|  |
| --- |
| 21.05.2023 |

|  |
| --- |
| HE-Arc – Ingénierie – ISC3il (2022-2023)  JEE/Spring II – cours 3293.1 |
| Saphir Lion & Topaz Lion |
| Rapport du projet (SP) |

|  |  |  |
| --- | --- | --- |
| 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> |
| Étudiant | Chappuis Sébastien – ISC3il-a | [sebastien.chappuis1@he-arc.ch](mailto:sebastien.chappuis1@he-arc.ch)  GitHub : <https://github.com/seba-chappuis> |
| Enseignant | Chèvre Sébastien Gérard Henri | [sebastien.chevre@he-arc.ch](mailto:sebastien.chevre@he-arc.ch) |
| Source du projet #1 (Semestre d’automne) | <https://github.com/Ozurah-HES/SaphirLion> | |
| Source du projet #2 (Semestre de printemps) | <https://github.com/Ozurah-HES/TopazLion> | |

Une image contenant clipart, illustration, dessin humoristique, art

Description générée automatiquement

Table des matières

[1 Introduction 2](#_Toc135509458)

[1.1 Description de Saphir Lion 2](#_Toc135509459)

[1.2 Description de Topaz Lion 2](#_Toc135509460)

[2 Modifications de Saphir Lion pour le projet du SP 3](#_Toc135509461)

[3 Schémas 4](#_Toc135509462)

[3.1 Schéma des composants 4](#_Toc135509463)

[3.2 Schéma des flux 5](#_Toc135509464)

[3.2.1 Saphir Lion 5](#_Toc135509465)

[3.2.2 Topaz Lion 11](#_Toc135509466)

[4 Routes des API 14](#_Toc135509467)

[4.1 Saphir Lion 14](#_Toc135509468)

[4.1.1 GET medias 14](#_Toc135509469)

[4.1.2 PUT media 14](#_Toc135509470)

[4.1.3 GET users 15](#_Toc135509471)

[4.1.4 GET user/medias 15](#_Toc135509472)

[4.1.5 PUT user/media 15](#_Toc135509473)

[4.1.6 DELETE user/media 16](#_Toc135509474)

[4.2 Topaz Lion 16](#_Toc135509475)

[4.2.1 GET media/last-response 16](#_Toc135509476)

[4.2.2 POST media 16](#_Toc135509477)

[4.2.3 POST media/fromANN 17](#_Toc135509478)

[5 Améliorations 18](#_Toc135509479)

[6 Conclusion 18](#_Toc135509480)

[7 Sources 19](#_Toc135509481)

[8 Annexes 19](#_Toc135509482)

# Introduction

Ce document fait état du semestre de printemps du projet Spring à réaliser dans le cadre de la troisième année de notre formation d’ingénieur en informatique en filière développement logiciel. Il s'agit d'un projet réalisé par groupe de 2 personnes.

Les membres du groupe sont :

* Allemann Jonas
* Chappuis Sébastien

Concernant le sujet, il était libre de choix. Il devait cependant contenir 2 projets Spring communiquant entre eux avec ActiveMQ, proposer des API REST et utiliser une API externe (ou un 3ème projet). Celui retenu consistait à améliorer le projet du semestre d’automne (**Saphir Lion**) en ajoutant une seconde application (**Topaz Lion**) pour la gestion des données communes à tous les utilisateurs (les médias).

Il s’agit d’un projet qui est évalué et permet de valider le module du semestre de printemps.

Ce présent document comprend :

* Une description des projets
* Les modifications des « données » apportées à Saphir Lion pour pouvoir correspondre aux demandes du semestre de printemps
* Les schémas de communication entre les différents projets
* Les routes API proposées par Saphir Lion et Topaz Lion.
* Quelques améliorations possibles
* Et pour finir, une conclusion

Un guide d’installation des différents projets (dépôt GIT et utilisation) est disponible en annexe.

## Description de Saphir Lion

L’application « **Saphir Lion** » qui a été développée en Spring durant le semestre d’automne est une application pour gérer ces médias (principalement des mangas, mais s’étend à tout autre média, comme des livres, des séries, etc.).

Dans sa version actuelle, elle dispose d’une base de données et d’une interface web pour les utilisateurs.

Les ajouts qui vont être faits sont les suivants :

* Ajout de routes **REST** pour :
  + Manipuler les médias des utilisateurs (création, édition, suppression)
  + Création/modification des médias (nom, image, nombre de parutions)
  + Obtention de tous les utilisateurs inscrits sur l’application
* Intégration **d’ActiveMQ** pour pouvoir communiquer avec l’application Topaz Lion

## Description de Topaz Lion

L’application « **Topaz Lion** » est une « extension » pour Saphir Lion. Son rôle sera de gérer la création et la modification des médias que les utilisateurs utilisent. Elle dispose d’une fonctionnalité supplémentaire pour la création des médias :

* Il est possible de récupérer le nom et l’image des mangas/animes en spécifiant un ID « ANN ».  
  « ANN » (Anime News Network) est une API fournissant un grand nombre de données sur les animes/mangas (en anglais).

