REPUBLIQUE DU BENIN
****

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR
ET DE LA RECHERCHE SCIENTIFIQUE
******

UNIVERSITE D'ABOMEY–CALAVI
*****

ECOLE NATIONALE D'ECONOMIE APPLIQUEE ET DE MANAGEMENT (ENEAM)

MEMOIRE DE FIN DE FORMATION DU CYCLE II

POUR L'OBTENTION DU DIPLOME DE MASTER

OPTION

Informatique de Gestion

Thème :

Mise en place d'une plate-forme mobile pour les Institutions de Microfinance

Réalisé et présenté par :

Ercias LOHOUNME

Sous la direction de:

Dr Théophile K. DAGBA
Professeur ENEAM/UAC

1ère Promotion : 2007-2009

Dédicace

A mes enfants, ce travail n'est qu'une partie de ce qui vous attend, c'est un impérial devoir pour vous de faire mieux.

Remerciements

« Soyons reconnaissants aux personnes qui nous donnent du bonheur ; elles sont les charmants jardiniers par qui nos âmes sont fleuries ».
Marcel Proust (1871-1922).

Je remercie Monsieur Théophile K. DAGBA, mon directeur de mémoire, pour avoir encadré mes travaux de recherche, et m'avoir soutenu jusqu'à la fin. Il n'a ménagé aucun effort pour que les travaux soient menés à bien.

Je remercie également la direction générale de l'Association pour la Promotion et l'Appui au Développement des Micro-Entreprises (PADME) pour son soutien.

Un grand merci à tous les camarades de l'ENEAM, à mes amis Ghislain, Azizou pour leur soutien.

Mes derniers remerciements qui ne sont pas les moindres sont pour les personnes les plus chères dans ma vie, ma femme Ahouéfa, mes parents et en particulier Olivier GAHOU qui ont tout sacrifié pour moi et qui m'ont toujours soutenu dans tout ce que j'entreprenais. Ce travail est le fruit de leurs patience et sacrifices.

Merci !

Avant-propos

Le travail présenté dans ce mémoire a été réalisé dans le cadre de notre projet de fin d'études de MASTER en Informatique de Gestion à l'Ecole Nationale d'Economie Appliquée et de Management (ENEAM).

En effet, les étudiants en cycle d'ingénieur de l'ENEAM sont tenus d'effectuer un stage afin de mettre en pratique les connaissances académiques. Ainsi, au cours de notre stage nous avons travaillé sur le thème « **Mise en place d'une plate-forme mobile pour les Institutions de Microfinance** » qui sera développé dans ce document.

Liste des acronymes

SIGLES	SIGNIFICATION
.NET	.NET (dit DotNet) est un ensemble de langages de programmation proposé par MicroSoft.
API	Application Programming Interface
CDC	Connected Device Configuration
CDMA	Code Division Multiple Access
CLDC	Connected Limited Device Configuration
COM	Component Object Model
CORBA	Common Object Request Broker Architecture
CSD	Circuit Switched Data
DCS	Digital Cellular System
DSP	Data Service Provider
EDGE	Enhanced Data for GSM Evolution
GPRS	General Packet Radio Services
GSM	Global System for Mobile
HTML	Hypertext Markup Language
HTTP	Hyper Text Transfer Protocol
IDE	Environnement de Développement Intégré
IMF	Institution de Micro-Finance
J2ME	Java 2 Micro Edtion
JCP	Java Community Process
JVM	Java Virtual Machine
MIDP	Mobile Information Device Profile
MVC	Model-View-Controller
NETBOOKS	Un netbook est un ordinateur de très petite taille, aux performances plus faibles qu'un ultraportable classique
OASIS	Organization for the Advancement of Structured Information Standards
ORB	Object Request Broker
OS	Operating System
PADME	Association pour la Promotion et l'Appui au Développement des Micro-Entreprises
PDA	Personal Digital Assistant ou Agenda électronique

REST	Representational State Transfer
SIGLES	**SIGNIFICATION**
SGBD	Système de Gestion de Bases de Données
SGBDR	Système de Gestion de Bases de Données Relationnelles
SIM	Subscriber Identity Module
SMS	Short Message Service
SOA	Servive Oriented Architecture
SOAP	Simple Object Access Protocol
TDMA	Time Division Multiple Access
TD-SCDMA	Time Division Synchronous CDMA
UDDI	Universal Description Discovery and Integration
UML	Unified Modeling Language
UMTS	Universal Mobile Telecommunications
URI	Universal Resource Identifier
URL	Uniform Resource Locator
W3C	World Wide Web Consortium
WAP	Wireless Application Protocol
W-CDMA	Wide Code Division Multiple Access
WS	Web Servives
WSDL	Web Service Description Language
WSOA	Web Services Oriented Architecture
XML	eXtensible Markup Language

Liste des figures

Table des matières

Résumé

L'informatique mobile offre aux utilisateurs un accès permanent et transparent à l'information et à des services contextuels en tout lieu et dans toutes les situations. Grâce à cette technologie, il est possible à un utilisateur de recevoir une demande d'achat par SMS, se rendre sur une application permettant de vérifier les stocks, confirmer la commande par un mail et recevoir la confirmation par un message vocal ; etc.

L'objectif principal que vise ce projet est de faire profiter cette technologie aux institutions de microfinance qui, malgré leur ferme volonté de s'étendre vers les zones rurales, sont limitées par l'isolement géographique et par le niveau élevé des coûts des transactions.

Pour cela, une solution applicative a été proposée pour l'accès à certaines fonctionnalités de l'application de gestion de crédits via Smartphone. Cette solution prend appui sur une architecture orientée service dans laquelle les principales fonctionnalités sont développées sous forme de Web services. L'appel des méthodes de ces Web services se fera à partir du Smartphone tandis que leur exécution s'effectuera sur un serveur qui retournera un résultat compréhensible grâce au fichier WSDL (Web Service Description Language) généré par les Web services et dont le transport est assuré par le protocole SOAP (Simple Object Access Protocol).

Pour l'élaboration de cet outil, quelques technologies disponibles sont présentées dans la première partie tandis que la seconde partie traite de sa conception et de sa mise en œuvre.

Abstract

Mobile computing offers a permanent and transparent access to information and services anywhere and in all situations. With this technology, it is possible for a user to receive a purchase request via SMS, go to an application for checking inventory, confirm the order and receive an email confirmation with a voice message, etc.

The main objective of this project is to make this technology benefit to microfinance institutions which, despite their strong desire to expand into rural areas, are limited to geographical isolation and the high transaction costs.

For this, a solution has been proposed to access some features of the enterprise application via Smartphone. This solution is based on a service-oriented architecture in which the main features are developed as web services. The invocation of the methods is performed from a Smartphone or a PC, while the execution will be on a server that returns an understandable result through WSDL (Web Service Description Language) generated by web services, where the transport is provided by SOAP (Simple Object Access Protocol)

By developing this tool, some available technologies are presented in the first part while the second part deals with the design and implementation.

Introduction

L'évolution technologique dans le domaine des communications sans fil a rendu possibles de nouvelles applications professionnelles qui offrent aux utilisateurs l'accès à l'information à n'importe quel moment et à n'importe quel endroit. Ainsi, une autre grande révolution a succédé à l'explosion de la téléphonie cellulaire : l'informatique mobile.

Grâce à ce nouveau paradigme, un utilisateur peut par exemple recevoir une demande d'achat par SMS, se rendre sur une application permettant de vérifier les stocks, confirmer la commande par un mail et recevoir la confirmation par un message vocal ; etc. Ces capacités d'interactions et de rapidité sont inhérentes au caractère principalement nomade de l'Internet et en particulier à la performance évolutive du mobile.

Mais en Afrique, la généralisation des Technologies de l'Information et de la Communication (TIC) dans les divers domaines d'activités n'est pas encore effective. La faible consommation de cet outil s'observe aussi au niveau des Institutions de Microfinance (IMF) pour qui il serait d'une grande utilité compte tenu de leur vision.

En effet, les IMF naissent généralement dans les grandes villes et ont dans leur majorité pour souci de s'étendre aux zones rurales, mais cet effort d'extension est souvent entravé par l'isolement géographique et par le niveau élevé des coûts des transactions, deux préoccupations majeures que, dans d'autres domaines, les technologies d'information ont toujours réussi à relever [WEB01].

Comment permettre de ce fait aux IMF de s'étendre aux zones rurales et reculées à moindre coût sans contrainte de technologie et ou de plates-formes ?

C'est à cette interrogation que nous tentons d'apporter une réponse dans le cadre de ce mémoire de fin d'études. Nous nous proposons à cet effet d'apporter une solution à partir de la richesse applicative et de la force novatrice du domaine multimédia mobile et de l'Internet.

Le projet a donc pour but le développement d'une application métier qui va permettre l'accès à certaines fonctionnalités de l'application de gestion de crédits via Smartphone pour une meilleure mobilité des agents nomades.

Le mémoire est structuré comme suit :

Dans la première partie, après avoir rappelé la problématique et les objectifs de ce sujet du projet de fin d'études, nous allons jeter un regard sur l'informatique mobile sans oublier les modèles d'architecture ainsi que les technologies et outils de son développement.

La deuxième partie traitera de la conception et de la réalisation du système.

1ère Partie : Généralités

« La principale fonction de l'art est de construire des types sur la base fournie par la science ».
Auguste Comte.

Chapitre 1 : Contexte général

Ce chapitre présente la problématique et les objectifs du sujet et aussi quelques concepts relatifs à l'informatique mobile.

I- Problématique et objectifs

A- Problématique

Les institutions de microfinance dans les pays en développement sont confrontées à deux grands défis dans leurs efforts pour atteindre les communautés rurales: les coûts de transaction et les risques. Les coûts élevés de transaction sont dus au faible volume de transaction, à la faible qualité ou à l'inadéquation des infrastructures de télécommunication dans certains pays. Quant aux risques, ils sont systémiques, comme les fléaux naturels tels que la sécheresse ou les inondations ou les prix fluctuants des produits de base sur les marchés locaux, régionaux et mondiaux.

Toutefois, de nos jours, l'évolution des TIC en l'occurrence l'informatique mobile permettrait de surmonter ces obstacles en favorisant les activités professionnelles nomades réduisant ainsi les coûts élevés de transaction.

Mais, les coûts d'acquisition, du déploiement et de la maintenance des solutions de mobilité proposées par les concepteurs ne sont facilement pas à la portée des petites et moyennes entreprises.

Comment rendre cette solution accessible aux institutions de microfinance à moindre coût et sans contrainte de technologie et ou de plates-formes? Il apparaît absolument nécessaire que les compétences locales apportent leur solution à travers l'informatique mobile.

B- Objectifs

Dans le cadre de ce mémoire, nous avons réalisé un prototype de plate-forme mobile de gestion des crédits pour les institutions de microfinance dont l'objectif principal est d'optimiser les traitements et se libérer des contraintes géographiques.

Cet outil sera utilisé pour gérer les opérations de crédits via téléphones mobiles dotés des services Internet. Il s'agira de mettre en place une application client-serveur dont un client mobile et un client Web qui consommeront les services Web conformément à l'architecture qui sera proposée.

Notre plate-forme a pour objectifs spécifiques de contribuer à:

- ✓ l'amélioration de l'efficacité, de la fiabilité et de la rapidité des opérations de crédit ;
- ✓ la facilitation de la collaboration entre bureaux distants ;
- ✓ la mise à disposition des informations en temps réel pour les prises de décision rapides ;
- ✓ l'amélioration du service à la clientèle ;
- ✓ la réduction du niveau de risque.

II- Regard sur l'informatique mobile

A- L'informatique mobile

1- Définition

L'informatique mobile traite de l'ensemble des solutions informatiques développées sur des plates-formes comme les téléphones portables, les Smartphones, les « netbooks », les tablettes, etc. En situation de nomadisme, l'informatique mobile permet à un utilisateur de conserver une partie de ces outils numériques tout en lui apportant de nouveaux services. Elle lui permet aussi d'interagir avec son environnement via de nombreux canaux comme le sans-contact, les technologies sans-fil, etc. [WEB02]

2- Classification des applications mobiles

Plusieurs variétés d'applications ayant trait à la mobilité ont été développées. La classification qui a paru la plus pertinente utilise comme critère discriminant le fait

que la mobilité soit liée à l'utilisateur, au serveur de données ou aux données elles-mêmes. Ainsi, les applications mobiles sont classifiées comme suit [WEB03] :

- Client mobile – serveur fixe : dans ce groupe se trouvent les applications client/serveur traditionnelles dans lesquelles certains utilisateurs sont mobiles. Il peut s'agir d'applications professionnelles dans lesquelles des employés itinérants doivent rester en contact avec le système d'information de l'entreprise *(exemples : voyageur de commerce, expert d'assurance, etc.)*. Il peut également s'agir d'applications personnelles dans lesquelles un utilisateur fait appel aux services d'un hébergeur de données ou DSP (Database Service Provider) qui va lui garantir un stockage sûr et un accès permanent à ses données privées *(données bancaires, agenda, dossier médical, etc.)*. On y rencontre également des applications effectuant de la dissémination d'information à destination d'un large public d'utilisateurs mobiles *(diffusion d'informations météo, trafic routier, cours de la bourse, etc.)*.

- Client mobile – serveur mobile : cette classe couvre les applications dans lesquelles client, serveur et données sont tous embarqués dans un calculateur mobile. Les dossiers portables de différentes natures sont des exemples typiques de cette classe d'application. Selon le terminal employé, ces dossiers portables peuvent devenir fortement sécurisés *(exemple : dossier médical sur carte à puce ou carnet d'adresses dans une carte SIM)*. La copie primaire des données est ici gérée par le serveur mobile même si certaines données peuvent être répliquées sur le réseau fixe pour des raisons de tolérance aux défaillances. Bien que les applications visées soient majoritairement personnelles, et donc mono-utilisateurs, on peut envisager des traitements coopératifs entre serveurs mobiles (exemple : prise de rendez-vous sur un ensemble d'agendas embarqués dans des assistants personnels).

- Client fixe – serveur mobile : on peut ranger dans cette classe des applications où des capteurs attachés à des mobiles diffusent des informations sur

leur état à destination d'une unité de traitement fixe et reçoivent éventuellement des commandes émanant de cette unité *(ex : contrôle à distance d'un véhicule)*. On peut également imaginer des applications statistiques réalisant de l'extraction de données détenues par plusieurs serveurs mobiles *(exemple : étude épidémiologique sur une large population de dossiers médicaux embarqués sur des cartes à puce).*

- Données mobiles dans un serveur fixe : par abus de langage, le terme données mobiles est utilisé pour caractériser les applications dans lesquelles les données gérées correspondent à la localisation de mobiles. De telles bases de données de localisation peuvent être exploitées dans des domaines aussi variés que la localisation de téléphones cellulaires par les opérateurs télécoms, la gestion de flottes de véhicules.

B- Les réseaux téléphoniques et les terminaux mobiles

1- La téléphonie mobile et les normes de communication

Les réseaux téléphoniques sont en permanente mutation, plusieurs types de réseau se sont succédés dans le monde de la téléphonie, chacun apportant des améliorations dans la qualité de service par rapport à son prédécesseur.
Dans cette partie, il s'agira pour nous de faire un bref aperçu des principaux réseaux qui ont marqué la téléphonie [WEB17].

- **Le CSD (*Circuit Switched Data*)**

La première technologie, CSD (Circuit Switched Data), consiste à établir une connexion à distance avec un ordinateur pendant toute la durée de la communication. Elle n'est plus utilisée aujourd'hui car elle était très chère et lente, avec une vitesse maximale de 9,6 kb/s juste suffisante pour vérifier ses courriels occasionnellement, et encore à condition qu'ils ne contiennent que du texte.

- **Le GSM (*Global System for Mobile*)**

Le GSM correspond à la norme actuelle des téléphones mobiles créée début 1990. La technologie de transmission est TDMA (Time Division Multiple Access). A l'origine, le GSM utilisait uniquement la bande de fréquence de 900 MHz. Elle a été étendue à celle de 1800 MHz et à la norme d'origine britannique DCS 1800 (Digital Cellular System). La famille GSM inclut le GPRS (General Packet Radio Services), EDGE (Enhanced Data for GSM Evolution) et le 3GSM. On parle aussi de 2G (téléphonie mobile de 2ème Génération) qui correspond au passage de la transmission analogique (les gros téléphones embarqués dans les voitures) à la transmission numérique (et à l'apparition des téléphones portables).

- **CDMA**

Code Division Multiple Access (CDMA) est une technologie de multiplexage des transmissions par codage d'informations numérisées dite à étalement de spectre. Elle est utilisée aux Etats-Unis et en Asie dans le standard du même nom.

- **TDMA**

Le Time Division Multiple Access (TDMA), AMRT (Accès Multiple à Répartition dans le Temps) en français est la technologie utilisée en Europe pour le GSM. Chaque canal radio est découpé en 8 tranches de temps de 0,57 ms. Les fréquences utilisées sont 900 et 1800 MHz. La parole est comprimée sur une bande de 22,8 KHz qui inclut un codage permettant la correction d'erreurs.

- **W-CDMA**

Le W-CDMA (Wide Code Division Multiple Access, Multiplexage par code large bande en français) est une variante de CDMA utilisée pour la téléphonie mobile de troisième génération. La norme 3G Européenne UMTS est basée sur cette technologie.

- **TD-SCDMA**

Le TD-SCDMA *(Time Division Synchronous CDMA)* est la technologie développée par les Chinois. Plus récente que CDMA et W-CDMA, elle possède un intérêt en performance et est poussée par le gouvernement chinois sur son immense marché.

- **WAP**

Le WAP (*Wireless Application Protocol*) est un protocole défini pour afficher des pages Web sur un téléphone portable. Le WAP possède 5 couches protocolaires d'une manière semblable à l'OSI (*Open Systems Interconnection*).

