



Rapport de Stage de Fin d'Année

Ingénierie en Réseaux et Systèmes d'Information

Sujet

Conception et Développement d'une Application de Gestion des Abonnés E-Banking, leurs Contrats d'Abonnement et les Offres Commerciales

Effectuer à



De 03 Juillet au 31 Août 2023

Réalisé par :

Ayoub TALBI

Encadré par :

Pr. Sara QASSIMI

Mme. Noura AIT EL HOUSSE

Année Universitaire : 2022-2023

Remerciement

J'ai le plaisir de garder cette page en signe de gratitude et de profonde reconnaissance à tous ceux qui m'ont aidé de près et de loin pour réaliser ce projet.

En premier lieu, je remercie **Mme Noura AIT EL HOUSSE**, mon encadrante professionnelle au sein de Adria Business & Technologies et mon encadrante académique **Pr. Sara Qassimi** pour leurs disponibilités, leurs judicieux conseils, leurs efforts et leurs encouragements.

Je tiens aussi à remercier la société qui m'a accueilli pendant toute la durée du stage **Adria Business & Technologies** pour cette opportunité de stage.

Résumé

Ce rapport présente le travail effectué lors de mon stage de fin d'année portant sur la conception et le développement d'une application de la gestion des abonnés E-Banking, leurs contrats d'abonnement ainsi que les offres commerciales. Le principal objectif de ce stage était de développer une application qui permet à un agent back-office au sein d'une banque à gérer les abonnés de son agence, et leurs contrats d'abonnement.

Le projet a été développé en utilisant le framework Java, Spring Boot en backend, et React en frontend, et donc le résultat final est une application web qui regroupe des fonctionnalités CRUD, ainsi que les opérations de recherches avancées avec des filtres.

Mots clés : Agent Back-Office, Gestion des abonnés, Gestion des contrats d'abonnement, Gestion des offres commerciales, Java, Spring Boot, React.

Abstract

This report presents the work carried out during my end-of-year internship focusing on the design and development of an application for the management of E-Banking subscribers, their subscription contracts, and commercial offers. The main objective of this internship was to develop an application that allows a back-office agent within a bank to manage subscribers of their branch and their subscription contracts.

The project was developed using the Java framework, Spring Boot for the backend, and React for the frontend. As a result, the final product is a web application that integrates CRUD (Create, Read, Update, Delete) functionalities, along with advanced search operations incorporating filters.

Keywords : Back-Office Agent, Subscribers Management, Subscription Contracts Management, Commercial Offers Management, Java, Spring Boot, React.

Table des matières

Table des figures	9
Liste des tableaux.....	11
Liste des abréviations	12
Introduction Générale	13
Chapitre 1	14
Contexte général du projet	14
1.1 Organisme d'accueil.....	15
1.1.1 Entreprise	15
1.1.2 Fiche technique de la société.....	15
1.1.3 Produits et services	16
1.1.4 Principales références	17
1.1.5 Présence internationale	17
1.2 Présentation du projet.....	18
1.2.1 Description du projet	18
1.2.2 Contexte du projet	18
1.2.3 Mes missions	19
1.3 Déroulement du projet.....	19
1.4 Conclusion	20
Chapitre 2	21
Analyse et conception du projet	21
2.1 Capture des besoins	22
2.1.1 Identification des acteurs du système	22
2.1.2 Besoins fonctionnels	22
2.1.3 Besoins non fonctionnels	23
2.2 Analyse des besoins.....	23
2.2.1 Diagramme de cas d'utilisation globale.....	24

2.2.2	Diagramme de cas d'utilisation pour la gestion des contrats	24
2.2.3	Diagramme de cas d'utilisation pour la gestion des abonnés.....	25
2.2.4	Diagramme de cas d'utilisation pour la gestion des offres	25
2.3	Diagrammes de séquence	26
2.3.1	Gestion des contrats d'abonnement	26
2.3.1.1	Diagramme de séquence pour l'ajout d'un contrat.....	26
2.3.1.2	Diagramme de séquence pour la modification d'un contrat.....	27
2.3.1.3	Diagramme de séquence pour la suppression d'un contrat	27
2.3.1.4	Diagramme de séquence pour la recherche d'un contrat.....	28
2.3.1.5	Diagramme de séquence pour la consultation des contrats	29
2.3.2	Gestion des abonnés	30
2.3.2.1	Diagramme de séquence pour l'ajout d'un abonné.....	30
2.3.2.2	Diagramme de séquence pour la modification d'un abonné	30
2.3.2.3	Diagramme de séquence pour la suppression d'un abonné	31
2.3.2.4	Diagramme de séquence pour la recherche d'un abonné	32
2.3.2.5	Diagramme de séquence pour la consultation des abonnés.....	33
2.3.3	Gestion des offres commerciales.....	34
2.3.3.1	Diagramme de séquence pour l'ajout d'une offre.....	34
2.3.3.2	Diagramme de séquence pour la modification d'une offre	34
2.3.3.3	Diagramme de séquence pour la suppression d'une offre	35
2.3.3.4	Diagramme de séquence pour la recherche d'une offre	35
2.3.3.5	Diagramme de séquence pour la consultation des offres	36
2.4	Diagrammes de classe	37
2.5	Conclusion	39
Chapitre 3	40
Etude Technique du projet	40
3.1	Introduction	41
3.2	Outils et environnement de travail	42
3.2.1	Outils de gestion de version	42
3.2.2	IDEs et éditeurs de text.....	42
3.3	Présentation de la partie Frontend du projet	44
3.3.1	Architecture logiciel de la partie Frontend	44
3.3.2	Langages et technologies	45

3.4 Présentation de la partie Backend du projet	46
3.4.1 Design Patterns.....	46
3.4.1.1 DAO (Data Access Object).....	47
3.4.1.2 IOC (Inversion of Control).....	47
3.4.1.3 DTO (Data Transfer Object)	48
3.4.2 Architecture logiciel de la partie Backend.....	49
3.4.3 Langages et technologies.....	50
3.4.4 Tests de la partie backend.....	52
3.4.4.1 Tests fonctionnels.....	52
3.4.4.2 Tests unitaires.....	55
3.5 Conclusion.....	56
Chapitre 4	57
Mise en œuvre	57
4.1 Page d'accueil.....	58
4.2 Page d'erreur.....	59
4.3 Gestion des abonnés.....	59
4.3.1 Ajout d'un abonné	59
4.3.2 Lister des abonnés	60
4.3.2.1 Lister tous les abonnés	60
4.3.2.2 Chercher des abonnés en utilisant la barre de recherche.....	61
4.3.3 Modifier les données d'un abonné	62
4.3.4 Suppression d'un abonné	62
4.4 Gestion des contrats d'abonnement	63
4.4.1 Ajout d'un contrat d'abonnement.....	63
4.4.2 Lister tous les contrats	63
4.4.2.1 Lister tous les contrats	63
4.4.2.2 Chercher les contrats en utilisant la barre de recherche	64
4.4.3 Modifier les données d'un contrat.....	65
4.4.4 Suppression d'un contrat	65
4.5 Gestion des offres commerciales	66
4.5.1 Ajout d'une offre commerciale	66
4.5.2 Lister les offres commerciales	66
4.5.2.1 Lister tous les offres.....	66

4.5.2.2 Naviguer entre les pages des offres	67
4.5.2.3 Chercher des offres en utilisant la barre de recherche	68
4.5.3 Modifier les données d'une offre	68
4.5.4 Suppression d'une offre.....	69
4.6 Conclusion.....	69
Conclusion Générale.....	70

Table des figures

Figure 1 : Produits et services Adria.....	16
Figure 2 : Principales références d'Adria.....	17
Figure 3 : La carte de la présence internationale d'Adria	17
Figure 4 : Drapeaux des pays illustrant la présence internationale d'Adria.....	17
Figure 5 : Diagramme de Gant.....	19
Figure 6 : Diagramme de cas d'utilisation globale.....	24
Figure 7 : Diagramme de cas d'utilisation pour la gestion des contrats	24
Figure 8 : Diagramme de cas d'utilisation pour la gestion des abonnés	25
Figure 9 : Diagramme de cas d'utilisation pour la gestion des offres.....	25
Figure 10 : Diagramme de séquence pour l'ajout d'un contrat	26
Figure 11 : Diagramme de séquence pour la modification d'un contrat.....	27
Figure 12 : Diagramme de séquence pour la suppression d'un contrat	27
Figure 13 : Diagramme de séquence pour la recherche d'un contrat.....	28
Figure 14 : Diagramme de séquence pour la consultation des contrats	29
Figure 15 : Diagramme de séquence pour l'ajout d'un abonné.....	30
Figure 16 : Diagramme de séquence pour la modification d'un abonné	30
Figure 17 : Diagramme de séquence pour la suppression d'un abonné	31
Figure 18 : Diagramme de séquence pour la recherche d'un abonné	32
Figure 19 : Diagramme de séquence pour la consultation des abonnés.....	33
Figure 20 : Diagramme de séquence pour l'ajout d'une offre.....	34
Figure 21 : Diagramme de séquence pour la modification d'une offre	34
Figure 22 : Diagramme de séquence pour la suppression d'une offre	35
Figure 23 : Diagramme de séquence pour la recherche d'une offre	35
Figure 24 : Diagramme de séquence pour la consultation des offres	36
Figure 25 : Diagramme de classe pour les entités du projet.....	37
Figure 26 : Diagramme de classe des trois couches repository, service et controller.....	38
Figure 27 : Schéma de l'aperçu global du projet	41
Figure 28 : Logo de Git.....	42
Figure 29 : Logo de GitHub	42
Figure 30 : Logo de IntelliJ IDEA	43
Figure 31 : Logo de VSCode.....	43
Figure 32 : La Racine de la partie front end.....	44
Figure 33 : Logo de JavaScript.....	45
Figure 34 : Logo de ReactJs	45
Figure 35 : Logo de Babel.....	46
Figure 36 : Logo de Bootstrap.....	46
Figure 37 : schéma illustrant le fonctionnement de DAO Design pattern	47
Figure 38 : Schéma illustrant le fonctionnement de IOC Container.....	48
Figure 39 : Racine de la partie backend.....	49

Figure 40 : Logo de Java.....	50
Figure 41 : Logo de Spring Boot.....	50
Figure 42 : Logo de Framework Hibernate.....	51
Figure 43 : Logo de MySQL.....	51
Figure 44 : Logo de Postman	52
Figure 45 : test de la méthode GET de l'api de gestion des abonnés	53
Figure 46 : test de la méthode POST de l'api de gestion des abonnés.....	53
Figure 47 : test de la méthode PUT de l'api de gestion des abonnés.....	54
Figure 48 : test de la méthode PUT pour changer le statut d'un abonné	54
Figure 49 : test de la méthode PUT pour associer une agence à un abonné.....	54
Figure 50 : test de la méthode DELETE pour supprimer tous les abonnés.....	54
Figure 51 : test de la méthode DELETE pour supprimer un abonné.....	54
Figure 52 : Logo de JUnit 4	55
Figure 53 : Logo de Mockito.....	55
Figure 54 : Page d'accueil	58
Figure 55 : Page d'erreur.....	59
<i>Figure 56 : Ajout d'un abonné</i>	59
Figure 57 : Lister tous les abonnés	60
Figure 58 : Chercher des abonnés en utilisant la barre de recherche	61
Figure 59 : Chercher des abonnés par email	61
Figure 60 : Modifier les données d'un abonné	62
Figure 61 : Suppression d'un abonné	62
Figure 62 : Ajout d'un contrat d'abonnement	63
Figure 63 : Lister tous les contrats.....	64
Figure 64 : Chercher les contrats en utilisant la barre de recherche	64
Figure 65 : Modifier les données d'un contrat	65
<i>Figure 66 : Suppression d'un contrat</i>	65
Figure 67 : Ajout d'une offre commerciale.....	66
Figure 68 : Lister tous les offres.....	67
Figure 69 : Naviguer entre les pages des offres.....	67
Figure 70 : Chercher des offres en utilisant la barre de recherche	68
Figure 71 : Modifier les données d'une offre.....	68
Figure 72 : Suppression d'une offre	69

Liste des tableaux

Tableau 1 : Fiche technique d'Adria B&T	15
Tableau 2 : Description des entités	37
Tableau 3 : Description des classes et interfaces du projet	39
Tableau 4 : Résumé de la couverture globale des tests unitaires.....	56

Liste des abréviations

Acronyme	Signification
ACID	Atomicité, Cohérence, Isolation, Durabilité
API	Application Programming Interface
BD	Base de données
CRUD	Create, Read, Update, Delete
DAO	Data Access Object
DOM	Document Object Model
DTO	Data Transfer Object
HTTP	HyperText Transfer Protocol
IDE	Integrated Development Environment
IoC	Inversion of Control
JPA	Java Persistence API
ORM	Object Relational Mapper
REST	Representational State Transfer
SQL	Structured Query Language
UML	Unified Modeling Language

Introduction Générale

L'adoption généralisée des services bancaires en ligne et mobiles connaît une croissance rapide, principalement attribuable à la pandémie de COVID-19, et cette tendance devrait perdurer au-delà de la fin de la crise sanitaire. Pour les institutions financières telles que les banques et les coopératives de crédit, cette évolution représente une opportunité significative d'exploiter les avantages offerts par la banque numérique. Elle leur permet également d'offrir à leurs clients la flexibilité d'effectuer des opérations bancaires, des transactions et des transferts d'argent selon leurs préférences.

Dans ce cadre, **Adria Business & Technology** propose à sa clientèle une diversité de fonctionnalités à travers ses applications web et mobiles. Dans le but de maintenir la dynamique technologique de ses logiciels, la société s'efforce de concevoir des solutions novatrices. Ce faisant, elle vise non seulement à consolider ses caractéristiques distinctives, mais également à construire une identité propre.

L'amélioration des solutions logicielles constitue l'un des objectifs primordiaux d'Adria. Ainsi, mon projet de stage se focalise sur la concrétisation de nouvelles idées visant à apporter une valeur ajoutée et à optimiser le produit.

Ce rapport se structure en quatre chapitres, au sein desquels je détaillerai les diverses tâches accomplies en vue d'atteindre l'objectif du projet :

- ❖ **Contexte général du projet**, dans lequel je vais présenter l'entreprise d'accueil et le déroulement général du projet.
- ❖ **Analyse et conception du projet**, où je présent une étude conceptuelle du projet.
- ❖ **Etude technique du projet**, où je parle des outils techniques utilisés pour le développement du projet.
- ❖ **Mise en œuvre**, dans lequel je présente des captures d'écran des interfaces graphiques accompagnées d'explications.

Chapitre 1

Contexte général du projet

Résumé

Ce chapitre vise à fournir une vue d'ensemble de la situation actuelle qui a motivé la réalisation de ce projet. Il présentera d'abord l'organisme d'accueil, ses activités, son organisation et ses clients, puis il décrira le cadre du projet, les contraintes identifiées et les objectifs poursuivis.

1.1 Organisme d'accueil

1.1.1 Entreprise



Adria Business & Technology est un expert dans l'édition et l'intégration des logiciels destinés aux banques et institutions financières. Sa mission est d'accompagner les organisations financières dans leurs projets de Digital Banking en leur offrant des solutions performantes et des services à haute valeur ajoutée.

Avec une profonde compréhension du secteur bancaire, Adria crée des produits bien conçus pour aider les banques à établir leur stratégie numérique au sein d'une structure de coûts définie et adaptée à chaque cas d'utilisation.

1.1.2 Fiche technique de la société

Raison sociale	ADRIA BUSINESS & TECHNOLOGY
Directeur général	Rachid BEKKAR
Chiffre d'affaires	1 000 000 DHs
Forme juridique	Société à Responsabilité Limitée à Associé Unique (SARLAU)
Secteur d'activité	Informatique, Digital Banking
Siege social	Shore 26 – 2e étage Casa Nearshore Park Casablanca, Maroc
Téléphone	+212 (5) 222-73-0-73
E-mail	contact@adria-bt.com
Site Web	www.adria-bt.com

Tableau 1 : Fiche technique d'Adria B&T

1.1.3 Produits et services

Adria a pour principal objectif d'assurer la qualité et l'adéquation des produits et services qu'elle offre à ses clients, afin de satisfaire amplement les besoins exprimés ou potentiels des consommateurs. Elle expose une variété de produits et services couvrant tous types de canaux de communication, à savoir le web et le mobile.

- ❖ **Adria banque directe Cross Canal :** Il s'agit d'un produit modulaire qui se positionne sur l'ensemble de la trajectoire commerciale et présente un espace en ligne intégré de la relation avec le client disposant ainsi de fonctionnalités de marketing, de vente dématérialisée et complètement en ligne, de services transactionnels, et de communication avec le client.
- ❖ **Adria Mobile Branch Banking :** Elle permet d'équiper les agences mobiles et les agents nomades d'une solution qui couvre toutes les fonctionnalités d'entrée en relation, de souscription de produits, de gestion de comptes et des services transactionnels...
- ❖ **Adria Analytics :** Il permet de collecter, d'analyser et visualiser l'ensemble des interactions du client avec la banque quel que soit le canal utilisé.

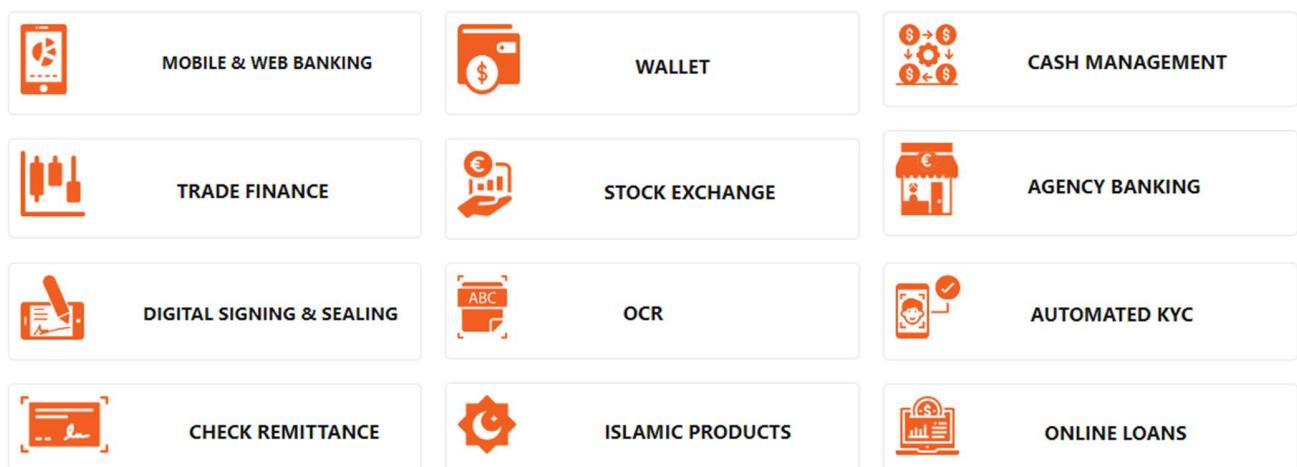


Figure 1 : Produits et services Adria

1.1.4 Principales références

Grâce à son potentiel inestimable, la société ADRIA a pu monopoliser 90% du marché bancaire au Maroc. La figure ci-dessous illustre les principales références d'Adria B&T :



Figure 2 : Principales références d'Adria

1.1.5 Présence internationale

En raison de ses services de qualités, Adria a pu créer une base clientèle qui regroupe toutes les grandes sociétés bancaires soit au Maroc ou à l'étranger. Le potentiel d'Adria l'a transformé d'une entreprise locale à une société visant l'internationale.

ADRIA a pu acquérir la confiance de clients à l'international. Ainsi, elle a pu rapprocher ses services au niveau Africain et Européen.

