UQAC

Université du Québec
à Chicoutimi

Stage-Projet I

Rapport N.2

Sujet

Développement d'une application mobile permettant la gestion simplifiée des dépenses dans un groupe.

Étudiants

Maël Garnier (GARM02020500)
Julien Becheras (BECJ27040300)

Maître de stage / tuteur

Hamid Mcheick

Date

29/05/2025

I. Le fond du projet picsou

A. Description du problème

Vivant au sein d'une colocation de 7 personnes, chaque jour nous rencontrons des problèmes pour gérer nos dépenses d'argent. Que ce soit des courses quotidiennes aux activités ou charge commune, nous éprouvons le besoin de partager de manière efficace et pratique nos factures au sein de notre colocation.

B. Solutions existantes

Pour résoudre ce problème nous avons opté pour une application mobile qui se nomme tricount. Son usage nous aide grandement au quotidien et sa facilité de prise en main et de partage en fond une bonne solution. Cependant, avec notre de temps d'utilisation conséquent quotidiennement de cette application, nous avons pu relever de nombreux défauts, notamment pour la gestion de stock en avance. En effet, la simplicité de cette application en fait aussi son défaut. Pour des gestions de ticket de caisse long et comportant de nombreux articles vous serez obligé de sortir un tableur excel avant de pouvoir entrer le coût des courses de chacun. De plus la difficulté pour réaliser des remboursements nécessitant de nombreuses applications pour communiquer, partager les dépenses et finalement rembourser les personnes.

II. Méthodologie

A. Solution proposée

Fort de notre expérience durant cette année avec l'application tricount, les points d'amélioration nous apparaissent clairement. Pour notre application de gestion de dépense nous avons souhaité nous axer sur le principe d'une messagerie. Ce choix permettra une communication à l'intérieur de l'application nous permettant de combler l'une des lacunes de Tricount: l'impossibilité de communiquer au sein de l'application.
Nous avons également souhaité nous attaquer à une autre faille de cette application: le manque d'efficacité dans la gestion de dépense complexe ou les sous-totaux par utilisateurs sont compliqué à configurer.

Pour nous, une solution idéale est une application sur laquelle chacun peut se créer un compte et synchroniser ses dépenses dans une base de données. Des remboursements simplifiés sont proposés aux personnes concernés en prenant en compte l'ensemble des dépenses afin de minimiser les transactions.

Notre application se découpe en groupe. Chaque groupe possède un propriétaire qui peut attribuer des rôles à chaque membre.
Parmis ces rôles:

- Owner : Peut tout modifier et supprimer dans le groupe
- Admin : Peut changer le rôle de membre (jusqu'à participant de confiance), supprimer des membres ou des spectateurs.
- Participant de confiance: Peut modifier / supprimer / ajouter n'importe quelle dépense
- Participant : Peut modifier / supprimer / ajouter une dépense qui lui appartient ou dont il fait parti
- membre : Peut éditer les dépenses qu'il a créée et où il apparaît, participer à des dépenses
- spectateur : peut uniquement voir le tricount, aucune modification ou création permise.

L'interface de l'application ressemble quant à elle à celle d'une messagerie : on ajoute des amis à partir de leurs identifiant, on peut ajouter des dépenses d'ami en ami, ou créer un groupe et ajouter des dépenses communes au groupe. Des comptes temporaires seront possibles, l'accès à ces comptes se fera grâce à un lien. Ceci seront définitivement sous le statut de spectateur.

Si un utilisateur est supprimé, les dépenses non remboursées dont il est contributeur ou pour lesquelles il est destinataire seront gérées par les admin/propriétaire du groupe.