Au début du travail, l’application n’existe pas. Les fonctionnalités implémentées sont :

* Routes REST pour créer des médias
* Récupération du nom et de l’image d’un média depuis ANN
* Intégration **d’ActiveMQ** pour pouvoir envoyer les modifications à l’application Saphir Lion.

# Modifications de Saphir Lion pour le projet du SP

En plus de l’ajout de la partie REST et ActiveMQ, quelques différences ont été apportées sur Saphir Lion pour s’adapter aux demandes pour le projet de printemps.

Modèles / Base de données

Les modifications apportées aux modèles sont les suivantes :

* Le nombre de publications (« nb\_published ») passe de « UserMedia » à « Media ».
* Une image est ajoutée au média. Cette image est chargée à partir d’une URL, qui est stockée dans « img\_url ».

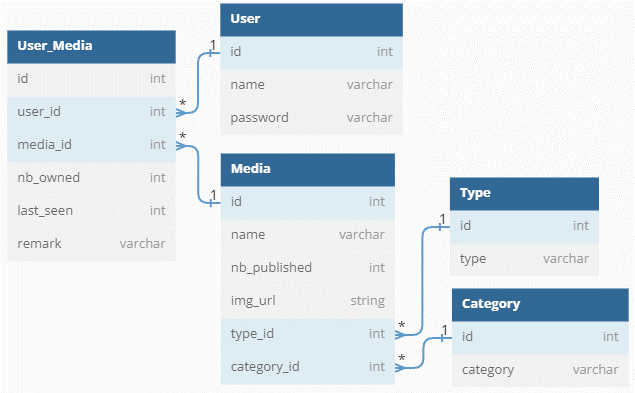


Diagramme 1 – Structure de la base de données

Contrôleurs

De nouveaux contrôleurs de type « REST » ont été ajoutés, ils sont disponibles dans le dossier « controller/REST »

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

Arborescence 1 – Contrôleurs ajoutés

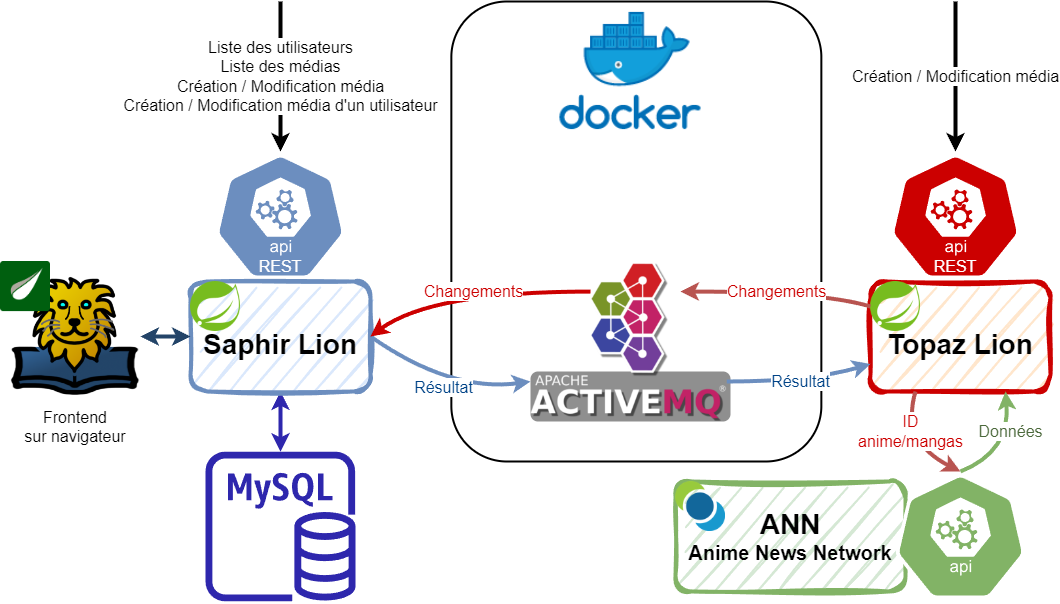
Repository & Services

De nouvelles fonctions ont été ajoutées, notamment pour obtenir tous les utilisateurs ou médias.

# Schémas

## Schéma des composants

Voici le schéma des composants du projet :



Nous y retrouvons :

* Les 2 projets Springs (**Saphir** & **Topaz Lion**)
* Les fonctionnalités des **API REST** des 2 projets
* La base de données liée à Saphir Lion
* Le frontend avec ThymeLeaf (sur navigateur) de Saphir Lion
* La communication **ActiveMQ** entre les 2 projets
* Le fait qu’ActiveMQ transite via **Docker**
* Le fait que Topaz Lion utilise une API externe (celle d’**ANN**).
* La base de données et le frontend de Saphir Lion faisaient partie du projet du semestre d’automne. Des changements mineurs ont été apportés pour être compatibles avec la version du semestre de printemps.

