|  |
| --- |
| 30.01.2023 |

|  |
| --- |
| HE-Arc – Ingénierie – ISC3il (2022-2023)  JEE/Spring I – cours 3291.2 |
| Saphir Lion |
| Rapport du projet |

|  |  |  |
| --- | --- | --- |
| Lieu de formation | Haute École Arc (Neuchâtel Berne Jura) – HE-Arc ingénierie  Espace de l'Europe 11, 2000 Neuchâtel | |
| Étudiant | Allemann Jonas — ISC3il-a | [jonas.allemann@he-arc.ch](mailto:jonas.allemann@he-arc.ch) GitHub : <https://github.com/Ozurah> |
| Enseignant | Chèvre Sébastien Gérard Henri | [sebastien.chevre@he-arc.ch](mailto:sebastien.chevre@he-arc.ch) |
| Source du projet | <https://github.com/Ozurah-HES/SaphirLion> | |



Table des matières

[1 Introduction 2](#_Toc126003476)

[1.1 Dates et délivrables 2](#_Toc126003477)

[1.2 Description du projet 2](#_Toc126003478)

[1.3 Contraintes 2](#_Toc126003479)

[2 Suivi 3](#_Toc126003480)

[2.1 Planning initial 3](#_Toc126003481)

[2.2 Planning effectif 3](#_Toc126003482)

[3 Divergences avec le cahier des charges 4](#_Toc126003483)

[3.1 Modèles / tables 4](#_Toc126003484)

[3.2 Routes remaniées 4](#_Toc126003485)

[3.3 Maquettes 5](#_Toc126003486)

[4 Réalisation 5](#_Toc126003487)

[4.1 Dépendances 5](#_Toc126003488)

[4.2 Architecture du projet 6](#_Toc126003489)

[4.3 Seeder 6](#_Toc126003490)

[4.4 Validation 7](#_Toc126003491)

[4.5 Séparation HTML 8](#_Toc126003492)

[4.6 Pages d’erreurs 8](#_Toc126003493)

[4.7 Logo 8](#_Toc126003494)

[4.8 Problèmes rencontrés 8](#_Toc126003495)

[4.9 Tests 9](#_Toc126003496)

[4.9.1 Configuration pour utiliser une BDD différente de la production 9](#_Toc126003497)

[4.9.2 Tests des modèles 10](#_Toc126003498)

[4.9.3 Tests des repository & services 10](#_Toc126003499)

[4.9.4 Execution des tests 11](#_Toc126003500)

[5 Résultats 12](#_Toc126003501)

[5.1 Page d’accueil 12](#_Toc126003502)

[5.2 Page de login 12](#_Toc126003503)

[5.3 Ma bibliothèque 13](#_Toc126003504)

[5.4 Ajout, édition d’un média 14](#_Toc126003505)

[5.5 Page d’erreur HTTP 14](#_Toc126003506)

[5.6 Autres résultats 14](#_Toc126003507)

[6 Améliorations 15](#_Toc126003508)

[7 Conclusion 15](#_Toc126003509)

[8 Remerciement 16](#_Toc126003510)

[9 Sources 16](#_Toc126003511)

[10 Annexes 16](#_Toc126003512)

# Introduction

Ce document fait état du semestre d’automne du projet Spring à réaliser dans le cadre de la troisième année de notre formation d’ingénieur en informatique en filière développement logicielle. Il couvre toute la durée du travail ayant débuté le 18 octobre 2022 et prenant fin le 30 janvier 2023.

Les chapitres qui sont abordés dans ce rapport sont :

1. Le suivi du développement de l’application (planning initial comparé au réel)
2. Les divergences avec le cahier des charges (les changements apportés durant la réalisation)
3. Ce qui a été réalisé, avec les problèmes rencontrés et les tests effectués
4. Les résultats (état de l’application, utilisation)
5. La conclusion avec un bilan personnel

* Les phases d’analyse et de conception ont été réalisées lors de la rédaction du cahier des charges et ne sont pas reprises dans ce document. Veuillez vous y référer pour les schémas ou diagrammes.
* Le cahier des charges est disponible en annexe. De même pour le guide d’installation du projet (expliquant comment l’obtenir, les configurations à effectuer, démarrer l’application).

Ce travail sera évalué afin de valider le cours.   
Concernant la quantité de travail demandé, elle est de 10h environ.

Le sujet était libre tant qu’il respectait certains critères. Celui retenu est un site permettant de suivre son avancée dans les séries (principalement les mangas), en termes de lecture et d’achats comparés à ce qui est déjà paru.

## Dates et délivrables

**Mardi 18 octobre 2022** : Début du travail ;  
**Mardi 08 novembre 2022** : Remise du cahier des charges ;  
Jusqu’à **lundi 30 janvier 2023, 23h59** : Remise des délivrables (application, rapport, guide d’installation).

## Description du projet

* La description est reprise du cahier des charges, sans modification.

