livre de sa liste de livre proposés

Résultats attendus :

Le livre est supprimé de la librairie d'Alice, et Alice est redirigée vers sa page de profil.

11. Participer à un évènement proposé sur la plateforme BookExchange

Préconditions :

- Alice doit être connectée à la plateforme BookExchange.

Description :

Alice se met sur la page "Events" à partir du menu général, elle se met sur l'évènement qui l'intéresse et clique sur le bouton "Attend".

Résultats attendus :

Alice est rajoutée à la liste des participants à l'évènement en question.

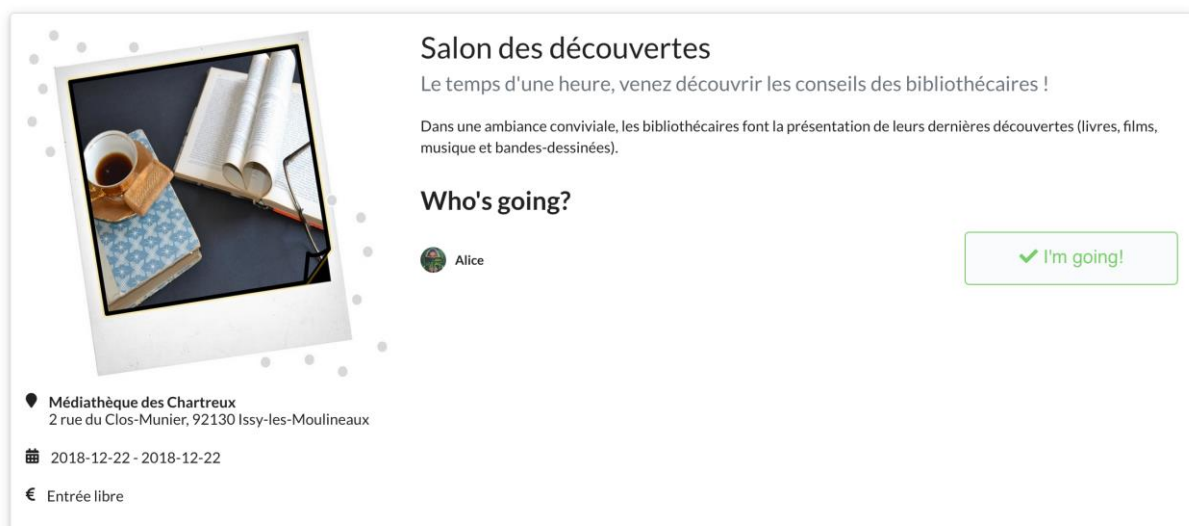


Figure 20: Participation à un évènement

Conception de l'application

Conception architecturale

Le schéma ci-après illustre l'architecture globale de notre application BookExchange

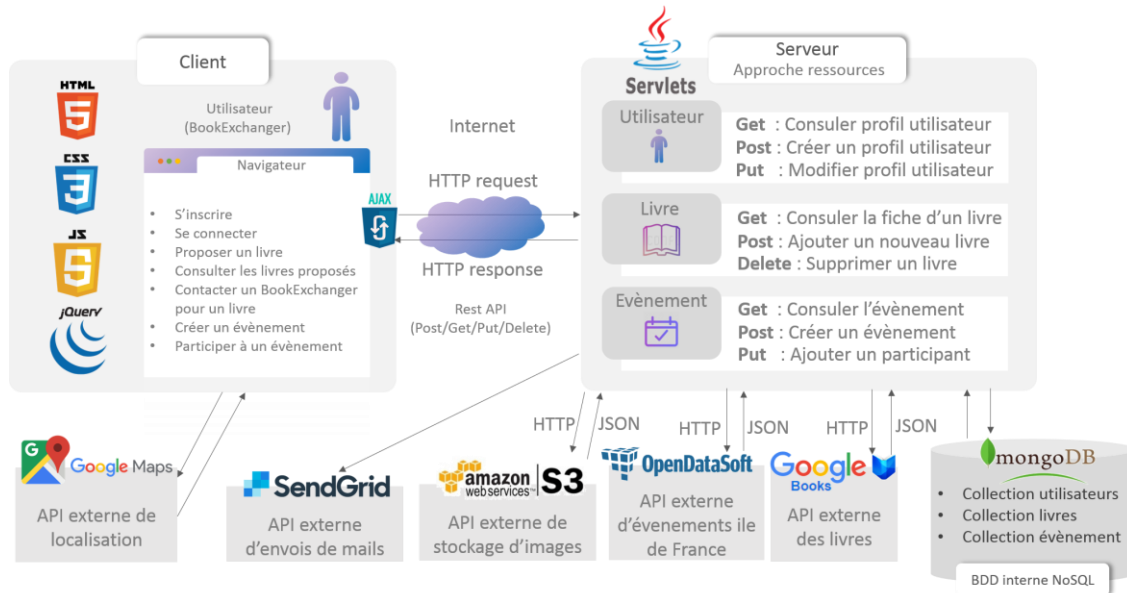


Figure 21 : Architecture globale de l'application "BookExchange"

Conception détaillée

1. Client :

Dans une architecture MVC (model-view-controller), le client représente la vue qui affiche les résultats retournés par les servlets de notre serveur qui représente les contrôleurs du modèle MVC. La communication entre notre client et notre serveur se fait à travers l'API REST qui offre les quatre méthodes (Get, Post, Put et Delete).

