République Algérienne Démocratique et Populaire

Ministère de L'enseignement Supérieur et de la Recherche Scientifique

UNIVERSITE ABDERAHMANE MIRA-BEJAIA

Faculté des Sciences Exactes

Département D'Informatique

Mémoire de fin de cycle

En vue de l'obtention du diplôme de Master en Informatique

Option : Administration et Sécurité des Réseaux

Thème

Conception et Implémentation d'une application web permettant la collecte d'informations des 48 wilayas du pays

Encadré par : M. DEMOUCHE

Devant le jury:

Président : Amroun Kamal
Examinateur : Salhi Nadir

Réalisé par :

- ✓ MEHRAZI Fahem
- ✓ MEDJKOUNE Yanis

Promotion 2014/2015

REMERCIEMENTS

Nous tenons à remercier en premier lieu le bon Dieu de nous avoir donné la force et le courage pour accomplir ce modeste travail.

Notre profonde gratitude et sincères remerciements vont à notre promoteur Mr. Demouche Mouloud pour nous avoir confié ce travail, pour son suivi, sa disponibilité, son orientation et ses conseils.

Nos remerciements vont aussi aux membres de jury qui ont accepté de juger notre travail.

A tout nos enseignants et les membres du département informatique de l'université
ABDERAHMANE MIRA.

Nous n'oublions pas de remercier, vivement, tous ceux qui ont contribué, de près ou de loin à la réalisation de ce mémoire de fin d'étude.

DÉDICACES

Ce modeste travail est dédié :

A nos chers parents qui nous ont soutenus et encouragés durant toute notre scolarité.

A nos frères et sœurs.

A nos amis(e).

A nos enseignants.

A toutes les personnes qui nous ont apportés de l'aide.

Table des matières

Chapitre 1

Etude préliminaire

Chapitre 2

Conception

Chapitre 3

Implémentation

Liste des figures

Liste des tableaux

Liste des abréviations

SQL: **S**tructured **Q**uery **L**anguage

CD: **C**ompact **D**isc

DVD: **D**igital **V**ersatile **D**isc

ISO **I**nternational **S**tandards **O**rganization

BLOB: **B**inary **L**arge **O**bjet

ANSI: **A**merican **N**ational and **S**tandards **I**nstitute

SPARC: **S**calable **P**rocessor **A**rchitecture

SGBD : **S**ystème de **G**estion de **B**ase de **D**onnées

SGBDR : **S**ystème de **G**estion de **B**ase de **D**onnées **R**elationnelle

MySQL: **M**y **S**tructured **Q**uery **L**anguage

DBA: **D**ata **B**ase **A**dministrator

DDL: Data **D**efinition **L**anguage

LMD : **L**angage de **M**anipulation des **D**onnées

DML : **D**ata **M**anipulation **L**anguage

LDD : **L**angage de **D**éfinition des **D**onnées

LCD : **L**angage de **C**ontrôle des **D**onnées

DCL : **D**ata **C**ontrol **L**anguage

MCD : **M**odèle **C**onceptuel de **D**onnées

MLD : **M**odèle **L**ogique de **D**onnées

MPD : **M**odèle **P**hysique de **D**onnées

OO : **O**rienté **O**bjet

DCU : **D**iagramme de **C**as d'**U**tilisation

DET : **D**iagramme d'**E**tat-**T**ransition

DAC : **D**iagramme d'**A**ctivité

DSE : **D**iagramme de **S**équence

DCO : **D**iagramme de **C**ommunication

DGI : **D**iagramme **G**lobal d'**I**nteraction

DTP : **D**iagramme de **T**emps

UP: Unified **P**rocess

GUI: **G**raphical **U**ser **I**nterface

IDE: **I**ntegrated **D**evelopment **E**nvironment

SWT: **S**tandard **W**idget **T**oolkit

IBM: International **B**isness **M**achine

OLAP: **O**n-**L**ine **A**nalytical **P**rocessing

OLTP: On-**L**ine **T**ransactional **P**rocessing

Introduction Générale

L'Algérie se situe au Nord-Ouest du continent africain. Bordée au Nord sur 1200 km par la mer méditerranéenne ; son littoral occupe plus de la moitié du rivage méridional occidental. Par sa superficie de (2.381.741 km²), elle occupe la seconde place en Afrique. Les 4/5ème du territoire sont occupés par le Sahara.

Les collectivités territoriales algériennes sont les communes et les wilayas, et depuis 1984, l'Algérie est composée de 48 wilayas et de 1 541 communes. Il n'existe pas d'autres collectivités territoriales en Algérie, les régions algériennes ne sont que des régions géographiques ou culturelles, et les daïras ne sont que des entités administratives.

Les caractéristiques des wilayas algériennes sont très disparates les unes comparées aux autres. Cela est dû à l'histoire du pays, à sa géographie et à sa démographie. Ainsi, on peut noter de grandes différences parmi les quarante-huit wilayas en termes de superficies, du nombre de communes ou de daïras et de population.

Malgré la richesse de l'Algérie dans tous les domaines, il reste un pays inconnu même pour la plupart des Algériens, et cela est dû au manque d'informations qu'on peu trouver sur le web. En effet malgré le grand nombre de sites web qu'on peu trouver sur le net qui parle de l'Algérie, cela reste insuffisant; puisque ils parle d'une manière générale, ou bien il citent les grandes wilayas seulement et les petites sont restés inconnues.

L'objectif de notre projet présenté dans ce mémoire est la conception et la réalisation d'une application web pour la collecte d'un maximum d'informations de toutes les wilayas algériennes, afin de permettre à tous les algériens de mieux connaitre toutes les wilayas du pays.

Afin d'y parvenir, nous avons organisé ce mémoire de la manière suivante :

- ✓ Le premier chapitre présente une étude préliminaire sur le domaine à étudier.
- ✓ Le deuxième chapitre présente la conception de notre application web que nous allons modéliser avec le langage UML.
- ✓ l'implémentation et La réalisation de notre application fera l'objet du troisième chapitre, dans lequel nous allons expliquer les différentes interfaces de l'application.
- ✓ Enfin, nous termineront par une conclusion générale.

Chapitre 1

Etude préliminaire

1.1 Introduction

L'objectif de cette étude est de réaliser une base de données permettant la collection d'informations relatives à l'ensemble des wilayas algériennes et de la faire manipuler et publier par une application web.

Dans ce chapitre, nous présentant en premier lieu le cahier de charge de notre domaine d'étude.

En deuxième lieu nous citons les choix techniques.

Nous finirons par présenter les différents organigrammes de notre méthode de conception du projet (UML).

1.2 Cahier de charges

1.2.1 Définition

Le cahier des charges se définit comme un document de référence qui permet de préciser les conditions, les règles et les exigences d'une mission, d'un travail à accomplir, en vue de résoudre un problème spécifique ou d'améliorer une situation donnée, tout en déterminant les résultats attendus.

1.2.2 définition d'un Acteur

Un acteur représente un rôle joué par une entité externe (utilisateur humain, dispositif matériel) qui interagit directement avec le système étudié.

Un acteur peut consulter et/ou modifier directement l'état du système, en émettant et/ou en recevant des messages susceptibles d'être porteurs de données.**[1]**

Acteur	Description
L'administrateur	C'est la personne responsable de l'administration de l'application web, il a le droit de gérer et d'accéder à toutes les rubriques.
Client	Le client a un compte dans l'application, il a le droit de chercher des wilayas, de consulter la base de données, modifier son compte.
Visiteur	C'est le visiteur de l'application web, il a seulement le droit de chercher des wilayas et de consulter la base de données.

Tableau 1.1 Les acteurs de l'application

1.2.3 Les besoins fonctionnels

Les besoins fonctionnels expriment une action que doit exécuter le système en réponse à une demande de l'administrateur ou d'un utilisateur.

L'application web qu'on va créer, devra regrouper un certain nombre d'informations sur les 48 wilayas du pays enregistrées dans notre base de données pour pouvoir faire une recherche claire, directe pour les utilisateurs.

Les utilisateurs doivent pouvoir récupérer les informations qui ils cherchent sur chacune des wilayas.

Le système doit :

- Offrir une interface qui permet d'accéder à l'application.
- Offrir à l'administrateur un espace d'authentification.
- Le système doit reconnaitre la personne connectée (Administrateur ou Utilisateur).
- Le système doit offrir le service de gestion des wilayas.

L'administrateur a pour rôle :

- Administrer le système.
- Gérer les wilayas.
- Gérer les clients.
- Maintenir et mettre à jour les tables de la base de données.

L'utilisateur et le visiteur vont pouvoir :

- Consulter la base de données.
- Consulter le contenu de l'application.
- Chercher des informations sue les différentes wilayas du pays.

1.2.4 Les besoins non fonctionnels

Ce sont des fonctionnalités qui participent à l'amélioration de l'application. Ces besoins regroupent les contraintes techniques, esthétiques et ergonomiques.

Pour faciliter l'interaction entre les utilisateurs et l'application, cette dernière doit répondre aux exigences de qualité et de performance suivantes :

Contraintes	Description
Techniques	La création d'une application web nécessite différents outils : - L'utilisation du langage de programmation java. - Le langage de modélisation UML. - Le système de gestion des bases de données SQL serveur
Esthétiques	- La police utilisée doit avoir un caractère intelligible et perceptible. - Utilisation des couleurs adaptées à l'application.
Ergonomiques	- Aider l'utilisateur à se repérer dans l'application - Simplifier la navigation, de manière à ce que de n'importes quelles page, on pourra accéder a toutes les autres pages de l'application.

Tableau 1.2 Besoins non fonctionnels

1.3 les choix techniques

Pour la réalisation de notre projet nous avons opté pour les choix techniques suivants :

UML : Méthode d'analyse et de conception

JAVA : Langage de programmation

ECLIPSE : Environnement de développement intégré

MICROSOFT SQL SERVEUR : Système de gestion de bases de données.

1.4 Diagrammes

Pour la modélisation des besoins, on utilise les diagrammes UML suivants :

1. Diagramme de contexte
2. Diagramme des cas d'utilisation
3. Diagrammes de séquences
4. Diagrammes d'activités

1.4.1 Diagramme de contexte

- Objectif d'utilisation

Le diagramme de contexte a pour but de représenter les flux d'informations entre l'organisation et les acteurs externes selon une représentation standard dans laquelle chaque objet porte un nom**. [2]**

Liste des intentions métier :

- I1 : Ajouter des utilisateurs
- I2 : Supprimer des utilisateurs
- I3 : Ajouter des administrateurs
- I4 : Supprimer des administrateurs
- I5 : Modifier le compte
- I6 : Ajouter une wilaya
- I7 : Modifier les informations des wilayas
- I8 : Supprimer une wilaya
- I9 : Consulter l'application
- I10 : Chercher une wilaya
- I11 : Faire une inscription

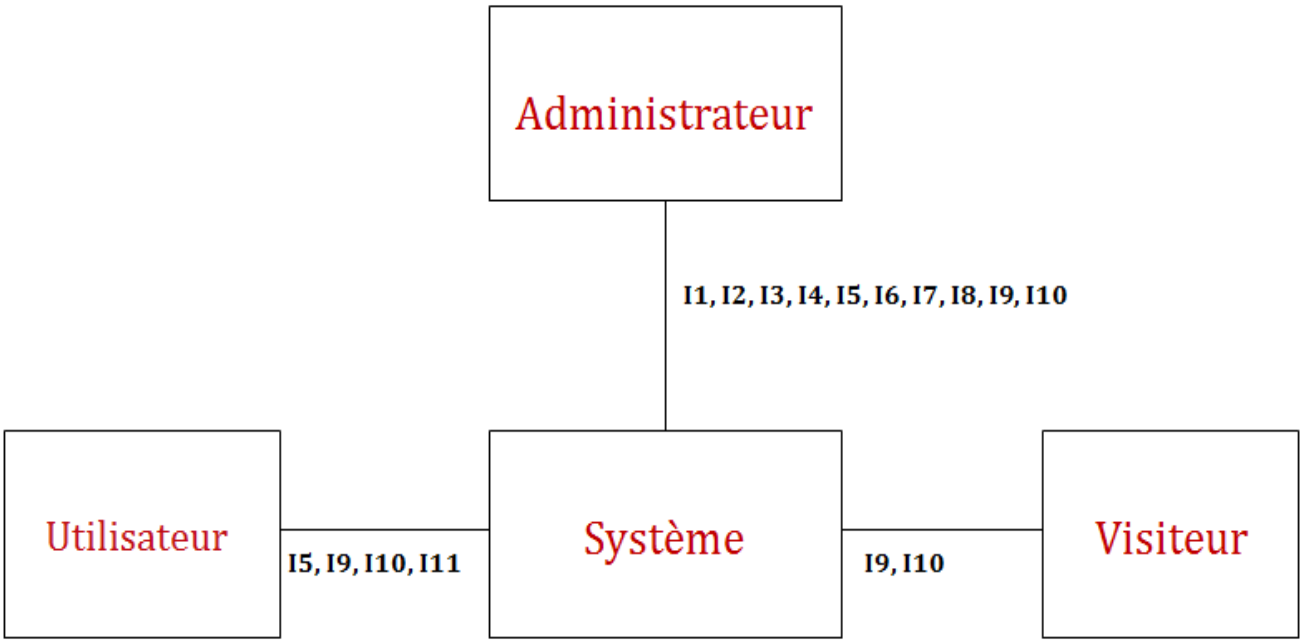

Figure 1.1 Diagramme de contexte

1.4.2 Diagramme de cas d'utilisation

1.4.2.1 Objectif d'utilisation

Les cas d'utilisation servent à exprimer le comportement du système en termes d'actions et de réactions, le modèle de cas d'utilisation comprend les acteurs, le système et les cas d'utilisations. Les fonctionnalités du système sont déterminées en examinant les besoins de chaque acteur qui est représenté sous forme de petits personnages et les cas d'utilisation se représentent sous forme d'une ellipse qui comporte le nom du cas. **[3]**

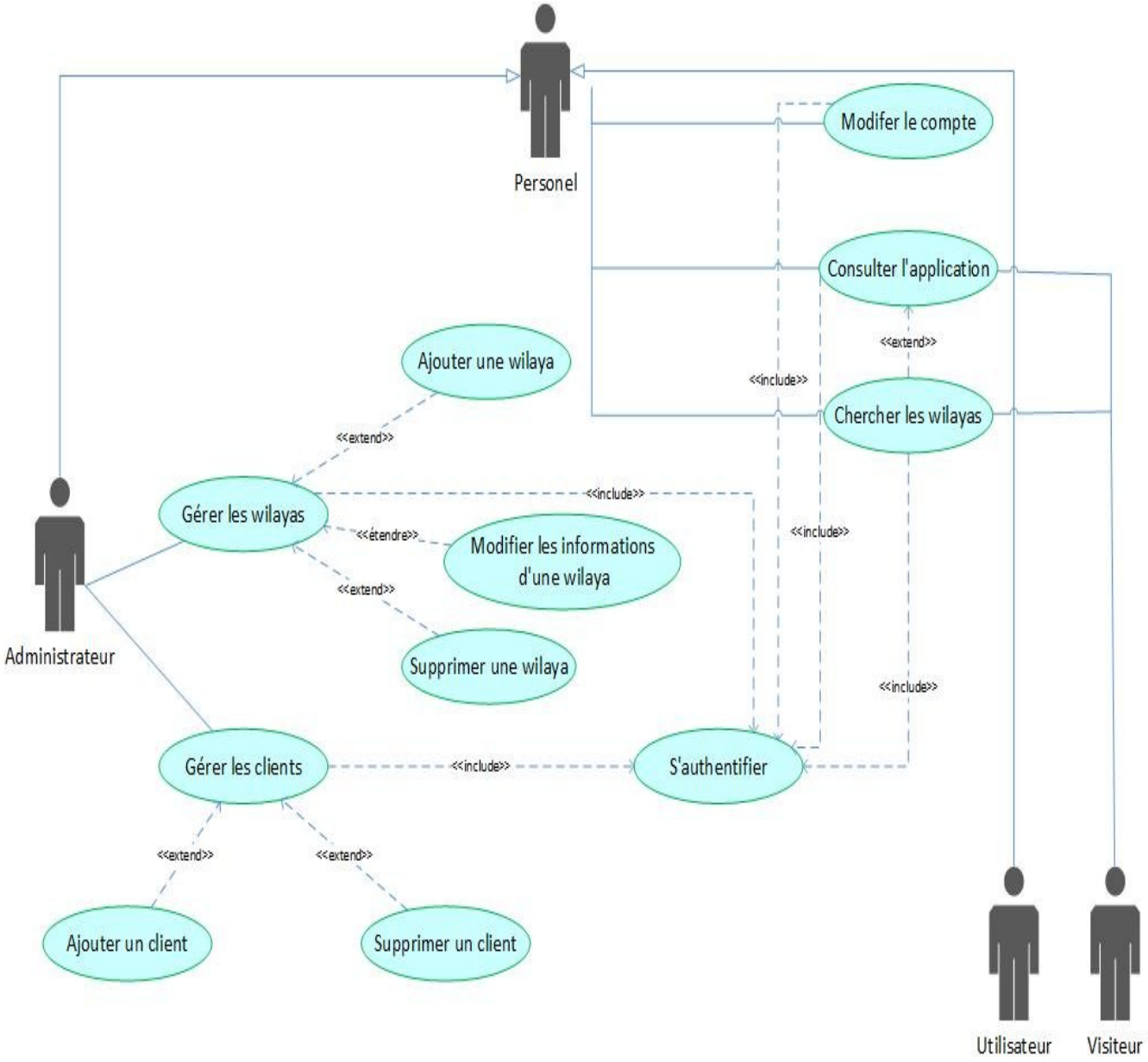

Figure 1.2 Diagramme de cas d'utilisation

1.4.2.2 Description textuelle du diagramme de cas d'utilisation

1. **Cas d'utilisation N° 1 : Authentification**

Cas d'utilisation N° 1 : Authentification	
objectif	Contrôler l'accès au système
Acteur	L'administrateur, l'utilisateur
Scénario nominal	[début] L'administrateur ou l'utilisateur saisit un login et mot de passe Le système vérifie l'existence dans la BDD Affichage de l'espace d'accueil adéquat [fin]
Scénario alternatif	Les informations fournies sont incorrectes Le système réaffiche le formulaire d'authentification et attend que l'administrateur ou l'utilisateur ressaisisse ses informations.