L'idée du projet retenu consiste à réaliser un site pour le suivi de lecture de ses mangas. De cette manière, le lecteur à la possibilité de savoir où il en est dans ses achats, se souvenir du dernier tome lut, etc.

Sur le site, il sera possible de saisir le nom du manga, le nombre de tomes déjà sortis/achetés, le nombre de tomes que l'on a déjà lu, et on peut laisser un commentaire sur le manga.  
Le site s'étendrait naturellement à d'autres types de médias (livres, films, séries, etc.).

Afin de mettre à disposition ce site à plusieurs utilisateurs, un système de login sera mis en place de manière à pouvoir sauvegarder les informations de chaque utilisateur.

## Contraintes

Les contraintes apposées par l’enseignant sont les suivantes :

* Application Spring Web avec UI
* Architecture découplée avec injection de dépendances
  + Structure claire avec utilisation correcte du framework Spring
* Accès DB
  + A minima: insert/update/read
  + Paginations
* Gestion utilisateur et sessions
* Sur repository Git
* Tests unitaires sur tout ce qui est public

# Suivi

Ce chapitre permet d’avoir un suivi du déroulement du projet, il est mention du planning initial et effectifs, avec une explication sur les différences.

## Planning initial

|  |  |  |
| --- | --- | --- |
| Date | Durée | Tâche |
| 24.10.2020 | - | Explication du type de projet à réaliser |
| 08.11.2022 | 1h30 | Remise du cahier des charges |
| 14.11.2022 | 1h30 | Mise en place de la structure du projet |
| 21.11.2022 | 1h30 | Page de connexion |
| 28.11.2022 | 1h30 | Page d'affichage de ses mangas |
| 05.12.2022 | 1h30 | Page d'ajout d'un manga |
| 12.12.2022 | 1h30 | Page d'édition d'un manga |
| 19.12.2022 | 1h30 | Design/Finitions du site |
| En parallèle des autres éléments | - | Tests unitaires Documentation |
| 16.01.2023 | - | Remise du projet |

## Planning effectif

* La durée de chaque tâche est disponible sur GitHub dans les [issues](https://github.com/Ozurah-HES/SaphirLion/issues)[[1]](#footnote-2). Chaque issue possède un « mini journal de travail » avec les heures effectives et des explications de ce qui a été réalisé.
* Dans le tableau, chaque tâche possède un lien vers l’issue qui lui est liée (« #X » ou X correspond au numéro de l’issue).
* La date de fin du projet a été déplacée du 16.01.2023 au 30.01.2023.

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Durée | Tâche | Issue liée |
| 24.10.2020 | - | Explication du type de projet à réaliser | - |
| 14-15.01.2023 | Indéterminé | Remise du cahier des charges | - |
| 14.11.2022 | 6h20 | Mise en place de la structure du projet | [#1](https://github.com/Ozurah-HES/SaphirLion/issues/1) |
| 15-16.01.2023 | 3h00 | Page de connexion | [#2](https://github.com/Ozurah-HES/SaphirLion/issues/2) |
| 17-18.01.2023 | 2h45 | Page d'affichage de ses mangas | [#3](https://github.com/Ozurah-HES/SaphirLion/issues/3) |
| 18-20.01.2023 | 7h55 | Page d'ajout & édition d'un manga | [#4](https://github.com/Ozurah-HES/SaphirLion/issues/4)*,* [#5](https://github.com/Ozurah-HES/SaphirLion/issues/5) |
| 20.01.2023 | 2h00 | Design | [#6](https://github.com/Ozurah-HES/SaphirLion/issues/6) |
| 21.01.2023 | 5h10 | Finitions du site & tests | [#12](https://github.com/Ozurah-HES/SaphirLion/issues/12) |
| En parallèle des autres éléments | Indéterminé | Documentation | - |
| 30.01.2023 | - | Remise du projet | - |

Il y a de grands écarts de dates à cause de la charge de travail que nous avions pour les autres cours. Le projet a réellement débuté une fois la théorie vue en classe. Concernant les durées, il y a plusieurs explications :

* De manière générale, il s’agit surtout de la correction de bug et de recherche des bonnes pratiques pour réaliser un projet Spring.
* La mise en place comprend tout ce qui est modèles (DB), repository et services. Il a donc été nécessaire de s’assurer du bon fonctionnement de ces classes avant de pouvoir réellement commencer le projet.
* Pour la page d’affichage de ses mangas (liste), ce qui a pris du temps était la mise en place de la pagination.
* Les pages d’ajout et d’édition ont été fusionnées (il s’agit de la même logique). Ce qui a pris du temps était la gestion correcte des erreurs de validations (grâce au « validator »).
* Pour les tests, il a fallu s’intéresser aux manières de réaliser les tests (non vu en cours). Quant aux finitions, il s’agissait de corriger quelques bugs mineurs et s’assurer du bon fonctionnement de l’application.

Au final, l’implémentation aura pris un **total de 27h10**.

