Comptes Bancaires.

Plan :

Nous allons vous présenter dans différentes parties :

1. **La conception du projet en UML**
2. **Les Outils déployés et leurs utilités**
3. **Analyse Technique**
4. **Implémentation**

Introduction :

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

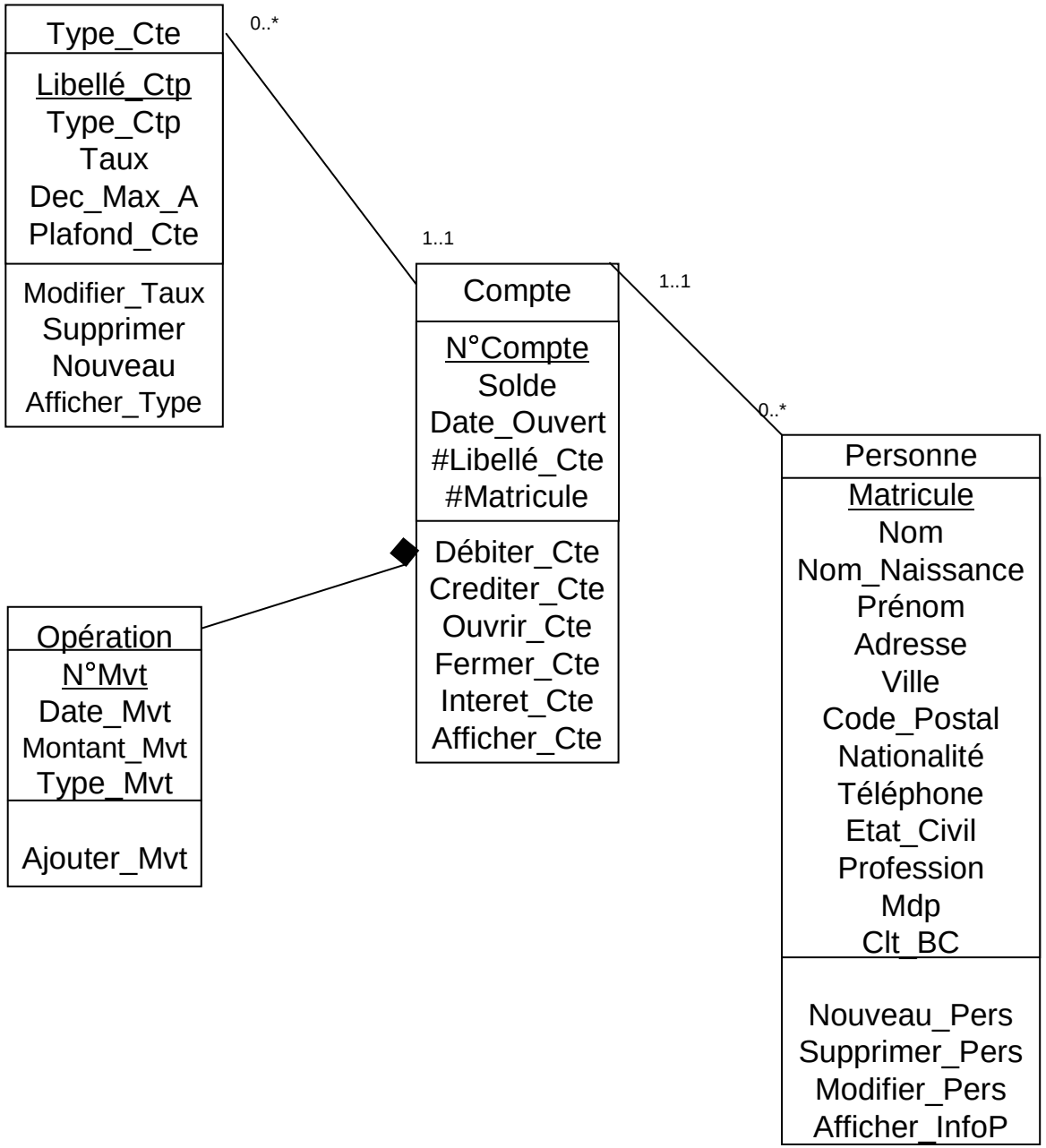
Sa réalisation a nécessiter une analyse de la demande engendrant l'utilisation d'une modélisation UML.

Une installation relativement complexe de composants.

Un développement de nombreuse servlets

Une implémentation adéquate.

1 - Schémas UML :



1.2 – Présentation des tables :

La table Type_Cte :

Ses champs :

- Libellé_Cte : Identifiant de cette table. Ce champ donne le sous-type de compte dont il s'agit.
- Type_Cte : Ce champs donne le type de compte dont il s'agit, il peu avoir pour valeur : Rémunérer, Courant, Sécuriser, Sécuriser & Rémunérer. [Dans l'application il sera indiqué par une liste déroulante].
- Taux : Le taux ne s'applique pas au compte courant. On le met a NULL pour les compte de se type.
- Dec_Mac_A : Le découvert n'est autorisé que sur les comptes courants, il est donc renseigné à NULL pour les comptes des autres types.
- Plafond_Cte : Le plafond ne s'applique pas a tous les comptes, les compte courant et les autres compte n'en ayant pas on se champs renseigné a 0.

Ses Fonctions :

- Modifier Taux : Permet de modifier le taux d'intérêt du compte.
- Supprimer : Vérifie si plus aucun client ne possède encore de compte de ce type puis supprime ce type de comte.
- Nouveau : Créer un nouveau type de compte.
- Afficher Type : Permet d'afficher toutes les propriétés d'un type de compte.

La table Compte :

Ses champs :

- N°Compte : Identifiant du compte.
- Solde : Solde du compte au jour de la requête.
- Date_Ouvert : Date où le compte a été ouvert.
- #Libellé_Cte : Clef étrangère de la table Type_Cte.
- #Matricule : Clef étrangère de la table Personne.