- **GPRS, 2,5G**

La technologie GPRS *(General Packet Radio Service)* est souvent appelée 2,5G du fait que c'est une technologie à mi-chemin entre le GSM (la deuxième génération, venant après la transmission analogique) et l'UMTS (3ème génération). L'avantage du GPRS est qu'il est disponible partout où le réseau de l'opérateur est accessible. La fiche technique d'un Smartphone indique la « classe » du GPRS qui détermine la vitesse maximale. La plupart des téléphones fonctionnent en classe 10, permettant une vitesse jusqu'à 80 kb/s en voie descendante et 40 kb/s en voie montante. On peut aussi trouver des téléphones ou des cartes pour ordinateur portable en classe 12. En classe 2 on n'aurait qu'une douzaine de kb/s.

- **EDGE, 2,75G**

EDGE (*Enhanced Data Rate for GSM Evolution*), présentée comme la génération 2,75G, est une amélioration technologique du GPRS. Comme pour le GPRS, il existe douze classes de vitesse. La classe 10 peut atteindre, en théorie, une vitesse de 384 kb/s dans le meilleur des cas, ce qui est comparable à la vitesse basse de l'UMTS. L'avantage principal de cette technologie réside dans sa couverture, qui est plus simple à déployer et donc nettement plus étendue. Avec de tels débits de données, l'accès à Internet et aux échanges de données est parfaitement exploitable.

- **UMTS, 3G**

La technologie UMTS *(Universal Mobile Telecommunications System ou 3G)* transforme le téléphone en porte d'accès à Internet et, plus globalement, sert de support aux nouveaux modes de communication. Ainsi, en zone de couverture 3G, équipé du matériel adéquat, il est possible d'envoyer et de recevoir des données à haute vitesse (plus de 384 kbit/s). De tels débits permettent avec un téléphone compatible 3G d'accéder à la visiophonie, de recevoir la télévision numérique ou encore de naviguer sur Internet. Plus encore, le réseau UMTS permet aux ordinateurs portables (via un téléphone portable ou une carte PCMCIA) d'avoir un accès total à Internet (sans restriction technologique et factuelle telle qu'un petit écran ou un navigateur trop sommaire) de la même façon que par une ligne ADSL *(Asymmetric Digital Subscriber Line)*. Cependant, la couverture UMTS est loin d'être complète.

2- Les terminaux mobiles et leur sécurité

a- Terminaux mobiles

Le développement sans cesse croissant de l'informatique et de l'électronique a favorisé l'émergence de la miniaturisation et une multiplication de terminaux dits portables. Fondamentalement, les terminaux sont classés portables s'ils peuvent fonctionner en mobilité tout en interagissant avec un backbone réseau. Au nombre de ces terminaux mobiles on peut citer quelques uns adaptés aux besoins professionnels [WEB18]:

- **PDA** *(Personal Data Assistant)*

Un assistant numérique personnel - ou PDA - est à l'origine un agenda électronique destiné à la prise de rendez-vous, à la planification des tâches et au transport de données personnelles. Il s'est depuis, beaucoup enrichi en fonctionnalités communicantes (push mail, carte 3G, connexions Bluetooth, etc.).

- **Smartphone**

Véritable croisement entre téléphone mobile et assistant personnel (PDA), le Smartphone emprunte les qualités et les défauts de ceux-ci. Comme un PDA le Smartphone permet de gérer les contacts et rendez-vous, de naviguer sur le Web, etc. Mais il possède aussi les fonctionnalités d'un téléphone mobile: téléphoner, envoyer des SMS, prendre des photos, etc. Avec un abonnement GPRS, le Smartphone offre l'accès à la connexion Internet. Tout comme le PDA, on peut consulter les e-mails et surfer sur le Web.

- **Tablet PC**

Le Tablet PC garde une forte identité "ordinateur", par le biais d'un écran plus large et tactile. Le clavier est presque inexistant puisque la plupart des commandes s'effectuent sur l'écran. Il est largement utilisé pour des démonstrations clients ou le recueil de signatures électroniques chez les transporteurs.

Dans notre contexte, l'équipement terminal est l'équipement dont dispose le personnel mobile pour effectuer ses opérations d'interrogation, de consultation et ou de modification des données du système de l'information de l'entreprise. Trois grandes familles d'équipement sont à distinguer :

- Les postes de travail (postes informatiques)
- Les assistants personnels numériques(PDA)
- Les téléphones mobiles évolués (téléphones WAP ou intelligents)

b- Sécurité des terminaux mobiles

L'équipement terminal constitue un des maillons faibles de la chaîne de la liaison de par ses caractéristiques [WEB04]:

- ✓ sa mobilité accroit le risque de vol, de détérioration et de divulgation ;
- ✓ la multiplicité des moyens de raccordement (Ethernet, ondes radio, modem, etc.) et la potentialité de leur simultanéité augmentent les risques d'accès ou d'écoute illicites et de rebond ;

- ✓ ses capacités de stockage (conservation d'informations issues du système d'information de l'entreprise ou produites par l'utilisateur) accroissent l'impact d'un vol ou d'accès illicites ;
- ✓ son caractère potentiellement multi-utilisateur (partage au sein de l'entreprise) augmente la probabilité d'altération ou de divulgation de données.

La vulnérabilité la plus importante réside, en cas de vol, dans les données embarquées en l'occurrence les PDA et les Smartphones. Pour y remédier, il faut [WEB04]:

- ✓ protéger le terminal et son contenu : chiffrement des données sensibles, verrouillage par mot de passe à l'arrêt, etc.;
- ✓ améliorer le processus de synchronisation : utilisation de plate-forme de synchronisation assurant l'authentification ;
- ✓ recourir à des applications en mode connecté (sur client léger) : lors des conceptions d'applications internes, privilégier les applications en mode connecté (sur client WAP ou navigateur HTML) aux applications client/serveur.

CONCLUSION

Dans cette partie, nous avons vu les objectifs du projet et ce qu'offre l'informatique mobile.

Nous allons à présent nous intéresser aux modèles d'architectures et outils de développement des applications mobiles et web.

Chapitre 2 : Architectures et outils de développement des applications mobiles et Web

Ce chapitre fait la présentation des modèles d'architecture logicielle ainsi que certains outils de développement des applications mobiles et web.

I- Les modèles d'architecture logicielle

Avant de faire le choix du modèle d'architecture à implémenter, il serait intéressant de faire un tour d'horizon des solutions existantes.

A- Le modèle client/serveur

L'architecture client/serveur est un mode de communication entre des ordinateurs et des logiciels. Les mots « serveur » et « client » désignent les logiciels de type serveur et client dans cette architecture, logiciels fonctionnant sur les ordinateurs qu'on nomme par abus de langage serveur informatique et poste client. Le principe est le suivant :

- ☞ le client prend l'initiative d'envoyer des requêtes au serveur, puis attend la réponse pour la donner, le cas échéant, à l'utilisateur ;
- ☞ le serveur est à l'écoute d'un réseau informatique, prêt à répondre aux requêtes envoyées par des clients.

Un serveur est capable de servir plusieurs clients simultanément, jusqu'à plusieurs milliers.

Le client/serveur est un concept dans lequel les tâches sont réparties entre le client et le serveur, permettant ainsi aux tâches d'être exécutées par des machines adaptées à une situation donnée.

C'est une architecture au sein de laquelle l'application est perçue sous trois niveaux conceptuels :

- ✓ la présentation : il s'agit de l'interface utilisateur ;

- ✓ la logique applicative : c'est ensemble des traitements effectués par l'application ;
- ✓ la gestion des données : Elle concerne l'accès aux données et le maintien de leur intégrité.

1- Principaux composants du modèle client/serveur

- ✓ Le client : Doté généralement d'une interface permettant à l'utilisateur de valider l'entrée des données, c'est un système autonome qui demande des services au serveur. Il peut dans certains cas assurer une partie des traitements.
- ✓ Le serveur : Il stocke les données de façon cohérente et par conséquent, est chargé de répondre aux sollicitations formulées par les utilisateurs à partir de leur poste client.
- ✓ Le middleware : C'est une couche logicielle permettant d'assurer le dialogue entre le client et le serveur. Il est utilisé généralement comme intermédiaire de communication entre des applications complexes, distribuées sur un réseau informatique. Il assure plusieurs services parmi lesquels :
  - o la présentation de l'interface utilisateur ;
  - o l'interrogation et la gestion des données ;
  - o la communication sur le réseau dans l'échange des données ;
  - o les services de conception et de programmation.

2- Principaux types de modèles client/serveur

☞ *Architecture « Un-tiers »*

Dans une application un tiers, les trois couches applicatives sont intimement liées et s'exécutent sur le même ordinateur. On ne parle pas ici d'architecture client-serveur, mais d'informatique centralisée. Ce modèle permet la cohabitation sur la même

machine physique du client et du serveur ; ce qui alourdit et rend pénible le déploiement et n'offre pas un gage de sécurité.

☞ *Architecture « Deux-tiers »*

Dans une architecture deux tiers, encore appelée client-serveur de première génération ou client-serveur de données, le poste client se contente de déléguer la gestion des données à un service spécialisé. Le cas typique de cette architecture est l'application de gestion fonctionnant sous MS-Windows et exploitant un SGBD (Système de Gestion des Bases de Données) centralisé [WEB05].

Contrairement à l'architecture « Un-tiers », nous n'avons plus une seule machine physique mais plutôt deux. Ce qui permet d'installer le client sur un poste et le serveur sur l'autre. Le client s'adresse directement au serveur (sans intermédiaire) et attend la réponse à sa requête tout en restant connecté.

Même si ce modèle offre plus de sécurité que le précédent, il faut remarquer qu'une fois le serveur mis en place, il faut installer les clients un à un ; ce qui est fastidieux.

☞ *Architecture « Trois-tiers »*

Le principe d'une architecture « *trois-tiers* » consiste à séparer la réalisation des trois parties vues précédemment (stockage des données, logique applicative, présentation). Les éléments permettant la réalisation classique d'un système en architecture « *trois-tiers »* sont les suivants :

- ✓ système de gestion de bases de données (SGBD) pour le stockage des données ;
- ✓ serveur applicatif pour la logique applicative ;
- ✓ navigateur Web pour la présentation.

Dans cette architecture « *trois-tiers* », le poste client est communément appelé « *client léger* » ou « *Thin client* », par opposition au « *client lourd* » des architectures « deux-tiers ». Il ne communique qu'avec la façade HTTP de

l'application et ne dispose d'aucune connaissance des traitements applicatifs ou de la structure des données exploitées.

L'avantage principal d'une architecture « *trois-tiers* » est la facilité de déploiement. L'application en elle-même n'est déployée que sur la partie serveur (serveur applicatif et serveur de base de données). Le client ne nécessite aucune installation, le navigateur étant généralement livré avec le système d'exploitation de la machine cliente.

Signalons qu'il existe également des architectures « *n-tiers* » qui généralisent les serveurs de données qui peuvent ainsi être répartis. Cela conduit vers des architectures réparties.

B- Le modèle à composants

1- Présentation

Dans les années 90, sont apparues les architectures orientées composant, pour une plate-forme distribuée avec consommation d'objets distants. Le modèle de composants le plus commun dans l'environnement Windows est le COM *(Component Object Model)*. S'ils sont conçus correctement, les composants COM dans chacune de ses couches peuvent être réutilisés par d'autres composants et applications. Dans le monde Java, CORBA *(Common Object Request Broker Architecture)* est le plus utilisé. Il rend possible la distribution des tâches de développement à travers plusieurs programmeurs, et fait que, le système soit plus robuste, scalable, et maintenable. L'utilisation d'architecture basée sur les composants est devenue ces dernières années le meilleur moyen pour créer des systèmes embarqués [WEB13].

Une réutilisation bien orchestrée nécessite la création et le maintien d'une bibliothèque logicielle. Créer une application revient à créer les composants de bibliothèque nécessaires puis à construire l'application à l'aide de ces composants. Une telle bibliothèque, facilitant le développement d'applications est un

« framework » d'entreprise et son architecture, ainsi que sa documentation sont les pierres angulaires de la réutilisation logicielle en entreprise [WEB14].

Si comme nous l'avons dit, le modèle à composants est basé sur les notions de composants et d'assemblage, un ensemble d'autres concepts communs aux différents modèles à composants peut aussi être distingué (notons qu'ils ne sont pas tous implémentés par ces modèles) : interface, conteneur et services techniques qui sont liés à la séparation de la logique applicative et du code technique.

2- Principaux types de modèles à composants

On distingue couramment deux grands types de modèles à composants :

- *Modèles à composants de présentation*

Les modèles à composants de présentation sont principalement dédiés au développement rapide d'interfaces graphiques et n'implémentent pas les notions de conteneur.

Comme exemple, nous pouvons citer le composant de présentation JavaBeans.

- *Modèles à composants métiers*

Les composants métiers sont quant à eux des composants destinés aux architectures distribuées objet.

CORBA (*Common Object Request Broker Architecture)* qui est actuellement un modèle distribué objet de référence fait partie de ces composants. CORBA constitue sans doute l'un des modèles les plus utilisés au monde. On ne compte plus aujourd'hui le nombre d'ORB (Object Request Broker) gratuits ou payants basés sur ce modèle de développement ou intégrés dans de multiples serveurs d'applications.

Comme exemple de composants métiers, nous pouvons citer EJB *(Enterprise Java Bean)*, CCM de CORBA, .NET.

Signalons toutefois l'émergence d'un troisième type de modèles à composants qu'on pourrait qualifier de « génériques ». C'est le cas de Fractal, ArticBeans, Avalon.

C- Les architectures orientées services

Les deux dernières décennies ont été marquées par le développement rapide des systèmes d'information distribués, et tout particulièrement par la diffusion de l'accès à Internet. Cette évolution du monde informatique a entraîné le développement de nouveaux paradigmes d'interaction entre applications tels que SOA (*Service Oriented Architecture*). Cette dernière a été mise en avant afin de permettre des interactions entre applications distantes.

L'architecture SOA est une méthode de conception basée sur des standards, permettant de créer une infrastructure informatique intégrée capable de répondre rapidement aux nouveaux besoins d'un utilisateur. Elle fournit les principes et directives permettant de transformer un réseau existant de ressources informatiques hétérogènes, distribuées, complexes et rigides en ressources intégrées, simplifiées et particulièrement souples pouvant être modifiées et combinées afin de mieux satisfaire les objectifs de l'utilisateur [WEB13].

Cette approche dont le but est de mettre en place une architecture logicielle globale décomposée en services correspondant aux processus métiers de l'entreprise, repose sur la réorganisation des applications en ensembles fonctionnels appelés services.

La SOA est une approche client-serveur de conception d'applications qui consiste en des fournisseurs, des consommateurs et des contrats de services dans un souci de réutilisabilité et d'interopérabilité [MEN09].

Lorsque l'architecture SOA s'appuie sur des Web services, on parle alors de WSOA (Web Services Oriented Architecture).

1- Principes généraux d'une SOA

Il n'existe pas à proprement parler de spécifications officielles d'une architecture SOA. Néanmoins les principales notions fédératrices que l'on retrouve dans une telle architecture sont les suivantes :

☞ *La notion de service*

Le service est un composant clé de l'architecture orientée services. Il consiste en une fonction ou fonctionnalité bien définie. C'est aussi un composant autonome qui ne dépend d'aucun contexte ou service externe ; ce qui en garantit la réutilisabilité. Il est divisé en opérations qui constituent autant d'actions spécifiques que le service peut réaliser. On peut faire un parallèle entre opérations et services d'une part, et méthodes et classes dans le mode orienté objet d'autre part [WEB06].

Un service est une fonction encapsulée dans un composant que l'on peut interroger à l'aide d'une requête composée d'un ou plusieurs paramètres et fournissant une ou plusieurs réponses.

Un service est une offre d'expertise par une interface bien définie et disponible pour la communauté. Plusieurs services peuvent être composés pour créer un service plus large.

Le service peut être codé dans n'importe quel langage et s'exécuter sur n'importe quelle plate-forme (matérielle et logicielle).

☞ *La description du service*

Il s'agit de décrire les paramètres d'entrée du service, le format et le type des données retournées. Le format de description de services le plus utilisé est WSDL (*Web Services Description Language*), normalisé par le W3C (*World Wide Web Consortium*).

☞ *La publication et la découverte des services*

La publication consiste à mettre dans un registre les services disponibles aux utilisateurs, tandis que la notion de découverte recouvre la possibilité de rechercher un service parmi ceux qui ont été publiés.

Le principal standard utilisé est UDDI (*Universal Description Discovery and Integration*), normalisé par l'OASIS (*Organization for the Advancement of Structured Information Standards*).

☞ *L'invocation*

Elle représente la connexion et l'interaction du client avec le service. Le principal protocole utilisé pour l'invocation de services est SOAP (*Simple Object Access Protocol*).

2- Avantages d'une SOA

Par son caractère standard, l'approche SOA contribue à améliorer la rapidité ainsi que la productivité des développements. Un composant exposé sous forme de service Web pouvant être réutilisé à loisir par d'autres applications. C'est le cas par exemple des dispositifs d'interrogation et d'inscription au sein d'un annuaire d'entreprise, solution classiquement conçue pour gérer les droits d'accès aux applications en stockant les profils des utilisateurs. Une fois publiées par le biais de services Web, ces fonctions peuvent être aisément intégrées aux différentes briques du système d'information sans que le développement de connecteurs spécifiques au cas par cas ne soit nécessaire.

