

RAPPORT DE STAGE

Elaboré par



Application mobile de transfert des fichiers à base de serveur Web Http

Réalisé au sein de

DEDICACES

Toutes les lettres ne sauraient trouver les mots qu'il faut...

Tous les mots ne sauraient exprimer la gratitude, l'amour, le respect, la reconnaissance...

Aussi, c'est tout simplement que je dédie ce projet de fin d'études...

Remerciements

C'est avec une grande gratitude et reconnaissance que je réserve ces lignes qui s'adressent à toutes les personnes qui ont contribué à l'élaboration de ce projet de fin d'études et qui m'ont aidé lors de la rédaction de ce rapport.

Sommaire

Dédicaces	2
Remercîments	3
Sommaire	4
Table des figures.....	7
Liste des abréviations	10
Introduction	11
Présentation de la société.....	11
I.1 Cycle de vie du projet de développement de l'application mobile	14
I.2 Introduction des phases	14
I.2.1 Conception Fonctionnelle	14
I.2.2 Design de l'interface Utilisateur.....	15
I.2.3 Codage.....	15
I.2.4 Test.....	.
I.3 Etude de besoin de l'application
I.3.1 Diagramme de cas d'utilisation globale
I.3.2 Diagramme MCD globale
I.3.3 Architecture d'une application Android
I.3.3.1 La composante <Activity>
I.3.3.2 La composante <Fragment>
I.4 Cycle de développement de l'application
I.4.1 Détermination des activités de l'application.....	.
I.4.1 Activité Connexion.....	.
I.4.1.1 Conception Fonctionnelle.....	.
I.4.1.2 Design de l'interface utilisateur.....	.
I.4.1.3 Conception de l'architecture de l'activité.....	.
I.4.1.4 Codage de l'activité

I4.1.5 Test de l'activité	
I.4.2 Activité Configuration.....	
I 4.2.1 Conception Fonctionnel.....	
I 4.2.2 Design	
I 4.2.3 Conception architecture.....	
I 4.2.4 Codage.....	
I.4.3 Activité File Transfer	
I 4.3.1 Conception Fonctionnel.....	
I 4.3.2 Desgin.....	
I 4.3.3 Conception architecture.....	
I 4.3.4 Conception codage.....	
I.4.4 Activité Administration.....	
I 4.4.1 Conception Fonctionnel.....	
I 4.4.2 Desgin.....	
I 4.4.3 Conception architecture.....	
I 4.4.4 Codage.....	
I.5 Validation	
I.6 Conclusion.....	

Listes des figures

Figure 1. Modèle en spirale	
Figure 2. Diagramme de cas d'utilisation globale.....	
Figure 3. Diagramme MCD globale	
Figure 4. Cycle de vie d'une activité.....	
.	

Listes des abréviations

UML: Unified Modeling Language

XML: eXtensible Markup Language

Introduction

Nous avons préparé un projet qui consiste à concevoir et développer une application Android mobile pour transférer des fichiers.

Dans un premier temps, on va analyser les besoins concernant cette application, en indiquant : les besoins fonctionnels et non fonctionnels, une étude de l'existant et des solutions pour la réalisation. On exploite cette partie pour présenter l'entreprise, le cadre du stage et décrire de façon détaillée le sujet.

Ensuite, on va décrire la phase de conception, en présentant des diagrammes du langage UML comme : les diagrammes de cas d'utilisation générale et détaillé en précisant les acteurs impliqués dans cette application, le diagramme de classes, les diagrammes d'activités.

Enfin, on va décrire la phase de la réalisation et du codage de l'application en indiquant les outils informatiques utilisés tout au long du développement. On va présenter les interfaces inclus dans l'application et on va indiquer des techniques pour réaliser cette application mobile.

Il s'agit en particulier de développer une application de transfert des fichiers à base de serveur web http qui sera réalisé avec trois technologies différentes :

- la 1ère technologie consiste à développer l'application avec le langage Java sur Android studio.

- La 2ème technologie consiste à développer l'application avec le langage Dart, le Framework Flutter sur Android studio .

-la 3ème technologie consiste à développer l'application avec le langage JavaScript et la technologie ReactNative.

Ce rapport se limitera à la réalisation du projet avec la 1^{er} technologie dont la période sera programmée pendant toute la durée du stage.

Après la réalisation du projet avec les trois technologies une étude comparative sera élaborée permettant d'énumérer les avantages et les inconvénients de chacune des trois technologies.

1-Cycle de vie du projet de developement de l'application mobile :

Le « cycle de vie d'un logiciel » (en anglais software life-cycle), ou les étapes mé1/thodes de développement, sélectionne et identifie toutes les étapes de développement d'un logiciel, dès sa conception en allant jusqu'à disparition. L'objectif était de définir des balises intermédiaires permettant la validation de la partie développement logiciel, c'est-à-dire si le logiciel répond aux besoins exprimés, et la vérification du processus de développement, et si les méthodes mises en œuvre respectant bien les contraintes prédéfinies auparavant.

-Cycle en spirale :

Le modèle en spirale (*spiral model*) est un modèle de Cycle de développement logiciel qui reprend les différentes étapes du cycle en V. Par l'implémentation de versions successives, le cycle recommence en proposant un produit de plus en plus complet et dur. Le cycle en spirale met cependant plus l'accent sur la gestion des risques que le cycle en V.

