République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université AMO de Bouira

Faculté des Sciences et des Sciences Appliquées

Département d'Informatique

Mémoire de Master

en Informatique

Spécialité : *ISIL*

Thème

Synchronisation de plusieurs bases de données (locales et distantes)

Encadré par

— DJOUABRI Abderrezak

Réalisé par

— CHERARAK Bekhi

— AMLIK Manaf

2019/2020

Remerciements

Nous remercions avant tout le bon Dieu le tout puissant qui nous a donné la santé , le courage, la force, la volonté et la patience pour terminer ce modeste travail.

Nos plus vifs remerciements s'adressent à nos familles qui nous ont soutenus beaucoup pendant toute la vie et qui continuerons à nous aider dans tous les projets de l'avenir.

Nous remercions très sincèrement notre promoteur, Monsieur**A.DJOUABRI** pour ses conseils, son orientation et son aide et d'avoir accepter de nous encadrer.

Nous remercions particulièrement avec gratitude tous les membres du jury qui nous ont fait l'honneur d'examiner et d'évaluer notre travail.

Nous remercions aussi tous nos amis et camarades qui nous ont soutenu et tous ceux qui ont contribué de près ou de loin à la réalisation de ce travail.

Nous remercions du fond de cœur à tous le monde.

Dédicaces

C'est avec un très grand honneur que je dédie ce modeste travail aux personne les plus chères au monde après mon Dieu pour leur sacrifices :

A mon père qui ma donné tout ce qu'il possède jusqu'à ce que je réalise ses espoirs pour lui, et qu'il me poussait vers la réalisation ce qui est désiré, et qui a pris soin de mon éducation avec de grands sacrifices traduit dans sa sanctification de la science.

A ma mère qui ma donné le frisson de son cœur de toute générosité et tendresse, et qui m'ont toujours encouragé et Soutenu dans mes études pour continuer La voie de la science et du succès.

A mes sœurs et mes frère .

A mes amies intimes .

A toute ma famille sans exception.

Cherarak Bekhi.

Dédicaces

Je dédie ce travail à mes très chers parents.

A ma mère qui se cesse jamais de m'encourager.

A mon père qui était toujours à mes coté à tout moment.

A mes frères et mes sœurs pour leur amour et leur soutien inconditionnel.

A toute ma famille et tous mes amis où qu'ils soient.

Amlik Manaf

Résumé

Avec l'émergences des moyens de communications intelligents, tel que les smartphones, les notebooks et autres. La disponibilité des services (en ligne et hors ligne) devient un facteur concurrentiel entre les prestataires de services. Ainsi, les algorithmes de synchronisation apparus afin de mettre à jour les clients qui ont des nouvelles et veulent obtenir les nouvelles.

Ce travail vise à proposer une solution permettant de synchroniser et répliquer en même temps plusieurs bases de données distantes en utilisant l'approche centralisée.

Pour cela nous faisons une comparaison entre certains algorithmes de synchronisation. Ensuite nous proposant un algorithme simple, claire et facile à implémenter qui assure une synchronisation et la réplication d'une base de données entre plusieurs clients en mode unidirectionnel et bidirectionnel.

Pour valider et tester l'algorithme proposé, nous utilisons une base de données MySQL hébergée dans un serveur centralisé Apache. Nous considérons plusieurs clients Android et les scriptes PHP portés par le protocole http comme support d'échange.

Mots clés : Synchronisation, Base de données, Réplication, Services Web, Applications Mobiles.

Abstract

With the emergence of intelligent means of communication, such as smartphones, notebooks and others. The availability of services (online and offline) is becoming a competitive factor among service providers. Thus, synchronization algorithms appeared in order to update customers who have news and want to get the news.

This work aims to propose a solution allowing to synchronize and duplicate at the same time several remote databases using the centralized approach.

For this we make a comparison between some synchronization algorithms. Then offering us a simple, clear and easy to implement algorithm. Which provides synchronization and replicationn of a database between multiple clients in one-way and two-way mode.

To validate and test the proposed algorithm ; we use a MySQL database hosted in a centralized Apache server. We are considering several Android clients. And the PHP scripts carried by the http protocol as an exchange medium.

Key words : Synchronization, Database, Replication, Web Services, Mobile Applications

Table des matières

Table des figures

Liste des tableaux

Liste des abréviations

BDD	Base De Données
SQL	Structured Query Language
NoSQL	Not Only SQL
SGBD	Systéme De Gestion De Base De Données
HTML5	HyperText Markup Language
XML	Extensible Markup Language
HTTP	HyperText Transfer Protocol
SDK	Software Development Kit
API	Application Programming Interface
ACID	Atomicité, Cohérence, Isolation et Durabilité
GCP	Google Cloud Platform
JSON	JavaScript Object Notation
WSDL	Web Services Description Language
SAMD	Synchronization Algorithms based on Message Digests
MRDMS	Mobile Replicated Database Management Synchronization
TCP	Traditional Transmission Control Protocol
IP	Internet Protocol
OODS	Object-oriented data synchronization
MANET	mobile ad-hoc networks
DBMS	Oracle Database Management Solution
SyncML	Synchronization Mark-up Language
EDI	Electronic Data Interchange
REST	epresentational state transfer

SOAP	Simple Object Access Protocol
SOA	service-oriented architecture
FTP	File Transfer Protocol
PHP	Hypertext Preprocessor
CSS	Cascading Style Sheets
XHTML	Extensible HyperText Markup Language

Introduction générale

Avec l'évolution de la technologie et la submersion des moyens de communications comme les appareils portables et les PC portables (notebook). La tendance des utilisateurs s'oriente vers le travail n'importe où n'importe quand (anywhere at any time) ! Ce qui pose le problème de connexion ? De ce fait, les prestataires des services web confrontent un grand problème concernant la disponibilité de leurs services.

L'objectif de notre travail consiste à proposer une solution afin d'assurer la continuité des services web. Et pour cela, nous proposerons un modèle qui permet de synchroniser et de répliquer plusieurs bases de données.

De ce fait, nous suivrons le plan suivant :

— **Dans le premier chapitre**, nous aborderons des notions de bases en ce qui concerne les bases de données et leurs approches et les architectures client-serveur. Ainsi nous parlerons de la synchronisation et la réplication.

— **Dans le deuxième chapitre**, nous présenterons les différentes méthodes et modèles de synchronisation, ensuite nous faisons une comparaison entre ces derniers.

— **Dans le troisième chapitre**, nous présenterons l'algorithme de synchronisation.

— **Dans le dernier chapitre**, nous présenterons l'implémentation de l'algorithme proposé. Aussi nous présenterons les outils de programmation utilisés au long de ce projet.

Finalement, nous clôturons avec une conclusion générale et quelques perspectives.

Chapitre 1

Base de données et synchronisation

1.1 Introduction

Une application qui ne peut garder trace des interactions avec l'utilisateur et conserver des données d'une session à une autre est une application sans vie, que l'utilisateur s'empressera souvent de désinstaller. Toute application doit pouvoir charger et enregistrer des données.
Dans ce chapitre, on va aborder quelques notions importantes sur les bases de données, les applications web et la synchronisation de données.

1.2 Notion de base

1.2.1 Définition d'une base de données

Est un ensemble de données organisé pour être utilisé par des programmes correspondant à des applications distinctes, de manière à faciliter l'évolution autonome des données et des programmes [1].

1.2.2 L'histoire des bases de données

L'histoire des bases de données se situer aux années 1960, avec l'apparition des bases de données réseau et des bases de données hiérarchiques. Dans les années 1980, ce sont les bases de données object-oriented qui ont fait leur apparition. Aujourd'hui, les bases de données les plus courantes sont les SQL, NoSQL et bases de données Cloud.

Il est également possible de classer les bases de données en fonction de leur contenu : bibliographique, textes, nombres ou images. Toutefois, en informatique, les bases de données sont classées généralement en fonction de leur approche organisationnelle[2].

1.2.3 Critère de base de données

Une base de données doit répondre aux critères suivants :

1. **Structure :** Assurer qu'il y a une adaptation entre le mode de stockage des données et le traitement qui les exploiterons et les mettrons à jour[3].
2. **Non redondance de donnée :** Implique l'unicité des informations dans la bdd.C.à.d éviter la duplication des données parce que cela pose des problèmes de cohérence lors des mises à jour de ces données[3].
3. **L'exhaustivité :** Signifie que la base de données doit disposer toutes les informations correspondant au sujet donné [3].

1.2.4 Les différents types de bases de données

On distingue 4 types de base de données, comme le montre la Figure 1.1 :

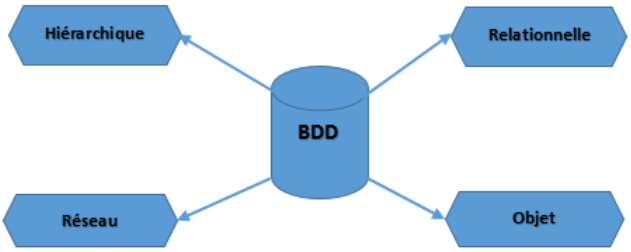


FIGURE 1.1 – Les types de base de données

1. **Les bases de données hiérarchiques :** C'est une forme du système de gestion de base de données qui relie les enregistrements dans une structure arborescente de manière à ce que chaque enregistrement contient un seul propriétaire[4]. Exemple :(voir la Figure 1.2 à la page 4)

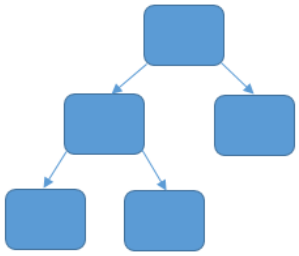


FIGURE 1.2 – Le modèle hiérarchique.

2. **Les bases de données réseau :** Le modèle de réseau est capable de résoudre de nombreuses difficultés dans le modèle hiérarchique grâce à la possibilité de créer des liens de type n-n, les liens entre objets pouvant exister sans contraintes. Pour trouver des données dans une telle modélisation, il est nécessaire de connaître le chemin d'accès (les liens) qui rend les programmes dépendants de la structure de données [4] . Exemple :(voir la Figure 1.3 à la page 4)

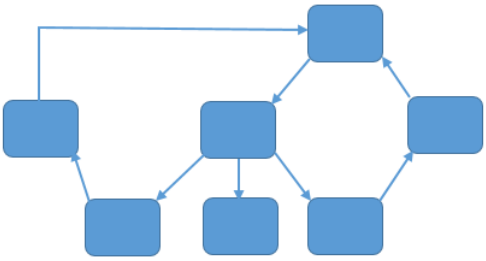


FIGURE 1.3 – Le modèle réseau.

3. **Les bases de données relationnelles :** Est une base de données structurée basé sur l'algèbre relationnelle [4]. Il permet de modéliser facilement et sans contraintes majeures des systèmes du monde réel et de créer des bases de données simples à maintenir, évolutives et indépendantes de leur support [5].
   Exemple :(voir la Figure 1.4 à la page 4)

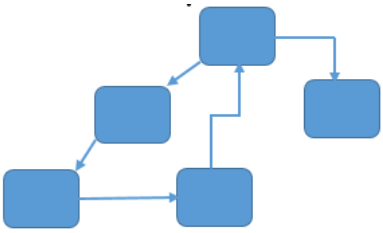


FIGURE 1.4 – Le modèle relationnelle.

4. **Les bases de données objet :** Les objets sont un concept de programmation qui simplifie la création de logiciel et offre de nombreux avantages aux grands projets informatiques[5]. Elle sera très probablement ajoutée au modèle relationnel[4].

1.3 Définition d'un SGBD

C'est un ensemble de logiciel informatique pour la gestion de l'information. Sa fonction principale est de fournir un support de stockage et de gestion de données aux applications informatiques via une interface de haut niveau. Il doit être capable de manipuler de très grands volumes de données d'une façon efficace tout en assurant la durabilité et la cohérence de l'information[6].

1.3.1 L'Objectif de SGBD

Les principaux objectifs fixés pour un système de gestion de base de données afin de résoudre les problèmes engendrés par la gestion sous forme de fichiers statiques sont les suivants :

1. **Indépendance physique :** La méthode de présenter les données doit être autonome des structures de stockage utilisées.
2. **Indépendance logique :** Toutes les différentes visions des données par les utilisateurs doivent être combinées dans une vision globale.
3. **Accès aux données :** Se fait par l'intermédiaire d'un langage de Manipulation de Données. Il est important que ce langage permette de répondre aux requêtes dans un délai raisonnable.
4. **Administration centralisée des données (intégration) :** Les données doivent être centralisées pour faciliter les résolutions des différentes visions des données (entre autres).
5. **Non redondance des données :** Chaque donnée doit être présentée une seule fois dans la base, pour éviter les problèmes de mise à jour.
6. **Cohérence des données :** Les données doivent être soumises à un certain nombre de restrictions d'intégration qui sont automatiquement vérifiées à chaque entrée ou modification afin d'assurer un état cohérent de la base de données.

7. **Partage des données :** Permettre à plusieurs utilisateurs d'accéder aux mêmes données au même temps, et cela peut être simple quand il s'agit d'interrogations uniquement, mais cela devient un problème complexe s'il s'agit de modifier des données quand il y a beaucoup d'utilisateurs.
8. **Sécurité des données :** Il faut pouvoir associer à chaque utilisateur des droits d'accès aux données, pour protéger ces données contre les accès non autorisés.
9. **Résistance aux pannes :** Il est impératif de pouvoir restaurer une base de données saine, si certains des fichiers contenant les données deviennent illisibles. Ainsi, après une panne au milieu de la modification, deux solutions peuvent être utilisées : soit restaurer les données dans l'état dans lequel elles étaient avant la modification, soit terminé l'opération interrompue [4].

1.3.2 L'architecture ANSI/SPARC

L'architecture Ansi/Sparc est une architecture fondamentale sur laquelle reposent les SGBD modernes, datant de 1975. Cette architecture est divisée en trois niveaux : le schéma interne (SI), le schéma conceptuel (SC) et les schémas externes (SE) [7], comme le montre la (Figure 1.5) :

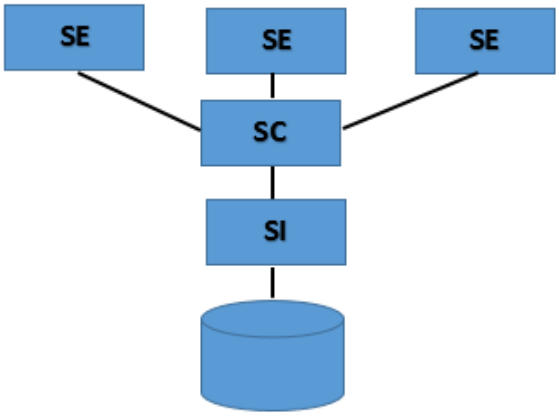


FIGURE 1.5 – L'architecture ANSI/SPARC [7].

1.3.3 Les niveaux de description des données

a **- Niveau externe :** Il définit les vues des utilisateurs qui concerne la façon dont les données sont consultées par les utilisateurs finaux.

b **-Niveau conceptuel :** Fusionne diverses opinions extérieures en une et composite ordinaire.

c **-Niveau physique (interne) :** Concerne la manière dont les données sont concrètement stockées [7].

1.3.4 La conception d'une base de données

On distingue quatre étapes dans la conception d'une base de données :

1. **L'analyse :** Consiste à étudier le problème et à consigner dans un document, les besoins, les choix et les contraintes.
2. **La modélisation logique :** Permet de décrire une solution, en prenant une orientation informatique générale (type de SGBD typiquement), formelle, mais de façon autonome de choix d'implémentation spécifiques.
3. **La modélisation conceptuelle :** Permettre de décrire le problème posé, en général graphique, en prenant des hypothèses de simplification. Ce n'est pas une description du réel, mais une représentation simplifiée d'une réalité.
4. **L'implémentation :** Concorder aux choix techniques, en termes de SGBD choisi et à leur mise en œuvre [8].

1.4 Application web

C'est un logiciel dont l'interface avec l'utilisateur s'exécute dans un navigateur et dont la logique est traitée par un serveur, c'est-à-dire une machine distante. Les échanges entre le navigateur et le serveur s'effectuent au moyen d'un réseau[9].

1.4.1 Les caractéristiques et les avantages d'une application web

— **Nécessitent un développement unique et un accès universel pour n'importe quel type d'appareil :** Un seul développement en HTML5 suffit pour n'importe quel système d'exploitation.

— **Il n'est pas nécessaire de les télécharger :** L'application est hébergée sur un serveur et accessible à partir d'un navigateur. Cela signifie qu'il est nécessaire d'être connecté pour y accéder.

— **Elles sont accessibles à partir de n'importe quel navigateur et depuis quel endroit de planète :** Si vous avez un navigateur installé sur votre appareil (Chrome, Firefox...), vous pouvez accéder à l'application web.

— **Elles apparaissent comme résultat dans les moteurs de recherche traditionnels :** Comme elles n'ont pas besoin d'être téléchargées, vous ne les trouverez pas dans les applications stores, mais elles apparaîtront en conséquence dans des moteurs tels que Google [9].

— Elles sont peu coûteuses et faciles à distribuer à de nombreux utilisateurs. Un navigateur web compatible est tout ce dont l'utilisateur a besoin [11].

— La maintenance est facile. Lorsque vous trouvez un bogue dans votre application ou lorsque vous avez ajouté de nouvelles fonctionnalités dont vous souhaitez faire profiter vos utilisateurs, il vous suffit de mettre à jour l'application sur le serveur web et vos utilisateurs peuvent profiter immédiatement de votre nouvelle application [11].

1.5 L'architecture client/serveur

L'architecture client/serveur permet de fournir des services d'un serveur à plusieurs clients. Le serveur est donc l'application qui fournit un ou plusieurs services (FTP, web...). Le serveur est à l'écoute des demandes : il attend les demandes. Le client est l'application qui a besoin de ces services. Il sera l'initiateur de la communication avec le serveur, c'est-à-dire celui qui enverra les demandes. Le client et le serveur peuvent être sur la même machine, comme ils peuvent être sur des machines distantes [12].

