PROGRAMMATION SYSTÈME ET RÉSEAUX

MINI-PROJET

TRANSACTIONS BANCAIRES

Résumé

Ce projet de développement a pour objectif la mise en place d'un système informatique. Ce système doit fournir un ensemble de fonctions (Partie 1) et doit être architecturé selon le modèle client-serveur (Partie 2). Il doit gérer des informations multiples de manière persistante (Partie 3) et doit utiliser un protocole de communication bien défini entre les clients et le serveur (Partie 4).

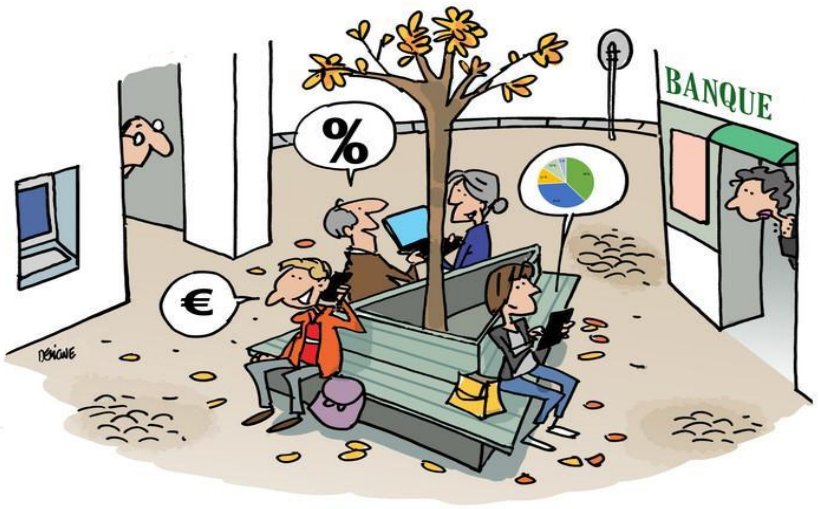

Table des matières

1. Fonctionnement : Transactions bancaires

L'objectif de ce projet est de mettre au point un système centralisé dans une banque en vue de gérer les transactions bancaires d'un ensemble de clients. Chaque client dispose d'un compte qui a une valeur qui correspond à la somme d'argent disponible. Un ensemble de transactions peut être réalisé sur chaque compte client. Chaque transaction peut concerner une opération de débiter ou de créditer. L'opération de débiter consiste à retirer une somme d'argent au compte. Alors que l'opération de créditer concerne l'ajout d'une somme d'argent au compte. Toutefois, la banque peut fixer pour chaque client une valeur indiquant le plafond de dépassement autorisé pour le débit si le solde est négatif.

1.1. Données manipulées

a) Données des comptes bancaires

Le système informatique dispose d'un ensemble de données où sont décrits tous les comptes bancaires d'une banque. Pour chaque compte, on dispose des informations suivantes :

- la référence précise du compte;
- la valeur du compte ;
- l'état du compte qui indique s'il est positif ou négatif
- la plafond de débit qui indique la valeur de dépassement maximale autorisée si le solde n'est pas suffisant au niveau d'une opération de débit.

Tout cet ensemble de données est enregistrée sous la forme d'un fichier texte **"comptes.txt"** et ce selon le format suivant :

Référence Compte	Valeur	Etat	Plafond Débit

Un exemple de contenu du fichier **"bien.txt"** est :

Référence Compte	Valeur	Etat	Plafond Débit
1000	500	Négatif	600
2000	300	Positif	0
3000	950	Positif	1000
4000	800	Négatif	800

Ce fichier doit être mis à jour à chaque fois où une transaction est réalisée. La mise à jour concerne la valeur et l'état si nécessaire.

N.B. Le Plafond Débit doit être initialisé au départ par la banque et ce pour chaque client (compte).

b) Données des transactions

Les données relatives aux transactions sont sauvegardées dans un fichier "**histo.txt**". Ce fichier permet d'assurer la traçabilité des différentes transactions reçues sur les différents comptes clients. Il est structuré selon le format suivant :

Référence Compte	Transaction	Valeur	Résultat	Etat

Un exemple du fichier "**histo.txt**" est le suivant :

Référence Compte	Transaction	Valeur	Résultat	Etat
1000	retrait	200	échec	négatif
2000	retrait	300	succès	positif
1000	retrait	100	succès	négatif
2000	ajout	450	succès	positif
3000	ajout	1000	succès	positif
4000	ajout	700	succès	négatif

Le compte 1000 sa valeur est « -500 » (d'après le fichier **"comptes.txt"**) et le plafond de dépassement autorisé est 600, donc le retrait de 200 n'est pas possible puisque la valeur du compte dépassera le plafond. En effet, -500-200 = -700> -600. Cependant, la deuxième transaction sur le même compte sera possible : -500-100 = -600 = plafond.

