UNIVERSITÉ D'OTTAWA

SEG 2505 Introduction au Génie Logiciel

Rapport Final - Projet PerfectPC

Professeur : Laurent FREREBEAU

Arsene KANANE 300264969
Freddy YAMAKO TCHAMO 300374652

Groupe 32

2024-11-29

TABLE DE MATIERE

Introduction

Référence GitHub

Description des Choix de Conception

Description des Exigences Supplémentaires

Retour d’Expérience

Tableau de Synthèse des Contributions

Estimation du Temps Global

Conclusion

Introduction

Le projet **PerfectPC** avait pour objectif de concevoir une application autonome permettant aux utilisateurs de commander des ordinateurs personnalisés. Ce projet s'inscrit dans une logique d'apprentissage pratique, visant à appliquer les concepts théoriques étudiés en classe, notamment la gestion des bases de données, la modélisation UML et le développement d'applications mobiles.

À travers ce projet, nous avons exploré plusieurs dimensions du génie logiciel, incluant le travail en équipe, la gestion des rôles, la conception d'interfaces utilisateur intuitives et la résolution de problèmes techniques complexes.

Ce rapport vise à détailler les étapes majeures du projet, nos choix de conception, les difficultés rencontrées et les solutions adoptées. Nous y partagerons également les leçons tirées de cette expérience, ainsi que notre retour d'expérience sur le travail en binôme. En outre, une estimation du temps consacré à chaque livrable est présentée, ainsi qu'un tableau récapitulatif des contributions de chaque membre de l'équipe.

Le projet PerfectPC a été une opportunité de mettre en pratique des compétences essentielles en développement logiciel tout en relevant des défis concrets, nous préparant ainsi aux exigences du monde professionnel.

Référence GitHub

Le dépôt GitHub du projet est accessible à l'adresse suivante :
https://github.com/uOttawa-2024-2025-seg2505-projet/groupe-32

Ce dépôt contient l'ensemble des livrables requis, y compris :

- Le code source du projet.
- Les documents de conception et d'analyse.
- Les fichiers de configuration nécessaires pour exécuter l'application.

Description des Choix de Conception

Nous avions à implémenter dans ce projet à la demande d'un «acheteur» une application qui permet à des utilisateurs de commander des PC sur mesure. L'application se devait d'être autonome, c'est-à-dire, une application auto-suffisante, sans architecture client-serveur et sans communication réseau. Elle n'a donc pas besoin d'une connexion à Internet ou à un serveur distant pour fonctionner. Toutes les données nécessaires sont stockées et gérées localement.

Sur Android Studio, notre application destinée à un terminal Android à été développée en java en français, avec du code source en anglais et des commentaires expliquant chaque méthodes en français.

SQLite a été sélectionné comme système de base de données pour enregistrer des informations telles que les utilisateurs, les composants du stock et les commandes. Étant entièrement locale et utilisable en mode hors ligne, sa performance est adaptée aux besoins de notre application autosuffisante. Une classe dédiée, « DatabaseHelper », à été implémentée pour créer et initialiser la base de données. Les informations ont été organisées en table, une table «REQUESTERS » pour les utilisateurs, « STOCK » pour les composants et «COMMANDE», pour les commandes.

Nous avons opté pour RecyclerView pour afficher la liste des composants du stock dans l'interface utilisateur, car elle gère mieux les listes volumineuses et dynamiques. ListView à été utilisé pour les listes des utilisateurs et des commandes.

L'application supporte les rôles d'administrateur, de magasinier et d'assembleur, chacun ayant un email et un mot de passe prédéfinie dans la base de données. Chacun des utilisateurs désirant commander sont enregistrés par l'administrateur peuvent s'authentifier avec leurs propres informations.

l'administrateur « Administrator » est chargé de gérer les utilisateurs de l'application. Ce rôle lui permet de créer, modifier et supprimer les comptes des utilisateurs souhaitant accéder au service. À partir de l'interface dédiée, l'administrateur peut enregistrer de nouveaux utilisateurs avec leurs informations (email, mot de passe, etc.) et s'assurer qu'ils disposent des droits nécessaires pour se connecter et passer des commandes. Toutes les actions de l'administrateur sont enregistrées dans la base de données, garantissant une gestion efficace et sécurisée des utilisateurs autorisés.

Le demandeur de PC « Requester » est capable de créer des commandes de PC sur mesure, chaque commande étant un composant matériel ou logiciel. Par la base de données, il a accès à la liste des composants du stock et peut passer une

commande en sélectionnant l’élément voulu. Il est ensuite possible de consulter sa commande et de la gérer.