1.5.1 Les avantages de l'architecture Client/Serveur

L'architecture Client/Serveur dispose de nombreux avantages [12] :

— **Des ressources centralisées :** C'est effectivement le serveur qui fournit les services aux nombreux clients présents sur un réseau.

— **Une sécurité :** Accrue grâce à des points d'entrée qui peuvent être limités ou filtrés plus facilement.

— **Une administration au niveau serveur :** Étant donné que les clients ayant peu d'importance dans ce modèle, ils ont moins besoin d'être administrés.

1.5.2 Les inconvénients de l'architecture client/serveur

— **Le coût souvent élevé d'un serveur :** Si le serveur est une machine qui doit répondre à de nombreuses demandes et rapidement, la configuration matérielle doit suivre. En outre, si elle fournit un ou plusieurs services très importants, ne supportant pas une interruption aussi courte qu'elle puisse l'être, elle nécessite également des configurations matérielles coûteuses.

— **Le serveur est l'élément faible du service :** Plus de serveur, plus de service Heureusement, il existe de nombreuses solutions telles que les solutions de redondance de serveurs, c'est-à-dire plusieurs serveurs identiques fournissant le même service.

Si l'un des serveurs tombe en panne, il y en a toujours un qui pourra répondre. Il existe aussi des systèmes RAID, c'est-à-dire des redondances de données [12].

1.5.3 Fonctionnement d'un système client/serveur

Ce système fonctionne selon le schéma de la Figure 1.6 .

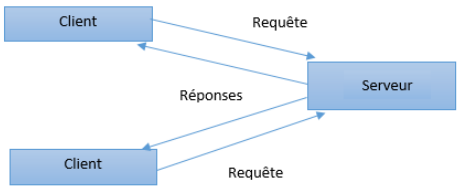


FIGURE 1.6 – Système client/serveur [44].

— Le client envoie une requête vers le serveur en utilisant son adresse IP et le port, qui désigne un service spécifique du serveur.

— Le serveur reçoit la demande et répond à l'aide de l'adresse de la machine client et son port[44].

1.6 Service web

Selon la définition du W3C (World Wide Web Consortium), un service Web est une application qui peut être appelée via Internet, par une autre application d'un autre site

Web, qui permet l'échange de données (de manière textuelle) afin que l'application appelante puisse intégrer le résultat de l'échange dans ses propres analyses. Les demandes et réponses sont soumises à des normes standardisées pour chacun de leurs échanges.
Un service web est une unité de logique d'application qui fournit des données et des services aux autres applications, les applications accèdent au web services via des protocoles et des formats de données omniprésentes telles que HTTP, SOAP, et XML [13].

1.7 La synchronisation :

En informatique, la synchronisation fait référence à l'un des deux concepts distincts mais liés : la synchronisation de thread ou de processus et la synchronisation des données [14].

1.7.1 La synchronisation de thread (processus)

C'est une application de mécanismes qui permet de garantir que deux threads ou processus s'exécutant simultanément n'exécutent pas des parties spécifiques d'un programme en même temps. Par contre si un processus a commencé à exécuter une partie sérialisée du programme, tout autre processus qui veut exécuter cette partie doit attendre jusqu'à le premier processus se termine son exécution. Cette synchronisation utilisée pour contrôler l'accès à la fois dans les systèmes multiprocesseurs à petite échelle, dans des environnements multithread, ordinateurs multiprocesseurs, dans des serveurs Web, etc [14].

1.7.2 La synchronisation des données

C'est le processus qui consiste à établir la cohérence entre les données d'une source vers le stockage de données cible et vice versa. Il est activé par le biais d'un logiciel spécialisé qui suit les versions des données lors de leur création et de leur utilisation, et garantit que quelles que soient les modifications des données, toutes les modifications sont fusionnées avec la source de données l'origine. Cette synchronisation est également utilisée pour la mise en miroir des données, où chaque ensemble de données est exactement reproduit ou synchronisé sur un autre appareil. Tel que les technologies de cette synchronisation conçues pour synchroniser un seul ensemble de données entre deux ou plusieurs périphériques et la copie automatique des modifications dans les deux sens [14].

1.7.3 Les types de synchronisations des données

Les processus de synchronisation comprennent une entité « source » et une entité « destination ». La synchronisation est divisée en deux catégories la synchronisation unidirectionnelle et la synchronisation bidirectionnelle [14].

A. La synchronisation unidirectionnelle

La synchronisation unidirectionnelle remplace les données de l'entité de destination par les données de l'entité source.

1. Les méthodes de synchronisation de sauvegarde créent des fichiers répliqués/miroir.
2. Si les anciennes versions des fichiers dans l'entité de destination doivent être conservées, une fonction d'archivage qui inclut que les fichiers supprimés dans l'entité source ne sont pas supprimés de l'entité de destination.
3. La consolidation ne garde pas de trace des conflits ou des suppressions de fichiers [14].

B. La synchronisation bidirectionnelle

La synchronisation bidirectionnelle fait par la fusion des données de la source et de la destination entité tel que :

1. Les fichiers nouveaux et mis à jour sont copiés dans les deux sens.
2. Les nouveaux fichiers ajoutés à l'entité source sont copiés vers la destination et vice versa.
3. Les fichiers supprimés dans la source sont supprimés des entités de destination et vice versa.
4. Les fichiers mis à jour dans la source sont copiés sur des fichiers plus anciens dans l'entité de destination et vice versa.
5. Si un fichier change dans les deux entités, le fichier est en conflit et doit être réconcilié manuellement [14].

1.7.4 La base de données en temps réel

Il s'agit d'une base de données NoSQL hébergée dans le cloud, qui permet de stocker les données au format JSON et toutes les modifications de ces données se reflètent

immédiatement en effectuant une synchronisation sur toutes les plateformes et tous les appareils [15].
Lorsque les utilisateurs passent en mode hors ligne, les SDK de la base de données en temps réel utilisent la cache pour enregistrer les modifications, et lorsque l'appareil est en ligne, les données locales sont automatiquement synchronisées. Dernière chose, les bases de données Firebase peuvent se joindre à l'authentification Firebase pour un processus d'authentification plus simple et plus rapide [16].

1.8 Définition de Firebase

C'est une plateforme qui permet de développer rapidement des applications pour mobile et le web. Elle peut être exploitée par plusieurs utilisateurs en même temps sans connaître un bug. Il offre de nombreuses fonctionnalités telles que l'authentification et la sécurité, la base de données en temps réel et le stockage des fichiers, les analyses, les notifications push, AdMod et bien d'autres. Il fournit le SDK pour Android, iOS, Web, NodeJS, C ++ et Java Server.
Firebase mettre à disposition des deux types d'outils :

— Les outils de développement et de test des applications.
— Les outils permettant d'augmenter et d'engager les cibles [16].

1.8.1 Une brève histoire de Firebase

En 2011, avant que Firebase ne soit Firebase, c'était une startup appelée Envolve. En tant qu'Envolve, il a fourni aux développeurs une API qui a permis l'intégration de la fonctionnalité de chat en ligne dans leur site Web.
Ce qui est intéressant, c'est que les gens ont utilisé Envolve pour transmettre des données d'application qui étaient plus que de simples messages de chat. Les développeurs utilisaient Envolve pour synchroniser les données d'application telles qu'un état de jeu en temps réel entre leurs utilisateurs. Cela a conduit les fondateurs d'Envolve, James Tamplin et Andrew Lee, à séparer le système de chat et l'architecture en temps réel. En avril 2012, Firebase a été créée en tant qu'entreprise distincte qui fournissait à Backend-as-a-Service des fonctionnalités en temps réel.

Après son acquisition par Google en 2014, Firebase est rapidement devenu le géant multifonctionnel d'une plateforme mobile et Web qu'elle est aujourd'hui [17].

1.8.2 Les nouvelles fonctionnalités de Firebase

Parmi les dernières fonctionnalités enregistrées sur Firebase se retrouvent :

— **Cloud Firestore :** Permet une synchronisation directe ainsi qu'une assistance hors connexion. En s'associant à d'autres produits Firebase, elle permet de créer des applications sans serveur. Cloud Firestore est équipé d'un émulateur local et vous permet de tester votre base de données.

— **ML kit** : Dédiée à l'apprentissage automatique et s'intègre facilement à votre application mobile. Vous n'aurez aucune difficulté à l'utiliser que vous soyez débutant ou professionnel.

— **Cloud Storage** : Firebase Storage vous permet de partager ou de stocker des contenus générés par les utilisateurs à ne citer que les images, les vidéos ou encore les fichiers audio. Il s'agit d'une solution de stockage d'objets puissante qui se distingue par sa simplicité et son caractère économique [16].

1.8.3 Les avantages de l'utilisation de la base de données en temps réel Firebase

— **En temps réel :** Cela signifier que s'il y a un changement dans les valeurs de la base de données, ce changement ne sera pas répercuté pour tous les utilisateurs à ce moment et il n'y aura pas de problème.

— **Grande accessibilité :** Sur la plateforme Firebase, il n'est pas nécessaire de taper plusieurs fois le même code pour différentes plateformes, car la base de données Firebase est accessible en temps réel à partir de différentes plateformes comme Android, iOS et web.

— **Mode hors ligne :** Si vous n'êtes pas connecté à l'internet et que vous avez modifié quelque chose sur votre application, cette modification ne sera reflétée dans votre application qu'à ce moment-là mais sur la base de données Firebase, la modification

sera mises à jour une fois que vous serez en linge, c'est-à-dire que votre appareil est connecté au l'internet, et ça à l'aide de l'activation de persistance de disque qui faite avec une seule ligne de code. Tels que tous les transitions effectue lorsque l'application est hors ligne sont mise en fille d'attend, une fois que l'application a retrouvé la connectivité réseaux, les transactions sont envoyer au serveur de base de donnée en temps réel. Si votre application utilise l'authentification Firebase, le client Firebase Realtime Database conserve le jeton d'authentification de l'utilisation hors des redémarrages de l'application. Si le jeton d'authentification expire alors que votre application est hors ligne, le client suspend les opérations d'écriture jusqu'à ce que votre application ré authentifie l'utilisateur, sinon les opérations d'écriture peuvent échouer en raison de règles de sécurité.

— **Contrôle l'accès aux données :** Par défaut, personne n'est autorisé à modifier les données dans la base de données Firebase Realtime, mais vous pouvez contrôler l'accès aux données, c'est-à-dire que vous pouvez définir quel utilisateur peut accéder aux données.
— **Pas de serveur d'application :** Les données sont directement accessibles depuis l'appareil mobile, donc il n'y a pas besoin de serveur d'application [15].
— Plus besoin d'infrastructures complexes[16].
— Une évolution constante et sûre [16].

1.8.4 L'interrogation des bases de données hors ligne

Les bases de données en temps réel de Firebase stockent les données renvoyées par une requête pour une utilisation hors ligne. Pour les requêtes effectuées hors ligne, la base de données Firebase en temps réel continue à travailler pour les données précédemment chargées. Si les données demandées n'ont pas été chargées, la base de données Firebase en temps réel charge les données du cache local. Lorsque la connectivité réseau est à nouveau disponible, les données se chargent et refléteront la requête [18].

1.8.5 Les configurations de base de données

Les données présentes dans la base de données sont très importantes et vous ne devez pas donner accès à tout le monde pour utiliser les données présentes dans votre base de

données. Ainsi, pour ce faire, Base de données en temps réel Firebase a des règles de configuration de base de données qui peuvent être utilisées pour fournir un accès différent à différents utilisateurs. Voici les règles de configuration de la base de données :

1. Par défaut, l'accès en lecture et écriture à votre bdd est désactivé.
2. Public : En utilisant des règles publiques, n'importe qui peut modifier les données présentes dans la base de données. Cette règle est généralement utilisée lorsque vous testez votre application et, après avoir testé l'application, vous pouvez définir la règle sur Utilisateur seulement [15].

1.9 La réplication des données

La réplication est un processus qui consiste à copier les données de références sur plusieurs serveurs pour améliorer la fiabilité, la tolérance aux pannes, un meilleur temps de réponse et la disponibilité des données dans les systèmes distribués.
Les serveurs répliqués sont de deux types : un primaire (maître) et un ou plusieurs serveurs secondaires (esclaves). Tel que le maître qui contient les données de référence peut exécuter des requêtes de type lecture et écriture. Par contre l'esclave qui s'y approvisionne ne peut exécuter que des requêtes de type lecture et des requêtes de réplication.
Toute modification de données doit s'effectuer sur le maître car l'information ne peut circuler que dans un seul sens, du maître vers l'esclave [43].

1.9.1 Les types de réplication des données

Il existe deux types de réplication des données :

A. La réplication des données synchrone :

Une réplication peut être synchrone : le serveur A envoie des données au serveur B et doit attendre que le serveur B termine le traitement et le notifie par l'envoi d'un accusé de réception, afin qu'il puisse continuer. En cas de catastrophe sur un site, la perte de données est donc minime, voire nulle.

B. La réplication des données asynchrone :

Une réplication peut également être asynchrone : dans ce cas, le serveur A envoie des données au serveur B et n'attend pas de réponse du serveur B pour continuer [43].

1.10 Problématique

Avec la submersion des nouvelles technologies mobiles tel que (les smartphones, les tablettes...) et l'évolution des réseaux mobiles de la 4G à la 5G. L'utilisation des bases de données distantes à plusieurs intérêt est devenue une pratique courante.
Assurer la continuité de service dépend de la disponibilité des données, Alors que garantir l'accès aux données partagées (en mode en ligne et hors ligne) n'est une tâche facile.
Les bases de données Firebase sont la seule alternative actuellement mais le problème des bases de données FireBase c'est qu'elles supportent un stockage très limité en mémoire cache, ce qui limite l'exploitation de service. En plus le risque de perdre les données si le système décide de vider la mémoire cache.

1.11 Objectifs de notre travail

Notre travail consiste à proposer des méthodes ou techniques de synchronisation entre les données distantes partagées qui viennent de différentes sources, et les données locales qu'on veut les partagées. Et garantir la continuité de services dans tous les modes (hors ligne et en ligne) et assure un partage entre clients qui sont aussi la source de partage.

1.12 Conclusion

Dans ce chapitre, nous avons présenté des notions de bases à propos des bases données, les SGBD, les applications web et l'architecture Client/Serveur. Aussi, nous avons parlé de la synchronisation et les problèmes qu'en rencontre habituellement.Nous avons clôturé par la problématique qui nous a conduit à proposer ce projet ainsi l'objectif de ce travail. Dans le prochain chapitre, nous présenterons un état de l'art des méthodes de synchronisation.

Chapitre 2

Etat de l'art des méthodes de synchronisation

2.1 Introduction

Dans ce chapitre, nous allons parler de différentes méthodes et techniques de synchronisation et leurs modes d'utilisation, avec une comparaison entre eux.

2.2 Les différentes méthodes et modèles de synchronisation

2.2.1 Modèle avec le marqueur de données en utilisant les données de service web

La méthode possible pour implémenter la synchronisation consiste à utiliser le service Web, qui joue le rôle d'un intermédiaire et facilitateur entre le client et le serveur de base de données [19], comme il montre la Figure 2.1 à la page 18.

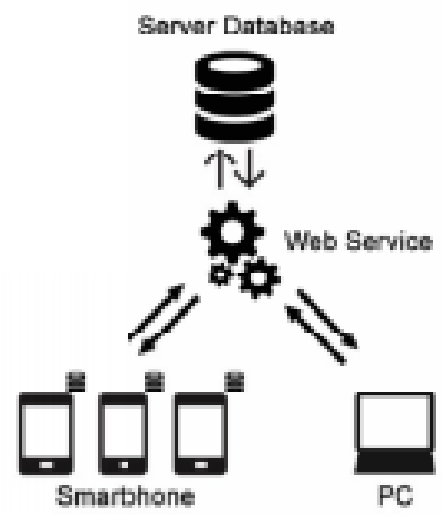


FIGURE 2.1 – Le Service Web [19].

Ce modèle de synchronisation de données est divisé en deux :
La synchronisation unidirectionnelle, comme le montre la **figure 2.3 à la page 22** et la synchronisation bidirectionnelle, comme le montre la **figure 2.2 à la page 20**.
Un exemples des étapes du processus de synchronisation bidirectionnelle :

1. l'utilisateur met à jour une donnée sur l'appareil A à 02h00.
2. Ensuite, l'utilisateur met à jour à nouveau la même donnée mais dans l'appareil B à 3h00.
3. Lorsque le processus de synchronisation est exécuté, les données sont entrées dans la base de données du serveur sont celles qui sont modifiées par le périphérique B en tant que dernière mise à jour.
4. Ainsi, l'utilisateur du l'appareil A obtiendra les dernières données mises à jour dans l'appareil B.

L'exemple du processus de synchronisation précédent peut être problématique, puisque n'est pas nécessairement que les dernières données entrées dans la base de données du serveur sont les plus récentes. Pour résoudre ces problèmes nous utilisons un marqueur qui sert d'une référence ou un signe de nouvelles données et qui sera écrit simultanément lors de la création des nouvelles données ou bien après la manipulation des données et comme un exemple de marqueur nous utilisons l'horodatage qui utilise le «temps» comme déterminant des données les plus récentes et pour enregistrer l'heure d'entrée des données.

Ainsi, les données qui ont le plus l'horodatage correspond aux données les plus récentes. Aussi pour éviter les demandes de données répétitives. Où le système ne demandera que les données qui ont été mises à jour après la dernière synchronisation.
Cette horodatage sera divisé en 3 types :

1. **Dernière synchronisation :** Signifie le temps de dernière synchronisation faite par l'utilisateur.
2. **Créé à :** Signifie le temps de création de ces données.
3. **Mis à jour à :** Signifie le temps de mise à jour de cette donnée [19].

