



Université Mohammed Premier
Ecole Nationale des Sciences Appliquées
Al Hoceima



PROJET DE FIN D'ETUDE

Filière : Génie informatique

Titre

*Réalisation d'une application permettant la montée de version
d'Amplitude*

Réalisé par

JAFINE Boutayna

OUAJJINI Zineb

Encadré par

M.ADISSA Mohamed Fayçal (Sopra)

Mme. MORADI Fouzia (Professeur à l'ENSAH)

Soutenu le 03/07/2018 devant le Jury :

- Pr. EL AKKAD Nabil (PA)
- Pr. MOQQADDEM Safaa (PA)
- Pr. MORADI Fouzia (PA)



بِسْمِ اللّٰهِ الرَّحْمٰنِ الرَّحِيْمِ

قَالُوا سُبْحَانَكَ لَا عِلْمَ لَنَا إِلَّا مَا عَلَمْتَنَا إِنَّكَ أَنْتَ الْعَلِيْمُ الْحَكِيمُ ٢٢
سورة البقرة

صَدْقَ الْعَظِيْمِ

Remarques

Page réservée aux remarques



Avant-propos

Noms : JAFINE Boutayna, OUAJJINI Zineb

Etablissement : L'Ecole Nationale des Sciences Appliquées d'Al Hoceima

Intitulé du travail : Réalisation d'une application permettant la montée de version d'Amplitude.

Organisme d'accueil : Sopra Banking Software Morocco

Adresse de l'organisme d'accueil :

Casablanca MARINA - Ivoire 5
Boulevard des Almohades
20000 Casablanca – Morocco
Phone : +212(0)5 22 64 57 48

Nom et prénom de l'encadrant du projet dans l'établissement d'accueil :

M. ADISSA Mohamed Fayçal

Nom et prénom de l'encadrante du projet à l'ENSAH :

Mme. MORADI Fouzia

Date de début et de fin du stage : 01/03/2018 – 31/08/2018

Dédicace :

OUAJJINI Zineb

Du profond de mon cœur, je dédie ce travail à tous ceux qui me sont chers,

A ma chère mère

Aucune dédicace ne saurait exprimer mon respect, mon amour éternel et ma considération pour les sacrifices que vous avez consenti pour mon instruction et mon bien être.

Je vous remercie pour tout le soutien et l'amour que vous me portez depuis mon enfance et j'espère que votre bénédiction m'accompagne toujours.

Que ce modeste travail soit l'exaucement de vos vœux tant formulés, le fruit de vos innombrables sacrifices. Puisse Dieu, le Très Haut, vous accorder santé, bonheur et longue vie.

A la mémoire de mon père

Ce travail est dédié à mon père, décédé trop tôt, qui m'a inspiré par ses pensées et sa philosophie de la vie.

J'espère que, du monde qui est sien maintenant, il apprécie cet humble geste comme preuve de reconnaissance de la part d'une fille qui a toujours prié pour le salut de son âme. Puisse Dieu, le tout puissant, l'avoir en sa sainte miséricorde !

À mes adorables sœurs Fadoua et Nouhaila pour leurs encouragements permanents, et leur soutien moral. À mon frère Anass pour son appui et son encouragement.



Dédicace :

JAFINE Boutayna

Je dédie ce modeste travail et ma profonde gratitude à tous ceux qui ont sacrifié pour m'offrir les conditions propices à ma réussite :

A l'âme de mon père

J'ai tant souhaité que tu sois avec moi ce jour-là. Certes j'ai été seule, mais ton visage et ton sourire sont toujours devant mes yeux... J'entends encore ta voix qui me dit ne t'en fais pas ma chérie tout ira bien.

Permettez-moi de t'exprimer mon grand amour, mon attachement et ma plus haute considération pour ta personne. Je suis très fière d'être ta fille et de pouvoir enfin réaliser, ce que tu as tant espéré et attendu de moi.

Que Dieu ait ton âme en sa sainte miséricorde

A Ma Chère Maman

Si Dieu a mis le paradis sous les pieds des mères, ce n'est pas pour rien.

Aucune dédicace ne saurait être assez éloquente pour exprimer ce que tu mérites pour tous les sacrifices que tu n'as cessé de me donner depuis ma naissance, durant mon enfance et même à l'âge adulte. La source de tendresse et l'exemple du dévouement qui n'a pas cessé de m'encourager et de prier pour moi. Ta prière et ta bénédiction m'ont été d'un grand secours pour mener à bien mes études.

Que Dieu, le tout puissant, te préserve et t'accorde santé, longue vie et bonheur.

A mes adorables sœurs pour leurs encouragements permanents, et leur soutien moral.

A mes frères pour leur appui et leur encouragement.

À tous ceux qui sont proches de mon cœur et dont je n'ai pas cité le nom.

Remerciement :

Avant tout, nous remercions ALLAH LE TOUT PUISSANT de nous avoir donné la force et le courage d'arriver là.

Nous tenons également à remercier toutes les personnes qui ont contribué de loin ou près au succès de ce travail et qui nous ont aidé lors de la rédaction de ce rapport.

Nous profitons de cette occasion pour remercier notre encadrant pédagogique au sein de l'ENSAH, Madame MORADI Fouzia, pour le savoir-faire qui nous a transmis tous au long de notre formation au sein de l'ENSAH, pour ses précieux conseils, son encouragement et surtout pour l'effort qu'elle a fourni et son assistance durant l'élaboration de ce rapport

Nous remercions également notre tuteur de stage monsieur ADISSA Mohamed Fayçal chef de pôle à Sopra Banking Software, pour son assistance quotidienne, pour la confiance qu'il nous a témoignée et ses encouragements, qui nous ont aidés à accomplir ce travail dans les meilleures conditions

A la même occasion, nous aimerais en profiter et remercier l'équipe de travail : nos collègues, Monsieur MOUAOU Youssef, Monsieur ERBATTI Amine et Monsieur MENGAT Mohamed Amin pour le transfert de compétences qui nous a été très utile afin de nous intégrer et faire une mise à niveau par rapport au métier ainsi pour leurs sens de collaboration et de partage d'expériences.

De même, nous tenons à témoigner notre gratitude à tous nos enseignants à l'Ecole Nationale des Sciences Appliquées, qui ont contribué à notre formation professionnelle

Sans oublier de remercier tous les membres du jury qui nous ont fait l'honneur d'accepter d'évaluer notre travail.



Résumé :

Le présent document constitue le fruit de travail accompli dans le cadre du projet de fin d'études au sein de la société de l'Entreprise de Service du Numérique (ESN)« Sopra Banking Software Morocco ».

Parmi les solutions qu'offre le groupe SOPRA STERIA, on trouve le Core Banking « Amplitude » qui représente l'activité principale du centre de services de Casablanca. Il offre une large gamme d'applications couvrant l'ensemble des métiers d'une banque universelle.

Depuis le lancement du Progiciel Amplitude, plusieurs mises à niveau de ce dernier ont été effectuées, afin de répondre aux fortes évolutions connues par le marché bancaire au fil des années.

Pour migrer d'une ancienne version d'Amplitude à une version plus récente, le pôle Montée de Version doit construire un kit de migration spécifique à chaque client selon sa base de données et ses besoins.

La construction du kit est un processus long, manuel et répétitifs comportant plusieurs étapes qui se déroulent en mode console. C'est dans ce cadre que s'inscrit le périmètre de notre projet de fin d'étude ayant pour objectif la mise en place d'une application java desktop permettant la Montée de version d'Amplitude.

Afin de mener à bien la mission qui nous a été confiée, il était convenable de faire une analyse de l'environnement, ainsi qu'une analyse de l'existant afin de comprendre son fonctionnement pour dégager les spécifications et les objectifs à atteindre pour l'application.

Ensuite, la phase de conception du projet a été entamée par l'élaboration d'un ensemble de diagrammes modélisant la future solution.

Le développement étant amorcé, le test et la qualification de la solution ont été réalisés pour assurer le bon fonctionnement de l'application.

Abstract:

This document is the result of work done in the final year project hosted by Sopra Banking Software Morocco.

Among the solutions offered by the Sopra Steria group, we find the Core Banking "Amplitude" which represents the main activity of the Casablanca branche.

It offers a wide range of applications covering all dealings of a universal bank.

Since Amplitude's first release, several upgrades have been made to respond to the strong changes in the banking market over the years.

To migrate from an older version to a newer one, the MDV division must build a migration kit specific to each client.

The Kit construction process is long, manual and repetitive. It is within this scope that my end-of-study project is set with the aim of setting up a java desktop application that constructs the migration kit.

In order to accomplish the mission entrusted to us, it was essential to carry out an analysis of the work environment, as well as an analysis of the existing solution in order to understand its functioning, to identify the specifications and the objectives that should be met for by our solution.

Then, the design phase of the project was started by developing a set of diagrams modeling the future solution.

The development being initiated, the test and qualification of the solution were made to ensure the proper functioning of the application.

Sommaire :

Résumé :	8
Abstract :	9
Sommaire :	10
Liste des figures :	14
Liste des tableaux :	16
Liste des abréviations :	17
Introduction Générale :	19
Chapitre 1 : Cadre général du projet	21
1. Présentation de l'organisme d'accueil :	22
1.1 Sopra group :	22
1.2 Steria group :	22
1.3 Sopra Steria group :	23
1.3.1 Fiche technique :	24
1.3.2 Métiers et secteurs d'activités :	24
1.3.3 Filiales :	25
2. Présentation générale du progiciel Amplitude :	30
2.1 Evolution du SI bancaire :	30
2.2 Enjeux du système d'informations :	30

2.3 Urbanisation des SI :	31
2.4 Présentation d'Amplitude :	31
2.4.1 Positionnement d'Amplitude :	32
2.4.2 Clients Sopra Banking Amplitude :	32
3. Présentation du projet :	33
3.1 Etude de l'existant et problématique :	33
3.2 Solution :	34
3.3 Conduite et planification du projet :	35
3.3.1 La méthode d'ingénierie eMedia :	35
3.3.1.1 Présentation de la méthodologie d'ingénierie eMedia :	35
3.3.1.2 Le processus de production : Delivery Process :	37
3.3.2 Application :	38
3.3.3 Diagramme de GANTT :	38
4. Conclusion :	39
Chapitre 2 : Analyse et conception	40
1. Les diagrammes de Cas d'utilisation :	41
1.1 Diagramme de cas d'utilisation global :	41
1.2 Classement des cas d'utilisation	42
1.3 Description des cas d'utilisation :	46
1.3.1 Description de « Gestion d'exécution » :	47
1.3.2 Correction des erreurs liés à l'exécution des scripts	49
1.3.3 Gestion des serveurs	51
2. Diagramme de classe :	53

3. Diagramme de séquence :	55
4. Diagramme d'activité :	59
5. Cinématique des écrans :	59
6. Conclusion :	62
Chapitre 3 : Etude technique et mise en œuvre	63
1. Architectures de l'application :	64
1.1 Architecture logicielle :	64
1.2 Architecture Applicative :	66
1.2.1 Design Pattern	66
1.2.1.1 MVC	66
1.2.1.2 Inversion de contrôle :	67
1.2.1.3 Injection des dépendances :	67
1.2.1.4 Programmation orienté aspects :	67
2. Les Framework et technologies utilisés :	68
2.1 Technologie :	68
3. Les outils et environnements :	73
4. Réalisation et présentation des interfaces de l'application :	76
4.1 Interface d'authentification :	77
4.2 Interface pour choisir la base de données :	78
4.3 Interface pour la gestion des erreurs :	79
4.4 Interface d'historique :	80
5. Conclusion :	81

Bilan de stage :	82
1. Objectifs atteints :	82
2. Obstacles affrontés :	82
3. Enseignements personnels	82
Conclusion générale :	84
Annexes :	85
Webographie :	106

Liste des figures :

Figure 1: Logo Sopra.....	22
Figure 2: Logo Steria.....	22
Figure 3: Logo Sopra Steria	23
Figure 4: Portefeuille d'activités couvert par Sopra	23
Figure 5: Secteur d'activité	24
Figure 6: Logo Sopra Steria Consulting.....	25
Figure 7: Logo Sopra Hr Software	25
Figure 8: Logo Sopra Banking Software.....	25
Figure 9: Logo du Sopra Banking Software.....	25
Figure 10: Chiffres clés de Sopra Banking Software	25
Figure 11: Organisation du Centre de Services Amplitude.....	27
Figure 12: Organigramme du centre de service.....	28
Figure 13: Organigramme de l'équipe	28
Figure 14: les principales étapes de la construction de kit de migration	29
Figure 15: Positionnement Amplitude en zone MEA	32
Figure 16: Les clients du Core Banking Amplitude	32
Figure 17: Le processus de construction du Kit de migration	33
Figure 18: Schéma de la solution proposée	34
Figure 19: Logo eMedia	35
Figure 20: Processus de production eMedia.....	37
Figure 21: Diagramme de Gantt	39
Figure 22: Diagramme global de la gestion de montée en version d'Amplitude	41
Figure 23 : Diagramme de « Gestion des environnements ».....	42
Figure 24 : Diagramme de « Gestion des serveurs».....	43
Figure 25 : Diagramme de « Gestion des bases de données ».....	43
Figure 26: Diagramme de « Gestion d'exécution des scripts »	44
Figure 27: Diagramme de « Correction des erreurs liées à l'exécution des scripts ».....	45
Figure 28 : Diagramme de « Gestion d'historique de scripts »	46
Figure 29: Diagramme de classe	53
Figure 30: Diagramme des classes du système	54
Figure 31: Diagramme de séquence de la fonctionnalité "Exécuter requête"	56
Figure 32: Diagramme de séquence de la fonctionnalité "Exécution script"	57
Figure 33: Diagramme de séquence de la fonctionnalité "Gestion des exceptions"	58
Figure 34:Diagramme d'activité du fonctionnement de l'exécution des scripts.....	59
Figure 35: la relation entre les différentes interfaces	60
Figure 36: La relation entre les différentes interface du menu Paramétrage	61
Figure 37: La relation entre les différentes interfaces du menu Historique et Scripts	62
Figure 38 : Architecture d'application.....	64
Figure 39 : Logo de Spring framework	68
Figure 40 : Les modules de Spring.....	69
Figure 41 : Logo de Spring Boot	69
Figure 42 : Logo de Spring Data	70
Figure 43: Logo de javafx	70
Figure 44: Logo de hibernate	71
Figure 45: Logo de SQLite.....	71
Figure 46: Architecture client/serveur du SGBDR	72



Figure 47: Architecture SQLite sans serveur	72
Figure 48: Logo de maven.....	73
Figure 49: Logo d'eclipse IDE	74
Figure 50 : Logo de Visual Paradigm.....	74
Figure 51: Logo de GanttProject	74
Figure 52: Logo de SVN	75
Figure 53: Logo de SQLite Studio	75
Figure 54: Logo de SQL Developer	76
Figure 55 : Interface d'authentification.....	77
Figure 56 : Interface du choix de la base de données.....	78
Figure 57 : Interface d'erreur lors de l'exécution.....	79
Figure 58 : Interface d'historique	80
Figure 59: Interface d'authentification.	85
Figure 60: Interface principale.	86
Figure 61 : Interface de création d'un environnement.....	87
Figure 62 : Interface d'affichage des environnements.....	88
Figure 63 : Interface de modification d'un environnement.....	89
Figure 64 : Interface de suppression d'un environnement	89
Figure 65 : message de confirmation de la suppression d'un environnement.	89
Figure 66 : Interface de création d'un serveur.	90
Figure 67 : Interface d'affichage du serveur.....	91
Figure 68: interface de modification d'un serveur	91
Figure 69 : Message de confirmation du suppression d'un serveur.....	92
Figure 70 : Interface de création d'une base de données.	93
Figure 71 : Interface d'affichage des bases de données.....	93
Figure 72 : Interface de modification d'une base de données.....	94
Figure 73 : Interface de suppression d'une base de données.	94
Figure 74 : Message de confirmation de suppression d'une base de données.	95
Figure 75 : Interface principale d'exécution des scripts	95
Figure 76 : Interface du choix de la base de données.....	96
Figure 77 : Interface du choix du script.....	96
Figure 78 : Interface d'erreur lors de l'exécution.	97
Figure 79 : Interface pour changer une requête SQL.	98
Figure 80 : Interface d'exécution d'une nouvelle requête SQL.....	98
Figure 81 : Interface de gestion de l'exécution	99
Figure 82: Répertoire de kit de migration	100
Figure 83: Figure montre le renommage des scripts	101
Figure 84: Exécution de tool 2	102
Figure 85: menu de montée de version.....	103
Figure 86 : Exécution des scripts	104
Figure 87: Gestion des erreurs.....	105
Figure 88: Fin d'exécution	105



Liste des tableaux :

Tableau 1: Fiche d'identité de Sopra Steria group.....	24
Tableau 2: Tableau de référence des cas d'utilisation de « Gestion des environnements ».....	42
Tableau 3: Tableau de référence des cas d'utilisation de « Gestion des serveurs ».....	43
Tableau 4: Tableau de référence des cas d'utilisation de « Gestion des bases de données ».....	44
Tableau 5: Tableau de référence des cas d'utilisation de « Gestion d'exécution des scripts ».....	44
Tableau 6: Tableau de référence des cas d'utilisation de « Gestion des exceptions d'exécution»	45
Tableau 7: Tableau de référence des cas d'utilisation de « Gestion d'historique d'exécution».....	46
Tableau 8: Tableau de description du cas d'utilisation « Gestion d'exécution »	47
Tableau 9: Tableau de description du cas d'utilisation « mettre en pause l'exécution des scripts »	48
Tableau 10: Tableau de description du cas d'utilisation « reprendre l'exécution ».....	49
Tableau 11: Tableau de description du cas d'utilisation « modifier un bloc SQL ».....	49
Tableau 12: Tableau de description du cas d'utilisation « ignorer un bloc SQL »	50
Tableau 13: Tableau de description du cas d'utilisation « Ajouter un bloc SQL »	51
Tableau 14: Tableau de description du cas d'utilisation « créer un serveur »	51
Tableau 15: Tableau de description du cas d'utilisation « consulter un serveur».....	52
Tableau 16: Tableau de description du cas d'utilisation « modifier un serveur ».....	52
Tableau 17: Tableau de description des classes du projet.....	55
Tableau 18: Tableau de webographie.....	55

Liste des abréviations :

Sigle	Signification
UML	Unified Modeling Language
SSII	Société de Services en Ingénierie d’Informatique
SI	Système d’information
OMG	Object Management Group
ESN	Entreprises de Services du Numérique
J2EE	Java Entreprise Edition
MEA	Middle East Africa
XML	eXtensible Markup Language
R&D	Recherche & développement
EJB	Entreprise JavaBeans
SGBD	Système de Gestion de Base de Données
SQL	Sturctred Query Language
JDBC	Java Data Base Connectivity
ORM	Object Relational Mapping
MDV	Montée De Version
DAO	Data Access Object
IDE	Integrated Development Environment
VM	Virtual Machine
CRUD	Create, Read, Update and Delete
SVN	Subversion
STS	Spring Tool Suite
MVC	Model View Controller
IoC	Inversion of Control
POJO	Plain Old Java Object
HTML	HyperText Markup Language
API	Application Programming Interface
REST	Representational State Transfer
RIA	Rich Internet Application
BPMN	Business Process Model and Notation
SSH	Secure Shell
JSCH	Java Shell Channel
IP	Internet Protocol
TCP	Transmission Control Protocol
POM	Project Object Model
GPL	General Public Licence
JSON	JavaScript Object Notation
AOP	Aspect Oriented Programming



JPA	Java Persistence API
ACID	Atomicité, Cohérence, Isolation et Durabilité
PERT	Program Evaluation and Review Technique
DBA	Database Administrator
PL	Procedural Language
PDF	Portable Document Format
CSV	Comma Separated Values
IT	Information Technology

Introduction Générale :

À l'issue des cinq années d'études effectuées avec succès dans le domaine d'ingénierie informatique et afin de décrocher notre diplôme d'ingénieur d'état au sein de l'École Nationale des Sciences Appliquées d'Al-Hoceima, nous avons été amené à réaliser notre stage de fin d'études à Sopra Banking Software Morocco, une multinationale de services en ingénierie informatique. Leader mondial dans l'édition et l'intégration logicielle pour le marché financier.

En tant qu'éditeur et intégrateur de progiciels bancaires, Sopra Banking Software est reconnue par ses solutions modulaires et multi-canaux ainsi que par son expertise dans le domaine du Core Banking.

Parmi ces progiciels, Amplitude se présente comme la solution de référence pour un système Core Banking intégré.

Au fil des années ce progiciel a connu plusieurs mises à niveau, afin d'assurer la migration d'une ancienne version d'Amplitude à une version plus récente, l'équipe MDV est chargée de construire un kit de migration adaptable avec les besoins de chaque client.

L'objectif de ce projet est de mettre en œuvre une application desktop permettant la montée de version d'Amplitude.

A cet égard, quatre objectifs ont été envisagés :

- Maîtriser les spécifications fonctionnelles liées aux évènements de construction du kit de migration.
- Concevoir une solution globale et ouverte qui couvre les besoins d'aujourd'hui et prévoit ceux du futur.
- Réaliser et mettre en œuvre la solution MDV APP.
- Qualifier et tester la solution élaborée.

Le présent document est divisé en quatre chapitres :

Le premier chapitre définit le contexte général du projet. Il présente l'organisme d'accueil, ainsi que l'objectif du projet et la méthode d'ingénierie eMedia adoptée.



Le second chapitre se consacre à la description des démarches suivies durant la phase d'analyse et de conception d'une manière détaillée, ainsi que la modélisation UML des composants du projet, mais aussi au livrables fournis l'issue des modèles proposés par la méthode Agile eMedia de la société :

Le troisième chapitre décrit les différentes architectures, technologies et outils utilisés pour la réalisation du projet, la solution informatique développée ainsi que les tests et qualifications de la solution.

A la fin de ce manuscrit, nous présentons le bilan du stage, la conclusion du travail ainsi que les perspectives du projet.

Chapitre 1 : Cadre général du projet

Introduction :

Ce chapitre cerne le contexte général du projet. D'abord, une vue globale sera élaborée sur l'histoire de l'organisme d'accueil, Sopra Banking Software, et son organisation. Par la suite, une description du cadre général du projet par la précision des objectifs majeurs de l'application, la méthodologie suivie au cours du projet, ainsi que le GANTT des tâches effectuées.

1. Présentation de l'organisme d'accueil :

Le système d'information (SI) est devenu pour les entreprises et les administrations un levier de performance et de conquête, le garant de leur capacité à anticiper les transformations indispensables à leur développement et leur pérennité, dans un environnement qui évolue de manière accélérée. La première partie de ce chapitre présente quelques informations sur le groupe Sopra Steria et décrit ses secteurs d'activités ainsi que ses différents pôles.

1.1 Sopra group :



Figure 1: Logo Sopra

Sopra Group, créé en janvier 1968 par Pierre Pasquier, François Odin et Léo Gantelet, figure parmi les plus anciennes Entreprises de Services du Numérique (ESN) en Europe. Ce groupe est un acteur majeur du conseil, des services technologiques et de l'édition de solutions en Europe, il accompagne ses clients dans la réussite de la transformation de leurs métiers et systèmes d'information.

Sopra Group figure dans le Top 10 des sociétés européennes de Conseil et de Services Informatiques. Elle se propose particulièrement une approche globale, du Conseil à l'Intégration de Systèmes jusqu'à l'Outsourcing applicatif et les Solutions applicatives.

1.2 Steria group :



Figure 2: Logo Steria

Steria, créée en 1969, est une ESN (nouvelle appellation des SSII) française, la cinquième sur le marché français en 2011 et la sixième à l'échelle européenne, avec un chiffre d'affaires de 1,75 Md d'euros en 2011 et 20 000 salarié(e)s réparti(e)s dans 16 pays. La société est cotée sur Euronext Paris. Le 8 avril 2014, Steria et Sopra ont annoncé une opération de fusion par

une OPE lancée par Sopra sur son concurrent Steria. Son siège social se situe à Vélizy-Villacoublay, dans le pôle technologique Paris-Saclay.

1.3 Sopra Steria group :



Figure 3: Logo Sopra Steria

Sopra Steria Group fruit de la fusion de Sopra group créé en 1968 et Steria qui a été créé en 1969, le projet de rapprochement des deux entreprises a donné naissance à Sopra Steria, leader européen de la transformation numérique, propose l'un des portefeuilles d'offres les plus complets du marché :

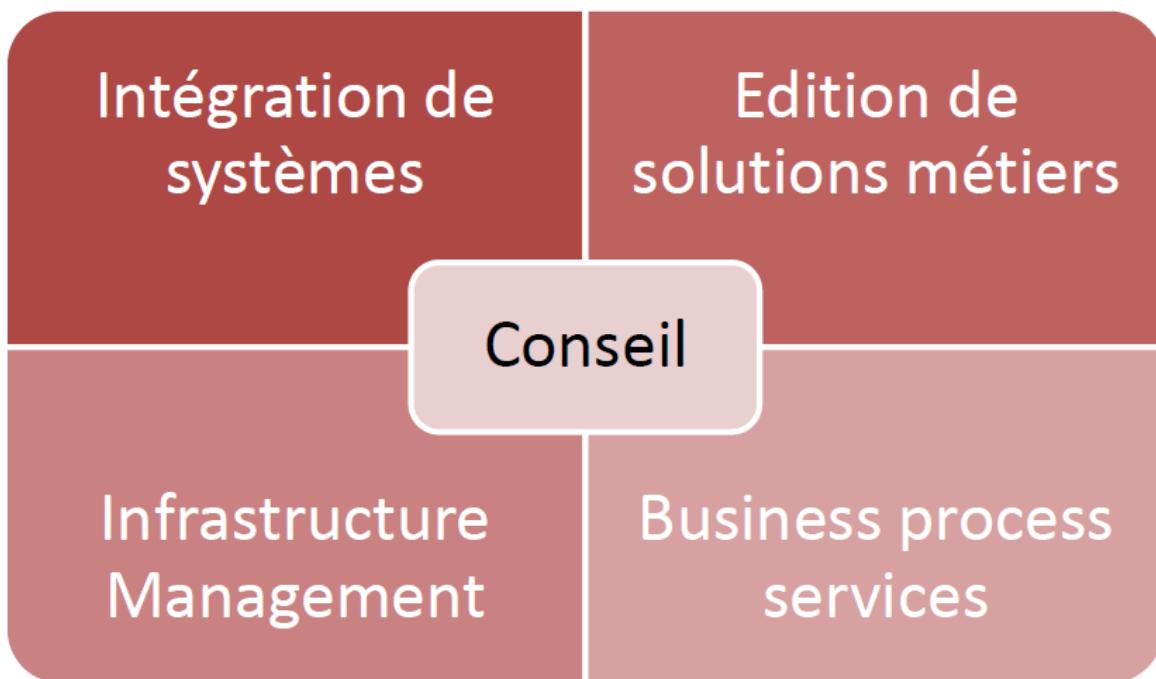


Figure 4: Portefeuille d'activités couvert par Sopra

Il apporte ainsi une réponse globale aux enjeux de développement et de compétitivité des grandes entreprises et organisations.

Sopra Steria accompagne ses clients dans leur transformation et les aide à faire le meilleur usage du numérique.