Tableau 1.3 Description textuelle du cas d'utilisation " Authentification ".

2. Cas d'utilisation N° 2 : Gérer les utilisateurs

Cas d'utilisation N° 2 : Gérer les utilisateurs	
Objectif	Gérer les utilisateurs
Acteur	L'administrateur
Scénario nominal	[début] L'administrateur effectue une demande de liste des Utilisateurs. Le système affiche la liste des utilisateurs. L'administrateur effectue (suppression, l'ajout, la recherche) d'un utilisateur. Le système valide la mise à jour [fin]

Tableau 1.4 Description textuelle du cas d'utilisation " Gérer les Utilisateurs ".

3. Cas d'utilisation N° 3 : Gérer les Wilayas

Cas d'utilisation N° 3 : Gérer les wilayas	
Objectif	gérer les wilayas
Acteur	L'administrateur
Scénario nominal	[début] L'administrateur effectue une demande de liste des wilayas. Le système affiche la liste des wilayas. L'administrateur effectue (suppression, modification, l'ajout) d'une wilaya. Le système valide la mise à jour [fin]

Tableau 1.5 Description textuelle du cas d'utilisation " Gérer les Wilayas ".

4. Cas d'utilisation N° 4 : Modifier le Compte

Cas d'utilisation N° 4 : Modifier le compte	
Objectif	Modifier le compte
Acteur	L'administrateur, l'utilisateur
Scénario nominal	[début] L'administrateur ou l'utilisateur effectue une demande d'affichage de ses coordonnées. Le système affiche ses coordonnées. L'administrateur ou l'utilisateur effectue une modification. Le système valide la mise à jour [fin]

Tableau 1.6 Description textuelle du cas d'utilisation " Modifier le compte ".

5. Cas d'utilisation N° 5 : Consulter l'application

Cas d'utilisation N° 5 : Consulter l'application	
Objectif	Consulter l'application
Acteur	L'administrateur, utilisateur, visiteur
Scénario nominal	L'administrateur ou l'utilisateur ou bien le visiteur demande l'interface voulue Le système affiche l'interface en question.

Tableau 1.7 Description textuelle du cas d'utilisation " Consulter l'application ".

6. Cas d'utilisation N° 6 : chercher les wilayas

Cas d'utilisation N° 5 : Chercher les wilayas	
Objectif	Chercher les wilayas
Acteur	L'administrateur, l'utilisateur, visiteur
Scénario nominal	L'administrateur ou l'utilisateur ou bien le visiteur demande la wilaya voulue. Le système affiche la wilaya en question

Tableau 1.8 Description textuelle du cas d'utilisation " Chercher les wilayas ".

1.4.3 Diagrammes de séquences

1.4.3.1 Objectif d'utilisation

Il représente des échanges de messages entre objets, selon un point de vue temporel. Il permet ainsi de décrire les scénarios de chaque cas d'utilisation en mettant l'accent sur la chronologie des opérations en interaction avec les objets. **[3]**

1.4.3.2 Diagramme de séquence : "Authentification "

Lorsque l'utilisateur veut s'authentifier, deux cas peuvent se présenter : Données correctes ou données incorrectes, c'est pourquoi on a utilisé l'operateur " Alt ". En effet, si les données d'authentification fournies par l'utilisateur sont correctes alors le système accorde l'accès à l'interface appropriée. Dans le cas contraire, un message d'erreur est généré et la page d'authentification est réaffichée. Ce procédé est exécuté à chaque fois que l'utilisateur tente de s'authentifier, ce qui justifie l'utilisation de l'operateur "Loop".

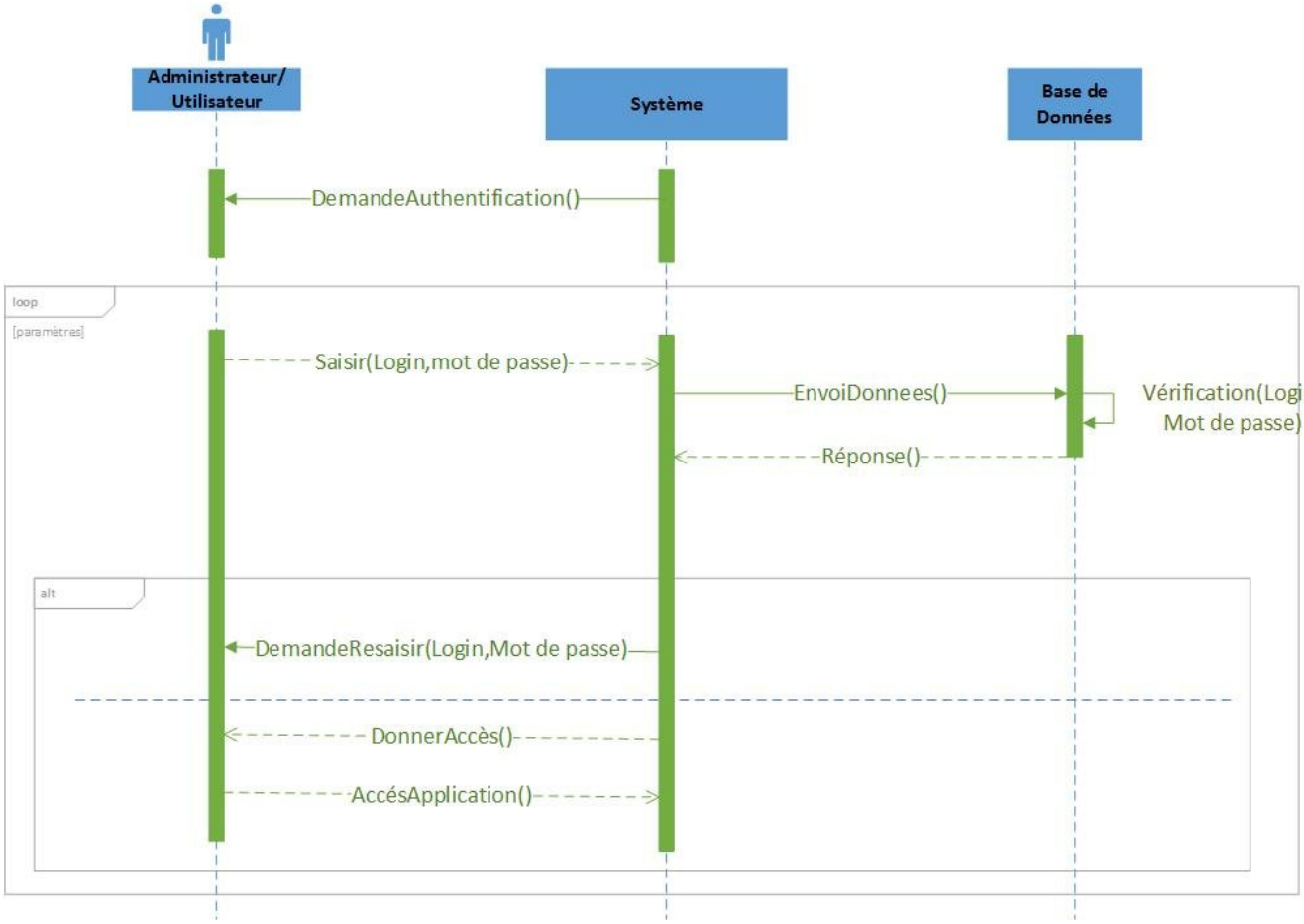

Figure 1.3 Diagramme de séquence " Authentification ".

1.4.3.3 Diagramme de séquence : "Ajout"

- L'ajout se fait par l'administrateur après authentification.
- Il demande le formulaire d'ajout qui sera affiché par le système.
- Il saisie les données à ajouter.
- Le système effectue une vérification de validité de données.
- Si les données sont correctes, ils seront enregistrés dans la base de données et un message de confirmation est affiché.
- Si les données ne sont pas correctes un message d'erreur est affiché.

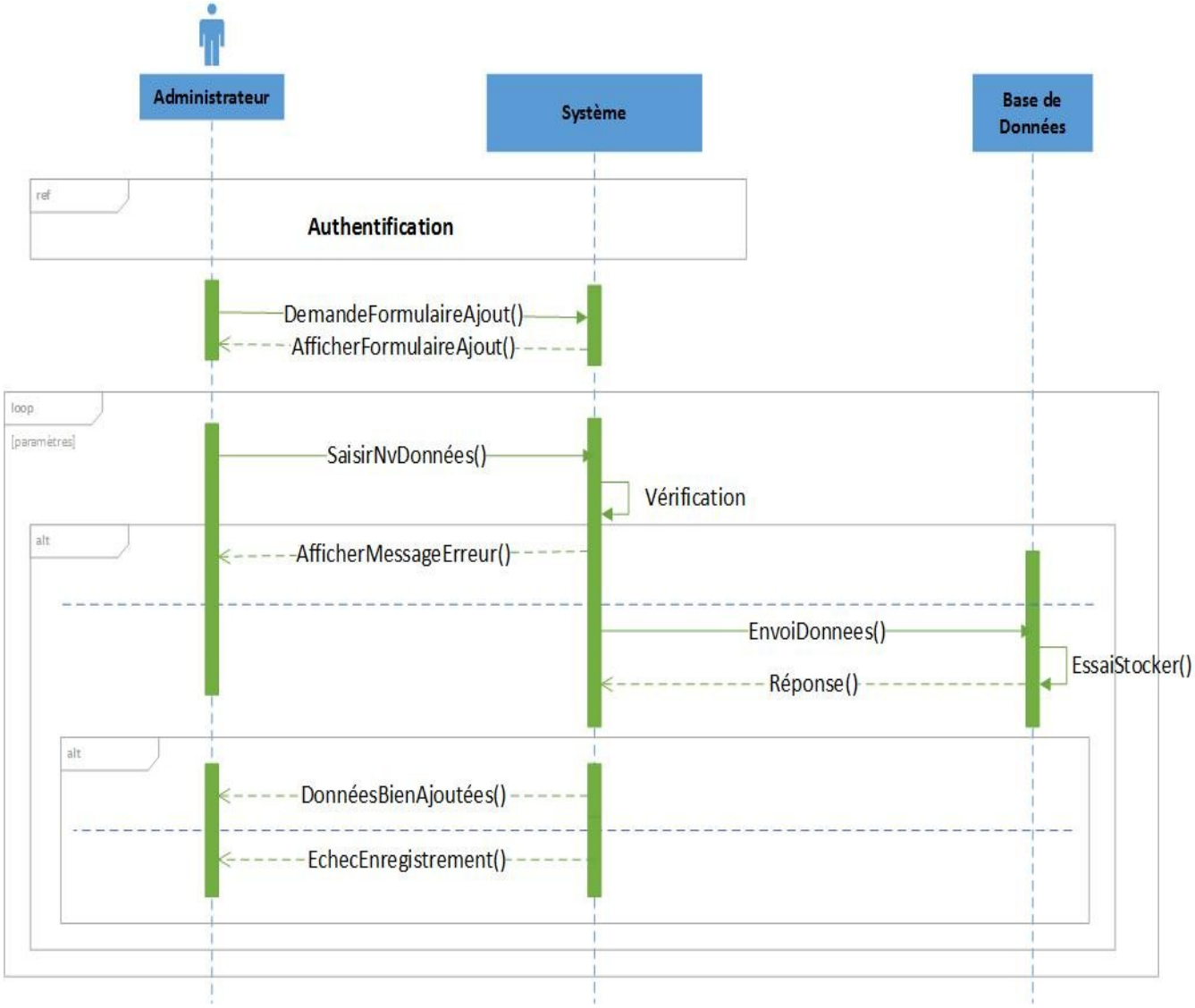

Figure 1.4 Diagramme de séquence " Ajout ".

1.4.3.4 Diagramme de séquence : "Modification"

- La modification se fait par l'administrateur après authentification.
- Il demande le formulaire de modification qui sera affiché par le système.

- Il saisie le code de la donnée à modifier.
- Le système effectue une vérification de validité du code saisi.
- Si le code n'est pas valide, le système demande la saisie d'un code valide.
- Si le code est valide, le système demande la saisie des nouvelles données.
- L'administrateur effectue la modification voulue.
- Si les nouvelles données saisies sont correctes elles seront enregistrées dans la base de données et un message de confirmation est affiché.
- Si les données ne sont pas correctes un message d'erreur est affiché.

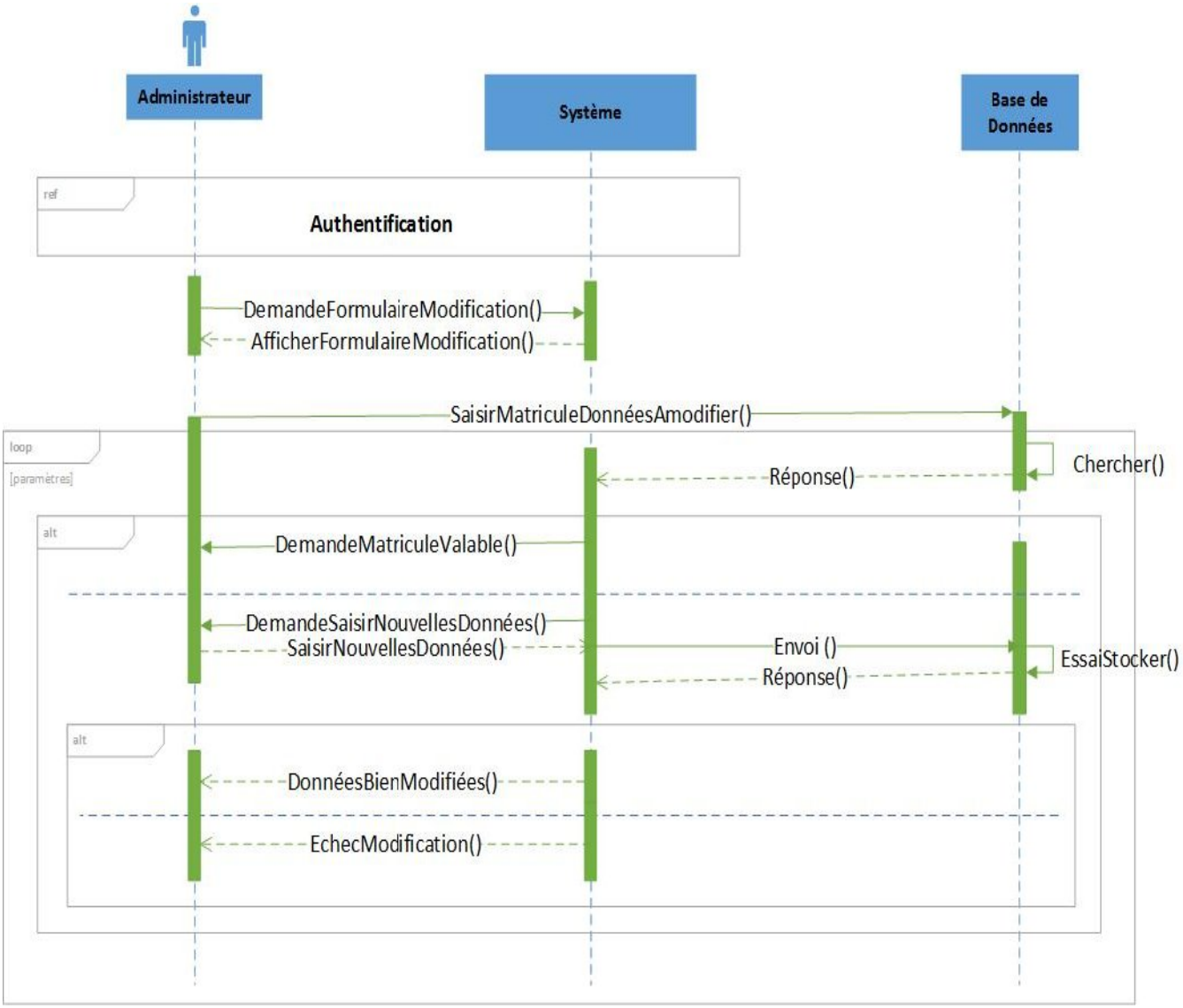

Figure 1.5 Diagramme de séquence " Modification ".