Ses Fonctions :

- Débiter_Cte : Permet de débiter d'un compte un somme donnée. Pour les compte de type Sécurisé ou Sécurisé et rémunéré on vérifie que le montant n'est pas supérieur a 5% du solde du compte.
- Créditer_Cte : Permet de créditer au compte un montant donné. Pour les comptes rémunérés, on vérifie que l'on dépasse pas le plafond autorisé, si il y en a un.
- Ouvrir_Cte : Permet à un agent de créer un nouveau compte à un client. On vérifie éventuellement que la personne ne possède pas déjà un compte de ce Libellé.
- Fermer_Cte : Permet à un agent de supprimer un compte d'un client. Avant cette opération, on s'assure que le solde du compte en question est bien nul.
- Interet_Cte : fonction qui permet de calculer les intérêts mensuels pour les comptes rémunérés
- Afficher_Cte : Permet d'afficher les détailles des opérations que le compte a subit.

- **La table Personne :**

- Ses champs :
- Matricule : Identifiant de la table Personne.
- Nom : Nom de famille du Titulaire du compte
- Nom_Naissance : Nom de naissance du titulaire du compte.
- Prénom : Prénom du titulaire du compte.
- Adresse : Adresse du titulaire du compte.
- Ville : Ville du titulaire du compte.
- Code_Postal : Code Postal du titulaire du compte.
- Nationalité : Nationalité du titulaire du compte.
- Téléphone : Téléphone du titulaire du compte.
- Etat_Civil : Etat civil du titulaire du compte.
- Profession : Profession du titulaire du compte.
- Clt_Emp : Il est renseigné a Client / Agent Bancaire. Il sert a savoir si la personne va être diriger vers la partie clients du site ou la partie Agence.

- Ses fonctions :
- Nouveau_Pers : Permet de rajouter un client ou un agent dans la base de données de l'entreprise.
- Supprimer_Pers : Permet de supprimer un client ou un agent dans la base de données si celui-ci ne possède plus aucun compte dans les établissements de l'entreprise.
- Modifier_Pers : Permet de modifier les informations personnelles d'un agent ou d'un client.
- Afficher_InfoP : Permet de consulter les informations personnelles d'un agent ou d'un client.

La table Opération :

La table Opération est une agrégation de la table Compte.

Ses champs :

- N°Mvt : Permet d'identifier chaque mouvement de man ière unique, c'est une entité faible de la table compte.
- Date_Mvt : Permet de garder la chronologie des mouvements du compte.
- Type_Mvt : Débit ou Crédit.
- Montant_Mvt : Sert à connaître le montant de l'opération. Peu aussi servir a calculer les intérêts.

Ses fonctions :

- Ajouter_Mvt : Cette fonction est utilisée à chaque mouvement d'un compte, il permet de garder une trace de tous les mouvements.

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 – Les Outils

- ***Eclipse :***

Eclipse est un IDE, mais surtout une plateforme pour héberger les applications.

Les IDE Java open source sont rares. La caractéristique essentielle d'Eclipse est l'extensibilité de l'environnement. Plus que de se focaliser sur un environnement de développement Java, les concepteurs d'Eclipse se sont efforcés avant tout de créer un socle applicatif.

Le cœur d'Eclipse est en fait composé :

1. D'un socle capable de charger des modules (plugins)
2. De modules de base permettant de gérer des ensembles de ressources (projets, fichiers, répertoires, …)
3. De modules pour permettre la création d'interfaces graphiques cohérentes. En utilisant cette librairie, les plugins ont un aspect homogène.
4. La partie développement Java n'est qu'un ensemble de plugins qui sont la première utilisation de ce socle. Leur principale caractéristique est d'être livrés avec Eclipse.

- ***Tomcat :***

Tomcat est le containeur de Servlet qui est utilisé dans les technologies d'implémentations Java. Les spécifications des Servlet Java et des Pages Java Server sont développées par Sun dans le cadre de la communauté Java.

Du point de vue des auteurs: Tomcat est le successeur de JServ qui n'est à ce jour plus développé. Tomcat supporte les dernières définitions de l'API jsp et des Servlets de SUN. Malheureusement, il est très difficile d'en compiler les sources à cause de "ant", son système de compilation. Il y a également une longue liste de dépendances.

