SEG 2505

Rapport Final – PC Manager

Université d'Ottawa

Automne 2024

Groupe 45 :

Mohamed Khalil EL OTHMANI-300389979

Nizar Taleb – 300371104

Zouheir Rachidi –300329396

Professeur :

Laurent Frerebeau

Introduction

Le projet **PC Manager** est une application mobile conçue pour optimiser la gestion des commandes et l'assemblage de PC. Ce système s'adresse aux entreprises et aux équipes spécialisées dans la fabrication et la livraison de composants informatiques. Grâce à une interface intuitive et des fonctionnalités robustes, l'application permet une gestion fluide et efficace des commandes, tout en intégrant un contrôle en temps réel du stock et des processus d'assemblage.

L'application repose sur un modèle à trois rôles principaux : **Requester, Storekeeper**, **Assembler**, et **Admin**, chacun disposant de fonctionnalités spécifiques.

- **Requester :** Soumission des commandes de composants PC.
- **Storekeeper:** Gestion de l'inventaire
- **Assembler :** Validation du stock, approbation ou rejet des commandes, et gestion de l'assemblage.
- **Admin :** Gestion des utilisateurs, des composants et des données globales.

Le projet utilise Firebase pour gérer les données en temps réel et garantir une synchronisation rapide entre les utilisateurs. Il met également l'accent sur une architecture évolutive et une sécurité adaptée pour répondre aux besoins actuels tout en offrant des perspectives d'améliorations futures.

Référence GitHub

Le projet est accessible sur GitHub à l'adresse suivante : Projet.

- Le code source complet.
- Les tests unitaires et les fichiers de configuration.
- Une documentation pour l'installation et l'exécution du projet.

1. Choix de Conception

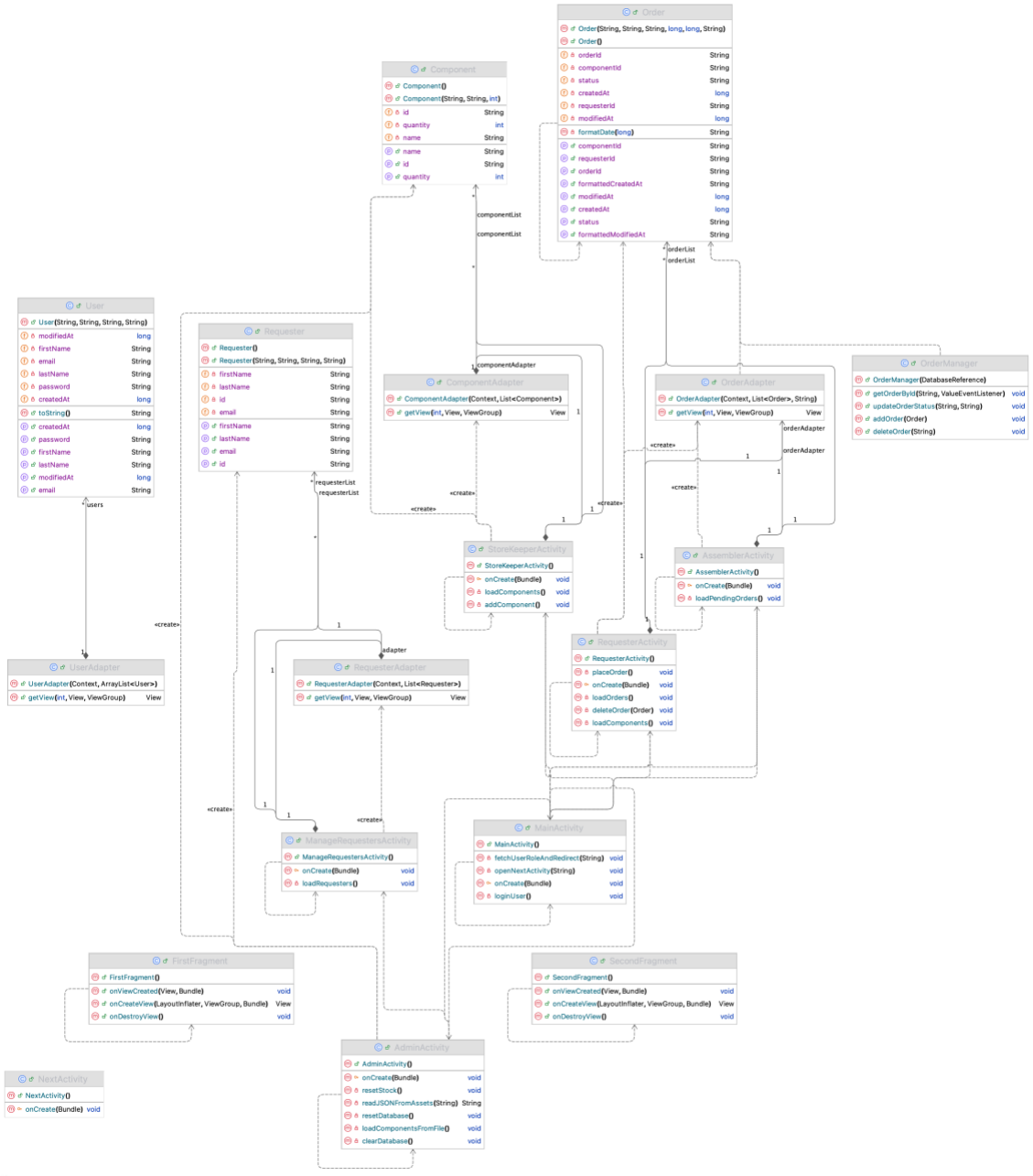
Ce diagramme UML illustre l'architecture globale de l'application **PC Manager**, en mettant en évidence les relations entre les principales classes, leurs attributs, et méthodes.

