Animanga

Rapport TPI et documentation technique

mai – juin 2020

Travail Pratique Individuel (TPI)

- Tanguy Cavagna
- Maître d'apprentissage : Pascal Bonvin

https://flask.tcsandbox.ch

Table des matière

- Animanga
  - Table des versions
  - Préambule
  - Introduction
  - Résumé de l'énoncé
  - Organisation
  - Livrable
  - Matériel et logicles à disposition
  - Descriptif complet du projet
    - Sitemap
    - Description succinte du contenu des pages du site
  - Méthodologie
    - 1. S'informer
    - 2. Planifier
    - 3. Décider
    - 4. Réaliser
    - 5. Contrôler
    - 6. Évaluer
  - Planification
    - Product backlog
    - Diagramme de Gantt
  - Implémentation
    - Base de données
    - Structure
    - Classes (Python)
    - API interne
  - Librairies et outils externes
    - Pip et NPM
    - Flask
    - Jinja
    - Flask-Login
    - Flask-Swagger
    - MySQL Connector/Python
    - SASS
    - Swagger
    - jQuery UI
    - ESLint
    - Pylint
    - Git
  - Analyse des fonctionnalités majeurs
    - Un CRUD complet permet de gérer une entrée manga de la bibliothèque
    - La base de données locale sqlite3 est synchronisée de façon unidirectionnelle avec la base de données d'un serveur mysql
    - Les données JSON de github sont importées dans la base de données locale
    - Le service http utilise Python Flask
    - Le planning réel est documenté et comparé au planning prescrit
    - Le projet est publié sur github et une url est communiqué

        - Le projet Python contient au moins une classe (python objet) conçu par le candidat
    - Plans de test et tests
        - Périmètre des tests
        - Environnement
        - Scénarios
        - Évolution des tests
    - Bibliographie
    - Glossaire
        - Termes métier
        - Termes technique
    - Conclusion
        - Difficultés majeures rencontrées
        - Améliorations possibles
        - Bilan personnel
        - Remerciements

Table des versions

N° de version	Date	Auteur	Modifications
1.0	2020-06-09	Tanguy Cavagna >	Version finale de la documentation du TPI

Préambule

Toute cette documentation a été rédigée en **Markdown**. PDF que vous avez entre les mains est généré automatiquement grâce au logiciel d'édition pour fichier Markdown : **Typora**. J'ai fait un script Bash servant à fusionner les différents fichiers PDF nécessaires à la composition finale de ce rapport. Par conséquent, il se peut que la mise en page soit quelque peu bancale. C'est pourquoi je vous invite chaleureusement à lire tout ce rapport sur le site **https://animanga.readthedocs.io/fr/latest/**. De plus, mon TPI est hébergé sur un serveur accessible à l'adresse **https://flask.tcsandbox.ch**.

Introduction

Ce projet a été réalisé dans le cadre du *Travail Pratique Individuel* (TPI) durant la session de mai - juin 2020. Il a pour but de valider les compétences acquises tout au long de la formation *Informaticien CFC* de l'école du CFPT-Informatique au Petit-Lancy.

Animanga est une application web, écrite en Python, permettant aux utilisateurs de faire leur propre bibliothèque d'anime. Pour ce faire, l'utilisateur a la possibilité de créer ses propres listes afin de correctement organiser sa bibliothèque, mettre des animes en tant que favoris, et rechercher les animes qu'il voudrait ajouter à sa collection directement depuis cette application.

Résumé de l'énoncé

Les informations suivantes sont éxtraites du cahier des charges du TPI.

Organisation

Éléve	Maître d'apprentissage	Experts
Tanguy Cavagna >	Pascal Bonvin <**edu-bonvinp@edu ge.ch**>	Nicolas Terrond <**nicolas.terrond@sig-ge.ch**> Robin Bouille <**robin.bouille@gmail.com**>

Livrable

Pour les experts et le formateur	Pour le formateur
Planning réel détaillé du projet	Accès au GitHub
Documentation du projet contenant le code source au format PDF	
Journal de bord	
Résumé du TPI (1 page A4)	

Matériel et logicles à disposition

- Un PC standard école, 2 écrans
- Pycharm
- Netbeans
- Suite office
- MySQl, Sqlite3, Flask, connexion internet

Descriptif complet du projet

Lorsqu'il s'agit de réaliser un site web, la tradition de l'école d'informatique encourage l'utilisation de PHP et Mysql. Dans le cas de ce diplôme, il s'agit de réaliser un site local avec Python Flask et de gérer les données dans une base de données Sqlite3. De plus la base de donnée locale est synchronisée unidirectionnelement avec une base de donnée Mysql sur un serveur distant. L'ambition de ce projet est de démontrer qu'il est possible de créer un application WEB sans passer par l'installation d'un serveur apache et d'une base données Mysql. A noter que le candidat est un élève brillant qui maîtrise le langage Python.

Les données initiales qui permettront de remplir la base de données sont accessibles sur github au format json (**https://github.com/manami-project/anime-offline-database**). Elles devront être converties et synchronisées dans la base de données locale qui reste à déterminer.

Sitemap

Description succinte du contenu des pages du site

- La page index permet d'avoir le champ de recherche, un bouton "aléatoire", ainsi que les favoris de l'utilisateur et son flux d'activité si connecter.
- La page à propos contient toutes les informations concernant le site ainsi que les librairies utilisées.
- La page connexion permet simplement de se connecter.
- La page inscription, de s'inscrire.
- La page anime / manga permet de voir les informations / actions sur un anime / manga sélectionner (rediriger sur cette page lorsque l'on clique sur un anime / manga sur la page index après une recherche).
- La page aide contient l'aide du site.
- La page profil contient les information de l'utilisateur ainsi que les listes personnelles, dans lesquelles des animes / manga peuvent être ajoutés, ainsi que leur contenu.
- La page favoris permet de modifier l'ordre des animes / manga mit en favoris ainsi que de les retirer de la liste des favoris.
- La page liste contient un CRUD sur les listes personnelles de l'utilisateur.

Méthodologie

Pour pouvoir planifier correctement ce projet, j'ai décidé d'utiliser la méthode en 6 étapes, décrite ci-dessous :

1. S'informer

La première étape est utile non seulement pour comprendre le projet dans son ensemble mais également pour se rendre compte de toutes les fonctionnalités nécessaires. Elle permet aussi d'éclaircir tous les points flous de l'énoncé.

2. Planifier

Le fait de planifier le projet permet de séparer les tâches, de lister et de définir les priorités. Ses dernières sont les suivantes : 🚫 *Bloquant* , 💥 *Critique* , ❗ *Important* , ❓ *Secondaire* .

Pour représenter le planning, j'ai choisi d'utiliser un diagramme de Gantt. Ce type d'outil de gestion permet de visualiser très correctement la progression quotidienne du projet ainsi que les différences entre ce qui a été prévu et la réalité.

3. Décider

S'il reste des points en suspens, c'est le dernier moment pour prendre des décisions (les éclaircir, les laisser de côté, les remettre à plus tard, etc.) afin de pouvoir ensuite "se jeter à l'eau" !

4. Réaliser

Cette partie permet de commencer le projet en tant que tel : implémenter toutes les fonctionnalités à développer et rédiger la documentation.

5. Contrôler

Cette étape invite à tester chacune des fonctionnalités indépendamment les unes des autres pour vérifier leur fonctionnement dans différents cas d'usage.

Une fois l'application terminée, il s'agit de tester son bon fonctionnement sur plusieurs navigateurs différents pour bien être certain que tout se déroule comme prévu dans l'import que cas d'utilisation.

6. Évaluer

Une fois toutes les étapes précédentes achevées, il s'agit de se lancer dans ce qui peut sembler le plus complexe ; une rétrospective de tout ce qui a été fait avec un regard critique afin de déceler les points à améliorer par la suite.

Pour ce faire, une section est prévue dans le rapport final dans laquelle sont répertoriés les problèmes rencontrés ainsi que les solutions trouvées pour ces derniers.

Pour que je puisse avoir une évaluation complète du projet, le rapport final se termine par une conclusion qui sert de bilan final au projet.

Planification

Product backlog

Nom	**S1 : Inscription à Animanga**
Description (*user story*)	En tant qu'utilisateur non connecté, je peux me créer un compte afin de pouvoir accéder au site.
Critère d'acceptation	Les tests *1.1* et *1.10* passent.
Priorité	🚫 Bloquant

Nom	**S2 : Connexion à Animanga**
Description (*user story*)	En tant qu'utilisateur non connecté, je peux me connecter afin de pouvoir accéder au site.
Critère d'acceptation	Les tests *2.1* et *2.6* passent.
Priorité	🚫 Bloquant

Nom	**S3 : Importation des animes**
Description (*user story*)	En tant qu'utilisateur connecté, je peux écraser les animes avec un nouveau set de données.
Critère d'acceptation	Le test *3.1* passe.
Priorité	💥 Critique

Nom	**S4 : Rechercher des animes**
Description (*user story*)	En tant qu'utilisateur connecté, je peux effectué une recherche afin d'ajouter des animes dans mes listes personnelles ou de les mettre en tant que *favoris* .
Critère d'acceptation	Les tests *4.1* et *4.2* passent.
Priorité	💥 Critique

Nom	**S5 : Affichage de la carte d'un anime**
Description (*user story*)	En tant qu'utilisateur connecté, je peux cliquer sur le titre d'un anime présent dans les résultats de ma précédente recherche afin d'accéder à ses informations.
Critère d'acceptation	Le test *5.1* passe.
Priorité	❗ Important

Nom	**S6 : Mise à jour d'un anime**
Description (*user story*)	En tant qu'utilisateur connecté, je peux mettre à jour le statut, l'appartenance à une liste personnel ainsi que le statut de favoris d'un anime.
Critère d'acceptation	Les tests *6.1* à *6.6* passent.
Priorité	❗ Important

Nom	**S7 : Affichage du profile**
Description (*user story*)	En tant qu'utilisateur connecté, je peux avoir accès à ma page de profile afin de pourvoir voir les statistiques et favoris. Il est également possible de voir la page de profile d'autre utilisateur du site.
Critère d'acceptation	Le test *7.1* passe.
Priorité	❗ Important

Nom	**S8 : Affichage des listes**
Description (*user story*)	En tant qu'utilisateur connecté, je peux avoir accès à ma page de listes afin de voir toutes mes listes et leur contenu. Il est également possible de voir les listes d'autre utilisateur du site.
Critère d'acceptation	Le test *8.1* passe.
Priorité	❗ Important

Nom	**S9 : Gestion des listes**
Description (*user story*)	En tant qu'utilisateur connecté, je peux gérer mes propres listes pour en ajouter, en supprimer, ou modifier leur nom.
Critère d'acceptation	Les tests *9.1* et *9.2* passent.
Priorité	❗ Important

Nom	**S10 : Affichage des favoris**
Description (*user story*)	En tant qu'utilisateur connecté, je peux avoir accès à ma page favoris afin de voir tout mes favoris. Il est également possible de voir les favoris d'autre utilisateur du site.
Critère d'acceptation	Les test *10.1* et *10.2* passent.
Priorité	❗ Important

Nom	**S11 : Gestion des favoris**
Description (*user story*)	En tant qu'utilisateur connecté, je peux organiser l'ordre de mes favoris selon mes envies.
Critère d'acceptation	Les tests *11.1* et *11.2* passent.
Priorité	❗ Important

Nom	**S12 : Affichage de la *landing page***
Description (*user story*)	En tant qu'utilisateur non connecté, je n'ai ni accès aux animes ni aux listes. La barre de navigation m'affiche un lien pour me connecter et un autre pour m'inscrire.
Critère d'acceptation	Le test *12.1* passe.
Priorité	❓ Secondaire

Nom	**S13 : Utilisation d'un git**
Description (*user story*)	En tant que développeur, je dois pouvoir faire du versionnage de code source et pouvoir accéder à un dépôt distant Github.
Critère d'acceptation	Le dépôt git local est configurer correctement et le lien sur le dépôt distant à été bien fait.
Priorité	🚫 Bloquant

Nom	**S14 : Configuration de la base de données**
Description (*user story*)	En tant que développeur, je dois pouvoir utilisé une base de données Sqlite3 et MySQL ayant un modèle identique à celui donné dans l'énoncé. Pour ce faire, j'ai une classe Python me permettant de faire des requêtes sur la base Sqlite3 et une autre classe me permettant de faire des requêtes sur la base MySQL. J'ai aussi un dump de la structure de la base MySQL dans les fichiers statiques de mon application.
Critère d'acceptation	Les tables `animes`, `status`, `type`, `url`, `list`, `user`, `list_has_anime`, `user_has_list` et `user_has_favorites` ont bien été crées et sont utilisable par les contrôleurs dédiés.
Priorité	🚫 Bloquant

Nom	**S15 : Synchronisation MySQL Sqlite3**
Description (*user story*)	En tant que développeur, je dois pouvoir synchroniser les bases MySQL et Sqlite3 unidirectionellement pour créer un backup sur serveur distant.
Critère d'acceptation	Le test *14.1* passe.
Priorité	❓ Secondaire

Nom	S16 : Configuration Flask
Description (*user story*)	En tant que développeur, je dois configurer l'application Flask afin d'avoir un site hébergé en local et pouvoir communiquer avec la base de données Sqlite3.
Critère d'acceptation	Une page web s'affiche sur l'url `localhost:5000`.
Priorité	🚫 Bloquant

Nom	S17: Vérifications syntaxique
Description (*user story*)	En tant que développeur, je peux lancé la commande `npm run lint` pour vérifier la syntaxe, basé sur le preset Airbnb, des fichiers JavaScript, et la commande `python3 -m pylint --output-format=colorized` pour vérifier la syntaxe des fichiers Python, basé sur les conventions PEP8.
Critère d'acceptation	Les test *12.1* et *12.2* passent.
Priorité	🚫 Bloquant

Nom	S18: Affichage des activités
Description (*user story*)	En tant qu'utilisateur connecté, je vois mon fil d'actualité contenant le temps écoulé depuis l'ajout d'un favoris et l'ajout d'un anime dans une liste.
Critère d'acceptation	Le test *15.1* passe.
Priorité	❓ Secondaire

Diagramme de Gantt

Jour	J1 lu.25	J2 ma.26	J3 me.27	J4 je.28	J5 ve.29	J6 ma.2	J7 me.3	J8 je.4	J9 ve.5	J10 lu.8	J11 ma.9
Lecture de l'énoncé											
Rédaction backlog											
Rédaction scénarios											
Rédaction planning											
S13 : Utilisation d'un git											
S14 : Configuration de la base de données											
S16 : Configuration de Flask											
S3 : Importation des données											
S1 : Inscription à Animanga											
S2 : Connexion à Animanga											
S12 : Affichage de la landing page											
S4 : Rechercher des animes											
S5 : Affichage de la carte de l'anime											
S6 : Mise à jour de l'anime											
S7 : Affichage du profile											
S10 : Affichage des favoris											
S8 : Affichage des listes											
S9 : Gestion des listes											
S11 : Organisation											

Jour	J1 lu.25	J2 ma.26	J3 me.27	J4 je.28	J5 ve.29	J6 ma.2	J7 me.3	J8 je.4	J9 ve.5	J10 lu.8	J11 ma.9
des favoris											
S18 : Affichage des activités											
S15 : Synchronisation MySQL Sqlite3											
S17 : Vérifications syntaxique											
Tests en profondeur et corrections des bugs											
Tenir à jour la documentation											
Résumé du TPI											
Finalisation / Impression											
Tenue du journal de bord											

Planification prévisionnelle Planification réelle Planification imprévue

Implémentation

Base de données

Le projet de TPI contient 2 types de base de données différents. La base principale est en Sqlite3 et la base de backup distante est en MySQL. Ces deux bases doivent pouvoir stocker les utilisateurs, les animes, les listes personnelles des utilisateurs, ainsi que les favoris des utilisateurs.

Voici le modèle réalisé :

Dictionnaire de données

anime

Colonne	Type	Null
idAnime	int(11)	Non
title	varchar(200)	Non
#type	int(11)	Non
episodes	int(3)	Non
#status	int(11)	Non
picture	varchar(200)	Non
thumbnail	varchar(200)	Non
synonyms	text	Oui
modificationDate	timestamp	Non

status

Colonne	Type	Null
idStatus	int(11)	Non
nameStatus	varchar(20)	Non
modificationDate	timestamp	Non

type

Colonne	Type	Null
idType	int(11)	Non
nameType	varchar(10)	Non
modificationDate	timestamp	Non

url

Colonne	Type	Null
idUrl	int(11)	Non
urlName	varchar(45)	Non
#idAnime	int(11)	Non
isRelation	tinyint(1)	Non
modificationDate	timestamp	Non

list

Colonne	Type	Null
idList	int(11)	Non
nameList	varchar(45)	Non
modificationDate	timestamp	Non

list_has_anime

Colonne	Type	Null
idListHasAnime	int(11)	Non
#idList	int(11)	Non
#idAnime	int(11)	Non
modificationDate	timestamp	Non

user

Colonne	Type	Null
idUser	int(11)	Non
emailUser	varchar(45)	Non
password	varchar(45)	Non
nicknameUser	varchar(45)	Non
modificationDate	timestamp	Non

user_has_list

Colonne	Type	Null
idUserHasList	int(11)	Non
#idUser	int(11)	Non
#idList	int(11)	Non
modificationDate	timestamp	Non

user_has_favorite

Colonne	Type	Null
idUserHasFavorite	int(11)	Non
#idUser	int(11)	Non
#idAnime	int(11)	Non
orderId	int(11)	Non
modificationDate	timestamp	Non

Structure

- **/** : Contient tout les fichiers de configuration ainsi que les routes
- **/docs** : Contient la documentation de tout mon projet
  - **/docs/compiled** : Contient la documentation compilée
  - **/docs/scripts** : Contient tout les scripts permettant de créer un fichier unique contenant toute la documentation
- ***/node_modules*** : Contient les dépendances JavaScript (géré par NPM)
- **/packages** : Contient les fichiers `python`
  - **/packages/controllers** : Contient les contrôleurs
  - **/packages/models** : Contient les modèles de données
- **/static** : Contient les fichiers statiques
  - **/static/css** : Contient les fichiers `css`
  - **/static/database** : Contient la base de données sqlite3
  - **/static/fonts** : Contient les polices
  - **/static/img** : Contient les images / svg
  - **/static/js** : Contient les fichiers `javascript`
  - **/static/json** : Contient le fichier `json` contenant tout les animes
  - **/static/scss** : Contient les fichiers `scss`
  - **/static/swagger-components** : Contient les composants pour swagger
- **/templates** : Contient les pages à afficher
  - **/templates/layouts** : Contient le layout générique ainsi que les fichier pouvant être inclus
- **/tests** : Contient le fichier `html` pour Katalon Recorder

Classes (Python)

Pour correctement interagir avec le code Python, et comme demandé dans le point **A20**, j'ai écrit les classes suivantes :

\packages\controllers\ActivitiesController

Classe permettant le contrôle des activités de l'utilisateur

\packages\controllers\AnimeController

Classe permettant le contrôle des animes

\packages\controllers\authentification

Contient les classe permettant la validation des données des formulaires d'authentification

\packages\controllers\ListController

Classe permettant le contrôle des listes

\packages\controllers\logger

Contient la fonction utilisée par tout les `except` afin de log les potentielles erreurs

\packages\controllers\MySQLController

Classe permettant d'interagir avec la base de données MySQL

\packages\controllers\SqliteController

Classe permettant d'interagir avec la base de données Sqlite3

\packages\controllers\StatusController

Classe permettant le contrôle des statuts

\packages\controllers\TypeController

Classe permettant le contrôle des types

\packages\controllers\UserController

Classe permettant le contrôle des utilisateurs

\packages\models\Anime

Modèle représentant un anime

\packages\models\List

Modèle représentant une liste

\packages\models\Status

Modèle représentant un statut

\packages\models\Type

Modèle représentant un type

\packages\models\User

Modèle représentant un utilisateur

API interne

Animanga possède un système d'API interne permettant de faire des actions sur les données depuis le front-end. Voici les différentes url d'entrées :

/get/favorites/

Permet de récupérer tous les favoris de l'utilisateur connecté

/get/favorites/<string:nickname>

Permet de récupérer tous les favoris d'un utilisateur

/set/favorite

Met à jour le statut de favoris d'un anime pour l'utilisateur connecté

/set/list

Met à jour l'association d'un anime à une liste

/delete/defaults

Permet de supprimer l'anime des listes par défauts de l'utilisateur connecté (Complétés, En cours, Planifiés, Abandonnés)

/get/animes

Permet de récupérer les animes d'une liste

/add/list

Permet d'ajouter une nouvelle liste à l'utilisateur connecté

/delete/list

Permet de supprimer une liste de l'utilisateur connecté

/set/list/name

Met à jour le nom d'une liste

/set/favorites-order

Met à jour l'ordre des favoris

/get/activities

Permet de récupérer toutes les activités des dernières 24h de l'utilisateur connecté

Librairies et outils externes

Pip et NPM

Pip et **NPM** sont deux gestionnaires de dépendances que j'ai utilisés pour mon TPI. Pip est le gestionnaire des dépendances Python tandis que NPM est sont équivalent pour JavaScript. Ces deux gestionnaires m'ont permis d'inclure toutes les librairies externes dont j'avais besoin pour mon TPI. Ceci me permet de ne pas avoir à télécharger manuellement les libraires et à les mettre dans mon projet. Leur utilisation m'a permis de faciliter grandement le développement du TPI et d'avoir des dépendances toujours à jour.

Flask

Flask est un micro framework web écrit en Python. Aucune couche autre que l'hébergement web n'est présent dans ce micro framework. Flask à été créé par **Armin Ronacher**, membre de **Pocoo**, un groupe de développeurs Python formé en 2004 - le 1er avril 2010. J'ai choisi d'utiliser ce framework pour mon TPI car il m'a permis d'aisément effectuer les tâches suivantes :

- Héberger mon site en local ainsi de pouvoir créer des routes web. Ces dernières sont des url écrites dans la barre d'adresse du navigateur. Elles sont utilisées non seulement pour éviter de devoir écrire en dur les nom des fichiers à afficher mais aussi de pouvoir exécuter du code avant d'afficher la page à l'utilisateur afin de récupérer des informations nécessaires au bon affichage des informations dynamiques. Un bon exemple d'utilisation est la page d'accueil : si l'utilisateur n'est pas connecté, un fond contenant une image est affiché et la barre de navigation du site ne permet que d'avoir accès à l'accueil, à la page à propos, à celle de connexion et enfin celle d'inscription. L'information comme quoi l'utilisateur n'est pas connecté est récupérée avant que la page soit affichée.

- Configurer le debug de mon site de manière générale. Il est possible de donner des paramètres de configuration à l'application Flask afin de faciliter le développement. J'ai utilisé ces paramètres pour faciliter le rafraichissement des pages dès lors qu'une modification est détectée dans un fichier.

Jinja

Jinja est un moteur de modèle de page web pour Python. Il a été créé par **Armin Ronacher**. Sa syntaxe est relativement identique au moteur de modèle Django mais adaptée pour la syntaxe de Python. Ce moteur de modèle est celui par défaut de **Flask**.

Flask-Login

Flask-Login donne accès à un gestionnaire de sessions pour **Flask**. Il prend en compte les tâches standards comme la connexion, la déconnexion, et l'enregistrement de l'utilisateur en session sur une longue période de temps. Dans mon TPI, je l'utilise afin de connecter / déconnecter mes utilisateur et pour pouvoir stocker leurs informations en session durant leur utilisation du site.

Flask-Swagger

Flask-Swagger donne accès à une méthode (swagger) qui inspecte les routes de **Flask** contenant une docstring en YAML afin de générer les spécifications nécessaires à **Swagger**.

MySQL Connector/Python

MySQL Connector/Python est une librairie permettant à Python de communiquer avec les serveurs MySQL. Cette librairie est indispensable si l'on veut communiquer avec une base de données MySQL, et elle apporte des avantages tels que la conversion de données entre Python et MySQL. Par exemple, le `datetime` Python et `DATETIME` MySQL.

SASS

SASS est un préprocesseur CSS. Cet outil permet d'étendre la syntaxe du langage CSS afin de pouvoir ajouter de nouvelles fonctionnalités. SASS permet aussi d'avoir un système de variable plus puissant que celui de CSS ainsi qu'un système d'import de fichier plus épuré à mon goût. En effet, il est possible de créer un fichier pour stocker toutes les couleurs sous forme de variables et ensuite importer ce fichier dans la feuille de style principale pour pouvoir utiliser les couleurs n'importe où.

Swagger

Swagger permet de visualiser les url d'une API automatiquement, en se basant sur les spécifications de chaque url. La génération du visuel est automatique et est optimisé pour une interaction avec le client. J'ai utilisé cet outil afin de visualiser correctement les routes utilisées par mon application et de récupérer des données.

jQuery UI

jQuery UI est un ensemble d'interactions utilisateur, d'effets, de widgets, et de thèmes construits sur la base de jQuery. J'ai utilisé cet outil afin de pouvoir gérer avec facilité la réorganisation des favoris d'un utilisateur. En effet, il est possible de drag&drop les couvertures des animes présent dans les favoris de l'utilisateur afin de réorganiser l'ordre de ces derniers.

ESLint

ESLint est un outil de vérification syntaxique automatique de code. La vérification est basée sur un ensemble de règles définissant la syntaxe à utiliser. Cet outil m'a été utile pour vérifier que mon code était conforme aux normes **Airbnb**, pour éviter d'avoir des morceaux de code potentiellement problématiques ou mal optimisé voire même plus utilisés.

```
..thon/Animanga
(✿◡‿◡) ~/Documents/programmation/python/Animanga  master ●  npm run lint static/js
npm WARN npm npm does not support Node.js v10.19.0
npm WARN npm You should probably upgrade to a newer version of node as we
npm WARN npm can't make any promises that npm will work with this version.
npm WARN npm Supported releases of Node.js are the latest release of 4, 6, 7, 8, 9.
npm WARN npm You can find the latest version at https://nodejs.org/

> Animanga@1.0.0 lint /home/cavagnat/Documents/programmation/python/Animanga
> eslint '**/*.js' --ignore-pattern node_modules/ "static/js"


/home/cavagnat/Documents/programmation/python/Animanga/static/js/user-lists-handler.js
  304:22  error  Missing semicolon  semi

× 1 problem (1 error, 0 warnings)
  1 error and 0 warnings potentially fixable with the `--fix` option.

npm ERR! code ELIFECYCLE
npm ERR! errno 1
npm ERR! Animanga@1.0.0 lint: `eslint '**/*.js' --ignore-pattern node_modules/ "static/js"`
npm ERR! Exit status 1
npm ERR!
npm ERR! Failed at the Animanga@1.0.0 lint script.
npm ERR! This is probably not a problem with npm. There is likely additional logging output above.

npm ERR! A complete log of this run can be found in:
npm ERR!     /home/cavagnat/.npm/_logs/2020-06-05T07_37_00_008Z-debug.log
✗ (✿◡‿◡) ~/Documents/programmation/python/Animanga  master ●
```

Cas d'utilisation de ESLint. La command npm run lint static/js m'indique qu'un point-virgule est manquant à la ligne 93 de mon fichier user-list-handler.js.

Pylint

Pylint est aussi un outil de vérification syntaxique comme **ESLint**. Cependant, il n'utilise pas de standard créé par la communauté mais les standards officiels de Python, le **PEP8**.

```
..thon/Animanga
(✿◡‿◡) ~/Documents/programmation/python/Animanga  master ● python3 -m pylint --output-format=colorized packag
es/controllers
************* Module packages.controllers.SqliteController
packages/controllers/SqliteController.py:9:0: C0411: standard import "import sqlite3" should be placed before "from f
lask import g" (wrong-import-order)
packages/controllers/SqliteController.py:10:0: C0411: standard import "from typing import Any" should be placed befor
e "from flask import g" (wrong-import-order)
packages/controllers/SqliteController.py:11:0: C0411: standard import "from sqlite3 import Error as SqliteError" shou
ld be placed before "from flask import g" (wrong-import-order)

------------------------------------------------------------------
Your code has been rated at 9.96/10 (previous run: 9.96/10, +0.00)

✗ (✿◡‿◡) ~/Documents/programmation/python/Animanga  master ●
```

Cas d'utilisation de Pylint. La command pylint --output-format=colorized packages/controllers/SqliteController.py m'indique, entre autre, que des imports ne sont pas bien placés.

Git

Git est un outil de gestion de version. Cet outil a été utilisé durant toute la durée de mon TPI afin de garder un historique des modifications apportées à mon projet ainsi qu'un système de sauvegarde externe sur **Github** en cas de problème technique sur mon ordinateur local.

Analyse des fonctionnalités majeurs

Un CRUD complet permet de gérer une entrée manga de la bibliothèque

L'utilisateur à un contrôle total sur ses propres listes. Cela comprend la création de nouvelles listes, le renommage et la suppression des listes existantes. Il n'a cependant pas la possibilité de créer lui-même une nouvelle entrée, mais il a la possibilité d'écraser toutes les entrées avec un nouveau jeu de données. Le CRUD est donc sur la gestion des entrées et non sur les entrées elles-mêmes, mais toutes les fonctions nécessaires à un CRUD sont présentes.

La base de données locale sqlite3 est synchronisée de façon unidirectionnelle avec la base de données d'un serveur mysql

Un système de synchronisation unidirectionnelle est en place avec un algorithme fait manuellement. Le fonctionnement de cette synchronisation est le suivant : récupérer tous les enregistrements Sqlite3 et MySQL, s'il y a plus d'enregistrements Sqlite3, ajouter les enregistrements manquants dans MySQL, sinon les supprimer. Ensuite, comparer les timestamps des enregistrements dont les IDS sont identiques. Si les enregistrements MySQL ont une date inférieure à ceux de Sqlite3, les mettre à jour. Ce processus mérite des améliorations, car il reste lent lorsque MySQL est vide ou qu'énormément de modifications ont été apportées aux données. J'explique comment cela pourrait être amélioré dans la parte prévue à cet effet. Voici le pseudo-code :

```
foreach table {
    stocker le nom de la table courante

    récupérer le nombre d'enregistrements de Sqlite3 et le stocker
    récupérer le nombre d'enregistrements de MySQL et le stocker

    if nombre enregistrements Sqlite3 différent de nombre enregistrements MySQL {
        récupérer ids et timestamp Sqlite3 pour stocker dans tableau id: timestamp
        récupérer ids et timestamp MySQL pour stocker dans tableau id: timestamp

        trier les tableaux par id

        if tableau Sqlite3 > tableau MySQL {
            stocker ids Sqlite3 non présent dans tableau MySQL
            retirer les ids ci-dessus du tableau Sqlite3

            récupérer les enregistrements Sqlite3 correspondant aux ids non présent MySQL

            foreach enregistrement Sqlite3 {
                insérer dans MySQL
            }
        } else {
            stocker ids MySQL non présent dans Sqlite3

            foreach id MySQL {
                supprimer l'entrée MySQL
            }
        }
    }

    déclarer tableau pour ids Sqlite3 dont timestamp diffère du MySQL
    foreach enregistrements Sqlite3 {
        if timestamp courant différent du timestamp MySQL correspondant {
            ajouter l'id dans le tablea créer à cet effet
        }
    }

```

```
    récupérer les enregistrements Sqlite3 correspondant aux ids
    foreach enregistrement {
        mettre à jour l'enregistrement correspondant dans MySQL
    }
}
```

Les données JSON de github sont importées dans la base de données locale

Le fichier donné dans l'énoncé comporte ~500k lignes de JSON et a la structure de données suivante :

```
{
    "data": [
    {
      "sources": [
        "https://anidb.net/anime/10143",
        "https://anilist.co/anime/102416",
        "https://kitsu.io/anime/8925",
        "https://myanimelist.net/anime/20707",
        "https://notify.moe/anime/Ff1bpKmmR"
      ],
      "title": "\"0\"",
      "type": "Special",
      "episodes": 1,
      "status": "FINISHED",
      "picture": "https://cdn.myanimelist.net/images/anime/6/54815.jpg",
      "thumbnail": "https://cdn.myanimelist.net/images/anime/6/54815t.jpg",
      "synonyms": [
        "Chiaki Kuriyama: 「0」",
        "「0」"
      ],
      "relations": []
    },
    ...
}
```

J'ai dû normaliser les données afin de les insérer correctement dans la base de données locale. Grâce à Sqlite3, l'insertion d'autant d'enregistrement est très rapide, il faut compter uniquement quelques secondes. L'écrasement des données peut être réalisé par tous les utilisateurs. Cet aspect peut être amélioré et j'en parle dans la section prévue à cet effet.

Le service http utilise Python Flask

Comme l'application est développée en Python, il me faut un moyen d'héberger mon site sur le service HTTP. Pour Python, il faut utiliser Flask. la configuration n'est vraiment pas compliquée et pour lancer le serveur, il faut exécuter la commande dans le dossier parent au projet : `python3 -m Animanga.app run`.

Le planning réel est documenté et comparé au planning prescrit

Un planning m'a été donné dans l'énoncé mais je ne le trouvais pas assez précis. J'ai alors pris l'initiative de refaire un planning prévisionnel. J'y ai inscrit toutes les *user stories* créées lors du premier jour du TPI. Mon planning réel est mixé au planning prévisionnel mais les couleurs sont différentes. Comme cela, il est plus facile de comparer les deux plannings. La légende concernant les couleurs est présente sous le planning.

Le projet est publié sur github et une url est communiqué

Mon projet utilise Git et est lié à GitHub. l'URL du répertoire distant est **https://github.com/TanguyCavagna/Animanga**.

Le projet Python contient au moins une classe (python objet) conçu par le candidat

Mon projet est rempli de classes. En effet, j'ai décidé de faire un contrôleur (classe de contrôle des données) par type de données utilisé et un modèle (classe de représentation des données) pour chaque type de données. Toutes ces classes sont expliquées dans la section adéquate, sous la partie *Implémentation* .

Plans de test et tests

Périmètre des tests

Pour *Animanga* , j'ai mis en place un protocole de tests afin que n'importe quel utilisateur puisse naviguer convenablement dans l'application, peu importe son navigateur WEB.