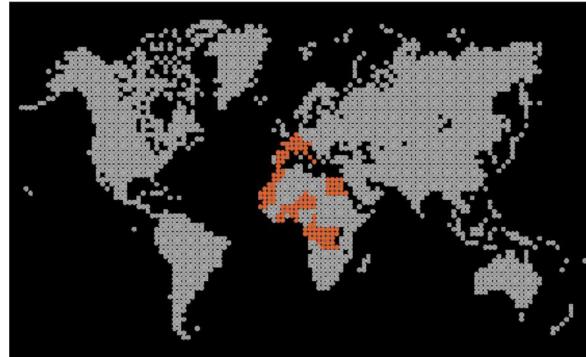


Figure 3 : La carte de la présence internationale d'Adria



Figure 4 : Drapeaux des pays illustrant la présence internationale d'Adria

1.2 Présentation du projet

Cette section vise à situer le contexte général du projet en décrivant la problématique, les objectifs visés et mes missions dans le projet.

1.2.1 Description du projet

Le cœur du projet réside dans la création d'une application conçue spécifiquement pour rationaliser la gestion des abonnés E-Banking au sein de l'institution bancaire. En mettant l'accent sur l'efficacité opérationnelle, l'application vise à équiper les agents back-office avec des fonctionnalités complètes de gestion. Les abonnés E-Banking, avec leurs contrats d'abonnement et les offres commerciales associées, sont au centre de l'attention. L'application permettra aux utilisateurs d'effectuer des opérations CRUD, englobant la création, la lecture, la mise à jour et la suppression, sur ces entités cruciales. Elle offrira également des fonctionnalités de recherche avancées par champ, permettant aux agents de naviguer et de filtrer les données de manière précise. Cette approche modulaire et complète vise à optimiser les processus internes de gestion, garantissant ainsi une expérience utilisateur fluide et une administration efficace des informations relatives aux abonnés E-Banking.

1.2.2 Contexte du projet

Au sein du secteur bancaire en constante évolution, l'impératif de rationaliser les processus de gestion des abonnés E-Banking devient de plus en plus prépondérant. La croissance exponentielle du nombre d'abonnés et la diversification des offres commerciales ont introduit une complexité significative dans la gestion des contrats d'abonnement. Face à cette réalité, il est devenu essentiel de mettre en place une solution informatique robuste capable d'assurer une gestion efficace et précise de ces entités interconnectées. L'application en développement s'inscrit comme une réponse stratégique à cette nécessité pressante en offrant un outil convivial et fonctionnel destiné aux agents back-office.

La complexité des contrats d'abonnement et des offres commerciales, avec leurs multiples relations, crée des défis uniques qui ne peuvent être surmontés que par une solution technologique avancée. L'objectif est d'autoriser les agents back-office à accéder de manière transparente, à modifier dynamiquement et à suivre de manière proactive les informations relatives aux abonnés et aux offres commerciales. Cette initiative de développement n'est pas simplement une réponse à une exigence technique, mais plutôt une démarche stratégique visant à optimiser les processus internes de l'institution bancaire. En contribuant à la simplification des opérations, cette démarche améliorera sensiblement l'expérience globale des abonnés E-Banking tout en offrant aux agents back-office un outil puissant pour relever les défis complexes de la gestion des abonnements.

Cette initiative s'inscrit dans une dynamique globale d'amélioration des services bancaires, mettant en avant la volonté de l'institution de demeurer à la pointe de l'innovation. En alignant les processus internes sur les normes technologiques les plus récentes, l'application en développement se positionne comme un levier stratégique pour la compétitivité de l'institution bancaire. Elle vise à introduire des niveaux inédits

d'efficacité opérationnelle, contribuant ainsi à une meilleure gestion des ressources tout en offrant aux abonnés E-Banking une expérience client optimale.

1.2.3 Mes missions

En tant que développeur principal sur ce projet, mes missions ont été diverses et couvraient l'ensemble du cycle de développement. La première étape consistait en la conception du service, impliquant la définition des entités du système, des relations entre elles, ainsi que l'élaboration d'une architecture logicielle cohérente. Ensuite, j'ai développé les API nécessaires pour les opérations CRUD et les filtres, en mettant l'accent sur la création des points de terminaison clairs et fonctionnels.

Au niveau du développement frontend, j'ai choisi le framework React pour créer une interface utilisateur moderne et réactive. Les composants ont été conçus pour refléter les entités du backend, assurant une intégration harmonieuse entre les deux parties. L'intégration backend-frontend a été effectuée avec soin pour garantir une communication efficace entre les deux couches de l'application.

En parallèle, j'ai élaboré des tests unitaires approfondis pour la couche service du backend, assurant ainsi la fiabilité et la stabilité de l'application.

Dans l'ensemble, mes missions ont été axées sur la création d'une solution complète et fonctionnelle, répondant de manière optimale aux besoins spécifiques de gestion des abonnés E-Banking dans le cadre bancaire.

1.3 Déroulement du projet

Le projet s'est déroulé selon le diagramme de Gantt ci-dessous :

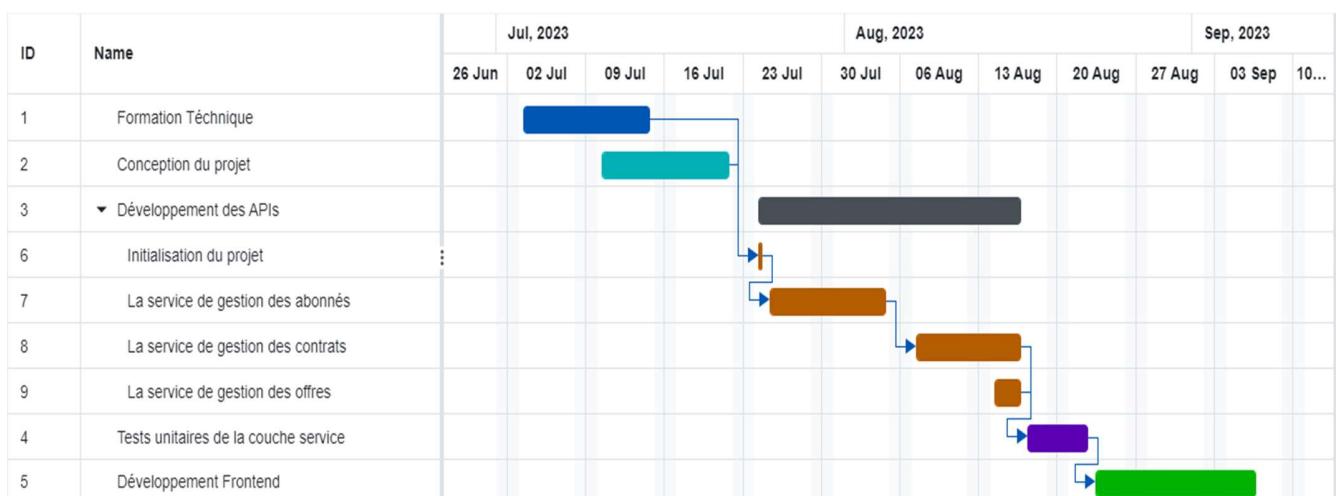


Figure 5 : Diagramme de Gantt

1.4 Conclusion

Le présent chapitre avait comme objectif de contextualiser le projet, en présentant d'une part la société **ADRIA B&T** en termes d'activités et d'organisation, ainsi que les concepts clés du sujet proposé et d'autre part, l'enchaînement et le déroulement du projet. La partie suivante sera réservée pour la branche fonctionnelle et conceptuelle.

Chapitre 2

Analyse et conception du projet

Résumé

La réussite d'un projet dépend en grande partie de la qualité de son initiation. Conséquemment, l'étape de spécification et l'analyse des besoins constituent le piédestal d'instruction de ce projet. Ce chapitre sera donc dédié à l'analyse des besoins fonctionnels du projet et la modélisation des différents aspects de la solution sous forme de diagrammes UML.

2.1 Capture des besoins

2.1.1 Identification des acteurs du système

Un acteur est une personne, un matériel ou un logiciel qui interagit avec le système. L'analyse du présent projet commence par une identification des acteurs agissants sur les différentes parties du système.

L'application à réaliser comprend un seul acteur :

- ❖ **Back office** : Employé d'une agence bancaire.

2.1.2 Besoins fonctionnels

Le cahier de charges reçu de la part de mon encadrante professionnelle contient les spécifications fonctionnelles suivantes :

- ❖ **Gestion des abonnés :**

- Création d'un nouvel abonné.
- Affichage de la liste des abonnés.
- Modification des données d'un abonné.
- Suppression d'un abonné.
- Chercher un ou plusieurs abonnés par champ (nom, prénom, email...).
- Filtrer les abonnés par champ.
- Associer un contrat à un abonné.

- ❖ **Gestion des contrats d'abonnement :**

- Création d'un contrat d'abonnement.
- Affichage de la liste des contrats d'abonnement.
- Modification d'un contrat.
- Suppression d'un contrat.
- Chercher un ou plusieurs contrats par champ (intitulé, statut...).
- Filtrer les contrats par champ.
- Associer un abonné à un contrat.

- ❖ **Gestion des offres commerciales :**

- Création d'une offre commerciale.
- Affichage de la liste des offres commerciales.
- Modification d'une offre.
- Suppression d'une offre.
- Chercher des offres par champ (libellé, description).

2.1.3 Besoins non fonctionnels

Les besoins ou les exigences non fonctionnelles sont les besoins qui caractérisent le système. Autrement dit, c'est un ensemble de critères essentiels pour le bon fonctionnement de notre application. Nous pouvons citer par exemple :

- ❖ **La simplification** : Gestion efficace des abonnés.
- ❖ **Ergonomie** : L'interface de l'application doit être simple et intuitive afin que l'utilisateur puisse l'exploiter sans se référer à des connaissances particulières, en d'autres termes, notre application doit être facile à manipuler par n'importe quel utilisateur (back office).
- ❖ **L'extensibilité** : L'application doit avoir la possibilité d'ajouter de nouvelles fonctionnalités ou de modifier celles existantes.

2.2 Analyse des besoins

Cette partie traite l'identification des acteurs interagissant avec le système et son organisation fonctionnelle. Les interactions seront présentées sous forme de cas d'utilisation de notre application. Durant Cette phase recours au formalisme UML étant le plus adéquat pour la modélisation d'une application modulaire et facilement extensible.

Les diagrammes de cas d'utilisation sont des diagrammes UML utilisés pour donner une vision globale du comportement fonctionnel d'un système logiciel. Ils sont utiles pour des présentations auprès de la direction ou des acteurs d'un projet.

Les cas d'utilisations et les acteurs de notre système sont schématisés dans les diagrammes ci-dessous :

2.2.1 Diagramme de cas d'utilisation globale

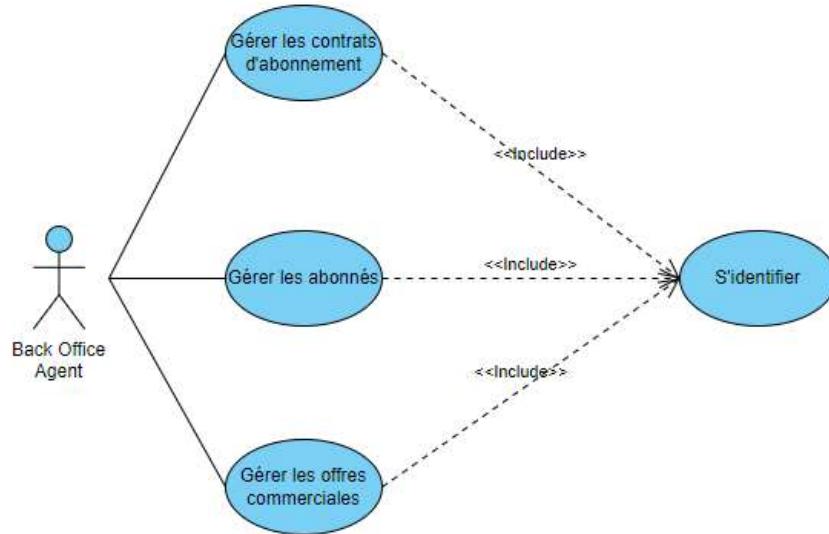


Figure 6 : Diagramme de cas d'utilisation globale

2.2.2 Diagramme de cas d'utilisation pour la gestion des contrats

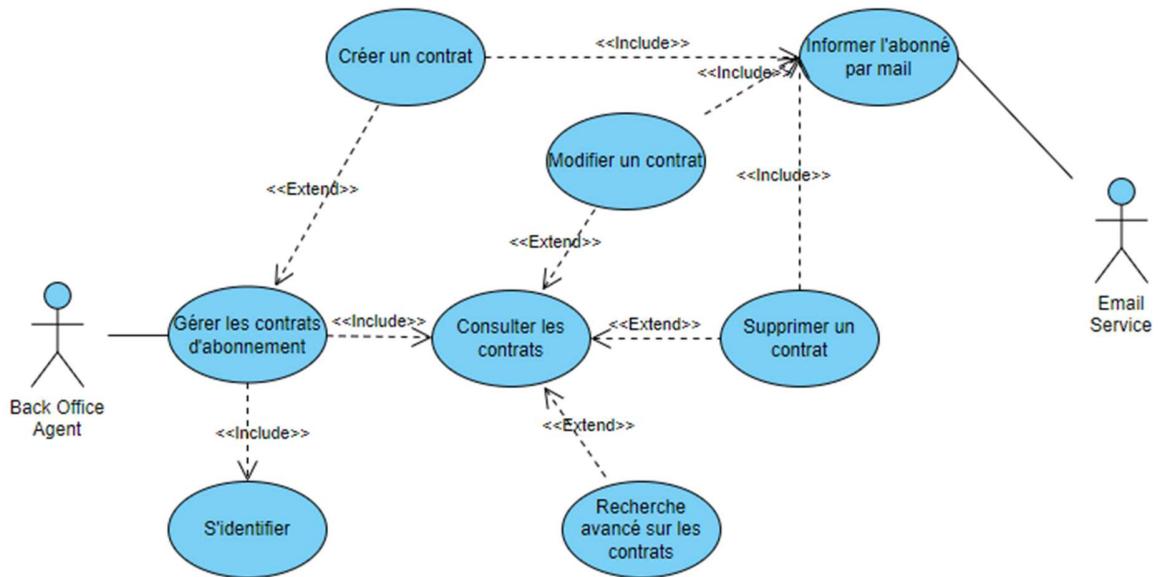


Figure 7 : Diagramme de cas d'utilisation pour la gestion des contrats

2.2.3 Diagramme de cas d'utilisation pour la gestion des abonnés

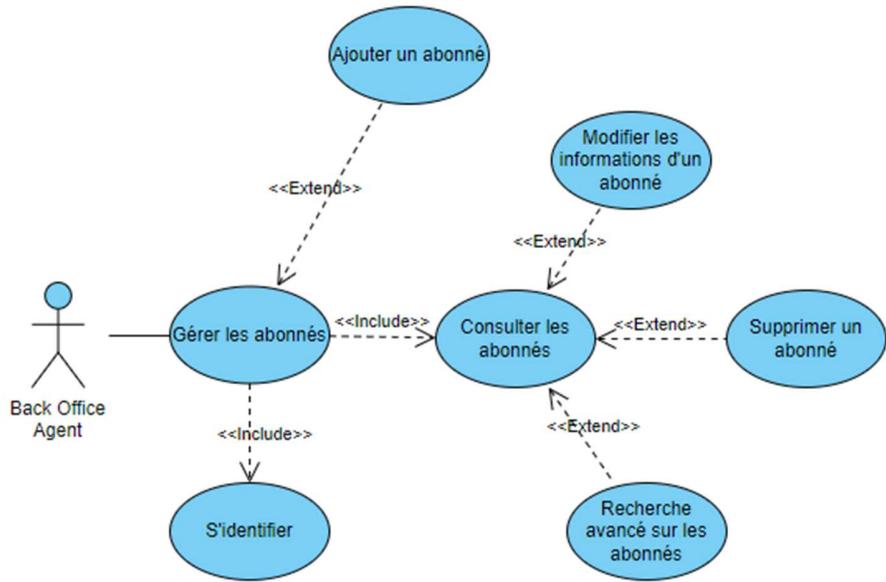


Figure 8 : Diagramme de cas d'utilisation pour la gestion des abonnés

2.2.4 Diagramme de cas d'utilisation pour la gestion des offres

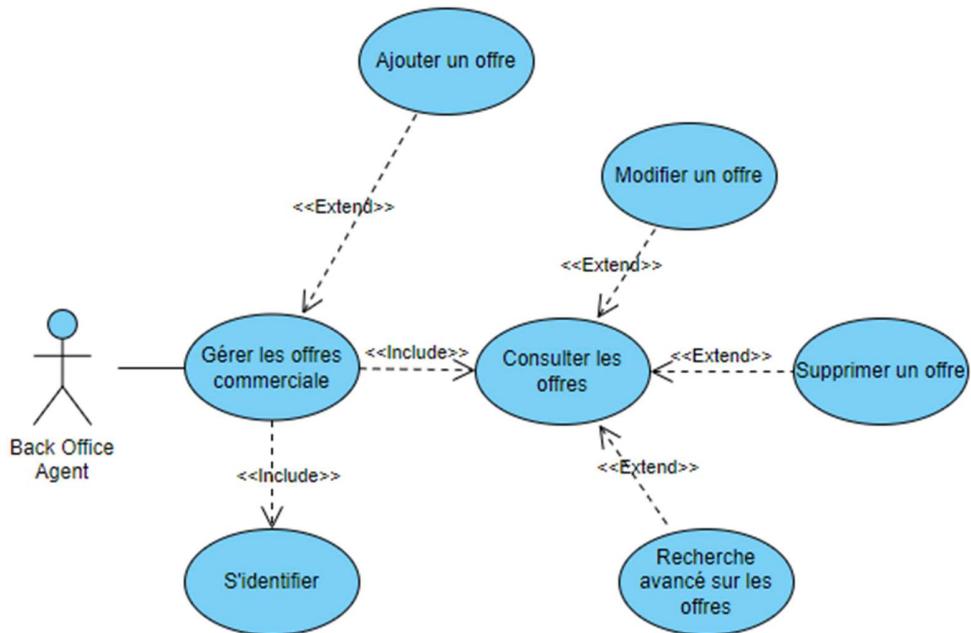


Figure 9 : Diagramme de cas d'utilisation pour la gestion des offres

2.3 Diagrammes de séquence

La description textuelle présente des inconvénients puisqu'il est difficile de montrer comment les traitements se succèdent. Il est donc recommandé de compléter la description textuelle par un ou plusieurs diagrammes dynamiques UML. Dans ce qui suit je vais présenter les principaux diagrammes de séquences élaborés.