Une architecture orientée services permet aussi d'obtenir tous les avantages d'une architecture client-serveur et notamment :

- ✓ une modularité permettant de remplacer facilement un composant (service) par un autre ;
- ✓ une réutilisabilité possible des composants (par opposition à un système tout-en-un fait sur mesure pour une organisation) ;
- ✓ de meilleures possibilités d'évolution car il suffit de faire évoluer un service ou d'ajouter un nouveau service ;
- ✓ une plus grande tolérance aux pannes ;
- ✓ une maintenance facilitée.

3- Les services Web

a- Présentation et fonctionnement

Les services Web poursuivent un vieux rêve de l'informatique: celui d'un monde où les ressources informatiques pourraient inter-opérer à travers un réseau, indépendamment de leurs plates-formes d'origine. Les Web services, apparus avec la mouvance Internet et le développement rapide des applications client-serveur accessibles via un navigateur Web, se veulent un aboutissement de la logique des applications distribuées. Ce concept peut être analysé comme une réponse au désir des entreprises et des administrations de relier différents composants logiciels internes et externes.

Deux protocoles fondamentaux sont utilisés : SOAP (Simple Object Access Protocol) qui définit la structure des messages XML utilisés par les différentes applications pour communiquer ensemble ; WSDL (Web Services Description Language) qui est un langage de contrat décrivant la façon pour une application d'invoquer une autre application : standardisation des schémas XML utilisés pour établir une connexion entre émetteurs et récepteurs. Enfin, dans l'idée d'applications distribuées sur le réseau Internet et utilisables comme des services, il est nécessaire de disposer d'une cartographie interrogeable de ces services : c'est l'objet des annuaires de services UDDI (Universal Description Discovery and Integration). UDDI est un annuaire mondial d'entreprises basé sur le Web, intégrant toutes sortes d'entrées (nom, carte d'identité des sociétés, description des produits et des services, etc.). Son objectif à terme est d'automatiser les communications entre prestataires, clients, etc.

Un service Web est décrit dans un document WSDL, précisant les méthodes pouvant être invoquées, leur signature, et les points d'accès du service (URL, port, etc.). Ces méthodes sont accessibles via SOAP : la requête et la réponse sont des messages XML transportés par HTTP. Un service Web est accessible depuis n'importe quelle plate-forme ou langage de programmation. On peut utiliser un service Web pour exporter des fonctionnalités d'une application et les rendre accessibles via des protocoles standards. Le service Web sert alors d'interface

d'accès à l'application, et dialogue avec elle au moyen de middleware (Corba, RMI, DCOM, EJB, etc.).

b- Utilités des Services Web en milieu mobile

En milieu mobile, les services Web ont l'avantage de pouvoir alléger considérablement la taille des applications sur les supports mobiles qui ont souvent un environnement de faible capacité. Ils permettent de faciliter les développements de services mobiles en offrant une manière simple d'accéder à distance à un serveur de données. Pour une application mobile il est alors possible de pouvoir effectuer un traitement lourd à distance par une simple requête/réponse, réduite à l'appel d'une méthode (au sens programmation orientée objet), voire de n'être qu'une simple application visuelle, le fonctionnel restant sur le serveur.

Les services Web fournissent l'interopérabilité entre divers logiciels fonctionnant sur diverses plates-formes et ils peuvent fonctionner au travers de nombreux pare-feux sans nécessiter des changements sur les règles de filtrage car basés sur le protocole http.

Des librairies existent déjà en Java permettant de faire appel à des services Web depuis des plates-formes pour mobile comme J2ME ou Personal Java. L'un des objectifs principaux de la plate-forme .NET de Microsoft est de permettre l'appel de Services Web depuis n'importe quel support, dont les mobiles utilisant des systèmes de l'offre Windows Mobile. Bien que ces deux plates-formes soient concurrentes, l'objectif principal des Services Web est qu'elles puissent communiquer.

Cependant, ils nécessitent que la connexion entre les applications fonctionne en mode connecté.

Rien n'est prévu dans les spécifications pour gérer la problématique des fluctuations des réseaux mobiles. Il est possible de pallier cela en utilisant un protocole assurant l'acheminement des données sur le réseau. Aujourd'hui le protocole HTTP ne permet pas d'assurer cela.

Au regard des architectures présentées, une architecture orientée services dans laquelle les principales fonctionnalités des applications seront développées et publiées sous forme de Web services répondrait mieux à notre projet.

II- Technologies et outils pour le développement des applications pour mobile et Web

A- Technologies pour le développement des applications pour mobile

Actuellement, il existe beaucoup de langages pour le développement des applications embarquées dont les plus utilisés sont : la technologie J2ME de SUN utilisant le langage Java et la technologie .NET Mobile (ou Microsoft Mobile Internet Toolkit) de Microsoft utilisant des langages de la technologie .NET.
Dans les parties suivantes, nous allons présenter ces deux technologies.

1- Java 2 Micro Edition (J2ME)

a- Présentation

Java 2 Micro Edition est une architecture technique dont le but est de fournir un socle de développement aux applications embarquées. L'intérêt est de proposer toute la puissance d'un langage tel que Java associé aux services proposés par une version bridée du Framework J2SE : J2ME [WEB15]. Les terminaux n'ayant pas les mêmes capacités en terme de ressources que les ordinateurs de bureau classiques (mémoire, disque et puissance de calcul), la solution passe par la fourniture d'un environnement allégé afin de s'adapter aux différentes contraintes d'exécution. La solution proposée par J2ME consiste à regrouper par catégories certaines familles de produits tout en proposant la possibilité d'implémenter des routines spécifiques à un terminal donné. L'architecture J2ME se découpe donc en plusieurs couches *(profils et configurations), qui ne sont en fait que les API (Application Programming Interface) Java de bases simplifiées* :

Les profils : Ils permettent à une certaine catégorie de terminaux d'utiliser des caractéristiques communes telles que la gestion de l'affichage, des événements d'entrées/sorties (pointage, clavier, …) ou des mécanismes de persistance (Base de données légère intégrée). Ces profiles sont soumis à spécifications suivant le principe du **JCP** (*Java Community Process*)

Les configurations : Elles définissent une plate-forme minimale en termes de services concernant un ou plusieurs profils donnés.

Les machines virtuelles : En fonction de la cible, la machine virtuelle pourra être allégée afin de consommer plus ou moins de ressources (KVM, CVM, …)

Le système d'exploitation : L'environnement doit s'adapter au système d'exploitation existant (Windows CE, Palm Os, SavaJe, …)

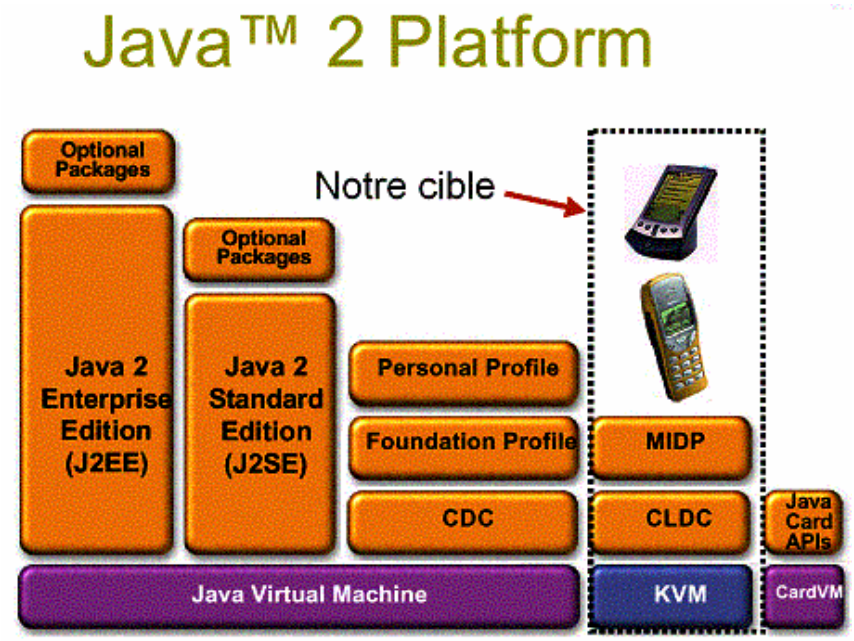


Figure 1 : Architecture de J2ME

Chaque profil repose sur une configuration qui elle-même repose sur une machine virtuelle spécifique. Aujourd'hui SUN a spécifié deux configurations basées chacune sur une machine virtuelle bien spécifique aux terminaux visés : CDC *(Connected Device Configuration)* et CLDC *(Connected Limited Device Configuration)*.

- La CDC est plus adaptée aux terminaux relativement puissants comme les PDA. En effet, elle nécessite une machine virtuelle java optimisée appelée CVM qui offre les mêmes fonctionnalités que la JVM classique.
- La CLDC est, par contre, dédiée aux appareils avec de faibles capacités comme les téléphones portables. La machine virtuelle allégée correspondante est la KVM *(KiloByte Virtual Machine)* et ne possède pas certaines fonctions de la JVM classique.

Lorsqu'une configuration définit le fondement d'une application, un profil en fournit la structure. Les profils définissent l'ensemble des API à utiliser dans une application J2ME et sont conçus spécialement pour chaque configuration.

SUN propose deux profils de référence J2ME : le profil Foundation et le profil MIDP (*Mobile Information Device Profile*).

- Le profil Foundation est destiné à la configuration CDC. Les développeurs qui utilisent ce profil ont accès à une implémentation complète des fonctionnalités de J2SE (Java 2 Standard Edition).
- Le profil MIDP est destiné à la configuration CLDC. Il prend en charge un nombre limité des classes de J2SE et définit des classes d'entrée/sortie et d'interface spécialisées pour une configuration CLDC.

b- Les outils de développement J2ME du marché

Il existe plusieurs environnements de développement intégrés (IDE) proposant l'intégration de l'API J2ME directement dans l'éditeur:

- Le MobileSet Plug-In de JBuilder : téléchargeable sur le site de Borland
- Forte de SUN : Le J2ME Toolkit pouvant être paramétré à l'installation comme un Plug-In de Fose.
- Visual Age Micro Edition (connu aussi sous le nom de VAME) : permet de tester des applications J2ME sous WebSphere Studio en ciblant des émulateurs tels que les PocketPC , Palm, etc ...
- Oracle 9i DS : Un plug-in pour l'IDE d'Oracle, JDeveloper.

- Eclipse : Cet IDE OpenSource intègre un Plug-In qui permet de développer des applications J2ME contenant la librairie graphique SWT à destination de divers périphériques (PocketPC, Palm, Mobiles, ...)
- Netbeans IDE avec Mobility Pack permet de développer des applications mobiles

2- Microsoft Windows Embedded

A travers le temps, l'approche de Microsoft concernant le monde de l'embarqué a considérablement évolué. De Microsoft Embedded Visual Tools (eVB et C++), on en est à la plate-forme .NET est actuellement disponible avec les différents langages qui l'accompagnent (C#, VB.NET, ...). Aujourd'hui, l'offre de Microsoft est essentiellement basée sur son système d'exploitation embarqué : Windows CE.

Microsoft Embedded propose une approche plus orientée « système » de l'embarqué. Un constructeur désireux d'intégrer un environnement Windows dans un terminal donné, devra extraire les briques logicielles d'une version standard en fonction des caractéristiques du récepteur. Par exemple, dans le cas d'un périphérique sans fil, les composants embedded « wireless » accompagnés des drivers adéquats seront proposés à l'intégrateur. Lorsque le terminal nécessite une base de données embarquée, il suffira de générer une image du Système d'exploitation prenant en compte cette contrainte. Microsoft propose une panoplie de composants systèmes en tout genre et adaptés à tout type de besoin, il revient ensuite à l'intégrateur de définir sa propre configuration.

Avec l'avènement du Framework .NET, la partie embarquée de l'offre de Microsoft a subi quelques évolutions majeures. Ainsi, le **.**NET Compact Framework (.NET CF) a fait son apparition au sein de l'architecture avec une philosophie proche de J2ME pour Java: un ensemble d'API allégé permettant de répondre aux exigences des périphériques en termes de ressources. Mais .NET CF se destine aux plates-formes Windows CE [WEB07].

De tout ce qui précède, la plate-forme de développement mobile J2ME paraît intéressante et par conséquent est pourrait être choisie pour l'implémentation de la solution qui sera retenue car la technologie est supportée par la majorité des téléphones mobiles. De plus, malgré l'existence d'autres alternatives pour les applications mobiles comme Symbian OS (*Operating System*), Windows Mobile, et Motorola, Java est portable grâce à ses machines virtuelles déployables sur une multitude de plates-formes.

B- Technologies pour le développement des applications Web

L'architecture des applications Web repose sur le fait que la présentation n'est plus créée dans le client, mais par un serveur d'application Web intégrant au moins l'une des technologies suivantes :

- PHP *(Hypertext Preprocessor)* : est un langage interprété (un langage de script) exécuté du côté du serveur. La syntaxe du langage provient de celle du langage C. Ses principaux atouts sont : sa gratuité, son intégration au sein de nombreux serveurs web (Apache, Microsoft IIS, ...), la simplicité d'écriture de scripts, et la simplicité d'interfaçage avec des bases de données.

- ASP.NET *(Active Server Pages)* : standard propriétaire (Microsoft) qui permet de développer des applications Web interactives, c'est-à-dire dont le contenu est dynamique. L'inconvénient majeur d'ASP est qu'il n'est disponible qu'avec le serveur Web de Microsoft (*IIS*) et donc ne s'exécute que sous le système d'exploitation Windows.

- JSP *(JavaServer Pages)* : est une technologie JAVA directement concurrente d'ASP, de plus elle est indépendante de la plateforme. Les commandes JSP sont placées dans le code HTML. Leur mode de fonctionnement est relativement lourd, il se déroule en 4 temps : La requête

est reçue par le serveur, la page demandée est traduite en servlets, puis compilée et exécutée pour être à la fin transmise au client.

- Servlets : sont un peu l'inverse des JSP, dans le sens où le code HTML est intégré dans du code JAVA. Leur mode de fonctionnement se passe en deux temps : la requête est reçue par le serveur puis la servlet est exécutée et la page HTML est transmise au client.

En vertu de sa gratuité, de son intégration au sein de nombreux serveurs Web, de sa simplicité d'écriture de scripts et d'interfaçage avec des bases de données d'une part, et de l'adoption du couple PHP/MySQL par la majorité des hébergeurs Web, PHP sera utilisé pour la réalisation de notre projet.

C- Les systèmes de gestion de base de données

Le système de gestion de bases de données (SGBD) ou en anglais DBMS (*Database management system*) est un ensemble de services (applications logicielles) permettant de gérer les bases de données.
De nombreux SGBD sont disponibles sur le marché, partant des SGBD gratuits jusqu'aux SGBD destinés spécialement aux professionnels, comportant de nombreuses fonctionnalités, mais plus coûteux. Au nombre de ceux-ci, on peut citer :

- **MySQL :** Selon le type d'application, sa licence est libre ou propriétaire. Il fait partie des logiciels de gestion de base de données les plus utilisés au monde, autant par le grand public (applications web principalement) que par des professionnels
- **Oracle :** est un système de gestion de base de données relationnel (SGBDR) qui depuis l'introduction du support du modèle objet dans sa version 8 peut être aussi qualifié de système de gestion de base de données relationnel-objet (SGBDRO).

- **Microsoft SQL Server :** est un Système de gestion de base de données relationnelles développé et commercialisé par la société Microsoft
- **PostgreSQL :** est un système de gestion de base de données relationnelle et objet (SGBDRO). C'est un outil libre disponible selon les termes d'une licence
- **SQLite :** est une bibliothèque écrite en C qui propose un moteur de base de données relationnelle accessible par le langage SQL. SQLite implémente en grande partie le standard SQL-92 et des propriétés ACID.

Vu que MySQL est une solution très courante en hébergement public et compte tenu de sa très bonne intégration dans l'environnement Apache/PHP et sa facilité de déploiement et de prise en main, nous l'avons préféré aux autres.

CONCLUSION

Il apparaît clair que nous que nous disposons des moyens technologiques évolués pour mettre en place une solution de plateforme mobile au profit des IMF. Pour atteindre nos objectifs nous devons choisir les outils adéquats et optimiser le plus possible les solutions proposées.

2ème partie : Conception et réalisation du système

« A long terme, la réalisation d'un événement improbable tend vers la certitude ».
Jean Largeault

Chapitre 3 : Analyse des besoins et conception du système

Cette partie de notre mémoire est essentiellement consacrée à l'analyse à la conception du système.

I- Etude préliminaire

Le projet de conception et réalisation d'une plate-forme mobile de gestion de crédits pour les Institutions de Micro Finances vise à faciliter la mobilité des agents de terrain dédiés aux opérations de mise en place et de suivi des micro-crédits.

L'outil à réaliser est destiné aux périphériques mobiles dotés des services Internet et doit être en mesure de fournir aux agents "nomades" la possibilité d'effectuer les opérations courantes de gestion de crédit.

A- Recueil des besoins

L'identification des besoins est l'étape la plus importante du processus de développement de logiciels. Elle aide à orienter le produit final vers son utilisation. Elle représente le point d'entrée du développement logiciel.

Le principal besoin est de réaliser à distance, à tout moment, et en tout point, les opérations relatives au portefeuille crédit. Pour recenser les besoins, nous nous sommes servis du manuel de procédures de l'Association pour la Promotion et l'Appui au Développement des Micro-Entreprises (PADME).

PADME est l'une des premières institutions béninoises de crédit direct qui apporte une solution plus simple au besoin de financement des populations à faibles revenus. Elle est le résultat de la transformation institutionnelle du Projet d'Appui au Développement des Micro Entreprises (PADME) initié par le gouvernement béninois le 1er Septembre 1993 pour juguler les effets sociaux du Programme

d'Ajustement Structurel (PAS). Elle a pour vision l'offre des services financiers adaptés aux micros entreprises et aux personnes à faibles revenus. Elle couvre aujourd'hui toute l'étendue du territoire du Benin.