Environnement

Lors de ces tests, j'utilise les navigateurs suivants :

- Mozilla Firefox 76.0.1 (64 bits) sur Windows 10 Entreprise 1903
- Google Chrome 81.0.4044.138 (64 bits) sur Windows 10 Entreprise 1903
- Microsoft Edge Version 81.0.416.72 (64 bits) sur Windows 10 Entreprise 1903

Scénarios

Les scénarios des tests sont détaillés afin qu'un autre professionnel puisse les exécuter. Pour rédiger mes scénarios, j'utilise la syntaxe **Gherkin** .

Nom	**1.1 Création d'un nouveau compte (informations valides)**
User Story	S1 : Inscription à Animanga
Situation	**Étant donné que** je suis un nouvel utilisateur de Animanga, je ne possède pas encore de compte. **Quand** je clique sur le bouton *Inscription* , je suis redirigé vers la page d'inscription et je remplis le formulaire avec les informations suivantes : email: `katalon@recorder.ch` , pseudo: `Katalon` , MDP: `MotDePasse` , confirmation: `MotDePasse` . **Alors**, je suis redirigé sur la page d'accueil avec mon nouveau compte connecté.
Résultats obtenus	Je suis redirigé vers la page d'accueil avec mon nouveau compte connecté.
Statut	✔ OK

Nom	**1.2 Création d'un nouveau compte (informations invalides)**
User Story	S1 : Inscription à Animanga
Situation	**Étant donné que** je suis un nouvel utilisateur de Animanga, je ne possède pas encore de compte. **Quand** je clique sur le bouton *Inscription* , je suis redirigé vers la page d'inscription et je remplis le formulaire avec les informations suivantes : email: `-` , pseudo: `Katalon` , MDP: `MotDePasse` , confirmation: `MotDePasse` . **Alors**, le formulaire est rechargé avec un message d'erreur indiquant ce qui ne s'est pas bien passé.
Résultats obtenus	Un message s'affiche m'indiquant que l'email n'est pas présent.
Statut	✔ OK

Nom	**1.3 Création d'un nouveau compte (informations invalides)**
User Story	S1 : Inscription à Animanga
Situation	**Étant donné que** je suis un nouvel utilisateur de Animanga, je ne possède pas encore de compte. **Quand** je clique sur le bouton *Inscription* , je suis redirigé vers la page d'inscription et je remplii le formulaire avec les informations suivantes : email: `a@b.c` , pseudo: `Katalon` , MDP: `MotDePasse` , confirmation: `MotDePasse` . **Alors**, le formulaire est rechargé avec un message d'erreur indiquant ce qui ne s'est pas bien passé.
Résultats obtenus	Un message s'affiche m'indiquant que l'email fourni est trop court.
Statut	✔ OK

Nom	**1.4 Création d'un nouveau compte (informations invalides)**
User Story	S1 : Inscription à Animanga
Situation	**Étant donné que** je suis un nouvel utilisateur de Animanga, je ne possède pas encore de compte. **Quand** je clique sur le bouton *Inscription* , je suis redirigé vers la page d'inscription et je remplis le formulaire avec les informations suivantes : email: `invalide/@recorder.ch` , pseudo: `Katalon` , MDP: `MotDePasse` , confirmation: `MotDePasse` . **Alors**, le formulaire est rechargé avec un message d'erreur indiquant ce qui ne s'est pas bien passé.
Résultats obtenus	Un message s'affiche m'indiquant que l'email fourni n'est pas correct.
Statut	✔ OK

Nom	**1.5 Création d'un nouveau compte (informations invalides)**
User Story	S1 : Inscription à Animanga
Situation	**Étant donné que** je suis un nouvel utilisateur de Animanga, je ne possède pas encore de compte. **Quand** je clique sur le bouton *Inscription* , je suis redirigé vers la page d'inscription et je remplis le formulaire avec les informations suivantes : email: `katalon@recorder.ch` , pseudo: `-` , MDP: `MotDePasse` , confirmation: `MotDePasse` . **Alors**, le formulaire est rechargé avec un message d'erreur indiquant ce qui ne s'est pas bien passé.
Résultats obtenus	Un message s'affiche m'indiquant que le pseudo est manquant.
Statut	✔ OK

Nom	**1.6 Création d'un nouveau compte (informations invalides)**
User Story	S1 : Inscription à Animanga
Situation	**Étant donné que** je suis un nouvel utilisateur de Animanga, je ne possède pas encore de compte. **Quand** je clique sur le bouton *Inscription* , je suis redirigé vers la page d'inscription et je remplis le formulaire avec les informations suivantes : email: `katalon@recorder.ch` , pseudo: `Katalon` , MDP: `-` , confirmation: `-` . **Alors**, le formulaire est rechargé avec un message d'erreur indiquant ce qui ne s'est pas bien passé.
Résultats obtenus	Un message s'affiche m'indiquant que le mot de passe est manquant.
Statut	✔ OK

Nom	**1.7 Création d'un nouveau compte (informations invalides)**
User Story	S1 : Inscription à Animanga
Situation	**Étant donné que** je suis un nouvel utilisateur de Animanga, je ne possède pas encore de compte. **Quand** je clique sur le bouton *Inscription* , je suis redirigé vers la page d'inscription et je remplis le formulaire avec les informations suivantes : email: `katalon@recorder.ch` , pseudo: `Katalon` , MDP: `Court` , confirmation: `Court` . **Alors**, le formulaire est rechargé avec un message d'erreur indiquant ce qui ne s'est pas bien passé.
Résultats obtenus	Un message s'affiche m'indiquant que le mot de passe est trop court.
Statut	✔ OK

Nom	**1.8 Création d'un nouveau compte (informations invalides)**
User Story	S1 : Inscription à Animanga
Situation	**Étant donné que** je suis un nouvel utilisateur de Animanga, je ne possède pas encore de compte. **Quand** je clique sur le bouton *Inscription* , je suis redirigé vers la page d'inscription et je remplis le formulaire avec les informations suivantes : email: `katalon@recorder.ch` , pseudo: `Katalon` , MDP: `-` , confirmation: `-` . **Alors**, le formulaire est rechargé avec un message d'erreur indiquant ce qui ne s'est pas bien passé.
Résultats obtenus	Un message s'affiche m'indiquant que le mot de passe de confirmation est manquant.
Statut	✔ OK

Nom	**1.9 Création d'un nouveau compte (informations invalides)**
User Story	S1 : Inscription à Animanga
Situation	**Étant donné que** je suis un nouvel utilisateur de Animanga, je ne possède pas encore de compte. **Quand** je clique sur le bouton *Inscription* , je suis redirigé vers la page d'inscription et je remplis le formulaire avec les informations suivantes : email: `katalon@recorder.ch` , pseudo: `Katalon` , MDP: `MotDePasse` , confirmation: `MotDePasseDifferent` . **Alors**, le formulaire est rechargé avec un message d'erreur indiquant ce qui ne s'est pas bien passé.
Résultats obtenus	Un message s'affiche m'indiquant que les mots de passes sont différent.
Statut	✔ OK

Nom	**1.10 Création d'un nouveau compte (informations invalides)**
User Story	S1 : Inscription à Animanga
Situation	**Étant donné que** je suis un nouvel utilisateur de Animanga, je ne possède pas encore de compte. **Quand** je clique sur le bouton *Inscription* , je suis redirigé vers la page d'inscription et je remplis le formulaire avec les informations suivantes : email: `katalon@recorder.ch` , pseudo: `Katalon` , MDP: `MotDePasse` , confirmation: `MotDePasseDifferent` . **Alors**, le formulaire est rechargé avec un message d'erreur indiquant ce qui ne s'est pas bien passé.
Résultats obtenus	Un message s'affiche m'indiquant que l'email est déjà utilisé.
Statut	✔ OK

Nom	**2.1 Connexion avec un compte existant (informations valides)**
User Story	S2 : Connexion à Animanga
Situation	**Étant donné que** je suis un utilisateur de Animanga, j'ai déjà un compte à disposition. **Quand** je clique sur le bouton *Connexion* , je suis redirigé vers la page de connexion et je remplis le formulaire avec les informations suivantes : email: `katalon@recorder.ch` , MDP: `MotDePasse` . **Alors**, je suis redirigé sur la page d'accueil avec mon compte connecté.
Résultats obtenus	Je suis redirigé vers la page d'accueil avec mon nouveau compte connecté.
Statut	✔ OK

Nom	**2.2 Connexion avec un compte existant (informations invalides)**
User Story	S2 : Connexion à Animanga
Situation	**Étant donné que** je suis un utilisateur de Animanga, j'ai déjà un compte à disposition. **Quand** je clique sur le bouton *Connexion* , je suis redirigé vers la page de connexion et je remplis le formulaire avec les informations suivantes : email: `-` , MDP: `MotDePasse` . **Alors**, je suis redirigé sur la page d'accueil avec mon compte connecté.
Résultats obtenus	Un message s'affiche m'indiquant que l'email est manquant.
Statut	✔ OK

Nom	**2.3 Connexion avec un compte existant (informations invalides)**
User Story	S2 : Connexion à Animanga
Situation	**Étant donné que** je suis un utilisateur de Animanga, j'ai déjà un compte à disposition. **Quand** je clique sur le bouton *Connexion* , je suis redirigé vers la page de connexion et je remplis le formulaire avec les informations suivantes : email: `invalide/@recorder.ch` , MDP: `MotDePasse` . **Alors**, je suis redirigé sur la page d'accueil avec mon compte connecté.
Résultats obtenus	Un message s'affiche m'indiquant que l'email est invalide.
Statut	✔ OK

Nom	2.4 Connexion avec un compte existant (informations invalides)
User Story	S2 : Connexion à Animanga
Situation	**Étant donné que** je suis un utilisateur de Animanga, j'ai déjà un compte à disposition. **Quand** je clique sur le bouton *Connexion* , je suis redirigé vers la page de connexion et je remplis le formulaire avec les informations suivantes : email: `katalon@recorder.ch` , MDP: `-` . **Alors**, je suis redirigé sur la page d'accueil avec mon compte connecté.
Résultats obtenus	Un message s'affiche m'indiquant que le mot de passe est manquant.
Statut	✔ OK

Nom	2.5 Connexion avec un compte existant (informations invalides)
User Story	S2 : Connexion à Animanga
Situation	**Étant donné que** je suis un utilisateur de Animanga, j'ai déjà un compte à disposition. **Quand** je clique sur le bouton *Connexion* , je suis redirigé vers la page de connexion et je remplis le formulaire avec les informations suivantes : email: `katalon@recorder.ch` , MDP: `MotDePasseInexistant` . **Alors**, je suis redirigé sur la page d'accueil avec mon compte connecté.
Résultats obtenus	Un message s'affiche m'indiquant que la combinaison email - mot de passe est invalide.
Statut	✔ OK

Nom	2.6 Déconnexion
User Story	S2 : Connexion à Animanga
Situation	**Étant donné que** je suis un utilisateur connecté au site. **Quand** je clique sur le bouton *Déconnexion* placé dans le dropdown du menu *Utilisateur* . **Alors**, je deviens un utilisateur non connecté et je suis redirigé sur la page de connexion.
Résultats obtenus	Je clique sur *Utilisateur* et *Déconnexion* . Je ne suis plus connecté et je suis redirigé sur la page de connexion.
Statut	✔ OK

Nom	**3.1 Importation des animes**
User Story	S3 : Importation des animes
Situation	**Étant donné que** je suis un utilisateur connecté au site. **Quand** je clique sur le bouton *Écraser tous les animes* placé dans le dropdown du menu *Utilisateur* . **Alors**, j'écrase toutes les données du site relatives aux animes eux-même. Cela comprend les animes en eux même, les favoris ainsi que les animes contenus dans les listes.
Résultats obtenus	Je clique sur *Utilisateur* et *Écraser tous les animes* . Je suis redirigé vers la page d'accueil et des alertes s'affichent en haut au centre de l'écran indiquant l'état de la mise à jours des animes.
Statut	✔ OK

Nom	**4.1 Recherche des animes**
User Story	S4 : Recherche des animes
Situation	**Étant donné que** je suis un utilisateur connecté au site. **Quand** je clique sur le bouton 🔍 placé dans la barre de navigation et que j'écris "k On" dans le champ de recherche de la modale. **Alors**, je suis redirigé vers la page d'accueil et les résultats de la recherche affiche l'anime "K-ON!".
Résultats obtenus	L'anime "K-ON!" est présent dans la zone de résultat de recherche.
Statut	✔ OK

Nom	**4.2 Recherche des animes avec raccourcis**
User Story	S4 : Recherche des animes
Situation	**Étant donné que** je suis un utilisateur connecté au site. **Quand** je fais le raccourcis clavier `Ctrl` + `S` et que j'écris "k On" dans le champ de recherche de la modale. **Alors**, je suis redirigé vers la page d'accueil et les résultats de la recherche affiche l'anime "K-ON!".
Résultats obtenus	L'anime "K-ON!" est présent dans la zone de résultat de recherche.
Statut	✔ OK

Nom	5.1 Affichage de la carte de l'anime
User Story	S5 : Affichage de la carte de l'anime
Situation	**Étant donné que** je suis un utilisateur connecté au site sur la page d'accueil ayant fait une recherche. **Quand** je clique sur le titre d'un anime présent dans la zone de résultat de la recherche. **Alors**, une modale s'affiche contenant l'image de couverture, le titre, le statut de visionnement de l'anime, et les listes personnelles de l'utilisateur.
Résultats obtenus	La modale s'affiche avec le contenu adéquat.
Statut	✔ OK

Nom	6.1 Mise à jour du statut de l'anime sélectionné
User Story	S6 : Mise à jour de l'anime
Situation	**Étant donné que** je suis un utilisateur connecté au site sur la page d'accueil ayant fait une recherche et ayant ouvert la modale d'informations d'un anime. **Quand** sélectionne un statut autre que "---". **Alors**, le combo-box se met à jour avec la nouvelle valeur sélectionnée.
Résultats obtenus	La valeur du combo-box s'est bien mise à jour.
Statut	✔ OK

Nom	6.2 Ajout de l'anime dans une liste personnelle
User Story	S6 : Mise à jour de l'anime
Situation	**Étant donné que** je suis un utilisateur connecté au site sur la page d'accueil ayant fait une recherche et ayant ouvert la modale d'informations d'un anime. **Quand** je clique sur une check-box blanche d'une des listes personnelles. **Alors**, l'état de la check-box se met à jour et elle se colore en bleu. L'anime est maintenant présent dans la liste personnelle.
Résultats obtenus	L'état de la check-box s'est bien mis à jour et est bien coloré en bleu.
Statut	✔ OK

Nom	6.3 Ajout de l'anime dans les favoris
User Story	S6 : Mise à jour de l'anime
Situation	**Étant donné que** je suis un utilisateur connecté au site sur la page d'accueil ayant fait une recherche et ayant ouvert la modale d'informations d'un anime. **Quand** je clique sur le cœur blanc pour ajouter au favoris. **Alors**, le cœur se colore et l'anime se rajoute dans la zone des favoris de la page d'accueil.
Résultats obtenus	Le cœur s'est coloré et l'anime s'est correctement ajouté dans la zone des favoris de la page d'accueil.
Statut	✔ OK

Nom	6.4 Suppression du statut de l'anime
User Story	S6 : Mise à jour de l'anime
Situation	**Étant donné que** je suis un utilisateur connecté au site sur la page d'accueil ayant fait une recherche et ayant ouvert la modale d'informations d'un anime. **Quand** je sélectionne le statut "---". **Alors**, le combo-box se met à jour avec la nouvelle valeur sélectionnée et l'anime n'est plus présent dans aucun autre statut.
Résultats obtenus	La valeur du combo-box s'est bien mise à jour et l'anime n'est effectivement plus présent dans les autres statuts.
Statut	✔ OK

Nom	6.5 Suppression de l'anime d'une liste personnelle
User Story	S6 : Mise à jour de l'anime
Situation	**Étant donné que** je suis un utilisateur connecté au site sur la page d'accueil ayant fait une recherche et ayant ouvert la modale d'informations d'un anime. **Quand** je clique sur une check-box bleue d'une des listes personnelles. **Alors**, l'état de la check-box se met à jour et se colore en blanc. L'anime n'est plus présent dans la cette liste personnelle.
Résultats obtenus	L'état de la check-box s'est bien mis à jour et est coloré en blanc.
Statut	✔ OK

Nom	6.6 Suppression de l'anime des favoris
User Story	S6 : Mise à jour de l'anime
Situation	**Étant donné que** je suis un utilisateur connecté au site sur la page d'accueil ayant fait une recherche et ayant ouvert la modale d'informations d'un anime. **Quand** je clique sur le cœur rose pour supprimer des favoris. **Alors**, le cœur se colore en blanc et l'anime se supprime de la zone des favoris de la page d'accueil.
Résultats obtenus	Le cœur s'est coloré en blanc et l'anime s'est correctement supprimé de la zone des favoris de la page d'accueil.
Statut	✔ OK

Nom	7.1 Affichage du profile de l'utilisateur connecté
User Story	S7 : Affichage du profile
Situation	**Étant donné que** je suis un utilisateur connecté. **Quand** je clique sur *Profile* dans la barre de navigation. **Alors**, la page de profile de l'utilisateur connecté s'affiche avec ses statistiques et ses favoris.
Résultats obtenus	La page de profile de l'utilisateur connecté s'affiche correctement et les statistiques ainsi que les favoris sont les siens.
Statut	✔ OK

Nom	**8.1 Affichage des listes de l'utilisateur connecté**
User Story	S8 : Affichage des listes
Situation	**Étant donné que** je suis un utilisateur connecté. **Quand** je clique sur *Listes* dans la barre de navigation. **Alors**, la page contenant toutes les listes de l'utilisateur connecté s'affiche ainsi que les animes contenus dans ces listes.
Résultats obtenus	La page contenant les listes de l'utilisateur connecté s'est correctement affichée et les animes sont correctement affiché aussi.
Statut	✔ OK

Nom	**9.1 Créer une liste**
User Story	S9 : Gestion des listes
Situation	**Étant donné que** je suis un utilisateur connecté. **Quand** je suis sur la page des listes et que j'écris "Ma nouvelle liste" dans le champ de texte *Nouvelle liste* et que j'appuie sur `Enter`. **Alors**, la liste apparaît en bas des listes déjà présentes avec une 🗑 à côté.
Résultats obtenus	La liste à bien été ajoutée en base des listes déjà présente.
Statut	✔ OK

Nom	**9.2 Supprimer une liste**
User Story	S9 : Gestion des listes
Situation	**Étant donné que** je suis un utilisateur connecté. **Quand** je suis sur la page des listes et que je clique sur 🗑 d'une liste présente. **Alors**, la liste ne sera plus présente dans les listes déjà existantes.
Résultats obtenus	La liste à bien été supprimée et n'est plus présente dans les listes déjà existantes.
Statut	✔ OK

Nom	**9.3 Renommer une liste**
User Story	S9 : Gestion des listes
Situation	**Étant donné que** je suis un utilisateur connecté. **Quand** je double-clique sur le nom de la liste, je peux renommer la liste et valider en appuyant sur `Entré`. **Alors**, le nom de la liste est changé.
Résultats obtenus	Le nom de la liste est bien mis à jour.
Statut	✔ OK

Nom	10.1 Affichage des favoris sur l'accueil
User Story	S10 : Affichage des favoris
Situation	**Étant donné que** je suis un utilisateur connecté. **Quand** je suis sur la page d'accueil. **Alors**, mes favoris sont présents sur la page.
Résultats obtenus	Mes favoris sont bien affichés.
Statut	✔ OK

Nom	10.2 Affichage des favoris du profile
User Story	S10 : Affichage des favoris
Situation	**Étant donné que** je suis un utilisateur connecté. **Quand** je suis sur ma page de profile. **Alors**, mes favoris sont présent sur la page.
Résultats obtenus	Mes favoris sont bien affiché.
Statut	✔ OK

Nom	11.1 Organisation des favoris
User Story	S11 : Organisation des favoris
Situation	**Étant donné que** je suis un utilisateur connecté. **Quand** je suis sur la page des favoris et je clique sur le bouton *Réorganiser les favoris* , je peux glisser déposer les animes dans l'ordre que je veux. Je clique sur le bouton *Sauvegarder* pour enregistrer l'ordre. **Alors**, mes favoris sont enregistrés dans l'ordre voulu.
Résultats obtenus	Mes favoris ont bien été réorganisé.
Statut	✔ OK

Nom	11.2 suppression des favoris
User Story	S11 : Organisation des favoris
Situation	**Étant donné que** je suis un utilisateur connecté. **Quand** je suis sur la page des favoris et je clique sur le bouton *Réorganiser les favoris* , je peux cliquer sur ⛔ pour enlever l'anime des favoris. **Alors**, l'anime ne fait plus partie des favoris.
Résultats obtenus	L'anime est bien retiré des favoris.
Statut	✔ OK

Nom	12.1 Affichage de la landing page
User Story	S12 : Affichage de la landing page
Situation	**Étant donné que** je suis un utilisateur non connecté. **Quand** je suis sur le site. **Alors**, une page d'accueil s'affiche avec comme possibilité : la visite de la page *À propos* , la connexion et l'inscription.
Résultats obtenus	La page d'accueil ainsi que la barre de navigation sont affichés correctement pour un utilisateur non connecté.
Statut	✔ OK

Nom	13.1 Respect du preset Airbnb
User Story	S17 : Vérification syntaxique
Situation	**Étant donné que** je suis un développeur. **Quand** j'exécute la commande `npm run lint` dans le dossier de mon projet. **Alors**, aucune erreur de syntaxe sur la base du preset Airbnb n'est relevée.
Résultats obtenus	> Animanga@1.0.0 lint /home/cavagnat/Documents/programmation/python/Animanga > eslint '*/ .js' --ignore-pattern node_modules/
Statut	✔ OK

Nom	13.2 Respect des conventions PEP8
User Story	S17 : Vérification syntaxique
Situation	**Étant donné que** je suis un développeur. **Quand** j'exécute la commande `python3 -m pylint --output-format=colorized packages` à la racine de mon projet. **Alors**, aucune erreur de syntaxe sur la base des conventions PEP8 n'est relevée.
Résultats obtenus	La note attribuée au code et supérieur à 10/10.
Statut	✔ OK

Nom	14.1 Synchronisation Sqlite3 - MySQL
User Story	S15 : Synchronisation MySQL Sqlite3
Situation	**Étant donné que** je suis un utilisateur connecté. **Quand** je clique sur l'icône de synchronisation. **Alors**, la page charge et je suis redirigé vers la page d'accueil.
Résultats obtenus	Les données sont identiques entre la base Sqlite3 et MySQL.
Statut	✔ OK

Nom	15.1 Affichage des activités des 24 dernières heures
User Story	S15 : Affichage des activités
Situation	**Étant donné que** je suis un utilisateur connecté. **Quand** je mets un anime en favoris et dans une liste (changement de statut aussi). **Alors**, une carte s'affiche sur l'accueil avec le nom de l'anime modifié ainsi que son image, le nom de la liste dans laquelle il a été ajouté, et le temps écoulé depuis la mise à jour.
Résultats obtenus	La carte s'affiche avec les informations correctes.
Statut	✔ OK

Évolution des tests

Nº Test	J1 lu.25	J2 ma.26	J3 me.27	J4 je.28	J5 ve.29	J6 ma.2	J7 me.3	J8 je.4	J9 ve.5	J10 lu.8	J11 ma.9
1.1	✗	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓
1.2	✗	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓
1.3	✗	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓
1.4	✗	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓
1.5	✗	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓
1.6	✗	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓
1.7	✗	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓
1.8	✗	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓
1.9	✗	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓
2.1	✗	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓
2.2	✗	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓
2.3	✗	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓
2.4	✗	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓
2.5	✗	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓
2.6	✗	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓
3.1	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
4.1	✗	✗	✗	✓	✓	✓	✓	✓	✓	✓	✓
4.2	✗	✗	✗	✓	✓	✓	✓	✓	✓	✓	✓
5.1	✗	✗	✗	✓	✓	✓	✓	✓	✓	✓	✓
6.1	✗	✗	✗	✗	✓	✓	✓	✓	✓	✓	✓
6.2	✗	✗	✗	✗	✗	✓	✓	✓	✓	✓	✓
6.3	✗	✗	✗	✗	✓	✓	✓	✓	✓	✓	✓
6.4	✗	✗	✗	✗	✓	✓	✓	✓	✓	✓	✓
6.5	✗	✗	✗	✗	✗	✓	✓	✓	✓	✓	✓
6.6	✗	✗	✗	✗	✓	✓	✓	✓	✓	✓	✓
7.1	✗	✗	✗	✗	✓	✓	✓	✓	✓	✓	✓
8.1	✗	✗	✗	✗	✗	✓	✓	✓	✓	✓	✓
9.1	✗	✗	✗	✗	✗	✓	✓	✓	✓	✓	✓
9.2	✗	✗	✗	✗	✗	✓	✓	✓	✓	✓	✓
10.1	✗	✗	✗	✓	✓	✓	✓	✓	✓	✓	✓
10.2	✗	✗	✗	✓	✓	✓	✓	✓	✓	✓	✓
11.1	✗	✗	✗	✗	✗	✓	✓	✓	✓	✓	✓
11.3	✗	✗	✗	✗	✗	✓	✓	✓	✓	✓	✓
12.1	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
13.1	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
13.2	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

N° Test	J1 lu.25	J2 ma.26	J3 me.27	J4 je.28	J5 ve.29	J6 ma.2	J7 me.3	J8 je.4	J9 ve.5	J10 lu.8	J11 ma.9
14.1	✗	✗	✗	✗	✗	✗	✗	✓	✓	✓	✓
15.1	✗	✗	✗	✗	✗	✗	✓	✓	✓	✓	✓

Bibliographie

Voici les différentes ressource techniques consultées lors lu développement de mon projet :

- Le documentation officielle Python : **https://docs.python.org/3/**
- MDN web docs, recueil très complet concernant le HTML, CSS, et JavaScript, créer par Mozilla : **https://developer.mozilla.org/en-US/**
- Site de questions en lignes pour développeurs de tout les horizons : **https://stackoverflow.com/**
- Le guide de style Airbnb, spécialisé pour la syntaxe JavaScript : **https://github.com/airbnb/javascript**
- Les sources des librairies externes utilisées lors de ce projet :
  - Documentation officielle de Flask : **https://palletsprojects.com/p/flask/**
  - Documentation officielle de Jinja2 : **https://jinja.palletsprojects.com/en/2.11.x/**
  - Documentation officielle de Flask-Login : **https://flask-login.readthedocs.io/en/latest/**
  - Exemple d'utilisation de flask-swagger : **https://pypi.org/project/flask-swagger/**
  - Documentation complète de Mysql Connector/Python : **https://dev.mysql.com/doc/connector-python/en/**
  - Explications de ce qu'est Swagger : **https://swagger.io/tools/swagger-ui/**
  - Documentation officielle de JquerUI : **https://jqueryui.com/**
  - Sources de SwaggerUI, répertoire distant comportant les fichiers utilisés pour l'affichage de mes points d'entrés pour mon API interne : **https://github.com/swagger-api/swagger-ui**

Glossaire

Termes métier

Termes	Explications
Anime (Anglais : *anime*)	Série, films ou épisodes spéciaux en dessins animé d'origine japonaise.
Favoris (Anglais : *favorite*)	Anime que l'utilisateur aime particulièrement beaucoup.
Liste (Anglais : *list*)	Conteneur pouvant accueil 1 ou plusieurs animes en son sein afin de pouvoir organisé correctement sa collection.
Statut (Anglais : *status*)	État de visionnement d'un anime. Ce dernier peut être *Complété* , *En cours* , *Abandonné* ou *Planifié.*

Termes technique

Termes	Explications
API (*Application Programming Interface*)	Permet d'avoir accès à des fonctionnalités d'un site facilement. En web, ces accesseurs sont des urls définissant clairement ce qu'elles font (Exemple : */get/users* ➡ récupère les utilisateurs).
Back-end	Partie caché d'une application permettant de communiquer avec tout le côté serveur. Cela comprend la base de données, l'authentification, etc...
Framework	En web, ensemble d'outils et de fonctionnalités construit spécialement pour le support de services web, gestion de ressources et déploiement web. Un framework apporte un manière standardisé de réalisé un projet et automatise des protocole fastidieux si fait manuellement .
Front-end	Partie visible par l'utilisateur d'une application. C'est cette partie qui comprend entre autre toute la partie ergonomie et les interfaces.
Librairies	Ensemble de fonctionnalités externes au projet conçu par des développeur pour des développeur.
Route	Url ne pointant pas sur un fichier directement mais fonctionnant comme une requête faite au serveur afin d'afficher des données ou faire une action précise.
Script	Programme chargé d'exécuter une tâche pré-défini permettant habituellement d'automatisé des actions.
Template	Page HTML comportant une mise en page précise et où les données sont inséré dynamiquement grâce à des requête ou au back-end lors du chargement de la page.

Conclusion

Difficultés majeures rencontrées

Durant tout le développement de mon projet, aucun problème bloquant n'a surgi. Voici cependant la liste des soucis majeurs que j'ai rencontrés :

- > L'outil de fusion de PDF que j'utilisais - **pdfunite** - ne prenait pas en charge les titres lors de la fusion de plusieurs PDF. Ainsi, si un PDF comportait des titres, après la fusion sa fusion avec les autres documents, les titres étaient considérés comme simple texte.

  ➡ J'ai pu corriger ce souci en changeant de librairie de fusion de document PDF. La recherche d'une nouvelle librairie n'a pas été compliquée du tout car il existe un nombre élevé de librairies permettant de fusionner des PDF, dont **pdftk** et le didacticiel disponible à **cette adresse**.

- > Lors de la synchronisation, j'utilise des timestamps pour savoir quels enregistrements ont été modifiés. Cependant, j'utilisais un format différent entre ma base Sqlite3 et MySQL. Dans Sqlite3, le format utilisé était `%Y-%m-%d %H:%M:%f` ce qui donne `2020-06-08 09:08:32.276`. Les millisecondes sont présentes avec ce format. Or, le format utilisé par défaut dans MySQL est `%Y-%m-%d %H:%M:%S` ce qui donne `2020-06-08 09:08:32`. Cette différence de format faisait que lorsqu'un timestamp Sqlite3 dont les millisecondes sont plus grandes que `.500`, ce timestamp était arrondi vers le haut et donc mes timestamps étaient différents.

  ➡ Le souci n'était de loin pas compliqué à régler mais j'ai mis un certain temps avant de trouver ce qui causait un différent entre Sqlite3 et MySQL. Une fois la cause trouvée, j'ai simplement fait en sorte que la date que j'insère dans Sqlite3 ne comporte pas les millisecondes En procédant ainsi, la synchronisation se fait sans problème.

Améliorations possibles

Étant donnée la courte période mise à disposition, il est clair que des améliorations sont possibles sur les fonctionnalités existantes. Voici un aperçu de ce qui pourrait être amélioré :

- Ajouter la fonctionnalité de pouvoir modifier son profile. Cela comporte le pseudo, l'email et le mot de passe ;
- Faire en sorte que l'interface du site soit *responsive design* (qu'il s'adapte sur tout type d'écran). Pour le moment, cette fonctionnalité n'est qu'à demi implémentée ;
- Proposer davantage de résultats lors d'une recherche. Pour le moment, seuls les neufs résultats les plus adéquats par rapport à la chaine recherché apparaissent ;
- Modifier la fonctionnalité de synchronisation unidirectionnelle entre Sqlite3 et MySQL. Pour le moment, l'algorithme utilisé est relativement efficace mais il pourrait être amélioré, par exemple, en stockant le timestamp de la dernière synchronisation dans une table, en supprimant tous les enregistrements qui ne sont plus présents dans Sqlite3 de MySQL, en ne sélectionnant que les enregistrements dont la date est supérieure ou égale à la dernière synchronisation, en mettant à jours les enregistrements MySQL et en ajoutant les enregistrements manquant. Cette façon de faire permettrait à l'algorithme d'être beaucoup plus rapide qu'actuellement.

Outre les fonctionnalités existantes, j'ai pensé à ces quelques idées durant le développement de l'application :

- Ajouter la fonctionnalité de pouvoir se faire une liste d'amis en cherchant le pseudo de l'utilisateur dans un champs prévu à cet effet et ensuite avoir une page dédiée à l'affichage de cette dite liste d'amis afin de pouvoir aller voir le profil de ces derniers ;
- Ajouter un système de rôle permettant aux administrateurs de pouvoir gérer les animes sans que les utilisateurs puissent le faire pour éviter toute fausse manipulation ;
- Ajouter la fonctionnalité permettant aux utilisateurs de mettre une note à un anime et une progression de visionnage (nombre d'épisodes regardés) ;
- Modifier le contenu des activités pour ajouter celles des amis.

Bilan personnel

Ce projet m'a énormément plus. Le sujet était parfait pour moi : j'adore réaliser des projets en Python, surtout web, et je suis passionné par les animes. Le fait de pouvoir lier ces deux passions était très agréable.

Je trouve très plaisant d'utiliser cette application de par sa simplicité et son contenu fourni. Le fait d'écrire une documentation aussi volumineuse était une première pour moi ! Ce fût non seulement satisfaisant mais aussi très enrichissant. J'ai en effet, appris quantité de choses.

Remerciements

J'apporte mes remerciements à :

- M. Pascal Bonvin pour son suivi assidu lors de ce TPI ;
- M. Nicolas Ettlin pour ses conseils avisés concernant l'utilisation d'eslint et quelques techniques CSS ;
- Ma famille pour la relecture et l'aide à la correction orthographique et grammaticale.

Annexes

Résumé TPI

https://flask.tcsandbox.ch

RÉSUMÉ DU RAPPORT TPI

Tanguy Cavagna - mai - juin 2020

École de métiers : CFPT Informatique, Petit-Lancy GE

Entreprise formatrice : Formation plein temps

Animanga

Situation de départ

Ce *Travail Pratique Individuel* (TPI), de 88 heures disposées sur 11 jours, a été réalisé dans le cadre de l'examen de validation d'acquis pour la formation CFC-Informatique Développement d'Application. L'énoncé qui m'a été attribué me donne le projet suivant : une application web permettant de faire des listes personnelles d'animes, ainsi que de mettre ces animes en favoris. Les utilisateurs doivent avoir un compte pour accéder à l'application et ainsi utiliser le site. Les données doivent provenir d'un fichier `json` donné et doivent être importées dans une base de données Sqlite3. D'autres fonctionnalités sont indispensables pour pouvoir rechercher les animes et affiché les activités de l'utilisateur.