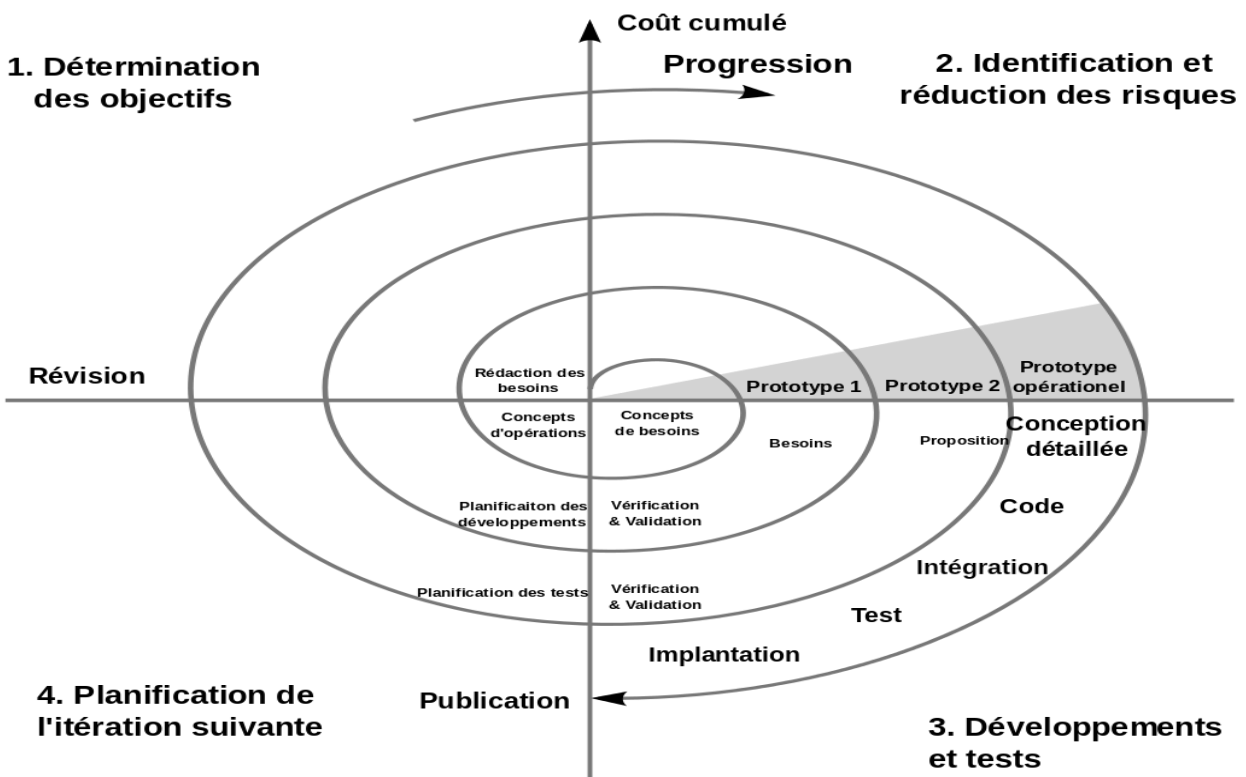


Figure : Modèle en spirale

On commence d'abord par l'étude des besoins fonctionnelles :

L'**étude de besoins** est un élément déterminant dans la démarche projet. Elle permet de réaliser un diagnostic stratégique aidant ensuite à s'engager dans une réflexion de projet en cohérence avec le territoire.

On a réalisé en particulier ce qui suit :

1. Le diagramme de cas d'utilisation global
2. Le modèle conceptuel de donnée global (MCD).
3. La détermination de la technologie à adopter (AndroidSDK) .

D'après le diagramme de cas d'utilisation on a identifié 4 activités principales pour notre application :

1. Connexion
2. Configuration
3. Administration
4. Files Transfer

Pour chaque Activité on passe par les 5 phases suivantes :

1. Conception Fonctionnelle :
2. Design des interfaces d'utilisateur :
3. Conception de l'architecture
4. Codage
5. Test

2-Introduction des phases :

2.1-Conception Fonctionnelle :

Nous avons utilisé pour la partie conception fonctionnelle le langage UML

(Unified Modeling Language) est un langage de modélisation unifié permet de modéliser une application logicielle d'une façon standard dans le cadre de conception orienté objet.

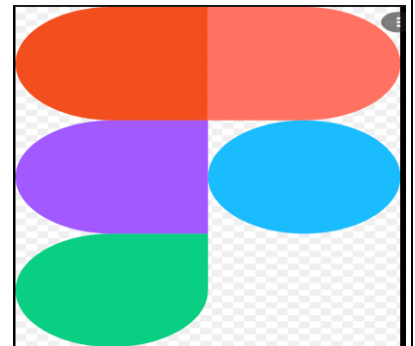
-On à utilis  en particulier les deux diagrammes cas d'utilisation et diagramme d'activit  .

-def diagramme cas utli et diagrm d'activit s .

2.2-Design de l'interface utilisateur :

Nous avons utilis  pour la partie design « Figma.com» :

Figma est un  diteur de graphiques vectoriels et un outil de prototypage. Il est principalement bas  sur le web, avec des fonctionnalit s hors ligne suppl mentaires activ es par des applications de bureau pour macOS et Windows.



3.3-Conception de l'architecture

La m thodologie merise - +MCD de la BD

-MCD : Le **MCD (Mod le Conceptuel des Donn es)** est une repr sentation graphique de haut niveau qui permet facilement et simplement de comprendre comment les diff rents  l ments sont li s entre eux   l'aide de diagrammes codifi s .

-MPD : Le **MPD (Mod le Physique des Donn es)** : L' tape de cr ation du MPD est presque une formalit  compar e   la cr ation du MCD. En s'appuyant sur des r gles simples (et qui fonctionnent   tous les coups), l'analyste fait  voluer sa mod lisation de haut niveau pour la transformer en un sch ma plus proche des contraintes des logiciels de bases de donn es. Il s'agit de pr parer l'impl mentation dans un SGBDR.

