***Justification***

**Log In et Sign Up**

Pour la connexion et l’enregistrement nous n’avons pas vraiment eu de difficultés si ce n’est un réarrangement total de la mise en page suite à un désaccord. Nous avons donc simplement fait 2 pages différentes liées entre elles par des boutons et gérer pas mal de cas pour la connexion et l’enregistrement. Pour l’itération 2 on a rajouté la gestion de conflits de connexions.

**Project menu**

Lors de l’itération 1 nous avons eu quelques difficultés pour intégrer la gestion de projet, surtout avec le format à utiliser, on a fini par choisir juste un tableau contenant les informations afin de nous simplifier la tâche. Une refonte graphique avait aussi dû être effectuée pour cette partie ce qui nous a ralenti dans notre avancement. Pour l’itération 2 on a du ré effectuer une refonte graphique et avons plus ou moins oublié l’idée du tableau, on l’a gardé mais seulement pour le nom des projets, ensuite les autres informations s’affichent sur l’écran.

**Tags**

Pour la représentation des tags nous avons choisi un texte ainsi qu'une couleur afin de les différencier plus facilement. Le programme dispose de tags prédéfinis et les utilisateurs peuvent en rajouter.

**Collaborators**

Pour l'ajout de collaborateurs a un projet, l'utilisateur précise un username et une invitation est envoyée a l'utilisateur correspondant si il existe, celui-ci recevra une notification a sa prochaine utilisation du programme ou lors de l'acces au menu principal, il y verra le nom du projet, sa description ainsi que l'username de l'utilisateur ayant envoyé l'invitation. Il peut alors l'accepter ou la refuser.

**Tasks**

Au départ nous avions utilisé un CheckComboBox afin de rajouter plusieurs collaborateurs a une tache, nous avons cependant été confronté a des problemes d'implémentation innés au module ControlFX qui nous a obligé a re-designer l'interface.

**Statistics**

En ce qui concerne la partie sur les statistiques nous avons rencontré pas mal de problèmes durant **l’itération 1**. Nous n’avons pas pu finir l’histoire, nous n’avions qu’un tableau reprenant le nom des projets car nous n’avions pas eu le temps de tout implémenter. Nous avons choisi cette mise en page aussi pour une question de simplicité même si ce n’est sûrement pas la plus adéquate, mais avec le temps dont nous disposons nous nous sommes dit qu’il en était mieux ainsi.

Pour l**’itération 2** nous avons réussi à rajouter les autres informations dans le tableau, ainsi que l’exportation des statistiques. Les formats choisis pour exporter les statistiques et donc pour les visualiser plus librement directement sur l’ordinateur sont « json » et « csv ».

L’implémentation de l’exportation sous format « .json » était beaucoup plus fastidieuse due à son format bien particulier. Mais une fois fait, nous apprécions ces deux formats car ils reprennent toutes les statistiques de manière simplifiée.

Malgré avoir « fini » le design des statistiques (il restera encore des détails et des précisions à rajouter), nous n’en sommes toujours pas convaincus.

Pour **l’itération 3**, nous nous sommes plus focalisés sur l’interface sans spécialement de fonctionnalités en plus. De ce fait, il est maintenant possible pour l’utilisateur de visualiser ses statistiques de manière individuelle ou globale.

La première consiste à choisir un projet afin d’en voir les statistiques spécifiques à lui (les nombres de collaborateurs, de tâches, de sous-projets et les dates de début, estimée et de fin). A noter que le nombre de collaborateurs compte le créateur du projet pour une question de cohérence. Nous estimions qu’il était plus pertinent de le compter. Cela dit, cette visualisation reste simple et efficace, elle donne les informations nécessaires pour situer l’utilisateur dans l’avancement du projet sélectionné à partir du tableau en arborescence sur la page.

Ensuite, nous avons opté pour une vue générale qui reprend tous les projets de l’utilisateur connecté avec deux graphiques. L’un d’eux contient un diagramme sous forme de « Pie Chart » afin de représenter le temps accordé (date de début – date de fin) pour chaque projet. Ainsi, l’utilisateur peut voir quel projet prend le plus de temps et ceux qui en prennent le moins pour fixer ses priorités. L’autre graphique reprend le nombre de tâches par projet sous forme de « Bar Chart » afin de voir facilement les projets qui ont plus ou moins de tâches à encore réaliser.