Grâce à son manuel de procédures bien élaborées, nous avons recensé les opérations usuelles effectuées par les agents dont leur fonction exige une certaine mobilité.

Simulation de prêt :

Elle consiste à déterminer le montant à payer par échéance à partir du montant sollicité, du nombre de paiement, du type de produit et de la périodicité du remboursement. Cette opération s'effectue lors des demandes de crédit.

Consultation :

Le système doit permettre aux utilisateurs de consulter :

- ✓ la fiche d'identification des clients
- ✓ le solde des crédits
- ✓ la liste des impayés
- ✓ l'échéancier de paiement

Enregistrement de la fiche d'identification d'un client :

Le chargé de prêts saisit les informations relatives à l'identité du client.

Traitement de la demande de crédit :

Le chargé de prêts saisit les informations afférentes aux demandes de crédit. Lors de l'enregistrement, le système doit pouvoir vérifier si le client n'a pas de crédit non soldé de même type en cours.

Le chef bureau, après évaluation du dossier, valide (approuve) la demande de crédit. Après décaissement, le crédit est affecté au portefeuille du chargé de prêts.

Pour se connecter au système, l'utilisateur doit être reconnu à partir d'un nom (login), un mot de passe.

De plus l'application devrait non seulement présenter une interface agréable et facile à utiliser mais aussi permettre d'effectuer un nombre minimal de requêtes

sur le serveur tout en tenant compte des facteurs de stabilité et de rapidité d'exécution.

B- Identification des acteurs

Un acteur en UML est une entité externe, un utilisateur humain, un dispositif matériel ou un autre système qui interagit directement avec le système étudié.

Un acteur peut consulter ou modifier l'état d'un système en émettant et/ou en recevant des messages susceptibles d'être porteurs de données. On les identifie en se concentrant sur les rôles joués par les entités extérieures au périmètre du système.

Le chargé de prêts : responsable de la mise en place, du suivi et du recouvrement des crédits dans son secteur, il a pour mission de saisir et éventuellement de modifier ou d'annuler les demandes de crédit. Il peut faire la simulation d'un prêt, consulter les informations sur le crédit et éditer la liste des impayés.

Le chef de Bureau : Il est responsable de la coordination des opérations de crédit dans le bureau de zone. Il effectue la validation des demandes de crédit dans le système et consulte les informations fournies.

L'administrateur du système : responsable du système, il gère les profils des utilisateurs et les mots de passe.

L'agent de recouvrement : il assure le recouvrement des impayés.

L'auditeur interne : il effectue par moment des contrôles sur les crédits mis en place.

II- Conception

Pour la conception du système, nous utiliserons UML (Unified Modeling Language) [AUD08, ROQ02], un langage de modélisation basé sur des diagrammes descriptifs permettant d'offrir une vision claire d'un système (que se soit informatique ou autre). Pour une modélisation, tous les diagrammes ne sont pas nécessairement produits [AUD08] ; nous allons donc nous accentuer sur les diagrammes des cas d'utilisation, des classes et des séquences.

A- Diagramme des cas d'utilisation

1- Identification des cas d'utilisation

Un cas d'utilisation représente un ensemble de séquences d'actions réalisées par le système et produisant un résultat observable intéressant pour un acteur particulier [ROQ02].

Un cas d'utilisation modélise un service rendu par le système. Il exprime les interactions acteurs/système et apporte une valeur ajoutée notable à l'acteur concerné.

Liste des cas d'utilisation

Cas d'utilisation	**Acteur principal, acteurs secondaires**	**Message(s) émis / reçus par les acteurs**
S'authentifier	Chargé de prêts, Chef bureau, Administrateur, Agent de recouvrement, Auditeur	
Simuler un prêt	Chargé de prêts, chef	Emet : Création, modification,

	bureau	annulation échéancier Reçoit : échéancier généré
Traiter demande	Chef bureau	Emet : Valider, rejeter demande
Enregistrer client	Chargé de prêts	Emet : création, modification, annulation Reçoit : confirmation enregistrement client
Enregistrer demande de prêt	Chargé de prêts	Emet : création, modification, annulation Reçoit : confirmation enregistrement
Consulter fiche d'identification client	Chargé de prêts, Chef bureau, Administrateur, Agent de recouvrement, Auditeur	Reçoit : fiche d'identification
Consulter échéancier	Chargé de prêts, Chef bureau, Administrateur, Agent de recouvrement, Auditeur	Reçoit : échéancier du crédit
Consulter le solde d'un crédit	Chargé de prêts, Chef bureau, Administrateur, Agent de recouvrement, Auditeur	Reçoit : Etat du crédit
Consulter liste des impayés	Chargé de prêts, Chef bureau, Administrateur, Agent de recouvrement,	Reçoit : liste des impayés

	Auditeur	
Consulter liste des paiements attendus	Chargé de prêts et Agent de recouvrement	Reçoit : liste des paiements attendus
Gérer utilisateur et données	Administrateur	Emet : création, modification, annulation Reçoit : confirmation enregistrement
Administrer le système	Administrateur	Emet : mise à jour Reçoit : confirmation de la mise à jour

A partir des cas d'utilisation et des différents acteurs recensés, nous obtenons le diagramme des cas d'utilisation ci-après (Figure 2) :

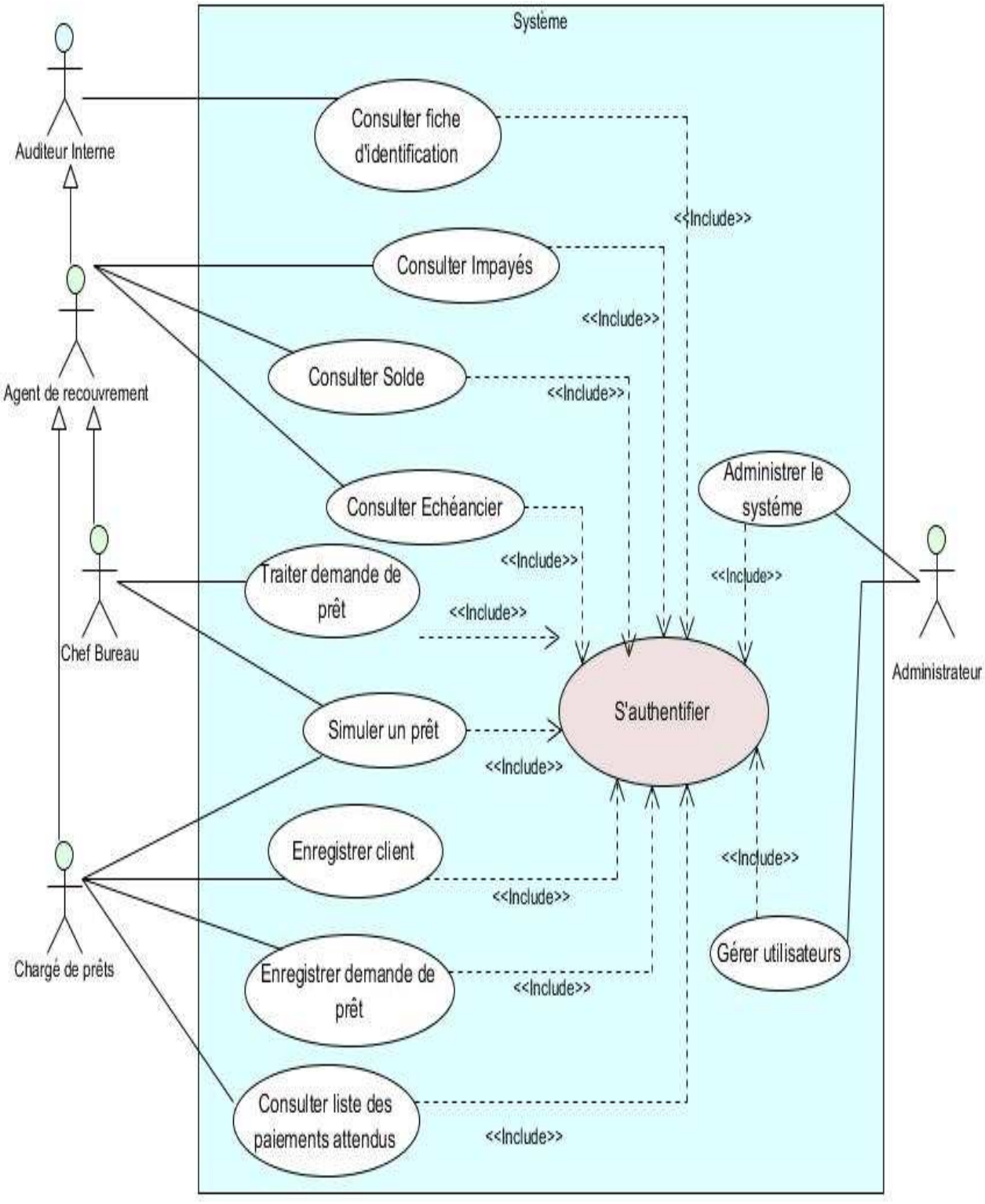


Figure 2 : Diagramme des cas d'utilisation

2- Description textuelle des cas d'utilisation

Cas d'utilisation : « S'authentifier »

Titre : S'authentifier

But : Vérifier l'identité de l'utilisateur

Résumé : Tout utilisateur avant l'exécution d'une action sur le système doit s'identifier en fournissant son identifiant et mot de passe

Acteurs : Tout acteur voulant se connecter au système

Pré conditions

- ✓ L'Internet est disponible sur le mobile

Scénario nominal

Ce cas d'utilisation commence quand un utilisateur est invité à se connecter au système.

Enchainement 1 : S'authentifier

L'utilisateur saisit son login et son mot de passe puis valide sa saisie. Si les informations sont erronées alors **Exception1[ErreurAuthentification]**

Exception

Exception1[ErreurAuthentification] : Le système affiche un message d'erreur d'authentification à l'utilisateur.

Ce cas d'utilisation prend fin lorsque l'utilisateur arrive à se connecter ou abandonne l'opération.

Post conditions

- ✓ L'utilisateur peut effectuer toutes actions sur le système selon son profil

Cas d'utilisation : « Simuler un prêt »

Titre : Simuler prêt

But : Avoir un aperçu de l'échéancier d'un prêt

Résumé : A partir de certaines données, on établit l'échéancier de paiement

Acteurs : Chargé de prêt, chef bureau

Pré conditions

- ✓ L'utilisateur est authentifié
- ✓ Au moins un produit est saisi dans la base

Scénario nominal

Ce cas d'utilisation commence lorsque l'utilisateur demande au système de simuler un prêt.

Enchainement 1 : Simuler prêt

L'utilisateur saisit les informations de base pour obtenir l'échéancier d'un prêt : le montant sollicité, le nombre d'échéances, la périodicité de remboursement. Il sélectionne le produit sollicité ainsi que le type d'échéancier. Il indique le nombre de mois de différé au besoin et valide sa saisie.

Si toutes les informations sont correctement saisies, le système génère et affiche l'échéancier ; sinon **Exception1[SaisieIncorrecte]**.

Exception

Exception1[SaisieIncorrecte] : Le système affiche un message d'erreur de saisie à l'utilisateur en fonction de la donnée erronée.

Ce cas d'utilisation prend fin lorsque l'utilisateur entre des données correctes et génère l'échéancier ou lorsqu'il abandonne l'opération.

Post conditions

- ✓ L'échéancier est généré

Cas d'utilisation : « Enregistrer client »

Titre : Enregistrer clients

But : Faire la mise à jour des clients

Résumé : A partir des informations fournies par le client, procéder à la création de sa fiche d'identification. Modification ou annulation de l'enregistrement.

Acteur : Chargé de prêt

Pré conditions

- ✓ L'utilisateur est authentifié
- ✓ Au moins une catégorie de clients est créée

Scénario nominal

Ce cas d'utilisation commence lorsque l'utilisateur demande au système d'enregistrer un client.

Enchainement 1 : Enregistrer client

L'utilisateur sélectionne la catégorie de client à enregistrer et saisit les informations permettant d'identifier le client. Il s'agit de l'adresse, de la date de naissance. Pour les personnes physiques, l'utilisateur précise les nom et prénoms ainsi que le sexe et pour les personnes morales, il indique la raison sociale, le statut juridique.

A la fin de la saisie, le système effectue une vérification pour s'assurer qu'il n'y a pas de doublons. Si le client n'est pas encore dans la base alors le système procède à son enregistrement après lui avoir attribué un code unique sinon

Exception1[DoublonClient].

Exception

Exception1[DoublonClient] : Le système affiche un message informant l'utilisateur que le client qu'il désire enregistrer figure déjà dans la base.

Enchainements alternatifs

Enchainement 2 : Modifier client

L'utilisateur accède à l'enregistrement d'un client par saisie dans le système du code du client ou de son nom ou encore de sa raison sociale.

Si les informations fournies sont correctes, le système affiche le client correspondant et l'utilisateur procède aux modifications voulues à l'exception de celle du code client et valide sa saisie ; sinon **Exception1[ClientInexistant]**.

A la fin de la saisie, le système effectue une vérification pour s'assurer que la modification ne fait pas apparaître de doublons. Si tel n'est pas le cas, le système procède à l'enregistrement des modifications sinon **Exception2[DoublonClient]**.

Exception

Exception1[ClientInexistant] : Le système affiche un message informant l'utilisateur que le client recherché ne se trouve pas dans la base.

Exception2[DoublonClient] : Le système affiche un message informant l'utilisateur que ses modifications feront apparaître des doublons et que l'opération ne peut aboutir.

Post Conditions

Le client est créé et peut être identifié

Cas d'utilisation : « Enregistrer une demande de prêt »

Titre : Enregistrer une demande de prêt

But : Enregistrer une demande de crédit sollicitée par client

Résumé : A partir des informations fournies par le client, enregistrer sa demande de crédit. Modification ou annulation de la demande de prêt.

Acteurs : Chargé de prêt

Pré conditions

- ✓ Le chargé de prêts est authentifié
- ✓ Au moins un produit est saisi dans la base
- ✓ Le client a un identifiant

Scénario nominal

Ce cas d'utilisation commence lorsque l'utilisateur demande au système d'enregistrer une nouvelle demande de crédit.

Enchainement 1 : Enregistrer demande

L'utilisateur saisit les informations relatives à la demande de crédit : l'objet du crédit, le montant du fonds de garantie, le montant sollicité, la date de déboursement, le nombre d'échéances, la périodicité de remboursement. Il sélectionne le client concerné ou le crée si c'est un nouveau client. Il sélectionne le produit sollicité, saisit les garanties du client au besoin et valide sa saisie.

Si le client possède déjà un crédit du même type en cours alors **Exception1[CreditEnCours]** ; sinon si le montant sollicité est trop élevé par rapport au type de crédit alors **Exception2[MontantElevé]** ; sinon si il y a des données erronées dans la saisie alors **Exception3[SaisieIncorrecte]** ; sinon le système procède à l'enregistrement des données dans la base.

Exception

Exception1[CreditEnCours] : Le système affiche un message informant l'utilisateur que ce client a déjà un crédit du même type en cours.

Exception2[MontantElevé] : Le système affiche un message informant l'utilisateur

que le montant sollicité est trop élevé par rapport au type de crédit sélectionné.

Exception3[SaisieIncorrecte] : Le système affiche un message d'erreur de saisie à l'utilisateur en fonction de la donnée erronée.

Enchainements alternatifs

Enchainement 2 : Modifier demande

L'utilisateur sélectionne une demande parmi celles non encore validées et procède aux modifications souhaitées à l'exception du numéro de demande. A la fin des modifications, il valide sa saisie.

Ce cas d'utilisation prend fin lorsque l'utilisateur quitte l'écran de création/modification de demandes de crédit.

Enchainement 2 : Annuler une demande

Le chargé de prêts peut annuler une demande de prêt si elle n'est pas encore validée, c'est-à-dire si elle n'est pas encore enregistrée dans la base de données centrale.

Post Conditions

La demande de prêt est enregistrée et a un numéro.

Cas d'utilisation : « Consulter fiche d'identification client »

Titre : Consulter fiche d'identification client

But : Obtenir les informations sur l'identité d'un client

Résumé : Consulter la fiche signalétique du client en cas de besoin. Elle fournit les informations sur son identité.

Acteurs : Acteurs humains

Pré conditions :

1. Les utilisateurs sont authentifiés
2. Les clients sont enregistrés dans le système

Scénario nominal

Ce cas d'utilisation est exécuté lorsque les utilisateurs ont besoin des informations pour identifier un client.

Enchaînement 1: consulter fiche d'identification

L'utilisateur accède à l'écran de consultation et fournit un paramètre de recherche tel que le nom du client ou son numéro.

Le système effectue les recherches et affiche le résultat.

Post Conditions

La fiche d'identification est affichée à l'écran.

Cas d'utilisation : « Consulter échéancier »

Titre : Consulter échéancier

But : Obtenir les informations sur l'échéancier des paiements

Résumé : Consulter l'échéancier pour avoir les dates et les montants des échéances d'un crédit.

Acteurs : Acteurs humains

Pré conditions :

1. Les utilisateurs sont authentifiés
2. Les crédits sont enregistrés et leurs échéances générées dans le système

Scénario nominal

Ce cas d'utilisation est exécuté lorsque les utilisateurs ont besoin des informations sur l'échéance du crédit d'un client.

Enchainement 1 : Consulter échéancier

L'utilisateur accède à l'écran de consultation des échéanciers et fournit un paramètre de recherche tel que le numéro du contrat, le nom du client ou le numéro client.
Le système effectue les recherches et affiche le résultat.

Post Conditions

L'échéancier est affiché à l'écran.

