



Animanga

Rapport TPI et
documentation
technique

**mai – juin
2020**

Travail Pratique Individuel (TPI)

- Tanguy Cavagna
- Maître d'apprentissage : Pascal Bonvin

Table des matière

- Animanga
 - Table des versions
 - Résumé de l'énoncé
 - Organisation
 - Livrable
 - Matériel et logiques à disposition
 - Descriptif complet du projet
 - Sitemap
 - Description succincte 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
 - Dictionnaire de données
 - Plans de test et tests
 - Périmètre des tests
 - Environnement
 - Scénarios
 - Évolution des tests

Table des versions

N° de version	Date	Auteur	Modifications
0.1	2020-05-25	Tanguy Cavagna < tanguy.cvgn@eduge.ch >	Création de la base de la documentation
0.2	2020-05-26	Tanguy Cavagna < tanguy.cvgn@eduge.ch >	Ajout de la partie <i>implémentation</i> et modification des <i>user stories</i> et <i>tests</i> .

Résumé de l'énoncé

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

Organisation

Élève	Maître d'apprentissage	Experts
Tanguy Cavagna < tanguy.cvgn@eduge.ch >	Pascal Bonvin < edu-bonvinp@eduge.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 logiques à disposition

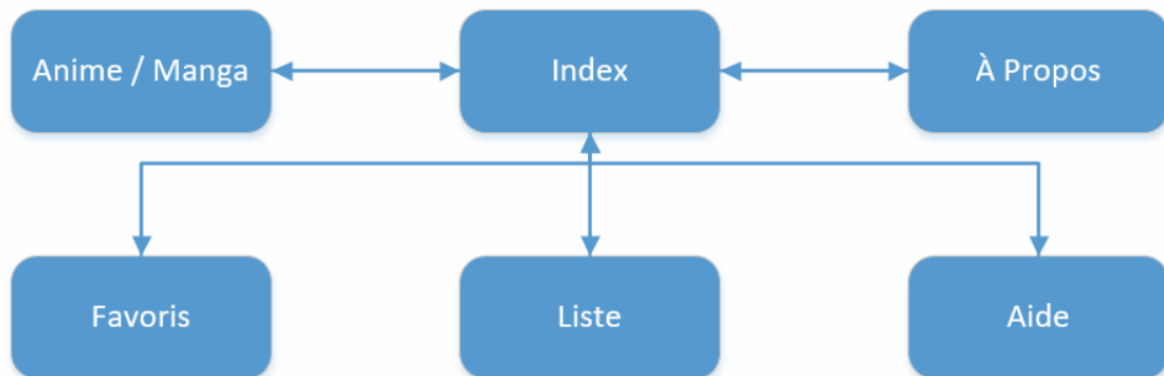
- Un PC standard école, 2 écrans
- Pycharm
- Netbeans
- Suite office
- MySQL, Sqlite3, Flask, connexion internet

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 unidirectionnellement 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

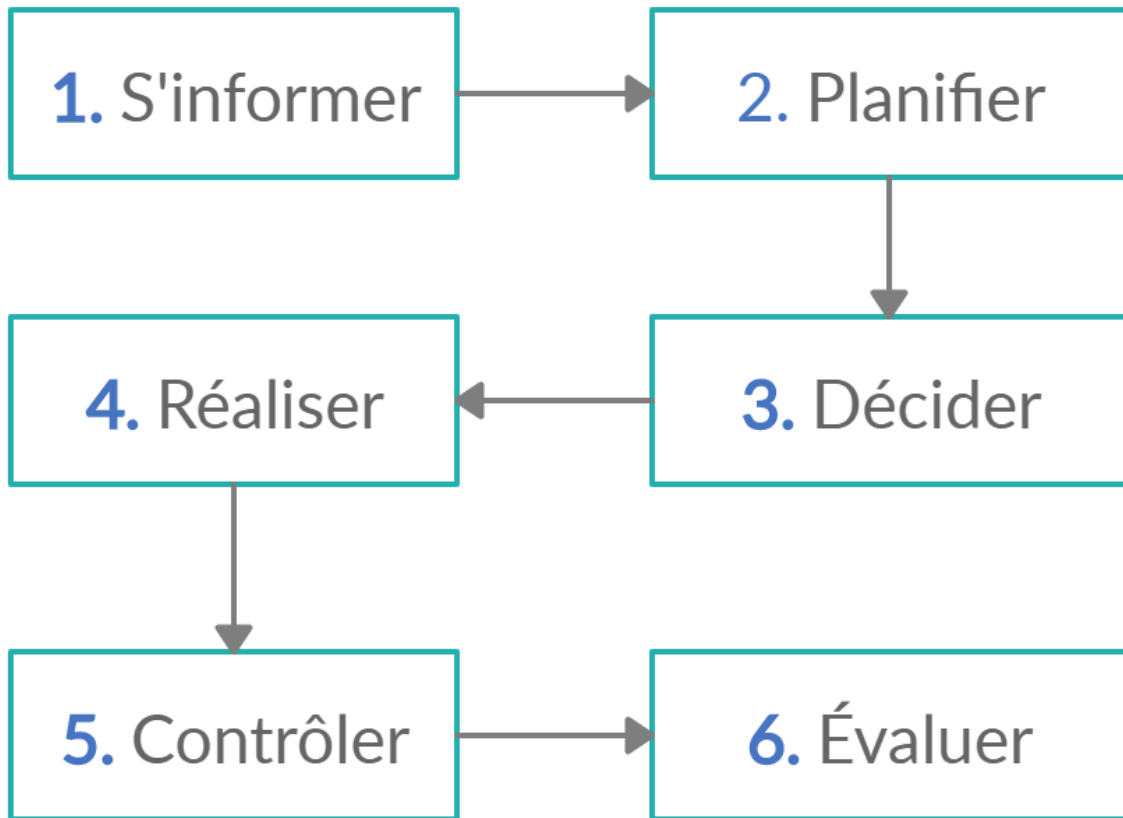


Description succincte du contenu des pages du site

- 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 sélectionné (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 informations 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 mis 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 pour pouvoir comprendre le projet dans son ensemble et comprendre toutes les fonctionnalités nécessaires. Il est aussi indispensable de demander d'éclaircir tous les points flous de l'énoncé.