# Divergences avec le cahier des charges

Quelques différences entre ce qui a été effectivement implémenté et le cahier des charges sont à noter.

## Modèles / tables

Modèles user avec la table UserAccount

Un modèle « User » permet la création des utilisateurs. Celui-ci génère également la table pour la base de données. Afin d’assurer une compatibilité avec H2, le nom de la table lié à ce modèle n’est plus « User », mais « UserAccount ». Ce changement n’a aucune incidence au niveau de Spring.

« type » et « catégories »

Dans la version actuelle du programme, ces deux tables (modèles) sont implémentées et liées correctement au « media », mais ne servent à rien. L’utilisateur n’a aucune gestion de ces deux éléments : il ne peut ni les modifier, ni en créer, ni en supprimer. Ces données pourraient être utilisées pour une prochaine version de l’application.

## Routes remaniées

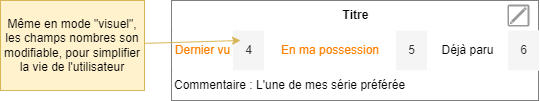
Certaines routes ont été remaniées, le tableau ci-après contient liste complète des routes. Celles qui changent ont leur cellule en vert, les routes supprimées ont leurs cellules en rouge.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Description | Route | Contrôleur | Méthode | Redirection |
| Page d'accueil | /, home, index | IndexController | Get | - |
| Page de connexion | /login | Géré par Spring Security | Get | - |
| Connexion de l'utilisateur | /login | Post | / |
| Déconnexion | /logout | Get | / |
| Page avec la liste des médias de l'utilisateur | /media | MediaController | Get | \* |
| Page d'ajout d'un média | /media/add | MediaController | Get | \* |
| Validation de l'ajout d'un média | /media/add | MediaController | Post | \* |
| Page d'édition d'un média | /media/edit/{id} | MediaController | Get | \* |
| Validation de l'édition/création d'un média | /media/edit | MediaController | Put | « \* » « /media »  En cas d’erreurs : « /media/edit/{id} »  ou « /media/add » |
| Suppression d'un média | /media/delete/{id} | MediaController | Delete | \* /media |
| Page d’erreur | /error | MyErrorController | - | - |

Les pages d’édition et d’ajout d’un média valident toutes deux leurs données en passant par la route « /media/edit », étant donné que la logique est presque la même.

## Maquettes

Initialement, il était prévu dans les maquettes d’avoir les champs modifiables directement dans la liste des médias enregistrés.



Pour faire ceci, il aurait nécessité l’usage de JavaScript pour correctement gérer cette fonctionnalité.   
Compte tenu du temps passé sur le projet, ces champs ne sont pas modifiables directement dans la page listant tous nos médias.

# Réalisation

## Dépendances

Les dépendances du projet sont indiquées dans le fichier « [pom.xml](https://github.com/Ozurah-HES/SaphirLion/blob/main/app/pom.xml)[[2]](#footnote-3) » du projet.  
Le projet a été réaliser avec la version « starter parent » 3.0.1 de Spring.

* Les exercices en cours employaient la version 2.7.1.  
  Certaines syntaxes diffèrent entre ces deux versions, par exemple les imports « javax » sont remplacés par « jakarta ».

La version Java acceptée est la 11. Cependant, le projet n’a été **testé que** sur une machine **avec Java 17**.

Les dépendances ajoutées sont :

|  |  |  |
| --- | --- | --- |
| Groupe | Artéfact | Utilité |
| org.springframework.boot | spring-boot-starter-data-jpa | Accès aux BDD avec JPA |
| com.h2database | h2 | Utilisation d’une BDD de type H2 |
| com.mysql | mysql-connector-j | Utilisation d’une BDD de type « MySql » |
| org.springframework.boot | spring-boot-starter | Fonctionnalités de base |
| org.springframework.security | spring-security-web | Fonctionnalités de sécurité (exemple le login des utilisateurs) |
| org.springframework.security | spring-security-crypto | Utilisation de BCrypt pour les mots de passe |
| org.springframework.security | spring-security-config | Configuration de Spring Security |
| org.springframework.boot | spring-boot-starter-validation | Validation des données saisies dans les formulaires |
| jakarta.validation | jakarta.validation-api | Validation des données Java |

* Les versions sont sélectionnées automatiquement

Bootstrap a été utilisé pour le design des pages. Plutôt que passer par les dépendances Spring, nous utilisons le CDN de la version 5.2.3. Pour les icônes, il s’agit du CDN de bootstrap icons, version 1.10.2. Ces CDN sont spécifiés dans le fichier « [header.html](https://github.com/Ozurah-HES/SaphirLion/blob/main/app/src/main/resources/templates/fragment/header.html)[[3]](#footnote-4) ».