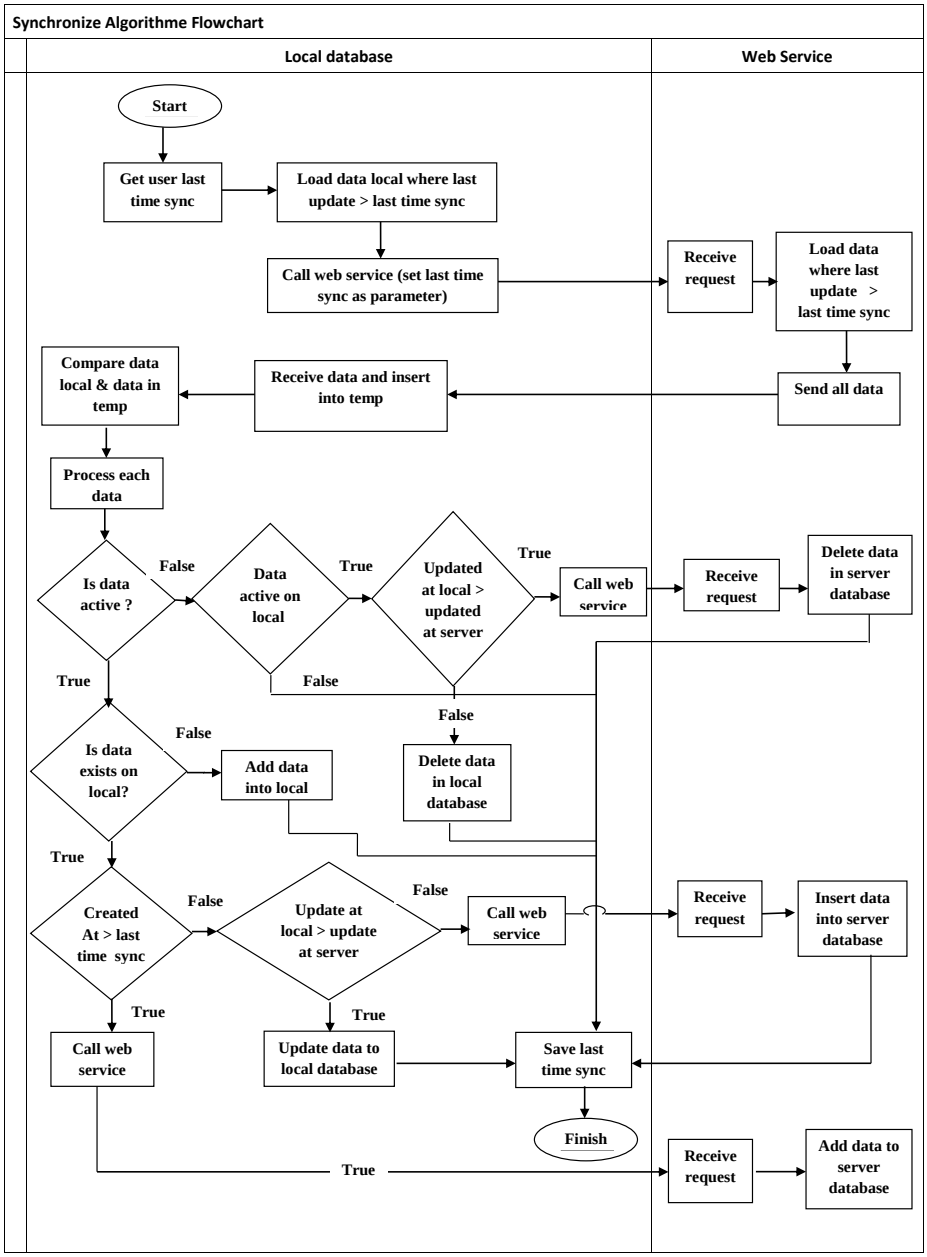


FIGURE 2.2 – Le flux de Synchronisation bidirectionnelle [19].

La figure2.2 montre le flux de synchronisation bidirectionnelle qui ce fait comme suite :

1. Lorsque nous démarrons la synchronisation nous obtenons le temps de dernière synchronisation de l'utilisateur ensuite nous chargeons les données locale où la dernière mise à jour est supérieure à la dernière synchronisation, puis nous appelons le service web et nous donnons comme paramètre le temps de dernière synchronisation fait par l'utilisateur.
2. Le service Web reçoit la demande de l'utilisateur, puis charge les données où la

dernière mise à jour est supérieure à la dernière synchronisation, puis envoie toutes ces données à la base de données locale.

3. L'utilisateur recevra ces données et les mettra en cache.
4. Comparez les données locales avec les données mises en cache, puis traitez ces données.
5. Vérifiez si ces données sont actives ou non.
6. Si ces données ne sont pas actives, vérifiez s'ils sont actives dans la bdd locale ou non.
7. Si ces données ne sont pas actives dans la BDD locale, nous enregistrons le temps de dernière synchronisation, puis nous terminons la synchronisation.
8. Sinon vérifiez si la mise à jour au niveau locale est supérieure à la mise à jour au niveau serveur.
9. Si la mise à jour au niveau locale est inférieure à la mise à jour au niveau serveur, nous supprimons les données dans la base de données locale, puis nous enregistrons le temps de dernière synchronisation et nous terminons la synchronisation.
10. Sinon nous appelons le service web, puis le service web reçoit la demande de l'utilisateur et supprime les données de la BDD du serveur, après nous enregistrons le temps de dernière synchronisation et nous terminons la synchronisation.
11. Si ces données sont actives, vérifiez si ces données existent dans locale.
12. Si ces données n'existent pas dans la BDD locale, nous ajoutons ces données dans la BDD locale, puis nous enregistrons le temps de dernière synchronisation et nous terminons la synchronisation.
13. Sinon vérifiez si le temps de création de ces données est supérieure au temps de dernière synchronisation.
14. Si le temps de création de ces données est inférieure au temps de dernière synchronisation, vérifiez si la mise à jour au niveau locale est supérieure à la mise à jour au niveau serveur.
15. Si la mise à jour au niveau local est inférieure à la mise à jour au niveau du serveur, nous appelons le service web. Puis le service web reçoit la demande de l'utilisateur et insère les données dans la BDD du serveur. Puis nous enregistrons le temps de dernière synchronisation et nous terminons la synchronisation.

16. Sinon modifiez les données dans la BDD locale, puis nous enregistrons le temps de dernière synchronisation et nous terminons la synchronisation.
17. Si le temps de création de ces données est supérieure à la dernière synchronisation, nous appelons le service web. Puis le service web reçoit la demande de l'utilisateur et ajoute ces données dans la BDD du serveur.

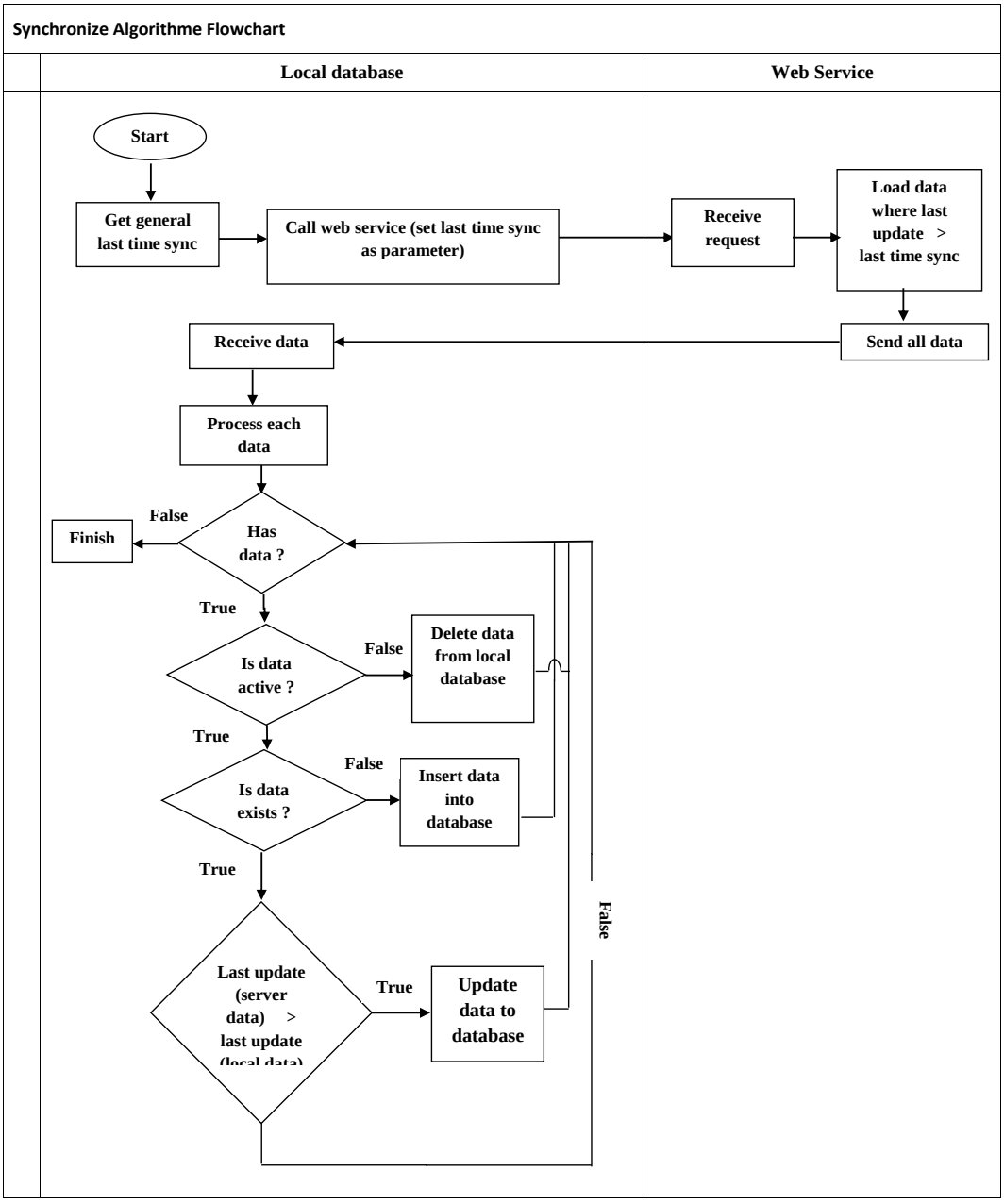


FIGURE 2.3 – Le flux de Synchronisation unidirectionnelle [19].

La figure2.3 montre le flux de synchronisation unidirectionnelle, un utilisateur comme un marqueur d'horodatage qui ce fait comme suite :

1. Lorsque nous démarrons la synchronisation nous obtenons le temps de dernière synchronisation et nous appelons le service web et nous donnons comme paramètre le temps de dernière synchronisation fait par l'utilisateur. Le rôle de ce service web est de télécharger toutes les données dans lesquelles le temps de dernière modification (last update) est supérieure au temps de dernière synchronisation (last time synchronisation).
2. Le service web reçoit la demande de l'utilisateur. Puis charge les données où la dernière mise à jour est supérieure au temps de dernière synchronisation puis envoie toutes ces données à la base de données locale.
3. L'utilisateur reçoit les données envoyées par le serveur et traite chaque donnée.
4. Pour toutes les données reçues et traitées, vérifiez si ces données sont actives.
5. Si ces données ne sont pas actives nous supprimons ces données de la base de données locale et encore une fois, nous vérifions en continu s'il y a des données ou non.
6. Sinon vérifiez s'il existe ces données dans la base de données locale.
7. Si ces données n'existent pas nous insérons ces données dans la base de données locale et encore une fois, nous vérifions en continu s'il y a des données ou non.
8. S'il existe ces données, vérifiez si la dernière mise à jour (données serveur) est supérieure à la dernière mise à jour (données locales).
9. Si la dernière mise à jour (données serveur) est supérieure à la dernière mise à jour (données locales) alors mettez à jour ces données dans la base de données locale et encore une fois, nous vérifions en continu s'il y a des données ou non.
10. Si la dernière mise à jour (données serveur) est inférieure à la dernière mise à jour (données locales) nous vérifions en continu s'il y a des données.
11. Lorsque nous effectuons toutes les vérifications pour toutes les données envoyées par le serveur nous terminons la synchronisation.

2.2.2 L'implémentation de la technique de synchronisation de base de données entre client et serveur

(Voir la figure 2.4 à la page 25) qui montre les étapes de l'algorithme de synchronisation des données qui sont indiquées ci-dessous :

Etape 1 : Saisissez les données dans le formulaire et téléchargez l'image à enregistrer dans la base de données.

Etape 2 : Vérifiez s'il y'a une connexion disponible ou non.

Etape3 : S'il n y'a pas de connexion disponible, nous activons le mode hors ligne. Puis les données stockées sur la base de données client et l'image stockée dans le dossier sur la machine client et encore une fois, il vérifiera en continu si la connexion est disponible ou non.

Etape4 : Si la connexion est disponible, nous activons le mode en ligne et il effectuera les opérations suivantes :

1. Si n'existe aucune entrée dans la base de données client, alors les données avec l'image stockées sur le serveur.
2. Si existe une entrée dans la base de données client, récupérez l'ID minimum et sa colonne correspondante avec l'image jusqu'à ce que la base de données soit vide.
3. Transférez les données avec une image de cet ID sur le serveur et supprimez-les de la base de données client et supprimez les images de la machine client. Puis des données avec l'image stockée sur le serveur[14].

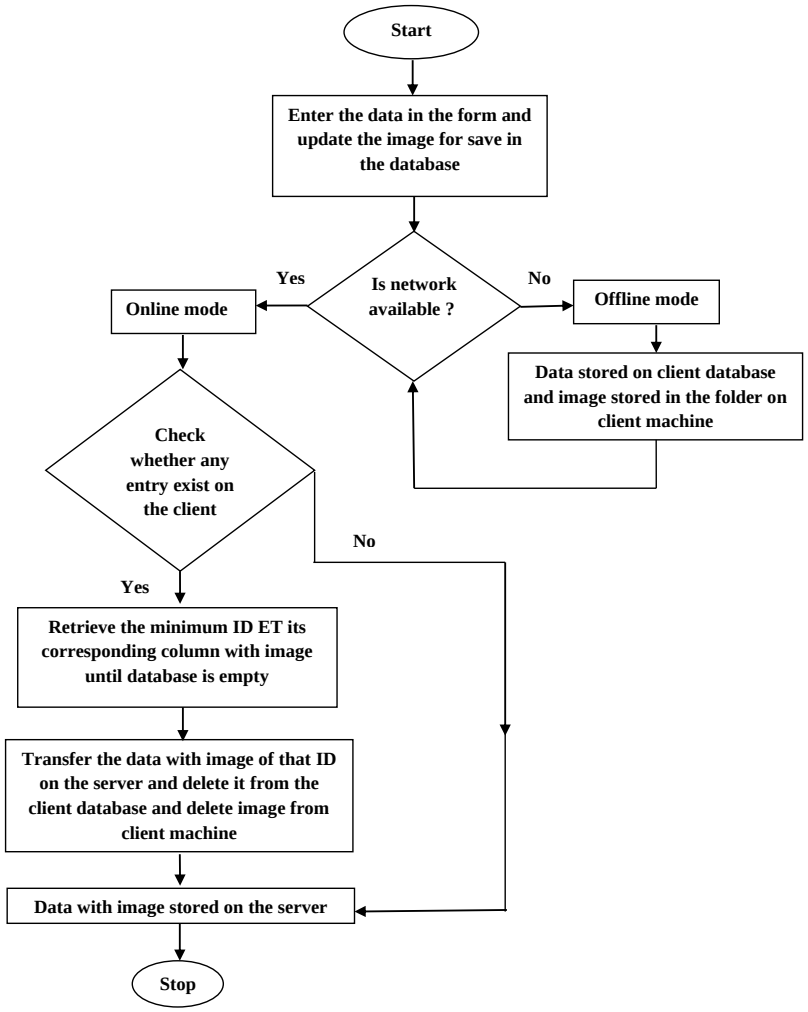


FIGURE 2.4 – Algorithme de synchronisation des données [14].

2.2.3 Algorithmes de synchronisations basées sur des messages digitaux (SAMD)

Les algorithmes de synchronisations basées sur les messages digitaux (SAMD) ont été proposés et mis en œuvre dans les travaux de[20],[21],[22],[23] et conçus pour faciliter la synchronisation des données entre une BDD côté serveur et une BDD mobile. Ces algorithmes sont indépendants des vendeurs de BDD. Ils dépendent plutôt de la fonction standard du langage SQL pour réaliser la procédure de synchronisation. Les algorithmes SAMD sont également sécurisés d'une manière ou d'une autre grâce à l'utilisation d'un résumé de message. Les inconvénients de ces algorithmes sont qu'ils utilisent des fonctions

de hachage, ils ne garantissant pas la sécurité lors de la transmission des données au serveur car ces valeurs de hachage résident dans une table de BDD aux deux extrémités. Leur principal objectif est de prouver les éventuelles incohérences des données. L'autre inconvénient est que le calcul du résumé des messages est gourmet en ressources[24].

2.2.4 Approche générique de la synchronisation des données

La synchronisation de la gestion des bases de données répliquées mobiles (MRDMS) qui a été proposé par sethia et al[25] est un système qui effectue une synchronisation basée sur les horodatages. Ceux-ci sont mis en œuvre au niveau des champs plutôt que des lignes. Une fois encore, le fait de conserver des horodatages basés sur les cellules plutôt que sur les lignes réduit les risques de conflit, car avec les horodatages basés sur les lignes, des cellules uniques peuvent ne pas avoir été modifiées. Le problème avec le système MRDMS est qu'il entraîne la création de plusieurs tables où les systèmes deviennent plus importants en termes de données requises, qu'il suppose une synchronisation initiale des temps, ce qui n'est pas toujours le cas et enfin, les insertions dans la base de données ne sont effectuées que lorsque l'appareil est en ligne. Cette solution fonctionne au niveau de la couche application.

