



Mémoire de Projet de Fin d'Études

En vue de l'obtention du diplôme

Diplôme d'Ingénieur d'État en Informatique

Filière : Ingénierie en Data Science et IoT

Développement d'un Système de Reconnaissance Vocale Pour l'Accessibilité des Personnes Non-Voyantes Dans Le eBanking

Effectué chez *Attijariwafa bank*

12 Février 2024 - 12 Juin 2024

Réalisé par :

Chaimaâ OURGANI

Encadré par :

Pr. Mohamed RADOUANE

M. Abdellah BERKAOUI

Soutenu le xx XX 2024, Devant le jury composé de :

Pr. Membre Du JURY 1 : ENSIAS - Président(e)

Pr. Mohamed RADOUANE : ENSIAS - Encadrant académique

M. Abdellah BERKAOUI : Attijariwafa Bank - Encadrant professionnel

Pr. Membre Du JURY 1 : ENSIAS - Examinateur/trice

Année Universitaire : 2023/2024

Dédicace

“

À mes parents, qui m'ont toujours encouragé à poursuivre mes rêves et à donner le meilleur de moi-même. Leur amour et leur soutien ont été ma source de motivation tout au long de mon parcours. Je leur dédie ce PFE en guise de gratitude pour tous les sacrifices qu'ils ont consentis à mon égard.

À ma soeur, pour son amour inconditionnel et son soutien indéfectible. Tu es ma source d'inspiration et ma motivation. Je te dédie ce PFE en guise de remerciement pour ta présence constante à mes côtés.

Je dédie également ce travail à mes meilleures amies Imane et Hajar et à mes chers amis Yousra, Ismail, Sofia et Nora pour leur amitié indéfectible et leur soutien constant. Vous avez rendu ce parcours plus agréable et enrichissant. Je vous dédie ce PFE en guise de remerciement pour votre présence dans ma vie.

À mes encadrants et professeurs, pour leurs précieux conseils et leur soutien constant. Je leur dédie ce PFE en reconnaissance de leur contribution à mon développement personnel et professionnel.

À tous les enseignants qui ont marqué mon parcours, je ne saurais point vous remercier comme il se doit, que ce travail soit le fruit de vos efforts,

Et à tous ceux qui de près ou de loin m'ont soutenu de jour comme de nuit

”

Chaimaa Ourgani

Remerciements

Tout d'abord, je remercie Dieu le Tout-Puissant de m'avoir donné le courage et la patience pour mener à bien ce travail et surmonter toutes les difficultés. Avant de présenter mon travail, je tiens à exprimer ma grande reconnaissance envers toutes les personnes qui m'ont apporté leur soutien, que ce soit de près ou de loin. Qu'ils trouvent ici collectivement et individuellement l'expression de toute ma gratitude.

J'adresse mes sincères remerciements au Professeur **Radouane Mohammed**, qui a su animer et orienter mon travail à partir de ses notes pertinentes, sa gentillesse et sa disponibilité à tout moment. Je tiens également à remercier chaleureusement **M. Abdellah Berkaoui** pour son soutien , son temps et sa précieuse contribution à mon développement professionnel. Je suis également reconnaissante envers toute l'équipe d'Attijariwafa bank pour leur dynamisme et leur partage d'expériences enrichissantes. Enfin, je remercie les membres du jury pour l'acceptation de mon travail pour évaluation et les professeurs de l'École Nationale d'Informatique et d'Analyse des Systèmes (ENSIAS) pour leur engagement envers l'amélioration de la qualité de l'éducation.

Enfin, je témoigne de mon profond respect et de ma gratitude envers tous ceux qui ont contribué à la réalisation de ce travail, en espérant qu'ils y trouveront l'expression de ma considération et le témoignage de mon estime.

Résumé

Le présent document synthétise mon expérience de stage chez Attijariwafa Bank, où j'ai contribué à un projet novateur : l'élaboration d'un système d'assistance vocale pour améliorer l'accessibilité de l'application eBanking pour les non-voyants.

Ce système de reconnaissance vocale (RV) vise à permettre une interaction fluide avec l'application bancaire uniquement par la voix, répondant ainsi aux besoins des utilisateurs aveugles ou malvoyants. Nous avons dû assurer la sécurité des données et garantir la précision de la reconnaissance vocale. La gestion des erreurs de reconnaissance a été un défi, tout en respectant des contraintes telles que le temps de développement réduit, les coûts maîtrisés et l'intégration fluide avec les systèmes existants. Nous avons également veillé à garantir une haute qualité de la synthèse vocale et une faible latence.

Pour répondre à ces exigences, nous avons opté pour une architecture intégrant des technologies spécifiques, notamment le développement d'une application mobile avec une sécurité biométrique. Le processus de développement s'est articulé autour de plusieurs phases, de l'analyse des besoins utilisateurs à la phase de déploiement et de documentation. Chaque phase comprenait des étapes telles que la collecte et le prétraitement des données, le développement du modèle de reconnaissance du locuteur, le développement des modèles NLP, le développement mobile et de l'API bancaire, l'intégration des différents composants et l'amélioration des interfaces utilisateur.

Le rapport débute par une présentation de l'entreprise et du contexte du projet, puis explore en détail l'état de l'art ,les choix technologiques effectués , ainsi que la modélisation et conception. Ensuite, il décrit la mise en œuvre du système biométrique, du système du traitement de langage naturel, du développement mobile et de l'API bancaire, ainsi que l'intégration des différents composants. Pour terminer, le rapport offre une évaluation complète des résultats obtenus, des défis rencontrés et propose des perspectives d'amélioration futures.

Mots-clés : Reconnaissance du locuteur, Reconnaissance vocale, NLP, NLU, Apprentissage profond, Apprentissage automatique, Apprentissage par transfert, Réseau de neurones siamois, RESTful API, UML, Reconnaissance d'intention, Chatbot conversationnel.

Abstract

This document summarizes my internship experience at Attijariwafa Bank, where I contributed to an innovative project : the development of a voice assistance system to improve accessibility to the eBanking application for the visually impaired people.

This voice recognition (VR) system aims to enable seamless interaction with the banking application solely through voice commands, thus meeting the needs of blind or visually impaired users. We had to ensure data security and accurate voice recognition. Handling recognition errors was a challenge while adhering to constraints such as reduced development time, controlled costs, and smooth integration with existing systems. We also aimed to ensure high-quality voice synthesis and low latency.