Mise en œuvre

L'application a été réalisée avec Python et le micro-framework Flask pour la partie back-end, JavaScript pour le front-end, ainsi que Sqlite3 et MySQL pour la base de données. L'application fonctionne grâce à des routes mises en place dans Flask et les pages chargent les données en utilisant `fetch` de JavaScript ainsi que des variables Python interprétées lors du chargement de la page. Les tâches de l'énoncé ont été divisées en *user stories* et la *méthodologie en 6 étapes* a été utilisé pour structuré la progression du projet. Un protocole de test complet a été créé pour rendre l'application conforme à l'énoncé. Un journal de bord quotidien a été tenu à jour et comporte toutes les étapes entreprises tout au long du travail, les problèmes rencontrés, solutions apportées et rendez-vous télévisuel fait.

Résultats

Toutes les fonctionnalités de l'énoncé ont été implémentées et les utilisateurs ont une expérience agréable lors de l'utilisation de l'application. Le code JavaScript est conforme aux normes Airbnb, vérifier grâce à l'analyseur syntaxique (*linter*) ESLint, tout comme le Python est conforme aux normes PEP8 grâce à l'analyseur syntaxique Pylint. Une documentation complète a également été rédigée.

Énoncé

République et canton de Genève
Département de l'instruction publique, de la formation et de la jeunesse (DIP)
Office pour l'orientation, la formation professionnelle et continue

Travail pratique individuel (TPI)
Informaticien-ne CFC
Dossier d'inscription et description du travail

Candidat :		**Entreprise formatrice :**	
Nom :	CAVAGNA	Société :	CFPT – Ecole d'informatique
Prénom :	Tanguy	Adresse :	10, Ch. Gérard de Ternier
Classe :	I.DA-P4B	Localité :	1213 Petit-Lancy
Tel professionnel :		Téléphone :	022 388 87 28
Tel mobile/privé :	0041 76 615 92 28	**Nom Formateur** :	Pascal Bonvin
E-Mail :	tanguy.cvgn@eduge.ch	Tel direct :	0033 632 17 84 11
		E-Mail :	pascal.bonvin@edu.ge.ch

Titre du travail :

Domaine :
☑ Développement d'applications ❑ Informatique d'entreprise ❑ Technique des systèmes

Durée du travail (comprise entre **70h** et **90h**) : **88h** **Date de début souhaitée** : **20 avril 2020**

Horaire hebdomadaire du travail : 7h30-11h40 / 12h40 -16h45
☑ lundi _____ ☑ mardi _____ ☑ mercredi _____ ☑ jeudi _____ ❑ vendredi _____

Lieu où se déroule le TPI si différent de l'adresse de l'employeur (adresse complète) :
Salle R111 (I.DA-P4B) / R113 (I.DA-P4A)

Résumé du travail :

Animanga
Ce projet permet à un utilisateur de constituer sa propre bibliothèque de mangas. Projet web python sqlite3 de gestion de bibliothèque privée de manga, il permet de proposer une alternative technique au traditionnel site web apache+php+mysql. Après installation locale, l'utilisateur accède à sa bibliothèque via un navigateur http.

RAPPEL :

Il est interdit au candidat de prendre connaissance de l'énoncé du travail de TPI avant le début de celui-ci.

L'énoncé lui sera transmis par les experts, par mail, le matin du 1er jour du TPI avant 7h30.

Devoir d'examen défini. L'entreprise formatrice :

Lieu : **Date :**

Signature :

Les pages suivantes contiennent la description du projet. Le dossier sera ensuite validé par le collège des experts qui désignera un (et dans ce cas le chef expert participera à la présentation) ou deux d'entre eux pour le suivi du déroulement du travail. L'acceptation de celui-ci sera confirmée par leurs signatures sur la feuille d'évaluation du TPI.
Rappel : Tous les dossiers incomplets seront automatiquement refusés.

TPI - Cahier des charges

Ce document sera connu du candidat uniquement au commencement du TPI. Il est interdit d'en communiquer le contenu au candidat avant la date de TPI convenue.

1. Titre

Animanga

2. Matériel et logiciels à disposition

- Un PC standard école, 2 écrans
- Pycharm
- Netbeans.
- Suite office.
- Mysql, Sqlite3, Flask, connexion internet.

3. Prérequis

Les notions spécifiques de ce projet ont été/seront étudiées en ateliers application et atelier TPI par le formateur et l'élève.

4. Descriptif complet du projet

Lorsqu'il s'agit de réaliser un site web, la tradition de l'école d'informatique encourage l'utilisation de PHP et Mysql. Dans le cas de ce diplôme, il s'agit de réaliser un site local avec Python Flask et de gérer les données dans une base de données Sqlite3. De plus la base de donnée locale est synchronisée unidirectionnel ment avec une base de donnée Mysql sur un serveur distant. L'ambition de ce projet est de démontrer qu'il est possible de créer un application WEB sans passer par l'installation d'un serveur apache et d'une base données Mysql. A noter que le candidat est un élève brillant qui maîtrise le langage Python.

Les données initiales qui permettront de remplir la base de données sont accessibles sur github au format json (https://github.com/manami-project/anime-offline-database). Elles devront être converties et synchronisées dans la base de données locale qui reste à déterminer.

Sitemap

Description succincte du contenu des pages du site :

- La page index permet d'avoir le champ de recherche, un bouton "aléatoire", ainsi que les favoris de l'utilisateur et son flux d'activité si connecter.
- La page à propos contient toutes les informations concernant le site ainsi que les librairies utilisées.
- La page connexion permet simplement de se connecter.
- La page inscription, de s'inscrire.
- La page anime / manga permet de voir les informations / actions sur un anime / manga sélectionner (rediriger sur cette page lorsque l'on clique sur un anime / manga sur la page index après une recherche).
- La page aide contient l'aide du site.
- La page profil contient les information de l'utilisateur ainsi que les listes personnelles, dans lesquelles des animes / manga peuvent être ajoutés, ainsi que leur contenu.
- La page favoris permet de modifier l'ordre des animes / manga mit en favoris ainsi que de les retirer de la liste des favoris.
- La page liste contient un CRUD sur les listes personnelles de l'utilisateur.

Le planning réel devra être comparé au planning prescrit suivant :

Jour	1		2		3		4		5		6		7		8		9		10		11	
Demi-Journée	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
Etude du sujet. Planification	■	■																				
Installation, import des données			■	■																		
Configuration Flask					■																	
Gestion CRUD						■	■	■														
Synchronisation sqlite/mysql									■	■	■	■										
Interface WEB													■	■								
Finalisation / Corrections															■	■						
Tests				■	■	■	■	■	■	■	■	■	■	■	■	■						
Documentation	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	
Résumé																			■			
Finalisation / Impressions																					■	■
Journal de bord	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■						

5. Livrables

Planning réel
Rapport de projet
Manuel utilisateur (si applicable)
Journal de travail

6. Points techniques évalués spécifiques au projet (obligatoire) correspondants aux points A14 à A20 du formulaire d'évaluation

A14 : Un CRUD complet permet de gérer une entrée manga de la bibliothèque

A15 : La base de données locale sqlite3 est synchronisée de façon unidirectionnelle avec la base de données d'un serveur mysql.

A16 : Les données JSON de github sont importées dans la base de données locale

A17 : Le service http utilise Python Flask.

A18 : Le planning réel est documenté et comparé au planning prescrit.

A19 : Le projet est publié sur github et une url est communiquée.

A20 : La projet Python contient au moins une classe (python objet) conçue par le candidat.

Journal de bord

https://flask.tcsandbox.ch

Journal de bord TPI - Tanguy CAVAGNA

J1 : lundi 25 mai 2020

Objectifs

L'objectif de cette journée est de lire l'énoncé dans son intégralité afin de prendre connaissance du cahier des charges, d'en extraire les *user stories* pour pouvoir correctement rédiger mon *product backlog* et enfin de rédiger les scénarios de tests fonctionnels, indispensables pour le bon fonctionnement de mon projet.

Déroulement

Je commence ma journée à 8h00. Je lis avec attention l'énoncé que M. Terrond m'a fait parvenir la veille. Ainsi, j'aborde la première étape de la **méthodologie en 6 étapes** : ***S'informer*** . Je vais utilisé cette méthodologie durant tout le déroulement de ce TPI.

Je réalise que quelques points de l'énoncé ne sont pas clairs dont un particulièrement embêtant. Je décide de poser mes questions à mon référent durant la matinée.

Je commence à ***Planifier*** , seconde étape de la méthodologie utilisée. Pour cela, je sépare mes journée en tranches de 4 heures (des demis journées), et je remplis les différentes tranches horaires avec les *user stories* extraites de mon cahier des charges.

8h15 : je décide d'utiliser des alias afin de nommer les jours de travail mis à disposition pour le TPI : **J1** à **J11**. Voici les alias :

- J1 : lundi 25 mai 2020
- J2 : mardi 26 mai 2020
- J3 : mercredi 27 mai 2020
- J4 : jeudi 28 mai 2020
- J5 : vendredi 29 mai 2020
- J6 : mardi 2 juin 2020
- J7 : mercredi 3 juin 2020
- J8 : jeudi 4 juin 2020
- J9 : vendredi 5 juin 2020

- J10 : lundi 8 juin 2020
- J11 : mardi 9 juin 2020

8h25 : Lors de la création des *user stories* je remarque qu'il me faut décider d'une manière de prioriser les tâches. Je choisis la méthode **MoSCoW**. Cependant, les niveaux de priorité ne correspondent pas entièrement pour mon TPI. Je décide alors de modifier les intitulés :

- **Must** devient 🚫 **Bloquant**
- **Should** devient 💥 **Critique**
- **Could** devient ❗ **Important**
- **Won't** devient ❓ **Secondaire**

Je décide aussi d'utiliser la syntaxe suivante pour de présenter mes *user stories* :

Nom	**S<n° de la *story* > : <Nom de la *user story* >**
Description (*user story*)	<Description de la story pour connaître avec précision le but à atteindre>
Critère d'acceptation	<n° des tests à passé pour valider cette *story* >
Priorité	<Priorité de la *story* >

9h : Je fais un script bash me permettant un rassembler tous mes fichiers Markdown de ma documentation dans un seul et même fichier. Ceci est nécessaire car je prévois de publier ma documentation en ligne, à l'aide du site **readthedocs.org**.

10h : En plus de la documentation publique, il faut une version PDF. Pour ce faire ,j'utilise le logiciel **Typora** qui me permet d'exporter mon fichier réunissant toute ma documentation en PDF. Une fois cela fait, j'utilise un autre script bash permettant de fusionner plusieurs fichiers PDF en un seul. Ce dernier se nomme : `Rapport du TPI et documentation technique`. Il contient le rapport, les annexes, le résumé, l'énoncé, le journal de bord, et le code source.

10h30 : Je décris mes mes outils de bureautique. J'utilise **Typora** (un éditeur Markdown compatible sous tout OS) pour rédiger l'entièreté de ma documentation. La création des fichiers PDF est faite grâce à l'export vers PDF de Typora ainsi qu'à un script écrit par moi-même.

Concernant le style appliqué à ma documentation, j'utilise la couleur 🔵 #**006EDB** comme base. La police s'appelle Poppins. J'utilise cette police également dans le projet en lui-même.

10h50 : J'ai un rendez-vous GMeet avec mon formateur pour vérifier que tout va bien. Je pose la question suivante :

> Est-ce que le planning que vous m'avez donné est celui qu'il faudra utiliser ?

⊡ Mon formateur me répond que le planning donné est un modèle permettant de suivre de façon basique l'avancée du projet.

Je comprends que si j'ai un planning plus précis, je peux l'utiliser et comparer le mien avec celui qui m'a été donné.

11h25 : Je termine la rédaction de mon *product backlog* temporaire. Des modifications peuvent encore être apportées si j'en trouve le besoin.

11h45 : Je compile une version de test de ma documentation pour vérifier qu'il n'y ait pas d'erreur. Je prends ma pause de midi.

12h50 : Reprise de la journée. Je m'attaque maintenant au diagramme de Gantt. Je choisis de le réaliser avec un tableau HTML car je ne suis pas à l'aise avec les outils spécialisé comme Gantter.

14h15 : Je remarque un souci lors de la fusion des fichiers Markdown. Une partie d'un fichier se dédouble mais je ne sais pas encore pourquoi.

15h50 : Mon souci de duplication est résolu. Ce problème venait du fait que j'ajoutais ma table des matières sans supprimer le contenu précédent. Désormais, je supprime le contenu du fichier avant de le remplacer par le contenu mis-à-jour avec la table des matières. Je peux désormais commencer l'écriture des scénarios de tests fonctionnels.

16h45 : J'écris une partie des scénarios de tests. Il m'en reste encore quelques-un que j'ajouterai demain matin.

Bilan

La journée c'est plutôt bien passée. J'ai cependant pris un peu de retard sur la rédaction des scénarios de tests à cause de mon problème de duplication lors de la compilation de la documentation. Cependant, ceci reste un écart très minime sur mon planning. Je suis tout de même satisfait de cette première journée.

J2 : mardi 26 mai 2020

Objectifs

L'objectif de cette journée est premièrement de rattraper le petit retard que j'ai pris hier sur les scénarios de tests. Ensuite, je prévois de faire le modèle de base de données, de configurer l'application Flask, et de faire le code permettant d'importer les données.

Déroulement

8h : Je commence ma journée. Je finis les scénarios de tests que je n'avais pas pu terminer hier. Ceci ne prend pas beaucoup de temps.

8h30 : Je termine les scénarios. Je passe à la conception de la base de données. Grâce à l'énoncé, j'extrais les différentes tables du projet : `anime`, `status`, `type`, `url`, `list`, `list_has_anime`, `user`, `user_has_list`, `user_has_favorite`.

8h55 : Je réalise le modèle de base de données que voici :

9h : Je fais la partie *Base de donnée* du chapitre *Implémentation* de la documentation étant donné que j'ai toutes les informations nécessaire.

9h25 : Je termine de documenter la partie *Base de données*. J'y ai mis le modèle ci-dessus ainsi que le dictionnaire de données.

9h30 : Je configure l'application Flask pour pouvoir avoir un environnement de développement fonctionnel et pouvoir ainsi faire la suite du projet.

J'inscris une `secret_key` à l'application Flask. Cette clef est utilisée dans les systèmes d'encryptions. Flask lui-même n'a pas besoin de cette clef mais d'autres librairies externes, tel que `flask-login`, que j'utiliserai afin de pouvoir connecter un utilisateur, doit avoir cette clef. La valeur de cette clef est `Super` en Sha256.

9h55 : J'ai une application Flask basique fonctionnelle. Je peux rendre des vues depuis une route sans problèmes.

10h : Je mets en place le système de documentation automatique d'API : **Swagger**.

Pour que ce système puisse être mis en place, il faut une librairie externe nommée `flask_swagger`. De plus, certains fichiers sont indispensables au bon fonctionnement de Swagger. Le plus primordiale est la page HTML. Je décide de la nommé `endpoints.html` et de la placer dans le dossier `templates`. Cette page est disponible **ici**. Cependant, je l'ai adapté pour qu'elle soit correctement implémentée dans l'application FLask.

En plus de la page HTML, je rajoute 2 fichiers `javascript` qui vont dans le dossier `static/js`, 2 images qui vont dans le `static/img` et enfin 1 fichier `css` qui va dans le `static/css`. Voici le lien pour télécharger les fichiers :

- **swagger-ui-bundle.js**
- **swagger-ui-standalone-preset.js**
- **favicon-16x16.png**
- **favicon-32x32.png**
- **swagger-ui.css**

La structure du dossier `static` ressemble désormais à ceci :

```
Animanga
└── static
    ├── css
    │   └── swagger-ui.css
    ├── js
    │   ├── swagger-ui-bundle.js
    │   └── swagger-ui-standalone-preset.js
    └── img
        ├── favicon-16×16.png
        └── favicon-32×32.png
```

10h25 : J'installe la police Poppins en local. De ce fait, je n'aurai pas de soucis si le site perd la connexion à internet et que les polices sont chargées depuis Google Fonts.

10h30 : Je commence l'importation des données en base.

11h40 : Je termine l'importation des types et statuts des animes depuis le fichier JSON. Il me reste à faire l'importation des animes eux-même. Je prends ma pause de midi.

13h : Reprise de la journée et continuation de l'importation des données dans la base de données.

14h : Je termine l'import des données. Tout semble parfaitement bien s'ajouter en base. Étant donné que j'ai un peu d'avance, je mets ma documentation en ligne sur **readthedocs.org**. Le site hébergeant la documentation utilise **Mkdocs** pour convertir la documentation de Markdown à HTML. C'est pourquoi j'installe mkdocs dans **WSL 2** sur ma machine pour vérifier si ma documentation compile correctement.

14h15 : La documentation est en ligne à l'adresse : **https://animanga.readthedocs.io/fr/latest/**.

Pour le moment, étant donné que je n'ai pas encore mis en place la connexion, je ne peux effectuer mes tests que manuellement. Cependant, dès lors que la connexion sera mise en place, j'utiliserai **Katalon Recorder** pour automatiser mes tests.

14h25 : J'effectue le test *3.1* pour vérifier que mes données soient correctement importées. Comme j'ai de l'avance, je fais l'affichage de la *landing page* .

15h20 : Je termine l'implémentation de la *landing page* . Pour le moment, je ne peux tester le basculement de l'état connecté à l'état déconnecté que via la variable `is_authenticated` de que je change dans le fichier `routes.py`. Demain, je pourrai changer puisque je mettrai en place la connexion et l'inscription. Je n'aurai donc plus besoin de cette variable. Le test *11.1* passe concernant l'affichage de la *landing page* .

Je configure deux vérificateurs de syntaxe différents pour mon projet. Le premier est **pylint** pour Python, et le second est **eslint** pour le JavaScript. Pour eslint, j'utilise un preset de vérification : airbnb. Cela me permet d'écrire du code JavaScript dans un cadre syntaxique strict et donc de ne pas sortir des conventions actuelles.

15h30 : Eslint est installé et je lance la commande `npm run lint static/js` pour vérifier mes fichiers JavaScript.

15h35 : Je corrige les erreurs que eslint m'a montrées et je m'attelle maintenant à pylint.

16h : Le vérification de syntaxe pylint fonctionne correctement. Je lance la commande `python3 -m pylint --output-format=colorized packages` pour vérifier la syntaxe des fichiers se trouvant dans le dossier `packages`.

16h10 : Je corrige les erreurs relevées par pylint. Je mets à jour mon planning et mes scénarios de tests afin d'ajouter la vérification syntaxique.

16h30 : J'ajoute la vérification syntaxique dans le planning comme *user story* et je crée un test pour le Python et un pour le JavaScript afin de validé la *user story* .

Bilan

Je suis très content de l'avancement d'aujourd'hui. J'ai eu un peu de retard hier, mais je l'ai rapidement rattrapé ce matin. Je réussis à prendre un peu d'avance dans le projet et donc je suis très confiant pour la suite. Cela me permet de rajouter la vérification syntaxique pour Python et pour JavaScript.

J3 : mercredi 27 mai 2020

Objectifs

L'objectif de cette journée est de faire le système de connexion et d'inscription. Comme j'ai déjà fait l'affichage de la landing page hier, je pense avancer sur une autre tâche.

Déroulement

8h : Je commence à faire la partie inscription Je crée un contrôleur pour les utilisateurs pour mieux pouvoir gérer ces derniers.

8h30 : Je fais le formulaire HTML et je commence à faire la partie prise en charges de ce dernier.

9h : J'ai la visite de M. Terrond. M. Bouille étant pris à la Protection Civile, il ne peut pas être présent. Le but de cette visite est de voir si tout se passe bien et de répondre à d'éventuelles questions. Étant donné que tout ce passe bien pour moi, la visite se révèle être très brève !

9h40 : J'ai des soucis avec mon système d'export de documentation vers PDF. L'export ne s'effectue pas et je n'ai aucun moyens de savoir pourquoi. Je cherche activement ce qui pose problème.

10h20 : Je parviens à régler le problème d'export pour le moment mais rien ne me dit que cela n'arrivera pas de nouveau. Le cas échéant, j'ai un moyen auxiliaire d'exporter ma documentation. Je me remets à travailler sur la partie inscription.

11h45 : Je termine l'inscription. L'insère de nouvel utilisateur semble fonctionner. Avec les quelques minutes qu'il me reste avant la pause de midi, je mets en place des automations **Katalon Recorder** pour les futurs tests.

12h : Je termine l'automation du test fonctionnel d'inscription avec valeurs correctes. Je ferai les autres tests dès que j'aurai du temps durant la journée. Je prends ma pause de midi.

13h : Je me remets au travail en commençant la partie connexion. Étant donné le retard pris à cause du problème d'exportation vers PDF, je ne commence la connexion que maintenant. Ceci ne va causer d'autres retards car j'ai pris de l'avance hier concernant la *landing page* que j'aurais dû réaliser aujourd'hui.

13h40 : La *connexion* est terminée et fonctionne à merveille. Je crée maintenant des tests d'automations pour éviter de retaper tout le temps la même chose lors des futurs tests. J'utilise Katalon Recorder pour cela.

14h20 : J'optimise un peu le code pour l'inscription et la connexion étant donné que je n'ai pas pu le faire ce matin à cause de mon souci d'erreur d'export de la documentation.

15h20 : J'ai un rendez-vous avec mon référent TPI pour faire un point de la situation. Comme tout ce passe bien, le rendez-vous ne dure que très peu de temps et je retourne tout de suite travailler.

17h : Je termine l'optimisation de la validation des champs. Je me suis basé sur la libraire **wtforms**. Cette librairie est utilisée pour générer et valider automatiquement des formulaires. Comme cette libraire est trop imposante pour un TPI, je décide de m'inspirer de cette dernière uniquement pour la partie validation de formulaire. Je demanderai demain matin à mon référent s'il est d'accord que je garde cette manière de valider mes champs de formulaire ou si c'est toujours trop imposant.

En plus de cela, je configure Katalon Recorder pour prendre aussi en compte la connexion. J'ai maintenant un dossier à la racine de mon projet nommé `tests` qui contient le fichier HTML contenant tous les *tests cases* pour Katalon Recorder.

Bilan

Je suis plutôt content de ma journée. J'ai pu correctement faire le code de l'inscription ainsi que la connexion. De plus, comme j'avais du temps restant avant la fin de la journée, j'ai décidé de mettre en place Katalon Recorder afin d'automatiser mes tests fonctionnels. J'ai aussi décidé d'optimiser le code de validation des champs de mes formulaires. Je n'ai cependant toujours pas fait valider cette idée d'optimisation donc je le ferai demain matin.

J4 : jeudi 28 mai 2020

Objectifs

Le but de cette journée est de réaliser la partie recherche et affichage de la carte d'un anime. C'est un des aspects principaux de l'application.

Déroulement

8h : Après réflexion, je réalise que ce que j'ai entrepris la hier sur l'optimisation de la validation des champs était beaucoup trop imposant. Je décide de le refaire de manière bien plus légère.

9h20 : Je termine ma nouvelle version de l'optimisation des champs. Je suis très content du résultat, c'est bien plus simple à lire et à maintenir ! Je commence la partie recherche d'anime. C'est une des parties les plus importantes du projet.

Pour les recherches en base, j'ai une table virtuelle Sqlite3 pour pouvoir faire de la recherche **Fulltext**. Sqlite3 ne supporte pas le Fulltexte sur une table standard, il faut créer une table virtuelle avec un template supportant Fulltext. Tout le procédé est expliqué **ici**.

10h10 : La recherche par chaine de caractères est terminée. Il me reste à faire l'affichage. Ce n'est pas long. Je décide cependant de le faire cet après-midi préférant m'occuper de la récupération d'une anime aléatoire.

10h20 : J'ai un rendez-vous GMeet avec mon référent pour voir mon avancement et pour lui poser une question concernant la validation des champs. Je demande si ce que j'avais fait convient et si je peux le garder. Sa réponse est positive. Je continue mon projet.

10h40 : La récupération d'un anime aléatoirement est terminée. Je remarque que j'ai des **try/except** dans pratiquement toutes les méthodes de classes. Le **except** est toujours le même mais rien n'est centralisé. Je décide de faire une fonction centralisant tous les logs.

11h35 : La fonction de log est terminée. Je place dans tous les **except** un appel à cette méthode.

Comme j'ai encore un peu de temps avant la pause de midi, je commence à afficher les résultats de la recherche.

12h : Je termine l'affichage des résultats de la recherche et je prends ma pause de midi.

13h : Je reprends le travail en faisant la prise en charge de la barre de recherche.

13h20 : Je termine la prise en charge de la barre de recherche et je commence à faire l'affichage de la carte pour les animes. Pour la barre de recherche. J'opte pour deux manières différentes de l'afficher. La première, classique, est de cliquer sur la 🔍. Une modale s'affichage alors avec un champ de type de texte pour y entrer la recherche, ainsi qu'une croix sur la droite pour effacer le contenu du champ. La seconde manière est par le raccourci `Ctrl` + `S`.

14h30 : Jr termine d'afficher la carte (modale) de l'anime. Je commence la mise à jour de l'anime pour l'utilisateur connecté.

17h : Je n'ai pas terminé la mise à jour de l'anime Comme je dois, en principe, le faire demain, ce n'est pas un soucis.

Bilan

Cette journée était très sympa. Je n'ai pas eu de problème et j'ai pris un peu d'avance. Le journal n'est pas très rempli pour cet après-midi. Cependant, tout ce qu'il y a à savoir sur cette journée y est. Il ne c'est rien de particulier. Je suis très content d'avoir mis en place les automations Katalon car dès que je change quelque chose, je peux lancer les tests pour voir si rien n'a régressé.

J5 : vendredi 29 mai 2020

Objectif

Le but de cette journée est de faire la mise à jour de l'anime pour un utilisateur, d'afficher sa page de profil, et d'afficher ses favoris. Je prévois de faire l'affichage des favoris en même temps que la mise à jour de l'anime pour pouvoir avoir un retour visuel sur le bon fonctionnement ou pas du code.

Déroulement

8h : Je commence ma journée par la mise à jour de l'anime. Ne pensant pas que cela va me prendre toute la matinée comme prévu, je vais faire la partie affichage des favoris en même temps.

9h30 : Je termine la partie mise à jour de l'état de favoris pour les animes. Je les affiche pour pourvoir avoir un retour visuel et aussi pour avoir un élément à tester en case de réussite lors des tests d'automation.

9h55 : Je termine l'affichage des favoris sur la page d'accueil. L'affichage spécifique des favoris est sur la page de profil donc pour le moment il n'y a que la page d'accueil pour voir ses propres favoris. La gestion de ces derniers ne viendra que plus tard. Je commence la mise à jour du statut de visionnement de l'anime.

10h15 : J'ai un rendez-vous GMeet avec mon référant pour vérifier mon avancement. Je lui montre ce que j'ai fait les jours précédents et ce que je suis en train de faire. La seule remarque concerne le planning. Je dois réécrire les dates car les numéros des jours sont faux et je dois mettre les cases en 🔴 #**F34C56** si elles n'étaient pas prévues dans le planning prévisionnel.

11h40 : Je termine la mise à jour du statut de visionnement des animes pour l'utilisateur connecté. Je dois encore afficher quel statut l'anime a quand on clique sur son nom pour afficher sa carte. Je décide de le faire maintenant. Ainsi, je ne prend pas de retard.

12h : Je termine l'affichage du statut correct dans le combo box de la carte de l'anime. Il me reste seulement à afficher les listes personnelles à l'utilisateur. Je ferai cela au moment où la tâche apparaîtra dans le diagramme de Gantt. Je prend ma pause de midi.

13h : Suite au rendez-vous GMeet, je me souviens que mon référent m'avait dit de mettre des captures d'écrans de Katalon dans mon journal de bord. Voici donc une capture d'écran montrant tous les tests créés depuis le début et leur état :

Tout les tests passent. Je n'ai pas encore mis les tests concernant la mise à jour d'un anime mais cela ne saurait tarder. Pour continuer mon travail, je m'occupe de l'affichage de la page de profil. Les images utilisées comme bannière de fond ainsi que l'image de profil sont arbitraires. Il serait préférable de faire un système d'upload pour chacune des images lors d'une amélioration du projet.

13h40 : Je termine l'affichage de la page de profil de l'utilisateur. En plus de seulement pouvoir afficher la page de profile de l'utilisateur en allant sur l'url `/profile/<pseudo>`, je fais en sort de rediriger l'utilisateur sur `/profile/<pseudo utilisateur connecté>` s'il va sur `/profile` ou `/profile/`. Je mets les liens pour les futures pages d'affichage des listes ainsi que pour la gestion des favoris. Pour le moment, les liens redirigent vers une 404. J'améliorerai cela sous peu. De plus, le lien `favoris` ne s'affiche que si l'on est sur sa propre page de profil. J'ajoute le contenu de la page profile, à savoir les statistiques de l'utilisateurs ainsi que ses favoris.

14h : J'ajoute les favoris sur la page de profil. Ils s'affichent de manière différente que sur la page d'accueil. En effet, c'est bien plus ergonomique horizontalement que verticalement sur une page telle que la page de profil. J'affiche les statistiques de l'utilisateur.

16h45 : Je termine d'afficher les statistiques de l'utilisateur. Ces dernières fonctionnent de cette manière : elles ne sont basées que sur les animes marqués comme `Complétés`. Je compte chaque type et je les affiche. J'ai alors quelque chose comme ceci :

```
{
    'TV': 2,
    'Movie': 1,
    'Ona': 0,
    'Special': 1
}
```

Je termine ma journée.

Bilan

Cette journée c'est très bien passée. J'ai pu faire tout ce qu'il fallait pour la journée et aucun bug ne s'est présenté pour le moment.

J6 : mardi 2 juin 2020

Objectif

Cette journée est relativement remplie. Premièrement, je dois afficher le contenu des listes d'un utilisateur. Ensuite, je dois m'occuper de la gestion des listes. Cela comprend l'ajout de nouvelles listes, la suppression des listes existantes et le renommage des listes existantes. Enfin, je prévois de faire la gestion de l'ordre des favoris.

Déroulement

8h : Je commence par l'affichage du contenu des listes d'un utilisateur. Une route doit être faite pour récupérer les animes d'une liste.

9h : Je remarque que le javascript ordonne automatiquement les listes lorsqu'il les reçoit via un fetch. Je mets donc l'id de la liste devant le nom pour que l'ordre soit fait directement sur l'id et non pas sur le nom de la liste. Cela permet de garder l'ordre de création des listes.

9h30 : Je termine l'affichage des listes. Dès que l'on clique sur le nom d'une liste, le site montre les animes contenus dans la liste cliquée. Par défaut, c'est `Tous` qui est sélectionné, ce qui permet de voir tous les animes de toutes les listes de l'utilisateur.

Je commence la gestion des listes.

10h : Je termine l'ajout de nouvelles listes. Cependant, je dois changer la manière de récupération des listes car l'ordre d'affichage est faux.

10h30 : J'ai un rendez-vous GMeet avec mon référant pour faire le point. Je lui explique sur quoi je travaille. Je lui dis que je n'ai pas de souci. Le rendez-vous se termine rapidement.

11h : Je résous le souci d'ordre d'affichage des listes. Ma route `/get/animes` renvoie le json suivant:

```
{
    'Complétés': [
        animes ...
    ],
    'En cours': [
        animes ...
    ],
    listes ...  : [
        animes ...
    ]
}
```

Les listes présentes dans le json ne sont que les listes contenant des animes.

Je commence la suppression des listes et de leur contenu.

11h40 : La suppression des listes et de leur contenu est terminée. Il ne me reste plus qu'à faire le renommage dans lequel je me lance.

12h : Je n'ai pas encore terminé la mise à jour du nom de la liste mais il ne me reste que la partie base de données à faire. Toute la partie logique est faite. Je poursuiverai cet après-midi.

13h : Je recommence la journée en travaillant sur la mise à jour du nom d'une liste.

13h10 : Je termine la mise à jour du nom d'une liste et je passe à la mise à jour de l'ordre des favoris.

14h : La mise à jour de l'ordre des favoris est terminée. Je m'occupe de l'affichage des activités de l'utilisateur connecté. L'ordre des favoris est modifié grâce à la librairie **JQueryUI**, qui permet rendre une `div` capable de prendre en charge du drag&drop. J'utilise le drag&drop pour déplacer les animes dans l'ordre que l'utilisateur veut sur la page des favoris.

Cette tâche n'est pas présente dans le planning prévisionnel car je l'avais oubliée lors de la rédaction du planning. Je l'ai cependant rajoutée dans le *product backlog* . C'est pourquoi il n'y a pas de case orange dans le planning prévisionnel mais uniquement une case rouge.

15h30 : Les activités sont implémentées. J'affiche toutes les activités des 24 dernières heures : les animes mis en favoris et les animes mis dans des listes. Ces activités sont présentes sur la page d'accueil si l'utilisateur n'a fait aucune recherche.

16h : Au début de la session de TPI, j'ai mis en place une méthode permettant de récupérer un aime aléatoire. J'ai maintenant terminé d'affiché cette méthode sur le site via un bouton placé dans la barre de navigation.

16h45 : Je termine l'affichage de l'anime tiré aléatoirement et clos ma journée.

Bilan

Comme prévu, cette journée était plutôt bien remplie. J'ai réussi à faire plusieurs tâches indispensables et toutes fonctionnent. Il ne me reste plus que la partie synchronisation à faire au niveau technique de l'application. Il me restera ensuite les partie de documentation, d'aide, et de page à propos à terminer. Je suis content du travail effectué aujourd'hui.

J7 : mercredi 03 juin 2020

Objectif

Le but de cette journée est de commencer une des parties principales de l'application : la synchronisation entre Sqlite3 et MySQL. En effet, cette partie est de très loin la plus complexe à mettre en place car aucune libraire ne permet de le faire automatiquement. C'est pour cela que je dois moi-même penser à un algorithme de synchronisation et le mettre en place. D'où les 4 jours de planification.