Une méthode inter-couches pour la synchronisation des bases de données mobiles a été mise en œuvre par Jiao et al[26]. Elle traite la synchronisation au niveau de la couche transport plutôt qu'au niveau de la couche application.

Les auteurs soulignent que la plupart des systèmes traitent la synchronisation au niveau de la couche application mais que cette couche ne connaît pas les informations sur les connexions sous-jacentes. Le protocole traditionnel TCP/IP (Transmission Control Protocol/Internet Protocol) a été remplacé par le protocole TCP Westwood, mieux adapté aux réseaux sans fil, car il réduit le gaspillage des ressources de communication dû à la réduction des transactions sur les réseaux sans fil. Touchsync ne fournit aucune garantie de sécurité et il est toujours possible d'intercepter le lien de communication.

La synchronisation des données orientée objet (OODS) pour les bases de données mobiles dans les réseaux mobiles adhoc (MANET) a été étudiée par Li et al [27]. Avec cette solution, tout est observé comme un objet et la notion d'abonné/éditeur est adoptée et il n'y a pas de preuves de la sécurité des données pendant la synchronisation. La façon dont ces données sont synchronisées entre l'éditeur et l'abonné est que les abonnés peuvent

reconnaître automatiquement les mises à jour des schémas de publication. Dans le cas des éditeurs, la résolution des conflits et les schémas de publication sont définis par l'utilisateur ou hérités de l'arbre d'héritage de schéma. OODS offre une solution complète pour les applications qui fonctionnent sur MANET mais il existe de nombreuses applications mobiles qui fonctionnent sur un réseau autre que MANET.

Malhotra et Chaudhary[28] ont proposé un algorithme pour résoudre le problème lorsque tous les clients s'appuient sur une seule base de données de serveur. En cas d'arrêt planifié ou de panne, les travailleurs à distance opèrent sur le stockage local et lorsque la connexion reprend, les données sont synchronisées du système client au serveur dans l'ordre séquentiel. Si le système est déconnecté, tous les fichiers téléchargés par l'utilisateur sont enregistrés dans le dossier de l'appareil client et lorsqu'il y a une connexion, ces fichiers sont automatiquement transférés du client au serveur. C'est une solution simple mais qui ne prend pas en compte de ce qui pourrait arriver aux données lors de transferts des fichiers en mode en ligne au cas où tous les appareils passeraient en mode hors ligne, car les appareils mobiles fonctionnent sur des réseaux sans fil à faible bande passante qui ne sont pas fiables. De plus, ces données pourraient être de nature sensible et donc nécessite les caractéristiques de confidentialité et de non-répudiation.

La synchronisation entre la base de données Oracle et la base de données dans une application mobile simple a été réalisée par Zechmeister et al[29]. L'objectif est de réaliser un transfert de données concret en utilisant des documents XML sur le réseau mobile. La solution réalisée est efficace et sûre et les documents XML sont faciles et simples à utiliser. La solution peut être utilisée dans des applications de mise en œuvre similaires qui utilisent la base de données oracle. Cependant, le XML utilise de longues balises inutiles qui consomment plus de bits, bien que les documents soient compressés pour réduire leur taille, la décompression au niveau de l'appareil mobile prend du temps. Aucune flexibilité n'est offerte au développeur d'applications car il est limité à la solution de gestion de base de données Oracle (DBMS) pour mettre en œuvre cette technique. Si, par exemple, le développeur d'applications n'est compétent qu'en SGBD MySQL (langage de requête structuré), il est désavantagé. En d'autres termes, la solution est propriétaire et utilise des informations dépendantes de la base de données, spécifiques à Oracle. Une fois encore, la couche application est la base de cette synchronisation [24].

2.2.5 Synchronisation basées sur XML

Le langage de balisage de synchronisation(SyncMl) suggère un nouveau standard pour la synchronisation des données entre les appareils. Il contient un protocole qui dépend du XML pour transporter des messages sur réseau. Ce langage a été proposé par lee et al[30] , et améliorer par li et li[31], qui ajoute le codage de huffman pour compresser les données et améliorer la performance de synchronisation.

Le système d'acquisition de données mobile a été développé par huang et al[32] pour changer la méthode traditionnelle de collecte de données sur papier et qui basé sur les éléments communs à différents systèmes. Ce système contient un module de synchronisation qui exporte et importe les données de BDD font et BDD SQlite. Les utilisateurs doivent spécifier les exigences de leur besoin. Ces informations sont enregistrées dans les fichiers XML et copier sur l'appareil avec un dictionnaire de données après le chargement de module de synchronisation.

En utilisant XML, defferedSync c'est une technique de synchronisation pour les BDD mobile qui a été développée par mille et al[33] et qui permet de transformer la BDD relationnelle en une structure arborescente XML et utilise ensuite des vues différées afin de minimiser la bande passante et l'espace de stockage sur le client mobile. La méthode de synchronisation des données a été développée par Guo[34] et basée sur XML dans des environnements hétérogènes distribués pour transférer des messages simples et pour réaliser la synchronisation des données entre différentes bases de données en utilisant un certain nombre de procédures :

1. Conversion des données incrémentielles en fichier XML via une couche de mappage.
2. Envoyer le fichier XML à la destination dans un message format.

2.3 Comparaison entre méthodes de synchronisation

méthode de synchronisation	**Points Fort**	**Points Faible**
SAMD	- Faciliter la synchronisation des données entre la BDD coté serveur et la BDD coté client. -Prouver les éventuelles incohérences des données. - Cet algorithme est sécurisé grâce à l'utilisation d'un résumé de message. - Ils peuvent être mis en œuvre dans toutes les combinaisons de BDD côté serveur et mobile.	-Utiliser des fonctions de hachage, et ne garantissant pas la sécurité lors de la transmission des données au serveur. - Le calcul du résumé des messages est gourmet en ressources.
MRDMS	- Conserver des horodatages basés sûr les cellules plutôt que sur les lignes réduit les risques de conflit.	- La sécurité des données n'est pas prise en compte. - Il entraîne la création de plusieurs tables à mesuré que le système devient plus important en termes de donnée qu'il suppose une synchronisation initiale des temps ce qui n'est pas toujours le cas. -Gaspillage de bande passante.

Inter-couches	- Traiter de la synchronisation au niveau de la couche transport plutôt qu'au niveau de la couche application. - Remplacer le protocole TCP/IP par le TCP Westword qui adapté mieux aux réseaux sans fil. -Réduit le temps et la bande passante. -Améliore les performances de synchronisation.	- Le taux de perte d'énergie n'est pas mesuré.
Touchsync	- Il est toujours possible d'intercepter le lien de communication.	sécurité.
OODS	- Donner une solution complète pour les applications qui fonctionnent sur MANET. -L'efficacité de synchronisation satisfaisante par rapport à la technologie de réplication de fusion(MR).	- Il n'y a pas de preuve de la sécurité des données pendant la synchronisation. - Il n'y a pas de preuve d'évaluation des performances algorithmiques.

BDD Oracle et mobile	- Efficace et sûre et les documents XML sont faciles et simples à utiliser. -Peut être utilisée dans les applications de mise en œuvre analogue qui utilisent la BDD Oracle. - Le XML utilise de longues balises inutiles qui consomment plus de bits. - Les documents soient compressés pour réduire leur taille. - La solution est propriétaire et utilise des informations dépendantes de la base de données, particulier à Oracle.	- Aucune flexibilité n'est offerte au développeur d'applications. -La décompression au niveau de l'appareil mobile prend du temps.
SyncML	- Ajouter le codage de huffman pour compresser les données et améliorer la performance de synchronisation.	-Dépends entièrement des balises XML qui enveloppent le message de synchronisation ce qui conduit à de longs fichiers inutiles qui sont trop chers à.
defferedSync	- Il utilise des vues pour minimiser la bande passante et l'espace de stockage sur le client mobile. -Réduit la charge de transmission.	- Il n'est pas explicite sur le temps nécessaire pour synchroniser les données. -Il n'y a pas de preuve à l'appui.

Marqueur Et Service web	- Éviter de demander des données répétitives. - Aider pour décider si les données doivent être ignorées ou synchronisées avec une autre base de données. - Peut utiliser un service web directement dans notre application s'est existé sans gaspiller le temps pour programmer un autre service qui fait la même chose .	-En utilisant le protocole HTTP, les services Web peuvent déborder les mesures de sécurité en place grâce à des pare-feu. -Les normes de services web dans certains domaines sont actuellement récentes. - Faibles performances par rapport aux approches de l'informatique répartie comme : RMI, CORBA, ou DCOM.

TABLE 2.1 – La comparaison entre les méthodes de synchronisation [24].

2.4 Conclusion

Nous avons présenté dans ce chapitre les méthodes de synchronisation des données, tel que le modèle avec le marqueur de données en utilisant les données de service web, ce dernier est utilisé comme facilitateur entre le client et le serveur qui garantit la cohérence des données appartenant à différentes BDD, la méthode de synchronisation MRDMS qui est basée sur les horodatages, l'algorithme de synchronisation SAMD qui est basée sur les résumés des messages, la méthode Inter_couche qui permet de traiter la synchronisation au niveau de la couche transport plutôt qu'au niveau de la couche application, la méthode defferedSync est une technique de synchronisation pour les BDD mobiles qui permet de transformer une BDD relationnelle en une structure arborescente XML.
Dans le prochain chapitre, nous détaillerons notre proposition.

Chapitre 3

Proposition

3.1 Introduction

Dans ce chapitre, nous présenterons un modèle qui est basé sur la synchronisation et la réplication. Ensuite nous présentons quelques moyens de conception afin d'expliquer le fonctionnement du système.

3.2 Proposition

Notre modèle proposé, est un modèle centralisé hybride qui est basé sur la synchronisation et la réplication, afin d'assurer une continuité de service.

Chaque client joue le rôle d'une base de données répliquées et le serveur comme une base centrale, à travers laquelle toutes les communications entre les clients sont passées. Au même temps tous les clients peuvent agir et communiquer leurs données avec les autres.

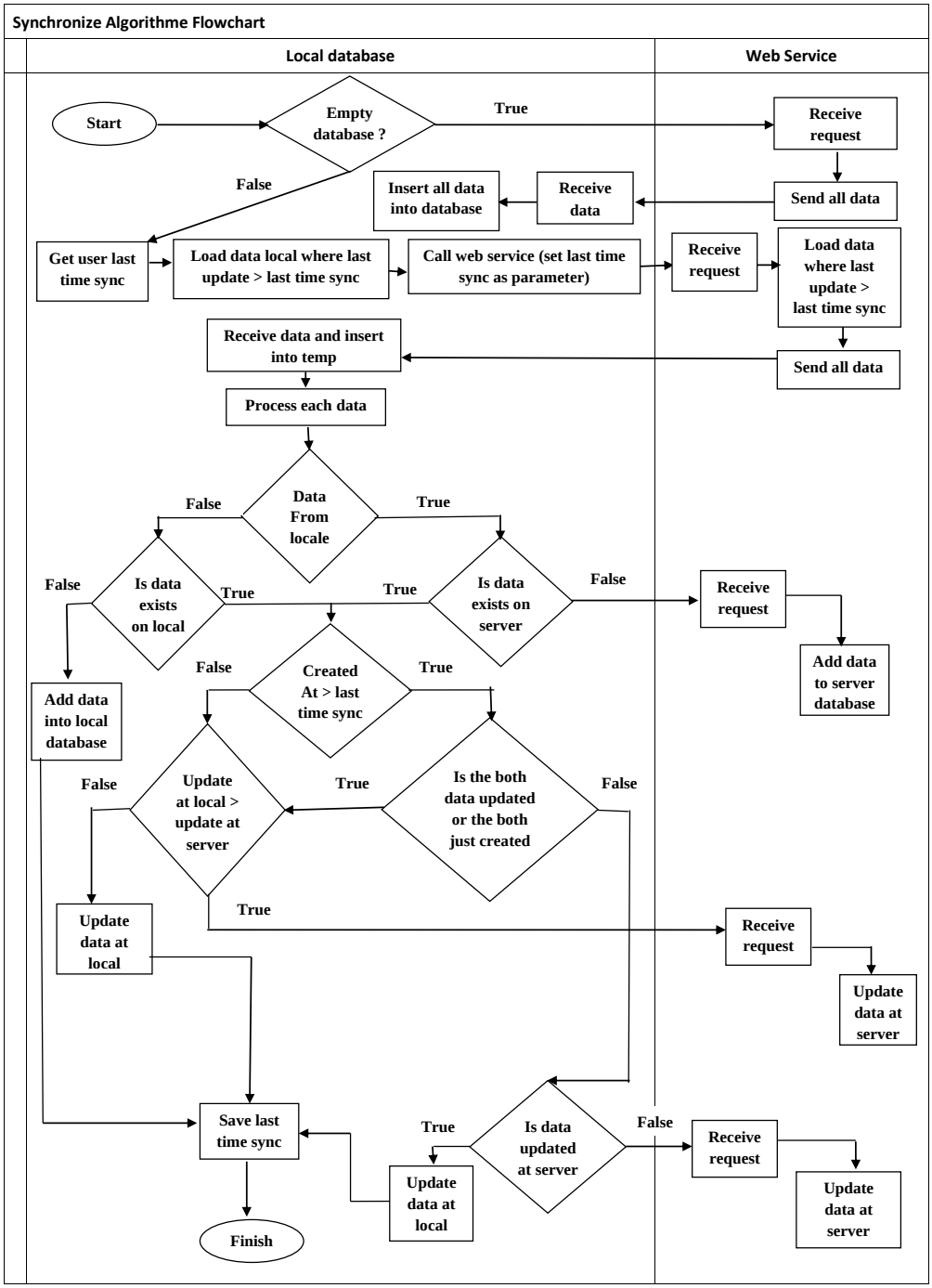


FIGURE 3.1 – Le schéma de synchronisation bidirectionnelle.

La figure 3.1 montre le flux de synchronisation bidirectionnelle qui ce fait comme suite :

1. Lorsque nous démarrons la synchronisation, nous vérifions si la base de données est vide ou non.

2. Si la BDD est vide, l'utilisateur envoie une demande au service web.
3. Le service web reçoit la demande de l'utilisateur, après envoie toutes les données présentes dans la base de données du serveur à la base de données locale.
4. Le client reçoit toutes ces données et les insère dans la BDD.
5. Sinon nous obtenons le temps dernière synchronisation de l'utilisateur et nous chargeons les données locales où la dernière mise à jour est supérieure au temps de dernière synchronisation, puis nous appelons le service web et nous donnons comme un paramètre le temps de dernière synchronisation fait par l'utilisateur.
6. Le service web reçoit la demande de l'utilisateur et charge les données où la dernière mise à jour est supérieure à le temps de dernière synchronisation. Ensuite, il envoie toutes ces données à la base de données locale.
7. L'utilisateur reçoit les données présentes dans la BDD du serveur et les insère dans le cache, puis traite ces données.
8. Vérifiez si ces données sont du local ou bien du serveur.
9. Si ces données sont du local, vérifiez si ces données existent dans le serveur ou non.
10. S'il ces données n'existent pas dans le serveur, l'utilisateur envoie une demande au service web, puis le service web reçoit cette demande et ajoute ces données à la base de données du serveur.
11. S'il existe ces données dans le serveur, vérifiez si le temps de création est supérieure au temps de dernière synchronisation.
12. Si le temps de création est inférieure au temps de dernière synchronisation, vérifiez si la mise à jour dans locale est supérieure à la mise à jour dans le serveur.
13. Si la mise à jour dans locale est supérieure à la mise à jour dans le serveur, l'utilisateur envoie une demande au service web, puis le service web reçoit cette demande et modifie ces données dans la base de données du serveur.
14. Sinon modifiez ces données dans la base de données locale et nous enregistrons le temps de dernière synchronisation, puis nous terminons la synchronisation.
15. Si le temps de création est supérieure au temps de dernière synchronisation, vérifiez si les deux données sont créées après le temps de dernière synchronisation et/ou les deux données sont mises à jour.

16. Si les deux données sont créées avant le temps de dernière synchronisation et/ou les deux données sont mettre à jour, vérifiez si ces données sont mises à jour dans le serveur.
17. Si ces données sont mises à jour dans le serveur, l'utilisateur modifie ces données dans la BDD locale et nous enregistrons le temps de dernière synchronisation, puis nous terminons la synchronisation.
18. Sinon l'utilisateur envoie une demande au service web, puis le service web reçoit cette demande et modifie ces données dans la base de données du serveur.
19. Si les deux données sont créées après la date de la dernière synchronisation et/ou les deux données sont mises à jour, vérifiez si la mise à jour dans locale est supérieure à la mise à jour dans le serveur.
20. Si la mise à jour dans locale est supérieure à la mise à jour dans le serveur, l'utilisateur envoie une demande au service web , puis le service web reçoit cette demande et modifie à jour ces données dans la base de données du serveur.
21. Sinon modifiez ces données dans la base de données locale et nous enregistrons le temps de dernière synchronisation, puis nous terminons la synchronisation.
22. Si ces données sont du serveur, vérifiez si ces données existent dans locale.
23. Si ces données n'existent pas dans locale, ajoute ces données dans la BDD locale et nous enregistrons le temps de dernière synchronisation, puis nous terminons la synchronisation.
24. S'il existe ces données dans locale, vérifiez si le temps de création est supérieure au temps de dernière synchronisation.
25. Si le temps de création est inférieure au temps de dernière synchronisation, vérifiez si la mise à jour dans locale est supérieure à la mise à jour dans le serveur.
26. Si la mise à jour dans locale est supérieure à la mise à jour dans le serveur, l'utilisateur envoie une demande au service web, puis le service web reçoit cette demande et modifie ces données dans la base de données du serveur.
27. Sinon modifiez ces données dans la base de données locale et nous enregistrons le temps de dernière synchronisation, puis nous terminons la synchronisation.
28. Si le temps de création est supérieure au temps de dernière synchronisation, vérifiez

si les deux données sont créées après la date de la dernière synchronisation et/ou les deux données sont mises à jour.

