***RAPPORT DE STAGE***

###### C:\Users\Admin\Desktop\a propos mon stage\des logos\apple-touch-icon@2 - Copie.pngElaboré 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

Remercîments

[Table des figures](#_TOC_250047)

[Liste des abréviations](#_TOC_250045)

1. Introduction

2. Présentation de la société

3. [Etude de besoin du projet](#_TOC_250041)

3.1 [Diagramme de cas d’utilisation global](#_TOC_250040)

3.2 [Diagramme MCD gl](#_TOC_250039)obal

[3.3 Architecture d’une application Android](#_TOC_250038)

[3.3.1 Le composant <Activity>](#_TOC_250037)

[3.3.2 Le composant <Fragment>](#_TOC_250037)

4. [Cycle de vie du projet](#_TOC_250044)

4.1 [Introduction des phases](#_TOC_250043)

4.1.1 Conception fonctionnelle

4.1.2 [Design de l’interface utilisateur](#_TOC_250042)

4.1.3 Conception de l’architecture

4.1.4 [Codage](#_TOC_250042)

4.1.5 Test

5. [Cycle de développement de l’application](#_TOC_250036)

5.1 Activité « Connexion »

5.1.1 Conception fonctionnelle

5.1.2 Design de l’interface utilisateur

5.1.3 Conception d’architecture

5.1.4 Codage

5.1.5 Test

5.2 Activité « Configuration »

5.2.1 Conception fonctionnelle

5.2.2 Design de l’interface utilisateur.

5.2.3 Conception d’architecture

5.2.4 Codage

5.2.5 Test

5.3 Activité « Files Transfer »

5.3.1 Conception fonctionnelle

5.3.2 Design de l’interface utilisateur

5.3.3 Conception d’architecture

5.3.4 Codage

5.3.5 Test

* 1. Activité « Administration »

5.4.1 Conception fonctionnelle

5.4.2 Design de l’interface utilisateur

5.4.3 Conception d’architecture

5.4.4 Codage

5.4.5 Test

6 [Validation](#_TOC_250036)

7 Conclusion

## Listes des figures

[Figure 1: Diagramme de cas d’utilisation global 10](#_Toc84610376)

[Figure 2: Diagramme MCD global 11](#_Toc84610377)

[Figure 3 Cycle de vie d’une activité 13](#_Toc84610378)

[Figure 4: Cycle de vie d’un fragment 14](#_Toc84610379)

[Figure 5: Modèle en spirale 15](#_Toc84610380)

[Figure 6 : Diagramme de cas d’utilisation de l’activité « Connexion » 21](#_Toc84610381)

[Figure 7:le diagramme d’activité pour le processus « Sign Up » 22](#_Toc84610382)

[Figure 8: le diagramme d’activité pour le processus Login 22](#_Toc84610383)

[Figure 9: Aperçu sur l’interface utilisateur de l’activité  « Connexion » 23](#_Toc84610384)

[Figure 10: : Le MCD de l’activité Connexion 25](#_Toc84610385)

[Figure 11: Le MPD généré pour l’activité « Connexion » 26](#_Toc84610386)

[Figure 12: le diagramme de package de l’activité "Connexion" 27](#_Toc84610387)

[Figure 13: Messages de validation des champs « User ID » et « Paswword » 29](file:///C:\Users\salim\Documents\+PFE\+Conception\rapport_PFE1.docx#_Toc84610388)

[Figure 14: Messages de confirmation des processus « Sign\_UP » et « Login » 30](#_Toc84610389)

[Figure 15: Le diagramme de cas d’utilisation de l’activité « Configuration » 31](#_Toc84610390)

[Figure 16: Design de l’interface utilisateur de l’activité « Configuration » 32](#_Toc84610391)

[Figure 17: Communications entre les sockets client et serveur 34](#_Toc84610392)

[Figure 18: le diagramme de Package de l’activité "Configuration" 34](#_Toc84610393)

[Figure 19: le diagramme de package pour l’activité "Configuration" 36](#_Toc84610394)

[Figure 20: Mécanisme de fonctionnement du serveur Http 37](#_Toc84610395)

[Figure 21: diagramme de cas d’utilisation de l’activité "FileTransfer" 43](#_Toc84610396)

[Figure 22: Aperçu de l’interface utilisateur de l’activité "FileTransfer" 44](#_Toc84610397)

[Figure 23: MCD pour l’activité "FileTransfer" 45](#_Toc84610398)

[Figure 24: Héritage de serveur http "NanoHTTPD" 46](#_Toc84610399)

[Figure 25: Personnalisation du serveur http 47](#_Toc84610400)

[Figure 26: le diagramme de package du processus "FileTransfer" 48](#_Toc84610401)

[Figure 27 : Mécanisme de téléchargement d’un fichier 49](#_Toc84610402)

[Figure 28:Mécanisme de téléversement d’un fichier 50](#_Toc84610403)

[Figure 29:diagramme de cas d’utilisation de l’activité "Administration" 57](#_Toc84610404)

[Figure 30:Le MCD de l’activité "Administration" 59](#_Toc84610405)

[Figure 31:le diagramme de package du processus "Administration" 60](#_Toc84610406)

## Listes des abréviations

**MCD** (Modèle Conceptuel de Données)

**MCD** (Modèle Physique de Données)

**UML: U**nified **M**odeling **L**anguage

**XML:** e**X**tensible **M**arkup **L**anguage

**MPD** (Modèle Physique de Données)

**SGBDR** (Système de Gestion Base de Données Relationelle)

**SDk** (softawre Development kit)

**IDE** (Integrate Development Editor)

**VCS** (version control system)

**GUI** (Graphical User Interface)

## 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 la le langage JavaScript et la technologie ReactNative.

Ce rapport se limitera à la réalisation du projet avec la 1er technologie et dont la période est programmée durant toute la période 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.

## Présentation de la société

## Etude de besoin du projet

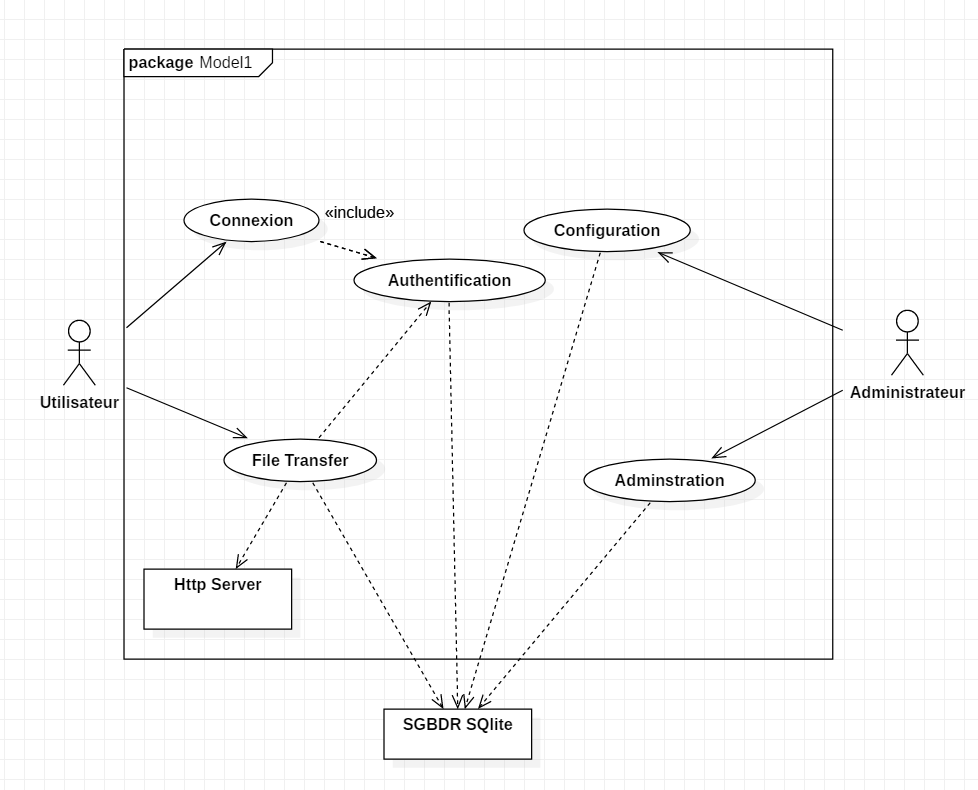
Lors de l’étude de besoin de ce projet on a déterminé ce qui suit :

1. Le diagramme de cas d’utilisation global

2. Le modèle conceptuel de données global (MCD).

3. La technologie à utiliser (Android SDK).

*3.1 Diagramme de cas d’utilisation global :*



**Figure 1**: Diagramme de cas d’utilisation global

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

**- « Connexion »**: permettant d’enregistrer et de connecter l’utilisateur.

**- « Configuration »** : permettant d’enregistrer et de mettre à jour les paramètres du serveur et de lancer et arrêter le serveur.

**- « File Transfer »** : permettant de se connecter vers un serveur distant et de télécharger/ téléverser des fichiers

**- « Adminstration »** : permettant d’analyser la liste des connexions des utilisateurs et la liste des transferts des fichiers.

*3.2 Diagramme MCD global :*

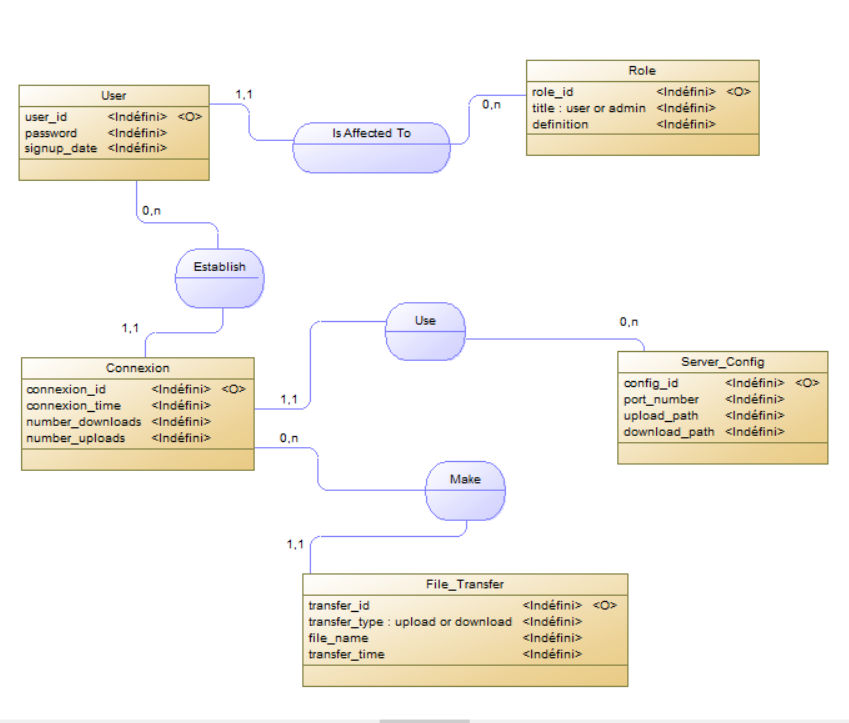


Figure 2: Diagramme MCD global

Dans le MCD On a les cinq tables suivantes :

* **Table « User »** : permettant d’enregistrer et d’authentifier les utilisateurs.
* **Table « Connexion »** : permettant d’enregistrer les connexions des utilisateurs.
* **Table « File\_Transfer »** : permettant d’enregistrer les transferts des fichiers de type "download" et "upload".
* **Table « Server\_Config »** : permettant de mettre à jour les paramètres de configuration du serveur qui sont : numéro du port, répertoire de téléchargement et répertoire de téléversement des fichiers.
* **Table « Rôle »** : permettant d’enregistrer les différents rôles à affecter aux utilisateurs qui sont "User" et "Admin".

*3.3 Architecture de l’application*

Une application Android comporte deux composants de types : « Activity » et « Fragment»

*3.1.1 Le composant « 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 évènements. Chaque événement est représenté dans une méthode.

****

Figure 3 Cycle de vie d’une activité

*3.1.2 Le composant « Fragment »*

Un Fragment est une partie d'une activité, qui contribue à sa propre interface utilisateur pour cette activité. Un Fragment peut être considéré comme une sous-activité, et 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 peut pas exister indépendamment. Il convient toujours de participer à une activité. Où que l'activité peut exister avec tout fragment en elle.

Le cycle de vie d’un fragment est plus complexe du cycle de vie d’une activité parce qu’il contient plus d’états.



Figure 4: Cycle de vie d’un fragment

## Cycle de vie du projet

Le « cycle de vie d'un logiciel » ou les méthodes de développement sélectionne et identifie toutes les étapes de développement de logiciel, dès sa conception en allant jusqu'à sa livraison. 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 respectent bien les contraintes prédéfinies auparavant.