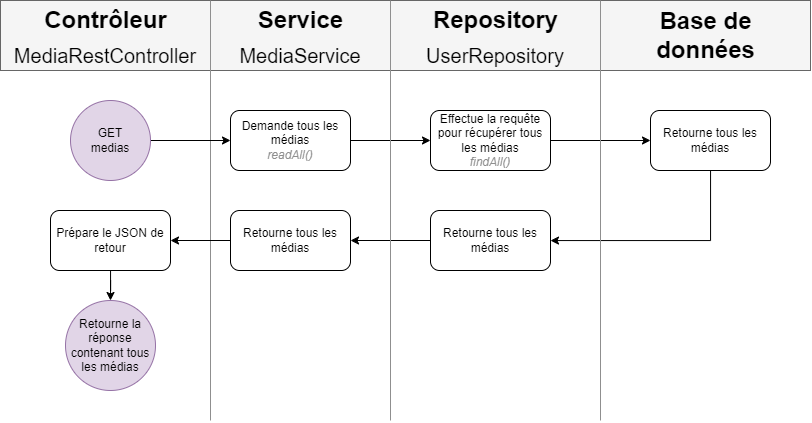
## Schéma des flux

Les différents schémas de flux suivis lors de requêtes REST sont indiqués dans ce chapitre.

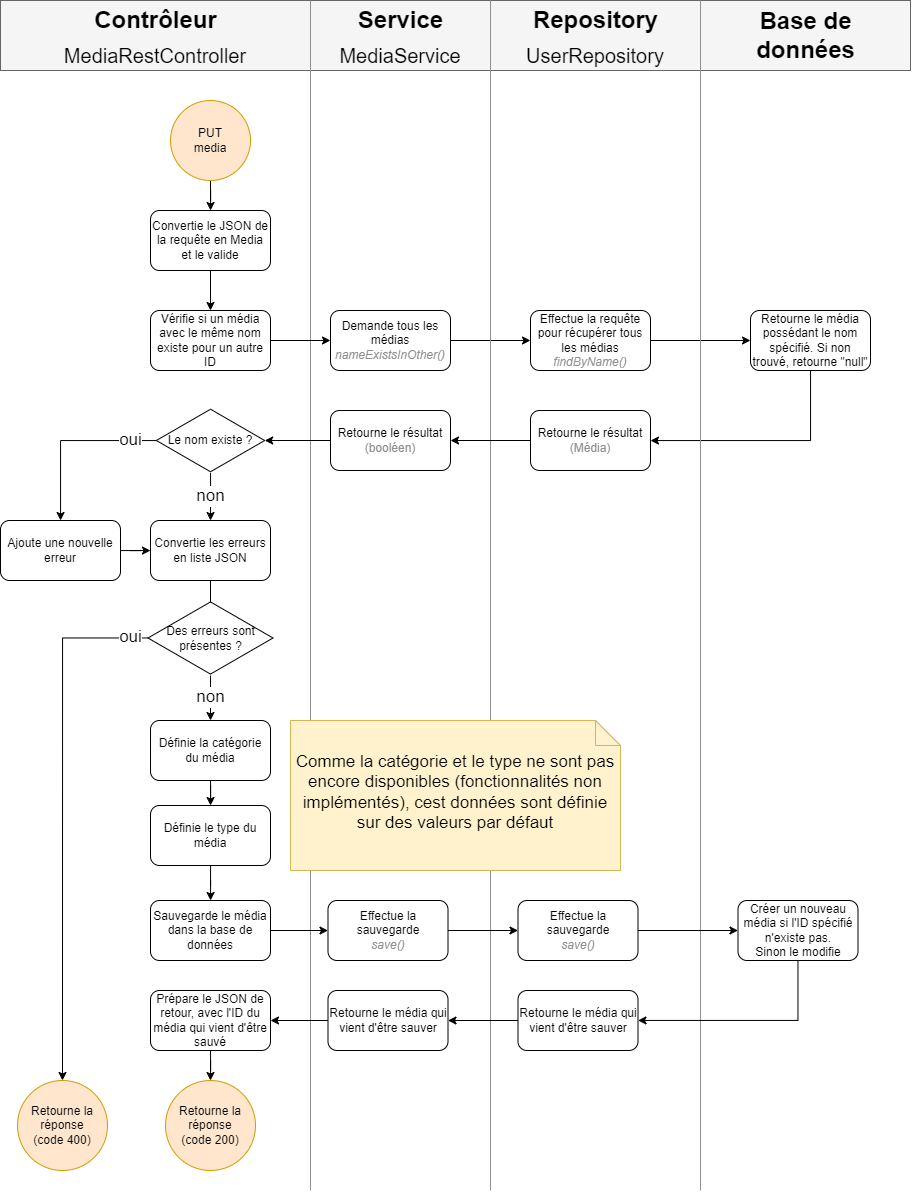
* Le titre de niveau 4 indique le point de départ du schéma.
* Si le point de départ des flux est une route API, il comprend le verbe et le suffixe de la route. Par exemple, si la bulle indique « **GET users** », la route complète sera : « localhost:[PORT]/api/**users** » et il faudra l’envoyer en « **GET** ».
  + Saphir Lion utilise le port 8080  
    Topaz Lion utilise le port 9090
* Les couleurs des points d’entrées correspondent à celles utilisées par Insomnia.
* Les ronds bleus correspondent à des queues d’ActiveMQ.

