UE Projet

Cahier des charges techniques
BADHOC

badhoc

Réalisé par :

Marie AMARU

Mouncef NAJI

Roumaissa SOUKEHAL

Encadré par :

Abderrezak RACHEDI

Aurélien CHAMBON

I. Contexte

Il existe parfois des situations où il est nécessaire de pouvoir communiquer avec des personnes autour de nous même lorsqu'une connexion Internet n'est pas disponible pour tout le monde. Dans certains lieux où obtenir une connexion Internet de qualité est compliqué, il peut être utile de pouvoir communiquer sans que chaque appareil soit relié à Internet. D'abord née d'un besoin de confidentialité entre les communicants, la communication en réseau local est aujourd'hui toujours utile. En effet, si les appareils communiquent à une portée courte grâce à un signal Bluetooth, la possibilité d'intercepter les communications est réduite. Il peut être pertinent de pouvoir communiquer dans des situations où une connexion n'est pas disponible : coupure d'électricité, concerts, lieux sinistrés, zones hors de portée de données cellulaires... C'est dans ce contexte que le projet badhoc trouve son utilité.

II. Outils de développement

A. Critères à évaluer pour la sélection

Afin de choisir quel outil serait le plus adapté pour nos besoins, il a fallu comparer plusieurs critères pour faire nos choix parmi les options que nous possédions.

- **la licence**

La licence d'un logiciel détermine le propriétaire de ce logiciel et donc l'accès à son utilisation. Il a donc fallu choisir des logiciels libres de droits.

- **la communauté**

La communauté d'un logiciel influe sur la disponibilité d'informations et de solutions lors de son utilisation. La possibilité d'accéder à des retours d'utilisateurs sur certains cas d'usages est un point positif.

- **la maintenance**

La maintenance d'un logiciel réfère à la fréquence à laquelle est mis à jour. Un logiciel bien maintenu sera corrigé plus fréquemment et présentera moins de bugs.

- **la facilité d'apprentissage**

Il est important que si un membre de l'équipe ne maîtrisait pas complètement une technologie, il puisse se mettre à niveau rapidement et qu'un accès à de l'information et de la documentation soit facilement disponible.

- **la connaissance des membres**

Afin d'être efficace dans la phase de développement, c'est un plus si les membres maîtrisent déjà les technologies employées.

B. Environnement de développement

Notre produit étant une application Android, il a fallu choisir l'IDE le plus adapté au développement d'applications Android.

Android Studio

Android Studio est un IDE sorti en 2013 qui est aujourd'hui la solution privilégiée pour le développement d'applications Android.

Il est open source et a gagné en popularité grâce à son interface idéale pour le développement d'applications mobiles. Il se base sur l'environnement IntelliJ IDEA, et possède des fonctionnalités très pratiques telles que la visualisation grâce à l'émulateur intégré.

Eclipse

Eclipse est un IDE open source. Il est nécessaire d'installer plusieurs utilitaires et plugins afin de développer des applications Android. Initialement utilisé comme outil de préférence pour le développement Android, il a cessé d'être mis à jour à partir de 2015 suite à l'évolution de l'environnement Android Studio.

	Open Source	**Communauté**	**Maintenance**	**Facilité d'apprentissage**	**Connaissance des membres**
Android Studio	oui	**	***	***	***
Eclipse	oui	***	*	**	**

Notre choix s'est porté sur l'IDE Android Studio. En effet ce choix s'est fait comme le plus naturel, Android Studio étant l'IDE le plus adapté pour nos besoins. C'est également l'IDE sur lequel les membres de l'équipe ont appris à développer des applications Android dans le cadre universitaire.

C. Gestion de projet

Notion

Notion est une plateforme web de collaboration avec prise en charge de markdown, de kanban, de tâches, des wikis et des bases de données. C'est un outil de gestion de fichiers offrant un espace de travail unifié, permettant aux utilisateurs de commenter les projets en cours, de participer à des discussions et de recevoir des commentaires.

Trello

Trello est un outil de gestion de projet en ligne qui repose sur une organisation des projets en planches listant des cartes, chacune représentant des tâches. Les cartes sont assignables à des utilisateurs et permettent de suivre l'avancement de chaque tâche. La version de base est gratuite, mais il existe une formule payante qui permet d'obtenir des services supplémentaires.groupe pour le développement en langage Java (IDE IntelliJ), cela est un plus. Enfin, c'est l'IDE le plus à jour et qui rend l'utilisation du SDK Android la plus intuitive possible.

	Open Source	**Communauté**	**Maintenance**	**Facilité d'apprentissage**	**Connaissance des membres**
Notion	oui	**	**	***	***
Trello	oui	***	**	***	**

Nous avons choisi Notion pour la gestion de notre projet. Les deux outils considérés étant très pratiques, notre choix a été motivé par le fait que Notion possède plus de possibilités de formats, et une structure plus proche d'un wiki.