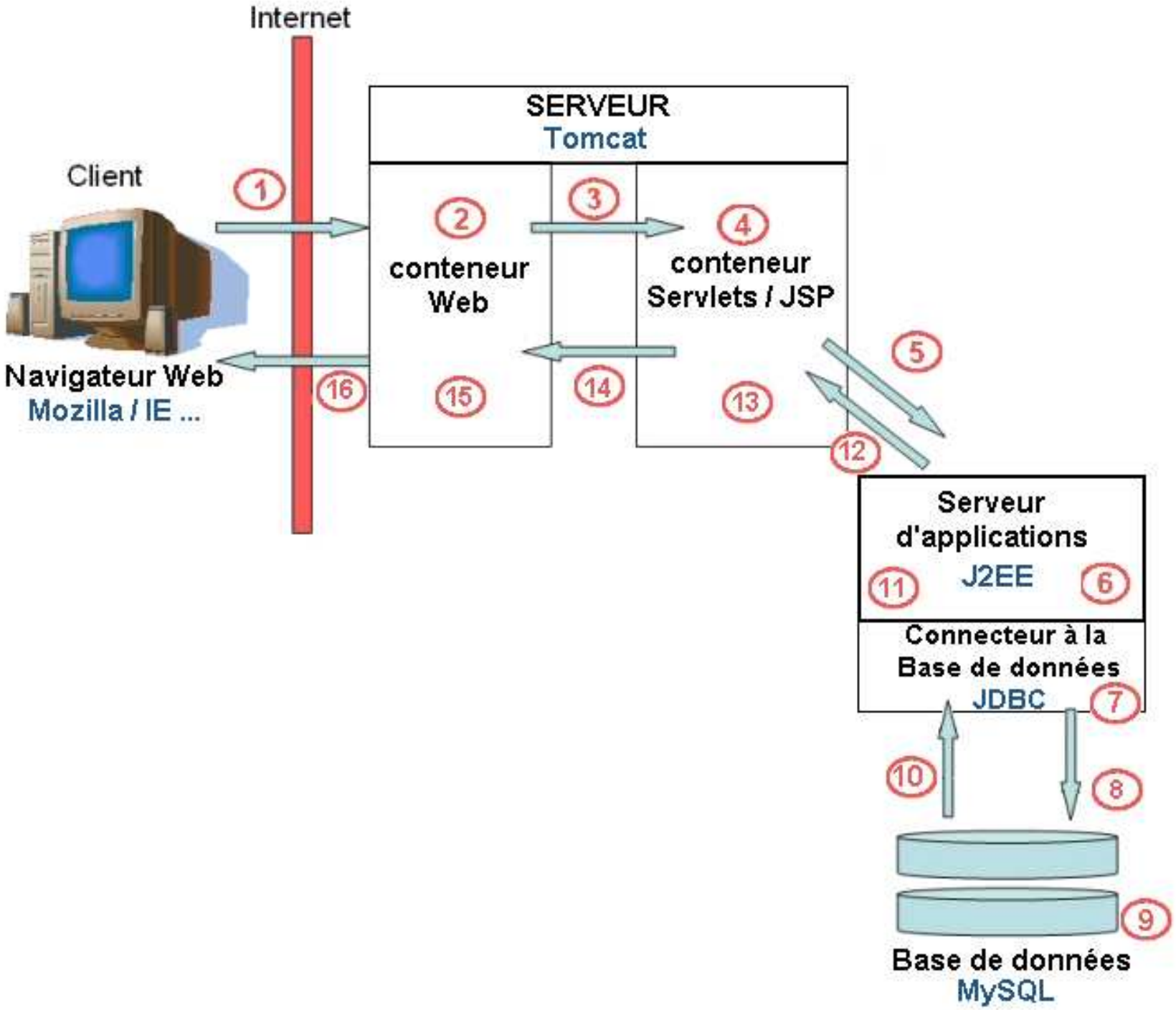
- ***JDBE :***

JDBC est l'acronyme de "**J**ava **D**ata **B**ase **C**onnectivite". Il s'agit tout simplement d'un ensemble de classes et d'interfaces, respectant une spécification et permettant de se connecter à n'importe quels types de bases de données en conservant le même code Java. Cette API est entièrement indépendante de votre système de base de donnée. Tant que vos requêtes sont compatibles vous pouvez sans problème réutiliser les mêmes méthodes sur une base Access et MySql par exemple.

- ***Mysql :***

Le logiciel MYSQL est un serveur de base de données (endroit ou l'on range plein d'informations) SQL très rapide, Multi-Threadé, Multi-Utilisateur et robuste. Le serveur MYSQL est destiné aux missions stratégiques et aux systèmes de production à forte charge, ainsi qu'à l'intégration dans des logiciels déployés à grande échelle.

MYSQL est à ce jour suffisamment élaboré pour être utilisé dans de très nombreux cas. Sa fiabilité est généralement excellente, sa disponibilité totale, et en plus il est gratuit. MYSQL permet de créer des tables contenant plusieurs milliards d'enregistrements.


Internet
SERVEUR
Tomcat
Client
1
2
conteneur
Web
3
4
conteneur
Servlets / JSP
5
Navigateur Web
Mozilla / IE ...
16
15
14
13
12
Serveur
d'applications
11
J2EE
6
Connecteur à la
Base de données
JDBC
7
10
8
9
Base de données
MySQL

1. Le navigateur Web transmet une requête demandant une ressource au serveur.
2. Le conteneur Web traitent les requêtes, qu'elles soient statiques ou dynamiques, mais ne peut répondre qu'aux requêtes statiques.
3. Transmission d'une requête dynamique.
4. Le conteneur de Servlets / JSP interprète leurs codes.
5. Le code java est transmis au serveur d'application.
6. Le serveur d'application interprète le code java.
7. Le JDBC permet d'établir la connexion entre le serveur d'application et la base de données.
8. La requête SQL est transmise à la base de données.
9. La base de donner exécute la requête
10. La réponse à la requête est transmise au serveur d'application via le connecteur JDBC.
11. Le serveur d'application intègre au code java interprété le résultat qu'il a reçut de la base de données.
12. Il transmet le tout au conteneur de Servlets.
13. Le conteneur de Servlets / JSP intègre ce qu'il a reçut du serveur d'application aux code qu'il interprète.
14. Il transmet le tout au serveur Web.
15. Le serveur Web se retrouve avec l'équivalent d'un requête statique qu'il peu donc traiter.
16. Le conteneur Web envoie la réponse finale au navigateur Web.