Pour le compte 2000, la valeur du solde est 300 donc il est possible de réaliser un retrait de 300 (300-300=0).

Pour le compte 4000, l'ajout de 700 au compte (-800) sera réalisé avec succès bien que la valeur du solde reste négative (-100).

Finalement, l'ajout engendre toujours un succès. Ce fichier "**histo.txt**" doit être mis à jour à chaque fois où une transaction est déclenchée.

c) Données des factures

Le système informatique dispose aussi d'un fichier qui décrit la somme que les clients devront payer. Cette somme est calculée sur la base du nombre d'occurrences d'un solde négatif sur un compte. Chaque occurrence encaisse un frais de 2% de la valeur retirée. Ce fichier de facture contient pour chaque compte :

- La référence du compte ;
- La somme totale à payer par le client correspondant.

Ces factures sont sauvegardées dans un fichier texte **"facture.txt"** et ce selon le format suivant :

Référence Compte	Somme à payer

Un exemple de contenu du fichier **"facture.txt"** est :

Référence Compte	Somme à payer
1000	2
2000	0
3000	0
4000	0

Le client du compte 1000 a réussi à retirer 100 et son compte est négatif. Donc il doit payer 100*2%= 2. Les autres clients n'ont rien à payer.

Dans le cadre de ce projet, on ne demande pas le détail des différentes factures. On ne s'occupe que par la somme totale que le client doit payer à la banque.

1.2. Opérations possibles sur les données

Le système informatique doit permettre de réaliser les opérations suivantes :

Pour la banque :

- **Consulter la liste des comptes :** en donnant la référence d'un compte, on doit pouvoir récupérer les informations le concernant (valeur, état et plafond) ;
- **Consulter la facture d'un compte :** il doit être possible de voir la facture correspondant à un compte en précisant sa référence.
- **Consulter l'historique des transactions** : il doit être possible de voir le contenu de l'historique des transactions.

Pour le client :

- **Réaliser des transactions :** un client doit pouvoir effectuer des transactions (retraits/ajouts). Il doit être capable de préciser à chaque fois une valeur pour la transaction ;
- **Recevoir une facture :** un client doit recevoir une facture à la fin de chaque transaction lui indiquant la somme qui lui reste à payer si nécessaire.

2. Architecture du système informatisé

La banque souhaite mettre en place un système informatique de gestion des comptes bancaires des clients composé des éléments suivants :

- un serveur central;
- des postes clients répartis chez les distributeurs, les agences et les terminaux bancaires.

Les postes clients doivent permettre, grâce à une Interface Homme-Machine appropriée (qui sera simulée ici sous forme d'une zone de saisie en mode caractères), de réaliser les différentes opérations prévues.

Le fonctionnement sera le suivant :

1. Selon la transaction, le poste client préparera une requête à envoyer au serveur.
2. La requête sera envoyée au serveur et le client se mettra en attente de la réponse.
3. Le serveur réceptionnera la requête et la traitera pour comprendre la demande du client.
4. Il effectuera ensuite le traitement associé,
5. Le serveur enverra le résultat de ce traitement au poste client concerné,
6. Le client réceptionnera le résultat et pourra enchaîner sur une nouvelle requête.

3. Gestion des données

Le serveur central aura à sa charge la gestion des comptes bancaires des clients et donc de l'ensemble des données (comptes, transactions et facturation). Les données manipulées par le serveur sont sauvegardées dans des fichiers. La concurrence d'accès au fichier exige l'emploi d'un mécanisme de synchronisation.

4. Gestion des échanges client/serveur

Les échanges entre les clients et le serveur doivent suivre un protocole bien défini pour que le serveur comprenne les requêtes des clients et pour que les clients comprennent les résultats renvoyés par le serveur. Le protocole à adopter dans ce cas de projet est TCP et ce pour assurer une communication en mode connecté et mettre en *temps réel* le contenu des différents fichiers.

5. Consignes

Le projet est travaillé et étudié en groupe de maximum **quadrinôme** mais la notation est **individuelle**.