To meet these requirements, we opted for an architecture integrating specific technologies, including the development of a mobile application with biometric security. The development process was structured around several phases, from user needs analysis to deployment and documentation. Each phase included steps such as data collection and preprocessing, the development of the speaker recognition system, natural language processing model development, mobile development and the banking API, integration of various components, and improvement of user interfaces.

The report begins with an introduction to the company and the project context, then explores the state-of-the-art and technological choices in detail. Next, it describes the implementation of the biometric system, the NLP system, mobile development, and the banking API, as well as the integration of the different components. Finally, the report provides a comprehensive evaluation of the results obtained, the challenges encountered, and proposes future improvement perspectives.

Keywords : Speaker Recognition, Speech To Text, NLP, NLU, Deep Learning, Machine Learning, Transfer Learning, Siamese Neural Network, RESTful API, UML, Intent Recognition, Conversational Chatbot.

Liste des abréviations

Abréviation	Désignation
IA	Intelligence Artificielle
EDA	Exploratory Data Analysis
LLM	Large Language Model
SVM	Support Vector Machine
XGBOOST	eXtreme Gradient Boosting
RF	Random forest
RNN	Réseaux de Neurones Récurrents
LSTM	Long-Short Term Memory
GRU	Gradient Recurrent Unit
BERT	Bidirectionnel Encoder Representations from Transformers
NLU	Natural Language Understanding
NLP	Natural Language Processing
NER	Named Entity Recognition
STT	Speech-to-Text
TTS	Text-To-Speech
SR	Speaker Recognition
FAQ	Frequently asked questions
MVC	Model View Controller
UI	User Interface
UML	Unified Modelling Language
SGBD	Système de Gestion des Bases de Données
JSON	JavaScript Object Notation

Table des figures

1	Logo Attijariwafa Bank	4
2	Actionnariat et Site web : d'Attijariwafa Bank https://www.attijariwafabank.com/fr	
3	Filiales d'AWB au Maroc	6
4	Filiales d'AWB à l'international	7
5	Organigramme AWB	8
6	Marques du Groupe	10
7	Valeurs de AWB	11
8	Présence mondiale de AWB	12
9	Méthodes d'accès au monde numérique pour les non-voyants	13
10	Logos des concurrents clés	16
11	Cycle de vie du projet	19
12	Diagramme de GANTT	21
13	Étude de l'état de l'art : SVM	26
14	Étude de l'état de l'art : Random Forest	26
15	Étude de l'état de l'art : XGBOOST	27
16	Étude de l'état de l'art : Architecture du CNN	28
17	Étude de l'état de l'art : Architecture des RNNs	29
18	État de l'art : Architecture LSTM	30
19	Étude de l'état de l'art : Architecture CamemBERT	31
20	Solution Théorique Globale	33
21	Architecture globale du système : Approche basée sur des APIs uniquement	34
22	Architecture globale du système : Approche basée uniquement sur des modèles développés en interne	35
23	Architecture globale du système vocal : Approche hybride	35
24	Solution Pratique Proposée	37
25	Logo Google Colab	37
26	Logo PyTorch et Tensorflow	38
27	Logos des bibliothèques pour la manipulation des données	39
28	Logos des bibliothèques de la visualisation de données	39
29	Logos des bibliothèques NLP	40
30	Logos des bibliothèques pour la pipeline FAQ	41
31	Logos des bibliothèques pour entraînement des modèles	41
32	Logo de bibliothèque pour traitement des signaux vocaux	42
33	Logo Postgresql	43
34	Logo Spring Boot	44
35	Logo IntelliJ IDEA	46
36	Logo Flutter	46
37	Logo Android Studio	47
38	Logo FastAPI	48

39	Logo PyCharm	49
40	Logo JUnit-Mockito	50
41	Logo Mocktail Flutter	51
42	Logo PyTest	51
43	Logo Postman	52
44	Logo Swagger	52
45	Diagramme des cas d'utilisation	57
46	Diagramme de séquence du cas d'utilisation : « Accès à l'assistance et à la FAQS »	62
47	Diagramme de séquence du cas d'utilisation : « Transaction Virement »	64
48	Diagramme de séquence du cas d'utilisation : « Consultation des Infos de comptes, de cartes et des opérations »	65
49	Diagramme de Classe	68
50	Diagramme d'activité du système	69
51	Diagramme d'activité du volet sécurité	70
52	Diagramme d'activité du volet NLP	71
53	Logo de la plateforme Figma	72
54	Prototype de l'application	72
55	Processus d'authentification via la biométrie vocale	77
56	Acquisition du dataset d'audios : LibriSpeech	78
57	Distribution de la longueur des audios dans le Dataset : LibriSpeech	78
58	Mesures statistiques de la distribution des audios par locuteur : LibriSpeech	79
59	Histogramme de la distribution du nombre d'audios par locuteur : LibriSpeech	79
60	Boîte à moustaches de la distribution des fichiers audio par locuteur : Li- briSpeech	80
61	Distribution des fichiers audio par genre : LibriSpeech	80
62	Distribution de la longueur des audios par locuteur : LibriSpeech	81
63	Formes d'ondes de certains échantillons d'audios : LibriSpeech	82
64	Spectrogrammes de certains échantillons d'audios : LibriSpeech	82
65	Les coefficients MFCC de certains échantillons d'audios	83
66	Gestion des valeurs manquantes : LibriSpeech	83
67	Gestion des doublons : LibriSpeech	84
68	Équilibrage des audios par locuteur :LibriSpeech	84
69	Visualisation de la forme d'onde d'un audio prétraité : LibriSpeech	85
70	Visualisation du spectrogramme de l'audio prétraité : LibriSpeech	85
71	Visualisation du MFCC de l'audio prétraité : LibriSpeech	86
72	Graphique de distribution des paires positives et négatives : LibriSpeech	86
73	Architecture du modèle Siamois pour la biométrie vocale	87
74	Les courbes d'apprentissage du modèle siamois pour la biométrie vocale	89
75	Rapport de classification et métriques de performances du modèle siamois pour la biométrie vocale	90