DEVELOPPEMENT DE LA WEB APPLICATION

Développer une web-application sous le modèle MVC , construite sur une architecture 4 tiers :

Un « tiers » : une couche logicielle de l'application

1er tiers: la VUE (l'Interface) 2 tiers: les Controllers 3 tiers: le Modèle (classes Métiers)

4 tiers: la Base de donnée

Pourquoi ?:

la dépendance entre les couches sont faibles

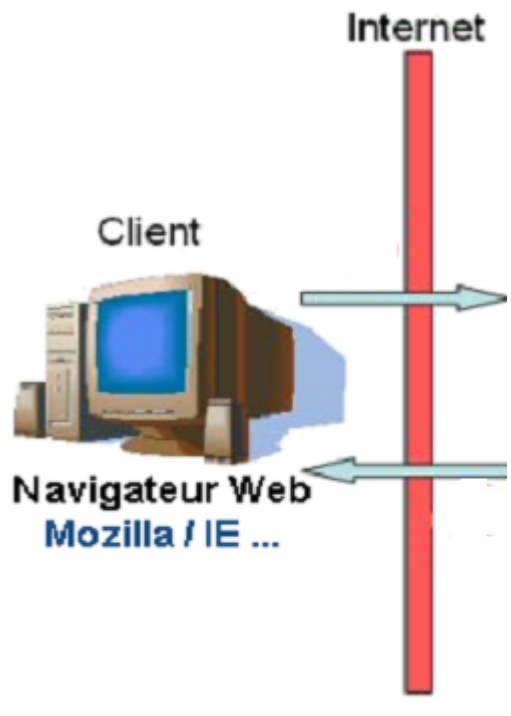
➔ les composants sont réutilisables, mieux construits,

➔ augmentation de la productivité

La démarche du développement de ce projet a nécessité 3 étapes

1. **L'analyse technique**
2. **L'analyse fonctionnelle**
3. **L'Implémentation**

Analyse Technique des Vues



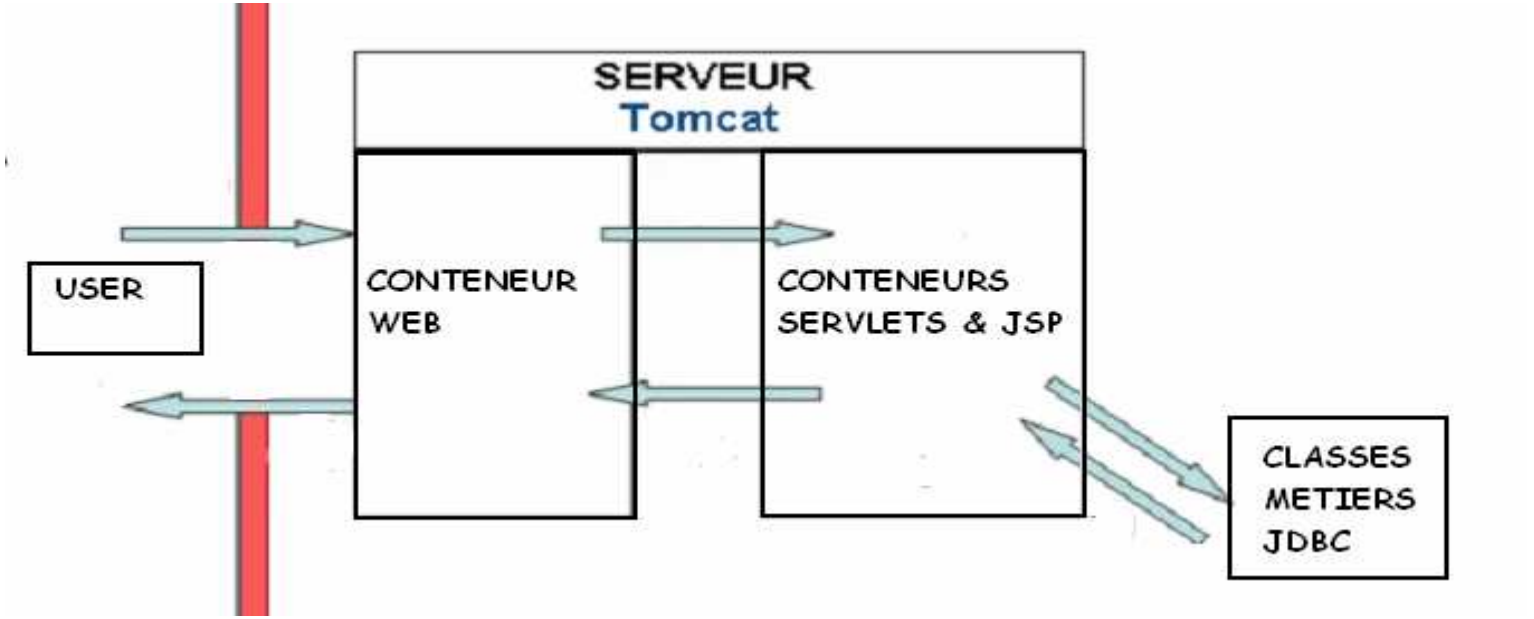
Créer des pages Webs

Créer l'Interface,la VUE

L'utilisateur doit savoir exactement comprendre quelles sont les actions qu'il peut entreprendre afin de satisfaire ses besoins !!!!

Le développement des Vues nécessite de connaître les codes HTML pour la structure, les CSS pour la présentation, le JavaScript ou Applets pour les scripts, et ceux de type SVG, Flash….(non utilisés dans le projet).

Analyse Technique des Controllers



Le développement des Controllers nécessite de connaître les fonctionnalités des 3 librairies TOMCAT (chacune des librairies sont compilées dans des conteneurs):

- jasper-runtime.jar source + class web tomcat (conteneur web)
- jsp-api.jar (JSP 2.0) source + class jsp **(conteneur JSP)**
- servlet-api.jar (Servlet 2.4) source + class servlets **(conteneur Servlet)**

Chaque requête d'un client exigeant une page JSP ou .class, instancie une servlet.class (donne naissance à la vie d'une servlet).

