Florent CARRE
Claire DURAND

Mars 2005

Projet Java

# #

Gestion de Compte Bancaire.

Sommaire

Introduction

Le but de cette application est de permettre à une agence bancaire, ainsi qu'à ces clients de gérer leurs différents comptes.

Afin de répondre à l'attente de la société BVN, nous allons réaliser un projet suivant plusieurs étapes :

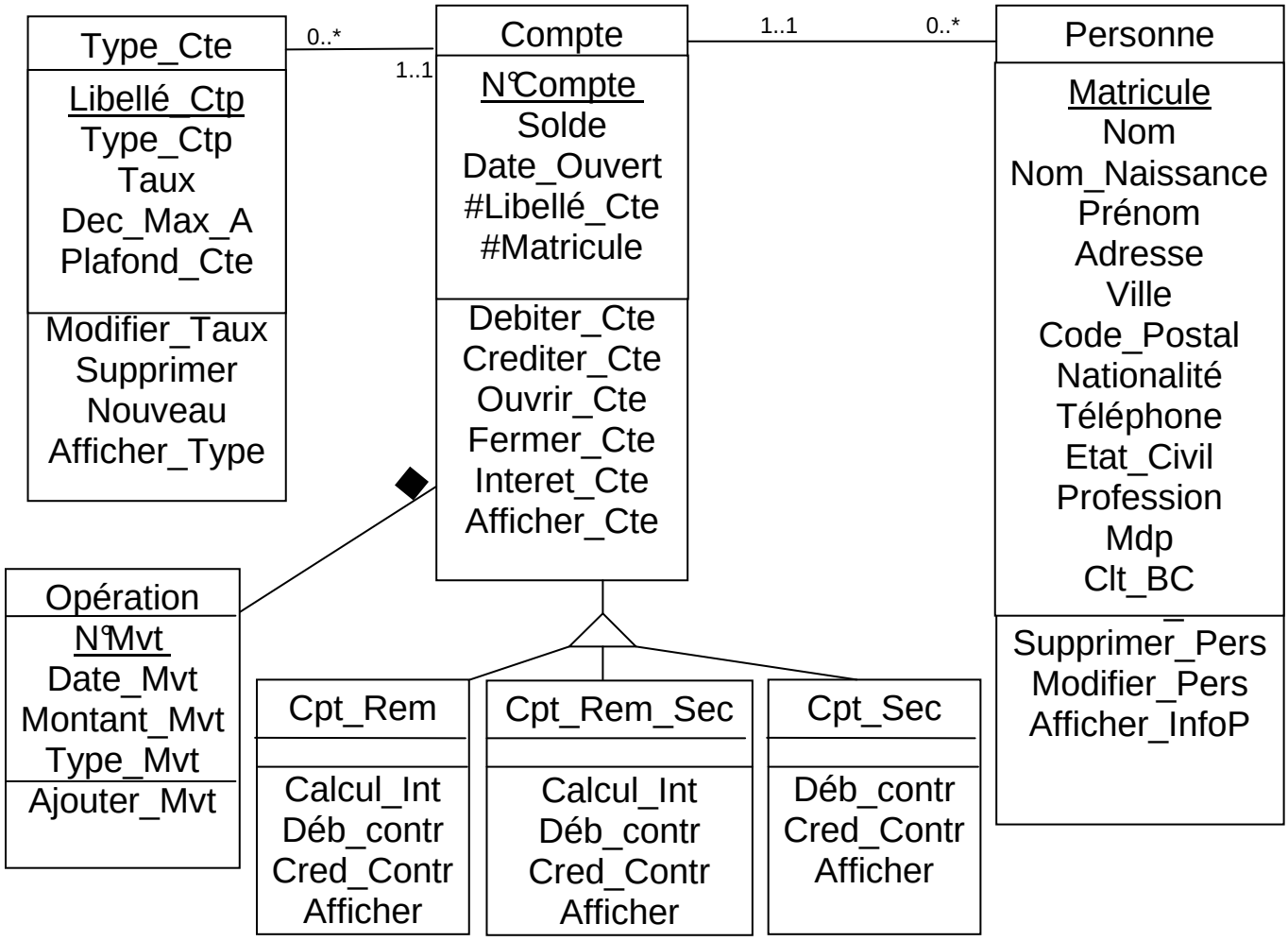
1. La première étape étant de concevoir un système d'information permettant la gestion des comptes. Pour sa mise en œuvre, nous avons utilisé le langage UML qui nous a semblé le plus approprié à ce cas.