2. Planifier

Le fait de planifier le projet permet de séparer les tâches et de définir des priorités. ses dernières sont les suivantes : ☹ Bloquant , ☠ Critique , ⚡ Important , 🕒 Secondaire .

Pour représenter le planning nous avons utilisé un diagramme de Gantt. Ce type de diagramme permet de visualiser très correctement la progression quotidienne ainsi que les différences entre les prévisions et le réel.

3. Décider

Cette partie nous permet de pouvoir se lancer dans la réalisation du projet. S'il nous reste des points en suspens, c'est le moment de prendre une décision et de se jeter à l'eau une bonne fois pour toute.

4. Réaliser

Nous pouvons enfin nous lancer dans l'implémentation de toutes les fonctionnalités à développer ainsi que la rédaction de la documentation.

5. Contrôler

Pour valider cette étape, nous avons tester chacune des fonctionnalités indépendamment des autres pour correctement vérifier leur fonctionnement dans différents cas d'usage.

Une fois l'application terminée, nous avons pu tester son bon fonctionnement sur plusieurs navigateurs différents pour bien être sûre que tout fonctionne comme prévu dans n'importe quel cas d'utilisation.

6. Évaluer

Une fois toutes les étapes précédentes achevées, nous avons pu nous lancer dans ce qui peut sembler le plus complexe. Nous avons fait une rétrospective de tout ce que nous avons fait avec un regard critique afin de chercher des points sur lesquels nous pourrions nous améliorer par la suite. Pour ce faire, nous avons une section dédiée dans le journal de bord répertoriant les problèmes rencontrés ainsi que les solutions trouvées pour ces derniers. Une conclusion est aussi présente à la fin de ce rapport servant 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 1.2 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 2.2 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 effectuer une recherche afin d'ajouter des animes dans mes listes personnelles ou de les mettre en tant que favoris .
Critère d'acceptation	<i>Pas encore écrits de tests</i>
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	<i>Pas encore écrits de tests</i>
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	<i>Pas encore écrits de tests</i>
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 pouvoir voir les statistiques et favoris. Il est également possible de voir la page de profile d'autre utilisateur du site.
Critère d'acceptation	<i>Pas encore écrits de tests</i>
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	<i>Pas encore écrits de tests</i>
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	<i>Pas encore écrits de tests</i>
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	<i>Pas encore écrits de tests</i>
Priorité	🚩 Important

Nom	S10 : 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	<i>*Pas encore écrits de tests*</i>
Priorité	🚩 Important

Nom	S11 : Affichage de la <i>landing page</i>
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	<i>Pas encore écrits de tests</i>
Priorité	🔗 Secondaire

Nom	S12 : 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 configuré correctement et le lien sur le dépôt distant a été bien fait.
Priorité	🔒 Bloquant

Nom	S13 : Configuration de la base de données
Description (user story)	En tant que développeur, je dois pouvoir utiliser une base de données SQLite 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 SQLite 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 <code>animes</code> , <code>status</code> , <code>type</code> , <code>url</code> , <code>list</code> , <code>user</code> , <code>list_has_anime</code> , <code>user_has_list</code> et <code>user_has_favorites</code> ont bien été créés et sont utilisables par les contrôleurs dédiés.
Priorité	🔒 Bloquant

Nom	S13 : Synchronisation MySQL SQLite3
Description (user story)	En tant que développeur, je dois pouvoir synchroniser les bases MySQL et SQLite3 unidirectionnellement pour créer un backup sur serveur distant.
Critère d'acceptation	<i>Pas encore écrits de tests</i>
Priorité	🔗 Secondaire

Nom	S15 : 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	<i>*Pas encore écrits de tests*</i>
Priorité	🔒 Bloquant

Nom	S17: Vérifications syntaxique
Description (user story)	En tant que développeur, je peux lancer la commande <code>npm run lint</code> pour vérifier la syntaxe, basé sur le preset Airbnb, des fichiers JavaScript, et la commande <code>python3 -m pylint --output-format=colored</code> pour vérifier la syntaxe des fichiers Python, basé sur les conventions PEP8.
Critère d'acceptation	Les tests 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	Pas encore de tests
Priorité	🔗 Secondaire

Diagramme de Gantt

Jour	J1 lu.25	J2 ma.26	J3 me.27	J4 je.28	J5 ve.29	J7 ma.2	J8 me.3	J9 je.4	J10 ve.5	J11 lu.8	J12 ma.9
Lecture de l'énoncé	<div></div>										
Rédaction backlog	<div></div>										
Rédaction scénarios	<div></div>	<div></div>									
Rédaction planning	<div></div>	<div></div>									
S13 : Utilisation d'un git	<div></div>	<div></div>									
S14 : Configuration de la base de données		<div></div>									
S16 : Configuration de Flask		<div></div>									
S3 : Importation des données		<div></div>	<div></div>								
S1 : Inscription à Animanga			<div></div>								
S2 : Connexion à Animanga			<div></div>	<div></div>							
S12 : Affichage de la landing page		<div></div>		<div></div>							
S4 : Rechercher des animes				<div></div>							

Jour	J1 lu.25	J2 ma.26	J3 me.27	J4 je.28	J5 ve.29	J7 ma.2	J8 me.3	J9 je.4	J10 ve.5	J11 lu.8	J12 ma.9
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 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											