1.4.3.5 Diagramme de séquence : "Suppression"

La Suppression se fait par l'administrateur après authentification.
Il demande le formulaire de suppression qui sera affiché par le système.
Il saisie la code de la donnée à supprimer.
Le système effectue une vérification de validité du code saisi
Si le code n'est pas valide, le système demande la saisie d'un code valide
Si le code est valide, le système demande la confirmation de la suppression.
L'administrateur confirme et effectue la suppression.
En cas d'échec de suppression, un message est affiché.

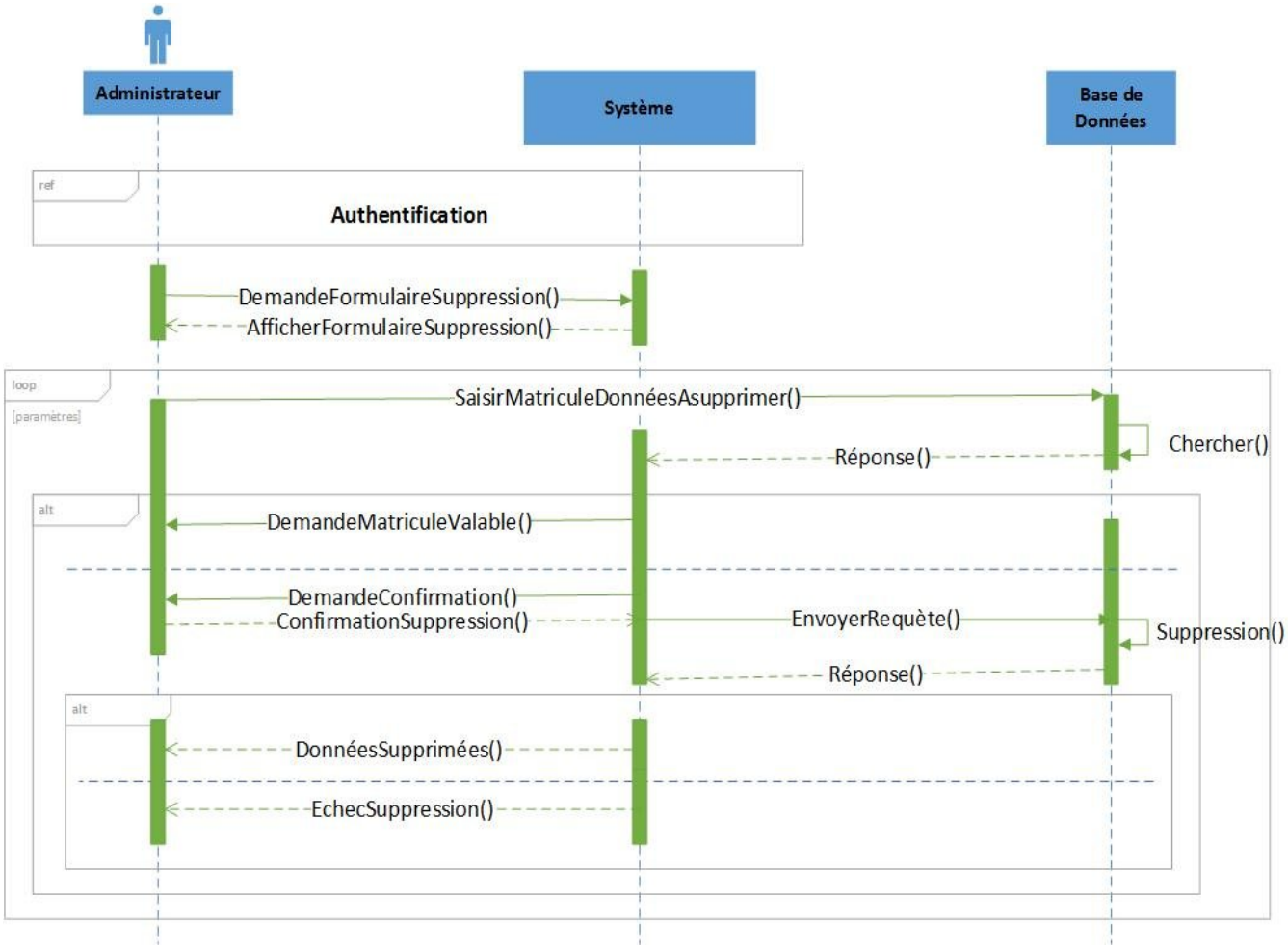

Figure 1.6 Diagramme de séquence " Suppression ".

1.4.4 Diagrammes d'activités

1.4.4.1 Objectif d'utilisation

Le diagramme d'activité représente les règles d'enchaînement des activités et actions dans le système. Il permet de consolider la spécification d'un cas d'utilisation. **[2]**

Une activité désigne une suite d'actions, le passage d'une action à l'autre est matérialisé par une transition. Les transitions sont déclenchées par la fin d'une action et provoquent le début immédiat d'une autre. **[3]**

1.4.4.2 Diagramme d'activité :"Authentification"

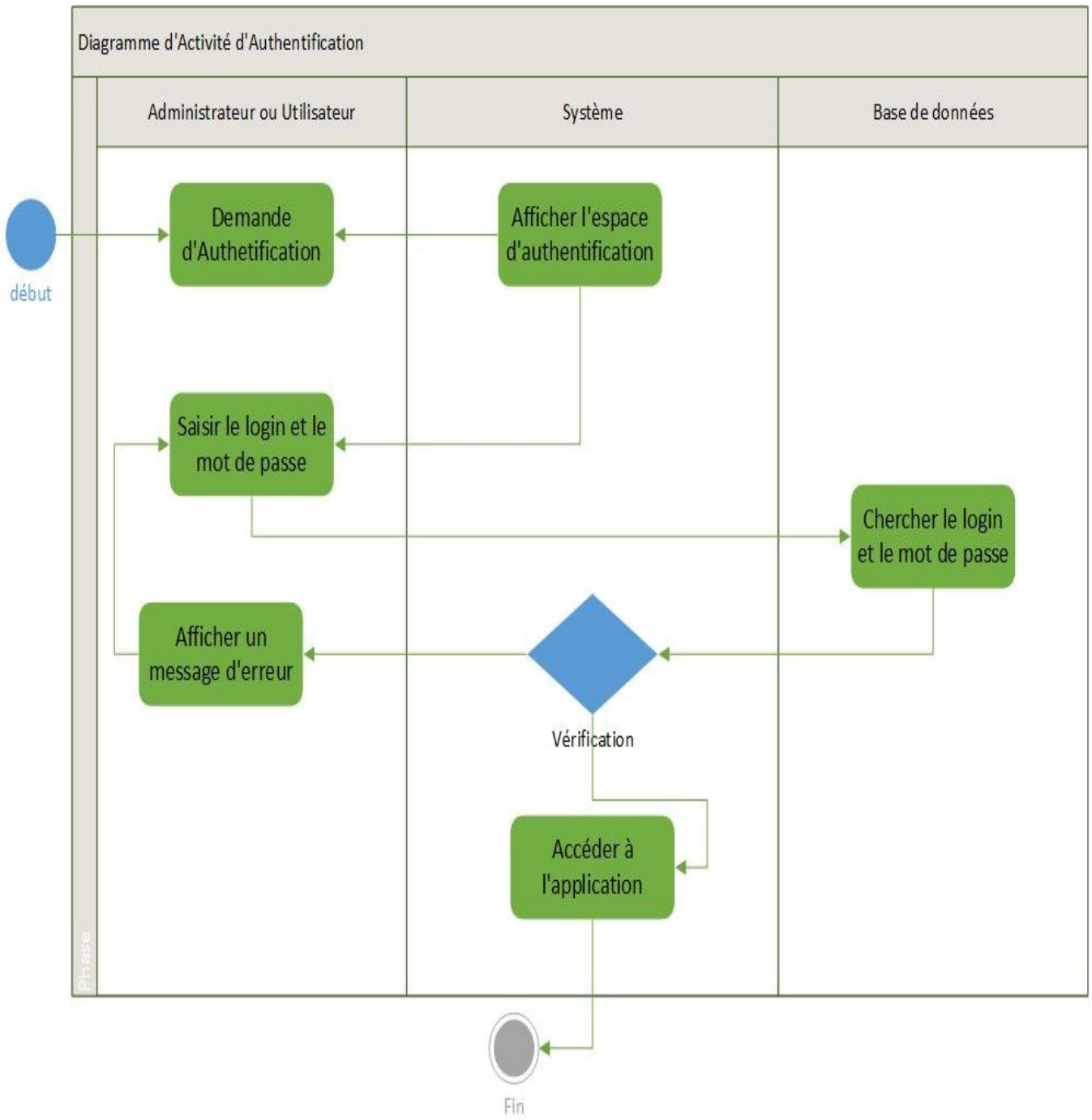

Figure 1.7 Diagramme d'Activité " Authentification ".

1.4.4.3 Diagramme d'activité :"Ajout"

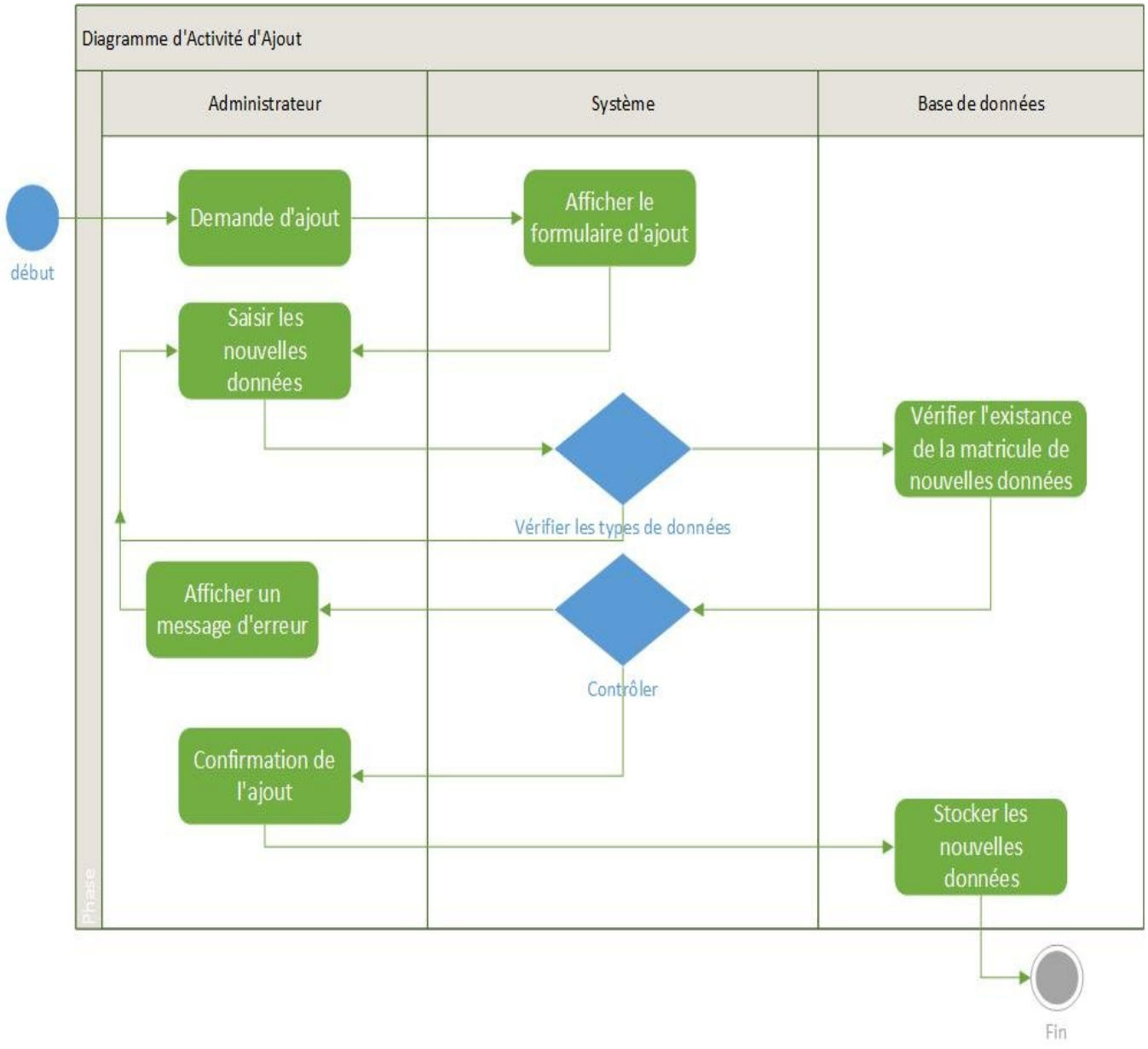

Figure 1.8 Diagramme d'Activité " Ajout ".

1.4.4.4 Diagramme d'activités : "Modification"

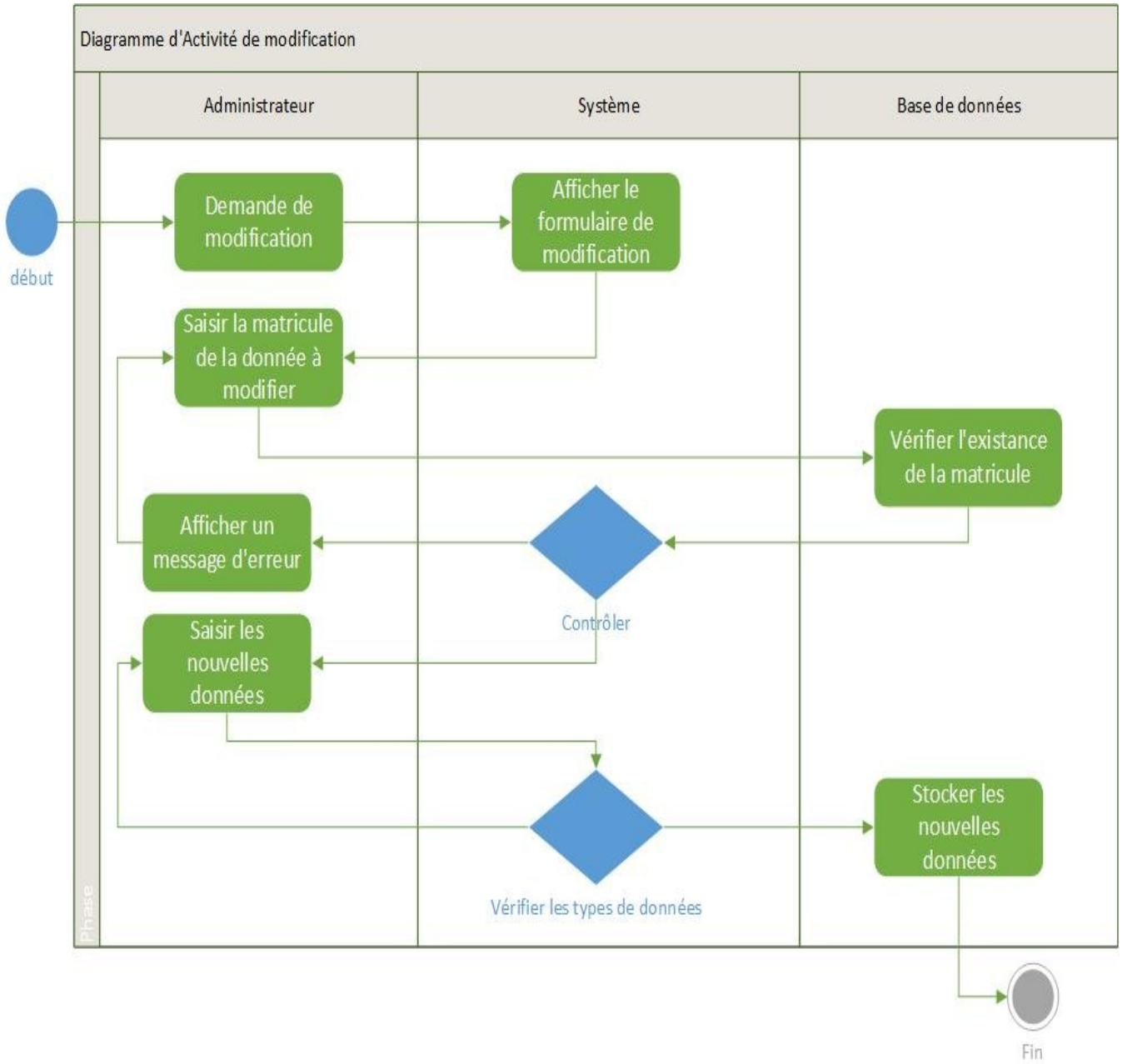

Figure 1.9 Diagramme d'Activité " Modification ".