-en particulier on a utilisé les modèles fonctionnelles et les modèles physiques :

Nous avons utilisé le logiciel Power AMC pour modéliser le diagramme ER (entité-relation) servira à modéliser le MCD (modèle conceptuel des données) de la base de données, c.-à-d. la structure de la base de données qui va contenir les informations sur les utilisateurs et leurs connexions.

MERISE :

MERISE (Méthode d'Etude et de Réalisation Informatique pour les Systèmes d'Entreprise) est une méthode d'analyse et de réalisation des systèmes d'information qui est élaborée en plusieurs étapes : schéma directeur, étude préalable, étude détaillée et la réalisation.

4.4-Codage

Nous allons présenter l'environnement matériel et logiciel utilisé pour le développement de la solution proposé tout en expliquant éventuellement nos choix techniques relatif aux langages de programmation et des outils utilisés. Enfin, nous allons donner une présentation des interfaces globales ainsi qu'une description du fonctionnement du système.

○ Technologies utilisées dans l'application:

Android SDK (Software Development Kit) :

Android est un système d'exploitation mobile pour Smartphones, tablettes tactiles, PDA, smartwatches (version Wear) et terminaux mobiles.



AndroidStudio IDE (Integrated Development Editor) :

Android Studio est un environnement de développement pour développer des applications mobiles Android. Il est basé sur IntelliJ IDEA et utilise le moteur de production Gradle. Il peut être téléchargé sous les systèmes d'exploitation Windows, macOS, Chrome OS et Linux.



Git VCS (version Control System): Git est de loin le système de contrôle de version le plus largement utilisé aujourd'hui. Git est un projet open source avancé, qui est activement maintenu.



SourceTree GUI (Graphical User Interface) : Une interface graphique Git qui offre une représentation visuelle de vos référentiels. Sourcetree est un client Git gratuit pour Windows et Mac.



5.Test :

3-Etude de besoin de l'application :

1/-Diagramme de cas d'utilisation globale :

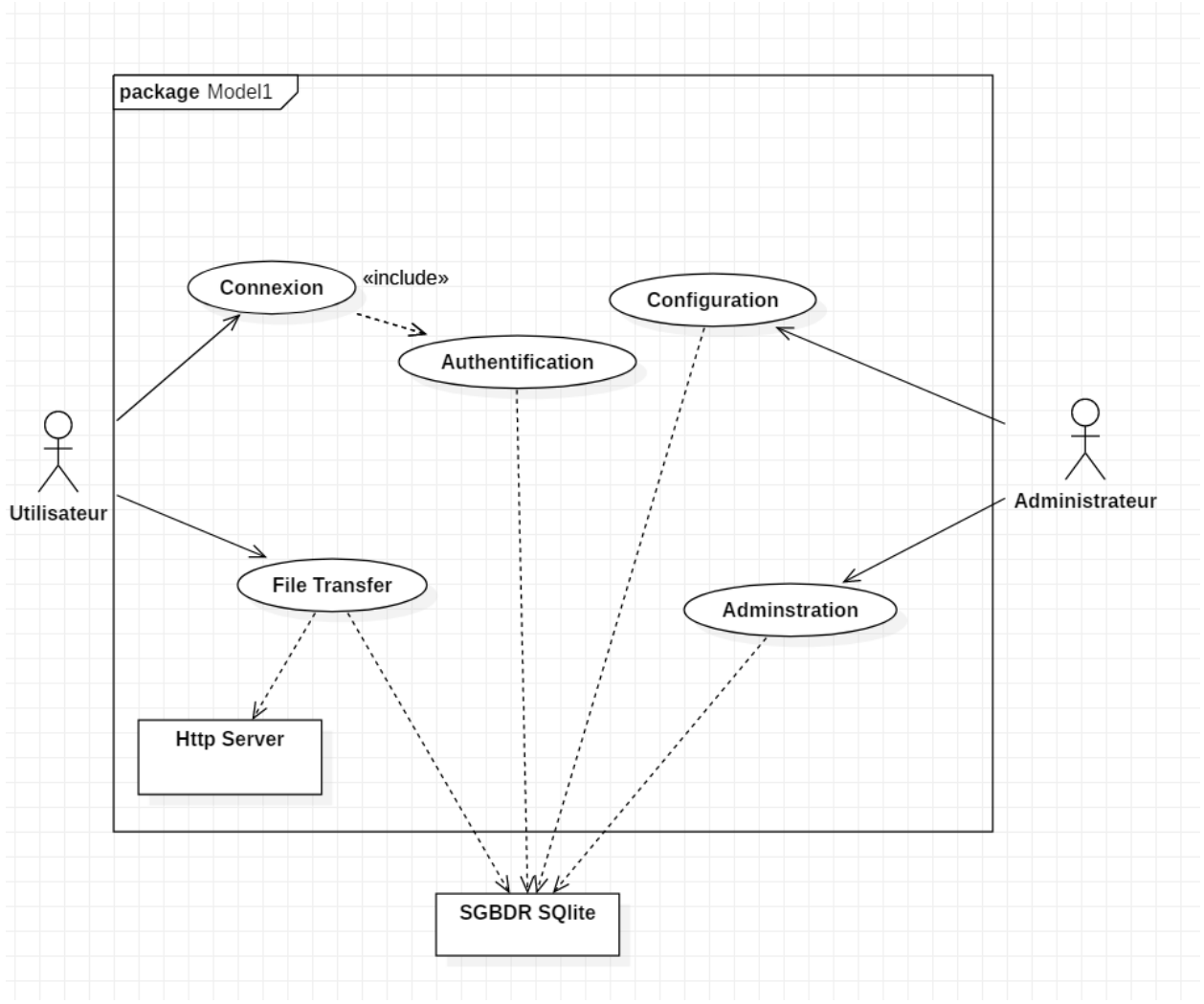


Figure : Diagramme de cas d'utilisation globale

D'après ce diagramme il y a 4 activités a prendre en considération par notre application mobile :