2.3.1 Gestion des contrats d'abonnement

2.3.1.1 Diagramme de séquence pour l'ajout d'un contrat

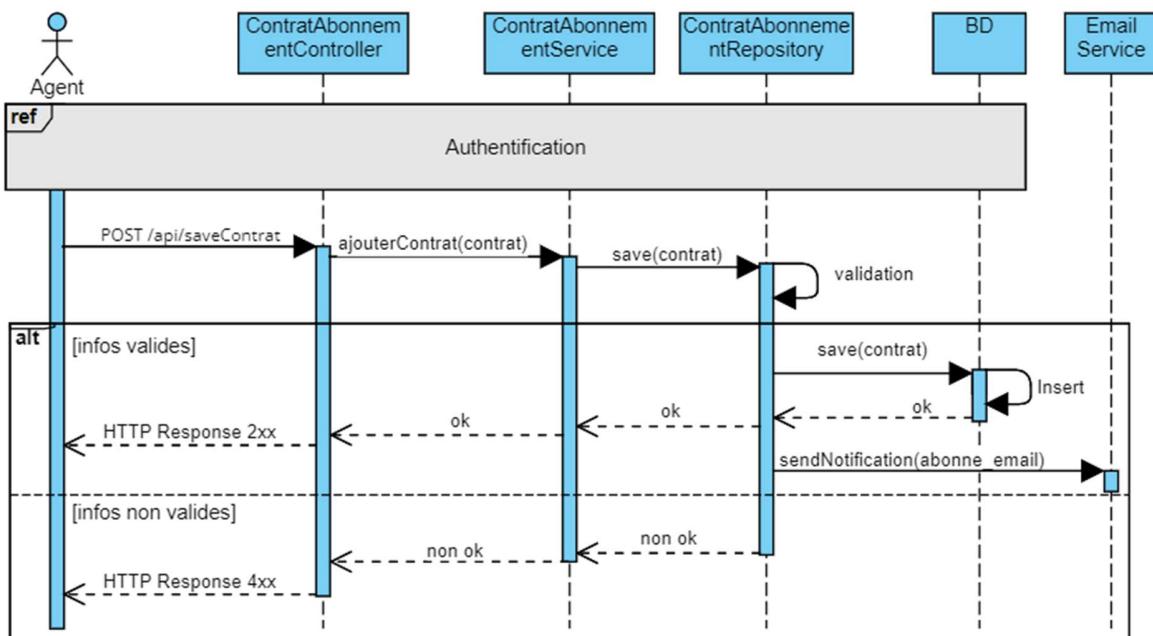


Figure 10 : Diagramme de séquence pour l'ajout d'un contrat

2.3.1.2 Diagramme de séquence pour la modification d'un contrat

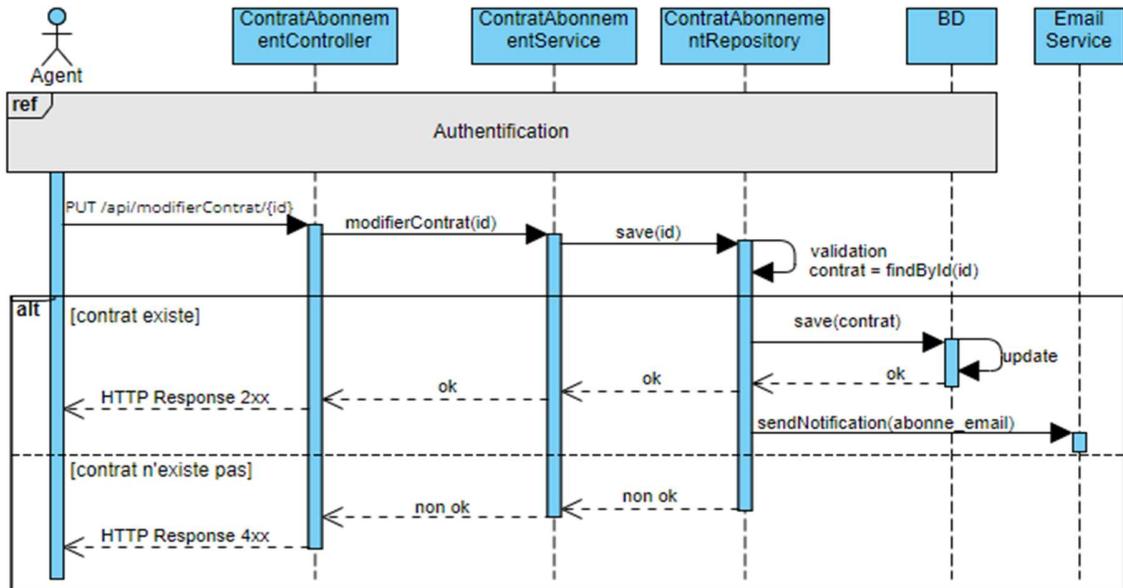


Figure 11 : Diagramme de séquence pour la modification d'un contrat

2.3.1.3 Diagramme de séquence pour la suppression d'un contrat

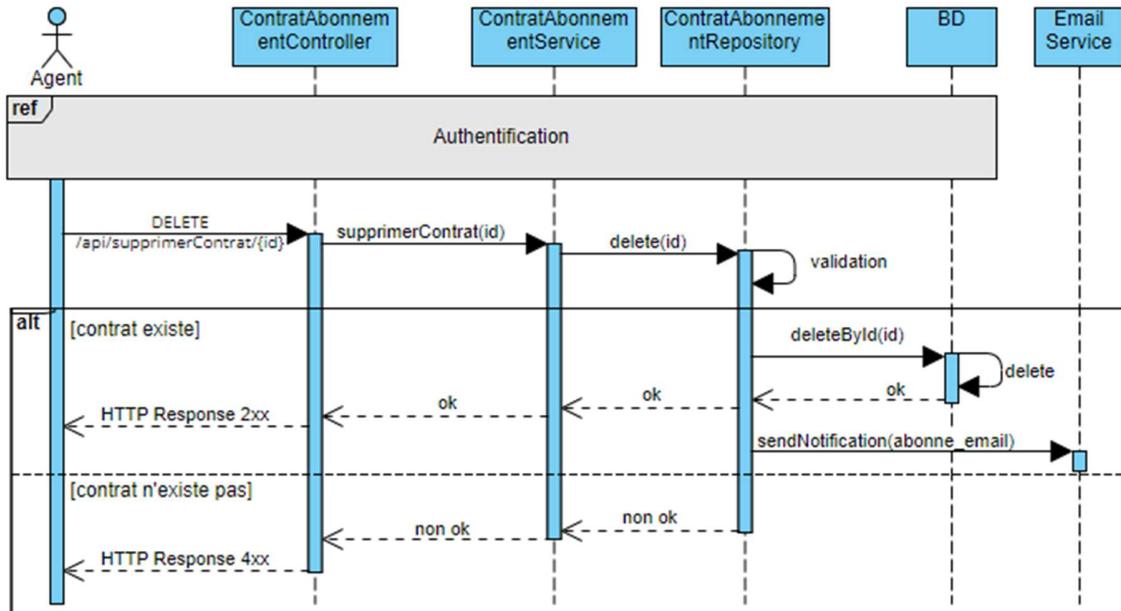


Figure 12 : Diagramme de séquence pour la suppression d'un contrat

2.3.1.4 Diagramme de séquence pour la recherche d'un contrat

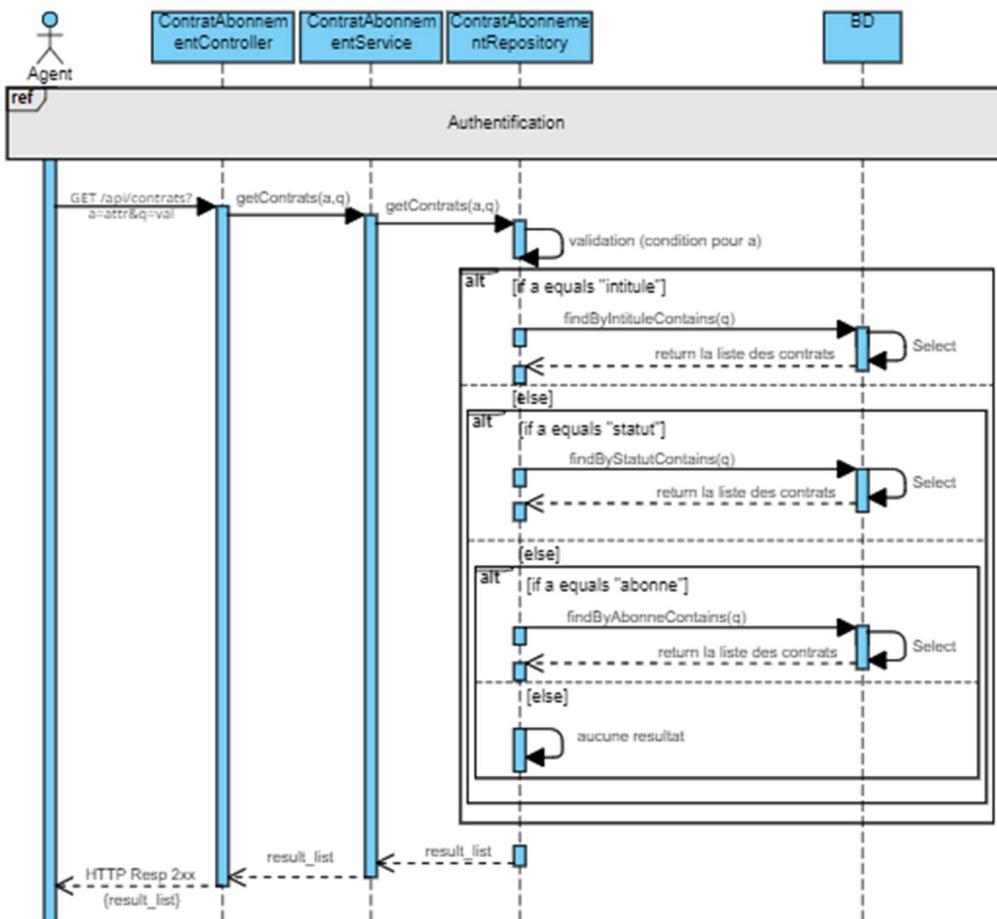


Figure 13 : Diagramme de séquence pour la recherche d'un contrat

2.3.1.5 Diagramme de séquence pour la consultation des contrats

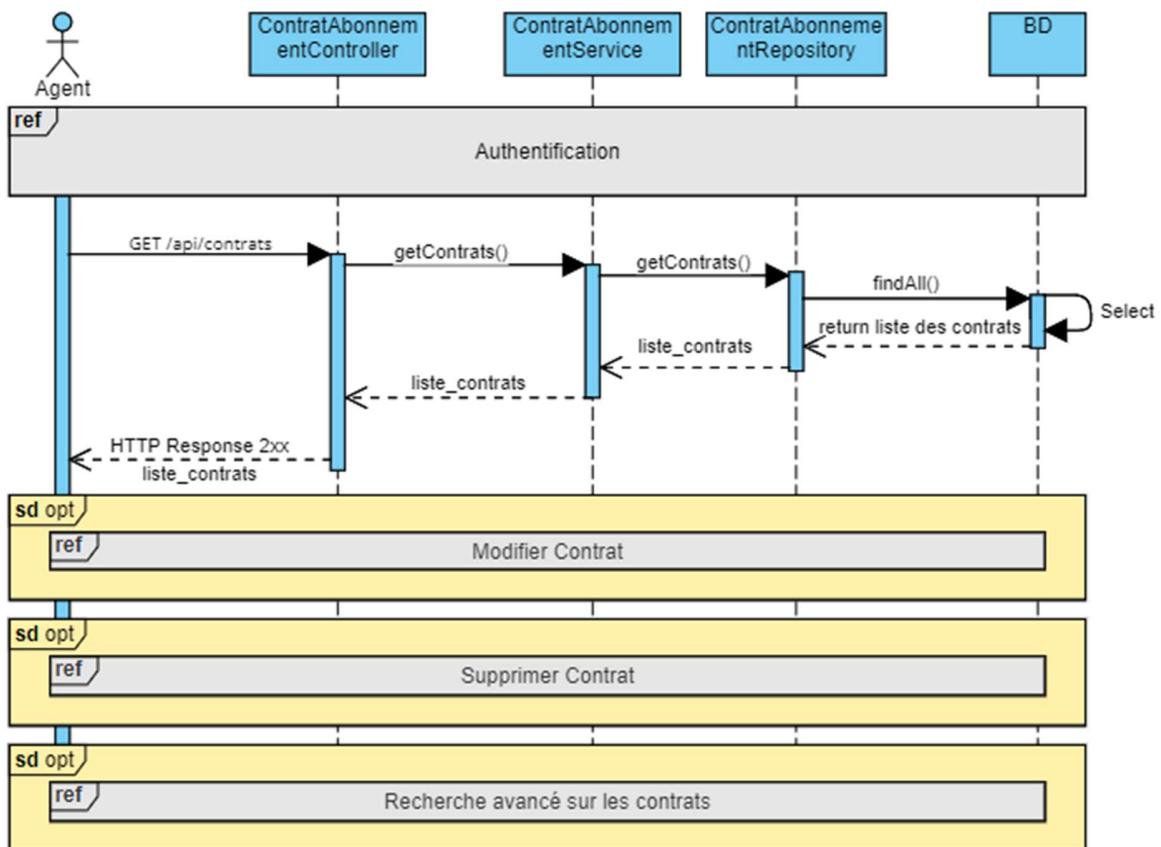


Figure 14 : Diagramme de séquence pour la consultation des contrats

2.3.2 Gestion des abonnés

2.3.2.1 Diagramme de séquence pour l'ajout d'un abonné

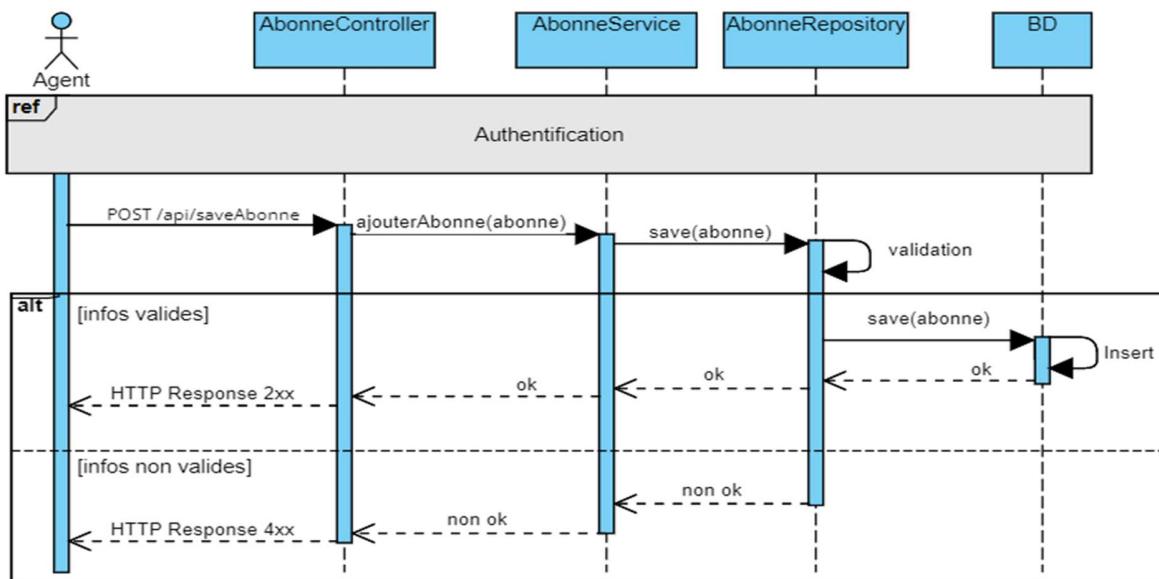


Figure 15 : Diagramme de séquence pour l'ajout d'un abonné

2.3.2.2 Diagramme de séquence pour la modification d'un abonné

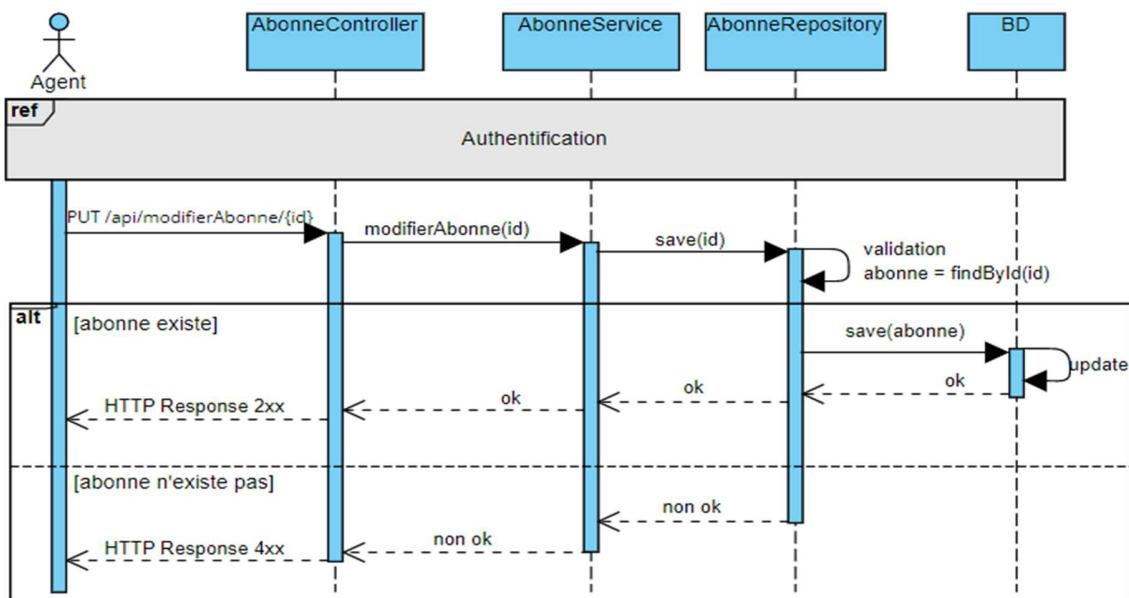


Figure 16 : Diagramme de séquence pour la modification d'un abonné

2.3.2.3 Diagramme de séquence pour la suppression d'un abonné

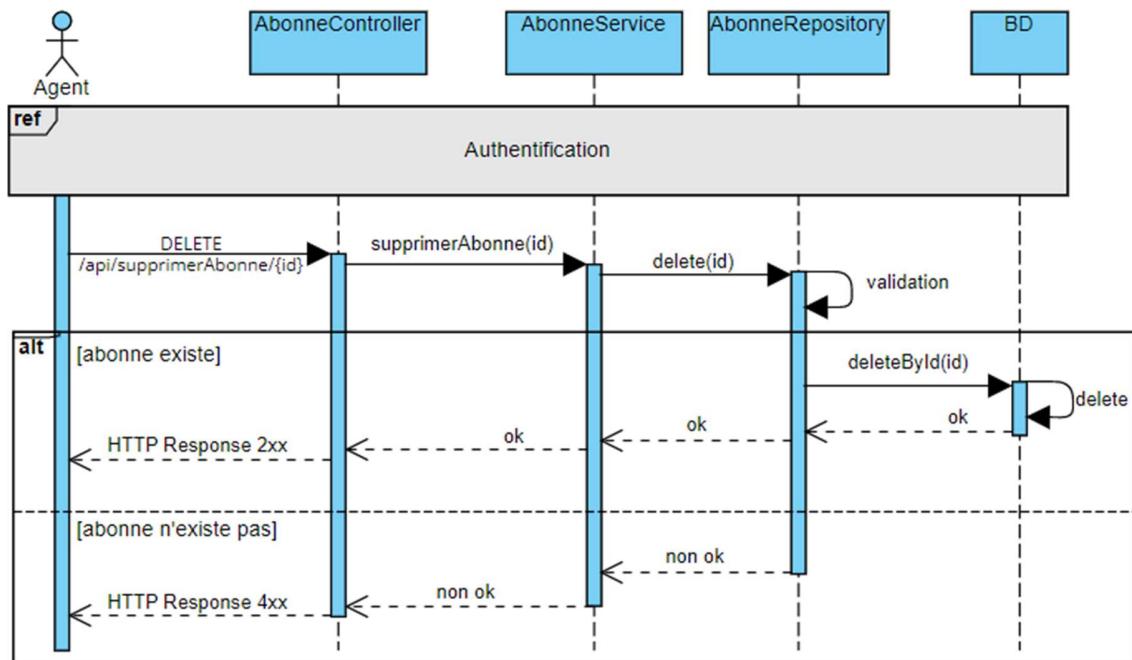


Figure 17 : Diagramme de séquence pour la suppression d'un abonné

2.3.2.4 Diagramme de séquence pour la recherche d'un abonné