76	Matrice de confusion du modèle siamois	91
77	Courbe ROC du modèle siamois	91
78	Taille du dataset	94
79	Distribution des mots par phrase	95
80	Distribution du nombre de phrases par classe	95
81	Répartition des intentions	96
82	Distribution de la longueur des phrases par classe	97
83	Nuage de mots fréquents par classe	97
84	Nettoyage du dataset	98
85	Visualisation du nombre d'échantillons par classe avant et après l'oversampling	98
86	Architecture du modèle CamemBERT-RNN pour la reconnaissance d'intention	101
87	Architecture du modèle CamemBERT avec le classificateur par défaut pour la reconnaissance des intentions	102
88	Architecture du modèle CamemBERT avec le classificateur personnalisé pour la reconnaissance des intentions	102
89	Les courbes d'apprentissage de CamemBERT-LSTM	104
90	Les courbes d'apprentissage de CamemBERT-GRU	105
91	Les courbes d'apprentissage du fine-tuning de CamemBERT avec le classificateur par défaut	106
92	Les courbes d'apprentissage du fine-tuning de CamemBERT avec le classificateur personnalisé	106
93	Rapport de classification et métriques de performances du meilleur modèle	109
94	Matrice de confusion du meilleur modèle	110
95	Architecture de la pipeline de la gestion des FAQ	112
96	Architecture Logicielle API Bancaire	116
97	Structure du projet de l'API bancaire	117
98	Tests unitaires et d'intégrations de l'API Spring Boot	119
99	Test fonctionnel de l'API Bancaire	119
100	Documentation de l'API Bancaire	120
101	Architecture logicielle de l'API NLP	121
102	Structure logicielle de l'API NLP	122
103	Tests unitaires et d'intégrations de l'API NLP	123
104	Test fonctionnel de l'API NLP	124
105	Documentation de l'API NLP à l'aide de Swagger	125
106	Architecture logicielle de l'API de la biométrie vocale	126
107	Structure du projet de l'API de la biométrie vocale	127
108	Test fonctionnel de l'API de biométrie vocale	128
109	Documentation de l'API de biométrie vocale	128
110	Architecture logicielle de l'application mobile	129

111	Struture du projet de l'application mobile	130
112	Implémentation des tests de l'application mobile	132
113	Interface d'accueil de l'assistant vocal	135
114	Interface d'authentification de l'assistant vocal	136
115	Interface principale de commandes de l'assistant vocal	137
116	Interface Google Maps	137

Liste des tableaux

1	Tableau récapitulatif de l'étude benchmarking des solutions des concurrents	17
2	Livrables pour chaque phase du projet	22
3	Comparaison des différentes approches d'architecture globale	36
4	Comparaison des différents Frameworks pour l'entraînement des modèles . .	38
5	Comparaison des différents SGBD disponibles	43
6	Comparaison des frameworks web disponibles pour l'API RESTful	44
7	Comparaison des environnements de développement pour API RESTful Spring Boot disponibles	45
8	Comparaison des Frameworks mobile disponibles	46
9	Comparaison des environnements de développement mobile disponibles . .	47
10	Comparaison des Frameworks de développement des APIs exploitant les modèles PyTorch	48
11	Comparaison des environnements de développement d'API FastAPI	49
12	Fiche descriptive du cas d'utilisation : Accès à l'Assistance et la FAQ . . .	57
13	Fiche descriptive du cas d'utilisation : Accès à la Géolocalisation des Agences	58
14	Fiche descriptive du cas d'utilisation : Consultation des Informations de Comptes, de Cartes et des Opérations	58
15	Fiche descriptive du cas d'utilisation : Gestion des Cartes et Leur Configu- ration	59
16	Fiche descriptive du cas d'utilisation : Paiement de Factures	59
17	Fiche descriptive du cas d'utilisation : Gestion des Bénéficiaires de Tran- sactions	60
18	Fiche descriptive du cas d'utilisation : Virement vers des Comptes Attija- riwafa et Confrères	60
19	Description des classes et opérations du système bancaire	66
20	Comparaison des différentes méthodes d'authentification	75
21	Meilleurs paramètres et score pour chaque modèle	103
22	Performance des différents modèles de classification	107
23	Performance du modèle NER	111

Table des matières

Dédicace	I
Remerciements	II
Résumé	III
Abstract	IV
Introduction générale	1
1 Organisme d'accueil et contexte général du projet	4
1.1 Introduction	4
1.2 Présentation de l'organisme	4
1.2.1 Attijariwafa Bank	4
1.2.2 Filiales de la société	5
1.2.3 Structure et organisation générale	7
1.2.4 Vision future du groupe :	9
1.2.5 Missions de AWB	9
1.2.6 Secteurs d'activités	9
1.2.7 Produits et services	10
1.2.8 Valeurs Fondamentales	10
1.2.9 Présence internationale	11
1.3 Présentation du projet	12
1.3.1 Problématique	12
1.3.2 Étude Benchmarking	13
1.3.3 Missions	17
1.3.4 Objectif du projet	18
1.4 Plannification	18
1.4.1 Cycle de vie du projet	18
1.4.2 Diagramme de Gantt	20
1.4.3 Les livrables	21
1.5 Conclusion	22
2 Étude de l'état de l'art	25
2.1 Introduction	25
2.2 État de l'art	25
2.2.1 Apprentissage automatique (Machine Learning)	25
2.2.2 Apprentissage profond (Deep Learning)	27
2.2.3 Apprentissage par Transfert (Transfer Learning)	32
2.2.4 Traitement du Langage Naturel (NLP)	32

2.3	Approche de Solution et Stratégie Technique : Exploration des Pistes et Méthodologies	33
2.4	Outils utilisés	37
2.4.1	Volet Data Science	37
2.4.2	Volet Développement Logiciel	42
2.4.3	Volet Test	50
2.5	Conclusion	52
3	Analyse et Conception	55
3.1	Introduction	55
3.2	Analyse de besoins et modélisation des cas d'utilisation	55
3.2.1	Capture des Besoins	55
3.2.2	Diagramme des cas d'utilisation	56
3.2.3	Descriptirion des cas d'utilisation	57
3.3	Diagrammes de séquence pour les cas d'utilisation clés	61
3.3.1	Diagramme de séquence cas d'utilisation : « Accès à l'assistance et à la FAQS »	61
3.3.2	Diagramme de séquence cas d'utilisation : « Transaction Virement »	62
3.3.3	Diagramme de séquence cas d'utilisation : « Consultation des Inofs de comptes, de cartes et des opérations »	64
3.4	Diagramme de classes	66
3.5	Diagrammes d'activité pour la conception du système	68
3.5.1	Diagramme d'activité : Système	68
3.5.2	Diagramme d'activité : Sécurité	69
3.5.3	Diagramme d'activité : Système NLP	70
3.6	Réalisation d'un protocole de l'application	72
3.7	Conclusion	73
4	Stratégies d'Authentification	75
4.1	Introduction	75
4.2	Exploration des Méthodes d'Authentification	75
4.2.1	Comparaison des différentes approches	75
4.2.2	Approche basée sur la biométrie vocale	76
4.3	Acquisition de Données et prétraitement pour la Biométrique Vocale	77
4.3.1	Analyse Exploratoire des Données (EDA)	77
4.3.2	Prétraitement des Données	83
4.4	Modèle de Biométrie Vocale	87
4.4.1	Architecture du modèle	87
4.4.2	Entraînement du modèle	88
4.4.3	Résultats et Évaluation	89
4.5	Conclusion	92