- Suppression de toutes les dépenses auxquelles il a contribué et/ou de toutes ses dettes (les répartitions au sein de la dépense sont recalculées)
- Obligation de remboursement des dettes avant départ
- Changement de propriétaires des dettes/emprunts (possibilité de faire au cas par cas)

La lenteur du processus de création d'une dépense quand on prend en compte l'utilisation des différents outils dans le cas d'application comme Tricount et un point clé sur lequel nous souhaitons capitaliser. Le fait de centraliser les fonctionnalités de gestion de factures, de création de dettes et d'une messagerie permettant de réaliser des demandes de remboursement rapidement entre utilisateurs fera gagner beaucoup de temps à nos utilisateurs. Cet outil intuitif et complet permettra de combler de nombreuses lacunes des outils aujourd'hui existants.

B. Fonctionnalités clés

Voici l'ensemble des fonctionnalités que nous souhaitons inclure dans ce projet. Celles-ci sont classées par ordre d'importance.

Priorité haute

- Création d'un compte pour un utilisateur
- Ajout et suppression d'amis
- Création de groupe d'utilisateurs
- Ajout et suppression de dépenses simple
- Algorithme de calcul de la somme dû ou de la somme remboursée spécifiquement dans des groupes ou d'amis en amis
- Dépense simple

Priorité Moyenne

- Tableau de bord récapitulatif
- Implémentation d'un message broker pour l'ajout de nouvelles dépenses
- Scan d'un ticket de caisse et création automatique de la dépense
- modification de dépense

Priorité Faible

- Ajout d'amis en rapprochant simplement les téléphones (tag nfc).
- Système de vote pour supprimer des dépenses
- Tag GPS sur les dépenses
- Commentaires pour des dépenses
- Messagerie implémentée dans l'application, d'utilisateur à utilisateur ou dans un groupe

- Deux modes de fonctionnement simple et expert
- Tutoriel pour chaque mode avec une partie “aide” dans l’application

Technologies

- Application mobile et application web : Flutter
- API : Rust, Diesel, Rocket
- Base de données : Postgres, Docker
- Système d’exploitation de l’API : raspberry OS (sur raspberry pi)

III. Environnement de travail

A. Outils utilisés

- Versionning et développement en équipe : Github
- IDE : Android Studio et RustRover
- Conception de l’interface et de l’architecture : Figma
- Conception de la base de données : Figma
- Plateforme SCRUM : Azure DevOps
- Test Api : Postman

B. Méthodologie

Afin d’améliorer et d'accélérer la gestion de projet et le développement, nous avons mis en place des concepts de la méthodologie SCRUM.

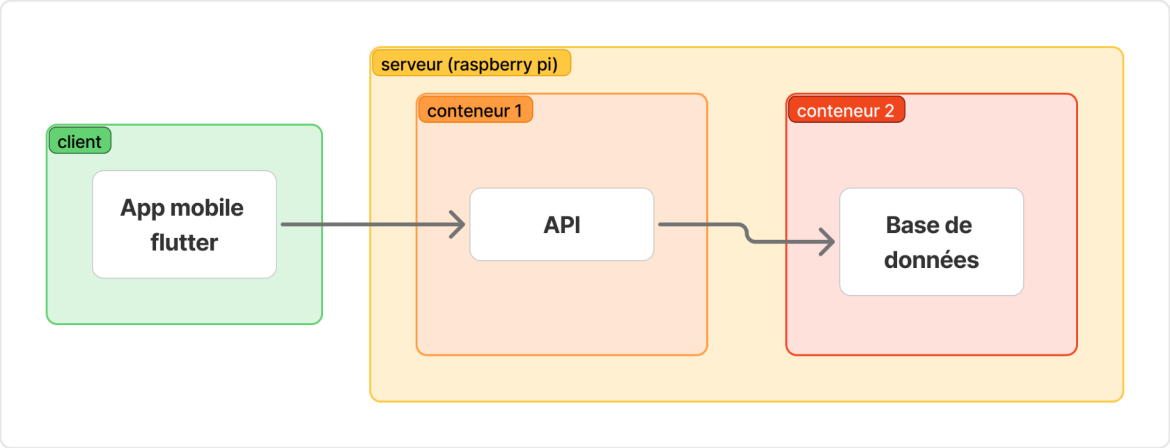
Nous avons réalisé des itérations de travail par période de deux semaines en se basant sur le principe des sprint : une première réunion au début du sprint pour se décider sur les fonctionnalités qui seront développées et par qui, ainsi que les deadline. Une seconde réunion pour mettre en commun le travail réalisé, discuter des problèmes rencontrés et si besoin ajuster les prochaines étapes du projet.

