RAPPORT

SAE 3.02 - Développement d'applications communicantes

Réalisé par :
GIANFERMI Matteo – BERTONCINI Hugo

Encadrant :
MADJAROV Ivan

Date :
23/01/2025

Organisation :
BUT Réseaux&Télécommunitcations

Table des matières

1. Introduction

Dans le cadre de ce projet, nous avons développé deux applications mobiles utilisant Python et Kivy, connectées à un serveur TCP. Ces applications ont été conçues pour répondre à deux besoins essentiels : la gestion des magasins en ligne et les services bancaires associés.

La solution proposée repose sur une architecture client-serveur, intégrant des fonctionnalités telles que l'interaction avec des bases de données MySQL, la gestion des articles et des comptes, et une interface utilisateur intuitive grâce à Kivy.

L'objectif principal est de fournir une plateforme efficace et sécurisée, adaptée aux besoins des commerçants et des utilisateurs bancaires.

2. Objectifs du projet

1.Développer une architecture modulaire et scalable permettant de gérer les magasins et les services bancaires.

2.Assurer une communication fluide et sécurisée entre les applications clientes et le serveur via des sockets TCP.

Objectifs Spécifiques :

Pour l'application de magasin :

- Permettre aux administrateurs de gérer les articles (création, mise à jour, suppression).
- Offrir une expérience utilisateur intuitive pour consulter et acheter des articles.
- Intégrer une base de données pour stocker les informations des articles et des commandes.

Pour l'application bancaire :

- Gérer les comptes clients et effectuer des transactions en toute sécurité.
- Offrir des écrans de connexion, consultation de comptes et historique des transactions.
- Permettre une gestion efficace des données bancaires via un serveur centralisé.

3. Analyse des besoins

3.1. Besoins Fonctionnels

Application de magasin :

- Gestion des articles : ajout, suppression, mise à jour.
- Visualisation des articles disponibles avec leurs détails (nom, description, prix, photo).
- Gestion des commandes client.

Application bancaire :

- Création et gestion de comptes clients.
- Affichage de l'historique des transactions.
- Gestion des transferts bancaires.

3.2. Besoins Non Fonctionnels

- Sécurité des données : communication cryptée entre client et serveur.
- Haute disponibilité : le serveur doit gérer plusieurs connexions simultanées.
- Simplicité et ergonomie des interfaces utilisateur.

4. Conception de l'application

4.1. Architecture Générale

Le projet repose sur une architecture client-serveur, où les deux applications (magasin et banque) communiquent avec un serveur centralisé via des sockets TCP. Voici les composants principaux :

4.1.1. Applications clientes :

- Application Magasin : Fournit une interface pour les utilisateurs et les administrateurs de magasin, gère les articles et les commandes.
- Application Bancaire : Permet aux utilisateurs de consulter leurs comptes et d'effectuer des transactions.

4.1.2. Serveur TCP :

- Gère la communication entre les clients et les bases de données.
- Exécute les requêtes des clients pour accéder aux données ou effectuer des actions.

4.1.3. Bases de données MySQL :

- Une base pour les articles et commandes du magasin.
- Une base distincte pour les données bancaires.

4.2. Rôles des fichiers

app.py : Point d'entrée pour l'application magasin. Il définit la gestion des écrans et la logique principale.

appbank.py : Point d'entrée pour l'application bancaire, avec des fonctionnalités adaptées aux services financiers.

administrateur.py : Contient les fonctionnalités spécifiques aux administrateurs, comme la gestion des articles.

tcp_server.py : Implémente le serveur TCP qui traite les requêtes des deux applications clientes.

database.py : Fournit les fonctions de connexion aux bases de données (magasin et bancaire).

articles.py : Implémente la classe pour gérer les articles.

services.py : Contient les services métiers pour les clients, commandes, et transactions.