*Cycle en spirale :*

Le modèle en spirale (*spiral model*) est un modèle de [cycle de développement logiciel](https://fr.wikipedia.org/wiki/Cycle_de_d%C3%A9veloppement_(logiciel)) qui reprend les différentes étapes du [cycle en V](https://fr.wikipedia.org/wiki/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 5: Modèle en spirale

On commence d’abord par l’étude des besoins fonctionnels :

L'**étude des 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.

D’après le diagramme de cas d’utilisation réalisé par l’étude de besoin on a identifié 4 activités principales pour notre application :

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

Pour chacune de ces activités on passe par les 5 phases suivantes :

1. Conception Fonctionnelle :
2. Design de l’interface utilisateur :
3. Conception de l’architecture
4. Codage
5. Test

*4.1 Introduction des phases*

*4.1.1 Conception fonctionelle*

Nous avons utilisé pour la partie conception fonctionnelle le langage UML (Unified Modeling Language): c’est un langage de modélisation unifié permettant de modéliser une application logicielle d'une façon standard dans le cadre de conception orienté objet.

On à utlisé en particulier les deux diagrammes « Cas d’utlisation » et diagramme « d’Activité ».

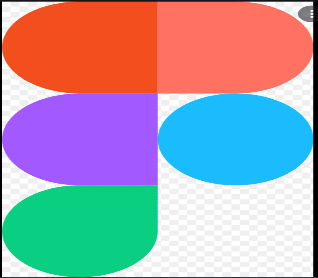
Le diagramme cas utlisation :

Les diagrammes de cas d'utilisation sont des diagrammes UML utilisés pour une représentation du comportement fonctionnel d'un système logiciel. Ils sont utiles pour des présentations auprès de la direction ou des acteurs d'un projet, mais pour le développement, les cas d'utilisation sont plus appropriés.

Le diagramme d’activités :

Le diagramme d'activité est un diagramme comportemental d'UML, permettant de représenter le déclenchement d'événements en fonction des états du système et de modéliser des comportements parallélisables. Le diagramme d'activité est également utilisé pour décrire un flux de travail.

* + 1. *Design de l’interface utilisateur*

Nous avons utilisé pour la partie design la plateforme web « 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.

*4.1.3 Conception de l’architecture*

Pour la modélisation des données on a utilisé la méthodologie merise permettant de réaliser le modèle conceptuel des données et le modèle physique des données.

**Le** **MCD (Modèle Conceptuel des Données)** : est une représentation graphique de haut niveau qui permet facilement et simplement de comprendre comment les différents éléments sont liés entre eux à l’aide de diagrammes codifiés.

**Le MPD** (**Modèle Physique des Données) :** L’étape de création du MPD est presque une formalité comparée à la création du MCD. En s’appuyant sur des règles simples (et qui fonctionnent à tous les coups), l’analyste fait évoluer sa modélisation de haut niveau pour la transformer en un schéma plus proche des contraintes des logiciels de bases de données. Il s’agit de préparer l’implémentation dans un [SGBDR](https://www.base-de-donnees.com/sgbd/).

Nous avons utilisé le logiciel Power AMC pour modéliser le diagramme ER (Entité-Relation) permettant de générer le MPD de la base de données, c.-à-d. sa structure finale physique.

La méthode 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.

*4.1.4 Codage*

Dans cette phase nous allons présenter l'environnement matériel et logiciel utilisé pour le développement de la solution proposée 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.

Les Technologies/Outils utilisés pour le codage de 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.

**Android Studio 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.**

*4.1.5 Test*

Au cours de cette phase en rédige le cahier de test dans lequel on valide les différentes fonctionnalités de chaque activité.

La validation des fonctionnalités est également accompagnée par des imprimes écran confirmant le bon fonctionnement des activités

## Cycle de développement de l’application

Notre application comporte les 4 activités suivantes :

- Activité « Connexion »

- Activité « Configuration »

- Activité « Files Transfer »

- Activité « Administration »

Le développement de chaque activité sera réalisé lors d’une itération dédiée dans le cycle en spirale du projet.

*5.1 Activité « Connexion »*

Dans cette activité on réalise l’enregistrement et l’authentification des utilisateurs de l’application.

*5.1.1 Conception fonctionnelle*

**Le Diagramme de cas d’utilisation pour l’activité « Connexion » :**

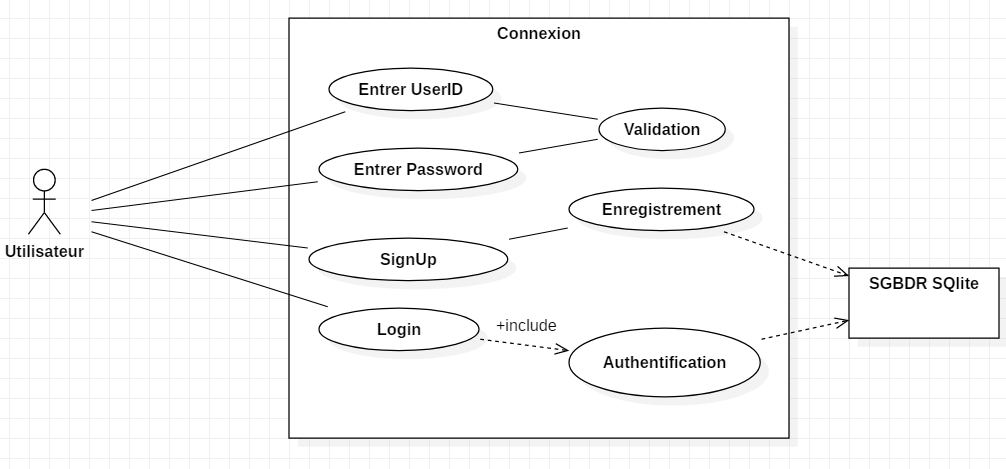


Figure 6 : Diagramme de cas d’utilisation de l’activité « Connexion »

**Le diagramme d’Activité du processus « Sign Up » :**

Au bout de ce processus l’utilisateur va créer un compte utilisateur.

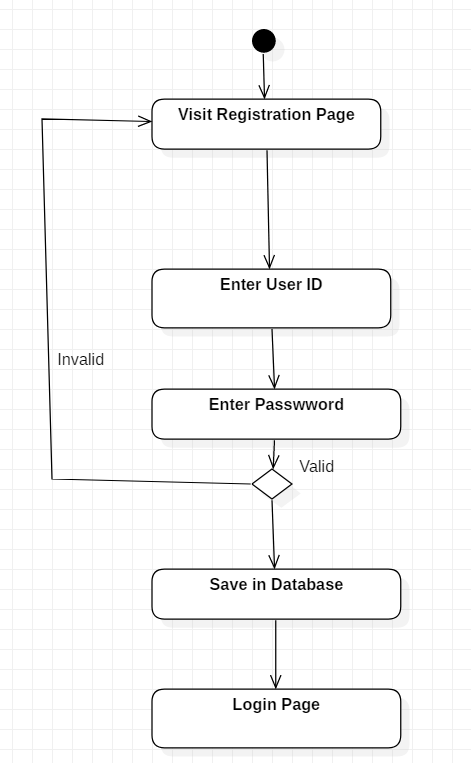


Figure 7:le diagramme d’activité pour le processus « Sign Up »

**Le diagramme d’activité pour le processus « Login » :**

Dans ce processus l’utilisateur saisie le "User ID" et le "Password" pour s’identifier à l’application.



Figure 8: le diagramme d’activité pour le processus Login

*5.1.2 Design de l’interface utlisateur*

Pour pouvoir utiliser l’application il faut créer un compte utilisateur nécessitant la validation de deux champs “User ID" et "Password" et dont le contrôle est réalisé en respectant plusieurs conditions de validation .

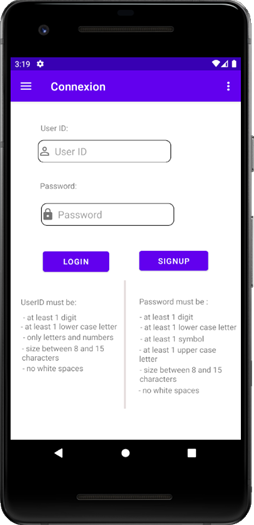


Figure 9: Aperçu sur l’interface utilisateur de l’activité  « Connexion »

Pour la validation du champ "User Id" il faut respecter les conditions suivantes :

- Au moins 1 chiffre

- Au moins une lettre minuscule

- Seulement des lettres minuscules et des chiffres

- Sans espaces.

- Taille entre 8 et 15 caractères.

Pour la validation du champ "Password" il faut respecter les conditions suivantes :

- Au moins 1 chiffre

- Au moins une lettre minuscule

- Au moins une lettre majuscule

- Au moins un symbole

- Sans espaces.

- Taille entre 8 et 15 caractères.

*5.1.3 Conception d’architecture*

*Dans cette partie on réalise le modèle conceptuel de donnée liées à l’activité en cours de développement.*

*En particulier on commence par créer les tables suivantes "User" et "Connexion" :*

**Table User** :

Elle permet d’enregistrer les utilisateurs et de les authentifier.

Elle contient les attributs suivants :

"user\_id" : identifiant de l’utilisateur

"password" : Mot de passe de l’utilisateur

"signup\_date" : Date d’enregistrement de l’utilisateur

**Table Connexion :**

Elle permet d’enregistrer les connexions des utilisateurs dans la base de donnée.

Cette table contient les attributs suivants :

"connexion\_id" : Identifiant de la connexion de l’utilisateur

"connexion\_time": Date et heure à laquelle la connexion a été effectuée

"number\_downloads": Nombre de téléchargement de fichiers réalisés par la connexion

"number\_uploads": Nombre de téléversements de fichiers réalisés par la connexion

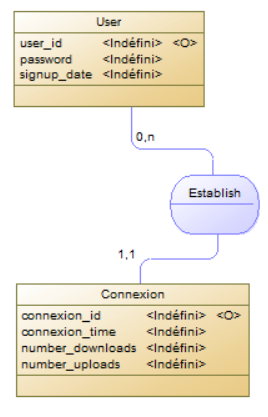


Figure 10: : Le MCD de l’activité Connexion

## L’association « Establish »

## Un utilisateur (« User ») peut établir plusieurs connexions (de 0 jusqu’à N)

## Une connexion est établie par un et un seul utilisateur.

**Le diagramme MPD (Modèle Physique de Données) :**

## Le modèle physique des données consiste à définir la structure finale de la base de données dans le SGBDR, dans lequel les clés étrangères seront générées

## La structure de la base de données générée est la suivante :

**Table User :**

Elle contient les attributs : « user-id », « role\_id », « password » et « signup\_date ».

**Table Connexion :**

Elle contient les attributs suivants :

« connexion\_id », « connexion\_time », « number\_downloads », « number\_uploads » et « user\_id ».

« user\_id » : c’est une clé étrangère qui va pointer vers la clé primaire de la table « User».

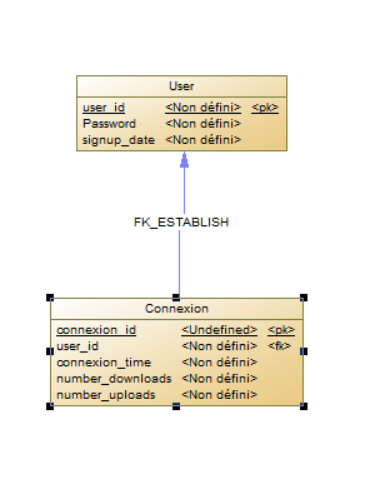


Figure 11: Le MPD généré pour l’activité « Connexion »

*5.1.4 Codage*

**Le diagramme de package :**

**Package MyApplication**

- MainActivity

**Package DB**

-DatabaseHelper

-User

-Connexion

**Package UI**

-ConnexionFragment

-ConnexionViewModel

Figure 12: le diagramme de package de l’activité "Connexion"

- MainActivity : c’est la classe de démarrage de l’application.

- **Le Pacakage DB** : contient les classes suivantes :

- DatabaseHelper : Elle permet la création et la gestion de la base de données à travers des requêtes SQL.

- User : Elle permet de représenter une ligne d’enregistrement dans la table « User » et contient les méthodes « Getters » et « Setters ».

- Connexion : Elle permet de représenter une ligne d’enregistrement dans la table « Connexion » et elle contient les méthodes « Getters » et « Setters ».

- **Le Pacakage Ui** : contient les classes suivantes :

- ConnexionFragment : Elle permet de définir les détecteurs des évènements des clicks sur les boutons de l’interface du fragment.

- ConnexionViewModel : Elle permet de stocker le modèle de données liée à l’interface du fragment.

