***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 globale](#_TOC_250039)

[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 de 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.** Modéle en spirale ............................................................................................

**Figure 2.** Diagramme de cas d’utlisation globale...................................................

**Figure 3.** Diagramme MCD globale ........................................................................................

**Figure 4.** Cycle de vie d’une activité.................................................................................

.

## Listes des abréviations

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

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

## 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

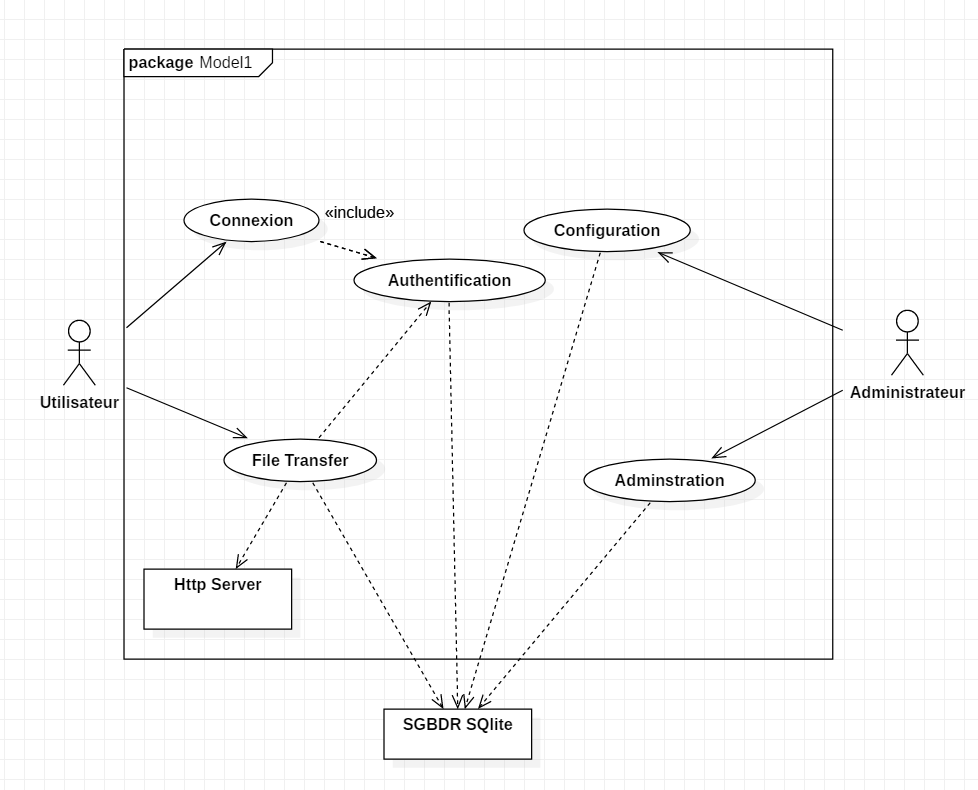
Lors de cette étude on a réalisé en particulier ce qui suit :

1. Le diagramme de cas d’utilisation global

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

3. La détermination de la technologie à adopter (Android SDK) .

*3.1 Diagramme de cas d’utilisation global :*



**Figure1** : Diagramme de cas d’utilisation globale

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

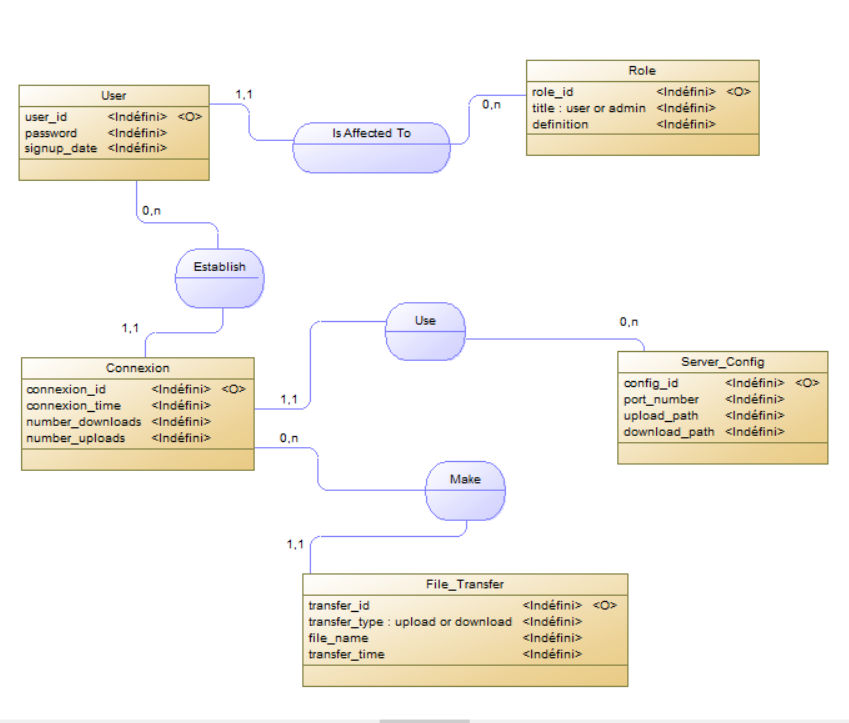
- « Connexion » : permettant d’enregistrer et de connecter l’utilisateur.