1.4.4.5 Diagramme d'activité : "Suppression"

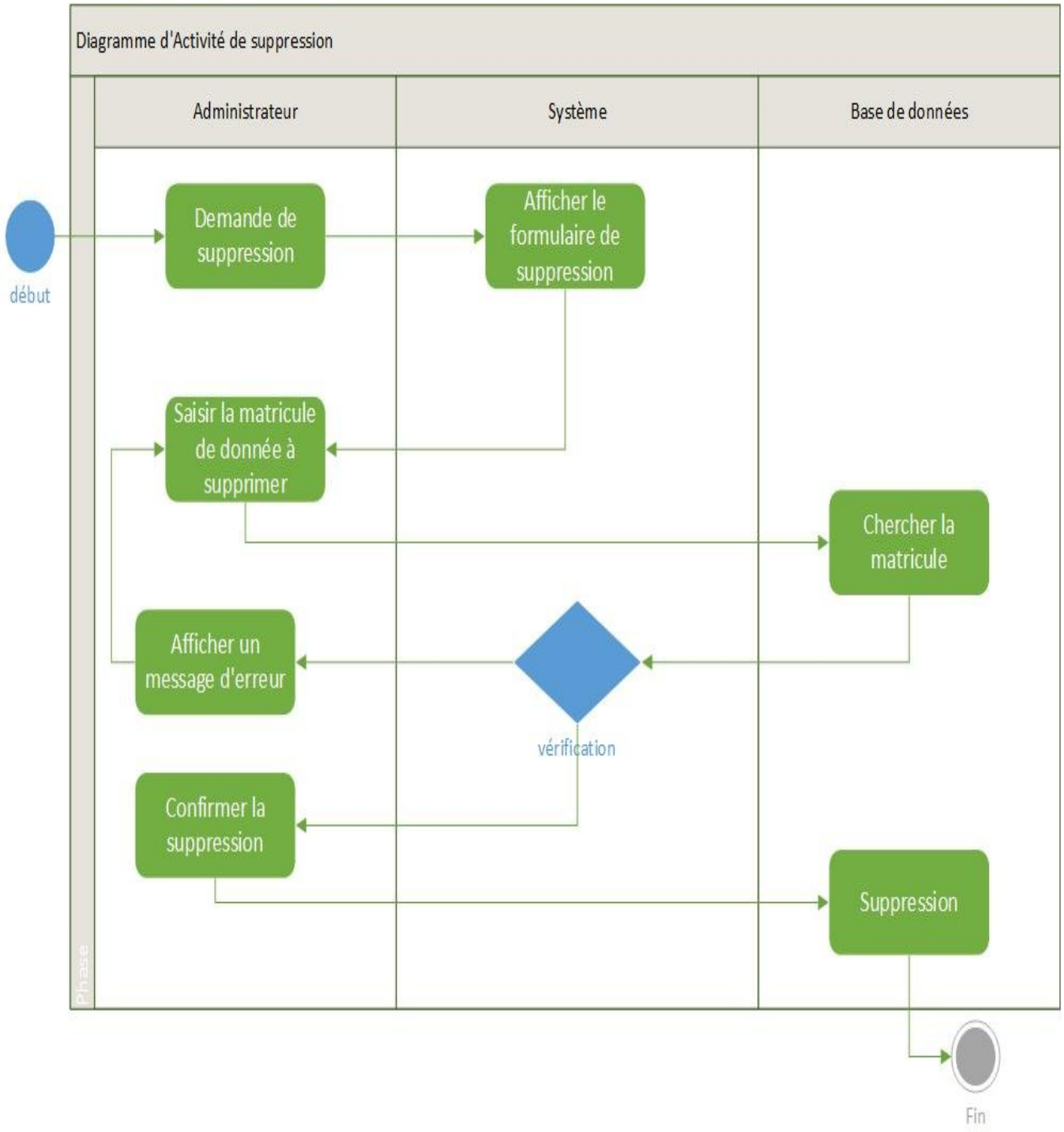

Figure 1.10 Diagramme d'Activité " Suppression ".

1.5 Conclusion

Ce chapitre nous a permet d'élaborer une étude préliminaire de notre application.
Nous avons cerné les besoins fonctionnels et opérationnels de l'acteur qui interagie avec notre application, et cela grâce aux différents diagrammes qu'on a pu réaliser.

Chapitre 02

Conception

2.1 Introduction

À la cour de ce chapitre nous allons établir la conception adaptée à notre application.

Nous commencerons ce chapitre par la Présentation de l'approche UML, ensuite nous Appliquons ce langage de modélisation (UML) à l'application étudiée.

Nous finirons par présenter le digramme de classes associé à notre application, ainsi que les attributs que comprend chacune des classes, suivi de la spécification des règles de passages du modèle de classe au modèle relationnel.

2.2 Paradigme orienté objet

2.2.1 Historique

Conçu à l'origine, au cours des années 1990, pour faciliter la construction d'applications orientées objet (OO), le langage UML a ensuite fortement évolué jusqu'à sa version 2.1 actuelle. Néanmoins, UML reste toujours très OO. Les concepts qu'il propose pour modéliser la vue structurelle sont donc les concepts de classe et d'objet.
UML définit cependant sa propre sémantique OO, laquelle ressemble à la sémantique des langages de programmation objet Java ou C++. Il est donc important de considérer UML comme un langage à part entière, et non comme une couche graphique permettant de dessiner des applications Java ou C++. **[4]**

2.2.2 Concepts élémentaires

Un minimum de connaissances du paradigme orienté objet est requis.
Les concepts élémentaires que nous présentons dans cette section sont les plus employés pour la réalisation de la vue structurelle d'un modèle UML.

2.2.2.1 Concept D'objet

Un objet représente une entité du monde réel (ou du monde virtuel pour les objets immatériels) qui se caractérise par un ensemble de propriétés (attributs), des états significatifs et un comportement. **[3]**
L'état d'un objet correspond aux valeurs de tous ses attributs à un instant donné.
Le comportement d'un objet est caractérisé par l'ensemble des opérations qu'il peut exécuter en réaction aux messages provenant des autres objets.

2.2.2.2 Concept de classe

En UML, une classe définit la structure commune d'un ensemble d'objets et permet la construction d'objets instances de cette classe. Une classe est identifiée par son nom. **[4]**

2.2.2.3 Propriété d'une classe

Chaque classe possède une ou plusieurs propriétés. Une propriété a un nom et un type. Le type peut être soit une classe UML, soit un type de base (integer, string, boolean, char, real). Un objet instance de la classe ou de l'interface doit porter les valeurs des propriétés de sa classe.

2.2.2.4 Héritage entre classes

En UML, une classe peut hériter d'autres classes. L'héritage entre classes UML doit être considéré comme une inclusion entre les ensembles des objets instances des classes. Les objets instances des sous-classes sont des objets instances des superclasses. En d'autres termes, si une classe A hérite d'une classe B, l'ensemble des objets instances de A est inclus dans l'ensemble des objets instances de B. **[4]**

2.2.2.5 Association entre classes

L'association représente une relation entre deux ou plusieurs classes. Elle correspond à l'abstraction des liens qui existent entre les objets dans le monde réel. Les multiplicités (cardinalités) et les rôles des objets participant aux relations complètent la description d'une association. Les exemples d'associations sont donnés directement dans les diagrammes de classe d'UML. **[3]**

2.2.2.6 Agrégation entre classes

L'agrégation est une forme particulière d'association entre plusieurs classes. Elle exprime le fait qu'une classe est composée d'une ou plusieurs autres classes. La relation composant-composé ou la relation structurelle représentant l'organigramme d'une entreprise sont des exemples types de la relation d'agrégation. **[3]**

2.2.2.7 Polymorphisme

Le polymorphisme est la capacité donnée à une même opération de s'exécuter différemment suivant le contexte de la classe où elle se trouve. Ainsi une opération définie dans une super-classe peut s'exécuter de manière différente selon la sous-classe où elle est héritée.

2.3 Présentation de l'approche UML

2.3.1 Définition d'UML

UML se définit comme un langage de modélisation graphique et textuel destiné à comprendre et décrire des besoins, spécifier et documenter des systèmes, esquisser des architectures logicielles, concevoir des solutions et communiquer des points de vue.

UML unifie à la fois les notations et les concepts orientés objet. Il ne s'agit pas d'une simple notation, mais les concepts transmis par un diagramme ont une sémantique précise et sont porteurs de sens au même titre que les mots d'un langage. UML a une dimension symbolique et ouvre une nouvelle voie d'échange de visions systémiques précises. Ce langage est certes issu du développement logiciel mais pourrait être appliqué à toute science fondée sur la description d'un système. Dans l'immédiat, UML intéresse fortement les spécialistes de l'ingénierie système. **[2]**

2.3.2 Les diagrammes d'UML

UML 2.0 s'articule autour de treize diagrammes complémentaires qui permettent la modélisation d'un projet. Dans le schéma de la figure ci-dessous, nous allons décrire ces diagrammes.

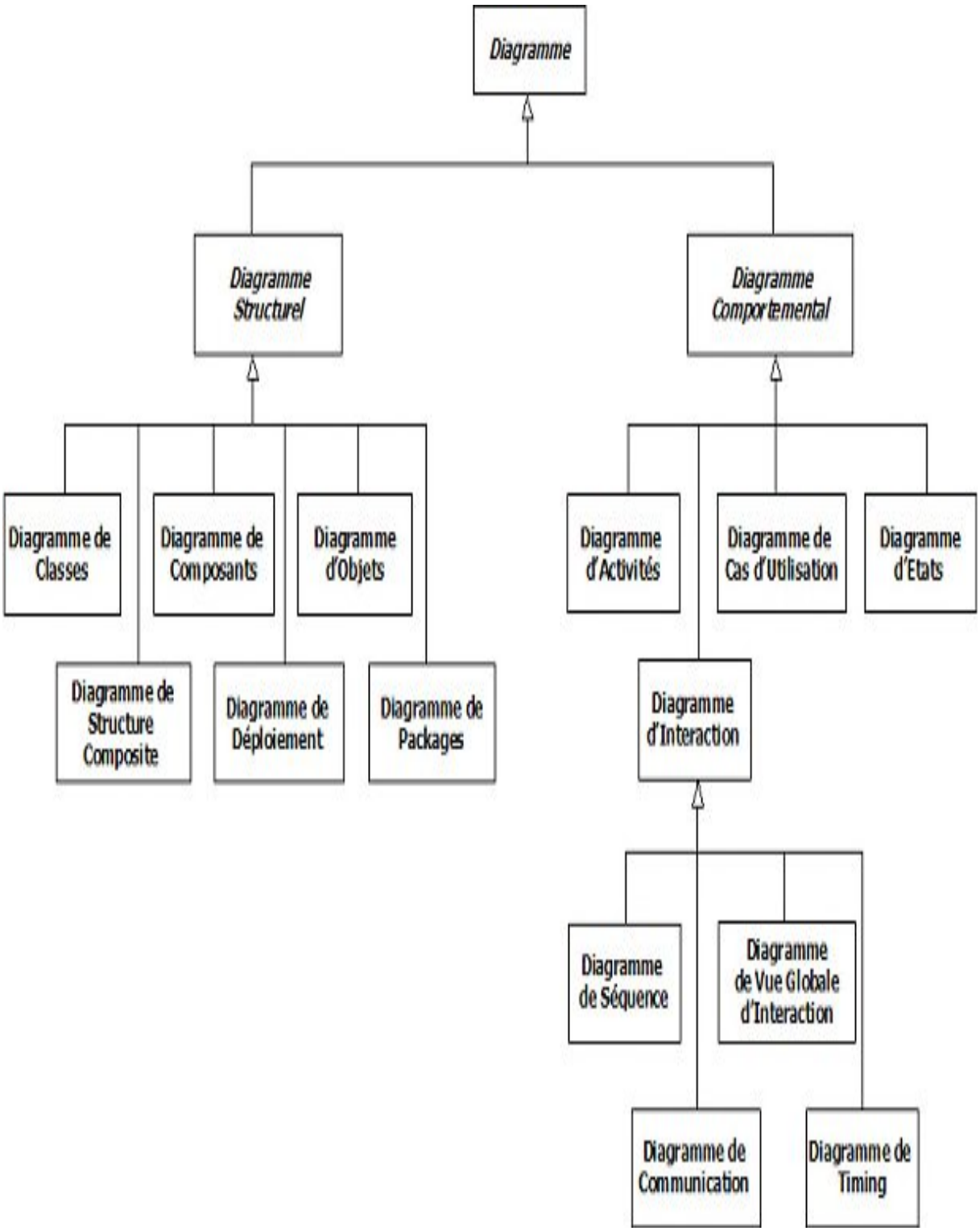

Figure 2.1 Les diagrammes d'UML 2

2.3.2.1 les diagrammes structurels

Les sept diagrammes structurels se présentent comme suit :

1. **Diagramme de classes**

Le diagramme de classe constitue l'un des pivots essentiels de la modélisation avec UML. En effet, ce diagramme permet de donner la représentation statique du système à développer. Cette représentation est centrée sur les concepts de classe et d'association. Chaque classe se décrit par les données et les traitements dont elle est responsable pour elle-même et vis-à-vis des autres classes. La description du diagramme de classe est fondée sur :

- le concept d'objet,
- le concept de classe comprenant les attributs et les opérations,
- les différents types d'association entre classes.

2. **Diagramme d'objet**

Le diagramme d'objets sert à illustrer des structures de classes compliquées en montrant des exemples d'instances. Ce diagramme est utilisé en analyse pour vérifier l'adéquation d'un diagramme de classes à différents cas possibles.

3. **Diagramme de composant**

Le diagramme de composants représente les concepts connus de l'exploitant pour installer et dépanner le système. Il s'agit dans ce cas de déterminer la structure des composants d'exploitation que sont les librairies dynamiques, les instances de bases de données, les applications, les progiciels, les objets distribués, les exécutables, etc.

4. **Diagramme de déploiement**

Le diagramme de déploiement correspond à la fois à la structure du réseau informatique qui prend en charge le système logiciel, et la façon dont les composants d'exploitation y sont installés.

5. **Diagramme de paquetage**

Il donne une vue d'ensemble du système structuré en paquetage. Chaque paquetage représente un ensemble homogène d'élément du système.

6. Diagramme de structure composite

Le diagramme de structure composite décrit la composition d'un objet complexe lors de son exécution.

2.3.2.2 les diagrammes comportementaux

Les diagrammes comportementaux sont focalisés sur la description de la partie dynamique du système à modéliser. Sept diagrammes sont proposés par UML 2 pour assurer cette description :

1. le diagramme des cas d'utilisation (DCU)

Les cas d'utilisation constituent un moyen de recueillir et de décrire les besoins des acteurs du système. Ils peuvent être aussi utilisés ensuite comme moyen d'organisation du développement du logiciel, notamment pour la structuration et le déroulement des tests du logiciel.

2. Le diagramme d'état-transition (DET)

Permet de décrire sous forme de machines à états finis le comportement du système ou de ses composants **[5].**

3. Le diagramme d'activité (DAC)

Il donne une vision des enchaînements des activités propres à une opération ou à un cas d'utilisation.

4. Le diagramme de séquence (DSE)

Permet de décrire les scénarios de chaque cas d'utilisation en mettant l'accent sur la chronologie des opérations en interaction avec les objets.

5. Le diagramme de communication (DCO)

Le diagramme de communication constitue une autre représentation des interactions que celle du diagramme de séquence. En effet, le diagramme de communication met plus l'accent sur l'aspect spatial des échanges que l'aspect temporel.

6. Le diagramme global d'interaction (DGI)

Le diagramme global d'interaction permet de représenter une vue générale des interactions décrites dans le diagramme de séquence et des flots de contrôle décrits dans le diagramme d'activité.

7. Le diagramme de temps (DTP)

Le diagramme de temps permet de représenter les états et les interactions d'objets dans un contexte où le temps a une forte influence sur le comportement du système à gérer. Autrement dit, le diagramme de temps permet de mieux représenter des changements d'états et des interactions entre objets liés à des contraintes de temps.

2.3.3 Processus pour UML

2.3.3.1 processus de développent logiciel

Un processus définit une séquence d'étapes, en partie ordonnées, qui concourent à l'obtention d'un système logiciel ou à l'évolution d'un système existant.
L'objet d'un processus de développement est de produire des logiciels de qualité qui répondent aux besoins de leurs utilisateurs dans des temps et des coûts prévisibles.

2.3.3.2 Processus unifié (UP)

Le processus unifié est un processus de développement logiciel c'est-à dire qu'il regroupe les activités à mener pour transformer les besoins d'un utilisateur en système logiciel, ce n'est pas un simple processus mais une infrastructure préfabriquée (Framework) de processus générique étant capable d'être adapter à une large classe de systèmes logiciels. **[3]**
Complément d'UML le processus unifié a pour but de spécifier les différentes phases d'un projet, de l'élaboration d'un cahier de charge au déploiement de l'application.

2.3.3.3 Caractéristiques d'un UP

Le processus de développement UP, associé à UML, met en œuvre les principes suivants :

1. Processus guidé par les cas d'utilisation

L'orientation forte donnée ici par UP est de montrer que le système à construire se définit d'abord avec les utilisateurs. Les cas d'utilisation permettent d'exprimer les interactions du système avec les utilisateurs, donc de capturer les besoins.
Une seconde orientation est de montrer comment les cas d'utilisation constituent un vecteur structurant pour le développement et les tests du système. Ainsi le développement peut se décomposer par cas d'utilisation et la réception du logiciel sera elle aussi articulée par cas d'utilisation.