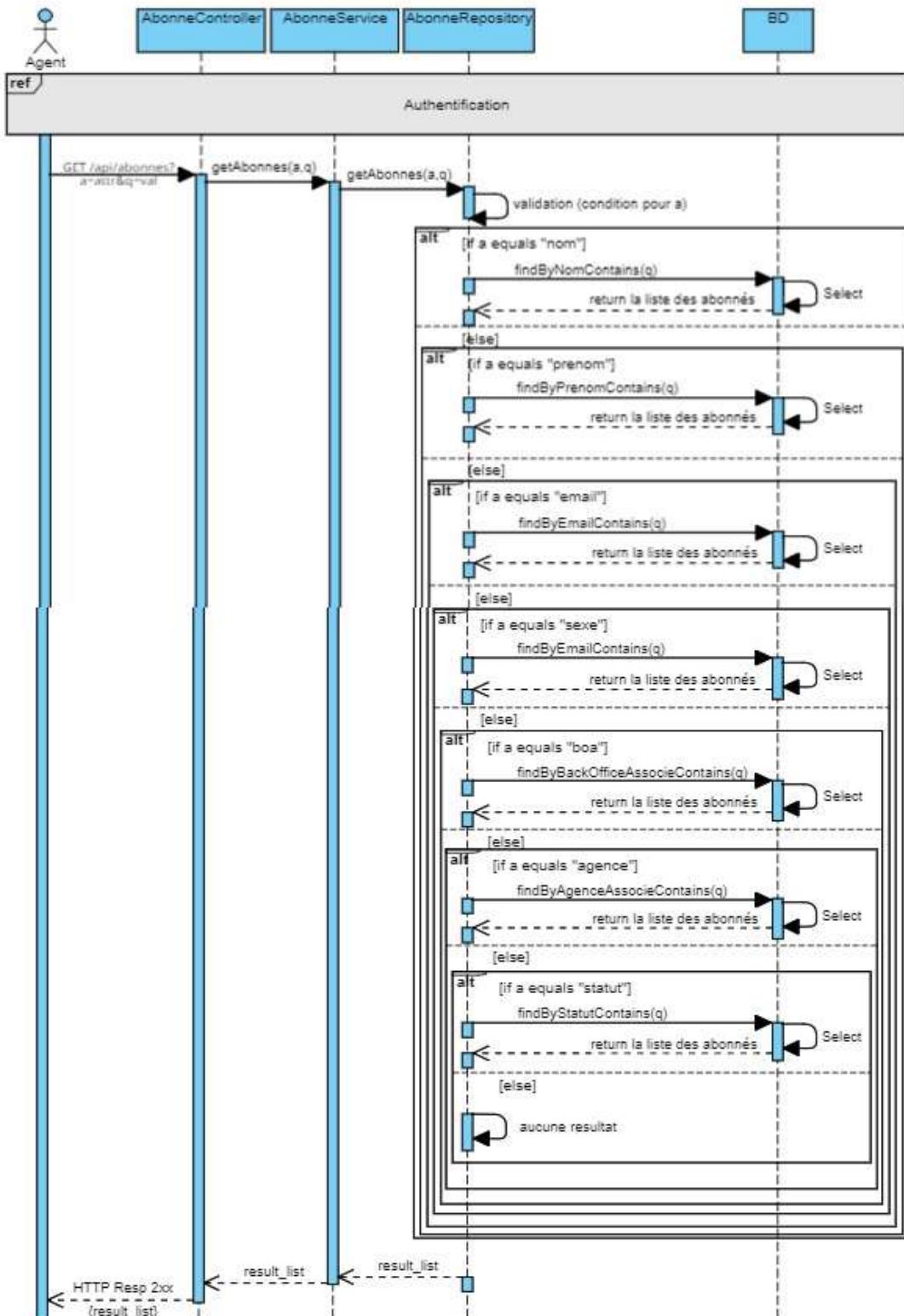


Figure 18 : Diagramme de séquence pour la recherche d'un abonné

2.3.2.5 Diagramme de séquence pour la consultation des abonnés

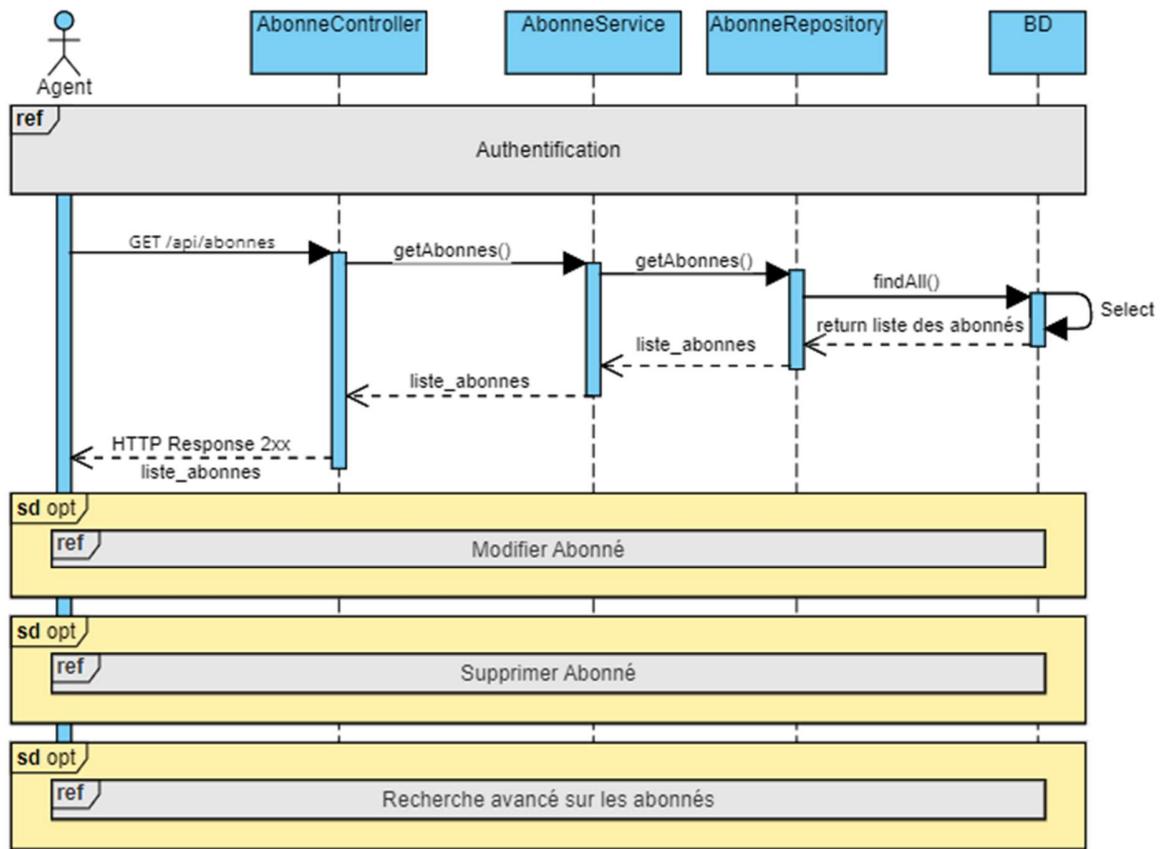


Figure 19 : Diagramme de séquence pour la consultation des abonnés

2.3.3 Gestion des offres commerciales

2.3.3.1 Diagramme de séquence pour l'ajout d'une offre

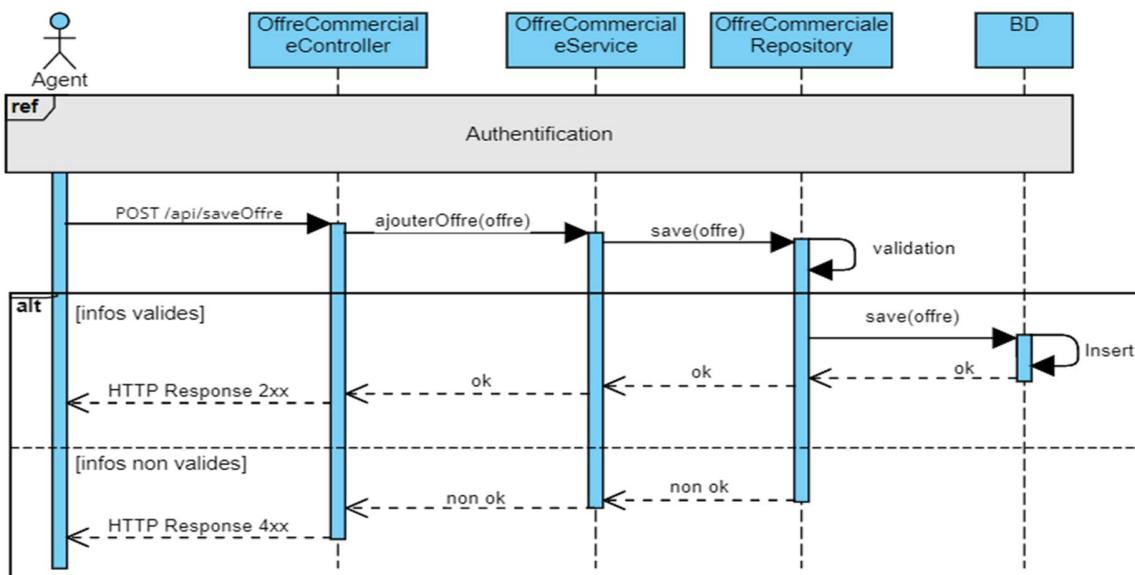


Figure 20 : Diagramme de séquence pour l'ajout d'une offre

2.3.3.2 Diagramme de séquence pour la modification d'une offre

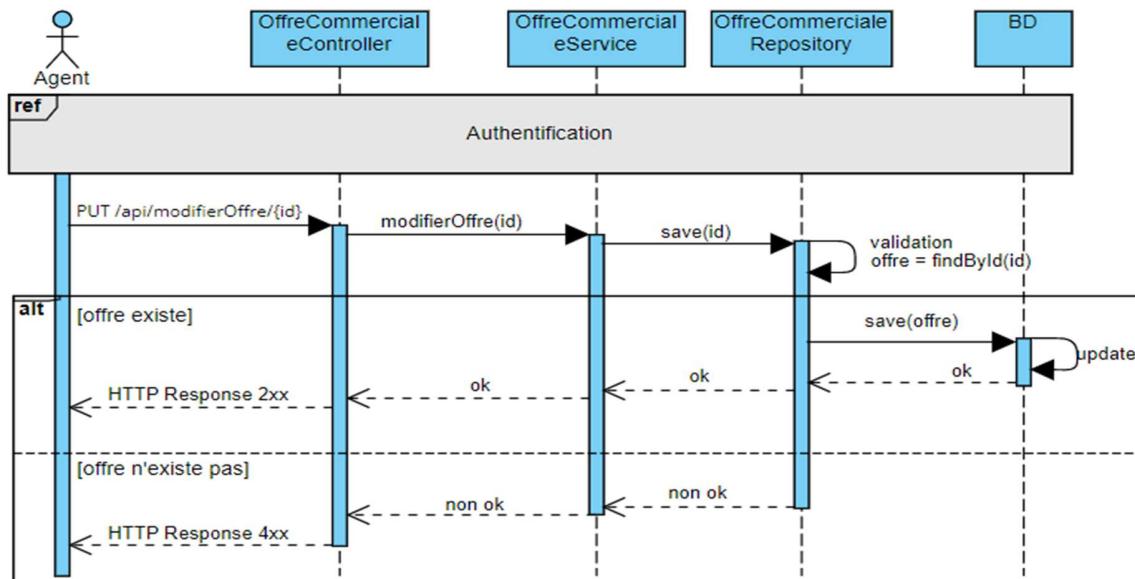


Figure 21 : Diagramme de séquence pour la modification d'une offre

2.3.3.3 Diagramme de séquence pour la suppression d'une offre

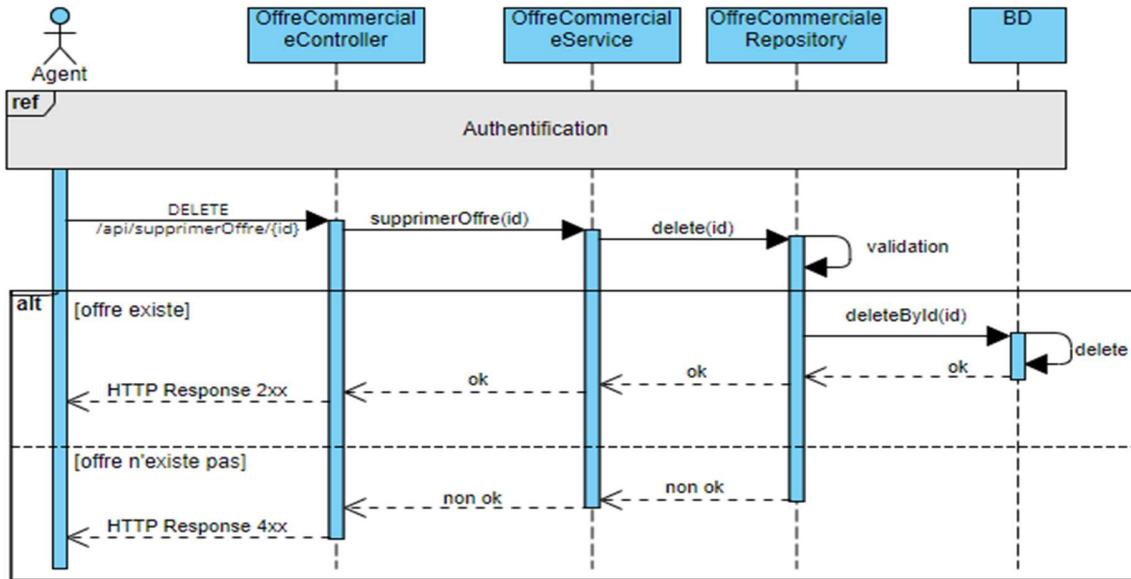


Figure 22 : Diagramme de séquence pour la suppression d'une offre

2.3.3.4 Diagramme de séquence pour la recherche d'une offre

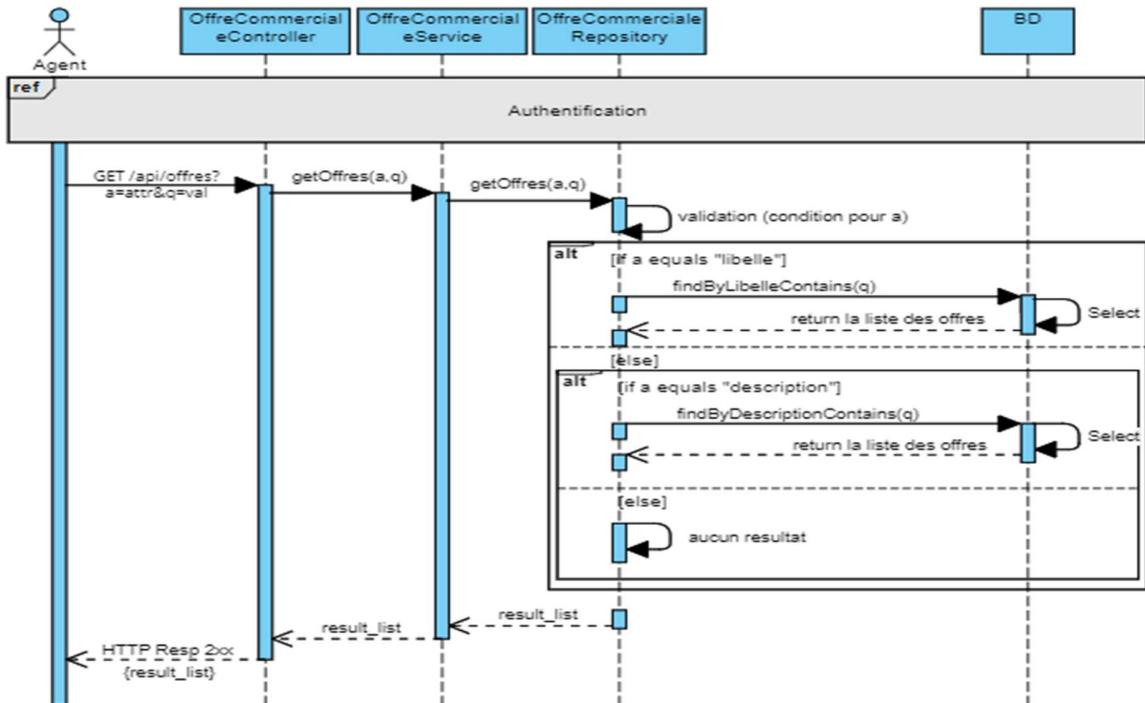


Figure 23 : Diagramme de séquence pour la recherche d'une offre

2.3.3.5 Diagramme de séquence pour la consultation des offres

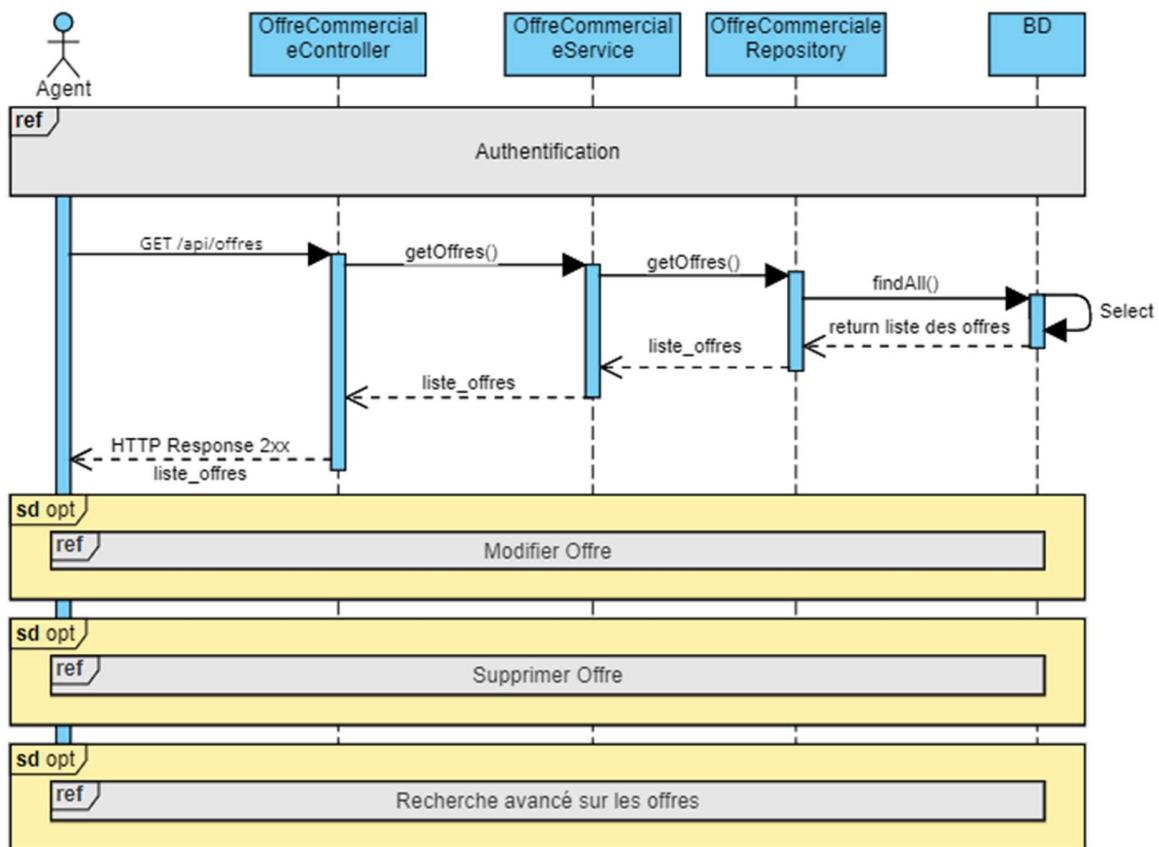


Figure 24 : Diagramme de séquence pour la consultation des offres

2.4 Diagrammes de classe

La modélisation de tout système passe essentiellement par un diagramme de classes qui englobe les différentes entités et les relations entre elles. Compte tenu des spécifications établies dans les chapitres précédents, le diagramme de classes global du système contient un certain nombre de classes :