5 Système de Traitement du Langage Naturel	94
5.1 Introduction	94
5.2 Acquisition de données et prétraitement	94
5.2.1 Analyse exploratoire des données (EDA)	94
5.2.2 Prétraitement et préparation des données	97
5.3 Modèles de reconnaissance d'intention	100
5.3.1 Exploration d'architectures des modèles entraînés	100
5.3.2 Entraînement des Modèles	103
5.3.3 Résultats et Comparaison	107
5.4 Extraction d'Entités Nommées (NER)	110
5.5 Pipeline de FAQs	111
5.6 Conclusion	113
6 Mise en oeuvre	115
6.1 Introduction	115
6.2 Développement des APIs	115
6.2.1 API bancaire de simulation	115
6.2.2 API NLP	120
6.2.3 API de la biométrie vocale	125
6.3 Développement de l'application mobile	128
6.3.1 Architecture physique de l'application	128
6.3.2 Architecture logicielle de l'application	129
6.3.3 Implémentation	130
6.3.4 Tests	131
6.4 Intégrations	132
6.4.1 Intégration de l'API bancaire avec l'API d'authentification	132
6.4.2 Intégration de l'API bancaire avec l'application mobile	132
6.4.3 Intégration de l'API bancaire avec l'API NLP	133
6.4.4 Intégration de l'API NLP avec l'application mobile	133
6.5 Conclusion	133
7 Démonstration et Défis	135
7.1 Introduction	135
7.2 Démonstration Finale	135
7.2.1 Interface d'accueil immersive	135
7.2.2 Interface d'Authentification	135
7.2.3 Interface principale de commandes	136
7.2.4 Interface Google Maps	137
7.3 Défis et perspectives	138
7.3.1 Défis rencontrés	138
7.3.2 Perspectives	138
7.4 Conclusion	139

Conclusion générale	140
Références	142

Introduction générale

L'équité numérique est désormais un enjeu majeur pour assurer un accès universel aux services numériques, quelles que soient les capacités physiques des individus. Au Maroc, cette problématique est particulièrement préoccupante : selon l'IAPB, près de 3,7 millions de personnes souffrent de déficience visuelle, dont environ 150 000 sont aveugles. Ces statistiques soulignent l'impérieuse nécessité de développer des solutions technologiques pour améliorer l'accessibilité de cette population. Une telle initiative est cruciale pour garantir une véritable inclusion numérique, permettant à tous de participer pleinement à une société numérique en constante évolution.

Parallèlement, les entreprises, conscientes de leur responsabilité sociale, s'engagent activement à proposer des services inclusifs. Attijariwafa Bank, un acteur majeur du secteur bancaire au Maroc, incarne cette vision à travers une stratégie plaçant le client au cœur de ses préoccupations. Dans ce contexte s'inscrit le présent rapport de projet de stage de fin d'études, visant à concilier ces impératifs en proposant le développement d'une solution novatrice : un système de reconnaissance vocale destiné à améliorer l'accessibilité des personnes non-voyantes dans l'application eBanking. Ce projet permet à Attijariwafa Bank de démontrer son engagement envers l'équité numérique et l'inclusion, en offrant des services adaptés aux personnes non-voyantes et en renforçant son image en tant qu'acteur socialement responsable.

L'avènement des technologies vocales ouvre de nouvelles perspectives pour l'intégration des personnes non-voyantes dans le domaine numérique. Ce projet vise à permettre aux utilisateurs très malvoyants ou non-voyants d'interagir aisément avec l'application bancaire en utilisant exclusivement leur voix, offrant ainsi une autonomie accrue dans leurs transactions financières quotidiennes. L'objectif principal est de concevoir un système de reconnaissance vocale capable de comprendre et d'exécuter des commandes spécifiques liées aux fonctionnalités de l'application eBanking, telles que la vérification du solde, les transferts d'argent et la consultation des transactions.

Pour atteindre cet objectif, diverses contraintes doivent être prises en compte, notamment la sécurité et la confidentialité des données, l'ergonomie de l'interface vocale pour une navigation intuitive et fluide, ainsi que l'intégration avec les fonctionnalités existantes de l'application eBanking. Nous avons opté pour le développement d'une application mobile sécurisée, utilisant une authentification à deux facteurs combinant l'authentification biométrique et des questions de sécurité génériques avant de traiter les demandes de l'utilisateur. La priorité a été accordée au développement de la compréhension du texte (NLU) pour le système de reconnaissance vocale, tandis que la conversion du texte en parole et vice versa sera gérée par des plugins intégrés à l'application mobile. Ces choix ont été motivés par la nécessité d'assurer une précision élevée dans un laps de temps restreint.

Le travail sera organisé en plusieurs phases distinctes, couvrant l'analyse des besoins des utilisateurs, la conception et la modélisation, le développement du système de reconnaissance vocale, l'intégration avec l'API bancaire, jusqu'aux tests et évaluations finales. Cette approche méthodique garantit une progression efficace et une validation rigoureuse de chaque étape du développement.

Ce rapport détaillera chaque phase du projet après avoir présenté le cadre général et l'étude de l'état de l'art, depuis l'analyse initiale des besoins jusqu'au déploiement final du système. Il examinera également les défis rencontrés, les solutions proposées, ainsi que les résultats obtenus lors des tests et des évaluations. Enfin, il mettra en lumière l'impact potentiel de cette innovation sur l'inclusion numérique des personnes non-voyantes au Maroc.