2. Processus itératif et incrémental

Ce type de démarche étant relativement connu dans l'approche objet, il paraît naturel qu'UP préconise l'utilisation du principe de développement par itérations successives. Concrètement, la réalisation de maquette et prototype constitue la réponse pratique à ce principe. Le développement progressif, par incrément, est aussi recommandé en s'appuyant sur la décomposition du système en cas d'utilisation.

Les avantages du développement itératif se caractérisent comme suit :

- les risques sont évalués et traités au fur et à mesure des itérations,
- les premières itérations permettent d'avoir un feed-back des utilisateurs,
- les tests et l'intégration se font de manière continue,
- les avancées sont évaluées au fur et à mesure de l'implémentation.

3. Processus centré sur l'architecture

Les auteurs d'UP mettent en avant la préoccupation de l'architecture du système dès le début des travaux d'analyse et de conception. Il est important de définir le plus tôt possible, même à grandes mailles, l'architecture type qui sera retenue pour le développement, l'implémentation et ensuite le déploiement du système. Le vecteur des cas d'utilisation peut aussi être utilisé pour la description de l'architecture.

4. Processus orienté par la réduction des risques

L'analyse des risques doit être présente à tous les stades de développement d'un système. Il est important de bien évaluer les risques des développements afin d'aider à la bonne prise de décision. Du fait de l'application du processus itératif, UP contribue

à la diminution des risques au fur et à mesure du déroulement des itérations successives.

2.4 Diagramme de classe de l'application

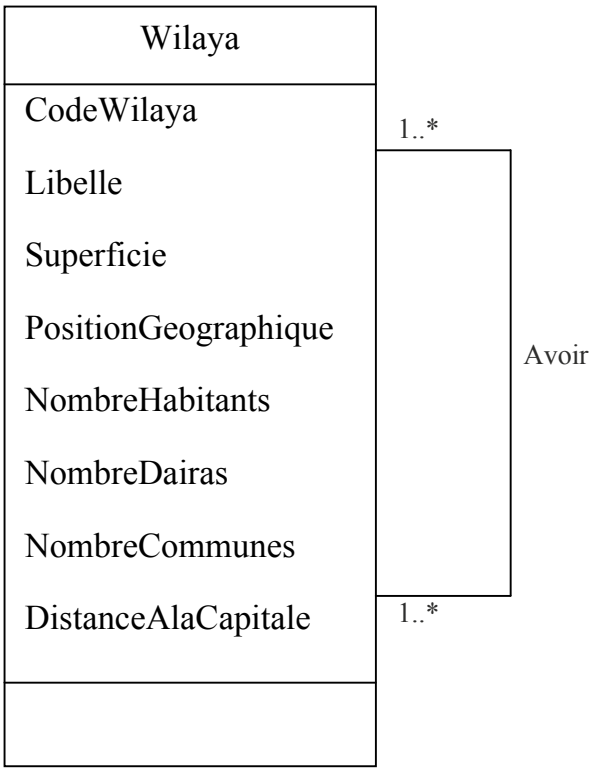

Figure 2.2 Diagramme de classes

2.5 Modèle relationnelle de données

2.5.1 Terminologie de l'approche relationnelle

- **Domaine** : Ensemble des valeurs admises pour un attribut, Il établit les valeurs acceptables dans une colonne.
- **Tuple** : une ligne dans une table.
- **Relation** : c'est une table qui contient des données appelées enregistrements (lignes), et de colonnes (champs) **[6].**
- **Clé primaire**: la clé primaire d'une table est l'ensemble minimal de colonnes qui permet d'identifier de manière unique chaque enregistrement.
- **Clé étrangère**: assure les liaisons entre les tables et les contraintes d'intégrités.

2.5.2 Règles de passage

- Toute entité donne naissance à une table dont la clé primaire est l'identifiant de l'entité.
- Toute association binaire (1..1) à (1..*) donne naissance à une table dérivée de l'entité du cardinalité (1..1), et sa clé primaire est déposé comme clé étrangère dans la table dérivée du cardinalité (1..*).
- Toute association binaire (1..*) à (1..*) avec une entité, donne naissance à trois tables, dont la table dérivée de l'entité association possèdera les clés primaires des deux autres tables comme clé primaire.
- Toute composante liée à une composite donne naissance à deux tables, dont la table dérivée de l'entité composante possédera la clé primaire de la table dérivée de l'entité composite comme clé primaire.
- L'agrégation de composition donnera naissance à deux tables dont la table agrégée possèdera la clé primaire de la deuxième table comme clé étrangère.
- Trois décompositions sont possibles pour traduire une association d'héritage en fonction des contraintes existantes :

1. Décomposition par distinction

Il faut transformer chaque sous-classe en une relation. La clé primaire de la sur-classe migre dans la (les) relation(s) issu(s) de la (des) sous-classe(s) et devient à la fois clé primaire et clé étrangère.

2. Décomposition descendante (push down)

   S'il existe une contrainte de totalité ou de partition sur l'association, il est possible de ne pas traduire la relation issue de la sur-classe. Il faut alors faire migrer tous ses attributs dans la (les) relation(s) issue(s) de la (des) sous classe(s).

3. Décomposition ascendante (push up)

   Il faut supprimer la (les) relation(s) issue(s) de la (des) sous-classe(s) et faire migrer les attributs dans la relation issue de la sur-classe **[7].**

2.5.3 Schéma relationnel

En appliquant les règles de transformation d'un modèle de classe vers un modèle relationnel, on aboutit au schéma relationnel suivant :

NB : dans ce qui suit nous allons souligner les clés primaires, et mettre # devant les clés étrangères.

Designation_Wilaya (CodeWilaya, Libelle) ;

Wilaya (Libelle, Superfecie, PositionGeographique, NombreHabitants,

NombreDairas, NombreCommues, DistanceAlaCapitale) ;

Avoisiner (#CodeWilaya, #CodeWilayaAvoisinante, Localisation) ;

2.6 Conclusion

Après une bonne compréhension de l'approche UML et son application à l'application étudiée, et en respectant les règles de passage, nous avons transformé le diagramme de classes obtenu en modèle relationnel qui nous a permis de déduire les tables de notre application que nous utiliseront dans la réalisation. Nous pouvons alors mener les phases d'implémentation qu'on va présenter dans le chapitre suivant.

Chapitre 3

Implémentation

3.1 Introduction

Dans ce chapitre nous allons faire une présentation de l'outil de développement utilisé pour la réalisation de notre travail, et aussi un aperçu sur le système de gestion des bases de données utilisé en l'occurrence Microsoft SQL serveur 2008 R2. Nous terminons ce chapitre par une projection de quelques prises d'écran d'essentielles fonctionnalités de l'application réalisée.

3.2 Langages de programmation

3.2.1 Java

Java est un langage objet permettant le développement d'applications complètes s'appuyant sur les structures de données classiques (tableaux, fichiers) et utilisant abondamment l'allocation dynamique de mémoire pour créer des objets en mémoire.

La notion de structure, ensemble de données décrivant une entité (un objet en Java) est remplacée par la notion de classe au sens de la programmation objet. Le langage Java permet également la définition d'interfaces graphiques (GUI : Graphical User Interface) facilitant le développement d'applications interactives et permettant à l'utilisateur de "piloter" son programme dans un ordre non imposé par le logiciel.

Le langage est aussi très connu pour son interactivité sur le Web facilitant l'insertion dans des pages Web, au milieu d'images et de textes, de programmes interactifs appelés "applets".

Un programme Java est portable au sens où il peut s'exécuter sur des ordinateurs fonctionnant avec différents systèmes d'exploitation. Java est portable d'une plate-forme (matériel et système d'exploitation) à une autre sans recompilation.

Le compilateur produit un langage intermédiaire appelé "bytecode" qui est interprété sur les différentes machines. Il suffit donc de communiquer le bytecode et de disposer d'un interpréteur de bytecode pour obtenir l'exécution d'un programme Java. **[8]**

Les programmes en Java ont l'extension .java, et le bytecode généré a l'extension .class (voir figures 4.1 et 4.2).

Figure 3.1Compilation d'un programme java

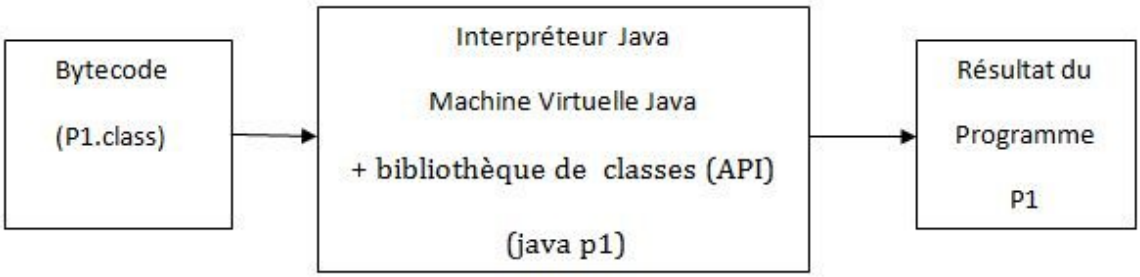

Figure 3.2 L'interprétation du bytecode

3.3 Outils de développement

3.3.1 Eclipse

Eclipse est un environnement de développement intégré(IDE) dont le but est de fournir une plate-forme modulaire pour permettre de réaliser des développements informatiques.

Eclipse utilise énormément le concept de modules nommés "plug-ins" dans son architecture.

Les principaux modules fournis en standard avec Eclipse concernent Java mais des modules sont en cours de développement pour d'autres langages notamment C++, Cobol, mais aussi pour d'autres aspects du développement (base de données, conception avec UML, ...). Ils sont tous développés en Java soit par le projet Eclipse soit par des tiers commerciaux ou en open source.

Bien que développé en Java, les performances à l'exécution d'Eclipse sont très bonnes car il n'utilise pas Swing pour l'interface homme-machine mais un toolkit particulier nommé SWT associé à la bibliothèque JFace. SWT (Standard Widget Toolkit) est développé en Java par IBM en utilisant au maximum les composants natifs fournis par le système d'exploitation sous jacent.

SWT et JFace sont utilisés par Eclipse pour développer le plan de travail (Workbench) qui organise la structure de la plate-forme et les interactions entre les outils et l'utilisateur.

Cette structure repose sur trois concepts : la perspective, la vue et l'éditeur. La perspective regroupe des vues et des éditeurs pour offrir une vision particulière des développements. En standard, Eclipse propose huit perspectives. Les vues permettent de visualiser et de sélectionner des éléments. Les éditeurs permettent de visualiser et de modifier le contenu d'un élément de l'espace de travail. **[9]**

3.3.1.1 Les points forts d'Eclipse

Eclipse possède de nombreux points forts qui sont à l'origine de son énorme succès dont les principaux sont :

- Une plate-forme ouverte pour le développement d'applications et extensible grâce à un mécanisme de plug-ins.
- Plusieurs versions d'un même plug-in peuvent cohabiter sur une même plate-forme.
- Un support multi langage grâce à des plug-ins dédiés : Cobol, C, PHP, C#, ...
- Support de plusieurs plate-formes d'exécution : Windows, Linux, Mac OS X, ...
- Malgré son écriture en Java, Eclipse est très rapide à l'exécution grâce à l'utilisation de la bibliothèque SWT.
- Un historique local des dernières modifications.

- La construction incrémentale des projets Java grâce à son propre compilateur qui permet en plus de compiler le code même avec des erreurs, de générer des messages d'erreurs personnalisés. **[10]**

3.3.1.2 Le plan de travail (Workbench)

Au lancement d'Eclipse, une seule fenêtre s'ouvre contenant le plan de travail (Workbench). Le plan de travail est composé de perspectives dont plusieurs peuvent être ouvertes mais une seule est affichée en même temps.
A l'ouverture, c'est la perspective "Ressource" qui est affichée par défaut.

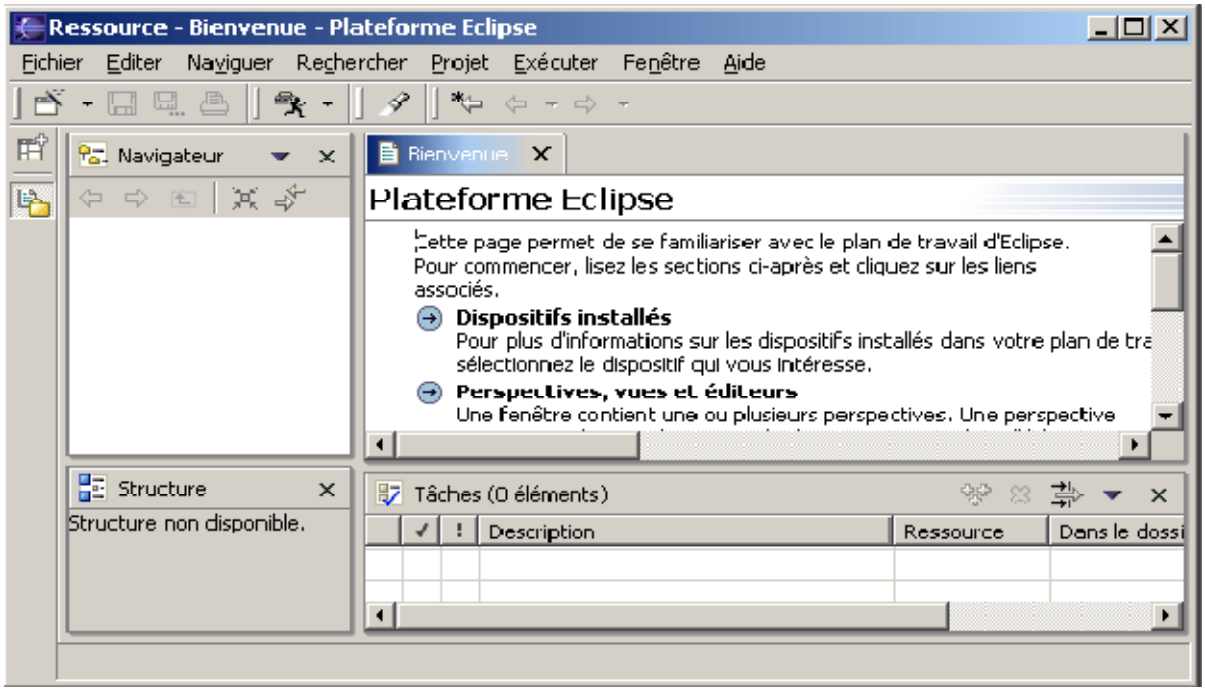

Figure 3.3 Perspective ressource

a. Les perspectives

Une perspective présente une partie du projet de développement selon un certain angle de vue.

Chaque perspective possède une icône qui permet de l'identifier plus rapidement.

Perspective	Icône	Rôle
Débogage		Débogueur
Java		Ecriture de code Java
Navigation Java		Navigation dans la hiérarchie et les éléments des classes
Hierarchie de type Java		
Développement de plug-in		Création de plug-ins
Ressource		Gestion du contenu de l'espace de travail
Synchronisation de l'équipe		
Exploration du référentiel CVS		Gestion du travail collaboratif avec CVS

Figure 3.4 les différentes perspectives d'éclipse

b. Les éditeurs

Il existe plusieurs éditeurs en fonction du type de l'élément qui est édité.

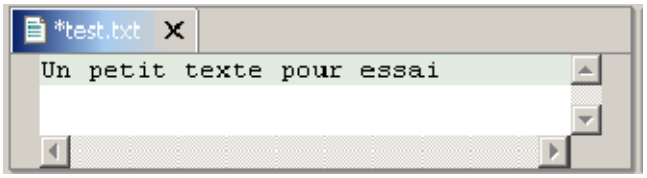

Figure 3.5 L'éditeur d'éclipse

L'onglet de l'éditeur contient le libellé de l'élément traité. Une petite étoile apparaît à droite de ce libellé si l'élément a été modifié sans être sauvegardé.
Pour fermer l'éditeur contenant l'élément en cours, il suffit de cliquer sur l'icône en forme de croix de l'onglet.
Une confirmation sera demandée en cas de fermeture alors que l'élément a été modifié sans sauvegarde.

Plusieurs raccourcis ont été ajoutés dans les éditeurs :

Raccourcis clavier	Rôle
Alt+flèche vers le haut / bas	Déplacement d'un ensemble de lignes sélectionnées
Ctrl +Alt+flèche vers le haut	Copie d'un ensemble de lignes sélectionnées
Ctrl+Maj+Entrée	Insérer une ligne au dessus de la ligne courante
Maj+Entrée	Insérer une ligne en dessous de la ligne courante
Ctrl+Maj+Y	Conversion du texte sélectionné en minuscule
Ctrl+Maj+X	Conversion du texte sélectionné en majuscule

Figure 3.6 raccourcis clavier d'Eclipse

c. Les vues

Les vues permettent de présenter des informations et de naviguer dans les ressources. Plusieurs vues peuvent être réunies dans une même sous fenêtre : dans ce cas, le passage d'une vue à l'autre se fait via un clic sur l'onglet concerné.