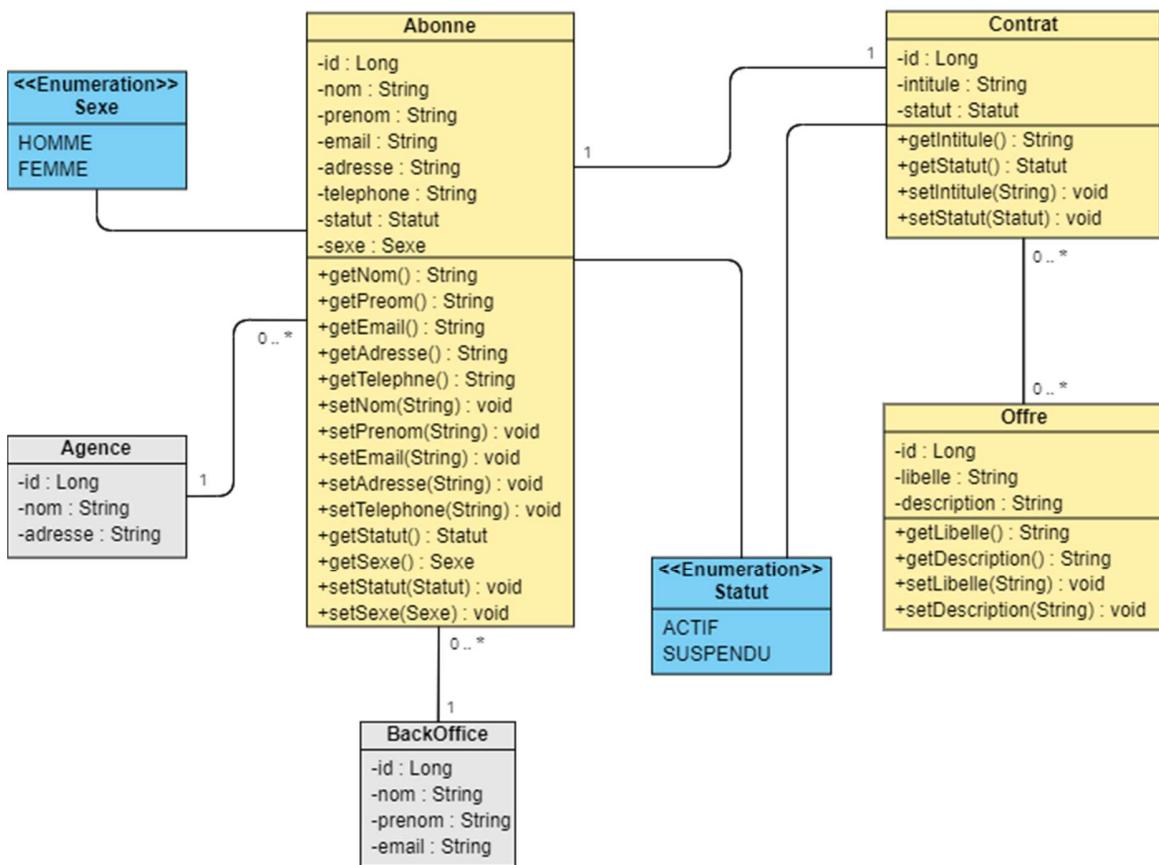


Figure 25 : Diagramme de classe pour les entités du projet

Class	Description
Abonne	Représente l'abonné de la banque
Contrat	Représente le contrat entre l'abonné et la banque
Offre	Représente une offre de la banque pour ces abonnés
Agence	Représente une Agence
BackOffice	Représente un employé d'une banque

Tableau 2 : Description des entités

Et comme mon application ne contient pas seulement des entités, elle contient les couches repository, service et controller, et donc voici un diagramme de classe qui illustre la relation entre les classes/interfaces de ces trois couches :

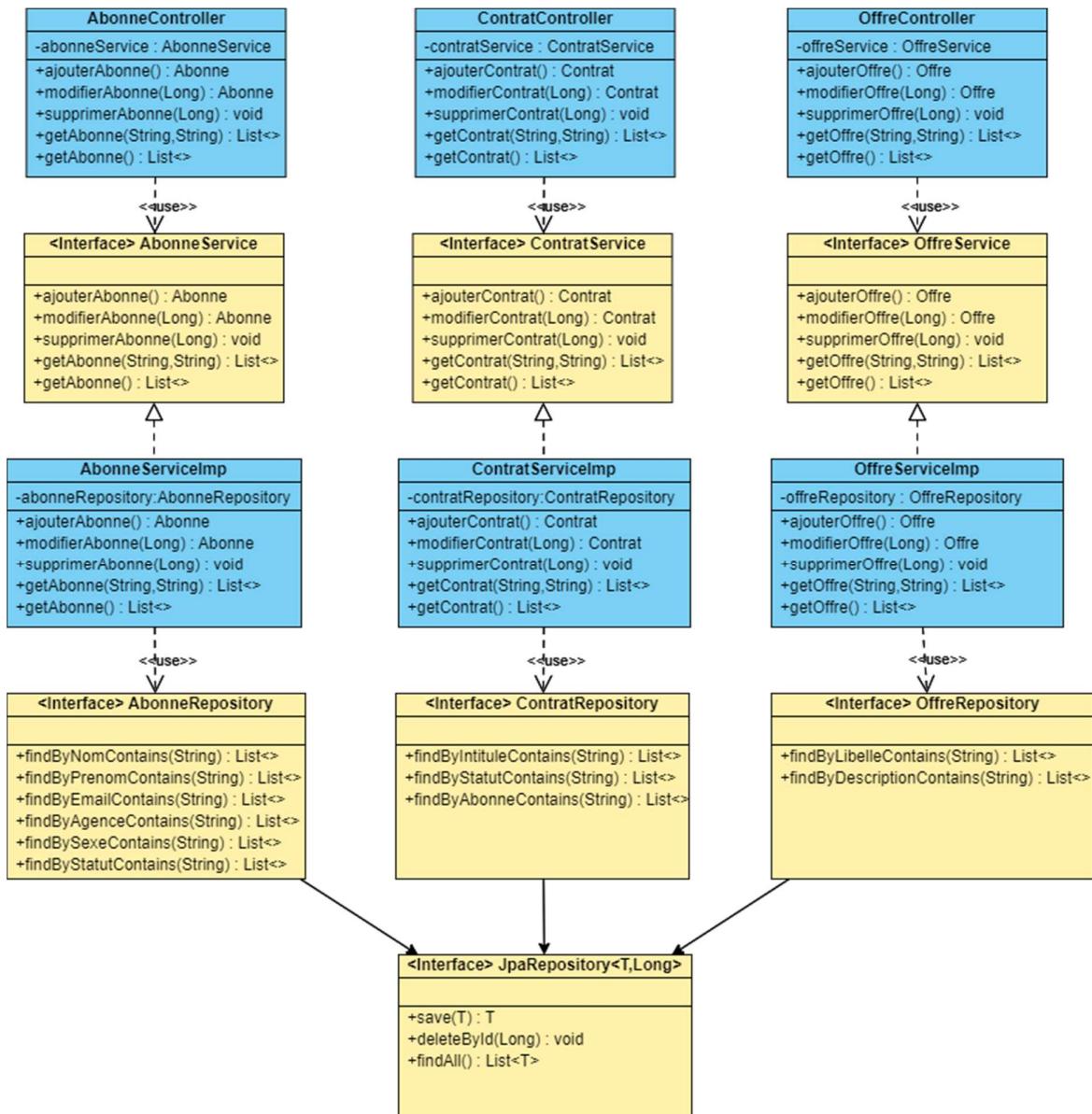


Figure 26 : Diagramme de classe des trois couches repository, service et controller

Class/Interface	Description
AbonneController	Représente la classe des points de terminaison pour l'abonné
ContratController	Représente la classe des points de terminaison pour le contrat
OffreController	Représente la classe des points de terminaison pour l'offre
AbonneService	Représente l'interface service de l'abonné
AbonneServiceImp	L'implémentation de service de l'abonné
ContratService	Représente l'interface service du contrat
ContratServiceImp	L'implémentation de service du contrat
OffreService	Représente l'interface service de l'offre
OffreServiceImp	L'implémentation de service de l'offre
AbonneRepository	Représente l'interface data de l'abonné
ContratRepository	Représente l'interface data du contrat
OffreRepository	Représente l'interface data de l'offre
JpaRepository	Représente l'interface Jpa qui définit les différentes méthodes pour interagir avec la BD

Tableau 3 : Description des classes et interfaces du projet

2.5 Conclusion

Ce chapitre était l'objet des spécifications fonctionnelles et l'analyse Conceptuelle du système. Les besoins fonctionnels étaient recensés, les acteurs interagissant avec le système ainsi que l'illustration des cas d'utilisation. Après nous avons présenté les résultats de l'analyse fonctionnelle qui nous a permis de dégager les objets métiers qui nous ont servi pour réaliser les diagrammes UML de la phase de conception, à savoir les diagrammes de séquences et les diagramme de classes. La mise en œuvre de notre application fera l'objet du chapitre suivant. Après avoir présenté une vue claire sur les différents besoins escomptés du projet d'un point de vue fonctionnel et conceptuel, l'étude technique sera le sujet du prochain chapitre.

Chapitre 3

Etude Technique du projet

Résumé

Ce chapitre a pour objectif de présenter l'étude technique du projet. Il présentera donc les architectures techniques et applicatives du système, ainsi que les Frameworks et les designs patterns utilisés.

3.1 Introduction

La création et la conception d'un système informatique est un processus qui implique l'utilisation de nouvelles technologies et outils. Parmi les outils utilisés, on trouve :

- ❖ Les langages de programmation web qui permettent de créer des applications sur navigateur.
- ❖ Langage de balisage pour garantir une bonne présentation visuelle.
- ❖ Un environnement de développement (IDE) dédié à la programmation web.
- ❖ Langage de programmation coté serveur pour la création d'une interface de communication avec ce dernier.
- ❖ Les systèmes de gestion de base de données qui répondent aux différentes opérations de stockage et de traitement des données.

L'objectif de ce chapitre est de présenter les différents outils et technologies qu'on a utilisés lors de la réalisation des tâches du projet.

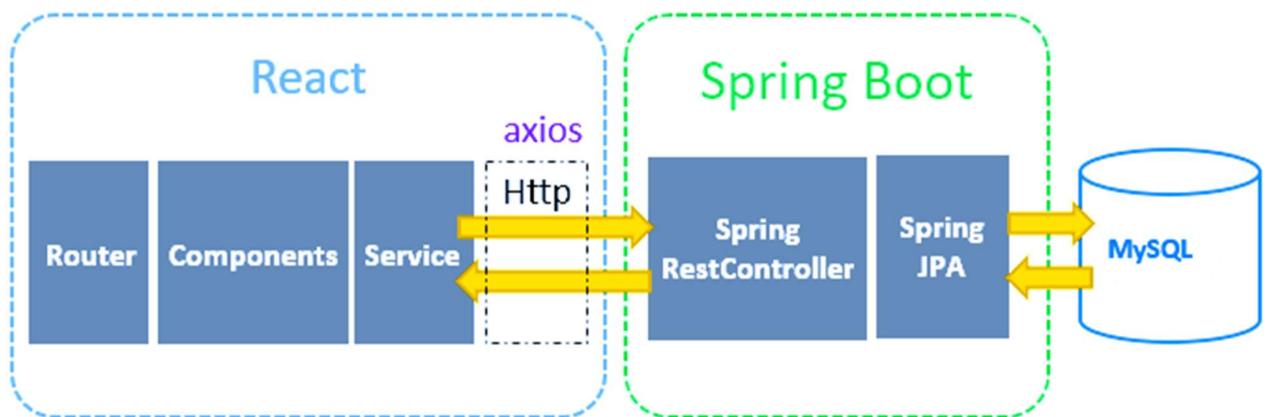


Figure 27 : Schéma de l'aperçu global du projet

3.2 Outils et environnement de travail

3.2.1 Outils de gestion de version

- ❖ **Git** : est un logiciel de gestion de versions distribué, créé par Linus Torvalds en 2005 et distribué gratuitement sous licence publique générale GNU version 2.



Figure 28 : Logo de Git

Dans ce système, peu importe le nombre de machines qui travaillent sur le même projet, chacune d'entre elles possède une copie intégrale du dépôt et est capable de travailler seule pour partager ses mises à jour en temps voulu avec les autres intervenants. Cela a un impact en termes de sécurité, puisque sans serveur central, la tolérance aux pannes est plus grande : il suffit qu'une seule machine soit en vie pour que le projet puisse être redistribué à volonté. Cela a aussi un impact en termes de performances : si on ne fait pas d'accès réseau en permanence on gagne forcément du temps. Le revers de la médaille est que des conflits peuvent survenir si plusieurs personnes modifient les mêmes lignes de code chacune de leur côté.

- ❖ **Github** : est une plateforme de gestion de versions et de collaboration pour le développement de logiciels. Il utilise le système de contrôle de version Git, qui permet de suivre les modifications apportées au code source d'un projet au fil du temps. GitHub offre une interface web conviviale qui facilite la collaboration entre les membres d'une équipe de développement.



Figure 29 : Logo de GitHub

3.2.2 IDEs et éditeurs de text

- ❖ **IntelliJ IDEA** : c'est un environnement de développement intégré (IDE) écrit en Java pour le développement de logiciels informatiques. Il est développé par JetBrains (anciennement connu sous le nom d'IntelliJ) et est disponible sous forme d'édition communautaire sous licence Apache 2 et dans une édition commerciale propriétaire. Les deux peuvent être utilisés pour le développement commercial.

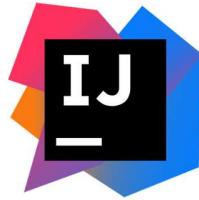


Figure 30 : Logo de IntelliJ IDEA

J'ai choisi IntelliJ IDEA comme environnement de développement principal pour la partie back-end de mon application, qui utilise le framework Spring Boot, en raison de plusieurs avantages. Tout d'abord, l'intégration native de Spring dans IntelliJ simplifie considérablement le développement avec ce framework. L'IDE offre des fonctionnalités spécifiques à Spring, telles que la création automatique de configurations, la navigation facilitée dans la hiérarchie des composants Spring, et une assistance intelligente pour l'injection de dépendances.

En outre, IntelliJ IDEA offre une excellente prise en charge de Maven et Gradle, les outils de gestion de dépendances couramment utilisés dans les projets Spring Boot. L'IDE automatise la configuration du projet, la gestion des dépendances et facilite l'exécution des tâches de build.

- ❖ **Visual Studio Code :** est un éditeur de code source léger, open source et multiplateforme développé par Microsoft. Il offre une interface utilisateur intuitive, des fonctionnalités puissantes de développement, et prend en charge une variété de langages de programmation. Grâce à son écosystème riche d'extensions, VSCode peut être personnalisé pour répondre aux besoins spécifiques des développeurs, ce qui en fait un choix populaire au sein de la communauté de développement logiciel.



Figure 31 : Logo de VSCode

J'utilise Visual Studio Code comme environnement de développement pour la partie front-end de mon application pour plusieurs raisons. Tout d'abord, la facilité d'utilisation de l'interface de VSCode me permet de travailler de manière productive tout en bénéficiant de fonctionnalités telles que la coloration syntaxique, l'autocomplétion intelligente et la navigation aisée dans le code.

De plus, les extensions disponibles pour VSCode sont extrêmement utiles dans le contexte du développement web. Des extensions telles que celles dédiées à HTML, CSS, JavaScript, et aux frameworks populaires comme React.

3.3 Présentation de la partie Frontend du projet

3.3.1 Architecture logiciel de la partie Frontend

La partie frontend de mon projet s'appuie sur la puissante bibliothèque ReactJS pour la construction d'interfaces utilisateur interactives. Adoptant le modèle de développement basé sur les composants, notre architecture garantit une modularité optimale et une facilité de maintenance. La structure est élaborée de manière hiérarchique, conformément aux meilleures pratiques de développement React, avec une organisation logique des composants par fonctionnalité.

La gestion de l'état de l'application est effectuée directement au niveau des composants, sans recourir à Redux. Cette approche offre une simplicité accrue tout en conservant une structure claire et compréhensible. Les fichiers sources sont regroupés selon une logique fonctionnelle, facilitant ainsi la collaboration et la compréhension du code.

L'organisation de la partie frontend inclut également l'utilisation de React Router pour faciliter la navigation au sein de l'application. De plus, les styles sont gérés à l'aide de Framework Bootstrap, assurant une encapsulation efficace des styles au niveau du composant.

Pour donner un aperçu visuel de la structure de notre frontend ReactJS, veuillez trouver ci-joint une prise d'écrans de la racine de notre projet, illustrant la disposition des fichiers et des répertoires.

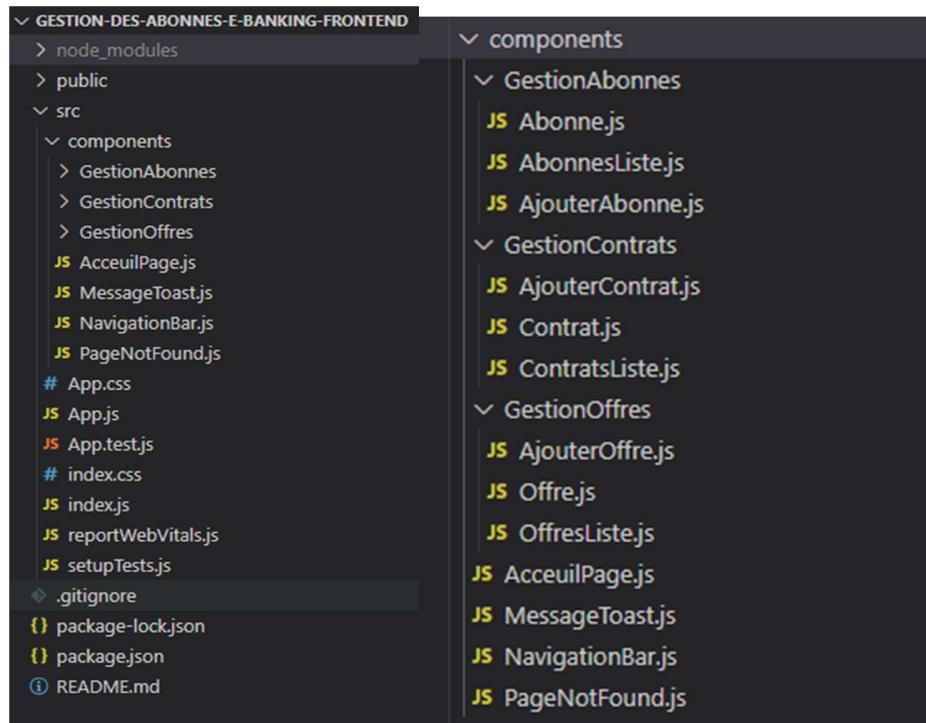


Figure 32 : La Racine de la partie front end

3.3.2 Langages et technologies

- ❖ **JavaScript** est un langage de programmation côté client, largement utilisé pour le développement web. Il permet d'ajouter des fonctionnalités interactives aux pages web, améliorant ainsi l'expérience utilisateur en permettant des manipulations dynamiques du contenu.

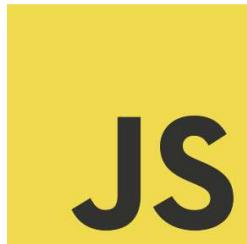


Figure 33 : Logo de JavaScript

JavaScript a été choisi en raison de sa polyvalence et de son rôle central dans le développement web. Il offre une exécution rapide des opérations côté client, facilitant la création d'interfaces utilisateur réactives et dynamiques, essentielles pour offrir une expérience utilisateur moderne et fluide.

