KEEP IN TOUCH

Projet de Génie Logiciel

Amel Dussier, Bastien Clément, Antoine Drabble et Guillaume Serneels

Table des matières

Introduction

Dans le cadre de notre projet de Génie Logiciel, nous allons réaliser une application de chat pour clients mobiles, ressemblant à des produits comme WhatsApp ou Telegram.

Cette application s'appellera « Keep in touch », et proposera dans sa première version toutes les fonctionnalités permettant une communication rapide et conviviale : possibilité de trouver ses amis et de rester en contact avec eux, discussion avec un ou plusieurs contacts via des groupes, discussions publiques ouvertes à tout le monde, etc.

L'aspect sécurité et protection des utilisateurs sera également présente : possibilité de signaler des comportements inadéquats, de déléguer l'administration d'un groupe, ou encore de bloquer des contacts si nécessaire.

Le but de notre projet est d'arriver à développer un produit fini, utilisable et qui soit suffisamment modulaire pour être étendu facilement grâce à des ajouts de fonctionnalités à l'avenir. Bien sûr, nous aimerions également que le produit soit esthétique et agréable à utiliser, afin que les utilisateurs partagent l'enthousiasme que nous avons eu lors de ce projet.

Ci-dessous le logo de notre première version, qui rappelle le nom de l'application grâce à son initiale, et qui symbolise également l'interaction sociale que « Keep in touch » veut promouvoir :

Figure 1 : Logo de l'application

Analyse

Fonctionnalités

La première fonctionnalité proposée à l'utilisateur sera bien sûr la création d'un compte, avec un login unique et un mot de passe. Ce compte lui permettra de s'authentifier dans l'application, et sera lié à toutes les données de l'utilisateur (contacts, messages, etc.).

Une fois connecté, l'utilisateur aura le choix entre trois types de chat différents :

- **Chat privé**
  Ce chat permettra de communiquer avec un contact de son choix. La première version permettra de communiquer par message texte puis, si le temps le permet, nous étudierons la possibilité d'implémenter des fonctionnalités supplémentaires (la communication audio, des discussions chiffrée de bout en bout avec un système utilisant des clés publiques/privées, etc.).

- **Chat groupe**
  Ce chat permettra de communiquer avec plusieurs contacts en même temps. Un utilisateur pourra créer un groupe et lui donner un nom. Il pourra y inviter les contacts de son choix. Toutes les personnes ajoutées pourront envoyer des messages, qui seront lisibles par tous les membres du groupe. Le créateur du groupe en sera l'administrateur et pourra y ajouter ou supprimer des membres. Il pourra également désigner d'autres administrateurs pour ce groupe.

- **Chat publique**
  Ce chat est similaire au chat de groupe, mais il a la particularité d'être ouvert à tout le monde. Les membres ne sont pas invités mais créent ou rejoignent eux-mêmes un chat public existant.

En cas de problème, une fonctionnalité de report de messages permettra aux utilisateurs de signaler un message abusif. Tous les reports seront envoyés à l'administrateur de l'application et à l'administrateur du groupe s'il ne s'agit pas d'un chat privé. Les reports envoyés à l'administrateur seront stockés dans une base de données et seront accessibles via une interface web.

Un utilisateur pourra également en bloquer un autre afin de ne plus recevoir de messages de sa part.

Partage des responsabilités entre le serveur et le client

Le serveur s'occupe de gérer l'ensemble des opérations, le client n'est qu'une vue des données du système. Il affiche les informations que l'utilisateur demande et transmet les actions qu'il souhaite effectuer. Le serveur s'assure qu'il en ait l'autorisation.

Le serveur fonctionne en permanence. Le client peut à tout moment s'y connecter en démarrant l'application et en fournissant ses informations de connexion (nom d'utilisateur et mot de passe).

Si l'application n'est pas lancée, l'utilisateur est considéré comme déconnecté. Après la première connexion, le client maintient un cookie (token) de session lui permettant de se reconnecter rapidement.

Un diagramme d'activité général

Le schéma ci-dessous montre l'architecture générale de notre solution, ainsi que la répartition des différentes responsabilités :

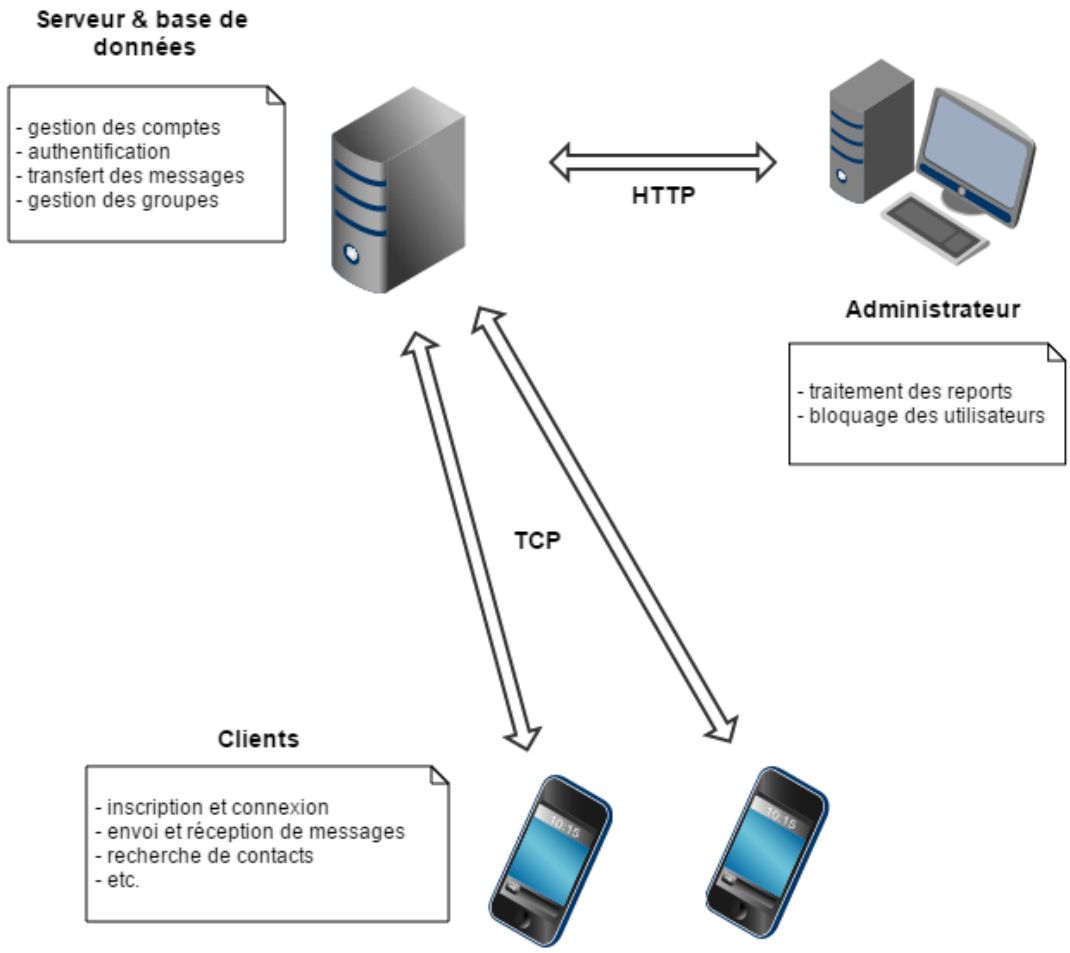

Figure 2 : Diagramme d'activité général

Le client permettra aux utilisateurs d'accéder à de nombreuses fonctionnalités. Seules les principales ont été mentionnées sur ce diagramme afin de ne pas le surcharger.