1.3.1 Fiche technique :

Le tableau ci-dessous présente l'organisme d'accueil dans lequel a été effectué notre stage de fin d'études :

Date de rapprochement	Septembre 2014
Fondateurs	Pierre Pasquier, François Odin, Léo Gantelet et Jean Carteron.
Chiffre d'affaires	3.7 milliards d'euros
Activité	Société de conseil, de services informatiques et d'édition de logiciels.
Effectif	37000
Forme juridique	SA à conseil d'administration
Implantations	+ 20 pays
Site Web	http://www.SopraSteria.com

Tableau 1: Fiche d'identité de Sopra Steria group

1.3.2 Métiers et secteurs d'activités :

L'activité de Sopra Steria Group est centrée sur divers secteurs d'activité parmi lesquels figurent les services financiers, les services transports & utilities, le service public, l'industrie, les télécoms et médias, la distribution ainsi que la défense.

La figure ci-dessous illustre la répartition du chiffre d'affaire de Sopra Group par marché.

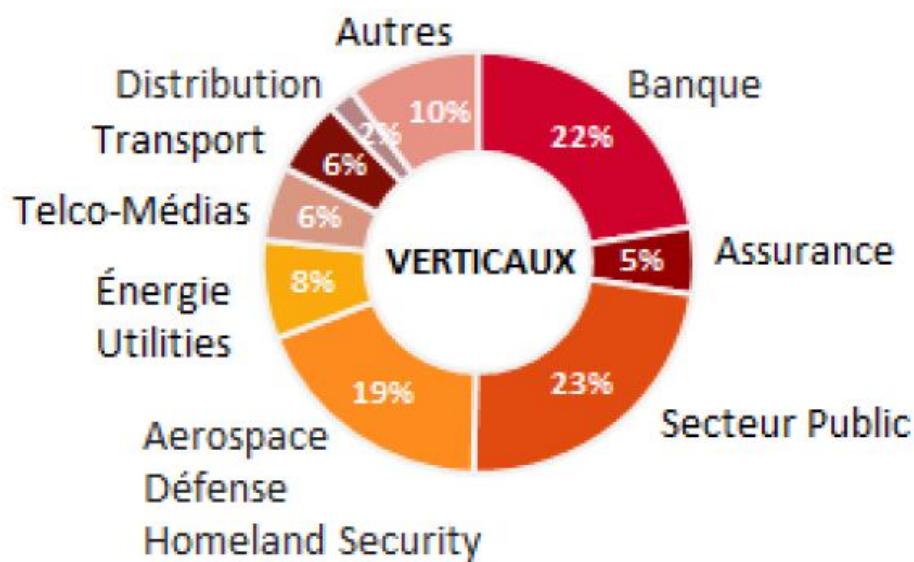


Figure 5: Secteur d'activité

1.3.3 Filiales :

Sopra Steria constitue aujourd’hui trois filiales importantes :



Figure 6: Logo Sopra Steria Consulting



Figure 7: Logo Sopra HR Software



Figure 8: Sopra Banking Software

❖ Sopra Banking Software :



Figure 9: Logo du Sopra Banking Software

Est un éditeur et intégrateur de logiciel bancaire et un acteur majeur du digital au service des banques depuis 40 ans. Avec une connaissance approfondie des métiers de la banque.

Début 2012, Sopra Group annonçait son projet de créer l’un des principaux acteurs européens dans l’édition de logiciels bancaires, avec l’ambition de devenir mondial.

Depuis le 1er juillet 2012, la filiale Sopra Banking Software est en place et a pour vocation d’offrir à l’ensemble des acteurs du secteur Banque & Finance des solutions progicielles à haute valeur ajoutée, capables de répondre à leurs enjeux.

Sopra Banking Software propose une offre très étendue et reconnue, constituée du rapprochement des solutions issues de Sopra Group, Callataÿ & Wouters et Delta Informatique. Sopra Banking Software sert d’ores et déjà plus de 600 clients dans plus de 70 pays avec un chiffre d’affaire de 217 millions d’euro en 2013. Avec l’appui du Groupe Sopra et grâce à son patrimoine de solutions, sa capacité d’intégration et d’accompagnement près de 2000 experts, Sopra Banking Software s’est mis en situation de répondre à tous les établissements.



Figure 10: Chiffres clés de Sopra Banking Software

- **Sopra Banking Software MEA (Moyen orient et Afrique) :**

Sopra Banking Software MEA est la filiale Moyen orient et Afrique de Sopra Banking software, dont la société mère est Sopra-Steria Group. L'Afrique fait partie intégrante des stratégies de Sopra Banking Software. Ce marché est en croissance et se transforme. Tandis que sur des marchés plus matures, on est sur des problématiques de rationalisation – on veut passer d'un système propriétaire à des progiciels en Afrique où le développement est rapide, Sopra Banking Software accompagne la création de banques (ou réseaux de banques) ainsi que le développement de nouveaux marchés. Ses clients doivent gérer le dynamisme du marché et donc la concurrence des nouveaux entrants (opérateurs télécom, établissements de micro finance...). Par ailleurs, il est également nécessaire pour ses clients d'accompagner la croissance du continent africain en se développant à l'international via des réseaux de banques interafricaines. Par exemple, ses clients marocains veulent investir sur l'Afrique subsaharienne en ouvrant des filiales. Elle leur accompagne alors les différentes directions (générale, métiers, IT) du siège dans leur stratégie de développement.

- **Sopra Banking Software Morocco :**

Sopra Banking Software Morocco est une partie intégrante de Sopra Banking Software MEA, et désormais un hub incontournable pour toute l'Afrique grâce à sa position géographique et la stratégie qu'adopte aujourd'hui le Maroc avec l'Afrique. Son siège social est implanté à Casablanca, créée en Septembre 2007 et emploie à présent plus de 300 collaborateurs Sopra Banking Software Morocco s'affirme comme une société de droit marocain. Ses principales missions sont les suivantes :

- Développer une compétence pérenne au Maroc pour ses marchés.
- Accompagner les projets de Solutions Bancaires Envolant dans les établissements de crédit, sociétés financières et associations de microcrédit au Maroc.
- Assurer des développements pour la R&D Banque Sopra Steria Group.

Le centre de service de Casablanca :

Au cœur de cette organisation, Sopra Banking Software Morocco héberge dans son agence de Casablanca, un centre de service dédié à Amplitude de plus de 100 ingénieurs, réparti sur 5 pôles d'activité, à savoir les pôles :

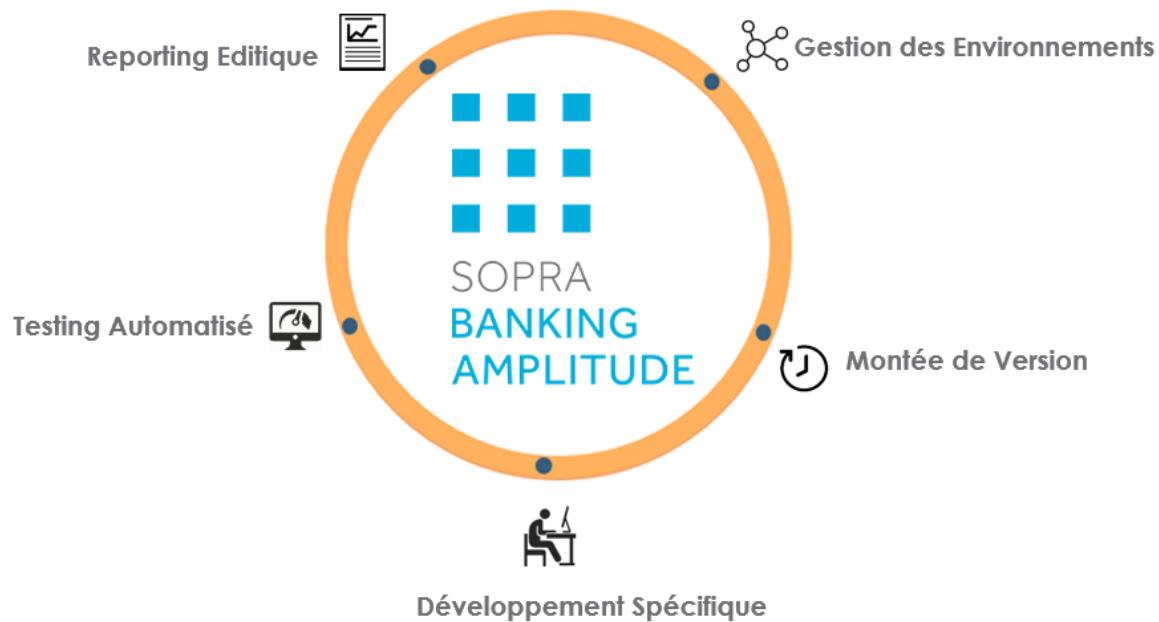


Figure 11: Organisation du Centre de Services Amplitude

Chacun de ces pôles a pour mission de servir et d'accompagner les différents projets menés par le groupe dans la zone et principalement en Afrique. L'activité principale du centre de service de Casablanca se concentre autour du Core Banking **Amplitude**. Ce stage se déroulera au sein du Pôle « **Montée de Version** », appelé également pôle « **MDV** ».

Organigramme du Centre de service :

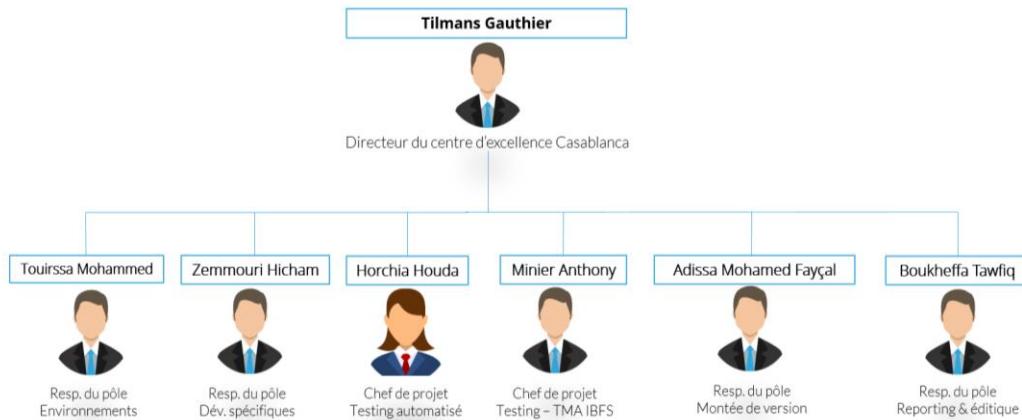


Figure 12: Organigramme du centre de service

Organigramme de l'équipe :

La réussite d'un projet passe impérativement par une organisation rigoureuse et efficace de l'équipe du projet, c'est ainsi le cas pour le pôle MDV.

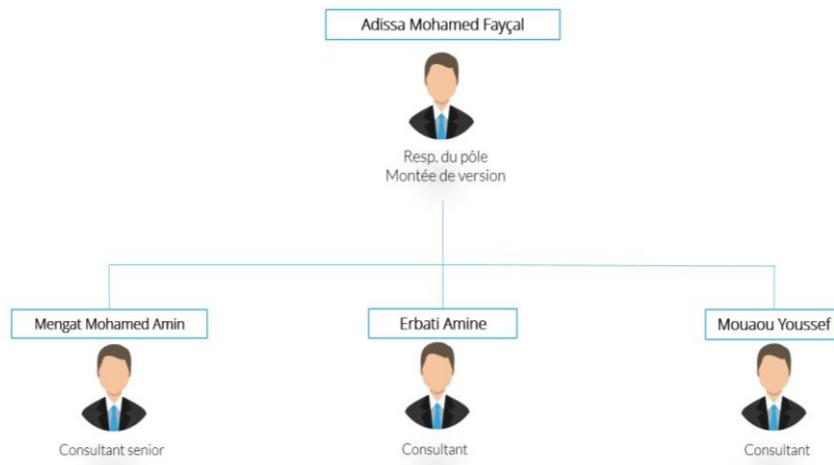


Figure 13: Organigramme de l'équipe

Présentation du pôle MDV :

Le pôle MDV est chargé d'assurer la montée de version d'Amplitude selon le besoin du client et le changement que subi son système.

Il a pour mission de construire un Kit de migration qui permet de passer d'une version source à une version cible d'Amplitude compatible avec la base de données du client.

La construction de ce Kit s'étale sur les étapes suivantes :



Figure 14: les principales étapes de la construction de kit de migration

Cependant, la construction du kit de migration n'est qu'une phase préliminaire de la montée de version. La mission principale du pôle MDV est de mener à bien le déroulement de ce kit chez le client afin d'avoir une version d'Amplitude adaptable avec sa base de données.

La mission du pôle MDV ne se restreint pas seulement sur les fonctions citées précédemment, il supporte à distance le bon fonctionnement du progiciel après la migration.

2. Présentation générale du progiciel Amplitude :

2.1 Evolution du SI bancaire :

Dans les années 80, le système d'informations des banques repose sur 2 axes :

- Une comptabilité auxiliaire (les comptes des clients) ;
- Une comptabilité générale.

Quelques systèmes spécifiques incluant des schémas comptables « en dur » dans les années 90-2000, de nouveaux concepts sont apparus tels que :

- Les opérations traitées par des applicatifs dédiés avec une logique de traitement des événements bancaires ;
- L'urbanisation des systèmes d'informations ;
- Les systèmes de synthèse (dont la comptabilité) alimentés via un interpréteur de règles.

Depuis 2005, de nouveaux changements ont modifié :

- Logique de séparation des applicatifs selon une logique distribution /production ;
- Arrivée de la distribution multi-canal ;
- Architectures orientées services.

2.2 Enjeux du système d'informations :

Les systèmes d'information sont au cœur de l'activité bancaire et financière. L'émergence de nouveaux concepts et de nouveaux outils a révolutionné la nature des services fournis par les systèmes d'information automatisés en augmentant notamment leur dimension stratégique et leur capacité à intégrer les nouvelles réglementations des professions financières.

La progression forte du PNB des banques africaines entraîne la nécessité d'une mutation importante de leurs systèmes d'information, pour faire face à l'augmentation des volumes et aux problématiques de rapidité de traitement des informations.

- S'adapter aux évolutions constantes du métier bancaire :



-
- Nouvelles techniques bancaires
 - Nouvelles méthodes commerciales
 - S'adapter aux évolutions réglementaires
 - S'adapter aux évolutions de l'activité de la banque :
 - Rapprochements/fusions
 - Evolutions d'organisation
 - Nouveaux canaux
 - Sans rupture d'activité

2.3 Urbanisation des SI :

Un système d'information urbanisé est un système où :

- Chaque application sert un métier et est optimisée pour ce métier (rythme, volume...) ;
- Chaque application saisit et maintient ses données propres ;
- La communication inter-applications est assurée par un bus applicatif garant de l'intégrité des échanges et de l'indépendance des applications ;
- La Distribution (FrontOffice) et le Back-Office sont indépendants.

2.4 Présentation d'Amplitude :

Adoptée avec succès par les banques de tailles petites et moyennes (moins de 4000 agences), la solution Sopra Banking Amplitude comprend des modules offrant un système d'information complet et facile à mettre en place pour répondre aux besoins les plus complexes d'une banque.

Sopra Banking Amplitude est une solution complète, modulaire et intégrée qui offre de riches fonctionnalités et permet une implémentation dans des délais maîtrisés.

Ce type d'architecture évolutif garantit l'intégrité des données et permet une implémentation rapide et progressive. Sa flexibilité et son ouverture permettent de répondre aux attentes des établissements financiers qui se trouvent aujourd'hui au marché, en termes d'interopérabilité recherchée.

2.4.1 Positionnement d'Amplitude :

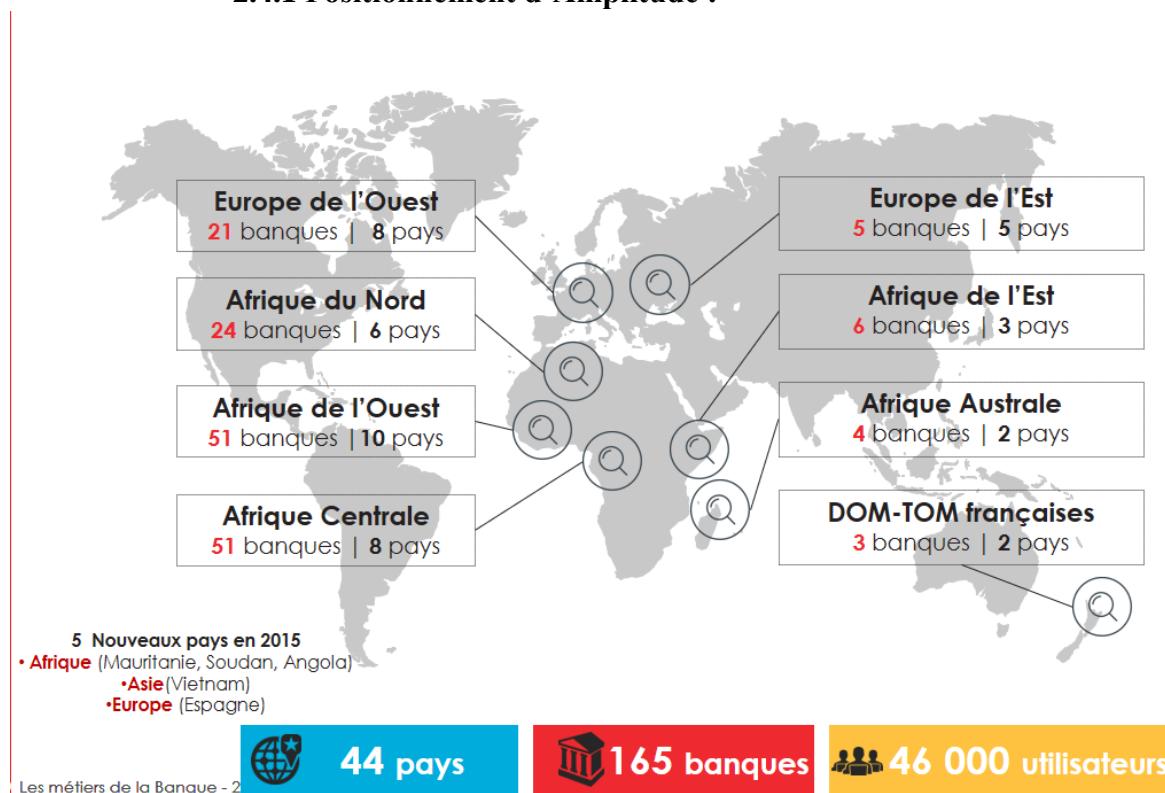


Figure 15: Positionnement Amplitude en zone MEA

2.4.2 Clients Sopra Banking Amplitude :

Parmi les groupes bancaires utilisant Sopra Banking Amplitude, on trouve :



Figure 16: Les clients du Core Banking Amplitude

3. Présentation du projet :

3.1 Etude de l'existant et problématique :

Cette figure résume les étapes du processus de la Montée de version d'Amplitude, la description détaillée se trouve dans l'annexe B.

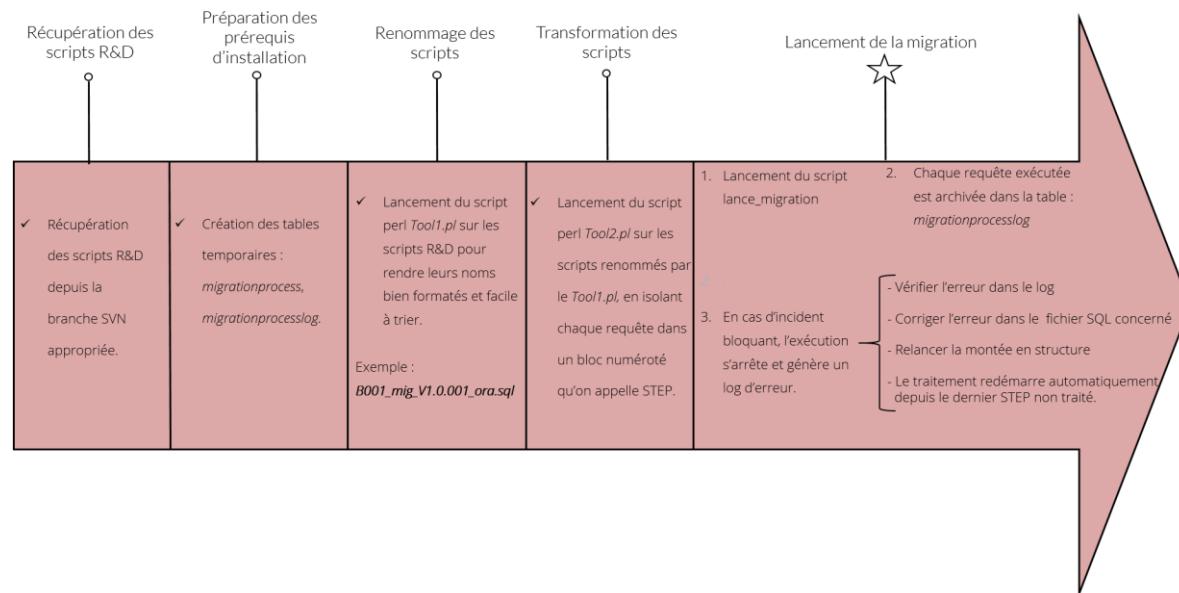


Figure 17: Le processus de construction du Kit de migration

Sopra Banking Software affronte une mission sérieuse consiste à migrer tous ses clients à la dernière version d'Amplitude, c'est un travail qui demande tant de gestion et d'économie.

Cela pousse l'équipe MDV à chercher à automatiser ce travail, particulièrement que le processus de la montée de version se déroule en mode console, donc c'est manuel et très gênant.

En outre, les outils utilisés dans cette montée de version sont inaptes de fournir une bonne qualité ; prenons l'exemple de l'outil 'Tool 2' qui est un script perl qui consomme assez de ressources pour accomplir sa tâche.

3.2 Solution :

Le problème décrit auparavant nous ramène à penser à concevoir une solution fiable afin de pouvoir optimiser le processus de la recette avant la livraison, de gagner en temps, en qualité et en ressource.

L'objectif ultime de ce projet est donc la mise en place d'un système qui va automatiser la montée de version.

Après l'étude de l'existant on remarque que les outils étudiés ont plus ou moins les mêmes fonctionnalités à savoir les fonctionnalités principales :

- La connexion à la base de données ;
- L'exécution des scripts SQL ;
- La gestion des exceptions SQL générées ;
- Le lancement des programmes de reprise.

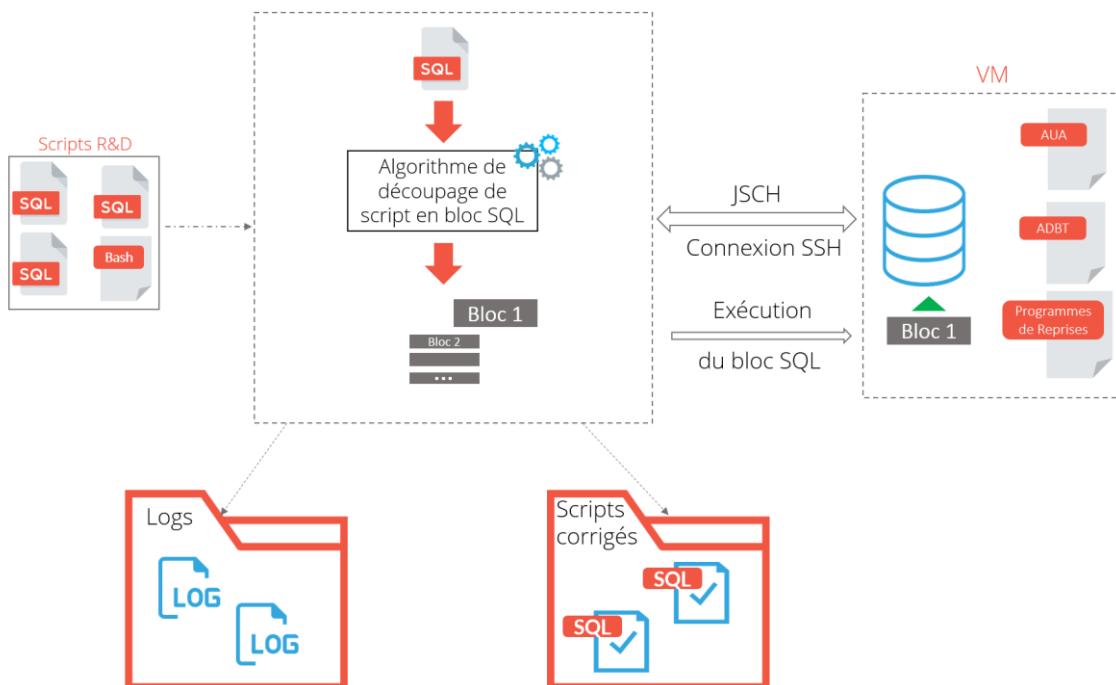


Figure 18: Schéma de la solution proposée

3.3 Conduite et planification du projet :

3.3.1 La méthode d'ingénierie eMedia :

eMedia est la méthode d'ingénierie des systèmes d'information de Sopra Steria Group. Elle définit l'ensemble des processus de production et de management nécessaires à la conduite des projets, de la conception au déploiement.

3.3.1.1 Présentation de la méthodologie d'ingénierie eMedia :



Figure 19:logo eMedia

La vocation première d'une méthode d'ingénierie logicielle est d'apporter une aide effective aux projets pour le développement et la maintenance des systèmes. Pour ce faire, cette méthode regroupe un ensemble de pratiques, permettant de conduire les différentes activités nécessaires, en une démarche et un processus d'ingénierie cohérents.

Fondamentalement, une méthode d'ingénierie vise à apporter des réponses à un certain nombre de questions qui se posent lors de tout développement ou évolution d'un système.

- Pourquoi on fait les choses (les objectifs) ;
- Sur quoi on travaille (les produits) ;
- Ce qu'il faut faire (les activités) ;
- Qui fait les choses (les rôles) ;
- Comment peut-on faire les choses (les pratiques et les techniques) ;
- Quand doit-on faire les choses (le processus d'ingénierie).

eMedia est une méthode d'ingénierie itérative et incrémentale qui a pour vocation de privilégier les pratiques standards et éprouvées de la communauté internationale, pratiques qu'elle adapte pour prendre en compte notre contexte particulier, notre savoir-faire et notre culture propre, en intégrant l'ensemble dans une démarche d'ingénierie cohérente.

L'objectif d'eMedia n'est pas de définir une méthode d'ingénierie « idéale » dans l'absolu avec des préoccupations purement académiques, mais bien de procurer une méthode d'ingénierie qui apporte des éléments de réponse concrets aux problèmes de notre projet. Cette vision résolument pragmatique ne s'oppose pas, bien au contraire, à la volonté de reposer sur des fondements méthodologiques solides : la pratique sans théorie est aussi inefficace que la théorie sans pratique, les deux doivent s'enrichir mutuellement

❖ **Principes d'EMedia :**

eMedia identifie un nombre limité – sept – de principes fondamentaux qui contribuent grandement à la réussite d'un projet. Chacun de ces principes apporte une valeur en lui-même, mais c'est bien leur mise en œuvre combinée et complémentaire qui procure le plus d'efficacité à un projet :

- L'ingénierie est d'abord une activité d'ingénieur ;
- Travailler en équipe et jouer collectif ;
- Ne faire que ce qui est nécessaire ... mais faire tout ce qui est nécessaire ;
- Affronter les risques o Travailler en professionnel ;
- Privilégier la production de valeur ;
- Optimiser la productivité du projet.

❖ **eMedia itérative et incrémentale :**

eMedia repose sur un processus itératif et incrémental maîtrisé qui marque un changement profond avec les processus purement séquentiels.