2. La seconde est la mise en place et en relation d'un certain nombre de composant WEB selon le modèle 4 tiers MVC. Les principaux composants que nous avons dû installer pour réaliser ce projet en java, sont les Rutines java, les packages (Servlets/JDBC) et l'outil de développement Eclipse.

3. La troisième est le développement même des classes et des servlets. L'une des contraintes qui nous avait été formulé était de programmer en Java.

4. La quatrième étape consiste à implémenter le projet. Pour un bon fonctionnement et une viabilité de l'application il est nécessaire de mettre en place une architecture structurée et une configuration rigoureuse des composants.

1 - La conception UML

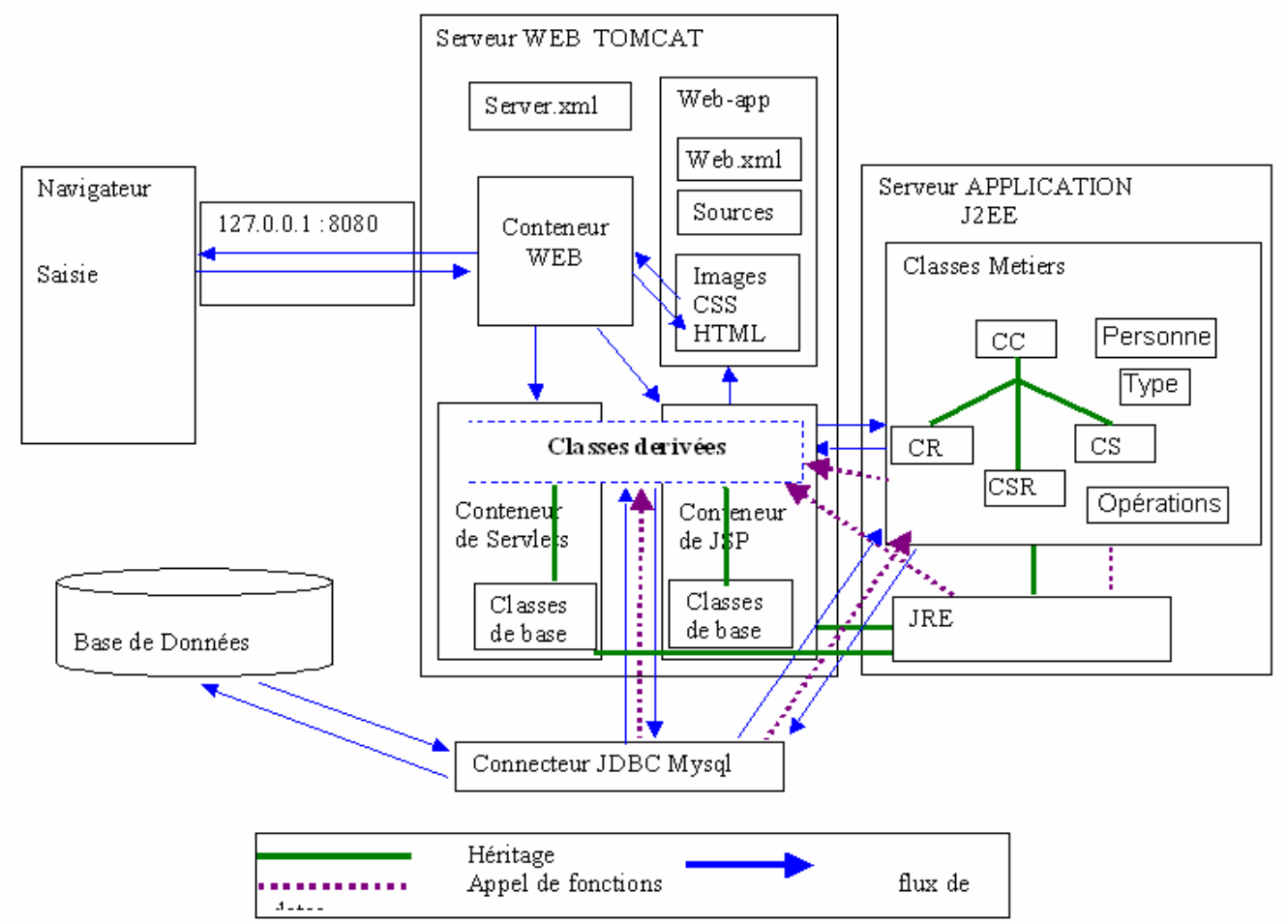


Les règles de gestions :

- Un client peut posséder plusieurs comptes.
- Un compte appartient à un et un seul Client.
- Un compte est de un et un seul type.
- Plusieurs comptes peuvent être du même type.
- Un mouvement ne dépend que d'un compte.
- Sur un compte peuvent être effectué plusieurs mouvements

2 - Le développement du Projet

Schémas de l'interactivité des composants :



