UNIVERSITÉ LIBRE DE BRUXELLES

Équipe de développeurs :

El Bakkali Tamara Soufian, Konstantopoulos Alexios, Sanou Pap, Bernard Loïc, Haal Benoît, Nguyen Khanh, Manzer Ali, Piekarski Maciej

Client : Dallant Justin

Planification du projet

Itération 3 :

Pour la troisième itération, il a été convenu de réaliser les histoires suivantes :

- Histoire 4 : calendrier pour les tâches sans le choix des couleurs, 25 points
- Histoire 6 : collaboration pour un projet, 30 points
- Histoire 8 : statistique sur la durée initiale d'un projet, 5 points
- Ainsi que 10 points sur l'amélioration de l'affichage

Les fonctionnalités demandées par le client ont été correctement implémentés dans l'application. Les détails sur la distribution des points sur les différentes tâches complétées peuvent être consultés à partir du document « BurndownChart_iteration3.ods » disponible dans le dossier « team ».

Voici la liste des fonctionnalités qui ont été ajoutées au projet :

- Un tab Calendrier :
  - L'utilisateur peut sélectionner un ou plusieurs de ses projets pour voir leurs tâches sur le calendrier.
  - L'utilisateur peut naviguer la vue par mois.
- Dans le tab Statistiques :
  - Une nouvelle statistique « durée initiale du projet » a été ajoutée.
  - La fonctionnalité de l'exportation du projet a été mis à jour.
- L'amélioration de l'affichage :
  - Les éléments de l'affichage ont été centralisés et s'ajustent correctement avec la taille de la fenêtre.
  - Une limite minimale a été appliqué à la taille de la fenêtre de l'application
- Collaboration de projet :
  - L'utilisateur peut maintenant ajouter un ou plusieurs collaborateurs à ses projets
  - Les projets sur lesquels il collabore sont visible dans l'accuei
  - L'utilisateur peut attribuer des tâches aux collaborateurs.

Motivation des choix pris lors de la conception :

- Model-View-Controller (MVC) : Architecture classique pour la conception de projets visuels utilisant JavaFx. Cela simplifie la tâche du développeur qui tenterait d'effectuer une maintenance ou une amélioration sur le projet et améliore la clarté de l'architecture.

- SQLite : Base de données locales afin de stocker les utilisateurs ainsi que leurs projets. Les raisons principales sont que cette librairie est simple à comprendre et nous n'avons pas besoin de serveur distant.

- Singleton design pattern : Le singleton est un patron de conception qui permet de restreindre le nombre d'instanciations d'une classe, nous l'utilisons pour réduire les connexions à la base de données. Il permet aussi de réduire les ressources utilisées en instanciant un seul et unique objet utilisé à travers tout le code. Il est aussi possible, grâce au singleton, d'utiliser l'héritage car les méthodes ne sont plus globales ("static").

- Observer design pattern : Celui-ci permet d'établir un lien entre des objets qui dépendent d'un « sujet ». Ce sujet peut par la suite tous les notifier d'éventuels changements qu'il subit. Nous avons plusieurs composants d'interface différents qui dépendent des données du même modèle. Afin de limiter le couplage entre les différentes vues, tous les observateurs sont informés à partir du modèle qui leur passe l'objet qui a été modifié.

Document remis le 26/04/2021.