Chapitre I : Organisme d'accueil et contexte général du projet

1 Organisme d'accueil et contexte général du projet

1.1 Introduction

Ce premier chapitre vise à dévoiler en profondeur les contours du projet. Tout d'abord, nous commencerons par une présentation succincte de l'institution hôte , puis nous aborderons la problématique qui l'anime et mettrons en lumière les objectifs qui le guident de manière explicite. Nous détaillerons également la planification méticuleuse qui s'érige en socle solide pour garantir l'accomplissement fructueux du projet.

1.2 Présentation de l'organisme

1.2.1 Attijariwafa Bank

Ce projet a été mené au sein du groupe Attijariwafa Bank (AWB), au sein des équipes du Digital Center du pôle Systèmes d'Information Groupe, partie intégrante du Pôle Transformation, Innovation, Technologies et Opérations.

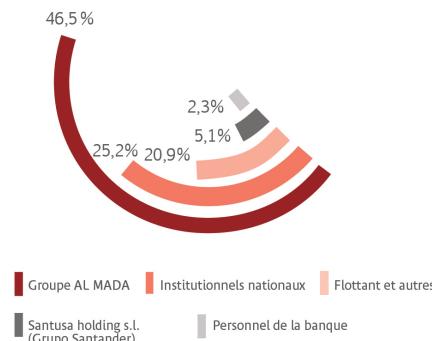
Attijariwafa Bank est un groupe bancaire et financier marocain issu de la fusion en 2003 entre la Wafabank et la Banque Commerciale du Maroc (BCM). Depuis lors, AWB est devenue la première banque du Maroc et d'Afrique du Nord, et la quatrième en Afrique. Dirigée par Mohamed el-Kettani, elle est cotée sur le MASI sous le symbole "ATW" et a son siège social à Casablanca. La majorité des actions est détenue par le groupe Al Mada.

AWB exerce ses activités dans la banque, la finance et l'assurance, offrant une gamme de services allant des comptes courants à l'assurance et au crédit à la consommation. Avec un effectif de 20 602 en 2020 et une capitalisation de 84,76 milliards de dirhams en 2022, elle maintient sa position de leader. Ses filiales, telles qu'Attijari Bank, Wafa Assurance, et Wafasalaf, et ses acquisitions comme Barclays Bank Egypt en 2017, illustrent son expansion. AWB est également engagée dans le développement durable et la responsabilité sociale, avec des initiatives dans l'éducation, l'entrepreneuriat, l'art, la culture et l'environnement.



FIGURE 1 – Logo Attijariwafa Bank

Actionnariat au 31 décembre 2023



Apple Pay

Avec Apple Pay, ton iPhone devient ta carte Attijariwafa bank.
Aussi simple que rapide.



[Je découvre](#)

Actualités



ACTUALITÉS | 19/03/2024

Google Pay : Paiements simplifiés pour les clients

d'Attijariwafa bank et L'bankalik



ACTUALITÉS | 19/03/2024

Avec votre nouvelle carte Attijariwafa bank –

Royal Air Maroc, gagnez des Miles Safar Flyer

pour vos paiements au quotidien

FIGURE 2 – Actionnariat et Site web : d'Attijariwafa Bank

<https://www.attijariwafabank.com/fr>

1.2.2 Filiales de la société

1. Filiales au Maroc :

Attijariwafa Bank (AWB) a étendu son offre de services bancaires au Maroc en développant des expertises spécialisées à travers ses Sociétés de Financements Spécialisés. Ces entités occupent des positions dominantes sur leur marché respectif, renforçant ainsi la diversification et la compétitivité du groupe. Parmi ses filiales au Maroc figurent Attijari Factoring, leader de l'affacturage, Attijari Finances Corp, fournisseur d'expertise en fusion-acquisition, et Attijari Global Research, offrant une couverture en macro-économie. Attijari Intermediation est le leader de l'intermédiation boursière, tandis que Bank Assafa est un pionnier de la banque participative. Wafa Assurance est le leader national de l'assurance, et Wafacash est le leader des transferts d'argent nationaux et internationaux. Ces filiales, ainsi que d'autres, contribuent à renforcer l'inclusion financière et à répondre aux besoins diversifiés des clients.



FIGURE 3 – Filiales d'AWB au Maroc

2. Filiales à l'international :

Attijariwafa Bank étend sa portée mondiale avec des opérations dans 26 pays, illustrant sa dimension internationale. Ses filiales internationales bénéficient de sa réputation solide, de son expertise et de sa compréhension des marchés locaux. Par exemple, Attijari bank Mauritanie propose des solutions innovantes pour le développement économique du pays, tandis qu'Attijari bank Tunisie est un acteur majeur du secteur bancaire avec une gamme complète de produits. En Égypte, Attijariwafa bank étend son réseau avec une stratégie axée sur la croissance et les synergies internationales. Attijariwafa bank Europe fournit des solutions bancaires aux Africains en Europe et facilite les échanges commerciaux avec l'Afrique. Ces filiales, ainsi que d'autres comme la Banque Internationale pour le Mali et la CBAO Sénégal, offrent une gamme variée de services financiers répondant aux besoins diversifiés des clients dans chaque région.



FIGURE 4 – Filiales d'AWB à l'international

1.2.3 Structure et organisation générale

Attijariwafa Bank, l'un des principaux groupes bancaires du Maroc, se distingue par une organisation rigoureuse et efficiente, conçue pour répondre aux besoins diversifiés de ses clients et assurer une gestion solide et transparente.

Structure Principale :

L'organigramme d'Attijariwafa Bank repose sur cinq pôles autonomes, chacun ayant ses propres missions et responsabilités :

1. **Pôle Banque de Détail Maroc et Europe** : ce pôle supervise les opérations de banque de détail au Maroc et en Europe, assurant ainsi la satisfaction des clients dans ces régions clés.
2. **Pôle Banque de Détail à l'International et Filiales de Financement Spécialisées** : responsable des activités de banque de détail sur les marchés internationaux et de la supervision des filiales spécialisées dans le financement, ce pôle contribue à l'expansion et à la diversification des services.
3. **Pôle Gestion Globale des Risques Groupe** : assumant la responsabilité cruciale de la gestion des risques, ce pôle garantit la solidité financière et la pérennité de la banque.
4. **Le pôle où s'est déroulé mon stage** : pôle Transformation, Innovation, Technologies et Opérations plus spécifiquement **Digital Center du Système d'Information Groupe** : en charge de piloter la transformation digitale et d'améliorer l'efficacité opérationnelle, ce pôle joue un rôle clé dans l'adaptation aux évolutions du marché et aux besoins changeants des clients.