## Architecture du projet

Le repo Git à la structure suivante :

|  |  |
| --- | --- |
| [Repo Git]/  ├── app/  └── doc/ | * App contient le projet Spring * Doc contient les documents du projet, tel que le cahier des charges ou les schémas. |

Le projet suit la structure vue en cours (arbre pour le dossier « /app/src ») :

|  |  |
| --- | --- |
|  |  |
| La structure du dossier « test » respecte la même structure que le dossier « main ».  Différences avec la structure vue en cours :   * Le dossier « seed » a été ajouté au « model ». Celui-ci contient des seeders qui permettent d’ajouter automatiquement des entrées dans la BDD. * Pour les services, les interfaces sont suffixées par « \_I », alors que l’implémentation ne possède pas le suffixe « Impl ». * Le dossier « utils » contient des classes utilitaires. * Le dossier « validator » contient des classes servant à la validation des données. * Pour le dossier « resources » :   + « static » contient 3 sous-dossier, un pour les styles (css), un pour les images et un pour le JS.   + « templates » contient les vues (.html). Les vues directement dans ce dossier sont destinées à être une page complète. Alors que les « .html » dans le sous-dossier « fragment » sont des parties à intégrer dans les vues. |

## Seeder

Trois seeders ont été créé, ils se trouvent dans le dossier « [model/seed/](https://github.com/Ozurah-HES/SaphirLion/tree/main/app/src/main/java/ch/hearc/SaphirLion/model/seed)[[4]](#footnote-5) ». Ces classes implémentent « CommandLineRunner » leur permettant d’être exécuté automatiquement au démarrage du projet.

Pour sélectionner quels seeders exécuter lors du démarrage de l’application, des profiles ont été mis en place. Ces profils sont à activer dans « spring.profiles.active=… » du fichier « [application.properties](https://github.com/Ozurah-HES/SaphirLion/blob/main/app/src/main/resources/application.properties)[[5]](#footnote-6) ».

Voici la liste des seeders :

|  |  |  |  |
| --- | --- | --- | --- |
| Ordre d’exécution | Classe | Profil d’activation | Utilité, remarque |
| 1 | DBCleaner | « clean-db » ou « seed-test » | Supprime le contenu des tables |
| 2 | DataSeederProd | « seed-prod » | Ajoute les données de bases pour la production. Ça comprend : 2 utilisateurs (« User 1 » et « User 2 ») ayant comme mot de passe « password » ; 1 type et 1 catégorie. Les données sont ajoutées que si les tables correspondantes ne possèdent pas d’entrées. |
| 2 | DataSeederTest | « seed-test » | Ajoute plusieurs entrées dans toutes les tables. |

## Validation

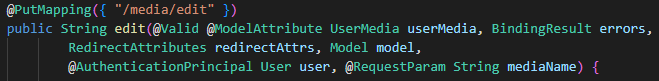
Pour s’assurer des données saisies par l’utilisateur dans les formulaires, un système de validation a été mis en place avec Spring.

Dans les modèles, il faut ajouter des annotations provenant de « jakarta.validation.contraints » au-dessus des champs ou méthodes. Voici un exemple pour des nombres positifs, une taille limite pour un string et une méthode personnalisée :

Une image contenant texte, capture d’écran, écran

Description générée automatiquement

Pour appliquer la validation, il faut ajouter dans les paramètres des méthodes du contrôleur « @Valid » devant le modèle utilisé pour le formulaire et « BindingResult » :



Il est également possible d’« Autowired » un validateur pour appliquer la validation manuellement. Nous l’avons par exemple fait pour valider qu’un « UserMedia » appartient bien à l’utilisateur qui tente de le modifier. Pour y parvenir, nous avons implémenté dans le dossier « validator » la classe BelongConnectedValidator (qui implémente « Validator »). Voici un exemple d’usage pour cette validation :



Une image contenant texte

Description générée automatiquement

* La méthode « addFlashAttribute » permet de transmettre des attributs à une autre méthode d’un contrôleur lors d’une redirection.

## Séparation HTML

Afin de simplifier la création des vues HTML, différents fragments ont été créés :

* « nav.html » : contiens le menu de navigation
* « header.html » : définit le « doctype », le « head » et ouvre le « body » en insérant le fragment « nav » et ouvre la div servant à contenir le corps du site.
* « footer.html » : referme les éléments du « header.html » et ajoute un footer.

De cette manière, la création d’une vue complète peut être faite de cette manière :

Une image contenant texte

Description générée automatiquement

Code 1 - Minimum d'un « .html » pour avoir une page complète

## Pages d’erreurs

Quand une erreur http survient, la page affiche un message d’erreur personnalisé.

Pour y parvenir, il a fallu ajouter l’entrée « server.error.path=/error » dans le fichier « [application.properties](https://github.com/Ozurah-HES/SaphirLion/blob/main/app/src/main/resources/application.properties)[[6]](#footnote-7) ». Un contrôleur spécifique (« MyErrorController ») permet de déterminer quel type d’erreur est survenue. Ce contrôleur va afficher la page « error.http.html ».