Donc, pour justifier tout ce nouveau design, nous insistons sur le fait qu’il a l’avantage d’être fort basé sur le visuel afin que tout utilisateur puisse facilement comprendre les statistiques.

**Refactoring**

Au début de l’itération 2 nous avons dû retravailler notre code afin de rentrer dans certaines normes de Java, nous nous sommes principalement penchés sur le MVC étant donné qu’il était absent de notre première itération. Afin de le respecter, le code a été séparé en différent package, contrôleur, contrôleur de vue,… Qui nous ont posé pas mal de problèmes pour gérer la liaison entre un contrôleur de vue et son contrôleur, surtout avec « ProjectsViewController » et « ProjectController ». Nous avons réglé les problemes en donnant une référence statique vers le controller dans le viewController.

**Export**

Lors de l’implémentation de l’export, nous avons commencé par insérer un bouton export dans l’interface graphique pour ensuite le lier à notre future méthode export dans, « ProjectsViewController » et « ProjectController ». Cette méthode permet de stocker les données de la base de données dans une fichier extension « json ». A partir de ce fichier, nous l’avons compressé dans une archive extension « tar.gz ». La difficulté était de bien écrire les données dans le fichier tout en respectant son format.

**Import**

Lors de l’implémentation de l’import, nous avons commencé par insérer un bouton import dans l’interface graphique pour ensuite le lier à notre future méthode import dans, « ProjectsViewController » et « ProjectController ». Tout d’abord, nous avons implémenté une méthode permettant de décompresser l’archive extension « tar.gz ». Ensuite, nous parcourons le fichier, une méthode va permettre de savoir si le projet existe déjà dans la base de données. Si ce n’est pas le cas, nous l’ajoutons dans la base de données en respectant les conventions.

**Intégration Cloud**

Chaque utilisateur dispose d’une quantité d’espace disque limitée. Pour définir cette limite, nous avons décidé de laisser l’administrateur du système la choisir. Le premier compte utilisateur créé se voit automatiquement attribuer ce rôle. Celui-ci peut changer ce paramètre depuis les réglages.

Nous avons décidé de calculer la taille sur disque occupée par chaque utilisateur sous forme de bytes. Pour le moment, seulement les projets, tags et collaborateurs sont comptés car prendre en compte les fichiers exportés s’est avéré d’être compliqué puisque nous n’avons pas moyen de localiser les fichiers exportés si l’utilisateur choisit de les stocker ou copier quelque part sans passer par l’application.

Concernant l’intégration cloud, nous avons implémenté l’intégration avec DropBox et Google Drive. Pour des raisons de sécurité, chaque utilisateur est prié de donner accès à l’application avec ses données de connexion. Nous avons choisi de permettre aux utilisateurs d’utiliser leurs comptes personnels car un compte centralisé pour tous les utilisateurs présente des risques de sécurité. Il suffit de connaitre les données de connexion (qui devront être incluses avec le code source) pour avoir accès total aux données de tous les utilisateurs.

**Implémentation :**

Nous utilisons un contrôleur, « CloudServiceController », qui englobe les fonctionnalités demandées. Celui-ci se charge d’initialiser et de servir d’intermédiaire entre les API et les vues de sélection service cloud et sélection de fichiers à télécharger.

**DropBox :**

A cause de certaines limitations de l’API, lors du téléchargement d’un fichier, nous avons choisi de seulement montrer les fichiers qui ont le bon format. L’API DropBox ne permet pas de distinguer les dossiers et fichiers. Une simple liste est affichée. De plus, lors de l’exportation vers le cloud, le fichier est automatiquement sauvegardé dans la racine du dossier cloud.

**Google Drive :**

L’API Google Drive n’offre pas une manière simple d’obtenir la signature hash d’un fichier. Pour le faire, nous nous sommes servis de la fonctionnalité AppProperties offerte par l’API qui permet de stocker des informations additionnelles sur le fichier. Pour prévenir l’utilisateur de l’existence préalable d’un fichier, la signature hash est calculée localement avant de charger le fichier et stockée dans le champ AppProperties.