- « Configuration » : permettant de mettre à jour les paramètres du serveur et de lancer/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 utilisateurs et les transfert des fichiers.

*3.2 Diagramme MCD globale :*



**Figure2** : Diagramme MCD globale

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 : « 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.

**Figure3**: Cycle de vie d’une activité

*3.1.2 Le 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 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 livraision. 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](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 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.

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

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

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*

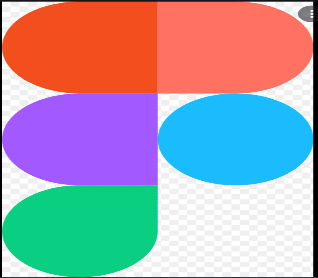
Nous avons utilisé pour la partie conception fonctionnelle le langage UML (Unified Modeling Language) : c’est un langage de modélisation unifié permettant de modéliser une application logicielle d'une façon standard dans le cadre de conception orienté objet.

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

-def diagramme cas utlisation :

Def diagrm d’activités :

* + 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 le modélisation des données on a 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*

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.

Les technologies utilisées 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*

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

La validation de fonctionnalités sera également accompagné par des imprimes écran confirmant le bon fonctionnement des activités

## Cycle de développement de l’application

*5.1 Détermination des activités de l’application*

Notre application comporte les 4 activités suivantes :

- Activité « Connexion »

- Activité « Configuration »

- Activité « Files Transfer »

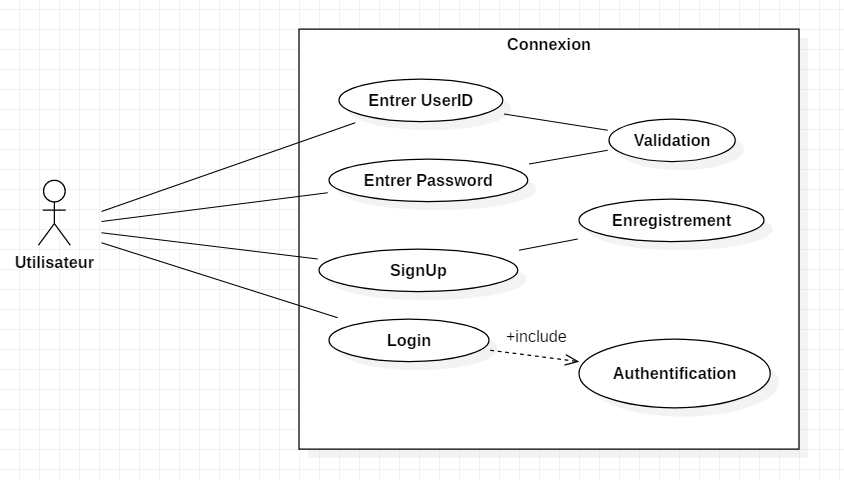
- Activité « Administration »