Les conditions de validations des champs « User ID » et « Password » sont implémenté à travers les expressions régulières suivantes :

Pour le champ User ID :

|  |  |
| --- | --- |
| Conditions de Validation | Expression régulière |
| Taille entre 8 et 15 caractères | **^.{8,15}$** |
| au moins un chiffre | **^.\*[0-9]+.\*$** |
| au moins une lettre minuscule | **^.\*[a-z]+.\*$** |
| seulement des lettres et des chiffres | **^[a-z0-9]+$** |
| pas d’espace | **^\\S+$** |

Pour le champ Password :

|  |  |
| --- | --- |
| Conditions de Validation | Expression régulière |
| Taille entre 8 et 15 caractères | **^.{8,15}$** |
| au moins un chiffre | **^.\*[0-9]+.\*$** |
| au moins une lettre minuscule | **^.\*[a-z]+.\*$** |
| seulement des lettres et des chiffres | **^[a-z0-9]+$** |
| pas d’espace | **^\\S+$** |
| au moins une lettre majuscule | **^.\*[A-Z]+.\*$** |
| n’importe quelle lettre | **^.\*[@#$%^&+=]+.\*$** |

*5.1.5 Test*

Pour la phase de test on valide les fonctionnalités de l’activité avec les imprimes écrans suivante :

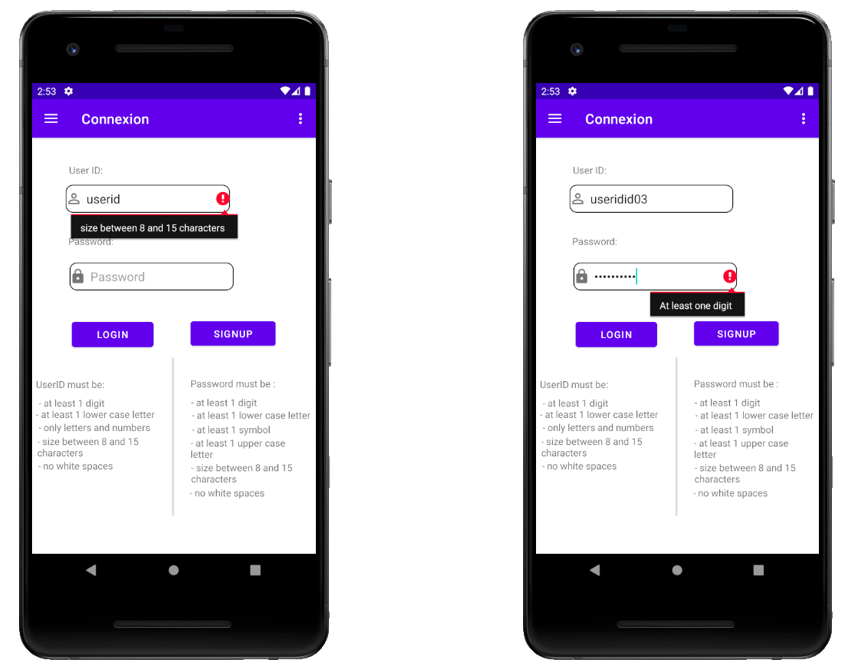
**

Figure 13: Messages de validation des champs « User ID » et « Paswword »

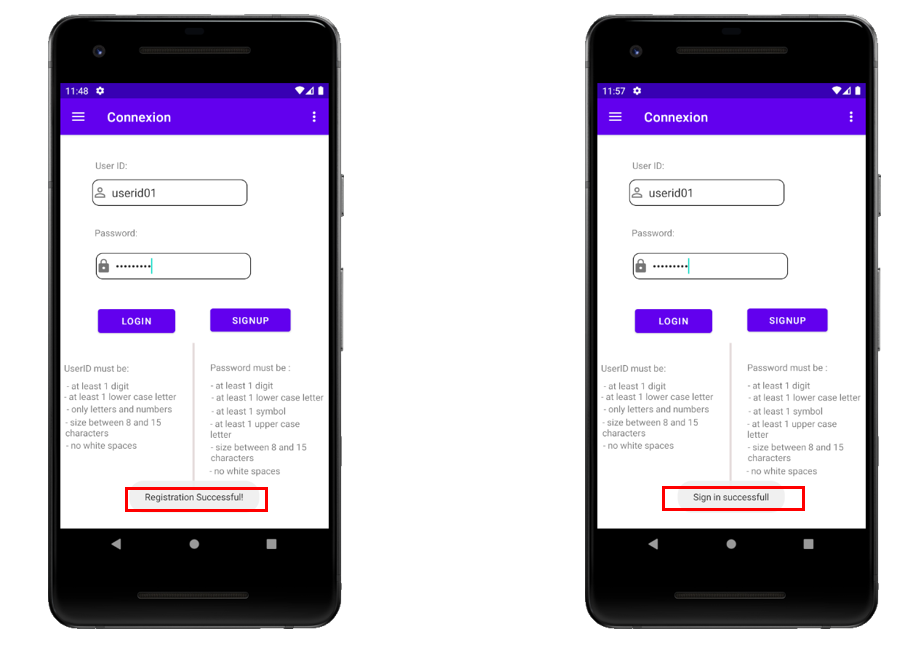
**

Figure 14: Messages de confirmation des processus « Sign\_UP » et « Login »

*5.2 Activité Configuration*

Dans cette activité on enregistre et on modifie les paramètres de configuration du serveur dans la base de données, on démarre et on arrête le serveur, on affiche l’adresse IP locale du serveur et le statut de fonctionnement du serveur (« Ruuning » or « Stopped »).

*5.2.1 Conception Fonctionnelle*

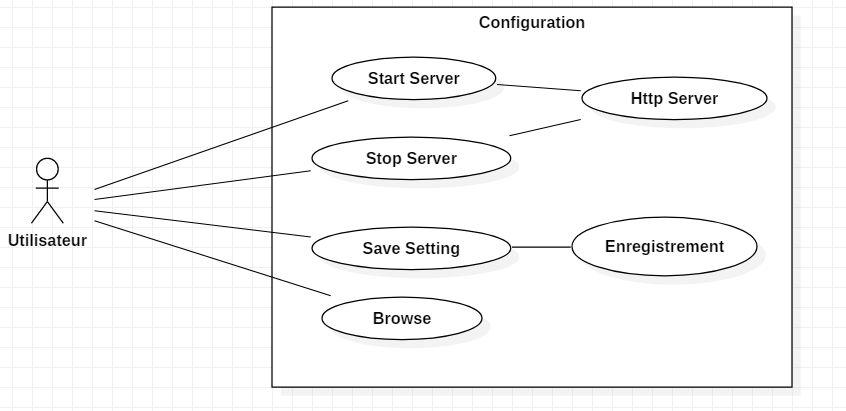
**Le diagramme de cas d’utilisation fonctionnelle**

Figure 15: Le diagramme de cas d’utilisation de l’activité « Configuration »

*5.2.2 Design de l’interface utilisateur*

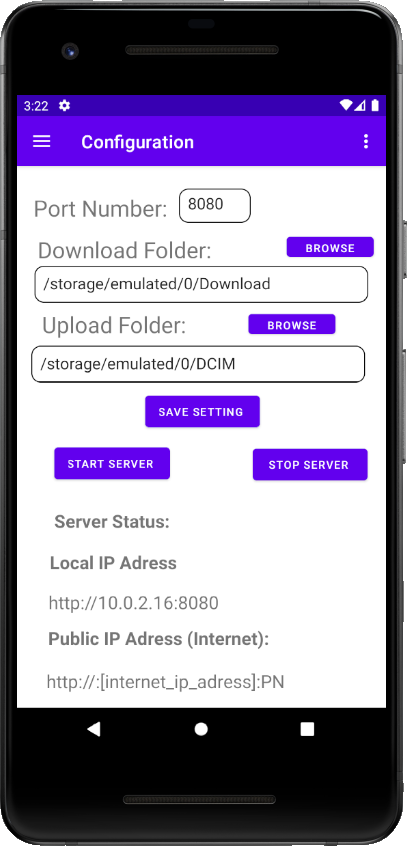
****

Figure 16: Design de l’interface utilisateur de l’activité « Configuration »

Le bouton « Save Settings » permet d’enregistrer et de modifier les paramètres de configuration du serveur dans la base de données et qui sont :

* le numéro de port « Port Number »,
* le répertoire de téléchargement « Download Folder »
* le répertoire de téléversement « Upload Folder »

Pour démarrer le serveur, il faut appuyer sur le bouton « Start Server » et pour l’arrêter il faut appuyer sur le bouton « Stop Server ».

Le « Server Status » permet d’afficher le statut de fonctionnement du serveur.

Le « Local Ip address » permet d’afficher l’adresse du serveur avec son adresse IP sur le réseau local.

Le « Public Ip Address » permet d’afficher l’adresse du serveur avec son adresse IP externe sur le réseau publique (Internet)

*5.2.3 Conception d’architecture*

**Définition des protocoles de télécommunications :**

**Le Protocole TCP/IP :** (Transmission Control Protocol/[Internet Protocol](https://www.journaldunet.fr/web-tech/dictionnaire-du-webmastering/1203389-ip-adresse-ip-internet-protocol-definition/)) 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).

**Le 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.

**Le Protocol FTP** : Un [serveur](https://www.journaldunet.fr/web-tech/dictionnaire-du-webmastering/1203337-serveur-informatique-definition-traduction/) FTP (File Transfer Protocol) est un logiciel utilisé dans le transfert de fichiers entre deux ordinateurs. Il est, avec le client FTP, l'une des deux composantes d'un transfert de fichiers via le langage FTP.

**Différence entre les protocoles HTTP et FTP :**

|  |  |
| --- | --- |
| **HTTP** | **FTP** |
| Il ne prend en charge que la connexion de données. | Il prend en charge à la connexion de données et la connexion de contrôle |
| Il utilise le protocole de contrôle de transmission sur le port Tcp 80. | Il utilise le protocole de contrôle de transmission sur le port Tcp 20 et 21. |
| L’URL utilisant le protocole http commencera par HTTP | L’URL utilisant le FTP commencera par FTP |

**Les Sockets**: Un socket est connu 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.

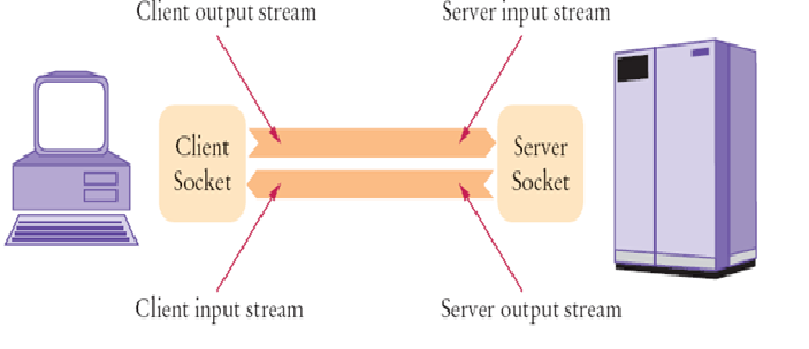


Figure 17: Communications entre les sockets client et serveur

**Le diagramme de Package :**

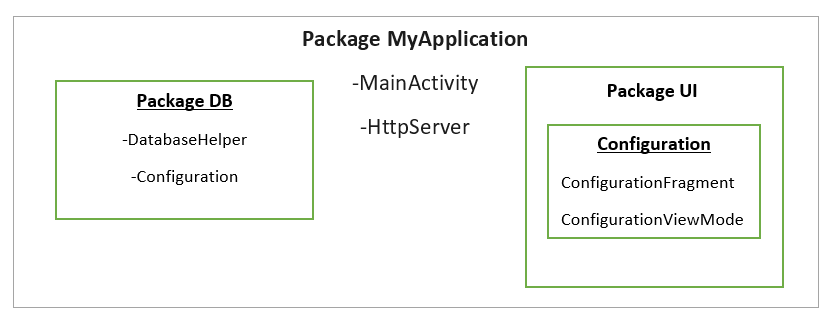


Figure 18: le diagramme de Package de l’activité "Configuration"

- « MainActivity » : c’est la classe de démarrage de l’application.

- **Le Pacakage DB** : il contient les classes suivantes :

- « DatabaseHelper » : elle permet la création et la modification de la base de données avec des

requêtes SQL

- « Configuration » : elle permet de représenter une ligne d’enregistrement dans la table

« Configuration » tout en proposant les méthodes « Getters » et « Setters ».

- **Le Pacakage Ui** : contient les classes suivantes :

- « ConfigurationFragment » : Elle permet de définir les détecteurs des évènements des clicks sur les boutons de l’interface du fragment.

- « ConfigurationViewModel » : elle permet de stocker le modèle de données lié à l’interface du fragment.

*5.2.4 Codage*

Pour le développement de cette activité on a fait appel au deux packages suivants :

**- "Java.IO"** : on a en particulier utilisé les classes "InputStream" et "OutputStream" pour gérer les transferts des données à travers les sockets.

**- "Java.NET"** : on a en particulier utilisé les classes "ServerSocket" et "Socket" permettant de gérer les instances des sockets et leur serveur.

**- "Android SDK"** : on a en particulier utilisé les classes permettant de gérer les composants "Activity" et "Fragment" et la configuration du fichier "AndroidManifest.xml".

