

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 basé 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'ajouterai demain matin.

Bilan

La journées 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

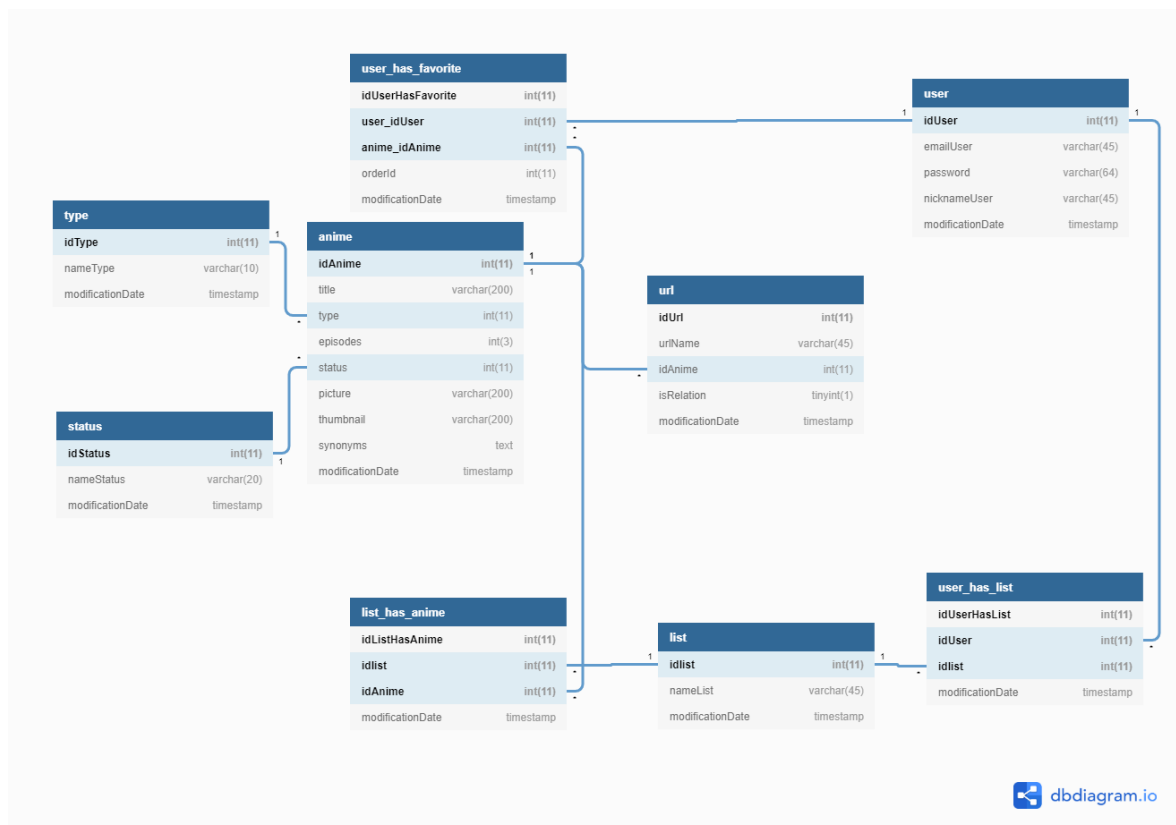
L'objectif de cette journée est premièrement de rattraper le peu de retard que j'ai eu hier sur les scénarios de tests. Ensuite, je ferai le modèle de base de données, je configurerai l'application Flask, et je ferai le code permettant d'importer les données.

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.