Cas d'utilisation

Diagramme général de contexte

Ci-dessous le diagramme général de contexte, avec les différents acteurs et cas d'utilisation que nous avons identifiés :

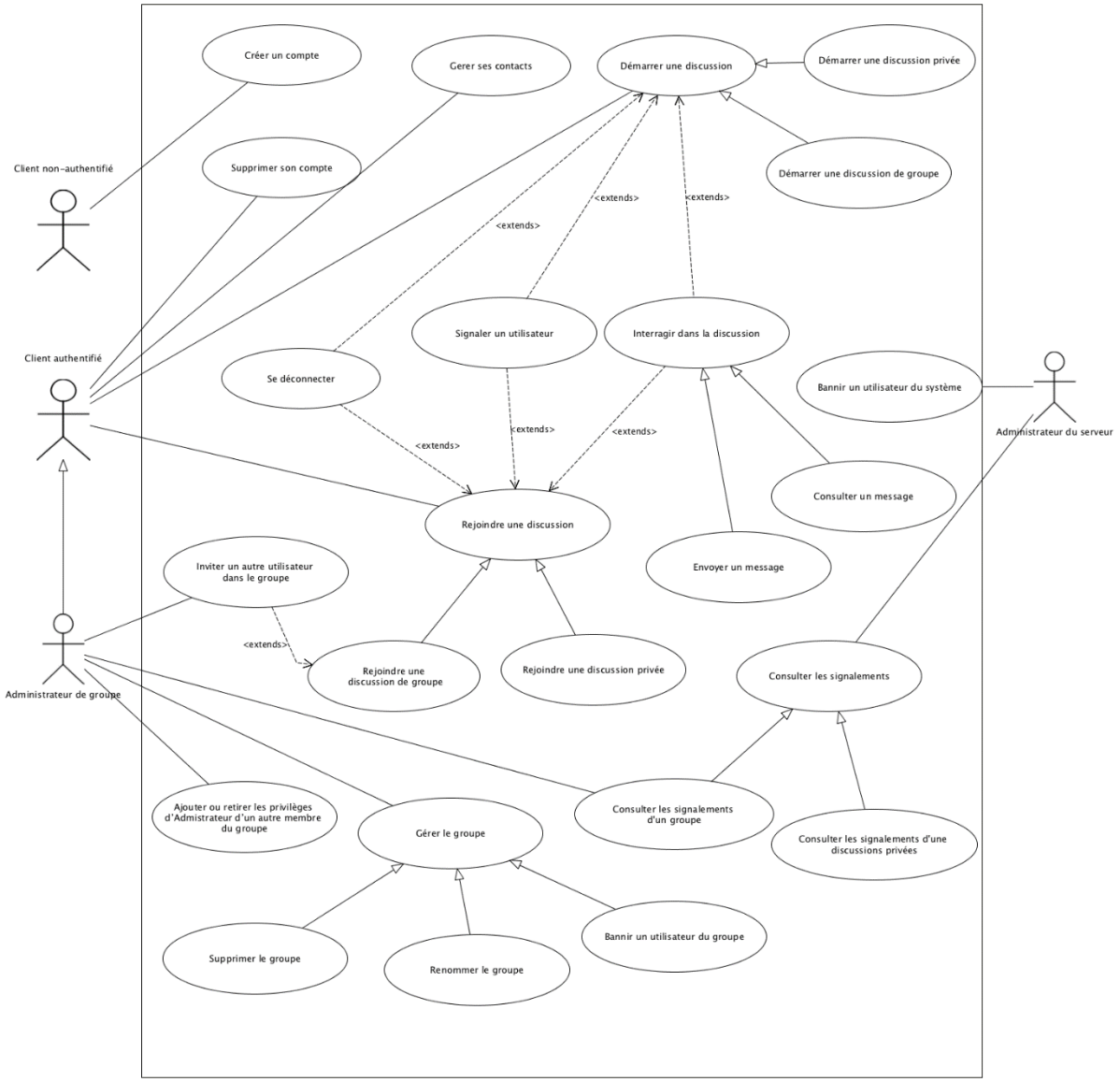

Figure 3 : Diagramme de contexte général

Description des acteurs

Acteurs principaux

- Client non-authentifié
  Il devra soit créer un compte soit s'authentifier pour se transformer en client authentifié.

- Client authentifié
  C'est l'acteur qui représente les utilisateurs qui possèdent un compte et qui sont connectés.

- Administrateur de groupe
  Il est notifié des reports des discussions de son groupe et peut gérer les membres du groupe. Le rôle d'administrateur de groupe étend le rôle de client authentifié.

Acteurs secondaires

- Administrateur du système
  Il est notifié des reports de groupes et de discussions privées et peut bannir des utilisateurs de l'application. Cet acteur n'a pas de rôle de client.

Description des cas d'utilisation

Prérequis : Authentification

Acteur(s) : Client non-authentifié

Description : Un client non-authentifié peut se connecter avec un compte existant

Scénario principal :

1. L'utilisateur ouvre l'application
2. L'utilisateur accède à l'interface de connexion
3. L'utilisateur fournit un pseudonyme et un mot de passe
4. Le serveur accepte les informations de connexion
5. L'application authentifie l'utilisateur

Scénarios d'échec :

a) Le serveur n'est pas disponible : l'application signale l'erreur et l'action est abandonnée
b) Les informations de connexion sont fausses : l'application signale l'erreur et l'action est abandonnée

Ajouter ou retirer les privilèges d'Administrateur d'un autre membre du groupe

Acteur(s) : Administrateur de groupe

Description : Un administrateur de groupe peut promouvoir ou dégrader un autre du groupe du rang d'administrateur de groupe

Scénario principal :

1. L'administrateur de groupe accède à une discussion de groupe qu'il administre
2. L'administrateur affiche la liste des membres du groupe
3. L'administrateur ouvre le menu contextuel du membre du groupe qu'il veut promouvoir ou dégrader
4. L'administrateur choisit « Promouvoir » ou « Dégrader»

Scénarios d'échec :

a. Le serveur n'est pas disponible : l'application signale l'erreur et l'action est abandonnée
b. Le pseudonyme de l'utilisateur à ajouter n'existe pas : l'application signale l'erreur et l'action est abandonnée

Ajouter un utilisateur dans un groupe

Acteur(s) : Administrateur de groupe

Description : Un administrateur de groupe peut ajouter un utilisateur au groupe

Scénario principal :

1. L'administrateur de groupe accède à une discussion de groupe qu'il administre
2. L'administrateur de groupe clique sur "Gérer la discussion"
3. L'administrateur de groupe clique sur "Ajouter un utilisateur"
4. L'administrateur de groupe choisit un de ces contacts enregistrés, ou entre le pseudonyme de l'utilisateur à ajouter
5. Le serveur accepte l'ajout de l'utilisateur

Scénarios d'échec :

a) Le serveur n'est pas disponible : l'application signale l'erreur et l'action est abandonnée
b) Le pseudonyme de l'utilisateur à ajouter n'existe pas : l'application signale l'erreur et l'action est abandonnée

Bannir un utilisateur du groupe

Acteur(s) : Administrateur de groupe

Description : Un administrateur de groupe peut supprimer un utilisateur d'une discussion de groupe

Scénario principal :

1. L'administrateur de groupe accède à une discussion de groupe qu'il administre
2. L'administrateur de groupe clique sur "Gérer le groupe"
3. L'application affiche la liste des membres
4. L'administrateur de groupe sélectionne un utilisateur membre du groupe
5. L'administrateur de groupe clique sur "Supprimer du groupe"
6. L'application demande confirmation
7. Le serveur autorise la suppression de l'utilisateur de la discussion publique
8. L'application met à jour la liste des utilisateurs membres du groupe

Scénarios d'échec :

a. Le serveur n'est pas disponible : l'application signale l'erreur et l'action est abandonnée

Bannir un utilisateur du système

Acteur(s) : Administrateur du serveur

Description : L'administrateur du serveur peut supprimer un utilisateur du système

Scénario principal :

1. L'administrateur du système accède à l'interface de gestion du système
2. L'administrateur clique sur « Bannir un utilisateur»
3. L'administrateur choisit parmi la liste des utilisateurs existants celui à bannir.
4. L'administrateur valide son choix

Scénarios d'échec :

a) Le serveur n'est pas disponible : l'application signale l'erreur et l'action est abandonnée

Charger les messages de l'historique

Description : Un client authentifié peut consulter les messages d'une discussion

Prérequis : Authentification

Scénario principal :