Une servlet.class est soit hébergée dans un conteneur Servlet soit dans celui des JSP

Analyse Technique des Controllers

a) Le développement d'une Servlet :

Toute servlet:

1. hérite des classes du conteneur jsp ou servlet (eux-mêmes héritant de la classe Objet)
2. fait appel aussi aux classes de ces conteneurs, si nécessaire aux classes Metiers et bien entendu au JRE

exemple :

```
import javax.servlet.*;  intégrant l'Interface Servlet
import javax.servlet.http.*; héritant des fonctions de l'Interface Servlet
import pack_metiers.XXX ;   appel aux classes metiers
import java.io.*;

public class login extends HttpServlet (classe du pack javax.servlet.http)
 {
Cycle de vie
}
```

Analyse Technique des Controllers

b) Le cycle d'une Servlet :

•La naissance d'un objet Servlet

L'instanciation déclenche la méthode init() *de l'Interface Servlet*

Elle peut donc être redéfinie : connexion réseau ou bdd

* init() est équivalente à un constructeur

Ex :

```
public void init() {
super.init(config) ;
//connexion réseau ou bdd}
```

La méthode init peut générer une exception de type **ServletException** ou **UnavailableException** (Pack javax.servlet). Une fois l'exception générée, le processus de chargement est abandonné et l'instance de la Servlet est immédiatement détruite.

Analyse Technique des Controllers

•Le traitement de la Servlet

Concernant les requettes http, il existe 2 types : GET et POST

GET : données provenant de l'URL POST : données provenant du corps

Selon le type, sera déclenché les méthodes doGET() ou doPost()

de type **HttpServlet** (service()*de l'Interface Servlet* est aussi utilisé en cas de doute)

Ex :

```
public void doGet(HttpServletRequest req, HttpServletResponse res)
throws ServletException, IOException
{   Traitement 1  (appel à des fonctions metiers, JDBC, coder HTML)        }
public void doPost(HttpServletRequest req, HttpServletResponse res)
throws ServletException, IOException
{   Traitement 2                                                          }
```

Analyse Technique des Controllers

Chaque méthode peut lever une exception grâce au marqueur « throws », et renverrait à l'utilisateur un code erreur 503

Elles sont surchargées par des objets de type :

- **HttpServletRequest** *(classe du pack javax.servlet.http)*

  possède tous les renseignements sur les paramètres passés à la requête

- **HttpServletResponse** *(classe du pack javax.servlet.http)*

  permet d'obtenir un flux de sortie pour communiquer la réponse au client

Elles peuvent acceder à plusieurs fonctions :

- récupération des variables saisies par le user getParameter("variable")
- redirection d'une page : sendRedirect("URL")
- génération de code HTML : getWriter() via la fonction println() de IO
- gestion des sessions

Analyse Technique des Controllers

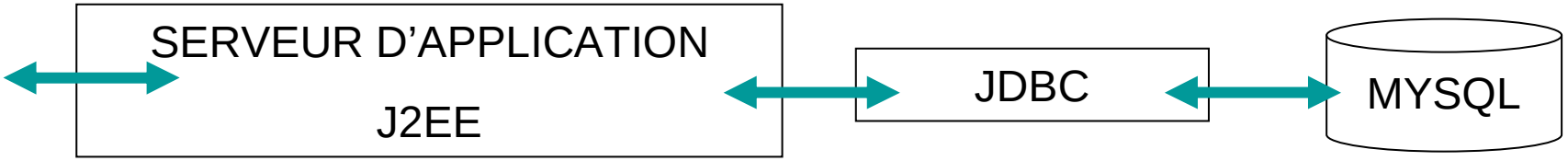
La mort d’un objet Servlet

Une fois la méthode destroy() appelée, le moteur de Servlets ne doit plus router les requêtes provenant des utilisateurs vers la Servlet concernée. Dans le cas où le moteur de servlets pense que l'instance de la servlet puisse être utile à nouveau, il doit créer une nouvelle instance de celle-ci et poursuivre la destruction de la précédente. Une fois que la fin du corps de la méthode destroy() est atteinte, le moteur de servlets doit détruire l'instance de la servlet et rendre l'objet éligible par le ramasse-miettes de la machine virtuelle.

c) le référencement d’une servlet

Seules les servlets du Conteneur servlet, sont à référencer (signature et mapping) dans le fichier de configuration **web.xml** (cf Partie 3 : Implémentation)

Analyse Technique des classes Métiers



Le développement des classes Metiers nécessite de connâitre le langage JAVA (sans oublier les méthodes des classes du Connecteur JDBC), et plus particulièrement certaines des techniques clés du POO :

- l'Héritage
- La Gestion des Exceptions

**notions utilisées dans l'analyse technique des Controllers*

Analyse Technique des classes Metiers

1) l'Héritage

Grâce à l'héritage, une classe peut disposer immédiatement de toutes les fonctionnalités d'une super-classe existante.

Les conséquences de l'Héritage : **Le Polymorphisme**

Le terme polymorphisme décrit la caractéristique d'un élément qui peut prendre plusieurs formes, comme l'eau qui se trouve à l'état solide, liquide ou gazeux.

En programmation Objet, le polymorphisme signifie que la même opération peut se comporter différemment sur différentes classes de la hiérarchie.