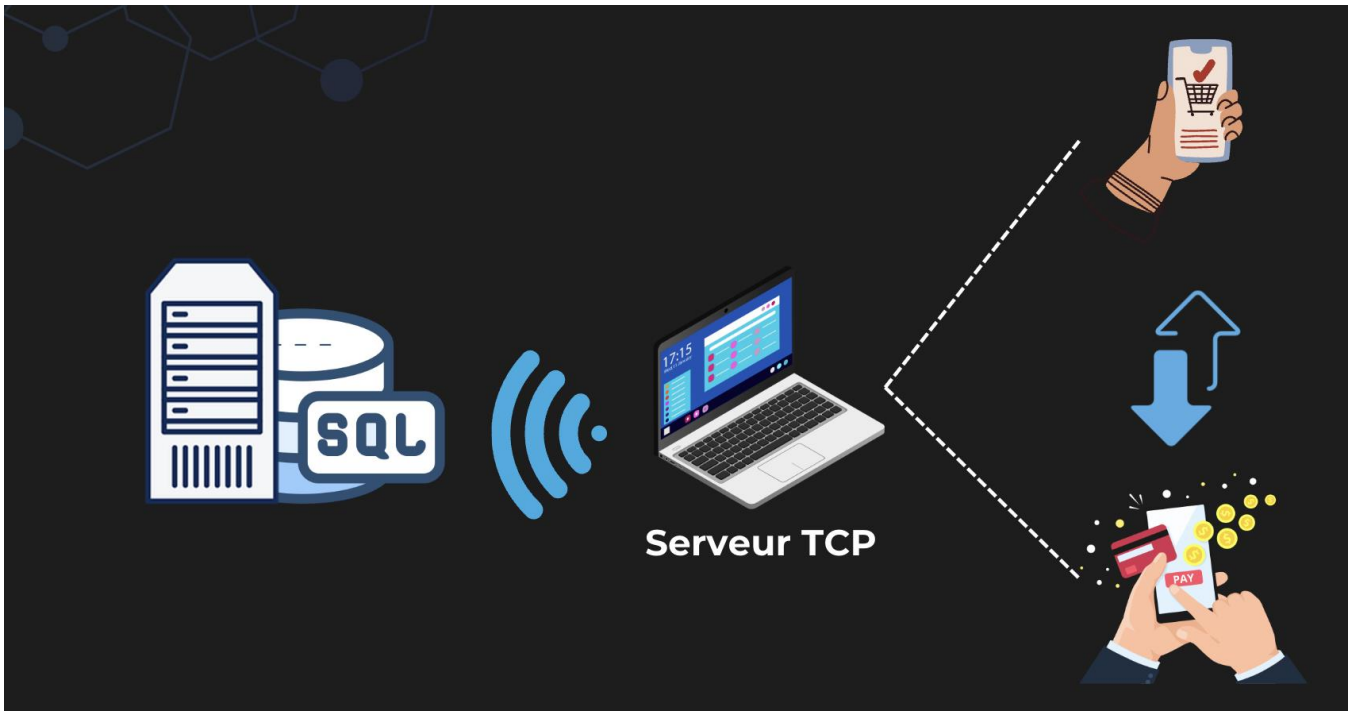
main_tcp.py : Contient les fonctions permettant d'envoyer des requêtes au serveur via des sockets.

mainbank.kv et **mainmagasin.kv** : Définissent les interfaces utilisateur pour les applications Kivy.

4.3. Schéma de l'Architecture

- Les applications Kivy (magasin et banque) se connectent au serveur TCP via un réseau local ou distant.
- Le serveur TCP agit comme un intermédiaire, recevant les requêtes des clients, les validant, puis les exécutant sur les bases de données MySQL.
- Les bases de données MySQL contiennent les données essentielles des articles, commandes, comptes bancaires, et transactions.

Exemple de schéma global:



De droite à gauche:

- Le client utilisateur (soit un PC, soit un smartphone) qui se connecte sur l'application magasin ou l'application bancaire.

- Les deux applications qui communiquent avec le serveur TCP pour permettre d'envoyer des requêtes SQL aux bases de données situé sur une VM pour récupérer des informations ou en envoyer.