1. L'utilisateur accède à une conversation
2. L'utilisateur remonte (scroll) dans la discussion
3. L'application met automatiquement à jour, au fur et à mesure du scroll, le contenu de la conversation

Scénarios d'échec :

a. Le serveur n'est pas disponible : l'application signale l'erreur et l'action est abandonnée

Créer un groupe de discussion public, consulter les signalements d'un groupe

Acteur(s) : Administrateur du système, Administrateur de groupe

Description : Un administrateur peut consulter les reports d'un groupe

Scénario principal :

1. L'administrateur accède à une discussion de groupe qu'il administre
2. L'administrateur clique sur "Gérer la discussion"
3. L'administrateur clique sur "Voir les signalements"

Scénarios d'échec :

a. Le serveur n'est pas disponible : l'application signale l'erreur et l'action est abandonnée

Consulter les signalements d'une discussion privée

Acteur(s) : Administrateur du système

Description : Un administrateur du système peut consulter les reports d'une discussion privée

Scénario principal :

1. L'administrateur du système accède à l'interface de gestion du système
2. L'administrateur du système clique sur "Voir les signalements"

Scénarios d'échec :

a. Le serveur n'est pas disponible : l'application signale l'erreur et l'action est abandonnée

Consulter un message

Acteur(s) : Client authentifié

Description : Un client authentifié peut consulter les messages d'une discussion

Prérequis : Authentification

Scénario principal :

1. L'utilisateur accède à une conversation
2. L'application met automatiquement à jour le contenu de la conversation

Scénarios d'échec :

a. Le serveur n'est pas disponible : l'application signale l'erreur et l'action est abandonnée

Créer un compte

Acteur(s) : Client non-authentifié

Description : Un client non-authentifié peut créer un nouveau compte

Prérequis : Authentification

Scénario principal :

1. L'utilisateur fournit un pseudonyme et un mot de passe (avec confirmation
2. Le serveur accepte et crée le nouveau compte
3. L'application authentifie l'utilisateur

Scénarios d'échec :

a. Le serveur n'est pas disponible : l'application signale l'erreur et l'action est abandonnée
b. Le compte existe déjà : l'application signale l'erreur et l'action est abandonnée

Démarrer une discussion de groupe

Acteur(s) : Client authentifié

Description : Un client authentifié peut créer une discussion publique

Pré-requis : Authentification

Scénario principal :

1. L'utilisateur clique sur "Créer un groupe"
2. L'utilisateur entre le nom du groupe qu'il veut créer
3. Le serveur autorise l'utilisateur à créer la discussion
4. L'application ouvre la discussion de groupe pour l'utilisateur

Scénarios d'échec :

a. Le serveur n'est pas disponible : l'application signale l'erreur et l'action est abandonnée
b. Un groupe public avec ce nom existe : l'application signale l'erreur et l'action est abandonnée

Démarrer une discussion privée

Acteur(s) : Client authentifié

Description : Un client authentifié peut créer une discussion privée avec un autre utilisateur

Pré-requis : Authentification

Scénario principal :

1. L'utilisateur clique sur "Discussion privée"
2. L'utilisateur choisit un de ces contacts enregistrés, ou entre le pseudonyme de l'utilisateur avec qui il veut avoir une discussion
3. Le serveur autorise l'utilisateur à créer la discussion
4. L'application ouvre la discussion privée pour l'utilisateur

Scénarios d'échec :

a. Le serveur n'est pas disponible : l'application signale l'erreur et l'action est abandonnée
b. Aucun utilisateur avec ce pseudonyme n'existe : l'application signale l'erreur et l'action est abandonnée

Envoyer un message

Acteur(s) : Client authentifié

Description : Un client authentifié peut envoyer un message dans une discussion

Prérequis : Authentification

Scénario principal :

1. L'utilisateur accède à une conversation
2. L'utilisateur saisit un message
3. L'utilisateur clique sur "Envoyer"
4. Le serveur accepte l'envoi du message
5. L'application met à jour le contenu de la discussion pour chaque membre de la discussion

Scénarios d'échec :

a) Le serveur n'est pas disponible : l'application signale l'erreur et l'action est abandonnée

Gérer ses contacts

Acteur(s) : Client authentifié

Description : Un client authentifié peut gérer ses contacts (ajout, suppression, blocage)

Prérequis : Authentification

Scénario principal :

1. L'utilisateur peut cliquer soit sur "Ajouter un contact" ou sélectionner un contact, puis cliquer sur "Supprimer" ou "Bloquer"
   1.1. Si « Ajouter un contact » :
      1.1.1. L'utilisateur entre le pseudonyme d'un utilisateur
      1.1.2. Le serveur accepte l'ajout de contact
   1.2. Si « Supprimer ou Bloquer »:
      1.2.1. L'application demande confirmation
      1.2.2. Le serveur accepte la suppression ou le blocage du contact

Scénarios d'échec :

a. Le serveur n'est pas disponible : l'application signale l'erreur et l'action est abandonnée
b. Le pseudonyme de l'utilisateur à ajouter n'existe pas : l'application signale l'erreur et l'action est abandonnée

Inviter un autre utilisateur dans le groupe

Acteur(s) : Client authentifié

Description : Un client authentifié et membre d'un groupe, peut ajouter un utilisateur dans ce groupe

Prérequis : Authentification, Être membre du groupe

Scénario principal :

1. L'utilisateur clique sur le groupe dans lequel il veut ajouter un utilisateur dans sa liste de discussions
2. L'utilisateur clique sur « Ajouter un utilisateur »
3. L'utilisateur sélection un utilisateur depuis sa liste de contacts
4. L'utilisateur valide son choix

Scénarios d'échec :

a. Le serveur n'est pas disponible : l'application signale l'erreur et l'action est abandonnée
b. L'utilisateur est banni du groupe et ne peut pas être ajouté : l'application signale l'erreur et l'action est abandonnée

Rejoindre une discussion de groupe ou privée

Acteur(s) : Client authentifié

Description : Un client authentifié peut

1. rejoindre une discussion de groupe s'il y a été invité
2. rejoindre une discussion privée

Prérequis : Authentification

Si discussion de groupe :

Un autre utilisateur membre du groupe nous y a invité

Si discussion privée:

Un autre utilisateur a démarré une discussion privée avec nous

Scénario principal :

1. L'utilisateur constate l'apparition d'une nouvelle discussion dans sa liste de discussions
2. L'utilisateur clique sur la discussion pour la rejoindre
3. L'application affiche la discussion

Scénarios d'échec :

a. Le serveur n'est pas disponible : l'application signale l'erreur et l'action est abandonnée

Renommer le groupe

Acteur(s) : Administrateur de groupe

Description : L'Administrateur d'un groupe peut le renommer

Prérequis : Authentification

Scénario principal :

1. L'administrateur choisit le groupe qu'il désire supprimer parmi sa liste de discussions
2. L'administrateur de groupe clique sur "Gérer le groupe"
3. L'administrateur clique sur « Renommer le groupe »
4. L'administrateur saisit le nouveau nom du groupe
5. L'administrateur confirme le changement

Scénarios d'échec :

a. Le serveur n'est pas disponible : l'application signale l'erreur et l'action est abandonnée

Signaler un utilisateur

Acteur(s) : Client authentifié

Description : Un client authentifié peut signaler un autre utilisateur

Prérequis : Authentification

Scénario principal :

1. L'utilisateur sélectionne un ou plusieurs messages
2. L'utilisateur clique sur "Signaler"
3. L'application demande confirmation
4. Le serveur accepte le signalement

Scénarios d'échec :

a. Le serveur n'est pas disponible : l'application signale l'erreur et l'action est abandonnée

Supprimer le groupe

Acteur(s) : Administrateur de groupe

Description : L'Administrateur d'un groupe peut le supprimer

Prérequis : Authentification

Scénario principal :

1. L'administrateur choisit le groupe qu'il désire supprimer parmi sa liste de discussions
2. L'administrateur de groupe clique sur "Gérer le groupe"
3. L'administrateur clique sur « Supprimer le groupe »
4. L'administrateur valide son choix