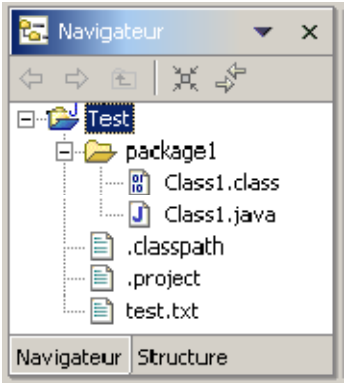

Figure 3.7 Vue Navigateur d'éclipse

d. Fermer le plan de travail

Pour fermer le plan de travail et donc quitter Eclipse, il y a deux possibilités :

- Fermer la fenêtre du plan de travail
- Sélectionner l'option "Quitter" du menu "Fichier"

Une boîte de dialogue permet de confirmation la fermeture de l'application.

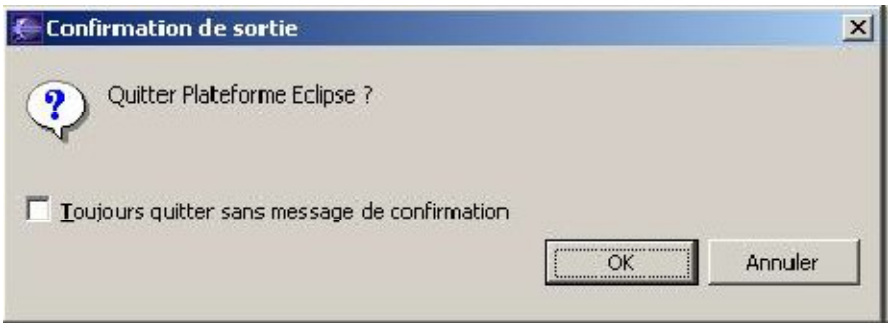

Figure 3.8 confirmation de fermeture d'éclipse

3.3.1.3 L'espace de travail (Workspace)

L'espace de travail est l'entité qui permet de conserver les projets et leur contenu. Physiquement c'est un répertoire du système d'exploitation qui contient une hiérarchie de fichiers et de répertoires. Il y a d'ailleurs un répertoire pour chaque projet à la racine de l'espace de travail.
Il est possible de parcourir cette arborescence et d'en modifier les fichiers avec des outils externes à Eclipse.

L'espace de travail contient tous les éléments développés pour le projet : il est possible de créer, de dupliquer, de renommer ou de supprimer des éléments. Ces opérations de gestion sont réalisées dans la vue "Navigateur" de la perspective "Ressource". **[10]**

3.3.2 Microsoft SQL serveur

Développé et commercialisé par la société Microsoft, SQL Server est un système de gestion de base de données relationnelles (abrégé en SGBDR) qui :

- gère le stockage des données pour les transactions et l'analyse ;
- répond aux requêtes des applications clientes

On peut utiliser SQL Server pour traiter des transactions, stocker et analyser des données, et développer de nouvelles applications.
SQL Server est une suite de produits et de technologies qui répondent aux exigences de stockage des données des environnements OLTP et OLAP.
SQL Server gère les bases de données **OLTP** et **OLAP**.

Les données d'une base de données OLTP sont généralement organisées en tables relationnelles afin de réduire les informations redondantes et d'accroître la vitesse des mises à jour. SQL Server autorise un grand nombre d'utilisateurs à effectuer des transactions et à changer les données en temps réel dans les bases de données OLTP. **[11]**

3.3.2.1 Outils de SQL Server

1. **Moteur de base de données**

   Le moteur de base de données permet de stocker, traiter et sécuriser les données.

2. **Analysis Services - Données multidimensionnelles**

   Analysis Services permet de créer et gérer des structures multidimensionnelles qui contiennent des données agrégées à partir d'autres sources de données, telles que des bases de données relationnelles.

3. **Integration Services**

   Integration Services est une plateforme permettant de créer des solutions d'intégration de données qui autorisent les processus d'extraction, de transformation et de chargement (ETL) pour le Data Warehouse.

4. **Reporting Services**

   Reporting Services fournit des fonctionnalités Web de création de rapports d'entreprise.

3.3.2.2 Types de bases de données

- OLTP (On-Line Transactional Processing) – Moteur SQL Server - est un service central qui permet de stocker, traiter et sécuriser les données.

- OLAP (On-Line Analytical Processing) - Moteur Analysis Services - a pour but d'organiser les données à analyser par domaine/thème et d'en ressortir des résultats pertinents pour le décideur.

3.3.2.3 Liste des bases de données

1. **MSDB**

   La base de données **msdb** est utilisée par l'Agent SQL Server pour planifier des alertes et des travaux, ainsi que par d'autres fonctionnalités telles que Service Broker et la messagerie de base de données.

2. **MASTER**

   La base de données **master** contient l'intégralité des informations système relatives à un système SQL Server.

3. **MODEL**

   La base de données **model** fait office de modèle pour toutes les bases de données créées sur une instance de SQL Server.

4. **TEMPDB**

   La base de données système **tempdb** est une ressource globale disponible pour tous les utilisateurs connectés à l'instance de SQL Server.

3.4 Présentation de quelques interfaces

3.4.1 Interface d'authentification

Au niveau de cette interface, nous avons trois modes d'accès.

1. **Le premier mode (Administrateur)** : l'accès n'est autorisé que pour l'administrateur, et cela est sécurisé par un mot de passe et un pseudo qu'il doit saisir avant de pouvoir accéder à l'application.

- En choisissant le ce mode d'accès, l'administrateur dois saisir son pseudo et son mot de passe, ensuite il doit cliquer sur le bouton login.
- Le système effectue une recherche dans la base de données vérifiant les données saisies.
- Dans le cas ou les données saisies sont correctes l'administrateur accède a l'application, et dans le cas contraire un message d'erreur sera afficher pour informer l'administrateur que sont mot de passe et/ou son pseudo est incorrect.

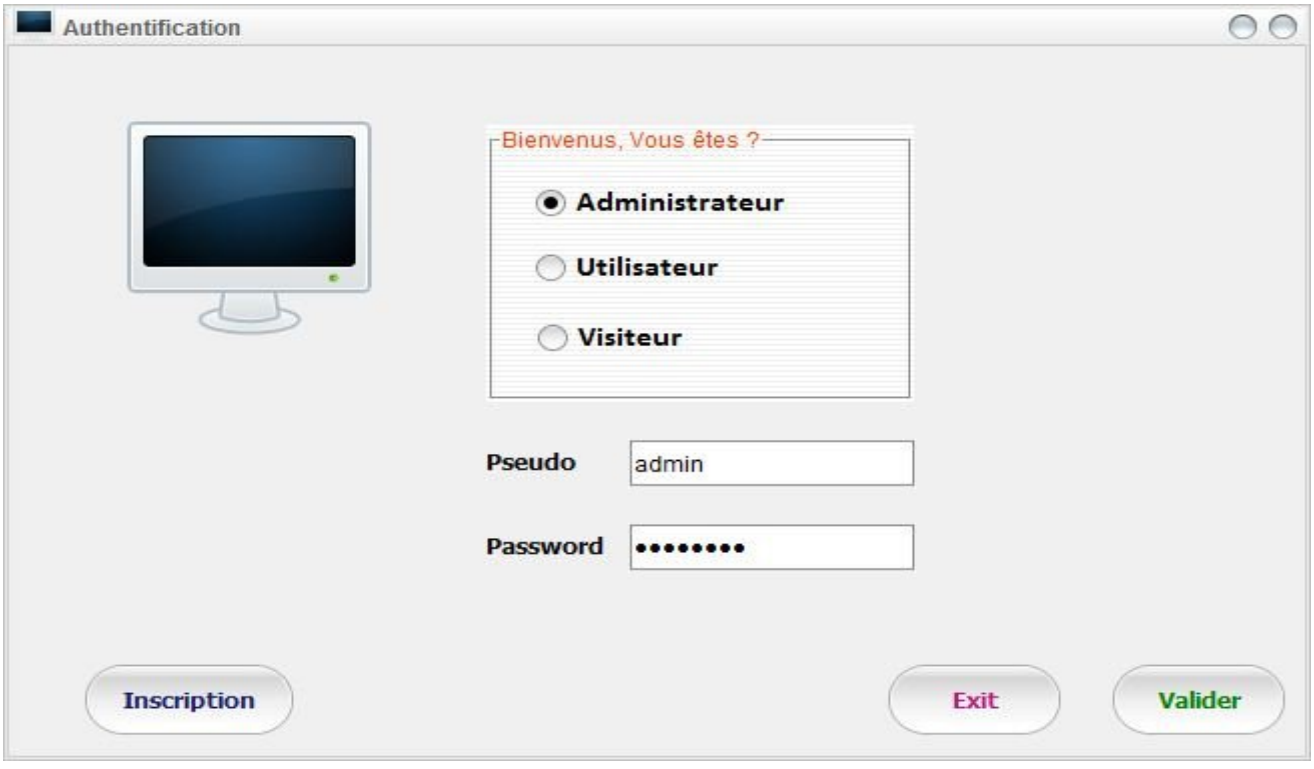

Figure 3.9 interface "d'authentification"

Figure 3.10 interface "message d'erreur d'authentification"

2. **Le deuxième mode (Utilisateur)** : en choisissant le mode d'accès **Utilisateur,** on peut accéder à l'application en saisissant le pseudo et le mot de passe, et en cliquant sur le bouton **login**.

- Apres cela, le système effectue une recherche dans la base de données.
- Dans le cas ou le pseudo et le mot de passe sont enregistrés, l'utilisateur accède à l'application.
- Dans le cas contraire un message d'erreur est affiché pour l'informer que sont mot de passe et/ou son pseudo est incorrect.

3. **Le troisième mode (Visiteur) :** le visiteur accède à l'application sans authentification.

3.4.2 Interface d'accueil Utilisateur

Dans cette interface, l'utilisateur peut sélectionner une des wilayas d'Algérie dans le combobox **Choisir Wilaya**.

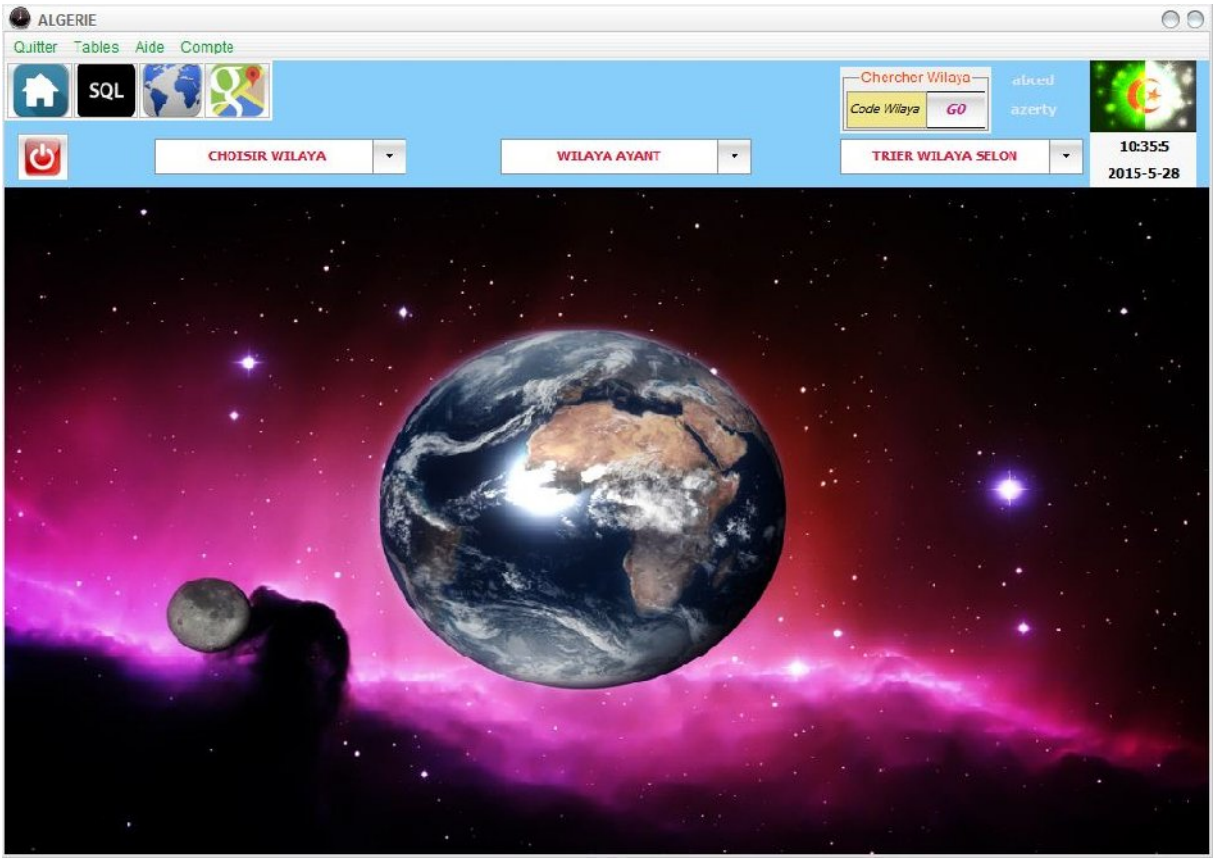

Figure 3.11 interface "d'accueil Utilisateur"

Apres la sélection d'une wilaya, une fenêtre s'affiche, elle comporte :

- Nom de wilaya.
- les renseignements : code de la wilaya, sa superficie, sa position géographique, le nombre d'habitants selon le dernier recensement, le nombre de communes, le nombre de daïras, et sa distance à la capitale.
- Un ensemble de photos des principaux sites touristiques et historiques de chaque wilaya.
- Les wilayas avoisinantes.
- Un lien vers Google Maps, permettant de voir le trajet entre cette wilaya et la capitale.

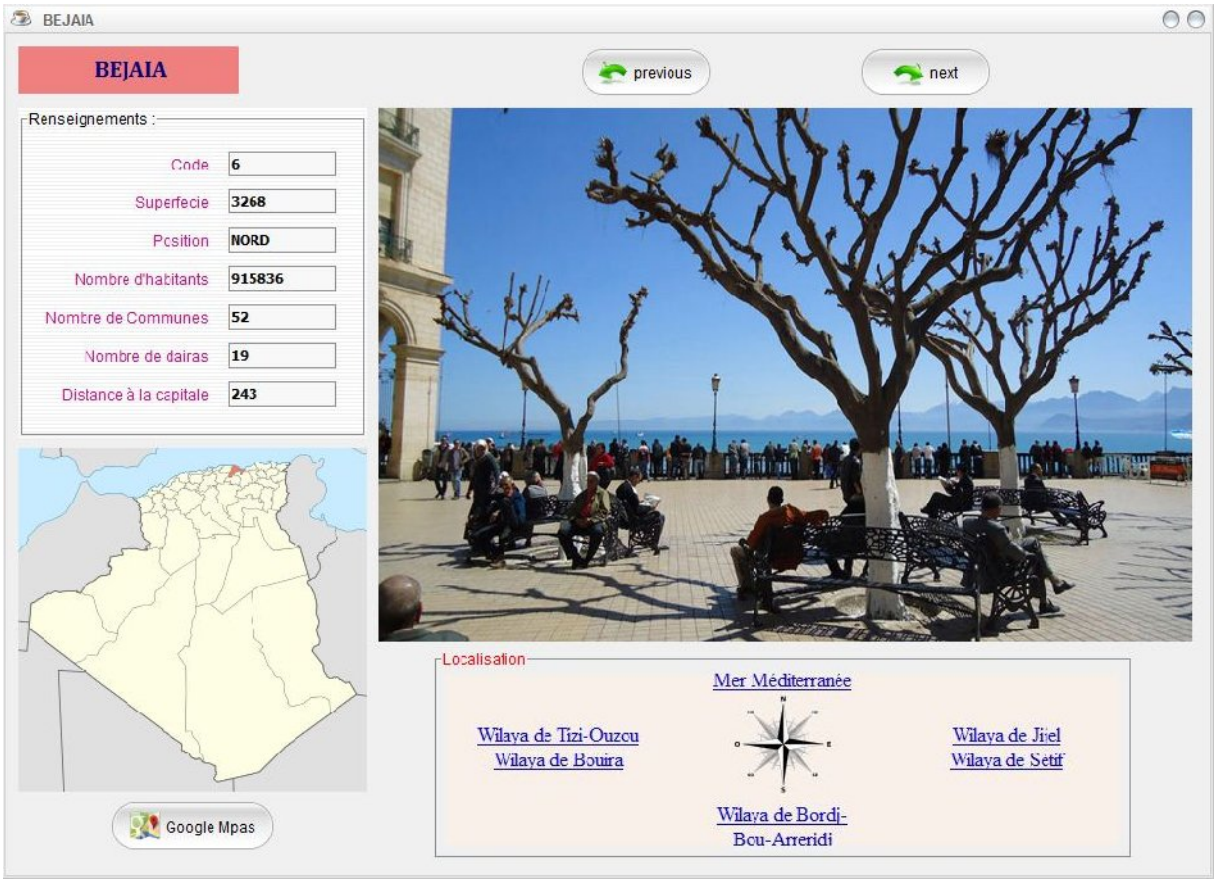

Figure 3.12 interface " choisir Wilaya "

- Dans le combobox **Wilaya Ayant**, le client peut choisir une proposition parmi les informations suggérées. Comme informations on trouve par exemple : (la wilaya ayant la plus grande superficie, celle qui a le plus grand nombre d'habitants, celle qui sont situées au nord, …)
- Ces dernières sont enregistrées dans la base de données, que le client peut interroger.
- En choisissant une proposition dans la liste, une nouvelle fenêtre s'affiche et la wilaya concernée est affichée avec toutes ses informations dans une table bien ordonnée.