-Connexion

-Administration

-Configuration

-File Transfer

2/Diagramme MCD globale :

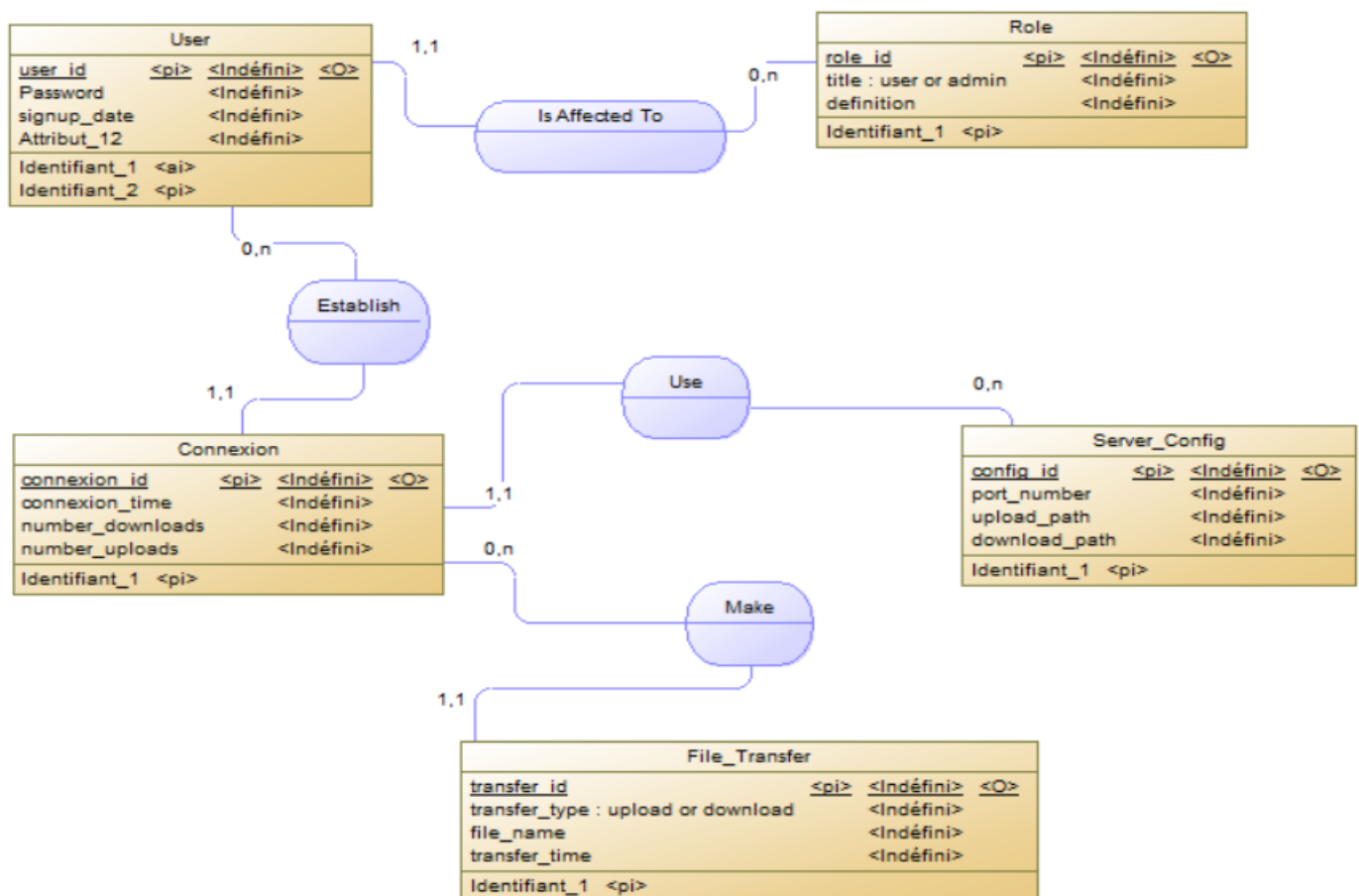


Figure : Diagramme MCD globale

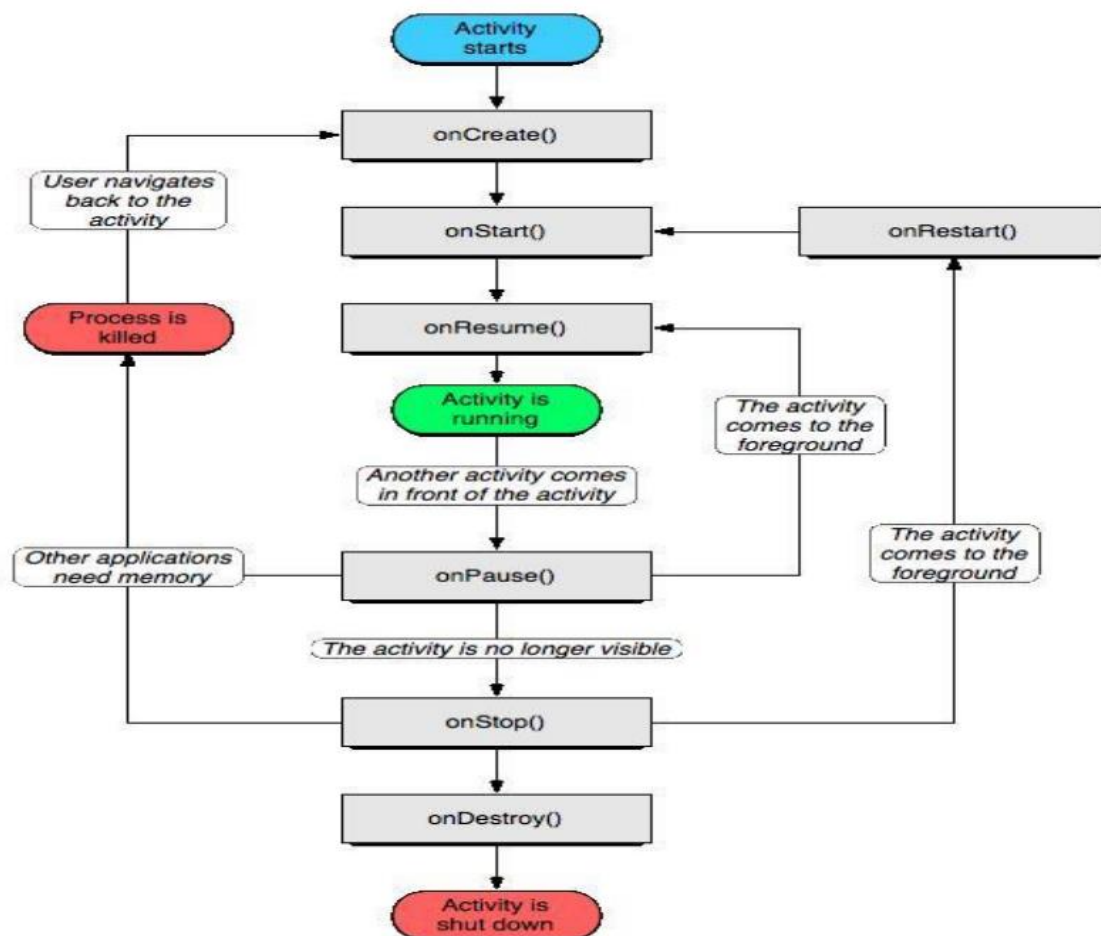
3-Architecture d'une application Android :

-une app android comporte de deux composantes
« Activity » et « Fragment»

3.1-La composante « Activity »:

Faisons un peu de théorie. Une activité représente en gros ce que l'on voit à l'écran, pour la définir simplement, elle est généralement composée d'une classe, et d'un layout xml.

Une activité est la composante principale pour une application Android. Elle représente l'implémentation métier dans une application Android, permettant de gérer l'ensemble des vues et ressources. Une activité peut être avec ou



sans interface utilisateur. Il est possible d'avoir plusieurs activités dans le même programme. Elle doit toujours être déclarée dans le fichier AndroidManifest.xml. Une activité n'est pas linéaire, elle est soumise à plusieurs événements. Chaque événement est représenté dans une méthode.

Figure : Cycle de vie d'une activité

La composante « Fragment »:

Fragment est une partie d'une activité, qui contribue à sa propre INTERFACE utilisateur pour cette activité. Fragment peut être considéré comme une sous-activité. Ils sont utilisés pour une utilisation efficace de l'espace dans l'ensemble de l'écran des appareils.

Une activité peut contenir 0 ou plusieurs fragments basé sur la taille de l'écran. Un fragment peut être réutilisé dans de multiples activités, de sorte qu'il agit comme un composant réutilisable dans les activités.

Un fragment qui ne peuvent pas exister indépendamment. Il convient toujours de participer à une activité. Où que l'activité peut exister avec tout fragment en elle.

Un fragment du cycle de vie est plus complexe que l'activité du cycle de vie parce qu'il a plus d'états



Figure : Cycle de vie d'un fragment

4-Cycle de développement de l'application :

4.1-Activité Connexion :

Dans cette partie nous avons travaillé sur l'Activité Connexion de l'application Android de notre projet.

1. Conception Fonctionnelle UML:

-Diagramme cas d'utilisation pour l'activité Connexion :