Résumé des éléments clés :

1. **Entités principales :**
    - **Order (Commande) :** Gère les informations sur les commandes (statut, composant, utilisateur).
    - **Component (Composant) :** Représente les éléments en stock (nom, quantité).
    - **User (Utilisateur) :** Inclut les rôles (admin, demandeur, assembleur).
2. **Activités et rôles :**

- **AdminActivity :** Gère les utilisateurs et les composants.
- **RequesterActivity :** Permet de passer des commandes.
- **AssemblerActivity :** Valide ou rejette les commandes selon le stock.

3. **Interactions clés :**
   - Relié à Firebase pour la gestion en temps réel des données (commandes, composants, utilisateurs).
   - Utilisation d'adaptateurs (`OrderAdapter`, `RequesterAdapter`) pour afficher dynamiquement les données.



Le diagramme UML met en évidence les différents composants du système et leur interaction, en regroupant les entités principales comme les utilisateurs, les commandes, et les composants, ainsi que les activités associées.

❖ Les choix de conception pour ce projet ont été guidés par les objectifs suivants : simplicité, évolutivité et intégration rapide avec Firebase.

- **Architecture logicielle :** MVC (Modèle-Vue-Contrôleur) pour séparer les responsabilités :
  - **Modèle :** Gestion des données et leur synchronisation avec Firebase.
  - **Vue :** Activités et fragments pour l'interface utilisateur.
  - **Contrôleur :** Logique métier et interaction avec Firebase via OrderManager.
- **Base de données :** Firebase Realtime Database pour une gestion en temps réel :
  - Users: Stocke les informations des utilisateurs (email, rôle).
  - Orders: Contient les commandes avec leurs détails (ID, timestamp, statut, etc.).
- **Tests unitaires :** Mockito utilisé pour les interactions Firebase avec des mocks.
- **Sécurité :** Règles de sécurité Firebase configurées selon les rôles des utilisateurs.

2. Description des Exigences Supplémentaires

Dans le cadre du projet **PC Manager**, plusieurs exigences supplémentaires ont été proposées pour enrichir l'expérience utilisateur et améliorer les fonctionnalités de l'application. Voici une description détaillée des exigences qui ont été proposées, celles qui ont été mises en œuvre, ainsi que celles qui ont été écartées.