Le principe des itérations est de découper la période d'activité du projet en périodes plus petites maitrisables. Les itérations sont organisées pour traiter en priorité les aspects les plus risqués (fonctionnels, techniques et organisationnels). Une Itération constitue un mini projet avec des objectifs précis préétablis et une période de temps déterminée, produisant des résultats tangibles. On peut également, tirer profit du mode de production itératif pour valider les exigences implémentées et pour affiner les exigences attendues.

❖ Un pilotage par les risques :

La méthode eMedia, composée d'un processus de production itératif, décrit un avantage probant en comparaison à une méthode purement séquentielle. Cet avantage s'aperçoit, lors de chaque itération produit, dans la production de versions intermédiaires de la solution, ce qui permet de discerner et de jauger objectivement l'état d'avancement du projet.

Par ailleurs, cette méthode propose une démarche efficace de résolution des risques en traitant en premier lieu les aspects fonctionnels, techniques et organisationnels les plus risqués du projet lors de chaque itération.

Cette méthode consiste à suivre un processus de 4 phases intitulé par : **Le processus de production** « *Delivery Process* ».

3.3.1.2 Le processus de production : Delivery Process :

Si les itérations permettent d'organiser et de maîtriser le travail à court terme, elles ne sont pas suffisantes. Elles doivent s'inscrire dans un plan à plus long terme qui prend en compte les objectifs et les jalons globaux du projet pour définir une stratégie d'ensemble.

C'est le rôle du Delivery Process.

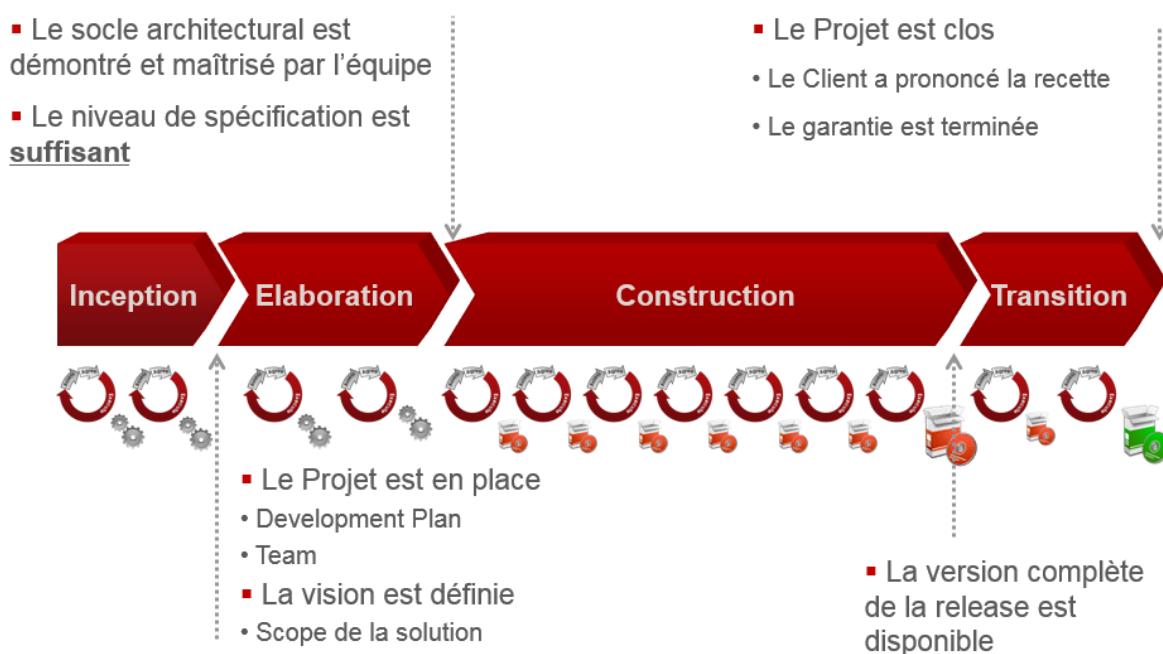


Figure 20: Processus de production eMedia

Comme le montre la figure ci-dessus (fig.20), eMedia repose sur deux stades fondamentaux :

- ***Un stade de définition*** : qui correspond à la définition d'une solution fonctionnelle et technique répondant aux besoins des parties prenantes du projet et à la démonstration de la faisabilité de cette solution. Il est constitué de deux phases séquentielles :
- ***Inception*** : On définit dans cette phase le périmètre global du projet ;
- ***Élaboration*** : À la fin de cette phase, le socle architectural est démontré et maîtrisé par l'équipe, et le niveau de spécification est suffisant pour alimenter une première itération de la construction.
- ***Un stade de production*** : correspond à la production de la solution précédemment définie et à sa livraison. Il est constitué de deux phases séquentielles :
- ***Construction*** : la version complète du projet est disponible à la fin de cette phase ;
- ***Transition*** : le projet est clos durant cette phase, et la recette client est prononcée.

3.3.2 Application :

L'application de la méthode eMedia au projet se traduit tout d'abord par une phase d'Inception où le projet est mis en place et le périmètre est défini. Suivi de la phase d'élaboration avec l'analyse et la rédaction des spécifications états et modèle. Puis la phase de réalisation avec la réalisation des états et des paramétrages et développement nécessaires pour produire les fichiers de déclaration. Finalement, la phase de transition pour intégrer le modèle chez le client.

Les phases exposées dans ce rapport sont la phase d'élaboration (chapitre II) et la phase de réalisation (chapitre III).

3.3.3 Diagramme de GANTT :

La gestion de projet consiste à planifier, à organiser et à gérer des tâches et des ressources dans le but d'atteindre l'objectif fixé.

Pour ce faire, l'élaboration d'un diagramme de GANTT est particulièrement adaptée, ce type d'outil permet à tout moment d'avoir une vision globale du projet et aide à faire le suivi de l'avancement du projet.



Le planning du projet a relativement été respecté sauf pour certains points qui ont nécessité plus de réflexion pour les traiter, car une évaluation exacte du temps consacré aux tâches requiert une parfaite maîtrise des outils et environnement de développement et de l'expérience.

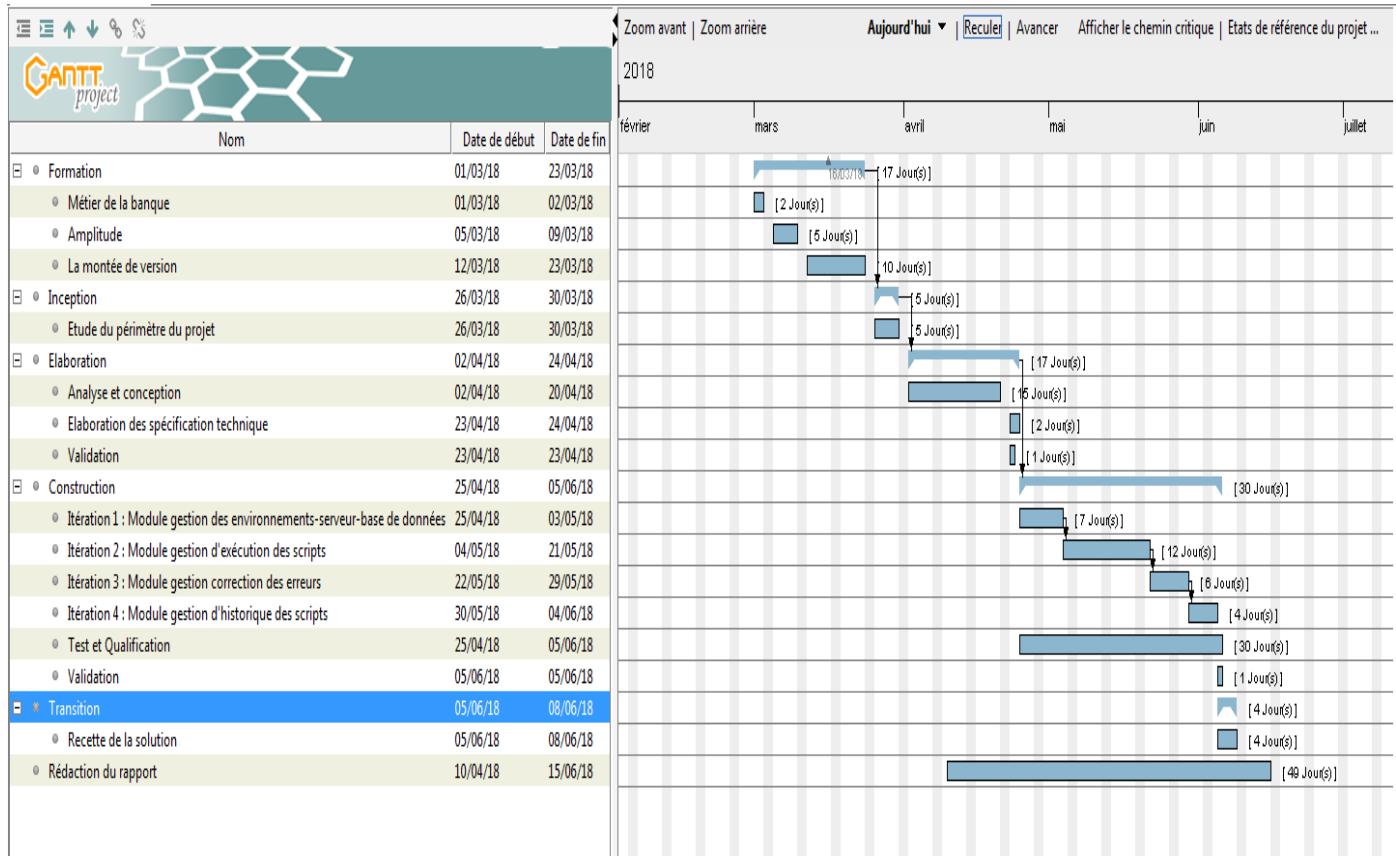


Figure 21: Diagramme de Gantt

4. Conclusion :

Ce chapitre a mis en place le point de départ de l'élaboration du projet, dans la mesure où il décrit son contexte général, en présentant successivement l'organisme d'accueil Sopra Banking Software Morocco, son produit phare Amplitude, les problèmes principaux de l'existant afin d'en sortir les objectifs primordiaux du projet, ainsi que sa démarche de développement ; eMedia, et les étapes de sa mise en œuvre.

Le chapitre suivant sera consacré à l'analyse et la conception des spécifications des besoins.

Chapitre 2 : Analyse et conception

Introduction :

L'étude fonctionnelle formalise et détaille ce qui a été ébauché au cours de l'étude préliminaire. Elle produit un modèle des besoins focalisé sur le métier des utilisateurs. Elle qualifie au plus tôt le risque de produire un système inadapté aux utilisateurs.

Cette étude va déterminer les interactions entre les acteurs et le système dans les diagrammes des cas d'utilisation, et ensuite, décrire l'essentiel du travail effectué durant la conception du projet, en modélisant les besoins identifiés précédemment sous forme de quelques diagrammes de modélisation UML.

1. Les diagrammes de Cas d'utilisation :

Après avoir identifié les acteurs qui interagissent avec le système, nous y développons un premier modèle UML, pour pouvoir établir précisément les frontières du système et construire un diagramme reliant les acteurs et les cas d'utilisation.

Ensuite, des descriptions textuelles de cas d'utilisation sont rédigées afin de détailler les différentes façons dont les acteurs peuvent utiliser le système.

1.1 Diagramme de cas d'utilisation global :

Pour améliorer le modèle, une organisation des cas d'utilisation sous forme d'ensembles fonctionnels cohérents est recommandée. Cette étape a conduit donc à un découpage du projet sous forme de modules fonctionnels, dont la cartographie est présentée sur la figure ci-dessous en utilisant la notion des packages

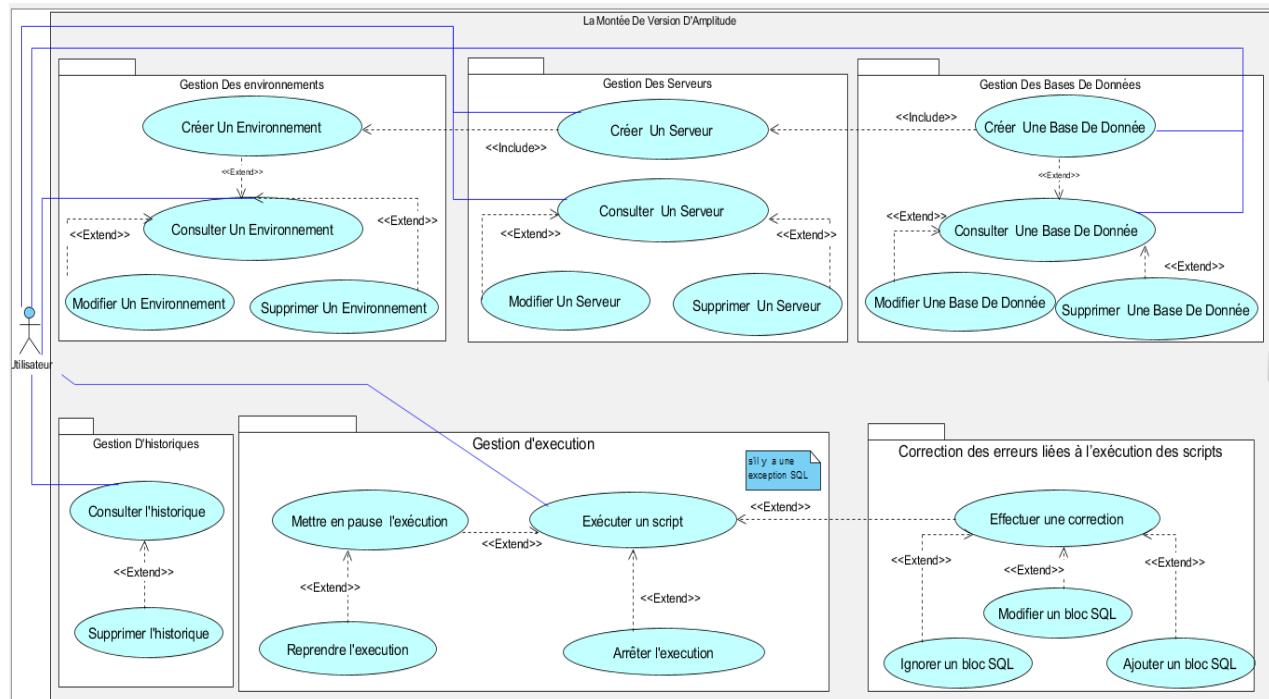


Figure 22: Diagramme global de la gestion de montée en version d'Amplitude

En ce qui suit, on va présenter les cas d'utilisation de chaque package en les attribuant une référence.

1.2 Classement des cas d'utilisation

Cette figure montre les différentes fonctionnalités du module « Gestion des environnements »:

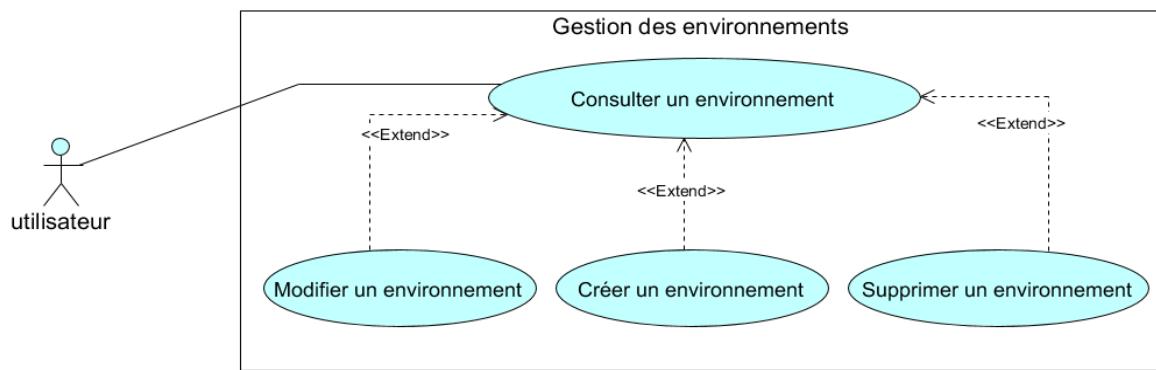


Figure 23 : Diagramme de « Gestion des environnements »

N° UC	Use cases	Type
UC_01	Consulter un environnement	UC
UC_02	Créer un environnement	UC
UC_03	Modifier un environnement	UC
UC_04	Supprimer un environnement	UC

Tableau 2: tableau de référence des cas d'utilisation de « Gestion des environnements »

La figure ci-dessous montre les différentes fonctionnalités du module « Gestion des serveurs » :

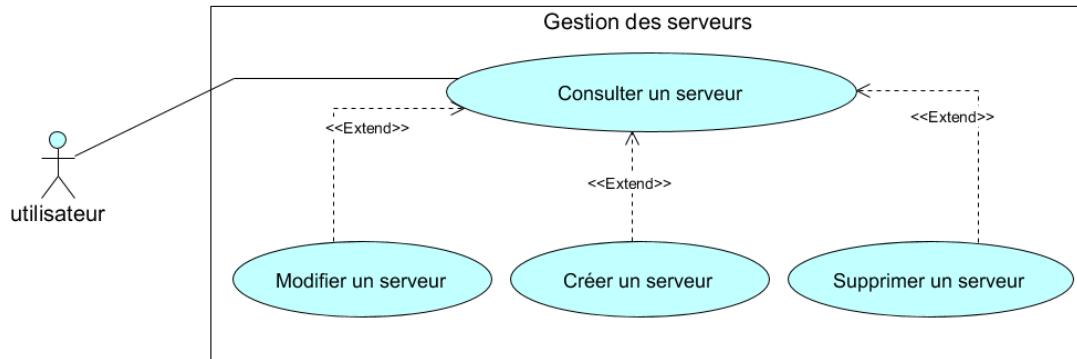


Figure 24 : Diagramme de « Gestion des serveurs »

N° UC	Use cases	Type
UC_05	Consulter un serveur	UC
UC_06	Créer un serveur	UC
UC_07	Modifier un serveur	UC
UC_08	Supprimer un serveur	UC

Tableau 3: tableau de référence des cas d'utilisation de « Gestion des serveurs »

Dans la figure ci-dessous on présente les différentes fonctionnalités du module « Gestion des bases de données » :

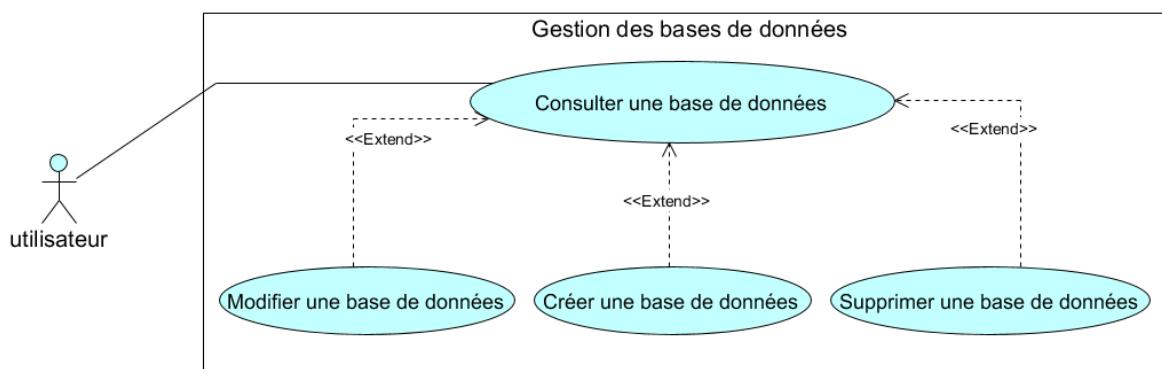


Figure 25 : Diagramme de « Gestion des bases de données »

N° UC	Use cases	Type
UC_09	Consulter une base de données	UC
UC_10	Créer une base de données	UC
UC_11	Modifier une base de données	UC
UC_12	Supprimer une base de données	UC

Tableau 4: tableau de référence des cas d'utilisation de « Gestion des bases de données »

Cette figure montre les différentes fonctionnalités du module « Gestion d'exécution des scripts » :

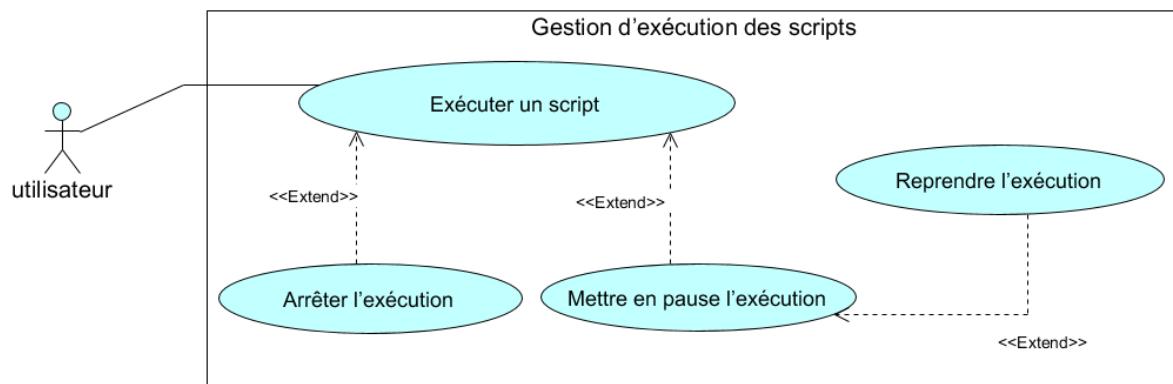


Figure 26: Diagramme de « Gestion d'exécution des scripts »

N° UC	Use cases	Type
UC_13	Exécuter un script	UC
UC_14	Mettre en pause l'exécution	UC
UC_15	Reprendre l'exécution	UC
UC_16	Arrêter l'exécution	UC

Tableau 5: tableau de référence des cas d'utilisation de « Gestion d'exécution des scripts »

Les différentes fonctionnalités du module « Correction des erreurs liées à l'exécution des scripts » sont clairement représentées dans la figure 27 :

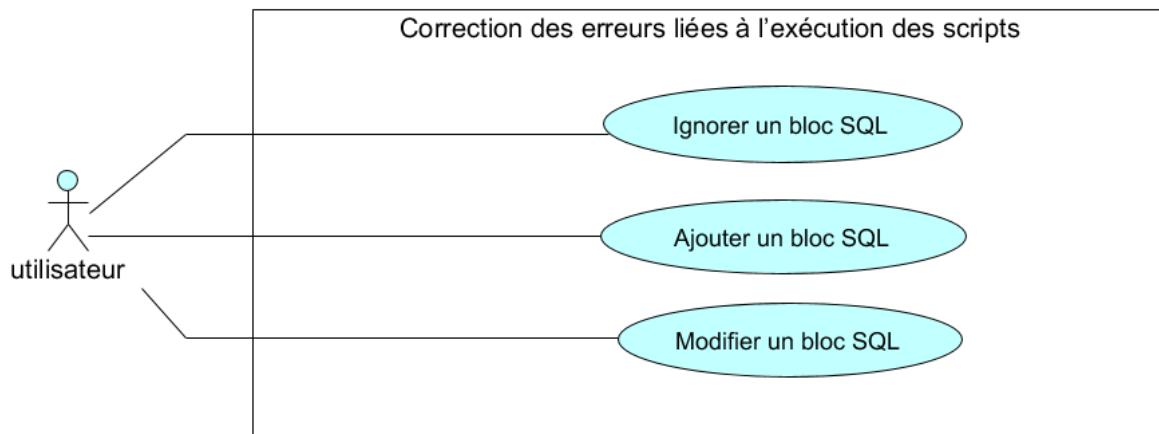


Figure 27: Diagramme de « Correction des erreurs liées à l'exécution des scripts »

N° UC	Use cases	Type
UC_17	Modifier un bloc SQL	UC
UC_18	Ajouter un bloc SQL	UC
UC_19	Ignorer un bloc SQL	UC

Tableau 6: tableau de référence des cas d'utilisation de « Gestion des exceptions d'exécution»

Finalement, la figure 28 montre les différentes fonctionnalités du module « Gestion d'historique de scripts » :

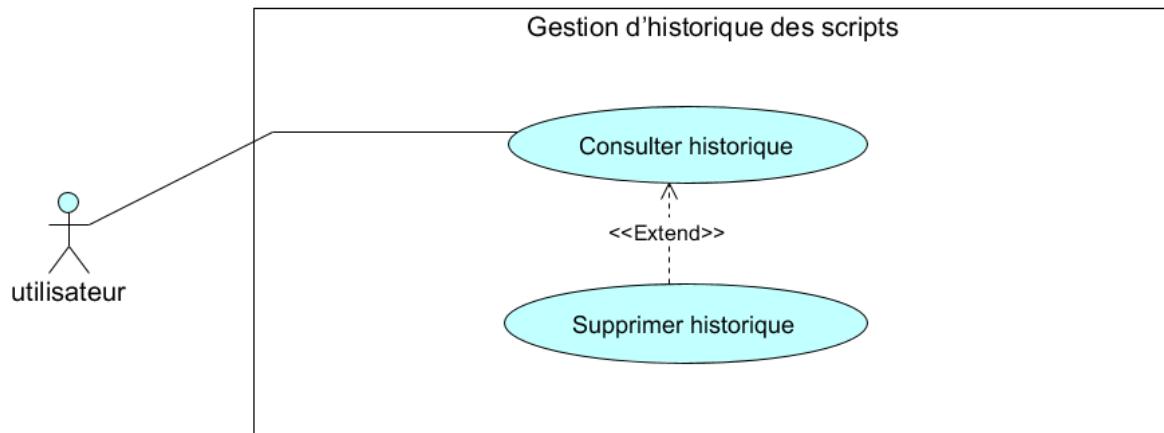


Figure 28Diagramme de « Gestion d'historique de scripts »

N° UC	Use cases	Type
UC_21	Consulter historique	UC
UC_22	Supprimer historique	UC

Tableau 7: tableau de référence des cas d'utilisation de « Gestion d'historique d'exécution»

1.3 Description des cas d'utilisation :

Cette section va décrire les cas d'utilisation en termes d'objectifs, d'acteurs, de pré-conditions et de post-conditions. Les descriptions illustrent le scénario de chaque cas d'utilisation dans des étapes aboutissant à la réalisation de la fonctionnalité ciblée, tout en identifiant les premières exceptions qui peuvent intervenir.

1.3.1 Description de « Gestion d'exécution » :

❖ Exécution d'un script :

Types	Cas d'utilisation
Intitulé	Exécuter un script.
Description	Ce cas d'utilisation décrit le processus de l'exécution d'un script SQL.
Acteur principal	Utilisateur.
Préconditions	<p>-l'utilisateur doit choisir l'environnement de travail (serveur et base de données).</p> <p>- l'utilisateur doit importer les scripts à exécuter.</p> <p>-Connexion réussie à la base de données.</p>
Post-conditions	Exécution réussie ou échouée.
<i>Description des étapes</i>	
<ul style="list-style-type: none"> ▪ Contrôle de sélection de l'environnement de travail. ▪ Contrôle de sélection de script à exécuter. ▪ Contrôle de connexion à la base de données. ▪ Répartir le script en des instructions unitaires (requête simple, procédure, fonction...). ▪ Exécuter les instructions une par une. ▪ Sauvegarder les instructions bien exécutées dans un fichier. 	
<i>Exception</i>	
<p>Le système peut rencontrer les exceptions ci-dessous :</p> <ul style="list-style-type: none"> ▪ L'utilisateur n'a pas sélectionné l'environnement de travail. ▪ Aucun script n'a été sélectionné. ▪ La connexion à la base de données a été échouée. ▪ Problème dans l'exécution de l'instruction. 	