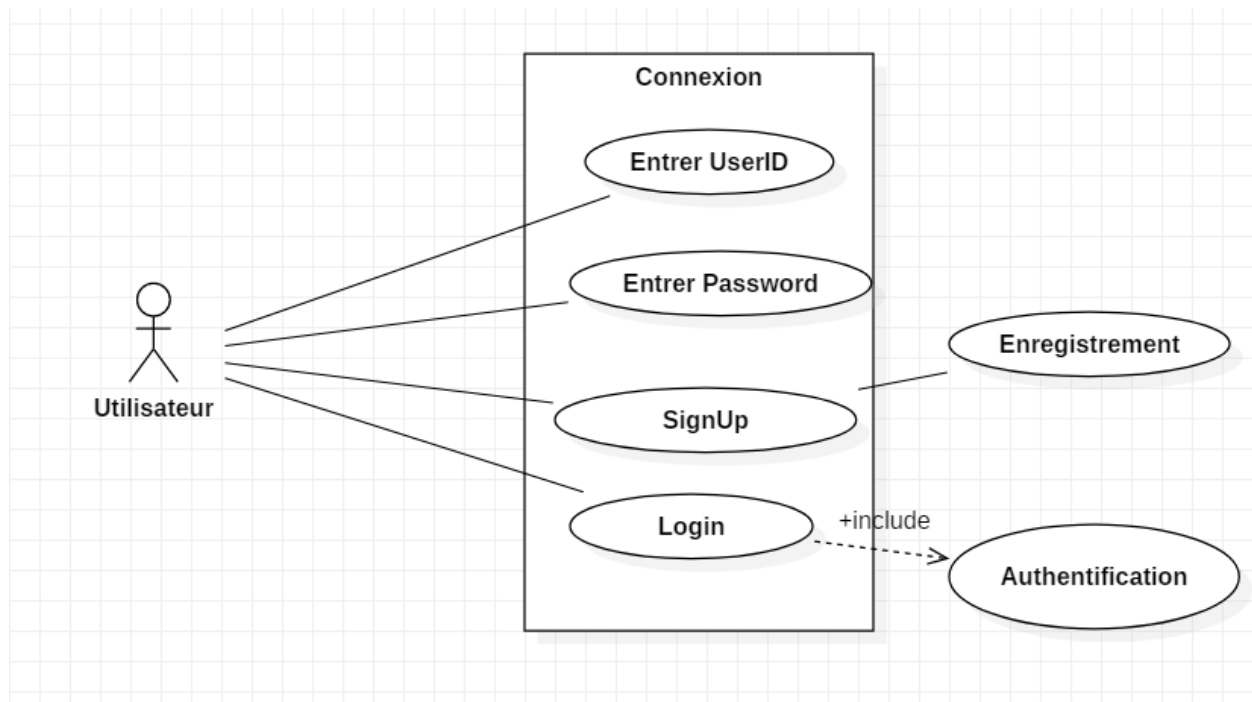


Figure : Diagramme cas d'utilisation

-Diagramme d'activité pour chaque cas d'utilisation :

*UML pour la page de « SignUp » :

Dans cette partie l'utilisateur créer un compte.

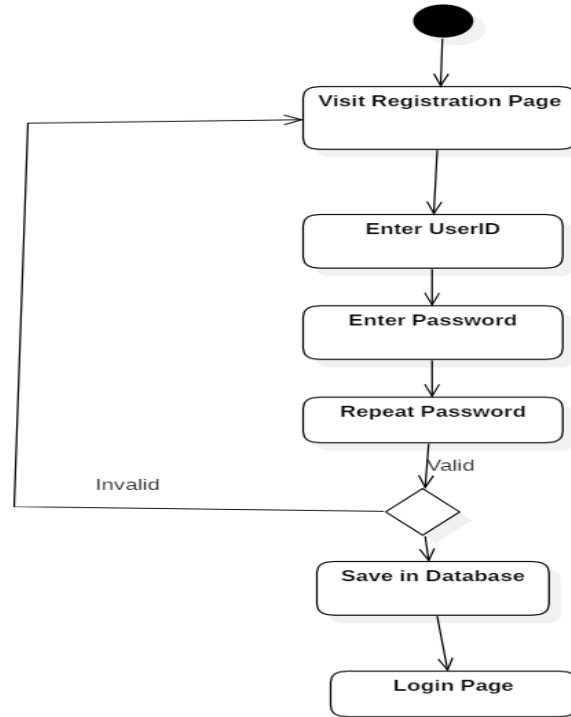


Figure : UML pour la page « SignUp »

*UML pour la page de Connexion (Login) :

Dans cette partie l'utilisateur saisie (userid)
Et (password) pour s'identifier a l'application.