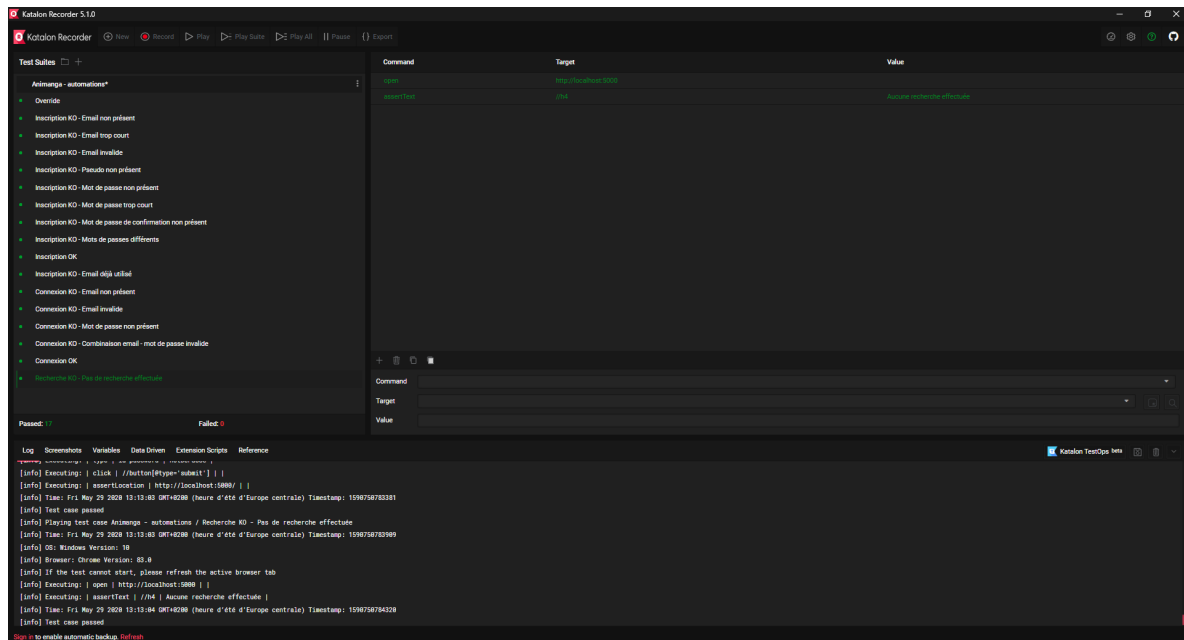
9h55 : J'ai terminé l'affichage des favoris sur la page d'accueil. L'affichage spécifique des favoris est sur la page de profil donc pour le moment il n'y a que la page d'accueil pour voir ses propres favoris. La gestion de ces derniers ne viendra que plus tard. Je vais maintenant commencer la mise à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

16h45 : J'ai terminé l'affichage de l'anime tiré aléatoirement. Je termine ma journée maintenant.

Bilan

Cette journée était plutôt bien remplie. J'ai réussi à faire plusieurs tâches indispensables et toutes fonctionnent. Il ne me reste que la partie synchronisation à faire au niveau technique de l'application et je n'aurai que la partie de documentation, d'aide, et de page à propos à terminer. Je suis content du travail fourni aujourd'hui.

Objectif

Le but de cette journée est de commencer une des parties principale 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 librairie 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 eu un rendez-vous avec mes experts (M. Terrond et M. Bouille) afin de faire le point sur mon avancement dans le travail. Je leur ai expliqué le cas de la tâche manquante dans le planning prévisionnel, et je leur ai montré le planning dans son ensemble pour qu'ils aient une idée global de l'avancement. Alors qu'ils voulaient que je leur fasse une démonstration, j'ai pu décelé 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.

En plus de cela, M. Bouille m'a donner le conseil de faire une slide spécifique dans ma présentation pour toutes les fonctionnalités qui pourraient être rajouté ou amélioré.

9h10 : Le rendez-vous c'est terminé et je vais maintenant corrigé les bugs trouvé lors de la démonstration.

9h30 : les bugs sont corrigé. Rien de grave, simplement une variable mal initialisé ainsi qu'un id HTML pas exclu lors d'un test javascript. Je vais maintenant commencé à réfléchir à l'algorithme que je pourrai utilisé lors de la synchronisation.

10h30 : J'ai fais une première version d'algorithme. Elle n'est de loin pas optimisé et donc je continue à chercher. Voici la première version :

```
1  foreach table {
2      stocker le nom de la table courante
3      récupérer les enregistrements et les stocker
4
5      découper les données en parties de 5k enregistrements
6      // [
7      //   [
8      //     {données}
9      //     ... x5k
10     //   ]
11     //   ... autant de tableau de 5k données pour avoir toutes les données
12     // ]
13
14     foreach paquet in tout les paquets découpé {
15         insérer les données dans la table courante MySQL via un insert
multiple
16         // INSERT INTO table( ... colonnes) VALUES( ... values), ( ... values), ...
17     }
18 }
```

Je vais maintenant essayer d'améliorer mon algorithme pour le rendre moins couteux en ressources et plus rapide.