Le gestionnaire de stock de pièces détachées « StoreKeeper » à accès à la liste de composants du stock et peut la modifier en, soit ajoutant des nouveaux éléments ou en augmentant et diminuant la quantité d’un élément, une quantité 0 étant une rupture de stock.

l’assembleur de PC sur mesure « Assembler » est responsable de la gestion et du suivi des commandes en cours. Ce rôle lui permet d’accepter ou de rejeter les commandes en attente après vérification des disponibilités en stock. Une fois qu'une commande est validée, l’assembleur peut marquer cette commande comme « livrée » après assemblage virtuel, ce qui consomme les composants correspondants dans le stock. L’assembleur peut également ajouter des commentaires aux commandes pour communiquer des informations spécifiques aux autres rôles. Cette fonctionnalité garantit un suivi clair et détaillé des processus de commande et d’assemblage.

Toutes ces mesures permettent une authentification simple et efficace des utilisateurs, la création, modification et annulation rapide des commandes, une consultation en temps réel des pièces en stock et une interface utilisateur intuitive adaptée à un terminal tactile.

Description des Exigences Supplémentaires

- Nous avons mis en place une validation des champs d'entrée d'email et de mot de passe pour la connexion des utilisateurs pour s'assurer que les données saisies sont correctes. Pour éviter que l'application permette la connexion d'un utilisateur qui n'existe pas ce qui pourrait causer un plantage.

- Nous avons ajouté un système de notifications pour informer les utilisateurs de la réussite de l'authentification, ainsi que pour informer l'administrateur de la réussite de la réinitialisation de la base de données des utilisateurs, la réinitialisation du stock et la vidange de la base de données.

- Nous avions l'option d'ajouter la prise en charge de plusieurs langues dans l'application pour améliorer l'accessibilité et répondre aux besoins d'utilisateurs francophones et anglophones et pour des points bonus. Bien que le projet n'exige pas plusieurs langues, offrir un support multilingue améliore l'expérience utilisateur. Cela nécessite l'adaptation de l'interface utilisateur pour s'assurer que tous les textes sont traduits correctement. Mais nous n'avons pas pu l'ajouter, car on était limité par le temps.

- Nous avons pensé à permettre aux utilisateurs de choisir la couleur des pièces qui pourrait avoir cette option (couleurs personnalisées). Bien que cette fonctionnalité améliore l'expérience utilisateur, elle n'était pas prioritaire par rapport aux exigences principales. De plus, sa mise en œuvre aurait nécessité plus de temps. L'application utilise les composants de couleur standard cohérent, mais des options de personnalisation pourraient être ajoutées ultérieurement.

- De même que les couleurs nous avons pensé à ajouter des images représentant les composants affichés dans le stock. Cette fonctionnalité aurait aussi améliorer l'expérience utilisateur, même si elle n'était pas prioritaire par rapport aux exigences principales. Aussi, sa mise en œuvre aurait nécessité plus de temps. L'application n'utilise que les noms composants pour les identifier.

- Nous avons mis en place deux interfaces pour l'utilisateur Requester, une pour voir le stock et commander et l'autre pour voir et gérer ses commandes. Pour éviter une surcharge d'une page utilisateur avec les deux listes et soigné l'aspect visuel de l'application.

Retour d’Expérience

Organisation en Équipe

Le projet **PerfectPC** a été réalisé en binôme avec une division équitable des livrables pour que chacun puisse contribuer à tous les aspects du projet. Cependant, bien que les tâches aient été réparties de manière équilibrée, nous avons également tiré parti de nos affinités respectives en nous concentrant sur des rôles spécifiques.

Ainsi, l’un des membres de l’équipe a travaillé principalement sur l’implémentation des fonctionnalités liées aux rôles d’administrateur et d’assembleur, en mettant l’accent sur la gestion des utilisateurs et les autorisations. Pendant ce temps, l’autre membre s’est concentré sur les fonctionnalités du StoreKeeper et du Requester, en développant des interfaces et des fonctionnalités permettant de gérer les commandes, les stocks, et les interactions utilisateur.

Malgré cette répartition, chaque fonctionnalité était rigoureusement testée et discutée ensemble pour garantir une intégration harmonieuse et une cohérence dans l’ensemble du projet. Cela nous a permis de combiner nos efforts tout en nous spécialisant dans des domaines spécifiques.