29. Si les deux données sont créées avant la date de la dernière synchronisation et/ou les deux données sont mises à jour, vérifiez si ces données sont mises à jour dans le serveur.
30. Si ces données sont mises à jour dans le serveur, l'utilisateur modifie ces données dans la BDD locale et nous enregistrons le temps de dernière synchronisation, puis nous terminons la synchronisation.
31. Sinon l'utilisateur envoie une demande au service web, puis le service web reçoit cette demande et modifie ces données dans la base de données du serveur.
32. Si les deux données sont créées après la date de la dernière synchronisation et/ou les deux données sont mises à jour, vérifiez si la mise à jour dans locale est supérieure à la mise à jour dans le serveur.
33. Si la mise à jour dans locale est supérieure à la mise à jour dans le serveur, l'utilisateur envoie une demande au service web, puis le service web reçoit cette demande et modifie ces données dans la BDD du serveur.
34. Sinon mettre à jour ces données dans la base de données locale et nous enregistrons le temps de dernière synchronisation, puis nous terminons la synchronisation.

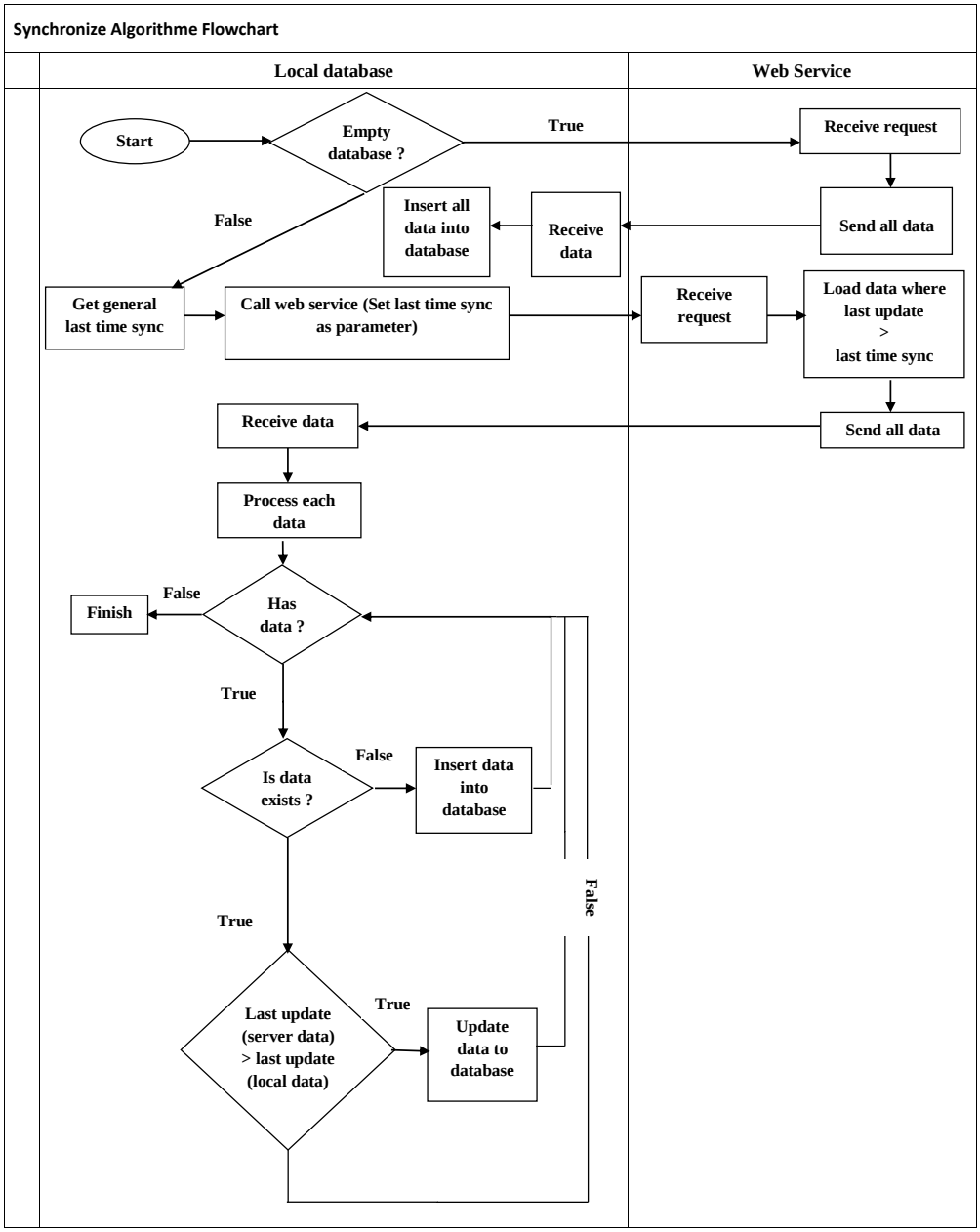


FIGURE 3.2 – Le schéma de synchronisation unidirectionnelle.

La figure 3.2 montre le flux de synchronisation unidirectionnelle un utilisateur comme un marqueur d'horodatage qui ce fait comme suite :

1. Lorsque nous démarrons la synchronisation, nous vérifions si la base de données est vide ou non.
2. Si la BDD est vide, l'utilisateur envoie une demande au service web.

3. Le service web reçoit la demande de l'utilisateur, puis envoie toutes les données stockées dans la base de données du serveur à la base de données locale.
4. L'utilisateur reçoit les données envoyées par le service web, puis insérez toutes ces données dans la base de données locale.
5. Sinon nous obtenons le temps de dernière synchronisation et nous appelons le service web et nous donnons comme un paramètre le temps de dernière synchronisation fait par l'utilisateur.
6. Le service web reçoit la demande de l'utilisateur, puis charge les données où la dernière mise à jour est supérieure au temps de dernière synchronisation, puis envoie toutes ces données à la base de données locale.
7. L'utilisateur reçoit les données envoyées par le serveur et traite chaque données.
8. Pour toutes les données reçues et traitées, vérifiez s'il existe ces données dans la BDD locale ou non.
9. Si ces données n'existent pas dans la BDD locale en insérant ces données dans la base de données et encore une fois, nous vérifions en continu s'il y a des données.
10. S'il existe ces données vérifiez si la dernière mise à jour (données serveur) est supérieure à la dernière mise à jour (données locales).
11. Si la dernière mise à jour (données serveur) est supérieure à la dernière mise à jour (données locales) alors mettre à jour les données dans la base de données locale et encore une fois, nous vérifions en continu s'il y a des données.
12. Si la dernière mise à jour (données serveur) est inférieure à la dernière mise à jour (données locales) vérifiez encore une fois s'il y a des données.
13. Lorsque nous effectuons toutes les vérifications pour toutes les données envoyées par le serveur nous terminons la synchronisation.

3.3 Diagramme de séquence de déroulement de synchronisation bidirectionnelle

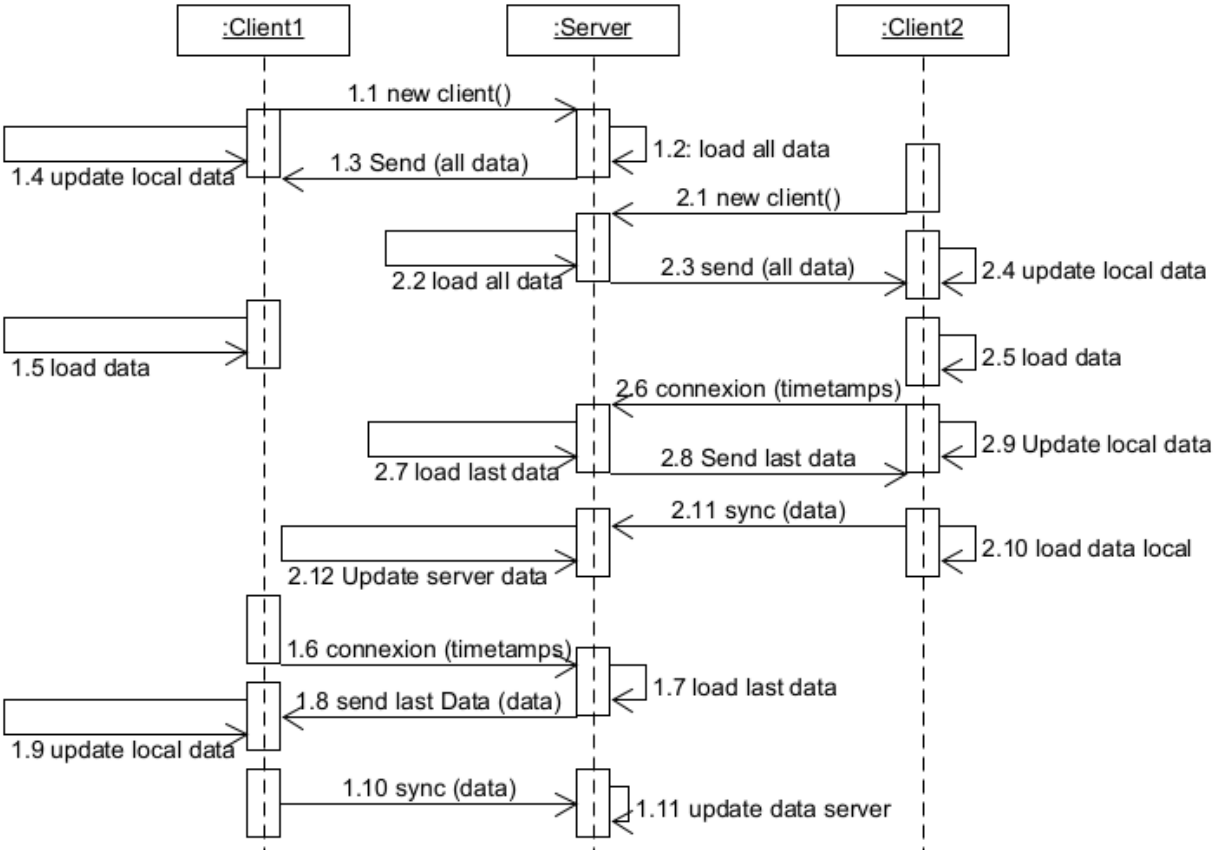


FIGURE 3.3 – Le diagramme de séquence de déroulement de synchronisation bidirectionnelle.

3.3.1 Scénario de déroulement de synchronisation bidirectionnelle

Scénario 1 : new clients :

1. Dès qu'un nouveau client se connecte au serveur. il envoie un message (Download all data). Afin de télécharger toutes les données stockées dans ce dernier.
2. Le serveur charge et envoie toutes les données présentes sur le serveur (load and send all data).
3. Le client crée sa propre base de données, et insère toutes les données envoyées par le serveur (insert all data sent by the server in local data base).

Scénario 2 : clients :

1. Chaque client insère ses données dans sa propre base de données.

2. Dès qu'une connexion s'établit avec un client, ce dernier envoie une demande de

synchronisation (Sync(timestamp)), pour récupérer les données stockées après la dernière sync (new entries or updated entries). En même temp le client mis ses données locales (new entry or update entry) qui est stockées après la dernière synchronisation (timestamp), dans une mémoire temporaire (Temp).

3.1 Pour chaque entry présente dans la mémoire Temp ou envoyée par le serveur.

3.2 Modifier les données locales par les données envoyer par le serveur.

3.3 Modifier les données de bdd serveur par les données envoyer comme paramètre.

3.4 Diagramme de séquence de déroulement de synchronisation unidirectionnelle

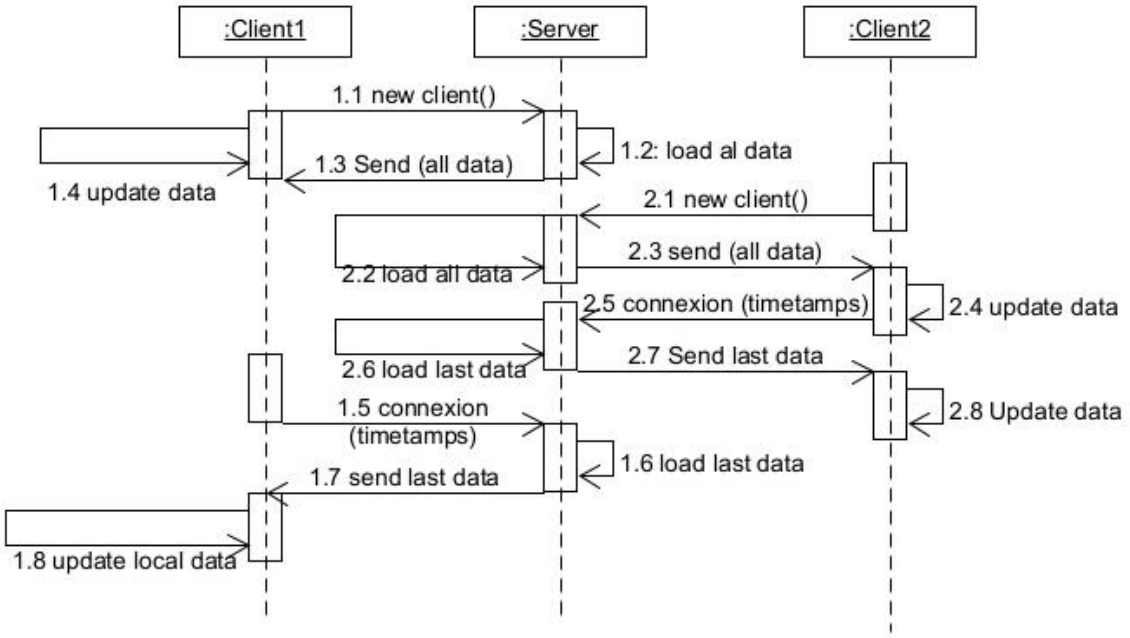


FIGURE 3.4 – Le diagramme de séquence de déroulement de synchronisation unidirectionnelle.

3.4.1 Scénario de déroulement de synchronisation unidirectionnelle

Scénario 1 : new clients :

1. Dès qu'un nouveau client se connecte au serveur. il envoie un message (Download all data). Afin de télécharger toutes les données stockées dans ce dernier.
2. Le serveur charge et envoie toutes les données présentes sur le serveur (load and send all data).
3. Le client crée sa propre base de données, et insère toutes les données envoyées par le serveur (insert all data sent by the server in local data base).

Scénario 2 : clients :

1. Chaque client insère ses données dans sa propre base de données.

2. Dès qu'une connexion s'établit avec un client, ce dernier envoie une demande de synchronisation (Sync(timestamp)), pour récupérer les données stockées après la dernière sync (new entries or updated entries). En même temp le client mis ses données locales (new entry or update entry) qui est stockées après la dernière sync (timestamp), dans une mémoire temporaire (Temp).

3.1 Pour chaque données envoyer par le serveur et recevoir par le client, vérifier si les données existe dans locales.

3.2 Si les données existe dans locales en vérifient si le temps de mis à jour dans le serveur supérieur à la mis à jour dans locale.

3.3 Si le temps de mis à jour dans le serveur supérieur à le temps de mis à jour dans locale, modifier la base de données locale par les données de serveur.

3.5 Conclusion

Dans ce chapitre, nous avons présenté notre algorithme, qui est basé sur la synchronisation et la réplication. En exploitant les marqueurs et les services web.
Prochainement nous présenterons les procédés pour l'implémentation et le test.

Chapitre 4

Implémentation et test

4.1 Introduction

Dans ce chapitre, nous allons définir l'environnement, les langages de programmation et des bases de données que nous avons utilisé dans le processus d'implémentation. Ainsi nous présenterons quelques tests pour valider les algorithmes proposés. Ensuite nous allons présenter les bouts de code les plus importants.

4.2 Présentation des langages et l'environnement utilisés

Afin d'implémenter notre solution, différents environnements de développement et les langages ont été utilisées.

4.2.1 Les langages de programmation

La méthode implémentée dans notre travail est réalisée avec les langages de programmation suivants :

Java

C'est un langage de programmation orienté objet et une plateforme informatique, développé par Sun Microsystems en 1995, et depuis son acquisition par la société Oracle en 2009, la technologie Java est inséparable du domaine de l'informatique et du web. nous la retrouvons donc sur les ordinateurs, mais également sur les téléphones mobiles, les consoles de jeux, etc. L'avènement du Smartphone et la puissance croissante des ordinateurs, ont conduit à un regain d'intérêt pour ce langage de programmation [36].

PHP

Le PHP est un langage informatique utilisé sur l'internet. Le terme PHP est un acronyme récursif de "PHP : Hypertext Preprocessor". Ce langage est principalement utilisé pour produire un site web dynamique. Il est courant que ce langage soit associé à une base de données, telle que MySQL.
Fonctionnant du côté du serveur (l'endroit où le site est hébergé) il n'est pas nécessaire que les visiteurs aient un logiciel ou un plugin particulier. Néanmoins, les webmasters qui souhaitent développer un site en PHP doivent s'assurer que l'hébergeur prend en compte ce langage. Lorsqu'une page PHP est exécuté par le serveur, alors celui-ci renvois généralement au client (les visiteurs du site) une page web qui peut contenir du HTML, XHTML, CSS, JavaScript ... etc[37].

JSON

JSON (JavaScript Object Notation) est un format léger d'échange de données. Il est facile à lire ou à écrire pour des humains.Il est également facile à analyser et à générer pour les machines.