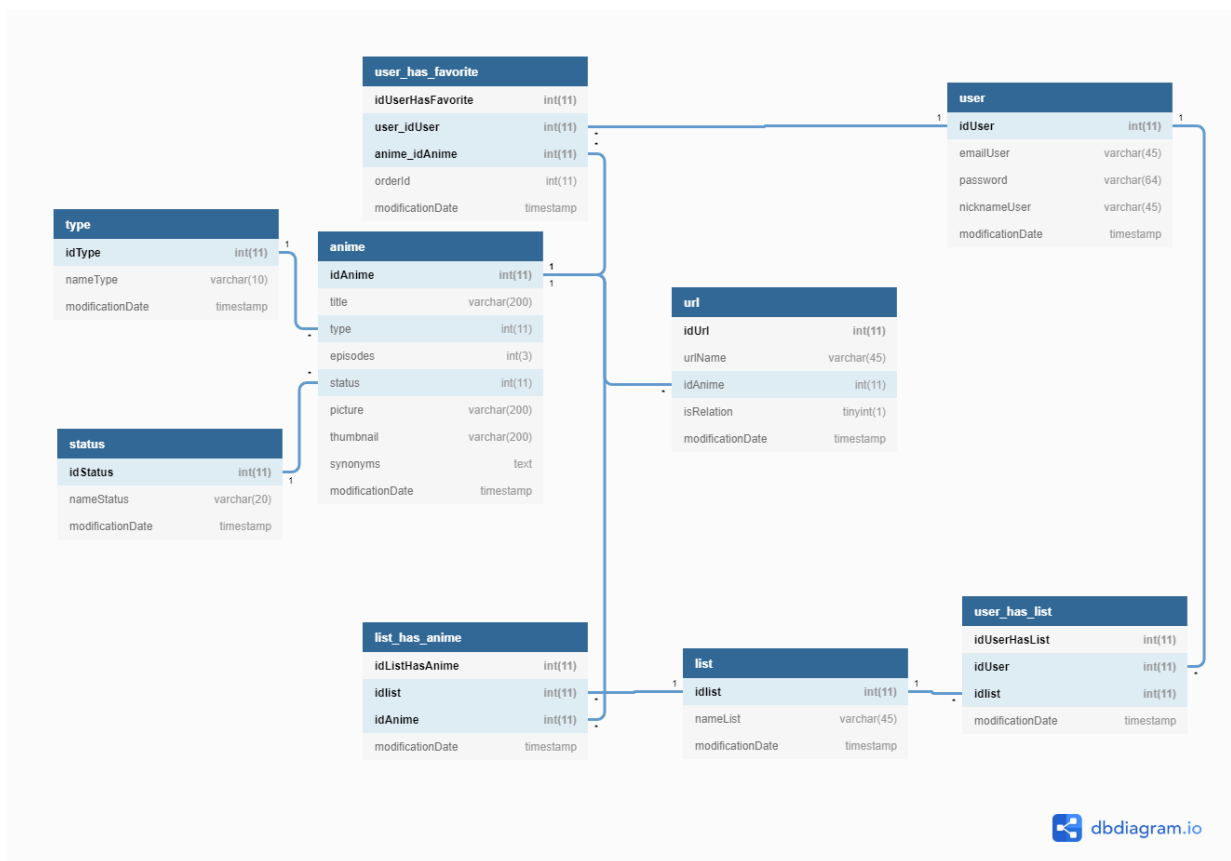
● Planification prévisionnelle
 ● Planification réelle
 ● Planification imprévue

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
<i>#idAnime</i>	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
<i>#idList</i>	int(11)	Non
<i>#idAnime</i>	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
<i>#idUser</i>	int(11)	Non
<i>#idList</i>	int(11)	Non
modificationDate	timestamp	Non

user_has_favorite

Colonne	Type	Null
idUserHasFavorite	int(11)	Non
<i>#idUser</i>	int(11)	Non
<i>#idAnime</i>	int(11)	Non
orderId	int(11)	Non
modificationDate	timestamp	Non

Plans de test et tests

Périmètre des tests

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

Environnement

Lors de ces tests, j'ai utilisé 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 que n'importe quelles personne puisse les exécuter. Pour rédiger mes scénarios j'ai utilisé la syntaxe **Gherkin**.


Nom	1.1 Création d'un nouveau compte
User Story	SI : 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 <i>Inscription</i> , je suis redirigé vers la page d'inscription et je rempli les informations demandées. Alors , je suis redirigé sur la page d'accueil avec mon nouveau compte connecté.
Résultats obtenus	Je clique sur <i>Inscription</i> . Je suis redirigé vers la page d'inscription. Je remplie le formulaire avec des informations valides. Je clique sur <i>Inscription</i> . Je suis redirigé vers la page d'accueil avec mon nouveau compte connecté.
Statut	✓ OK

Nom	1.2 Création d'un nouveau compte (avec erreurs)
User Story	SI : 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 <i>Inscription</i> , je suis redirigé vers la page d'inscription et je rempli le formulaire avec des informations erronées. Alors , le formulaire est rechargé avec un message d'erreur indiquant ce qu'il ne c'est pas bien passé.
Résultats obtenus	Je clique sur <i>Inscription</i> . Je suis redirigé vers la page d'inscription. Je remplie le formulaire avec un email invalide. Je clique sur <i>Inscription</i> . Un message s'affiche m'indiquant que l'email fourni n'est pas correct.
Statut	✓ OK

Nom	2.1 Connexion avec un compte existant
User Story	S2 : Connexion à Animanga
Situation	<p>Étant donné que je suis un utilisateur de Animanga, j'ai déjà un compte à disposition.</p> <p>Quand je clique sur le bouton <i>Connexion</i> , je suis redirigé vers la page de connexion et je rempli les informations demandées.</p> <p>Alors, je suis redirigé sur la page d'accueil avec mon compte connecté.</p>
Résultats obtenus	Je clique sur <i>Connexion</i> . Je suis redirigé vers la page d'inscription. Je remplie le formulaire avec des informations valides. Je clique sur <i>Connexion</i> . Je suis redirigé vers la page d'accueil avec mon nouveau compte connecté.
Statut	✓ OK

Nom	2.2 Déconnexion
User Story	S2 : Connexion à Animanga
Situation	<p>Étant donné que je suis un utilisateur connecté au site.</p> <p>Quand je clique sur le bouton <i>Déconnexion</i> placé dans le dropdown du menu <i>Utilisateur</i> .</p> <p>Alors, je deviens un utilisateur non connecté et je suis redirigé sur la page de connexion.</p>
Résultats obtenus	Je clique sur <i>Utilisateur</i> et <i>Déconnexion</i> . Je ne suis plus connecté et je suis revenue sur la page de connexion.
Statut	✓ OK