Exigences Supplémentaires Proposées et Mises en Œuvre :

1. **Gestion des rôles des utilisateurs :**
   - **Proposition :** Différencier les fonctionnalités en fonction du rôle des utilisateurs (`Requester`, `StroeKeeper`, `Assembler`, `Admin`).
   - **Mise en œuvre :**
     - Les *Requesters* peuvent passer des commandes et visualiser leurs statuts.
     - Les *Assemblers* valident les commandes en fonction de la disponibilité du stock, approuvent ou rejettent les commandes, et visualisent les composants du PC assemblé.
     - Les *Admins* gèrent les utilisateurs, réinitialisent les données, et contrôlent les stocks.
2. **Validation du stock :**
   - **Proposition :** Mettre en place une vérification automatique du stock avant d'approuver une commande.
   - **Mise en œuvre :**

        - Les *Assemblers* reçoivent une alerte en cas de stock insuffisant, et la commande est mise en statut "En attente" jusqu'à ce que le stock soit réapprovisionné.
3. **Fonctionnalité de visualisation du PC assemblé :**
    - **Proposition :** Permettre aux assembleurs de visualiser les composants du PC une fois la commande approuvée.
    - **Mise en œuvre :**
        - Une activité dédiée (*VisualizePCActivity*) a été créée, où les assembleurs peuvent voir les détails des composants associés à une commande.
4. **Réinitialisation de la base de données par les administrateurs :**
    - **Proposition :** Ajouter une fonctionnalité permettant de réinitialiser les utilisateurs, les commandes et les stocks.
    - **Mise en œuvre :**
        - Les administrateurs peuvent réinitialiser complètement la base de données via des boutons dédiés dans l'interface.
5. **Affichage en temps réel des données :**
    - **Proposition :** Synchroniser les commandes, composants et utilisateurs en temps réel.
    - **Mise en œuvre :**
        - Grâce à Firebase Realtime Database, toutes les modifications sont automatiquement synchronisées sur les appareils connectés.

Exigences Supplémentaires Écartées :

1. **Notifications en temps réel :**
    - **Proposition :** Envoyer des notifications push aux utilisateurs pour les informer des changements de statut des commandes.
    - **Raison de l'exclusion :**
        - Cette fonctionnalité nécessitait l'intégration de Firebase Cloud Messaging, ce qui aurait augmenté la complexité du projet au-delà des contraintes de temps.
2. **Support multilingue :**
    - **Proposition :** Offrir la possibilité de basculer l'application entre plusieurs langues (anglais, français, etc.).
    - **Raison de l'exclusion :**
        - Bien que pertinent pour les utilisateurs internationaux, ce besoin n'était pas prioritaire dans le contexte du projet actuel.
3. **Analyse des données et tableaux de bord :**
    - **Proposition :** Ajouter un tableau de bord analytique pour permettre aux administrateurs de suivre les performances des commandes et du stock.
    - **Raison de l'exclusion :**
        - Cette fonctionnalité nécessitait des calculs et des visualisations complexes, qui auraient rallongé le développement.

Exigences Futures Envisagées :

1. **Gestion des paiements en ligne :**
   - Intégrer un module de paiement sécurisé pour permettre aux utilisateurs de régler leurs commandes directement via l'application.
2. **Tableau de bord analytique :**
   - Permettre aux administrateurs de consulter des graphiques et des rapports sur les commandes, le stock, et les performances.
3. **Intégration avec des API externes :**
   - Ajouter des services externes comme des APIs de fournisseurs pour automatiser la gestion du stock.
4. **Mode hors ligne :**
   - Permettre une utilisation limitée de l'application sans connexion Internet, avec une synchronisation automatique lors de la reconnexion.