D. Système d'intégration continu permettant l'utilisation de tests

Le besoin de l'équipe est un outil permettant d'effectuer une intégration continue du code en surveillant la non régression de ces features au cours de la phase de développement. La vérification des tests est donc également primordiale.

Github Actions

GitHub Actions permet l'automatisation de workflows logiciels. Il permet de tester et déployer du code directement depuis GitHub. Étant directement lié à GitHub, dans le cas où Git est la plateforme choisie pour la gestion du projet, cela est un avantage car la gestion du système d'intégration continue est fortement facilitée.

Jenkins

Jenkins est un outil open source de serveur d'automatisation. Il aide à automatiser les parties du développement logiciel liées au build, aux tests et au déploiement, et facilite l'intégration continue.

	Open Source	**Communauté**	**Maintenance**	**Facilité d'apprentissage**	**Connaissance des membres**
GitHub Actions	oui	**	***	**	***
Jenkins	oui	***	**	**	*

Notre choix s'est porté sur le système GitHub Actions. Les deux logiciels disponibles pour notre projet étaient adaptés à nos besoins, cependant le choix s'est fait sur notre familiarité avec GitHub Actions. Marie Amaru ayant travaillé avec ce système de développement continu au cours de son travail en alternance, il semblait logique d'utiliser cette solution afin de gagner du temps sur la découverte d'un nouveau logiciel au cours du développement du projet.

E. Logiciel de gestion de version

Git

GIT est un logiciel de gestion de version décentralisé. Ce logiciel est proposé gratuitement avec une version Community et une version Entreprise qui est payante. Ce logiciel est sous licence publique générale, ce qui lui permet d'être toujours à jour.

SVN

Subversion (en abrégé svn) est un logiciel de gestion de versions, distribué sous licence Apache. Il repose sur un système de versionning centralisé. Cela signifie qu'un seul répertoire général existe et tous les utilisateurs y ont accès. Les modifications ne pouvant être fusionnées, le système empêche que deux utilisateurs modifient un même fichier simultanément.

	Open Source	**Communauté**	**Maintenance**	**Facilité d'apprentissage**	**Connaissance des membres**
Git	oui	***	***	**	***
SVN	oui	**	**	**	*

Nous avons choisi Git pour versionner notre projet. C'est en effet le logiciel de versionning le plus courant et le plus utilisé. Il s'agit également du logiciel le plus utilisé dans des projets précédents des membres de l'équipe et au sein d'expériences professionnelles. De plus, avec l'utilisation des workflows GitHub Actions, l'utilisation de Git semble logique.

III. Analyse des besoins techniques

A. Langage de programmation

Langage de programmation	**Qualités**	**Inconvénients**
Java	- Contient de multiples librairies disponibles. - Utilisé dans la plupart des solutions. - Open source. - Facilité d'assemblage de plusieurs projets.	- Compliqué en comparaison avec les autres langages. - Plus verbeux. - Problèmes avec les API d'android (limites d'héritage) - Plus lent en exécution. - Requiert plus de mémoire.
Kotlin	- Supporté par la plupart des librairies Android Jetpack - Moins verbeux.	- Lent en compilation. - Ressources d'apprentissage limitées. - Syntaxe concise.
Javascript (react native)	- Pour différentes plateformes. - Populaire.	- N'est pas native. - Choix de composants limités.

	- Rafraichissement rapide.	- Un UI fragile. - Applications plus lourdes que les natives.
Dart (Flutter)	- Pour différentes plateformes. - Exécution des applications plus rapides.	- Pas si riche en librairies. - Compilation et déploiement non automatique. - Lourd.

B. Format des données

Format de données	**Qualités**	**Inconvénients**
Json	- Basé sur javascript. - Supporte les listes	- Ne supporte pas les namespaces. - moins sécurisé. - supporte l'encodage UTF-8 seulement.
XML	- Basé sur SGML - Représente des data items. - supporte les namespaces. - plus sécurisé. - supporte différents encodages.	- Ne supporte pas les listes. - Difficile à interpréter.