- ❖ **ReactJS** est une bibliothèque JavaScript déclarative utilisée pour construire des interfaces utilisateur interactives. Sa structure basée sur les composants permet une gestion modulaire du code, favorisant la réutilisabilité et la facilité de maintenance.

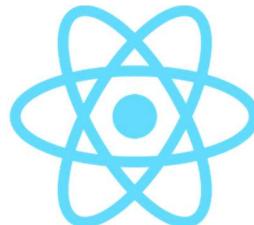


Figure 34 : Logo de ReactJs

J'ai opté pour ReactJS en raison de sa nature déclarative et de son approche axée sur les composants. La modularité inhérente à React facilite le développement en permettant une réutilisation optimale du code, tout en garantissant des mises à jour efficaces de l'interface utilisateur grâce à sa virtual DOM.

- ❖ **Babel** est un transpileur JavaScript qui convertit le code ECMAScript moderne en une version compatible avec les navigateurs les plus anciens. Il facilite le développement en assurant une compatibilité étendue du code.



Figure 35 : Logo de Babel

L'intégration de Babel dans notre pile technologique découle de la nécessité d'assurer une compatibilité maximale avec différents navigateurs. Transpiler notre code garantit une expérience utilisateur uniforme, indépendamment de la plateforme ou du navigateur utilisé, contribuant ainsi à la portabilité de notre application.

- ❖ **Bootstrap** est une bibliothèque front-end open source, comprenant des outils et des composants pour le développement rapide et efficace de sites web responsives.



Figure 36 : Logo de Bootstrap

La décision d'utiliser Bootstrap pour la gestion des styles repose sur sa capacité à accélérer le développement tout en fournissant une esthétique cohérente. La grille flexible et les composants prêts à l'emploi de Bootstrap simplifient le processus de conception de l'interface utilisateur, permettant de maintenir une apparence professionnelle et responsive avec efficacité.

3.4 Présentation de la partie Backend du projet

3.4.1 Design Patterns

Dans le contexte de Spring Boot, les design patterns sont des solutions architecturales réutilisables qui résolvent des problèmes courants rencontrés lors du développement d'applications. Spring Boot favorise l'utilisation de plusieurs design patterns, offrant ainsi une structure robuste et modulaire. Parmi les design patterns clés adoptés dans notre projet, nous mettons particulièrement l'accent sur trois d'entre eux : DAO (Data Access Object), IOC (Inversion of Control) et DTO (Data Transfer Object).

3.4.1.1 DAO (Data Access Object)

Le modèle de conception DAO (Data Access Object) a été rigoureusement adopté dans la partie backend de notre projet basé sur Spring Boot. Le DAO est un modèle architecturale qui sépare la logique d'accès aux données du logique métier, favorisant ainsi une structure modulaire et une gestion efficace des opérations sur la base de données. Concrètement, chaque entité de notre application, qu'il s'agisse d'utilisateurs, de produits ou d'autres données, dispose d'une classe DAO dédiée. Ces classes DAO encapsulent les opérations CRUD (Create, Read, Update, Delete) spécifiques à chaque entité, offrant ainsi une abstraction claire et cohérente de la manipulation des données.

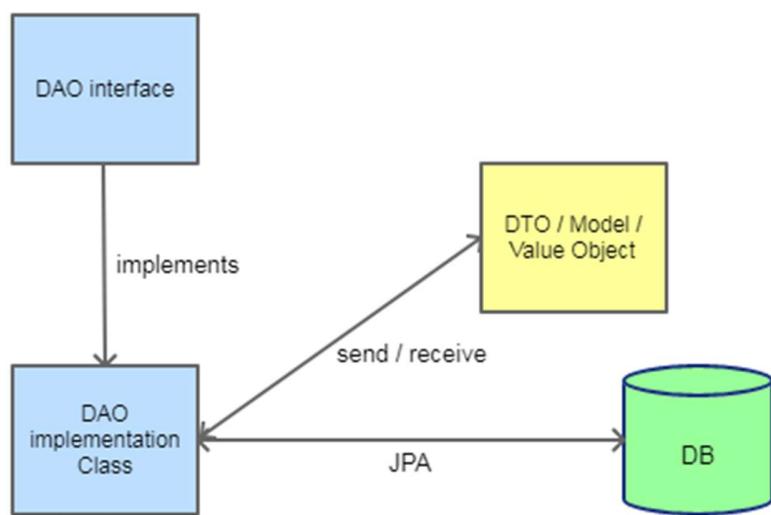


Figure 37 : schéma illustrant le fonctionnement de DAO Design pattern

L'adoption du modèle DAO dans notre projet Spring Boot garantit une flexibilité accrue en permettant des modifications à la logique d'accès aux données sans perturber le logique métier sous-jacent. De plus, cela facilite la gestion des transactions, assurant une intégrité des données tout en optimisant les performances d'accès à la base de données. Globalement, l'implémentation du design pattern DAO renforce la clarté architecturale de notre backend, contribuant ainsi à la maintenabilité et à la scalabilité de notre application.

3.4.1.2 IOC (Inversion of Control)

L'Inversion de Contrôle (IoC) est un principe de conception fondamental adopté par Spring Boot, représentant un changement de paradigme dans la manière dont les composants sont gérés et les dépendances sont traitées au sein d'une application. Dans la programmation traditionnelle, le flux de contrôle est déterminé par le programme lui-même, mais dans le conteneur IoC de Spring, le contrôle est inversé ou remis au framework. Dans le contexte de Spring Boot, le conteneur IoC, souvent appelé le

conteneur Spring, gère le cycle de vie des objets Java (beans) et est responsable de l'injection de dépendances, permettant une architecture plus faiblement couplée et modulaire.

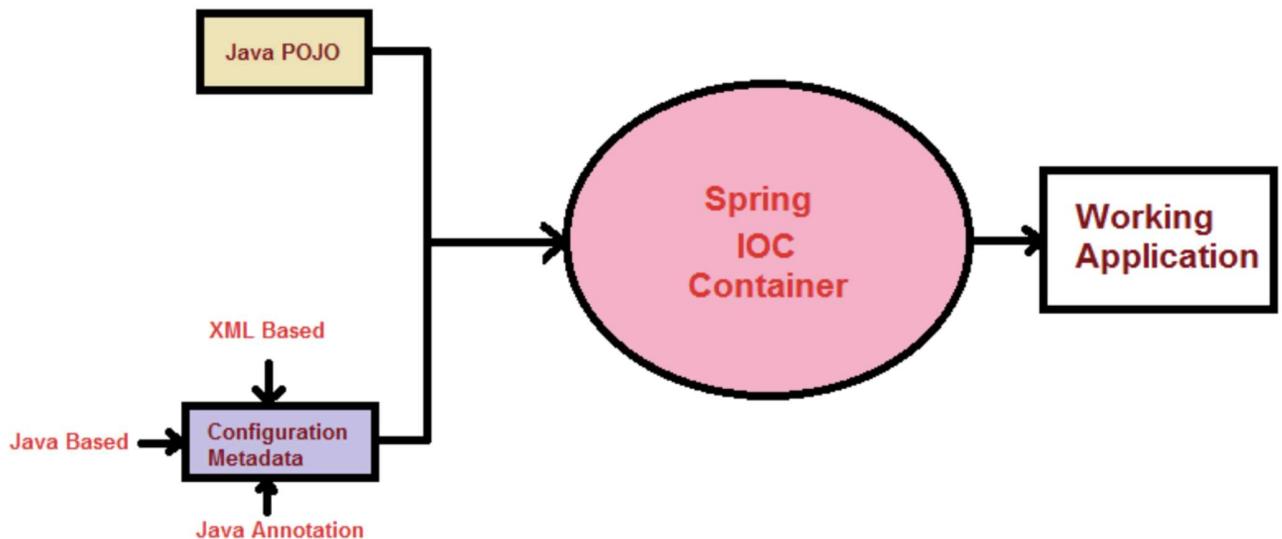


Figure 38 : Schéma illustrant le fonctionnement de IOC Container

L'implémentation de l'IoC dans Spring Boot est facilitée par l'utilisation du conteneur IoC de Spring. Les développeurs déclarent les beans, représentant divers composants de l'application, dans des fichiers de configuration ou des classes Java, et le conteneur est responsable de l'instanciation, de la gestion et de la liaison de ces beans entre eux. L'injection de dépendances, un concept clé de l'IoC, permet l'injection automatique des dépendances dans les beans, réduisant le couplage entre les composants et favorisant la réutilisabilité. Les avantages de l'IoC dans Spring Boot incluent une meilleure maintenabilité, testabilité et évolutivité. En permettant au framework de gérer le flux de contrôle et les dépendances, les développeurs peuvent se concentrer sur l'écriture de la logique métier, résultant en une base de code plus modulaire, extensible et facilement maintenable. De plus, l'IoC améliore la flexibilité de l'application, car les modifications des dépendances ou l'ajout de nouveaux composants peuvent être intégrés sans modifier le code existant, contribuant à un processus de développement plus agile et adaptable.

3.4.1.3 DTO (Data Transfer Object)

Le modèle DTO est utilisé pour transférer des données entre les différentes couches de l'application, optimisant ainsi les échanges d'informations. Dans notre projet Spring Boot, les objets DTO sont employés pour encapsuler les données spécifiques nécessaires à la communication entre le frontend et le backend. Cette approche favorise une communication claire et structurée, évitant le transfert inutile de données et optimisant les performances globales de l'application.

3.4.2 Architecture logiciel de la partie Backend

La partie backend de notre application est structurée selon une architecture logicielle basée sur le framework Spring Boot. Cette architecture monolithique offre une approche cohérente pour le développement, la gestion, et la maintenance de la logique métier de notre application. Les composants essentiels, tels que les contrôleurs, les services, et les repositories, sont organisés de manière à favoriser la modularité, la réutilisabilité, et la lisibilité du code.

Les contrôleurs, responsables de la gestion des requêtes HTTP, interagissent avec les services pour traiter les opérations métier. Les services encapsulent la logique métier, permettant une séparation claire des responsabilités. Les repositories, utilisant le Framework Hibernate, facilitent l'interaction avec la base de données MySQL, assurant une persistance des données efficace et cohérente.

La gestion des dépendances est simplifiée grâce à l'inversion de contrôle (IoC) offerte par Spring Boot. Les beans, définis dans les classes de configuration, sont automatiquement injectés là où ils sont nécessaires, réduisant ainsi le couplage entre les composants et favorisant une architecture flexible.

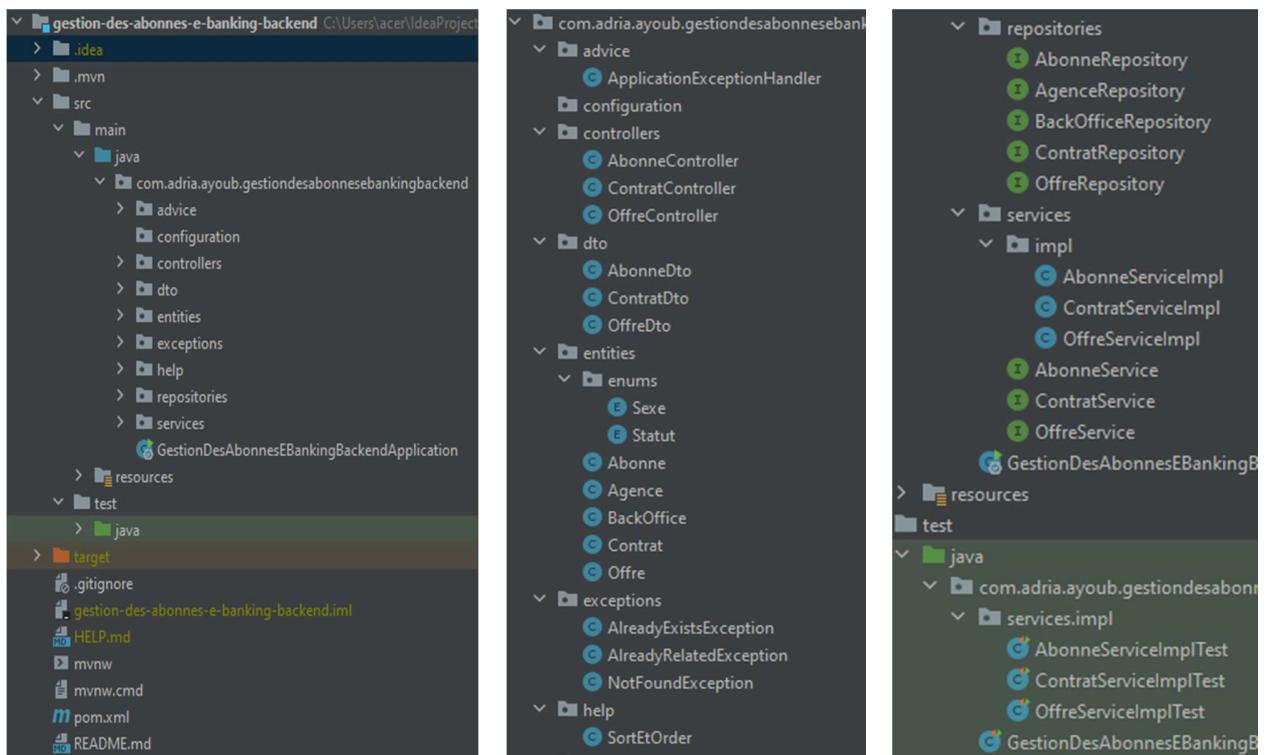


Figure 39 : Racine de la partie backend

3.4.3 Langages et technologies

- ❖ **Java** : en tant que langage fondamental dans la partie backend de notre projet, est choisi pour ses performances fiables, sa portabilité et sa robustesse. La plateforme Java offre une approche polyvalente pour le développement d'applications, favorisant une gestion sécurisée de la mémoire, une facilité de maintenance et une portabilité élevée entre différentes plates-formes. La richesse de l'écosystème Java, avec ses bibliothèques étendues et son support communautaire, renforce notre capacité à développer des composants backend solides et évolutifs.



Figure 40 : Logo de Java

- ❖ **Spring Boot** : est au cœur de notre architecture backend, offrant un cadre de développement qui simplifie la création d'applications Java autonomes. Les fonctionnalités prêtes à l'emploi de Spring Boot, telles que l'injection de dépendances, la gestion des transactions et la configuration automatique, accélèrent le développement tout en assurant une structure modulaire et bien organisée. L'utilisation de Spring Boot améliore également la productivité du développement en réduisant la configuration boilerplate, permettant ainsi à l'équipe de se concentrer davantage sur la logique métier.



Figure 41 : Logo de Spring Boot

L'adoption de Spring Boot dans mon projet a été motivée par sa capacité à simplifier de manière significative le développement d'applications Java tout en offrant un cadre solide et cohérent. En tant que framework basé sur Spring, Spring Boot accélère le processus de développement en fournissant des conventions par défaut, des configurations automatiques et des fonctionnalités prêtes à l'emploi. Sa philosophie "convention over configuration" réduit la complexité liée à la configuration manuelle, permettant aux développeurs de se concentrer davantage sur la logique métier plutôt que sur les détails techniques.

- ❖ **Framework Hibernate :** a été intégré à mon projet pour la gestion efficace de la persistance des données dans la base de données. En tant que framework ORM (Object-Relational Mapping), Hibernate simplifie l'interaction avec la base de données en permettant le mapping objet-relationnel. Cela se traduit par une écriture de code plus intuitive, où les objets Java sont directement associés aux entités de base de données, réduisant ainsi la complexité des requêtes SQL et facilitant la maintenance. L'utilisation de Hibernate contribue à la gestion transparente des opérations CRUD et à l'amélioration des performances de l'accès aux données, renforçant ainsi la fiabilité et l'efficacité de notre backend.

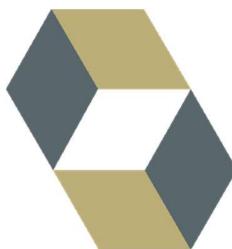


Figure 42 : Logo de Framework Hibernate

- ❖ **MySQL :** L'adoption de MySQL en tant que système de gestion de base de données dans notre projet repose sur plusieurs considérations fondamentales qui convergent vers la fiabilité, la performance, et la facilité de gestion des données. MySQL est un système de gestion de base de données relationnelle open-source reconnu pour sa stabilité, sa robustesse, et sa conformité aux normes SQL, offrant ainsi une solution éprouvée pour le stockage et la récupération des données.



Figure 43 : Logo de MySQL

La raison qui a guidé notre choix vers MySQL est sa réputation en tant que base de données fiable et stable, utilisée avec succès dans une multitude d'applications à travers le monde. Sa gestion des transactions ACID (Atomicité, Cohérence, Isolation, Durabilité) assure l'intégrité des données, garantissant des opérations fiables même dans des scénarios complexes.

3.4.4 Tests de la partie backend

Cette partie constitue un volet essentiel de ce rapport, dédié à l'évaluation et à la validation de la robustesse, de la qualité, et de la performance de la partie serveur de notre application. Cette section est divisée en deux sous-sections distinctes, à savoir les Tests Fonctionnels et les Tests Unitaires. Les tests fonctionnels, réalisés avec l'outil Postman, se concentrent sur la validation des fonctionnalités exposées par nos API REST, simulant des interactions réelles pour garantir la cohérence des réponses et la stabilité globale de notre application. D'un autre côté, les tests unitaires, exploitant JUnit 5 et Mockito, se penchent sur l'évaluation des unités individuelles de code, comme les services et les classes utilitaires, pour assurer une qualité et une fiabilité approfondies au niveau du code. Cette section témoigne de notre engagement envers des pratiques rigoureuses d'assurance qualité, contribuant ainsi à la livraison d'une partie backend solide, performante, et conforme aux attentes fonctionnelles de notre application Spring Boot.

3.4.4.1 Tests fonctionnels

Les tests fonctionnels de la partie backend ont été réalisés avec Postman, un outil puissant dédié à l'évaluation du comportement de nos API REST exposées par l'application Spring Boot. Ces tests ont permis de valider la conformité des points de terminaison, d'assurer la stabilité des réponses, et de vérifier la logique métier globale de notre application. En utilisant Postman, nous avons pu simuler des scénarios d'utilisation réels, évaluant ainsi la cohérence des réponses dans des conditions proches de celles du monde réel. Ces tests fonctionnels ont été cruciaux pour garantir la fiabilité et la performance de notre backend, offrant une assurance qualité pour les fonctionnalités clés de notre application.



Figure 44 : Logo de Postman

Les tests fonctionnels de la partie backend ont impliqué l'utilisation de quatre méthodes HTTP principales GET, POST, PUT, et DELETE. Chaque méthode a été déployée stratégiquement pour garantir la fonctionnalité, la sécurité et la stabilité de notre application Spring Boot.