*5.1.1 Activité Connexion*

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

*5.1.1.1 Conception Fonctionnelle*

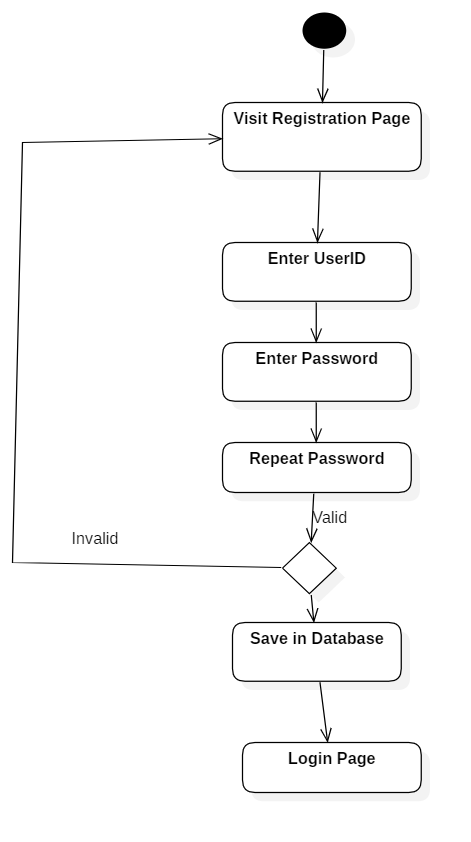
**Le Diagramme cas d’utilisation pour l’activité Connexion :**



**Figure 6** : Diagramme cas d’utilisation de l’activité connexion

**Le diagramme d’activité pour l’opération SignUp :**

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



**Figure** 7 : le diagramme d’activité pour le processus SignUp :

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

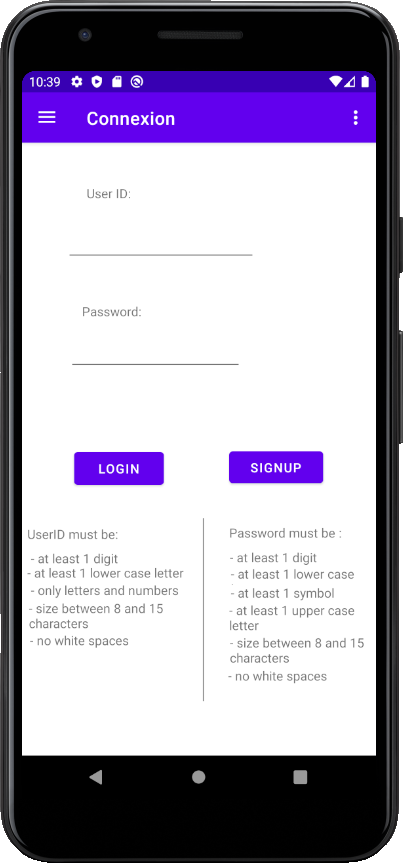
Dans ce processus l’utilisateur saisie "userid" et "password" pour s’identifier à l’application.



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

*5.1.1.2 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 la contrôle est effectuer respectant plusieurs conditions de validation



**Figure 8** :Activité Connexion

Pour la validation de "User Id" il  faut respecter les règles suivantes :

- Au moins 1 chiffre

- Au moins une lettre minuscule

- Au moins une lettre majuscule

- Que des lettres et des chiffres

- Pas d’espace.

- Taille entre 8 et 15 caractères.

Pour la validation de "Password" il faut respecter les règles suivantes :

- 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.

*5.1.1.3 Conception d’architecture*

*Dans cette partie on réalise le modèle conceptuel de donnée liées à l’activité en particulier on va créer les tables suivantes "User" et "Connexion"*

**Table User** :

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

Elle contient les champs : "user\_id", "password" et "signup\_date".

**Table Connexion :**

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

Cette table contient les champs : "connexion\_id", "connexion\_time", "number\_downloads" et "number\_uploads".



**Figure 9** : Le MCD de l’activité Connexion

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

## 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.

## Dans le MPD on va faire apparaitre les clés étrangères.

## La structure de la base de données devient la suivante :

**Table User :**

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

« role\_id » : c’est une clé étranger qui va pointer vers la clé primaires de la table rôle .

**Table Connexion :**

Elle contient les champs suivantes :

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

« config\_id » : c’est une clé étranger qui va pointer vers la clé primaire de la table « server\_config » .