Déroulement

8h30 : J'ai un rendez-vous avec mes experts (M. Terrond et M. Bouille) afin de faire le point sur mon avancement dans le travail. Je leur explique le cas de la tâche manquante dans le planning prévisionnel, et je leur montre le planning dans son ensemble pour qu'ils aient une idée globale de l'avancement. Ils souhaitent que je leur fasse une démonstration. C'est alors que je décèle un bug apparu après avoir corrigé ma syntaxe hier vers 16h. Je n'avais alors pas retesté le bon fonctionnement de mon application et le bug c'est manifesté ce matin.

M. Bouille me suggère de faire une slide spécifique dans ma présentation pour toutes les fonctionnalités qui pourraient être rajoutées ou améliorées.

9h10 : Le rendez-vous est terminé. Je corrige les bugs rencontrés lors de la démonstration.

9h30 : Les bugs sont corrigés. Rien de grave, simplement une variable mal initialisée ainsi qu'un id HTML pas exclu lors d'un test javascript. Je commence à réfléchir à l'algorithme que je pourrai utilisé lors de la synchronisation.

10h30 : Je fais une première version d'algorithme. Elle n'est de loin pas optimisée. Je continue donc à chercher. Voici la première version :

```
foreach table {
    stocker le nom de la table courante
    récupérer les enregistrements et les stocker

    découper les données en parties de 5k enregistrements
    // [
    //   [
    //     {données}
    //      ... x5k
    //   ]
    //   ... autant de tableau de 5k données pour avoir toutes les données
    // ]

    foreach paquet in tout les paquets découpé {
        inserer les données dans la table courante MySQL via un insert multiple
        // INSERT INTO table( ... colonnes) VALUES( ... values), ( ... values), ...
    }
}
```

J'essaye d'améliorer mon algorithme pour le rendre moins coûteux en ressources et plus rapide.

11h40 : Je travaille encore sur la seconde version de mon algorithme. Je continuerai cet après-midi.

13h : Je reprends ma journée et je continue l'élaboration de mon algorithme de synchronisation.

13h10 : J'ai une seconde version de mon algorithme. Le voici :

```
foreach table {
    stocker le nom de la table courante

    récupérer le nombre d'enregistrements de Sqlite3 et le stocker
    récupérer le nombre d'enregistrements de MySQL et le stocker

```

```
    if nombre enregistrements Sqlite3 égale nombre enregistrements MySQL {
        passé à la table suivante
    } else {
        vider la table MySQL

        découper les données en parties de 5k enregistrements

        foreach paquet in tout les paquets découpé {
            inserer les données dans la table courante MySQL via un insert multiple
        }
    }
}
```

Je suis persuadé que je peux faire autrement que de découper mes données en paquets de 5000 enregistrements. Je continue a chercher une manière plus efficace.

16h : Je pense avoir une version presque finale de l'algorithme. Je décide de travailler avec les timestamps que j'ai en base. Je ne sais pas pourquoi je n'y avais pas pensé plus tôt mais avec les timestamps, je peux savoir quand est-ce que les données ont été altérées pour la dernière fois et donc trier plus facilement les enregistrements à modifier, $ supprimer, ou à ajouter.

Il se peut qu'il y ait des modifications à venir sur cet algorithme mais voici la version actuelle avec laquelle je pense continuer l'application :

```
foreach table {
    stocker le nom de la table courante

    récupérer le nombre d'enregistrements de Sqlite3 et le stocker
    récupérer le nombre d'enregistrements de MySQL et le stocker

    if nombre enregistrements Sqlite3 différent de nombre enregistrements MySQL {
        récupérer ids et timestamp Sqlite3 pour stocker dans tableau id: timestamp
        récupérer ids et timestamp MySQL pour stocker dans tableau id: timestamp

        trier les tableaux par id

        if tableau Sqlite3 > tableau MySQL {
            stocker ids Sqlite3 non présent dans tableau MySQL
            retirer les ids ci-dessus du tableau Sqlite3

            récupérer les enregistrements Sqlite3 correspondant aux ids non présent MySQL
```

```

            foreach enregistrement Sqlite3 {
                insérer dans MySQL
            }
        } else {
            stocker ids MySQL non présent dans Sqlite3

            foreach id MySQL {
                supprimer l'entrée MySQL
            }
        }
    }

    déclarer tableau pour ids Sqlite3 dont timestamp diffère du MySQL
    foreach enregistrements Sqlite3 {
        if timestamp courant différent du timestamp MySQL correspondant {
            ajouter l'id dans le tablea créer à cet effet
        }
    }

    récupérer les enregistrements Sqlite3 correspondant aux ids
    foreach enregistrement {
        mettre à jour l'enregistrement correspondant dans MySQL
    }
}
```

J'implémente cet algorithme dans l'application.

17h : Je n'ai pas terminé l'implémentation. Je continuerai demain. Je clos ma journée.

Bilan

Cette journée n'a pas été très variée. Je n'ai fait que développé mon algorithme et j'ai à peine commencé à l'implémenter dans l'application. Je terminerai l'implémentation demain je pense.

J8 : jeudi 04 juin 2020

Objectif

Aujourd'hui, je prévois de terminer l'implémentation de l'algorithme de synchronisation. Je suis entièrement dans les temps et très content de la vitesse d'avancement du projet.

Déroulement

8h10 : Je commence ma journée. J'implémenter la synchronisation entre les bases de données.

9h30 : J'ai un rendez-vous avec mon référent pour faire le point. Comme tout ce passe bien, le rendez-vous ne dure que très peu de temps. Mon référent va cependant prendre un moment pour regarder le code et nous allons faire une conférence pour corriger les parties qui ont besoin de modifier.

10h30 : Un élève (Vincent Steinmann) me demande de l'aide pour son projet. Comme j'ai de l'avance, je prend l'initiative de l'aider un moment.

11h30 : J'ai terminé d'aider Vincent. Cette aide ne m'a pas fait prendre de retard sur mon planning donc tout va bien. Je continue à implémenter la synchronisation.

12h : Je prends ma pause de midi.

13h : Je reprends l'implémentation de l'algorithme.

14h : Je remarque, lors de la mise à jour des données dans MySQL, que j'ai stocké les timestamp avec les millisecondes dans Sqlite3. Cela pose problème pour MySQL car lorsque je mets les nouveaux enregistrements, les timestamps sont arrondis.

> Exemple:
>
> J'ai un enregistrement dans Sqlite3 dont le timestamp est de 2020-05-04 14:28:30.67293
>
> Dans MySQL, il deviendra : 2020-05-04 14:28:31

Je modifie par conséquent le code de l'ajout des timestamp dans Sqlite3 pour retirer les millisecondes à la source directement. Je remplace tous les `dt.now()` par `dt.now().strftime('%Y-%m-%d %H:%M:%S')`.

16h : La synchronisation entre les bases est terminée. Je gagne 1 jour d'avances grâce à cela. J'en profite pour corriger tous les potentiels bugs qui pourraient y avoir et faire de la documentation. De plus, ça me permettra de vérifier que je n'ai rien oublié des points mentionnés dans le cahier des charges.

Je mets à jour la documentation.

17h : Je termine ma journée.

Bilan

Cette journée a été très fructueuse. J'ai pu finir la synchronisation et j'ai 1 jour d'avance sur mon planning. Cela m'a permis de terminer les fonctionnalités de mon application. Maintenant, je vais vérifier que je n'ai rien oublié et corriger les potentiels bugs qui pourraient se présenter.

J9 : vendredi 05 juin

Objectif

Le but de cette journée est de faire de la documentation. Je vais vérifier si toutes les fonctionnalités sont correctement implémentées.

Déroulement

8h : Je remarque que je me suis trompé dans le numéro des jours sur mon planning. Ceci m'a rendu confus lors de la rédaction du journal de bord. Les jours indiqués dans le planning étaient : J1, J2, J3, J4, J5, J7, J8, J9, J10, J11, J12.

Je n'ai pas remarqué cela lors de mes mises à jour du planning et donc les plannings donnés précédemment aux experts étaient erronés.

Je corrige cette erreur.

8h30 : Le planning est corrigé. J'ajoute la partie *libraires et outils externes* .

10h : Les librairies sont documentées et j'ajoute la structure du projet dans la partie *implémentation* .

10h30 : Je remarque que je pourrais ajouter dans les scénarios de tests, toutes les possibilités d'erreurs pour l'inscription et la connexion. Cela ajoute ~10 tests supplémentaires. Étant donné que je suis relativement à jour dans la documentation, je vérifie si toutes les fonctionnalités sont bien implémentées.

10h35 : Il manque une fonctionnalité : la possibilité de supprimer un animes des favoris depuis la page d'organisation des favoris. Je décide de l'implémente.

11h50 : La possibilité de supprimer un animes des favoris depuis la page d'organisation des favoris est implémentée. Pour pouvoir faire ceci, j'ai mis une icône, ⛔, pour pouvoir retirer les animes. Cette icône s'affiche lorsque l'utilisateur clique sur le bouton `Réorganiser les favoris`.

Je prends ma pause de midi.

13h : Je reprends ma journée et je vérifie tout le reste du projet pour m'assurer qu'il n'y a aucun bug.

14h30 : Toute l'application est propre. Je n'ai détecté qu'un seul bug et je l'ai corrigé très rapidement. Un simple problème de type. Je m'attelle à la documentation. Cela comprend corrections orthographiques et ajout de parties manquantes si le cas se présente.

Code source

```python
"""Contient le coeur de l'application

Toutes les routes de l'appplication sont décrites dans ce fichier.

@author: Tanguy Cavagna
@copyright: Copyright 2020, TPI
@version: 1.0.0
@date: 2020-05-26
"""
from flask import Flask, url_for, redirect, flash, g
from flask_login import LoginManager
from dotenv import load_dotenv

from .config import DevConfig, ProdConfig
from .packages.controllers.UserController import UserController

app = Flask(__name__)
app.config.from_object(DevConfig())
app.secret_key = "8185c8ac4656219f4aa5541915079f7b3743e1b5f48bacfcc3386af016b55320"

login_manager = LoginManager()
login_manager.init_app(app)

# Chargement des routes
with app.app_context():
    from . import routes, auth

    # Enregistre les routes
    app.register_blueprint(routes.main_bp)
    app.register_blueprint(auth.auth_bp)

# ================
#      OTHER
# ================
@login_manager.user_loader
def load_user(user_id):
    """Test si l'utilisateur est connecté sur toutes les pages. Si oui, retourne l'utilisateur"""
    if user_id is not None:
        return UserController().get_by_id(user_id)
    return None

# ================
```

```python
#        Run
# ================
if __name__ == "__main__":
    app.run()
```

```python
"""Contient les routes d'authentifiation de l'application

@author: Tanguy Cavagna
@copyright: Copyright 2020, TPI
@version: 1.0.0
@date: 2020-05-27
"""
import hashlib
import re
from typing import Any

from flask import redirect, render_template, flash, Blueprint, request, url_for
from flask_login import login_required, current_user, login_user, logout_user

from .packages.controllers.UserController import UserController
from .packages.controllers.authentication import SignupController, LoginController

# Configuration du Blueprint
auth_bp = Blueprint('auth_bp', __name__,
                    template_folder='templates',
                    static_folder='static')

@auth_bp.route('/signup', methods=['GET', 'POST'])
def signup():
    """Affiche la page d'inscription
        \nGET: Affiche la page
        \nPOST: Si les informations de connexion sont valides, redirection sur l'accueil
    """
    auth_validation = SignupController(request.form)

    if request.method == 'POST':
        if auth_validation.validate():
            email = request.form.get('email')
            nickname = request.form.get('nickname')
            password = request.form.get('password')

            if not UserController().exists(email):
                hashed_password = hashlib.sha256(password.encode('utf8')).hexdigest()
                user = UserController().insert(email, nickname, hashed_password)
                UserController().setup_default_lists(user.id)
```

```
41
42                 if user is not None:
43                     login_user(user)
44                     return redirect(url_for('main_bp.home'))
45
46             flash('Un utilisateur utilise déjà cet email')
47             return redirect(url_for('auth_bp.signup'))
48
49     return render_template('signup.html',
50                            current_user=current_user,
51                            auth_errors=auth_validation.
   get_state())
52
53 @auth_bp.route('/login', methods=['GET', 'POST'])
54 def login():
55     """Affiche la page de connexion
56         \nGET: Affiche la page
57         \nPOST: Si les informations de connexion sont valides
   , redirection sur l'accueil
58     """
59     auth_validation = LoginController(request.form)
60
61     if request.method == 'POST':
62         if auth_validation.validate():
63             email = request.form.get('email')
64             password = request.form.get('password')
65             hashed_password = hashlib.sha256(password.encode('
   utf8')).hexdigest()
66
67             if UserController().check_password(email,
   hashed_password):
68                 user = UserController().get_by_email(email)
69                 login_user(user)
70                 return redirect(url_for('main_bp.home'))
71
72             flash('Combinaison email - mot de passe invalide')
73             return redirect(url_for('auth_bp.login'))
74
75     return render_template('login.html',
76                            current_user=current_user,
77                            auth_errors=auth_validation.
   get_state())
78
79 @auth_bp.route('/logout', methods=['GET'])
80 @login_required
81 def logout():
82     """Déconnecte l'utilisateur"""
```

```
    logout_user()
    return redirect(url_for('auth_bp.login'))

```

```python
"""Contient les différentes configuration du serveur Flask

@author: Tanguy Cavagna
@copyright: Copyright 2020, TPI
@version: 1.0.0
@date: 2020-05-26
"""
class ProdConfig:
    TESTING = False
    DEBUG = False
    TEMPLATES_AUTO_RELOAD = False

class DevConfig:
    TESTING = True
    DEBUG = True
    TEMPLATES_AUTO_RELOAD = True
```

```python
"""Contient les routes de l'application

@author: Tanguy Cavagna
@copyright: Copyright 2020, TPI
@version: 1.0.0
@date: 2020-05-26
"""
from flask import render_template, Response, stream_with_context, jsonify, request, Blueprint, redirect, url_for
from flask import current_app as app
from flask_swagger import swagger
from flask_login import current_user, login_required
import markdown, os

from .packages.controllers.AnimeController import AnimeController
from .packages.controllers.ListController import ListController
from .packages.controllers.UserController import UserController
from .packages.controllers.ActivitiesController import ActivitiesController
from .packages.controllers.SqliteController import SqliteController

# Configuration du Blueprint
main_bp = Blueprint('main_bp', __name__,
                    template_folder='templates',
                    static_folder='static')

# ================
#      STREAM
# ================
def stream_template(template_name, **context):
    """Le but est de récupérer l'objet template de Jinja2 et d'appeler la fonction stream() qui renvoie un stream au lieu de render() qui renvoie un string.
    Étant donné que l'on bypass le template de Flask, on doit être sûre de mettre à jour de template nous meme en appelant update_template_context().
    Le template sera évalué chaque fois que le stream sera modifier. À chaque yield, le serveur va envoyer le contenu au client et donc modifier le stream.

    Arguments:
        template_name {string} -- Nom du template a appeler
```

```
        **context -- Tous les autres argument mixes représentant les variables à envoyer au template

    Returns:
        stream -- Stream à envoyer
    """
    # https://flask.palletsprojects.com/en/1.1.x/patterns/streaming/
    app.update_template_context(context)
    t = app.jinja_env.get_template(template_name)
    rv = t.stream(context)
    return rv

# ================
#       GET
# ================
@main_bp.route('/')
def home():
    """Affiche la page d'accueil
    """
    return render_template('index.html',
                           current_user=current_user)

@main_bp.route('/search/<string:search_string>', methods=['GET'])
@login_required
def search(search_string: str = None):
    """Recherche un anime
    """
    # Parse la chaine de recherche pour tansformer la chaine comme suit (facilite la recherche fulltext) :
    # Chaine de base : Ma chaine de recherche
    # Chaine parsée : "Ma" "chaine" "de" "recherche"
    search_string_parsed = ' '.join(['"' + escaped + '"' for escaped in search_string.split(' ')])
    searched_animes = AnimeController().get_from_search_string(search_string_parsed)

    # Ajout des relations
    for anime in searched_animes:
        anime['relations'] = AnimeController().get_relations_by_anime_id(anime['id'])
        anime['is_favorite'] = AnimeController().is_anime_in_user_favorite(current_user.id, anime['id'])
        anime['in_list'] = [l['id'] for l in ListController().get_list_of_an_anime(anime['id'])]

```

```python
    return render_template('index.html',
                           current_user=current_user,
                           search_string=search_string,
                           search_results=searched_animes,
                           user_list=ListController().get_user_lists(current_user.id))

@main_bp.route('/random', methods=['GET'])
@login_required
def random():
    """Affiche un anime random
    """
    random_anime = AnimeController().get_random_anime()

    random_anime['relations'] = AnimeController().get_relations_by_anime_id(random_anime['id'])
    random_anime['is_favorite'] = AnimeController().is_anime_in_user_favorite(current_user.id, random_anime['id'])
    random_anime['in_list'] = [l['id'] for l in ListController().get_list_of_an_anime(random_anime['id'])]

    return render_template('index.html',
                           current_user=current_user,
                           search_string='aléatoire',
                           search_results=[random_anime],
                           user_list=ListController().get_user_lists(current_user.id))

@main_bp.route('/profile/<string:nickname>', methods=['GET'])
@login_required
def profile(nickname: str = None):
    """Affiche la page de profile de l'utilisateur mentionner dans l'url
    """
    if nickname is None:
        nickname = current_user.nickname

    searched_user = UserController().get_by_nickname(nickname)

    if searched_user is not None:
        return render_template('profile.html',
                               searched_user=searched_user,
                               stats=UserController().get_stats_by_id(searched_user.id))

```

```python
    return redirect(url_for('main_bp.home'))

@main_bp.route('/profile', methods=['GET'])
@main_bp.route('/profile/', methods=['GET'])
@login_required
def profile_redirect():
    """Redirige l'utilisateur sur sa page de profile si aucun pseudo indiqué dans l'url
    """
    return redirect(url_for('main_bp.profile', nickname=current_user.nickname))

@main_bp.route('/lists/<string:nickname>')
@login_required
def lists(nickname: str = None):
    """Affiche toutes les liste et leur contenu de l'utilisateur mentionné dans l'url
    """
    if nickname is None:
        nickname = current_user.nickname

    searched_user = UserController().get_by_nickname(nickname)

    if searched_user is not None:
        user_lists = ListController().get_user_lists(searched_user.id)
        return render_template('lists.html',
                               current_user=current_user,
                               user_lists=user_lists,
                               searched_user=searched_user)

    return redirect(url_for('main_bp.home'))

@main_bp.route('/lists')
@main_bp.route('/lists/')
@login_required
def lists_redirect():
    """Redirige l'utilisateur si aucuns pseudo n'est inscrit dans l'url
    """
    return redirect(url_for('main_bp.lists', nickname=current_user.nickname))

@main_bp.route('/favorites/<string:nickname>')
@login_required
def favorites(nickname: str = None):
```

```python
    """Affiche tout les favoris de l'utilisateur mentionné dans l'url
    """
    if nickname is None:
        nickname = current_user.nickname

    searched_user = UserController().get_by_nickname(nickname)

    if searched_user is not None:
        user_lists = ListController().get_user_lists(searched_user.id)
        return render_template('favorites.html',
                               current_user=current_user,
                               searched_user=searched_user)

    return redirect(url_for('main_bp.home'))

@main_bp.route('/favorites')
@main_bp.route('/favorites/')
@login_required
def favorites_redirect():
    """Redirige l'utilisateur si aucuns pseudo n'est inscrit dans l'url
    """
    return redirect(url_for('main_bp.favorites', nickname=current_user.nickname))

@main_bp.route('/about')
def about():
    """Affiche la page à propos
    """
    with open(os.path.dirname(os.path.realpath(__file__)) + '/docs/librairies.md', 'r', encoding='utf-8') as input_file:
        text = input_file.read()

    return render_template('about.html', current_user=current_user, html=markdown.markdown(text))

@main_bp.route('/help')
def help():
    """Affiche la page d'aide
    """
    with open(os.path.dirname(os.path.realpath(__file__)) + '/docs/aide.md', 'r', encoding='utf-8') as input_file:
        text = input_file.read()

```

```
190     return render_template('help.html', current_user=
    current_user, html=markdown.markdown(text))
191
192 @main_bp.route('/get/favorites', methods=['GET'])
193 @login_required
194 def get_favorites_for_loged_user():
195     """
196     Récupère tous les favoris de l'utilisateur connecté
197     ---
198     tags:
199       - favorites
200     responses:
201       200:
202         $ref: '/static/swagger-components/components.yml#/
    components/responses/LoggedUserFavorites'
203     """
204     return jsonify({'animes': AnimeController().
    get_favorite_by_user_id(current_user.id)})
205
206 @main_bp.route('/get/favorites/<string:nickname>', methods=['
    GET'])
207 @login_required
208 def get_favorites_for_user(nickname = None):
209     """
210     Récupère tous les favoris d'un utilisateur
211     ---
212     tags:
213       - favorites
214     responses:
215       200:
216         $ref: '/static/swagger-components/components.yml#/
    components/responses/LambdaUserFavorites'
217     """
218     searched_user = UserController().get_by_nickname(nickname
    )
219     if searched_user is not None:
220         user_id = searched_user.id
221         return jsonify({'animes': AnimeController().
    get_favorite_by_user_id(user_id)})
222
223     return jsonify({'animes': []})
224
225 @main_bp.route('/endpoints', methods=['GET'])
226 def endpoints():
227     """Affiche les points de sorties de l'api. Comprend
    uniquement les urls documentées avec de la docstring swagger
228     """
```

```
    return render_template('endpoints.html')

@main_bp.route("/specs", methods=['GET'])
def spec():
    """
    Retourne un JSON comportant toutes les informations contenues dans les docstring des fonctions
    ---
    tags:
      - API
    response:
      200:
        description: objet JSON comportant toutes les informations nécessaire à l'affichage des points de sortie de l'API
    """
    swag = swagger(app)
    swag['info']['version'] = "1.0"
    swag['info']['title'] = "Animanga"
    return jsonify(swag)

# J'ai été obliger de mettre la route en GET, car ce n'est pas possible de streamer des infos si elles viennent d'un post
# Lors de l'affichage des specs - '/enpoints' - j'ai remplacer le type de methode de GET à PUT pour une meilleur comprehention de l'api
# La fonction stream_with_context() permet de garder le contexte de la requete qui se fait supprimer en tant normale. Indispensable lors de stream avec template contenant un layout
@main_bp.route('/override')
def override_datas():
    """
    Écrase les données existantes de la bd avec de nouvelles données
    ---
    tags:
      - animes
    responses:
      200:
        description: Remplacement effecuté et redirection sur l'index
    """
    current_directory = '/'.join(__file__.split('/')[:-1])
    return Response(stream_with_context(stream_template('index.html',
```

```
263
    override_messages=AnimeController().override_local_with_json(
    current_directory + '/static/json/anime.json'))))
264
265 @main_bp.route('/sync')
266 @login_required
267 def sync():
268     """Synchronise les données entre Sqlite3 et MySQL
269     """
270     SqliteController().synchronise_with_mysql()
271
272     return redirect(url_for('main_bp.home'))
273
274 # ================
275 #      POST, PATCH, PUT, DELETE
276 # ================
277 @main_bp.route('/set/favorite', methods=['PATCH'])
278 @login_required
279 def set_favorite():
280     """
281     Met à jour le status de favoris pour un anime de l'
    utilisateur connecté
282     ---
283     tags:
284       - favorites
285     parameters:
286       - $ref: '/static/swagger-components/components.yml#/
    components/parameters/idAnime'
287     responses:
288       200:
289         $ref: '/static/swagger-components/components.yml#/
    components/responses/GenericResponse'
290     """
291     anime_id = request.get_json()['idAnime']
292
293     if anime_id is not None:
294         return jsonify({'Status': AnimeController().
    set_anime_in_user_favorite(current_user.id, anime_id)})
295
296     return jsonify({'Status': False})
297
298 @main_bp.route('/set/list', methods=['PUT'])
299 @login_required
300 def set_list():
301     """
302     Met à jour le status de favoris d'un anime pour l'
    utilisateur connecté
```

```
    ---
    tags:
      - lists
    parameters:
      - $ref: '/static/swagger-components/components.yml#/components/parameters/idAnime'
      - $ref: '/static/swagger-components/components.yml#/components/parameters/idList'
    responses:
      200:
        $ref: '/static/swagger-components/components.yml#/components/responses/GenericResponse'
    """
    id_anime = request.get_json()['idAnime']
    id_list = request.get_json()['idList']

    if id_anime is not None:
        return jsonify({'Status': AnimeController().set_anime_in_list(id_anime, id_list)})

    return jsonify({'Status': False})

@main_bp.route('/delete/defaults', methods=['DELETE'])
@login_required
def delete_status():
    """
    Supprime l'anime des listes par défaut de l'utilisateur connecté
    ---
    tags:
      - animes
    parameters:
      - $ref: '/static/swagger-components/components.yml#/components/parameters/idAnime'
    responses:
      200:
        $ref: '/static/swagger-components/components.yml#/components/responses/GenericResponse'
    """
    anime_id = request.get_json()['idAnime']

    if anime_id is not None:
        return jsonify({'Status': AnimeController().delete_anime_status(current_user.id, anime_id)})

    return jsonify({'Status': False})

```

```python
@main_bp.route('/get/animes', methods=['POST'])
@login_required
def get_animes_from_list():
    """
    Récupère les animes d'une liste
    ---
    tags:
      - animes
    parameters:
      - $ref: '/static/swagger-components/components.yml#/components/parameters/userNickname'
      - $ref: '/static/swagger-components/components.yml#/components/parameters/idList'
      - $ref: '/static/swagger-components/components.yml#/components/parameters/search_terms'
    responses:
      200:
        $ref: '/static/swagger-components/components.yml#/components/responses/AnimeListResponse'
    """
    if request.json.get('nicknameUser') is not None and request.json.get('nicknameUser') != current_user.nickname:
        user_id = UserController().get_by_nickname(request.json.get('nicknameUser')).id
    else:
        user_id = current_user.id

    return jsonify({ 'animes': AnimeController().get_animes_in_list_for_user(user_id, request.json.get('idList'), request.json.get('search-terms')) })

@main_bp.route('/add/list', methods=['PUT'])
@login_required
def add_list():
    """
    Ajoute une nouvelle liste à l'utilisateur connecté
    ---
    tags:
      - lists
    parameters:
      - $ref: '/static/swagger-components/components.yml#/components/parameters/newListName'
    responses:
      200:
        $ref: '/static/swagger-components/components.yml#/components/responses/ListResponse'
    """
```

```python
    return jsonify({ 'list': ListController().add_new_list(request.json.get('newListName'), current_user.id) })

@main_bp.route('/delete/list', methods=['DELETE'])
@login_required
def delete_list():
    """
    Supprime une liste à l'utilisateur connecté
    ---
    tags:
      - lists
    parameters:
      - $ref: '/static/swagger-components/components.yml#/components/parameters/idList'
    responses:
      200:
        $ref: '/static/swagger-components/components.yml#/components/responses/GenericResponse'
    """
    return jsonify({ 'Status': ListController().delete_by_id(request.json.get('idList'), current_user.id) })

@main_bp.route('/set/list/name', methods=['PATCH'])
@login_required
def rename_list():
    """Renomme une liste
    """
    list_id = request.get_json()['idList']
    new_list_name = request.get_json()['newListName']

    return jsonify({'Status': ListController().rename(list_id, new_list_name)})

@main_bp.route('/set/favorites-order', methods=['PATCH'])
@login_required
def set_favorites_order():
    """
    Met à jour l'ordre des favoris
    ---
    tags:
      - favorites
    parameters:
      - $ref: '/static/swagger-components/components.yml#/components/parameters/idAnimeList'
    responses:
      200:
        $ref: '/static/swagger-components/components.yml#/
```

```
components/responses/GenericResponse'
    """
    return jsonify({ 'Status': AnimeController().reorder_favorites(current_user.id, request.json.get('ids')) })

@main_bp.route('/get/activities', methods=['GET'])
@login_required
def get_activities_for_logged_user():
    """
    Récupère tous les activitées de l'utilisateur connecté
    ---
    tags:
      - activities
    responses:
      200:
        $ref: '/static/swagger-components/components.yml#/components/responses/LoggedUserActivities'
    """
    return jsonify({'activities': ActivitiesController().get_last_24h(current_user.id)})

```

```python
from subprocess import Popen, PIPE

# Execute la commande : python3 -m md_toc github ../compiled/final.md
# Cette commande sert à générer une table des matières pour un document markdown.
p = Popen(['python3', '-m', 'md_toc', 'github', '../compiled/final.md'], stdin=PIPE, stdout=PIPE, stderr=PIPE)
# Récupère la sortie de la commande (la table des matière)
(output, err) = p.communicate()
toc = output.decode('utf-8') # Decode la sortie

# Ajoute le titre "Table des matières" juste avant la table des matières
toc = '## Table des matière\n\n' + toc + "\n<div style='page-break-after: always; break-after: page; text-align:right;'></div>"

updated_lines = []
# Ajoute la table des matière dans les lignes du fichier
with open('../compiled/final.md', 'r+') as f:
    lines = f.readlines()
    del lines[0]
    del lines[1]
    f.seek(0)
    lines.insert(0, toc)
    updated_lines = lines
    f.truncate(0) # Suppression de l'ancien contenu
    f.writelines(updated_lines) # Ajout du nouveau contenu
    f.close()
```

```bash
#!/bin/bash

# Fichier final
final=../compiled/final.md

# Écrase le fichier
echo "" > $final

# Parcour tout les fichiers md souhaité dans un certain ordre
for f in ../index.md \
         ../resume-enonce.md \
         ../methodologie.md \
         ../planning.md \
         ../implementation.md \
         ../librairies.md \
         ../fonctionnalites-majeurs.md \
         ../scenarios-tests.md \
       ../bibliographie.md \
       ../conclusion.md
do
    echo >> $final
    cat $f >> $final
done

# Éxecuter le script python pour générer la table des matières
exec python3 toc.py

```

```
#!/bin/sh

# Fusionne la page de garde avec le contenu du document final
pdftk ../compiled/cover.pdf \
         ../compiled/final.pdf \
         ../compiled/annexes-cover.pdf \
         ../compiled/resume-cover.pdf \
         ../compiled/resume-tpi.pdf \
         ../compiled/enonce-cover.pdf \
         ../compiled/"Enoncé TPI CAVAGNA Tanguy  I.DA-P4B v2.pdf" \
         ../compiled/journal-de-bord-cover.pdf \
         ../compiled/journal-de-bord.pdf \
         ../compiled/code-source-cover.pdf \
         ../compiled/code-source.pdf \
         cat output \
         ../compiled/rapport-tpi-et-documentation-technique.pdf
```

```javascript
/**
 * Script pour la prise en charge des raccourcis clavier
 *
 * @author: Tanguy Cavagna
 * @copyright: Copyright 2020, TPI
 * @version: 1.0.0
 * @date: 2020-05-28
 */

/**
 * Prise en charge des évènements keydown
 *
 * @param {event} e Évènement keyup
 */
function keydownHandler(e) {
    // Prise en charge du CTRL+S
    if (e.ctrlKey && e.keyCode === 83) {
        e.preventDefault(); // Annule l'event de sauvegarde natif

        const searchModalTrigger = document.getElementById('search-modal-trigger');

        if (searchModalTrigger !== undefined) {
            searchModalTrigger.click();
        }
    }
}

document.addEventListener('keydown', keydownHandler, false);

```

```javascript
/**
 * Script permettant le redimensionnement des image en fonction de la taille de l'écran
 *
 * @author: Tanguy Cavagna
 * @copyright: Copyright 2020, TPI
 * @version: 1.0.0
 * @date: 2020-05-28
 */

const resultsItemPictures = document.querySelectorAll('.search-results .result-item .left .picture');
const resultsItem = document.querySelectorAll('.search-results .result-item');

/**
 * Redéfini la taille des images pour garder un ratio correct
 */
function resizeThumbnails() {
    let newHeight = -1;
    // Re defini la hauteur de l'image en fonction de sa largeur
    resultsItemPictures.forEach((item) => {
        const currentItem = item;
        newHeight = parseInt(currentItem.getBoundingClientRect().width / 0.7333, 10);
        currentItem.style.height = `${newHeight}px`;
    });
    // Defini la hauteur max de la carte en foncion de la hauteur de l'image principale
    resultsItem.forEach((item) => {
        const currentItem = item;
        currentItem.style.maxHeight = `${currentItem.querySelector('.left .picture').getBoundingClientRect().height}px`;
    });
}

window.addEventListener('load', resizeThumbnails);
window.addEventListener('resize', resizeThumbnails);