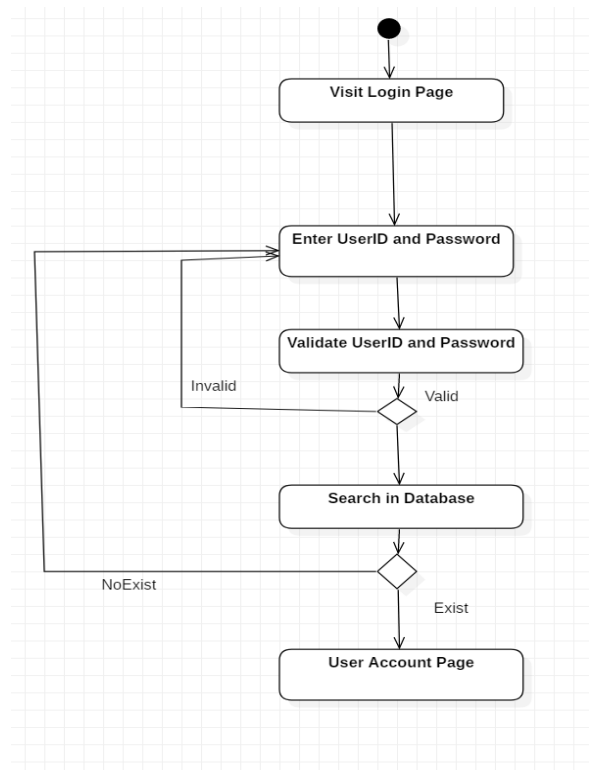


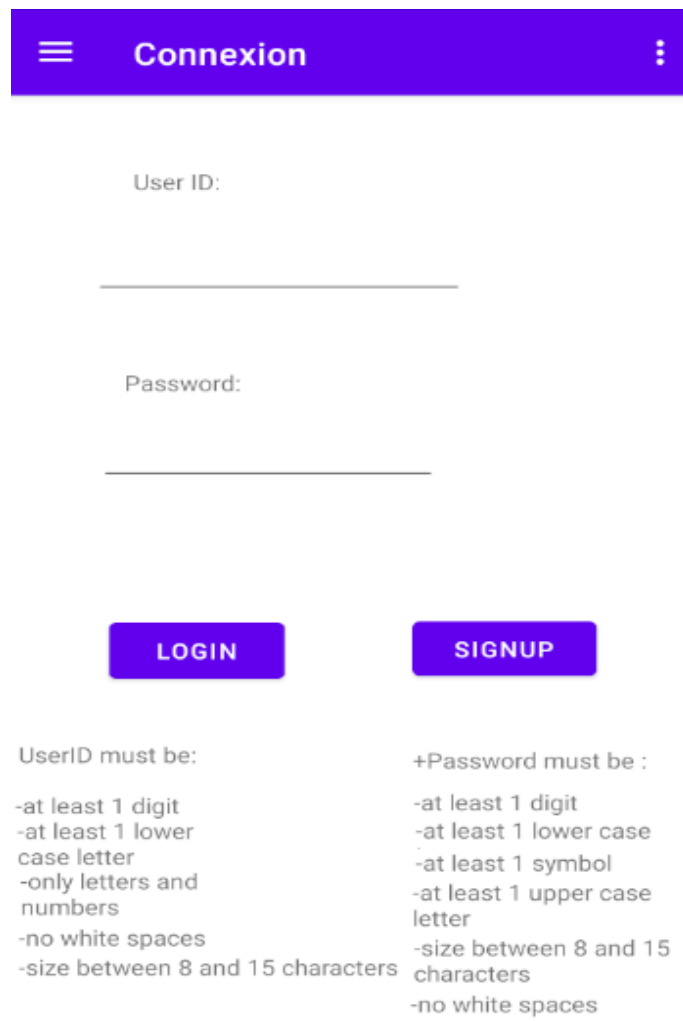
Figure :UML pour la page « Login »

2. Design :

Nous avons utilisé pour la partie design « Figma.com » :

Figma est un éditeur de graphiques vectoriels et un outil de prototypage. Il est principalement basé sur le web, avec des fonctionnalités hors ligne supplémentaires activées par des applications de bureau pour macOS et Windows.

Pour lancer l'application il faut créer un compte dans l'activité Connexion lors de clique sur le bouton « Signup » il faut créer un « User ID » et « Password » selon les conditions Ci-dessous la figure de l'activité Connexion :



The image shows a web form for user authentication. At the top is a dark blue header with a hamburger menu icon, the word "Connexion" in white, and a vertical ellipsis icon. Below the header are two input fields: "User ID:" and "Password:". Each field has a horizontal line representing the input area. Below the input fields are two blue buttons: "LOGIN" and "SIGNUP". At the bottom, there are two columns of text detailing the requirements for the User ID and Password.

User ID must be:

- at least 1 digit
- at least 1 lower case letter
- only letters and numbers
- no white spaces
- size between 8 and 15 characters

+Password must be :

- at least 1 digit
- at least 1 lower case
- at least 1 symbol
- at least 1 upper case letter
- size between 8 and 15 characters
- no white spaces

Figure : Activité Connexion

Pour la création d'un nom utilisateur il faut respecter les règles suivantes :

- au moins 1 chiffre
- au moins une lettre minuscule
- au moins une lettre majuscule
- n'importe quelle lettre
- pas d'espace.
- Taille entre 8 et 15 caractères.

Pour la création de mot de passe sécurisé dans l'application il doit comporter :

- au moins 1 chiffre
- au moins une lettre minuscule
- uniquement des lettres minuscules et des chiffres
- pas d'espace.
- Taille entre 8 et 15 caractères.

3. Conception d'architecture:

-MCD (table user and table connexion)

-MPD

- Modélisation de Base de données avec la méthodologie merise :
- Diagramme entité relation :

Table User : Enregistrer les utilisateurs.

Cette table contient les champs : User-id, Password et Signup_date.

Table Connexion : Enregistrer les connexions d'un utilisateur dans la base donnée, chaque utilisateur faire la connexion est enregistré dans la table de connexion. Cette table contient les champs : Connexion_id, Connexion_time, number_downloads et number_uploads.

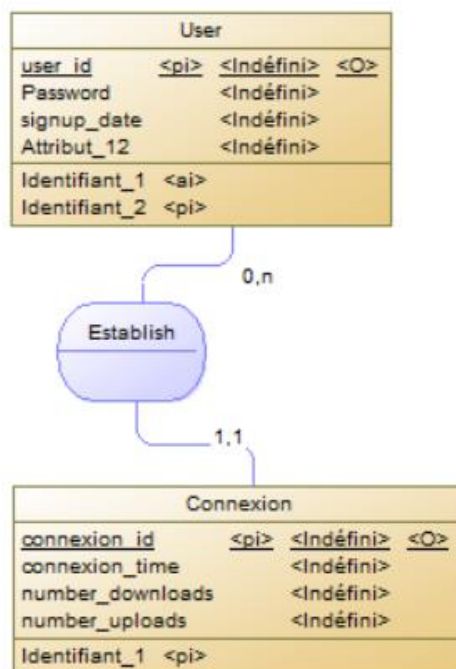


Figure : Diagramme entité relation :

Diagramme MPD (Multi Protocol Driver) :

le modèle physique des données consiste à implanter une base de données dans un SGBDR. Le langage utilisé pour ce type d'opération est le SQL.

Table User : Enregistrer les utilisateurs.

Cette table contient les champs : User-id, role_id, password et Signup_date.

Table Connexion : Enregistrer les connexions d'un utilisateur dans la base donnée, chaque utilisateur faire la connexion est enregistré dans la table de connexion. Cette table contient les champs : Connexion_id, User, config_id, Connexion_time, number_downloads et number_uploads.