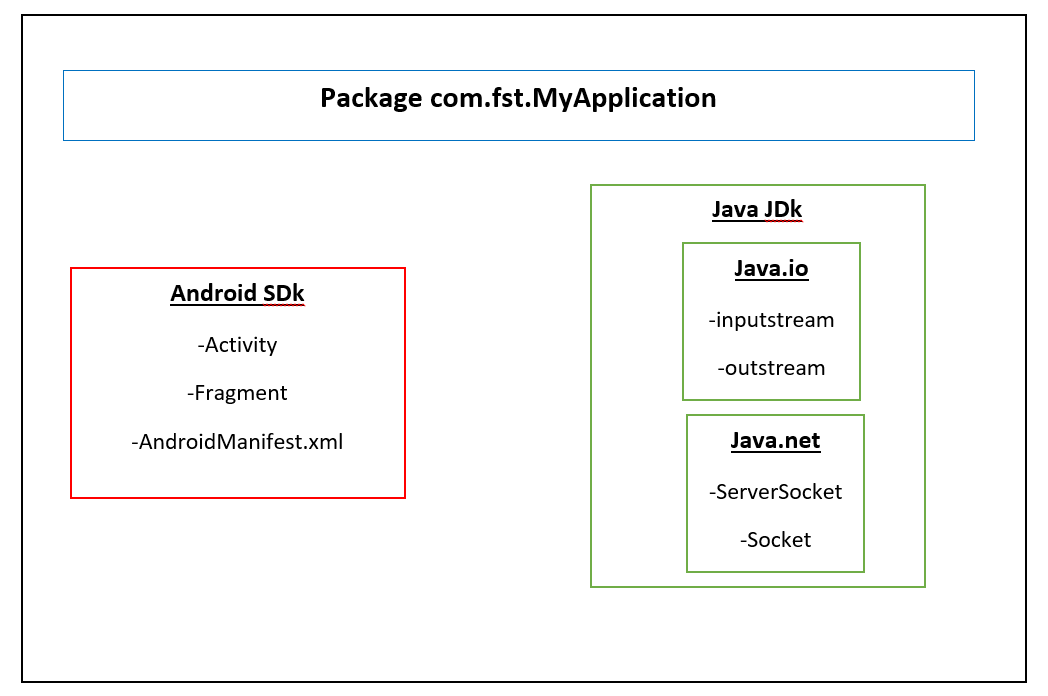
**

Figure 19: le diagramme de package pour l’activité "Configuration"

**Les permissions d’utilisateurs d’Android :**

Chaque application Android s'exécute dans un bac à sable à accès limité. Si votre application doit utiliser des ressources ou des informations en dehors de son propre bac à sable, vous pouvez déclarer une autorisation et configurer une demande d'autorisation qui fournit cet accès.

Dans notre application nous avons utilisé les permissions suivantes :

* "INTERNET" :

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

* "ACCESS\_NETWORK\_STATE":  
  <**uses-permission android:name ="android.permission.ACCESS\_NETWORK\_STATE"** />
* " ACCESS\_WIFI\_STATE":  
  <**uses-permission android:name="android.permission.ACCESS\_WIFI\_STATE"**/>

Ces permissions sont configurables au niveau du fichier "AndroidManifest.xml"

**Implémentation du serveur HTTP :**

Notre serveur http est implémenté avec deux types de processus d’arrière-plan :

- "ServerListenerThread" : c’est la classe de type Thread permettant de lancer la classe "ServerSocket" sur un numéro spécifique en arrière-plan tout en acceptant les nouvelles connexions vers le serveur.

- "HttpConnectionThread" : c’est la deuxième classe de type Thread permettant de gérer les communications de chaque socket dans un processus en arrière-plan à part.

**Mécanisme de fonctionnement du serveur HTTP :**

On crée une seule instance de type "ServerlistenerThread" pour accepter les connexions sur le numéro de port spécifié.

Pour chaque nouvelle connexion on crée une nouvelle instance de type "HttpConnectionThread" permettant de gérer la socket créée en association avec la connexion.

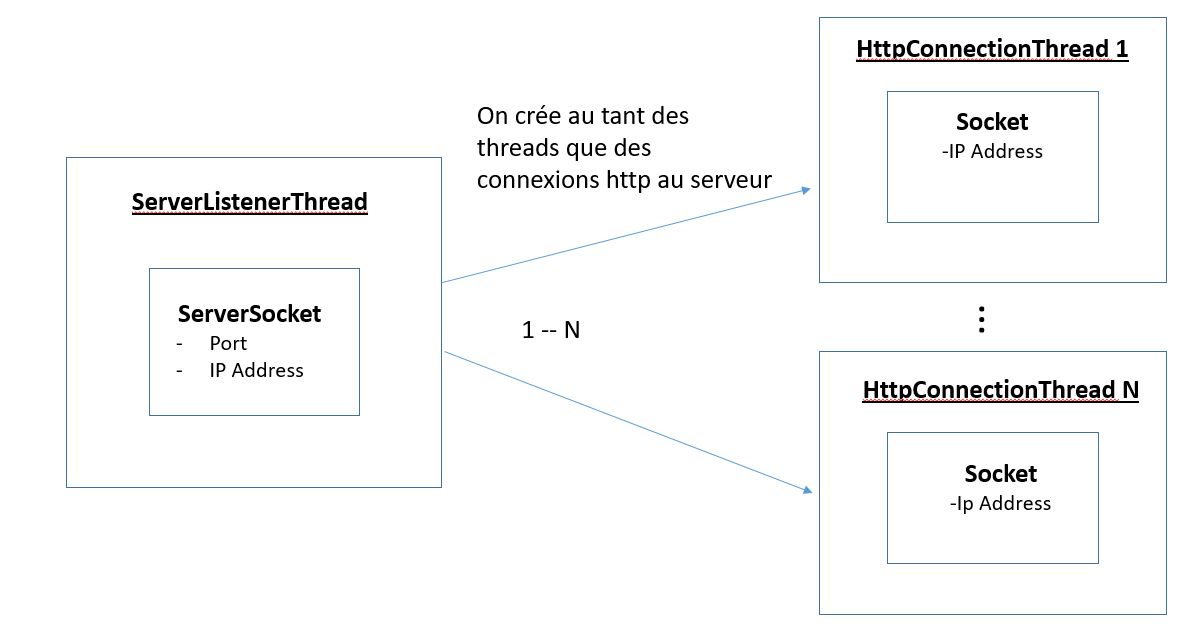


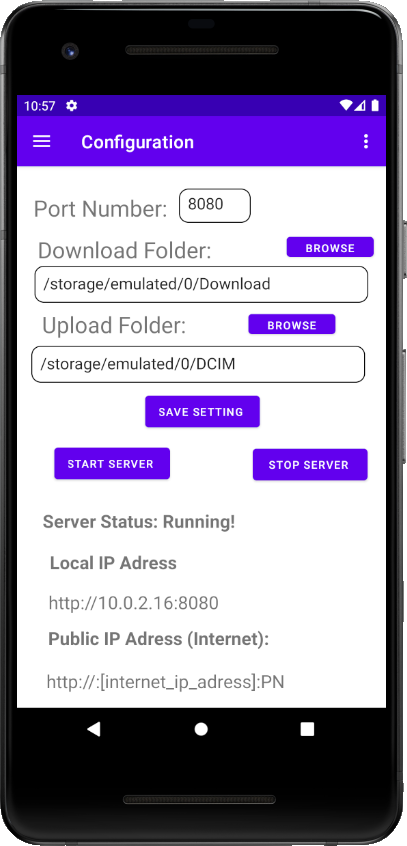
Figure 20: Mécanisme de fonctionnement du serveur Http

*5.2.5 Test*

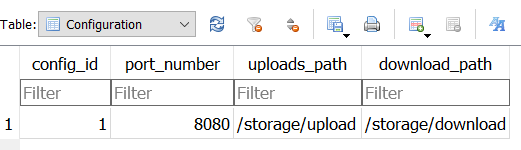
**Le bouton "Save Settings” :**

Il permet d’ajouter une nouvelle ligne dans la table "Configuration" ou bien mettre à jour une ligne existante dans la base de données.

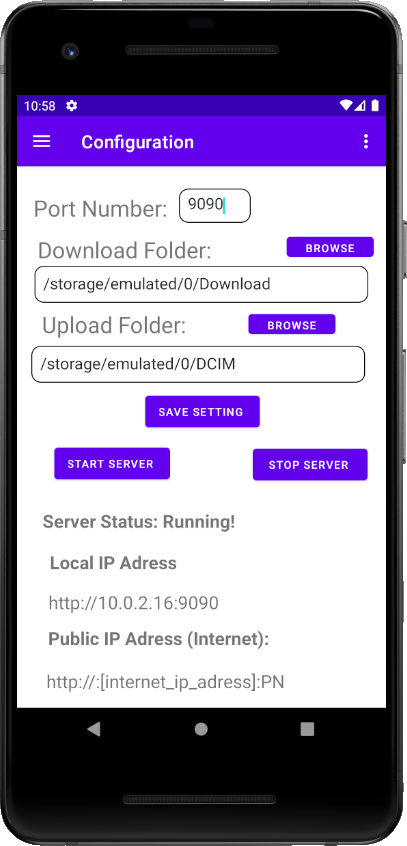
Test de l’enregistrement d’une nouvelle ligne dans la table "Configuration" avec le numéro de port : 8080.

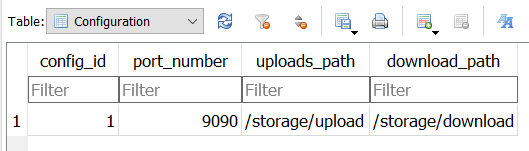


Vérification de l’ajout dans la base de données avec l’outil DB Sqlite Browser.



Mise à jour d’une ligne déjà existante dans la table "Configuration" avec le nouveau numéro de port : 9090.





**Le bouton "Start Server" :**

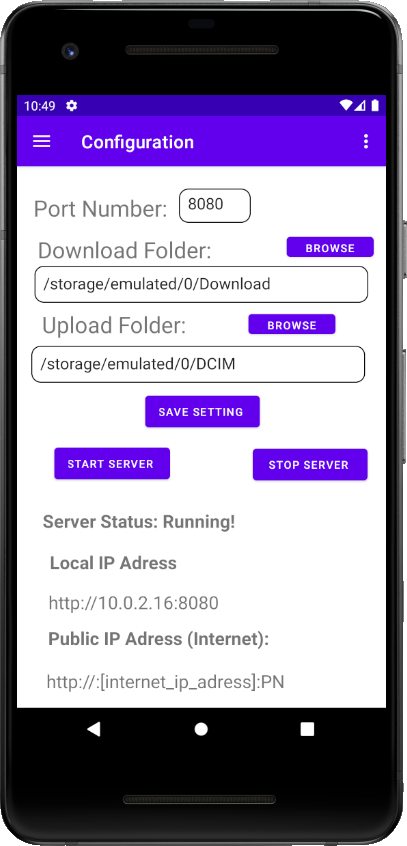
Il permet de démarrer le serveur sur le numéro du port spécifié.

On commence par tester le serveur avec le navigateur web "Chrome" utilisant l’adresse suivante : "http://localhost:8080"

**

D’après l’imprime écran ci-dessus on voit bien que le serveur http répond au navigateur et revoie un message d’accueil et une confirmation du bon fonctionnement du serveur.

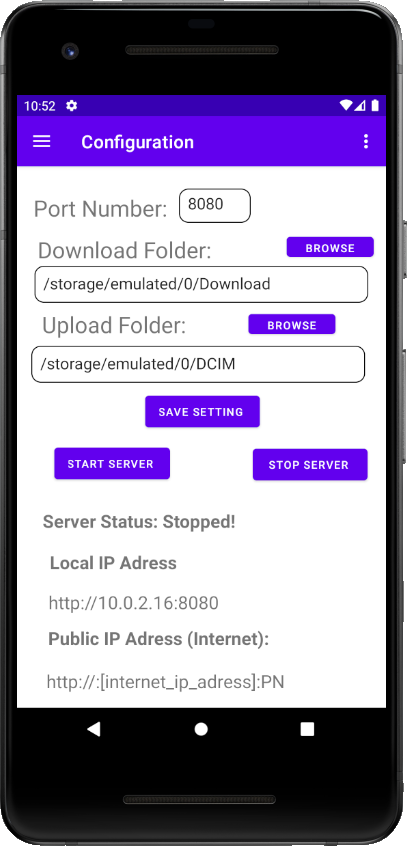
**Affichage du statut** : "Server Status Running !"

****

Dans l’imprime écran ci-dessus on indique à l’utilisateur les informations suivantes :

* Le statut de fonctionnement de serveur : "Running !"
* L’adresse IP sur le réseau local : "http://10.0.2.16:8080"

**Affichage du statut**: "Server Status Stopped !"



Lorsqu’on clique sur le bouton "Stop Server" on voit bien que le statut de fonctionnement du serveur bascule vers l’état "Stopped !"

* 1. *Activité File Transfer*

Cette activité nous permet de se connecter un serveur HTTP distant et d’effectuer des téléchargements et des téléversements des fichiers avec le serveur.