### Saphir Lion

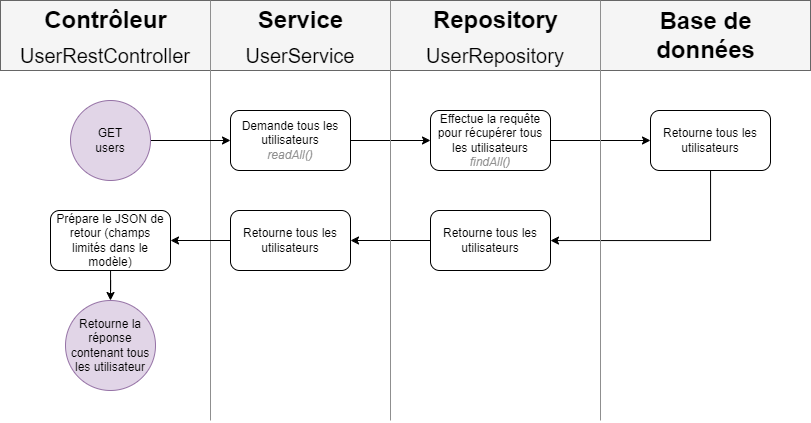
#### GET medias



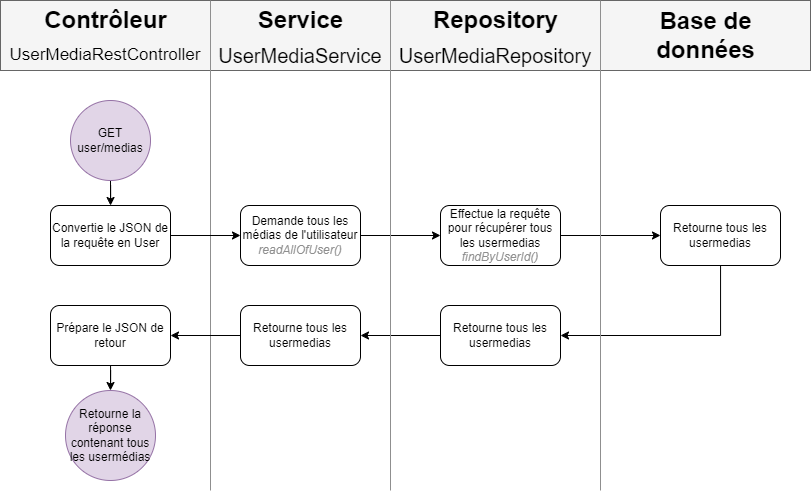
#### PUT media



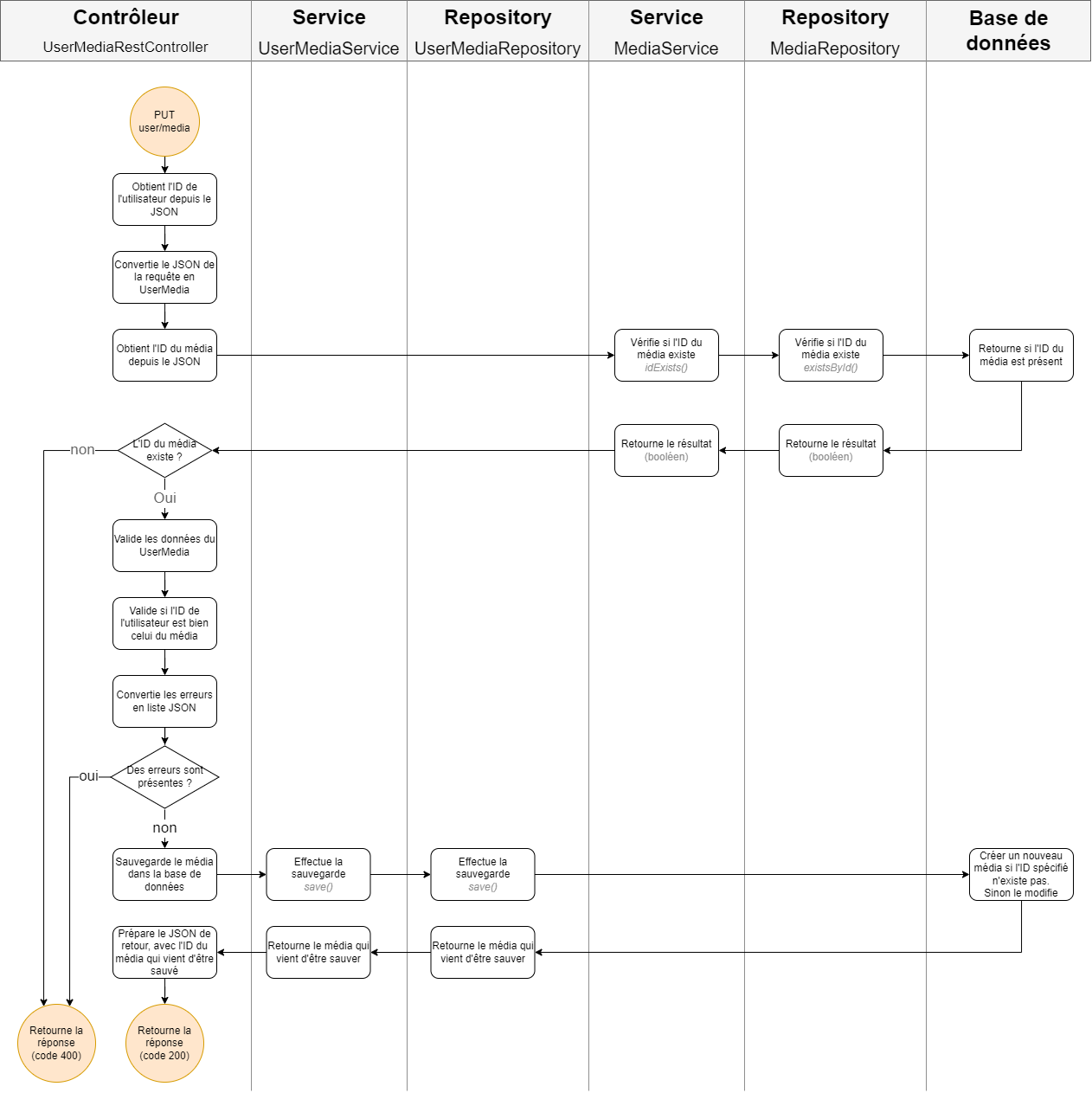
#### GET users



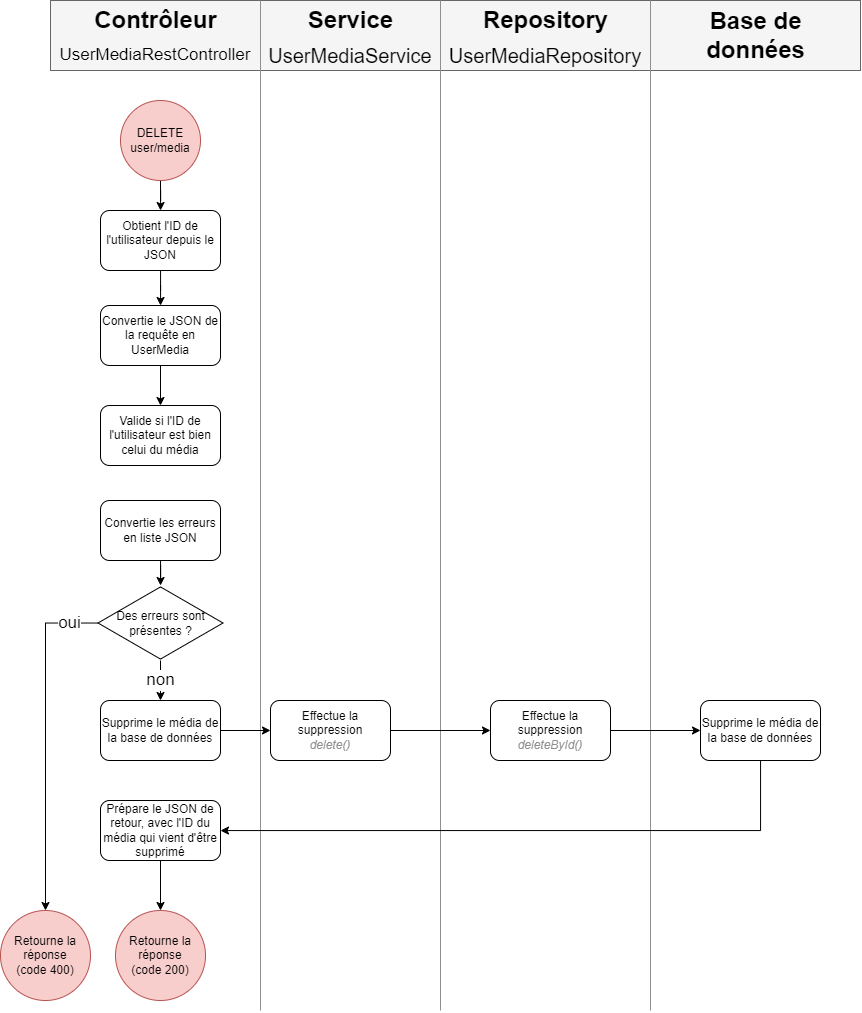
#### GET user/medias



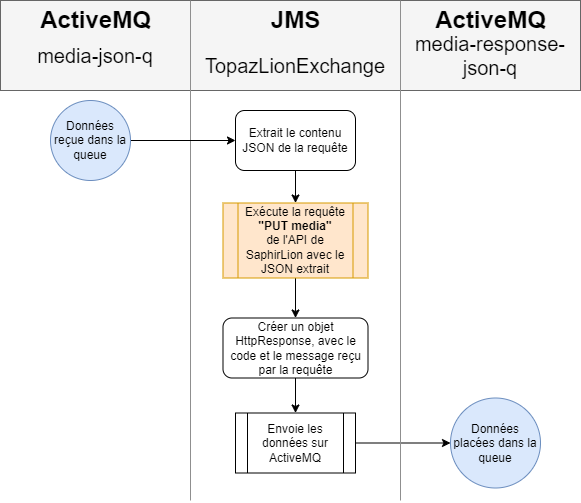
#### PUT user/media



#### DELETE user/media

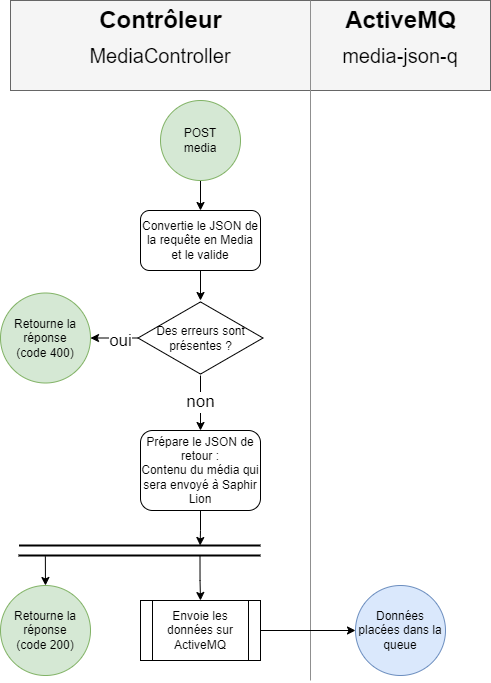


#### ActiveMQ

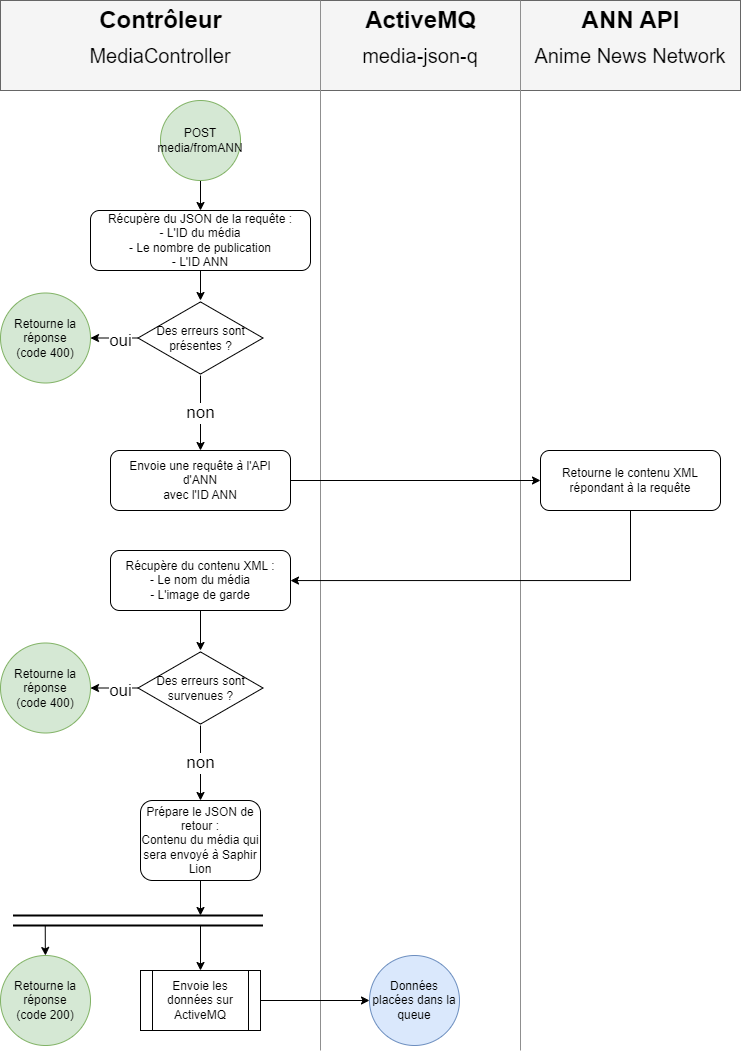


### Topaz Lion

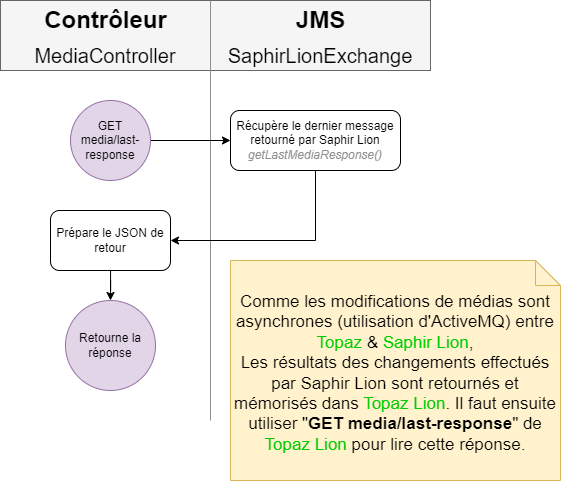
#### POST media



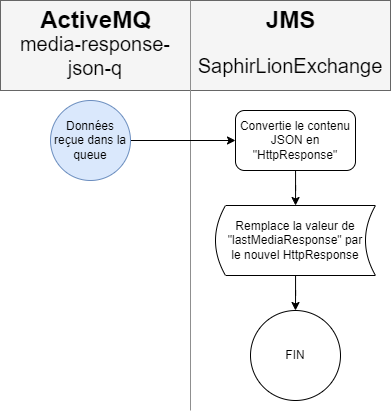
#### POST media/fromANN



#### GET media/last-response



#### ActiveMQ



# Routes des API

Les différentes routes disponibles pour les API de Saphir Lion et Topaz Lion sont indiquées dans ce chapitre, chaque route contient les données à envoyer (corps), et la réponse (tous deux au format JSON).