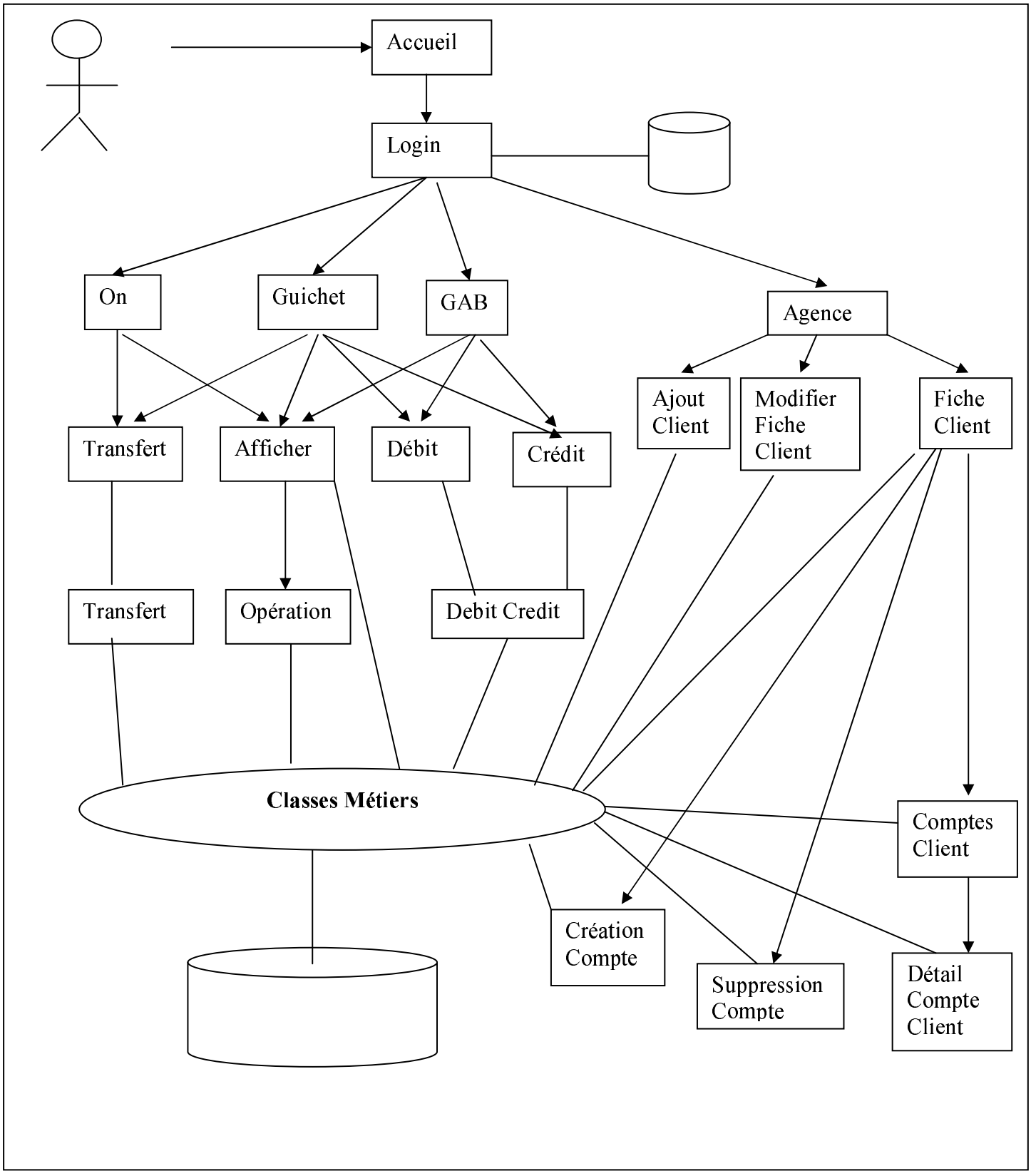
Description du schéma fonctionnel

- L'utilisateur doit se connecter au serveur via un navigateur traitant les données de format html, JavaScript, css …
- Chacune des requêtes est traitées par le Conteneur Web :
  → Pour les requêtes de type statique, le conteneur renvoie directement des pages html.
  → Pour les requêtes dynamiques, le conteneur Web transmet les instructions aux conteneurs de servlets.
- Une servlet a pour ordre, d'exécuter les instructions :
  → D'une part : soit en héritant (flux vert), soit en faisant appel à des méthodes du JRE (flux violet)
  → D'autre part : en faisant appel à des méthodes spécifiques (celles des classes métiers ou JDBC).
- Le résultat des instructions provoque la génération de code html, un code qui est traité par le Conteneur Web.
- Le Conteneur Web renvoie donc des pages Web dynamiques.
- Les classes Métiers, composées de méthodes métiers, sont structurées selon le respect du diagramme de classes : chaque classe est propriétaire de fonctions spécifiques et de propriétés alimentées par les ressources des tables correspondantes de la base de données.
  Ainsi, les méthodes de type JDBC, représentent le lien indispensable pour relier les propriétés du classe X, à une table Y.
  Les classes dérivées Compte_rémunéré, Compte_sécurisé et Compte_rémunéré et sécurisé, héritent des méthodes de la classe Compte_Courant (flux vert).

3 - Description du Programme

Cette application permet des sessions de 15 minutes maximum. Ce délai dépassé, l'utilisateur est automatiquement redirigé vers la page d'accueil (page par laquelle on accède au site). C'est à cette page que l'on s'identifie, pour ensuite être dirigé vers la partie client ou la partie agence du site.

Schéma :



3.1 - Les servlets/Class communes au deux modes de connexions.

Login :

Elle se connecte à la base de donnée « Bancaire » afin d'identifier de l'utilisateur. Chaque personne possède

S'il s'agit d'un client il sera dirigé vers la partie cliente de l'application et si c'est un agent, vers la partie agence.

Dans l'optique de préserver la confidentialité des données de l'utilisateur, l'envoie des critères se fait par la méthode POST.

En cas de succès, durant toute la session, l'utilisateur pourra accéder aux différentes pages Web lui donnant accès aux ressources (transactions, consultation) de la base.