## Logo

Le logo a été spécialement réalisé pour l’application avec Gimp. Il s’agit d’un lion (référence au nom du site) au-dessus d’un livre (le contenu principal du site). Le livre est de couleur bleu saphir.

## Problèmes rencontrés

Plusieurs problèmes ont été rencontrés durant la réalisation du projet. Certains ont pu être corrigé rapidement après quelques recherches sur internet (notamment grâce aux tutoriels du site **Baeldung)**, d’autres ont été plus bloquants. Ci-après figurent quelques problèmes principaux, avec leur solution.

Java version 61 au lieu de 55

Au début du projet, je n’étais pas en mesure d’exécuter le « Maven clean install » pour le projet, il y avait une erreur avec les versions de Java. Il y avait deux causes :

* Maven utilise la version de java spécifié dans la variable d’environnement « JAVA\_HOME ». Sur mon PC, ce chemin pointait vers une ancienne version.
* L’autre cause était liée à VSCode. Celui-ci prend par défaut la première version de Java qu’il trouve sur le PC, indépendamment des variables d’environnement « PATH » et « JAVA\_HOME ». Il a fallu spécifier à VSCode d’utiliser spécifiquement « JAVA\_HOME ».

Spring Security

Plusieurs problèmes ont été rencontrés avec la mise en place de Spring Security :

1. La mise en place de Spring Security a nécessité quelques adaptations par rapport à celle vue en cours. Avec l’usage de Spring « 3.0.1 », certaines méthodes étaient devenues obsolètes.
2. Si les ressources statiques (css, images, etc.) ne sont pas explicitement autorisées pour les utilisateurs, la redirection après la connexion avait un comportement étrange : plutôt que rediriger sur la page d’accueil ou précédente, nous étions redirigés sur une image présente dans la page (par exemple l’adresse du Logo : « <http://localhost:8080/images/logo.png> » ).
3. Lors de l’édition de certains champs dans les formulaires, une erreur 500 était retournée lors de la validation

Voici les changements effectués pour corriger ces problèmes (dans la méthode « [SecurityConfiguration .securityFilterChain](https://github.com/Ozurah-HES/SaphirLion/blob/f48dcbb8eac0b53d87a340e3449bf0aedb97d3fe/app/src/main/java/ch/hearc/SaphirLion/SecurityConfiguration.java#L36)[[7]](#footnote-8)»  :

1. « antMatchers » est remplacé par « requestMatchers »
2. Permission des ressources statiques : « .requestMatchers("/js/\*\*", "/css/\*\*", "/images/\*\*").permitAll() »
3. Désactivation des vérifications « CSRF » : « .csrf().disable() »

## Tests

Les tests unitaires ont été effectués avec JUnit et Spring Boot Test.  
JUnit met à disposition des méthodes de type « assert », tel que « assertTrue », « assertNull », etc. Ces méthodes font échouer le test si la condition ne correspond pas au type d’assertion.

Comme mentionné dans le chapitre « 4.2 Architecture du projet » et par convention, l’arborescence du dossier de test correspond à celle du dossier « main ». Chaque classe testée est suffixée par « Test ». Exemple « main/…/model/UserMedia.java » 🡪 « test/…/model/UserMediaTest.java ».

Les tests effectués couvrent les méthodes publiques des classes « [model](https://github.com/Ozurah-HES/SaphirLion/tree/main/app/src/test/java/ch/hearc/SaphirLion/model)[[8]](#footnote-9)», « [repository](https://github.com/Ozurah-HES/SaphirLion/tree/main/app/src/test/java/ch/hearc/SaphirLion/repository)[[9]](#footnote-10)» et « [service](https://github.com/Ozurah-HES/SaphirLion/tree/main/app/src/test/java/ch/hearc/SaphirLion/service)[[10]](#footnote-11)».

Pour les tests utilisant la base de données (repository et service), il est nécessaire de les dissocier et de les rendre indépendants de la BDD de production. Pour y parvenir, il a fallu les configurer pour qu’ils utilisent une autre BDD. Dans notre cas, ils emploient la BDD H2 en mémoire.

### Configuration pour utiliser une BDD différente de la production

Les classes utilisant la base de données, ainsi que les « beans » Spring possèdent l’annotation suivante :



L’annotation « TestPropertySource » permet d’override certaines propriétés chargées. Dans notre cas, les fichiers appelés permettent d'utiliser un seeder spécifique (profile) et modifier les informations d’accès à la BDD (pour utiliser H2). Ces fichiers de propriétés sont définis [ici sur le dépôt](https://github.com/Ozurah-HES/SaphirLion/tree/main/app/src/test/resources)[[11]](#footnote-12).

### Tests des modèles

* Les getters/setters ne sont pas testés.

Si l'on prend par exemple la classe UserMedia.java, elle contient deux méthodes publiques retournant un booléen. Les tests effectués vérifient les cas où le résultat retourne vrai, et faux.