Tableau 8: tableau de description du cas d'utilisation « Gestion d'exécution »

❖ Mettre en pause l'exécution de script :

Types	Cas d'utilisation
Intitulé	Mettre en pause l'exécution de script.
Description	Ce cas d'utilisation permet d'arrêter temporairement l'exécution du script.
Acteur principal	Utilisateur.
Pré-conditions	<ul style="list-style-type: none"> - Le script doit être en exécution. - Le système doit vérifier que l'instruction en cours d'exécution a été terminée.
Post-conditions	
<i>Description des étapes</i>	
<ul style="list-style-type: none"> ▪ Le cas où la dernière instruction avant l'arrêt de l'exécution a été exécutée avec succès : <ul style="list-style-type: none"> - On sauvegarde la dernière ligne de cette instruction ; - Déconnexion de la base de données. ▪ Le cas où l'exécution de l'instruction a été échouée : <ul style="list-style-type: none"> - On sauvegarde la dernière instruction exécutée avec succès 	
<i>Exception</i>	

Tableau 9:tableau de description du cas d'utilisation « mettre en pause l'exécution des scripts »

❖ Reprendre l'exécution :

Types	Cas d'utilisation
Intitulé	Reprendre l'exécution.
Description	Ce cas d'utilisation permet de reprendre l'exécution à partir du point d'arrêt.
Acteur principal	Utilisateur.
Pré-conditions	<ul style="list-style-type: none"> - Le script doit être en pause.
Post-conditions	Le script continue l'exécution du script.
<i>Description des étapes</i>	
<ul style="list-style-type: none"> ▪ Connexion à la base de données. ▪ Continue l'exécution du script à partir du point d'arrêt. 	
<i>Exception</i>	

- Un problème au niveau de la connexion à la base de données.

Tableau 10: tableau de description du cas d'utilisation « reprendre l'exécution »

1.3.2 Correction des erreurs liés à l'exécution des scripts

❖ Modifier un bloc SQL :

Types	Cas d'utilisation
Intitulé	Modifier un bloc SQL.
Description	Ce cas d'utilisation permet de modifier un bloc SQL afin de corriger une exception liée à ce bloc.
Acteur principal	Utilisateur.
Pré-conditions	<ul style="list-style-type: none"> - Le bloc SQL est affiché dans la console.
Post-conditions	<ul style="list-style-type: none"> -Si l'exécution de l'instruction modifiée a été bien passée, on l'ajoute dans le fichier corrigé. -Sinon une exception s'affiche.
<i>Description des étapes</i>	
<ul style="list-style-type: none"> ▪ Ajouter les modifications nécessaires au bloc SQL. ▪ Connexion à la base de données ▪ Exécuter le bloc. ▪ Continue l'exécution du script à partir du point d'arrêt. 	
<i>Exception</i>	
<ul style="list-style-type: none"> ▪ Problème dans l'exécution de l'instruction. ▪ Problème au niveau de la connexion 	

Tableau 11: tableau de description du cas d'utilisation « modifier un bloc SQL »

❖ Ignorer un bloc SQL :

Types	Cas d'utilisation
Intitulé	Ignorer un bloc SQL
Description	Ce cas d'utilisation est utile en cas d'exception, il permet d'ignorer l'exécution du bloc SQL.
Acteur principal	Utilisateur.
Pré-conditions	-L'instruction est affichée dans la console.
Post-conditions	- Ajouter l'instruction en commentaire dans le fichier principal.
<i>Description des étapes</i>	
<ul style="list-style-type: none"> ▪ Ajouter l'instruction en commentaire dans le fichier principal. ▪ Connexion à la base de données. ▪ Continue l'exécution du script à partir du point d'arrêt. 	
<i>Exception</i>	
<ul style="list-style-type: none"> ▪ Un problème au niveau de la connexion à la base de données. 	

Tableau 12: tableau de description du cas d'utilisation « ignorer un bloc SQL »

❖ Ajouter un bloc SQL :

Types	Cas d'utilisation
Intitulé	Ajouter un bloc SQL.
Description	Ce cas d'utilisation permet d'ajouter un nouveau bloc SQL au script.
Acteur principal	Utilisateur.
Pré-conditions	
Post-conditions	Exécution réussie ou échouée.
<i>Description des étapes</i>	
<ul style="list-style-type: none"> ▪ En cliquant sur « Nouveau Bloc SQL » une fenêtre s'affiche pour ajouter le bloc SQL ; ▪ Connexion à la base de données ; ▪ L'exécution de ce bloc peut générer deux cas : <ul style="list-style-type: none"> -exécution réussie, avec l'ajout de ce bloc dans le fichier corrigé ; -exécution échouée, avec la possibilité de corriger le bloc SQL. 	



<i>Exception</i>
<ul style="list-style-type: none"> ▪ Un problème au niveau de la connexion à la base de données. ▪ Une exception au niveau de l'exécution du bloc SQL.

Tableau 13: tableau de description du cas d'utilisation « Ajouter un bloc SQL »

1.3.3 Gestion des serveurs

❖ Crée un serveur :

Types	Cas d'utilisation
Intitulé	Créer un serveur
Description	Ce cas d'utilisation permet la création d'un serveur dans un environnement.
Acteur principal	Utilisateur.
Pré-conditions	
Post-conditions	Création de serveur réussie ou rejetée.
<i>Description des étapes</i>	
1. Contrôle de présence des données obligatoires :	
2. Contrôle de la validité des données en entrée.	
<i>Exception</i>	

Tableau 14: tableau de description du cas d'utilisation « créer un serveur »

❖ **Consulter un serveur :**

Types	Cas d'utilisation
Intitulé	Consulter un serveur
Description	Ce cas d'utilisation permet de consulter les informations d'un serveur
Acteur principal	Utilisateur.
Pré-conditions	
Post-conditions	Informations sur le serveur
<i>Description des étapes</i>	
3. Informations de serveur récupéré. 4. Affichage de résultat.	
<i>Exception</i>	

Tableau 15: tableau de description du cas d'utilisation « consulter un serveur »

❖ **Modifier un serveur :**

Types	Cas d'utilisation
Intitulé	Modifier un serveur
Description	Ce cas d'utilisation décrit les étapes suivies pour une modification des informations d'un serveur.
Acteur principal	Utilisateur.
Pré-conditions	
Post-conditions	Modification réussie ou rejetée.
<i>Description des étapes</i>	
5. Contrôle de présence des données obligatoires : 6. Contrôle de la validité des données en entrée.	

Tableau 16: tableau de description du cas d'utilisation « modifier un serveur »



2. Diagramme de classe :

Le diagramme de classes est le point central dans un développement orienté objet. En analyse, il a pour objectif de décrire la structure des entités manipulées par les utilisateurs. En conception, le diagramme de classes représente la structure d'un code orienté objet ou, à un niveau de détail plus important, les modules du langage de développement.

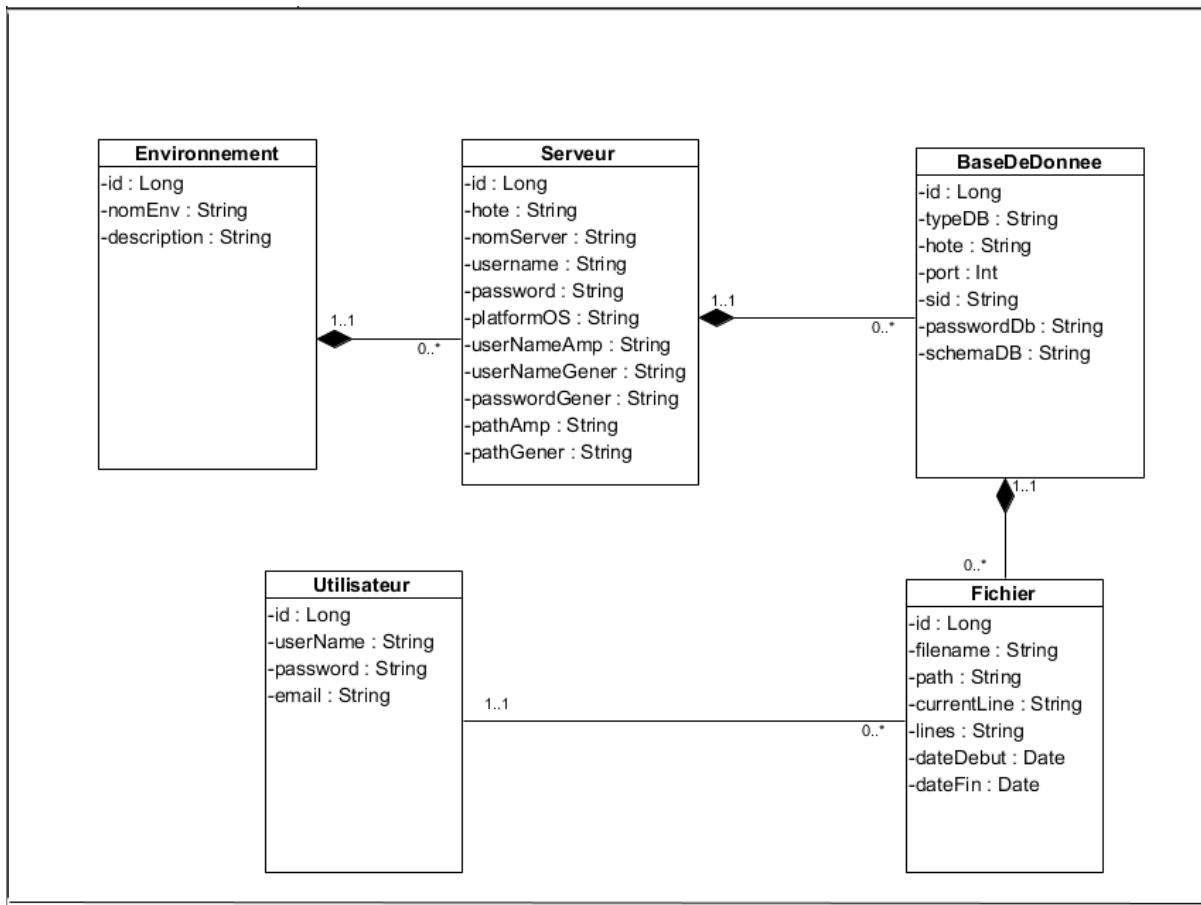


Figure 29: Diagramme de classe

Vu que le nombre de classes du système est relativement grand et que cela pourrait nuire à la qualité de l'image présentée, on va se limiter dans ce rapport sur la description de l'entité « Base De Données », en structurant les différentes classes interagissant par packages.

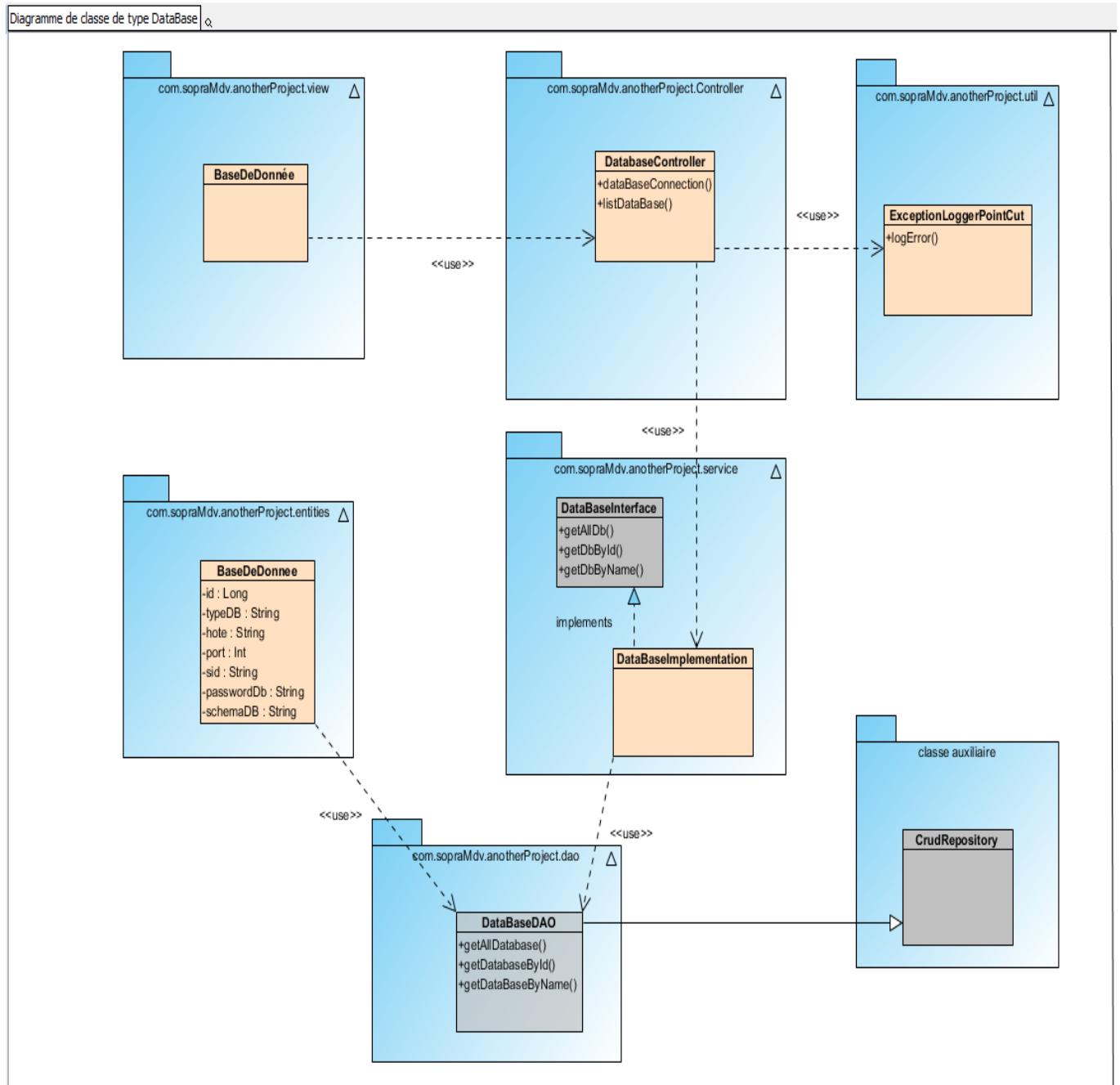


Figure 30: Diagramme des classes du système

Ce tableau montre la relation entre les packages de notre solution :

Package	Classe	Description
com.sopraMdv.anotherProject.entities	DataBase	Une classe POJO contenant les attributs de l'entité 'DataBase', les accesseurs et les constructeurs.
com.sopraMdv.anotherProject.dao	DataBaseDAO	Cette interface étend de CrudRepository, elle contient les CRUD.
com.sopraMdv.anotherProject.service	DataBaseInterface	C'est une interface contenant la définition des services métiers.
	DataBaseImplementation	C'est une implémentation de l'interface 'DataBaseInterface', elle contient les différents services métiers.
com.sopraMdv.anotherProject.controller	DataBaseController	Cette classe contient les contrôleurs qui seront appelé au moment du déclenchement d'un évènement au niveau du fichier FXML.
com.sopraMdv.anotherProject.view	BaseDeDonnee	C'est un fichier FXML servant à définir des interfaces graphiques.
com.sopraMdv.anotherProject.util	ExceptionLoggerPointCut	Cette classe permet d'attraper les exceptions SQL et les traiter selon nos désirs.
Classe auxiliaire	CrudRepository	C'est une interface générique des CRUD fournie par Spring Data.

Tableau 17: tableau de description des classes du projet

3. Diagramme de séquence :

Les diagrammes de séquences représentent une vue dynamique de système qui met en évidence les interactions entre les objets et leurs messages. Ci-après les principaux diagrammes de séquence qui décrit le déroulement du cas d'utilisation précédemment cité.

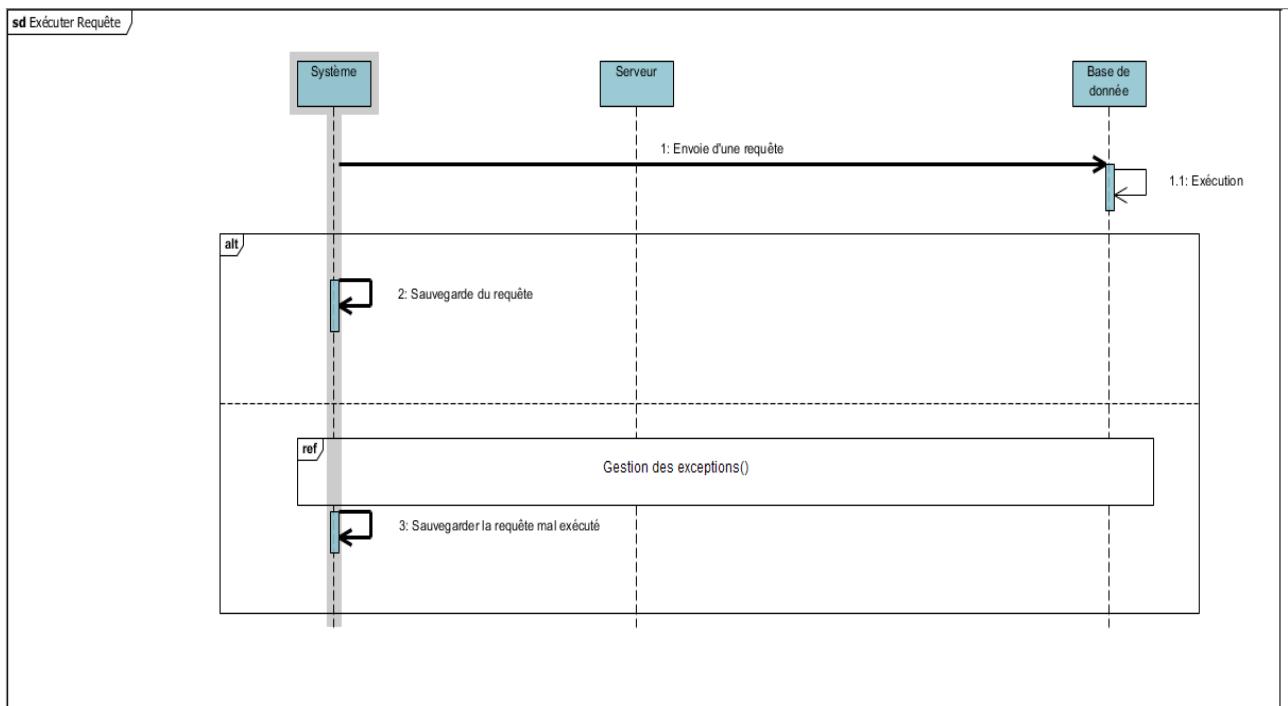


Figure 31: Diagramme de séquence de la fonctionnalité "Exécuter requête"

Cette figure montre l'enchaînement du processus de la fonctionnalité « Exécution script » :

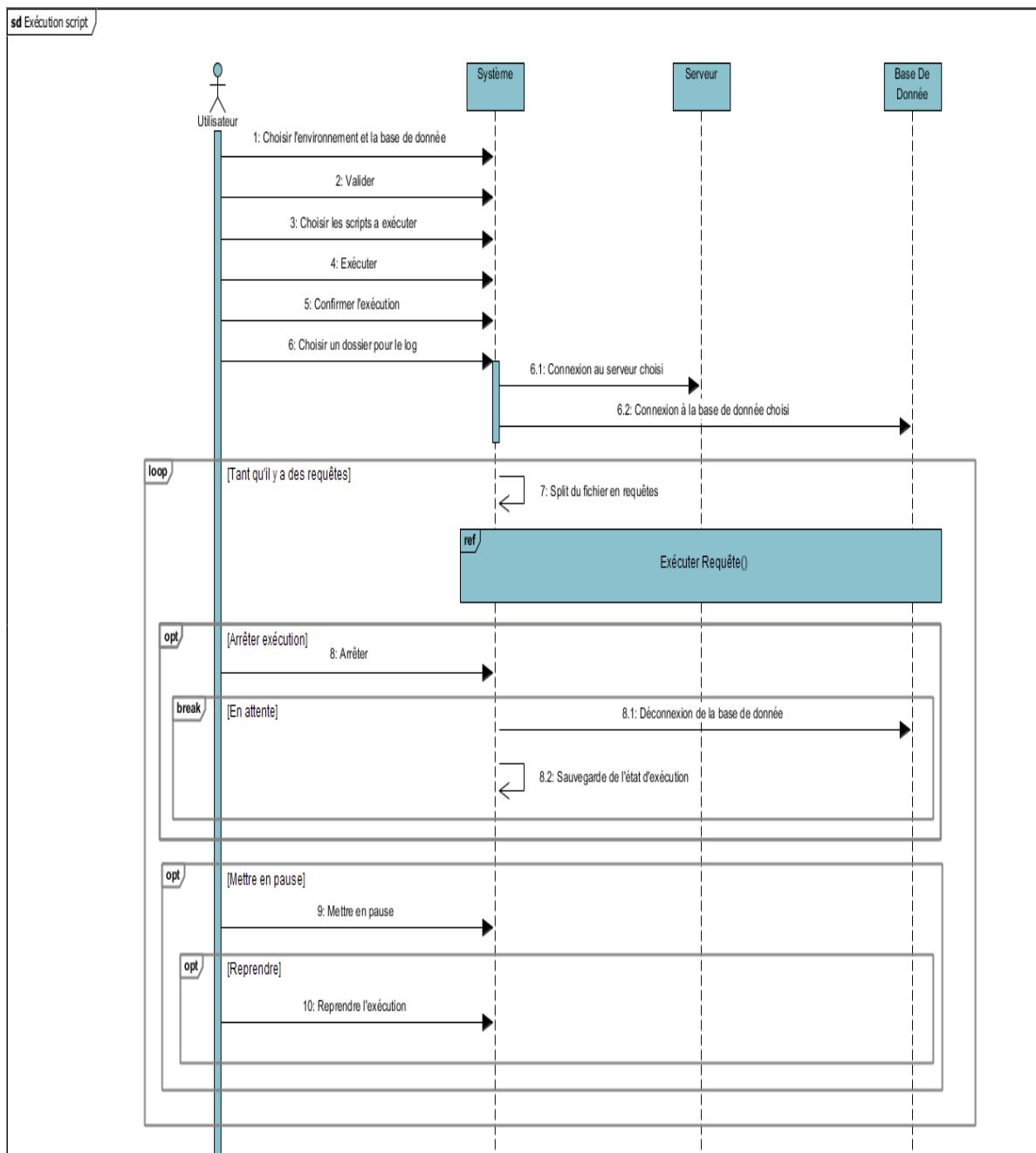


Figure 32: Diagramme de séquence de la fonctionnalité "Exécution script"

Cette figure montre l'enchaînement du processus de la fonctionnalité « Gestion des exceptions » :

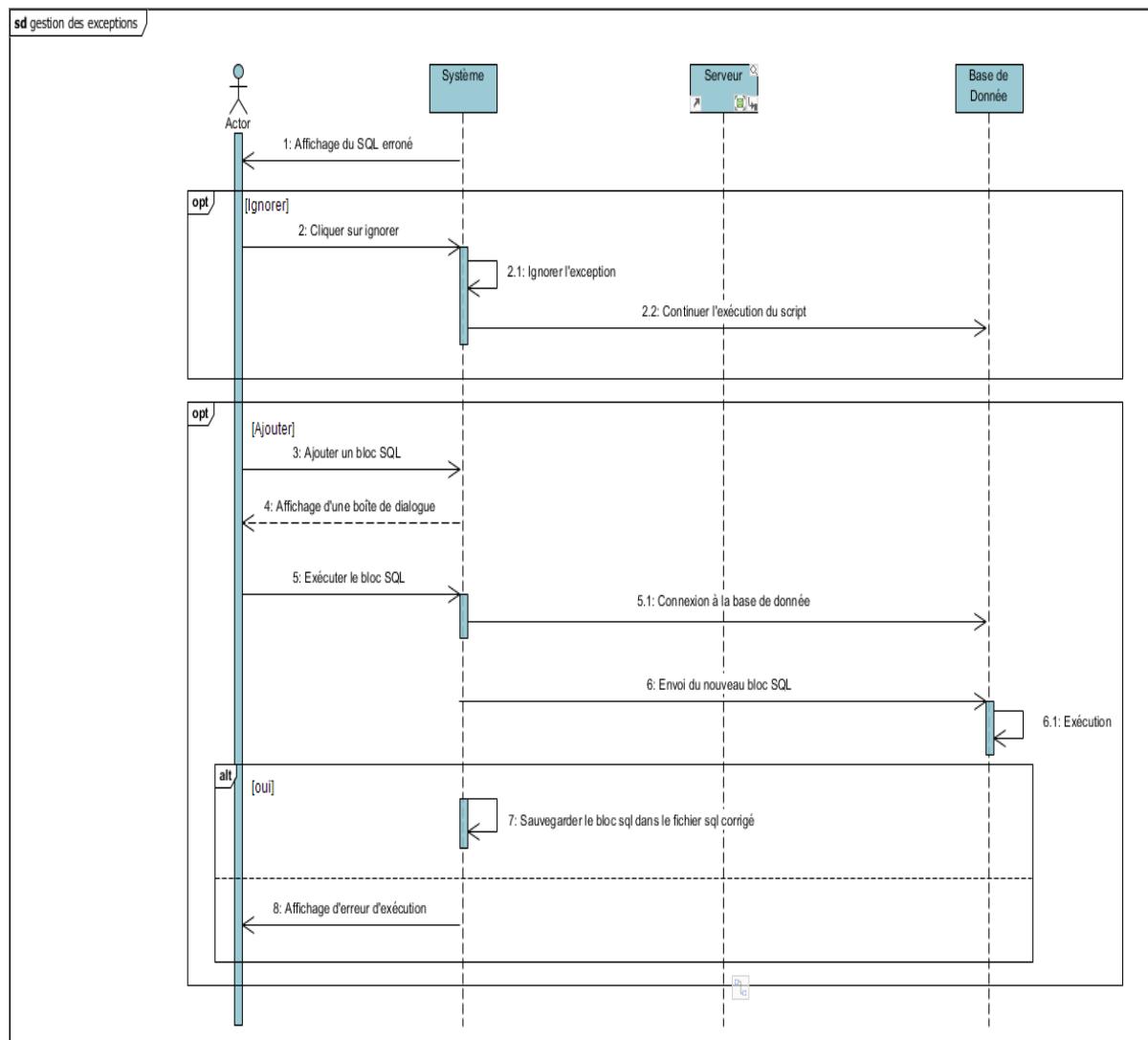


Figure 33: Diagramme de séquence de la fonctionnalité "Gestion des exceptions"

4. Diagramme d'activité :

Dans cette partie, on présente un diagramme d'activité élaboré après une étude approfondie du périmètre du projet, des spécifications fonctionnelles et définition des cas d'utilisation.