2. Serveur :

Pour le serveur de notre application, nous avons opté pour une architecture REST, ce que nous estimons le plus adapté pour une application web légère telle que la nôtre, qui manipule des ressources (livres, utilisateurs et des événements).

Dans cette approche, nous avons divisé notre serveur en un ensemble de servlets pour déposer, récupérer et mettre à jour nos données au niveau de la base de données interne, pour accéder aux APIs [Google Books](#), [Google Maps](#) et [OpenDataSoft](#) afin de récupérer des données concernant les livres qu'on manipule sur notre application et des données de localisation et pour construire nos pages web (JavaScript, HTML/CSS + AJAX).

Pour convertir des chaînes JSON vers des objets de nos modèles de données et vice versa, nous avons utilisé la bibliothèque [GSON de Google](#), ce qui nous a beaucoup facilité la récupération des

données à partir des API externes, et le stockage et restitution des données à partir de notre base de données interne.

3. Base de données interne NoSQL

Pour la couche persistance de notre application, nous avons opté pour un stockage en base de données noSQL orientés documents sous MongoDB, qui utilise des objets JSON pour l'exploitation des données, qui fournit une excellente performance et scalabilité et qui permet une évolution de la taille de données de notre application.

Notre base de données est constituée de documents des différents livres, utilisateurs et évènements de notre système, organisés en trois collections ; ci-dessous, nous présentons deux exemples de documents, l'un représentant un livre, et l'autre un utilisateur :