5. Implémentation

5.1. Outils et Technologies

Python : Langage principal utilisé pour développer les applications et le serveur.

Kivy : Framework pour créer des interfaces utilisateur interactives.

MySQL : Base de données pour stocker les informations des magasins et des banques.

Sockets TCP : Protocoles pour la communication entre le serveur et les applications.

bcrypt : Bibliothèque pour le hachage des mots de passe et la gestion de la sécurité.

5.2. Fonctionnement des Modules

Communication TCP :

- Le client envoie des requêtes sous forme de JSON au serveur.
- Le serveur analyse la requête, exécute les actions nécessaires (ex. requêtes SQL), et renvoie une réponse.

Gestion des données :

- Les bases de données MySQL sont accédées via les fonctions de database.py.
- Les modules comme articles.py et services.py encapsulent la logique métier pour éviter des duplications.

5.3. Interfaces Utilisateur

- mainmagasin.kv : Conçu pour des écrans comme la connexion, la visualisation des articles, et les détails des commandes.
- mainbank.kv : Fournit des écrans pour la gestion des comptes bancaires, avec des formulaires simples et intuitifs.

6. Tests et validation

6.1. Scénarios de Test

Application Magasin :

- Ajouter, mettre à jour et supprimer un article.
- Consulter la liste des articles disponibles.
- Passer une commande et vérifier son traitement.

Application Bancaire :

- Créer un compte client.
- Effectuer une transaction et vérifier son historique.
- Consulter le solde du compte.
- Gérer l'agios et, par conséquent, les intérêts.

Serveur TCP :

- Gérer plusieurs connexions clients simultanément.
- Résister à des requêtes malveillantes ou incorrectes.
- Mettre à jour les bases de données ou envoyer des informations selon les requêtes des applications.

6.2. Résultats Obtenus

Les tests montrent une communication fluide entre les clients et le serveur. Les performances sont optimales pour des bases de données de taille modérée, et la sécurité des mots de passe est assurée via bcrypt. Les interfaces des applications communicantes permettent l'inscription ou la connexion d'un utilisateur et d'utiliser les services proposés par le magasin et la banque.

7. Défis rencontrés et solutions apportées

Problème : Gestion des connexions simultanées au serveur.

- Solution : Implémentation de threads pour chaque connexion client.

Problème : Sécurisation des mots de passe.

- Solution : Utilisation de la bibliothèque bcrypt pour le hachage des mots de passe.

Problème : Synchronisation des bases de données entre magasin et banque.

- Solution : Utilisation de bases de données distinctes pour isoler les données critiques.

8. Conclusion et perspectives

Le projet peut être déclaré comme un succès. En effet, les deux applications communicantes (magasin et banque) peuvent communiquer avec, différentes requêtes, le serveur TCP et les bases de données qu'il contient.

L'enregistrement et l'aspect de sauvegarde des données des comptes utilisateurs fonctionnent comme voulu, permettant ainsi de gérer les tentatives de connections et de vérifier qui peut se connecter ou non aux applications.

Les services proposés comme le suivi des commandes, passer commande ou voir les transactions effectuées permettent aux utilisateurs de prendre en main les différents services proposés sur les deux applications tout en communiquant avec le serveur.