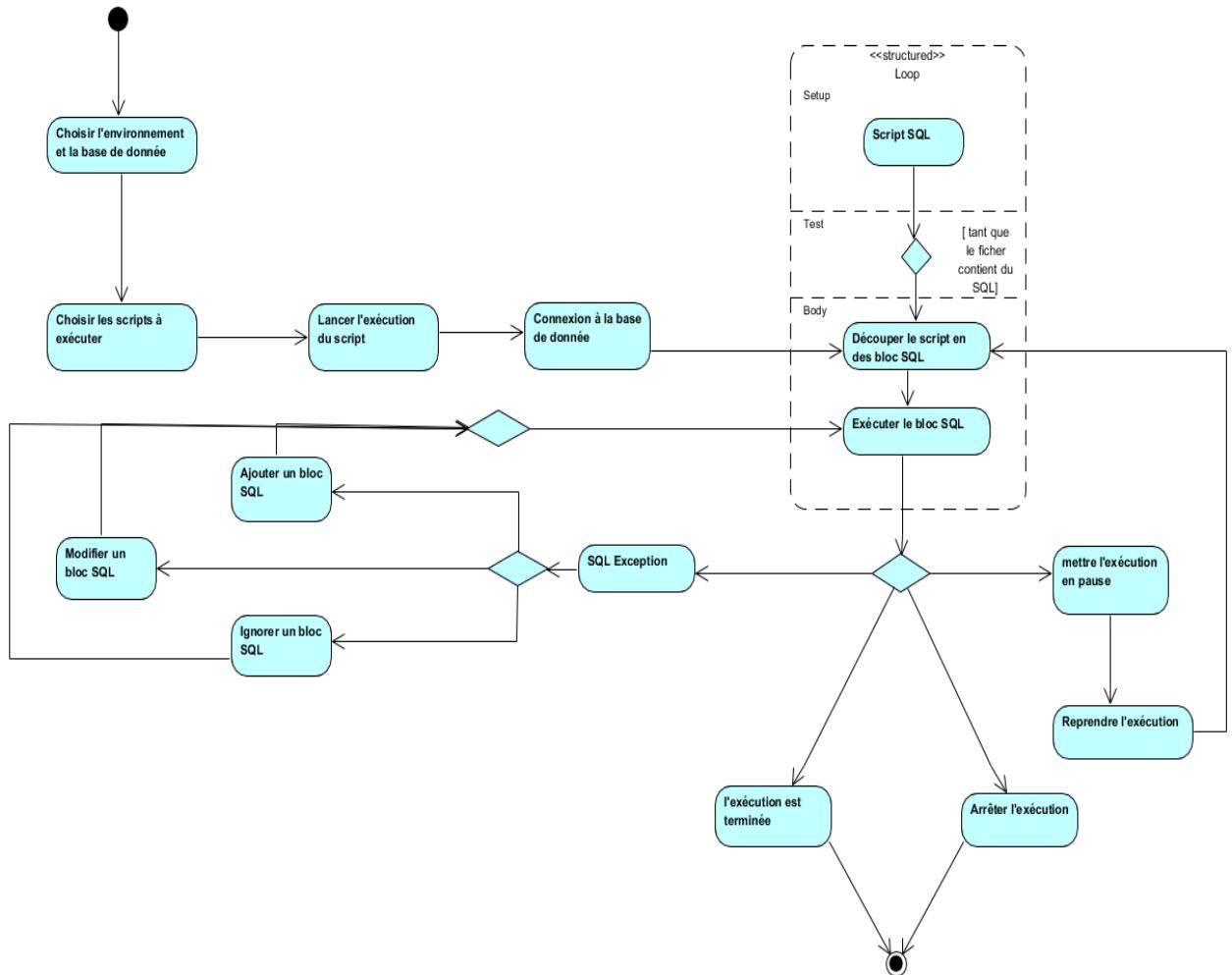


Figure 34: Diagramme d'activité du fonctionnement de l'exécution des scripts

5. Cinématique des écrans :

Dans cette partie, une illustration des cinématiques des écrans permet de comprendre la navigabilité entre les différents écrans de notre application.

Notre application contient trois volets principaux :

- Historique : Permet de consulter l'historique des scripts exécutés avec la possibilité de continuer l'exécution des scripts ;
- Scripts : Sert à gérer l'exécution des scripts ;
- Paramétrage : Menu déroulant contient trois sous menu ; Environnement, Base de données et Serveur.

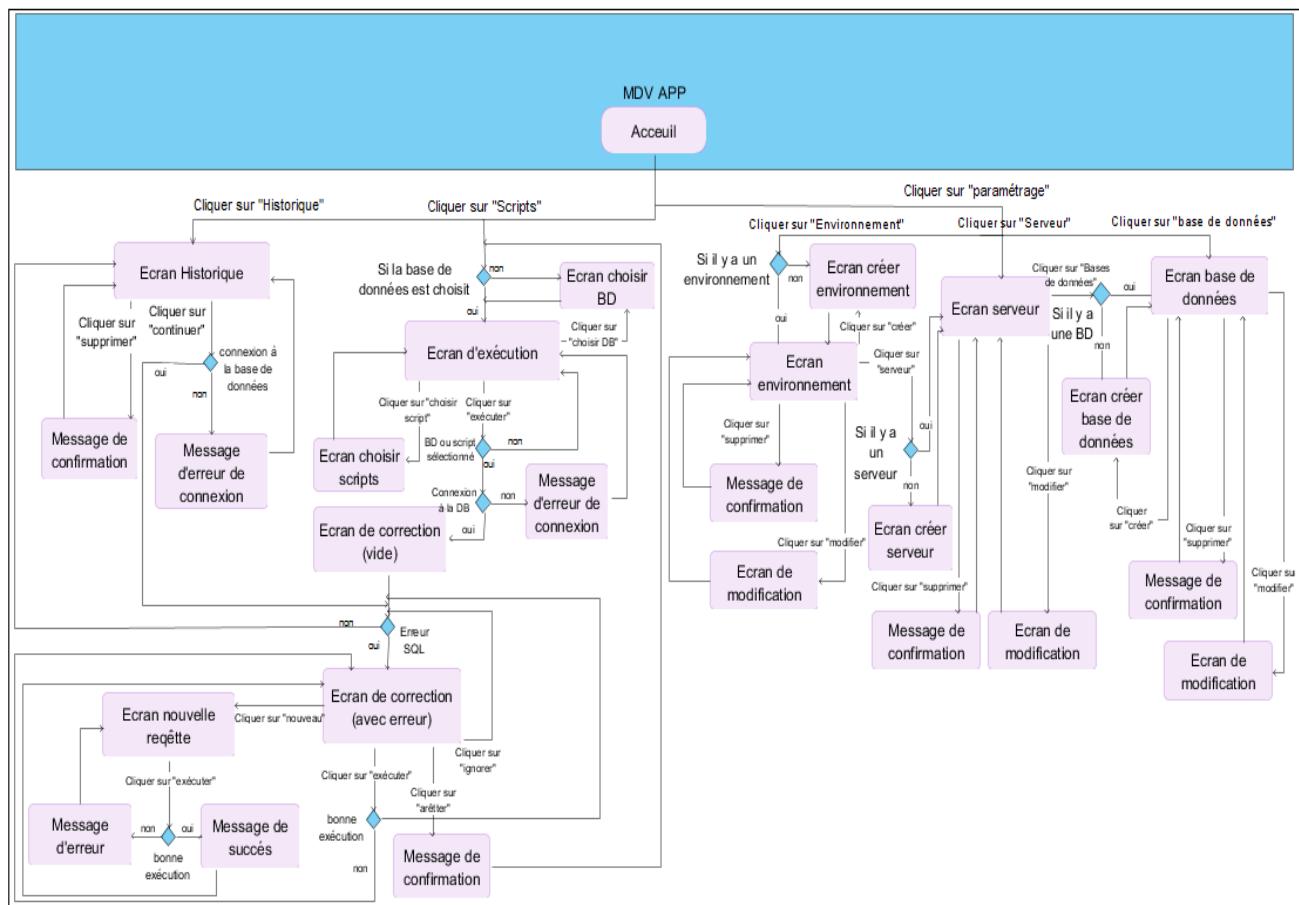


Figure 35: la relation entre les différentes interfaces

Vu que le nombre des écrans de notre application est relativement grand et cela pourrait nuire à la qualité de l'image présentée, on va exposer la cinématique de ces écrans en ces deux figures ci-dessous.

Menu Paramétrage :

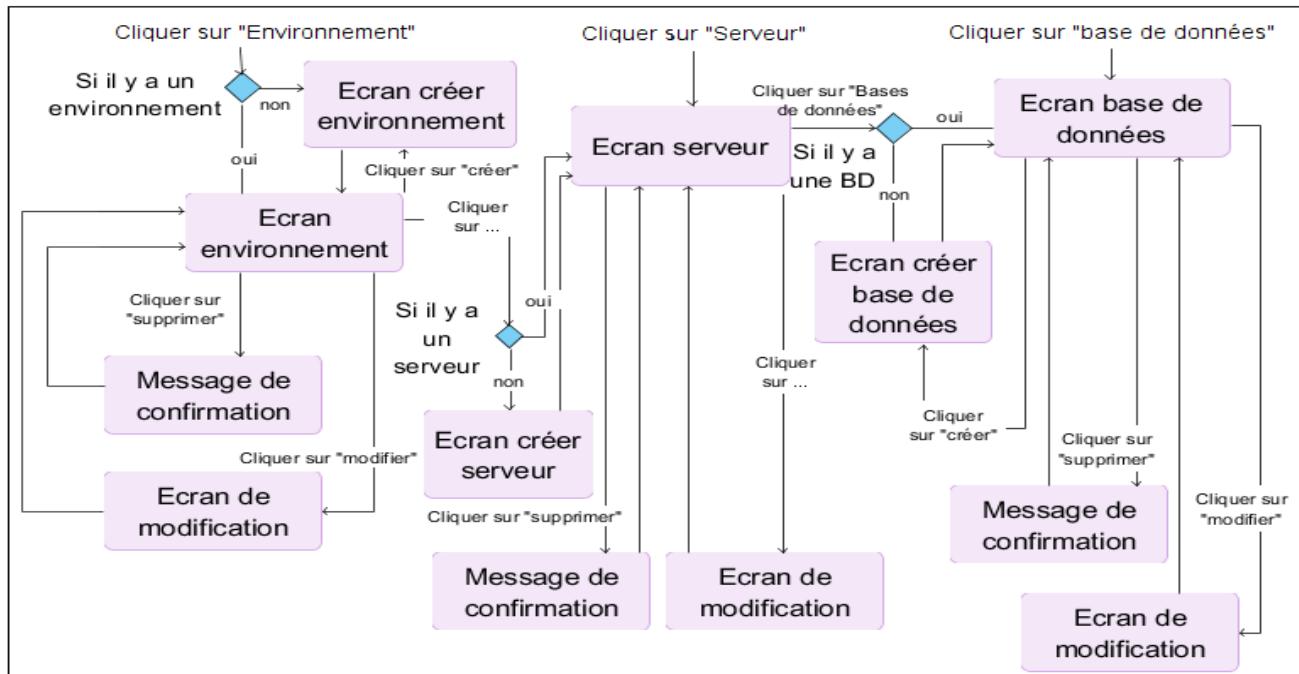


Figure 36: La relation entre les différentes interfaces du menu Paramétrage

Menu Historique et Script :

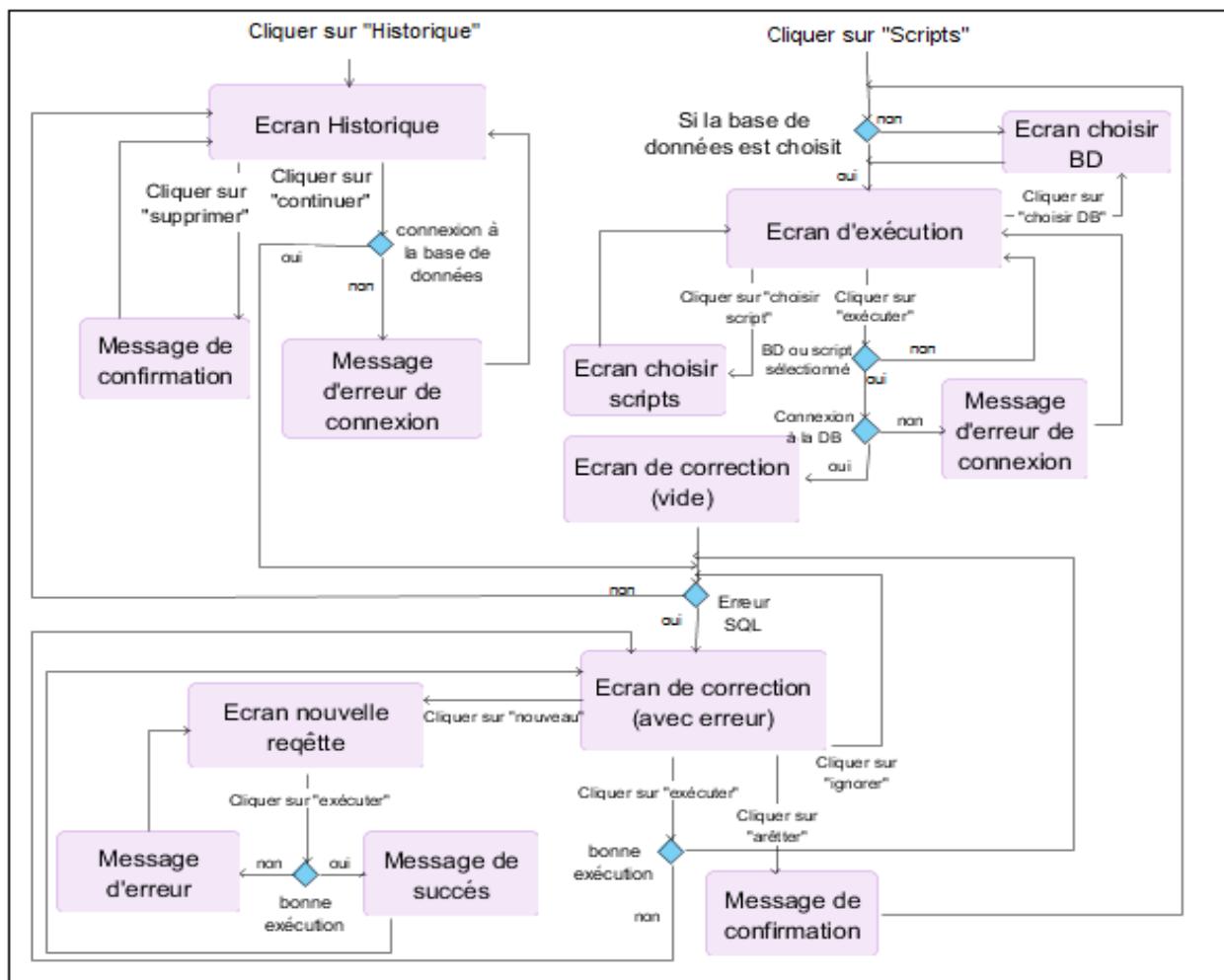


Figure 37: La relation entre les différentes interfaces du menu Historique et Scripts

6. Conclusion :

Dans ce chapitre, nous avons fait la description des démarches suivies durant les phases d'analyse et de conception ainsi que le planning suivi tout au long du projet.

La modélisation UML des composants du projet était fournie, ceci nous a donné une bonne visibilité sur la réalisation, et nous a permis de définir les critères de sélection de notre solution technique et plus précisément le choix d'architecture et des outils de développement. Dans le chapitre suivant, nous détaillerons ces choix et présenterons les Framework utilisés lors de la mise en place de la solution.

Chapitre 3 : Etude technique et mise en œuvre

Introduction :

Comme tout projet informatique, une étude technique de celui-ci est cruciale afin de concrétiser les besoins fonctionnels voulus du projet. Pour se faire nous avons consacré un temps très important pour en sortir d'une application qui suit une architecture bien organisée et structurée.

Dans ce chapitre, nous avons divisé l'étude technique en deux grandes parties commençant premièrement par l'architecture de l'application que ce soit applicative ou bien logicielle, et deuxièmement par une description des différentes technologies que nous avons choisies pour la réalisation de ce projet et que nous croyons utile. Et ce chapitre se terminera par la partie réalisation qui va comporter la description détaillée des différentes interfaces de notre application.

1. Architectures de l'application :

1.1 Architecture logicielle :

La figure ci-dessous montre l'architecture qui a été mise en œuvre dans le cadre de développement de notre solution.

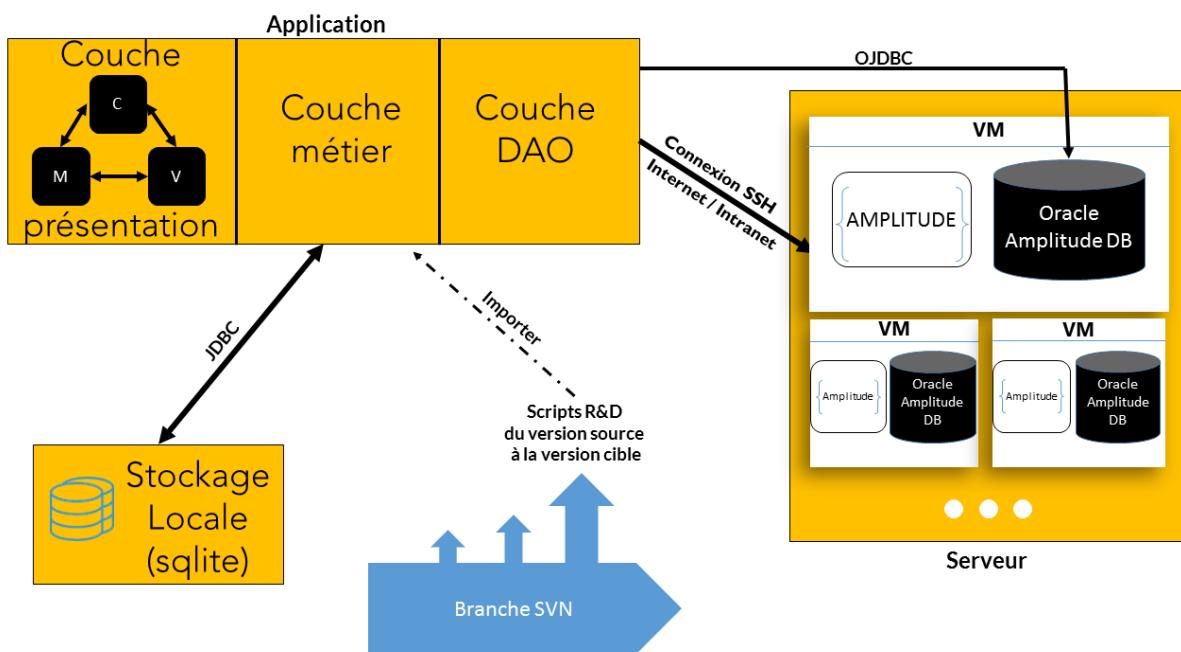


Figure 38 : Architecture d'application.

L'architecture « 3-tiers » implémentée dans l'application est un modèle logique d'architecture applicative qui vise à séparer très nettement trois couches logicielles au sein d'une même application ou système.

L'application est considérée comme un empilement de trois couches, étages, niveaux ou strates qui répondent aux contraintes applicatives majeures, l'architecture devrait répondre aux principes structurants suivants :

❖ Couche Présentation :

La couche présentation relaie les requêtes de l'utilisateur à destination de la couche service, et en retour lui présente les informations renvoyées par les traitements de cette couche. Il s'agit donc d'un assemblage de services métiers et applicatifs offerts par la couche inférieure.

❖ Couche métier :

Elle correspond à la partie fonctionnelle de l'application, celle qui implémente la « logique », et qui décrit les opérations que l'application opère sur les données en fonction des requêtes des utilisateurs, effectuées au travers de la couche présentation.

La couche Métier joue le rôle principal dans la synchronisation de la couche présentation et la couche accès aux données. Elle reçoit tous les événements de l'utilisateur et enclenche les actions à effectuer. Si une action nécessite un changement de données, c'est cette couche qui possède les outils pour demander la modification des données à la couche accès aux données et ensuite avertit la couche présentation que les données ont été changées pour que celles-ci se mettent à jour. Certains événements de l'utilisateur ne concernent pas les données mais la couche présentation elle-même. Dans ce cas, elle demande à la couche présentation de se modifier.

Elle consiste en la partie gérante l'accès aux gisements de données du système. La couche métier n'a pas à s'adapter à ce dernier, ceci est transparent pour elle, ainsi elle accède aux données de manière uniforme (couplage faible) en interrogeant la couche DAO.

❖ Couche accès aux données :

La couche accès aux données assure la gestion des données de l'application et garantit leur intégrité. Elle offre des méthodes pour l'insertion, la suppression et la mise à jour de ces données (CRUD). Cette couche a été gérée grâce au Framework Hibernate qui gère la persistance des objets en base de données relationnelle.

Dans cette approche, les couches communiquent entre elles à travers un "modèle d'échange", et chacune d'entre elles propose un ensemble de services rendus. Les services d'une couche sont mis à disposition de la couche supérieure. On s'interdit par conséquent qu'une couche invoque les services d'une couche plus basse que la couche immédiatement inférieure ou plus haute que



la couche immédiatement supérieure (chaque niveau ne communique qu'avec ses voisins immédiats).

1.2 Architecture Applicative :

1.2.1 Design Pattern

1.2.1.1 MVC

L'organisation d'une interface graphique est délicate. L'architecture "MVC" ne prétend pas à éliminer tous les problèmes, mais fournit une première approche pour ce faire. Offrant un cadre normalisé pour structurer une application, elle facilite aussi le dialogue entre les concepteurs.

L'idée est de bien séparer les données, la présentation et les traitements. Il en résulte les trois parties énumérées en dessous : le modèle, la vue et le contrôleur.

Le MVC est un patron de conception, à trois couches, qui reposent sur la volonté de séparer les données, les traitements et la présentation. Ainsi l'application se retrouve segmentée en trois composants essentiels :

❖ Le modèle :

Représente les données et les règles métiers. C'est dans ce composant que s'effectuent les traitements liés au cœur du métier. Les données peuvent être liées à une base de données, des EJBs ou même des services Web. Il est important de noter que les données sont indépendantes de la présentation. En d'autres termes, le modèle ne réalise aucune mise en forme. Ainsi ces données pourront être affichées par plusieurs vues. Du coup le code du modèle est factorisé : il est écrit une seule fois puis réutilisé par chaque vue.

❖ La vue :

Correspond à l'IHM. Elle présente les données et interagit avec l'utilisateur. Dans notre cas (le cas des applications Desktop), il s'agit d'une interface javaFX.

❖ **Le contrôleur :**

Quant à lui, se charge d'intercepter les requêtes de l'utilisateur, d'appeler le modèle puis de rediriger le résultat vers la vue adéquate. Il ne doit faire aucun traitement. Il ne fait que de l'interception et de la redirection.

1.2.1.2 Inversion de contrôle :

Inversion de contrôle est un principe de génie logiciel par lequel le contrôle d'objets ou de parties d'un programme est transféré dans un conteneur. Il est le plus souvent utilisé dans le contexte de la programmation orientée objet.

Contrairement à la programmation traditionnelle, dans laquelle le code personnalisé appelle une bibliothèque, IoC permet à un Framework de prendre le contrôle du flux d'un programme et de passer des appels au code personnalisé. Pour permettre cela, les Frameworks utilisent des abstractions avec un comportement supplémentaire intégré. Si nous voulons ajouter notre propre comportement, nous devons étendre les classes du Framework ou greffer nos propres classes.

1.2.1.3 Injection des dépendances :

L'injection de dépendances est un mécanisme qui permet d'implémenter le principe de l'inversion de contrôle.

Il consiste à créer dynamiquement (injecter) les dépendances entre les différents objets en s'appuyant sur une description (fichier de configuration ou métadonnées) ou de manière programmatique. Ainsi les dépendances entre composants logiciels ne sont plus exprimées dans le code de manière statique mais déterminées dynamiquement à l'exécution.

1.2.1.4 Programmation orienté aspects :

La programmation orientée aspect, ou POA est un paradigme de programmation qui permet de traiter séparément les préoccupations transversales qui relèvent souvent de la technique, des préoccupations métier, qui constituent le cœur d'une application. Un exemple classique d'utilisation est la journalisation, mais certains principes architecturaux ou modèles de



conception peuvent être implémentés à l'aide de ce paradigme de programmation, comme l'inversion de contrôle.

2. Les Framework et technologies utilisés :

De nos jours, le choix des technologies à utiliser dans un projet informatique influence beaucoup sur la qualité et les performances du logiciel produit, raison pour laquelle nous avons consacré plus de temps et d'effort en travaillant là-dessus, car nous assumons qu'un mauvais choix va dévaloriser notre produit final, ce qui nécessite une analyse profonde avant tout choix final d'une technologie.

Après l'étude des fonctionnalités des différentes technologies, nous avons opté pour une liste des technologies les plus sophistiquées que nous allons développer dans le paragraphe ci-dessous :

2.1 Technologie :

❖ Spring :



Figure 39 : Logo de Spring framework

Spring est un Framework libre pour construire et définir l'infrastructure des applications java, dont il facilite le développement et les tests. Le Framework Spring a tout d'abord été écrit par Rod Johnson et réalisé sous la licence Apache 2.0 en 2003.

Spring est considéré comme un conteneur dit « léger » c'est-à-dire une infrastructure similaire à un serveur d'applications J2EE. Il prend donc en charge la création d'objets et la mise en relation d'objets par l'intermédiaire d'un fichier de configuration qui décrit les objets à fabriquer et les relations de dépendances entre ces objets. Le gros avantage par rapport aux serveurs d'application est qu'avec Spring, les classes n'ont pas besoin d'implémenter une interface quelconque pour être prises en charge par le Framework (au contraire des serveurs

d'applications J2EE et des EJBs). C'est en ce sens que Spring est qualifié de conteneur « léger ».

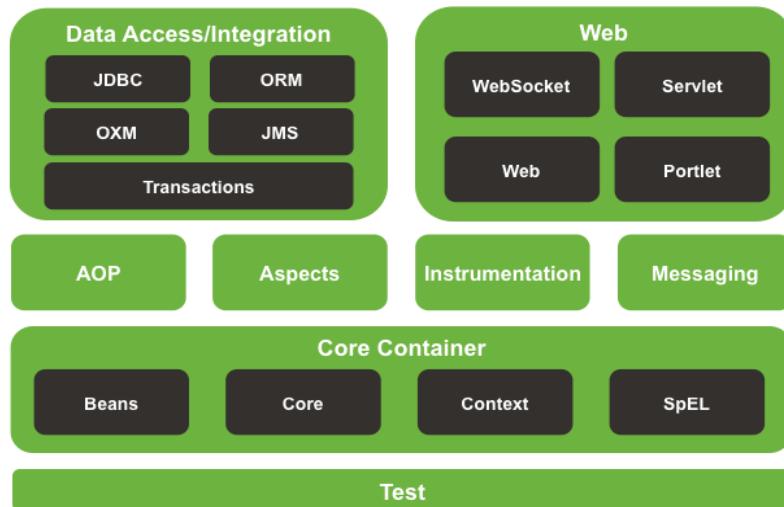


Figure 40 : Les modules de Spring

❖ Spring boot :



Figure 41 : Logo de Spring Boot

Spring Boot est un projet ou un micro framework qui a notamment pour but de faciliter la configuration d'un projet Spring et de réduire le temps alloué au démarrage d'un projet. Pour arriver à remplir cet objectif, Spring Boot se base sur plusieurs éléments :

- Un site web qui vous permet de générer rapidement la structure de votre projet en y incluant toutes les dépendances Maven nécessaires à votre application. Cette génération est aussi disponible via le plugin Eclipse STS.
- L'utilisation de « Starters » pour gérer les dépendances. Spring a regroupé les dépendances Maven de Spring dans des « méga dépendances » afin de faciliter la gestion de celles-ci.
- L'auto-configuration, qui applique une configuration par défaut au démarrage de votre application pour toutes dépendances présentes dans celle-ci. Cette configuration s'active à partir du moment où vous avez annoté votre application avec `@Enable Auto`

Configuration ou @Spring Boot Application. Bien entendu cette configuration peut être surchargée via des propriétés Spring prédefinie ou via une configuration Java.