IV. Evolutions depuis le premier rapport

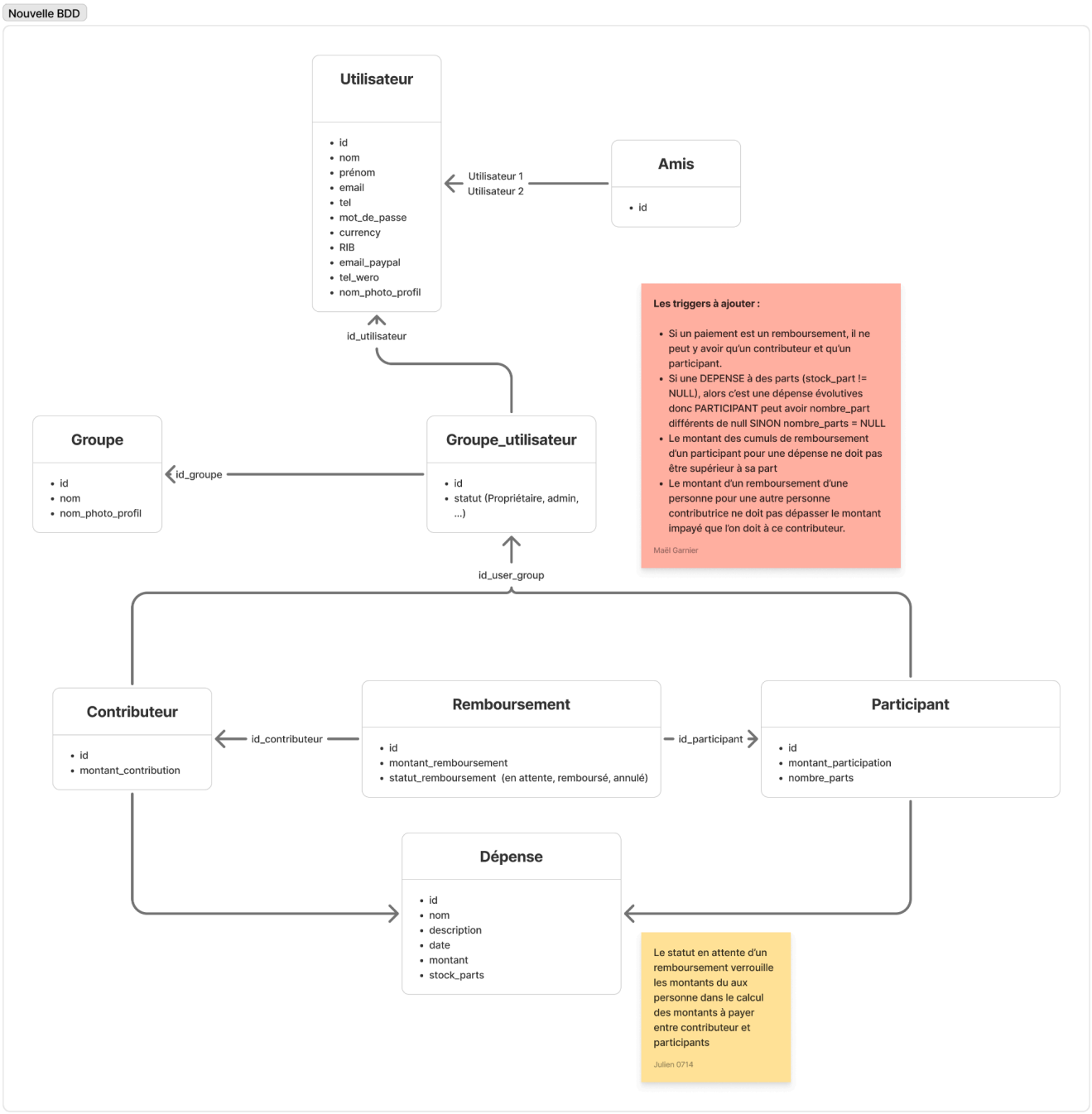
A. Architecture

L’architecture de notre application repose sur un principe simple: l'adaptabilité. Il faut que notre solution soit disponible sur toutes les plateformes. Du téléphone android ou iphone à la tablette tactile en passant par l’ordinateur, peu importe la plateforme notre solution doit pouvoir être disponible. Pour cela, le choix du flutter nous permet une solution qui tourne sur n'importe quelle plateforme.
Afin de communiquer entre utilisateurs de manière efficace, sécurisée et fonctionnelle toutes les données d'utilisateur sont hébergées sur une base de données postgresql. La robustesse et la performance de cette plateforme notamment pour la gestion de bases de données volumineuses en font une solution idéale.
Afin de communiquer de manière sécurisée avec notre base de données, notre application passera par une API en Rust. Cette API aura pour mission de filtrer les requêtes lui parvenant avant de communiquer par son intermédiaire avec la base de données.
L’API et la base de données sont déployées au sein de conteneur Docker permettant plus tard une migration facilitée vers des serveurs cloud permettant de faire de la mise à l'échelle horizontale (horizontal scaling).

Pour le moment notre projet en cours de développement est installé sur un raspberryPi.



La base de données à aussi évoluée depuis le dernier rapport : nous nous sommes concentrés sur la simplification de la gestion des dépenses, réduisant également le nombre de tables et de transactions.



B. Développement continu / Intégration continue

Afin de simplifier le développement de l'API, nous avons mis en place du développement continu.