Déconnexion

Elle permet a l'utilisateur de se des enregistrer et ainsi d'être renvoyé a la page d'accueil.

Classe Métier Compte_courant - Consultation des comptes

Paramètre d'entrés : Un numéro de compte

Paramètre de sortie : Un tableau double dimension de chaîne de caractère.

Traitement effectué : Récupération des propriétés de tous les comptes d'un Client

Classe Métier Compte_courant - Consultation d'opérations d'un compte

Paramètre d'entrés : Un numéro de compte

Paramètre de sortie : Un tableau double dimension de chaîne de caractère.

Traitement effectué : Récupération du détail de tous les mouvements effectué sur un compte.

Classe Métier Type_Compte - Consultation des types de comptes

Paramètre d'entrés : Un numéro de compte.

Paramètre de sortie : Un tableau double dimension de chaîne de caractère.

Traitement effectué : Récupération des occurrences d'un tuple de la table type_compte en fonction du numéro de compte passé en paramètre.

Classe Métier Compte rémunère - Calcul du montant de l'Intérêt

Paramètre d'entrés : Un numéro de compte.

Paramètre de sortie : Retourne le montant de l'intérêt mensuel.

Traitement effectué : Cette fonction permet de calculer les intérêts du mois. Pour cela il calcule les intérêts que le compte a généré entre chaque mouvement à l'aide la fonction : ((Nb_jour_Entre_Mvt)*taux*solde)/30, puis il additionne tous les résultats pour obtenir les intérêts du mois.

3.2 - Les servlets du mode Client.

La servlet : on.jsp :

Elle génère une page html qui permet au client de sélectionner l'environnement sur dans lequel il se trouve. Ceux – ci se comptent au nombre de trois :

- L'accès au distributeur
- L'accès en ligne via le service BVN.Net
- L'accès au guichet

Suivant l'accès sélectionné, il est possible d'accéder ou non à différentes servlets.

La servlet : Créditer.jsp

Elle génère une page html permettant à l'utilisateur de déposer de l'argent sur ses comptes. Pour se faire il saisie le montant qu'il veut ajouter et sélectionne dans la liste déroulant (préalablement remplie à l'aide d'un requête SQL) le compte sur lequel il veut le faire puis clic sur créditer pour valider l'opération qui sera traitée par la servlet Débit_Crédit.

La servlet : Débiter.jsp

Elle génère une page html permettant à l'utilisateur de retirer de l'argent sur ses comptes. Pour se faire il saisie le montant qu'il veut prendre et sélectionne dans la liste déroulant le compte sur lequel il veut le faire, puis clic sur débiter pour valider l'opération qui sera traitée par la servlet Débit_Crédit.

La servlet : Débit_Crédit

La servlet s'assure de la conformité de l'opération en fonction du type de compte. Si elle est autorisée, cela déclenche des transactions gérées par les classes Métiers Comptes et Opération. Le résultat de l'opération est ensuite transmis à l'utilisateur par une page html.

Classe Métier Opération - retrait et dépôt

Paramètre d'entrés : Un numéro de compte, un montant, une opération (débit/crédit).
Paramètre de sortie : Un entier déterminant l'échec ou le succès de l'exécution (0/1).
Traitement effectué : Enregistrement du mouvement dans la table opération relative au compte en question.

Classe Métier Compte_courant - mise à jour du solde

Paramètre d'entrés : Un numéro de compte, un montant, une opération (débit/crédit).
Paramètre de sortie : Un entier déterminant l'échec ou le succès de l'exécution (0/1).
Traitement effectué : Récupération du solde du compte grâce a l'héritage de con_compte.

	A : Avec plafond.	B : Sans Plafond
Crédit	Si Montant_Plafond > Solde + Somme ➔ Exécution d'une requête SQL effectuant l'opération.	➔ Exécution d'une requête SQL effectuant l'opération.
Débit	➔ Exécution d'une requête SQL effectuant l'opération.	