Analyse Technique des classes Metiers

*Extrait d'un code d'une méthode de **super-classe** :*

```
public class compte_courant
{
public int maj_compte(String post_compte,String trans,char sens)
{
/*************************************************************
cette méthode mets à jour le solde d'un compte courant
mettre à jour le compte XXXX du montant XX euros en crédit
*************************************************************/
}
}
```

Analyse Technique des classes Metiers

a) La redéfinition d'une methode d'une super classe :

Redéfinir, consiste à garder le même nom, avec les mêmes signatures (type de retour et types d'arguments), seul le corps de la fonction change.

```
public class compte_securise extends compte_courant
{
    public int maj_compte(String post_compte,String trans,char sens)
    {
        compte_courant secu= new compte_courant();
     // contrôle d'un retrait à 5%
        if(solden>=(1.05*transn))
            { secu.maj_compte(post_compte,trans,sens); return 1;} else return 0 ;
     }
}
```

Analyse Technique des classes Metiers

b) La surdéfinition d'une methode d'une super classe :

Sur définir, consiste à garder le même nom mais avec des signatures différentes

```
public class compte_remunere extends compte_courant
{
public int maj_compte(String post_compte,String trans,char sens,String solde)
        {
          compte_courant remu= new compte_courant();
      // execution d'un algorithme d'Interêt
      // condition X          return maj_compte(post_compte,trans,sens);
        }
}
```

Analyse Technique des classes Metiers

2) La Gestion des Exceptions :

Une exception est un signal qui indique qu'un événement anormal est survenu dans un programme

Java dispose d'un mécanisme très souple nommé gestion d'exception, qui permet à la fois :

•de dissocier la détection d'une anomalie de son traitement

•de séparer la gestion des anomalies du reste du code

On distingue plusieurs types d'erreurs :

•Les **Error** correspondent à des exceptions qu'il est rare d'attraper.

•Les **RuntimeException** que l'on peut rattraper mais que l'on n'est pas obligé.

•Les **Exception** que l'on est obligé d'attraper **(try{}/catch{})** ou de dire que la méthode appelante devra s'en occuper **(throw{}).**

Quoi qu'il arrive : use **finally{}**

Analyse Technique des classes Metiers

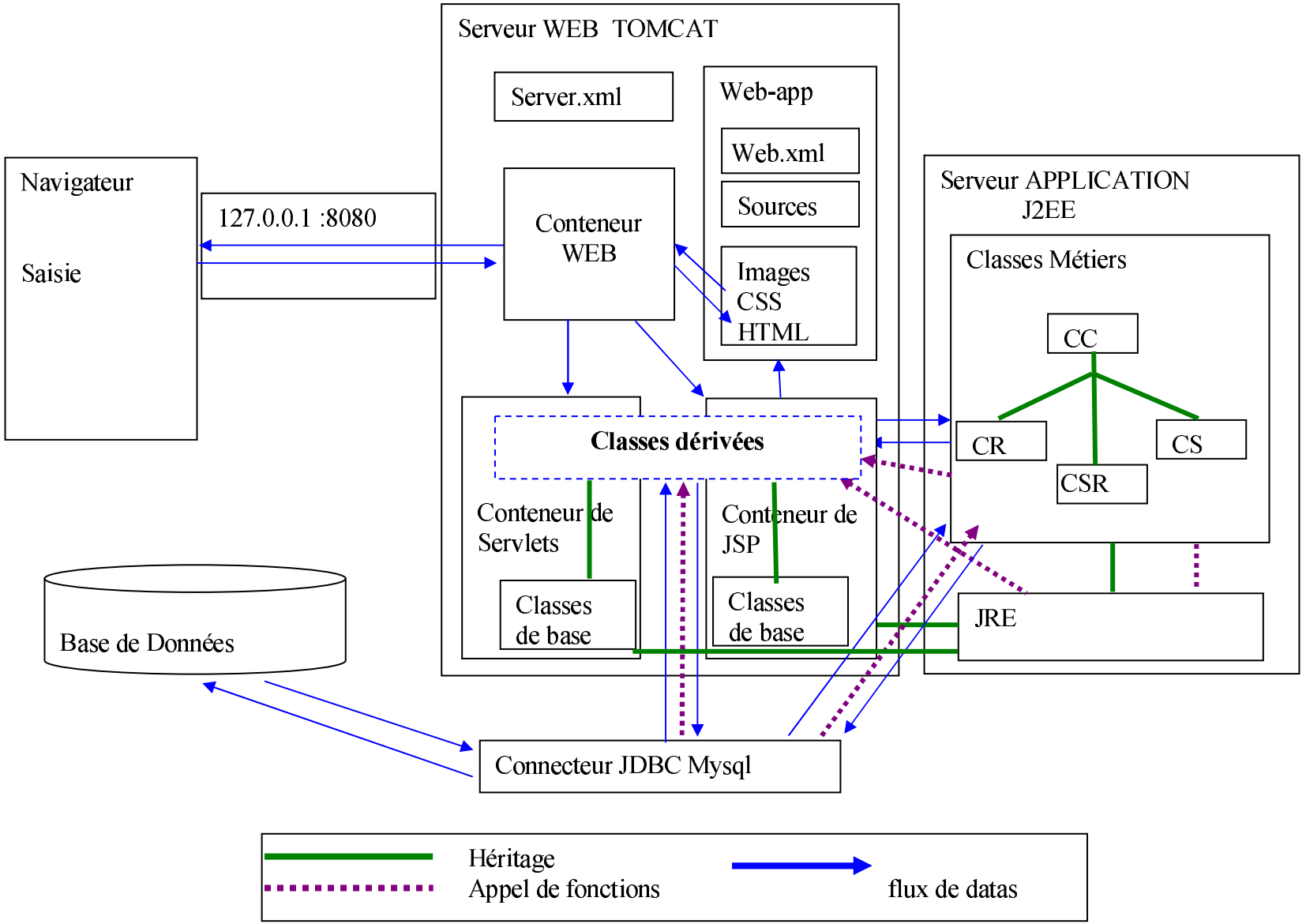
L'objet Servlet A tente de charger le driver JDBC pour Mysql

```
try      {Class.forName("com.mysql.jdbc.Driver");  }
Catch (ClassNotFoundException e)
        {System.out.println("Erreur au chargement du driver : " + e.toString()); }
```