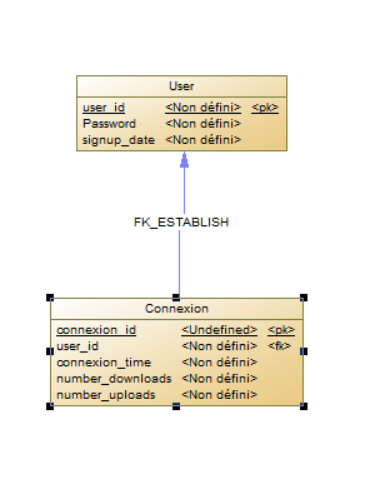


Figure10 :Diagramme MPD

*5.1.1.4 Codage*

**Le diagramme de package:**

**Package MyApplication**

- MainActivity

**Package DB**

-DatabaseHelper

-User

-Connexion

**Package UI**

-ConnexionFragment

-ConnexionViewModel

**Figure 11** : le diagramme de package du processus "Connexion"

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

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

- DatabaseHelper : Elle permet la création de la base de données avec la requête SQL

- User : Elle permet de représenter une ligne d’enregistrement dans la table « user » et elle

contient les méthodes « getters » et « setters ».

- Connexion : Elle permet de représenter une ligne d’enregistrement et elle contient les lignes « getter » et « setter ».

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

- ConnexionFragment : Elle permet de definir les détecteurs des évènements des clicks sur les de

l’interface de fragment.

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

fragment.

Les conditions de validations des champs « user\_id » et « password » sont implémenté avec

les expressions régulières suivantes:

Pour le champ User ID :

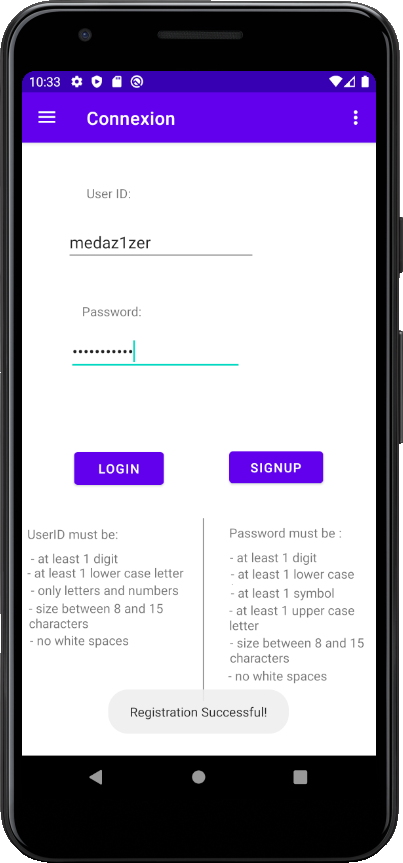
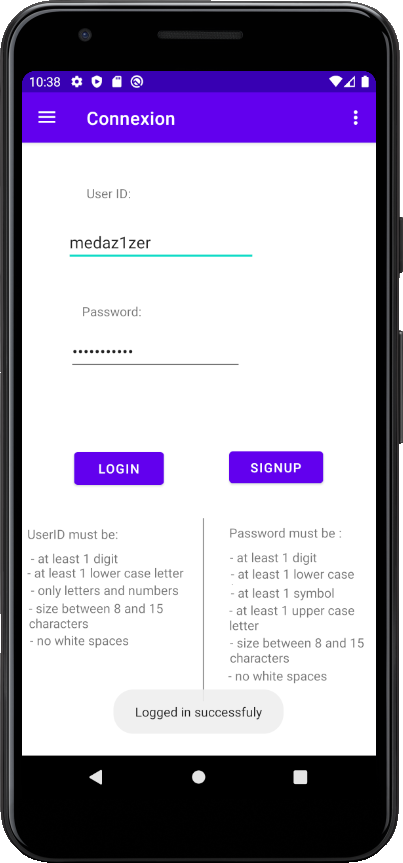
|  |  |
| --- | --- |
| Conditions de Validation | Expression réguliers |
| Taille entre 8 et 15 caractères | **^.{8,15}$** |
| au moins un chiffre | **^.\*[0-9]+.\*$** |
| au moins une lettre minuscule | **^.\*[a-z]+.\*$** |
| seulement des lettres et des chiffres | **^[a-z0-9]+$** |
| pas d’espace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

Pour le champ Password :

|  |  |
| --- | --- |
| Conditions de Validation | Expression réguliers |
| 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 | **^.\*[@#$%^&+=]+.\*$** |

*5.1.1.5 Test*

Pour la phase de test on valide les fonctionalités des activités avec les imprimes écrans suivantes :