Scénarios d'échec :

b. Le serveur n'est pas disponible : l'application signale l'erreur et l'action est abandonnée

Supprimer son compte

Acteur(s) : Client authentifié

Description : Un client authentifié peut supprimer son compte

Prérequis : Authentification

Scénario principal :

1. L'utilisateur clique sur "Gérer mon compte"
2. L'utilisateur clique sur "Supprimer mon compte"
3. L'application demande confirmation
4. Le serveur accepte la suppression du compte et supprime les infos du compte présentes sur la base de données
5. L'application déconnecte l'utilisateur

Scénarios d'échec :

c. Le serveur n'est pas disponible : l'application signale l'erreur et l'action est abandonnée

Modèles de domaine

Client

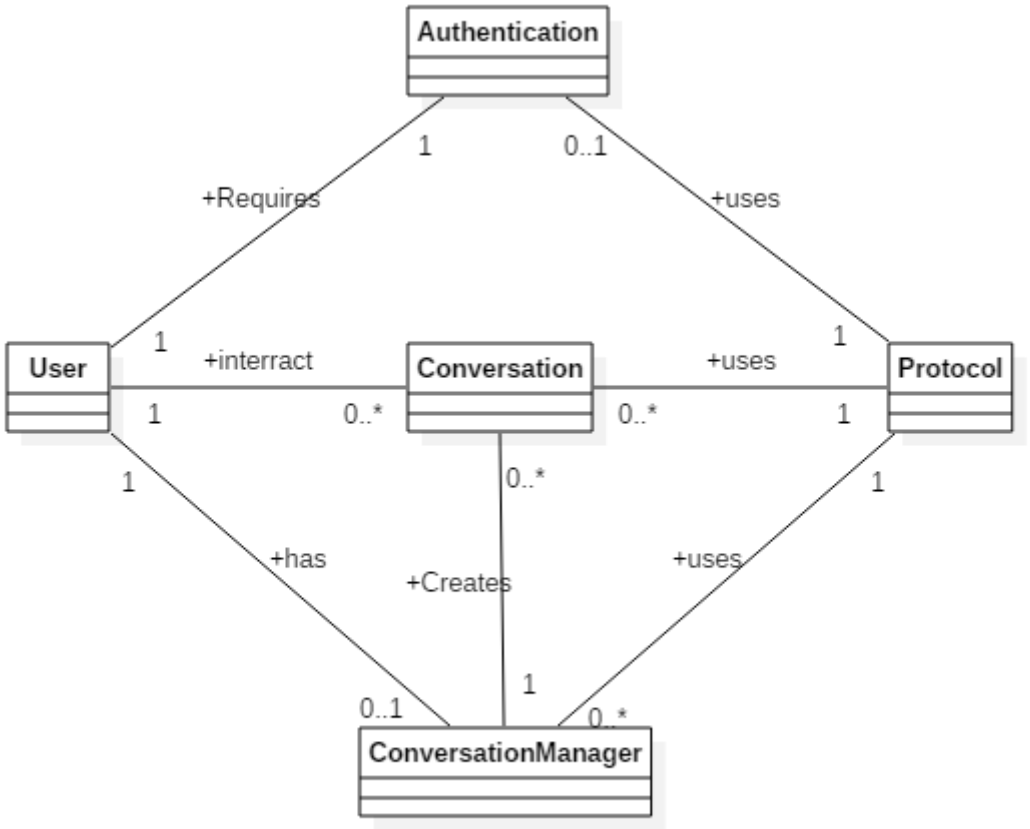

Figure 4 : Modèle de domaine client

Serveur

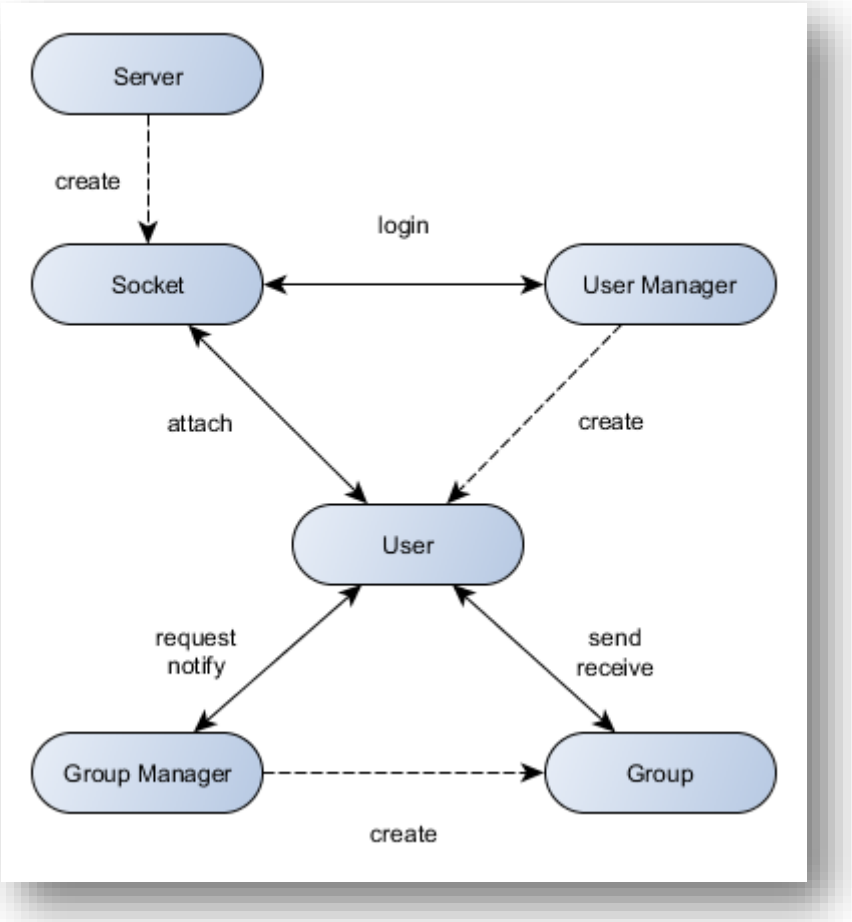

Figure 5 : Modèle de domaine serveur

Base de données

La base de données définit des utilisateurs qui peuvent être membre de conversations. Les conversations privées seront différenciées des conversations de groupe par l'attribut type dans conversation.

Une conversation contient plusieurs messages qui sont liés à un utilisateur et les messages peuvent être reportés. Seuls les administrateurs de groupes (membre dont l'attribut admin est à true) pourront voir les messages reportés.

Un utilisateur peut en bloquer un autre via l'association block.

Modèle conceptuel

Ci-dessous notre modèle conceptuel (ou entité-association) de notre base de données :

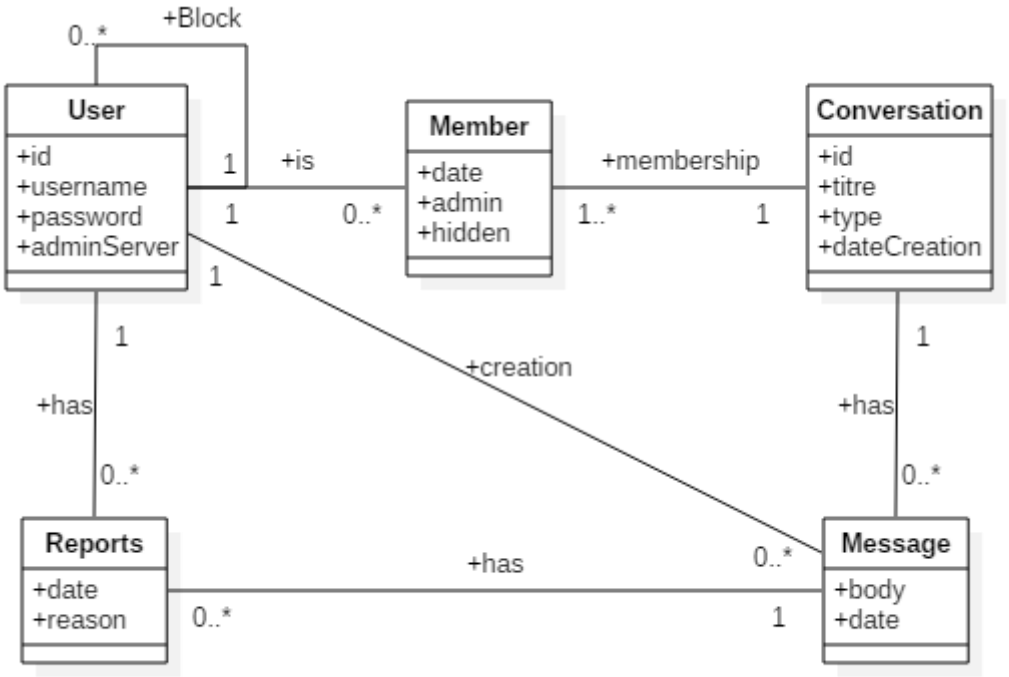

Figure 6 : Modèle conceptuel de la base de données

Conception du projet

Protocole d'échange entre le client et le serveur

Le protocole de communication client-serveur devra permettre les commandes suivantes :