JSON est un format de texte totalement autonome du langage, mais qui utilise des conventions familières aux programmeurs de la famille des langages C,notamment C, C++, C#, Java, JavaScript, Perl, Python et bien d'autres. Ces propriétés font de JSON un langage parfait pour l'échange de données.
JSON ne comprend que deux éléments structurels :

1. Des ensembles de paires nom/valeur .
2. Des listes ordonnées de valeurs [40].

StarUML

StarUML est un logiciel de modélisation UML (Unified Modeling Language) open source qui peut remplacer dans bien des situations des logiciels commerciaux et coûteux. Étant simple d'utilisation, nécessitant peu de ressources système, supportant UML 2, ce logiciel constitue une excellente option pour une familiarisation à la modélisation. Cependant, seule une version Windows est disponible [35].

4.2.2 L'environnement de développement

Nous avons utilisé l'environnement Android Studio, WampServer, Postman .

Android Studio

C'est un environnement de développement intégré (IDE) pour le développement des applications Android basé sur IntelliJ IDEA, un environnement de développement java intégré pour les logiciels, et intégré ses outils d'édition de code et de développement. Pour prendre en charge le développement d'application dans le système d'exploitation Android, Android Studio utilise un système de construction basé sur Gradle, un émulateur, des modèles de code.
Chaque projet dans Android Studio a une ou plusieurs modalités avec le code source et les fichiers de ressources. Ces modalités incluent les modules d'application Android, les modules de bibliothèques et les modules Google app Engine[38].

WampServer

C'est une plateforme de développement Web sous Windows pour des applications Web dynamiques utilisant le serveur Apache2, le langage de scripts PHP et une base de données MySQL. Il dispose aussi de PHPMyAdmin pour gérer plus facilement vos bases de données[39].

Apache

Le logiciel libre Apache est un serveur HTTP créé et maintenu au sein de la fondation Apache. C'est le serveur HTTP le plus populaire du World Wide Web. Il est distribué selon les termes de la licence Appache.
Pour permettre à notre application de se connecter, il est nécessaire d'ajouter la permission dans le fichier de configuration d'Apache (httpd.conf) et de remplacer la directive « ALLOW FROM 127.0.0.1 » par « ALLOW FROM ALL ».
Le serveur va ainsi pouvoir répondre automatiquement aux requêtes provenant de l'application.

Postman

Est un environnement de développement d'API qui aide les utilisateurs à créer, documenter, tester, surveiller et publier la documentation de leurs API [41].
Nous avons utilisé cette environnement pour tester les scripts PHP que nous avons utilisé dans notre implémentation.

Une base de données SQLite

SQLite est une base de données open source, qui supporte les fonctionnalités standards des bases de données relationnelles comme la syntaxe SQL, les transactions et les prepared statement. La base de données nécessite peu de mémoire lors de l'exécution (env. 250 ko), ce qui en fait un bon candidat pour être intégré dans d'autres environnements d'exécution.
SQLite prend en charge les types de données TEXT (similaire à String en Java), INTEGER (similaire à long en Java) et REAL (similaire à double en Java). Tous les autres types doivent être convertis en l'un de ces types avant d'être enregistrés dans la base de données. SQLite ne vérifie pas si les types des données insérées dans les colonnes correspondent au type défini, par exemple, vous pouvez écrire un nombre entier dans une colonne de type chaîne de caractères et vice versa. [42].

4.3 L'implémentation :

Pour l'implémentation de notre solution nous avons suivi le principe du schéma illustré à la figure 4.1 à la page 48. Cette figure montre que l'application Android envoie une requête HTTP au serveur avec l'adresse du script PHP et ce script récupère les données à partir de la base de données MySQL. La base de données est responsable d'insérer et modifier les données dans les tables ou bien de renvoyer le résultat d'une sélection. Le résultat retourné est en format JSON qui permet de représenter l'information structurée. Le résultat est transféré vers l'application, Puis l'application convertir ce résultat pour le réutiliser.

FIGURE 4.1 – La communication android avec MySQL et PHP [10].

Donc nous avons commencé l'implémentation de notre proposition avec la création d'une base de données interne SQLite et la base de données externe MYSQL.

4.3.1 La base de données externe MYSQL :

Nous avons créé une base de données dans WAMPServeur grâce au phpMyAdmin qui permet de gérer la création des bases de données, les tables et la gestion des utilisateurs et leurs privilèges. Cette base de données contient deux tables : la table «etudiants» et la table «moyenne», comme le montre la figure 4.2 à la page 48 et la figure 4.3 à la page 48.

#	Nom	Type	Interclassement	Attributs	Null	Défaut	Extra	Action
1	Matricule	varchar(20)	latin1_swedish_ci		Non	Aucune		Modifier Supprimer Primaire Unique Index Spatial plus
2	Nom	varchar(20)	latin1_swedish_ci		Non	Aucune		Modifier Supprimer Primaire Unique Index Spatial plus
3	Prenom	varchar(20)	latin1_swedish_ci		Non	Aucune		Modifier Supprimer Primaire Unique Index Spatial plus
4	DateN	date			Non	Aucune		Modifier Supprimer Primaire Unique Index Spatial plus
5	Adresse	varchar(30)	latin1_swedish_ci		Non	Aucune		Modifier Supprimer Primaire Unique Index Spatial plus
6	Niveau	varchar(10)	latin1_swedish_ci		Non	Aucune		Modifier Supprimer Primaire Unique Index Spatial plus
7	code_Fac	varchar(10)	latin1_swedish_ci		Non	Aucune		Modifier Supprimer Primaire Unique Index Spatial plus
8	Create_a	datetime			Non	Aucune		Modifier Supprimer Primaire Unique Index Spatial plus
9	temps_mis_ajour	datetime			Non	Aucune		Modifier Supprimer Primaire Unique Index Spatial plus

Tout cocher *Pour la sélection :* Afficher Modifier Supprimer Primaire Unique Index

FIGURE 4.2 – La table étudiant.

Afficher Structure SQL Rechercher Insérer Exporter Importer Privilèges Opérations Déclencheurs

#	Nom	Type	Interclassement	Attributs	Null	Défaut	Extra	Action
1	Matricule	varchar(50)	latin1_swedish_ci		Non	Aucune		Modifier Supprimer Primaire Unique Index Spatial plus
2	moyenne_s1	double			Non	Aucune		Modifier Supprimer Primaire Unique Index Spatial plus
3	moyenne_s2	double			Non	Aucune		Modifier Supprimer Primaire Unique Index Spatial plus
4	décision	varchar(50)	latin1_swedish_ci		Non	Aucune		Modifier Supprimer Primaire Unique Index Spatial plus
5	create_a	datetime			Non	Aucune		Modifier Supprimer Primaire Unique Index Spatial plus
6	temps_mis_ajour	datetime			Non	Aucune		Modifier Supprimer Primaire Unique Index Spatial plus

Tout cocher *Pour la sélection :* Afficher Modifier Supprimer Primaire Unique Index

FIGURE 4.3 – La table moyenne.

4.3.2 La base de données interne SQLite :

Nous avons créé dans l'application trois clients (étudiant, service d'inscription, scolarité), et chaque client a sa propre base de données locale qui contient deux tables(timesynchro, client) : «timesynchro» pour sauvegarder le temps de synchronisation et «client» pour sauvegarder les données concernant leur travaille.
Pour pouvoir travailler en locale nous avons besoin d'une base de données SQLite, et pour la créer nous avons implémenté une classe «Database» qui hérite de «SQLiteOpenHelper». Cette classe permettra de définir la table qui sera produit lors de l'instanciation. Pour appeler la base de données nous avons utilisé la méthode onCreate(SQLiteDatabase), et nous avons créé la table avec la méthode «execSQL()» qui permet d'exécuter une instruction SQL, comme le montre la figure 4.4 à la page 49.

```
public class Database extends SQLiteOpenHelper {
    public static final String  DATABASE_NAME="student.db";
    public static final String  TABLE_NAME="student_table";
    public static final String  col_1="MAT";
    public static final String col_2 ="NOM";
    public static final String col_3="Prenon";
    public static final String  col_4="Adresse";
    public static final String  col_5="DateN";
    public static final String  col_6="Nivaux";
    public static final String  col_7="code_f";
    public static final String  col_8="Create_a";
    public static final String  col_9="Mis_ajour_a";
    public Database(@Nullable Context context) {
        super(context,  DATABASE_NAME,  factory: null,  version: 1);

    }
    @Override
    public void onCreate(SQLiteDatabase db) {
db.execSQL("create table "+TABLE_NAME+" (MAT TEXT PRIMARY KEY ,NOM TEXT,Prenon TEXT ,Adresse TEXT,DateN TEXT,Nivaux TEXT,code_f TEXT," +
        "Create_a TEXT, Mis_ajour_a TEXT )");
    }
```

FIGURE 4.4 – La création de la base de données locale.

Et pour insérer ou bien mettre à jour les données de notre base de données nous devons implémenter la méthode «insert()» qui permet d'insérer une ligne qui prend les valeurs qui sont dans l'objet «ContentValues», comme le montre la figure 4.5 à la page 50 et la méthode «update()» pour apporter des modifications aux lignes existantes en utilisant la condition «**WHERE**» pour spécifier quelles lignes doivent être modifiées, comme le montre la figure 4.6 à la page 50.

```
public boolean insertdata(String  mat,String  nom,String prenom,String adresse,String daten,
                          String nivaux,String code_f,String Create_a,String Mis_ajour_a ) {
    SQLiteDatabase db=this.getWritableDatabase();
    ContentValues  contentValues=new  ContentValues();
    contentValues.put(col_1,mat);
    contentValues.put(col_2,nom);
    contentValues.put(col_3,prenom);
    contentValues.put(col_4,adresse);
    contentValues.put(col_5,daten);
    contentValues.put(col_6,nivaux);
    contentValues.put(col_7,code_f);
    contentValues.put(col_8,Create_a);
    contentValues.put(col_9,Mis_ajour_a);
   long result=db.insert(TABLE_NAME, nullColumnHack: null,contentValues);

    if(result==-1){
        return false ;
    }
    else return  true;
}
```

FIGURE 4.5 – L'insertion des données dans la base de données locale.

```
public boolean updatedata(String  mat,String  nom,String prenom,String adresse,String daten,
                          String nivaux,String code_f,String Create_a,String Mis_ajour_a){
    SQLiteDatabase db=this.getWritableDatabase();
    ContentValues contentValues=new  ContentValues();
    contentValues.put(col_1,mat);
    contentValues.put(col_2,nom);
    contentValues.put(col_3,prenom);
    contentValues.put(col_4,adresse);
    contentValues.put(col_5,daten);
    contentValues.put(col_6,nivaux);
    contentValues.put(col_7,code_f);
    contentValues.put(col_8,Create_a);
    contentValues.put(col_9,Mis_ajour_a);
    long result = db.update (TABLE_NAME, contentValues,  whereClause: "MAT=?",new String []{String.valueOf (mat)} );
    if(result==-1){
        return false ;
    }
    else return  true;
}
```

FIGURE 4.6 – La mise à jour des données dans la base de données locale.

Et nous avons appelé ces deux méthodes dans la classe principale «MainActivity», comme le montre la figure 4.7 et la figure 4.8 :

```
isInserteed= database.insertdata(mat.getText().toString()
    ,nom.getText().toString()
    ,preno.getText().toString(),adresse.getText().toString(),dateN.getText().toString(),
    nivaux.getText().toString(),code_f.getText().toString(),new SimpleDateFormat( pattern: "yyyy-MM-dd HH:mm:ss").format(new Date()),
        new SimpleDateFormat( pattern: "yyyy-MM-dd HH:mm:ss").format(new Date())) ;
if (isInserteed=true){
    Toast.makeText( context: InsetEtu .this, text: "Data inserted",Toast.LENGTH_LONG).show();
}
else  Toast.makeText( context: InsetEtu .this, text: "Data not inserted",Toast.LENGTH_LONG).show();
```

FIGURE 4.7 – L'appel de la méthode insert.

```
isInserteed= database.updatedata (mat.getText().toString()
        ,nom.getText().toString()
        ,preno.getText().toString(),adresse.getText().toString(),dateN.getText().toString(),
        nivaux.getText().toString(),code_f.getText().toString(),new SimpleDateFormat( pattern: "yyyy-MM-dd HH:mm:ss").format(new Date())
        ,new SimpleDateFormat( pattern: "yyyy-MM-dd HH:mm:ss").format(new Date())) ;
if (isInserteed=true){
    Toast.makeText( context: InsetEtu .this, text: "data updated",Toast.LENGTH_LONG).show();
}
else  Toast.makeText( context: InsetEtu .this, text: "data not updated",Toast.LENGTH_LONG).show();
```

FIGURE 4.8 – L'appel de la méthode update.

La figure 4.9 à la page 51 représente la méthode «getAllData()» qui permet de récupérer toutes les données de notre base de données locale en utilisant la méthode «rawQuery()» qui exécute une requête SQL «SELECT» et retourne le résultat dans un cursor pour faciliter l'utilisation.

```
public Cursor getAllData(){
    SQLiteDatabase db=this.getWritableDatabase();
    Cursor res=db.rawQuery( sql: "Select * from  student_table" , selectionArgs: null);
    return res;
}
```

FIGURE 4.9 – La récupération des données de la base de données locale.

4.3.3 Les scripts PHP

Nous avons créé des scripts PHP, qui permettent d'effectuer plusieurs opérations tel que : la connexion à la base de données, l'insertion et la mise à jour d'un élément, la récupération de toute les données de la base de données,...

Fichier de connexion à la base de données

Pour accéder à la base de données nous avons créé un fichier «connection.php» qui permet de se connecter au serveur de base de données grâce au simple appel de la fonction «mysql_connect()» en donnant comme paramètre le nom de serveur, le nom de la base de données, le nom de l'utilisateur et le mot de passe de la base, comme le montre la figure 4.10 :

```php
<?php
$db_name="Database";
$mysql_username="root";
$mysql_password="";
$server_name="localhost";
// selection de la base
$connexion=mysql_connect($server_name,$mysql_username,$mysql_password,$db_name);

 if($connexion){
    echo"connexion success";
}
else{

    echo"connexion echoué";
}
```

FIGURE 4.10 – Le fichier de connexion à la base de données.

Fichier d'insertion d'un étudiant dans la base de données

A fin que nous pouvons insérer des données, nous avons créé un fichier «insert-Data.php» qui permet d'insérer un étudiant en envoyant les données à insérer via la requête HTTP POST. Ce dernier permet d'ajouter l'étudiant que nous avons inséré à notre base de données, comme le montre la figure 4.11 à la page 52 et la figure 4.13 à la page 53 :

```php
<?php
if($_SERVER['REQUEST_METHOD']=='POST'){
 require 'connection.php';

    insetData();
}
    function insetData(){
      global $connect;
     $Matricule = $_POST['Matricule'];
      $Nom = $_POST['Nom'];
      $Prenom = $_POST['Prenom'];
      $DateN = $_POST['DateN'];
      $Adresse = $_POST['Adresse'];
      $Niveau = $_POST['Niveau'];
      $code_Faculti = $_POST['code_Faculti'];
      $query=" INSERT INTO etudiant(Matricule, Nom, Prenom, DateN, Adresse, Niveau,
       code_Faculti, Create_a, temps_mis_ajour)
       VALUES ('$Matricule',
      '$Nom','$Prenom','$DateN','$Adresse','$Niveau','$code_Faculti',CURRENT_TIMESTAMP(),CURRENT_
      mysqli_query($connect,$query)or die(mysqli_error($connect));
      mysqli_close($connect);
    }
```

FIGURE 4.11 – L'insertion d'un étudiant dans la BDD du serveur.

Ensuite, pour tester ce script nous avons utilisé le logiciel «PostMan» en insérant les champs nécessaires et en cliquant sur send comme indiqué sur la figure 4.12 :

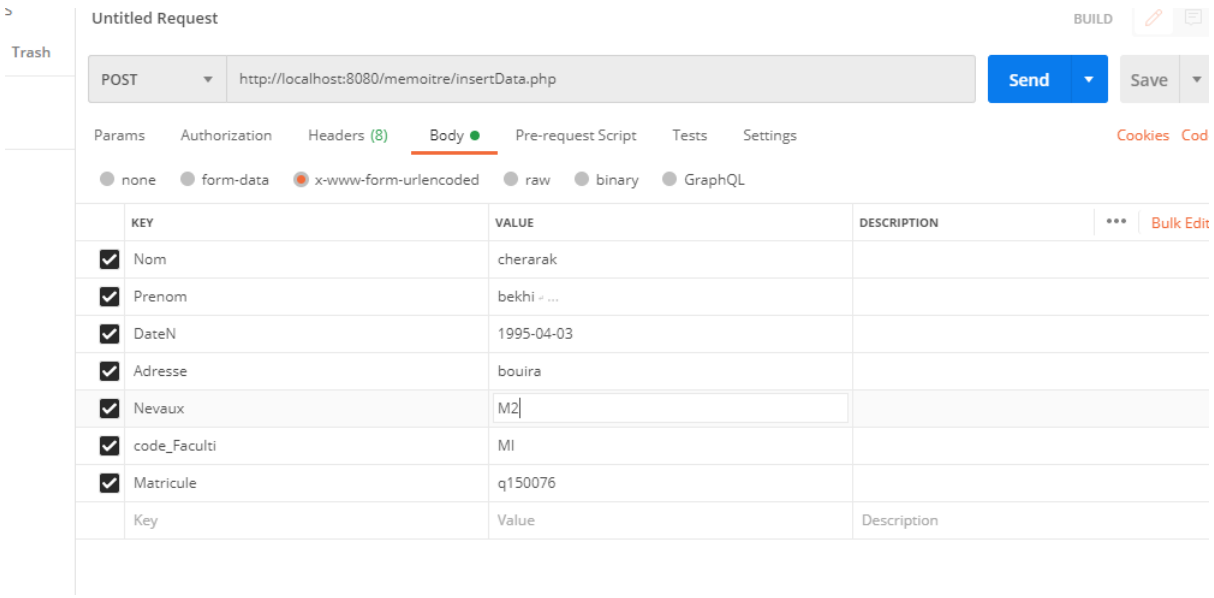


FIGURE 4.12 – L'insertion des champs de l'étudiant.