- Spring Boot simplifie le déploiement d'application Spring en offrant la possibilité d'intégrer directement un serveur Tomcat dans votre exécutable. Au lancement de celui-ci, un Tomcat embarqué sera démarré afin de faire tourner votre application.

❖ **Spring data :**



Figure 42 : Logo de Spring Data

Spring Data est un projet supplémentaire de Spring créé il y a quelques années pour répondre aux besoins d'écrire plus simplement l'accès aux données et d'avoir une couche d'abstraction commune à de multiples sources de données.

Spring Data s'interface avec plusieurs sources de données parmi lesquelles JPA, Neo4j, MongoDB, GemFire, Hadoop, ElasticSearch, REST, Redis, Couchbase et quelques autres.

❖ **JavaFX :**



Figure 43: Logo de javafx

JavaFX est une technologie créée par Sun Microsystems qui appartient désormais à Oracle, c'est une évolution de Java, elle permet de créer des RIA (Rich Internet Applications), c'est-à-dire des applications contenant des vidéos, de la musique, des effets graphiques très intéressants, etc. JavaFX permet de créer des applications mobiles, des applications sur poste de travail et des applications Web. JavaFX devient la bibliothèque de création d'interface graphique officielle du langage Java remplaçant son prédecesseur Swing étant abandonné.

❖ **Hibernate :**



Figure 44: Logo de hibernate

Hibernate est une solution open source de type ORM qui permet de faciliter le développement de la couche persistance d'une application. Hibernate permet donc de représenter une base de données en objets Java et vice versa. Il facilite la persistance et la recherche de données dans une base de données en réalisant lui-même la création des objets et les traitements de remplissage de ceux-ci en accédant à la base de données. La quantité de code ainsi épargnée est très importante d'autant que ce code est généralement fastidieux et redondant. Hibernate est très populaire notamment grâce à de ses bonnes performances et à son ouverture à de nombreuses bases de données. Les bases de données supportées sont les principales du marché : DB2, Oracle, MySQL, PostgreSQL, Sybase, SQL Server, SQLite, etc.

❖ **SQLite :**



Figure 45: Logo de SQLite

SQLite est une bibliothèque logicielle qui fournit un système de gestion de base de données relationnelle. La lite dans SQLite signifie poids léger en termes d'installation, d'administration de base de données et de ressource requise. SQLite a les caractéristiques notables suivantes :

- **Serverless :**

Normalement, un SGBDR tel que MySQL, PostgreSQL, etc., nécessite un processus de serveur séparé pour fonctionner. Les applications qui souhaitent accéder au serveur de base de données utilisent le protocole TCP / IP pour envoyer et recevoir des demandes. C'est ce qu'on appelle l'architecture client / serveur.

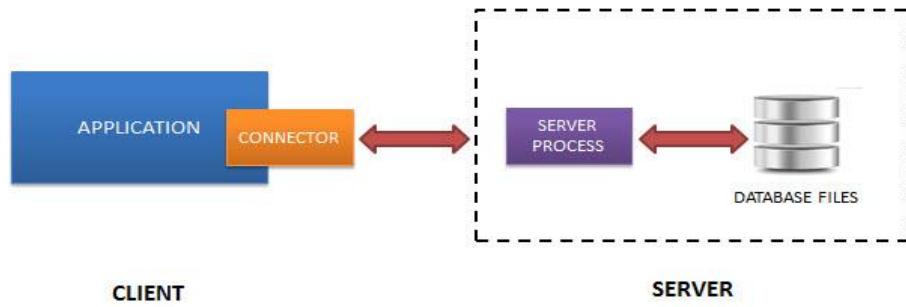


Figure 46: Architecture client/serveur du SGBDR

Au contraire SQLite ne nécessite pas l'exécution d'un serveur. Les applications interagissent avec la base de données SQLite lire et écrire directement à partir des fichiers de base de données stockés sur le disque.

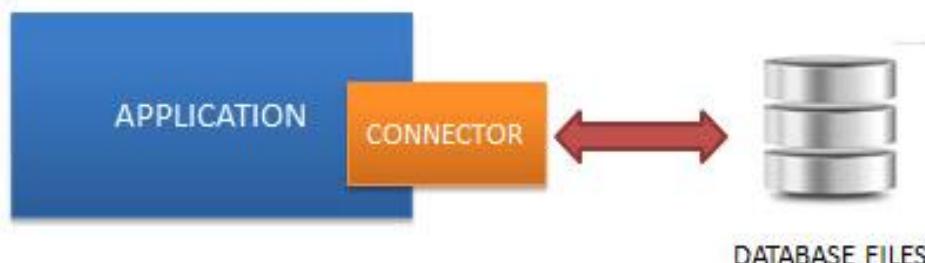


Figure 47: Architecture SQLite sans serveur

- **Zéro configuration**

En raison de l'architecture sans serveur, vous n'avez pas besoin d'installer SQLite avant de l'utiliser. Aucun processus de serveur n'a besoin d'être configuré, démarré et arrêté. En outre, SQLite n'utilise aucun fichier de configuration.

- **Transactionnel**

Toutes les transactions dans SQLite sont entièrement compatibles ACID. Cela signifie que toutes les requêtes et les modifications sont atomiques, cohérentes, isolées et durables.

❖ **Le protocole SSH :**

SSH est à la fois un programme informatique et un protocole de communication sécurisé. Le protocole de connexion impose un échange de clés de chiffrement en début de connexion. Par la suite, tous les segments TCP sont authentifiés et chiffrés. Il devient donc impossible d'utiliser un sniffer pour voir ce que fait l'utilisateur. Le protocole SSH a été conçu avec l'objectif de remplacer les différents protocoles non chiffrés comme telnet, ftp et ssh.

Le protocole SSH existe en deux versions majeures : la version 1.0 qui souffrait néanmoins de problèmes de sécurité dans la vérification de l'intégrité des données envoyées ou

reçues, La version 2 qui était à l'état de brouillon jusqu'en janvier 2006 est déjà largement utilisée à travers le monde.

Nous avons utilisé une implémentation pure de SSHv2 en java avec le module JSCH (Java Secure Channel) qui permet de se connecter à un serveur sshd et d'utiliser le transfert de port, le transfert X11, le transfert de fichiers, etc.

❖ JDBC

La technologie JDBC (Java Data Base Connectivity) est une API fournie avec Java depuis sa version 1.1, permettant de se connecter à des bases de données, c'est-à-dire que JDBC constitue un ensemble de classes permettant de développer des applications capables de se connecter à des serveurs de bases de données.

L'API JDBC a été développée de telle façon à permettre à un programme de se connecter à n'importe quelle base de données en utilisant la même syntaxe, c'est-à-dire que l'API JDBC est indépendante du SGBD.

3. Les outils et environnements :

❖ Maven :



Figure 48: Logo de maven

Maven, un mot yiddish signifiant accumulateur de connaissances, a été initialement lancé comme une tentative de simplifier les processus de construction dans le projet Jakarta Turbine. C'est un outil de construction de projets open source développé par la fondation Apache. Il permet notamment :

- D'automatiser certaines tâches : compilation, tests unitaires et déploiement des applications qui composent le projet ;
- De gérer des dépendances vis-à-vis des bibliothèques nécessaires au projet ;
- De générer des documentations concernant le projet.

❖ Eclipse :



Figure 49: Logo d'eclipse IDE

Eclipse est un environnement de développement intégré utilisé dans la programmation informatique, et l'IDE Java le plus utilisé. Il contient un espace de travail de base et un système de plug-in extensible pour personnaliser l'environnement. Eclipse est principalement écrit en Java et son utilisation principale est le développement d'applications Java, mais il peut également être utilisé pour développer des applications dans d'autres langages de programmation via des plug-ins. Il peut également être utilisé pour développer des documents avec LaTeX (via un plug-in TeXlipse) et des packages pour le logiciel Mathematica.

❖ Visual Paradigm :



Figure 50 : Logo de Visual Paradigm

Visual Paradigm est un outil UML CASE prenant en charge UML 2, SysML et la notation BPMN du groupe OMG. En plus de la prise en charge de la modélisation, il fournit des capacités de génération de rapports et d'ingénierie de code, y compris la génération de code. Il peut effectuer une ingénierie inverse des diagrammes à partir du code et fournir une ingénierie d'aller-retour pour différents langages de programmation.

❖ Gantt Project :



Figure 51: Logo de GanttProject

GanttProject est un logiciel libre de gestion de projet écrit en Java, ce qui permet de l'utiliser sur divers système d'exploitation (Windows, Linux, MacOS); il permet d'éditer un diagramme de Gantt. L'outil permet de créer des diagrammes de Gantt, des diagrammes de ressources et des réseaux PERT.

❖ SVN:



Figure 52: Logo de SVN

Subversion ou SVN est un logiciel de gestion de versions, distribué sous licence Apache, fonctionne donc sur le mode client-serveur, avec un serveur centralisé et unique où se situent les fichiers constituant la référence, des postes clients sur lesquels se trouvent les fichiers recopiés depuis le serveur.

Dans notre cas nous avons utilisé le client TortoiseSVN pour récupérer le kit de migration depuis la branche SVN du serveur.

❖ SQLite Studio :



Figure 53: Logo de SQLite Studio

SQLite Studio est un gestionnaire de base de données SQLite avec les fonctionnalités suivantes :

- Portable - pas besoin d'installer ou de désinstaller. Il suffit de télécharger, décompresser et exécuter ;
- Interface intuitive et simple ;
- Puissant, léger et rapide ;
- Cross-plate-forme - fonctionne sur Windows, Linux et MacOS X ;
- Exportation vers différents formats (instructions SQL, CSV, HTML, XML, PDF, JSON) ;
- Importation de données à partir de différents formats (CSV, fichiers texte personnalisés [expressions régulières]) ;
- De nombreux petits ajouts, comme le code de formatage, l'historique des requêtes exécutées dans les fenêtres de l'éditeur, la vérification de la syntaxe à la volée, etc ;
- Support Unicode ;

-
- Open source et gratuit - Publié sous licence GPLv3.

❖ **SQL Developer :**



Figure 54: Logo de SQL Developer

Oracle SQL Developer est un environnement de développement intégré multiplateforme, fourni gratuitement par Oracle Corporation et utilisant la technologie Java (Java Development Kit). C'est un outil graphique permettant d'interroger des bases de données Oracle à l'aide du langage SQL.

Oracle SQL Developer permet le développement de A à Z d'applications en PL/SQL, la mise à disposition de feuilles de travail pour exécuter les requêtes et les scripts, une console pour l'administration de bases de données (DBA), une interface pour la génération de rapports (reporting), une solution complète de conception du modèle de données et une interface de migration permettant de migrer les bases de données d'éditeurs tiers vers Oracle.

4. Réalisation et présentation des interfaces de l'application :

Après un long chemin du développement de ce projet, commençant de l'analyse des besoins fonctionnels passant par une conception détaillée et en concrétisant cette dernière dans un projet complet, nous sommes finalement arrivés à la phase où nous devons vous présenter le fruit de notre travail par le biais d'interfaces graphiques bien choisies.

Cette dernière partie donc se focalisera sur la présentation de quelques interfaces Homme/Machine maintenues par une description minutieuse.

Pour une vue plus générale sur toutes les interfaces de l'application, nous avons décrit d'une manière exhaustive la totalité des interfaces dans les annexes (voir l'annexe A).

4.1 Interface d'authentification :

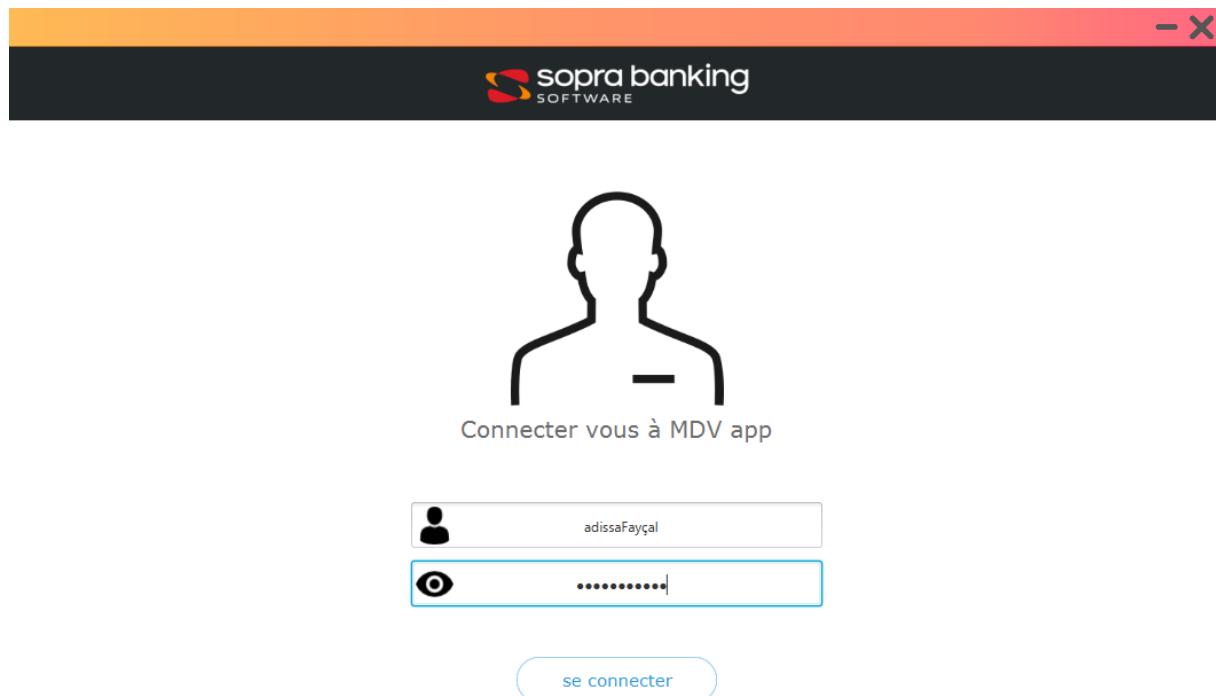


Figure 55 : Interface d'authentification.

Lors du lancement de l'application, une interface d'authentification apparaît indiquant à l'utilisateur d'insérer son nom d'utilisateur et son mot de passe. Au cas où les identités sont invalides l'authentification échoue.

4.2 Interface pour choisir la base de données :

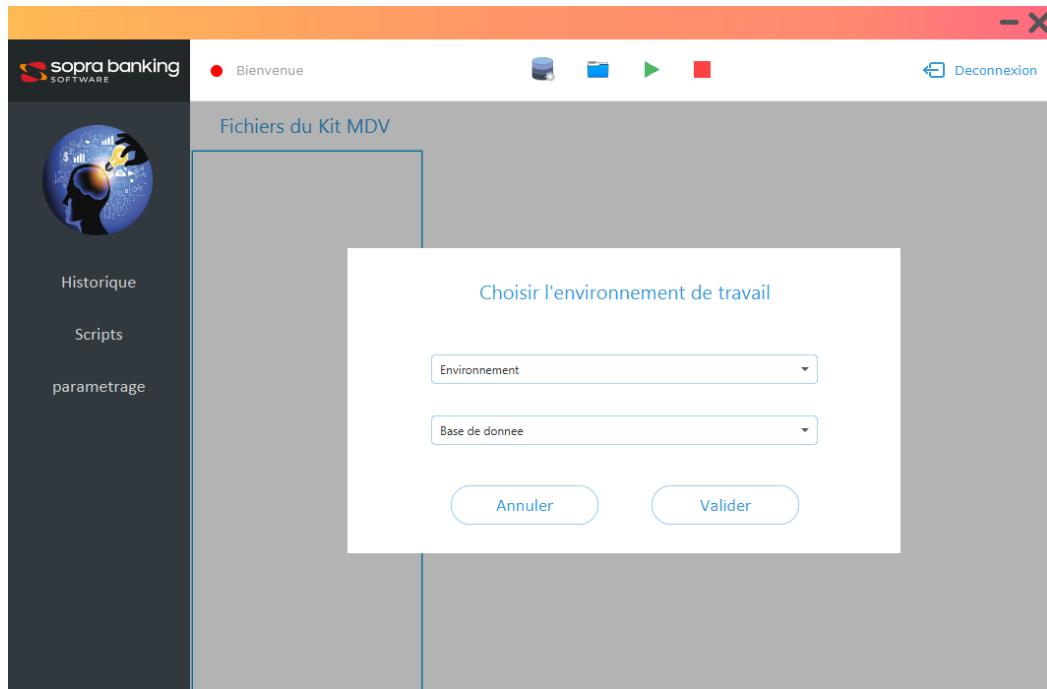


Figure 56 : Interface du choix de la base de données.

Lorsque l'utilisateur veut exécuter les scripts R&D, il se dirige vers l'onglet « Scripts » c'est à cet endroit là où il peut dérouler ses scripts, mais avant l'exécution des scripts une boîte de dialogue se présente afin de choisir l'environnement et la base de données cible pour exécuter les scripts que cette interface ci-dessus nous montre. Par suite, l'utilisateur doit choisir à la fois le dossier des scripts pour que l'application les importent, et le fichier à exécuter après l'importation de ceux-ci.

La création de l'environnement, serveur et base de données et l'importation des scripts sont donc des étapes préalables que nous n'avons pas citées ici, mais dans l'annexe (voir l'annexe A).

4.3 Interface pour la gestion des erreurs :

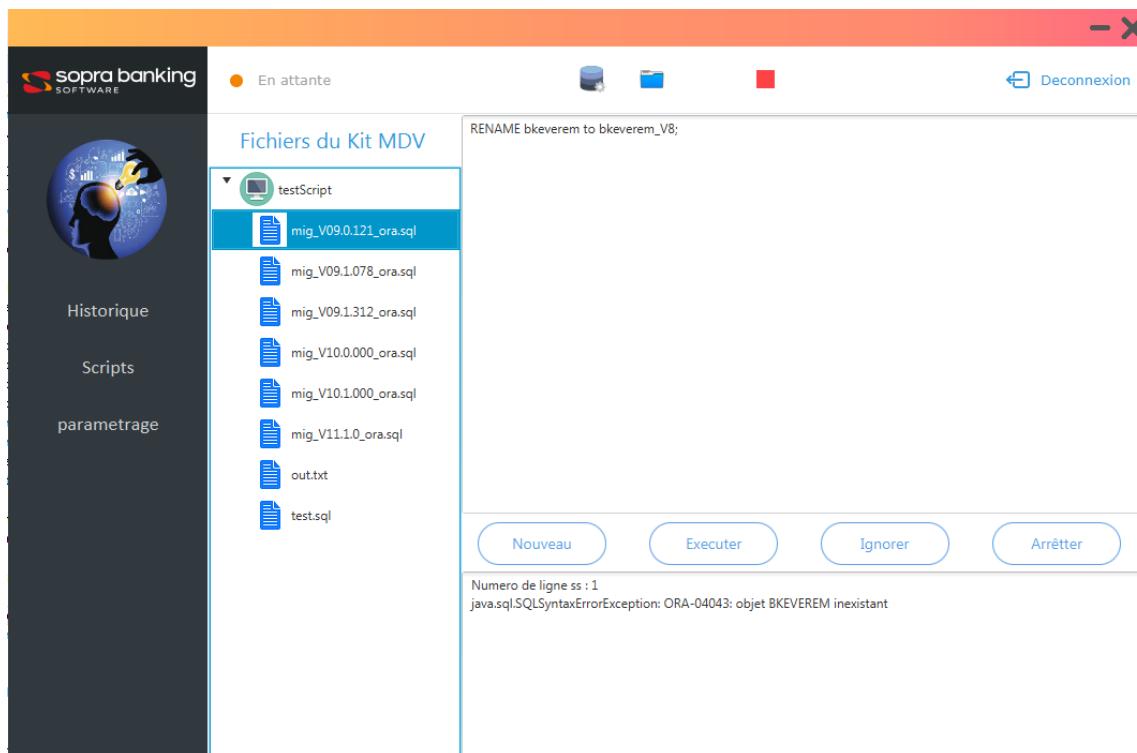


Figure 57 : Interface d'erreur lors de l'exécution.

Après le lancement du script, l'exécution du script dans la base de données choisie s'effectue sans arrêt, jusqu'à l'occurrence d'une exception que l'utilisateur doit corriger.

Cette interface présente à l'utilisateur plusieurs objets graphiques. Une zone de texte qui affiche la requête erronée et une autre zone contenant l'erreur SQL déclenchée avec les différents boutons pour effectuer une correction à savoir :

- Nouveau : bouton pour l'exécution d'une nouvelle requête ;
- Exécuter : bouton pour l'exécution de la requête en question après modification ;
- Ignorer : bouton pour commenter et donc bien ignorer la requête en question ;
- Arrêter : bouton pour arrêter le script en cours d'exécution.

4.4 Interface d'historique :

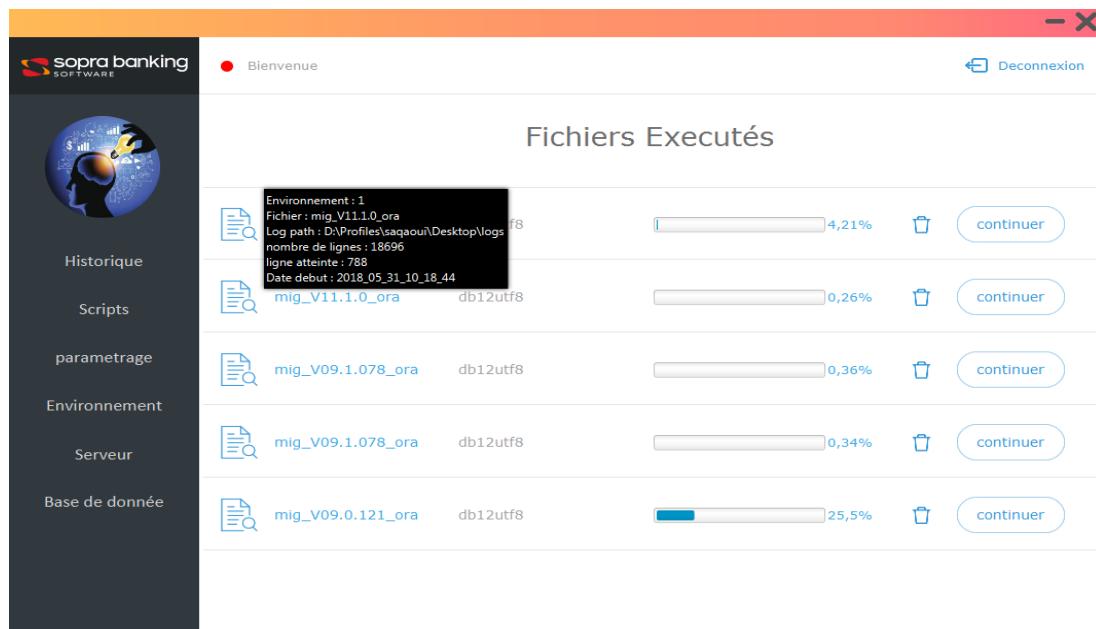


Figure 58 : Interface d'historique.

L'utilisateur peut à tout moment arrêter l'exécution d'un script donné, ou bien se déconnecter avant que le script soit complètement exécuter, l'application à comme avantage un mécanisme de sauvegarde de l'état d'exécution du script pour que l'utilisateur puisse continuer l'exécution d'une façon simple via l'onglet « Historique ».

Dans cette interface l'utilisateur peut :

- Voir tous les scripts qu'il a déjà exécuté ;
- Afficher les informations d'un script en pressant ce dernier ;
- Reprendre l'exécution d'un script ;
- Supprimer une exécution.

5. Conclusion :

Dans ce chapitre nous avons achevé la dernière phase de l'élaboration de notre projet.

Nous avons entamé par la présentation de l'architecture logicielle et applicative, ensuite nous avons cité les outils de développement et les technologies utilisées durant la mise en œuvre de notre solution, en concluant par une exposition des interfaces du projet.

Bilan de stage :

1. Objectifs atteints :

Tout au long de la période du stage, nous avons été amenés à passer par toutes les phases inhérentes du développement logiciel. Nous avons commencé par la récolte et l'analyse des besoins du client, jusqu'à la réalisation effective de la solution finale, son implémentation et son test. Tout ceci, sous une contrainte de délai et de qualité qui nous a forcé à travailler dur et à user de notre esprit d'analyse pour pouvoir livrer à temps un produit optimal et de qualité.

Finalement, nous reconnaissions être très satisfaits de notre projet ainsi que de notre réalisation qui répond aux besoins initiaux.

2. Obstacles affrontés :

Bien que le projet ait été bénéfique en termes de plusieurs aspects, sa réalisation effective n'a pas été évidente. En effet, l'un des obstacles qui a ralenti le départ du projet, était le fait de cerner le métier bancaire et notamment les concepts clés sur lesquels repose la problématique,

Une des difficultés qui a obstrué le travail réalisé est aussi l'utilisation d'une multitude de Framework et d'outils, notamment des outils gestion de projets et d'intégration continue, que nous n'avons jamais eu l'occasion de mettre en pratique. Nous avons dû alors nous documenter à propos de ces technologies, et aussi nous renseigner auprès de nos collègues pour des informations complémentaires.

3. Enseignements personnels

Nous avons eu la chance de trouver un stage dans ce champ qui nous intéresse particulièrement. Cette expérience nous a avant tout permis d'acquérir de solides bases tant théoriques que pratiques dans ce domaine.

Sur le plan technique, ce projet a été une occasion pour nous de connaître de nouvelles technologies. Nous avons également pu apprendre quelques bonnes pratiques de conception et de développement pour fournir une application réutilisable et maintenable dans le temps.

Au sein de l'entreprise, nous avons été confrontés à de réelles problématiques, ce qui a changé notre vision des projets informatiques professionnels, et nous a aidé à monter en compétence en tout ce qui concerne les relations professionnelles.

La contribution à un tel projet de grande envergure, et la criticité de la qualité des logiciels qui doivent être fournis, notamment pour le domaine bancaire, ont créé en nous un esprit de rigueur qui vise la perfection et qui n'admet pas de compromis. Tout ceci en cherchant les solutions les plus optimales que ce soit en termes de temps ou de ressources.

Conclusion générale :

Comme il a été décliné tout au long de ce rapport, notre projet de fin d'études avait pour but la mise en place d'une application permettant la montée de version d'Amplitude.

La production de cette application a réussi en adoptant la méthodologie Agile eMedia, qui commence par une étude préliminaire des spécifications et des exigences du client, afin de ressortir un cahier des charges fonctionnelles reflétant son besoin.

Ensuite, une analyse conceptuelle qui nous a permis de cerner toutes les fonctionnalités requises pour la solution.

Quant aux perspectives envisagées pour la continuité du projet, au court et moyen terme notre équipe vise à ajouter d'autres fonctionnalités dans notre solution qui vont faciliter la mission de l'équipe de la Montée de version dans un futur proche.