5. Pôle Corporate Investment Banking : Offrant des services de banque d'affaires et d'investissement aux grandes entreprises et aux investisseurs institutionnels, ce pôle participe activement au développement économique et financier.

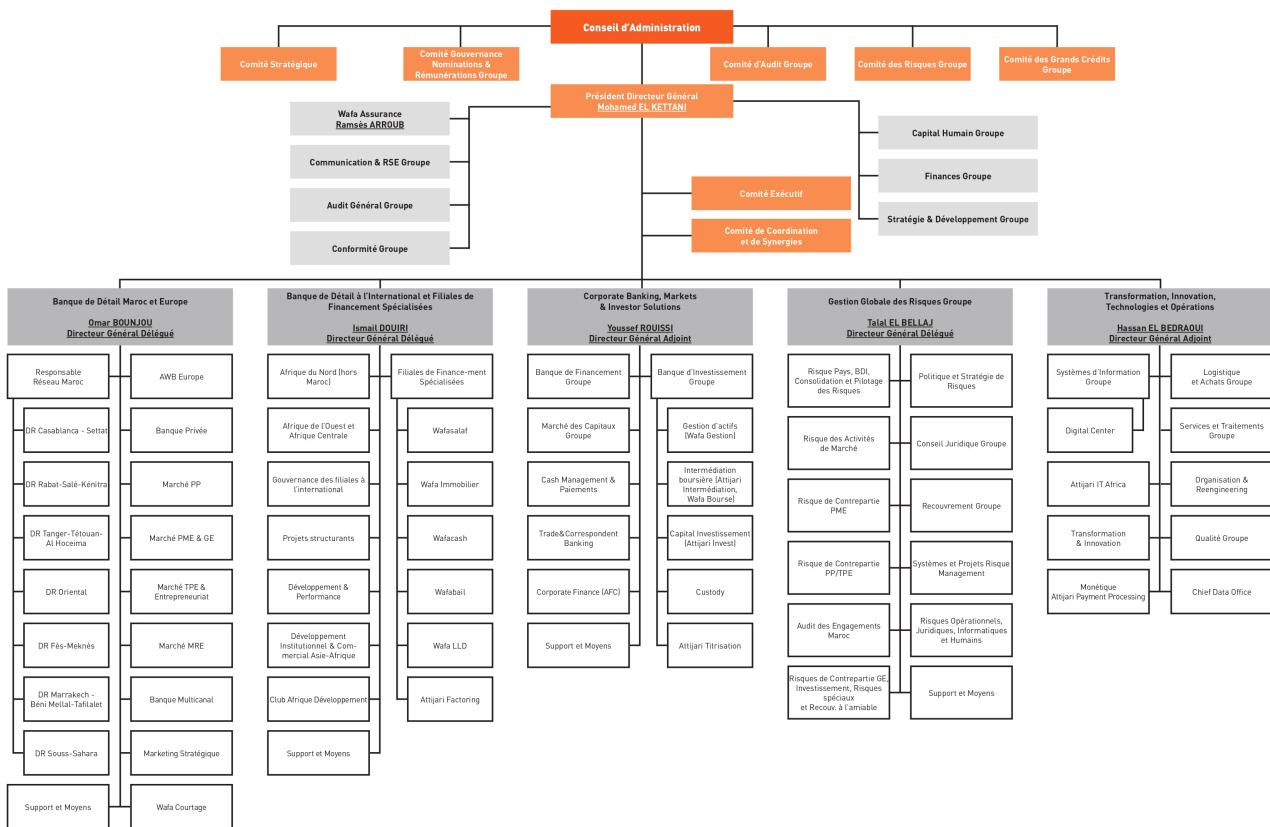


FIGURE 5 – Organigramme AWB

Système de Gouvernance :

Le système de gouvernance repose sur un **Conseil d'Administration** composé de 13 membres, chargé de la solidité financière de la banque et de la définition des orientations stratégiques. Pour l'assister, cinq comités spécialisés ont été institués, couvrant des domaines tels que l'audit, les risques, la gouvernance, la stratégie et les grands crédits.

Un principe de collégialité des décisions a été instauré, avec un **Comité Exécutif** chargé de piloter la performance de l'établissement et de décliner la stratégie approuvée par le Conseil d'Administration. Ce Comité Exécutif est dirigé par le Président Directeur Général et est composé de plusieurs directeurs généraux délégués responsables de différents pôles et fonctions clés de l'entreprise.

Parallèlement, le **Comité de Coordination et de Synergies** favorise les échanges entre les directions et assure la coordination des projets transversaux. Il réunit régulièrement les principaux responsables en plus des membres du Comité Exécutif, renforçant

ainsi la collaboration et la cohérence des initiatives au sein du groupe.

Dans l'ensemble, l'organisation d'Attijariwafa Bank reflète son engagement envers l'excellence opérationnelle, l'innovation et la satisfaction client, tout en maintenant des normes élevées de gouvernance et de gestion des risques.

1.2.4 Vision future du groupe :

Le groupe bancaire a récemment dévoilé son nouveau plan stratégique, intitulé @MBI-TIONS 2025, articulé autour de trois principales ambitions. Tout d'abord, il vise à renforcer sa position en tant que leader financier en Afrique, en favorisant une croissance responsable et durable dans les régions où il est présent ainsi que sur de nouveaux marchés. Ensuite, la banque s'engage à consolider son image en tant que banque relationnelle et citoyenne de référence, axée sur l'innovation, l'agilité et la compétitivité, en exploitant les opportunités offertes par la transformation digitale, le Big Data et les synergies internes. Enfin, Attijariwafa Bank s'efforcera de maintenir son alignement avec les normes internationales en matière d'efficience opérationnelle, de gestion des risques et de conformité, assurant ainsi une croissance durable et stable.

1.2.5 Missions de AWB

- Être le partenaire de référence des entreprises européennes opérant sur les marchés maghrébins et subsahariens et des entreprises présentes sur cette zone géographique, développant des courants d'affaires en Europe.
- Développer les synergies avec tous les pays où Attijariwafa bank est présente.
- Accompagner les investissements vers les pays du Maghreb et d'Afrique francophone.
- Offrir un service et un suivi personnalisé aux clients (réactivité/agilité) ainsi qu'une bonne connaissance du marché et une expertise en ce qui concerne le risque africain.
- Se positionner comme un acteur régional de référence en matière de syndication avec les banques internationales.