* Bien que les deux méthodes puissent se ressembler, il est quand même nécessaire de les tester distinctement, car rien ne prouve que leur comportement soit le même.

|  |  |
| --- | --- |
| **isNbPublishedGreaterThanLastSeen()** | **isNbPublishedGreaterThanNbOwned()** |
|  |  |

### Tests des repository & services

Un autre exemple de test implémenté consiste à vérifier les fonctions personnalisées utilisant @Query dans les repository.

* Même si beaucoup de tests ont été faits, seul un exemple pour UserMediaRepository.java est mentionné dans ce chapitre. La structure des autres tests est similaire.

La fonction ci-après récupère les données depuis une base de données. Ces données sont les UserMedia d'un utilisateur en particulier.

UserMediaRepository.java



Et le test lié :

UserMediaRepositoryTest.java

Une image contenant texte, écran, capture d’écran

Description générée automatiquement

Quant aux services, les tests effectués vérifient que les opérations CRUD fonctionnent correctement.

### Execution des tests

Pour exécuter les tests, nous pouvons soit le faire avec « Maven », soit directement depuis VSCode.

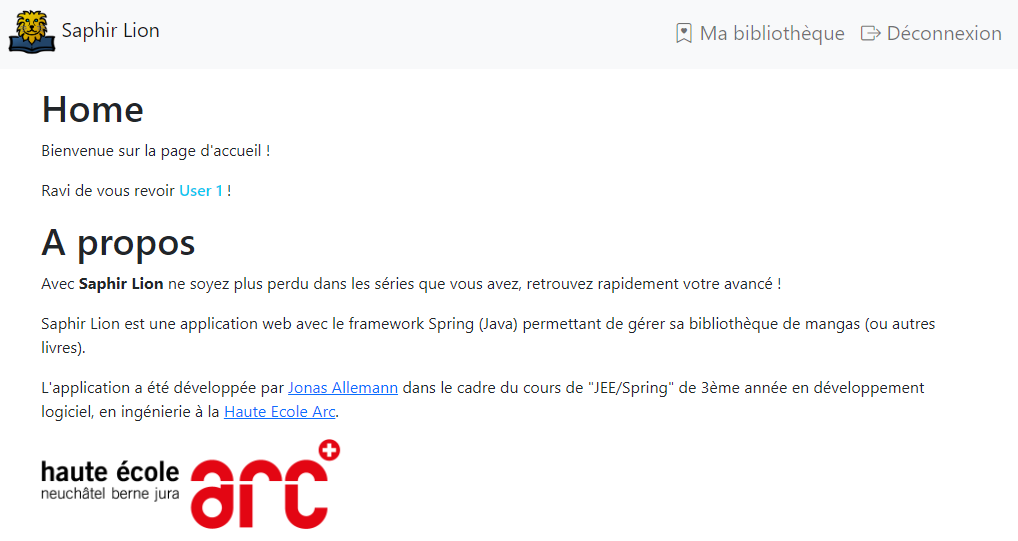
|  |  |
| --- | --- |
| Avec maven « [projet]/app/mvnw.cmd test » | Depuis VSCode, onglet "Testing" |
| [INFO] Results:  [INFO]  [INFO] Tests run: 20, Failures: 0, Errors: 0, Skipped: 0  [INFO]  [INFO] ------------------------------------------------------------------------  [INFO] BUILD SUCCESS  [INFO] ------------------------------------------------------------------------  [INFO] Total time: 15.493 s |  |

Comme nous pouvons le voir, tous les tests sont passés.

# Résultats

## Page d’accueil

La page d’accueil est très simple. Pour éviter d’avoir une page vide, elle comprend également le « A propos ».



Page du site 1 – L'accueil « <http://localhost:8080/> »

Si l’utilisateur est connecté, une phrase indique de qui il s’agit.

## Page de login

Si l’utilisateur n’est pas connecté, un formulaire de connexion s’affiche dès qu’il souhaite accéder à une page autre que l’accueil. Ce formulaire est géré par Spring Security. Bien qu’il soit possible de créer une vue personnalisée pour le traduire, j’ai laissé la page par défaut pour gagner du temps.