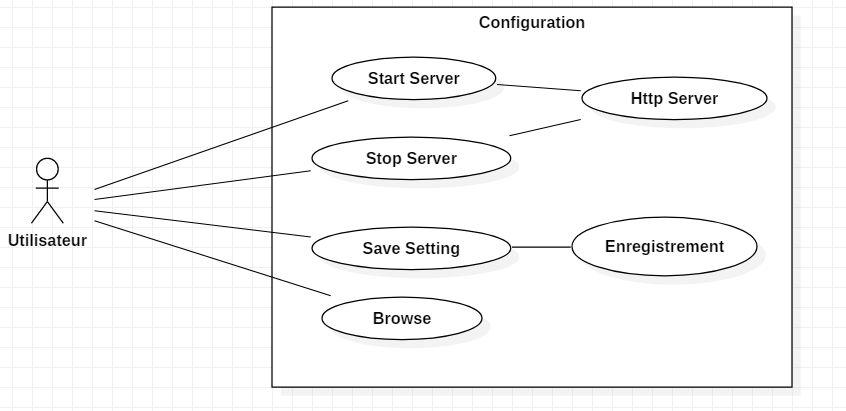


**Figure 12** : Message de confirmation de processus « sign\_up » et « login »

*5.2 Activité Configuration*

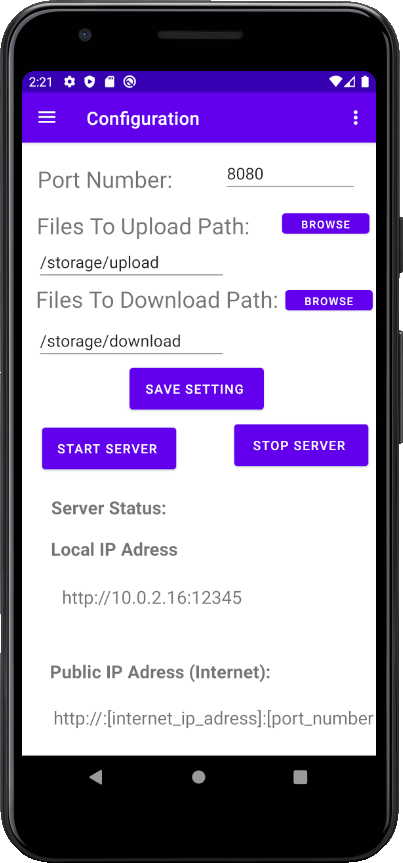
Dans cette activité on enregistre et on modifie les paramètres de configuration du serveur dans la base de données, on démarre et on arrête le serveur, on afficher l’adresse ip locale du serveur, et le statut de fonctionnement du serveur (« ruuning » or « stopped ») .

*5.2.1 Conception Fonctionnelle*

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

**Figure 13** : diagramme de cas d’utilisation de l’activité configuration

*5.2.2 Design de l’interface utilisateur*



**Figure 14** : Design de l’activité Configuration

Le button « Save Setting » 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 » et le répertoire de téléversement « Upload Folder ».

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

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

Le « Local Ip address » permet d’afficher l’adresse de serveur avec l’adresse ip sur le réseau local .

Le « Public Ip Address » permet d’afficher l’addresse de serveur avec l’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).

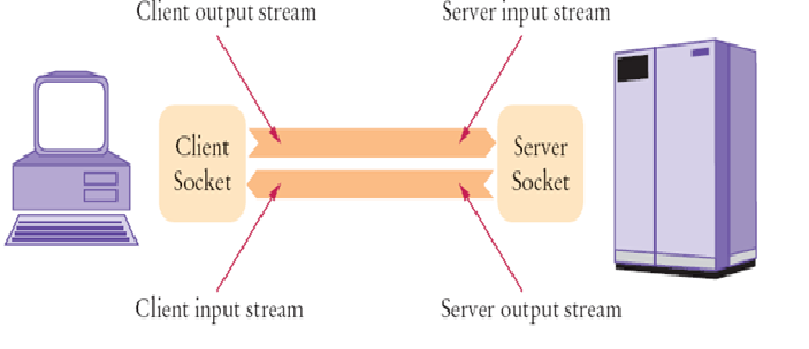
**Le Protocol HTTP** : HTTP signifie « **Hypertext Transfer Protocol** ». Ce protocole a été développé par Tim Berners-Lee au CERN (Suisse) avec d’autres concepts qui ont servi de base à la création du World Wide Web : le HTML et l’URI. Alors que le HTML (Hypertext Markup Language) définit comment un 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 15 :** Communications entre les sockets client et serveur

**Le diagramme de package :**

**Package MyApplication**

-MainActivity

-HttpServer

**Package UI**

**Package DB**

-DatabaseHelper

-Configuration

**Configuration**

ConfigurationFragment

ConfigurationViewModel

**Figure 16** : le diagramme de package du processus "Configuration"

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

- **Le Pacakage DB** : 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 » et elle contient les méthodes « getters » et « setters ».