* + 1. *Conception Fonctionnelle*

**Le diagramme de cas d’utilisation :**

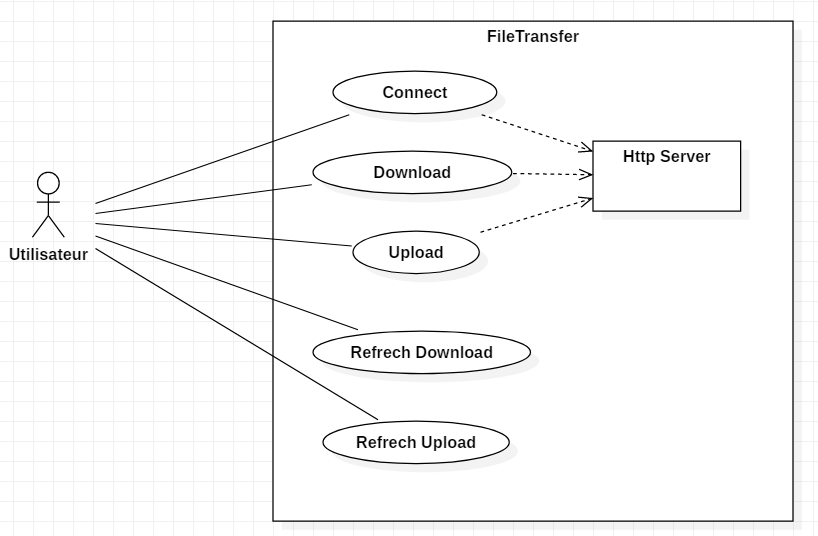
**

Figure 21: Le diagramme de cas d’utilisation de l’activité "FileTransfer"

*5.3.2 Design de l’interface utilisateur*

A travers l’interface utilisateur on peut effectuer les opérations suivantes :

* **"Connect" :** permet de connecter au serveur HTTP et d’afficher son statut de fonctionnement (Running ou Stopped !)
* **"Download" :** permet de télécharger le fichier sélectionné sous le répertoire "Upload Folder" du serveur vers le répertoire local "Download Folder".
* **"Upload" :** permet de téléverser le fichier sélectionné sous le répertoire local "Download Folder" vers le répertoire distant "Upload Folder" du serveur.

****

Figure 22: Aperçu de l’interface utilisateur de l’activité "FileTransfer"

* + 1. *Conception d’architecture*

Pour notre modèle conceptuel de donnée on a ajouté la table "File\_Transfer" permettant d’enregistrer les transferts des fichiers réalisés par chaque connexion et qui contient les attributs suivant :

* "transfer\_id" : identifiant de la table
* "transfer\_type" : c’est le type de transfert, ayant pour valeur "upload" ou bien "download"
* "file\_name" : c’est le nom du fichier
* "transfer\_time" : c’est la date à laquelle le transfert a été effectué .

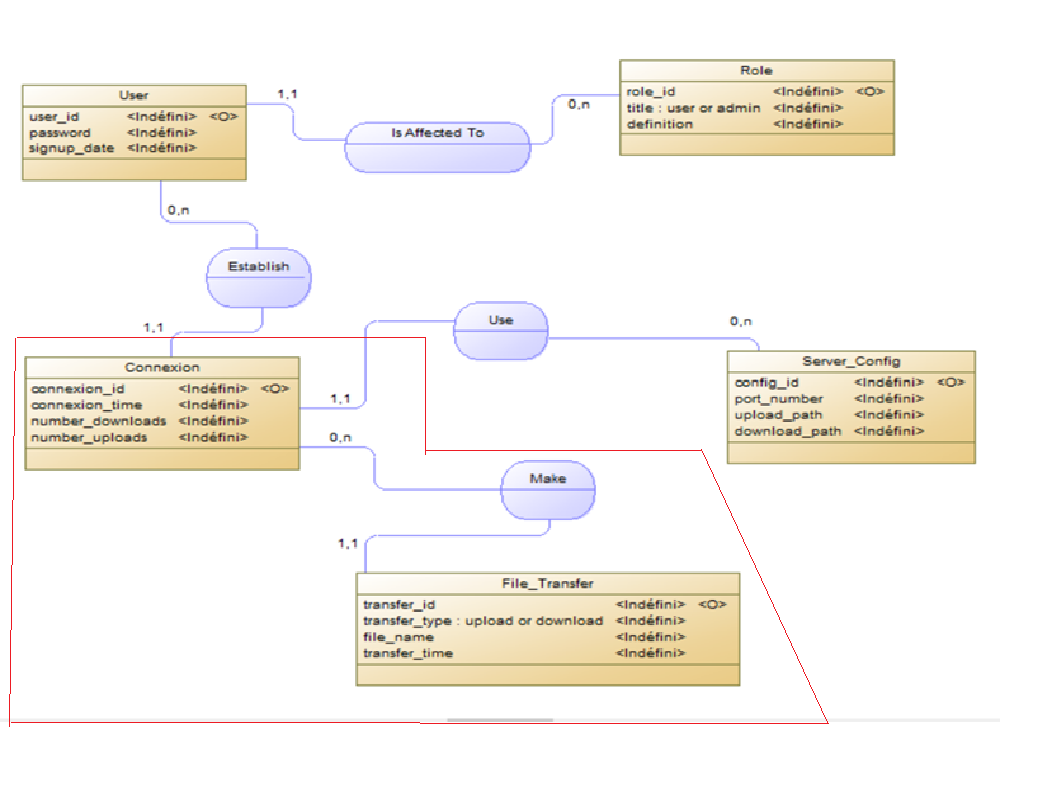
**

Figure 23: Le MCD pour l’activité "FileTransfer"

## L’association « Make »

## Une Connexion peut faire plusieurs Transferts des fichiers (de 0 jusqu’à N)

## Un Transfert de fichier est fait par une et une seul Connexion.

**Héritage du serveur http "NanoHTTPD"**

Pour ce projet on a importé un serveur http déjà implémenté distribué librement (en open source) sur GitHub sous le nom de "NanoHTTPD" et dont le lien se trouve sur l’adresse suivante : "https://github.com/NanoHttpd/ ".

Pour notre application notre serveur http "MyHttpServer" va hériter l’implémentation du serveur "NanoHTTPD".

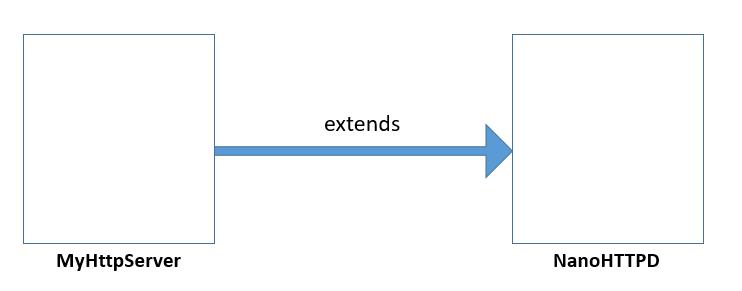
****

Figure 24: Héritage du serveur http "NanoHTTPD"

**Personnalisation du serveur HTTP**

La personnalisation du serveur consiste à implémenter le code permettant de renvoyer deux types de réponses au navigateur :

* **1er réponse en format HTML** : ce type de réponse est renvoyé quand la requête http ne contient pas de paramètres et constitue comme données la liste des fichiers se trouvant sous la racine du serveur sous forme de liens hypertexte (téléchargeables).
* **2ème réponse en format JSON** : ce type de réponse est renvoyé quand la requête http contient le paramètre "request=upload" et constitue comme données la liste des fichiers sous le répertoire "Upload Folder" du serveur.

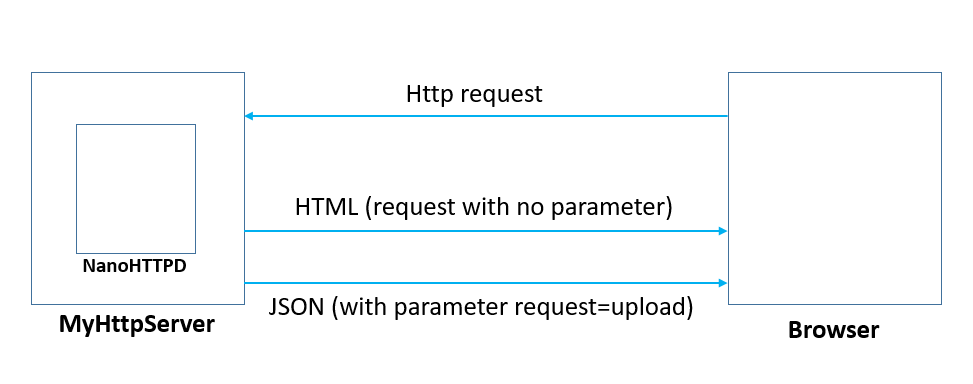


Figure 25: Personnalisation du serveur HTTP

*5.3.4 Codage*

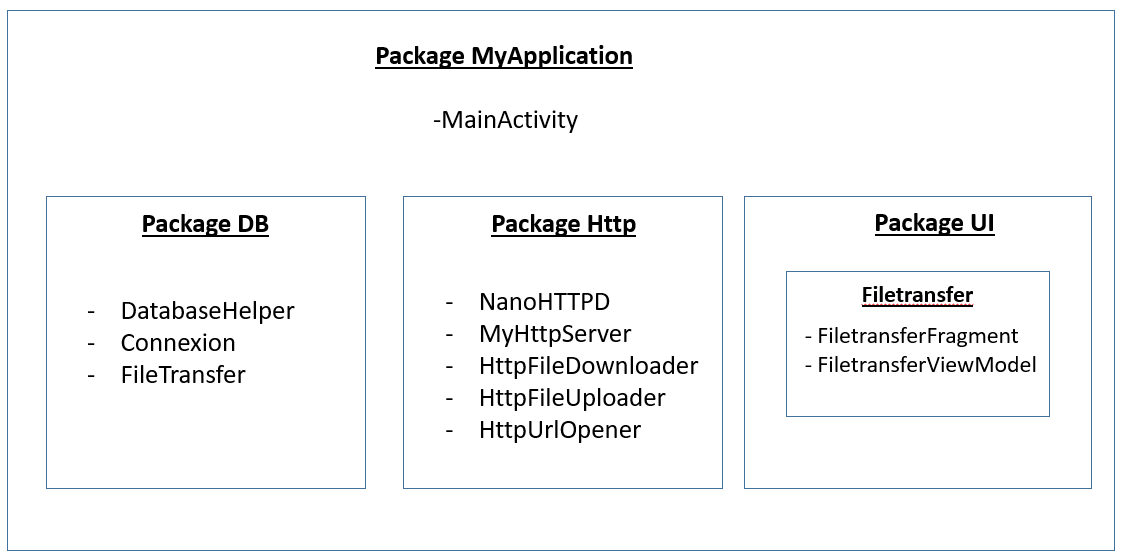
**Le diagramme de package :**

Figure 26: le diagramme de package du processus "FileTransfer"

- MainActivity : c’est la classe de démarrage de l’application.

- **Le Pacakage Http** : il contient les classes liées au serveur Http

- NanoHTTPD : c’est une implémentation open source d’un serveur Http.

- MyHttpServer : elle hérite de la classe NanoHTTPD et permettant le lancement et l’arrêt du serveur.

- HttpFileDownloader : elle permet de télécharger un fichier

- HttpFileUploader : elle permet de téléverser un fichier

- HttpUrlOpener : elle permet de se connecter au serveur et de vé0rifier son fonctionnement.

- **Le Pacakage Ui** : il contient les classes en rapport avec l’interface utilisateur :

- FiletranferFragment : elle permet de contenir les composants de l’interface utilisateur et de définir les détecteurs des évènements des clicks réalisés sur les boutons.

- FiletransferViewModel : elle permet de stocker le modèle de données liée à l’interface du

Fragment.

**Mécanisme de téléchargement d’un fichier :**

Pour réaliser l’opération de téléchargement d’un fichier on crée une instance de la classe "HttpUrlConnection" qui va nous permettre de gérer la connexion sous le protocole http avec le serveur.

Pour chaque nouveau téléchargement on crée un Thread pour réaliser le téléchargement des données en arrière-plan à travers la classe "HttpUrlConnection".

L’opération de téléchargement est effectuée en deux étapes :

* **1ere étape :** on crée une instance de la classe "HttpFileDownloader" permettant de d’envoyer une requête de connexion au serveur avec la classe "HttpUrlConnection".
* **2ème étape :** on lance le téléchargement des données avec la classe "InputStream" puis l’écriture dans un fichier avec la classe "FileOutputStream".