Nom	3.1 Importation des animes
User Story	S3 : Importation des animes
Situation	<p>Étant donné que je suis un utilisateur connecté au site.</p> <p>Quand je clique sur le bouton <i>Écraser tous les animes</i> placé dans le dropdown du menu <i>Utilisateur</i> .</p> <p>Alors, j'écrase toutes les données du site relatives aux animes. Cela comprend les animes en eux même, les favoris ainsi que les animes contenu dans les listes.</p>
Résultats obtenus	Je clique sur <i>Utilisateur</i> et <i>Écraser tous les animes</i> . Je suis redirigé vers la page d'accueil et des alertes s'affiche 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	<p>Étant donné que je suis un utilisateur connecté au site.</p> <p>Quand je clique sur le bouton  placé dans la barre de navigation et que j'écris "k On" dans le champs de recherche de la modale.</p> <p>Alors, je suis redirigé vers la page d'accueil et les résultats de la recherche affiche l'anime "K-ON!".</p>
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	<p>Étant donné que je suis un utilisateur connecté au site.</p> <p>Quand je fais le raccourcis clavier <input type="text" value="Ctrl"/> + <input type="text" value="S"/> et que j'écris "k On" dans le champs de recherche de la modale.</p> <p>Alors, je suis redirigé vers la page d'accueil et les résultats de la recherche affiche l'anime "K-ON!".</p>
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	<p>Étant donné que je suis un utilisateur connecté au site sur la page d'accueil ayant fait une recherche.</p> <p>Quand je clique sur le titre d'un anime présent dans la zone de résultat de la recherche.</p> <p>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.</p>
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	<p>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.</p> <p>Quand sélectionne un statut autre que "---".</p> <p>Alors, le combo-box se met à jour avec la nouvelle valeur sélectionnée.</p>
Résultats obtenus	La valeur du combo-box c'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	<p>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.</p> <p>Quand je clique sur une check-box blanche d'une des listes personnelles.</p> <p>Alors, l'état de la check-box ce met à jour et elle se colore en bleu. L'anime est maintenant présent dans la liste personnelle.</p>
Résultats obtenus	L'état de la check-box c'est bien mis à jour et est bien coloré en bleu.
Statut	✗ KO

Nom	6.3 Ajout de l'anime dans les favoris
User Story	S6 : Mise à jour de l'anime
Situation	<p>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.</p> <p>Quand je clique sur le cœur blanc pour ajouté au favoris.</p> <p>Alors, le cœur se colore et l'anime se rajoute dans la zone des favoris de la page d'accueil.</p>
Résultats obtenus	Le cœur c'est coloré et l'anime c'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	<p>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.</p> <p>Quand sélectionne le statut "---".</p> <p>Alors, le combo-box se met à jour avec la nouvelle valeur sélectionnée et l'anime n'est plus présent dans aucun autre statut.</p>
Résultats obtenus	La valeur du combo-box c'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	<p>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.</p> <p>Quand je clique sur une check-box bleue d'une des listes personnelles.</p> <p>Alors, l'état de la check-box ce met à jour et se colore en blanc. L'anime n'est plus présent dans la cette liste personnelle.</p>
Résultats obtenus	L'état de la check-box c'est bien mis à jour et est coloré en blanc.
Statut	✗ KO

Nom	6.6 Suppression de l'anime des favoris
User Story	S6 : Mise à jour de l'anime
Situation	<p>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.</p> <p>Quand je clique sur le cœur rose pour supprimer des favoris.</p> <p>Alors, le cœur se colore en blanc et l'anime se supprime de la zone des favoris de la page d'accueil.</p>
Résultats obtenus	Le cœur c'est coloré en blanc et l'anime c'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 <i>Profile</i> 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	7.2 Affichage du profile d'un autre utilisateur
User Story	S7 : Affichage du profile
Situation	Étant donné que je suis un utilisateur connecté. Quand je modifie l'url pour entré <i>"/profile/Test"</i> . Alors , la page de profile de "Test" s'affiche avec ses statistiques et ses favoris.
Résultats obtenus	La page de profile de "Test" 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 <i>Listes</i> dans la barre de navigation. Alors , la page contenant toutes les listes de l'utilisateur connecté s'affiche ainsi que les animés contenu dans ces listes.
Résultats obtenus	La page contenant les listes de l'utilisateur connecté c'est correctement affiché et les animés sont correctement affiché aussi.
Statut	✗ KO

Nom	8.2 Affichage des listes d'un autre utilisateur
User Story	S8 : Affichage des listes
Situation	Étant donné que je suis un utilisateur connecté. Quand je modifie l'url pour entré <i>"/listes/Test"</i> . Alors , la page contenant toutes les listes de l'utilisateur "Test" s'affiche ainsi que les animés contenu dans ces listes.
Résultats obtenus	La page contenant les listes de l'utilisateur "Test" c'est correctement affiché et les animés sont correctement affiché aussi.
Statut	✗ KO

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 champs de texte <i>Nouvelle liste</i> et que j'appuie sur <input type="button" value="Enter"/> . Alors , la liste apparaîtra 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	✗ KO

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 présentes.
Résultats obtenus	La liste a bien été supprimée et n'est plus présente dans les listes déjà existantes.
Statut	✗ KO

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	✗ KO

Nom	10.2 Affichage des favoris du profil
User Story	S10 : Affichage des favoris
Situation	Étant donné que je suis un utilisateur connecté. Quand je suis sur ma page de profil. Alors , mes favoris sont présents sur la page.
Résultats obtenus	Mes favoris sont bien affichés.
Statut	✗ KO

Nom	10.3 Affichage des favoris d'un autre
User Story	S10 : Affichage des favoris
Situation	Étant donné que je suis un utilisateur connecté. Quand je modifie l'URL pour entrer "/favorites/Test". Alors , les favoris de l'utilisateur "Test" s'affichent dans l'ordre que "Test" a décidé.
Résultats obtenus	Mes favoris sont bien affichés.
Statut	✗ KO

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 <i>Réorganiser les favoris</i> , je peux glisser déposer les animés dans l'ordre que je veux. Je clique sur le bouton <i>Sauvegarder</i> pour enregistrer l'ordre. Alors , mes favoris sont enregistrés dans l'ordre voulu.
Résultats obtenus	Mes favoris ont bien été réorganisés.
Statut	✗ KO

Nom	12.1 Affichage de la landing page
User Story	S12 : Affichage de la landing page
Situation	<p>Étant donné que je suis un utilisateur non connecté.</p> <p>Quand je suis sur le site.</p> <p>Alors, une page d'accueil s'affiche avec comme possibilité : la visite de la page <i>À propos</i>, se connecter et s'inscrire.</p>
Résultats obtenus	La page d'accueil ainsi que la barre de navigation sont affichée correctement pour un utilisateur non connecté.
Statut	✓ OK