L'objet Servlet A tente de se connecter et d'exécuter une requête SQL

```
try {  Statement stmt = null; ResultSet rs = null;
       Connection conn DriverManager.getConnection(“ // Connnection à la base");
       stmt = conn.createStatement();
       rs = stmt.executeQuery(“  //  requete SQL '");                    }
catch (SQLException ex)
{System.out.println("SQLException: "+ ex.getMessage()); //message: contient une description de l'erreur SQL
System.out.println("SQLState: " + ex.getSQLState()); // code défini par les normes X/Open et SQL99
System.out.println("VendorError: " + ex.getErrorCode()); // le code d'erreur du fournisseur du pilote }
```

ANALYSE FONCTIONNELLE



Cas n°1 Connexion au site:

VUE : Le client appel la page via une Url

http://www.bvn.net:8080/acceuil.jsp

au niveau réseau :

Entête Ethernet	Entête IP	EntêteTCP	EntêteHTTP	Datas
carte du serveur	ip bvn	tcp 8080	GET /accueil.jsp	

Controllers :

Analyse de TOMCAT :

- Autoriser la connexion via le socket 8080
- Charger la page JSP ➔ exécuter la servlet acceuil. class dans le Conteneur JSP.

Traitement de TOMCAT :

- Exécuter les instructions de la servlet accueil via Javaw.exe

Cas n°1

Instructions exécutées par Javaw.exe :

Instanciation de la servlet accueil

Action de la Méthode doGET : Génération d'une clé ID de session

Génération du code HTML

- Charger la feuille de style style.css
- Charger le logo.gif
- Création du message http

Code reponse :200	Set Cookie : JSESSION=XXXA Images CSS	<html>……</html>

VUE : le client reçoit sa page de connexion

Cas n°2 Demande de Session:

VUE : le client saisit son formulaire de connexion, il valide son identifiant et son mot de passe

Contollers :

Analyse de Tomcat :

Entête Ethernet	Entête IP	EntêteTCP	EntêteHTTP	Datas
carte du serveur	ip bvn	tcp 8080	POST /bancaire/login Cookie : JSESSIONID= XXXA	ID=C001Mdp=c1bvn

Traitement de TOMCAT :

- Lecture du cookie, et chargement du fichier de session XXXA
- Executer les instructions de la servlet login.class dans le Conteneur Servlet

Cas n°2

Instructions exécutées par Javaw.exe :

Instanciation de la servlet login, Chargement du driver JDBC

Action de la Méthode doPOST :

Récupération des datas VUE

Connexion à la base Mysql : JDBC tente de créer un objet Connexion

Si objet non créé : Génération d'une exception et traitement de celle çi :

Génération du code HTML : échec d'authentification

Création d'un Objet Statement qui exécute 1 requete SQL

Récupération d'un objet Resultset qui contient le tuple identifiant le User

Analyse du Resulset : Si objet contenant la valeur Cli :Gen Html : page client
Enregistrement de la variable user dans le fichier de session

Si objet contenant la valeur Emp : Gen Html : page agent
Enregistrement de la variable user dans le fichier de session

- Chargement images,css…
- Création du message http

Cas n°3 Consultation des comptes:

VUE : Le client choisit son environnement :

•Acces à un distributeur Acces à un guichet Acces à un service online

Controllers :

Analyse de Tomcat :

Entête Ethernet Entête IP EntêteTCP EntêteHTTP Datas

carte du serveur ip bvn tcp 8080 GET choix.jsp?service=gab Cookie : JSESSIONID= XXXA

Traitement de TOMCAT :

- Lecture du cookie, et chargement du fichier de session XXXA
- Exécuter les instructions de la servlet login.class dans le Conteneur JSP

  Instanciation de la servlet choix

  Action de la Méthode doGET :

  Récupération des datas VUE

  Exécution de la condition : si (service== « gab ») : Gen Html : page GAB

  Enregistrement de la variable service dans le fichier de session
- Chargement images,css…
- Création du message http

Cas n°3

VUE : Le client se trouvant le distributeur a le choix entre plusieurs fonctions :

•Créditer son compte X Débiter son compte X Consulter ses comptes

Controllers :

Analyse de Tomcat :

Entête EthernetEntête IPEntêteTCPEntêteHTTPDatas

carte du serveur ip bvn tcp 8080 GET /bancaire/consulter Cookie : JSESSIONID= XXXA

Traitement 1/2 de TOMCAT :

- Lecture du cookie, et chargement du fichier de session XXXA
- Exécuter les instructions de la servlet consulter.class dans le ConteneurServlet

Instructions exécutées par Javaw.exe :

Instanciation de la servlet consulter

Action de la Méthode doGET :

Création d'un objet clientExecution de la fonction consultation des comptes pour un Client

Métiers :

Traitement du Serveur Applicatif :

Instructions exécutées par Javaw.exe :

Dés la naissance de l'objet Client, son constructeur charge JDBC et la connexion

Création d'un Objet Statement qui exécute 1 requête SQL

Récupération d'un objet Resultset qui contient les tuples de comptes du client

Récupération tuples par tuples, des valeurs du compte,du solde…

Renvoie d'un tableau 2 dimension de String contenant les informations des comptes du Client.

Traitement 2/2 de TOMCAT :

Récupération d'un tableau 2 dimensions contenant les comptes

Génération du code HTML : page contenant des tableaux récapitulatifs des comptes du client C001.

- Chargement images,css…
- Création du message http

Remarque : A chaque fois que le client décide d'acceder aux fonctions suivantes :

- Consulter ses comptes
- Consulter les opérations d'un de ces comptes
- Effectuer un transfert de type interne-interne ou interne-externe
- Effectuer un retrait
- Effectuer un dépôt

Toute servlet fera appel aux classes métiers hébergeant ces fonctions.