- **Les 3 classes présentées ci – après héritent de la classe Compte_Coutant.**

Classe Métier Dérivée Compte_Remunere - mise à jour du solde

Paramètre d'entrés : Un numéro de compte, un montant, une opération (débit/crédit).
Paramètre de sortie : Un entier déterminant l'échec ou le succès de l'exécution (0/1).
Traitement effectué : Récupération du solde du compte grâce a l'héritage de con_compte et le taux d'intérêt concerné grâce a la méthode algo.

	C : Avec plafond.	D : Sans Plafond
Crédit	Si Plafond > Solde + Somme ➔ exécution d'une requête SQL effectuant l'opération.	➔ exécution d'une requête SQL effectuant l'opération.
Débit	Si Solde - Solde*Taux d'intérêt > 0 ➔ Exécution de la requête SQL.	

Classe Métier Dérivée Compte_Sécurisé - mise à jour du solde

Paramètre d'entrés : Un numéro de compte, un montant, une opération (débit/crédit).
Paramètre de sortie : Un entier déterminant l'échec ou le succès de l'exécution (0/1).
Traitement effectué : Récupération du solde du compte grâce a l'héritage de con_compte.

	E : Avec plafond.	F : Sans Plafond
Crédit	Si Plafond > Solde + Somme ➔ exécution d'une requête SQL effectuant l'opération.	➔ exécution d'une requête SQL effectuant l'opération.
Débit	Si Solde > Montant*1,05 ➔ Exécution d'une requête SQL effectuant l'opération.	

Classe Métier Dérivée Compte - mise à jour du solde

Paramètre d'entrés : Un numéro de compte, un montant, une opération (débit/crédit).
Paramètre de sortie : Un entier déterminant l'échec ou le succès de l'exécution (0/1).
Traitement effectué : Récupération du solde du compte grâce a l'héritage de con_compte et le taux d'intérêt concerné grâce a la méthode algo.

	G : Avec plafond.	H : Sans Plafond
Crédit	Si Plafond > Solde + Somme + Intérêt mensuel ➔ exécution d'une requête SQL effectuant l'opération.	➔ exécution d'une requête SQL effectuant l'opération.
Débit	Si Solde + Intérêt Mensuel > Montant*1,05 ➔ Exécution d'une requête SQL.	

La servlet : transfert.jsp

Cette servlet est en fait un appelle successif des deux servlets précédentes. L'utilisateur sélectionne dans une liste déroulante le compte qu'il souhaite débiter et saisie le numéro du compte qu'il souhaite créditer ainsi que le montant de la transaction. La validation de cette page déclanche une seconde servlet.

La servlet : transfert

La servlet s'assure que le transfert des opérations de crédit et de débit nécessité par le transfert est conforme aux types de compte. Si c'est le cas, les transactions qui sont traitées par les classes Métiers Comptes et Opération. Le résultat de l'opération est ensuite transmis à l'utilisateur par une page html.

La servlet : Afficher

Elle récupère par l'intermédiaire d'une méthode métier les propriétés de tous les comptes du Client puis affiche le solde de chacun d'entre eux.

La servlet : Opération

Elle effectue un appel à une méthode métier grâce auquel elle peut récupérer les propriétés d'un compte pour ensuite générer une page html récapitulant les détails de tous mouvements réalisé sur le compte.

3.3 - Les servlets du mode Agence.

La servlet : Agence.jsp

Elle génère une page html qui propose à l'utilisateur 3 actions :

- Ajouter un Client
- Gérer les comptes d'un Client.
- Mettre à jour la fiche Client

La servlet : Ajout_Cli.jsp

Elle génère une page html qui permet à l'agent bancaire d'enregistrer un nouveau client.
La validation du formulaire, déclenche l'insertion des tuples dans la table client. Le résultat de la requête est transmis à l'utilisateur par une page html.