1.2.6 Secteurs d'activités

Attijariwafa Bank, opère dans une multitude de secteurs d'activité, reflétant sa position en tant que groupe bancaire universel et diversifié. Elle joue un rôle majeur dans la banque de détail, fournissant des services bancaires courants tels que les comptes chèques, les prêts et les cartes de crédit, tout en ciblant également les petites et moyennes entreprises avec des solutions de financement adaptées. En outre, le groupe est un acteur important sur le marché des capitaux, proposant des services de gestion d'actifs, de courtage en bourse et de conseil financier. Attijariwafa Bank est également impliquée dans le secteur de la banque d'investissement, facilitant les fusions et acquisitions, ainsi que le

financement de projets d'infrastructure. Enfin, elle s'engage dans des initiatives parabana- caires innovantes telles que les services mobiles et les plateformes d'accompagnement des entrepreneurs, contribuant ainsi au développement économique et financier du Maroc et des pays où elle est implantée. Enfin, la banque s'engage dans le secteur de la finance islamique, offrant des produits conformes à la Charia pour répondre aux besoins des clients soucieux de respecter les principes de l'islam en matière de finance.

1.2.7 Produits et services

Attijariwafa Bank se démarque par la diversité et l'innovation de ses produits et services, répondant aux besoins variés de sa clientèle tout en stimulant le développement économique et financier. Son offre comprend une gamme complète de services bancaires de détail pour les particuliers et les professionnels, ainsi que des solutions personnalisées de gestion patrimoniale via sa Banque Privée. Pour les entreprises et les institutionnels, elle propose des services sur mesure en financement et investissement, incluant le conseil en investissement et des produits spécialisés comme le crédit-bail. Le groupe offre également des services parabanaquaires innovants, tels que L'bankalik, une banque 100% mobile, et Dar Al Moukawil, un service dédié aux entrepreneurs. Attijariwafa Bank accorde une attention particulière à sa clientèle internationale et soutient activement les entreprises à travers des solutions adaptées. Enfin, elle joue un rôle clé en facilitant le financement des petites, moyennes et grandes entreprises, contribuant ainsi au développement économique du Maroc.



FIGURE 6 – Marques du Groupe

1.2.8 Valeurs Fondamentales

Les fondements de la culture d'entreprise reposent sur cinq valeurs fondamentales, guidant sa stratégie, influençant les principes éthiques et déontologiques au quotidien, et façonnant l'identité de l'institution bancaire.

Ces valeurs sont profondément incarnées par les membres du Groupe, donnant un sens à

leur dévouement envers le développement de la banque et à leur engagement à satisfaire pleinement nos clients.

- **Citoyenneté** : Contribuer activement au progrès de la nation.
- **Solidarité** : Faire croître la cohésion et la collaboration au sein de l'équipe.
- **Engagement** : Travailler sans relâche pour répondre aux besoins et aux attentes des clients.
- **Leadership** : Affirmer la détermination à atteindre l'excellence et à obtenir des résultats positifs.
- **Éthique** : Agir en accord avec les normes morales et les principes éthiques.



FIGURE 7 – Valeurs de AWB

1.2.9 Présence internationale

Attijariwafa bank étend son empreinte à l'échelle mondiale avec une présence dans 26 pays en Afrique, en Europe, en Amérique et au Moyen-Orient. Doté du réseau le plus dense du continent africain, avec 5835 agences, le groupe opère dans divers pays, consolidant ainsi son positionnement stratégique. Au Maroc, son pays d'origine, Attijariwafa bank compte 3494 agences, tandis qu'en Afrique du Nord, il dispose de 293 agences. En Europe, au Moyen-Orient et en Amérique, le groupe étend ses activités à travers 69 points de vente. Basé à Marrakech, Attijariwafa bank exerce son influence à travers ses filiales bancaires majoritairement contrôlées dans 26 pays, notamment en Afrique (Tunisie, Mauritanie, Egypte, Sénégal, Burkina Faso, Niger, Bénin, Côte d'Ivoire, Mali, Togo, Cameroun, Gabon, Congo) et en Europe (Belgique, France, Allemagne, Pays-Bas, Italie et Espagne). En outre, il déploie ses activités internationales à travers des bureaux de représentation stratégiquement situés à Genève, Londres, Dubaï, Abu Dhabi, Riyad et Montréal, renforçant ainsi sa présence mondiale et sa capacité à répondre aux besoins de sa clientèle diversifiée.



FIGURE 8 – Présence mondiale de AWB

1.3 Présentation du projet

1.3.1 Problématique

Les statistiques alarmantes au Maroc révèlent qu'en 2020 près de 3,7 millions d'individus souffrent de perte de vision, dont 150 000 étaient aveugles, et éprouvent des difficultés d'accès aux services numériques en raison de leurs déficiences visuelles. Cette réalité souligne l'importance urgente de développer des solutions technologiques pour surmonter les barrières d'accès et favoriser une inclusion numérique authentique. L'émergence des technologies vocales ouvre de nouvelles perspectives pour l'intégration des personnes non-voyantes dans le domaine numérique.

Dans ce contexte, Attijariwafa bank , plaçant le client au coeur de ses préoccupations, s'engage à fournir des services inclusifs. Le projet proposé vise à permettre aux utilisateurs très malvoyants ou non-voyants d'interagir aisément avec l'application bancaire en utilisant exclusivement leur voix. Cette initiative répond à un besoin crucial d'accessibilité, offrant aux individus non-voyants une autonomie accrue dans leurs transactions financières quotidiennes.

L'objectif principal de ce projet est donc de concevoir un système de reconnaissance vocale capable de comprendre et d'exécuter des commandes spécifiques liées aux fonctionnalités de l'application eBanking. Cela implique la vérification du solde, les transferts d'argent, la consultation des transactions, et bien d'autres opérations essentielles. En rendant l'ensemble du processus bancaire accessible par la voix, nous aspirons à créer une interface intuitive et conviviale pour les utilisateurs non-voyants , tout en garantissant la sécurité, la confidentialité des données et une interface intuitive pour les utilisateurs non-voyants.