Voici quelques exemples pour les tests de l'API de gestion des abonnés :

- ❖ **Méthode GET** : a été employée pour récupérer la liste des abonnés. Cette opération inclut une fonctionnalité de pagination pour optimiser les performances en limitant la quantité de données retournée à la fois. J'ai également mis en place des filtres de recherche, permettant aux utilisateurs de spécifier des critères afin de récupérer des ensembles spécifiques d'abonnés. Ces tests ont validé la conformité des points de terminaison GET, assuré la stabilité des réponses, et vérifié la performance de la récupération des données dans des scénarios réalistes.

```

http://localhost:8089/api/abonnes
http://localhost:8089/api/abonnes/1
http://localhost:8089/api/abonnes?search=nom&val=Zad&page=1&sort=nom,desc

```

Figure 45 : test de la méthode GET de l'api de gestion des abonnés

- ❖ **Méthode POST :** a été déployée pour la création d'entités telles que les abonnés, les contrats, et les offres. Cette opération a été soumise à des tests approfondis afin de garantir la stabilité et la cohérence du processus de création. Les tests ont vérifié la conformité des points de terminaison POST, assurant ainsi la fiabilité des données introduites dans notre application. En simulant des scénarios de création dans des conditions proches du monde réel, nous avons renforcé la qualité et la robustesse des fonctionnalités clés de notre application.

```

http://localhost:8089/api/abonnes
{
    "nom": "Saadani",
    "prenom": "Brahim",
    "email": "brahim@gmail.com",
    "adresse": "852, Av Gueliz, Marrakech",
    "telephone": "+212611111111",
    "sexe": "homme",
    "statut": "actif"
}

```

Figure 46 : test de la méthode POST de l'api de gestion des abonnés

- ❖ **Méthode PUT :** a été utilisée pour les opérations de modification des abonnés. Ces tests ont validé la conformité des points de terminaison PUT et ont assuré que les modifications apportées étaient traitées de manière sécurisée et cohérente. Les scénarios de test ont inclus des mises à jour de divers champs associés aux abonnés, évaluant ainsi la robustesse de notre logique métier lors de la modification d'informations existantes.

```
http://localhost:8089/api/abonnes/1
```

```
{  
    "nom": "SAADANI",  
    "prenom": "Brahim",  
    "email": "brahim@yahoo.fr",  
    "adresse": "852, Av Gueliz, Marrakech",  
    "telephone": "00212622222222",  
    "sexe": "HOMME",  
    "statut": "Suspendu",  
    "contratId": 1,  
    "agenceId": 1,  
    "backOfficeId": 1  
}
```



Figure 47 : test de la méthode PUT de l'api de gestion des abonnés

```
http://localhost:8089/api/abonnes/1/statut
```

```
"ACTIF"
```



Figure 48 : test de la méthode PUT pour changer le statut d'un abonné

```
http://localhost:8089/api/abonnes/2/agence/3
```



Figure 49 : test de la méthode PUT pour associer une agence à un abonné

- ❖ **Méthode DELETE** : été implémentée pour gérer les opérations de suppression d'entités telles que les abonnés, les contrats, et les offres. Ces tests ont été essentiels pour garantir une gestion sécurisée et fiable de la suppression des données dans notre application.

```
http://localhost:8089/api/abonnes
```



Figure 50 : test de la méthode DELETE pour supprimer tous les abonnés

```
http://localhost:8089/api/abonnes/1
```



Figure 51 : test de la méthode DELETE pour supprimer un abonné

3.4.4.2 Tests unitaires

Les tests unitaires constituent une étape cruciale dans le processus de développement pour garantir la qualité du code et la stabilité des fonctionnalités. Pour cette phase, j'ai opté pour l'utilisation de JUnit 4, un framework de test unitaire robuste pour Java, combiné avec Mockito, une bibliothèque de mocking puissante.

- ❖ **JUnit 4 :** a été sélectionné comme framework principal pour la réalisation des tests unitaires. Sa structure claire et sa flexibilité m'a permis d'organiser les tests de manière modulaire et de les exécuter efficacement. J'ai créé des classes de test distinctes pour chaque service clé de l'application, assurant ainsi une granularité appropriée pour les tests unitaires.



Figure 52 : Logo de JUnit 4

L'utilisation de JUnit 4 m'a également offert des fonctionnalités avancées telles que les assertions conditionnelles, les paramètres de test dynamiques et les extensions, améliorant la lisibilité des tests et la facilité de maintenance du code.

- ❖ **Mockito :** Pour garantir l'isolation des tests unitaires et simuler le comportement des dépendances, j'ai intégré Mockito à mon suite de tests. Mockito a permis de créer des mocks des dépendances externes (la couche Repository), éliminant la nécessité d'accéder à des ressources réelles telles que des bases de données pendant les tests unitaires.



Figure 53 : Logo de Mockito

Les mocks ont été utilisés pour simuler des réponses spécifiques et contrôler le comportement des dépendances, ce qui m'a permis de focaliser les tests sur des unités spécifiques de code sans les complications potentielles liées à des composants externes. Cette approche a non seulement accéléré l'exécution des tests, mais a également facilité la détection et la correction d'éventuelles erreurs d'implémentation.

En combinant JUnit 4 et Mockito, j'ai établi une suite de tests unitaires robuste qui a validé la logique interne des services, assurant ainsi la cohérence et la fiabilité de l'application à un niveau microscopique.

Voici une table qui illustre la couverture des tests de la couche service :

Package	Class, %	Method, %	Line, %
com.adria.ayoub.gestiondesabonnesebankingbackend.services.impl	100% (3/3)	91,7% (33/36)	96,9% (315/325)
Class	Class, %	Method, %	Line, %
AbonneServiceImpl	100% (1/1)	93,8% (15/16)	97,3% (181/186)
ContratServiceImpl	100% (1/1)	91,7% (11/12)	96,3% (103/107)
OffreServiceImpl	100% (1/1)	87,5% (7/8)	96,9% (31/32)

Table 4 : Résumé de la couverture globale des tests unitaires

3.5 Conclusion

Ce chapitre a été consacré à la présentation de l'environnement de travail du projet. J'ai présenté les différentes technologies que j'ai mis en place pour développer l'application. Dans le chapitre suivant, je présenterai la partie implémentation du système.

Chapitre 4

Mise en œuvre

Résumé

La partie de mise en œuvre se place à la fin du processus de développement, elle comporte communément le codage et le test de l'application. Le présent chapitre a donc pour but la description de la phase de réalisation du projet, ceci exige une éventuelle présentation de l'environnement de développement, ainsi que certaines prises d'écrans exposantes de quelques cas d'utilisations de l'application.

4.1 Page d'accueil

Voici un aperçu de la page d'accueil qui s'affiche à l'agent back office de l'agence, qui contient les liens vers tous ce qu'il peut faire dans l'application :

- ❖ Consulter les listes des abonnés
- ❖ Consulter les listes des contrats d'abonnement
- ❖ Consulter les listes des offres commerciales
- ❖ Ajout des abonnés
- ❖ Ajout des contrats
- ❖ Ajout des offres

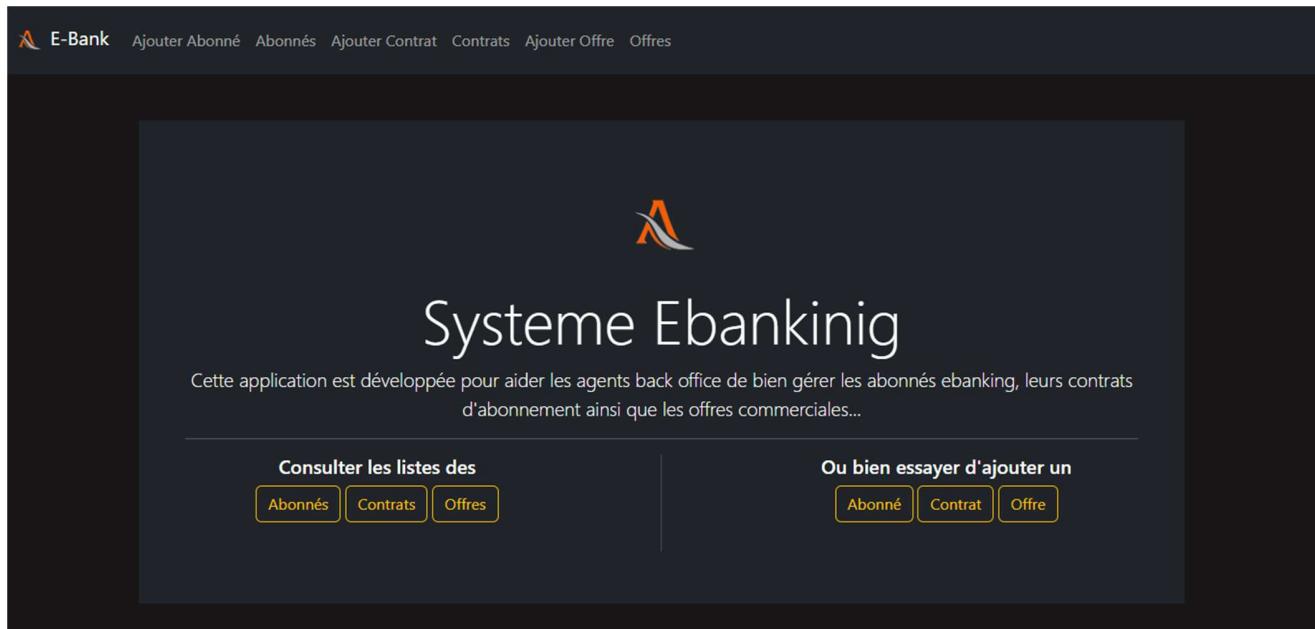


Figure 54 : Page d'accueil

4.2 Page d'erreur

Cette page doit être affichée quand le back office a entré un chemin qui ne contient aucune contenu, ou une page inexiste.

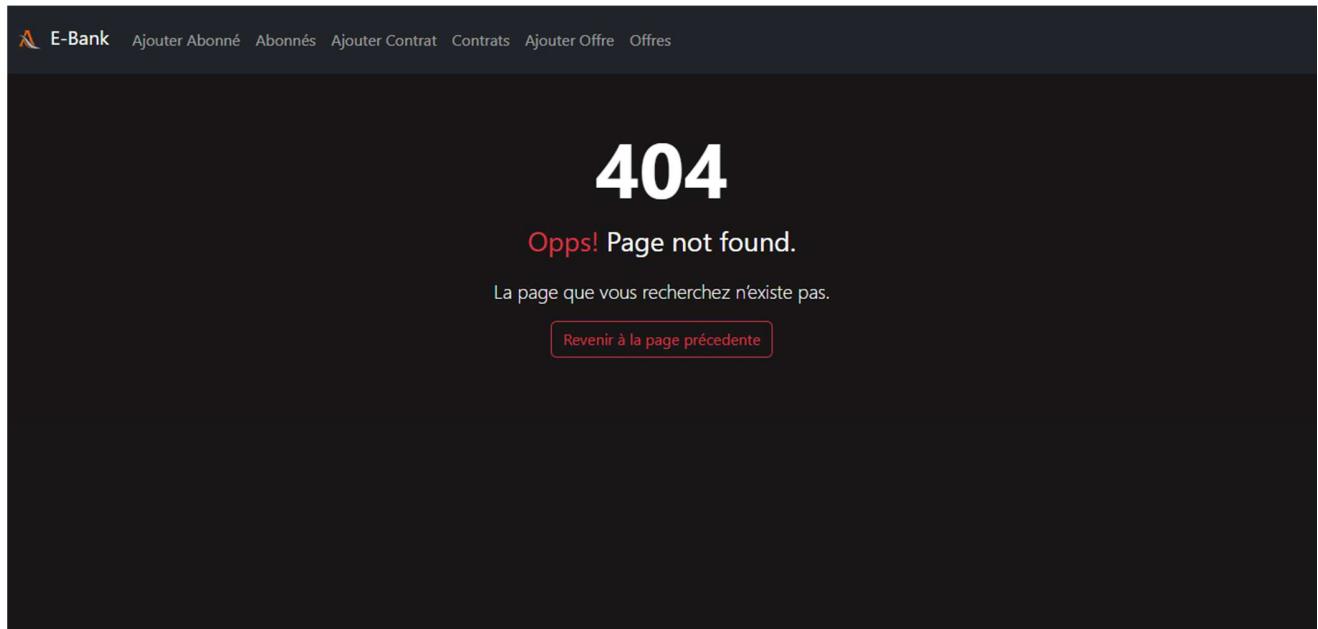


Figure 55 : Page d'erreur

4.3 Gestion des abonnés

4.3.1 Ajout d'un abonné

A screenshot of a web form titled 'Ajouter un Abonné'. The form has several input fields: 'Nom' (with placeholder 'Le nom de l'abonné'), 'Prenom' (with placeholder 'Le prenom de l'abonné'), 'Email' (with placeholder 'L'email de l'abonné'), 'Numéro de Téléphone' (with placeholder 'Le numéro de téléphone de l'abonné'), 'Sexe' (with placeholder 'Choisir le sexe de l'abonné'), 'Statut' (with placeholder 'Choisir le statut de l'abonné'), 'Adresse' (with placeholder 'L'adresse de l'abonné'), 'Contrat' (with placeholder 'Choisir le contrat à associer à cet abonné'), 'Agence' (with placeholder 'Choisir l'agence à associer à cet abonné'), and 'Back Office' (with placeholder 'Choisir le back office à associer à cet abonné'). At the bottom right are two buttons: 'Réinitialiser' and 'Enregistrer'.

Figure 56 : Ajout d'un abonné

Pour l'ajout d'un abonné, cette interface est utilisée, elle contient les champs suivants :

- ❖ Nom de l'abonné
- ❖ Prénom
- ❖ Email qui doit être unique
- ❖ Numéro de Téléphone (seulement Marocaine)
- ❖ Sexe (deux valeurs possibles, Homme ou Femme)
- ❖ Statut de l'abonné (Actif ou Suspendu)
- ❖ Adresse actuelle de l'abonné
- ❖ Le contrat associé à cet abonné (peut être laissé nulle pour le moment avant d'ajouter un contrat)
- ❖ Agence associé (pour cet exemple, y a que trois agences)
- ❖ Back Office associé (c'est l'agent de la banque qui est en train de saisir les informations)

4.3.2 Lister des abonnés

4.3.2.1 Lister tous les abonnés

Ici le back office peut lister tous les abonnés (dans des pages de 10 utilisateurs par page pour ne pas déranger la base de données pour lister tous les abonnés à la fois) pour bien les gérer, il y a aussi la possibilité d'organiser les abonnés de la liste selon l'ordre des lettres de l'alphabet :

ID	NOM	PRENOM	EMAIL	TEL	SEXЕ	STATUT	CONTRAT	ADRESSE	AGENCE	BACKOFFICE
1	TALBI	Ayoub	ayoub@gmail.com	0639325687	HOMME	ACTIF	Contrat 1	Marrakech	MARRAKECH ABDELRIM EL KHATABI	Ei Amrani Amine
2	AMRANI	Khaled	khaled@gmail.com	069987986	HOMME	ACTIF	Contrat 3	Casablanca	AIN CHOCK	Moussaoui Leila
3	Saadani	Khadija	khadija@gmail.com	0669835212	FEMME	ACTIF	Contrat 2	Agadir	AGADIR DRARGA	Benjelloun Youssef
4	Moussaoui	Adil	adil@gmail.com	0650236985	HOMME	ACTIF	Contrat 4	Casablanca	AIN CHOCK	Moussaoui Leila

Figure 57 : Lister tous les abonnés

4.3.2.2 Chercher des abonnés en utilisant la barre de recherche

Ici le back office a la possibilité de faire des recherches par champ, il doit d'abord choisir par quel champ il veut effectuer la rechercher. Je veux mentionner que la recherche faite automatiquement quand il saisit quelque chose, sans cliquer sur un bouton de recherche.

The screenshot shows a web browser window with the URL `localhost:42791/abonnes`. The page title is "E-Bank". A search bar at the top contains the placeholder "Chercher...". Below the search bar is a table header with columns: ID, NOM, PRENOM, EMAIL, TEL, SEXE, STATUT, CONTRAT, ADRESSE, AGENCE, BACKOFFICE. The table body contains four rows of data. At the bottom of the table, it says "Totale des abonnés : 4 - Page 1/1." and includes navigation buttons.

ID	NOM	PRENOM	EMAIL	TEL	SEXЕ	STATUT	CONTRAT	ADRESSE	AGENCE	BACKOFFICE
1	Amrani	Ayoub	ayoub@gmail.com	0639325687	HOMME	ACTIF	Contrat 1	Marrakech	MARRAKECH ABDELRIM EL KHATABI	<input checked="" type="checkbox"/> <input type="checkbox"/>
2	Saadani	Khaled	khaled@gmail.com	0698987986	HOMME	ACTIF	Contrat 3	Casablanca	AIN CHOCK	<input checked="" type="checkbox"/> <input type="checkbox"/>
3	Saadani	Khadija	khadja@gmail.com	0669835212	FEMME	ACTIF	Contrat 2	Agadir	AGADIR DRARGA	<input checked="" type="checkbox"/> <input type="checkbox"/>
4	Moussaoui	Adil	adil@gmail.com	0658236985	HOMME	ACTIF	Contrat 4	Casablanca	AIN CHOCK	<input checked="" type="checkbox"/> <input type="checkbox"/>

Figure 58 : Chercher des abonnés en utilisant la barre de recherche

The screenshot shows a web browser window with the URL `localhost:42791/abonnes`. The search bar now contains "kh". The table header and body are identical to Figure 58, but only two rows of data are visible, corresponding to the search results for "kh".

ID	NOM	PRENOM	EMAIL	TEL	SEXЕ	STATUT	CONTRAT	ADRESSE	AGENCE	BACKOFFICE
2	AMRANI	Khaled	khaled@gmail.com	0698987986	HOMME	ACTIF	Contrat 3	Casablanca	AIN CHOCK	<input checked="" type="checkbox"/> <input type="checkbox"/>
3	Saadani	Khadija	khadja@gmail.com	0669835212	FEMME	ACTIF	Contrat 2	Agadir	AGADIR DRARGA	<input checked="" type="checkbox"/> <input type="checkbox"/>

Figure 59 : Chercher des abonnés par email

4.3.3 Modifier les données d'un abonné

Le back office peut modifier les données d'un utilisateur :

E-Bank Ajouter Abonné Abonnés Ajouter Contrat Contrats Ajouter Offre Offres

Modifier l'abonné de l'id : 1

Nom	Prenom	Email
TALBI	Ayoub	ayoub@gmail.com
Numéro de Téléphone	Sexe	Statut
0639325687	Homme	Suspendu
Adresse		
Marrakech		
Agence	Back Office	
MARRAKECH ABDELRIM EL KHATABI	El Amrani Amine	

Réinitialiser Enregistrer les modifications

Figure 60 : Modifier les données d'un abonné

4.3.4 Suppression d'un abonné

Le back office peut supprimer un abonné, mais d'abord une alerte est affichée pour confirmer la suppression :

E-Bank Ajouter Abonné Abonnés Ajouter Contrat Contrats Ajouter Offre Offres

NOM Chercher...

La Liste des Abonnés

ID	NOM	PRENOM	EMAIL	SEXE	STATUT	ADRESSE	AGENCE	BACKOFFICE
1	TALBI	Ayoub	ayoub@gn			Marrakech	MARRAKECH ABDELRIM EL KHATABI	El Amrani Amine
2	AMRANI	Khaled	khaled@gu			isablanca	AIN CHOCK	Moussaoui Leila
3	Saadani	Khadija	khadija@g			jadir	AGADIR DRARGA	Benjelloun Youssef
4	Moussaoui	Adil	adil@gmai			isablanca	AIN CHOCK	Moussaoui Leila

Total des abonnés : 4 - Page 1/1.

Voulez-vous vraiment supprimer cet abonné ?

ID : 2
NOM : AMRANI
PRENOM : Khaled

Oui, supprimer l'abonné définitivement Annuler

Figure 61 : Suppression d'un abonné

4.4 Gestion des contrats d'abonnement

4.4.1 Ajout d'un contrat d'abonnement

Pour l'ajout d'un contrat d'abonnement, cette interface est utilisée, elle contient les champs suivants :

- ❖ Intitulé du contrat
- ❖ Statut du contrat (Actif ou Suspendu)
- ❖ L'abonné associé à ce contrat (peut être laissé nulle pour le moment avant d'ajouter un abonné)
- ❖ Les offres commerciales à ajouter

The screenshot shows a dark-themed web application window. At the top, there's a header bar with icons for back, forward, search, and refresh, followed by the URL 'localhost:42791/ajouterContrat'. Below the header is a navigation menu with links: E-Bank, Ajouter Abonné, Abonnés, Ajouter Contrat, Contrats, Ajouter Offre, and Offres. The main content area is titled 'Ajouter un Contrat'. It contains several input fields:

- Intitulé:** A text input field labeled 'L'intitulé du contrat'.
- Statut:** A dropdown menu labeled 'Choisir le statut du contrat'.
- Abonne:** A dropdown menu labeled 'Choisir l'abonné à associer à cet contrat'.
- Offres:** A dropdown menu labeled 'Ajouter un ou plusieurs offres'.