```

```js
/* global fetchFavorites */

/**
 * Script pour la prise en charge du chargement des favoris, ajout de favoris,
 * ajout d'anime dans les listes
 *
 * @author: Tanguy Cavagna
 * @copyright: Copyright 2020, TPI
 * @version: 1.0.0
 * @date: 2020-05-29
 */

const favoriteTogglers = document.querySelectorAll('.favorite-toggler');
const statusCombo = document.querySelectorAll('.status_combo');
const customChecks = document.querySelectorAll('.custom-check');

/**
 * Supprime un anim des listes par défaut
 *
 * @param {int} idAnime Id de l'anime à supprimer des listes par défaut
 * @param {function} callback Function à éxecuter en tant que callback du fetch
 */
function deleteAnimeFromDefaultLists(idAnime, callback = () => {}) {
    return fetch('/delete/defaults', {
        method: 'DELETE',
        headers: {
            Accept: 'application/json',
            'Content-Type': 'application/json',
        },
        body: JSON.stringify({ idAnime }),
    }).then((response) => response.json()).then(() => {
        callback();
    });
}

/**
 * Ajout un anime dans une liste
 *
 * @param {int} idAnime Id de l'anime
 * @param {int} idList Id de la liste
```

```javascript
 * @param {function} callback Fonction à éxecuter en tant que callback du fetch
 */
function putAnimeInList(idAnime, idList, callback = () => {}) {
    fetch('/set/list', {
        method: 'PUT',
        headers: {
            Accept: 'application/json',
            'Content-Type': 'application/json',
        },
        body: JSON.stringify({ idAnime, idList }),
    }).then((response) => response.json()).then((json) => {
        callback(json);
    });
}

// Changement du status de favoris de l'anime
favoriteTogglers.forEach((favoriteToggler) => {
    favoriteToggler.addEventListener('click', (e) => {
        const toggler = e.target.closest('.favorite-toggler');

        fetch('/set/favorite', {
            method: 'PATCH',
            headers: {
                Accept: 'application/json',
                'Content-Type': 'application/json',
            },
            body: JSON.stringify({ idAnime: toggler.dataset.id }),
        }).then((response) => response.json()).then((json) => {
            // Check le toggle favoris selon l'état de l'anime
            if (json.Status === true) {
                toggler.classList.add('is-favorite');
            } else {
                toggler.classList.remove('is-favorite');
            }

            fetchFavorites();
        });
    });
});

// Changement du statut de visionnement de l'anime. (Complété, En cous, Abondoné, Planifié)
statusCombo.forEach((comboOption) => {
```

```javascription>
    comboOption.addEventListener('change', async () => {
        const idAnime = comboOption.value.split('-')[0];
        const idList = comboOption.value.split('-')[1];


        if (idList !== 'none') {
            await deleteAnimeFromDefaultLists(idAnime);
            putAnimeInList(idAnime, idList);
        } else {
            deleteAnimeFromDefaultLists(idAnime);
        }
    });
});

// Ajout d'un anime dans une liste
customChecks.forEach((customCheck) => {
    customCheck.addEventListener('click', (e) => {
        const check = e.target;
        const input = check.querySelector('input[type="checkbox"]');
        input.checked = !input.checked;

        const idAnime = input.value.split('-')[0];
        const idList = input.value.split('-')[1];

        putAnimeInList(idAnime, idList, (json) => {
            if (json.Status === true) {
                check.classList.add('checked');
            } else {
                check.classList.remove('checked');
            }
        });
    });
});

```

```javascript
/**
 * Script générant une table des matières
 *
 * @author: Tanguy Cavagna
 * @copyright: Copyright 2020, TPI
 * @version: 1.0.0
 * @date: 2020-06-08
 */
const titles = document.querySelectorAll('h1, h2, h3, h4, h5, h6');
const toc = document.querySelector('.toc');

for (let i = 0; i < titles.length; i += 1) {
    const title = titles[i];
    const tocTitle = document.createElement(title.tagName.toLowerCase());
    const tocLink = document.createElement('a');

    if (!title.hasAttribute('id')) {
        title.id = `h${i}`;
    }
    tocLink.href = `#${title.getAttribute('id')}`;
    tocLink.innerHTML = title.textContent;
    tocTitle.appendChild(tocLink);
    toc.appendChild(tocTitle);
}

```

```javascr
/**
 * Script pour la prise en charge de la recherche d'anime
 *
 * @author: Tanguy Cavagna
 * @copyright: Copyright 2020, TPI
 * @version: 1.0.0
 * @date: 2020-05-29
 */

const searchClear = document.querySelector('.search-clear');
const searchString = document.querySelector('.search-form .search-string');
const pickRandom = document.querySelector('#pick-random');

if (searchClear != null) {
    searchClear.addEventListener('click', () => {
        searchString.value = '';
    });
}

if (pickRandom != null) {
    pickRandom.addEventListener('click', () => {
        window.location.replace('/random');
    });
}

if (searchString != null) {
    // Redirige sur la page de recherche lors de l'appuie sur la touche entrée
    // dans le champs de recherche
    searchString.onkeydown = (e) => {
        if (e.keyCode === 13) { // Touche entrée
            window.location.replace(`/search/${searchString.value}`);
        }
    };
}

```

```javascript
/**
 * Récupère les animes favoris de l'utilisateur
 *
 * @param {boolean} col Est-ce l'affichage avec colonne de Bootstrap doit être utilisé
 * @param {string} searchedUser Pseudo de l'utilisateur cherché si sur la page d'un autre
 *                              utilisateur que celui connecté
 */
/* eslint-disable-next-line no-unused-vars */
function fetchFavorites(col = false, canDelete = false, searchedUser = null) {
    const favorites = document.querySelector('.favorites');

    if (favorites != null) {
        let url = '/get/favorites';

        // Ajout du slug `nickname` si l'on veut récupérer les favoris d'un
        // autre utilisateur que celui connecté
        if (searchedUser != null) {
            url += `/${searchedUser}`;
        }

        fetch(url, {
            method: 'GET',
            header: {
                Accept: 'application/json',
                'Content-Type': 'application/json',
            },
        }).then((response) => response.json()).then((json) => {
            favorites.innerHTML = '';

            // Affiche les favoris
            if (json.animes.length > 0) {
                json.animes.forEach((anime) => {
                    const favoriteItem = document.createElement('div');
                    const favoritePicture = document.createElement('img');
                    const favoriteRemover = document.createElement('img');

                    favoriteItem.classList.add('favorite-item');
                    favoriteItem.dataset.id = anime.id;
```

```

                    if (col === true) {
                        favoriteItem.classList.add('col-xl-4');
                    }

                    favoritePicture.src = anime.picture;
                    favoriteRemover.classList.add('favorite-remover');
                    if (canDelete === false) {
                        favoriteRemover.classList.add('d-none');
                    }
                    favoriteRemover.addEventListener('click', () => {
                        fetch('/set/favorite', {
                            method: 'PATCH',
                            headers: {
                                Accept: 'application/json',
                                'Content-Type': 'application/json',
                            },
                            body: JSON.stringify({ idAnime: anime.id }),
                        }).then((response) => response.json()).then(() => {
                            fetchFavorites(false, true);
                        });
                    });
                    favoriteRemover.src = '/static/img/remove.svg';
                    favoriteItem.appendChild(favoritePicture);
                    favoriteItem.appendChild(favoriteRemover);
                    favorites.appendChild(favoriteItem);
                });
            } else {
                const noFavorite = document.createElement('div');
                const h4 = document.createElement('h4');
                const span = document.createElement('span');

                noFavorite.classList.add('no-favorites');
                h4.innerHTML = 'Aucun favoris pour le moment';
                span.innerHTML = '(>人<;)';

                noFavorite.appendChild(h4);
                noFavorite.appendChild(span);
```

```
                    favorites.append(noFavorite);
                }
            });
        }
}

```

```javascript
/**
 * Script pour la prise en charge des activitée de l'utilisateur
 *
 * @author: Tanguy Cavagna
 * @copyright: Copyright 2020, TPI
 * @version: 1.0.0
 * @date: 2020-06-02
 */

/**
 * Retourne le nombre d'heure, minutes et secondes entre une date et aujourd'hui
 *
 * @param {timestamp} olderTime Timestamp antérieur à aujourd'hui
 */
/* eslint-disable-next-line no-unused-vars */
function timeEllapsedFromNow(olderTime) {
    let timeEllapsed = 0;

    // Récupération des différentes mesures de temps
    const delta = Math.abs((Date.now() - olderTime) / 1000);
    const seconds = Math.floor(delta % 60);
    const hours = Math.floor(delta / 3600) % 24;
    const minutes = Math.floor(delta / 60) % 60;

    // Assignation de la mesure de temps la plus appropriée
    if (hours > 0) {
        timeEllapsed = `${hours} heures`;
    } else if (minutes > 0) {
        timeEllapsed = `${minutes} minutes`;
    } else {
        timeEllapsed = `${seconds} secondes`;
    }

    return `Il y a ${timeEllapsed}`;
}

/**
 * Récupère les activité de l'utilisateur
 */
/* eslint-disable-next-line no-unused-vars */
function fetchActivities() {
    const activities = document.querySelector('.activities');

    if (activities != null) {
```

```javascript
        const url = '/get/activities';

        fetch(url, {
            method: 'GET',
            headers: {
                Accept: 'application/json',
                'Content-Type': 'application/json',
            },
        }).then((response) => response.json()).then((json) => {
            activities.innerHTML = '';

            // Affiche les activités
            if (json.activities.length > 0) {
                json.activities.forEach((activity) => {
                    const activityContainer = document.createElement('div');
                    const wrapper = document.createElement('div');
                    const activitySubject = document.createElement('div');
                    const subjectTitle = document.createElement('span');
                    const activityTimestamp = document.createElement('span');
                    const activityImg = document.createElement('img');

                    activityContainer.classList = 'activity';
                    subjectTitle.innerHTML = activity.title;
                    activitySubject.classList.add('activity__subject');
                    activitySubject.appendChild(subjectTitle);
                    activitySubject.innerHTML += `<br> À été ajouté à : ${activity.list}`;
                    activityTimestamp.classList.add('activity__timestamp');
                    activityTimestamp.innerHTML = timeEllapsedFromNow(new Date(activity.date));
                    activityImg.src = activity.picture;

                    wrapper.appendChild(activitySubject);
                    wrapper.appendChild(activityTimestamp);
                    wrapper.appendChild(activityImg);
                    activityContainer.appendChild(wrapper);
                    activities.appendChild(activityContainer);
                });
```

```
            } else {
                const noFavorites = document.createElement('div');
                const h4 = document.createElement('h4');
                const span = document.createElement('span');

                noFavorites.className = 'no-activities';
                h4.innerHTML = 'Aucune activité pour le moment';
                span.innerHTML = '(>人<;)';

                noFavorites.appendChild(h4);
                noFavorites.appendChild(span);

                activities.appendChild(noFavorites);
            }
        });
    }
}

```

```javascript
/**
 * Script pour la prise en charge du changement d'ordre des favoris
 *
 * @author: Tanguy Cavagna
 * @copyright: Copyright 2020, TPI
 * @version: 1.0.0
 * @date: 2020-06-02
 */

const reorder = document.querySelector('.reorder');
let removers = document.querySelectorAll('.favorite-remover');
let toggleReorder = false;

/**
 * Affiche ou cache les boutton de suppression des favoris
 */
function toggleHideFavoriteRemovers(hide = true) {
    removers.forEach((remover) => {
        if (hide === true) {
            remover.classList.add('d-none');
        } else {
            remover.classList.remove('d-none');
        }
    });
}

reorder.addEventListener('click', () => {
    $('.favorites').sortable({ disabled: toggleReorder });
    reorder.innerHTML = toggleReorder === false ? 'Sauvegarder' : 'Réordonner les favoris';

    // Si bouton cliqué pour sauvegarder
    if (toggleReorder === true) {
        const favorites = document.querySelectorAll('.favorites .favorite-item');
        toggleHideFavoriteRemovers(true);

        const ids = [];
        favorites.forEach((favorite) => {
            ids.push(favorite.dataset.id);
        });

        fetch('/set/favorites-order', {
            method: 'PATCH',
            headers: {
                Accept: 'application/json',
```

```
                'Content-Type': 'application/json',
            },
            body: JSON.stringify({ ids }),
        }).then((response) => response.json()).then(() => {
            // Ne rien faire
        }).catch(() => {
            // Ne rien faire
        });
    } else { // Clique sur `Réordonner les favoris` pour afficher les boutons de suppression
        removers = document.querySelectorAll('.favorite-remover');
        toggleHideFavoriteRemovers(false);
    }

    toggleReorder = !toggleReorder;
});

```

```javascript
/**
 * Script pour la prise en charge des listes de l'utilisateur
 *
 * @author: Tanguy Cavagna
 * @copyright: Copyright 2020, TPI
 * @version: 1.0.0
 * @date: 2020-06-02
 */

let lists = document.querySelectorAll('#lists .lists-container ul li');
let listNames = document.querySelectorAll('#lists .lists-container ul li .list__name');
let listRemovables = document.querySelectorAll('#lists .lists-container ul li .remove__list');
let listCancelRenames = document.querySelectorAll('#lists .lists-container ul li .cancel__rename');
let listRenameInput = document.querySelectorAll('#lists .lists-container ul li input');
const listSearchString = document.querySelector('#search-list .search-string');
const newListString = document.querySelector('#new-list__name');

/**
 * Récupère tout les animes d'une liste et les affiche
 *
 * @param {string} searchTerms Chaine de recherche
 * @param {int} idList Id de la liste à fetch
 */
function fetchListAnimes(searchTerms, idList) {
    fetch('/get/animes', {
        method: 'POST',
        headers: {
            Accept: 'application/json',
            'Content-Type': 'application/json',
        },
        body: JSON.stringify({
            nicknameUser: document.getElementById('searched_user').dataset.nickname,
            idList,
            'search-terms': searchTerms,
        }),
    }).then((response) => response.json()).then((json) => {
        const { animes } = json; // Récupère la valeur de la clef `animes` de l'object `json`

```

```javascript
        const animesContainer = document.querySelector('#animes-container');

        if (Object.keys(animes).length > 0) {
            const animeLists = Object.keys(animes);

            animesContainer.innerHTML = '';

            animeLists.forEach((list) => {
                const listSection = document.createElement('div');
                const listSectionTitle = document.createElement('h2');

                listSection.classList = 'mb-5 list-section';
                listSectionTitle.innerHTML = list;

                animes[list].forEach((anime) => {
                    const animeItem = document.createElement('div');
                    const animeItemTitle = document.createElement('h4');
                    const animeItemPicture = document.createElement('img');

                    animeItem.classList.add('anime-item');
                    animeItemTitle.innerHTML = anime.title;
                    animeItemPicture.src = anime.picture;

                    animeItem.appendChild(animeItemTitle);
                    animeItem.appendChild(animeItemPicture);

                    listSection.appendChild(animeItem);
                });

                animesContainer.appendChild(listSectionTitle);
                animesContainer.appendChild(listSection);
            });
        } else {
            animesContainer.innerHTML = '';

            const infos = document.createElement('div');
            infos.classList.add('lists-empty');

            const title = document.createElement('h4');
            const separator = document.createElement('span');
            const sub = document.createElement('span');
```

```
            const img = document.createElement('img');
            title.innerHTML = 'Aucun anime';
            separator.classList.add('separator');
            sub.innerHTML = 'Remplis tes listes pour la rendre heureuse !';
            img.src = '/static/img/whale.svg';

            infos.appendChild(title);
            infos.appendChild(separator);
            infos.appendChild(sub);
            infos.appendChild(img);

            animesContainer.appendChild(infos);
        }
    });
}

/**
 * Récupère les animes de la list cliquée
 */
function listNamesClickEventSetup() {
    listNames.forEach((listName) => {
        const currentListName = listName;

        // Event clique pour le nom des listes afin d'afficher les animes de cette dite liste
        currentListName.addEventListener('click', () => {
            const list = currentListName.parentElement;

            if (!list.classList.contains('new-list')) {
                // Parcour toutes les listes existantes dans la page et supprime la classe
                // `selected` pour l'ajouté sur la liste courante
                lists.forEach((l) => l.classList.remove('selected'));
                list.classList.add('selected');

                fetchListAnimes(listSearchString.value, list.dataset.list);
            }
        });

        if (
            ![
                'Complétés',
                'En cours',
```

```
                'Planifiés',
                'Abandonés',
            ].includes(currentListName.innerHTML)
        ) {
            // Event double clique pour renommer la liste seulement si elle ne fait
            // pas parti de celles par défaut
            currentListName.addEventListener('dblclick', () => {
                const newNameInput = currentListName.parentNode.querySelector('input');
                const cancelButton = currentListName.parentNode.querySelector('.cancel__rename');
                const removeButton = currentListName.parentNode.querySelector('.remove__list');

                newNameInput.classList.remove('d-none');
                newNameInput.value = currentListName.innerHTML;
                cancelButton.classList.remove('d-none');
                currentListName.classList.add('d-none');
                removeButton.classList.add('d-none');
            });
        }
    });
}

/**
 * Supprime une liste
 *
 * @param {int} idList Id de la liste à supprimer
 */
function deleteList(idList) {
    fetch('/delete/list', {
        method: 'DELETE',
        headers: {
            Accept: 'application/json',
            'Content-Type': 'application/json',
        },
        body: JSON.stringify({
            idList,
        }),
    }).then((response) => response.json()).then(() => {
        window.location.reload();
    });
}

```

```
/**
 * Supprime la liste cliqueé
 */
function removeListClickEventSetup() {
    // Event clique pour la suppression d'une liste
    listRemovables.forEach((removeTrigger) => {
        removeTrigger.addEventListener('click', () => {
            const list = removeTrigger.parentNode;
            deleteList(list.dataset.list);
        });
    });
}

/**
 * Annulle le renommage de la lsite
 */
function cancelListRenameClickEvent() {
    // Event clique pour l'annullation du renommage d'une liste
    listCancelRenames.forEach((cancelTrigger) => {
        cancelTrigger.addEventListener('click', () => {
            const list = cancelTrigger.parentNode;
            const newNameInput = list.querySelector('input');
            const cancelButton = list.querySelector('.cancel__rename');
            const removeButton = list.querySelector('.remove__list');
            const listName = list.querySelector('.list__name');

            newNameInput.classList.add('d-none');
            newNameInput.value = '';
            cancelButton.classList.add('d-none');
            listName.classList.remove('d-none');
            removeButton.classList.remove('d-none');
        });
    });
}

/**
 * Ajoute une nouvelle liste personnelle à l'utilisateur
 *
 * @param {string} newListName Nom de la nouvelle liste
 */
function addNewList(newListName) {
    fetch('/add/list', {
        method: 'PUT',
```

```
        headers: {
            Accept: 'application/json',
            'Content-Type': 'application/json',
        },
        body: JSON.stringify({
            newListName,
        }),
    }).then((response) => response.json()).then((json) => {
        newListString.value = '';
        const createdList = json.list;

        const li = document.createElement('li');
        const listName = document.createElement('span');
        const udpateNameInput = document.createElement('input');
        const removeButton = document.createElement('span');

        li.dataset.list = createdList.id;
        listName.classList.add('list__name');
        listName.innerHTML = createdList.name;
        udpateNameInput.type = 'text';
        udpateNameInput.name = 'new_list_name';
        udpateNameInput.style.width = '80%';
        udpateNameInput.classList.add('d-none');
        udpateNameInput.id = `${createdList.id}_new_list_name`;
        removeButton.classList.add('remove__list');
        removeButton.innerHTML = '?';

        li.appendChild(listName);
        li.appendChild(udpateNameInput);
        li.appendChild(removeButton);

        // Ajoute la nouvelle liste juste avant le champ text permettant d'ajouter
        // une nouvelle liste
        document.querySelector('#lists .lists-container ul').insertBefore(li, lists[lists.length - 1]);

        // Mise à jour des éléments DOM pour les listes
        lists = document.querySelectorAll('#lists .lists-container ul li');
        listNames = document.querySelectorAll('#lists .lists-container ul li .list__name');
        listRemovables = document.querySelectorAll('#lists .lists-container ul li .remove__list');
        listCancelRenames = document.querySelectorAll('#lists
```

```
 .lists-container ul li .cancel__rename');
        listRenameInput = document.querySelectorAll('#lists .lists-container ul li input');

        // Ajout des events pour la liste venant d'être créer
        listNamesClickEventSetup();
        removeListClickEventSetup();
        cancelListRenameClickEvent();
    });
}

/**
 * Renomme la liste
 *
 * @param {int} listId Id de la liste à modifier
 * @param {string} newListName Nouveau nom de la liste
 */
function renameList(idList, newListName) {
    fetch('/set/list/name', {
        method: 'PATCH',
        headers: {
            Accept: 'application/json',
            'Content-Type': 'application/json',
        },
        body: JSON.stringify({
            idList,
            newListName,
        }),
    }).then((response) => response.json()).then(() => {
        window.location.reload();
    });
}

// Affiche les animes des listes de l'utilisateur dont le titre concorde avec
// la chaine recherchée lorsque la touche entrés est pressée
listSearchString.onkeydown = (e) => {
    if (e.keyCode === 13) { // Touche entrée
        fetchListAnimes(listSearchString.value, null);
    }
};

// Créer une nouvelle liste pour l'utilisateur connecté lorsque la touche entrée et pressée
if (newListString !== undefined) {
    newListString.onkeydown = (e) => {
        if (e.keyCode === 13) {
```

```
            addNewList(newListString.value);
        }
    };
}

// Renomme une liste
if (listRenameInput !== undefined) {
    listRenameInput.forEach((renameInput) => {
        const currentRenameInput = renameInput;

        if (currentRenameInput.id !== 'new-list__name') {
            currentRenameInput.onkeydown = (e) => {
                if (e.keyCode === 13) {
                    renameList(
                        currentRenameInput.parentNode.dataset.list,
                        currentRenameInput.value,
                    );
                }
            };
        }
    });
}

window.addEventListener('load', () => {
    fetchListAnimes(null, null);
    listNamesClickEventSetup();
    removeListClickEventSetup();
    cancelListRenameClickEvent();
});

```

```javascript
/**
 * Script pour la prise e charge de la page d'accueil lorsque utilisateur connecté ou pas
 *
 * @author: Tanguy Cavagna
 * @copyright: Copyright 2020, TPI
 * @version: 1.0.0
 * @date: 2020-05-26
 */

const background = document.querySelector('.background');
const navbar = document.querySelector('.navbar');

if (background != null) {
    navbar.style.background = 'transparent';
}

```

```scss
/**
 * @filesource style.scss
 * @brief Fichier contenant tout le style de mon application
 * @author Tanguy Cavagna <tanguy.cvgn@eduge.ch>
 * @date 2020-04-18
 * @version 1.0.0
 */

@import 'typography';
@import 'colors';
@import 'sizes';

// Global
//=================
*{
    font-family: $f-regular;
    color: $c-white;
}
body {
    background: $c-background-medium-dark;
}
input {
    background: $c-background-darkest;
    color: $c-light-gray;
    border: none;
    padding: 10px;
    border-radius: $s-small-border-radius;
}

// Scrollbar
//=================
// Largeur
::-webkit-scrollbar {
    width: 10px;
}
// Fond
::-webkit-scrollbar-track {
    background: $c-background-darker;
}
// Barre
::-webkit-scrollbar-thumb {
    background: $c-background-medium-dark;
}
// Hover de la barre
::-webkit-scrollbar-thumb:hover {
    background: $c-background-darkest;
}
```

```scss

// Navbar
//=================
.navbar {
    background: $c-background-dark;
    height: $s-nav-height;
}
.nav-link {
    color: $c-white;
    font-family: $f-nav-link;
    letter-spacing: .15rem;
    font-size: .9rem;
    margin-right: 20px;

    &:hover {
        color: #c3c3c3;
    }
}
.dropdown-item span.danger {
    padding: .25rem .5rem;
    background-color: $c-danger-background;
    color: $c-danger-foreground;
    border-radius: 5px;
}
.navbar-toggler:not(:disabled) ~ .navbar-collapse.show,
.navbar-toggler:not(:disabled) ~ .navbar-collapse.collapsing {
    margin-top: 15px;
}
.navbar-toggler-icon {
    width: auto;
    height: auto;
    display: flex;
    justify-content: center;
    align-items: center;
}

// Login & Regsiter
//=================
.login-form {
    width: $s-login-form-width;
    height: $s-login-form-height;
    padding-bottom: 60px;
}
.signup-form {
    width: $s-register-form-width;
    height: $s-register-form-height;
    padding-bottom: 60px;
```

```scss
}
.login-form,
.signup-form {
    background: $c-background-darker;
    border-radius: $s-small-border-radius;
    display: flex;
    flex-direction: column;
    align-items: center;
}
.login-form h3,
.signup-form h3 {
    color: $c-light-gray;
    font-family: $f-section-title;
    padding-top: 60px;
    padding-bottom: 60px;
    text-align: center;
}
.login-form form,
.signup-form form {
    height: 100%;
    width: 80%;
    display: flex;
    flex-direction: column;

    fieldset {
        width: 100%;
    }
    ul {
        list-style-type: none;
        background-color: $c-danger-background;
        padding: 5px 0 5px 0;
        display: flex;
        flex-direction: column;
        align-items: center;
        border-radius: $s-small-border-radius;
    }
    ul li {
        color: $c-danger-foreground;
        font-family: $f-regular;
        font-weight: 600;
    }
    input {
        margin-bottom: 7%;
        width: 100%;
        font-family: $f-regular;
        font-weight: bold;
    }
```

```scss
    .submit {
        margin-top: 5%;
        display: flex;
        flex-direction: column;
        align-items: center;

        button {
            width: auto;
            font-size: 1.3rem;
            border: none;
            border-radius: $s-small-border-radius;
            background: $c-accent;
            color: $c-white;
            padding: 5px 20px 5px 20px;
            margin-bottom: 15px;
        }

        span {
            color: $c-white;
        }
    }
}

// Search modal
//=================
.search-modal {
    margin-top: 4%;

    .modal-dialog {
        max-width: none;
        width: 40%;
    }
}
.search-form {
    display: flex;
    flex-direction: row;
    background: $c-background-medium-dark;

    div:first-child {
        position: relative;
        margin-left: 10px;

        img {
            position: absolute;
            top: 50%;
            left: 50%;
            transform: translateY(-50%);
```

```scss
                width: 20px;
                opacity: .4;
            }
        }
        div:last-child {
            position: relative;
            margin-left: 10px;

            svg {
                position: absolute;
                top: 50%;
                left: 50%;
                transform: translateY(-50%);
                width: 20px;
                opacity: .4;
            }
        }
        input {
            width: 85%;
            margin-left: 30px;
            font-family: $f-regular;
            font-weight: bold;
        }
    }
    .search-string {
        background-color: $c-background-medium-dark;
    }
    .search-clear:hover {
        cursor: pointer;
    }

    // Home
    //=================
    #override-info {
        position: absolute;
        top: 90px;
        left: 50%;
        transform: translateX(-50%);
        color: $c-white;
        width: 40%;
        padding: 0;
        background: $c-background-darker;
        border-radius: $s-small-border-radius;
        text-align: center;
        z-index: 10;
    }
    #overrided {
```

```
    position: absolute;
    left: 50%;
    width: 40%;
    transform: translateX(-50%);
    top: 90px;

    button {
        margin: auto;
    }
}
.main-home {
    max-width: 100%;

    section {
        padding-left: 80px;

        h4 {
            font-family: $f-section-title;
            letter-spacing: .3rem;
            font-size: 1.7rem;
            margin: auto;
            text-align: center;
            margin-bottom: 30px;
        }
        .search-results {
            display: grid;
            grid-template-columns: repeat(auto-fill, 350px);
            column-gap: 50px;
            row-gap: 20px;

            .result-item {
                display: flex;
                position: relative;
                height: auto;

                .left, .right {
                    flex: 50%;
                    position: relative;
                }
                .left {
                    .picture {
                        width: 100%;
                        height: calc(100% * 0.73333);
                        border-radius: $s-small-border-radius 0 0 $s-small-border-radius;
                    }
                    .title {
```

```
                        position: absolute;
                        left: 0;
                        width: 100%;
                        bottom: 0;
                        text-align: center;
                        background: rgba($color: $c-background-dark, $alpha: .8);
                        border-radius: 0 0 0 $s-small-border-radius;
                        font-size: .9rem;
                        padding: .5rem;

                        &:hover {
                            cursor: pointer;
                        }
                    }
                }
                .right {
                    .status,
                    .type,
                    .relations,
                    .episodes {
                        width: 100%;
                        min-height: 30px;
                        text-align: center;
                        display: flex;
                        justify-content: center;
                        align-items: center;
                    }
                    .status {
                        border-radius: 0 $s-small-border-radius 0 0;
                        background: $c-background-darkest;
                        color: $c-primary-light;
                    }
                    .type {
                        background: $c-background-darkest--;
                        color: $c-light-gray;
                    }
                    .relations {
                        justify-content: space-between;
                        align-items: flex-start;
                        flex-wrap: wrap;
                        background: $c-background-darker;
                        max-height: calc(100% - (3 * 30px));
                        min-height: calc(100% - (3 * 30px));
                        overflow-x: hidden;
```

```
                        overflow-y: auto;
                        padding: 15px;
                        margin-left: 0;

                        img {
                            border-radius: $s-small-border-radius;
                            margin-bottom: 15px;
                            padding: 0;
                        }
                    }
                    .episodes {
                        border-radius: 0 0 $s-small-border-radius 0;
                        background: $c-background-darkest--;
                        color: $c-light-gray;
                    }
                }
                .modal .modal-dialog {
                    max-width: none;
                    width: 50%;

                    .modal-content {
                        background: $c-background-darkest--;

                        .header {
                            position: relative;
                            height: 170px;
                            width: 100%;

                            .header-background {
                                height: 100%;

                                img {
                                    top: 0;
                                    left: 0;
                                    width: 100%;
                                    height: 100%;
                                    object-fit: cover;
                                    border-radius: $s-small-border-radius $s-small-border-radius 0 0;
                                }
                                .dark-layer {
                                    top: 0;
                                    left: 0;
                                    height: 100%;
                                    width: 100%;
```

```
                                    background: rgba($color: #000000, $alpha: .5);
                                    border-radius: $s-small-border-radius $s-small-border-radius 0 0;
                                }
                            }
                            .thumbnail,
                            .dark-layer,
                            .title,
                            .favorite-toggler,
                            .save {
                                position: absolute;
                            }
                            .thumbnail {
                                left: 5%;
                                top: 20%;
                                width: 140px;
                                border-radius: $s-small-border-radius;
                            }
                            .title {
                                left: 25%;
                                bottom: 5%;
                                max-width: 400px;
                            }
                            .favorite-toggler {
                                bottom: 7%;
                                right: 2%;
                            }
                            .favorite-toggler:hover {
                                cursor: pointer;
                            }
                            .favorite-toggler:hover svg path,
                            .favorite-toggler.is-favorite svg path {
                                fill: $c-favorite;
                            }
                            .save {
                                right: 2%;
                                bottom: 2%;
                                background: $c-primary-light;
                                color: $c-white;
                            }
                        }
                        .content {
                            padding: 80px 5% 0 5%;
                            min-height: 200px;
```

```scss
                            height: auto;
                            background: $c-background-darkest--;
                            border-radius: 0 0 $s-small-border-radius $s-small-border-radius;

                            fieldset {
                                width: 100%;
                                display: flex;
                                flex-direction: column;
                                align-items: flex-start;
                                margin-bottom: 20px;

                                & > span:first-child {
                                    margin-bottom: 10px;
                                    font-family: $f-regular;
                                    font-weight: bold;
                                    letter-spacing: .2rem;
                                }
                            }
                            .status {
                                select {
                                    background: $c-background-medium-dark;
                                    color: $c-light-gray;
                                    padding: 5px 0 5px 15px;
                                    width: 30%;
                                    outline: none;
                                    border: none;
                                    border-radius: 5px;
                                    margin-left: 20px;

                                    option:hover {
                                        background: $c-background-darkest;
                                    }
                                }
                            }
                            .custom .lists {
                                width: 100%;
                                display: flex;
                                flex-direction: column;
                                flex-wrap: wrap;
                                max-height: 130px;
                                overflow-x: auto;
                                padding-left: 20px;