3. Retour d'expérience

1. Organisation en Équipe

Dans le cadre du projet **PC Manager**, nous avons adopté une approche collaborative et structurée pour répartir les responsabilités et atteindre nos objectifs. Voici comment nous nous sommes organisés :

- **Répartition des rôles :**
  - Un membre de l'équipe s'est concentré sur le **backend**, notamment l'intégration avec Firebase et la mise en place des règles de sécurité.
  - Un autre membre a pris en charge le **frontend**, avec un accent particulier sur le design de l'interface utilisateur et les interactions utilisateur-application.
  - Un troisième membre a géré les **tests unitaires**, la validation des fonctionnalités et la documentation.
- **Outils de collaboration :**
  - **GitHub** a été utilisé pour le contrôle de version et la gestion des branches, ce qui a permis un développement parallèle sans conflit majeur.
  - **Slack** a servi de plateforme de communication pour partager les progrès, poser des questions et résoudre rapidement les problèmes.
  - Nous avons planifié des **réunions hebdomadaires** pour faire le point sur l'état d'avancement, ajuster les priorités et résoudre les blocages éventuels.
- **Planification et suivi :**
  - Nous avons subdivisé le projet en **sprints hebdomadaires**, chaque sprint ayant des objectifs clairs (ex. : intégration Firebase, développement de fonctionnalités clés, etc.).
  - Un tableau Kanban a été utilisé pour suivre les tâches en cours, terminées ou bloquées.

---

2. Difficultés Rencontrées et Solutions

Problème 1 : Intégration avec Firebase

- **Description :**
  - Les interactions avec Firebase, en particulier avec Realtime Database, ont généré des erreurs fréquentes, notamment des `NullPointerException` et des problèmes de permissions.
- **Solution :**
  - Nous avons utilisé des **mocks** pour simuler les réponses Firebase lors des tests unitaires.
  - Une révision approfondie des **règles de sécurité Firebase** a été effectuée pour s'assurer que les utilisateurs n'essaient pas d'accéder à des données non autorisées.

Problème 2 : Gestion des conflits Git

- **Description :**
  - Des conflits sont apparus lors de la fusion des branches, en particulier sur des fichiers critiques comme `build.gradle` ou des layouts XML.
- **Solution :**
  - Nous avons introduit une règle stricte de **revue de code avant fusion**, où chaque membre devait valider les changements avant qu'ils ne soient intégrés à la branche principale.

Problème 3 : Tests unitaires avec Firebase

- **Description :**
  - Les callbacks Firebase, qui fonctionnent de manière asynchrone, ont rendu les tests difficiles, car il était complexe de valider les résultats en temps réel.
- **Solution :**
  - Nous avons converti certaines méthodes en appels synchrones pendant les tests et utilisé des frameworks comme Mockito pour capturer les résultats des callbacks.

Problème 4 : Design de l'interface utilisateur

- **Description :**

    - Il était difficile de concevoir une interface qui soit à la fois esthétique et fonctionnelle, notamment pour la gestion des différents rôles utilisateurs (Requesters, Assemblers, Admins).
- **Solution :**
    - Nous avons adopté une approche modulaire en concevant des layouts réutilisables pour minimiser les redondances.
    - Un feedback régulier a été sollicité auprès des membres de l'équipe pour garantir une meilleure expérience utilisateur.

---

3. Leçons Tirées et Recommandations pour l'Avenir

Ce qui s'est bien passé :

1. **Collaboration efficace :**
    - Grâce à l'utilisation d'outils comme GitHub et Slack, la communication entre les membres de l'équipe a été fluide.
    - Les réunions hebdomadaires ont permis de maintenir une vision claire des objectifs et des priorités.
2. **Répartition des responsabilités :**
    - La spécialisation des tâches a permis à chaque membre de l'équipe de se concentrer sur ses points forts.
    - Cela a également réduit le risque de duplication d'efforts ou de confusion.
3. **Documentation continue :**
    - Une documentation régulière a été maintenue sur GitHub, ce qui a facilité le suivi des changements et la résolution des problèmes.
4. **Gestion des tests :**
    - Les tests unitaires et la simulation des réponses Firebase ont joué un rôle crucial dans la stabilité du produit final.

Ce qui aurait pu être amélioré :

1. **Standardisation du code :**
    - Des incohérences dans le style de codage ont été observées au début du projet.
    - **Recommandation :** Mettre en place des **linter tools** ou des conventions dès le début.
2. **Anticipation des problèmes d'intégration :**
    - Plusieurs bugs sont apparus en phase finale en raison de l'intégration tardive des différents modules.
    - **Recommandation :** Effectuer des tests d'intégration dès les premières semaines.
3. **Gestion du temps :**

- Certains sprints ont pris plus de temps que prévu en raison d'une estimation initiale incorrecte des efforts requis.
- **Recommandation :** Allouer plus de temps aux tâches critiques, comme les tests et l'intégration Firebase.