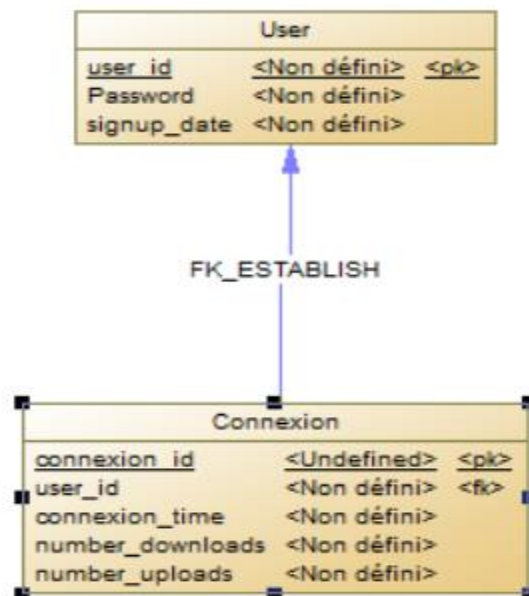
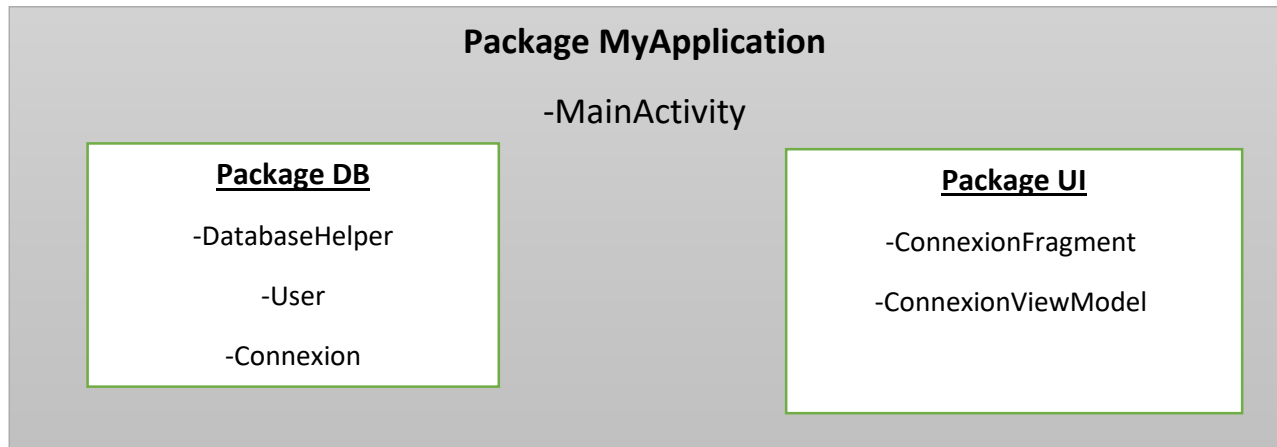


Figure : Diagramme MPD

4. Codage :

- les classes DatabaseHelper , Connexion.java, mainActivity
- interface : Connexionfragment , viewmodel.



-MainActivity : est la classe de démarrage de l'application .

-Le Pacakage db : contient les classes suivantes :

-DatabaseHelper : la création de la base de donnée , la requête pour la création des tables <User> et <Connexion> ainsi la création de la méthode <adduser> et <addConnexion>

-User : Création de la table User avec les champs <user_id>,<password> et <signup_date> ainsi que le constructeur et getter et setter dans la classe <User>

-Connexion : Création de la table Connexion avec les champs <connexion_id>,<connexion_time>,<connexion_downloads>,<connexion_uploads> et <user_id> ainsi que le constructeur et getter et setter_

-Le Pacakage Ui : contient les classes suivantes :

-ConnexionFragment : détecteur des évènements des clicks des buttons (Login) et (Signup).

-ConnexionViewModel :

Activité Configuration :

1. Conception Fonctionnel

2. Design

3. Conception d'architecture :

-Protocol tcp/ip : (Transmission Control Protocol/Internet Protocol) réunit les deux protocoles TCP et IP. Il s'agit donc d'une suite de protocoles associée au domaine d'Internet pour lequel elle facilite le transfert de données.

Présenté simplement, le protocole **TCP/IP** est un standard de communication entre deux processus. Il détermine et fixe les règles inhérentes à l'émission et à la réception de données sur un réseau. L'association des deux protocoles permet d'apporter des garanties de fiabilité dans le transfert des données. Avec le **TCP/IP**, vous êtes certain(e) que les informations envoyées arriveront bel et bien au bon destinataire. (+photo)

-Protocol http : HTTP signifie « **Hypertext Transfer Protocol** ». Ce protocole a été développé par Tim Berners-Lee au CERN (Suisse) avec d'autres concepts qui ont servi de base à la création du World Wide Web : le HTML et l'URI. Alors que le HTML (Hypertext Markup Language) définit comment un site Internet est construit, le HTTP détermine comment la page est transmise du serveur au client. Le troisième concept, l'URL (Uniform Resource Locator), fixe la façon dont une ressource (par exemple un site Internet) doit être adressée sur le Web. (+photo).

-Protocol ftp :

-Différence entre http et ftp ☺+photo

-Socket : Une socket est connue comme un type de logiciel qui agit comme un point d'extrémité qui fonctionne en établissant une liaison de communication réseau bidirectionnelle entre l'extrémité du serveur et le programme de réception du client. On l'appelle aussi souvent un point d'aboutissement dans un canal de communication bidirectionnel. Ces sockets sont réalisés et mobilisés en même temps qu'un ensemble de requêtes de programmation identifiées comme appels de fonction, qui est techniquement appelé interface de programmation d'application (API). Une socket est capable de simplifier le fonctionnement d'un programme car les programmeurs n'ont plus qu'à se soucier de

manipuler les fonctions de la socket, ce qui leur permet de compter sur le système d'exploitation pour transporter correctement les messages sur le réseau.

4. Codage

5. Test