```

```
                                .list-item {
                                    display: flex;
                                    margin-bottom: 10px;
                                    min-width: 300px;

                                    .list-title {
                                        font-size: .9rem;
                                    }
                                }
                            }
                        }
                    }
                }
            }
        }
        .activities {
            margin-top: 5%;
            padding-top: 5%;
            height: auto;
            border-top: 3px solid $c-background-darkest;
            display: grid;
            grid-template-columns: repeat(auto-fill, minmax(300px, 1fr));
            gap: 30px;
            margin-bottom: 50px;

            .activity {
                width: 100%;
                height: 200px;
                position: relative;
                overflow: hidden;
                display: flex;
                justify-content: center;
                align-items: center;

                div {
                    width: 100%;
                    z-index: 0;

                    img {
                        position: absolute;
                        top: 0;
                        left: 50%;
                        transform: translateX(-50%);
                        width: 100%;
                        opacity: .3;
                        z-index: -1;
```

```
                    }
                    .activity__subject {
                        width: 100%;
                        text-align: center;
                        z-index: 1;
                        background-color: rgba($color: $c-background-darker, $alpha: .7);

                        span {
                            color: $c-primary-light;
                            font-weight: bold;
                            font-size: 1.8rem;
                        }
                    }
                    .activity__timestamp {
                        z-index: 1;
                        position: absolute;
                        top: 5%;
                        right: 5%;
                        font-family: $f-nav-link;
                        color: $c-light-gray;
                    }
                }
            }
            .no-activities {
                width: 100%;
                display: flex;
                justify-content: center;
                align-items: center;
                flex-direction: column;

                h4, span {
                    color: $c-light-gray;
                }
                span {
                    font-weight: bold;
                    font-size: 4rem;
                    opacity: .5;
                }
            }
        }
    }
    aside {
        padding: 0 80px;

        h4 {
            padding-left: 30px;
```

```
            color: $c-light-gray;
            font-family: $f-nav-link;
            font-size: .9rem;
            letter-spacing: .25rem;
            margin-bottom: 20px;
        }
    }
}

// About
//=================
.about-background {
    opacity: .2;
    position: fixed;
    z-index: -1;
    top: 50%;
    left: 50%;
    transform: translate(-50%, -50%);
}
.about-content {
    z-index: 1;
    text-align: justify;

    h1 {
        font-family: $f-nav-link;
        text-transform: uppercase;
    }
    h2 {
        font-weight: bold;
    }
    h2 + h3 {
        margin-top: 0;
    }
    h3 {
        margin-top: 15%;
    }
    img {
        max-width: 100%;
    }
}

// Profile
//=================
.profile-header {
    .header {
        position: relative;
        height: 40vh;
```

```scss

        .banner,
        .overlay {
            position: absolute;
            width: 100%;
            height: 100%;
            left: 0;
            top: 0;
            object-fit: cover;
        }
        .overlay {
            background: linear-gradient(180deg, rgba(196, 196, 196, 0) 0%, rgba(0, 0, 0, 0.55) 100%);;
        }
        .profile-picture {
            position: absolute;
            width: 10%;
            min-width: 125px;
            height: auto;
            left: 5%;
            bottom: 0;
        }
        .profile-nickname {
            position: absolute;
            bottom: 40px;
            font-family: $f-nav-link;
            font-size: 1.8rem;
        }
    }
    .links {
        height: 6vh;
        background: $c-background-dark;
        padding-left: 17%;
        display: flex;
        align-items: center;

        span {
            margin-right: 45px;
            font-family: $f-nav-link;
            color: $c-white;

            &:hover {
                cursor: pointer;
                color: $c-light-gray;
            }
            a {
                color: inherit;
```

```
                font-family: inherit;
                text-decoration: none;
            }
        }
    }
}
.profile {
    display: flex;
    flex-direction: column;
    align-items: flex-end;

    .stats {
        width: 95%;
        background: $c-background-dark;
        border-radius: 10px;
        display: flex;
        flex-wrap: wrap;
        justify-content: left;
        align-items: center;
        position: relative;
        padding: 30px;
        margin: auto;
        margin-top: 20px;
        margin-bottom: 20px;
        overflow-x: auto;
    }
    .favorites-container {
        width: 95%;
        height: auto;
        margin: auto;
        background: $c-background-dark;
        overflow-x: auto;
    }
    .stats {
        display: flex;
        align-items: center;
        justify-content: space-around;

        .watched {
            display: flex;
            flex-direction: column;

            span {
                text-align: center;
            }
            .count,
            .category {
```

```scss
                font-family: $f-nav-link;
            }
            .count {
                color: $c-accent;
                font-size: 2rem;
            }
        }
    }
}

// Lists
//=================
#lists {
    display: flex;
    justify-content: center;

    .lists-container {
        width: 80%;

        #search-list {
            display: flex;
            justify-content: left;
            align-items: center;
            background: $c-background-darkest;
            border-radius: $s-small-border-radius;

            img { margin-left: 5px; }
            .search-string {
                width: calc(100% - 25px);
                background: inherit;
            }
        }
        h4 {
            margin-top: 5%;
            font-size: 1.3rem;
            font-weight: bold;
        }
        ul {
            list-style-type: none;
            padding-left: 20px;

            li {
                padding: 5px;
                display: flex;
                justify-content: space-between;

                & .list__name:hover,
```

```scss
                & .remove__list:hover,
                & .cancel__rename:hover {
                    cursor: pointer;
                }
                &.selected {
                    background: $c-background-darker;
                    border-radius: $s-small-border-radius;
                }
            }
        }
    }
}
#animes-container {
    h2 {
        width: 100%;
    }
    .list-section {
        display: flex;
        justify-content: left;
        flex-wrap: wrap;

        .anime-item {
            position: relative;
            display: flex;
            justify-content: center;
            height: auto;
            width: 200px;
            margin: 25px;

            h4 {
                position: absolute;
                left: 0;
                width: 100%;
                bottom: 0;
                text-align: center;
                background: rgba($color: $c-background-dark, $alpha: .8);
                border-radius: 0 0 0 $s-small-border-radius;
                font-size: .9rem;
                padding: .5rem;
                margin: 0;
            }
            img {
                width: 100%;
                height: calc(200px / .73333);
                border-radius: $s-small-border-radius;
            }
```

```scss
        }
    }
}
.lists-empty img {
    margin-top: 30px;
}
.new-list #new-list__name {
    width: 100%;
}

// Favorites
//=================
.reorder {
    margin-top: 20px;
    margin-right: 2.5%;
    padding: 5px;
    border: none;
    width: 12%;
    background: $c-primary;
    border-radius: $s-small-border-radius;

    &:focus {
        outline: none;
    }
}
.favorites-container {
    background: $c-background-darker;
    border-radius: 10px;
    width: 100%;
    height: 750px;
}
.favorites {
    padding: 20px;
    position: relative;
    display: grid;
    grid-template-columns: repeat(auto-fill, 90px);
    grid-template-rows: repeat(auto-fill, 121px);
    gap: 20px;

    .no-favorites {
        display: flex;
        flex-direction: column;
        align-items: center;
        position: absolute;
        top: 50%;
        left: 50%;
        transform: translateX(-50%) translateY(-50%);
```

```scss
        width: 100%;

        h4 {
            font-family: $f-section-title;
            letter-spacing: .3rem;
            font-size: 1.7rem;
            text-align: center;
            padding: 0 1.2rem;
        }
        span {
            font-family: $f-nav-link;
            font-size: 2rem;
            margin-top: 5%;
            opacity: .5;
        }
    }
    .favorite-item {
        width: 100%;
        height: 100%;
        position: relative;

        img {
            width: 100%;
            height: 100%;
            border-radius: 3px;

            &.favorite-remover {
                position: absolute;
                right: -10px;
                top: -10px;
                width: 20px;
                height: 20px;

                &:hover {
                    cursor: pointer;
                }
            }
        }
    }
}

// miscellaneous
//=================
.background {
    position: fixed;
    left: 0;
    top: 0;
```

```scss
    z-index: -1;
}
.container-center {
    display: flex;
    flex-direction: column;
    justify-content: center;
    align-items: center;
    width: 100%;
    height: calc(100vh - #{$s-nav-height});
}
.alert button {
    width: 16px;
    height: 16px;
    display: flex;
    margin-top: 4px;
    margin-left: 10px;
}
.alert-success {
    color: $c-success-foreground;
    background: $c-success-background;
    font-weight: bold;
    border: none;
}
input:focus {
    outline: none;
}
.custom-check {
    width: 20px;
    height: 20px;
    background: $c-light-gray;
    border-radius: $s-small-border-radius;
    display: inline-block;
    margin-right: 10px;
    position: relative;

    input {
        display: none;
    }
}
.custom-check.checked::before {
    content: '';
    position: absolute;
    top: 50%;
    left: 50%;
    transform: translateX(-50%) translateY(-50%);
}
.custom-check.checked {
```

```
    background: $c-primary-light;
    position: relative;
}
.no-search,
.lists-empty {
    display: flex;
    flex-direction: column;
    align-items: center;

    span,
    h4 {
        color: $c-light-gray;
        margin: 0;
    }
    h4 {
        font-family: $f-section-title;
        letter-spacing: .3rem;
        font-size: 1.7rem;
    }
    .separator {
        width: 80px;
        height: 3px;
        background: $c-light-gray;
        margin: 1.5rem 0;
    }
    span:nth-child(3) {
        font-size: 1.2rem;
        text-align: center;
        font-family: $f-nav-link;
        letter-spacing: .2rem;
    }
    span:nth-child(4) {
        font-family: $f-nav-link;
        font-size: 4rem;
        opacity: .5;
        margin-top: 5%;
    }
}
.toc {
    width: 15%;
    height: auto;
    position: fixed;
    top: 100px;
    left: 40px;

    a {
        color: $c-light-gray;
```

```
    }
    h1:hover, h2:hover, h3:hover, h4:hover {
        cursor: pointer;
    }
    h1 {
        font-size: 1.6rem;
        font-weight: bold;
    }
    h2 {
        font-size: 1.4rem;
        margin-left: 10px;
    }
    h3 {
        font-size: 1.1rem;
        margin-left: 20px;
    }
    h4 {
        font-size: 1rem;
        margin-left: 30px;
    }
}

// Media queries
//=================
@media (min-width: 1450px) {
    .main-home aside .favorites {
        max-height: 750px;
        overflow-x: hidden;
        overflow-y: auto;
    }
}
@media (max-width: 1449px) {
    .main-home section {
        padding-right: 80px;
    }
}
@media (min-width: 992px) {
    .profile-header .profile-nickname {
        left: 17%;
    }
}
@media (max-width: 991px) {
    .profile-header .profile-nickname {
        right: 17%;
    }
}
```

```scss
/**
 * @filesource _typography.scss
 * @brief Fichier contenant toutes les taiiles utilisées dans mon application
 * @author Tanguy Cavagna <tanguy.cvgn@eduge.ch>
 * @date 2020-04-18
 * @version 1.0.0
 */


// Navbar
//=================
$s-nav-height: 60px;

// Login
//=================
$s-login-form-width: 450px;
$s-login-form-height: auto;

// Register
//=================
$s-register-form-width: 450px;
$s-register-form-height: auto;

// Miscellaneous
//=================
$s-small-border-radius: 5px;
```

```scss
/**
 * @filesource _typography.scss
 * @brief Fichier contenant toutes les couleurs utilisées dans mon application
 * @author Tanguy Cavagna <tanguy.cvgn@eduge.ch>
 * @date 2020-04-18
 * @version 1.0.0
 */

$c-white: #fff;
$c-light-gray: #C4C4C4;
$c-primary-light: #1CA5F2;
$c-primary: #006EDB;
$c-background-medium-dark: #272C38;
$c-background-dark: #1F2631;
$c-background-darker: #1F232D;
$c-background-darkest--: #191D26;
$c-background-darkest: #13171D;
$c-accent: #B368E6;
$c-favorite: #F34C56;
$c-danger-background: #e7c2c2;
$c-danger-foreground: #6f2323;
$c-success-background: #9ad6a8;
$c-success-foreground: #0d4419;
```

```scss
/**
 * @filesource _typography.scss
 * @brief Fichier contenant toutes les polices utilisées dans mon application
 * @author Tanguy Cavagna <tanguy.cvgn@eduge.ch>
 * @date 2020-04-18
 * @version 1.0.0
 */

@font-face {
    font-family: 'Regular';
    src: url('../fonts/Poppins-Regular.ttf');
}

@font-face {
    font-family: 'Nav link';
    src: url('../fonts/Poppins-Bold.ttf');
}

@font-face {
    font-family: 'Section title';
    src: url('../fonts/Poppins-ExtraBold.ttf');
}

$f-regular: 'Regular';
$f-nav-link: 'Nav link';
$f-section-title: 'Section title';
```

```python
"""Contient le model d'une liste

@author: Tanguy Cavagna
@copyright: Copyright 2020, TPI
@version: 1.0.0
@date: 2020-05-28
"""

class List:
    """Modèle d'une liste
    """

    def __init__(self, list_id: int, name: str):
        self.id = list_id
        self.name = name

    def serialize(self) -> dict:
        """Sérialise les données
        """
        return {
            'id': self.id,
            'name': self.name
        }
```

```python
"""Contient le model d'un type

@author: Tanguy Cavagna
@copyright: Copyright 2020, TPI
@version: 1.0.0
@date: 2020-05-26
"""

class Type:
    """Modèle d'un type
    """

    def __init__(self, type_id: int, name: str):
        self.id = type_id
        self.name = name

    def serialize(self) -> dict:
        """Sérialise les données
        """
        return {
            'id': self.id,
            'name': self.name
        }
```

```python
"""Contient le model d'un utilisateur

@author: Tanguy Cavagna
@copyright: Copyright 2020, TPI
@version: 1.0.0
@date: 2020-05-27
"""
from flask_login import UserMixin

class User(UserMixin):
    """Modèle d'un utilisateur. Classe héritée de UserMixin pour pouvoir être prise en compte comme classe de flask-loggin
    """

    def __init__(self, user_id: int, email: str, nickname: str):
        self.id = user_id
        self.email = email
        self.nickname = nickname

    def serialize(self) -> dict:
        """Sérialise les données
        """
        return {
            'id': self.id,
            'email': self.email,
            'nickname': self.nickname
        }
```

```python
"""Contient le model d'un anime

@author: Tanguy Cavagna
@copyright: Copyright 2020, TPI
@version: 1.0.0
@date: 2020-05-28
"""

class Anime:
    """Modèle d'un anime
    """

    def __init__(self, anime_id: int, title: str, anime_type: str, episodes: int, status: str, picture: str, thumbnail: str, synonyms: str):
        self.id = anime_id
        self.title = title
        self.type = anime_type
        self.episodes = episodes
        self.status = status
        self.picture = picture
        self.thumbnail = thumbnail
        self.synonims = synonyms
        self.relations = None
        self.is_favorite = None
        self.in_list = None

    def serialize(self) -> dict:
        """Sérialise les données
        """
        return {
            'id': self.id,
            'title': self.title,
            'type': self.type,
            'episodes': self.episodes,
            'status': self.status,
            'picture': self.picture,
            'thumbnail': self.thumbnail,
            'synonims': self.synonims,
            'relations': self.relations,
            'is_favorite': self.is_favorite
        }

```

```python
"""Contient le model d'un status

@author: Tanguy Cavagna
@copyright: Copyright 2020, TPI
@version: 1.0.0
@date: 2020-05-26
"""

class Status:
    """Modèle d'un statut
    """

    def __init__(self, status_id: int, name: str):
        self.id = status_id
        self.name = name

    def serialize(self) -> dict:
        """Sérialise les données
        """
        return {
            'id': self.id,
            'name': self.name
        }
```

```python
"""Contient tout les loggers

@author: Tanguy Cavagna
@copyright: Copyright 2020, TPI
@version: 1.0
@date: 2020-05-26
"""
import os

def log(e, in_terminal: bool = True, in_file: bool = True) -> None:
    """Log les erreurs

        Arguments:
            in_terminal {bool} -- Ajoute le log dans le temrinal
            in_file {bool} -- Aout le log dans un fichier
    """
    try:
        assert in_terminal or in_file, 'Au moins `in_terminal` ou `in_file` doit être à `True`'
        assert e is not None, 'L\'exception donnée n\'est pas valide'

        if in_terminal:
            print(str(e))

        if in_file:
            # Écrit dans un fichier à la racine du projet
            log_file_path = os.path.dirname(os.path.realpath(__file__)) + '/../../log.txt'

            append_write = 'a+' if os.path.exists(log_file_path) else 'w+'

            with open(log_file_path, append_write) as log_file:
                log_file.write(str(e) + '\n')
                log_file.close()
    except AssertionError as assert_error:
        log(assert_error)
        raise

```

```python
"""Contient le contrôleur d'authentification

@author: Tanguy Cavagna
@copyright: Copyright 2020, TPI
@version: 1.0.0
@date: 2020-05-27
"""
import re
from typing import Any

class BaseAuthenticationControlleur:
    """Contrôleur pour l'authentification
    """

    def __init__(self, form):
        """
            Arguments:
                form {dict} -- Dictionnaire provenant de `request.form`
        """
        self.form = form
        self._auth_errors = dict()

    def update_auth_errors_dict(self, field_name: str, error_condition: bool, message: str) -> bool:
        """Met à jour le dictionnaire d'erreurs si la condition d'erreur passe

            Arguments:
                field_name {str} -- Nom du champs pour lequel la condition s'applique
                error_condition {bool} -- Condition d'erreur. Si True, alors l'erreur doit être prise en compte.
                message {str} -- Message à afficher si l'erreur est levée

            Returns:
                bool -- Est-ce que l'erreur à été levée
        """
        if error_condition:
            if field_name not in self._auth_errors.keys():
                self._auth_errors.update({ field_name: [] })

            self._auth_errors.get(field_name).append(message)

        return self._auth_errors

```

```
    def reset_auth_errors(self):
        """Supprime toutes les anciennes erreurs
        """
        self._auth_errors = dict()

    def get_state(self) -> Any:
        """Est-ce que l'authentification est validé

            Returns:
                bool -- `True` si aucune erreurs levées
                dict -- Toutes les erreurs levées
        """
        return self._auth_errors if self._auth_errors else True

    def field_empty(self, field_name: str, message: str = None) -> bool:
        """Vérifie si le champ est vide

            Arguments:
                field_name {str} -- Nom du champ à vérifier
                message {str} -- Message à afficher en cas d'erreur
        """
        error_condition = bool(self.form.get(field_name) == '')

        if error_condition:
            if message is None:
                message = 'Ce champ est obligatoire'

            return self.update_auth_errors_dict(field_name, error_condition, message)

        return None

    # pylint: disable=redefined-builtin
    def length(self, field_name: str, min: int = -1, max: int = -1, message: str = None) -> bool:
        """Vérifie si le champ respecte les longueurs imposée

            Arguments:
                field_name {str} -- Nom du champ à vérifier
                min {int} -- Valeur minimum
                max {int} -- Valeur maximum
                message {str} -- Message à afficher en cas d'erreur
```

```
        """
        length = len(self.form.get(field_name))
        error_condition = bool(length < min or (max != -1 and length > max))

        if error_condition:
            if message is None:
                if max == -1: # Seulement une limite minimum est présent
                    message = f'Le champ doit au moins avoir {min} caractères.'
                elif min == -1: # Seulement une limite maximum est présent
                    message = f'Le champ ne peut pas excéder {max} caractères.'
                else:
                    message = f'Le champ doit être compris entre {min} et {max} caractères'

            return self.update_auth_errors_dict(field_name, error_condition, message)

        return None

    def regex(self, field_name: str, regex: str, message: str = None) -> bool:
        """Vérifie si le champ valide le regex

            Arguments:
                field_name {str} -- Nom du champ à vérifier
                regex {str} -- Expression régulière à vérifier
                message {str} -- Message à afficher en cas d'erreur
        """
        p = re.compile(regex)
        error_condition = bool(not p.match(self.form.get(field_name)))

        if error_condition:
            if message is None:
                message = 'Champ invalide'

            return self.update_auth_errors_dict(field_name, error_condition, message)

        return None
```

```python

    def email(self, field_name: str, message: str = None) -> bool:
        """Vérifie si le champ valide le regex

            Arguments:
                field_name {str} -- Nom du champ à vérifier
                message {str} -- Message à afficher en cas d'erreur
        """
        if message is None:
            message = 'Email invalide'
        return self.regex(field_name, r'^[a-zA-z0-9._-]+@[^\.]+\..+$', message)

    def equal_to(self, field_name: str, other_field_name: str, message: str = None) -> bool:
        """Vérifie si le champ valide le regex

            Arguments:
                field_name {str} -- Nom du champ à vérifier
                other_field_name {str} -- Nom du champ contre lequel tester l'égalité
                message {str} -- Message à afficher en cas d'erreur
        """
        try:
            field_to_equal = self.form.get(other_field_name)

            if field_to_equal is None:
                raise KeyError
        except KeyError:
            self.update_auth_errors_dict(other_field_name, True, f'Nom de champs à égalé incorrect `{other_field_name}`')

        error_condition = bool(self.form.get(field_name) != field_to_equal)

        if error_condition:
            if message is None:
                message = f'Le champ doit être égale à {other_field_name}'

            return self.update_auth_errors_dict(field_name, error_condition, message)

```

```
        return None

class SignupController(BaseAuthenticationControlleur):
    """Contrôleur de l'inscription
    """

    def validate(self):
        """Valide les champs du forumlaire
        """
        if not self.field_empty('email'):
            self.length('email', min=6, message='Choisissez un email plus long')
            self.email('email', 'Entrez un email valide')

        self.field_empty('nickname')

        if not self.field_empty('password'):
            self.length('password', min=6, message='Choisissez un mot de passe plus long')

        if not self.field_empty('confirm'):
            self.equal_to('confirm', 'password', 'Les mots de passes doivent correspondre')

        return not self._auth_errors

class LoginController(BaseAuthenticationControlleur):
    """Contrôleur de la connexion
    """

    def validate(self):
        """Valide les champs du forumlaire
        """
        if not self.field_empty('email'):
            self.email('email', 'Entrez un email valide')

        self.field_empty('password')

        return not self._auth_errors

```

```python
"""Contient la classe de contrôle des utilisateurs

@author: Tanguy Cavagna
@copyright: Copyright 2020, TPI
@version: 1.0.0
@date: 2020-05-28
"""
from datetime import datetime as dt

from .SqliteController import SqliteError, SqliteController
from .logger import log
from ..models.List import List

class ListController:
    """Controlleur d'une list
    """

    @classmethod
    def __encapsulate_lists(cls, raw_results: dict) -> [List]:
        """Encapsule les résultats d'une requete

            Arguments:
                raw_resutls {dict} -- Résultats provenant de `SqliteController.execute`

            Returns:
                [Anime] -- Liste de listes encapsuler
        """
        return [
            List(result['idList'],
                 result['nameList']
            ) for result in raw_results
        ]

    @classmethod
    def __serialize_encapsulated_lists(cls, encapsulated_lists: list) -> [dict]:
        """Sérialize toutes les listes contenu dans une liste de liste encapsulé

            Arguments:
                encapsulated_lists {list} -- Liste des animes encapsulé

            Returns:
                [dict] -- Liste de tout les animes sérialisé
        """
```

```python
        return [_list.serialize() for _list in encapsulated_lists]

    @classmethod
    def get_defaults_for_user(cls, user_id: int) -> [List]:
        """Retourne les listes par défaut d'un utilisateur

            Arguments:
                user_id {int} -- Id de l'utilisateur en question

            Returns:
                [List] -- Liste des listes par défaut de l'utilisateur
        """
        try:
            sql_user_list = """SELECT list.idList, nameList
                               FROM list
                               JOIN user_has_list ON list.idList = user_has_list.idList
                               WHERE user_has_list.idUser = ?
                               AND nameList IN ('Complétés', 'En cours', 'Abandonés', 'Planifiés')"""

            results = SqliteController().execute(sql_user_list, values=(user_id,), fetch_mode=SqliteController.FETCH_ALL)

            encapsulated = cls.__encapsulate_lists(results)

            return cls.__serialize_encapsulated_lists(encapsulated)
        except SqliteError as e:
            log(e)
            return []

    @classmethod
    def get_list_of_an_anime(cls, anime_id: int) -> [List]:
        """Récupère toutes les listes d'un anime

            Arguments:
                anime_id {int} -- Id de l'anime

            Returns:
                [List] -- Liste de toutes les listes de l'anime
        """
        try:
```

```
            sqli_lists = """SELECT list.idList, nameList
                            FROM list
                            JOIN list_has_anime ON list.idList = list_has_anime.idList
                            WHERE list_has_anime.idAnime = ?"""

            results = SqliteController().execute(sqli_lists, values=(anime_id,), fetch_mode=SqliteController.FETCH_ALL)

            encapsulated = cls.__encapsulate_lists(results)

            return cls.__serialize_encapsulated_lists(encapsulated)
        except SqliteError as e:
            log(e)
            return []

    @classmethod
    def get_user_lists(cls, user_id: int) -> [List]:
        """Récupère toutes les listes d'un utilisateur

            Arguments:
                user_id {int} -- Id de l'utilisateur

            Returns:
                [List] -- Toutes les listes de l'utilisateur
        """
        try:
            sql_user_lists = """SELECT list.idList, nameList
                                FROM list
                                JOIN user_has_list ON list.idList = user_has_list.idList
                                WHERE user_has_list.idUser = ?"""

            results = SqliteController().execute(sql_user_lists, values=(user_id,), fetch_mode=SqliteController.FETCH_ALL)

            encapsulated = cls.__encapsulate_lists(results)

            return cls.__serialize_encapsulated_lists(encapsulated)
        except SqliteError as e:
            log(e)
            return []
```

```python

    @classmethod
    def get_by_id(cls, list_id: int) -> List:
        """Récupère une liste via son id

            Arguments:
                list_id {int} -- Id de la liste

            Return:
                List -- Liste trouvée
        """
        try:
            sql_select = "SELECT idList, nameList FROM list WHERE idList = ?"

            row = SqliteController().execute(sql_select, values=(list_id,), fetch_mode=SqliteController.FETCH_ONE)

            return List(row['idList'], row['nameList'])
        except SqliteError as e:
            log(e)
            return None

    @classmethod
    def add_new_list(cls, new_name: str, user_id: int) -> List:
        """Ajoute une nouvelle liste pour l'utiliateur connecté

            Arguments:
                new_name {str} -- Nom de la nouvelle liste
                user_id {int} -- Id de l'utilisateur connecté

            Returns:
                List -- Nouvelle liste créée
        """
        try:
            sql_insert = "INSERT INTO list(nameList, modificationDate) VALUES(?, ?)"
            sql_link_to_user = "INSERT INTO user_has_list(idUser, idList, modificationDate) VALUES(?, ?, ?)"

            current_date = dt.now().strftime('%Y-%m-%d %H:%M:%S')

            list_id = SqliteController().execute(sql_insert, values=(new_name, current_date,), fetch_mode=SqliteController
```

```
.NO_FETCH)
            SqliteController().execute(sql_link_to_user, values=(user_id, list_id, current_date,), fetch_mode=SqliteController.NO_FETCH)

            return cls.get_by_id(list_id).serialize()
        except SqliteError as e:
            log(e)
            return None

    @classmethod
    def delete_by_id(cls, list_id: int, user_id: int) -> bool:
        """Supprime une liste via son id

            Arguments:
                list_id {int} -- Id de la list à supprimer
                user_id {int} -- Id de l'utilisateur connecté

            Returns:
                bool -- États de la requete
        """
        try:
            sql_unlink_anime = "DELETE FROM list_has_anime WHERE idList = ?"
            sql_unlink_to_user = "DELETE FROM user_has_list WHERE idUser = ? AND idList = ?"
            sql_delete = "DELETE FROM list WHERE idList = ?"

            SqliteController().execute(sql_unlink_anime, values=(list_id,), fetch_mode=SqliteController.NO_FETCH)
            SqliteController().execute(sql_unlink_to_user, values=(user_id, list_id,), fetch_mode=SqliteController.NO_FETCH)
            SqliteController().execute(sql_delete, values=(list_id,), fetch_mode=SqliteController.NO_FETCH)

            return True
        except SqliteError as e:
            print(str(e))
            raise e

    @classmethod
    def rename(cls, list_id: int, new_list_name: str) -> bool:
        """Renomme une liste

```

```
            Arguments:
                list_id {int} -- Id de la liste à renommé
                new_list_name {str} -- Nouveau nom de la liste

            Returns:
                bool -- État de la requête
        """
        try:
            sql_update = "UPDATE list SET nameList = ? WHERE idList = ?"

            SqliteController().execute(sql_update, values=(new_list_name, list_id,), fetch_mode=SqliteController.NO_FETCH)

            return True
        except SqliteError as e:
            log(e)
            return False

```

```python
"""Contient la classe de contrôle des types
Les types d'anime sont : Special, Movie, ONA, TV

@author: Tanguy Cavagna
@copyright: Copyright 2020, TPI
@version: 1.0.0
@date: 2020-05-26
"""
from sqlite3 import Error as SqliteError

from .SqliteController import SqliteController
from ..models.Type import Type
from .logger import log

class TypeController:
    """Controlleur des types
    """

    def __init__(self):
        pass

    @classmethod
    def get_all(cls) -> [Type]:
        """Récupère tout les types

            Returns:
                [Type] -- Listes contenant des types
        """
        try:
            rows = SqliteController().execute("SELECT `idType`, `nameType` FROM `type`", fetch_mode=SqliteController().FETCH_ALL)
            types = [Type(t['idType'], t['nameType']) for t in rows]

            return types
        except SqliteError as e:
            log(e)
            raise e

    @classmethod
    def get_all_as_dict(cls) -> dict:
        """Récupère tout les types sous forme de dictionnaire

            Returns:
                dict -- Dictionnaire contenant les types avec le format "name": id
```

```
        """
        types = {}
        for t in cls.get_all():
            types.update({t.name: t.id})

        return types
```

```python
"""Contient la classe de contrôle des utilisateurs

@author: Tanguy Cavagna
@copyright: Copyright 2020, TPI
@version: 1.0.0
@date: 2020-05-27
"""
from datetime import datetime as dt

from sqlite3 import Error as SqliteError
from .SqliteController import SqliteController
from .TypeController import TypeController
from ..models.User import User
from .logger import log

class UserController:
    """Contrôleur d'un utilisateur
    """

    @classmethod
    def exists(cls, email: str) -> bool:
        """Test is l'email existe déjà en base

            Arguments:
                email {str} -- Email à tester

            Returns:
                bool -- Est-ce que l'email existe
        """
        try:
            sql_exists = "SELECT COUNT(*) AS `Count` FROM user WHERE emailUser = ?"

            count = SqliteController().execute(sql_exists, values=(email,), fetch_mode=SqliteController.FETCH_ONE)['Count']

            return bool(count > 0)
        except SqliteError as e:
            log(e)
            return True

    @classmethod
    def check_password(cls, email: str, hashed_password: str) -> bool:
        """Test la validité du mot de passe

```

```
            Arguments:
                email {str} -- Email de l'utiliateur à tester
                hashed_password {str} -- Mot de passe à tester

            Returns:
                bool -- Est-ce que le mot de passe est valide
        """
        try:
            sql_check = "SELECT COUNT(*) AS `Count` FROM user WHERE emailUser = ? AND password = ?"

            count = SqliteController().execute(sql_check, values=(email, hashed_password,), fetch_mode=SqliteController.FETCH_ONE)['Count']

            return bool(count == 1)
        except SqliteError as e:
            log(e)
            return False

    @classmethod
    def insert(cls, email: str, nickname: str, password: str) -> User:
        """Insère un nouvel utilisateur en base

            Arguments:
                email {str} -- Email
                nickname {str} -- Pseudo
                passowrd {str} -- Mot de passe

            Returns:
                User -- Nouvel utilisateur créé
        """
        try:
            sql_insert = "INSERT INTO user(emailUser, password, nicknameUser, modificationDate) VALUES(?, ?, ?, ?)"

            values_user = ((
                email,
                password,
                nickname,
                dt.now().strftime('%Y-%m-%d %H:%M:%S')
            ))

            last_row_id = SqliteController().execute(sql_insert, values=values_user, fetch_mode=SqliteController.NO_FETCH)
```

```python

            return User(last_row_id, email, nickname)
        except SqliteError as e:
            log(e)
            return None

    @classmethod
    def get_by_id(cls, user_id: int) -> User:
        """Récupère l'utilisateur via son id

            Arguments:
                user_id {int} -- Id de l'utilisateur à récupèrer

            Returns:
                User -- Utilisateur trouvé
        """
        try:
            sql_select = "SELECT * FROM user WHERE idUSer = ?"

            user_dict = SqliteController().execute(sql_select, values=(user_id,), fetch_mode=SqliteController.FETCH_ONE)

            if user_dict is not None:
                return User(user_dict['idUser'], user_dict['emailUser'], user_dict['nicknameUser'])

            return None
        except SqliteError as e:
            log(e)
            return None

    @classmethod
    def get_by_email(cls, email: str) -> User:
        """Récupère un utilisateur via son email

            Argument:
                email {str} -- Email de l'utilisateur

            Returns:
                User -- Utilisateur trouvé
        """
        try:
            sql_select = "SELECT * FROM user WHERE emailUser = ?"