Figure 3.13 interface "Wilaya ayant "

- Dans le combobox **Trier Wilaya Selon**, le client aura aussi le choix entre plusieurs propositions pour trier les wilayas selon différents critères comme par exemple : la position géographique, le nombre de communes, ...
- Apres que le client fait son choix dans la liste du combobox, une nouvelle fenêtre s'affiche comportant la liste des wilayas triée selon le critère choisit.

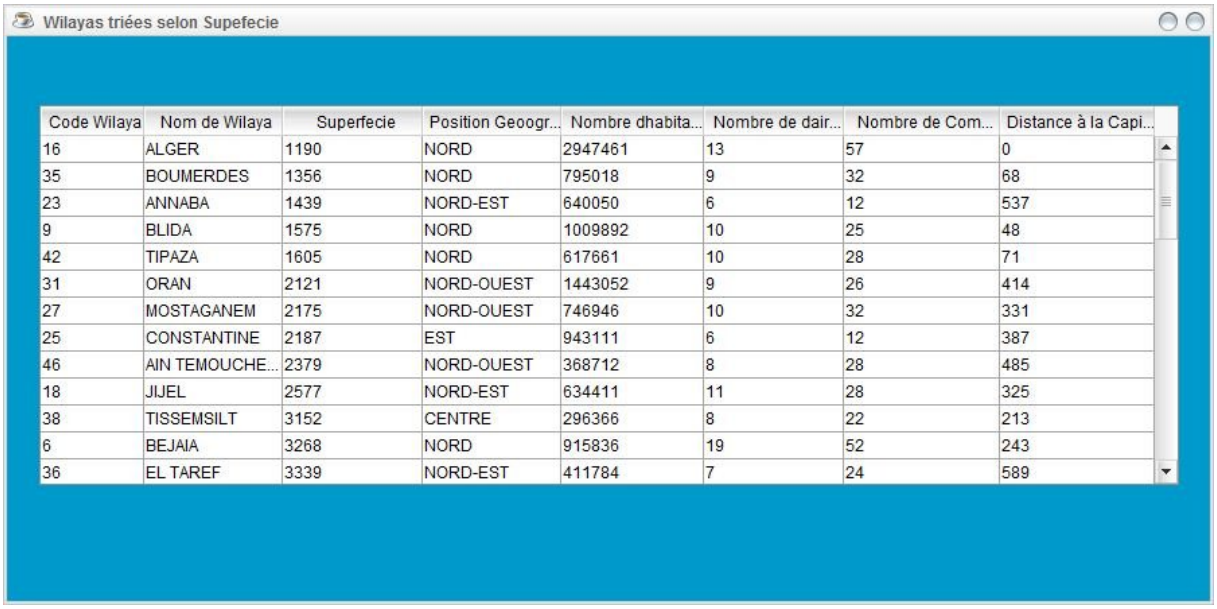

Figure 3.14 interface "Trier Wilaya Selon"

- Dans la barre du menu le client dispose aussi du menu **Compte,** dans lequel il a la possibilité de changer ses coordonnés.

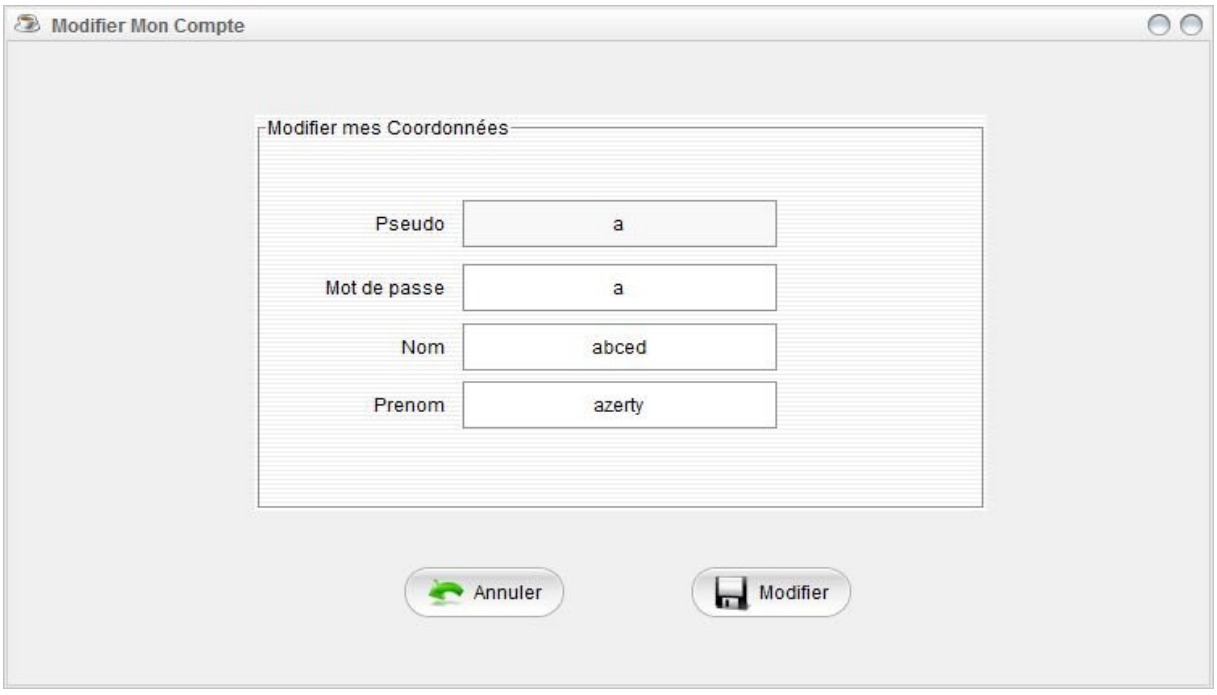

Figure 3.15 interface "Modifier Mon Compte"

3.4.3 Interface d'accueil Administrateur

Apres l'authentification de l'administrateur l'interface d'accueil sera affichée.

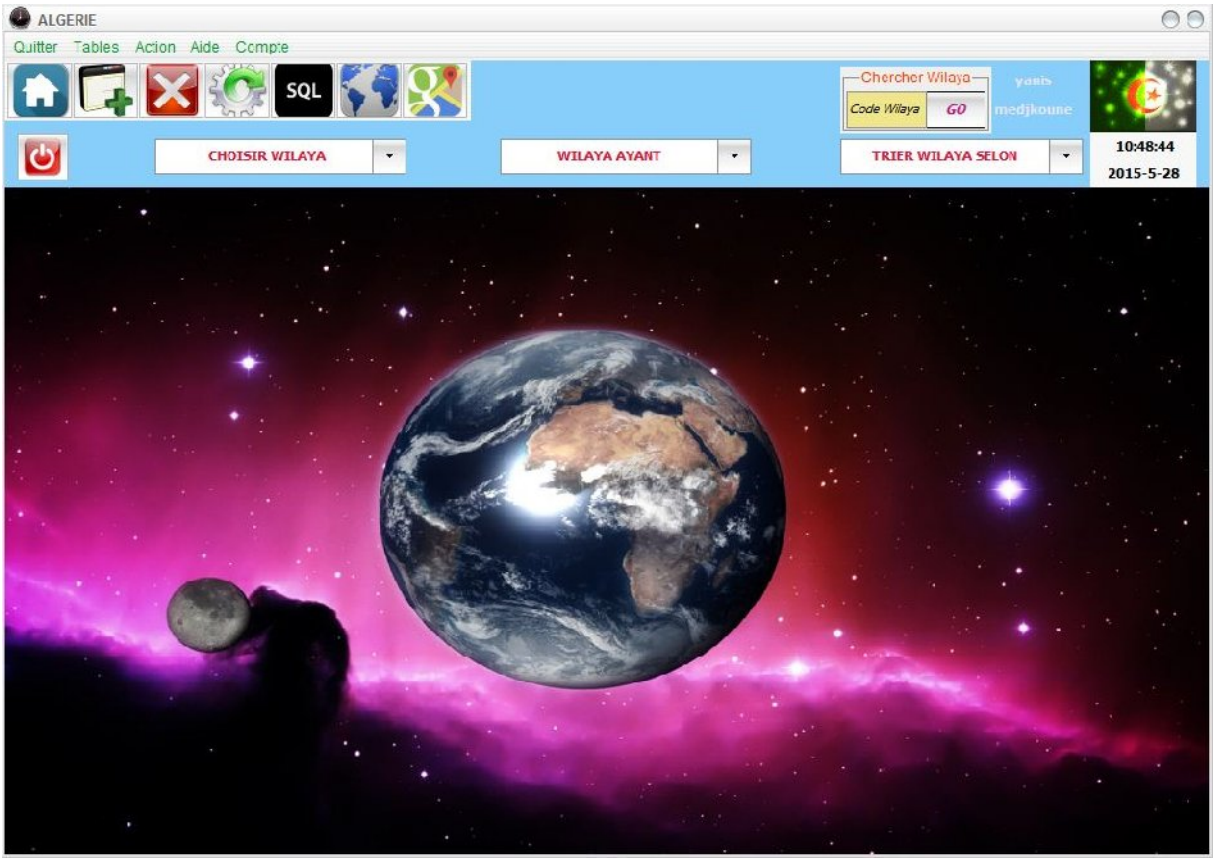

Figure 3.16 Interface" d'accueil Administrateur"

En plus des fonctionnalités offertes au client, l'administrateur dispose de quelques privilèges. Nous citons :

3.4.3.1 Interface Ajouter Wilaya

- Dans cette interface, l'administrateur a la possibilité d'ajouter une nouvelle wilaya à la liste, en remplissant les champs (Code wilaya, nom, superficie, …).
- Apres avoir rempli tous les champs d'une manière correcte, l'administrateur clique sur le bouton **Enregistrer** pour sauvegarder cette wilaya dans la base de données. Un message de confirmation d'enregistrement sera affiché.
- Dans le cas ou l'administrateur fait une erreur en remplissant les champs, par exemple au lieu d'écrire un entier dans un champ, il écrit un caractère ou quelque chose d'autre, un message d'erreur lui sera affiché.
- Apres avoir enregistrer une wilaya, au lieu d'effacer les champs manuellement pour pouvoir ajouter une autre wilaya, l'administrateur n'a qu'a cliquer sur le bouton **Effacer** , et tous les champs seront effacés.

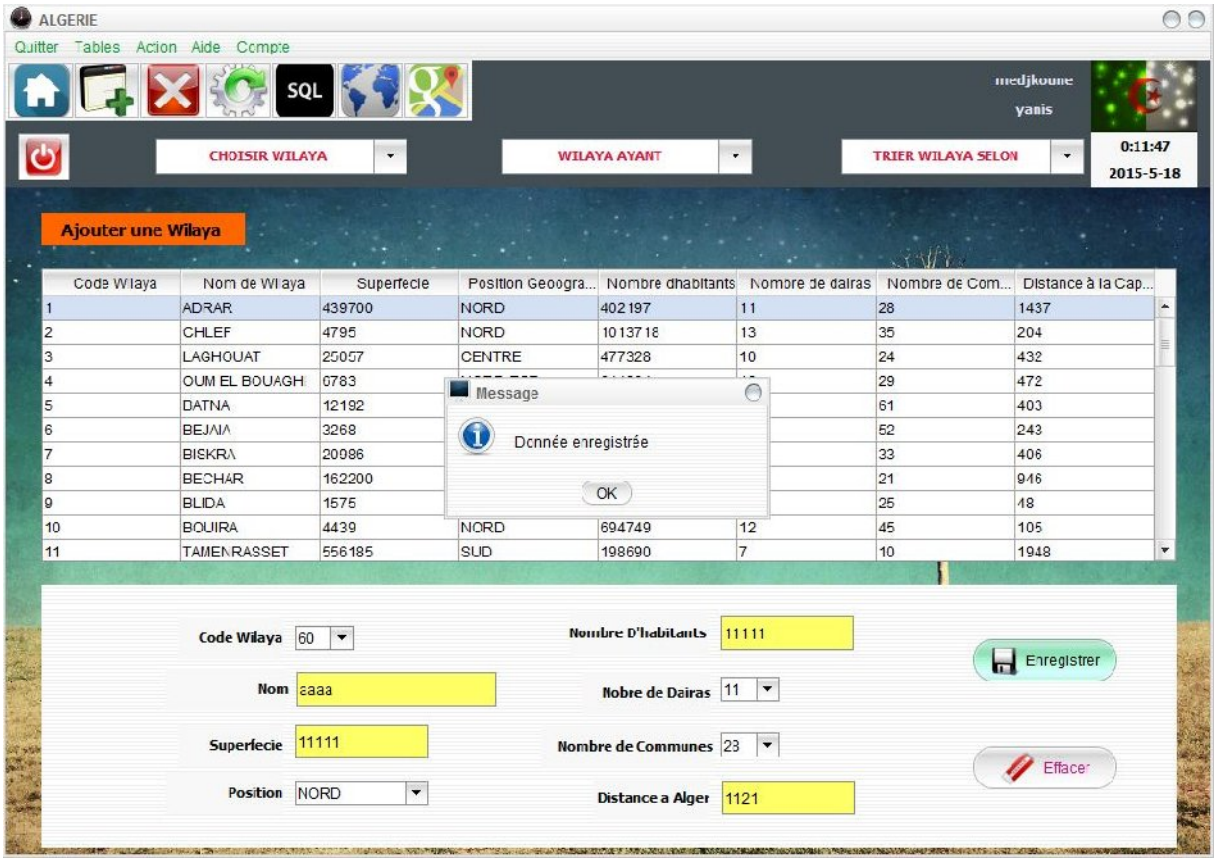

Figure 3.17 interface "Ajouter une wilaya"

3.4.3.2 Interface Mettre à jour Wilaya

- Dans cette interface, l'administrateur à la possibilité de faire une mise à jour de chaque wilaya.
- Pour pouvoir réaliser cette opération, il doit cliquer sur une ligne de la liste des wilayas, et qu'il en choisit une. Ensuite les données de cette dernière seront affichées chacune dans son champ, et l'administrateur pourra changer la donnée qu'il veut.
- Apres modification, il clique sur le bouton **Modifier** et la wilaya sera mise à jour, et un message de confirmation de modification sera affiché.

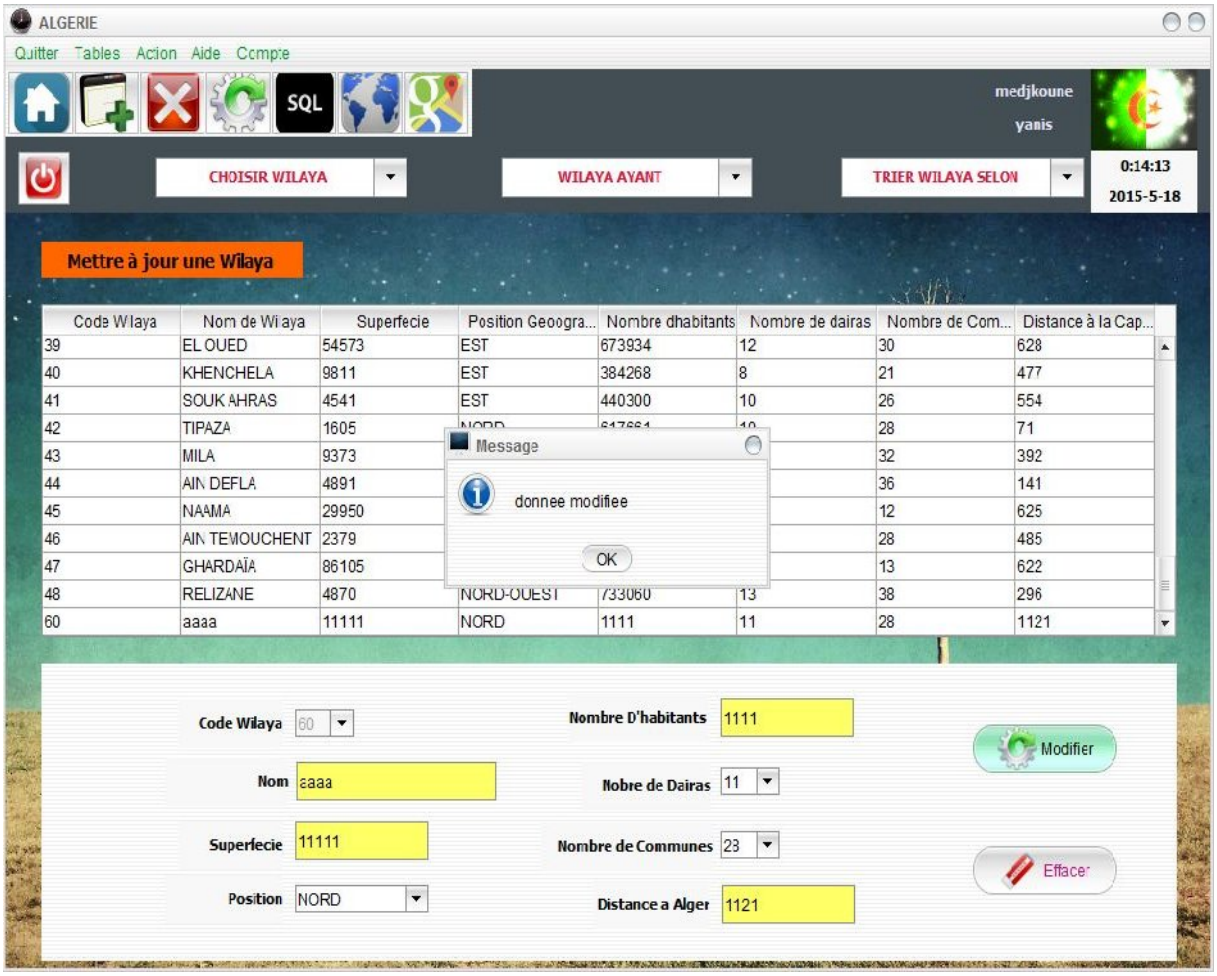

Figure 3.18 interface "Modifier Wilaya"

3.4.3.3 Interface Requête SQL

- Dans cette interface on peut interroger la base de données, et cela en écrivant une requête SQL dans le champ réservé à ce but, ensuite en cliquant sur le bouton **Exécuter** le résultat sera affiché.
- Dans le cas ou la requête n'est pas juste, un message d'erreur sera affiché pour spécifier l'erreur commise.