Cas d'utilisation : « Consulter le solde d'un crédit »

Titre : Consulter le solde d'un crédit

But : Connaître le solde d'un crédit

Résumé : Consulter en cas de besoin ou de demande le solde d'un crédit à une date donnée.

Acteurs : Acteurs humains

Pré conditions :

1. Les utilisateurs sont authentifiés
2. Les crédits sont enregistrés

Scénario nominal

Ce cas d'utilisation est exécuté lorsque les utilisateurs ont besoin ou par sollicitation du solde d'un crédit.

Enchainement 1 : Consulter le solde

L'utilisateur accède à l'écran de consultation de solde et fournit un paramètre de recherche tel que le numéro du contrat, le nom du client ou le numéro client.

Le système effectue les recherches et affiche le résultat.

Post Conditions

Le solde du crédit est affiché à l'écran.

Cas d'utilisation : « Consulter liste des impayés »

Titre : Consulter liste des impayés

But : Identifier les crédits dont les échéances ne sont pas honorées

Résumé : Consulter tous les jours la liste des crédits dont les échéances sont échues mais pas encore payées à une date donnée.

Acteurs : Acteurs humains

Pré conditions :

1. Les utilisateurs sont authentifiés
2. Les Crédits sont déjà enregistrés dans le système
3. Echéances des crédits sont enregistrées

Scénario nominal

Ce cas d'utilisation est exécuté lorsque les utilisateurs ont besoin de faire les recouvrements des crédits en souffrance.

Enchainement 1 : Consulter liste des impayés

L'utilisateur précise les critères de l'édition tels que le nom du chargé de prêts ou le code du bureau de zone.

Post Conditions

La liste des crédits en impayés est éditée.

Cas d'utilisation : « *Traiter demande* »

Titre : Traiter demande

But : Mettre le crédit sollicité en place ou rejeter les demandes de crédit

Résumé : Valider ou rejeter les demandes de crédit

Acteur : Chef bureau

Pré conditions

- ✓ L'utilisateur est authentifié
- ✓ Au moins une demande non validée existe dans la base

Scénario nominal

Ce cas d'utilisation commence lorsque l'utilisateur demande au système de traiter une demande de crédit.

Enchainement 1 : Valider demande

Le système affiche les demandes non validées. L'utilisateur pour chaque demande à valider, clique sur le bouton correspondant en face de la demande. Le système vérifie que la date de déboursement enregistrée au niveau de la demande est postérieure à la date de validation ; sinon **Exception1[DateDéboursementPassée]**.

Le système met en œuvre les règles de gestion pour que soit marquée la demande comme définitivement validée et mettre en place le crédit puis créer un compte au client, soit marquée la demande comme provisoirement validée et la rendre visible au supérieur hiérarchique de l'acteur ayant opéré la validation. Dans ce cas, ce dernier effectue l'enchainement 1 ou l'enchainement 2 à son niveau.

Exception

Exception1[DateDéboursementPassée] : Le système affiche un message informant l'utilisateur que la date de déboursement de cette demande est déjà passée et que la validation ne peut se faire.

Enchainement 2 : Rejeter demande

Le système affiche les demandes non validées. L'utilisateur pour chaque demande à rejeter, clique sur le bouton correspondant en face de la demande. Le système invite

l'utilisateur à saisir le motif du rejet. Ce dernier procède à la saisie et valide l'information. Le système procède à l'enregistrement dans la base.

Ce cas d'utilisation prend fin lorsque l'utilisateur quitte l'écran de validation/rejet de demandes de crédit.

Post Conditions

La demande de prêt est validée ou rejetée

B- Diagramme des classes

Le diagramme de classes est considéré comme le plus important de la modélisation orientée objet. Contrairement au diagramme des cas d'utilisation qui montre le système du point de vue des acteurs, il précise quant à lui la structure interne du système. Il permet de fournir une représentation abstraite des objets du système qui vont interagir ensemble pour réaliser les cas d'utilisation

Pour le concevoir, nous allons d'abord identifier les différentes classes du système.

Identification des classes

Une classe est la description formelle d'un ensemble d'objets ayant une sémantique et des propriétés communes. Un objet est une instance de classe ; C'est une entité discrète dotée d'une identité, d'un état et d'un comportement que l'on peut invoquer. Les objets sont des éléments individuels d'un système en cours d'exécution. Dans notre étude, nous pouvons dégager les classes suivantes :

Utilisateur : Cette classe regroupe des propriétés permettant de décrire les utilisateurs

Produit : Elle contient les propriétés des produits offerts aux clients.

DemandeCredit : C'est dans cette classe que l'on a les propriétés des demandes de prêt.

Credit : Cette classe contient les propriétés qui caractérisent les crédits octroyés qu'ils soient soldés ou non.

Echeance : Les propriétés permettant de décrire les échéances se trouvent dans cette classe.

PersonnePhysique : Elle contient les propriétés des personnes de type physique.

PersonneMorale : Elle contient les propriétés des personnes de type morale.

Client : Cette classe recense les propriétés permettant de décrire les bénéficiaires de crédit.

TypeDemande : C'est dans cette classe qu'on les propriétés des types de demande.

Garantie : Elle contient les propriétés permettant de décrire une garantie de prêt.

TypeGarantie : Cette classe contient les propriétés des types de garantie acceptée.

Fonction : Elle contient les caractéristiques qui décrivent les fonctions occupées par chaque utilisateur du système.

BureauZOne : Dans cette classe on a les propriétés des bureaux de zone où s'effectuent les opérations de crédit.

DroitsUtilisateur : Cette classe contient les propriétés qui caractérisent les droits

DroitsAttribues : Cette classe regroupe les propriétés des droits déjà attribués à certains utilisateurs du système.

Agence : Les propriétés qui caractérisent une agence figurent dans cette classe.

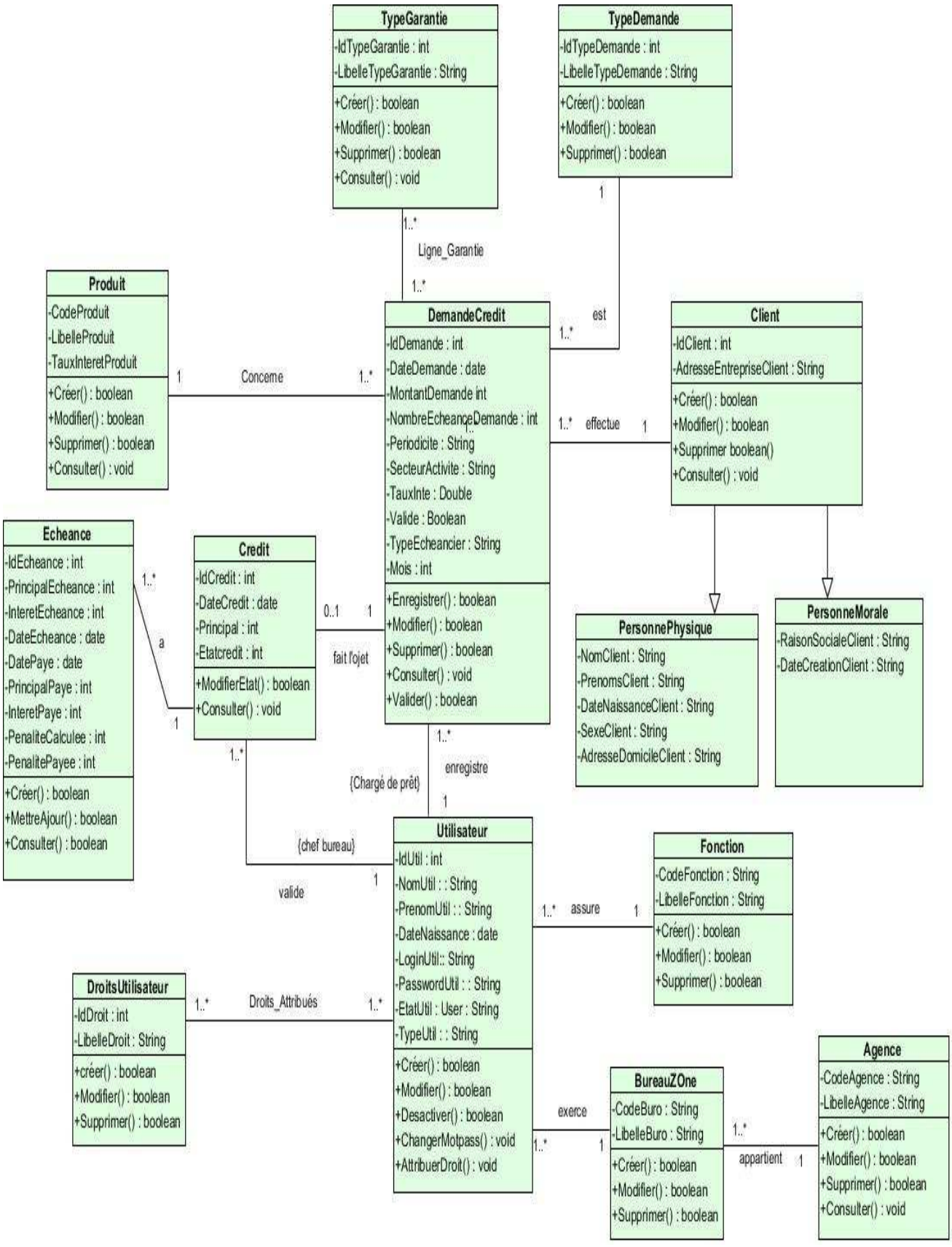


Figure 3 : Diagramme des classes

C- Diagramme des séquences

Le diagramme de séquence représente la succession chronologique des opérations réalisées par un acteur. Il indique les objets que l'acteur va manipuler et les opérations qui font passer d'un objet à l'autre. Même s'il est une variante du diagramme de collaboration, il privilégie la représentation temporelle par rapport à la représentation spatiale et est plus apte à modéliser les aspects dynamiques du système [AUD08]. C'est pourquoi nous avons choisi de développer notre modèle dynamique à l'aide des diagrammes de séquence.

Le diagramme de séquence permet de visualiser les messages par une lecture de haut en bas. L'axe vertical représente le temps et l'axe horizontal les objets qui collaborent.

- *S'authentifier*

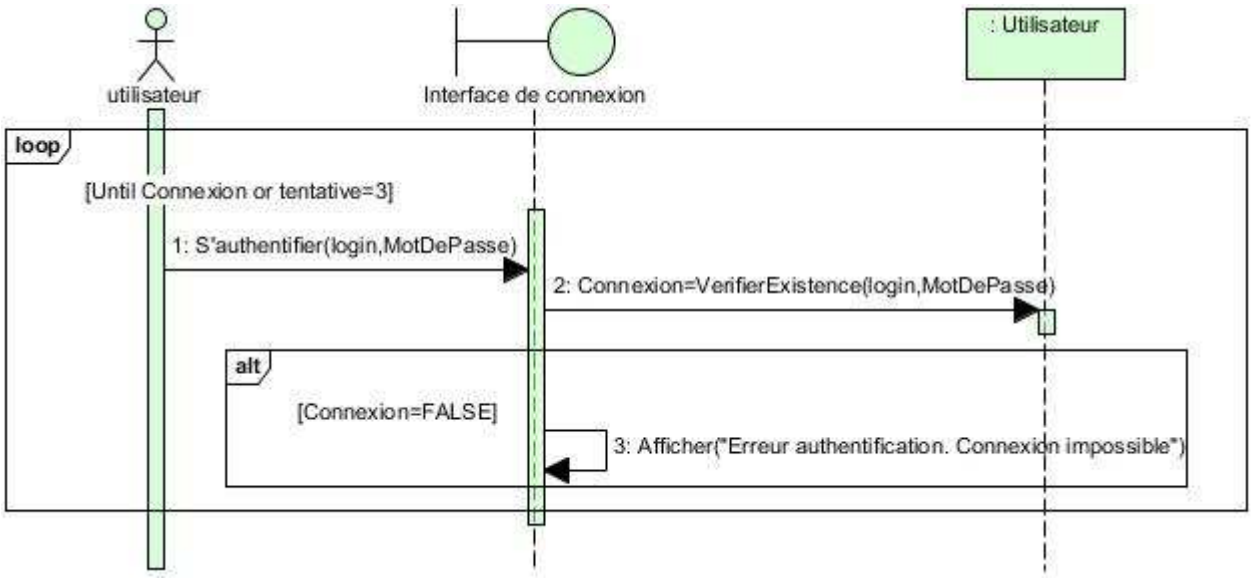
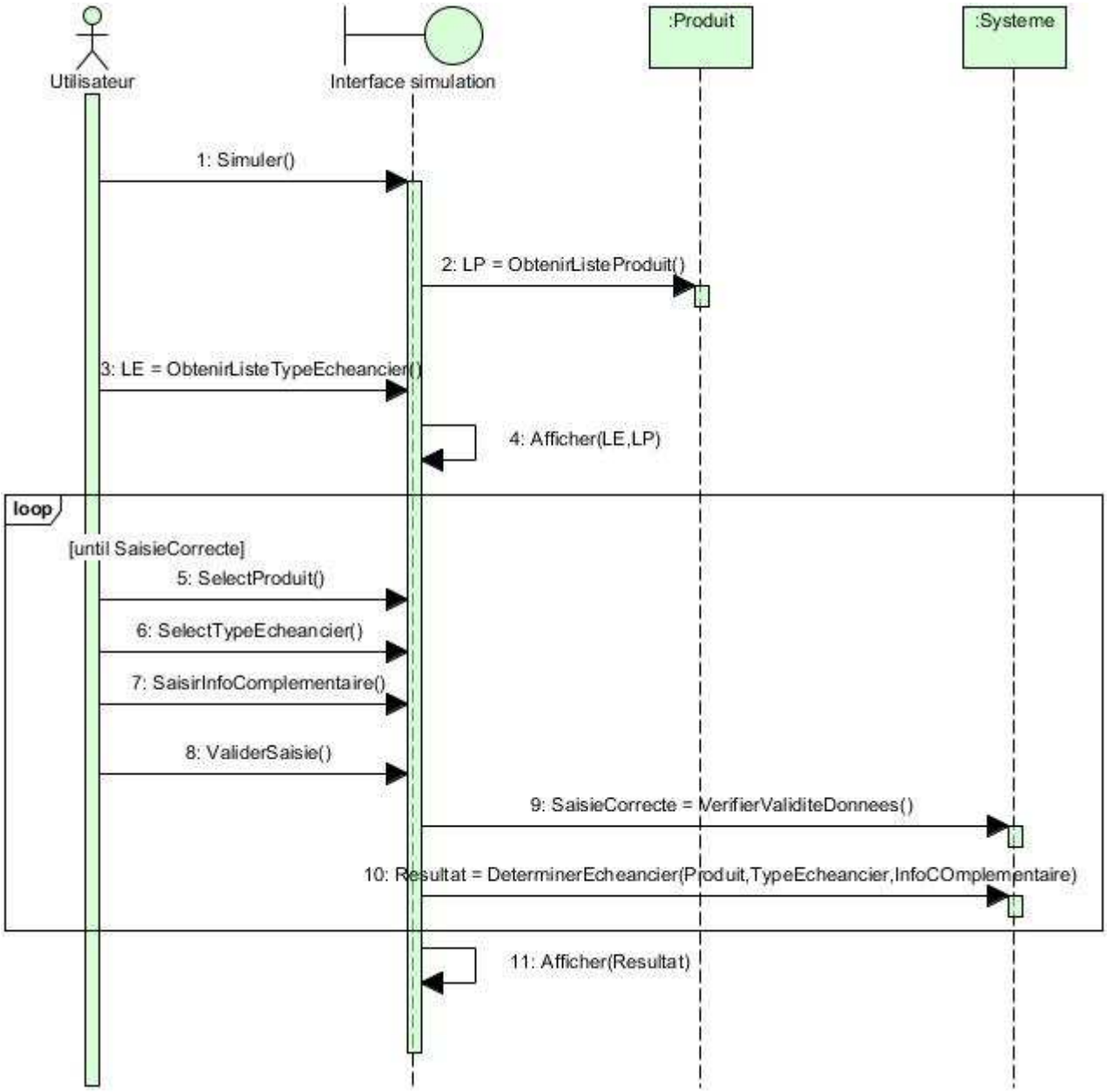


Figure 4 : Diagramme de séquence « S'authentifier »

- *Simuler prêt*



LP=Liste Produits
LE=Liste Echéancier

Figure 5 : Diagramme de séquence « Simuler prêt »