```

```python
            user_dict = SqliteController().execute(sql_select, values=(email,), fetch_mode=SqliteController.FETCH_ONE)

            if user_dict is not None:
                return User(user_dict['idUser'], user_dict['emailUser'], user_dict['nicknameUser'])

            return None
        except SqliteError as e:
            log(e)
            return None

    @classmethod
    def get_by_nickname(cls, nickname: str) -> User:
        """Récupère un utilisateur via son pseudo

            Arguments:
                nickname {str} -- Pseudo de l'utilisateur cherché

            Returns:
                User -- Utilisateur trouvé
        """
        try:
            sql_select = "SELECT * FROM user WHERE nicknameUser = ?"

            user_dict = SqliteController().execute(sql_select, values=(nickname,), fetch_mode=SqliteController.FETCH_ONE)

            if user_dict is not None:
                return User(user_dict['idUser'], user_dict['emailUser'], user_dict['nicknameUser'])

            return None
        except SqliteError as e:
            log(e)
            return None

    @classmethod
    def setup_default_lists(cls, user_id: int) -> bool:
        """Créer les listes par défaut si inexistantes

            Arguments:
                user_id {int} -- Id de l'utilisateur

            Returns:
```

```
                bool -- `True` si tout c'est bien passé
        """
        try:
            sql_default_list = "INSERT INTO list(nameList, modificationDate) VALUES(?, ?)"
            sql_link_list_user = "INSERT INTO user_has_list(idUser, idList, modificationDate) VALUES(?, ?, ?)"

            values_list = [
                'Complétés',
                'En cours',
                'Abandonés',
                'Planifiés'
            ]

            current_time = dt.now().strftime('%Y-%m-%d %H:%M:%S')
            for list_name in values_list:
                list_id = SqliteController().execute(sql_default_list, values=(list_name,current_time,), fetch_mode=SqliteController.NO_FETCH)
                SqliteController().execute(sql_link_list_user, values=(user_id,list_id,current_time,), fetch_mode=SqliteController.NO_FETCH)

            return True
        except SqliteError as e:
            log(e)
            return False

    @classmethod
    def get_stats_by_id(cls, user_id: int) -> []:
        """Récupère les statistiques pour un utilisateur

            Arguments:
                user_id {int} -- Id de l'utilisateur

            Returns:
                [] -- Liste des statistiques
        """
        try:
            sql_stats = """SELECT COUNT(*) AS `Count`
                           FROM anime
                           WHERE idAnime IN (
                               SELECT list_has_anime.idAnime
                               FROM user_has_list
                               JOIN list_has_anime ON
```

```
user_has_list.idList = list_has_anime.idList
                                WHERE user_has_list.idUser = ?
                                AND user_has_list.idList = (
                                    SELECT list.idList
                                    FROM list
                                    JOIN user_has_list ON list.idList = user_has_list.idList
                                    WHERE user_has_list.idUser = ?
                                    AND list.nameList LIKE "Complétés"
                                )
                            )
                            AND type = (
                                SELECT idType
                                FROM type
                                WHERE nameType LIKE ?
                            )"""

            types = TypeController().get_all()

            stats = {}
            for t in types:
                stats[t.name] = SqliteController().execute(sql_stats, values=(user_id,user_id,t.name,), fetch_mode=SqliteController.FETCH_ONE)['Count']

            return stats
        except SqliteError as e:
            log(e)
            return []
```

```python
"""Contient la classe de contrôle des animes

@author: Tanguy Cavagna
@copyright: Copyright 2020, TPI
@version: 1.0
@date: 2020-05-26
"""
import json
import pathlib
#import sys
#import os
from sqlite3 import Error as SqliteError
from datetime import datetime as dt
from random import randint

from .SqliteController import SqliteController
from .TypeController import TypeController
from .StatusController import StatusController
from .ListController import ListController
from ..models.Anime import Anime
from .logger import log

class AnimeController:
    """Controlleur d'un anime
    """

    #pylint: disable=too-many-return-statements
    @classmethod
    def override_local_with_json(cls, path: str) -> None:
        """Écrase toutes les données en base

            Arguments:
                path {string} -- Chemin du fichier json
        """
        with open(pathlib.Path(__file__).parent / path, 'r', encoding="utf8") as f:
            animes = json.load(f)

        try:
            # Mise en place des tables si elles ne sont pas présente et truncate tout ce qui est en lien direct avec la table 'anime'
            if not SqliteController().setup_type_table(): return False
            if not SqliteController().setup_status_table(): return False
            if not SqliteController().setup_anime_table():
```

```
42 return False
43             if not SqliteController().setup_url_table(): return False
44             if not SqliteController().setup_user_table(): return False
45             if not SqliteController().setup_list_table(): return False
46             if not SqliteController().setup_user_has_list_table(): return False
47             if not SqliteController().setup_list_has_anime_table(): return False
48             if not SqliteController().setup_user_has_favorite_table(): return False
49             if not SqliteController().truncate_anime_related(): return False
50
51             # Requêtes pour les insertions de nouvelles données
52             sql_query_type = "INSERT INTO type(nameType, modificationDate) VALUES(?, ?)"
53             sql_query_status = "INSERT INTO status(nameStatus, modificationDate) VALUES(?, ?)"
54             sql_query_anime = "INSERT INTO anime(idAnime, title, type, episodes, status, picture, thumbnail, synonyms, modificationDate) VALUES(?, ?, ?, ?, ?, ?, ?, ?, ?)"
55             sql_query_anime_ft = "INSERT INTO anime_ft(idAnime, title) VALUES(?, ?)"
56             sql_query_url = "INSERT INTO url(urlName, idAnime, isRelation, modificationDate) VALUES(?, ?, ?, ?)"
57
58             # Tableaux ayant pour but de stocké les futures valeurs à mettre dans les table pour faire un insert_many
59             values_type = []
60             values_status = []
61             values_anime = []
62             values_anime_ft = []
63             values_source = []
64             values_relation = []
65
66             current_time = dt.now().strftime('%Y-%m-%d %H:%M:%S')
67
68             yield "Récupération des types et statuts"
69
70             # Récupération des statuts et des types
71             for anime in animes['data']:
72                 # Suppression de tout les OVA car souvent
```

```
ayant un contenu innaproprié pour un TPI
                if anime['type'] == 'OVA':
                    continue

                # Ajout distinct des types
                if not any(anime['type'] in i for i in values_type):
                    values_type.append((
                        anime['type'],
                        current_time
                    ))

                # Ajout distinct des statuts
                if not any(anime['status'] in i for i in values_status):
                    values_status.append((
                        anime['status'],
                        current_time
                    ))

            yield "Insertion des types et status"
            SqliteController().execute_many(sql_query_type, values_type)
            SqliteController().execute_many(sql_query_status, values_status)

            yield "Récupération des animes et urls"
            types = TypeController().get_all_as_dict()
            status = StatusController().get_all_as_dict()

            # Reparcourir le fichier json pour extraire cette fois si les animes et leurs relations
            idAnime = 1
            for anime in animes['data']:
                # Suppression de tout les OVA car leur contenu est souvent innaproprié pour un TPI
                if anime['type'] == 'OVA':
                    continue

                values_anime.append((
                    idAnime,
                    anime['title'],
                    types[anime['type']],
                    anime['episodes'],
                    status[anime['status']],
                    anime['picture'],
                    anime['thumbnail'],
```

```python
                    ','.join(anime['synonyms']) if len(anime['synonyms']) > 0 else None,
                    current_time
                ))

                values_anime_ft.append((
                    idAnime,
                    anime['title']
                ))

                for source in anime['sources']:
                    values_source.append((
                        source,
                        idAnime,
                        False,
                        current_time
                    ))

                for relation in anime['relations']:
                    values_relation.append((
                        relation,
                        idAnime,
                        True,
                        current_time
                    ))

                idAnime += 1

            yield "Insertion des animes et urls"
            SqliteController().execute_many(sql_query_anime, values_anime)
            SqliteController().execute_many(sql_query_anime_ft, values_anime_ft)
            SqliteController().execute_many(sql_query_url, values_source)
            SqliteController().execute_many(sql_query_url, values_relation)

            yield "redirect"
            return True
        except SqliteError as e:
            log(e)
            return False

    @classmethod
    def __encapsulate_animes(cls, raw_results: dict) -> [Anime]:
```

```
154         """Encapsule les résultats d0une requete
155
156             Arguments:
157                 raw_resutls {dict} -- Résultats provenant de
    `SqliteController.execute`
158
159             Returns:
160                 [Anime] -- Liste d'anime encapsuler
161         """
162         return [
163             Anime(result['idAnime'],
164                   result['title'],
165                   result['nameType'],
166                   result['episodes'],
167                   result['nameStatus'],
168                   result['picture'],
169                   result['thumbnail'],
170                   result['synonyms']
171             ) for result in raw_results
172         ]
173
174     @classmethod
175     def __serialize_encapsulated_animes(cls,
    encapsulated_animes: list) -> [dict]:
176         """Sérialize tout les animes contenu dans une liste d
    'anime encapsulé
177
178             Arguments:
179                 encapsulated_animes {list} -- Liste des
    animes encapsulé
180
181             Returns:
182                 [dict] -- Liste de tout les animes sérialisé
183         """
184         return [anime.serialize() for anime in
    encapsulated_animes]
185
186     @classmethod
187     def __get_count(cls) -> int:
188         """Retourne le nombre d'anime présent en base
189
190             Returns:
191                 int -- Nombre d'anime en base
192         """
193         try:
194             sql_count = """SELECT COUNT(*) AS `Count` FROM
    anime"""
```

```python

            count = SqliteController().execute(sql_count, fetch_mode=SqliteController.FETCH_ONE)['Count']

            return count
        except SqliteError as e:
            log(e)
            return None

    @classmethod
    def get_from_search_string(cls, search_string: str) -> [Anime]:
        """Retourne les animes concordant avec la chaine de recherche

            Arguments:
                search_string {str} -- Chaine de recherche

            Returns:
                [Anime] -- Liste des animes concordant
        """
        try:
            # Le join sur la table virtuelle est obligatoire pour réalisé des recherches fulltext
            sql_search = """SELECT anime.idAnime, anime.title, nameType, episodes, nameStatus, picture, thumbnail, synonyms
                                FROM anime
                                JOIN anime_ft ON anime.idAnime = anime_ft.idAnime
                                JOIN status ON anime.status = status.idStatus
                                JOIN type ON anime.type = type.idType
                                WHERE anime_ft.title MATCH ?
                                LIMIT 9"""

            results = SqliteController().execute(sql_search, values=(search_string,), fetch_mode=SqliteController.FETCH_ALL)

            encapsulated = cls.__encapsulate_animes(results)

            return cls.__serialize_encapsulated_animes(encapsulated)
        except SqliteError as e:
            log(e)
```

```
            return []

    @classmethod
    def get_relations_by_anime_id(cls, anime_id: int) -> [Anime]:
        """Retourne toutes les relations d'un anime

            Arguments:
                anime_id {int} -- Id de l'anime du quel nous voulons les relations

            Returns:
                [Anime] -- Liste de toutes les relations
        """
        try:
            # Cette requete est relativement complexe. Pour avoir toutes les relations d'un anime, voila comment elle procède :
            #   Récupérer toutes les url liée à un anime, récupérer tout les ids des animes ayant des urls référencé comme relation à l'anime principal
            #   et ne pas reprendre une seconde  fois l'anime principale. Ne reste plus qu'à récupérer les informations de ces animes
            sql_relations = """SELECT anime.idAnime, anime.title, nameType, episodes, nameStatus, picture, thumbnail, synonyms
                                FROM anime
                                JOIN status ON anime.status = status.idStatus
                                JOIN type ON anime.type = type.idType
                                WHERE idAnime IN (
                                    SELECT idAnime
                                    FROM url
                                    WHERE urlName IN (
                                        SELECT urlName
                                        FROM url
                                        WHERE idAnime = ?
                                    )
                                    AND isRelation = 1
                                    AND idAnime != ?
                                )"""

            results = SqliteController().execute(sql_relations, values=(anime_id, anime_id,), fetch_mode=SqliteController.FETCH_ALL)
```

```python

            encapsulated = cls.__encapsulate_animes(results)

            return cls.__serialize_encapsulated_animes(encapsulated)
        except SqliteError as e:
            log(e)
            return []

    @classmethod
    def get_random_anime(cls) -> Anime:
        """Récupère un anime aléatoire

            Returns:
                Anime -- Anime tiré aléatoirement
        """
        try:
            sql_select = """SELECT anime.idAnime, anime.title, nameType, episodes, nameStatus, picture, thumbnail, synonyms
                            FROM anime
                            JOIN status ON anime.status = status.idStatus
                            JOIN type ON anime.type = type.idType
                            WHERE idAnime = ?"""

            random_anime_id = randint(1, cls.__get_count())

            results = SqliteController().execute(sql_select, values=(random_anime_id,), fetch_mode=SqliteController.FETCH_ONE)

            encapsulated = cls.__encapsulate_animes([results])

            return cls.__serialize_encapsulated_animes(encapsulated)[0]
        except SqliteError as e:
            log(e)
            return None

    @classmethod
    def is_anime_in_user_favorite(cls, user_id: int, anime_id: int) -> bool:
        """Retourne si l'anime est dans la liste des favoris de l'utilisateur
```

```python

            Arguments:
                user_id {int} -- Id de l'utilisateur en question
                anime_id {int} -- Id de l'anime sur lequel tester si il est favoris

            Returns:
                bool -- Est-ce que l'anime est dans les favoris de l'utilisateur
        """
        try:
            sql_is_favorite = """SELECT COUNT(*) AS `Count`
                                 FROM user_has_favorite
                                 WHERE idUser = ?
                                 AND idAnime = ?"""

            results = SqliteController().execute(sql_is_favorite, values=(user_id,anime_id,), fetch_mode=SqliteController.FETCH_ONE)['Count']

            return bool(results == 1)
        except SqliteError as e:
            log(e)
            return False

    @classmethod
    def is_anime_in_list(cls, anime_id: int, list_id: int) -> bool:
        """Est-ce que l'anime est dans la liste

            Arguments:
                anime_id {int} -- Id de l'anime
                list_id {int} -- Id de la liste

            Returns:
                bool -- Est-ce que l'anime est dans la liste
        """
        try:
            sql_is_in_list = """SELECT COUNT(*) AS `Count`
                                FROM list_has_anime
                                WHERE idAnime = ?
                                AND idList = ?"""

            results = SqliteController().execute(sql_is_in_list, values=(anime_id,list_id,), fetch_mode=SqliteController.FETCH_ONE)['Count']
```

```python

            return bool(results == 1)
        except SqliteError as e:
            log(e)
            return False

    @classmethod
    def set_anime_in_user_favorite(cls, user_id: int, anime_id: int) -> bool:
        """Met à jour l'état de favoris d'un anime pour un utiliateur

            Arguments:
                user_isd {int} -- Id de l'utilisateur
                anime_id {int} -- Id de l'anime

            Returns:
                bool -- État de favoris de l'anime
        """
        try:
            sql_delete_favorite = """DELETE FROM user_has_favorite
                                     WHERE idAnime = ?
                                     AND idUser = ?"""
            sql_get_last_order_id = """SELECT orderId
                                       FROM user_has_favorite
                                       WHERE idUser = ?
                                       ORDER BY orderId DESC
                                       LIMIT 1"""
            sql_add_favorite = "INSERT INTO user_has_favorite (idAnime, idUser, orderId, modificationDate) VALUES (?, ?, ?, ?)"

            if cls.is_anime_in_user_favorite(user_id, anime_id):
                SqliteController().execute(sql_delete_favorite, values=(anime_id,user_id,), fetch_mode=SqliteController.NO_FETCH)

                # Réattribué les orderId

                is_favorite = False
            else:
                last_order_id = SqliteController().execute(sql_get_last_order_id, values=(user_id,), fetch_mode=SqliteController.FETCH_ONE)
                last_order_id = int(last_order_id['orderId'
```

```
374 ]) + 1 if last_order_id is not None else 1
375                 SqliteController().execute(sql_add_favorite,
    values=(anime_id,user_id,last_order_id,dt.now().strftime('%Y
    -%m-%d %H:%M:%S'),), fetch_mode=SqliteController.NO_FETCH)
376
377                 is_favorite = True
378
379             return is_favorite
380         except SqliteError as e:
381             log(e)
382             return False
383
384     @classmethod
385     def get_favorite_by_user_id(cls, user_id: int) -> [Anime
    ]:
386         """Récupère tout les favoris d'un utilisateur
387
388             Arguments:
389                 user_id {int} -- Id de l'utilisateur
390
391             Returns:
392                 [Anime] -- Liste de tout les favoris
393         """
394         try:
395             sql_favorites = """SELECT anime.idAnime, anime.
    title, nameType, episodes, nameStatus, picture, thumbnail,
    synonyms
396                                FROM anime
397                                JOIN status ON anime.status =
    status.idStatus
398                                JOIN type ON anime.type = type
    .idType
399                                JOIN user_has_favorite ON
    anime.idAnime = user_has_favorite.idAnime
400                                WHERE user_has_favorite.idUser
     = ?
401                                ORDER BY orderId ASC"""
402
403             results = SqliteController().execute(
    sql_favorites, values=(user_id,), fetch_mode=SqliteController
    .FETCH_ALL)
404
405             encapsulated = cls.__encapsulate_animes(results)
406
407             return cls.__serialize_encapsulated_animes(
    encapsulated)
408         except SqliteError as e:
```

```
            log(e)
            return []

    @classmethod
    def set_anime_in_list(cls, anime_id: int, list_id: int) -> bool:
        """Met à jour l'appartenance d'un anime à une liste

            Arguments:
                anime_id {int} -- Id de l'anime
                list_id {int} -- Id de la liste
        """
        try:
            sql_delete_from_list = """DELETE FROM list_has_anime
                                      WHERE idAnime = ?
                                      AND idList = ?"""
            sql_add_to_list = """INSERT INTO list_has_anime(idAnime, idList, modificationDate)
                                 VALUES(?, ?, ?)"""

            if cls.is_anime_in_list(anime_id, list_id):
                SqliteController().execute(sql_delete_from_list, values=(anime_id,list_id,), fetch_mode=SqliteController.NO_FETCH)
                is_in_list = False
            else:
                SqliteController().execute(sql_add_to_list, values=(anime_id,list_id,dt.now().strftime('%Y-%m-%d %H:%M:%S'),), fetch_mode=SqliteController.NO_FETCH)
                is_in_list = True

            return is_in_list
        except SqliteError as e:
            log(e)
            return False

    @classmethod
    def delete_anime_status(cls, user_id: int, anime_id: int) -> bool:
        """Supprime le stats actuel d'un anime pour un utilisateur

            Arguments:
                user_id {int} -- Id de l'utilisateur
                anime_id {int} -- Id de l'anime

```

```
            Returns:
                bool -- État de la requete
        """
        try:
            sql_delete = """DELETE FROM list_has_anime
                            WHERE idList IN (?, ?, ?, ?)
                            AND idAnime = ?"""

            default_lists = ListController().get_defaults_for_user(user_id)
            in_clause = [l['id'] for l in default_lists] # Créer un tableau regroupant les ids des listes par défaut
            values = [] + in_clause # Fusionne les deux listes
            values.append(anime_id)

            SqliteController().execute(sql_delete, values=tuple(values), fetch_mode=SqliteController.NO_FETCH)

            return True
        except SqliteError as e:
            log(e)
            return False

    @classmethod
    def get_animes_in_list_for_user(cls, user_id: int, list_id: int, search_terms: str = None) -> [Anime]:
        """Récupère la liste de tout les animes d'une liste

            Arguments:
                user_id {int} -- Id de l'utilisateur
                list_id {int} -- Id de la liste
                search_terms {str} -- Chaine de caractères pour la recherche d'anime dans une liste
            Returns:
                [Anime] -- Liste de tous les animes trouvés
        """
        try:
            animes = {}

            sql_select = """SELECT list.idList, list.nameList, anime.idAnime, anime.title, nameType, list_has_anime.idList, anime.episodes, nameStatus, anime.picture, anime.thumbnail, anime.synonyms
                            FROM anime
                            JOIN anime_ft ON anime.idAnime = anime_ft.idAnime
```

```
                                JOIN list_has_anime ON list_has_anime.idAnime = anime.idAnime
                                JOIN user_has_list ON user_has_list.idList = list_has_anime.idList
                                JOIN list ON list.idList = list_has_anime.idList
                                JOIN type ON type.idType = anime.type
                                JOIN status ON status.idStatus = anime.status
                                WHERE user_has_list.idUser = ?"""

            if search_terms is not None and search_terms != "":
                sql_select += " AND anime_ft.title MATCH ?"

                if list_id is not None:
                    sql_select += " AND list_has_anime.idList = ?"
                    values = (user_id, search_terms, list_id,)
                else:
                    values = (user_id, search_terms,)
            elif list_id is not None:
                sql_select += " AND list_has_anime.idList = ?"
                values = (user_id, list_id,)
            else:
                values = (user_id,)

            sql_select += " ORDER BY list.idList"

            rows = SqliteController().execute(sql_select, values=values, fetch_mode=SqliteController.FETCH_ALL)

            # Ajoute les animes dans la bonne liste
            for row in rows:
                new_name = row['nameList']
                if new_name not in animes:
                    animes[new_name] = []
                animes[new_name].append(Anime(row['idAnime'], row['title'], row['nameType'], row['episodes'], row['nameStatus'], row['picture'], row['thumbnail'], row['synonyms']).serialize())

            return animes
        except SqliteError as e:
```

```
            log(e)
            return {}

    @classmethod
    def reorder_favorites(cls, user_id: int, ids: list) -> bool:
        """Met à jour l'ordre des animes

            Arguments:
                user_id {int} -- Id de l'utilisateur
                ids {list} -- Liste des ids des animes

            Returns:
                bool -- État de la mise à jour
        """
        try:
            sql_update = "UPDATE user_has_favorite SET orderId = ?, modificationDate = ? WHERE idAnime = ? AND idUser = ?"

            order_id = 1
            modification_date = dt.now().strftime('%Y-%m-%d %H:%M:%S')

            for anime_id in ids:
                SqliteController().execute(sql_update, values=(order_id, modification_date, anime_id, user_id,), fetch_mode=SqliteController.NO_FETCH)
                order_id += 1

            return True
        except SqliteError as e:
            log(e)
            return False

```

```python
"""Contient la classe de contrôle de la base de données MySQL

@author: Tanguy Cavagna
@copyright: Copyright 2020, TPI
@version: 1.0.0
@date: 2020-06-04
"""
from typing import Any

import mysql.connector
from mysql.connector import Error
from .logger import log

class MySQLController:
    """Classe permettant d'accéder à ma base de données MySQL
    """

    FETCH_ALL = 0
    FETCH_ONE = 1
    NO_FETCH = 2

    def __init__(self):
        """Constructeur vide
        """

    def __getattr__(self, attr):
        """Essaie d'accéder à une variable ou fonction innexistante
        """
        return "Tried to handle unknown method / attribute : {0}".format(attr)

    @classmethod
    def __connection(cls) -> mysql.connector.MySQLConnection:
        """Créer et retourne la connexion

            Returns:
                {mysql.connector.MySQLConnection} -- Nouvelle connexion
        """
        return mysql.connector.connect(host='localhost',
                                       database='tpi',
                                       user='root',
                                       password='root')

    @classmethod
    def openConnection(cls) -> mysql.connector.MySQLConnection
```

```
:
        """Ouvre une connexion

            Returns:
                {mysql.connector.MySQLConnection} -- Connection to the database
        """
        return cls.__connection()

    @classmethod
    def closeConnection(cls, connection) -> None:
        """Ferme une connexion données

            Arguments:
                connection {mysql.connector.MySQLConnection} -- Connexion à fermer
        """
        connection.close()

    @classmethod
    def executeMany(cls, connection, query, values) -> None:
        """Fait une requete de type many. (INSERT INTO <table> VALUES(...values), (...values), (...values), ...)

            Arguments:
                connection {mysql.connector.MySQLConnection} -- Connexion sur laquelle faire la requête
                query {str} -- Requête à faire
                values {tuple} -- Valeurs à attribué à la requête
        """
        try:
            cursor = connection.cursor(prepared=True)
            cursor.executemany(query, values)
            connection.commit()
        except Error as e:
            log(e)

    @classmethod
    def executePrepare(cls, connection, query, values = None, fetch_mode = 0) -> Any:
        """Éxecute une requête préparée

            Arguments:
                connection {mysql.connector.MySQLConnection} -- Connexion sur laquelle faire la requête
                query {str} -- Requête à faire
```

```python

            Keyword Arguments:
                values {tuple} -- Valeurs à attribué à la requête (default: {None})
                fetch_mode {int} -- Type de fetch (default: {0})

            Returns:
                Depending on which fetch mod you are, you could get bool or a list of tuples
        """
        rows = None

        try:
            cursor = connection.cursor(prepared=True)

            cursor.execute(query, values)

            fields_names = [i[0] for i in cursor.description]

            # Fetch datas
            if fetch_mode == cls.FETCH_ALL:
                results = cursor.fetchall()

                rows = [dict(zip(fields_names, results[i])) for i in range(len(results))]
            elif fetch_mode == cls.FETCH_ONE:
                rows = cursor.fetchone()
            elif fetch_mode == cls.NO_FETCH:
                rows = None

            # Return datas
            if rows is not None:
                return rows

            return None
        except Error as e:
            log(e)
            return False
        finally:
            cls.closeConnection(connection)

    @classmethod
    def executeQuery(cls, connection, query, values = None, fetch_mode = 0, last_insert_id = False, close_connection = True) -> Any:
        """Execute and retrieve data from querie
```

```python

            Arguments:
                connection {mysql.connector.MySQLConnection} -- Connexion sur laquelle faire la requête
                query {str} -- Requête à faire

            Keyword Arguments:
                values {tuple} -- Valeurs à attribué à la requête (default: {None})
                fetch_mode {int} -- Type de fetch (default: {0})
                last_insert_id {bool} -- Retourne le last_insert_id ou pas (default: {False})
                close_connection {bool} -- Fermer la connexion à la fin de l'éxecution de la requête

            Returns:
                Depending on which fetch mod you are, you could get bool or a list of tuples
        """
        rows = None
        last_id = None

        try:
            cursor = connection.cursor(dictionary=True)

            # Execute the query
            if (values is not None):
                cursor.execute(query, values)
                if not query.startswith("SELECT"):
                    connection.commit()
            else:
                cursor.execute(query)
                if query.startswith("DELETE"):
                    connection.commit()


            # Fetch datas
            if fetch_mode == cls.FETCH_ALL:
                rows = cursor.fetchall()
            elif fetch_mode == cls.FETCH_ONE:
                rows = cursor.fetchone()
            elif fetch_mode == cls.NO_FETCH:
                rows = None

            if last_insert_id:
                last_id = cursor.lastrowid
```

```python

            if last_id is not None:
                return last_id

            # Return datas
            if rows is not None:
                return rows

            return None
        except Error as e:
            log(e)
            return False
        finally:
            if close_connection:
                cls.closeConnection(connection)

```

```python
"""Contient la classe de contrôle de la base de données Sqlite

@author: Tanguy Cavagna
@copyright: Copyright 2020, TPI
@version: 1.0.0
@date: 2020-05-26
"""
import sqlite3
from typing import Any
from sqlite3 import Error as SqliteError
from mysql.connector import Error as MysqlError
from flask import g

from .MySQLController import MySQLController
from .logger import log

class SqliteController:
    """Classe permettant d'accéder à ma base de données Sqlite
    """

    FETCH_ALL = 0
    FETCH_ONE = 1
    NO_FETCH = 2

    connection = None

    @classmethod
    def __get_cursor(cls) -> object:
        """Récupère le curseur de ma connexion
        """
        return cls.get_instance().cursor()

    #pylint: disable=no-self-use
    @classmethod
    def __dict_factory(cls, cursor: object, row: object) -> dict:
        """Retourne les valeurs du fetch comme dictionnaire
            see: https://stackoverflow.com/questions/3300464/how-can-i-get-dict-from-sqlite-query

            Arguments:
                cursor {object} -- Cursor of the connection
                row {object} -- Returned rows

            Returns:
                dict -- Full dict
        """
```

```
        d = {}
        for idx, col in enumerate(cursor.description):
            d[col[0]] = row[idx]
        return d

    @classmethod
    def get_instance(cls) -> object:
        """Récupère l'instance de connexion à la base
        """
        SqliteController.connection = getattr(g, '_database', None)
        if SqliteController.connection is None:
            SqliteController.connection = g._database = sqlite3.connect("/home/cavagnat/Documents/programmation/python/Animanga/static/database/tpi.db", detect_types=sqlite3.PARSE_DECLTYPES)
            SqliteController.connection.row_factory = cls.__dict_factory

        return SqliteController.connection

    #pylint: disable=no-self-use
    @classmethod
    def close(cls) -> None:
        """Ferme la connexion courante
        """
        SqliteController.connection = getattr(g, '_database', None)
        if SqliteController.connection is not None:
            SqliteController.connection.close()

    @classmethod
    def execute_many(cls, query: str, values: tuple) -> None:
        """Fait une requete de type many. (INSERT INTO <table> VALUES(...values), (...values), (...values), ...)
        """
        cursor = cls.__get_cursor()

        cursor.executemany(query, values)

        cls.get_instance().commit()

    @classmethod
    def execute(cls, query: str, values = None, fetch_mode = 2) -> Any:
        """Exécute une requete en base

```

```
            Arguments:
                query {str} -- Requete à executer

            Keyword Arguments:
                values {tuple} -- Valeurs à utilisé (default : {None})
                fetch_mode {int} -- Type de fetch voulu (default: {2}) -> pas de fetch

            Returns:
                None|[]|int -- Si pas de fetch, retourne le last_row_id, sinon retourne le fetch voulu
        """
        try:
            cursor = cls.__get_cursor()

            if values is not None:
                cursor.execute(query, values)
            else:
                cursor.execute(query)

            last_row_id = cursor.lastrowid

            cls.get_instance().commit()

            if fetch_mode == cls.FETCH_ONE:
                result = cursor.fetchone()
            elif fetch_mode == cls.FETCH_ALL:
                result = cursor.fetchall()
            else:
                result = None

            if (fetch_mode == cls.NO_FETCH):
                return last_row_id

            return result
        except sqlite3.Error as e:
            raise e

    # Données de la base de données

    @classmethod
    def truncate_anime_related(cls) -> None:
        """Vide toutes les tables relatives aux animes
        """
        try:
            cls.execute("DELETE FROM `user_has_favorite`")
```

```
            cls.execute("DELETE FROM `list_has_anime`")
            cls.execute("DELETE FROM `anime_ft`")
            cls.execute("DELETE FROM `anime`")
            cls.execute("DELETE FROM `url`")
            cls.execute("DELETE FROM `type`")
            cls.execute("DELETE FROM `status`")

            return True
        except SqliteError as e:
            log(e)
            return False

    @classmethod
    def setup_anime_table(cls) -> None:
        """Créer la table anime si elle n'existe pas
        """
        try:
            cls.execute(
                """
                    CREATE TABLE IF NOT EXISTS `anime` (
                        `idAnime` integer NOT NULL PRIMARY KEY,
                        `title` text NOT NULL,
                        `type` integer NOT NULL,
                        `episodes` integer NOT NULL,
                        `status` integer NOT NULL,
                        `picture` text NOT NULL,
                        `thumbnail` text NOT NULL,
                        `synonyms` text DEFAULT NULL,
                        `modificationDate` timestamp DEFAULT NULL,
                        FOREIGN KEY (`type`) REFERENCES `type`(`idType`),
                        FOREIGN KEY (`status`) REFERENCES `status`(`idStatus`)
                    )
                """,
                None
            )
            # Obligatoire pour pouvoir faire une recherche full text
            cls.execute("CREATE VIRTUAL TABLE IF NOT EXISTS `anime_ft` USING FTS5(`idAnime`, `title`)")
            return True
        except SqliteError as e:
            log(e)
            return False
```

```python

    @classmethod
    def setup_type_table(cls) -> None:
        """Créer la table type si elle n'existe pas
        """
        try:
            cls.execute(
                """
                    CREATE TABLE IF NOT EXISTS `type` (
                        `idType` integer NOT NULL PRIMARY KEY,
                        `nameType` text NOT NULL,
                        `modificationDate` timestamp DEFAULT NULL
                    )
                """,
                None
            )
            return True
        except SqliteError as e:
            log(e)
            return False

    @classmethod
    def setup_status_table(cls) -> None:
        """Créer la status anime si elle n'existe pas
        """
        try:
            cls.execute(
                """
                    CREATE TABLE IF NOT EXISTS `status` (
                        `idStatus` integer NOT NULL PRIMARY KEY,
                        `nameStatus` text NOT NULL,
                        `modificationDate` timestamp DEFAULT NULL
                    )
                """,
                None
            )
            return True
        except SqliteError as e:
            log(e)
            return False

    @classmethod
    def setup_url_table(cls) -> None:
```

```
        """Créer la url anime si elle n'existe pas
        """
        try:
            cls.execute(
                """
                    CREATE TABLE IF NOT EXISTS `url` (
                        `idUrl` integer NOT NULL PRIMARY KEY,
                        `urlName` text NOT NULL,
                        `idAnime` integer NOT NULL,
                        `isRelation` integer NOT NULL,
                        `modificationDate` timestamp DEFAULT NULL,
                        FOREIGN KEY (`idAnime`) REFERENCES `anime`(`idAnime`)
                    )
                """
            )
            return True
        except SqliteError as e:
            log(e)
            return False

    @classmethod
    def setup_user_table(cls) -> None:
        """Créer la user anime si elle n'existe pas
        """
        try:
            cls.execute(
                """
                    CREATE TABLE IF NOT EXISTS `user` (
                        `idUser` integer NOT NULL PRIMARY KEY,
                        `emailUser` text NOT NULL,
                        `password` text NOT NULL,
                        `nicknameUser` text NOT NULL,
                        `modificationDate` timestamp DEFAULT NULL
                    )
                """,
                None
            )
            return True
        except SqliteError as e:
            log(e)
            return False

    @classmethod
```

```
    def setup_list_table(cls) -> None:
        """Créer la list anime si elle n'existe pas
        """
        try:
            cls.execute(
                """
                    CREATE TABLE IF NOT EXISTS `list` (
                        `idList` integer NOT NULL PRIMARY KEY,
                        `nameList` text NOT NULL,
                        `modificationDate` timestamp DEFAULT NULL
                    )
                """,
                None
            )
            return True
        except SqliteError as e:
            log(e)
            return False

    @classmethod
    def setup_user_has_list_table(cls) -> None:
        """Créer la user_has_list anime si elle n'existe pas
        """
        try:
            cls.execute(
                """
                    CREATE TABLE IF NOT EXISTS `user_has_list` (
                        `idUserHasList` integer NOT NULL PRIMARY KEY,
                        `idUser` integer NOT NULL REFERENCES `user`(`idUser`),
                        `idList` integer NOT NULL REFERENCES `list`(`idList`),
                        `modificationDate` timestamp DEFAULT NULL
                    )
                """,
                None
            )
            return True
        except SqliteError as e:
            log(e)
            return False