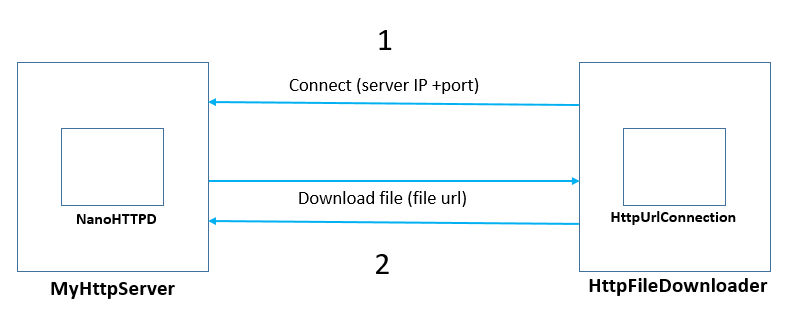
**

Figure 27 : Mécanisme de téléchargement d’un fichier

**Mécanisme de téléversement d’un fichier (Upload) :**

Pour réaliser l’opération de téléversement d’un fichier on envoie une demande de téléversement du fichier au serveur, ce dernier va créer une instance de la classe "HttpFileUploader" permettant le téléchargement du fichier et l’envoie de sa progression.

L’opération de téléversement est effectuée en trois étapes :

* **1ere étape :** envoie d’une demande de téléversement par le serveur 1 vers le serveur 2.
* **2ème étape :** instanciation de la classe "HttpFileUploader" par le serveur 2
* **3ème étape :** lancement de téléchargement par le serveur 2 et envoie la progression vers le serveur 1**.**

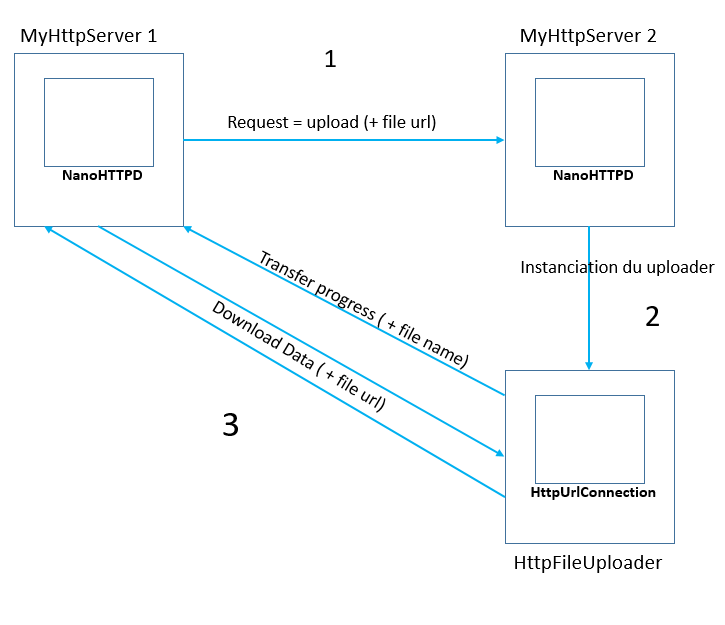
**

Figure 28: Mécanisme de téléversement d’un fichier

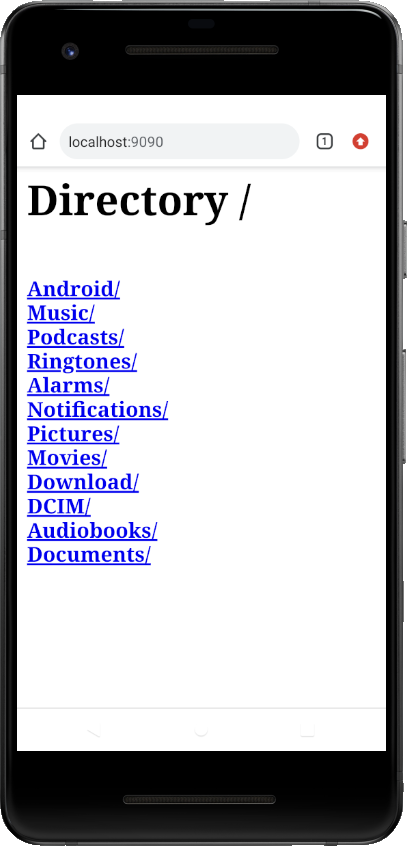
*5.3.5 Test*

L’interface utilisateur de l’activité "FileTransfer" permet les actions suivantes :

* **Le bouton "Connect"** : l’action sur ce bouton permet la connexion au serveur et la vérification de son fonctionnement

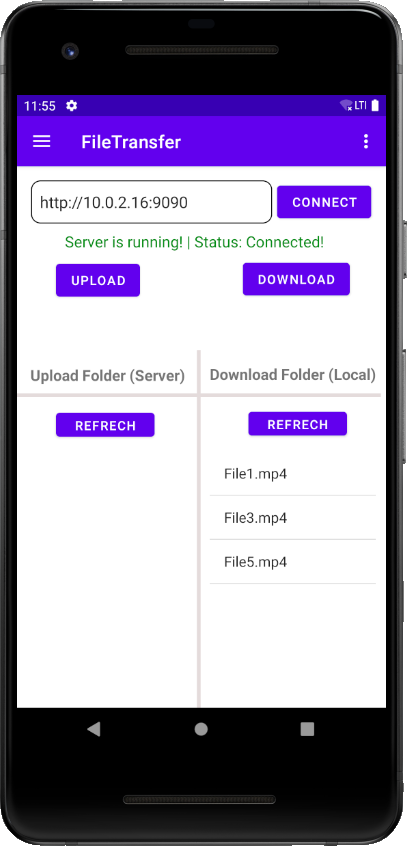
**

Sur l’imprime écran ci-dessus on voit bien le message de vérification du bon fonctionnement du serveur ("Server is Running ! | Status : Connected !).

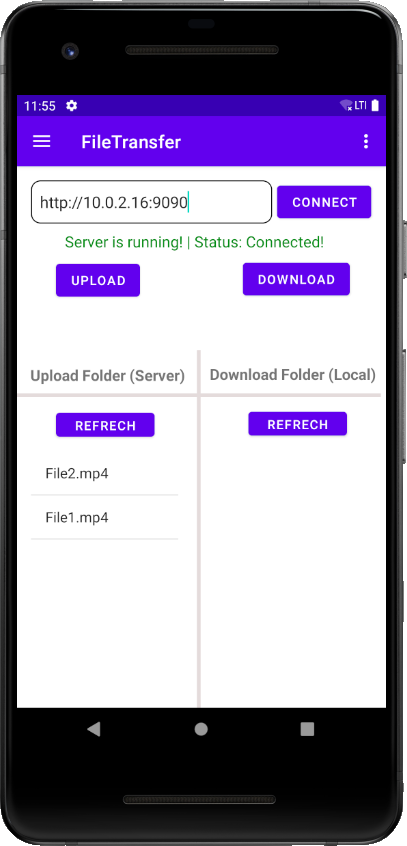


Dans l’imprime écran ci-dessus on voit bien que le serveur http (NanoHTTPD) répond au navigateur et revoie la liste des répertoires sous la racine du serveur " / " et qui correspond au répertoire "/storage/emulated/0".

* **Le bouton "Refrech" sous "Download Folder"** : l’action sur ce bouton permet la génération de la liste des fichiers sous le répertoire "Download Folder" sur l’appareil local.

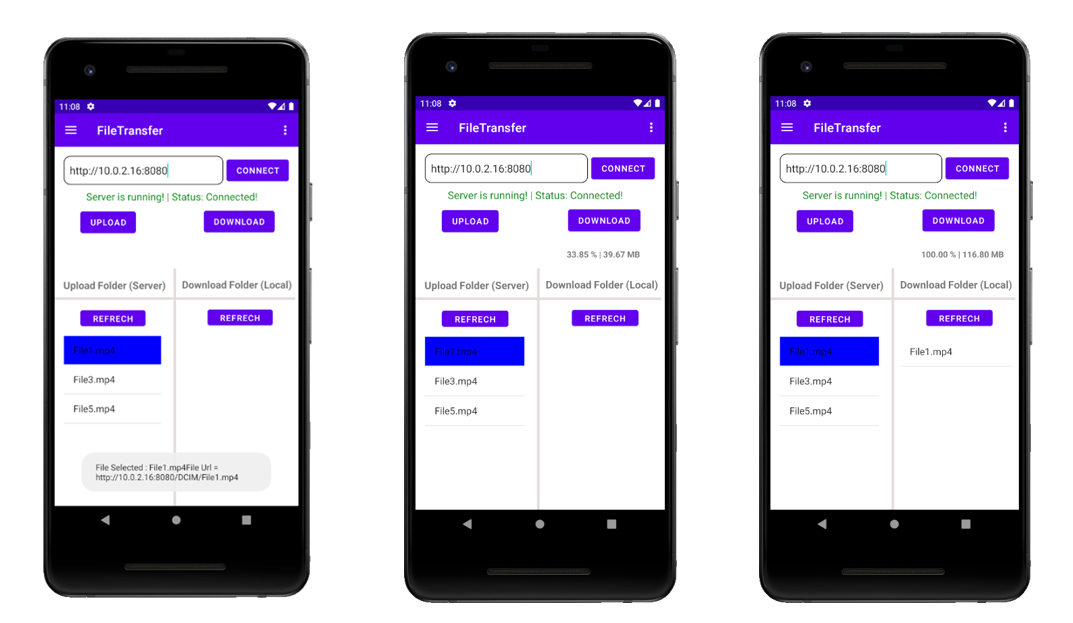
**

* **Le bouton "Refrech" sous "Upload Folder"** : l’action sur ce bouton permet d’afficher la liste des fichiers sous le répertoire "Upload Folder" sur l’appareil distant du serveur.

**

* **Le bouton "Download"** : l’action sur ce bouton permet de lancer le téléchargement d’un fichier depuis le répertoire distant "Upload Folder" du serveur vers le répertoire local "Download Folder".

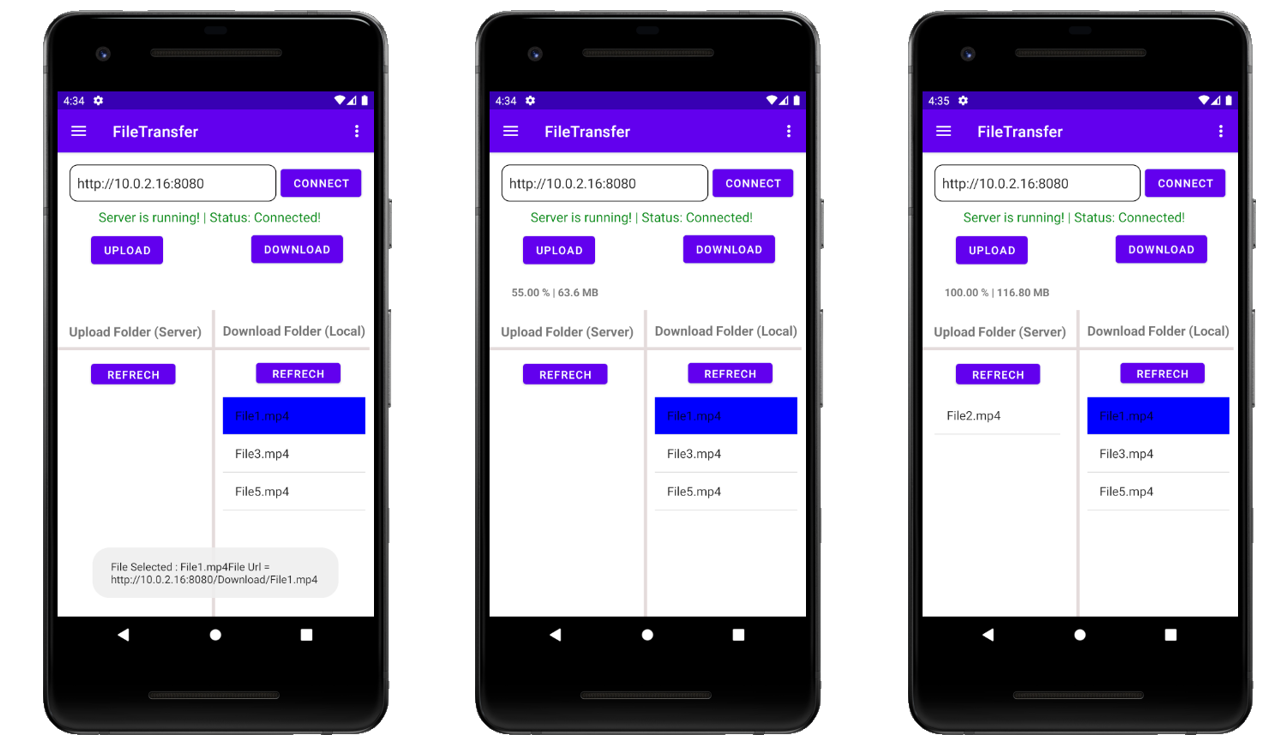
Pour se faire on sélectionne d’abord un fichier de la liste des fichiers sous le répertoire « Upload Folder" du serveur puis on lance le téléchargement en cliquant sur le bouton "Download".