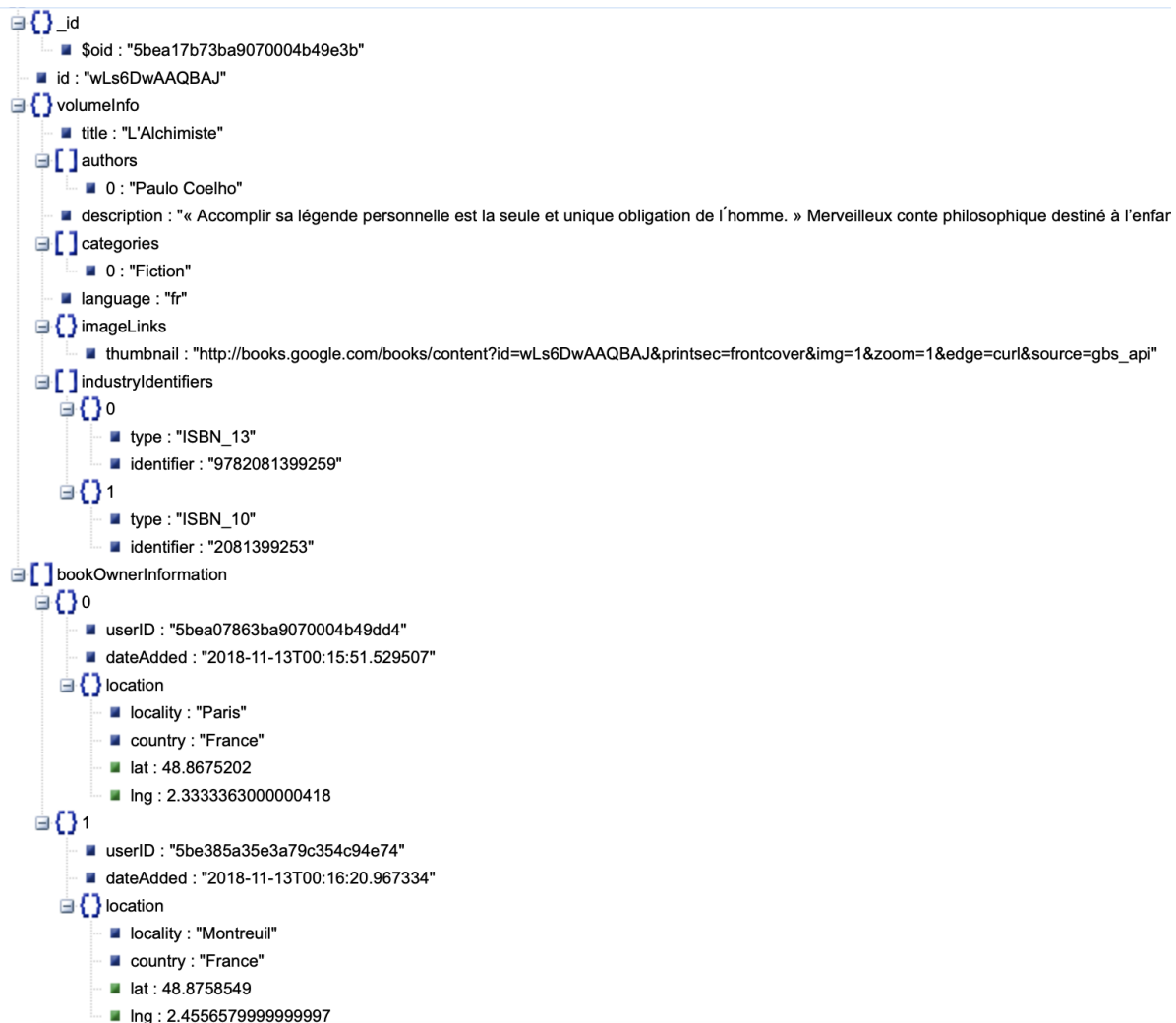


Figure 22: Document d'un livre sur MongoDB

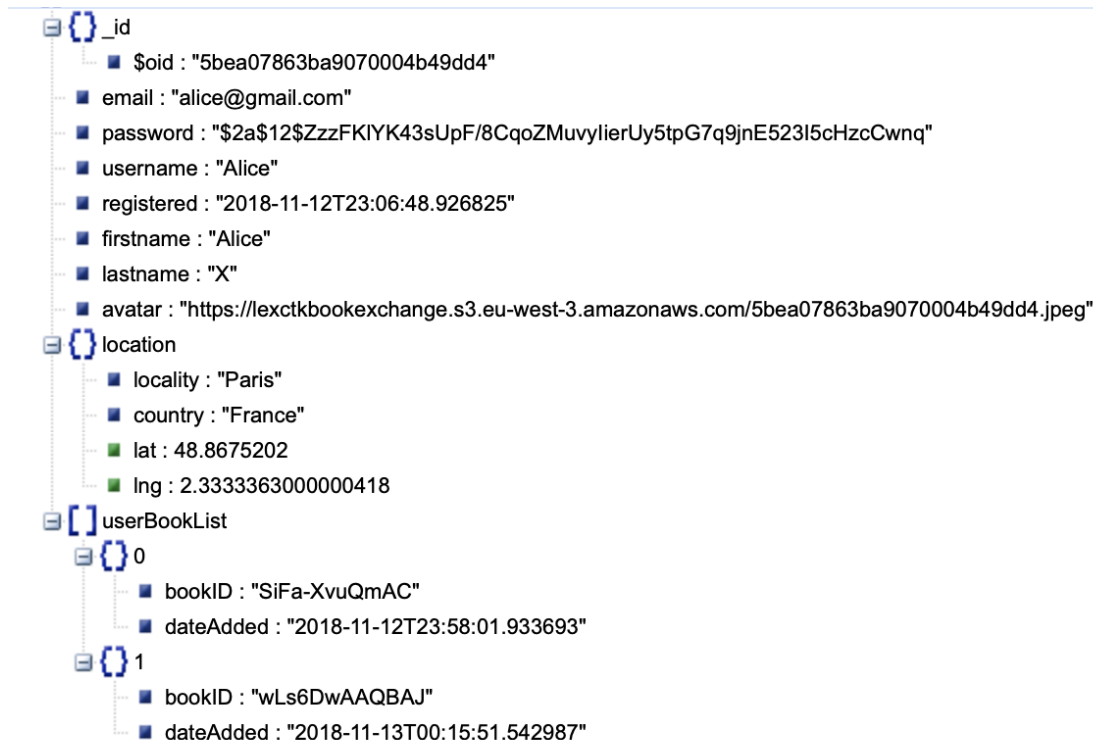


Figure 23: Document d'un utilisateur sur MongoDB

4. Les APIs utilisés

API Google Books :

Pour faciliter aux utilisateurs l'ajout de leurs livres à la plateforme et leur éviter l'ajout manuel, nous utilisons l'API Google Books qui permet à notre application d'effectuer des recherches en texte intégral pour trouver n'importe quel livre et toutes les informations qui le concerne (titre, auteurs, ISBN, langue, photo, description, catégories, ...).

L'API est une API REST, qui permet à travers l'opération HTTP GET de récupérer des livres à travers l'URL suivante :

```
https://www.googleapis.com/books/v1/volumes?q=search+terms
```

Nous filtrons ensuite afin de récupérer uniquement les données qui nous intéressent :

```
&fields=items(id,volumeInfo/title,volumeInfo/authors,volumeInfo/description,volumeInfo/industryIdentifiers,volumeInfo/categories,volumeInfo/language,volumeInfo/imageLinks)
```

Chaque recherche est limitée à 40 résultats, et par défaut, l'API retourne 10 résultats. Pour des raisons de performance, et pour un meilleur affichage, nous avons choisi de retourner 12 résultats par recherche (paramètre maxResults=12).

Ensuite, lorsque l'utilisateur souhaite chercher dans les résultats suivants, nous utilisons le paramètre startIndex pour les afficher.

Cette requête retourne tous les résultats contenant la chaîne de caractères q. Pour effectuer des recherches par attributs, il suffit de spécifier le mot clé dans la partie « search » et de mettre la valeur dans la partie « terms ». Ci-dessous trois exemples de mots clés :

intitle : Renvoie les résultats où le texte suivant ce mot clé se trouve dans le titre.

inauthor : Renvoie les résultats où le texte suivant ce mot clé est trouvé dans l'auteur.

isbn : Renvoie les résultats où le texte suivant ce mot clé est le numéro ISBN.

Exemple de requête, qui retourne l'ensemble de livres dont « [Paulo Coelho](#) » figure dans la liste des auteurs :

```
https://www.googleapis.com/books/v1/volumes?q=inauthor:Paulo coelho
```

Limitations de l'API

- La recherche est limitée par rapport aux résultats affichés sur [Google Books](#), qui utilise la même API, mais avec le moteur de recherche Google. Les résultats dépendent aussi de la géolocalisation du serveur (dans notre cas, le serveur se trouve en Irlande, les résultats sont donc souvent en Anglais, même lorsqu'on cherche un titre en Français).
- Toutes les URL des photos sont en http:// et pas en https://. Afin d'éviter des warnings, nous devons modifier les liens en https:// à chaque recherche.