- *Enregistrer_demande*

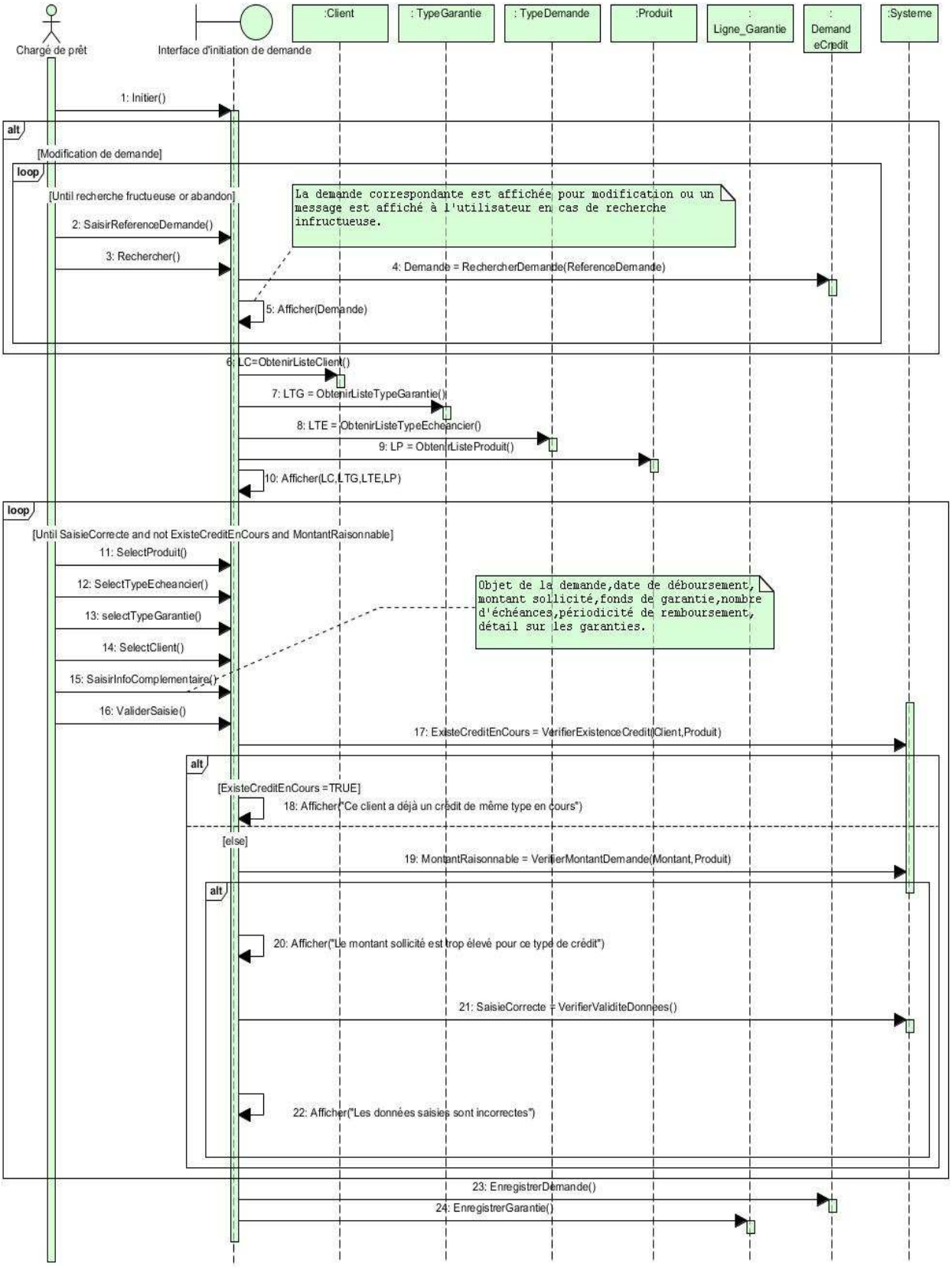
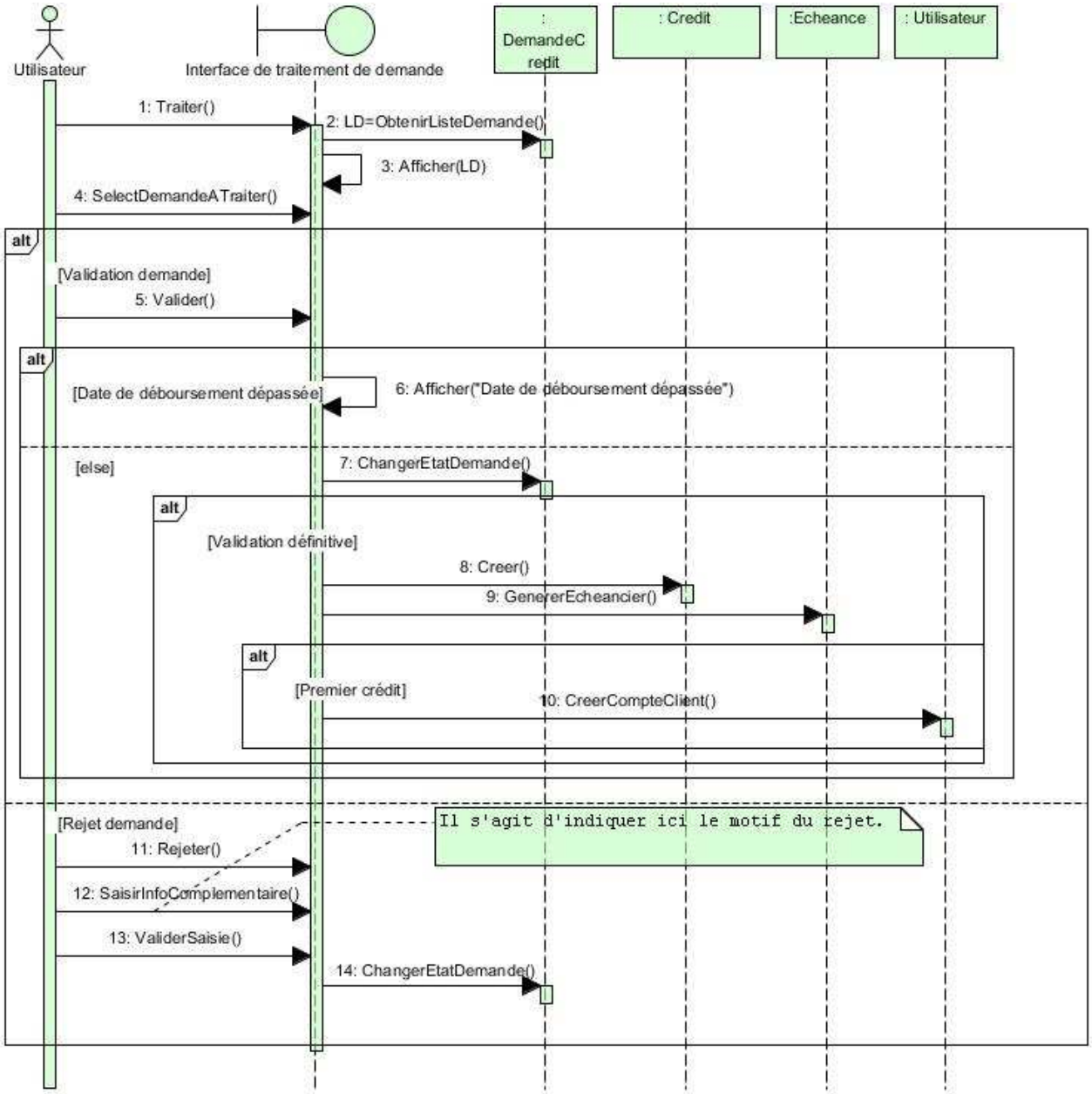


Figure 6 : Diagramme de séquence « Enregistrer demande »

- *Traiter demande*



LD=Liste Demande

Figure 7 : Diagramme de séquence « Traiter demande »

- *Enregistrer client*

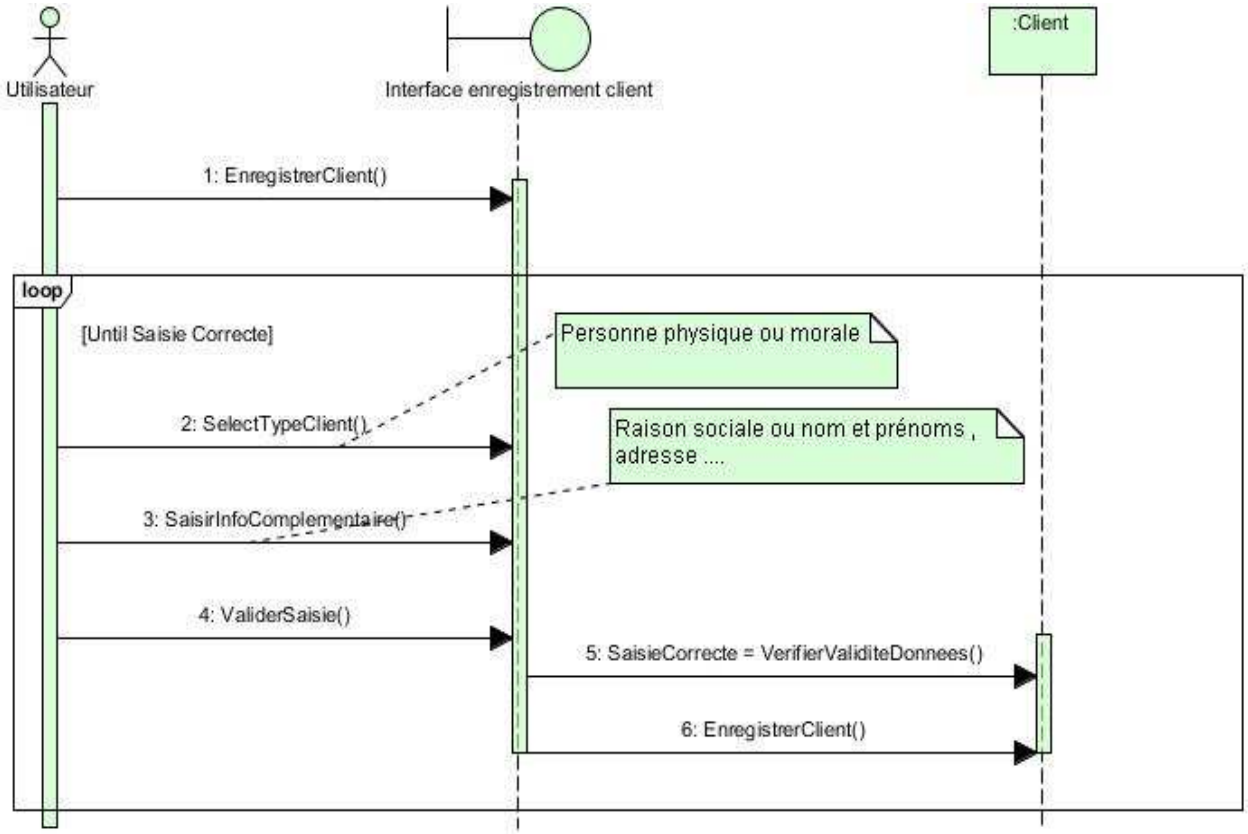


Figure 8 : Diagramme de séquence « Enregistrer client »

- *Consulter_solde*

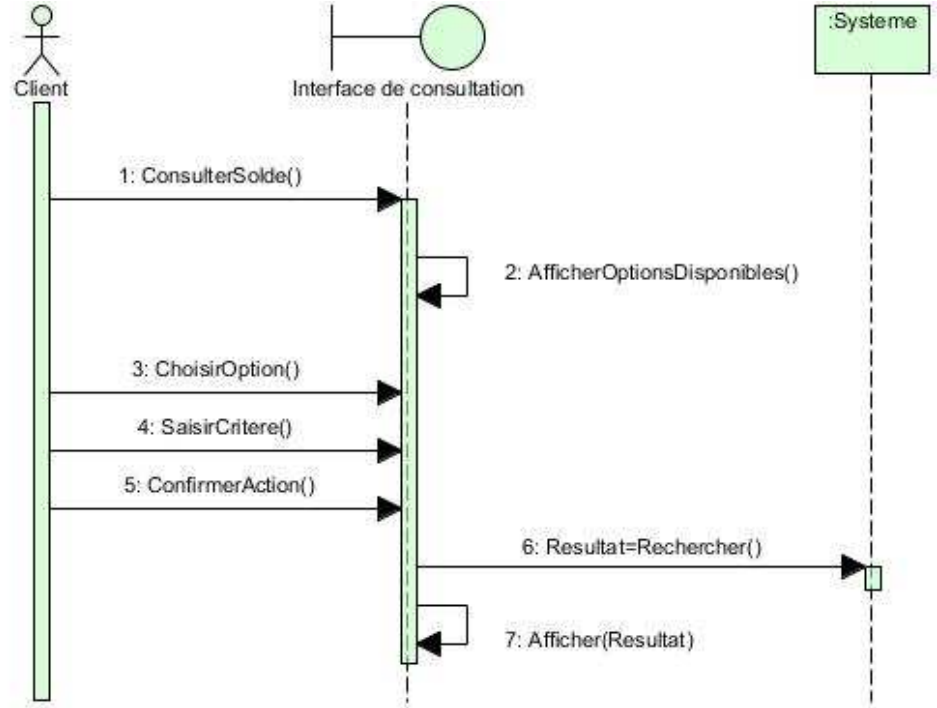


Figure 9 : Diagramme de séquence « Consulter solde »

- *Gérer utilisateur*

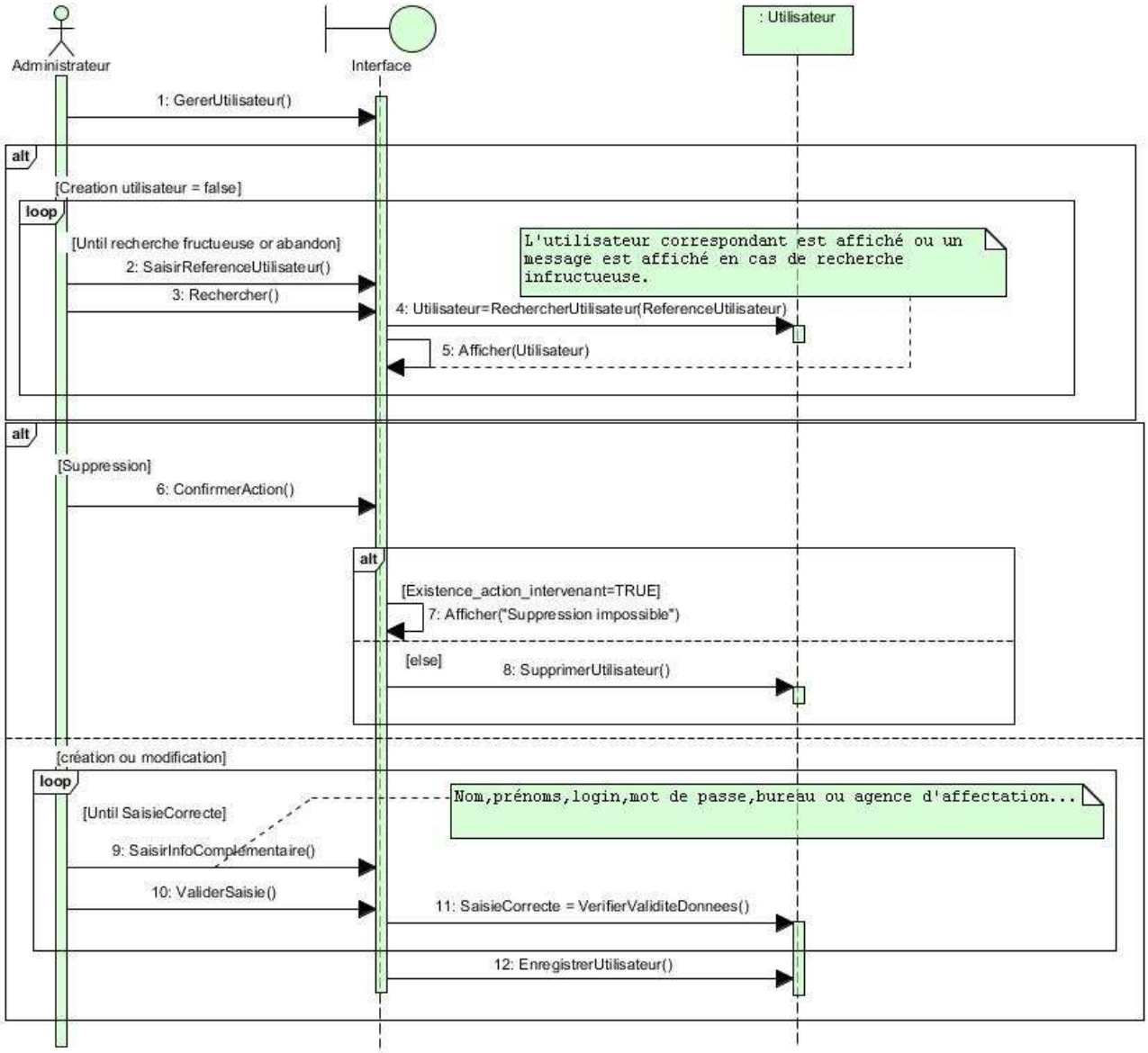


Figure 10 : Diagramme de séquence « Gérer utilisateur »

CONCLUSION

Les résultats attendus en termes de fonctionnalités, de performance, d'extensibilité ont été explorés lors de l'analyse. La conception nous a permis de décrire le fonctionnement de notre système. Ainsi, nous avons une idée claire du système à mettre en place.

Chapitre 4 : Réalisation du système

I- Architecture du système

A- Architecture technique

C'est un système client/serveur n-tiers (n>=3) orientée service: la base de données peut être installée sur un serveur différent du serveur Web. Grâce à l'Internet fourni par les opérateurs GSM, les téléphones intelligents(Smartphones) pourront accéder aux données et consommer les services proposés par les services Web. Il est de même pour l'utilisateur disposant d'un ordinateur PC si celui-ci a accès à Internet.

En effet, les méthodes du service Web seront disponibles à partir de l'application client, l'appel de ces méthodes se fera à partir du Smartphone tandis que leur exécution s'effectuera sur le serveur qui retournera un résultat compréhensible par l'application sur le mobile, ceci grâce au fichier WSDL généré par le service Web et téléchargé par l'application cliente ; l'échange de messages sera assuré par le protocole SOAP.

La figure 11 montre l'architecture de la solution proposée.