Client -> Serveur

- Demande de création de compte
- Validation/Refus de création de compte
- Demande de tentative de connexion
- Validation/Refus de tentative de connexion
- Demande d'ajout d'un contact
- Validation/Refus d'ajout de contact (par exemple s'il est bloqué ou s'il y a une erreur) (Ajout d'un lien dans les deux sens entre les deux contacts, création de discussions vide)
- Demande de création de groupe
- Validation/Refus de création de groupe
- Envoi d'un message à un contact/groupe
- Validation/Refus d'envoi du message
- Demande de déconnection
- Validation/Refus de déconnection

Serveur -> Client

- Envoi d'un message au client
- Acceptation/Refus de réception
- Création d'un groupe dont le client est membre
- Acceptation/Refus de création

Une classe Java Protocole va définir les différentes commandes. Elle sera utilisée du côté serveur et client. Une commande sera représentée sur 3 bytes. Le premier byte va définir la commande à utiliser. On aura donc 255 commandes différentes possibles ce qui devrait suffire. Les 2 bytes suivants définiront la longueur des messages soit une longueur maximum de 65535. Le message sera situé directement après ces 3 bytes.

Diagrammes de classes

Serveur

Todo : Diagramme + description

Client

Todo : Diagramme + description

Base de données

Modèle relationnel

Todo : Schéma + description

Ebauches des interfaces utilisateurs

Todo : descriptions

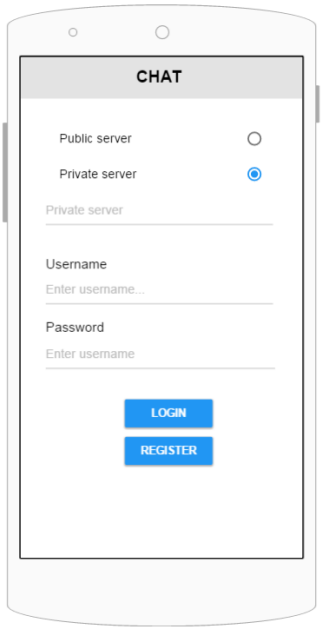

Figure 7 : Concept d'interface pour la fenêtre de login

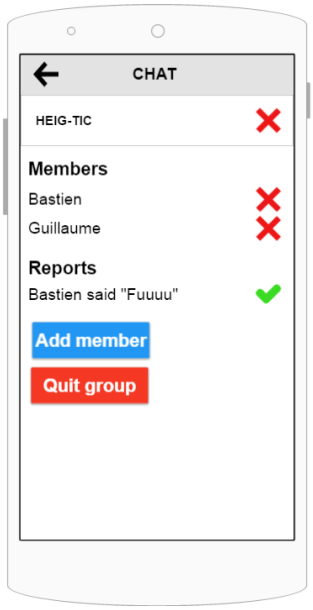

Figure 8 : Concept d'interface pour l'écran principal

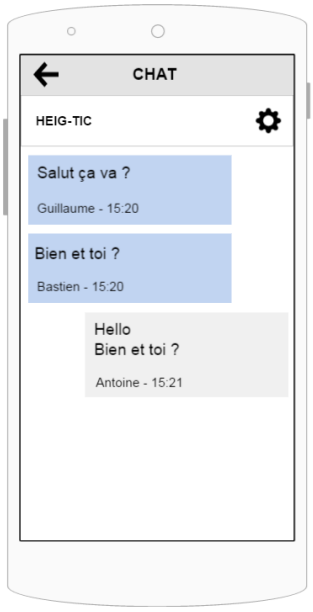

Figure 9 : Concept d'interface pour la fenêtre de discussion

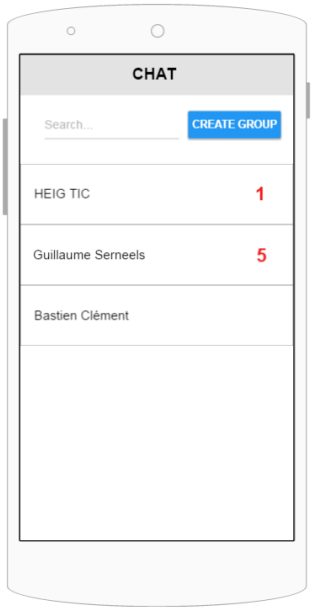

Figure 10 : Concept d'interface pour la fenêtre de gestion de groupe

Implémentation du projet

Technologies utilisées

Todo : justification, langages, bibliothèques spécifiques, leur intérêt, leur utilisation, etc.

Problèmes éventuels rencontrés et solutions apportées

Gestion du projet

Rôle des participants au sein du groupe de développement

Représentant des utilisateurs

Amel Dussier

Responsabilités :

- Collecte des besoins
- Spécification des tests de fonctionnalités
- Explication des aspects métiers

Chef de projet

Amel Dussier

Responsabilités :

- Planification
- Coordination avec les utilisateurs

Analyste

Antoine Drabble

Responsabilités :

- Spécifications
- Collecte des demandes de changement

Software architect

Bastien Clément

Responsabilités :

- Conception de l'architecture du produit

Programmeurs

Bastien Clément, Antoine Drabble & Guillaume Serneels

Responsabilités :

- Participe à la conception du produit
- Ecriture des tests unitaires
- Codage

Responsable des tests

Guillaume Serneels

Responsabilités :

- Participe à l'intégration continue des composants
- Ecrit les tests fonctionnels
- Met en place l'architecture permettant de lancer régulièrement les tests fonctionnels

Responsable de la configuration

Bastien Clément

Responsabilités :

- Gestion de la base des artefacts du projet
- Gestion des releases
- Allocation des droits
- Responsable de la configuration logicielle & matérielle
- Intégration des changements

Plan des itérations initial

Itération 1

Objectif général : Création de la base de données et mise en place d'une première communication simple entre client-serveur

Objectifs détaillés :

- Apprendre les bases du développement Scala et Android (Gestion)
- Mettre en place la base de données (Développement de l'infrastructure)
- Définir une première version du protocole de communication (Gestion : Conception)
- Réaliser une première communication entre le client et le serveur (Développement de l'infrastructure)
    - Cet objectif est un prérequis de tous les cas d'utilisation. A l'issue de l'itération, il sera possible de vérifier qu'un message simple a bien été transmis du client au serveur.