API Google Maps :

Nous utilisons: [Google Maps Javascript API](#) et [Google Geolocation API](#). Nous utilisons également la librairie [Marker Cluster Plus](#) de Google pour afficher les résultats sous forme de "clusters".

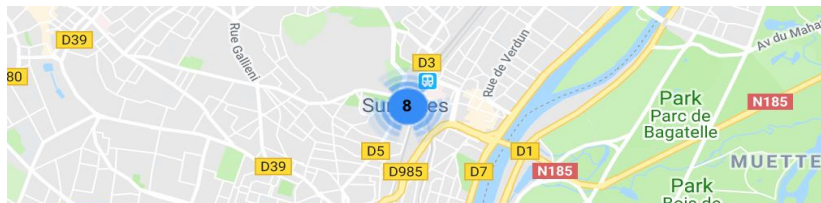


Figure 24: Cluster Google Maps

L'API permet de récupérer des résultats avec une fonction callback en Javascript :

```
<!-- Google API -->
<script src="https://maps.googleapis.com/maps/api/js?key=key&callback=initMap" async defer></script>
```

Nous n'avons pas d'appel à l'API Google Maps au niveau serveur. Les appels se font uniquement en JavaScript. Dans notre base de données, nous gardons seulement les coordonnées géographiques (latitude et longitude) avec la ville et le pays uniquement pour l'affichage, afin d'éviter des requêtes supplémentaires.

Limitations de l'API

- L'API est payante avec un crédit mensuel de \$200 gratuit, ce qui est suffisant pour notre application a ce stade. Cependant, nous avons limité le nombre de requêtes : la position d'un

utilisateur n'est pas modifiée en temps réel, mais l'utilisateur doit éditer son profil pour la modifier. De même, il n'y a pas de recherche par "défaut" sur Google Maps lorsqu'un utilisateur souhaite consulter notre bibliothèque, il faut aller sur "Search near you"

- Lorsqu'un utilisateur ajoute plusieurs livres, tous les livres se retrouvent sur la même position géographique. Pour l'affichage sur une carte et afin de permettre aux autres utilisateurs de cliquer facilement sur un résultat, nous avons introduit une variation aléatoire sur la position (latitude et longitude) de chaque livre, de l'ordre de ± 0.0005 . Les résultats donnent :



Figure 25: Application de la variation sur les positions des livres d'un même utilisateur sur Google Maps

API Amazon S3 (AWS)

Sur Heroku, notre hébergeur d'application, le système de fichiers est éphémère. Chaque dyno (conteneur d'application) obtient son propre système de fichiers éphémère, avec une nouvelle copie du code le plus récemment déployé.

Pendant le cycle de vie du dyno, ses processus en cours peuvent utiliser le système de fichiers comme un bloc-notes temporaire, mais aucun fichier écrit n'est visible pour les processus de tout autre dyno et tous les fichiers écrits seront ignorés dès que le dyno sera arrêté ou redémarré. Par exemple, cela se produit chaque fois qu'un dyno est remplacé en raison du déploiement de l'application et environ une fois par jour dans le cadre de la gestion normale du dyno.

Pour cette raison, nous avons opté pour l'API [Amazon S3](#) pour stocker nos images (images profil des utilisateurs, images des livres lorsque les utilisateurs souhaitent uploader manuellement sans l'API Google Books). Amazon S3 propose une API de haut niveau qui nous permet de charger et déposer des fichiers par de simples appels de méthodes. De plus, il prend en charge plusieurs mécanismes de contrôle d'accès ainsi que le cryptage pour un transit et un stockage sécurisé des données utilisateurs.

API OpenDataSoft

Pour pouvoir récupérer la liste des événements les plus récents de la région île de France, nous utilisons la plateforme Cloud [OpenDataSoft](#), qui propose une API REST prenant en charge les deux méthodes POST et GET pour la recherche et l'analyse de données publiques dans plusieurs domaines.

API SendGrid

Comme il n'est pas possible de configurer un serveur SMTP sur Heroku, nous utilisons l'API [SendGrid](#) pour l'envoi des emails.

Sécurité de l'application

1. Authentification

Pour pouvoir sécuriser les mots de passe des utilisateurs de notre système, nous utilisons la bibliothèque `BCrypt` de Java qui implémente l'algorithme `Bcrypt`. Cet algorithme permet le hachage des mots de passe unidirectionnel basé sur le chiffrement de Blowfish¹ qui utilise un algorithme de hachage adaptatif pour stocker les mots de passe. `BCrypt` génère en interne un sel aléatoire lors du codage des mots de passe et fournit donc un résultat codé différent pour la même chaîne. Mais une chose commune est que chaque fois, il génère une chaîne de longueur 60.

Pour hacher le mot de passe d'un utilisateur avant de le rajouter à la base de données, nous appelons la méthode `hashpw` avec un sel aléatoire, comme ceci :

```
String password = BCrypt.hashpw(plainPassword, BCrypt.gensalt(12));
```

Où la méthode `gensalt ()` utilise le paramètre facultatif (`log_rounds`) qui détermine la complexité de calcul du hachage.