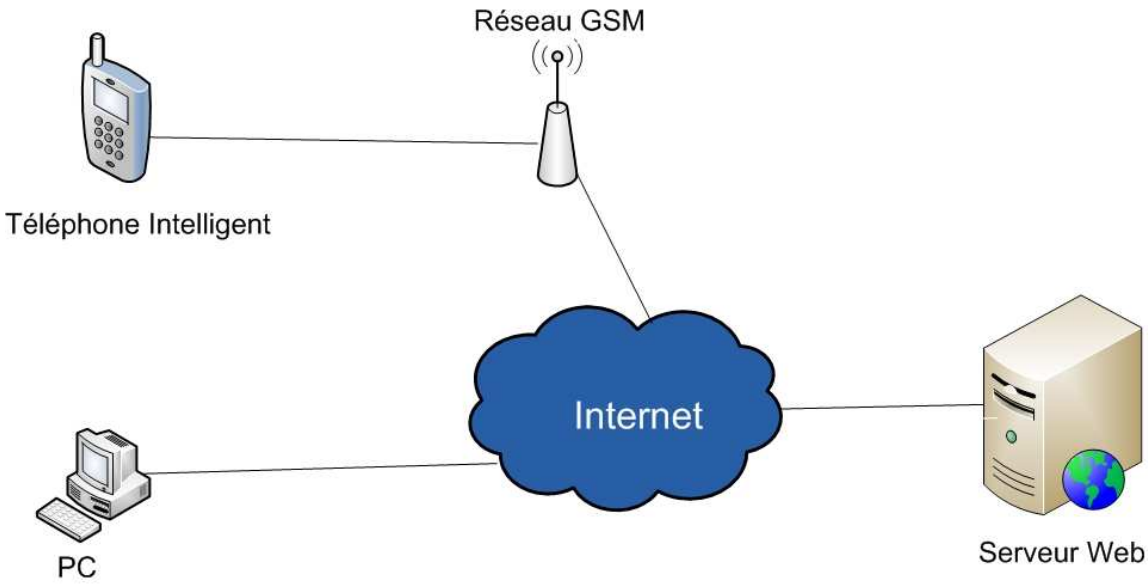


Figure 11 : Architecture du système

B- Architecture logicielle

Du point de vue logiciel, l'architecture repose sur le modèle MVC (Model-View-Controller) qui organise l'interface homme-machine d'une application logicielle en un modèle (objet métier, encore appelé modèle de données), une vue (présentation, interface utilisateur) et un contrôleur (logique de contrôle, gestion des événements, traitement), chacun ayant un rôle précis dans l'interface. Ainsi, le stockage des données est isolé des actions et de la présentation de l'application. Le modèle MVC est utilisé par différents projets JAVA et PHP [WEB10].

En résumé, lorsqu'un client envoie une requête à l'application :

- la requête est analysée par le contrôleur
- le contrôleur demande au modèle approprié d'effectuer les traitements
- le contrôleur renvoie la vue adaptée.

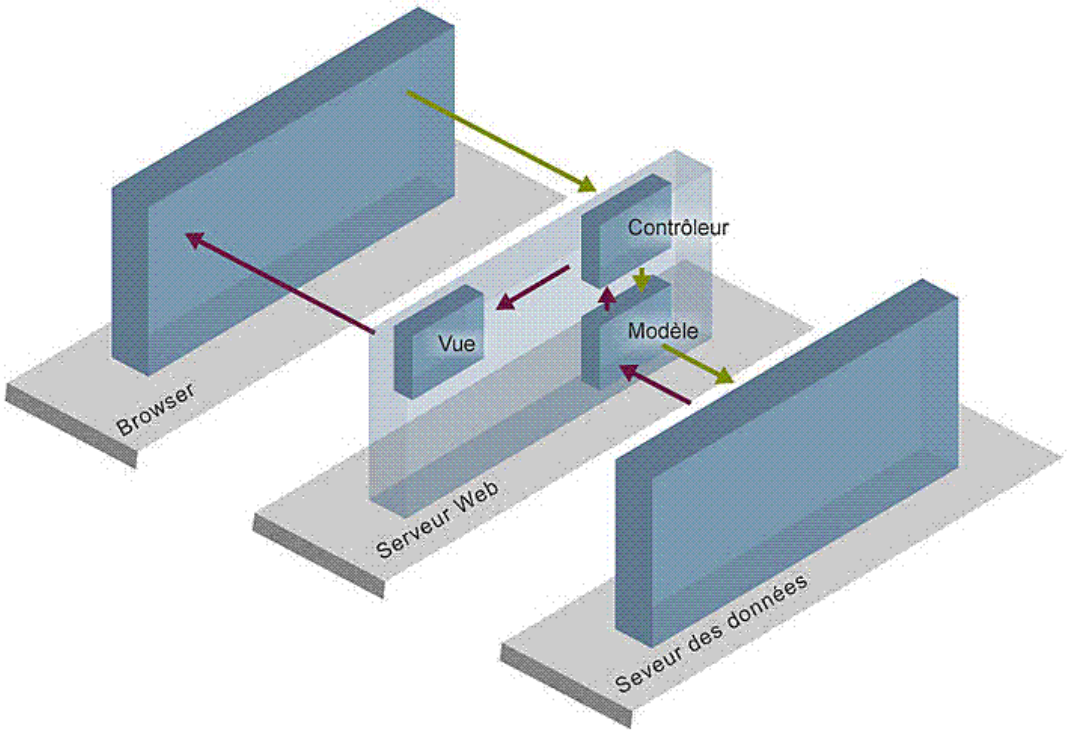


Figure 12 : Architecture MVC

II- Implémentation et simulations

A- Environnement et outils de développement

Les choix techniques utilisés pour le développement sont les suivants :

- ✓ la plate-forme de développement J2ME pour le client mobile dans un Environnement de Développement Intégré (EDI) Netbeans ;
- ✓ pour le client Web, c'est le langage de programmation PHP, dédié à la création des applications Web dynamique, avec Dreamweaver comme environnement de développement qui lui est largement compatible ;
- ✓ les pages écrites en PHP sont testées grâce à la plate-forme wampserver version 2.0 qui inclut tous les outils nécessaires pour le test d'un site Web dynamique à savoir le serveur apache version 2.2.8, MySQL version 5.0.51b et la version phpmyadmin 2.9.1.1.

Pour la connexion à la base de données le mot de passe a été crypté avec l'algorithme MD5. Aussi, la technologie PDO (PHP Data Objects) qui est une extension PHP définissant une interface d'accès à une base de données a été utilisée afin d'éviter les injections SQL [WEB16].

B- Développement et déploiement de l'application

1- Développement

Pour créer les services Web, après avoir mis en place un serveur Apache compilé avec PHP5 et SOAP, nous avons effectué les opérations conformément au modèle MVC.

- ❖ **Modèle**

Dans le modèle nous avons d'abord installé la base de données puis écrit les différentes classes permettant d'interagir avec cette base de données.

Notre base de données étant MySQL, pour l'administrer, nous avons utilisé l'interface d'administration phpmyadmin contenu dans « wamp » que nous avons installé pour déployer notre serveur.

Un exemple de classe permettant d'interagir avec la table « DEMANDE » est présenté en Annexe 1.

❖ **Contrôleur**

A cette étape, nous avons écrit les classes qui sont appelées à échanger avec le modèle puis à restituer les données aux vues. Etant donné que nous utilisons une architecture orientée service, nous devons mettre en place notre serveur SOAP qui doit servir d'intermédiaire entre le modèle et le contrôleur.

En effet, dans un MVC classique, le contrôleur échange directement avec le modèle, ce qui n'est pas le cas dans notre architecture qui elle a un intermédiaire.

➢ **Création du serveur SOAP en PHP5**

La première étape de ce processus est de traduire (réécrire) nos fonctions disponibles et surtout devant être consommées en tant que services par des clients externes dans un fichier dit de description qui a pour extension WSDL (voir un exemple de fichier WSDL en Annexe 2).

La seconde étape consiste à charger les librairies SOAP disponibles dans Apache si elles ne l'étaient par défaut.

En troisième et dernière étape, nous indiquons au serveur SOAP le fichier de description (WSDL) qui mettra à disposition des clients les services qui y sont définis. Un exemple de ce procédé est présenté en annexe 3.

➢ **Création du client SOAP en PHP5 (Client Web)**

Il s'agit ici de créer une instance du client soap en lui indiquant le fichier de description de notre serveur afin de consommer les services qui sont disponibles. Un exemple de ce procédé est présenté en annexe 4.

➢ **Création du client SOAP en Java J2ME (Client mobile)**

Dans le JRE (*Java Runtime Environment*) de Java, il n'existe pas de bibliothèque nous permettant de consommer des services SOAP. Il nous a donc fallu télécharger une bibliothèque appelée ksoap2 qui nous permet de créer l'instance du client

SOAP. Cette librairie comporte un certain nombre de fonctions permettant de créer ou d'analyser un objet SOAP. Un exemple de ce procédé est présenté en Annexe 5.

❖ Vue

A ce stade, nous montrerons quelques exemples de vue et comment ces vues échangent avec leurs contrôleurs.

➢ Cas PHP

Au lancement de notre client PHP, la première fenêtre est celle d'authentification.

Une fois les données d'authentification saisies et validées, elles sont transmises au contrôleur par un appel de fonction comme suit :

```
< ?php
    If(isset($_POST[‘identifiant’])){
       Identifiant($_POST[‘identifiant’], $_POST[‘password’]) ;
    }
?>
```

Si cet appel de fonction du contrôleur nécessite une réponse, alors cette dernière est retournée à notre vue qui la récupère dans une variable.

➢ Cas Java

Le principe est le même que celui du cas PHP. L'appel de la fonction d'identification se présente comme suit :

```
if (command == Valider) {
     this.identification();
}
```

2- Empaquetage et déploiement

Comme nous l'avons souligné ci-haut, nous avons développé un Client mobile et un client Web. Ainsi, l'application pour le client Web est accessible à partir d'un navigateur Web en lui fournissant l'URL utilisée. Pour déployer l'application sur un téléphone intelligent, on utilise un réseau entre ordinateur et téléphone intelligent par connexion USB ou Bluetooth ou autre technologie pour recopier les fichiers Java Archive (.JAR) et Java Application Descriptor (.JAD) (fichiers obtenus après compilation de l'application avec l'IDE Netbeans) vers le téléphone intelligent. La plupart de téléphones mobiles disponibles permettent d'installer une application implémentée en J2ME sous forme de .JAR.

C- Simulations

Nous présentons dans cette partie quelques résultats obtenus.

1- Simulations sur Smartphone

Les figures 13, 14, 15 et 16 présentent quelques résultats obtenus pour le cas d'un Smartphone.

Figure 13 : Menu

Figure 14 : Ecran Simulation échéancier

Figure 15 : Ecran échéancier généré

Figure 16 : Ecran Enregistrer demande de prêt

2- Simulations sur PC

Pour l'application Web, les figures 17 et 18 montrent respectivement le résultat de l'implémentation des cas d'utilisation s'authentifier et enregistrer client.

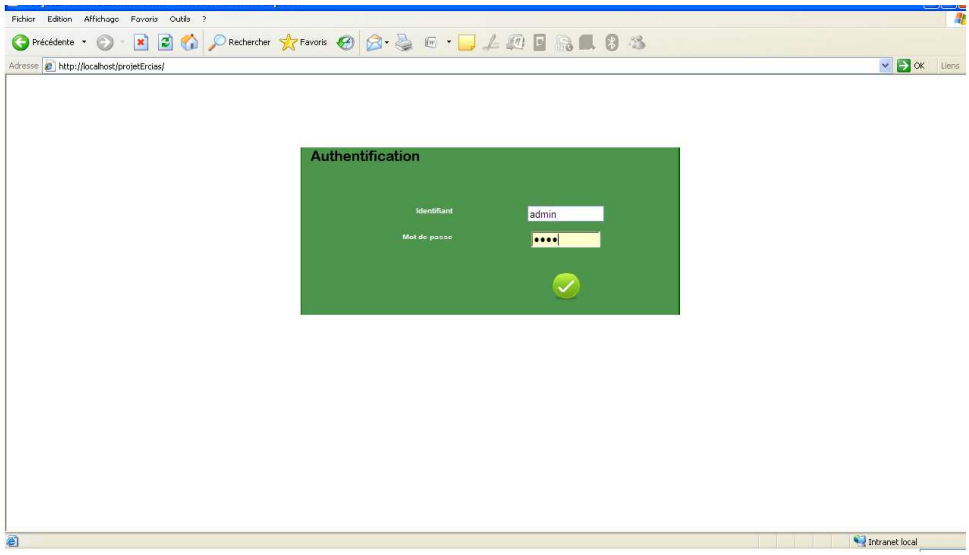


Figure 17 : Ecran S'authentifier

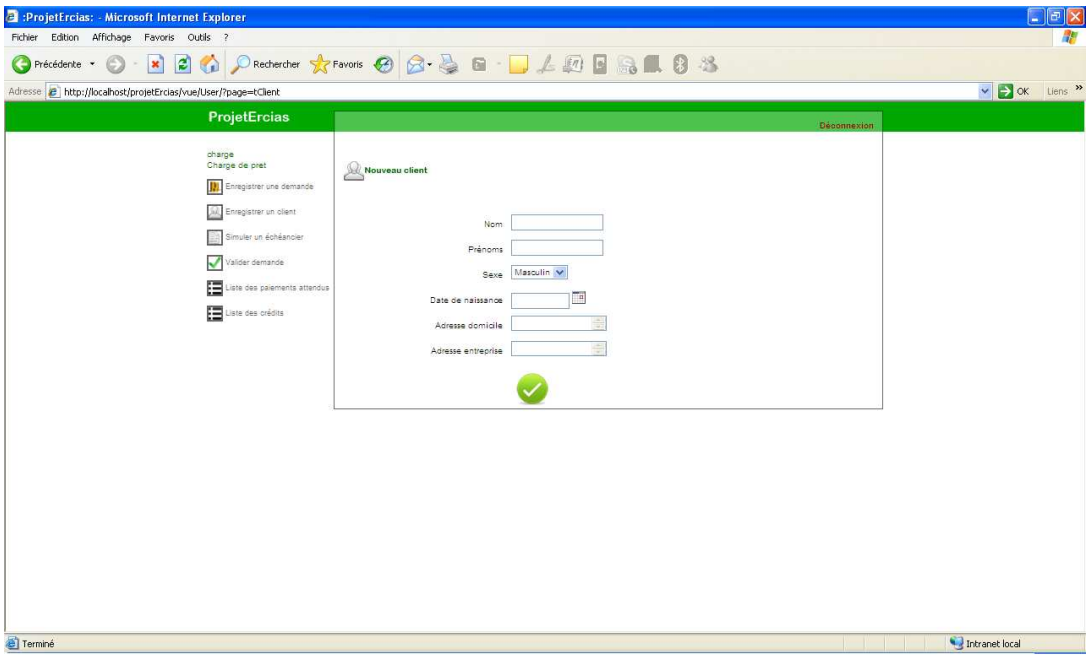


Figure 18 : Ecran Enregistrer client

Conclusion et perspectives

Au terme de notre projet dont l'objectif principal est d'ouvrir la porte de l'informatique mobile aux institutions de microfinance en leur mettant en place une solution applicative pour l'accès à certaines fonctionnalités de l'application de gestion de crédits via Smartphone, nous pouvons affirmer que les résultats obtenus sont encourageants. Le prototype d'application proposé et développé a été testé et a affiché des résultats qui augurent une suite meilleure.

L'élaboration de ce projet nous a permis d'approfondir nos connaissances en informatique et surtout dans le domaine du développement d'applications destinées aux périphériques mobiles : la technologie relative aux services Web et à J2ME a été expérimentée.

Certes, des améliorations restent à intégrer. En perspectives nous pouvons améliorer notre application en termes de sécurité car bien que nécessitant une authentification, les messages SOAP transmis sur le protocole http ne sont pas cryptés, ce qui constitue une faiblesse non négligeable. Aussi, reste t-il à prendre en compte la synchronisation afin de pouvoir aussi travailler en mode déconnecté.

Bibliographie

[AUD08] : Laurent AUDIBERT, UML 2, Institut Universitaire de Technologie de Villetaneuse - Département Informatique, Édition 2007-2008

[MEN09] : Joël MENGOLO ASSAMBA, Refonte de la plate-forme architecturale du patrimoine applicatif, Mémoire d'ingénieur, IAI GABON, 2009.

[ROQ02] : Pascal ROQUES et Franck VALLEE, UML en action, édition Eyrolles, 2002, Paris, 388 pages.