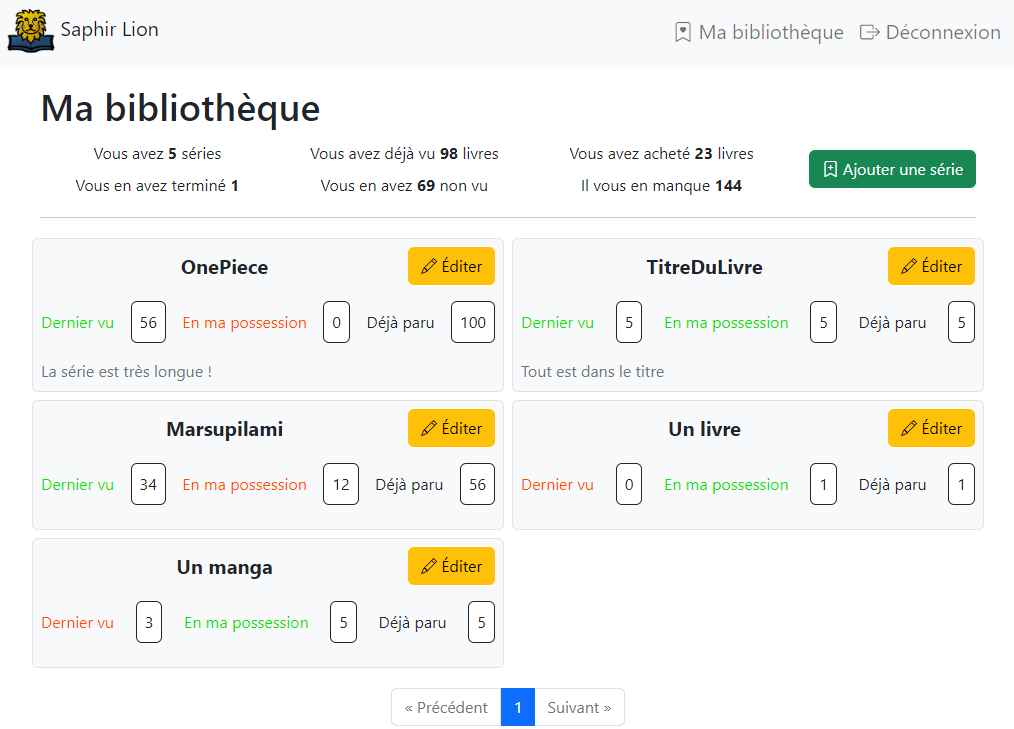


Page du site 2 – Connexion « <http://localhost:8080/login>»

Pour se déconnecter, l’URL est « <http://localhost:8080/logout> ».

## Ma bibliothèque

Quand l’utilisateur est connecté, il a accès à l’onglet « Ma bibliothèque ». Cette page liste tous les médias qu’il a enregistrés.



Page du site 3 – Ma bibliothèque « <http://localhost:8080/media> »

Sur cette page, nous y voyons quelques statistiques et tous les médias enregistrés sur notre compte.

Le label « dernier vu » devient vert s’il est supérieur à « en ma possession », et orange s’il est inférieur.  
Le label « en ma possession » est vert s’il est égal à « déjà paru », sinon orange.

Seule une dizaine de médias sont affichés par page.

## Ajout, édition d’un média

Lors de l’édition ou de l’ajout d’une série, le formulaire suivant s’affiche. Il s’agit du même pour les deux actions.

|  |  |
| --- | --- |
| Une image contenant texte  Description générée automatiquement  *Page du site 4 – Edition/ajout d'un média à sa collection* « <http://localhost:8080/media/add> » « <http://localhost:8080/media/edit/99> » | Le champ « média » permet de saisir le nom d’un média (manga, bd, etc.). Une liste des médias non présents dans sa collection est disponible. Il est également possible d’en créer un nouveau en saisissant un titre n’existant pas.   * Une fois qu’un média est enregistré, les autres utilisateurs auront la possibilité de le choisir dans la liste.   Les champs « dernier vu » et « en ma possession » doivent être inférieurs ou égaux au champ « déjà paru ». Ces 3 champs sont de type numérique et doivent être positifs.  Les champs « texte » sont soumis à une limite de caractère (relativement grande). |

## Page d’erreur HTTP

En cas d’erreur, une page adéquate s’affiche. Celle-ci récupère le code d’erreur (par exemple 403, 404, etc.) et affiche l’image du célèbre site « [http.cat](https://http.cat/)[[12]](#footnote-13) » liée à l’erreur.

* « http.cat » affichera l’image « 404 » s’il ne possède pas d’image pour le code spécifié.

Une image contenant texte, intérieur

Description générée automatiquement

Page du site 5 – Contenu visible en cas d'erreur http

## Autres résultats

Le site est responsif

Utilisation d’une version Spring plus récente que celle en cours

* + Correction du code devenu obsolète entre les deux versions

Validation et sécurité « backend » : Les valeurs des champs sont vérifiées avec Spring Validation ; un utilisateur ne peut modifier que ses entrées ; etc.

Un service d’authentification « maison » avait été implémenté, avant de passer à Spring Security. L’implémentation réalisée (« [AuthService](https://github.com/Ozurah-HES/SaphirLion/blob/main/app/src/main/java/ch/hearc/SaphirLion/service/impl/AuthService.java)[[13]](#footnote-14) ») est toujours disponible dans le projet, mais, a été marquée comme dépréciée.

# Améliorations

Diverses améliorations peuvent être effectuées sur le site. En voici quelques une :

* Actuellement, les entrées « catégorie » et « types » sont liées à un média, mais ne sont pas utilisables par l’utilisateur.
* La page de connexion est en anglais, et n’a pas la même interface que le reste du site.
* Les données saisies telles que le nombre de parutions devraient être le même pour un média entre tous les utilisateurs.
* Il y a des messages d’erreur si la saisie des champs est invalide, mais il n’y a pas de message de réussite (exemple : série correctement créée).