Pour ce faire, nous avons dans un premier temps configuré un github action sur la branche staging du dépôt github permettant de compiler le projet à chaque nouvelle version. Nous avons ainsi un indicateur clair nous informant si la nouvelle version n'est pas fonctionnelle.
Nous avons également créé un second github action sur la branche main (branche de production) afin automatiquement déployer chaque nouvelle version sur notre Raspberry Pi. Pour cela, nous avons rédigé un docker-compose (utilisant un Dockerfile et un script shell) déployant automatiquement une image fonctionnelle de l'api. Ainsi, nous avons fait en sorte que le github action se connecte via SSH sur le Raspberry Pi, construise la nouvelle image du docker-compose et la re-déploie.
Ce processus a grandement amélioré notre aisance dans le développement du projet, nous permettant de porter beaucoup moins d'importance au déploiement ainsi qu'à la configuration.

C. Interface et expérience utilisateur

Nous avons développé l'application en suivant principalement les designs effectués sur figma. Nous avons cependant effectué quelques améliorations au cours du développement notamment après l'avoir testé auprès d'autres utilisateurs. L'interface et la palette de couleurs restent malgré tout similaires.
Au cours du développement, nous avons porté une très grande importance à la fluidité de l'application. En effet, au-delà d'avoir une interface utilisateur attrayante, une application moderne se doit d'avoir des animations et des transitions fluides et travaillées. Pour cela, nous avons créé une version modifiée du routeur natif de Flutter nous permettant de créer nos propres transition lors des changements de pages, et choisir celles qui nous conviennent en fonction du contexte.
Nous avons également utilisé des nouvelles fonctionnalités de Flutter tel que "Hero" garantissant des transition et animations de composants fluides et modernes.