Durée : 2 semaines

Date de début : vendredi 22 avril

Date de fin : vendredi 6 mai (Ascension)

Partage du travail, les heures sont indiquées par semaine :

- Antoine :
    - Apprendre les bases du développement Scala (3h)
    - Définition de la première version du protocole de communication (2h)
- Bastien :
    - Création de la base de données (2h)
    - Mise en place de la communication côté serveur (2h)
- Guillaume :
    - Apprendre les bases du développement Android (3h)
    - Mise en place de la communication côté client (2h)
- Amel :
    - Apprendre les bases du développement Android (3h)
    - Première ébauche de l'interface de connexion (2h)

Temps consacré : environ 19 heures par semaines (38 heures au total)

Itération 2

Objectif général : Mise en place des fonctionnalités de création / suppression de compte et de connexion

Objectifs détaillés :

- Ajouter les fonctionnalités de gestion de compte et de connexion au protocole de communication (Gestion : Conception)
- Interfacer l'application serveur avec la base de données (Développement de l'infrastructure)
- Implémenter la gestion des comptes et de connexion au niveau du serveur et du client (Développement des fonctionnalités)
    - Cas d'utilisation réalisés complètement : Création de compte, Connexion à l'application
    - Cas d'utilisation réalisés partiellement : Supprimer son compte, Se déconnecter
        - Ces fonctionnalités seront implémentées au niveau du code, mais leur intégration à l'interface utilisateur se fera lors des itérations suivantes.
          Il sera néanmoins possible de voir qu'une commande de déconnection est reçue coté serveur.
          Lors d'une suppression de compte, il sera possible de voir qu'une commande de suppression est reçue coté serveur et que les informations du compte sont effectivement supprimées de la base de données.
- Commencer la rédaction du rapport final, avec la structure des chapitres (Gestion : Rédaction)

Durée : 1 semaine

Date de début : mardi 3 mai

Date de fin : lundi 9 mai

Partage du travail :

- Antoine :
    - Ajouter les fonctionnalités de gestion de compte et de connexion au protocole de communication (3h)
    - Interfacer l'application serveur avec la base de données (2h)
- Bastien :
    - Implémenter la gestion des comptes et de connexion au niveau du serveur (5h)
- Guillaume :
    - Implémenter la gestion des comptes et de connexion au niveau du client (5h)
- Amel :
    - Finaliser l'interface utilisateur pour la création de compte et la connexion (3h)
    - Commencer la rédaction du rapport final (2h)

Temps consacré : environ 20 heures (5 heures par personne)

Itération 3

Objectif général : Mise en place des fonctionnalités de recherche et de gestion de contacts

Objectifs détaillés :

- Ajouter les fonctionnalités de recherche et de gestion de contacts au protocole de communication (Gestion : Conception)
- Implémenter la recherche et de gestion de contact au niveau du serveur et du client, ainsi que les interfaces utilisateur correspondantes (Développement des fonctionnalités)
    - Cas d'utilisation réalisés complètement : Gérer les contacts
    - Cas d'utilisation réalisés partiellement : Créer une discussion privée, Ajouter un utilisateur dans un groupe public
        - La fonctionnalité de recherche est utilisée dans plusieurs cas d'utilisation. La réalisation complète de ces cas d'utilisation se fera lors des itérations suivantes.
- Continuer la rédaction du rapport final (Gestion : Rédaction)

Durée : 1 semaine

Date de début : mardi 10 mai

Date de fin : lundi 16 mai

Partage du travail :

- Antoine :
    - Ajouter les fonctionnalités de recherche et de gestion de contact au protocole de communication
    - Continuer la rédaction du rapport
- Bastien :
    - Implémenter la recherche et de gestion de contact au niveau du serveur
- Guillaume :
    - Implémenter la recherche et de gestion de contact au niveau du client
    - Ajout de l'interface de recherche de contact
- Amel :
    - Ajout de l'interface de gestion de contact

Temps consacré : environ 20 heures (5 heures par personne)

Itération 4

Objectif général : Mise en place des discussions privées

Objectifs détaillés :

- Ajouter les fonctionnalités de création de discussion privée, d'envoi de messages et d'historique au protocole de communication (Gestion : Conception)
- Implémenter la création / suppression de discussion, l'envoi de messages et l'affichage de l'historique au niveau du serveur et du client, ainsi que les interfaces utilisateur correspondantes (Développement des fonctionnalités)
    - Cas d'utilisation réalisés complètement : Créer une discussion privée, Envoyer un message, Consulter les messages, Charger les messages de l'historique
    - Cas d'utilisation réalisés partiellement : Créer un groupe de discussion public
        - Une discussion publique est une extension d'une discussion privée, donc certaines fonctionnalités seront identiques. La réalisation complète de ce cas d'utilisation se fera lors des itérations suivantes.
- Continuer la rédaction du rapport final (Gestion : Rédaction)

Durée : 1 semaine

Date de début : mardi 17 mai

Date de fin : lundi 23 mai

Partage du travail :

- Antoine :
    - Ajouter les fonctionnalités de création de discussion privée, d'envoi de messages et d'historique au protocole de communication
    - Implémenter la gestion de l'historique de discussion au niveau du serveur
- Bastien :
    - Implémenter la création / suppression de discussion, l'envoi de messages au niveau du serveur
- Guillaume :
    - Implémenter la création / suppression de discussion, l'envoi de messages et l'affichage de l'historique au niveau du client
    - Continuer la rédaction du rapport
- Amel :
    - Ajout de l'interface de création et d'affichage de discussion
    - Ajout de l'interface de saisie de message

Temps consacré : environ 20 heures (5 heures par personne)

Itération 5

Objectif général : Création de discussion publique (discussion de groupe)

Objectifs détaillés :

- Ajouter les fonctionnalités de discussion publique (créer, supprimer, rejoindre), et de gestion des membres (ajouter, supprimer, promouvoir administrateur de groupe) d'une discussion publique au protocole de communication (Gestion : Conception)
- Implémenter toutes les fonctionnalités relatives aux discussions publiques au niveau du serveur et du client, ainsi que les interfaces utilisateur correspondantes (Développement des fonctionnalités)
    - Cas d'utilisation réalisés complètement : Rejoindre une discussion publique, Créer un groupe de discussion public, Supprimer un utilisateur d'un groupe de discussion public, Quitter un groupe, Ajouter un utilisateur dans un groupe
- Continuer la rédaction du rapport final (Gestion : Rédaction)

Durée : 2 semaines

Date de début : mardi 24 mai

Date de fin : lundi 6 juin

Partage du travail :

- Antoine :
    - Ajouter toutes les fonctionnalités relatives aux discussions publiques au protocole de communication
    - Aider Bastien et Guillaume pour implémenter les fonctionnalités au niveau serveur ou client
- Bastien :
    - Implémenter toutes les fonctionnalités relatives aux discussions publiques au niveau du serveur
- Guillaume :
    - Implémenter toutes les fonctionnalités relatives aux discussions publiques au niveau du client
    - Extension de l'interface d'affichage de discussion privée pour gérer les discussions publiques
- Amel :
    - Ajout de l'interface de gestion de groupes
    - Continuer la rédaction du rapport

Temps consacré : environ 40 heures (5 heures par personne et par semaine)

Itération 6

Objectif général : Signalement et blocage

Objectifs détaillés :

- Ajouter les fonctionnalités de signalement de message et de blocage d'utilisateur au protocole de communication (Gestion : Conception)
- Implémenter toutes les fonctionnalités relatives au signalement de message et de blocage d'utilisateur au niveau du serveur et du client, ainsi que les interfaces utilisateur correspondantes (Développement des fonctionnalités)
    - Cas d'utilisation réalisés complètement : Consulter les signalements d'un groupe public, Consulter les reports d'un groupe privé, Reporter un autre utilisateur
- Terminer la première version du rapport final (Gestion : Rédaction)

Durée : 1 semaine

Date de début : mardi 7 juin

Date de fin : lundi 13 juin

Partage du travail :

- Antoine :
    - Ajouter les fonctionnalités de signalement de message et de blocage d'utilisateur au protocole de communication
    - Terminer la première version du rapport final
- Bastien :
    - Implémenter toutes les fonctionnalités relatives au signalement de message et de blocage d'utilisateur au niveau du serveur
    - Finaliser l'application serveur
- Guillaume :
    - Implémenter toutes les fonctionnalités relatives au signalement de message et de blocage d'utilisateur au niveau du client
    - Finaliser l'application cliente et les différentes interfaces
- Amel :
    - Ajouter les options de signalement et de blocage aux différentes interfaces utilisateur
    - Finaliser les différentes interfaces

Temps consacré : environ 20 heures (5 heures par personne)

Bilan des itérations

Itération 1

Durée
2 semaines

Date de début
vendredi 22 avril

Date de fin
lundi 9 mai

Objectif
Création de la base de données et mise en place d'une première communication simple entre client-serveur.