* 1 2 3*

Sur l’imprime écran ci-dessus on peut suivre la progression du téléchargement sous le bouton « Download » (33.9 5% | 39.67 MB sur la deuxième photo)

* **Le bouton "Upload"** : l’action sur ce bouton permet de lancer le téléversement d’un fichier depuis le répertoire local "Download Folder" vers le répertoire distant "Upload Folder" du serveur.

On commence d’abord par sélectionner un fichier dans la liste des fichiers sous le répertoire "Download Folder" puis on lance l’opération de téléversement avec le bouton "Upload".



*1 2 3*

Sur l’imprime ecran ci-dessus on peut noter la progression du téléversement sous le bouton « Upload » (55.00 % | 63.6 MB)

* 1. *Activité Administration*

Dans cette activité on analyse les connexions établie avec l’application et les transferts des fichiers réalisés par ces connexions.

*5.4.1 Conception Fonctionnelle*

**Le diagramme de cas d’utilisation**

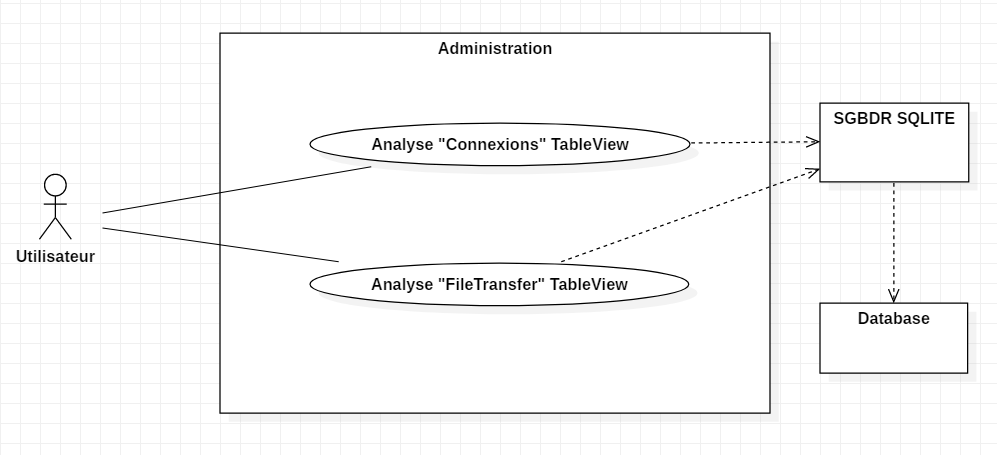


Figure 29: Le diagramme de cas d’utilisation de l’activité "Administration"

*5.4.2 Design de l’interface utilisateur*

A travers l’interface utilisateur on peut distinguer deux types de tableaux :

* **Tableau "Connexions" :** ilcontient les champs suivants :

"userId" : est l’identifiant de l’utilisateur.

"Downloads" : est le nombre des téléchargements effectués par une connexion utilisateur.

"Uploads" : est le nombre des téléversements effectués par une connexion utilisateur.

"ConnexionTime " : est la date et heure à laquelle la connexion a été établie par l’utilisateur.

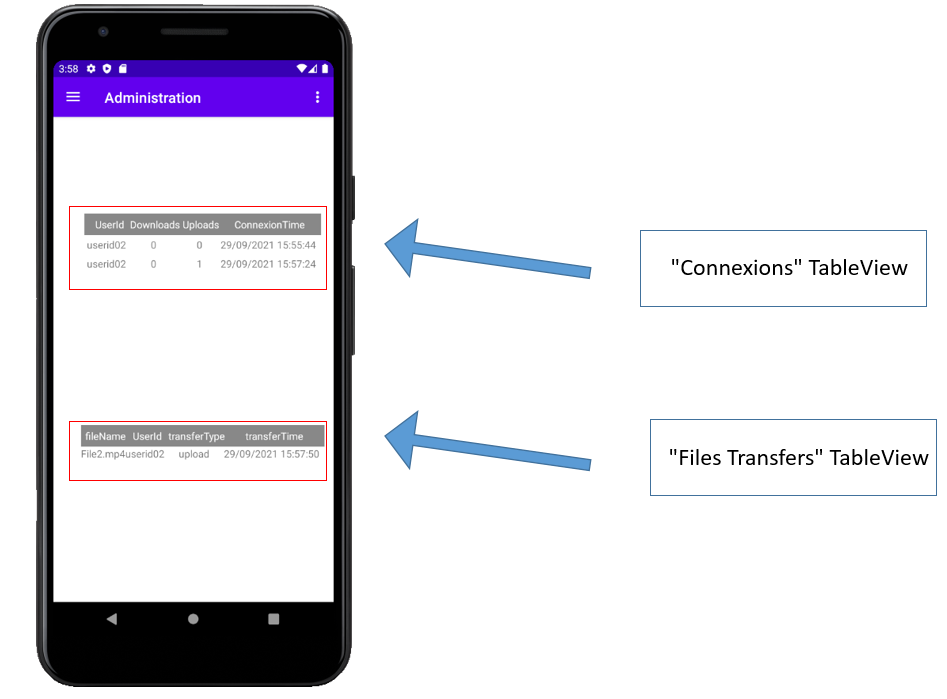
* **Tableau "Files Transfers" :** il contient les champs suivants :

"fileName" : est le nom du fichier transféré

"UserId" : est l’identifiant d’utilisateur qui a effectué le transfert du fichier

"TransferType" : il renseigne sur le type de transfert ayant pour valeur "download" ou bien "upload".

"TransferTime " : est la date et l’heure à laquelle le transfert a été effectué.

**

* + 1. *Conception d’architecture*

**Le modèle conceptuel des données**

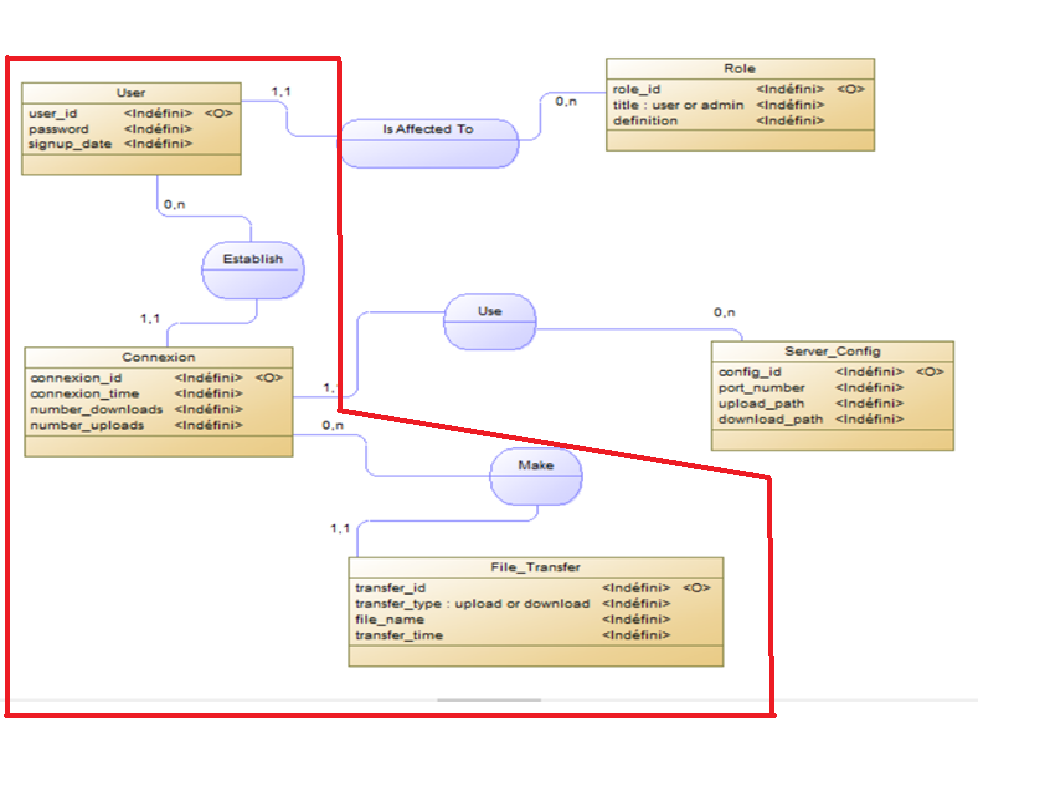
****

Figure 30: Le MCD de l’activité "Administration"

Pour afficher les deux tableaux on va utiliser les trois tables suivantes :

* Table "User"
* Table "Connexion"
* Table "File Transfer"

En exécutant des requêtes SQL sur la base de données.

*5.4.4 Codage*

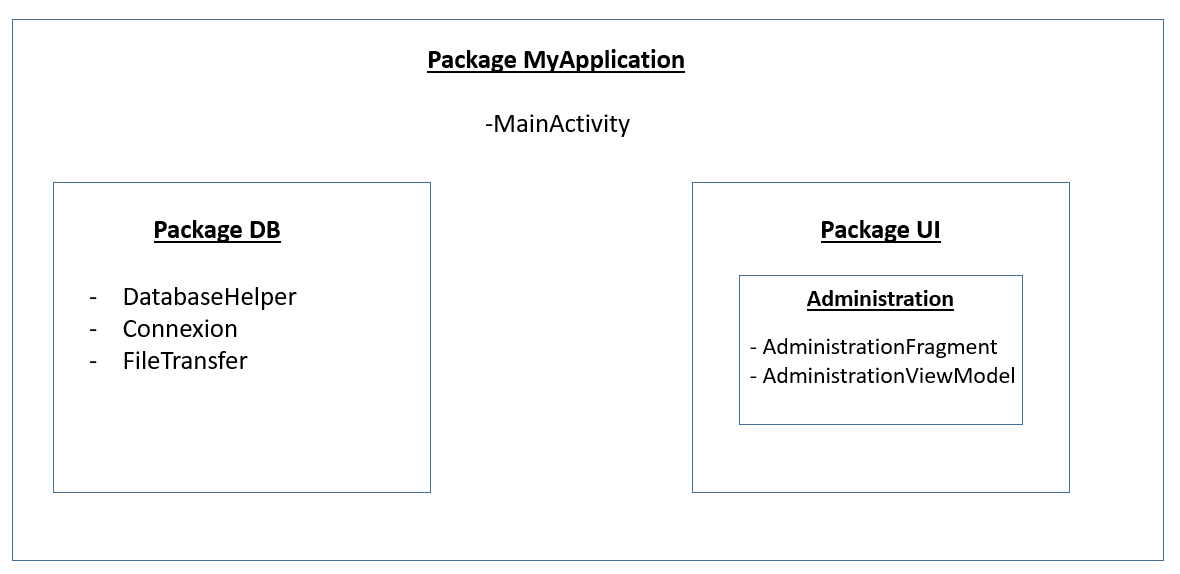
****Le diagramme de package :**

Figure 31: Le diagramme de package du processus "Administration"

- MainActivity : c’est la classe de démarrage de l’application.

**Le Pacakage DB** : il contient les classes suivantes :

- "DatabaseHelper" : est la classe permettant de gérer la base de données

- "Connexion" : est la classe représentant la table "Connexion" dans la base de données.

- "File\_Transfer" : est la classe représentant la table "File\_Transfer" dans la base de données.