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

Nom	13.2 Respect des conventions PEP8
User Story	S17 : Vérification syntaxique
Situation	<p>Étant donné que je suis un développeur.</p> <p>Quand j'exécute la commande <code>python3 -m pylint --output-format=colored packages</code> à la racine de mon projet.</p> <p>Alors, les erreurs <i>Too many return statements</i> n'ont pas besoin d'être prises en compte, tout comme <i>Method could be a function</i> du fichier <code>SqliteController.py</code>.</p>
Résultats obtenus	La note attribuée au code est supérieure à 9.5/10.
Statut	✓ OK

Évolution des tests

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
1.1	×	×	✓	✓	✓	✓	×	×	×	×	×
1.2	×	×	✓	✓	✓	✓	×	×	×	×	×
2.1	×	×	✓	✓	✓	✓	×	×	×	×	×
2.2	×	×	✓	✓	✓	✓	×	×	×	×	×
3.1	×	✓	✓	✓	✓	✓	×	×	×	×	×
4.1	×	×	×	✓	✓	✓	×	×	×	×	×
4.2	×	×	×	✓	✓	✓	×	×	×	×	×
5.1	×	×	×	✓	✓	✓	×	×	×	×	×
6.1	×	×	×	×	✓	✓	×	×	×	×	×
6.2	×	×	×	×	×	✓	×	×	×	×	×
6.3	×	×	×	×	✓	✓	×	×	×	×	×
6.4	×	×	×	×	✓	✓	×	×	×	×	×
6.5	×	×	×	×	×	✓	×	×	×	×	×
6.6	×	×	×	×	✓	✓	×	×	×	×	×
7.1	×	×	×	×	✓	✓	×	×	×	×	×
7.2	×	×	×	×	✓	✓	×	×	×	×	×
8.1	×	×	×	×	×	✓	×	×	×	×	×
8.2	×	×	×	×	×	✓	×	×	×	×	×
9.1	×	×	×	×	×	✓	×	×	×	×	×
9.2	×	×	×	×	×	✓	×	×	×	×	×
10.1	×	×	×	✓	✓	✓	×	×	×	×	×
11.1	×	×	×	×	×	✓	×	×	×	×	×
11.2	×	×	×	×	×	✓	×	×	×	×	×
11.3	×	×	×	×	×	✓	×	×	×	×	×
12.1	×	✓	✓	✓	✓	✓	×	×	×	×	×
13.1	×	✓	✓	✓	✓	✓	×	×	×	×	×
13.2	×	✓	✓	✓	✓	✓	×	×	×	×	×



Annexes



Résumé TPI



Énoncé



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 1^{er} 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

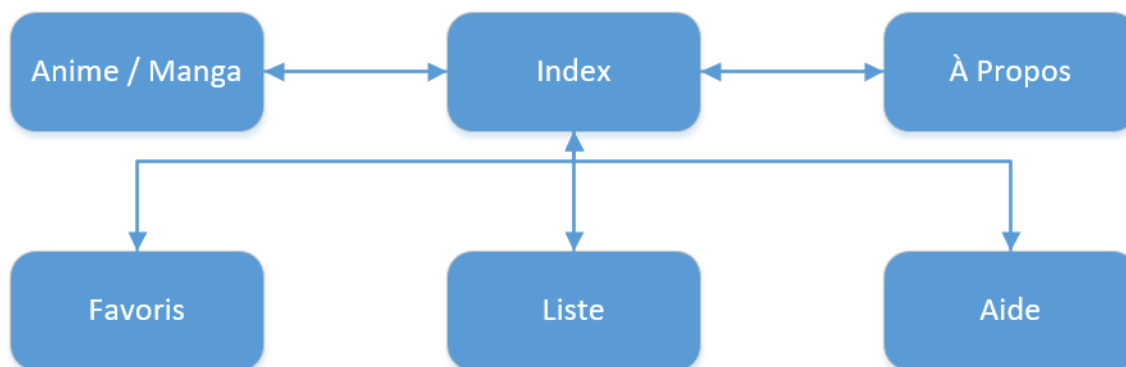
Les notions spécifiques de ce projet ont été/seront étudiées en ateliers application et atelier TPI par le formateur et l'élève.

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

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, extraire les *user stories* de ce dernier pour pouvoir correctement rédiger mon *product backlog* et enfin rédiger les scénarios de tests fonctionnels, indispensable pour le bon fonctionnement de mon projet.

Déroulement

Je commence ma journée à 8h00. M. Terrond m'a fait parvenir mon énoncé la veille, que j'ai lu avec attention ce dernier. Par ce biais, j'ai complété avec succès la première étape de la **méthodologie en 6 étapes**, méthodologie que je vais utiliser durant tout le déroulement de ce TPI : **S'informer**.

J'ai quelques points incertains concernant mon énoncé dont un quelque peu embêtant. Je poserai mes questions à mon formateur durant la matinée. Je vais maintenant commencer à **Planifier**, seconde étape de la méthodologie utilisée. Je séparerai ma journée en tranches de 4 heures, soit par demi-journée, et remplirai des différentes tranches horaires avec les *user stories* extraites de mon cahier des charges.

8h15 : J'ai décidé d'utiliser des alias afin de nommer les jours de travail mis à disposition pour le TPI. Les jours seront nommer de **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* j'ai remarqué qu'il me fallait décider d'une manière de prioriser les tâches. J'ai opté pour me baser sur la méthode **MoSCoW**. Cependant les niveaux de priorité ne correspondaient pas entièrement pour un TPI. J'ai alors décidé de modifier les intitulés :

- **Must** devient ☹ **Bloquant**
- **Should** devient ⚠ **Critique**
- **Could** devient 📌 **Important**
- **Won't** devient ? **Secondaire**


J'ai aussi décidé d'utiliser la syntaxe suivante afin 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 : J'ai fait un script bash me permettant un rassembler tout 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** pour exporter mon fichier réunissant toute ma documentation en PDF. Une fois cela fait, j'utilise un autre script bash que j'ai réalisé 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 : Descriptif de 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'ai utilisé la couleur  **#006EDB** comme principale. La police est Poppins, aussi utilisée dans le projet en lui même.

10h50 : J'ai eu un rendez-vous GMeet avec mon formateur pour vérifier que tout allait bien. J'ai posé la question suivante et voici la réponse donnée :

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

 Le planning que j'ai donné est un modèle permettant de suivre de façon basique l'avancée du projet. Si vous avez un planning plus précis, vous pouvez sans autre l'utiliser et comparer ensuite le vôtre avec celui que j'ai donné.