Sur le plan humain, l'enrichissement fut incontestable puisque nous avons pu, d'un côté, développer notre indépendance mais surtout jouir des avantages que présente le travail en équipe. De plus, nous avons appris l'importance de la recherche et de la communication pour l'obtention des bonnes informations, ainsi que l'importance de la gestion du temps et de la planification des tâches pour le bon déroulement des travaux.

Dans une vision professionnelle, ce projet fut une première grande expérience dans notre carrière. Il nous a permis de mieux appréhender l'activité globale de l'organisation qui nous a accueilli, et il nous a également offert l'occasion de vivre sur le terrain avec des professionnels afin de mieux comprendre leur vie dans l'organisme, leur méthodologie de travail, les problèmes qu'ils rencontrent et comment ils les résolvent.

Pour conclure, ce projet nous a offert une bonne préparation à notre insertion et intégration professionnelle et fut pour nous une expérience enrichissante et complète qui conforte notre désir d'exercer notre futur métier d'ingénieur.

Annexes

Annexe A : Interfaces de l'application

Cette annexe a pour objectif d'introduire d'une façon exhaustive les différentes fonctionnalités que propose notre application destinée à la montée de version d'une base de données. Pour se faire, ci-dessous un cas concret de ce qu'un utilisateur peut affronter en utilisant l'application.

❖ Authentification :

Quand l'application se lance l'utilisateur fait face à une interface d'authentification, il doit donc taper son nom et son mot de passe dans les champs correspondants. Une fois que le client clique sur « connecter », le système vérifie les données entrées. Si ces dernières sont valides le système renvoie l'interface d'accueil, et il réaffiche l'interface d'authentification avec un message d'erreur dans le cas contraire.

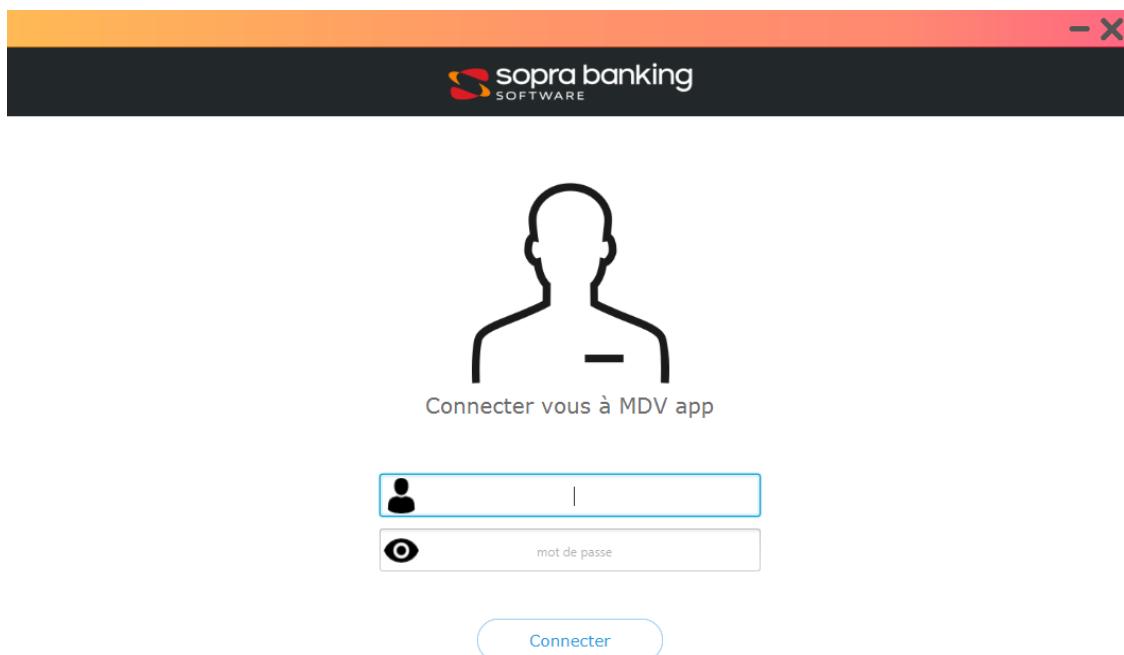


Figure 59: Interface d'authentification.

❖ Interface Principale :

Lorsque la connexion est bien établie, l'interface ci-dessous est renvoyée, elle représente l'historique des exécutions déjà faites par l'utilisateur qu'il peut les poursuivre ou bien les supprimer.

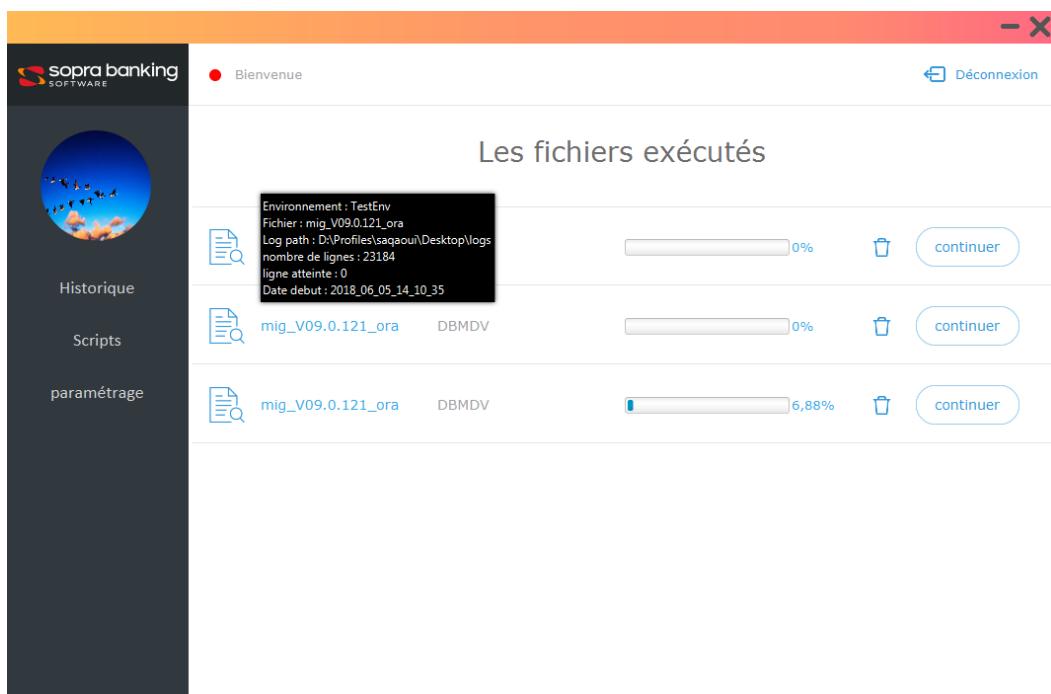


Figure 60:Interface principale.

❖ Gestion des environnements :

Cette partie va présenter les différentes interfaces qui gèrent les environnements à savoir la création, l'affichage, la modification et la suppression.

- Création d'un environnement :

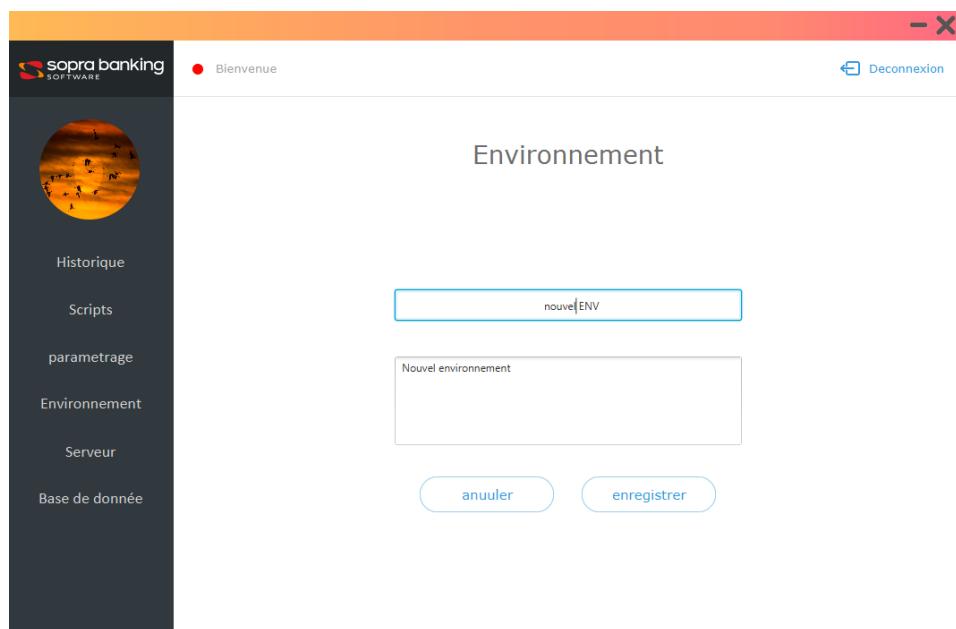
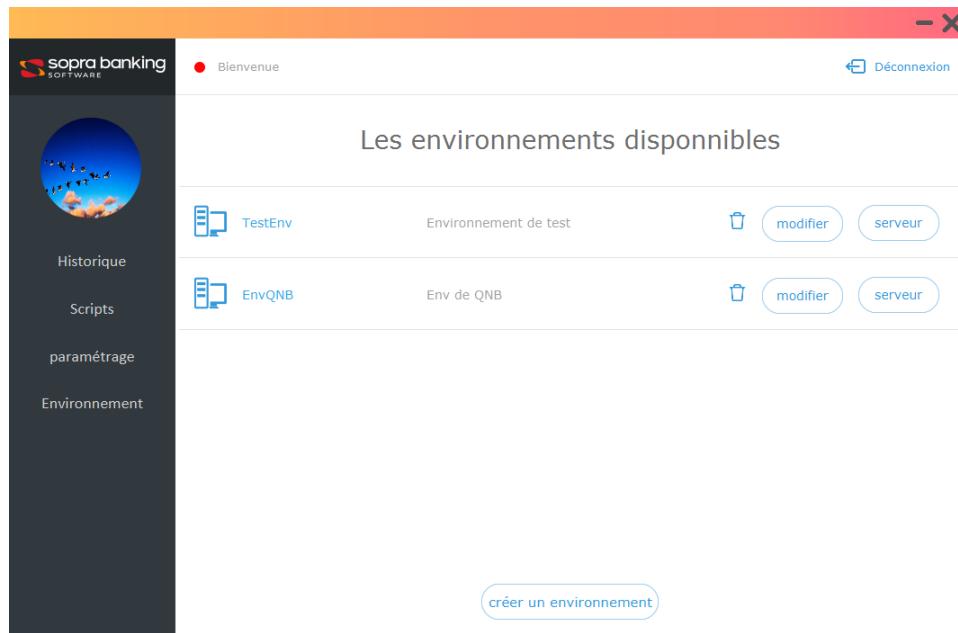


Figure 61 : Interface de création d'un environnement.

- Affichage des environnements :



The screenshot shows the 'Les environnements disponibles' (Available Environments) page. At the top right, there are 'Bienvenue' (Welcome), 'Déconnexion' (Logout), and a close button. On the left, a sidebar has icons for 'Historique' (History), 'Scripts', 'paramétrage' (Configuration), and 'Environnement'. The main area lists two environments: 'TestEnv' (Environnement de test) and 'EnvQNB' (Env de QNB). Each environment entry includes a trash icon, a 'modifier' (Edit) button, and a 'serveur' (Server) button. At the bottom right is a 'créer un environnement' (Create Environment) button.

Environnement	Description	Actions
TestEnv	Environnement de test	trash, modifier, serveur
EnvQNB	Env de QNB	trash, modifier, serveur

Figure 62 : Interface d'affichage des environnements.

- Modification d'un environnement :

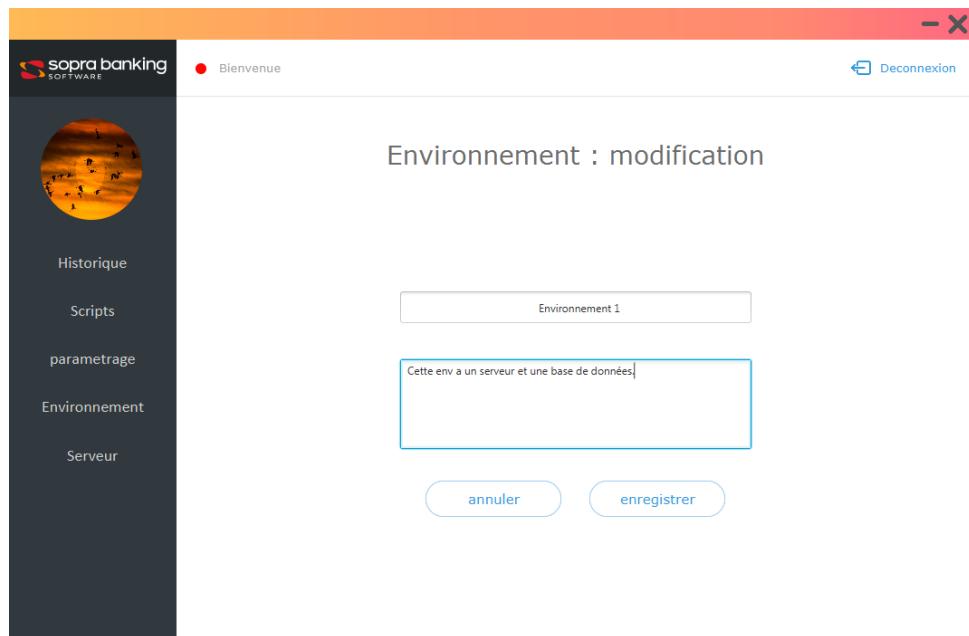


Figure 63 : Interface de modification d'un environnement.

- Suppression d'un environnement :



Figure 64 : Interface de suppression d'un environnement.

Si l'utilisateur clique sur le bouton de suppression l'application affiche le message de confirmation suivant :

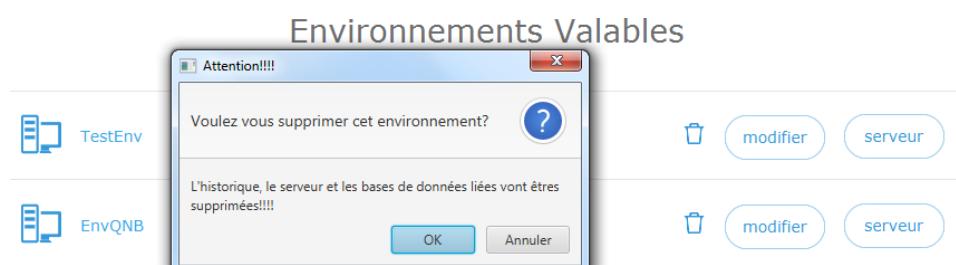


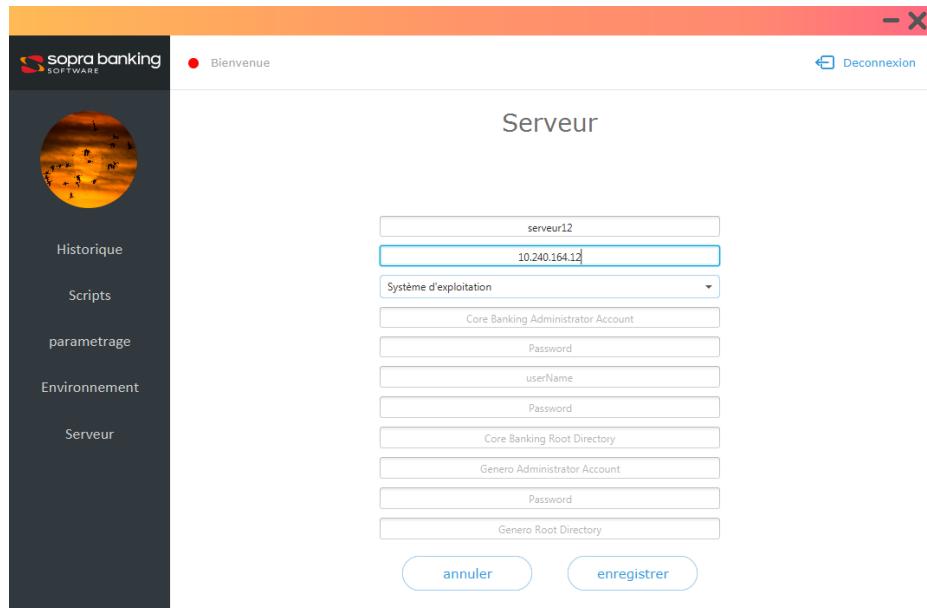
Figure 65 : message de confirmation de la suppression d'un environnement.

❖ Gestion des Serveurs :

Cette partie va présenter les différentes interfaces qui gèrent les serveurs à savoir la création, l'affichage, la modification et la suppression.

- Création d'un serveur :

Cette interface permet de créer un nouveau serveur. Une fois que l'utilisateur clique sur « enregistrer », le système vérifie la connexion avec le serveur utilisant les données entrées. En cas d'échec, il réaffiche la même l'interface avec un message d'erreur. Si la connexion est établie le système affiche le serveur créé.



The screenshot shows the 'sopra banking SOFTWARE' interface. On the left, there's a sidebar with icons for Historique, Scripts, paramétrage, Environnement, and Serveur. The main area has a title 'Serveur' and contains a form for creating a new server. The fields are as follows:

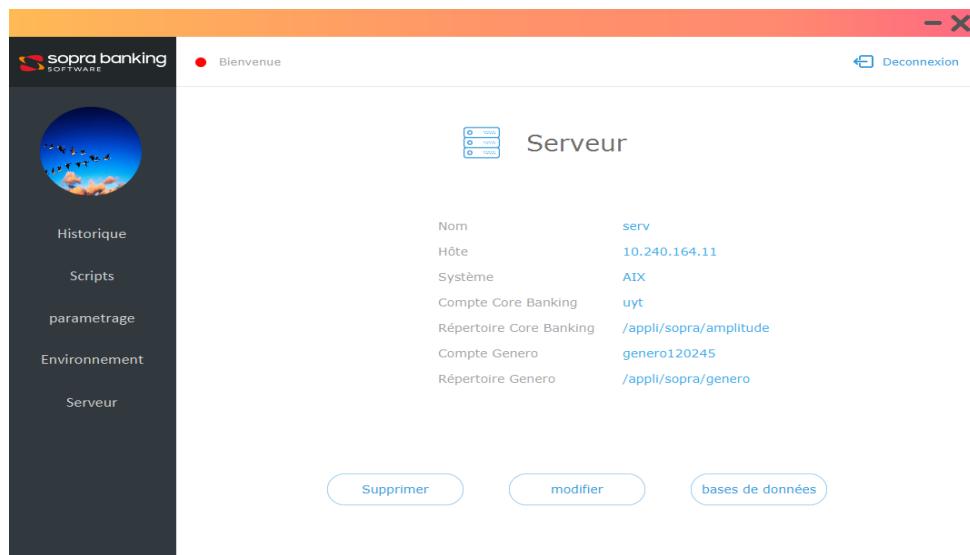
- Nom : serveur12
- IP : 10.240.164.12
- Système d'exploitation : Core Banking Administrator Account
- Core Banking Root Directory
- Genero Administrator Account
- Genero Root Directory

At the bottom are two buttons: 'annuler' (cancel) and 'enregistrer' (register).

Figure 66 : interface de création d'un serveur.

- Affichage d'un serveur :

Cette interface nous permet de visualiser les informations du serveur.



Nom	serv
Hôte	10.240.164.11
Système	AIX
Compte Core Banking	uyt
Répertoire Core Banking	/appli/sopra/amplitude
Compte Genero	genero120245
Répertoire Genero	/appli/sopra/genero

Figure 67 : Interface d'affichage du serveur.

- Modification d'un serveur :

Cette interface permet la modification des informations du serveur.



Figure 68: interface de modification d'un serveur

-
- Suppression d'un serveur :

Si l'utilisateur clique sur le bouton de suppression l'application affiche le message de confirmation suivant :

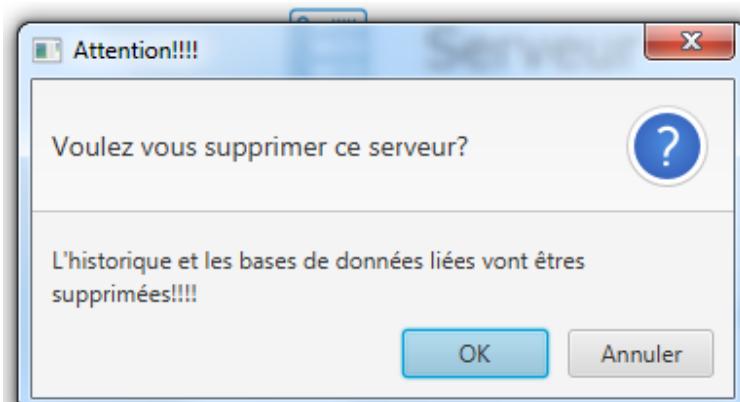


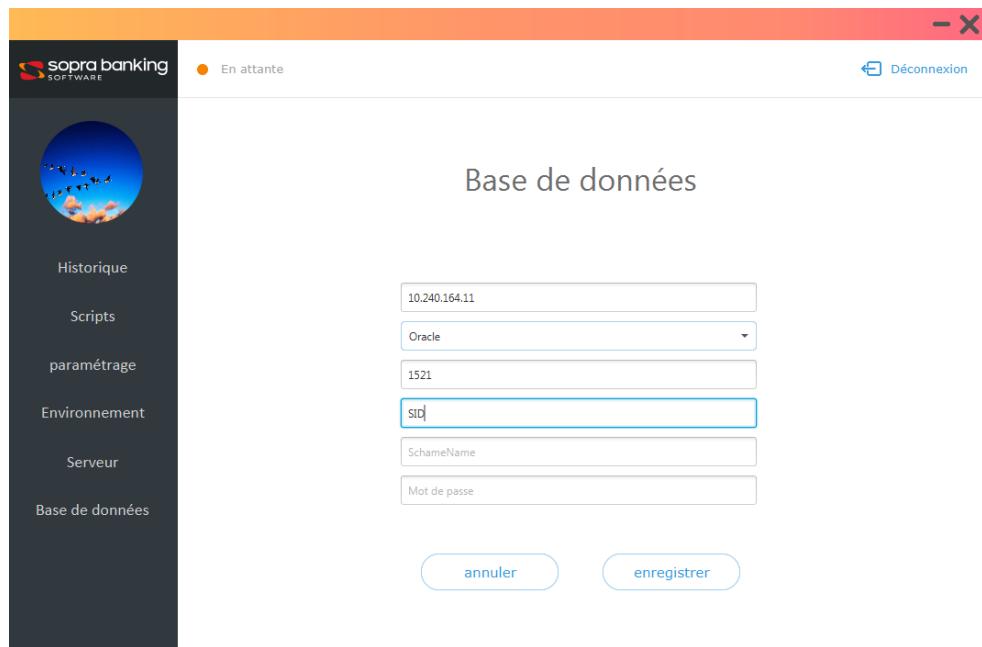
Figure 69 : Message de confirmation du suppression d'un serveur.

❖ Gestion des bases de données :

Cette partie va présenter les différentes interfaces qui gèrent les bases de données.

- Création d'une base de données :

Cette interface permet de créer une nouvelle base de données. Une fois que l'utilisateur clique sur « enregistrer », le système vérifie la connexion avec la nouvelle base de données en utilisant les données entrées. En cas d'échec, il réaffiche l'interface avec un message d'erreur. Si la connexion est établie le système affiche la base de données créée.

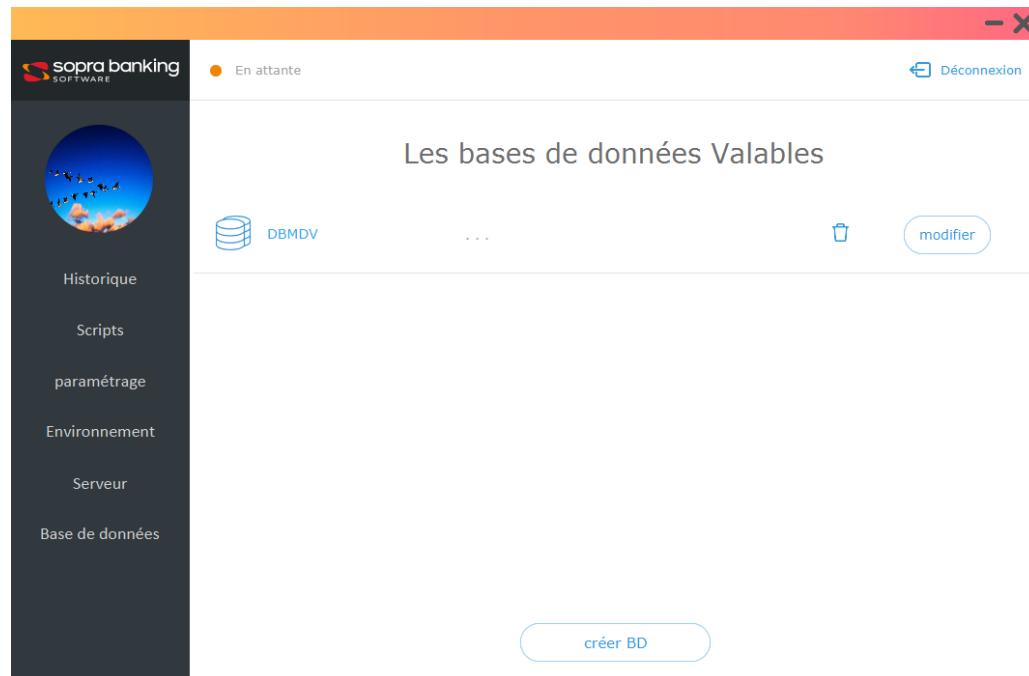


The screenshot shows the 'Base de données' creation screen. On the left, a sidebar lists navigation options: Historique, Scripts, paramétrage, Environnement, Serveur, and Base de données. The 'Base de données' option is selected. The main area is titled 'Base de données'. It contains several input fields: 'IP' (10.240.164.11), 'Type de base de données' (Oracle), 'Port' (1521), 'SID' (sid), 'SchameName' (empty), and 'Mot de passe' (empty). Below the fields are two buttons: 'annuler' (cancel) and 'enregistrer' (register).

Figure 70 : Interface de création d'une base de données.

- Affichage des bases de données :

Cette interface affiche les bases de données du serveur choisi.