A l'authentification d'un utilisateur, pour vérifier si un mot de passe en texte clair correspond à un mot de passe haché précédemment, nous utilisons la méthode `checkpw`, comme ceci :

```
if (profile.containsKey("password") && BCrypt.checkpw(password, (String)profile.get("password")))
```

Où `profile.get("password")` représente le mot de passe haché et récupéré depuis la base de données.

2. Sessions

Lorsqu'un utilisateur s'enregistre ou se connecte, nous utilisons des variables de session :

```
//generate a new session
HttpSession newSession = request.getSession(true);

newSession.setAttribute("username", username);
newSession.setAttribute("_id", _id);

//setting session to expire
newSession.setMaxInactiveInterval(15*60);
```

Uniquement les pages login, logout et register sont accessibles pour des utilisateurs non-enregistrés. L'accès à l'application est géré par un Filtre, qui vérifie les variables de session.

```
@WebFilter("/app/*")
public class LoginFilter implements Filter {}
```

¹ Algorithme de chiffrement symétrique (à clé secrète), par blocs conçu par Bruce Schneier en 1993 (Schneier, B. (1993, December). Description of a new variable-length key, 64-bit block cipher (Blowfish). In International Workshop on Fast Software Encryption (pp. 191-204). Springer, Berlin, Heidelberg.)

3. Sécurité de la base de données

La base de données est hébergée sur [mLab](#). Tous les serveurs mLab sont exécutés en « mode auth » (la commande mongod est exécutée avec l'indicateur `-auth`). Nous nous connectons à la base de données avec un utilisateur et mot de passe, avec le [driver Java](#)

```
mongodb://<dbuser>:<dbpassword>@ds115523.mlab.com:15523/book_exchange
```

Le compte admin mLab utilisé est protégé par un mot de passe généré automatiquement, et utilise l'authentification à double facteur (Google Authenticator) avec backup SMS.

Les options de sécurité mLab sont payantes, nous ne pouvons pas bénéficier pour notre application

- Connexions SSL.
- Pare feu afin de limiter l'accès aux plages d'adresses IP spécifiques et / ou groupes de sécurité AWS.
- Chiffrement Amazon EBS.

Réalisation de l'application

APIs est technologies utilisées

1. Choix persistance :

Pourquoi MongoDB ?

Dans le cadre de notre système, nous connaissons au préalable les données à gérer par notre système (Utilisateurs, Livres, Evènements), et nous ne ressentons pas le besoin de faire des requêtes assez complexes pour les manipuler. Nous avons donc opté pour une base de données NOSQL, qui permet la modélisation des données sous forme de documents JSON, ce qui réduit au maximum le nombre de relation dans notre base de données, et simplifie de ce fait sa structure, augmente sa lisibilité et offre une grande souplesse d'utilisation et une vraie évolutivité.

2. Choix des Framework et librairies Frontend

Pourquoi JQuery ?

Afin de simplifier la manipulation du DOM, et la gestion des événements et AJAX, nous utilisons la bibliothèque jQuery de Javascript. De plus, Bootstrap et Semantic UI nécessitent jQuery.

Pourquoi Bootstrap ? Semantic UI ?

Nous avons utilisé principalement le Framework [Bootstrap](#) au début de notre projet. Afin de gagner du temps sur la réalisation de la partie frontend, nous avons souvent utilisé des composants du Framework [Semantic UI](#), moins connu, mais qui propose une librairie de composants plus riche sur le plan esthétique.

Semantic UI n'étant pas assez robuste, surtout le côté responsif, nous n'avons pas eu le temps de finir la transition de Bootstrap vers Semantic UI. Actuellement, notre application utilise les 2 Framework : le grid responsive de Bootstrap, et les composants de Semantic UI. Même s'il n'y a pas de conflit, à terme, il faut rester sur un seul Framework.

Filterizr, ImagesLoaded :

Nous avons opté pour l'utilisation des deux plugins [Filterizr](#) et [ImagesLoaded](#) pour filtrer la liste des livres de notre bibliothèque de livres, puisque Filterizr nous permet de filtrer uniquement en frontend, et imagesLoaded nous permet d'appeler filterizr uniquement lorsque toutes les images sont téléchargées ce qui évite des images superposées. De plus, Les deux plugins sont très "lightweight", filterizr ~20kb et ImagesLoaded ~5kb.

3. Implémentation des servlets

Pour mieux organiser notre système et faciliter son évolution et sa maintenance, nous avons organisé nos différents servlets en plusieurs packages, que nous présentons ci-dessous avec quelques exemples de code de nos servlets :

- Package de gestion des requêtes Ajax : qui regroupe les servlets d'affichage de la liste des livres de notre bibliothèque en les chargeant depuis la base de données MongoDB de livre ainsi que la recherche d'un livre sur l'API Google Books.
- Package de gestion de l'authentification : qui regroupe les servlets d'inscription, de connexion et de déconnexion d'un utilisateur à l'application, nous présentons ci-dessus un exemple basique du servlet qui permet la déconnexion d'un utilisateur :

```
/**
 * Servlet implementation class LogoutServlet
 */
@WebServlet("/logout")
public class LogoutServlet extends HttpServlet {

    /**
     * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)
     */
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        RequestDispatcher requestDispatcher = request.getRequestDispatcher("/login.jsp");
        requestDispatcher.forward(request, response);
    }

    private static final long serialVersionUID = 1L;

    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

        response.setContentType("text/html");
        Cookie loginCookie = null;
        Cookie[] cookies = request.getCookies();

        if (cookies != null) {
            for (Cookie cookie : cookies) {
                if (cookie.getName().equals("username")) {
                    loginCookie = cookie;
                    break;
                }
            }
        }

        if (loginCookie != null) {
            loginCookie.setMaxAge(0);
            response.addCookie(loginCookie);
        }

        HttpSession oldSession = request.getSession(false);
        if (oldSession != null) {
            oldSession.invalidate();
        }

        response.sendRedirect("login");
    }
}
```