11h25 : J'ai terminé la rédaction de mon *product backlog* temporaire. Des modifications peuvent encore être apportés si j'en trouve le besoin.

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

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

14h15 : J'ai remarqué un soucis lors de la fusion des fichiers Markdown. Une partie d'un fichier se dédouble mais je ne sais pas encore pourquoi.

15h50 : Mon soucis 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 vais pouvoir commencer l'écriture des scénarios de tests fonctionnels.

16h45 : J'ai écrit une partie des scénarios de tests. Il m'en reste encore quelques un que j'ajouterais 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 que très minime sur mon planning. Je sort tout de mais satisfait de cette première journée.

J2 : mardi 26 mai 2020

Objectifs

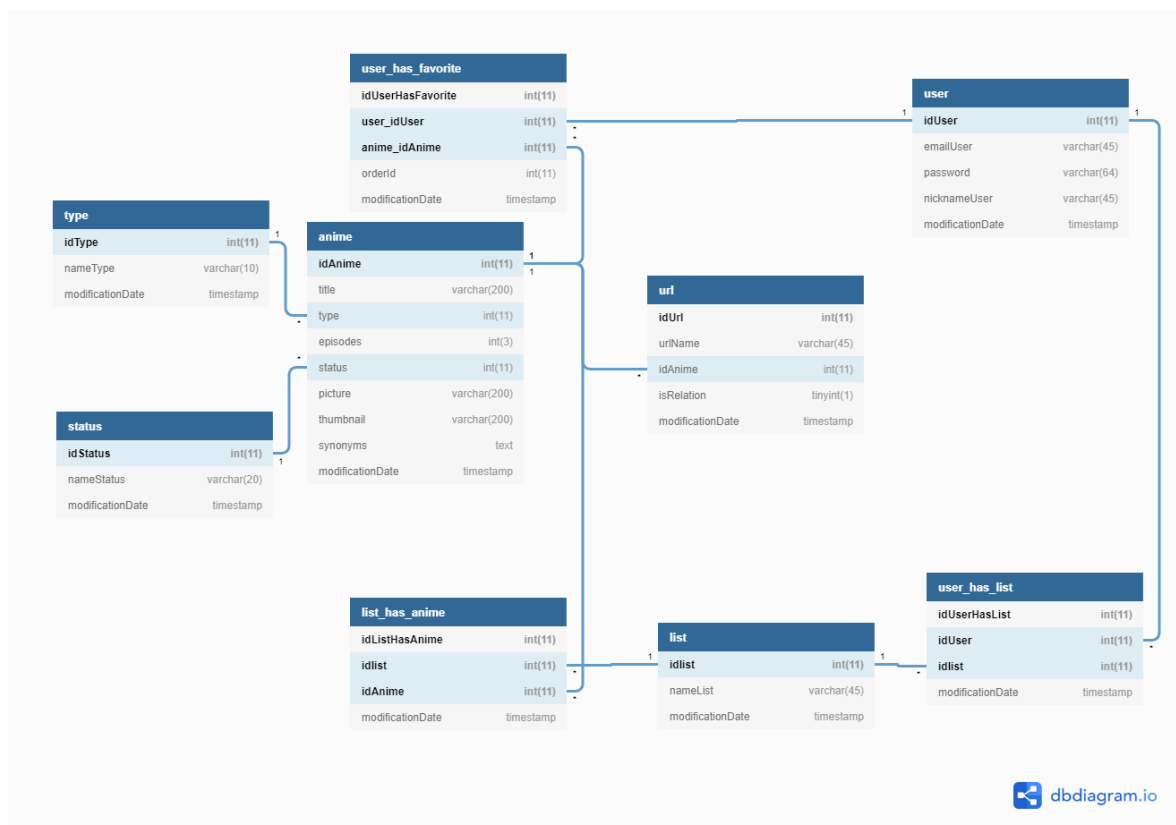
L'objectif de cette journée est premièrement de rattraper le peu de retard que j'ai eu hier sur les scénarios de tests. Ensuite, je ferai le modèle de base de données, je configurerai l'application Flask, et je ferai le code permettant d'importer les données.

Déroulement

8h : Je commence ma journée. Je dois finir les scénarios de tests que je n'avais pas pu terminer hier. Ceci ne devrait pas me prendre beaucoup de temps.

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

8h55 : J'ai réalisé le modèle de base de données que voici :



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

9h25 : J'ai terminé 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 ainsi pouvoir faire la suite du projet.

J'ai inscrit une `secret_key` à l'application Flask. Cette clef est utilisé dans les systèmes d'encryptions. Flask lui-même n'a pas besoin de cette clef mais d'autre 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 : J'ai mis en place le système de documentation automatique d'API : **Swagger**.

Pour que ce système puisse être mis en place, il faut une librairie externe nommé `flask_swagger` . De plus, certains fichiers sont indispensable au bon fonctionnement de Swagger. Le plus primordiale est la page HTML. J'ai décidé de la nommée `endpoints.html` et de la placé 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, il nous faut rajouter 2 fichiers `javascript` qui iront dans le dossier `static/js` , 2 images qui iront dans le `static/img` et enfin 1 fichier `css` qui ira 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 :



10h25 : J'ai installer 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 maintenant l'importation des données en base.

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

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

14h : J'ai terminé l'import des données et tout semble parfaitement bien s'ajouter en base. Étant donné que j'ai un peu d'avance, je vais mettre 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'ai installé mkdocs dans **WSL 2** sur ma machine pour vérifier si ma documentation compilait 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'ai effectué le test *3.1* pour vérifier que mes données soient correctement importées. Comme j'ai de l'avance de vais faire l'affichage de la *landing page* .

15h20 : J'ai terminé 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 changé puisque je met 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 vais configurer deux vérificateurs de syntaxe différent pour mon projet. Le premier est **pylint** pour Python, et le second est **eslint** pour le JavaScript. Pour eslint, je vais utilisé un preset de vérification : `airbnb`. Cela me permet d'écrire mes script 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 : J'ai corrigé les erreurs que eslint m'avait montré et je m'attaque maintenant à pylint.

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

16h10 : J'ai corrigé les erreurs relevées par pylint. Je vais mettre à jour mon planning et mes scénarios de tests afin d'ajouter la vérification syntaxique.