Voilà la figure 4.13 qui indique que l'étudiant a été bien ajouté à notre base de données.

Matricule	Nom	Prenom	DateN	Adresse	Niveau	code_Faculti	Create_a	temps_mis_ajour
q150076	cherarak	bekhi	1995-04-03	bouira	M2	MI	2020-10-13 20:31:01	2020-10-13 20:31:01
q150195	amlik	manaf	1995-12-27	bouira	M2	MI	2020-10-14 11:53:48	2020-10-14 11:53:48

FIGURE 4.13 – L'affichage d'un étudiant dans la base de données du serveur.

Pour récupérer toutes les données de notre base de données, nous avons créé un fichier GetAllData.php.

Dans ce fichier nous avons appelé la fonction «mysql_query()» qui permet d'exécuter une requête sql et retourne un résultat qui sera ultérieurement coder en JSON grâce au simple appel de la fonction «json_encode()», comme le montre la figure 4.14.

```php
<?php
$host = "localhost";
$username = "root";
$pwd = "";
$db = "database";
$con=mysqli_connect($host,$username,$pwd,$db) or die("Unable to Connect");
if(mysqli_connect_error($con))
{echo "Failed to connect";}
$query=mysqli_query($con,"select * from etudiants ");
if($query)
{while($row=mysqli_fetch_array($query))
    {$flag[]=$row;}
echo json_encode($flag);
}
mysqli_close($con);
?>
```


FIGURE 4.14 – La sélection de toutes les données.

Ensuite, nous avons testé ce code dans le logiciel « PostMan » et nous avons obtenu le résultat affiché dans la figure 4.15 à la page 54.

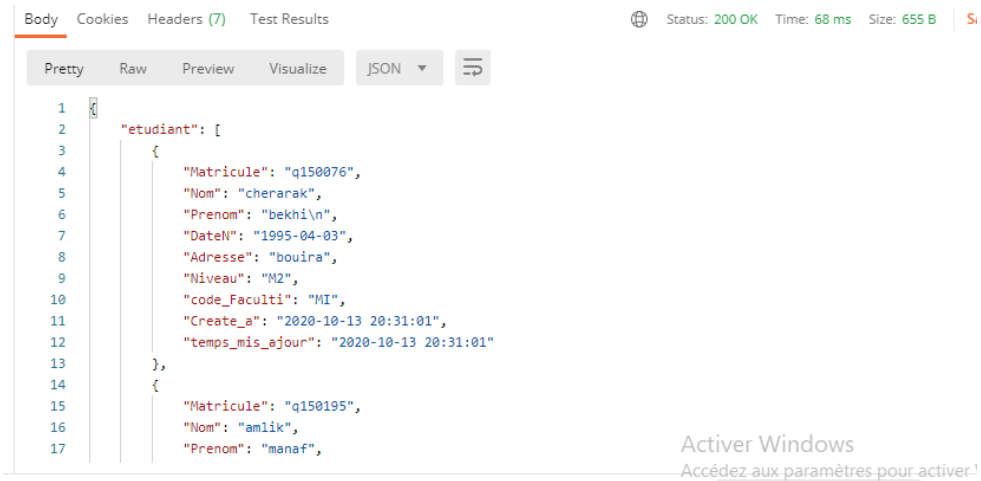


FIGURE 4.15 – Le résultat de la sélection des données sous forme JSON.

La figure 4.16 représente le code de fichier « synchro.php » qui permet de récupérer les données qui sont modifiées après le temps de dernière synchronisation et afficher le résultat sous forme JSON.

```php
<?php
$host = "localhost";
$username = "root";
$pwd = "";
$db = "database";
$con=mysqli_connect($host,$username,$pwd,$db) or die("Unable to Connect");
if(mysqli_connect_error($con))
{echo "Failed to connect";}
$temp=$_GET['las'];
$query=mysqli_query($con,"select * from etudiants where temps_mis_ajour >'".$te
if($query)
{while($row=mysqli_fetch_array($query))
  {$flag[]=$row;}
echo json_encode($flag);}
mysqli_close($con);?>
```


FIGURE 4.16 – La récupération des données qui sont modifiées après le temps de dernière synchronisation

Nous avons testé le code précédent avec le logiciel « PostMan » tel que nous avons donné un temps et nous avons obtenu comme résultat les données qui sont mises à jour après le temps que nous avons donné, comme le montre la figure 4.17 à la page 55.

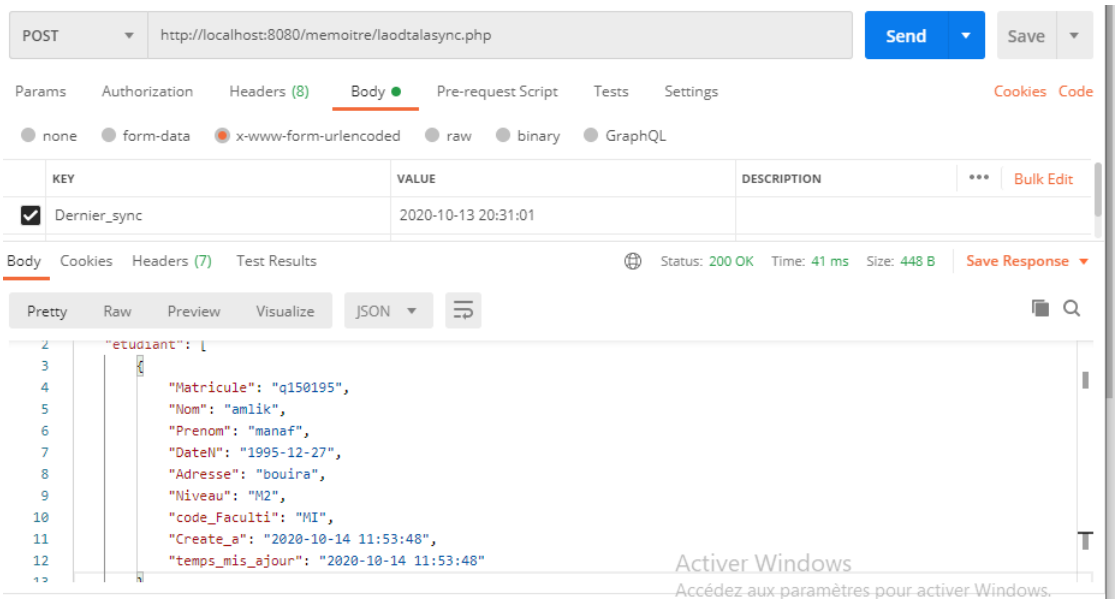


FIGURE 4.17 – Le résultat des données modifiées après le temps de dernière synchronisation sous forme JSON.

Et pour faire les mêmes opérations pour la table moyenne nous laissons les mêmes scripts php que nous avons fait pour la table « etudiants » nous remplaçons juste le nom de la table « etudiants » par « moyenne », et les attributs de table « etudiants » par les attributs de table « moyenne ».

4.3.4 La création de classe unidirectionnelle et bidirectionnelle

Pour programmer les algorithmes unidirectionnelle et bidirectionnelle nous avons créé deux classes : la première classe s'appelle « SyncroUnid » qui représente l'algorithme unidirectionnelle et la deuxième classe s'appelle « SynchroBidi » qui représente l'algorithme bidirectionnelle.

La classe unidirectionnelle :

Dans cette classe nous avons implémenté la méthode « start()», dans cette méthode nous avons vérifié, si la base de données est vide ou non. Si la base de données est vide nous appelons la méthode « getal() » qui permet de récupérer les données de base de données du serveur et de les insérer dans la base de données locale et si la BDD n'est pas vide nous appelons la méthode « timelASTSyncr() » qui permet de récupérer le temps de dernière synchronisation qui à été sauvegardé dans la table « timesynchro ». Puis nous appelons la méthode « syn()» qui prend comme paramètre la date récupérée par la méthode « timelASTSyncr() », comme indiqué dans la figure 4.18 :

```
public void Start() {
    Cursor res = database.getAllData ( );
    if (res.getCount ( ) == 0) {
        //apelle service pour inser dans bdd;
        StrictMode.setThreadPolicy ((new StrictMode.ThreadPolicy.Builder ( ).permitNetwork ( ).build ( )));
        getall ( );
    } else {
        String date = timelASTSyncr ( );
        Toast.makeText ( context: SyncroUni.this, date, Toast.LENGTH_LONG).show ( );
        StrictMode.setThreadPolicy ((new StrictMode.ThreadPolicy.Builder ( ).permitNetwork ( ).build ( )));
        syn (date);
    }

}
```

FIGURE 4.18 – La méthode start de la classe unidirectionnelle.

Et puisque les méthodes « syn() » et « getal() » communiquent avec le serveur il est nécessaire d'établir une connexion pour récupérer le résultat en ajoutant les lignes de code suivantes dans ces méthodes, comme le montre la figure 4.19 :

```
URL url = new URL (urladdress);
HttpURLConnection con = (HttpURLConnection) url.openConnection ( );
con.setRequestMethod ("GET");
is = new BufferedInputStream (con.getInputStream ( ));
```

FIGURE 4.19 – La connexion avec url.

La classe bidirectionnelle :

Dans cette classe nous avons implémenté la méthode « start() » et dans cette méthode nous avons vérifié si la base de données est vide ou non. Si la base de données est vide nous chargeons les données du serveur et nous les insérons dans la base de données locale et si la BDD n'est pas vide nous appelons la méthode «timelASTSyncr()» qui permet de récupérer le temps de dernière synchronisation qui à été sauvegardé dans la table « time-synchro ». Puis nous appelons la méthode « comserlo()» qui prend comme paramètre la date qui permet de synchroniser les données locales avec les données du serveur. Ensuite nous appelons la méthode «comloser()» qui prend comme paramètre la date qui permet de synchroniser les données du serveur avec les données du locale, comme le montre la figure 4.20 à la page 57 :

```
private void start() {
    //is bdd empty
    Cursor res = database.getAllData ( );
    if (res.getCount ( ) == 0) {

        //laod data from seveur and insrted into database locale
        new ServiceStubAsyncTask ( context: this,  activity: this).execute ( );
    } else {
        //get last time syncro
        String date = timelASTSyncr ( );
        //synro data local with data serveur
        StrictMode.setThreadPolicy((new StrictMode.ThreadPolicy.Builder().permitNetwork().build()));
       comserLo(date);
         //synro data serveur with data local
        StrictMode.setThreadPolicy((new StrictMode.ThreadPolicy.Builder().permitNetwork().build()));
        comloser(date);
    }
}
```

FIGURE 4.20 – La méthode start de classe bidirectionnel.

Pour que notre application puisse utiliser le wifi, il faut obtenir l'autorisation nécessaire en ajoutant la ligne suivante au fichier "AndroidManifest.xml".

```
<uses-permission android:name="android.permission.INTERNET" />

<application
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
    android:label="mémoire"
```

FIGURE 4.21 – L'ajout de la permission de connexion dans le fichier manifest.

4.4 Les tests

Dans cette partie, nous avons fait quelques tests pour valider notre proposition et nous avons vérifié que les deux algorithmes(bidirectionnelle et unidirectionnelle) fonctionnent correctement.

En premier lieu, nous avons commencé par tester l'algorithme bidirectionnelle, et nous avons lancé trois clients différents qui ont saisi les moyennes des étudiants, comme indiqué dans la figure 4.22 à la page 58 :

FIGURE 4.22 – (a) Client 1 et (b) Client 2 et (c) Client 3.

Lorsque les clients cliquent sur le bouton « ADD », les données de chaque client sont ajoutées à sa propre base de données, comme indiqué dans la figure 4.23 :

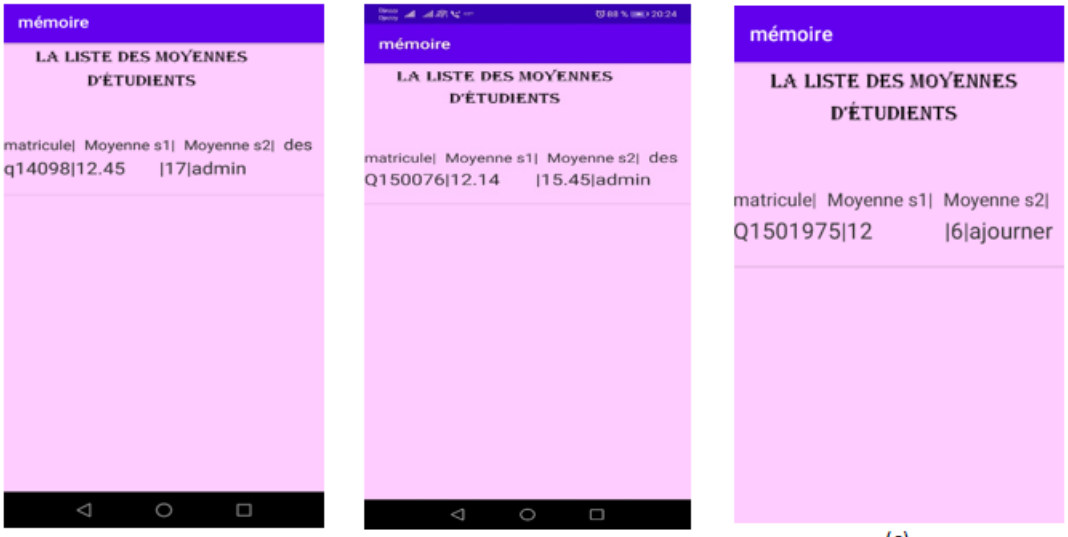


FIGURE 4.23 – La base de données locale de client 1, client 2 et client 3.

Après avoir saisi les informations dans la base de données de chaque client et après avoir effectué la synchronisation par le « client 1 », les données saisies sont transférées vers le serveur, comme le montre la figure 4.24 à la page 59 :

Options

	Matricule	moyenne_s1 moyenne_s2	moyenne_s2 décision	decision create_a	create_a temps_mis_ajour	temps_mis_ajour
Modifier Copier Effacer	q14098	12.45	17	admin	2020-11-15 20:25:30	2020-11-18 11:10:02
Modifier Copier Effacer	q150195	15	15	admin	2020-11-11 10:10:10	2020-11-11 10:10:10

FIGURE 4.24 – La base de données serveur après la synchronisation de client 1.

Et après la synchronisation de client 2 et client 3, ces données sont également transférées vers la base de données du serveur qui va devenir, comme le montre la figure 4.25 :

Nombre de lignes : 25

rier sur l'index: Aucune

Options

	Matricule	moyenne_s1 moyenne_s2	moyenne_s2 décision	decision create_a	create_a temps_mis_ajour	temps_mis_ajour
Modifier Copier Effacer	q14098	12.45	17	admin	2020-11-15 20:25:30	2020-11-15 20:25:30
Modifier Copier Effacer	Q150076	12.14	15.45	admin	2020-11-15 20:25:41	2020-11-15 20:25:41
Modifier Copier Effacer	q150195	15	15	admin	2020-11-14 00:00:00	2020-11-14 05:00:00
Modifier Copier Effacer	Q1501975	12	6	ajourner	2020-11-15 20:29:46	2020-11-15 20:29:46

FIGURE 4.25 – La base de données serveur après la synchronisation de ces trois clients.

Et la base de données de chaque client devient, comme le montre la figure suivante :

mémoire

LA LISTE DES MOYENNES D'ÉTUDIENTS

matricule| Moyenne s1| Moyenne s2| des

q14098|12.45 |17|admin

q150195|15 |15|admin

Q150076|12.14 |15.45|admin

Q1501975|12 |6|ajourner

FIGURE 4.26 – La base de données locale de chaque client après la synchronisation.

Pour voir ce qui se passe après la synchronisation dans le cas où les données existent dans le serveur et dans locale, mais qu'il y a des données modifiées en locale(même matricule, mais les autres attributs sont modifiés). Donc nous avons lancé un client qui a une base de données locale, comme indiqué dans la figure suivante :

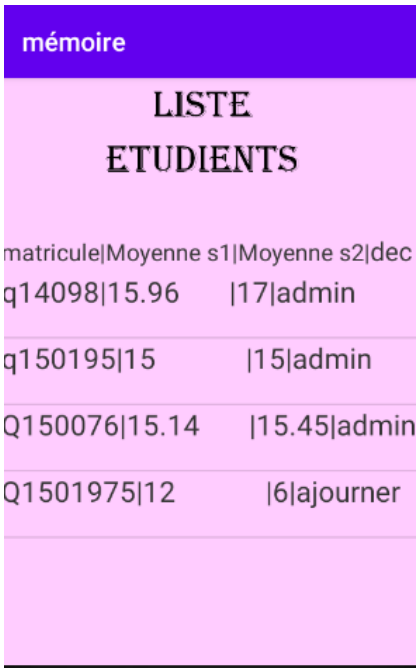


FIGURE 4.27 – La base de données locale.

Et la base de données du serveur soyez comme ce qui montre sur la figure suivante :

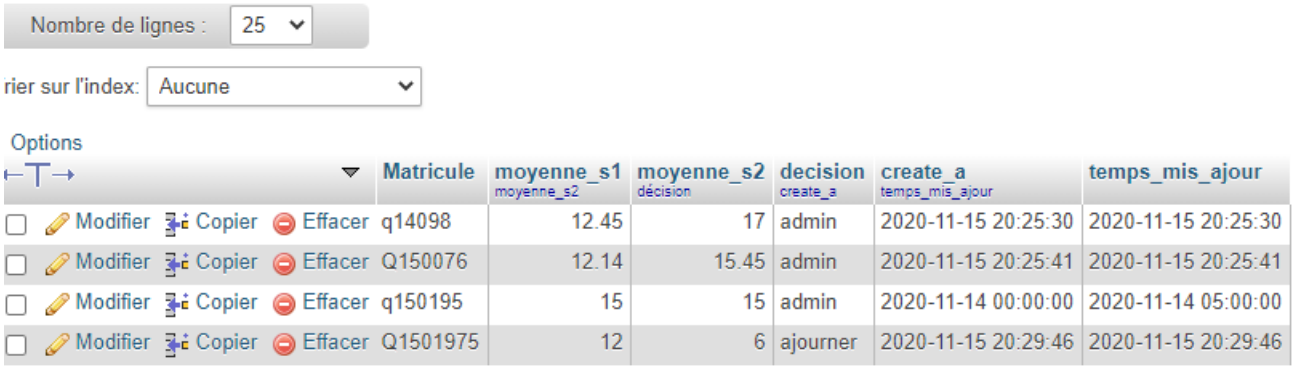


FIGURE 4.28 – La base de données du serveur.