# Conclusion

Le site reprend tous les éléments mentionnés dans le cahier des charges, avec quelques ajouts.

L’utilisation de Bootstrap a permis de rendre le site « agréable » visuellement, et ce rapidement.

Concernant le volume de travail, l’implémentation du site a nécessité bien plus de temps que demandé. En effet, nous avons environ 27h de développement contre les 10h demandés. Avec de l’expérience dans le Framework, ce temps peut grandement être réduit, car il s’agit principalement de la recherche de bonne pratique et de solutions adéquate pour mettre en place telle ou telle fonctionnalité.

Pour conclure, le site est fonctionnel et correspond aux attentes. Pour moi, l’apprentissage de Spring fut intéressant et m’a permis d’explorer de nouvelles manières de développer un site web. Certes, le projet a pris plus de temps que nécessaire, mais je suis satisfait du résultat final et des connaissances acquises.

# Remerciement

Je remercie M. Loureiro pour la vérification du guide d’installation sur son PC et d’avoir remonté des bugs (qui ont été corrigés) sur le site.

# Sources

Les sources principales ayant été utilisées pour la réalisation du projet sont les suivantes :

De manière générale, StackOverflow pour la recherche d’erreur et les pistes à envisager.

Tutoriels de Baeldung

* Pagination
  + Pagination and Sorting using Spring Data JPA  
    <https://www.baeldung.com/spring-data-jpa-pagination-sorting>
  + Spring with Thymeleaf Pagination for a List  
    https://www.baeldung.com/spring-thymeleaf-pagination
* Partie validation :
  + Java Bean Validation Basics<https://www.baeldung.com/javax-validation>
  + Validation in Spring Boot  
    https://www.baeldung.com/spring-boot-bean-validation
* Page d’erreur
  + Spring Boot: Customize Whitelabel Error Page  
    https://www.baeldung.com/spring-boot-custom-error-page
* Réalisation des tests
  + Override Properties in Spring’s Test  
    <https://www.baeldung.com/spring-tests-override-properties>

Documentation

* La documentation officielle Spring  
  <https://docs.spring.io/spring-framework/docs/current/reference/html/>
* La documentation officielle Thymeleaf  
  <https://www.thymeleaf.org/documentation.html>

# Annexes

Le projet se trouve sur GitHub à l’adresse suivante : <https://github.com/Ozurah-HES/SaphirLion>

Les documents en annexes de ce rapport sont :

* Le cahier des charges
* Le guide d’installation
* Ces documents sont également accessibles sur le wiki du projet :  
  <https://github.com/Ozurah-HES/SaphirLion/wiki>

1. <https://github.com/Ozurah-HES/SaphirLion/issues> [↑](#footnote-ref-2)
2. <https://github.com/Ozurah-HES/SaphirLion/blob/main/app/pom.xml> [↑](#footnote-ref-3)
3. <https://github.com/Ozurah-HES/SaphirLion/blob/main/app/src/main/resources/templates/fragment/header.html> [↑](#footnote-ref-4)
4. <https://github.com/Ozurah-HES/SaphirLion/tree/main/app/src/main/java/ch/hearc/SaphirLion/model/seed> [↑](#footnote-ref-5)
5. <https://github.com/Ozurah-HES/SaphirLion/blob/main/app/src/main/resources/application.properties> [↑](#footnote-ref-6)
6. <https://github.com/Ozurah-HES/SaphirLion/blob/main/app/src/main/resources/application.properties> [↑](#footnote-ref-7)
7. <https://github.com/Ozurah-HES/SaphirLion/blob/f48dcbb8eac0b53d87a340e3449bf0aedb97d3fe/app/src/main/java/ch/hearc/SaphirLion/SecurityConfiguration.java#L36> [↑](#footnote-ref-8)
8. <https://github.com/Ozurah-HES/SaphirLion/tree/main/app/src/test/java/ch/hearc/SaphirLion/model> [↑](#footnote-ref-9)
9. <https://github.com/Ozurah-HES/SaphirLion/tree/main/app/src/test/java/ch/hearc/SaphirLion/repository> [↑](#footnote-ref-10)
10. <https://github.com/Ozurah-HES/SaphirLion/tree/main/app/src/test/java/ch/hearc/SaphirLion/service> [↑](#footnote-ref-11)
11. <https://github.com/Ozurah-HES/SaphirLion/tree/main/app/src/test/resources> [↑](#footnote-ref-12)
12. <https://http.cat/> [↑](#footnote-ref-13)
13. <https://github.com/Ozurah-HES/SaphirLion/blob/main/app/src/main/java/ch/hearc/SaphirLion/service/impl/AuthService.java> [↑](#footnote-ref-14)