16h30 : J'ai ajouté la vérification syntaxique dans le planning comme *user story* et j'ai créer 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 très vite rattrapé ce matin. J'ai réussi à prendre un peu d'avance dans le projet et donc je suis très confiant pour la suite. Cela m'a permis de rajouté 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 peut être avancer sur une autre tâche.

Déroulement

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

8h30 : J'ai fait le formulaire HTML et je commence à faire la partie prise en charges de ce dernier.

9h : J'ai eu la visite de M. Terrond. M. Bouille étant pris à la Protection Civile, il n'a pas pu être présent. Le but de cette visite était de voir si tout ce passe bien et de répondre au possible questions. Étant donné que tout ce passe bien pour moi, la visite n'a duré que très peu de temps.

9h40 : J'ai des soucis avec mon système d'export de documentation vers PDF. L'export ne s'effectue pas mais aucun moyen de savoir quelle est la cause. Je cherche activement ce qui pourrait poser problème.

10h20 : J'ai réussi à régler le problème d'export pour le moment mais rien ne me dit que cela ne réarrivera pas. Si cela doit être le cas, j'ai un moyen auxiliaire d'exporter ma documentation. Je me remet à travailler sur la partie inscription.

11h45 : J'ai terminé l'inscription. L'insère de nouvel utilisateur semble fonctionner. Avec les quelques minutes qu'il me reste avant la pause de midi, je met en place des automatisations Katalon Recorder pour les futurs tests.

12h : J'ai terminé l'automation du test fonctionnel d'inscription avec valeurs correctes. Je ferai les autres dès que j'aurai du temps aujourd'hui. Je prend ma pause de midi.

13h : Je me remet 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'autre retards car j'avais pris de l'avance hier concernant la *landing page* que j'aurais du réaliser aujourd'hui.

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

14h20 : Je vais maintenant pouvoir optimiser 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 soucis d'erreur d'export de la documentation.

15h20 : J'ai eu un rendez-vous avec mon référent TPI pour faire un point. Comme tout ce passe bien le rendez-vous n'a duré que très peu de temps et je suis tout de suite retourné travailler.

17h : J'ai terminé l'optimisation de la validation des champs. Je me suis basé sur la librairie **wtforms**. Cette librairie est utilisé pour générer et valider automatiquement des formulaires. Comme cette librairie est trop imposant pour un TPI, j'ai décidé 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, j'ai configuré Katalon Recorder pour prendre en compte la connexion aussi. J'ai maintenant un dossier à la racine de mon projet nommé `tests` qui contient le fichier HTML contenant tout 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 et 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

Déroulement

8h : J'ai réfléchi hier soir et j'ai réalisé que ce que j'avais entrepris la veille sur l'optimisation de la validation des champs était beaucoup trop imposant. Je me décide à le refaire de manière bien plus légère.

9h20 : J'ai terminé ma nouvelle version de l'optimisation des champs. Bien plus simple à lire et à maintenir. Très content du résultat. Je commence maintenant la partie recherche d'anime. 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 Fulltext 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 chaîne de caractères est terminée. Il me reste à faire l'affichage mais ceci ne sera pas long. Je le ferai cet après-midi car maintenant je vais faire la récupération d'un anime aléatoire.

10h20 : J'ai eu un rendez-vous GMeet avec mon référent pour voir mon avancement et pour lui poser une question concernant la validation des champs. J'ai demandé si ce que j'avais fait était bien et si je pouvais le garder. Sa réponse a été positive et j'ai pu continuer mon projet.

10h40 : La récupération d'un anime aléatoirement est terminée. J'ai remarqué en programmant que j'ai des **try/except** dans pratiquement toutes les méthodes de classes. Le **except** est toujours le même mais rien n'est centralisé. J'ai alors décidé de commencer à faire une fonction centralisant tout les logs.

11h35 : La fonction de log est terminée et j'ai placé dans tout les **except** un appel à cette méthode.

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

12h : J'ai terminé l'affichage des résultats de la recherche et je prend ma pause de midi.

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

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

14h30 : J'ai terminé d'afficher la carte (modal) de l'anime. Je vais commencer la mise à jour de l'anime pour l'utilisateur connecté.

17h : Je n'ai pas terminé la mise à jour de l'anime mais comme je dois le faire demain normalement ce n'est pas un soucis.

Bilan

Cette journée était très sympa. Je n'ai pas eu de problèmes et j'ai pris un peu d'avance. Le journal n'est pas très rempli pour cet après-midi mais tout ce qu'il y a à savoir sur ma journée y est. Rien de spécial ne s'est passé. Je suis très content d'avoir mis en place les automatisations 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, afficher sa page de profil, et afficher ses favoris. Je pense que je vais 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. Je ne pense pas que cela va me prendre toute la matinée comme prévue et c'est donc aussi pour ceci que je vais faire la partie affichage des favoris en même temps.

9h30 : J'ai terminé la partie mise à jour de l'état de favoris pour les animes. Je vais maintenant les afficher pour pouvoir avoir un retour visuel et aussi pour avoir un élément à tester en case de réussite lors des tests d'automations.