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

- ConfigurationFragment : Elle permet de definir les détecteurs des évènements des clicks sur les

boutons de l’interface du fragment.

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

fragment.

*5.2.4 Codage*

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

- "Java.io" : on a en particulier utilisé les classes "InputStream" et "OutputStream" pour gérer les transferts de données entre les sockets.

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

- "Android SDK" : on a particulier utilisé les classes permettant de gérer les composantes "Activity" et "Fragment" et le fichier de configuration "AndroidManifest.xml".

**Package com.fst.MyApplication**

**Java JDk**

**Android SDk**

-Activity

-Fragment

-AndroidManifest.xml

**Java.io**

-inputstream

-outstream

-AndroidManifest.xml

**Java.net**

-ServerSocket

-Socket

-AndroidManifest.xml

**Figure 17** : 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 :**

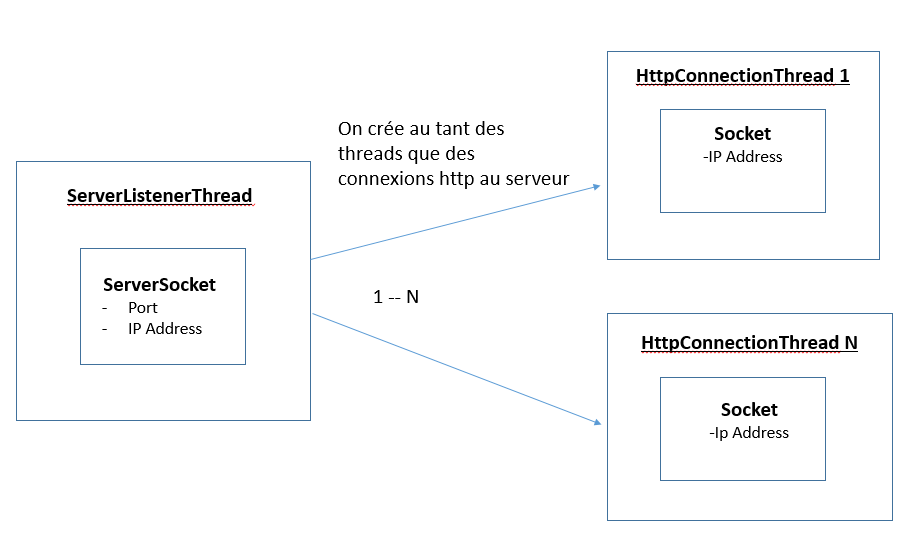
Notre serveur http est implémenté avec deux types de processus d’arrière-plan :

- "ServerListenerThread" : c’est la classe de type Thread permettant de faire fonctionner la classe "ServerSocket" en arrière-plan ce qui consiste précisément à accepter les connexions vers le serveur.

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

On a une seule instance de type "ServerlistenerThread"

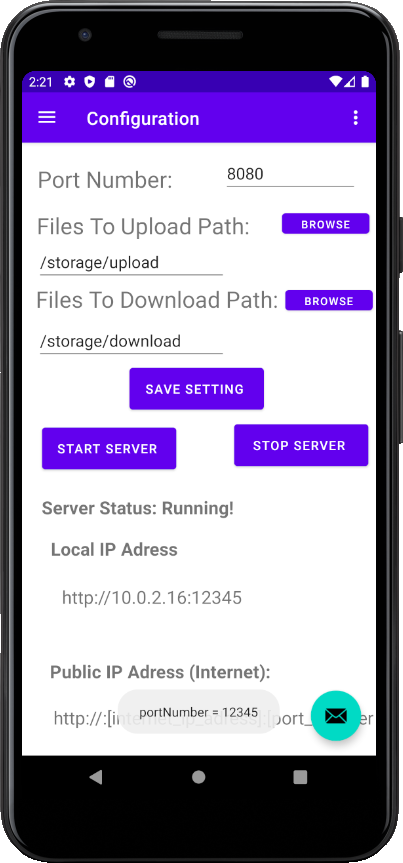
Pour chaque nouvelle connexion on crée une nouvelle instance de type HttpConnectionThread