Avancement
La base de données est créée et le processus de déploiement du logiciel serveur est prêt.

Pas encore de communication client-serveur suite à un changement de plan après discussion avec Jonathan. Nous utiliserons une API REST pour la majorité des opérations.

Nous pensons conserver une interface Socket lorsque l'application est ouverte uniquement pour permettre des notifications push au client Android (au lieu d'utiliser les services GCM). Le protocole du socket est donc grandement simplifié et sera défini lors d'une future itération.

Bilans personnels (heures effectives / heures planifiées)
Antoine (5h/5h)

- Apprendre les bases du développement Scala
    - J'ai mis en place mon environnement de développement. J'ai commencé à apprendre le langage Scala. Je vais devoir également étudier le framework Play que l'on va utiliser pour faire le serveur.
- Définition de la première version du protocole de communication
    - Nous avions d'abord prévu d'utiliser une connexion TCP ainsi qu'un protocole de communication binaire, mais après discussion avec l'assistant, nous allons mettre en place une communication REST en JSON et une communication TCP pour les notifications de type PUSH. Nous devons donc encore en parler avant de tout mettre en place.
    - J'ai également commencé à mettre en place une communication simple du côté client.

Bastien

- Aucun souci particulier à mentionner. La mise en place d'un hook GitHub pour automatiser le déploiement du serveur permettra d'avoir facilement une version « stable » du serveur accessible à tout moment pour le développement de l'application Android.

- La communication côté serveur a été développée avec en tête un protocole entièrement basé sur un socket bidirectionnel. Après discussion, ces fonctionnalités ne seront pas utiles puisque nous pouvons développer une grande partie de l'application en utilisant une API REST, très simple à mettre en œuvre avec Play.

Guillaume (5h/6h)

- Apprendre les bases du développement Android (3h) Début de l'apprentissage d'Android. Création du projet client avec une première Activity. Plusieurs interrogations concernant la gestion des IO sur Android et la mise en place de l'API Rest à clarifier avec l'assistant.
- Mise en place de la communication côté client Pas encore de communication effective coté client (cf. ci-dessus) Mise à jour du rapport (1h)

Amel

- Apprendre les bases du développement Android
    - Installation et configuration d'Android Studio, et d'un « device » pour tester et débugger les applications.
    - Prise en main de l'environnement de développement, de la structure des projets Android (ressources, contrôleurs en Java), etc. Un peu de peine à comprendre certaines notions, comme les fichiers de configuration « gradle » par exemple.
- Première ébauche de l'interface de connexion
    - Première version de l'activité Login, avec le placement de boutons, de champs textes et de boutons radio. Pas mal de recherche pour trouver les attributs nécessaires pour placer correctement les éléments (alignements, espaces entre les éléments, etc.).

Itération 2

Durée
1 semaine

Date de début
mardi 10 mai

Date de fin
lundi 16 mai

Objectif
Mise en place des fonctionnalités de création / suppression de compte et de connexion.

Avancement
-

Bilans personnels (heures effectives / heures planifiées)
Antoine

- Ajouter les fonctionnalités de gestion de compte et de connexion au protocole de communication
    - J'ai refactorisé le client Android et j'ai implémenté les boutons de logins et d'inscription afin qu'ils communiquent avec le serveur.
- Interfacer l'application serveur avec la base de données
    - J'ai créé la base de données et Bastien s'est occupé de l'interfacer avec la base de données.

Bastien

- Une gestion de compte relativement simple est disponible côté serveur. Les opérations de connexion et d'inscription sont disponibles.

Guillaume (5h/5h)

- Implémenter la gestion des comptes et de connexion au niveau du client
    - Envoi d'un register/login.
    - Pas encore de token retourné par le serveur
    - Planification: Si possible discuter vendredi 20 mai avec Jonathan Bischof et Bastien Clément au sujet des IO pour déterminer le protocole à utiliser.

Amel

- Finaliser l'interface utilisateur pour la création de compte et la connexion
    - L'activité de Login a été complétée avec quelques fonctionnalités :
        - l'affichage ou non du champ texte pour le nom du serveur privé est maintenant automatique, selon la sélection des boutons radio

        - un clic sur le bouton d'inscription lance maintenant l'activité de *Subscription*
    - Création de l'activité de Subscription :
        - placement des différents éléments de l'interface
        - mise en place des évènements lors des saisies de texte, par exemple pour vérifier que les champs ne sont pas vides ou que les deux mots de passe sont identiques
        - début de réflexion concernant la validation du nom de l'utilisateur: il faut qu'on se mette d'accord sur le format (commence par une lettre, pas d'espaces, lettres autorisées?) et sur la vérification de doublons (contact avec le serveur pour interdire l'utilisation d'un username déjà existant par exemple)
- Commencer la rédaction du rapport final
    - Pas encore eu le temps de commencer

Itération 3

Durée
1 semaine

Date de début
mardi 17 mai

Date de fin
lundi 23 mai

Objectif
Mise en place des fonctionnalités de recherche et de gestion de contacts

Avancement
Le projet avance bien, le refactoring de l'assistant nous a permis d'avoir un code plus propre côté client mais nous a fait perdre un peu de temps pour la fusion et l'adaptation à la nouvelle architecture. Nous avons un petit retard sur la gestion des contacts mais ça devrait être rapidement rattrapé maintenant que tout est en place.

Du côté serveur tout se passe bien.

Bilans personnels (heures effectives / heures planifiées)
Antoine (9h/5h)

- Ajouter les fonctionnalités de recherche et de gestion de contact au protocole de communication.
    - Comme l'assistant a refactorisé le client et Amel a fait des changements en même temps, j'ai dû fusionner les deux ce qui m'a pris pas mal de temps. J'ai également dû faire refonctionner le login et le register (itération 2).
    - J'ai créé les classes RequestPUT et RequestDelete pour l'envoi de requête HTTP PUT et DELETE.
    - J'ai créé l'activité ContactViewActivity qui permet de voir les messages envoyés avec un contact.
    - J'ai implémenté le bouton suppression d'un contact, mais je n'ai pas encore pu le tester
    - J'ai fait fonctionner la récupération du token pendant l'authentification/enregistrement.
    - J'ai fait fonctionner les fonctions GetToken et SetToken du client.
    - J'ai commencé à récupérer la liste des contacts afin de les afficher.
    - J'ai créé l'activité de recherche d'utilisateurs, mais je n'ai pas encore remplis la liste des utilisateurs.
- Implémenter la recherche et la gestion de client au niveau du client.

- Je n'ai pas eu le temps d'implémenter la recherche et la gestion car j'ai d'abord dû faire fonctionner le login/register et fusionner les deux projets.

Bastien (5h/4h)

- Implémenter la recherche et de gestion de contacts au niveau du serveur
  - La fonctionnalité a été implémentée sans difficulté. L'API REST est entièrement fonctionnelle pour les opérations de gestion de contacts.
  - Le temps supplémentaire relatif à la planification est lié à la mise de mécanisme de traitement d'erreur au niveau du serveur qui n'est pas directement liés à la gestion de contact. Il est maintenant plus aisé de communiquer un échec au client de l'API et le serveur devrait maintenant retourner les exceptions non-attrapées au consommateur de l'API en format JSON.
  - Par la suite, il sera possible de se baser sur le statut administrateur du client pour déterminer si l'exception doit ou non être détaillée.