La route à envoyer correspond au nom du titre de niveau 3. Par exemple, si le nom est « **GET users** », la route complète est : « localhost:[PORT]/api/**users** » et il faut l’envoyer en « **GET** ».

* Saphir Lion utilise le port 8080  
  Topaz Lion utilise le port 9090

## Saphir Lion

### GET medias

#### Corps

Aucun.

#### Réponse

Liste JSON de médias contenant chacun :

* id
* name
* type (objet JSON)
  + id
  + type
* category (objet JSON)
  + id
  + category
* nbPublished
* imgUrl

### PUT media

#### Corps

Objet JSON d’un média :

* id (si non-spécifié ou null, va créer un nouveau média, sinon va modifier un média existant)
* name
* type (objet JSON)
  + id
* category (objet JSON)
  + id
* nbPublished
* imgUrl
* « type » et « category » sont optionnels. De toute façon, dans la version actuelle du projet, ils ne sont pas traités et remplacés par une valeur par défaut

#### Réponse

Liste contenant le message de succès ou liste contenant les erreurs.

### GET users

#### Corps

Aucun.

#### Réponse

Liste JSON d’utilisateurs, contenant chacun :

* id
* username
* userMediaIds (liste contenant les IDs des UserMedia)

### GET user/medias

#### Corps

Objet JSON contenant :

* id (ID d’un utilisateur)

#### Réponse

Liste JSON de UserMedias (pour l’utilisateur spécifié), contenant chacun :

* id
* nbOwned
* lastSeen
* remark
* userId
* mediaId
* nbPublishedGreaterThanNbOwned
* nbPublishedGreaterThanLastSeen

### PUT user/media

#### Corps

Objet JSON :

* id (si non-spécifié, va créer un UserMedia, sinon va modifier un UserMedia existant)
* nbOwned
* lastSeen
* remark
* mediaId
* userId

#### Réponse

Liste JSON de messages d’erreurs (si échec) ou d’un message de succès.

Exemple : ["Edited item Id: 44"]

### DELETE user/media

#### Corps

Objet JSON contenant :

* id (ID du UserMedia)
* userId

#### Réponse

Liste JSON de messages d’erreurs (si échec) ou d’un message de succès.

Exemple 1 : ["Ce média ne vous appartient pas"]  
Exemple 2 : ["Deleted item ID: 44"]

## Topaz Lion

### GET media/last-response

#### Corps

Aucun.

#### Réponse

Objet JSON contenant :

* last-media-response (objet JSON)
  + id
  + status (statut HTTP)
  + message (liste contenant messages d’erreurs/de succès)
* L’id correspond au nombre de messages qui ont été envoyés par Saphir Lion. Un ID à 4 signifie donc qu’il y a eu 3 réponses avant celle obtenue actuellement.