Dues à des limitations de cette API, les données des utilisateurs sont stockées automatiquement dans un fichier StoredCredentials après avoir donné accès au compte Google Drive à l’application. Il n’y a donc pas moyen d’extraire les données et de les sauvegarder dans la base de données de l’application car tout le processus d’autorisation est géré par la librairie. De plus, nous avons du modifier le .jar google-oauth-client-java6-1.25.0 (marqué « custom ») de la libraire afin d’obtenir le lien d’autorisation, car la librairie n’offre pas de getter, pour offrir à l’utilisateur la possibilité de l’ouvrir dans le cas où son navigateur n’est pas ouvert automatiquement par la librairie.

**Ajout d’un calendrier**

Pour l’itération 3 un calendrier a dû être implémenté, pour ce faire nous avons utilisé la librairie calendarFX. Cependant il y avait un problème, cette librairie ne comprenait que 9 couleurs pour différencier les différents « objets » dans le calendrier. De ce fait nous avons modifié le fichier « .jar » de la librairie afin de rajouter une grande quantité de couleurs. C’est au final ce qui nous a posé le plus de problèmes, le reste s’est fait tout seul, nous avons choisi de faire l’affichage en semaines, et de séparer les tâches des projets.

Dans l'itération 4 nous avons dû rajouter la vue par mois pour le calendrier, cependant nous trouvions que les outils fournis par calendarFX était pas incroyable en ce qui concernait les mois. Nous avons donc décider d'un peu « tricher » et utiliser la vu par semaine mais en 5 fois afin de faire un mois. On a ensuite adapter l'affichage et créer les méthodes pour gérer tout ça .

**Ajout de la section d’aide :**

Pour l’**itération4**, nous avons décidés d’implémenter une section d’aide sous forme de « Tooltips » directement sur les fenêtres pour une simplicité d’accès mais également un menu à part où il est possible de voir une vidéo qui montre étape par étape l’utilisation de la fenêtre en question. Nous avons aussi ajouté des boutons afin de faciliter la lecture de la vidéo et pour une question d’ergonomie. Nous n’avons pas fait de tutoriel avec de simples images car au niveau du code c’était plus rapide et efficace mais aussi au niveau de la compréhension des fonctionnalités plus complexes.

Sécurité :

La base de données est chiffrée lorsque l’application est fermée et déchiffrée lorsque l’application est lancée avec les algorithmes fournis par java*x.crypto*. Plus précisément, l’algorithme utilisé sest *password based* (PBE) avec MD5 et TripleDES. Ceci garantit que des tierces n’auront pas accès aux données stockées dans celle-ci. De plus, les mots de passe des utilisateurs sont aussi chiffrés avant d’être stockés dans la base de données pour offrir une couche de sécurité supplémentaire afin de guarantir la confidentialité des données dans le cas (très improbable) où des tierces puissent avoir accès aux données dans la base de données.

Concernant les projets exportés, l’utilisateur est prié de saisir un mot de passe pour le protéger. L’application applique l’algorithme mentionné ci-dessus avec le mot de passe saisi par l’utilisateur pour chiffrer et protéger le fichier. A l’importation de ces fichiers, l’utilisateur est prié de saisir le mot de passe utilisé pour les chiffrer. Ceci garantit évite de stocker les mots de passe quelque part et seul l’utilisateur qui a exporté le fichier puisse y avoir accès.

Un problème a été rencontré lors de l’importation d’un fichier chiffré. Il n’y a pas moyen de détecter si le fichier a été chiffré par l’application ou même si le fichier a été déchiffré proprement. Hasher le fichier non-chiffré et stocker son hash dans la base de données pour déchiffrer le fichier et comparer son hash à celui stocké aurait pu être une solution. Désormais, il n’y a pas moyen d’associer le fichier à une entrée dans la base de données car changer le nom du fichier suffit pour invalider cette solution. Par conséquent, lorsque l’utilisateur saisit un mot de passe erroné pour déchiffrer le fichier, une exception est levée par *Archiver.extract* de la librairie *jarchivelib* qui cause que le fichier temporaire où le projet exporté sera extrait soit bloqué par l’application. Ceci n’arrive pas lorsque le mot de passe est correct en utilisant les mêmes méthodes. Les fichiers temporaires sont donc stockés dans un dossier *temp* qui sera vidé à chaque nouveau démarrage de l’application pour supprimer les fichiers erronés causés par *jarchivelib.*