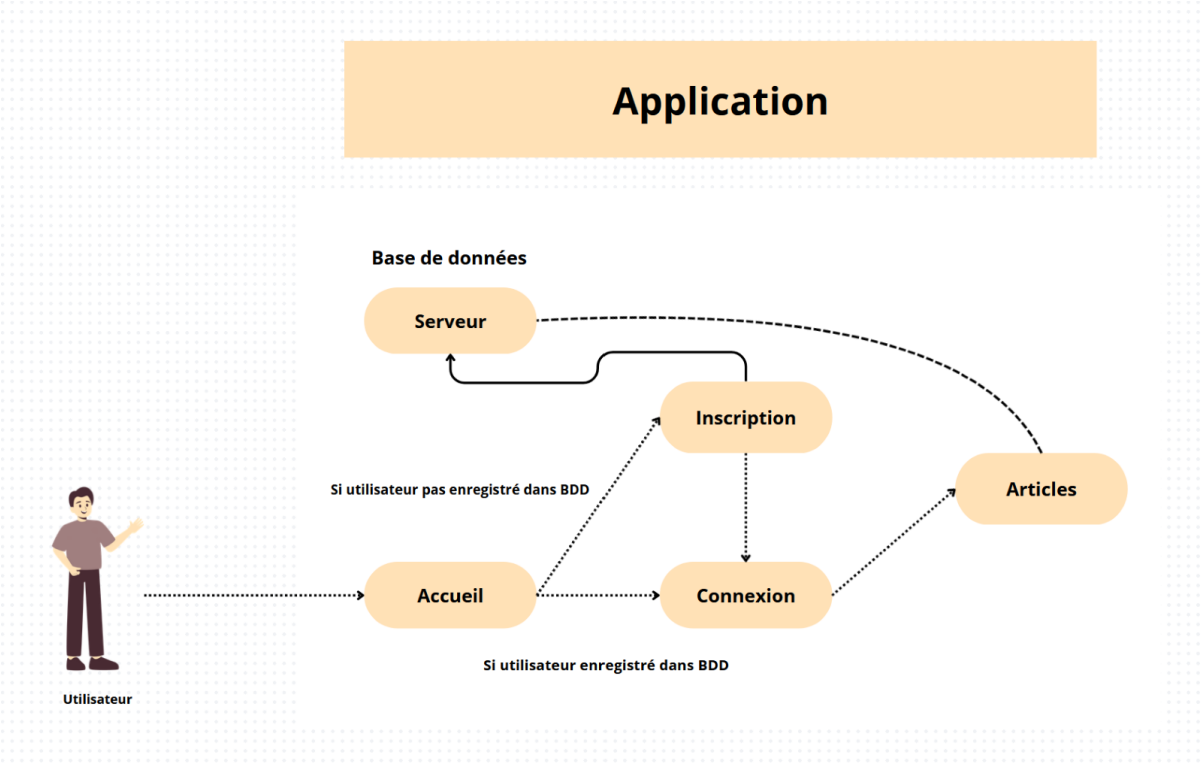
Comme améliorations possibles à envisager, il y a l'option de panier pour le magasin qui permettrait d'aller plus loin dans la simulation et de se rapprocher encore plus de ce qui se fait dans la réalité.

D'envisager une meilleure sécurité des données et éviter des attaques comme les injections SQL en utilisant différent outils de cybersécurité à disposition.

9. Annexes

Schémas et diagrammes

Le schéma qui suit montre, de manière simplifiée, comment l'application du magasin fonctionne.