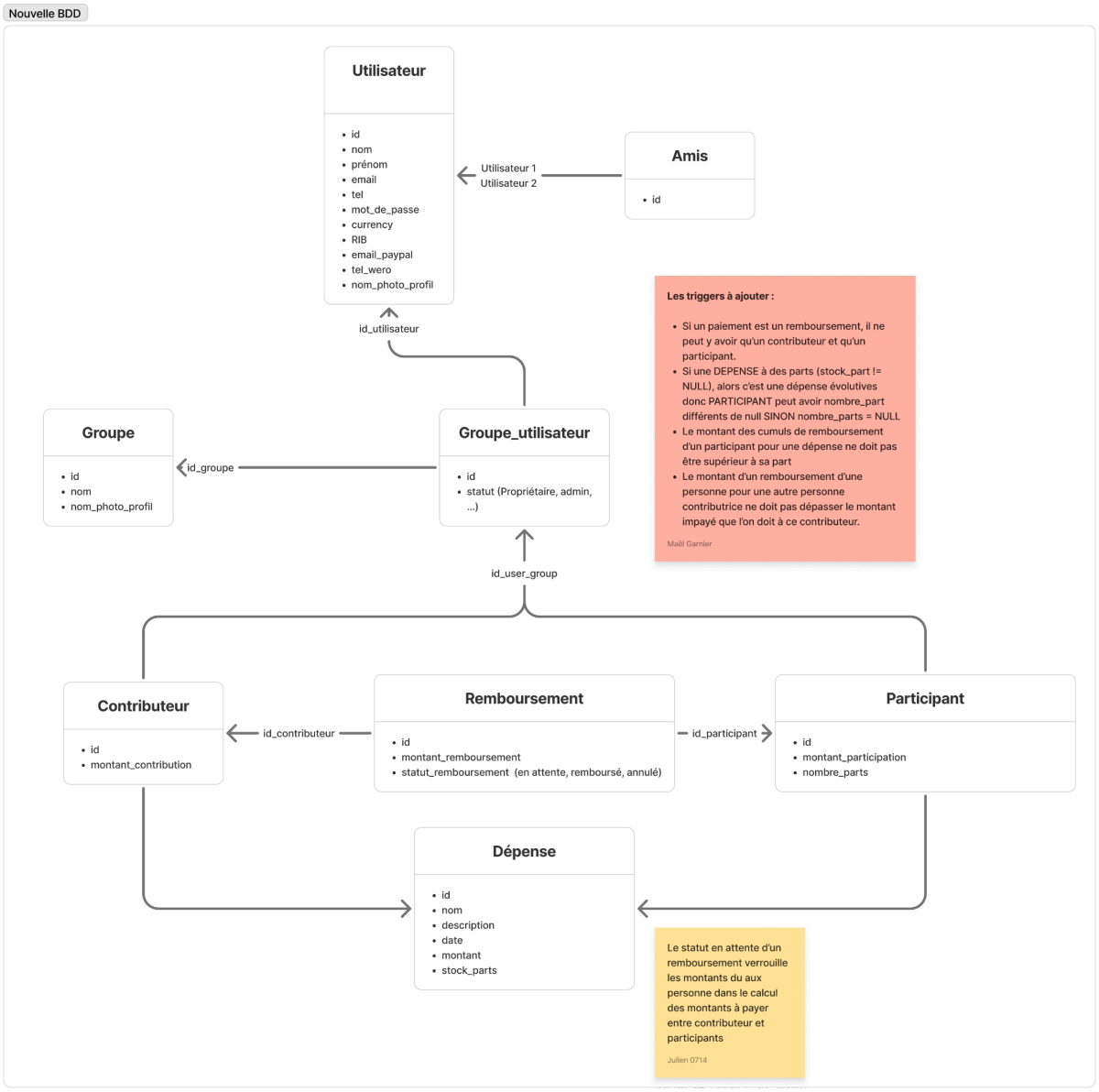
D. Fonctionnalités prêtes à l'emploi

1. Application mobile

L'application mobile permet maintenant d'effectuer la grande majorité des fonctionnalités que nous avons placé en "priorité haute". Ainsi, nous somme maintenant capable de créer un compte, se connecter, modifier nos informations personnelles, rechercher des amis, envoyer des requêtes d'amis, accepter ou supprimer des requêtes d'amis, voir et supprimer un ami, créer un groupe, créer une discussion, étudier les détails d'un groupe ou d'une discussion, créer une dépense et y ajouter des contributeurs et des participants et enfin voir les dépenses par groupes et discussions.
Il est important de noter que la phase la plus longue du développement à consisté en la création de tous les composants essentiels de l'application. Maintenant que cette phase est terminée, l'implémentation de nouvelles pages et nouvelles fonctionnalités sera relativement rapide.