Difficultés Rencontrées et Solutions

1. Gestion des rôles multiples dans l'application

- **Problème :** La gestion des différents rôles (Administrator, StoreKeeper, Requester, Assembler) a présenté un défi important, car chaque rôle dispose d’un ensemble unique de permissions. Cela a conduit à une complexité accrue dans la gestion du code, notamment pour éviter que les utilisateurs n’accèdent à des fonctionnalités qui ne leur sont pas destinées. De plus, les interactions entre les rôles, comme l’accès partagé à des données (ex. commandes ou stock), ont introduit des risques de confusion et d’incohérences fonctionnelles.
- **Solution :** Pour résoudre ce problème, une structure centralisée associant chaque rôle à une liste prédéfinie de permissions a été mise en place. Cela permet de définir clairement ce que chaque rôle peut ou ne peut pas faire dès sa création. Cette méthode réduit considérablement le risque d’erreurs et facilite la gestion du code.

2. Gestion des erreurs dans les formulaires utilisateur

- **Problème :** La validation des formulaires d’enregistrement des utilisateurs et d’authentification a nécessité un effort conséquent pour gérer différents

scénarios d’erreurs tels que des champs vides, des adresses email non valides ou des mots de passe invalides. Ces validations ont entraîné une duplication importante du code et une expérience utilisateur incohérente, les messages d’erreur variant parfois selon les écrans. De plus, un mauvais traitement des erreurs pouvait conduire à des bugs critiques, notamment des enregistrements incorrects en base de données.

- **Solution :** Une approche centralisée pour la validation des formulaires a été adoptée. Un seul groupe de fonction gère toutes les vérifications nécessaires, telles que la validité des adresses email ou la présence de champs obligatoires. Ces fonctions génèrent également des messages d’erreur uniformes et compréhensibles pour l’utilisateur. Ce processus réduit la redondance dans le code, améliore la cohérence de l’expérience utilisateur et permet d’identifier et de corriger rapidement les éventuelles erreurs.

3. Synchronisation du travail d’équipe sur GitHub

- **Problème :** Travailler en équipe sur un dépôt partagé a souvent engendré des conflits lors de la fusion des branches. Ces conflits, dus à des modifications simultanées sur les mêmes fichiers ou fonctionnalités, ont parfois entraîné des retards.
- **Solution :** Pour pallier ces difficultés, un workflow structuré a été adopté. Chaque membre de l’équipe travaille désormais sur une branche spécifique à une fonctionnalité donnée, ce qui limite les interférences. Avant toute fusion dans la branche principale, une revue collective rapide est effectuée pour valider la qualité du code et détecter les conflits potentiels.

Leçons Apprises

1. Importance d'une planification claire dès le début

L'un des principaux enseignements de ce projet est l'importance cruciale de bien planifier chaque étape avant de commencer l'implémentation. Définir des objectifs clairs, attribuer des rôles spécifiques aux membres de l'équipe, et établir un calendrier réaliste a permis de mieux structurer le travail. Cela a également aidé à identifier rapidement les risques potentiels et à répartir équitablement la charge de travail.

2. Collaboration efficace au sein de l'équipe

La collaboration a joué un rôle clé dans la réussite du projet. Nous avons appris que des outils comme GitHub ne sont pas seulement utiles pour partager le code, mais qu'ils nécessitent un workflow bien structuré pour éviter les conflits. Organiser des réunions régulières et communiquer fréquemment a permis de maintenir un bon alignement entre les membres, tout en favorisant une ambiance de travail productive.

3. Gestion des priorités et des fonctionnalités

Nous avons découvert qu’il est souvent préférable de se concentrer sur la mise en œuvre des fonctionnalités de base avant d’ajouter des éléments plus avancés ou optionnels. Cette approche incrémentale a non seulement réduit le stress lié au respect des délais, mais elle a également permis de tester et d'améliorer progressivement les fonctionnalités existantes.

4. Simplification et maintenabilité du code

Une leçon importante a été l'importance de maintenir un code simple, lisible et bien documenté. Nous avons appris à éviter les solutions complexes qui peuvent sembler innovantes mais qui sont difficiles à maintenir. Cela inclut également l'utilisation de bonnes pratiques comme les commentaires, les conventions de nommage et la séparation claire des responsabilités dans le code.

5. Adaptation aux imprévus

Le projet a mis en évidence que, malgré une bonne planification, des imprévus peuvent survenir, comme des erreurs inattendues ou des retards. Nous avons appris à rester flexibles et à ajuster nos priorités pour faire face à ces défis tout en respectant les délais.