Guillaume (4h/5h)

- Implémenter la gestion des comptes et de connexion au niveau du client
  - Compréhension et intégration du client v2 refactorisé par l'assistant.
  - Login / obtention du token fonctionnel
- Implémenter la recherche et la gestion de contact au niveau du client
  - Recherche fonctionnelle sur liste de contacts codés en dur
  - Suite à l'intégration du client v2, pas encore pu
- Ajout de l'interface de recherche de contact
  - Interface de recherche fonctionnelle

Amel (5h/5h)

- Ajout de l'interface de recherche de contact
  - Création d'une interface pour lister les contacts de l'utilisateur
  - Possibilité de faire une recherche avec un widget SearchView (passé un peu de temps à comprendre comment configurer la recherche avec un Adapter, et comment personnaliser l'affichage)
  - La sélection d'un contact permet d'accéder à l'interface de gestion du contact
  - Pas encore réussi à charger la liste de contacts directement depuis le serveur
- Ajout de l'interface de gestion de contact
  - Création d'une première version de l'interface de gestion d'un contact
  - Récupère le contact passé en paramètre depuis l'activité précédente
  - Pour l'instant l'interface contient seulement un bouton pour supprimer le contact (fonctionnel)

Itération 4

Durée
1 semaine

Date de début
mardi 24 mai

Date de fin
lundi 30 mai

Objectif
Mise en place des discussions privées

Avancement
Création d'une discussion privée coté client.

Bilans personnels (heures effectives / heures planifiées)
Antoine (9h/5h)

J'ai pu rattraper le retard de l'itération précédente. J'ai fait fonctionner la recherche et l'ajout de contacts, l'affichage de la liste des contacts, la suppression d'un contact et les boutons de retour en arrière. (4h)

J'ai également ajouté l'affichage des erreurs dans le client. (30min)

J'ai commencé à mettre en place la vue de discussion 1 à 1 avec Guillaume, et nous avons préparé l'adapter qui sera utilisé pour l'affichage de la conversation. (3h)

J'ai implémenté la réception des messages depuis le serveur et leur affichage dans la fenêtre de discussion que j'ai renommé en ContactDiscussionActivity. (1h)

J'ai implémenté le bouton d'envoi pour que les messages soient bien envoyés sur le serveur. (30min)

- Ajouter les fonctionnalités de création de discussion privée, d'envoi de messages et d'historique au protocole de communication.
    - Les messages fonctionnent mais il reste les notifications à faire
- Implémenter la gestion de l'historique de discussion au niveau du serveur.
    - Bastien s'en est occupé.

Bastien (7h/5h)

- Quelques mises à jour des API de l'itération 3
- Changement de la gestion des contacts côté serveur, au lieu d'avoir une paire d'entrée dans une table, il n'y a plus qu'une entrée avec un ordre précis des contacts
- Mise en place du système de push d'événement en utilisant la technique du long-polling HTTP
- Mise en place de l'API pour la gestion des discussions privées et des événements associés
- Mise en place de la gestion des messages lus / non-lus et des événements associés

- Amélioration du processus de mise à jour du serveur en utilisant Docker

Guillaume (5h/5h)

- Implémenter la création / suppression de discussion, l'envoi de messages et l'affichage de l'historique au niveau du client
    - Modification de ContactViewActivity, qui permet de modéliser une discussion contenant la liste de messages passé
    - La suppression de discussion correspond à la suppression du contact
    - En attente de l'implémentation des messages cotés serveur
- Continuer la rédaction du rapport
    - Ajout d'un fichier planification_iterations.md sur le git, avec la planification mise en forme. Mise à jour de bilan_iteration.md avec les dates.

Amel (5h/5h)

- Ajout de l'interface de création et d'affichage de discussion
    - Modifications de l'activité ContactViewActivity
    - Ajout d'un bouton "Envoyer message" dans l'interface de gestion de contact
- Ajout de l'interface de saisie de message
    - Intégrée à l'interface de saisie de message
- Correction de bugs dans l'interface de recherche de contacts

Itération 5

Durée
1 semaine

Date de début
mardi 31 mai

Date de fin
lundi 6 juin

Objectif
-

Avancement
-

Bilans personnels (heures effectives / heures planifiées)
Antoine (33h/5h)

J'ai dû refactoriser le code car on n'enregistrait pas la liste des utilisateurs et des messages et pour des questions de performance, il fallait la stocker dans l'application. J'ai également généré toute la javadoc des classes du client. 3h

J'ai mis la fonctionnalité du bouton de retour dans le fichier manifest comme l'assistant m'avait dit de faire mais ça change le comportement et du coup j'ai du rollback. (1h)

J'ai commencé à chercher comment faire le système de notification depuis Android. (Faire une classe qui permette de mettre à jour les adapteurs des différentes vues en allant chercher les évènements sur le serveur). J'ai trouvé une façon pas très propre et je cherche mieux. (3h)

J'ai changé la façon de faire la gestion des évènements en mettant en place un service comme vu avec Jonathan. (2h)

J'ai dû résoudre le problème que RequestGET est un task et pas un thread et bloque l'exécution des autres requêtes quand on veut récupérer les évènements. (2h)

J'ai fait marcher les évènements et la récupération du JSON. J'ai fait que la recherche n'affiche pas les utilisateurs déjà en contact et soi-même. (1h)

J'ai fait marcher les évènements d'ajout et suppression de contact. Et les évènements de nouveaux messages. (2h)

J'ai fait marcher le système de notification pour afficher une notification dans la liste des contacts quand un nouveau message a été reçu. (1h30)

On a eu le cours où on a fait la démo et avancer sur les évènements et la vue de création de groupe. (4h)

J'ai refactorisé un peu le code après les changements fait par Jonathan et j'ai résolu un problème avec les évènements. (1h)

J'ai ajouté les vues discussion de groupe et édition de groupe. J'ai implémenté l'envoi de message, la réception de message, la création de groupe, l'affichage de la liste des groupes... (4h)

J'ai ajouté le tri des contacts/groupes par dernier message reçu et j'ai ajouté le bouton de déconnection. (1h)

J'ai résolu quelques problèmes notamment la mise à jour des listes des fragments. (1h)

J'ai résolu des problèmes d'évènements (création de groupe et messages non lus). (1h)

Afficher la liste des membres d'un groupe et les boutons de suppression de membre. (1h30)

Ajouter la suppression et l'ajout de membres dans un groupe. Permettre de supprimer/quitter le groupe. Les fonctionnalités dépendent de si on est admin ou pas. (4h)

- Ajouter toutes les fonctionnalités relatives aux discussions publiques au protocole de communication.
    - J'ai ajouté une bonne partie des fonctionnalités de discussion publique.
- Aider Bastien et Guillaume pour implémenter les fonctionnalités au niveau serveur ou client.
    - Bastien c'est occupé de la partie serveur (je lui ai communiqué les quelques bugs).

Bastien (TODO/TODO)

- TODO

Guillaume (7h/5h)

- Ajout dans l'activité CreateGroup de l'affichage de la liste des contacts avec des cases à cocher. (2h)
- Refactorisation de l'activité principale sous la forme de deux fragments (contact/groupes) afin de pouvoir les afficher sous la forme d'ongles facilement navigables, avec l'aide de l'assistant. (3h)
- pour implémentation de l'onglet GroupFragment: création de l'adaptateur pour l'affichage des groupes et du modèle groupe. (2h)

Amel (5h/5h)

- Rédaction du rapport final
    - Définition de la structure de base
    - Intégration et mise en page du contenu du rapport intermédiaire
    - Complété la partie "Analyse"
    - Création du diagramme d'activité général
    - Premier état des lieux pour les illustrations manquantes et les sujets à éclaircir avec l'ensemble du groupe

Itération 6

Durée

Date de début

Date de fin

Objectif

Avancement

Bilans personnels (heures effectives / heures planifiées)

Stratégie de tests

Todo : Guillaume fait une petite description pour les tests unitaires

Stratégie d'intégration du code

On a utilisé GIT

Création de branche pour chaque nouvelle fonctionnalité

Commits fréquents

Etc.

Etat des lieux

Ce qui fonctionne

(Résultats des tests)

Ce qu'il resterait à développer

(en proposant une planification)

Autocritique

Relativement à votre solution technique, votre gestion de projet, votre plan d'itération

Ce que vous auriez pu améliorer et comment

Todo : améliorations possibles

- Cryptage
- Notifications
- Client IOS et Windows Phone
- Client Web
- Etc.

Conclusion

… c’était trop cool

Annexe

Table des illustrations

Manuel d'utilisation

Installation

Serveur + client

Utilisation

Todo : captures d'écran