C. Protocoles de communication

Criteria	Zigbee	Z-Wave	WiFi	Bluetooth
Range	Bonne grâce à l'héritage du mesh networking	Bonne grâce à l'héritage du mesh networking	Bonne si on utilise des répéteurs ou bien WiFi mesh	Moins bonne
Power Use (in theory)	Bas	Bas	Elevé	Bas
Bandwidth	Médiocre	Médiocre	Excellent	Médiocre
RF Band	2.4 GHz	908.42 MHz	2.4 GHz/5 GHz	2.4 GHz
Needs hub?	Oui	Oui	Non (router)	Non (smartphones)
# of smart devices	Modéré	Pas beaucoup (à part les capteurs)	Beaucoup	Presque aucun
Price of smart devices	Elevé	Elevé	Bas	Moyen
Part of CHIP?	Oui	Non	Oui	Oui

D. Protocoles de communication

	MQTT	**COAP**	**XMPP**	**DDS**
	Message Queuing Telemetry Transport	Contraint Application Protocol	Extensible Messaging and Presence Protocol	Data Distribution Service
Standard	OASIS	IETF	OMG	IETF
Modèle de communication	Publication/abonnement	Requête/réponse	Publication/abonnement	Publication/abonnement
Protocole Transport	TCP	UDP	TCP	TCP
Applications sous NDN	Le réseau d'information véhiculaire IoT	NDN-ACE, COAP observe sur ICN	La messagerie instantané, chat multi-utilisateurs sans serveur Chronos	IOT robotique.

E. SDK pour la fonction de messagerie

Il existe plusieurs applications qui proposent la messagerie par bluetooth:

- Briar (https://briarproject.org/)
- Bridgefy (https://bridgefy.me/sdk/)
- Serval mesh (http://www.servalproject.org/)
- Firechat
- ...

Cependant Bridgefy reste la seule solution proposant l'utilisation du SDK en open source.

F. Choix de développement pour notre projet

Conformément aux critères et besoins définis précédemment pour le projet, nous optons pour les différents choix suivants :

- Langage de programmation : Java.
- Format de données : Json.
- Communication entre les smartphones : Bluetooth.
- Communication avec le serveur : Wifi.
- SDK pour la messagerie : Bridgefy.

IV. Architecture

A. Architecture globale du système

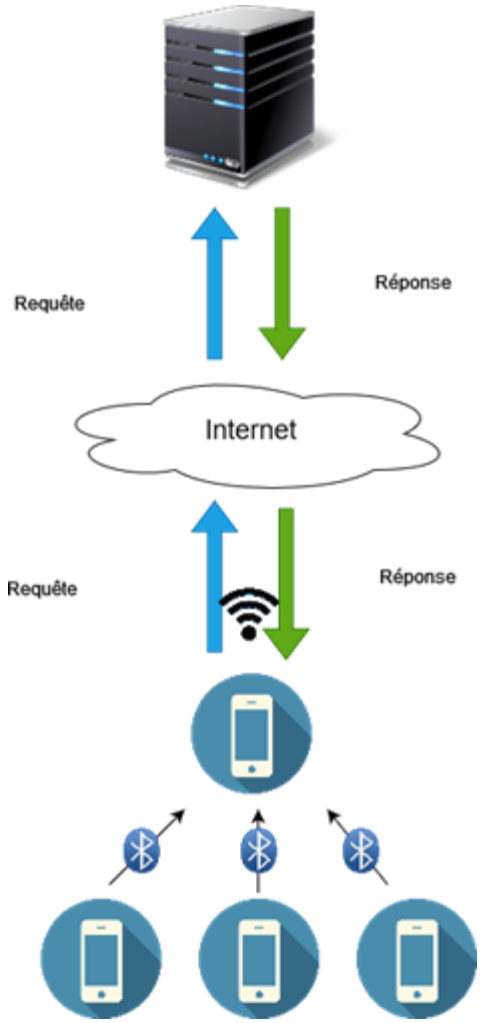

Figure 1 : Architecture du système à développer

B. Use cases génériques

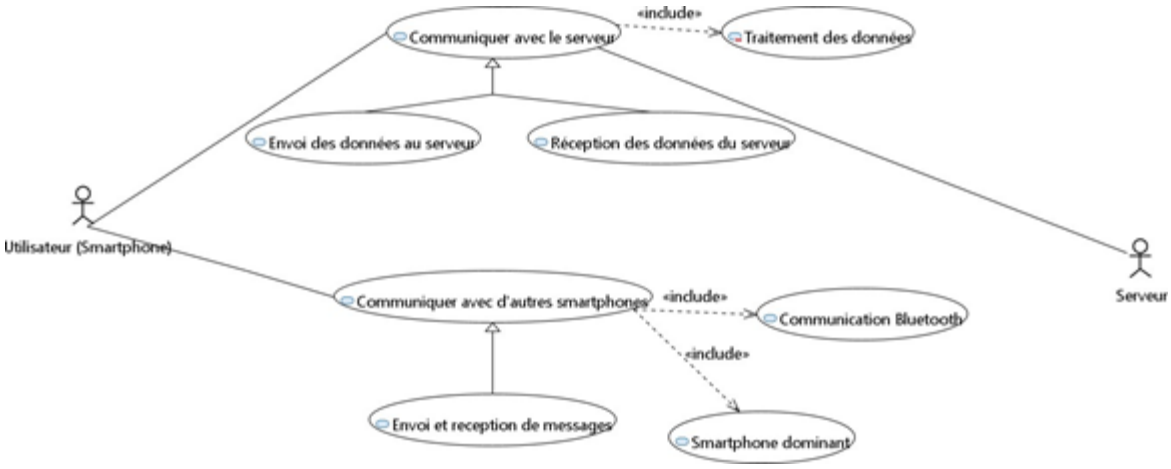

Figure 2: Diagramme de cas d'utilisation du système.