6. Apprendre en faisant

Enfin, ce projet nous a permis de renforcer nos compétences techniques et organisationnelles. Chaque difficulté rencontrée a été une opportunité d’apprentissage, et nous avons terminé le projet avec une meilleure compréhension du développement logiciel, de la gestion de projet et de la collaboration en équipe.

Ces leçons serviront non seulement pour des projets académiques futurs, mais également dans le cadre professionnel.

Tableau de Synthèse des Contributions

Contributeur : Freddy Yamako Tchamo 300374652

Rôle : Développeur principal pour le StoreKeeper et le Requester

Description des travaux réalisés :

- J'ai conçu et implémenté la fonctionnalité de gestion de stock pour le rôle « StoreKeeper **»**. Cela comprenait la création de la liste des composants matériels et logiciels (par exemple, boîtiers, cartes mères, logiciels bureautiques, etc.) et la modification de ceux-ci.
- J'ai développé la fonctionnalité du rôle « Requester », permettant aux utilisateurs de créer et gérer des commandes pour des PC sur mesure.
- J'ai mis en place une interface où les utilisateurs peuvent voir et sélectionner les composants nécessaires à la configuration du PC, puis enregistrer cette commande dans la base de données. et une interface pour voir ses commandes.

Relations avec l'équipe :

- J'ai collaboré avec mon coéquipier qui à mis en place la base de données pour m'assurer qu'elle est bien implémentée et fonctionne bien avec la liste de composants et des commandes.
- Nous avons effectué des tests ensemble pour valider le bon fonctionnement de l'application pour notre scénario de démonstration pendant toute la durée du projet.

Contributeur : Arsene Kanane Akili 300264969

Rôle : Développeur principal pour l'administrateur, l'assembleur, le système d'authentification et la base de données

Description des travaux réalisés :
J'ai conçu et implémenté les fonctionnalités pour le rôle « Administrator », permettant la gestion des utilisateurs, incluant leur création, modification et suppression. J'ai également implémenté le rôle « Assembler », responsable de la gestion des commandes, comme l'acceptation, le rejet, et la clôture des commandes une fois assemblées.

En parallèle, j'ai développé le **système d'authentification** pour garantir une connexion sécurisée des utilisateurs. Cela inclut la validation des informations d'identification (email et mot de passe) ainsi que la gestion des erreurs en cas de données incorrectes.

J'ai aussi pris en charge la conception et la mise en œuvre de la **base de données SQLite**, où toutes les informations, incluant les utilisateurs, les commandes et les composants du stock, sont stockées. Une classe dédiée, « DatabaseHelper », a été créée pour initialiser la base de données et simplifier les interactions avec celle-ci.

Relations avec l'équipe :
J'ai collaboré étroitement avec mon coéquipier, qui a développé les fonctionnalités liées au gestionnaire de stock (StoreKeeper) et au demandeur (Requester). Nous avons travaillé ensemble pour nous assurer que les interactions entre les rôles, notamment via la base de données, soient fluides et sans erreur.

Estimation du Temps Global

Livrable	Temps estimé au total (heures)
Livrable 1	25
Livrable 2	20
Livrable 3	20
Livrable 4	30

Conclusion

Le projet PerfectPC a été une expérience enrichissante et formatrice, tant sur le plan technique qu'organisationnel. À travers ce projet, nous avons appliqué de manière concrète les concepts de génie logiciel, en concevant une application fonctionnelle répondant à des exigences spécifiques et en relevant des défis variés, comme la gestion des rôles multiples, la création d'une base de données robuste et l'optimisation des interactions utilisateur.

La collaboration en équipe a été essentielle pour mener à bien ce projet. En combinant nos compétences respectives, nous avons pu répartir efficacement les tâches, tout en garantissant une intégration harmonieuse de nos contributions. Cette expérience a renforcé notre compréhension de l'importance de la communication et de la planification dans un cadre de développement collaboratif.

Nous avons également tiré des leçons précieuses, notamment sur l'importance de la simplification du code et de la documentation. Malgré les défis rencontrés, comme les conflits de synchronisation ou les limitations temporelles, nous avons su nous adapter et apporter des solutions pratiques pour atteindre les objectifs fixés.

PerfectPC représente non seulement un aboutissement académique, mais aussi une préparation solide aux réalités du développement logiciel en milieu professionnel. Les compétences et enseignements acquis au cours de ce projet seront d'une grande utilité pour nos projets futurs, tant académiques que professionnels.