La servlet : Maj_Cli.jsp

Elle permet à l'utilisateur de modifier les informations relatives à un client. Le résultat de la requête est transmis à l'utilisateur par une page html.

La servlet : Afficher_Cli.jsp

La page qu'elle génère est remplie grâce à l'appel d'une méthode métier.
Elle offre la possibilité l'utilisateur d'accéder à l'ajout, la suppression, l'affichage des informations clientes ainsi qu'à la consultation des comptes d'un client.

Classe Métier Client - fiche

Paramètre d'entrés : Le matricule du Client
Paramètre de sortie : Un tableau double dimension de chaîne de caractère.
Traitement effectué : Récupération de toutes les informations relative à un client.

La servlet : Comptes_Client

Elle fait appel à une méthode de la classe métier afin de récupérer les informations sur les différents comptes que possède une personne et d'en restitués les soldes.

La servlet : Détail_Comptes_Client

Elle fait appel à une méthode de la classe métier afin de récupérer les propriétés d'un compte. Puis génère une page html récapitulant les détails de tous mouvements effectuer sur ce compte.

La servlet : Création Compte.jsp

Elle génère une page html qui permet à un agent bancaire de créer un nouveau Compte Client
La validation du formulaire, déclenche l'insertion des tuples dans la table compte, ainsi que la génération d'une seconde page alertant le succès ou l'échec de l'insertion.

La servlet : Suppression_Compte.jsp

Elle génère une page html qui permet à l'utilisateur de supprimer un Compte Client
La validation du formulaire, déclenche la suppression des tuples dans la table compte, ainsi que la génération d'une seconde page alertant le succès ou l'échec de la suppression

4 - Implémentation

4.1 – Installation et Configuration

Avant toute chose, quelques vérifications s'imposent :

- Vérifier les composants déjà présents sur votre poste afin de ne pas perturber l'implémentation de notre environnement du projet:
- Vérifier que le protocole TCP/IP est actif.
- Vérifier que votre firewall autorise les requêtes vers votre local host.
- Installation du pack J2SE Rutine Environnent + JDK 1.5
- Installation d'un IDE : EclipseV3
- Installation d'une base de donnée : MySQL Servers and Clients 4.0.21
- Installation d'un serveur Web : Tomcat 5.5.4
- Configuration : Ouvrir la console :
  - cd c:\mysql\bin\ [ou le chemin qui va bien]
  - >mysqld-nt –install
  - >mysql –u root
  - mysql>create database bancaire;
  - mysql>use bancaire;

- Télécharger le plug-in Tomcat pour Eclipse à mettre dans le répertoire …\éclipse\plugins\ : com.sysdeo.eclipse.tomcat_3.0.0·
- *Indiquer le chemin de Tomcat (permettre à Eclipse d'exécuter le serveur Web) et du fichier server.xml dans Eclipse (option de mise à jour a activer : lors de la création d'un nouveau projet, Tomcat insère dans son fichier de configuration server.xml, le nom du contexte et son path)*

4.2 – Déploiement.

- Importer les données de la base bancaire : mysql –u root bancaire < d:\ressources.sql
- Extraire le fichier zip « BVN.zip » dans le dossier « workspace » (espace de travail par default lors de l'installation d'Eclipse)
- Ouvrir Eclipse.exe
- File ➔ Import ➔ Existing Project into Workspace
  → Project name:BVN
  → Project contents: …\eclipse\workspace\BVN
- Dans le "Package Explorer": clic droit au niveau du repertoire BVN → mise à jour du contexte
- Démarer le service Tomcat
- Dand votre navigateur Web saisir l'URL : http://localhost:8080/BVN/accueil.jsp

5 - Conclusion

La réalisation de cette application à necessite beaucoup de travail, alternant les phases de développement et phase de teste. Néanmoins, le debuguage a été facilite par l'architecture n-tier qui nous également permis de prendre conscience de l'importance des risques d'erreur du type de « non-regression ».