D'après ces figures, nous avons remarqué que dans la base de données locale la ligne qui contient la matricule "q14098" a été modifié.
Lorsque nous avons lancé la synchronisation, il y avait l'ensemble de vérification qui était présenté dans l'algorithme bidirectionnelle de notre proposition et lorsque nous avons trouvé que le temps de création dans la bdd locale inférieur au temps de dernière synchronisation et le temps de mise à jour dans locale supérieur au temps de mise à jour dans le serveur, comme indiqué dans la figure suivante :

rowid	mat	moS1	mos2	dicition	Create_a
3	Q150076	15.14	15.45	admin	2020-11-15 20:25:41
1	Q1501975	12.0	6.0	ajourner	2020-11-16 09:42:43
2	q14098	15.96	17.0	admin	2020-11-05 11:05:36
4	q150195	15.0	15.0	admin	2020-11-14 00:00:00

FIGURE 4.29 – La liste des données dans la BDD locale.

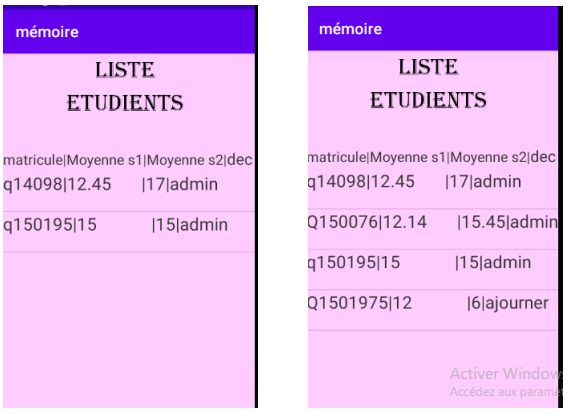
Après la fin de la synchronisation cette donnée ont été modifiées dans le serveur, comme indiqué dans la figure suivante :

Options

	Matricule	moyenne_s1 moyenne_s2	moyenne_s2 décision	decision create_a	create_a temps_mis_ajour	temps_mis_ajour
Modifier Copier Effacer	q14098	15.96	17	admin	2020-11-15 20:25:30	2020-11-18 11:10:02
Modifier Copier Effacer	Q150076	15.14	15.45	admin	2020-11-15 20:25:41	2020-11-15 21:33:41
Modifier Copier Effacer	q150195	14	14	admin	2020-11-11 10:10:10	2020-11-11 10:10:10
Modifier Copier Effacer	Q1501975	12	6	ajourner	2020-11-15 20:29:46	2020-11-15 20:29:46

FIGURE 4.30 – La base de données serveur après la mise à jour.

En seconde lieu, nous avons testé l'algorithme unidirectionnelle, et nous avons lancé deux clients qui consultent la base de données avant et après la synchronisation. Donc avant de faire la synchronisation, les données de base de données locale du « client 1 » et « client 2 » seront comme indiqué sur la figure 4.31 à la page 62, et la base de données du serveur sera comme indiqué sur la figure 4.32 à la page 63 :



(a) (b)

FIGURE 4.31 – (a) BDD locale du client 1 et (b) BDD locale du client 2 avant la synchronisation

Options

	Matricule	moyenne_s1 moyenne_s2	moyenne_s2 décision	decision create_a	create_a temps_mis_ajour	temps_mis_ajour
Modifier Copier Effacer	q14098	12.45	17	admin	2020-11-15 20:25:30	2020-11-15 20:25:30
Modifier Copier Effacer	Q150076	15.14	15.45	admin	2020-11-15 20:25:41	2020-11-15 21:33:41
Modifier Copier Effacer	q150195	15	15	admin	2020-11-14 00:00:00	2020-11-14 05:00:00
Modifier Copier Effacer	Q1501975	12	6	ajourner	2020-11-15 20:29:46	2020-11-15 20:29:46

FIGURE 4.32 – La base de données du serveur avant la synchronisation du client 1 et client 2.

Comme nous remarquons, dans le « client 1 », il n'a que deux étudiants dans sa propre base de données. Et il y a des données dans la bdd serveur qui ne sont pas dans la bdd locale, et après la synchronisation, les données qui ne sont pas présentes sont ajoutées à la base de données du client 1, comme le montre sur la figure 4.33 :

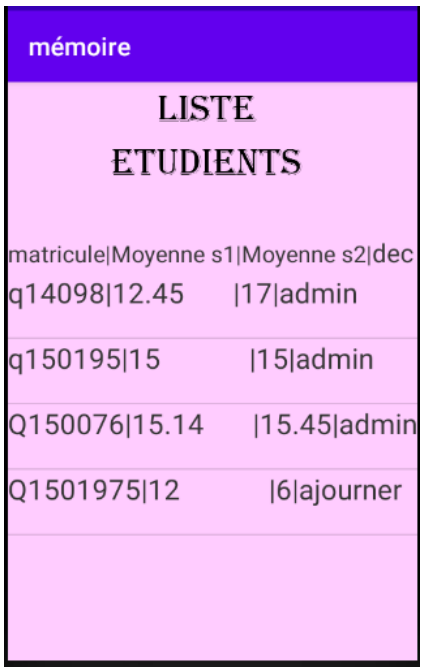


FIGURE 4.33 – La base de données locale du client 1 après la synchronisation.

Et dans le « client 2 » il contient les mêmes données que la base de données du serveur mais il y a des données qui sont modifiées lorsque le client fait la synchronisation, nous comparons l'heure de modification dans la base de données locale et dans la base de données du serveur, si la modification du serveur est supérieure à la modification de la base de données locale donc ça signifie que ces données doivent être mises à jour et c'est notre cas. Après la fin de synchronisation les données sont mises à jour dans la base de

données locale, comme indiqué dans la figure 4.34 :

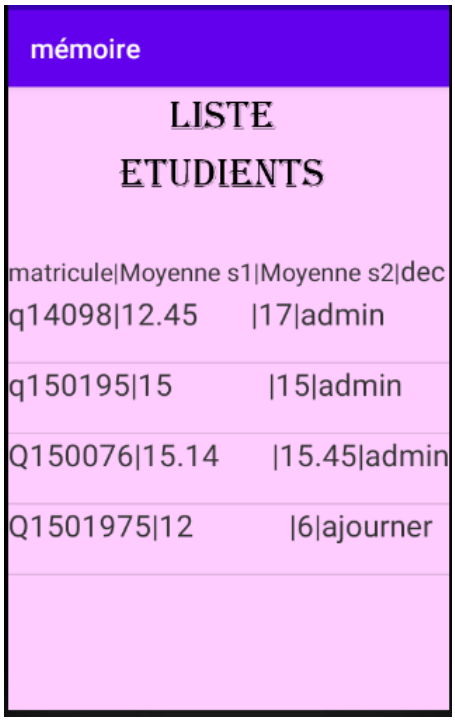


FIGURE 4.34 – La base de données locale du client 2 après la synchronisation.

4.5 Conclusion

Dans ce chapitre, nous avons parlé dans un premier lieu sur les différents langages, les outils de développement que nous avons utilisé afin d'implémenter notre solution.
Et dans un second lieu nous avons parlé sur la création de base de données interne et externe, des scripts php, la création des classes unidirectionnelle et bidirectionnelle et quelques captures sur le fonctionnement de notre solution.

Conclusion générale

Au cours de ce travail, nous avons proposé un modèle hybride, basé sur l'approche centralisée qui assure la disponibilité des services en mode en ligne et hors ligne. Cela en exploitant les méthodes de synchronisation(unidirectionnelle et bidirectionnelle) et de réplication.

Nous avons commencé par un aperçu des bases de données, des applications web et la synchronisation des données ainsi que la réplication. Puis nous avons montré les différents modèles et méthodes de synchronisation actuels qui nous ont aidés à proposer notre modèle.

Pour finir, avant de passer aux perspectives, ce travail nous a permis de mettre en pratique nos connaissances sur la synchronisation et la réplication des données et d'autres choses.

Il reste beaucoup de choses à améliorer dans le modèle proposé, par exemple :
Réduire les données stockées dans le laps de temps entre l'envoie de requête et la réponse de serveur.

Bibliographie

[1] Groupe MadeInFutura, FUTURA TECH, 2001-2020, disponible à l'adresse : https ://www.futura-sciences.com/tech/definitions/informatique-base-donnees-518/ , consulter le 26/03/2020

[2] Groupe Publithings, LeBigData, 2020, disponible à l'adresse : https ://www.lebigdata.fr/base-de-donnees , consulter le 26/03/2020

[3] Delvin DIUMI OMOKOKO, conception et réalisation d'une base de données pour la gestion de facturation à l'office congolais de contrôle direction provinciale du Kasaà occidental. Université Notre-Dame du Kasayi - Diplôme de graduat en informatique de gestion 2009.

[4] https ://laurent-audibert.developpez.com/Cours-BD/ ?page=introduction-bases-de-donnees#L1-1-2-a, consuler le 28/03/2020

[5] *Base de données*, disponible à l'adresse, https ://www.base-de-donnees.com/comprendre-bases-de-donnees/les-4-types-de-bases-de-donnees/, consuler le 26/03/2020

[6] Bernd AMANN, Michel SCHOLL, « SYSTÈMES INFORMATIQUES- Systèmes de gestion de bases de données », Encyclopædia Universalis [en ligne], 2020, disponible à l'adresse : https ://www.universalis.fr/encyclopedie/systemes-informatiques-systemes-de-gestion-de-bases-de-donnees/, consulter le 26/03/2020

[7] Sandhu, Ravi S and Feinstein, Hal, A three tier architecture for role-based access control, Proc. of the 17th NIST-NCSC National Computer Security Conference, 138–149, 1994, consulter 02/12/2020

[8] *MYSCENARI*, disponible à l'adresse : https ://stph.scenari-community.org/bdd/0/co/pri1c21.html, consulter le 29 /03/2020

[9] Eric Quinton. La sécurisation d'une application web : Risque, chiffrement et traitement des vulnérabilités avec PHP. ISTE Group, page 25, 2017.

[10] *ENIS Android Club*, 2020, disponible à l'adresse : http ://enis-androidclub.blogspot.com/2014/04/tutoAndroidMysql.html, consulter le 10/10/2020.

[11] Andreas Anyuru, Professional WebGL programming : developing 3D graphics for the Web.John Wiley & Sons, page 2, 2012.

[12] Mikaël Pirio, Apache (version 2) : installation, administration et sécurisation, page : 33-34.

[13] *SlidePlayer*, 2020, disponible à l'adresse : http ://slideplayer.fr/slide/173472/, consulter le 06/10/2020.

[14] Malhotra, Naveen et Chaudhary, Anjali. Implementation of Database Synchronization Technique between Client and Server.International Journal Of Engineering And Computer Science,3(07),2014.

[15] *MindOrks*, 2017-2020, disponible à l'adresse : https ://blog.mindorks.com/firebase-realtime-database-android-tutorial, consulter le 22/04/2020

[16] *Junto*,2020, disponible à l'adresse : https ://junto.fr/blog/firebase/, consulter le 22/04/2020.

[17] https ://hackernoon.com/introduction-to-firebase-218a23186cd7, consulter le 22/04/2020.

[18] https ://firebase.google.com/docs/database/android/offline-capabilities, consulter le 22/04/2020.

[19] Shodiq, Muhsin and Wongso, Rini and Pratama, RendySetya and Rhenardo, Eko and others. Implementation Of Data Synchronization With Data Marker Using Web Service Data. Procedia Computer Science,59 :366—372,2015.

[20] M. Y. Choi, E. A. Cho, D. H. Park, J. Y.Bae, C. J. Moon, and D. K. Baik, "A synchronization algorithm of mobile database for ubiquitous computing," in NCM 2009 - 5th International Joint Conference on INC, IMS, and IDC, 2009, pp. 416–419.

[21] V. Balakumar and I. Sakthidevi, "An efficient database synchronization algorithm for mobile devices based on secured message digest," in 2012 International Conference

on Computing, Electronics and Electrical Technologies, ICCEET 2012, 2012, pp. 937–942.

[22] T. A. Alhaj and M. M. Taha, "Synchronization Wireless Algorithm Based on Message Digest (SWAMD) For Mobile Device Database," in International Conference on Computing, Electrical and Electronic Engineering (ICCEEE), 2013, pp. 259–262.

[23] B. Saravanan, J, A. Mary, and J. Jagadeesan, "A DATABASE SYNCHRONIZATION ALGORITHM FOR MOBILE DEVICES," Int. J. Comput. Sci. Mob. Appl., vol. 2, no. 4, pp. 46–58, 2014.

[24] Kekgathetse, Mopati B and Letsholo, Keletso J. A survey on database synchronization algorithms for mobile device. Journal of Theoretical and Applied Information Technology, 86(1) : 1-9, 2016.

[25] D. Sethia, S. Mehta, A. Chwodhary, Replicated Database Management Synchronization," in 2014 International Conference on Signal Processing and Integrated Networks, 2014, pp. 624–631.

[26] Y. Jiao, Z. Jin, and Z. Ma, "A cross-layer method to improve mobile database synchronization performance," in Proceedings - 5th International Conference on Wireless Communications, Networking and Mobile Computing, WiCOM 2009, 2009, no. 1, pp. 2–5.

[27] Y. Li, X. Zhang, and Y. Gao, "Objectoriented data synchronization for mobile database over mobile ad-hoc networks," in 2008 International Symposium on Information Science and Engineering, ISISE 2008, 2008, vol. 2, pp. 133–138.

[28] N. Malhotra and A. Chaudhary, "Implementation of database Synchronization Technique between Client and Server," Int. J. Eng. Sci. Innov. Technol., vol. 3, no. 4, pp. 460–465, 2014.

[29] J. Zechmeister, D. Zak, T. Vana, and J. Lebduska, "Realization of Data Transfer between Mobile Device and Oracle Database," in Proceedings of International Conference on Advances in Communication and Information Technology 2012, pp. 48–50.

[30] B. Lee, T. Kim, D. Kim, and C. National, "Data synchronization protocol in mobile computing environment using S y ncML," in 5th IEEE International Conference on

High Speed Networks and Multimedia Communication (Cat. No.02EX612), 2002, pp. 133–137.

[31] J.-L. Li and J.-P. Li, "Data synchronization protocol in mobile computing environment using SyncML and Huffman coding," Wavelet Act. Media Technol. Inf. Process. (ICWAMTIP), 2012 Int. Conf., pp. 260–262

[32] Z. Huang, Q. Chang, Z. Shen, Y. Zhou, B.Yan, and L. Liu, "Design and Implementation of an XML-Based Universal Mobile Data Acquisition System," 2009 Fifth Int. Conf. Semant. Knowl. Grid, 2009, pp. 456–457.

[33] K. Miller, C. Gee, R. Inaba, T. Ozyer, A. Lo, and R. Alhajj, "Synchronization of Mobile XML Databases by Utilizing Deferred Views," in Information Reuse and Integration,. IRI 2004. Proceedings of the 2004 IEEE International Conference on, 2004, pp. 186 – 191.

[34] Z. Y. Lu and Z. B. Guo, "A Method of Data Synchronization Based on Message Oriented Middleware and Xml in Distributed Heterogeneous Environments," in Proceedings of International Conf. on Artificial Intelligence and Industrial Eng. no. Aiie, pp. 210–212, 2015.

[35] GÉNIE LOGICIEL, https ://inf1410.teluq.ca/teluqDownload.php ?file=2014/01/INF1410-PresentationStarUML.pdf, consulter le 21/11/2020.

[36] https ://www.journaldunet.fr/web-tech/dictionnaire-du-webmastering/1203555-java-definition/, consulter le 30/09/2020.

[37] *nfoWebMaster*,2007-2020, disponible à l'adresse : http ://glossaire.infowebmaster.fr/php/, consulter le 30/09/2020.

[38] *TechTarget*, 2003-2020, disponible à l'adresse : https ://searchmobilecomputing.techtarget.com/definition/Android-Studio, consulter le 30/09/2020.

[39] https ://www.wampserver.com/, consulter le 30/09/2020.

[40] https ://www.json.org/json-en.html, consulter le 10/10/2020.

[41] https ://riptutorial.com/fr/postman/example/30196/qu-est-ce-que-postman-, consulter le 10/10/2020.

[42] https ://vogella.developpez.com/tutoriels/android/utilisation-base-donnees-sqlite/, consulter le 10/10/2020.

[43] BRAND UP DIGITAL, *COORSALINE*, 2017, disponible à l'adresse : https ://www.coursaline.com/tutoriels/base-donnees/replication-base-de-donnees# : :text=Les%20serveurs%20r%C3%A9pliqu%C3%A9s%20sont%20de,un%20ou%20plusieurs%20serveurs%20esclaves , consulter le 18/11/2020.

[44] Emna Guermazi, Application web pour la gestion de la bibliothèque. Institut supérieur d'informatique et du multimédia de sfax (ISIMS) - Ingénieur en informatique, techniques web et multimédia 2010. Memoireonline : https ://www.memoireonline.com/04/11/4453/m_Application-web-pour-la-gestion-de-la-bibliotheque7.html, consulté le 03/12/2020.