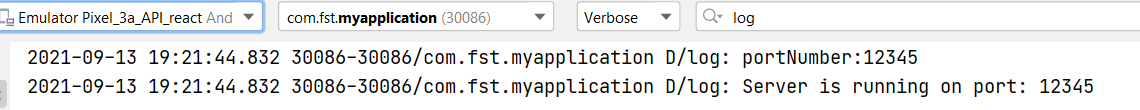
****

*5.2.5 Test*

Voici ci-dessous les différentes imprime écran correspondes a toutes les fonctionnalités de l’activités Configuration

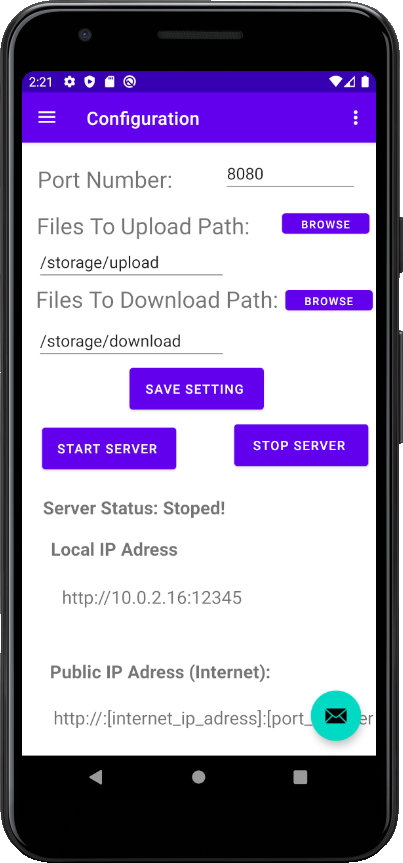
le serveur est en marche : (Server Status : Runnig !)

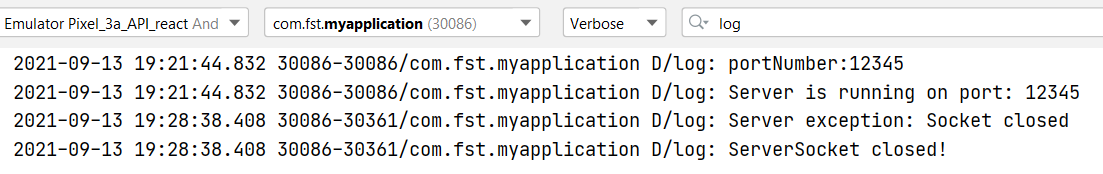
****

**Figure** : serveur en marche

**Figure : Serveur en arrêt**

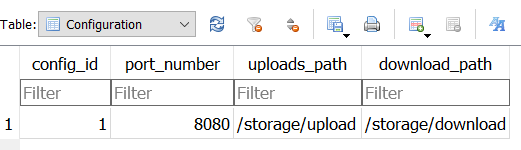
le serveur est en arret : (Server Status : Stopped !)

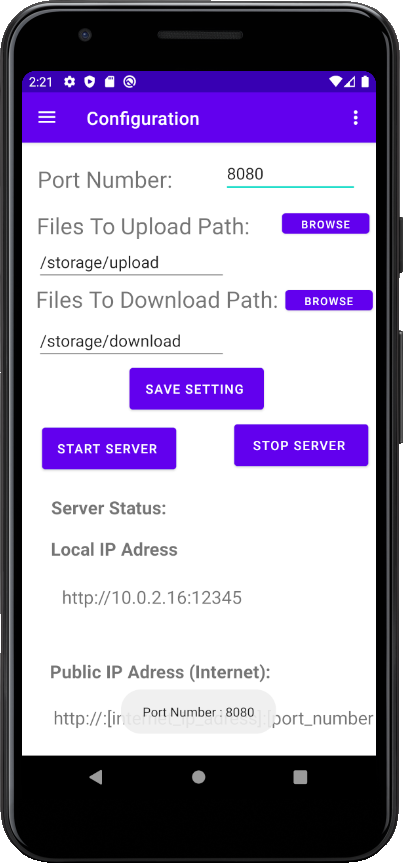




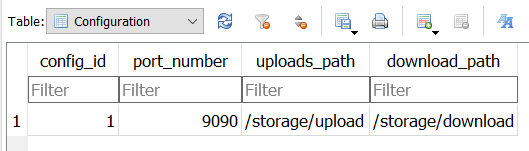
Le Button <SaveSetting> : permet d’ajouter une ligne dans la table de configuration ou bien mettre à jour d’une ligne existence dans la base de données.