```

```
    @classmethod
    def setup_list_has_anime_table(cls) -> None:
        """Créer la table list_has_anime si elle n'existe pas
        """
        try:
            cls.execute(
                """
                    CREATE TABLE IF NOT EXISTS `list_has_anime` (
                        `idListHasAnime` integer NOT NULL PRIMARY KEY,
                        `idList` integer NOT NULL REFERENCES `list`(`idList`),
                        `idAnime` integer NOT NULL REFERENCES `anime`(`idAnime`),
                        `modificationDate` timestamp DEFAULT NULL
                    )
                """,
                None
            )
            return True
        except SqliteError as e:
            log(e)
            return False

    @classmethod
    def setup_user_has_favorite_table(cls) -> None:
        """Créer la table user_has_favorite si elle n'existe pas
        """
        try:
            cls.execute(
                """
                    CREATE TABLE IF NOT EXISTS `user_has_favorite` (
                        `idUserHasFavorite` integer NOT NULL PRIMARY KEY,
                        `idUser` integer NOT NULL REFERENCES `user`(`idUser`),
                        `idAnime` integer NOT NULL REFERENCES `anime`(`idAnime`),
                        `orderId` integer NOT NULL,
                        `modificationDate` timestamp DEFAULT NULL
                    )
                """,
```

```
                None
            )
            return True
        except SqliteError as e:
            log(e)
            return False

    # pylint: disable=too-many-locals,too-many-nested-blocks
    @classmethod
    def synchronise_with_mysql(cls) -> bool:
        """Synchronise la base MySQL avec Sqlite3

            Returns:
                bool -- Status de la synchronisation
        """
        try:
            tables = {
                'type': 'idType',
                'status': 'idStatus',
                'anime': 'idAnime',
                'url': 'idUrl',
                'user': 'idUser',
                'list': 'idList',
                'user_has_list': 'idUserHasList',
                'list_has_anime': 'idListHasAnime',
                'user_has_favorite': 'idUserHasFavorite',
            }
            sql_table_columns = "PRAGMA table_info("
            sql_entries = "SELECT * FROM "
            sql_insert = "INSERT INTO "
            sql_delete = "DELETE FROM "

            connection = MySQLController().openConnection()
            MySQLController().executeQuery(connection, "SET FOREIGN_KEY_CHECKS=0;", fetch_mode=MySQLController.NO_FETCH, close_connection=False)

            for current_table in tables:
                print('TABLE COURANTE :', current_table)

                # Récupération des noms de colonnes de la table courante
                current_tables_columns = SqliteController().execute(sql_table_columns + current_table + ')', fetch_mode=SqliteController.FETCH_ALL)
                current_tables_columns = [item['name'] for item in current_tables_columns]
```

```python

                # Récupération des enregistrements de la table courante sur sqlite et mysql
                sqlite_entries = cls.execute(sql_entries + current_table, fetch_mode=SqliteController.FETCH_ALL)
                mysql_entries = MySQLController().executeQuery(connection, sql_entries + current_table, fetch_mode=MySQLController.FETCH_ALL, close_connection=False)

                # Création d'un tableau contenant l'id et la modificationDate de chaque enregistrement
                sqlite_ids_timestamps = [{'id': sqlite_entry[tables[current_table]], 'timestamp': sqlite_entry['modificationDate'].replace(microsecond=0)} for sqlite_entry in sqlite_entries]
                mysql_ids_timestamps = [({'id': mysql_entry[tables[current_table]], 'timestamp': mysql_entry['modificationDate'].replace(microsecond=0)}) for mysql_entry in mysql_entries]

                # Tri des tableaux par id
                sqlite_ids_timestamps = sorted(sqlite_ids_timestamps, key=lambda k: k['id'])
                mysql_ids_timestamps = sorted(mysql_ids_timestamps, key=lambda k: k['id'])

                print('Data same', len(sqlite_entries) == len(mysql_entries))
                ids_to_skip_for_update = []
                # Ajout des données manquantes mysql et suppression des données en trop mysql
                if len(sqlite_entries) != len(mysql_entries):
                    # Ajout des éléments manquants dans mysql
                    ids_non_present_in_mysql = []
                    if len(sqlite_ids_timestamps) > len(mysql_ids_timestamps):
                        # Éléments non présents dans la table myqsl
                        ids_non_present_in_mysql = [i['id'] for i in sqlite_ids_timestamps if i['id'] not in [j['id'] for j in mysql_ids_timestamps]]

                        # Récupère les valeurs sqlite a ajouté dans mysql seulement si elles ne sont déjà présentes dans mysql
                        values_to_append = []
                        for sqlite_entry in sqlite_entries:
```

```
                                if sqlite_entry[tables[current_table]] in ids_non_present_in_mysql:
                                    values_to_append.append(tuple(sqlite_entry.values()))
                                    ids_to_skip_for_update.append(sqlite_entry[tables[current_table]])

                            # Construction de la requête d'ajout de données et insertion des données dans mysql
                            columns_list = ''
                            for column in current_tables_columns: columns_list += column + ','
                            parameter_list = ''
                            for _ in range(len(current_tables_columns)): parameter_list += '%s,'
                            # Format de la requete :
                            #       INSERT INTO <table>(...champs) VALUES(...valeurs)
                            MySQLController().executeMany(connection,
                                                          sql_insert + current_table + "(" + columns_list[:-1] + ") VALUES(" + parameter_list[:-1] + ")",
                                                          values=values_to_append)
                        # Suppression des éléments en trop de mysql
                        elif len(sqlite_ids_timestamps) < len(mysql_ids_timestamps):
                            # Éléments non présents dans la tablea sqlite
                            ids_non_present_in_sqlite = [i['id'] for i in mysql_ids_timestamps if i['id'] not in [j['id'] for j in sqlite_ids_timestamps]]

                            # Construction de la requête de suppressions de données
                            delete_list = ''
                            for delete in ids_non_present_in_sqlite: delete_list += str(delete) + ','

                            # Format de la requete :
                            #       DELETE FROM <table> WHERE <id table> IN (...valeurs)
                            MySQLController().executeQuery(connection,
                                                           
```

```
sql_delete + current_table + " WHERE " + tables[current_table] + " IN (" + delete_list[:-1] + ")",
fetch_mode=MySQLController.NO_FETCH,
close_connection=False)

                # Réafectation des variables pour avoir les données à jour
                mysql_entries = MySQLController().executeQuery(connection, sql_entries + current_table, fetch_mode=MySQLController.FETCH_ALL, close_connection=False)
                mysql_ids_timestamps = [({'id': mysql_entry[tables[current_table]], 'timestamp': mysql_entry['modificationDate'].replace(microsecond=0)}) for mysql_entry in mysql_entries]
                mysql_ids_timestamps = sorted(mysql_ids_timestamps, key=lambda k: k['id'])

                # Récupération des id à mettre à jour dans mysql
                elements_to_updates = []
                for sqlite_element in mysql_ids_timestamps:
                    if sqlite_element['id'] not in ids_to_skip_for_update:
                        # Format de la requete :
                        #       SELECT <id table> AS `id` FROM <table> WHERE modificationDate NOT LIKE '<date mysql>%' AND <id table> = <id mysql>
                        # Je suis obliger de mettre le `NOT LIKE` pour la date car Sqlite contient les millisecondes et pas mysql
                        entry = SqliteController().execute("SELECT " + tables[current_table] + " AS `id` FROM " + current_table + " WHERE modificationDate NOT LIKE '" + str(sqlite_element['timestamp']) + "%' AND " + tables[current_table] + " = " + str(sqlite_element['id']),
fetch_mode=SqliteController.FETCH_ONE)
                        if entry is not None:
                            elements_to_updates.append(entry['id'])

                print(elements_to_updates)
                # A voir avec Bonvin car trop long
                for index in elements_to_updates:
                    # Récupération des valeurs sqlite à
```

```
mettre dans mysql
                    values_to_append = None
                    for item in sqlite_entries:
                        if item[tables[current_table]] == index:
                            # [1:] pour éviter de prendre l'id
                            values_to_append = tuple(list(item.values())[1:])
                            break

                    # Mise à jours des données mysql avec les valeurs sqlite
                    set_list = ''
                    # [1:] pour éviter de mettre à jour l'id
                    for column in current_tables_columns[1:]: set_list += column + "=%s,"
                    MySQLController().executeQuery(connection,
                                                    "UPDATE " + current_table + " SET " + set_list[:-1] + " WHERE " + tables[current_table] + " = " + str(index),
                                                    values=values_to_append,
                                                    fetch_mode=MySQLController.NO_FETCH,
                                                    close_connection=False)
            MySQLController().executeQuery(connection, "SET FOREIGN_KEY_CHECKS=1;", fetch_mode=MySQLController.NO_FETCH, close_connection=False)
            MySQLController().closeConnection(connection)
            return True
        except (SqliteError, MysqlError, ValueError, TypeError) as e:
            log(e)
            return False

```

```python
"""Contient la classe de contrôle des status
Les status sont : FINISHED, CURRENTLY, UPCOMING, UNKNOWN

@author: Tanguy Cavagna
@copyright: Copyright 2020, TPI
@version: 1.0.0
@date: 2020-05-26
"""
from sqlite3 import Error as SqliteError
from .SqliteController import SqliteController
from ..models.Status import Status
from .logger import log

class StatusController:
    """Controlleur des status
    """

    def __init__(self):
        pass

    @classmethod
    def get_all(cls) -> [Status]:
        """Récupère tout les statuts

            Returns:
                [Status] -- Listes contenant les statuts
        """
        try:
            rows = SqliteController().execute("SELECT `idStatus`, `nameStatus` FROM `status`", fetch_mode=SqliteController().FETCH_ALL)
            status = [Status(t['idStatus'], t['nameStatus']) for t in rows]

            return status
        except SqliteError as e:
            log(e)
            raise e

    @classmethod
    def get_all_as_dict(cls) -> dict:
        """Récupère tout les statuts sous forme de dictionnaire

            Returns:
                dict -- Dictionnaire contenant les statuts avec le format "name": "id"
```

```
        """
        status = {}
        for s in cls.get_all():
            status.update({s.name: s.id})

        return status
```

```python
"""Contient la classe de contrôle des activités

@author: Tanguy Cavagna
@copyright: Copyright 2020, TPI
@version: 1.1.0
@date: 2020-06-02
"""
from datetime import timedelta
from datetime import datetime as dt
from sqlite3 import Error as SqliteError

from .SqliteController import SqliteController

class ActivitiesController:
    """Controlleur d'une activité
    """

    def __init__(self):
        pass

    @classmethod
    def get_last_24h(cls, user_id: int) -> []:
        """Récupère les activitées d'un utilisateur

            Arguments:
                user_if {int} -- Id de l'utilisateur
            Returns:

                [] -- Liste des activitées
        """
        activities = cls.__get_list_related(user_id) + cls.__get_favorite_related(user_id)
        # Trie toutes les activitées par date
        activities.sort(key=lambda item: item['date'], reverse=True)

        # Change le format de toutes les dates des activitées
        for a in activities:
            a['date'] = a['date'].strftime('%Y-%m-%d %H:%M:%S')

        return activities

    @classmethod
    def __get_list_related(cls, user_id: int) -> []:
        """Récupère les activitées relatives aux listes

```

```
            Arguments:
                user_id {int} -- Id de l'utilisateur
            Returns:

                [] -- Liste des activités
        """
        try:
            current_datetime = dt.now().replace(microsecond=0)

            sql_list_additions = """SELECT anime.title, anime.picture, list_has_anime.modificationDate, list.nameList
                                    FROM list_has_anime
                                    JOIN list ON list.idList = list_has_anime.idList
                                    JOIN user_has_list ON user_has_list.idList = list.idList
                                    JOIN anime ON anime.idAnime = list_has_anime.idAnime
                                    WHERE user_has_list.idUser = ? AND list_has_anime.modificationDate > ?"""

            user_list_additions = SqliteController().execute(sql_list_additions, values=(user_id,(current_datetime - timedelta(days=1)),), fetch_mode=SqliteController.FETCH_ALL)

            results = []
            for add in user_list_additions:
                results.append({
                    'title': add['title'],
                    'list': add['nameList'],
                    'picture': add['picture'],
                    'date': add['modificationDate']
                })

            return results
        except SqliteError as e:
            print(str(e))
            return []

    @classmethod
    def __get_favorite_related(cls, user_id: int) -> []:
        """Récupère les activitées relatives aux favoris

            Arguments:
                user_id {int} -- Id de l'utilisateur
            Returns:

```

```
                [] -- Liste des activités
        """
        try:
            current_datetime = dt.now().replace(microsecond=0)

            sql_favorite_additions = """SELECT anime.title, anime.picture, user_has_favorite.modificationDate
                                        FROM user_has_favorite
                                        JOIN anime ON anime.idAnime = user_has_favorite.idAnime
                                        WHERE user_has_favorite.idUser = ? AND user_has_favorite.modificationDate > ?"""

            user_favorite_additions = SqliteController().execute(sql_favorite_additions, values=(user_id, (current_datetime - timedelta(days=1)),), fetch_mode=SqliteController.FETCH_ALL)

            results = []
            for add in user_favorite_additions:
                results.append({
                    'title': add['title'],
                    'list': 'favoris',
                    'picture': add['picture'],
                    'date': add['modificationDate']
                })

            return results
        except SqliteError as e:
            print(str(e))
            return []
```

```
{% extends 'layouts/layout.html' %}

{% block title %}
    Aide
{% endblock title %}

{% block content %}
    <div class="toc"></div>
    <div class="container about-content mt-5">
        <h1>Aide</h1>
        {{ html | safe }}
    </div>
{% endblock content %}

{% block script %}
    <script src="{{ url_for('static', filename='js/toc-generator.js') }}"></script>
{% endblock script %}
```

```
1 {% extends 'layouts/layout.html' %}
2
3 {% block title %}
4     À propos
5 {% endblock title %}
6
7 {% block content %}
8     <div class="toc"></div>
9     <div class="container about-content mt-5">
10        <h1>À propos</h1>
11        <p>
12            Ce projet a été réalisé dans le cadre du <em>
  Travail Pratique Individuel</em> (TPI) durant la session de
  mai - juin 2020.
13            Il a pour but de valider les compétences acquises
  tout au long de la formation *Informaticien CFC* de l'école du
   CFPT-Informatique au Petit-Lancy.
14        </p>
15        <p>
16            Animanga est une application web, écrite en Python
  , permettant aux utilisateurs de faire leur propre
  bibliothèque d'anime.
17            Pour ce faire, l'utilisateur a la possibilité de
  créer ses propres listes afin de correctement organiser
18            sa bibliothèque, mettre des animes en tant que
  favoris, et rechercher les animes qu'il voudrait ajouter à sa
  collection directement depuis cette application.
19        </p>
20        {{ html | safe }}
21    </div>
22 {% endblock content %}
23
24 {% block script %}
25    <script src="{{ url_for('static', filename='js/toc-
  generator.js') }}"></script>
26 {% endblock script %}
```

```
1 {% extends 'layouts/layout.html' %}
2
3 {% block title %}
4     Accueil
5 {% endblock title %}
6
7 {% block content %}
8     <div id="override-info"></div>
9
10    {% if not current_user.is_authenticated %}
11        <div class="background">
12            <img src="{{ url_for('static', filename='img/home-background.png') }}">
13        </div>
14    {% else %}
15        <main class="mt-5 main-home row">
16            <section class="col-md-12 col-xl-8">
17                {% if search_results|length <= 0 %}
18                    <div class="no-search">
19                        <h4>Aucune recherche effectuée</h4>
20                        <span class="separator"></span>
21                        <span>
22                            Faites une recherche pour obtenir des informations sur l'anime souhaité
23                        </span>
24                        <span>(´。_。`)</span>
25                    </div>
26
27                    <div class="activities"></div>
28                {% else %}
29                    <h4>Résultats pour "{{ search_string }}" :</h4>
30
31                    <div class="search-results">
32                        {% for anime in search_results %}
33                            <div class="result-item">
34                                <div class="left">
35                                    <img class="picture" src="{{ anime.picture }}">
36                                    <span class="title" data-toggle="modal" data-target="#result-item-{{ anime.id }}">{{ anime.title }}</span>
37                                </div>
38                                <div class="right">
39                                    <div class="status">{{ anime.status }}</div>
40                                    <div class="type">{{ anime
```

```
.type }}</div>
                                    <div class="relations row">
                                        {% for relation in anime.relations %}
                                            <img class="relation col-5" src="{{ relation.picture }}" title="{{ relation.title }}" alt="404">
                                        {% endfor %}
                                    </div>
                                    <div class="episodes">{{ anime.episodes }}</div>
                                </div>

                                <!-- Modal -->
                                <div class="modal fade" id="result-item-{{ anime.id }}" tabindex="-1" role="dialog" aria-labelledby="exampleModalLabel" aria-hidden="true">
                                    <div class="modal-dialog modal-dialog-centered" role="document">
                                        <div class="modal-content">
                                            <div class="header">
                                                <div class="header-background">
                                                    <img src="{{ url_for('static', filename='img/home-background.png') }}">
                                                    <div class="dark-layer"></div>
                                                </div>
                                                <img class="thumbnail" src="{{ anime.picture }}">
                                                <div class="title">{{ anime.title }}</div>
                                                <div class="favorite-toggler {{ 'is-favorite' if anime.is_favorite else '' }}" data-id="{{ anime.id }}">
                                                    <svg width="21" height="19" fill="none" xmlns="http://www.w3.org/2000/svg">
                                                        <path d="M19.291 1.612a5.5 5.5 0 00-7.78 0l-1.06 1.06-1.06-1.06a5.501 5.501 0 10-7.78 7.78l1.06 1.06 7.78 7.78 7.78-7.78 1.06-1.06a5.5 5.5 0 000-7.78z" fill="#fff"/>
                                                    </svg>
```

```
                                                    </div>
                                                </div>

                                                <div class="content">
                                                    <fieldset class="status">
                                                        <span>Status</span>
                                                        <select name="status" class="status_combo">
                                                            <option value="{{ anime.id }}-none">---</option>
                                                            {% for list in user_list[:4] %}
                                                                <option value="{{ anime.id }}-{{ list.id }}" {{ 'selected' if list.id in anime.in_list else '' }}>{{ list.name }}</option>
                                                            {% endfor %}
                                                        </select>
                                                    </fieldset>

                                                    <fieldset class="custom">
                                                        <span>Listes personnelles</span>

                                                        <div class="lists">
                                                            {% for list in user_list[4:] %}
                                                                <div class="list-item">
                                                                    <div class="custom-check {{ 'checked' if list.id in anime.in_list else '' }}">
                                                                        <input type="checkbox" value="{{ anime.id }}-{{ list.id }}">
                                                                    </div>
                                                                    <span class="list-title">{{ list.name }}</span>
                                                                </div>
                                                            {%
```

```
 89 endfor %}
 90                                                        </div>
 91                                                    </fieldset>
 92                                                </div>
 93                                            </div>
 94                                        </div>
 95                                    </div>
 96                                </div>
 97                            {% endfor %}
 98                        </div>
 99                    {% endif %}
100                </section>
101
102                <aside class="col-md-12 col-xl-4">
103                    <h4>Favoris</h4>
104
105                    <div class="favorites-container">
106                        <div class="favorites"></div>
107                    </div>
108                </aside>
109            </main>
110        {% endif %}
111 {% endblock content %}
112
113 {% block script %}
114        <script src="{{ url_for('static', filename='js/landing-
    page-handler.js') }}"></script>
115        <script src="{{ url_for('static', filename='js/fetch-
    favorites.js') }}"></script>
116        <script src="{{ url_for('static', filename='js/fetch-
    activities.js') }}"></script>
117        <script src="{{ url_for('static', filename='js/ratio-
    keeper.js') }}"></script>
118        <script src="{{ url_for('static', filename='js/anime-
    handler.js') }}"></script>
119        <script>
120            window.addEventListener('load', () => {
    fetchFavorites(false, false, null); fetchActivities(); });
121
122            // Déclaration obligatoire pour que les message du
    stream puissent être mis dans un élément DOM
123            let override_info = document.getElementById('override
    -info');
124        </script>
125        <!-- Affiche les messages envoyé par la fonction
    stream_template -->
126        {% for message in override_messages: %}
```

```
        <script>
            // Prise en compte du message special 'redirect' pour rediriger l'utilisateur et de ce fait, arréter le stream courant
            if ("{{ message }}" == "redirect") {
                override_info.style.padding = '0';
                window.location.replace("/");
            }

            // Condition obligatoire à cause du script ci-dessus ayant déjà cette variable de déclaré
            if (typeof override_info !== 'undefined') {
                override_info = document.querySelector('#override-info');
            } else {
                let override_info = document.querySelector('#override-info');
            }

            // Utilisation de 'var' pour ne pas avoir de soucis avec des variables déjà déclarlé lors de message précédents
            var msg = document.createElement('span');
            var loader = document.createElement('img');

            override_info.innerHTML = '';
            override_info.style.padding = '.75rem 0';
            msg.innerHTML = '{{ message }}';
            loader.src = '/static/img/loader.svg';
            loader.classList.add('ml-5');
            override_info.appendChild(msg);
            override_info.appendChild(loader);
        </script>
    {% endfor %}
{% endblock script %}
```

```
1 {% extends 'layouts/layout.html' %}
2
3 {% block metas %}
4 <meta id="searched_user" data-nickname="{{ searched_user.
  nickname }}">
5 {% endblock metas %}
6
7 {% block title %}
8     Listes - {{ searched_user.nickname }}
9 {% endblock title %}
10
11 {% block content %}
12     {% include 'layouts/profile-header.html' %}
13
14     <div class="row mt-4">
15         <div class="col-xl-2 col-md-12 mb-5" id="lists">
16             <div class="lists-container">
17                 <div id="search-list">
18                     <img src="{{ url_for('static', filename='
  img/search.svg') }}" width="20">
19                     <input type="text" class="search-string"
  placeholder="Rechercher">
20                 </div>
21
22                 <h4>Listes</h4>
23
24                 <ul>
25                     <li class="selected"><span class="
  list__name">Tous</span></li>
26                     {% for list in user_lists %}
27                         {% if loop.index > 4 and current_user.
  nickname == searched_user.nickname %}
28                             <li data-list="{{ list.id }}">
29                                 <span class="list__name">{{
  list.name }}</span>
30                                 <input type="text" class="d-
  none" style="width: 80%" name="new_list_name" id="{{ list.id
   }}_new_list_name">
31                                 <span class="cancel__rename d-
  none">?</span>
32                                 <span class="remove__list">?</
  span>
33                             </li>
34                         {% else %}
35                             <li data-list="{{ list.id }}"><
  span class="list__name">{{ list.name }}</span></li>
36                         {% endif %}
```

```
                    {% endfor %}
                    {% if current_user.nickname == searched_user.nickname %}
                        <li class="new-list">
                            <input type="text" name="new-list__name" id="new-list__name" placeholder="Nouvelle liste">
                        </li>
                    {% endif %}
                </ul>
            </div>
        </div>
        <div class="col-xl-10 col-md-12 p-md-5 p-xl-0" id="animes-container"></div>
    </div>
{% endblock content %}

{% block script %}
<script src="{{ url_for('static', filename='js/user-lists-handler.js') }}"></script>
{% endblock script %}
```

```
1 {% extends 'layouts/layout.html' %}
2
3 {% block title %}
4     Connexion
5 {% endblock title %}
6
7 {% block content %}
8     <div class="container-center">
9         {% for message in get_flashed_messages() %}
10            <div class="alert alert-warning" id="flash">
11                <button type="button" class="close" data-dismiss="alert"><img src="{{ url_for('static', filename='img/close.svg') }}"></button>
12                {{ message }}
13            </div>
14        {% endfor %}
15
16        <div class="login-form">
17            <h3>Connexion</h3>
18
19            <form action="/login" method="post">
20                <fieldset>
21                    <input type="email" name="email" id="email" placeholder="Email">
22                    {% if auth_errors.email %}
23                        <ul id="email-error">
24                            {% for error in auth_errors.email %}<li>{{ error }}</li>{% endfor %}
25                        </ul>
26                    {% endif %}
27                </fieldset>
28
29                <fieldset>
30                    <input type="password" name="password" id="password" placeholder="Mot de passe">
31                    {% if auth_errors.password %}
32                        <ul id="password-error">
33                            {% for error in auth_errors.password %}<li>{{ error }}</li>{% endfor %}
34                        </ul>
35                    {% endif %}
36                </fieldset>
37
38                <div class="submit">
39                    <button type="submit">Connexion</button>
40                    <span>Pas encore inscris ? <a href="/signup">Créer un compte</a></span>
```

```
                </div>
            </form>
        </div>
    </div>
{% endblock content %}
```

```
1 {% extends 'layouts/layout.html' %}
2
3 {% block content %}
4     <div class="container-center">
5         {% for message in get_flashed_messages() %}
6             <div class="alert alert-warning" id="flash">
7                 <button type="button" class="close" data-dismiss="alert"><img src="{{ url_for('static', filename='img/close.svg') }}"></button>
8                 {{ message }}
9             </div>
10        {% endfor %}
11
12        <div class="signup-form">
13            <h3>Inscris toi <br> sur Animanga</h3>
14
15            <form action="/signup" method="post">
16                <fieldset>
17                    <input type="email" name="email" id="email" placeholder="Email">
18                    {% if auth_errors.email %}
19                        <ul id="email-errors">
20                            {% for error in auth_errors.email %}<li>{{ error }}</li>{% endfor %}
21                        </ul>
22                    {% endif %}
23                </fieldset>
24
25                <fieldset>
26                    <input type="text" name="nickname" id="nickname" placeholder="Pseudo">
27                    {% if auth_errors.nickname %}
28                        <ul id="nickname-errors">
29                            {% for error in auth_errors.nickname %}<li>{{ error }}</li>{% endfor %}
30                        </ul>
31                    {% endif %}
32                </fieldset>
33
34                <fieldset>
35                    <input type="password" name="password" id="password" placeholder="Mot de passe">
36                    {% if auth_errors.password %}
37                        <ul id="password-errors">
38                            {% for error in auth_errors.password %}<li>{{ error }}</li>{% endfor %}
39                        </ul>
```

```
                    {% endif %}
                </fieldset>

                <fieldset>
                    <input type="password" name="confirm" id="confirm" placeholder="Confirmer le mot de passe">
                    {% if auth_errors.confirm %}
                        <ul id="confirm-errors">
                            {% for error in auth_errors.confirm %}<li>{{ error }}</li>{% endfor %}
                        </ul>
                    {% endif %}
                </fieldset>

                <div class="submit">
                    <button type="submit">Inscription</button>
                    <span>Déjà un compte ? <a href="{{ url_for('auth_bp.login') }}">Connecte toi</a></span>
                </div>
            </form>
        </div>
    </div>
{% endblock content %}
```

```html
{% extends 'layouts/layout.html' %}

{% block metas %}
<meta id="searched_user" data-nickname="{{ searched_user.nickname }}">
{% endblock metas %}

{% block title %}
    {{ current_user.nickname }} - Profil
{% endblock title %}

{% block content %}
    {% include 'layouts/profile-header.html' %}

    <div class="profile">
        <div class="stats">
            <div class="watched">
                <span class="count">{{ stats['TV'] }}</span>
                <span class="category">TV Regardés</span>
            </div>
            <div class="watched">
                <span class="count">{{ stats['Movie'] }}</span>
                <span class="category">Films Regardés</span>
            </div>
            <div class="watched">
                <span class="count">{{ stats['ONA'] }}</span>
                <span class="category">Ona Regardés</span>
            </div>
            <div class="watched">
                <span class="count">{{ stats['Special'] }}</span>
                <span class="category">Speciaux Regardés</span>
            </div>
        </div>

        <div class="favorites-container">
            <div class="favorites"></div>
        </div>
    </div>
{% endblock content %}

{% block script %}
    <script src="{{ url_for('static', filename='js/fetch-favorites.js') }}"></script>
    <script>
```

```
        window.addEventListener('load', () => { fetchFavorites (false, false, document.getElementById('searched_user').dataset.nickname) });
    </script>
{% endblock script %}
```

```
1 {% extends 'layouts/layout.html' %}
2
3 {% block title %}Titre de la page{% endblock title %}
4
5 {% block extra %}
6 <link rel="stylesheet" type="text/css" href="{{ url_for('
  static', filename='css/swagger-ui.css') }}" >
7 <link rel="icon" type="image/png" href="{{ url_for('static',
  filename='img/favicon-32x32.png') }}" sizes="32x32" />
8 <link rel="icon" type="image/png" href="{{ url_for('static',
  filename='img/favicon-16x16.png') }}" sizes="16x16" />
9 <style>
10     html
11     {
12     box-sizing: border-box;
13     overflow: -moz-scrollbars-vertical;
14     overflow-y: scroll;
15     }
16
17     *,
18     *:before,
19     *:after
20     {
21     box-sizing: inherit;
22     color: black;
23     }
24
25     body
26     {
27     margin:0;
28     background: #fafafa;
29     }
30 </style>
31 {% endblock extra %}
32
33 {% block content %}
34 <div id="swagger-ui"></div>
35 {% endblock content %}
36
37 {% block script %}
38 <script src="{{ url_for('static', filename='js/swagger-ui-
  bundle.js') }}"> </script>
39 <script src="{{ url_for('static', filename='js/swagger-ui-
  standalone-preset.js') }}"> </script>
40 <script>
41 window.onload = function() {
42     // Begin Swagger UI call region
```

```
43     const ui = SwaggerUIBundle({
44         url: "/specs",
45         dom_id: '#swagger-ui',
46         deepLinking: true,
47         supportedSubmitMethods: [], // Used to show the "try
   it out" button. Possible item: ["get", "post", "delete", "put
   "]
48         presets: [
49             SwaggerUIBundle.presets.apis,
50             SwaggerUIStandalonePreset
51         ],
52         plugins: [
53             SwaggerUIBundle.plugins.DownloadUrl
54         ],
55         layout: "StandaloneLayout"
56     });
57     // End Swagger UI call region
58
59     window.ui = ui;
60 }
61 </script>
62 {% endblock script %}
```

```html
{% extends 'layouts/layout.html' %}

{% block title %}
    Listes - {{ searched_user.nickname }}
{% endblock title %}

{% block content %}
    {% include 'layouts/profile-header.html' %}

    <div class="profile">
        <button class="reorder">Réordonner les favoris</button>
        <div class="favorites-container">
            <div class="favorites"></div>
        </div>
    </div>
{% endblock content %}

{% block script %}
<script src="https://code.jquery.com/ui/1.12.1/jquery-ui.min.js" integrity="sha256-VazP97ZCwtekAsvgPBSUwPFKdrwD3unUfSGVYrahUqU=" crossorigin="anonymous"></script>
<script src="{{ url_for('static', filename='js/fetch-favorites.js') }}"></script>
<script src="{{ url_for('static', filename='js/favorites-reorder.js') }}"></script>
<script>
    window.addEventListener('load', () => { fetchFavorites(false, false); });
</script>
{% endblock script %}
```

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">

    {% block metas %}{% endblock metas %}

    <title>{% block title %}{% endblock title %}</title>

    <link rel="stylesheet" href="{{ url_for('static', filename='css/bootstrap.css') }}">
    <link rel="stylesheet" href="{{ url_for('static', filename='css/style.min.css') }}">
    {% block extra %}{% endblock extra %}
</head>
<body style="overflow-x: hidden">

    <nav class="navbar navbar-expand-lg">
        <div class="container">
            <a class="navbar-brand" href="/">
                <img src="{{ url_for('static', filename='img/logo_small.svg') }}" width="40">
            </a>
            <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarSupportedContent" aria-controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle navigation">
                <span class="navbar-toggler-icon"><img src="{{ url_for('static', filename='img/burger.svg') }}"></span>
            </button>

            <div class="collapse navbar-collapse" id="navbarSupportedContent">
                <ul class="navbar-nav mr-auto">
                    <li class="nav-item active">
                        <a class="nav-link" href="/">Accueil</a>
                    </li>

                    {% if current_user.is_authenticated %}
                        <li class="nav-item">
                            <a class="nav-link" href="/profile/{{ current_user.nickname }}">Profil</a>
                        </li>
```

```
                        <li class="nav-item">
                            <a class="nav-link" href="/lists/{{ current_user.nickname }}">Listes</a>
                        </li>
                    {% endif %}

                    <li class="nav-item active">
                        <a class="nav-link" href="/about">À propos</a>
                    </li>
                    <li class="nav-item active">
                        <a class="nav-link" href="/help">Aide</a>
                    </li>
                </ul>

                <ul class="navbar-nav">
                    {% if current_user.is_authenticated %}
                        <li class="nav-item" style="display: flex">
                            <button type="button" id="synchronize" class="btn mr-5">
                                <img src="{{ url_for('static', filename='img/sync.svg') }}" width="25">
                            </button>
                        </li>
                        <li class="nav-item" style="display: flex">
                            <button type="button" id="pick-random" class="btn mr-5">
                                <img src="{{ url_for('static', filename='img/random.svg') }}" width="25">
                            </button>
                        </li>
                        <li class="nav-item" style="display: flex">
                            <!-- Button trigger modal -->
                            <button type="button" id="search-modal-trigger" class="btn mr-5" data-toggle="modal" data-target="#exampleModal">
                                <img src="{{ url_for('static', filename='img/search.svg') }}" width="25">
                            </button>

                            <!-- Modal -->
                            <div class="modal fade search-modal" id="exampleModal" tabindex="-1" role="dialog" aria-
```

```
labelledby="exampleModalLabel" aria-hidden="true">
                                <div class="modal-dialog" role="document">
                                    <div class="modal-content search-form">
                                        <div><img src="{{ url_for('static', filename='img/search.svg') }}"></div>
                                        <input class="search-string" type="text" name="search-string" placeholder="Rechercher sur Animanga">
                                        <div class="search-clear">
                                            <svg width="21" height="21" fill="none" xmlns="http://www.w3.org/2000/svg"><path d="M17.393 1.834h-14a2 2 0 00-2 2v14a2 2 0 002 2h14a2 2 0 002-2v-14a2 2 0 00-2-2zM7.393 7.834l6 6M13.393 7.834l-6 6" stroke="#fff" stroke-width="2" stroke-linecap="round" stroke-linejoin="round"/></svg>
                                        </div>
                                    </div>
                                </div>
                            </div>
                        </li>
                        <li class="nav-item dropdown">
                            <a class="nav-link dropdown-toggle" href="#" id="navbarDropdown" role="button" data-toggle="dropdown" aria-haspopup="true" aria-expanded="false">
                                <img src="{{ url_for('static', filename='img/profil-icon.svg') }}">
                            </a>
                            <div class="dropdown-menu" aria-labelledby="navbarDropdown">
                                <a class="dropdown-item" href="/logout">Déconnexion</a>
                                <div class="dropdown-divider"></div>
                                <a class="dropdown-item" href="/override" style="padding: 0 1rem">
                                    <span class="danger">Écraser tous les animes</span>
                                </a>
                            </div>
                        </li>
                    {% else %}
                        <li class="nav-item">
                            <a class="nav-link" href="/login">Connexion</a>
```

```
                        </li>
                        <li class="nav-item">
                            <a class="nav-link" href="/signup">Inscription</a>
                        </li>
                    {% endif %}
                </ul>
            </div>
        </div>
    </nav>

    <div>
        {% block content %}{% endblock content %}
    </div>

    <script src="https://code.jquery.com/jquery-3.4.1.min.js" integrity="sha256-CSXorXvZcTkaix6Yvo6HppcZGetbYMGWSFlBw8HfCJo=" crossorigin="anonymous"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.7/umd/popper.min.js" integrity="sha384-UO2eT0CpHqdSJQ6hJty5KVphtPhzWj9WO1clHTMGa3JDZwrnQq4sF86dIHNDz0W1" crossorigin="anonymous"></script>
    <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.min.js" integrity="sha384-JjSmVgyd0p3pXB1rRibZUAYoIIy6OrQ6VrjIEaFf/nJGzIxFDsf4x0xIM+B07jRM" crossorigin="anonymous"></script>
    <script src="{{ url_for('static', filename='js/search-handler.js') }}"></script>
    <script src="{{ url_for('static', filename='js/shortcut.js') }}"></script>
    <script>
        const syncButton = document.querySelector('#synchronize');

        if (syncButton != null) {
            syncButton.addEventListener('click', () => {
                window.location.replace('/sync');
            });
        }
    </script>
    {% block script %}{% endblock script %}
</body>
</html>
```

```html
<div class="profile-header">
    <div class="header">
        <img class="banner" src="{{ url_for('static', filename='img/banner.jpg') }}" alt="">
        <div class="overlay"></div>
        <img class="profile-picture" src="{{ url_for('static', filename='img/pp.png') }}" alt="">
        <span class="profile-nickname">{{ searched_user.nickname }}</span>
    </div>

    <div class="links">
        <span><a href="/profile/{{ searched_user.nickname }}">Vue d'ensemble</a></span>
        <span><a href="/lists/{{ searched_user.nickname }}">Listes</a></span>
        {% if current_user.nickname == searched_user.nickname %}
            <span><a href="/favorites/{{ searched_user.nickname }}">Favoris</a></span>
        {% endif %}
    </div>
</div>
```