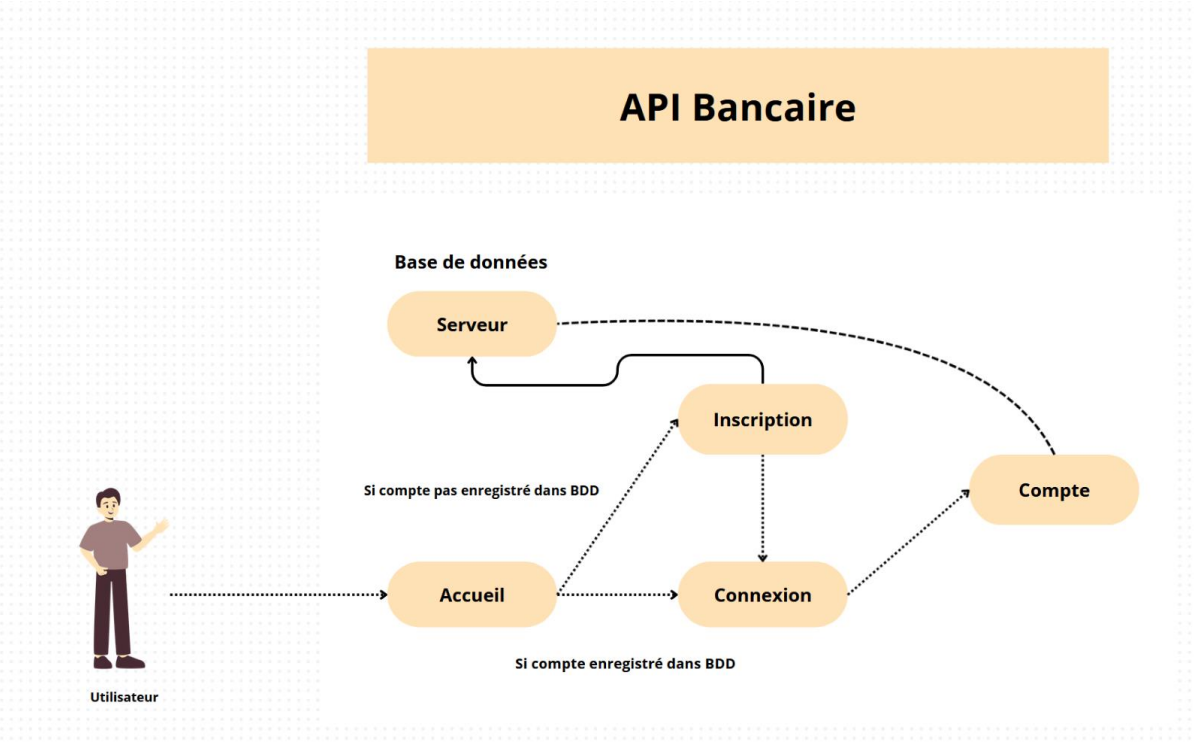


L'utilisateur, lorsqu'il lance l'application, arrive sur un écran d'accueil. Deux options lui sont alors proposées: s'inscrire si ce dernier ne possède pas de compte enregistré ou de se connecter s'il vient de s'inscrire ou s'il possède déjà un compte enregistré.

Lors de l'inscription, l'application communique avec le serveur TCP qui possède la base de données. Grâce à différentes requêtes, les données nouvellement entrées par l'utilisateur sont sauvegardées dans la base de données. Lorsqu'il voudra se connecter, l'application fera une requête au serveur mais pour vérifier les informations que l'utilisateur entre. Cela permettra d'autoriser ou refuser l'accès aux services.

Après s'être connecté, l'utilisateur arrive sur l'écran des articles qui sont eux-mêmes enregistrés dans la base de données et affichés grâce à des requêtes. D'ici, l'utilisateur peut passer commande.

De la même manière, le schéma qui suit montre, de manière simplifiée, comment l'application bancaire fonctionne.



Comme pour l'application du magasin, l'utilisateur, lorsqu'il lance l'application, arrive sur un écran d'accueil. Deux options lui sont alors proposées: s'inscrire si ce dernier ne possède pas de compte enregistré ou de se connecter s'il vient de s'inscrire ou s'il possède déjà un compte enregistré.

Lors de l'inscription, l'application communique avec le serveur TCP qui possède la base de données. Grâce à différentes requêtes, les données nouvellement entrées par l'utilisateur sont sauvegardées dans la base de données. Lorsqu'il voudra se connecter, l'application fera une requête au serveur mais pour vérifier les informations que l'utilisateur entre. Cela permettra d'autoriser ou refuser l'accès aux services.

Après s'être connecté, l'utilisateur arrive sur l'écran de son compte bancaire. Des informations sont alors affichées sur l'écran comme le nom, le numéro de compte, sa solde et ses intérêts. Il a la possibilité de se verser de l'argent, ou alors, de voir les différentes transactions qu'il a pu effectuer.

10. Documentation technique

Dans le cadre de ce projet, tous les codes sources ont été soigneusement documentés et organisés. Pour une consultation facilitée, ils sont disponibles sous forme de fichiers PDF inclus dans une archive compressée jointe à ce rapport.