[WEB01] : http://ictupdate.cta.int/fr/content/download/3470/35945/file/ICTupdate_36_pf+FR+lowres.pdf du 14/09/2010

[WEB02] : http://www.gfi.fr/gfilabs/common/docs/Offre-Technologique-Informatique-Mobile.pdf du 20/01/2011

[WEB03] : http://www-smis.inria.fr/dataFiles/P03a.pdf du 02/11/2010

[WEB04] : http://www.clusif.asso.fr/fr/production/ouvrages/pdf/SecuriteMobilite.pdf du 11/02/2010

[WEB05] : http://mrproof.blogspot.com/2011/03/larchitecture-client-serveur.html du 15/02/2011

[WEB06] : http://fr.wikipedia.org/wiki/Architecture_orientee_services du 15/01/2011

[WEB07] : http://www2.ifi.auf.org/rapports/tpe-promo12/tpe-nguyen_van_tien.pdf

[WEB08] : http://www.mysql.fr/ du 15/09/2010

[WEB09] : http://www.dotnetguru.org/articles/J2MEvsSDE/J2MEvsSDE.htm du 25/11/2010

[WEB10] : http://notes.mazenod.fr/le-design-pattern-mvc1.html du 12/10/2010

[WEB11] : http://netbeans.org/index_fr.html du 13/09/2010

[WEB12] : http://www.adobe.com/fr/products/dreamweaver.html du 16/09/2010

[WEB13] : http://www.etudionet.com/v0/communaute/xuser/etudionet/docs/3ACHOURI_Rached.pdfdu 22 /03/2011

[WEB14] : http://fr.wikipedia.org/wiki/Architecture_logicielle du 22/02/2011

[WEB15] : http://www.3ie.fr/nouvelles_technologies/fiche/fiche_J2ME.htm du 16/03/2011

[WEB16] : http://www.commentcamarche.net/faq/27489-pdo-une-autre-facon-d-acceder-a-vos-bases-de-donnees du 16/12/2010

[WEB17] : http://www.guideinformatique.com/fiche-telephone_mobile-735.htm du 27/03/2011

[WEB18] : livre blanc II sur la mobilité/sécurité édité en 2006 : http://www.chambre-monegasque-nouvelles-technologies.net/ du 12/01/2011

Annexes

Annexe 1 : Exemple de classe modèle

Annexe 2 : Exemple de fichier WSDL

Annexe 3 : Extrait de code pour le serveur SOAP en PHP5

Annexe 4 : Extrait de code pour le client SOAP en PHP5

Annexe 5 : Extrait de code pour le client SOAP en J2ME

Annexe 1 : Exemple de classe modèle

```
<?php
class DemandeCreditTable
{
    /*
            *Requêtes préparées
            */
            private $selectAll;
    private $insert;
    private $updateState;

    public function __construct()
    {
        require("connexion.php");
        $this->selectAll=$db->prepare("SELECT idDemande, codeProduit, idTypeDemande, idClient,
                idUser, secteurActivite, montantDemande, periodicite, nombreEcheanceDemande,
                typeecheancier, mois, dateDemande, valide FROM demande");

        $this->updateState=$db->prepare("UPDATE demande SET valide = ? WHERE idDemande=?");
     }

     public function insert($codeProduit, $idTypeDemande, $idClient, $idUser, $secteurActivite,
                    $montantDemande, $periodicite, $nombreEcheanceDemande, $typeecheancier, $mois,
                    $dateDemande, $valide)
    {
          require("connexion.php");
          $this->insert=$db->prepare("INSERT INTO demande (codeProduit, idTypeDemande, idClient,
                    idUser, secteurActivite, montantDemande, periodicite, nombreEcheanceDemande,
                    typeecheancier, mois, dateDemande, valide) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)");
        $this->insert->execute(array($codeProduit, $idTypeDemande, $idClient, $idUser, $secteurActivite,
                $montantDemande, $periodicite, $nombreEcheanceDemande, $typeecheancier, $mois,
                $dateDemande, $valide));
        return $db->lastInsertId();
     }

            public function updateState($valide, $idDemande)
    {
        $this->updateState->execute(array($valide, $idDemande));
        return $db->rowCount();
     }

     public function selectAll()
     {
        $this->selectAll->execute();
        return $this->selectAll->fetchAll();
     }
}
?>
```

Annexe 2 : Exemple de fichier WSDL

```
<?xml version="1.0"?>
<!-- partie 1 : Definitions -->
<definitions name="ProjetErcias"
        targetNamespace="urn:ProjetErcias"
        xmlns:typens="urn:ProjetErcias"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema"
        xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
        xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
        xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
        xmlns="http://schemas.xmlsoap.org/wsdl/">

<!-- partie 2 : Types-->
  <types>
    <xsd:schema xmlns="http://www.w3.org/2001/XMLSchema" targetNamespace="urn:ProjetErcias/">
        <xsd:complexType name="clients">
            <xsd:sequence>
                <xsd:element name="idClient" type="xs:string" minOccurs="0"/>
                <xsd:element name="dateNaissanceClient" type="xs:string" minOccurs="0"/>
                <xsd:element name="adresseEntrepriseClient" type="xs:string" minOccurs="0"/>
                <xsd:element name="nomClient" type="xs:string" minOccurs="0"/>
                <xsd:element name="prenomsClient" type="xs:string" minOccurs="0"/>
                <xsd:element name="sexeClient" type="xs:string" minOccurs="0"/>
                <xsd:element name="adresseDomicileClient" type="xs:string" minOccurs="0"/>
                <xsd:element name="raisonSocialeClient" type="xs:string" minOccurs="0"/>
                <xsd:element name="dateCreationClient" type="xs:string" minOccurs="0"/>
             </xsd:sequence>
        </xsd:complexType>
        <xsd:complexType name="tableauClients" final="#all">
            <xsd:sequence>
                <xsd:element name="item" type="tns:clients" minOccurs="0" axOccurs="unbounded"
                nillable="true"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:schema>
  </types>

<!-- partie 3 : Message -->
        <message name="nouveauClientParticulierRequest">
            <part name="dateNaissanceClient" type="xsd:string"/>
            <part name="adresseEntrepriseClient" type="xsd:string"/>
            <part name="nomClient" type="xsd:string"/>
            <part name="prenomsClient" type="xsd:string"/>
            <part name="sexeClient" type="xsd:string"/>
            <part name="adresseDomicileClient" type="xsd:string"/>
        </message>
        <message name="nouveauClientParticulierResponse">
            <part name="return" type="xsd:string"/>
        </message>
        <message name="nouveauClientMoraleRequest">
            <part name="raisonSocialeClient" type="xsd:string"/>
            <part name="adresseEntrepriseClient" type="xsd:string"/>
            <part name="dateCreationClient" type="xsd:string"/>
        </message>
        <message name="nouveauClientMoraleResponse">
            <part name="return" type="xsd:string"/>
        </message>
```

```
<!-- partie 4 : Port Type -->
        <portType name="ProjetErciasPort">
<!-- partie 5 : Operation -->
                </operation>
                      <operation name="enregistrementParticulier">
                       <input message="typens:nouveauClientParticulierRequest"/>
                       <output message="typens:nouveauClientParticulierResponse"/>
                </operation>
                                              <operation name="enregistrementMorale">
                       <input message="typens:nouveauClientMoraleRequest"/>
                       <output message="typens:nouveauClientMoraleResponse"/>
                </operation>
        </portType>
<!-- partie 6 : Binding -->
        <binding name="ProjetErciasBinding" type="typens:ProjetErciasPort">
                <soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>

                <operation name="enregistrementParticulier">
                        <soap:operation soapAction="ClientAction"/>
                        <input name="nouveauClientParticulierRequest">
                                <soap:body use="encoded"
                                namespace="urn:Client"
                                encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
                        </input>
                        <output name="nouveauClientParticulierResponse">
                                <soap:body use="encoded"
                                namespace="urn:Client"
                                encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
                        </output>
                </operation>

                <operation name="enregistrementMorale">
                        <soap:operation soapAction="ClientAction"/>
                        <input name="nouveauClientMoraleRequest">
                                <soap:body use="encoded"
                                namespace="urn:Client"
                                encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
                        </input>
                        <output name="nouveauClientMoraleResponse">
                                <soap:body use="encoded"
                                namespace="urn:Client"
                                encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
                        </output>
                </operation>
        </binding>
<!-- partie 7 : Service -->
        <service name="ProjetErciasService">
                <documentation>Retourne une phrase simple </documentation>
<!-- partie 8 : Port -->
                <port name="ProjetErciasPort" binding="typens:ProjetErciasBinding">
                        <soap:address location="http://127.0.0.1/projetErcias/controleur/serveur.php"/>
                </port>
        </service>
</definitions>
```

Annexe 3 : Extrait de code pour le serveur SOAP en PHP5

```
<?php
include("../modele/User.php");
include("../modele/Client.php");
include("../modele/Credit.php");
include("../modele/DemandeCredit.php");

// première étape : désactiver le cache lors de la phase de test
ini_set("soap.wsdl_cache_enabled", "0");

// on indique au serveur à quel fichier de description il est lié
$serveurSOAP = new SoapServer('serveur.wsdl');

// ajouter les fonctions de la classe user au serveur
try
{
$serveurSOAP->addFunction('enregistrementParticulier');
$serveurSOAP->addFunction('enregistrementMorale');
$serveurSOAP->addFunction('selectionClients');
$serveurSOAP->addFunction('selectionGaranties');
$serveurSOAP->addFunction('enregistrementDemande');
$serveurSOAP->addFunction('enregistrementUser');
$serveurSOAP->addFunction('selectionUsers');
$serveurSOAP->addFunction('modifierEtatUser');
$serveurSOAP->addFunction('selectionDemandes');
$serveurSOAP->addFunction('selectionTypeDemandeParId');
$serveurSOAP->addFunction('selectionClientParId');
$serveurSOAP->addFunction('selectionUserParId');
$serveurSOAP->addFunction('enregistrementCredit');
$serveurSOAP->addFunction('modifierEtatDemande');
$serveurSOAP->addFunction('validerDemande');
$serveurSOAP->addFunction('selectionCredits');
//$serveurSOAPUser->setClass('User');
}
catch
(Exception $e)
{
echo $e->getMessage();
}
// lancer le serveur
if ($_SERVER['REQUEST_METHOD'] == 'POST')
{
$serveurSOAP->handle();
}
else
{
echo 'désolé, je ne comprends pas les requêtes GET, veuillez seulement utiliser POST';
}
?>
```

Annexe 4 : Extrait de code pour le client SOAP en PHP5

```
function validerDemande($vId, $vProduit, $vMontant, $vPeriodicite, $vEcheance,
$vTypeecheancier, $vMois)
  {
  $clientSOAP = new SoapClient('../controleur/serveur.wsdl');

  $date = getdate();
  try{
    $idCredit = $clientSOAP->enregistrementCredit($date['year']."-".$date['mon']."-
                ".$date['mday'], $vMontant, 1, $vId);
    $resultat2 = $clientSOAP->modifierEtatDemande(1, $vId);
  }
  catch (SoapFault $exception)
  {
          echo 'EXCEPTION='.$exception;
  }

     switch ($vPeriodicite){
          case "journalier":
          $periodicite=0.033;
          $jour=1;
          break;
          case "hebdomadaire":
          $periodicite=0.23;
          $jour=7;
          break;
          case "mensuel":
          $periodicite=1;
          $jour=30;
          break;
          case "trimestriel":
          $periodicite=3;
          $jour=90;
          break;
          case "semestriel":
          $periodicite=6;
          $jour=180;
          break;
          case "annuel":
          $periodicite=12;
          $jour=360;
          break;
     }

  $tauxMensuel=$vProduit/100;
  $tauxEquivalent=bcpow((1+$tauxMensuel),$periodicite,2)-1;
  $echeance=$vEcheance;
  if ($vMois==""){
     $mois=0;
  }else{
     $mois=$vMois;
  }
  $montant=$vMontant;
  $principal=0;

  if ($vTypeecheancier=="simple"){
     $constant=ceil(($montant * $tauxMensuel)/(1-bcpow((1/(1+$tauxMensuel)),$echeance,10)));
          $dateEcheance = $date['year']."-".$date['mon']."-".$date['mday'];
     for($i=0;$i<$echeance;$i++){
```

```
                    $montant=$montant-$principal;
                    if(($i+1)!=$echeance){
                    $interet=floor($montant*$tauxMensuel);
                    }else{
                    $interet=$constant-$montant;
                    }
                    $principal=$constant-$interet;

                    $dateEcheance = strtotime ( "+$jour day" , strtotime ( $dateEcheance ) ) ;
        $dateEcheance = date ( 'Y-m-j' , $dateEcheance );
                    try{
           $resultat3 = $clientSOAP->enregistrementEcheance($principal,$interet,$dateEcheance,
                        0,0,0,0,0,$idCredit);
        }
        catch (SoapFault $ex)
        {
                echo 'EXCEPTION='.$ex;
        }
            }
}
if ($vTypeecheancier=="sansinteret"){
    $interetDiff=$montant*$tauxMensuel*$mois;
    $constInt=round(($interetDiff * $tauxMensuel)/(1-bcpow((1/(1+$tauxMensuel)),$echeance,10)));
    $constant=ceil(($montant * $tauxMensuel)/(1-bcpow((1/(1+$tauxMensuel)),$echeance,10)));
          $dateEcheance = $date['year']."-".$date['mon']."-".$date['mday'];
    for($i=0;$i<$echeance;$i++){
                    $montant=$montant-$principal;
              if(($i+1)<$echeance){
                    $interet=floor($montant*$tauxMensuel)+$constInt;
                    }else{
                       if(($i+1)==$echeance){
                            $interet=$constant-$montant;
                         }else{
                            $interet=0;
                         }
                    }
                    $principal=$constant-(round($montant*$tauxMensuel));

                    $dateEcheance = strtotime ( "+$jour day" , strtotime ( $dateEcheance ) ) ;
        $dateEcheance = date ( 'Y-m-j' , $dateEcheance );
                    try{
           $resultat3 = $clientSOAP->enregistrementEcheance($principal,$interet,$dateEcheance,
                        0,0,0,0,0,$idCredit);
        }
        catch (SoapFault $ex)
        {
                echo 'EXCEPTION='.$ex;
        }
            }
}
if ($vTypeecheancier=="avecinteret"){
    $constant=ceil(($montant * $tauxMensuel)/(1-bcpow((1/(1+$tauxMensuel)),$echeance,10)));
          $dateEcheance = $date['year']."-".$date['mon']."-".$date['mday'];
    for($i=0;$i<($echeance+$mois);$i++){
                    $montant=$montant-$principal;
                    if($mois>=$i+1){
                    $constant2=floor($montant*$tauxMensuel);
              }else{
                       if(($echeance+$mois)>=$i+1){
                            $constant2=$constant;
```

```
                        }else{
                        $constant2=0;
                        }
                }
            if(($i+1)<($echeance+$mois)){
                $interet=floor($montant*$tauxMensuel);
                }else{
                   if(($i+1)==($echeance+$mois)){
                        $interet=$constant2-$montant;
                    }else{
                        $interet=0;
                    }
                }
                $principal=$constant2-$interet;
        $dateEcheance = strtotime ( "+$jour day" , strtotime ( $dateEcheance ) ) ;
        $dateEcheance = date ( 'Y-m-j' , $dateEcheance );
                try{
           $resultat3 = $clientSOAP->enregistrementEcheance($principal,$interet,$dateEcheance,
                0,0,0,0,0,$idCredit);
        }
        catch (SoapFault $ex)
        {
                echo 'EXCEPTION='.$ex;
        }
            }
    }
    return 0;
    }
```

Annexe 5 : Extrait de code pour le client SOAP en J2ME

```
package Classes;
```

```
import java.io.IOException;
import java.util.Vector;
import org.ksoap2.serialization.SoapObject;
import org.ksoap2.serialization.SoapSerializationEnvelope;
import org.ksoap2.transport.HttpTransport;
import org.xmlpull.v1.XmlPullParserException;
/**
 *
  */
   public class ConnexionSoap {
   public Object resultatRequeteSOAP = null;
   public SoapObject retour = null;
   public Object[] retourTab = null;
   public Object[][] retourTabDouble = null;
   public SoapObject objetSOAP;
   public Vector vector;
   private HttpTransport connexionServeur = null;
   private SoapSerializationEnvelope envelope = null ;

    public HttpTransport getConnexionServeur() {
        return connexionServeur;
    }

   // nom du service
   private String nomService = "";
   // url du service
   private String urlService = "";
   // méthode du service
   private String methodeChoisie = "";
   // constructeur soap
   public  ConnexionSoap (String url){
         this.urlService = url;
        try{   // etape 1 création du module de connexion HTTP
            this.connexionServeur = new HttpTransport ( this.urlService );
            this.connexionServeur.debug = true;
         }catch(Exception e){
           e.printStackTrace ();
         }
    // return connexionServeur;
   }
   // création de l'objet SOAP et retourner l'objet
   public  void objetSoap(String nomService, String methodeChoisie, Vector arg){

        this.nomService = nomService;
        this.methodeChoisie = methodeChoisie;
        try{
           //création objet SOAP
          this.objetSOAP = new SoapObject (nomService, methodeChoisie );
          //ajout des proprietés pour cette méthode
          if(arg != null){
           for(int i=0; i<arg.size(); i++)
           this.objetSOAP.addProperty("arg"+i, arg.elementAt(i));
          }
        }catch(Exception e){
           e.printStackTrace ();
        }
      //  return objetSOAP;
   }
package Principal;
```

```
import Classes.ConnexionSoap;
import java.util.*;
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
import org.netbeans.microedition.lcdui.SimpleTableModel;
import org.netbeans.microedition.lcdui.SplashScreen;
import org.netbeans.microedition.lcdui.TableItem;

/**
 * @author Erlos
 */
public class ProjetErcias extends MIDlet implements CommandListener, ItemCommandListener {

   private boolean midletPaused = false, droit13=false, droit14=false, droit15=false, droit17=false,
                                        droit18=false, droit19=false, droit25=false;
   private ConnexionSoap cs = new ConnexionSoap("http://127.0.0.1/projetErcias/controleur/serveur.php");
   private String varId, varIdentifiant="", varNom, varPrenom, varCodeFonction, varCodeBureau,
           Message="", NumTemp="", IdTemp="", titreTableau, varCredit, numEcheance, idEcheance;
   private double vPeriodicite, vTaux, vTauxEquivalent;
   private String[][] tableEcheanceData = null, tableCreditsData = null, tableEcheancesData = null;
 public void envelope(){
      try{

       // création d'un objet qui contiendra nos propriétés
       this.envelope = new SoapSerializationEnvelope(SoapSerializationEnvelope.VER11);
       this.envelope.bodyOut = this.objetSOAP;
       // création de l'objet SOAP ok
       }catch(Exception e){
         e.printStackTrace ();
       }
      //return this.envelope;
  }
   public SoapSerializationEnvelope getEnvelope() {
      return envelope;
   }
  public Object resultatRequeteSOAP()  {

      // connexion au serveur
       try
           {
             // invoquation de la méthode sur le serveur
            this.connexionServeur.call(null, this.envelope);
             // recuperation de la réponse du serveur
            this.resultatRequeteSOAP = this.envelope.getResponse();
           }catch (Exception aE)
           {
           aE.printStackTrace();
           }
      return this.resultatRequeteSOAP;
  }
}
 }
```