4. **Enrichissement de l'application :**
   - Bien que l'objectif initial ait été atteint, certaines fonctionnalités supplémentaires (ex. : notifications en temps réel, tableau de bord analytique) auraient pu être intégrées.
   - **Recommandation :** Définir une feuille de route pour intégrer ces améliorations dans des versions futures.

4. Tableau de synthèse des contributions

Tableau de Synthèse des Contributions

Nom	Prénom	Numéro Étudiant	Contributions principales
El Othmani	**Mohamed Khalil**	**300a389979**	Intégration Firebase, logique backend, mise en place des règles de sécurité et gestion des interactions avec la base de données.
RACHIDI	**Zouheir**	**300329396**	Développement des vues Android (UI/UX), conception des interfaces utilisateurs et intégration des layouts XML.
TALEB	**Nizar**	**300371104**	Écriture des tests unitaires avec Mockito, coordination GitHub, gestion de la documentation et suivi des tâches.

Détails par Contributeur

EL OTHMANI Mohamed Khalil (Team Leader):

- **Travaux réalisés :**
  - Responsable de l'intégration de Firebase pour les modules utilisateurs, commandes et composants.
  - Implémentation des règles de sécurité Firebase pour assurer la gestion basée sur les rôles (Requester, Assembler, Admin).
  - Développement du backend pour gérer les commandes en attente, leur validation ou rejet, et la mise à jour des stocks.

    - Coordination avec l’équipe pour intégrer le backend avec le frontend développé par Zouheir.
- **Relations avec l'équipe :**
    - Collaboration étroite avec Zouheir pour ajuster les données renvoyées aux interfaces.
    - Discussions régulières avec Nizar pour valider les tests unitaires des méthodes backend.
    - Échange constant pour documenter et expliquer les fonctions Firebase utilisées dans le projet.

RACHIDI Zouheir

- **Travaux réalisés :**
    - Responsable de la conception des vues Android, incluant les dashboards pour les rôles Requester, Assembler, et Admin.
    - Mise en place des interfaces interactives telles que les listes de commandes, les boutons d’approbation/rejet, et les formulaires de création de commande.
    - Intégration des layouts XML avec les classes Java et gestion des événements utilisateur (onClickListeners).
    - Tests fréquents des interfaces pour garantir une expérience utilisateur fluide.
- **Relations avec l'équipe :**
    - Communication avec Mohamed Khalil pour assurer que les données Firebase s’affichent correctement dans les vues.
    - Interaction avec Nizar pour tester les fonctionnalités UI/UX et résoudre les bugs liés aux interfaces.

TALEB Nizar

- **Travaux réalisés :**
    - Responsable des tests unitaires pour les fonctionnalités critiques, notamment la gestion des commandes et la validation des rôles utilisateurs.
    - Utilisation de Mockito pour simuler les réponses Firebase et valider les interactions asynchrones.
    - Coordination des branches GitHub et résolution des conflits lors des fusions.
    - Rédaction de la documentation du projet, incluant l’explication des modules et des tests.
- **Relations avec l'équipe :**
    - Collaboration étroite avec Mohamed Khalil pour comprendre la logique backend et écrire des tests ciblés.
    - Participation aux revues de code et feedback sur les interfaces développées par Zouheir.

- Rôle de coordinateur pour s'assurer que les livrables respectent les délais convenus.

Synthèse

Chaque membre de l’équipe a apporté des contributions spécifiques et complémentaires, ce qui a permis de mener à bien ce projet. La coordination, l’écoute mutuelle, et l’engagement de chacun ont été des facteurs clés de succès pour la finalisation de cette application.

5. Temps global passé

Tâches	**Temps total (heures)**
Initialisation du projet	15 h
Développement des fonctionnalités	39 h
Tests et validation	20 h
Documentation	13 h
Total	**87 h**