IMPLEMENTATION

1) Installations & Configurations

Pré-requis :

Vérifier les installations déjà présentes sur votre poste afin de ne pas perturber l'implémentation de notre environnement du projet: (J2EE,mysql….)

Vérifier que le protocole TCP/IP est actif.

Vérifier que votre firewall autorise les requêtes vers votre local host.

a) Installation de l'environnement Java de base :

· Installation du pack J2SE Runtime Environnent + JDK 1.5

· Vérifier que Java est fonctionnel:

Compilation du code source java → javac MaClasse.java

Interprétation du byte code → java MaClasse.class

(jawaw est aussi un interpréteur mais sans ouverture de console dos)

IMPLEMENTATION

b) Installation d'un IDE : EclipseV3

· Indiquer le chemin du Jre 1.5 afin d'une part, d'hériter ou d'appeler des classes des librairies de base, et d'autre part de donner l'accès à l'executable javaw.exe

· Vérifier l'opérationnalité en créant un projet puis une classe java (tester le fameux « Hello Word ») contenant la méthode main()

c) Installation d'une base de donnée : MySQL Servers and Clients 4.0.21

· Activer le serveur Mysql : **mysqld-nt –install**

d) Installation d'un serveur Web : Tomcat 5.5.4

· Indiquer le chemin du Jre 1.5 afin d'une part, d'hériter des librairies de base, et d'autre part de donner l'accès à l'exécutable javaw.exe

IMPLEMENTATION

e) Télécharger le connecteur mysql pour Java (cf l'arborescence du projet pour son emplacement) : mysql-connector-java-3.0.16-ga-bin.jar

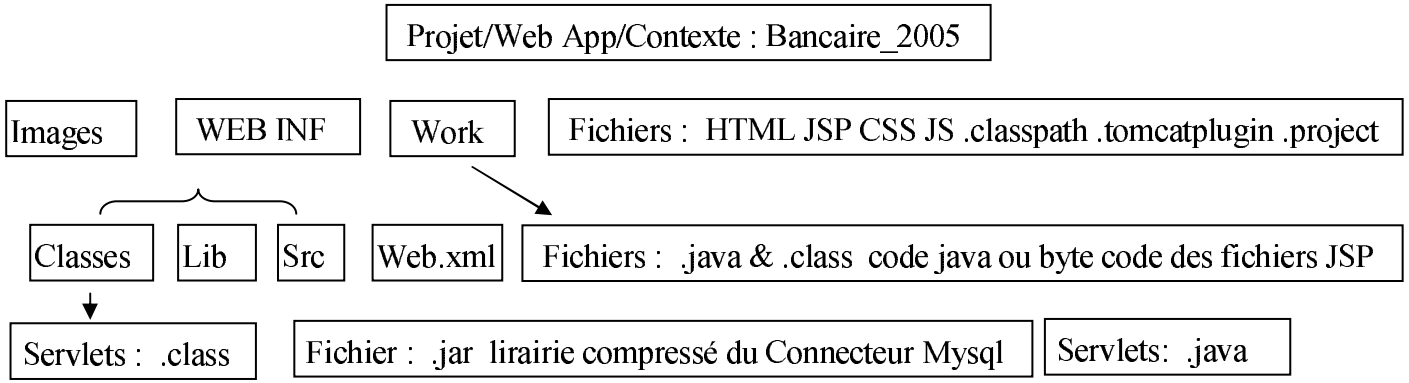
f) Télécharger le plug-in Tomcat pour Eclipse à mettre dans le repertoire …\eclipse\plugins\ : com.sysdeo.eclipse.tomcat_3.0.0

· Indiquer le chemin de Tomcat (permettre à Eclipse d'executer 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)

IMPLEMENTATION

2) Implementation de notre application web

a) Création d'un projet et de son arborescence



b) Test opérationnel du serveur TOMCAT

- Activer le serveur Tomcat dans Eclipse
- Tapez l'Url : http://localhost:8080/ et exécuter les exemples JSP & Servlets

IMPLEMENTATION

c) Test opérationnel des servlets et Jsp du Projet

· Mettre à jour le fichier web.xml à chaque développement d'une servlet :

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE web-app PUBLIC "-//Sun Microsystems, Inc.//DTD Web
Application 2.3//EN" "http://java.sun.com/dtd/web-app_2_3.dtd">
<web-app>
      <display-name>nom_du_Projet</display-name>
      <servlet> // On reference la servlet
            <servlet-name>ma_servlet</servlet-name>
            <servlet-class>mon_package.ma_servlet</servlet-class>
      </servlet>
      <servlet-mapping> // On attribute un chemin Url à notre servlet
            <servlet-name>ma_servlet</servlet-name>
            <url-pattern>/ma_servlet</url-pattern>
      </servlet-mapping>
</web-app>
A Chaque lancement de Tomcat, ce dernier le le charge
```

· Tapez :

Création d'un projet java tomcat (mise à jour auto du fichier xml)

test d'une page JSP : http://localhost:8080/bancaire_2005/acceuil.jsp

test d'une servlet : http://localhost:8080/bancaire_2005/ma_servlet

IMPLEMENTATION

e) Test opérationnel du pilote JDBC (*permet de transmettre des instructions SQL via Java)*

· Configuration du serveur Mysql :

Sous root, créer une base Bancaire :

mysql –u root mysql>create database Bancaire mysql>show databases;

Créer une classe java de test, contenant une levée d'exception afin de s'assurer du bon fonctionnement du pilote JDBC pour Mysql :

g) Créer votre base de donnée à l'aide des schémas de conception

h) Administrer votre base de donnée

· Création d'utilisateurs et de leurs droits sur la base bancaire :

Mysql>grant (privilèges) on bancaire.tableX to 'userX'@'localhost' identified by'mdp' ;