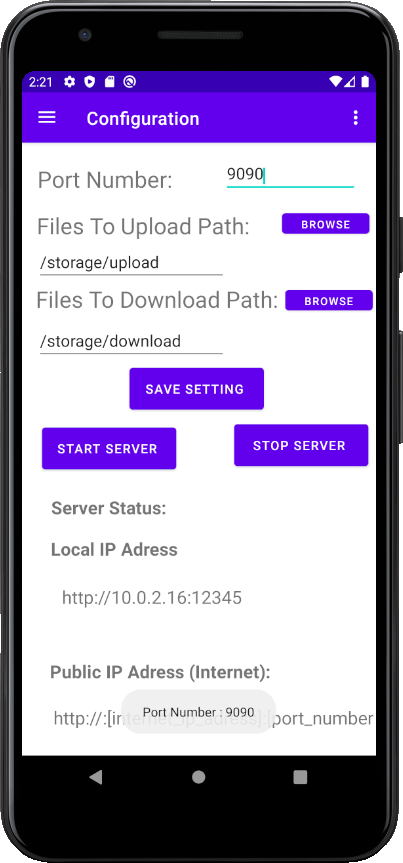
Ajouter une ligne dans table de configuration avec le numéro de port ajouté 8080 :





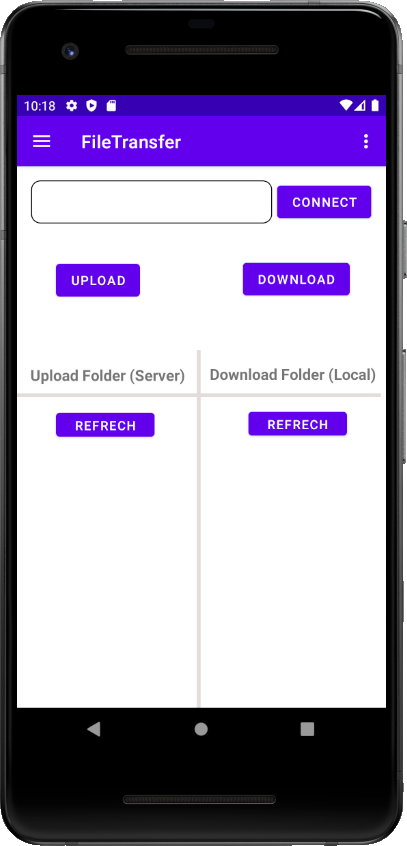
Mettre à jour la table de configuration avec le nouveau numéro de port ajouté 9090 :





*5.3 Activité File Transfer*

*5.3.1 Conception Fonctionnelle*

*5.3.2 Design*

L’utilisateur se connecte à un serveur distant une fois le serverur est connecté le Server Status est en marche ( Server Status : is Running !) .

L’utilisateur permet de télécharger des fichiers à partir de <Upload Server> , il sélectionné le ficher à télécharger puis en cliquant sur le bouton <Download> une progression de téléchargement est afficher sous le bouton , une fois le téléchargement est terminé , il faut appuiera sur le bouton < Refrech> pour vérifier que le téléchargent est bien télécharge dans le répertoire de téléchargement locale de l’appareil utilisateur .

Pour le téléversemnt des fichiers dans le serveur distant :

L’utilisateur sélectionne les fichiers à téléverser depuis le répertoire <Download Local> puis en cliquant sur le bouton <Upload> une progression de télé versement est afficher sous le button , une fois le télé versement est terminé les fichiers seront télé verser dans <Upload Server>

*5.3.3 Conception d’architecture*

*5.3.4 Codage*

*5.3.5 Test*

*5.4 Activité Administration*

*5.4.1 Conception Fonctionnelle*

*5.4.2 Design*

*5.4.3 Conception d’architecture*

*5.4.4 Codage*

*5.4.5 Test*

*6 Validation*

*7 Conclusion*