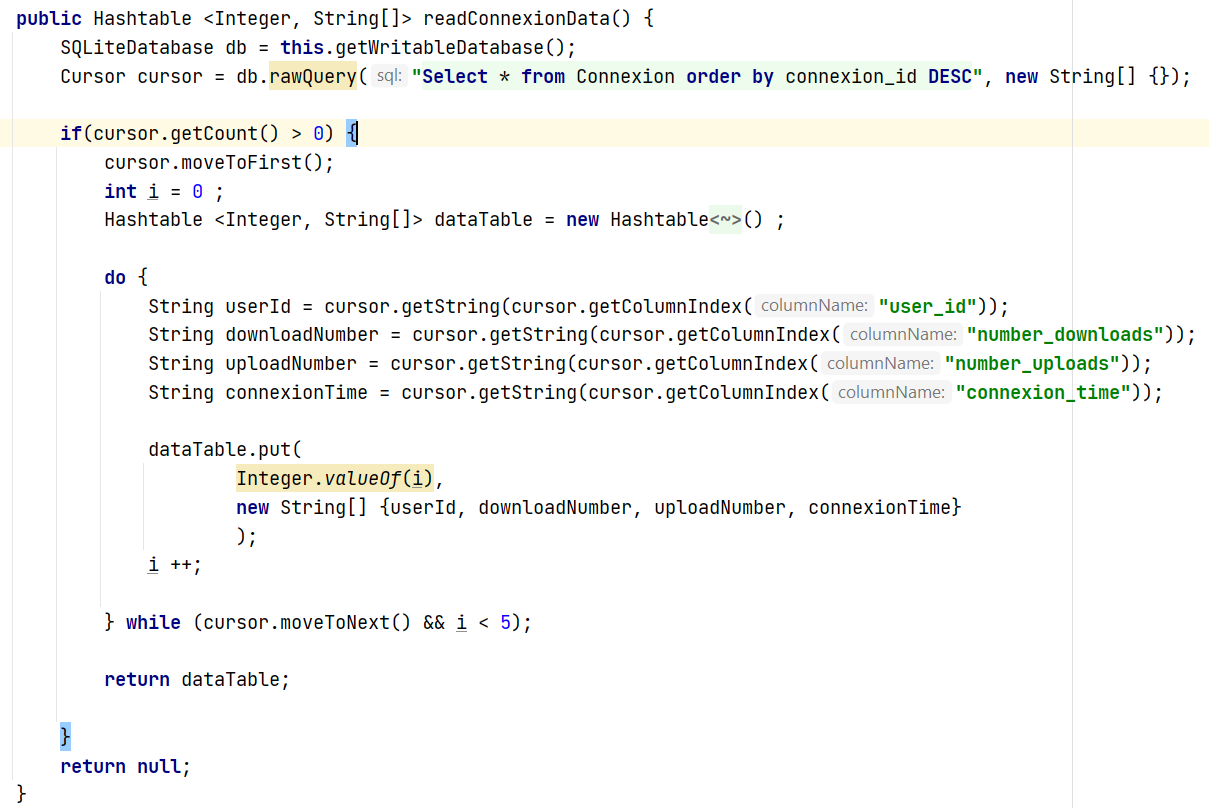
**Le Pacakage Ui** : il contient les classes en rapport avec l’interface utilisateur :

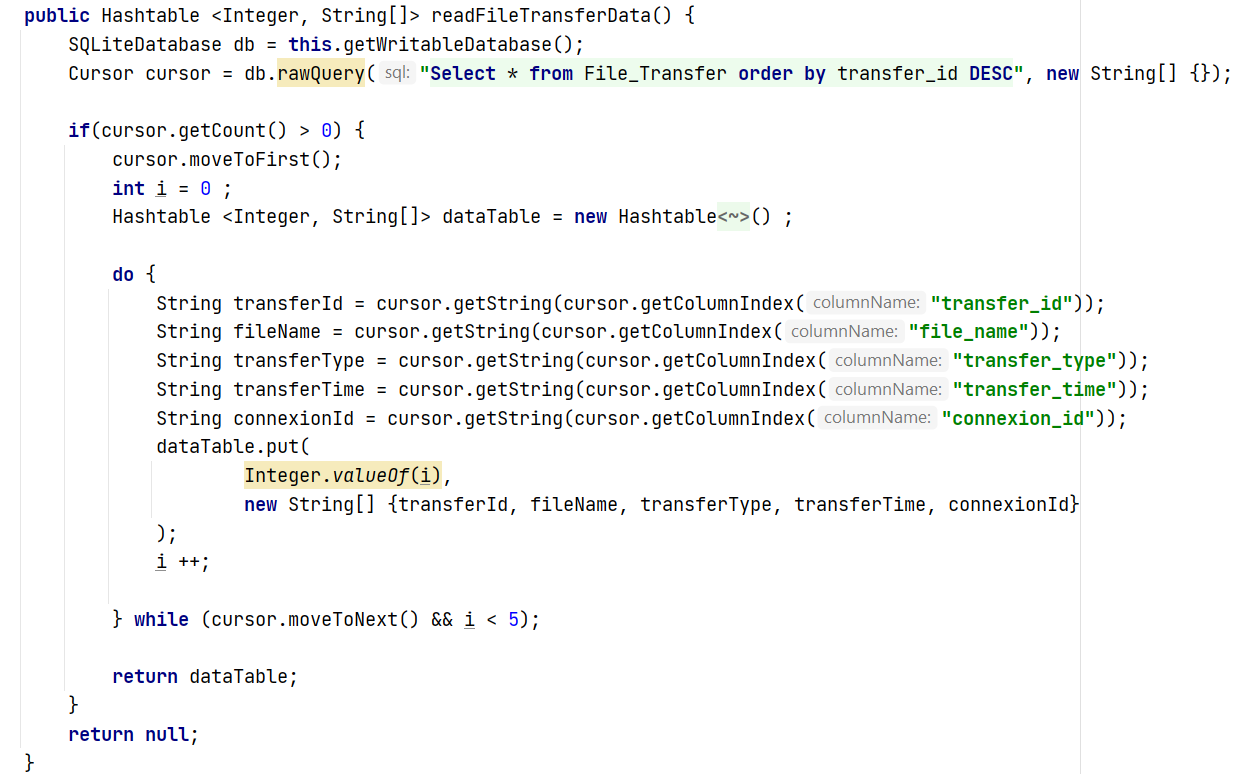
- AdministrationFragment : elle permet de contenir les composants de l’interface utilisateur.

- AdministrationViewModel : elle permet de stocker le modèle de données liée à l’interface du

Fragment.

**Implémentation en Java**

**On commence par implémenter les deux méthodes "readConnexionData" et "readFileTransferData" permettant respectivement de lire les données des tables "Connexion" et "File\_Transfer" en exécutant les requêtes SQL suivantes :

****

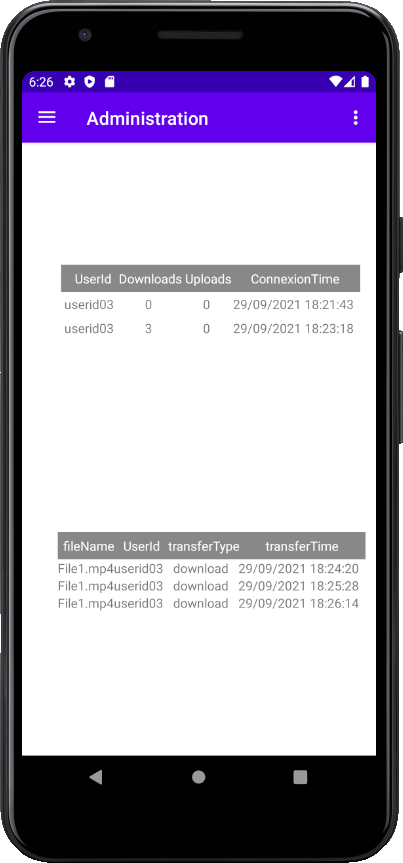
Une fois les données récupérées dans une "HashTable" on passe à leur affichage en utilisant la méthode "addView()" des classes suivantes :

* « TableLayout » : permet de construire la vue du tableau.
* « TableRow » : permet de construire la vue d’une ligne dans le tableau.
* « TextView » : permet de construire les vues des cases dans le tableau.

*5.4.5 Test*

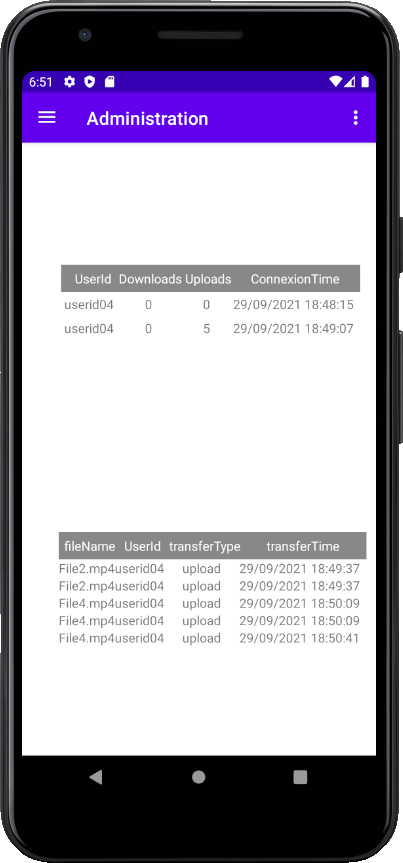
**Analyse de nombre des téléchargements d’une connexion d’utilisateur**

Dans le cas suivant l’utilisateur identifié par "userid03" a effectué trois téléchargements durant une seule connexion :

**

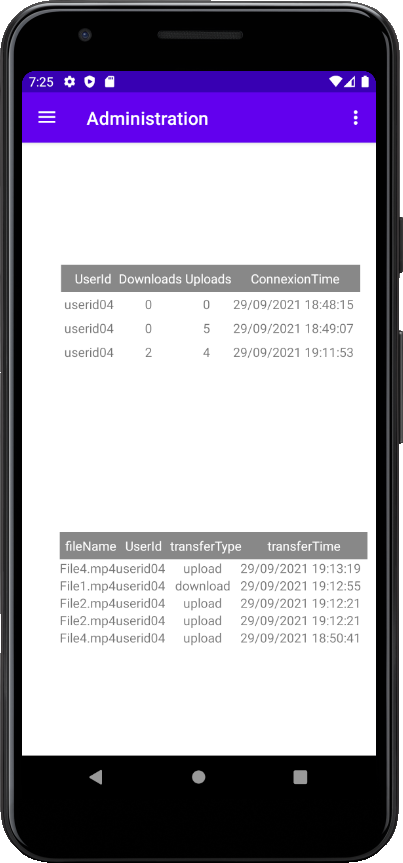
**Analyse de nombre des téléversements d’une connexion d’utilisateur**

Dans le cas suivant l’utilisateur identifié par "userid04" a effectué quatre téléversements durant une seule connexion :



**Analyse des transfert des fichiers des utilisateurs**

Pour chaque transfert de fichier on peut déterminer le type de transfert (download ou upload), l’identifiant de son utilisateur et la date et l’heure de son transfert.



1. *Validation*

Dans cette partie on va présenter le tableau de validation de différentes fonctionnalités réalisable par toutes les activités de l’application :

|  |  |
| --- | --- |
| *Fonctionnalités* | *Validation* |
| Activité "Connexion" |  |
| * "Sign Up" * "Login " | La fonctionnalité d’enregistrement des utilisateurs fonctionne correctement avec la vérification des conditions du format de saisie et l’enregistrement dans la base de données.  La fonctionnalité de "Login" des utilisateurs fonctionne bien avec la vérification des conditions du format de saisie et authentification au niveau de la base de données. |
| Activité "Configuration" |  |
| * "Save Settings" | La fonctionnalité "Save Settings" permet à la fois d’enregistrer et de mettre à jour les champs suivants dans la base de donnée :  - "Port Number"  - "Download Folder"  - "Upload Folder" |
| * "Start Server" | La fonctionnalité "Start Server" permet de démarrer le serveur en écoute sur le numéro de port spécifié tout en affichant son statut de fonctionnement ("Running !") et son adresse IP locale sur le réseau |
| * "Stop Server" | La fonctionnalité "Stop Server" permet d’arrêter le serveur tout en mettant son statut de fonctionnement à "Stopped !". |
| Activité "File Transfer" |  |
| * "Connect" | La fonctionnalité "Connect" permet de se connecter au serveur à l’adresse IP et numéro de port spécifiés et d’afficher son statut de fonctionnement ("Running !"). |
| * "Refrech" (sous "Download Folder") | La fonctionnalité "Refrech" permet d’afficher la liste des fichiers sous le répertoire "Download Folder" local. |
| * "Refrech" (sous "Upload Folder") | La fonctionnalité "Refrech" permet d’afficher la liste des fichiers sous le répertoire "Upload Folder" distant. |
| * "Download" | La fonctionnalité "Download" permet de télécharger des fichiers depuis le répertoire distant "Upload Folder" du serveur vers le répertoire local "Download Folder" |
| * "Upload" | La fonctionnalité "Upload" permet de téléverser des fichiers depuis le répertoire local "Download Folder" vers le répertoire distant "Upload Folder" du serveur. |
| Activité "Administration" |  |
| * Affichage du tableau "Connexions" | La fonctionnalité d’affichage permet d’afficher les cinq dernières lignes enregistrées dans la table "Connexion" avec les champs suivants :   * "User Id" * "Download Number" * "Upload Number" * "Connexion Time " |
| * Affichage du tableau "Files Transfers" | La fonctionnalité d’affichage permet d’afficher les cinq dernières lignes enregistrées dans la table "File\_Transfer" avec les champs suivants :   * "File Name" * "User Id" * "Transfer Type" * "Transfer Time " |

*7 Conclusion*

Durant ce projet j’ai eu l’occasion d’acquérir plusieurs connaissances dans le domaine de développement des applications Android parmi lesquelles je note :

* La construction des interfaces graphiques d’utilisateurs avec l’IDE Android Studio en utilisant les composants "Activity" et "Fragment".
* Les bases de programmation et de débogage avec le langage Java.
* Travailler avec le système de control des versions avec les outils Git et SourceTree ainsi le service d’hébergement de dépôt GitHub.
* Exploitation des bases de données relationnelles en Java à travers le SGBDR SQLite.
* La communication et L’échange de données entre sockets au-dessus de protocole HTTP
* Comprendre l’implémentation Multithreading du serveur HTTP permettant la gestion de plusieurs sockets/connexions simultanément.

Toutes ces connaissances me procurent une bonne base pour démarrer avec confiance dans le domaine de développement des applications mobiles en général et les applications Android en particulier.