At the bottom right of the form are two buttons: 'Réinitialiser' (Reset) and 'Enregistrer' (Save).

Figure 62 : Ajout d'un contrat d'abonnement

4.4.2 Lister tous les contrats

4.4.2.1 Lister tous les contrats

Ici le back office peut lister tous les contrats d'abonnement (dans des pages de dix contrats par page pour ne pas déranger la base de données pour lister tous les contrats à la fois), il y a aussi la possibilité d'organiser les contrats de la liste selon l'ordre des lettres de l'alphabet :

ID	INTITULE	STATUT	OFFRES
1	Contrat 1	ACTIF	Offre 1, Offre 3, Offre 8, Offre 10 <input checked="" type="checkbox"/> <input type="checkbox"/>
2	Contrat 2	SUSPENDU	Offre 1, Offre 4, Offre 6, Offre 7, Offre 8, Offre 9 <input checked="" type="checkbox"/> <input type="checkbox"/>
3	Contrat 3	ACTIF	Offre 1, Offre 2, Offre 3, Offre 4, Offre 5, Offre 6, Offre 7 <input checked="" type="checkbox"/> <input type="checkbox"/>
4	Contrat 4	ACTIF	Offre 1, Offre 2, Offre 13, Offre 14 <input checked="" type="checkbox"/> <input type="checkbox"/>

Figure 63 : Lister tous les contrats

4.4.2.2 Chercher les contrats en utilisant la barre de recherche

Ici le back office a la possibilité de faire des recherches par champ pour les contrats, il doit d'abord choisir par quel champ il veut effectuer la recherche (Intitulé ou Statut). Je veux mentionner que la recherche faite automatiquement quand il saisit quelque chose, sans cliquer sur un bouton de recherche.

ID	INTITULE	STATUT	OFFRES
2	Contrat 2	SUSPENDU	Offre 1, Offre 4, Offre 6, Offre 7, Offre 8, Offre 9 <input checked="" type="checkbox"/> <input type="checkbox"/>

Figure 64 : Chercher les contrats en utilisant la barre de recherche

4.4.3 Modifier les données d'un contrat

Le back office peut modifier les données d'un contrat :

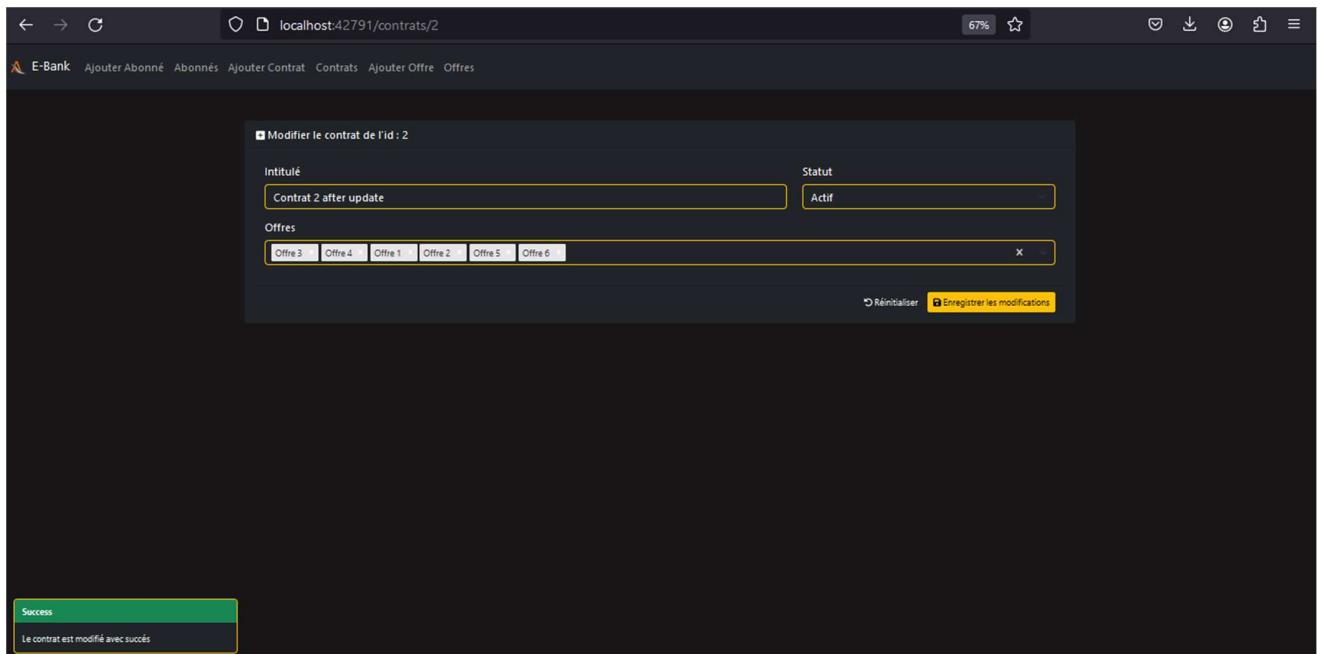


Figure 65 : Modifier les données d'un contrat

4.4.4 Suppression d'un contrat

Le back office peut supprimer un contrat d'abonnement, mais d'abord une alerte sera être affichée pour confirmer la suppression :

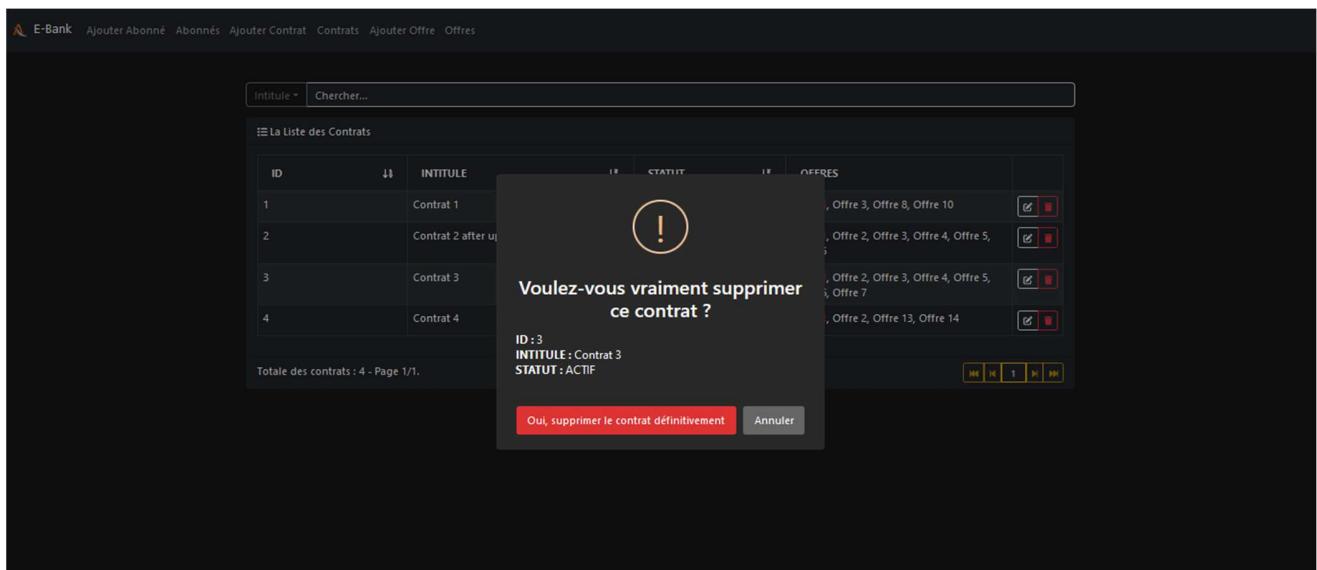


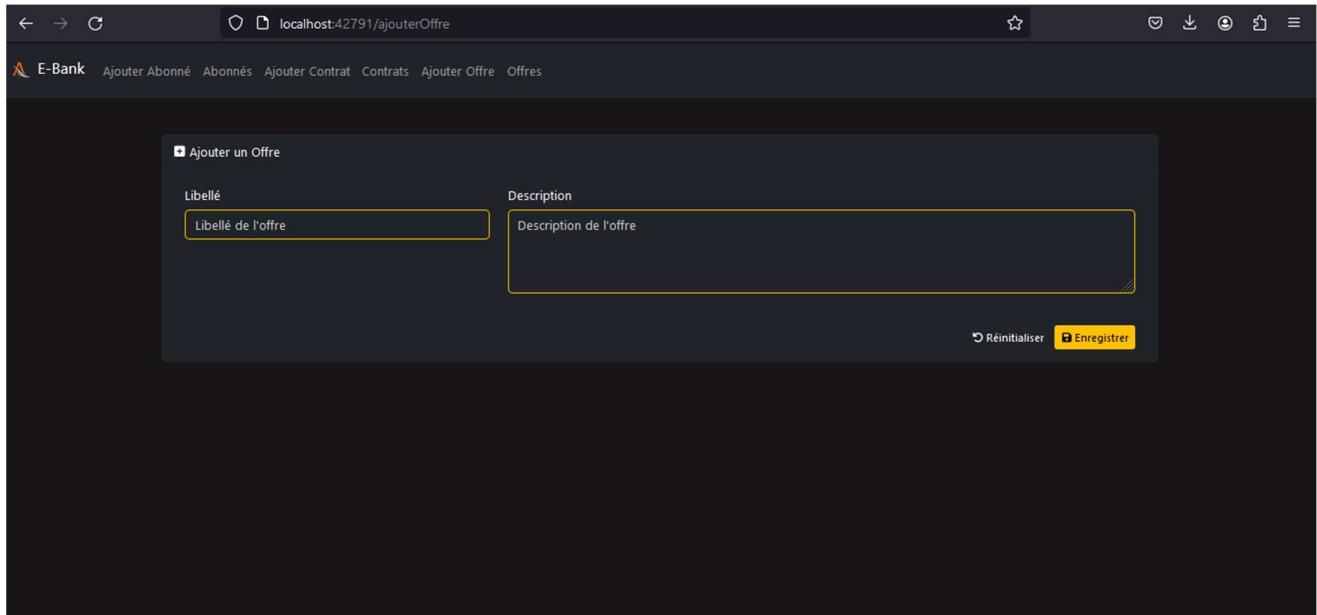
Figure 66 : Suppression d'un contrat

4.5 Gestion des offres commerciales

4.5.1 Ajout d'une offre commerciale

Pour l'ajout d'une offre commerciale, cette interface est utilisée, elle contient les champs suivants :

- ❖ Libellé de l'offre
- ❖ Sa description



The screenshot shows a dark-themed web application window. At the top, the address bar displays "localhost:42791/ajouterOffre". The main content area is titled "Ajouter un Offre". It contains two input fields: "Libellé" (labeled "Libellé de l'offre") and "Description" (labeled "Description de l'offre"). At the bottom right of the form are two buttons: "Réinitialiser" and "Enregistrer".

Figure 67 : Ajout d'une offre commerciale

4.5.2 Lister les offres commerciales

4.5.2.1 Lister tous les offres

Ici le back office peut lister tous les contrats d'abonnement (dans des pages de dix contrats par page pour ne pas déranger la base de données pour lister tous les contrats à la fois), il y a aussi la possibilité d'organiser les contrats de la liste selon l'ordre des lettres de l'alphabet :

ID	LIBELLE	DESCRIPTION	
1	Offre 1	Description de l'offre 1	<input checked="" type="checkbox"/> <input type="checkbox"/>
2	Offre 2	Description de l'offre 2	<input checked="" type="checkbox"/> <input type="checkbox"/>
3	Offre 3	Description de l'offre 3	<input checked="" type="checkbox"/> <input type="checkbox"/>
4	Description de l'offre 4	Description de l'offre 4	<input checked="" type="checkbox"/> <input type="checkbox"/>
5	Offre 5	Description de l'offre 5	<input checked="" type="checkbox"/> <input type="checkbox"/>
6	Offre 6	Description de l'offre 6	<input checked="" type="checkbox"/> <input type="checkbox"/>
7	Offre 7	Description de l'offre 7	<input checked="" type="checkbox"/> <input type="checkbox"/>
8	Offre 8	Description de l'offre 8	<input checked="" type="checkbox"/> <input type="checkbox"/>
9	Offre 9	Description de l'offre 9	<input checked="" type="checkbox"/> <input type="checkbox"/>
10	Offre 10	Description de l'offre 10	<input checked="" type="checkbox"/> <input type="checkbox"/>

Totale des offres : 15 - Page 1/2.

Figure 68 : Lister tous les offres

4.5.2.2 Naviguer entre les pages des offres

Ce n'est pas seulement pour les offres, mais aussi pour les abonnés et les contrats, ici le back office peut naviguer entre les pages des offres (chaque page contient dix offres) :

ID	LIBELLE	DESCRIPTION	
11	Offre 11	Description de l'offre 11	<input checked="" type="checkbox"/> <input type="checkbox"/>
12	Offre 12	Description de l'offre 12	<input checked="" type="checkbox"/> <input type="checkbox"/>
13	Offre 13	Description de l'offre 13	<input checked="" type="checkbox"/> <input type="checkbox"/>
14	Offre 14	Description de l'offre 14	<input checked="" type="checkbox"/> <input type="checkbox"/>
15	Offre 15	Description de l'offre 15	<input checked="" type="checkbox"/> <input type="checkbox"/>

Totale des offres : 15 - Page 2/2.

Figure 69 : Naviguer entre les pages des offres

4.5.2.3 Chercher des offres en utilisant la barre de recherche

Ici le back office a la possibilité de faire des recherches par champ pour les offres, il doit d'abord choisir par quel champ il veut effectuer la recherche (Libellé ou Description).

The screenshot shows a dark-themed web application interface for 'E-Bank'. At the top, there is a navigation bar with links: Ajouter Abonné, Abonnés, Ajouter Contrat, Contrats, Ajouter Offre, and Offres. Below the navigation, a search bar contains the text 'Description' followed by the number '3'. A modal window titled 'La Liste des Offres' is displayed, showing a table with two rows of data. The columns are labeled 'ID', 'LIBELLE', and 'DESCRIPTION'. Row 1 has ID 3, Libellé 'Offre 3', and Description 'Description de l'offre 3'. Row 2 has ID 13, Libellé 'Offre 13', and Description 'Description de l'offre 13'. Each row has edit and delete icons. Below the table, a message says 'Totale des offres : 2 - Page 1/1.' with a page navigation bar.

Figure 70 : Chercher des offres en utilisant la barre de recherche

4.5.3 Modifier les données d'une offre

Le back office peut modifier les données d'une offre :

The screenshot shows a browser window with the URL 'localhost:42791/offres/4'. The page title is 'E-Bank'. The main content area is titled 'Modifier l'offre de l'id :'. It contains two input fields: 'Libellé' with the value 'Offre 4' and 'Description' with the value 'Description de l'offre 4 after update'. At the bottom are two buttons: 'Réinitialiser' and 'Enregistrer les modifications'. A green 'Success' bar at the bottom left indicates 'L'offre est modifiée avec succès'.

Figure 71 : Modifier les données d'une offre

4.5.4 Suppression d'une offre

Le back office peut supprimer une offre commerciale, mais d'abord une alerte sera être affichée pour confirmer la suppression :

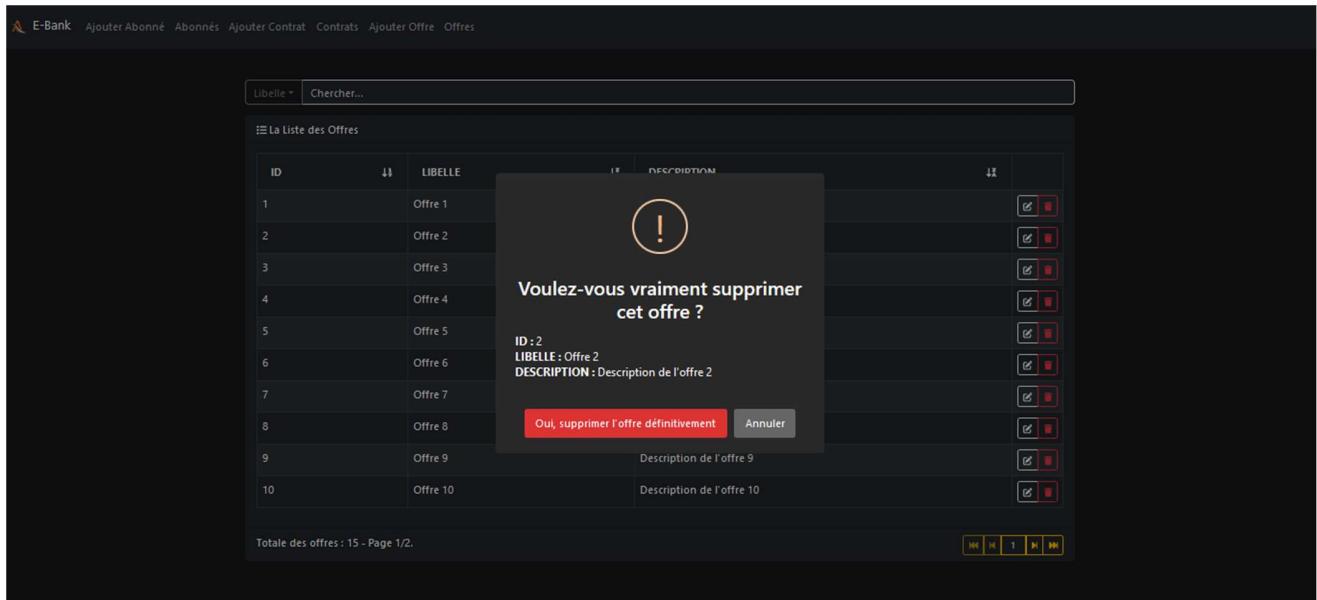


Figure 72 : Suppression d'une offre

4.6 Conclusion

Ce chapitre était consacré à la navigation sur la plateforme. Il a pour objectif de décrire les principales fonctionnalités implémentées suite au développement par des captures d'écran témoignant les différentes interfaces que contient le système.

Conclusion Générale

Au terme de ce projet de fin d'année au sein de la société Adria Business & Technology, la conception et le développement de l'application de Back Office dédiée à la gestion des abonnés e-banking, de leurs contrats d'abonnement, ainsi que des offres commerciales, ont constitué une expérience enrichissante et formatrice.

La démarche entreprise a débuté par une phase d'étude approfondie des besoins, permettant une identification claire des exigences fonctionnelles. La formalisation de ces dernières a été réalisée à travers une modélisation systématique du système à l'aide du langage UML, définissant avec précision les acteurs impliqués et les scénarios fonctionnels attendus.

L'étude conceptuelle a constitué une étape cruciale dans le processus, se matérialisant par la création de diagrammes de classes et de séquences, offrant ainsi une vision détaillée de l'architecture du système. Par la suite, une étude technique approfondie a été entreprise, mettant en lumière l'ensemble des choix architecturaux et des technologies adoptées pour la réalisation du projet.

Dans la phase de réalisation, toutes les spécifications émises par l'encadrante professionnelle ont été méticuleusement mises en œuvre, garantissant ainsi le bon fonctionnement de la gestion des abonnés, des contrats d'abonnement et des offres commerciales.

Ce stage m'a permis d'appliquer concrètement les connaissances acquises au cours de ma formation, mais également de développer de nouvelles compétences techniques.