### POST media

#### Corps

Objet JSON d’un média, contenant :

* id (si non-spécifié ou null, va créer un nouveau média, sinon va modifier un média existant)
* nbPublished
* name

#### Réponse

Objet JSON du média qui sera créé ou modifié :

* media-will-be-add (objet JSON)
  + id
  + name
  + nbPublished
  + imgUrl
* information (message indiquant la route à emprunter pour connaitre la réponse de Saphir Lion)

### POST media/fromANN

#### Corps

Objet JSON contenant :

* id (si non-spécifié, va créer un média, sinon va modifier un média existant)
* ann-id (ID d’un média dans la base de données ANN)
* nb-published

#### Réponse

Si le corps est valide, un objet JSON du média qui sera créé ou modifié :

* media-will-be-add
  + id
  + name (récupéré de l’API ANN)
  + nbPublished
  + imgURL (récupéré de l’API ANN)
* information (message indiquant la route à emprunter pour connaitre la réponse de Saphir Lion)

Sinon : liste de messages d’erreurs

# Améliorations

Améliorations Saphir Lion (SA)

Les améliorations suggérées pour Saphir Lion dans le rapport du semestre d’automne sont toujours d’actualités.

Maintenant que Saphir Lion dispose de routes REST, il serait intéressant de les intégrer au site utilisateur (frontend) et ne plus utiliser les anciens contrôleurs. Le site pourrait même être dissocié de Saphir Lion pour séparer le backend et le frontend.

Modifications via l’API

Les requêtes de modification proposées par les deux API sont de type « PUT », donc exécuter l’une de ces requêtes nécessitera de spécifier toutes les valeurs (même celles qui ne changent pas) sinon elles seraient « écrasées » par des valeurs par défaut. Il serait intéressant d’ajouter des requêtes de type « PATCH » pour pouvoir ne modifier que certaines informations.

Ajout d’un média avec Topaz Lion

Comme Topaz Lion envoie les demandes de changement à Saphir Lion au travers d’ActiveMQ, il est nécessaire d’appeler une seconde requête « /media/last-responses » pour s’assurer que les données ont correctement été mises à jour. Il serait intéressant de fusionner la réponse de modification avec « last-responses ».

Création & Edition

Actuellement, l’édition et la création sont gérées par Spring de manière unique, si l’ID spécifié existe, l’élément est écrasé par les nouvelles valeurs, sinon une nouvelle entrée est créée.

Il pourrait être intéressant de dissocier ce comportement afin d’avoir le principe de « responsabilité unique ». Ce comportement serait appréciable notamment pour les routes de l’API, qui actuellement informe juste avec une « édition effectuée » autant pour l’ajout ou la modification.

# Conclusion

Saphir Lion dispose à présent d’une API REST offrant quelques routes. Le site de Saphir Lion est toujours fonctionnel et permet donc aux utilisateurs de modifier leurs données de deux manières : par le site, ou par requêtes avec l’API.

* L’API est moins sécurisée que le site, étant donné qu’il n’a pas de système d’authentification. N’importe qui peut modifier n’importe quelle donnée, même si elle appartient à un autre utilisateur.

Une communication bidirectionnelle entre l’application Saphir Lion et Topaz Lion est présente. Cette communication se fait avec **ActiveMQ** au travers d’un **Docker**.

Topaz Lion propose une API REST relativement simple. Celle-ci permet de créer ou modifier des médias de l’application Saphir Lion. De plus, il offre une nouvelle fonctionnalité pour éditer les médias : la récupération des données (nom et image de couverture) d’une **API externe (Anime News Network)**.

# Sources

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

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

Tutoriels de Baeldung

* REST with Spring Tutorial  
  <https://www.baeldung.com/rest-with-spring-series>
* Spring Remoting with JMS and ActiveMQ   
  <https://www.baeldung.com/spring-remoting-jms>
* XML Serialization and Deserialization with Jackson  
  <https://www.baeldung.com/jackson-xml-serialization-and-deserialization>
* Mapping Nested Values with Jackson  
  <https://www.baeldung.com/jackson-nested-values>

Documentation

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

Anime News Network (API externe)

* Encyclopedia API  
  <https://www.animenewsnetwork.com/encyclopedia/api.php>

# Annexes

Les projets (Saphir Lion & Topaz Lion) se trouvent sur GitHub aux adresses suivantes :

* <https://github.com/Ozurah-HES/SaphirLion/tree/SP>
* <https://github.com/Ozurah-HES/TopazLion>

Les documents en annexes de ce rapport sont :

* Le guide d’installation (SP)
* Ces documents sont également accessibles sur le wiki du projet :  
  [*https://github.com/Ozurah-HES/SaphirLion/wiki*](https://github.com/Ozurah-HES/SaphirLion/wiki)[*https://github.com/Ozurah-HES/TopazLion/wiki*](https://github.com/Ozurah-HES/TopazLion/wiki)