- Package de gestion des livres : qui regroupe l'ensemble des servlets permettant la récupération des informations des livres et du filtrage des livres ainsi que la recherche des livres les plus proches de soi.
- Package de gestion des événements : qui regroupe l'ensemble de servlets qui permettent de manipuler les événements de l'application, notamment la récupération des plus récents événements autour des livres et de la lecture de la région île de France en utilisant l'API OpenDataSoft.

Tests de l'application

Tests unitaires :

Afin de tester le bon fonctionnement de certaines parties de notre code, plus spécialement les méthodes doGet et doPost de nos servlets, nous nous sommes servis de Junit pour écrire nos programmes de tests. Voici un exemple de test pour la méthode doPost du servlet "LoginServlet" s'occupant de l'authentification d'un utilisateur :

```
@Test
public void testDoPost() throws Exception {
    parameters.put("email", "ds_righi@esi.dz");
    parameters.put("password", "cicile");
    servlet.doPost(request, response);
    verify(request, atLeast(1)).getParameter("email"); //verify email was called
    assertEquals(HttpStatus.SC_OK, response.getStatus()); //verify statut code
}
```

Figure 26: Exemple de test Junit du servlet de login

Tests de compatibilité :

Nous avons testé la compatibilité de nos interfaces web avec les différents navigateurs : Chrome, firefox, safari, ainsi que sur dispositif mobile, afin de vérifier son adaptabilité. Et effectivement, comme notre frontend a été fait mobile-first, les résultats de ces tests étaient plutôt positifs :

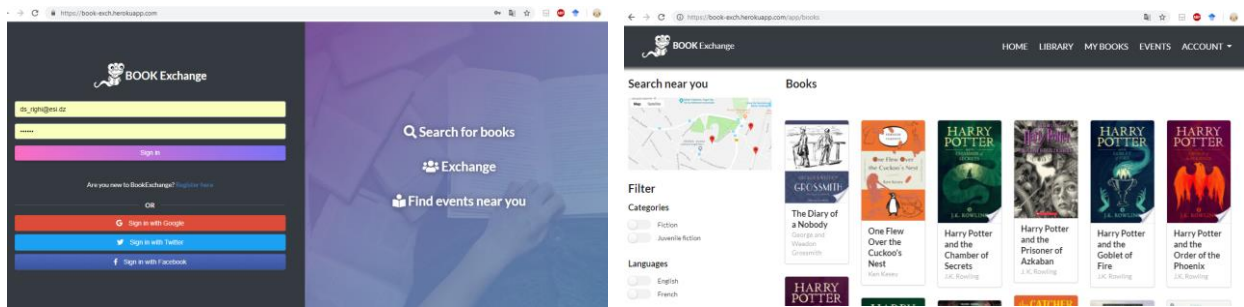


Figure 27: Résolution de l'application sur Google Chrome

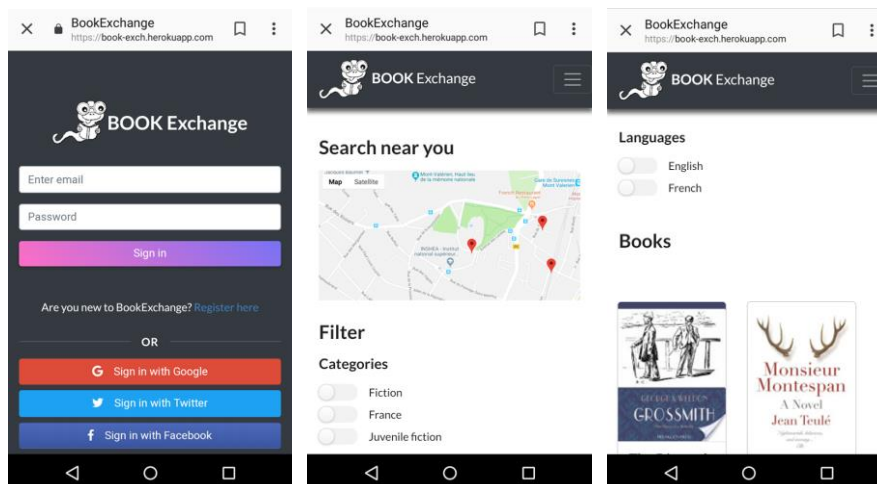


Figure 28: Résolution de l'application sur smartphone

Tests fonctionnels :

Afin d'assurer la qualité de notre application, nous avons mis en place un ensemble de tests fonctionnels pour vérifier qu'elle fonctionne adéquatement. Pour chaque fonctionnalité nous avons comparé le résultat attendu par celle-ci avec le résultat que nous avons obtenus, qui étaient tous les deux conformes pour toutes nos fonctionnalités. En voici un exemple :