C. Présentation des différents paquets

- **Package activity**
    - **Classe MainActivity** : Cette classe est l'activité principale de l'application, elle articule les fragments entre eux et gère le client Bridgefy, ainsi que le service de connexion au serveur.
- Package adapter
    - **Classe MessagesBadhocAdapter** : Cette classe gère l'affichage des listes de messages dans les interfaces de chat : si le message est entrant ou sortant, si c'est une image ou du texte. Elle héberge les

listes de messages en fonction du type de conversation : privée ou publique.

- **Classe NeighborsAdapter** : Cette classe gère l'affichage des appareils alentour connectés : affichage du nom du voisin, et l'icône de couleur en fonction de sa disponibilité. Elle contient la liste de voisins de l'appareil.
- **Classe NotificationAdapter** : Cette classe gère l'affichage des notifications reçues par le serveur, ainsi que des messages envoyés vers le serveur. Elle contient la liste des notifications reçues et envoyées au serveur.

- Package fragment
  - **Classe AroundMeFragment** : Cette classe implémente le fragment qui représente l'onglet AroundMe.
  - **Classe BroadcastChatFragment** : Cette classe implémente le fragment qui représente l'onglet Broadcast.
  - **Classe NotificationFragment** : Cette classe implémente le fragment qui représente l'onglet Notification.
  - **Classe PrivateChatFragment** : Cette classe implémente le fragment qui représente la page de discussion privée lorsque l'on clique sur un utilisateur depuis l'onglet Around Me.
- Package listener
  - **Interface ItemClickListener** : Cette interface permet de rendre cliquable le nom des utilisateurs dans l'onglet Around Me.

- **Classe MessageListenerImpl** : Cette classe implémente l'interface MessageListener du sdk Bridgefy et permet de gérer la réception de messages selon leur type. (public, privé)
- **Classe StateListenerImpl** : Cette classe implémente l'interface StateListener du sdk Bridgefy qui permet de gérer le démarrage du client Bridgefy ainsi que la détection d'utilisateurs alentour.

- **Package model**
  - **Classe FromServerNotification** : Cette classe représente une notification en provenance du serveur.
  - **Classe MessageBadhoc** : Cette classe représente un message avec les informations nécessaires à l'affichage des messages dans les fragments de conversations
  - **Classe Neighbor** : Cette classe représente un nœud mais dans un format simplifié, ne contenant que les informations pertinentes qu'un nœud doit avoir sur ses voisins proches.
  - **Classe Node** : Cette classe représente un nœud dans son format le plus complet.
  - **Classe NotificationDisplay** : Cette classe représente une notification reçue par soit le serveur dans le cas d'un nœud dominant, soit par le nœud dominant du réseau dans le cas d'un nœud dominé. Il sert à afficher correctement le contenu de la notification dans le fragment NotificationFragment.
  - **Classe Status** : Cette énumération représente les deux statuts qu'un nœud peut avoir : dominant ou dominé.

- **Classe Tag** : Cette énumération contient toutes les clés associées à des chaînes de caractère. Les clés sont liées soit à l'envoi de broadcast dans le contexte de la communication entre l'activité, les fragments et le service, soit dans les payloads personnalisées envoyées dans les messages privés ou publics entre les appareils.
- **Classe ToServerNotification** : Cette classe représente une notification à destination du serveur.

- Package serializer
  - **Classe NeighborDominatingSerializer** : Cette classe sert à sérialiser au bon format le contenu de la classe Neighbor pour que le message au format JSON ait le format attendu par le serveur.

- Package service
  - **Classe LocationService** : Cette classe implémente le service qui gère la localisation de l'appareil et récupère ses informations (position et vitesse)
  - **Classe ServerService** : Cette classe gère la connexion au serveur et les interactions avec ce dernier. Elle gère les pertes de connexion et les messages reçus et envoyés vers le serveur.

- Package util
  - **Classe DeviceUtil** : Cette classe implémente toutes les méthodes liées à la récupération des informations techniques liées à l'appareil. (adresse MAC, signal LTE...)
  - **Classe ParserUtil** : Cette classe permet de formater les messages aux formats attendus par leur destinataire, en fonction des topics.

D. Diagramme de classe