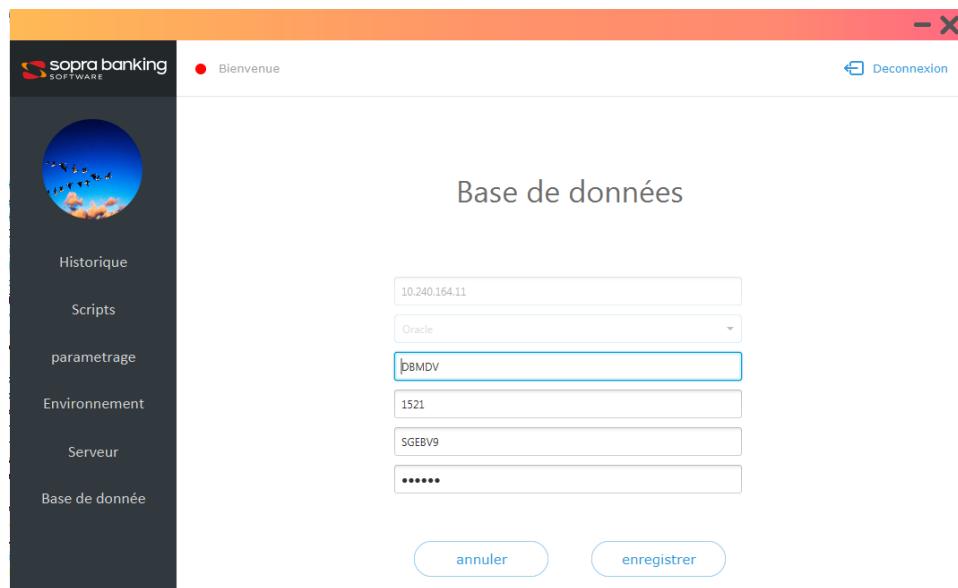


The screenshot shows the 'Les bases de données Valables' list screen. On the left, a sidebar lists navigation options: Historique, Scripts, paramétrage, Environnement, Serveur, and Base de données. The 'Base de données' option is selected. The main area is titled 'Les bases de données Valables'. It displays a table with one row: 'DBMDV'. To the right of the table are icons for 'supprimer' (delete) and 'modifier' (edit). At the bottom of the screen is a button labeled 'créer BD' (create BD).

Figure 71 : Interface d'affichage des bases de données.

- Modification d'une base de données :

Cette interface permet de modifier les informations d'une base de données.



The screenshot shows a software interface titled "sopra banking SOFTWARE". On the left, there's a sidebar with a blue circular icon containing a globe and the following menu items: Historique, Scripts, paramétrage, Environnement, Serveur, and Base de donnée. The main area is titled "Base de données" and contains several input fields:

- IP address: 10.240.164.11
- Database type: Oracle
- Database name: DBMDV (highlighted with a blue border)
- Port: 1521
- Service name: SGEBV9
- Password: masked as ****

At the bottom are two buttons: "annuler" (cancel) and "enregistrer" (register).

Figure 72 : Interface de modification d'une base de données.

- Suppression d'une base de données :

Si l'utilisateur clique sur le bouton de suppression un message de confirmation s'affiche

Les bases de données Valables



DBMDV

...



Figure 73 : Interface de suppression d'une base de données.

Pour valider la suppression on clique sur OK.

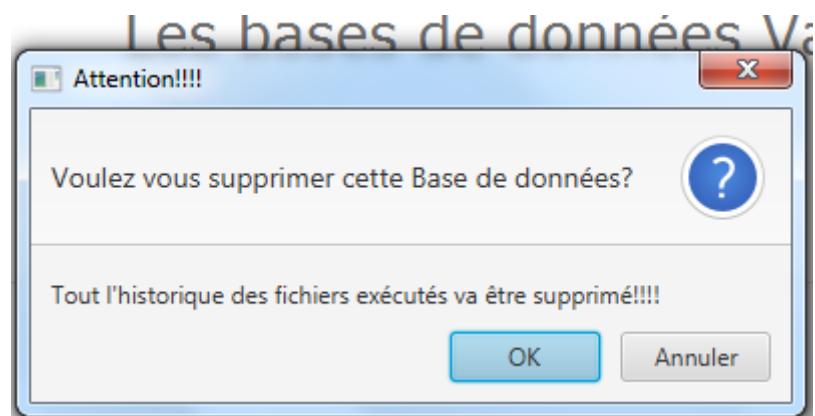


Figure 74 : Message de confirmation de suppression d'une base de données.

❖ Exécution des scripts :

Cette interface permet à l'utilisateur de gérer l'exécution des scripts.

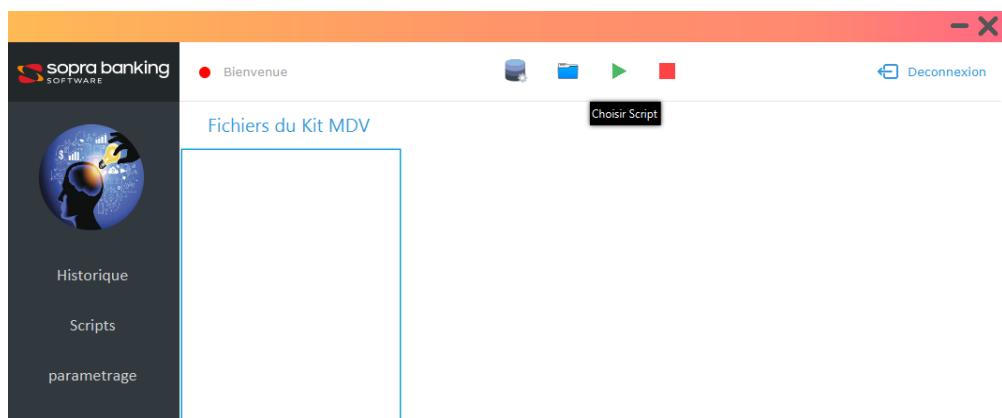


Figure 75 : Interface principale d'exécution des scripts

Pour exécuter un script, il faut d'abord choisir ce dernier et choisir la base de données dont on veut l'exécuter.

- Choisir la base de données :

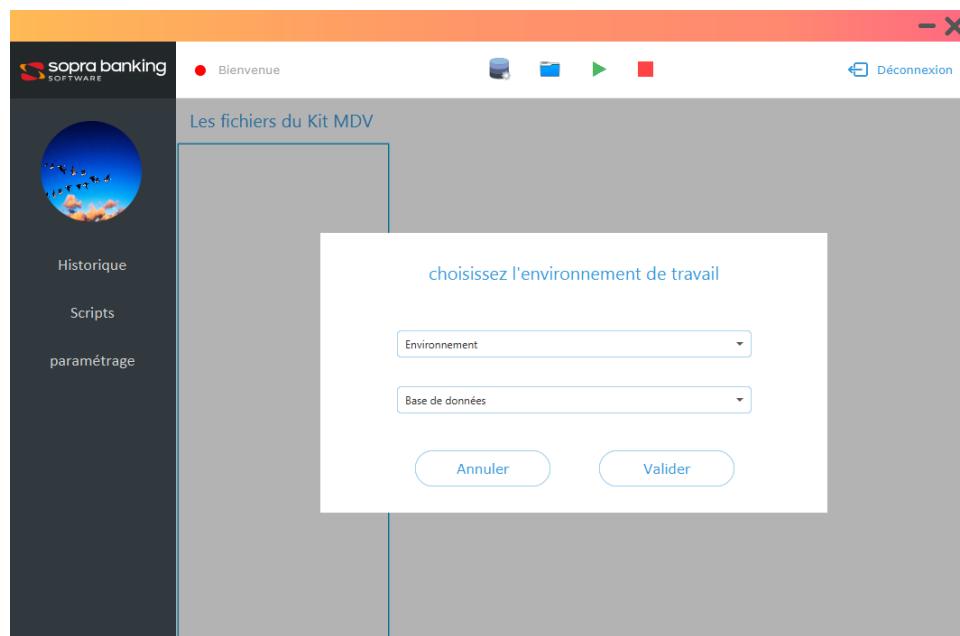


Figure 76 : Interface du choix de la base de données.

- Choisir le script :

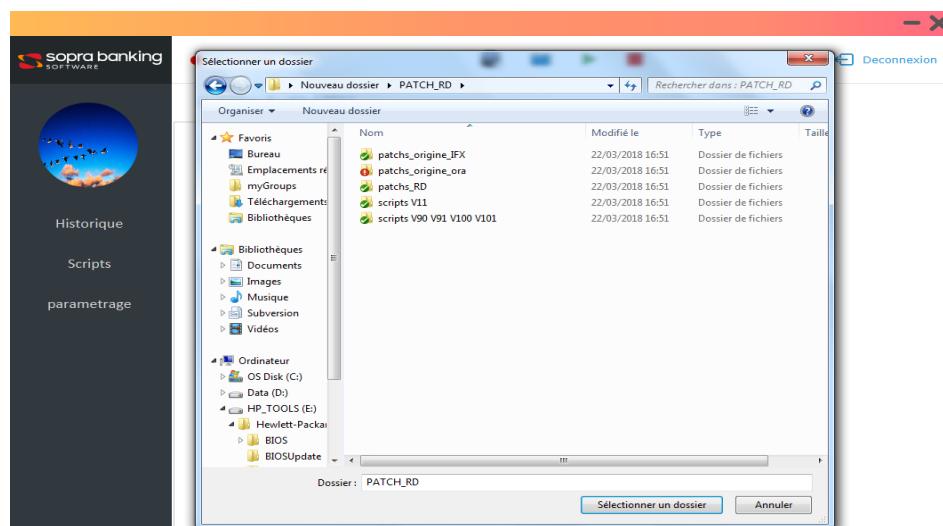


Figure 77 : interface du choix du script.

❖ Gestion des erreurs :

Après le lancement de l'exécution du script l'utilisateur est amené à corriger les erreurs SQL qui s'affiche. Cette figure présente à l'utilisateur plusieurs objets graphiques. Une zone de texte qui affiche la requête erronée et une autre zone contenant l'erreur SQL déclenchée avec les différents boutons pour effectuer une correction.

- Gestion des exceptions SQL :

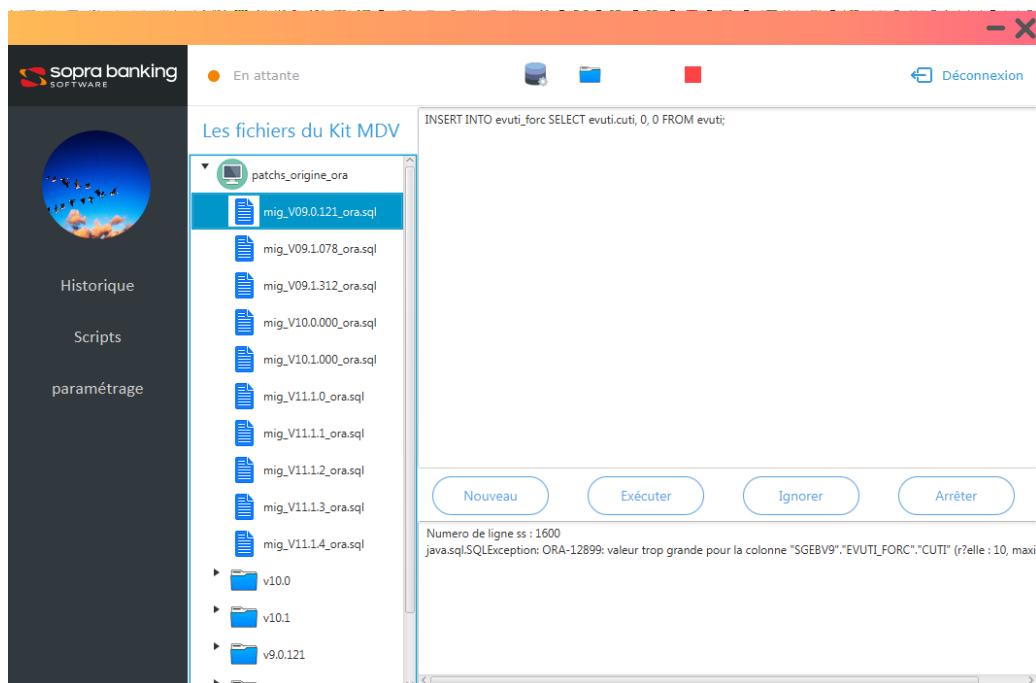


Figure 78 : Interface d'erreur lors de l'exécution.

Afin de corriger les erreurs liées à l'exécution des scripts, l'utilisateur a trois possibilités :

- Modifier le bloc SQL :

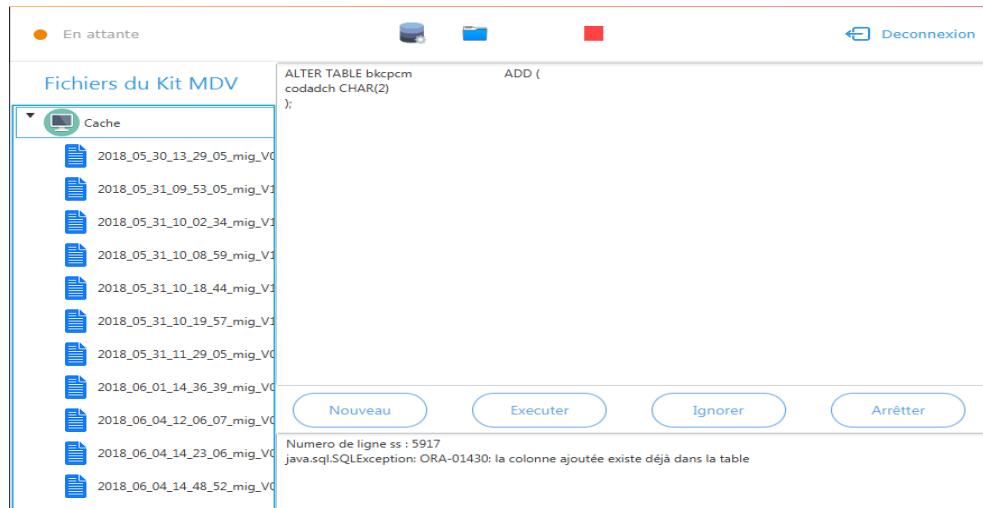


Figure 79 : interface pour changer une requête SQL.

Dans cette interface l'utilisateur peut changer la requête et exécuter cette dernière dans la base de données cible en cliquant sur le bouton « Exécuter ».

- Exécuter un nouveau bloc SQL :

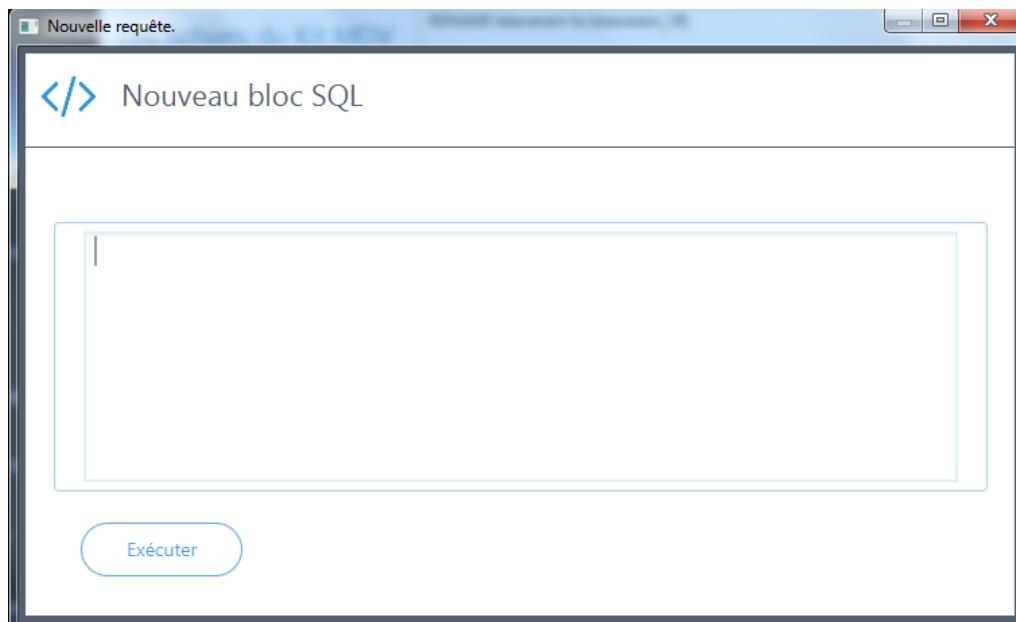


Figure 80 : interface d'exécution d'une nouvelle requête SQL

-
- Ignorer un bloc SQL :

Pour que l'application ignore une requête SQL l'utilisateur doit cliquer sur le bouton « ignorer ».

❖ **Contrôle de l'exécution :**



Figure 81 : interface de gestion de l'exécution.

Cette interface permet à l'utilisateur de gérer l'exécution du script :

- Le bouton en orange permet de mettre l'exécution en pose.
- Le bouton en rouge permet d'arrêter l'exécution.

Annexe B : Montée de la version

❖ Description de Kit de migration :

Le kit de migration qui va être installé chez le client est un répertoire qui contient tous les éléments nécessaires pour faire la montée de version Amplitude.

Les éléments du répertoire sont:

/appli/sopra/SGBFV_Upgrade/Migration_Kit	
Nom	
..	
flags	
logs	
progs	
Referentiel	
scripts	
working	
AmpSchema.body.sql	
ampschema.sql	
create_table_version_amplitude.sql	
infos_apres	
infos_avant	
lance_migration	
migration.sh	
migrationprocess.sh	
migrationprocess.sql	
migrationprocesslog.sql	
parametragement.sh	
premigration.sh	

Figure 82: répertoire de kit de migration

❖ Préparation et traitement des scripts de migration

Afin de mettre en forme les scripts R&D, et de les encapsuler pour permettre de gérer les erreurs, l'équipe migration lance les Outils tool1 et Tool2.

- **Tool1.pl**

Pour exécuter Tool1, il faut se placer dans le répertoire dans lequel le kit de migration a été installé et se mettre sur le dossier Tools et puis lancer l'exécution du script « Tool1.pl » par la commande suivante :

```
./Tool1.pl
```

Le script commence par faire le renommage des scripts puis lance le processus d'analyse.

Le renommage :

```
B001_mig_V9_1_001_ora.sql  
B002_mig_V9_1_002_ora.sql  
B003_mig_V9_1_003_ora.sql  
B004_mig_V9_1_004_ora.sql  
B005_mig_V9_1_005_ora.sql  
C001_mig_V10_0_001_ora.sql  
C002_mig_V10_0_002_ora.sql  
C003_mig_V10_0_003_ora.sql  
C004_mig_V10_0_004_ora.sql  
C005_mig_V10_0_005_ora.sql  
C006_mig_V10_0_006_ora.sql  
D001_mig_V10_1_001_ora.sql  
D002_mig_V10_1_002_ora.sql  
D003_mig_V10_1_003_ora.sql  
D004_mig_V10_1_004_ora.sql  
D005_mig_V10_1_005_ora.sql  
D006_mig_V10_1_006_ora.sql  
D007_mig_V10_1_007_ora.sql  
D008_mig_V10_1_008_ora.sql  
D009_mig_V10_1_009_ora.sql
```

Figure 83: Figure montre le renommage des scripts

Les scripts générés par Tool1.pl seront placés dans le répertoire « prepa_scripts »

- **Tool2.l**

L'étape suivante est de lancer le script Tool2.pl qui permet d'organiser les requêtes SQL en ajoutant avant et après chacune d'elle des procédures SQL permettant d'archiver les logs d'exécutions dans les tables de logs et d'exécuter les requêtes avec une bonne gestion des exceptions.

Le script Tool2.pl est exécuté de la façon suivante :

```
./Tool2.pl
```

```
[root@slnxtmaibfsbfvsgmdv03 ~]# su - oracle
[oracle@slnxtmaibfsbfvsgmdv03 ~]$ cd /appli/sopra/SGBFV_Upgrade/Tools/
[oracle@slnxtmaibfsbfvsgmdv03 Tools]$ ./Tool2.pl
DDL sup  rieur   32767 caract  res - Ex  cution du DDL sans passage par ampschema.executeSql| D002_mig_V10_1_002_ora.sql | STEP-1370 | CREATE OR : REPLACE
DDL sup  rieur   32767 caract  res - Ex  cution du DDL sans passage par ampschema.executeSql| D002_mig_V10_1_002_ora.sql | STEP-1371 | CREATE OR : REPLACE
DDL sup  rieur   32767 caract  res - Ex  cution du DDL sans passage par ampschema.executeSql| B003_mig_V9_1_003_ora.sql | STEP-29 | CREATE OR : REPLACE
DDL sup  rieur   32767 caract  res - Ex  cution du DDL sans passage par ampschema.executeSql| B003_mig_V9_1_003_ora.sql | STEP-35 | CREATE OR : REPLACE
DDL sup  rieur   32767 caract  res - Ex  cution du DDL sans passage par ampschema.executeSql| B003_mig_V9_1_003_ora.sql | STEP-190 | CREATE OR : REPLACE
DDL sup  rieur   32767 caract  res - Ex  cution du DDL sans passage par ampschema.executeSql| B003_mig_V9_1_003_ora.sql | STEP-196 | CREATE OR : REPLACE
DDL sup  rieur   32767 caract  res - Ex  cution du DDL sans passage par ampschema.executeSql| B002_mig_V9_1_002_ora.sql | STEP-3304 | CREATE OR : REPLACE
DDL sup  rieur   32767 caract  res - Ex  cution du DDL sans passage par ampschema.executeSql| B002_mig_V9_1_002_ora.sql | STEP-3310 | CREATE OR : REPLACE
```

Figure 84: Ex  cution de tool 2

Les scripts g  n  r  s par Tool2.pl seront plac  s dans le r  pertoire « Trans_Scripts ».

Il faut par la suite copier les scripts de Trans_Scripts dans le dossier « scripts » du kit du client.

❖ Lancement du kit de migration

Cette tape permet de lancer le KIT de migration, dont les diff  rentes phases sont pr  sent  es par des choix au niveau du menu principal du KIT.

Le lancement du KIT de migration se d  roule comme suit :



Se positionner au niveau du répertoire du KIT de migration /appli/sopra/amplitude/

```
cd /appli/sopra/amplitude/XXX_Migration
```

Exécuter la commande

```
./lance_migration.sh
```

On a l'affichage du menu ci-dessous :

```
- Montage de version UIB -  
  
le répertoire de travail est :  
/appli/amplitude/UIB_Migration/Migration_Kit  
  
-1- Collection des informations avant migration  
-2- Execution de la montée en structure de la base de données  
-3- Execution des programmes de reprise - HS  
-4- Execution de script de régularisation - HS  
-5- Chargement du référentiel - HS  
-6- Collection des informations après migration - HS  
  
Etape (R=Retour) :  
[ ]
```

Figure 85: menu de montée de version

Notre application s'intéresse particulièrement sur l'étape 2 : Exécution de la montée en structure de la base de données.

Cette étape permet de monter la structure de la base, en partant d'un schéma source, pour arriver au schéma de la version cible.

➤ Exécution de migrationprocess.sh

En choisissant l'étape 2, le script commence à préparer les prérequis d'installation en créant les tables « migrationprocess », « migrationprocesslog » et « VERSION_AMPLITUDE_9 »



ainsi que le package « ampchschema » et son body. Pour le cas d'informix, le package est sous forme d'un ensemble procédures à créer pour le bon déroulement du kit de migration.

➤ Exécution du script « pre-migration.sh »

Une fois terminée, il exécute le script « lance_migration.sh » exécute le script « premigration.sh », ce dernier permet de générer des fichiers « *.lst » comme résultat de la comparaison de la base source avec la base de référence source en se basant sur le script « schema_version.sh »

➤ Exécution du script « migration.sh »

Une fois terminée, il exécute le script « lance_migration.sh » exécute le script « migration.sh » qui exécute les scripts de la montée de version. Cette phase vise à effectuer la montée de version de la base de données source de référence vers la version cible. Dans notre cas, s'aligner sur la v10.1.

Dès qu'une requête est exécutée, elle est archivée dans les tables migrationprocess et migrationprocesslog.

```

Running migration sql script: "A001_migtables_spaces.sql"
Running migration sql script: "B001_mig_V9_1_001_ora.sql"
Running migration sql script: "B002_mig_V9_1_002_ora.sql"
Running migration sql script: "B003_mig_V9_1_003_ora.sql"
Running migration sql script: "B004_mig_V9_1_004_ora.sql"
Running migration sql script: "B005_mig_V9_1_005_ora.sql"
Running migration sql script: "C001_mig_V10_0_001_ora.sql"
Running migration sql script: "C002_mig_V10_0_002_ora.sql"
Running migration sql script: "C003_mig_V10_0_003_ora.sql"
Running migration sql script: "C004_mig_V10_0_004_ora.sql"
Running migration sql script: "C005_mig_V10_0_005_ora.sql"
Running migration sql script: "C006_mig_V10_0_006_ora.sql"
Running migration sql script: "D001_mig_V10_1_001_ora.sql"
Running migration sql script: "D002_mig_V10_1_002_ora.sql"
Running migration sql script: "D003_mig_V10_1_003_ora.sql"
Running migration sql script: "D004_mig_V10_1_004_ora.sql"
Running migration sql script: "D005_mig_V10_1_005_ora.sql"
Running migration sql script: "D006_mig_V10_1_006_ora.sql"
Running migration sql script: "D007_mig_V10_1_007_ora.sql"
Running migration sql script: "D008_mig_V10_1_008_ora.sql"
Running migration sql script: "D009_mig_V10_1_009_ora.sql"
10_1_119

Migration Complete - Amplitude User Applications are now available.

Ensure that statistics have been gathered on all tables, to enable
use of the Oracle Cost Based Optimizer.

The migration process may have recreated a number of tables,
which removes any optimizer statistics present before migration.
A script is supplied to gather statistics on any large recreated tables;
see schema/source/gatherstats_postmigration.sql.

The script U2_gather_stats.sql, run during migration, will have gathered
statistics on any small tables containing data.
New tables that have not been populated with data will not have statistics;
statistics should be gathered on these tables once they are populated.

ven. déc. 16 18:03:40 CET 2016 Fin de l'exécution de la montée en structure de la base de données
Presser une touche pour continuer ..

```

Figure 86 :Exécution des scripts



En cas d'incident, L'exécution s'arrête et génère un log d'erreur.

```
Running migration sql script: "B001_mig_V9_1_001_ora.sql"
Error messages detected in log file - aborting migration.
Refer to log file: logs/run001_B001_mig_V9_1_001_ora.log

Please investigate and correct problem before restarting.
Migration is incomplete - User Applications are NOT available.
[oracle@slnxtmaibfsbfvsgmdv03 Migration_Kit]$ vi logs/run001_B001_mig_V9_1_001_ora.log
```

Figure 87:Gestion des erreurs

Dans ce cas, il faut procéder comme suit :

- Le traitement des scripts s'arrête
- Affichage du log du script en erreur
- Correction du fichier sql concerné
- Relance du script « migration.sh »
- Le traitement redémarre automatiquement à partir de la dernière requête qui a causé l'erreur pour l'exécuter à nouveau.

Une fois le script “migration.sh” passe par l’ensemble des scripts de la migration et exécute l’ensemble des requêtes. Un message de fin s’affiche.

Message de fin:

```
Migration Complete - Amplitude User Applications are now available.

Ensure that statistics have been gathered on all tables, to enable
use of the Oracle Cost Based Optimizer.

The migration process may have recreated a number of tables,
which removes any optimizer statistics present before migration.
A script is supplied to gather statistics on any large recreated tables;
see schema/source/gatherstats_postmigration.sql.

The script U2_gather_stats.sql, run during migration, will have gathered
statistics on any small tables containing data.
New tables that have not been populated with data will not have statistics;
statistics should be gathered on these tables once they are populated.

ven. déc. 16 18:03:40 CET 2016 Fin de l'execution de la montée en structure de la base de donnees
Presser une touche pour continuer ..
```

Figure 88: Fin d'exécution



Webographie :

Webographie	Date de consultation
https://www.w3schools.com/	28/03/2018
https://openclassrooms.com	28/03/2018
http://www.developpez.com/	04/04/2018
https://docs.spring.io	25/04/2018
http://www.oracle.com	29/04/2018
https://mvnrepository.com	30/04/2018
https://www.flaticon.com/	25/05/2018
https://stackoverflow.com	26/05/2018
http://code.makery.ch/library/javafx-8-tutorial/fr/	30/05/2018

Tableau 18: Tableau de webographie