ID	Test fonctionnel	Résultat attendu	Résultat obtenu
01	Se connecter à son compte utilisateur pour consulter sa liste de livres : sur la page de login je saisis mon adresse mail et mon mot de passe et je clique sur le bouton « Login »	Je suis loggé, je peux aller dans « My Books » et consulter la liste des mes livres ajoutés	Succès

Test de validation de vitesse :

Afin de mesurer la vitesse de chargement de nos pages, nous avons utilisé l'outil Google Page Speed insights, qui nous a retourné des résultats satisfaisants en général. Ci-dessus, un exemple de résultats obtenus pour la page de login :

Sur web :

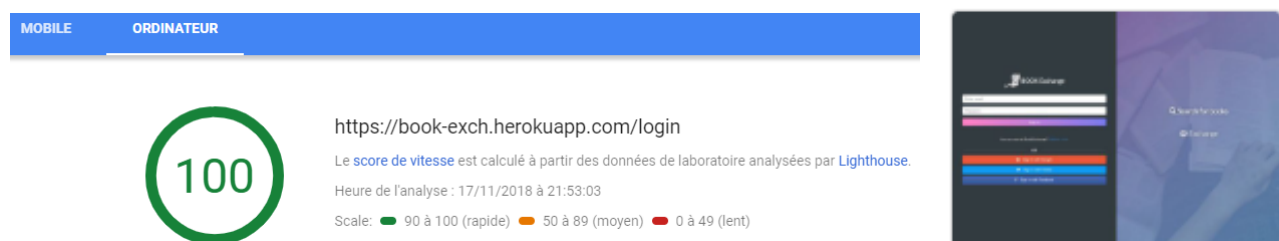


Figure 29: Exemple du résultat du test de vitesse de chargement sur un navigateur web

Sur mobile :

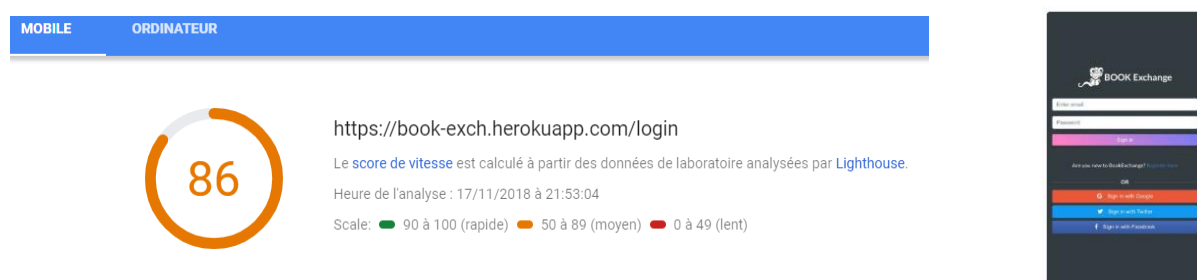


Figure 30: Exemple du résultat du test de vitesse de chargement sur un navigateur web sur smartphone

Déploiement de l'application

Choix de l'hébergement

1. Application web

Nous avons hébergé notre application web sur la plateforme Cloud Heroku (PaaS), en lui attribuant le domaine unique "<https://book-exch.herokuapp.com/>", qui permet de router les demandes HTTP vers le dyno (conteneur d'application) approprié.

Pour ce faire nous avons utilisé le système de gestion de version Git via le client GitHub. Le déploiement est ensuite automatique à partir de GitHub :

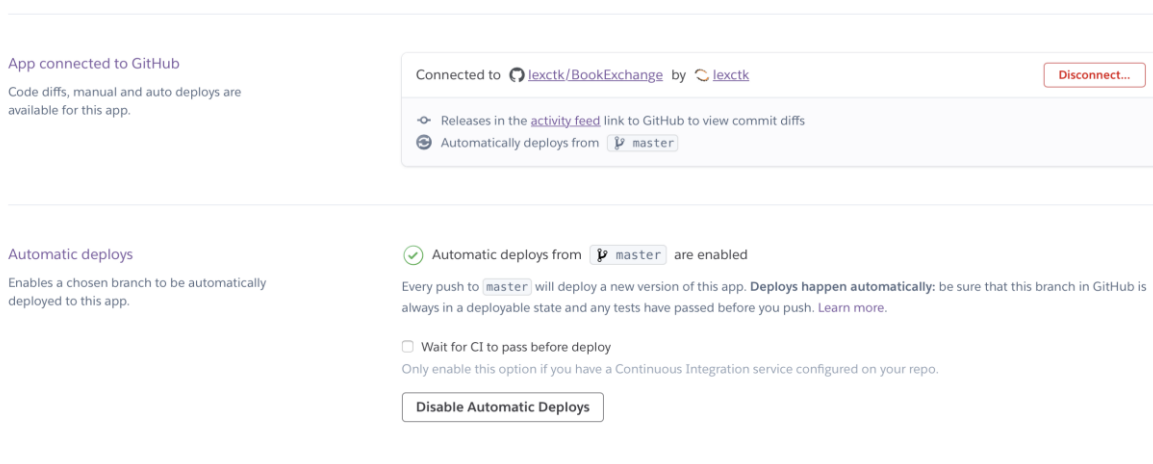


Figure 31: Déploiement de l'application sur Heroku

2. Base de données

Pour héberger notre base de données MongoDB, nous utilisons le service Cloud mLab (SaaS) qui est très compatible et facile à intégrer avec Heroku notre hébergeur de l'application web. Celui-ci offre un provisionnement et une mise à l'échelle automatisés pour notre base de données, une sauvegarde, une récupération et une surveillance 24 heures sur 24, 7 jours sur 7.