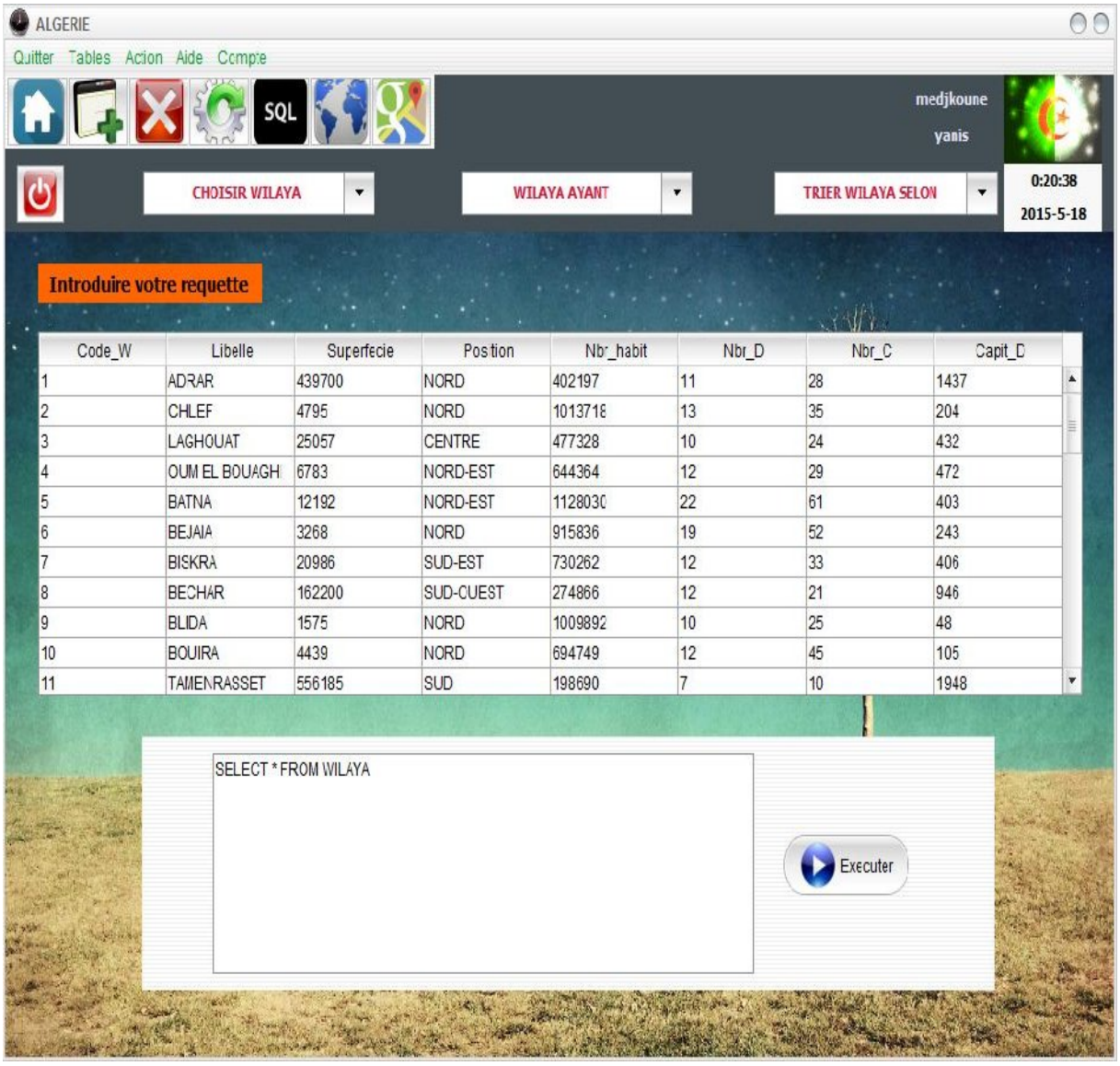

Figure 3.20 interface "requête SQL"

3.4.3.4 Interface Table Voisinage

Dans cette interface, l'administrateur à la possibilité de :

1. Enregistrer le voisinage (wilayas avoisinantes) de chaque wilaya : cela en choisissant le code de la wilaya, le code de la wilaya avoisinante et sa localisation, et enfin en cliquant sur le bouton **Enregistrer.**

2. Supprimer le voisinage : l'administrateur doit cliquer sur la ligne qu'il désire supprimer, en suite il clique sur le bouton **Supprimer**.
   Un message de confirmation de la suppression est affiche, si il veut réellement supprimer la ligne il clique sur le bouton **Oui**, sinon sur le bouton **Non**.

3. Mettre à jour le voisinage : l'administrateur doit cliquer sur une ligne, automatiquement ses données (code wilaya, code wilaya avoisinante, localisation) seront affichées dans l'espace mettre à jour, ensuite il pourra changer la localisation en choisissant dans le combobox dédié.
   Apres la modification il doit cliquer sur le bouton **Update** pour enregistrer les modifications.

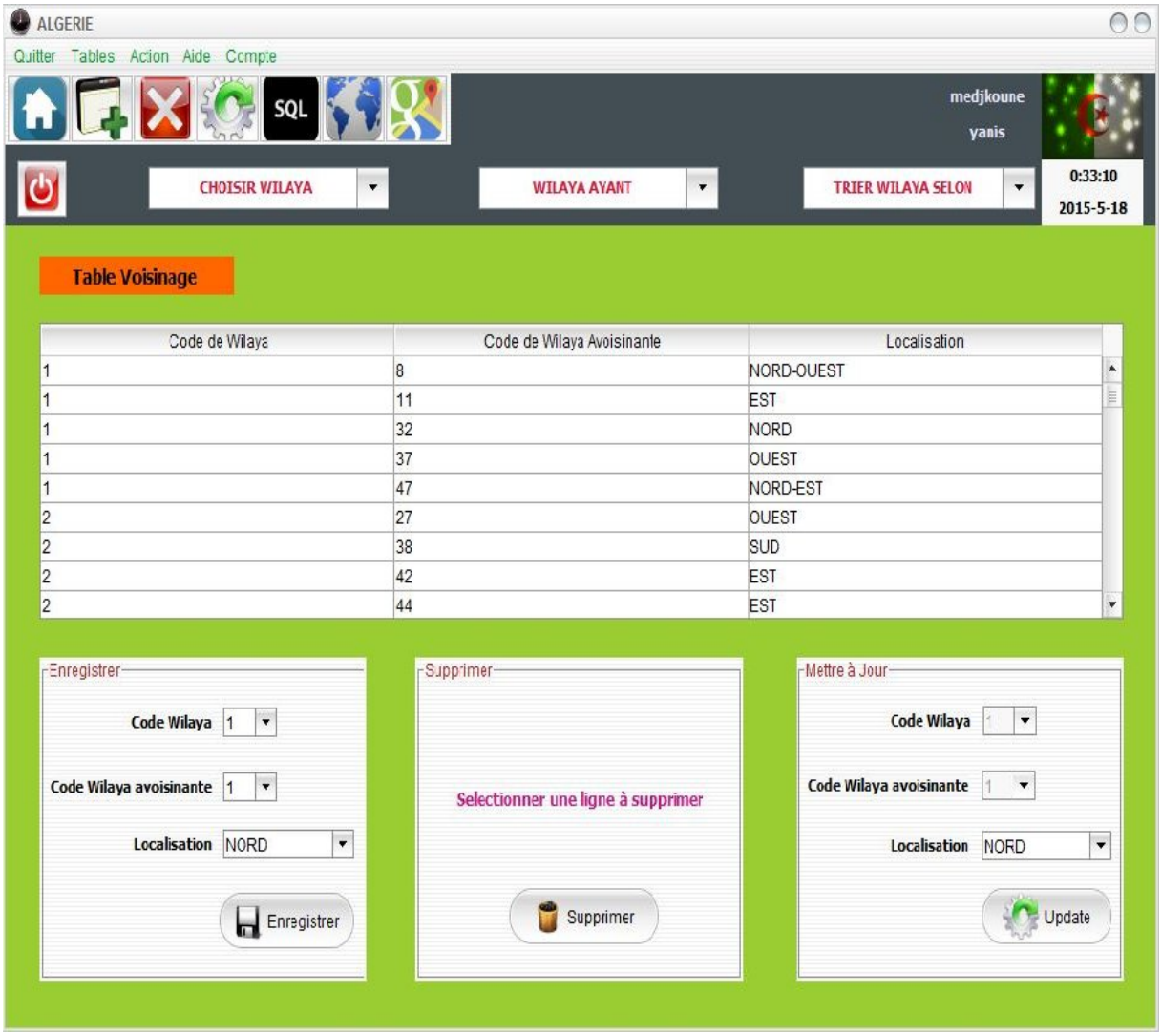

Figure 3.21 "interface Table voisinage"

3.4.3.5 Interface Gestion Utilisateur

Dans cette interface l'administrateur a la possibilité d'ajouter et de supprimer des utilisateurs.

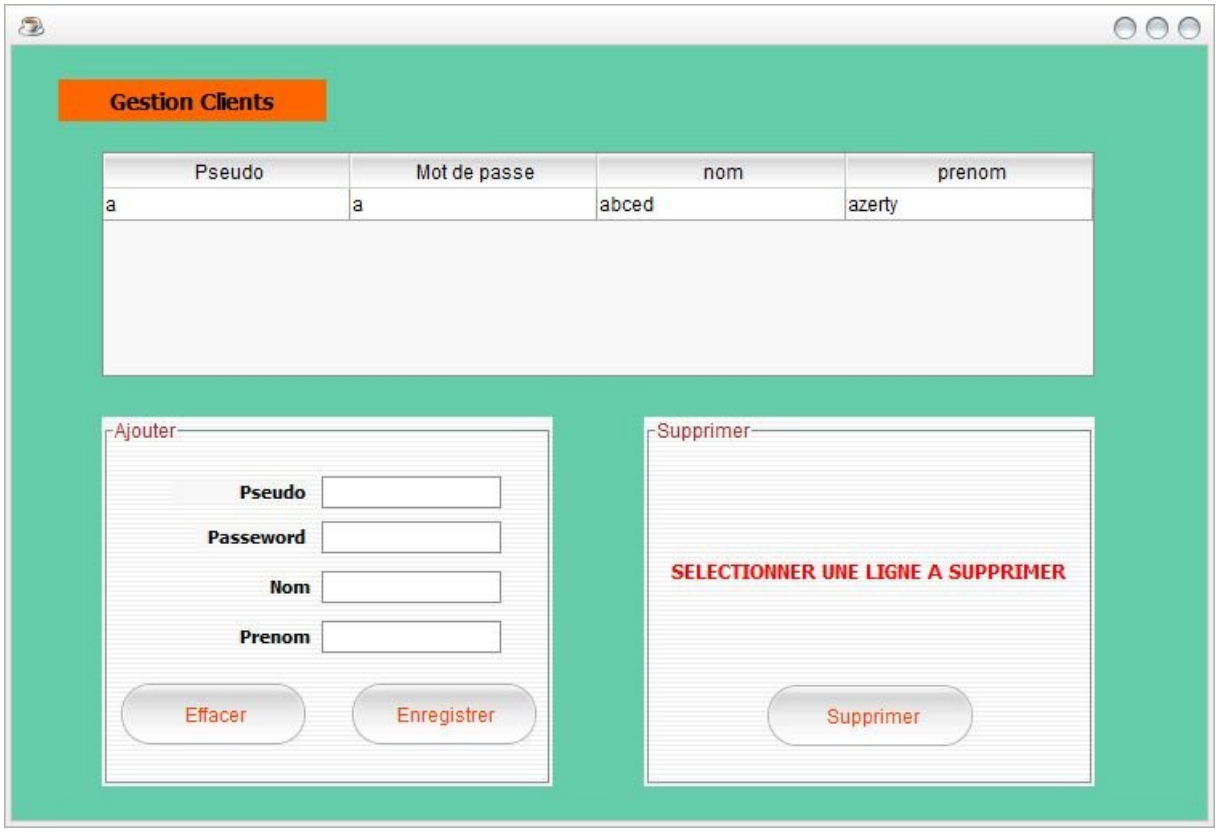

Figure 3.21 interface "Gestion Utilisateurs"

3.5 Conclusion

Ce chapitre été consacré à la phase de la réalisation. Cette phase est le fruit de nos efforts tout au long de ce projet.

La réalisation de ce travail nous a permis de mettre en œuvre nos connaissances théoriques sur les bases de données et le langage java qui un langage incontournable dans le domaine de la programmation.

Conclusion générale et perspectives

Dans notre travail, nous avons présenté les différentes étapes de développement et de la réalisation de notre application web.
Nous nous sommes appuyés durant tout le processus de modélisation sur UML, qui est un langage de modélisation très utilisé pour illustrer notre démarche.
Le but de l'application été de rassembler le maximum d'information sur chacune des wilayas algériennes et de les mettre à la disposition des utilisateurs.

La réalisation de ce projet nous a permis d'acquérir de l'expérience dans le domaine de la programmation avec le langage java, qui est un langage très puissant et très utilisé, et aussi la manipulation de SQL serveur qui est un système de gestion de base de données performant.

Nous espérons que cette application sera bénéfique pour ses utilisateurs qui cherchent des informations relatives à des wilayas algériennes.

Il faut toutefois noter que ce projet n'est qu'une initiative pour la réalisation d'une application qui sera à la hauteur des exigences de tous les internautes ; effectivement, il reste encore du travail à faire afin de réaliser une application complète.

Parmi les perspectives qu'on espère ajouter dans les futurs projets :

- Ajouter un forum de discussion pour les utilisateurs.
- donner aux utilisateurs la possibilité de commenter les photos.
- Ajouter un espace pour l'actualité de chaque wilaya.
- rendre cette application disponible pour le mobile.

Bibliographie

[1] Pascal Roques, "UML 2 par la pratique Etudes de cas et exercices corrigés", édition Eyrolles, 2006.

[2] Pascal Roques Franck Vallée, "UML 2 en Action De l'analyse des besoins à la conception", édition Eyrolles, 2007.

[3] Joseph Gabay David Gabay, "UML2 analyse et conception", édition Dunod, 2008.

[4] Xavier Blanc Isabelle Mounier, "UML 2 pour les développeurs», édition Eyrolles, 2009

[5] P. Muller, N.Gerstner, " Modélisation objet avec UML", édition, 2004.

[6] G. Roy, " conception des bases de données avec UML ", 2009.

[7] Ch.Soutou, " UML2 pour les bases de données", Eyrolles, édition ,2006.

[8] Michel Divay, "LA PROGRAMMATION OBJET EN JAVA", édition DUNOD, 2006.

[9] Timothy R. Fisher, "Java l'essentiel du code et des commandes", édition CampusPress, 2009.

[10] Jean Michel DOUDOUX, "Développons en Java avec Eclipse", 2004

[11] Jérôme Gabillaud"SQL Server 2005 SQL, Transact SQL", edition ENI, 2005

Résumé

Le présent travail porte sur la conception et la réalisation d'une application web pour la gestion des wilayas en Algérie. Le résultat attendu de ce système est d'offrir à l'utilisateur le maximum d'informations concernant les wilayas algériennes.
A la phase de conception et de modélisation du SI, on a utilisé le langage UML en suivant les phases du processus du développement UP.
Pour réaliser ce travail et arriver à notre objectif, nous avons utilisé comme serveur local, Microsoft SQL Serveur, comme langage de programmation, JAVA, et comme environnement de développement intégré, ECLIPSE.

Abstract

This work focuses on the design and implementation of a web application for managing wilayas in Algeria. The expected result of this system is to give the user as much information about the Algerian wilayas.
In the design and modeling phase of the SI was used UML following the phases of the development process UP.
To carry out this work and reach our goal, we have used as a local server, Microsoft SQL Server, as a programming language, JAVA, and as integrated development environment, ECLIPSE.