5.1. Travail de base demandé

Vous disposez d'une squelette de programme fournie au niveau du cours. Après lecture des différents documents, il faudra le compléter pour fournir :

1. **Aspect réseau :** Protocole TCP ; Connexion entre au moins entre **trois** machines physiques ;
2. **Aspect système :** un serveur parallèle. Il s'agit de modifier le serveur TCP pour qu'il puisse gérer des requêtes en parallèle. Le serveur doit donc créer un nouveau thread chaque fois qu'il reçoit une requête ;
3. **Aspect programmation :** une sauvegarde/chargement des données dans des fichiers. Cela permet que les informations soient persistantes et non pas limitées à la session en cours.

5.2. Travaux complémentaires

Après la réalisation du travail demandé dans la section précédente, vous pourrez choisir un ou plusieurs points présentés ici pour améliorer votre programme.

1. **Aspect réseau :** Faites en sorte que votre serveur et votre client puissent fonctionner indifféremment en TCP ou en UDP. Le protocole TCP ou UDP sera choisi selon la valeur d'un argument de la ligne de commande ;
2. **Évolution fonctionnelle :** Introduisez la notion de profil en distinguant un profil administrateur d'un profil utilisateur. Désormais, l'administrateur sera le seul à avoir le droit d'effectuer une requête listant les utilisateurs et récupérant les données.

Dans tous les cas, avant de commencer à coder ces questions, faites une analyse des problèmes que vous voulez résoudre et des solutions que vous allez proposer. Si cette analyse est bien faite, le codage sera facile. Une bonne analyse (même sans implémentation) dans le rapport et la soutenance sera fortement valorisée.

5.3. Organisation du travail

Les groupes travaillent en quadrinôme au maximum. La répartition des tâches entre les quatre est libre.

Nous vous conseillons de :

- bien respecter le cahier de charges ;
- bien gérer le temps qui vous est imparti ;
- bien discuter dans le groupe ;
- réfléchir avant de programmer.

Pour la réalisation logicielle, nous conseillons vivement de suivre les étapes suivantes :

1. définir les fichiers *.txt;
2. réalisation d'une communication en TCP de chaque côté (client et serveur) ;
3. réalisation des opérations possibles de chaque côté (client et serveur) ;
4. ajout du mode parallèle (thread) ;
5. ajout de la synchronisation ;
6. ajout des fonctionnalités supplémentaires.

Note : certaines étapes peuvent être faites en parallèle. Pensez à vous répartir le travail.

5.4. Évaluation du travail

a) Soutenance

Chaque groupe aura 15 minutes pour présenter son travail. Le groupe expliquera brièvement la répartition du travail entre ses membres, la réalisation du projet et les problèmes rencontrés. Des questions seront posées et la démonstration est évidemment demandée.

b) Rapport

Le groupe doit rendre juste avant la soutenance un rapport de 10 pages au maximum donnant:

- l'organisation du travail dans le groupe ;
- la méthodologie utilisée dans le développement ;
- la pertinence de certains choix ;
- l'état courant du projet ;
- les difficultés rencontrées ;
- un rapide bilan de ce que vous a apporté ce projet.

Il s'agit de mettre en valeur la qualité de votre travail à l'aide d'un rapport. Pour cela le rapport doit explicitement faire le point sur les fonctionnalités du logiciel (lister les objectifs atteints, lister ce qui ne fonctionne pas et expliquer - autant que possible - pourquoi).

Ensuite le rapport doit mettre en valeur le travail réalisé sans paraphraser le code, bien au contraire : il s'agit de rendre explicite ce que ne montre pas le code, de démontrer que le code produit a fait l'objet d'un travail réfléchi et même parfois minutieux. Par exemple, on pourra évoquer comment vous avez su résoudre un bug, comment vous avez su éviter/éliminer des redondances dans votre code, comment vous avez su contourner une difficulté technique ou encore expliquer vos choix, pourquoi certaines pistes examinées voire réalisées ont été abandonnées.

Il s'agira aussi de bien préciser l'origine de tout texte ou toute portion de code empruntée (sur internet, par exemple) ou réalisée en collaboration avec tout autre groupe. Il est évident que tout manque de sincérité sera lourdement sanctionné.

Le rapport ne doit pas redonner des informations présentes dans ce document.

c) Code source

Vous devrez également fournir l'intégralité de votre code source en **Python** sous le format d'un dossier compressé qui doit avoir un nom selon le modèle suivant :

Groupe_Nom1_ Nom2_ Nom3_Nom4

Le groupe indique votre section : A, B, C, D ou E.

N'oubliez pas de commenter vos programmes. L'archive doit être nettoyée i.e. ne doit pas contenir des fichiers objets ou des exécutables.

Le dossier compressé est à charger au niveau de la partie concernée.