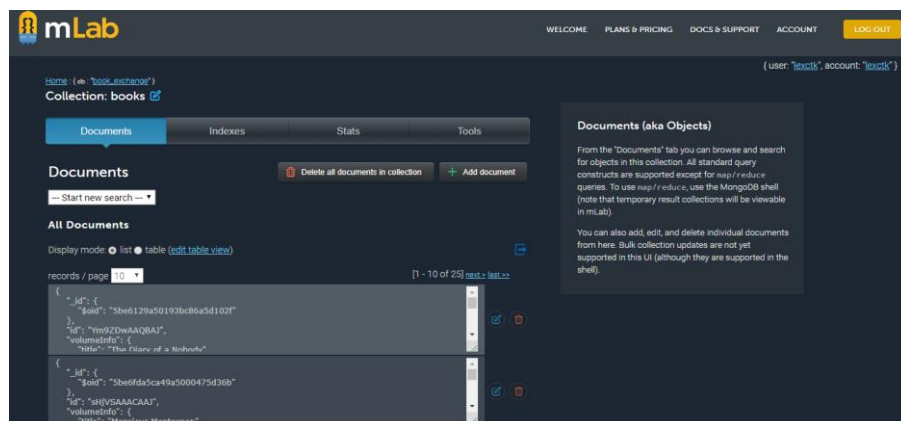


Figure 32: Hébergement de la base de données MongoDB sur mLab

Compléments du projet

Organisation et méthodologie de travail suivis

Pour réaliser le projet nous avons travaillé en binôme en suivant la méthodologie Agile Kanban, ce qui nous a permis d'avoir une meilleure répartition des tâches et une très grande visibilité sur l'avancement du projet grâce au tableau kanban sur l'outil trello :



Figure 33: Tableau kanban du suivi du projet sur Trello

Nous avons également appliqué les bonnes pratiques des méthodologies agiles qui nous ont beaucoup aidé tel que la programmation par paire qui a favorisé l'échange de connaissances entre nous ainsi que l'intégration continue et la gestion de versions qui a beaucoup amélioré la cohérence de notre code grâce à Git avec le client GitHub.

Extensions possibles du système

1. Extension du système par le système du troc à point

Nous souhaitons intégrer prochainement un système de Troc à point pour les échanges livre-contre-livre sur notre plateforme. Chaque livre donné à une autre personne fait gagner un ensemble de points à la personne qui l'avait donné, (le nombre de point peut être fixe par la plateforme à 2 points par livre par exemple, ou alors c'est renseigné par le propriétaire du livre, si le livre est de grande valeur il attribue une valeur de 5 points par exemple)

1. Si le livre que je propose est pris par une autre personne je gagne ses points ;
2. Pour acquérir un nouveau livre il faut avoir le nombre de points nécessaires pour l'avoir ;
3. Une personne peut choisir de donner ses points à un autre membre de la plateforme telle qu'une association caritative.

2. Evaluation des utilisateurs

Dans une prochaine version de l'application, nous souhaitons intégrer un système d'évaluation des différents utilisateurs de l'application à travers des étoiles, et des commentaires, pour permettre aux utilisateurs d'avoir une idée sur l'honnêteté des utilisateurs avant d'échanger leurs livres avec eux.

3. Chats et messagerie interne

Une messagerie interne est également envisageable afin de permettre aux BookExchangers de communiquer entre eux sans avoir à passer par la messagerie Gmail pour l'échange de livre, et de discuter entre eux de manière générale.

Comment monétiser notre application ?

Pour rentabiliser l'application financièrement, nous avons plusieurs idées que nous présentons ci-dessous :

1. Via le système à points :

En intégrant le système à points que nous avons évoqué en tant qu'extension du système, si une personne n'a pas de points, et n'a pas encore gagné de points, elle peut acheter 2 premier points au prix de 5 euros.

2. Affiliation Amazon :

Nous pouvons envisager également de miser sur l'ancêtre des modèles de business du web qui est la publicité, notamment par l'affiliation Amazon (numéro 1 du e-commerce en France), en recommandant un ou plusieurs livres proposés par Amazon afin de toucher des commissions de 7% sur chaque vente.

Conclusion générale

Ce projet est pour nous une expérience très fructueuse, qui nous a permis d'appliquer plusieurs connaissances acquises durant les cours de DAR en pratique, et d'approfondir nos connaissances et compétences personnelles dans le cadre de développement d'applications réticulaires. Cette expérience nous a permis de concrétiser une idée que nous avons proposée, en passant par la conception architecturale de l'application, son développement en Java EE, l'intégration des différentes API REST dont nous avons eu besoin, jusqu'à son déploiement sur le Cloud afin de la rendre visible au grand public.

En prenant en compte les ressources humaines, les contraintes techniques posées ainsi que les délais, nous estimons que nous avons atteint les objectifs initiaux de notre projet, et ça nous tient à cœur de continuer à travailler sur cette application dans le futur afin de l'enrichir et de viser un public plus large.