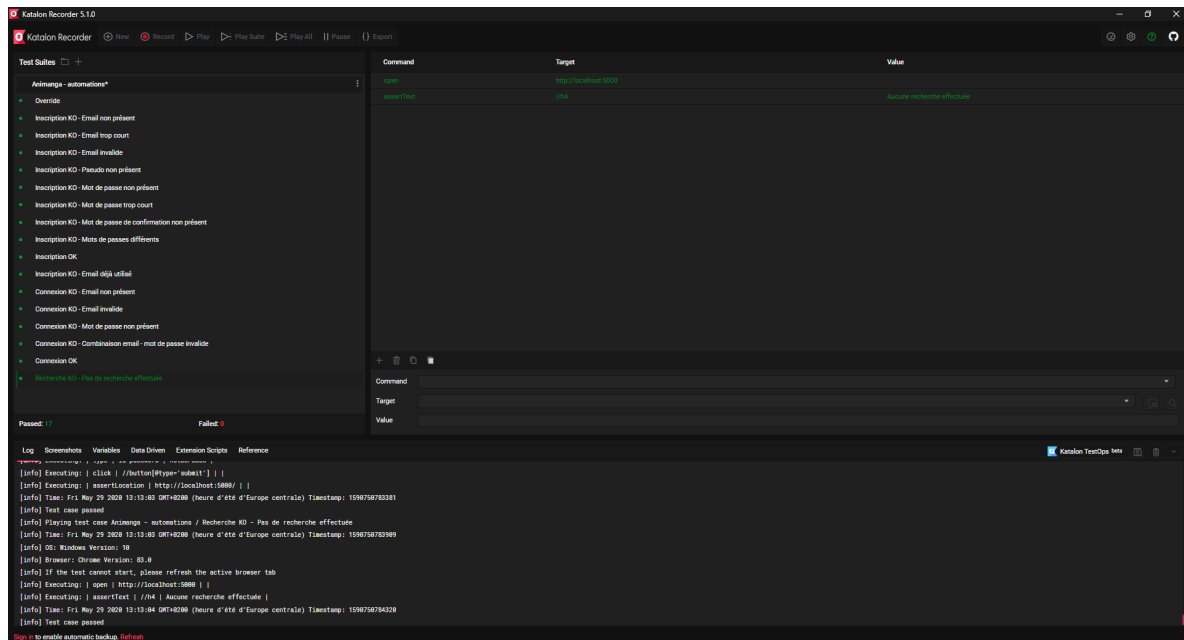
9h55 : J'ai terminé l'affichage des favoris sur la page d'accueil. L'affichage spécifique des favoris est sur la page de profil donc pour le moment il n'y a que la page d'accueil pour voir ses propres favoris. La gestion de ces derniers ne viendra que plus tard. Je vais maintenant commencer la mise à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

10h15 : J'ai eu un rendez-vous GMeet avec mon référent pour vérifier mon avancement. Je lui ai montré ce que j'ai fait les jours précédents et ce que je suis en train de faire. La seule remarque était sur 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 : J'ai terminé la mise à jour du statut de visionnement des animes pour l'utilisateur connecté. Je dois encore affiché quel statut l'anime a quand on clique sur son nom pour affiché sa carte. Je pense le faire maintenant comme cela je ne prend pas de retard.

12h : J'ai terminé l'affichage du statut correct dans le combo box de la carte de l'anime. Il me reste seulement à affiché les listes personnelles à l'utilisateur mais je ne ferai cela que lorsque la tâche arrivera. Je prend ma pause de midi.

13h : Suite au rendez-vous GMeet, je me suis souvenu 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 tout les tests créer depuis le début et leur état :



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

13h40 : J'ai terminé l'affichage de la page de profil de l'utilisateur. En plus de seulement pouvoir affiché la page de profile de l'utilisateur en allant sur l'url `/profile/<pseudo>`, j'ai fait en sort de redirigé l'utilisateur sur `/profile/<pseudo utilisateur connecté>` s'il va sur `/profile` ou `/profile/`. J'ai mis les liens pour le future pages d'affichage des listes ainsi que pour la gestion des favoris. Pour le moment les liens redirige vers une 404 mais je devrais changé cela sous peu. De plus, le lien `favoris` ne s'affiche que si l'on est sur sa propre page de profil. Je vais maintenant ajouté le contenu de la page profile, à savoir les statistiques de l'utilisateurs ainsi que ses favoris.

14h : J'ai ajouté les favoris sur la page de profil. Ils sont affiché différemment que sur la page d'accueil car c'est bien plus ergonomique horizontalement que verticalement sur une page telle que la page de profil. Je vais maintenant affiché les statistiques de l'utilisateur.

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

```
1  {  
2      'TV': 2,  
3      'Movie': 1,  
4      'Ona': 0,  
5      'Special': 1  
6  }
```

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 n'est présent pour le moment.

J6 : mardi 2 juin 2020

Objectif

Cette journée est relativement remplie. Premièrement, je dois affiché le contenu des listes d'un utilisateur. Secondement, il faut que je fasse la gestion des listes. Cela comprend l'ajout de nouvelle liste, la suppression des listes existantes et le renommage des listes existantes. Enfin, il va falloir que je fasse la gestion de l'ordre des favoris.

Déroulement

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

9h : J'ai remarqué que le javascript ordonne automatiquement les listes lorsqu'il les reçoit via un fetch. J'ai donc dû mettre l'id de la liste devant le nom pour que l'ordre soit fait sur l'id et non sur le nom de la liste. Cela permet de garder l'ordre de création des listes.

9h30 : J'ai terminé l'affichage des listes. Dès que l'on clique sur le nom d'une liste, le site nous montre les animes contenu dans la liste cliquée. Par défaut, c'est **Tous** qui est sélectionné, ce qui permet de voir tout les animes de toutes les listes de l'utilisateur.

Je commence maintenant maintenant la gestion des listes.

10h : J'ai terminé l'ajout de nouvelles listes. Cependant, je dois changé la manière de récupération des listes car l'ordre d'affichage est faux.

10h30 : J'ai eu un rendez-vous GMeet avec mon référant pour faire le point. Je lui ai expliqué sur quoi je travaillais et que je n'avais pas de soucis. Le rendez-vous a duré que très peu de temps.

11h : J'ai résolu le soucis d'ordre d'affichage des listes. Ma route `/get/animes` renvoie le json suivant:

```
1  {
2      'Complétés': [
3          animes ...
4      ],
5      'En cours': [
6          animes ...
7      ],
8      listes... : [
9          animes ...
10     ]
11 }
```

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

Je commence maintenant la suppression des listes et de leur contenu.

11h40 : J'ai terminé la suppression des listes et de leur contenu. Il ne me reste qu'à faire le renommage. Je commence cela maintenant.

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 recommencerai cet après-midi.

13h : Je recommence la journée et je vais terminer la mise à jour du nom d'une liste.


13h10 : J'ai terminé la mise à jour du nom d'une liste et je passe maintenant à la mise à jour de l'ordre des favoris.

14h : La mise à jour de l'ordre des favoris est terminée et je vais maintenant faire 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'ai utilisé 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 oublié lors de la rédaction du planning. Je l'ai cependant rajouté dans le product backlog. C'est pourquoi il n'y a pas de case orange dans le planning prévisionnel et seulement une case rouge.

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

16h : J'avais fait au début de la session de TPI la méthode qui permet de récupérer un anime aléatoire. J'ai maintenant terminé de l'afficher sur le site via un bouton placé dans la barre de navigation.



Code source

<https://flask.tcsandbox.ch>