2. Api

En entrant plus en profondeur dans la réalisation de notre projet nous nous sommes aperçus que notre base de données comporte plusieurs défauts majeurs. Une reconception et implémentation de celle ci à donc était nécessaire afin de mieux répondre à nos objectifs pour cette application.



De plus nous avons essayé de concevoir une API se rapprochant de l'architecture REST il a donc fallu réaliser un travail de conception préalables nous amenant à cette architecture:

Pour chaque route, il a donc également fallu déterminer les règles nécessaires afin de sécuriser les données transmises ou insérées en BD. Dans un souci d'optimisation, il a également fallu minimiser la quantité de requêtes effectuées pour chaque route à la BD afin d'éviter de futur problème de surcharge de la base de données et de l'API. Pour le moment les routes créées sont celles nécessaires à la gestion des amis, la gestion d'un utilisateur (création, connexion, modification) mais aussi la gestion des groupes et des dépenses. La gestion de toutes routes vérifient également si l'utilisateur possède le droit de réaliser les actions qu'il appelle. Voici le tableau récapitulatif des différents privilèges des utilisateurs :

Table de référence

UserGroup

Status	Rôle	Avantage
0	Propriétaire	supprimer groupe, changer le role de membre (jusqu'à n°0)
1	Admin	Ajouter des utilisateur, changer le rôle de membre (jusqu'à n°2), suprimer des membre ou des spectateur.
2	Participant de confiance	éditer toutes les dépenses
3	Participant	éditer les dépenses dans lesquelles il apparait
4	Membre	éditer les dépenses qu'il a crée et ou il apparait, participer à des dépenses
5	Spectateur	voir les dépenses et les messages

E. Fonctionnalités en cours de développement

1. Application mobile

Nous sommes actuellement en train de poursuivre le développement de l'interface de gestion de groupe, dépenses et discussion. Ainsi, les prochaines fonctionnalités immédiates seront le visionnage de détails de dépenses, la suppression de dépenses, de groupe et de discussions.

2. Api

La gestion des remboursements est en cours d'implémentation permettant de générer des remboursements intelligents entre personnes partageant des groupes en commun.

F. Futures fonctionnalités

1. Application mobile

Les futures fonctionnalités importantes seront la gestion des remboursements, ainsi que des études financières plus poussées permettant de facilement comprendre nos dépenses et remboursements d'une manière plus globale.

2. Api

Notre application étant hébergée sur un Raspberry Pi durant la durée de ce stage, nous allons bientôt devoir changer de support. Afin de préparer la suite il est envisagé que nous l'hebergions sur une solution similaire le temps du développement des prochaines partie clefs mais le but et de passer d'ici peu de temps sur une infrastructure cloud comme azure en mettant en place du scaling horizontal mais également un répartiteur de charge (load balancing) et une base de données noSQL afin d'héberger les photos que les utilisateurs déposent sur notre application.

De plus, il sera nécessaire de mettre en place un message broker afin de diffuser les messages et ajouts de dépenses à tous nos utilisateurs.

Afin de permettre une utilisation plus rapide et moins contraignante il sera possible d'accéder à des groupes grâce à des liens créant ainsi des compte

temporaire à notre application. La logique de compte temporaire ainsi que son utilisation et ses répercussions sur la base de données restent à concevoir et implémenter.
Une des fonctionnalités phares de notre projet est l'ajout d’une possibilité de scan des tickets de caisse grâce à une IA qui viendrait détecter les champs sur l’image. Cette fonctionnalité reste à implémenter.