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

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

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

```
1  foreach table {
2      stocker le nom de la table courante
3
4      récupérer le nombre d'enregistrements de Sqlite3 et le stocker
5      récupérer le nombre d'enregistrements de MySQL et le stocker
6
7      if nombre enregistrements Sqlite3 égale nombre enregistrements MySQL {
8          passé à la table suivante
9      } else {
10         vider la table MySQL
11
12         découper les données en parties de 5k enregistrements
13
14         foreach paquet in tout les paquets découpé {
15             insérer les données dans la table courante MySQL via un insert
multiple
16         }
17     }
18 }
```

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

16h : Je pense avoir une version presque final de l'algorithme. J'ai décidé 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 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 :

```
1  foreach table {
2      stocker le nom de la table courante
3
4      récupérer le nombre d'enregistrements de Sqlite3 et le stocker
5      récupérer le nombre d'enregistrements de MySQL et le stocker
6
7      if nombre enregistrements Sqlite3 différent de nombre enregistrements
MySQL {
8          récupérer ids et timestamp Sqlite3 pour stocker dans tableau id:
timestamp
9          récupérer ids et timestamp MySQL pour stocker dans tableau id:
timestamp
10
11         trier les tableaux par id
12
13         if tableau Sqlite3 > tableau MySQL {
14             stocker ids Sqlite3 non présent dans tableau MySQL
```

```

15         retirer les ids ci-dessus du tableau Sqlite3
16
17         récupérer les enregistrements Sqlite3 correspondant aux ids non
présent MySQL
18
19         foreach enregistrement Sqlite3 {
20             insérer dans MySQL
21         }
22     } else {
23         stocker ids MySQL non présent dans Sqlite3
24
25         foreach id MySQL {
26             supprimer l'entrée MySQL
27         }
28     }
29 }
30
31 déclarer tableau pour ids Sqlite3 dont timestamp diffère du MySQL
32 foreach enregistrements Sqlite3 {
33     if timestamp courant différent du timestamp MySQL correspondant {
34         ajouter l'id dans le tablea créer à cet effet
35     }
36 }
37
38 récupérer les enregistrements Sqlite3 correspondant aux ids
39 foreach enregistrement {
40     mettre à jour l'enregistrement correspondant dans MySQL
41 }
42 }

```

Je commence maintenant à implémenter cet algorithme dans l'application.

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

Bilan

Cette journée n'a pas eu beaucoup de diversité. Je n'ai fait que développé mon algorithme et j'ai à peine commencé à l'implémenté dans l'application. Je terminerai l'implémentation demain je pense. Cela me fera prendre de l'avance de 2 jours environ sur le planning prévisionnel.

J8 : jeudi 04 juin 2020

Objectif

Aujourd'hui je prévois de terminé l'implémentation de l'algorithme de synchronisation. Cela me fera prendre 2 jours d'avance sur mon planning prévisionnel.

Déroulement

8h10 : Je commence ma journée. Aujourd'hui je vais implémenter la synchronisation entre les bases de données.

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

10h30 : J'ai un élève (Vincent Steinmann) qui m'a demandé de l'aide pour son projet. Comme j'ai de l'avance, j'ai pris l'initiative d'aller l'aider pendant un moment.

11h30 : J'ai terminé d'aidé 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 prend ma pause de midi.

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

14h : J'ai remarqué, 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 met les nouveau 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

J'ai donc modifier le code de l'ajout des timestamp dans Sqlite3 pour retiré les millisecondes à la source directement. J'ai remplacer tout les `dt.now()` par `dt.now().strftime('%Y-%m-%d %H:%M:%S')`.

16h : J'ai terminé d'implémenté la synchronisation entre les bases. J'ai pris 2 jours d'avances grâce à cela. Je vais en profiter pour corriger tout les potentiels bugs qui pourrais y avoir et faire de la documentation. De plus, ça me permettra de vérifier que je n'ai rien oublié des points mentionné dans le cahier des charges.

Je met maintenant à jour la documentation.

17h : Je termine ma journée.

Bilan

Cette journée était très fructueuse. J'ai pu finir la synchronisation et donc j'ai 2 jours 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 oublier et corriger les potentiels bugs qui pourraient être présent.