Pour ce faire, une application mobile sécurisée à l'aide de l'authentification biométrique et intégrant un système de reconnaissance vocale est envisagée, offrant ainsi une solution

complète et efficace pour répondre aux besoins de ces utilisateurs.

1.3.2 Étude Benchmarking

1. Méthodes d'accès au monde numérique pour les non-voyants

Dans le monde numérique, les non-voyants peuvent accéder à différents outils et technologies pour faciliter leur navigation et leur interaction avec les appareils électroniques. Sur les sites Web, des lecteurs d'écran comme NVDA et des navigateurs web accessibles sont disponibles, conformément aux directives d'accessibilité telles que WCAG 2.0 et ARIA. Pour les transactions bancaires, des guichets automatiques offrent des fonctionnalités vocales. Sur smartphones, des lecteurs d'écran tels que TalkBack pour Android et VoiceOver pour iOS sont utilisés, accompagnés de claviers braille physiques ou virtuels et de gestes tactiles intuitifs. Des applications dédiées comme Lookout et Be My Eyes enrichissent également l'expérience numérique des non-voyants.

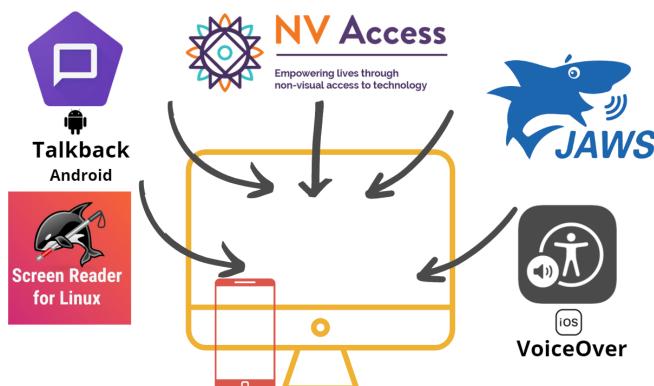


FIGURE 9 – Méthodes d'accès au monde numérique pour les non-voyants

2. Identification des concurrents clés et analyse des fonctionnalités clés

— Société Générale :

Application iPhone accessible aux non-voyants : La Société Générale a rendu son application iPhone entièrement accessible aux non-voyants en améliorant sa compatibilité avec VoiceOver, un logiciel de lecture d'écran.

ELIOTT, un callbot pour une banque réactive : ELIOTT est un robot téléphonique basé sur la reconnaissance du langage naturel, visant à offrir un accompagnement plus réactif aux clients. Il traite efficacement 4 millions de conversations par an avec les clients, offrant ainsi une gestion des conversations robuste. Avec un taux de réponse automatique de 22%, Eliott assure le traitement complet de nombreuses demandes clients de manière autonome. De plus, il garantit une disponibilité 24/7, permettant aux clients d'accéder aux services bancaires à tout moment, et maintient un taux de réponse élevé de 80% tout en assurant une interprétation précise des demandes des clients, avec un taux de confirmation de 90%.

- **Répondre aux FAQs :** Elliott peut répondre à un large éventail de questions sur les produits et services de Société Générale, telles que les comptes bancaires, les crédits, les cartes et les assurances.
- **Effectuer des transactions courantes :** telles que la consultation de solde, le transfère d'argent et le payement des factures.
- **Prendre des rendez-vous :** aider les clients à prendre rendez-vous avec un conseiller bancaire pour des questions plus complexes.
- **Signaler des problèmes :** aider les clients à signaler des problèmes tels que des cartes perdues ou volées ou des transactions frauduleuses.
- **Suivre les demandes :** Elliott permet aux clients de suivre l'état de leurs demandes, telles que les demandes de crédit ou d'ouverture de compte.
- **Obtenir des conseils personnalisés :** fournir des conseils personnalisés aux clients en fonction de leurs besoins et objectifs financiers.

— **Ally Bank :**

Ally Assist : L'un des premiers chatbots bancaires, Ally Assist, lancé en 2015, fournit un service client personnalisé via l'application iPhone d'Ally Bank. Ally Bank a également introduit Ally Skill sur Amazon Alexa, permettant des tâches bancaires simples via des commandes vocales. Les utilisateurs peuvent donc interagir avec Ally Assist en parlant ou en tapant leurs requêtes

- **Effectuer les transactions courantes :** Permet de payer des factures via l'application mobile Ally Bank et facilite les transferts de fonds entre comptes. .
- **Demandes d'informations sur les comptes et de détails sur les transactions :** Fournit des informations détaillées sur les comptes et sur les transactions, notamment en cas de noms de commerçants inconnus.
- **Analyse des dépenses et budgeting :** Offre des recommandations de budgétisation et une analyse historique des dépenses.
- **Alertes et notifications :** Envoie des alertes de paiement de factures et des notifications en cas de fraude ou d'activités suspectes sur leur compte.
- **Prévisions et suggestions personnalisées :** Utilise l'intelligence artificielle pour anticiper les besoins des clients et fournir des solutions pertinentes.

— **Emirates NBD :**

Eva : Le chatbot Eva offre un support instantané aux clients, les aidant à vérifier leurs soldes, suivre leurs dépenses et trouver le distributeur automatique le plus proche. Eva se distingue par ses capacités de reconnaissance vocale, rendant les tâches bancaires plus pratiques et conviviales.

- **Banque en ligne :** Accès à tous les services bancaires de Emirates NBD.
- **Assistant virtuel basé sur la voix et le texte :** EVA est conçu pour

fonctionner à la fois via des commandes vocales et des interactions par chat, offrant une expérience client fluide et intuitive.

- **Self-service et redirection :** EVA peut traiter de manière autonome certaines demandes des clients et les rediriger vers le bon menu ou vers un agent de service client si nécessaire.
- **Intégration multicanal :** EVA permet une expérience omnicanal, où les clients peuvent commencer une interaction via le chatbot et la poursuivre via la voix sur différents appareils.

— Bank of America :

Erica : Erica est l'assistant virtuel de Bank of America, accessible via l'application mobile de la banque, capable d'effectuer diverses tâches pour les clients. Erica est disponible 24 heures sur 24, 7 jours sur 7, dans l'application mobile et permet aux utilisateurs d'interagir avec Erica via des commandes vocales, des chats textuels ou simplement en tapant sur l'écran de leur téléphone dans l'application mobile.

- **Alertes financières et rappels de paiement :** Erica alerte les utilisateurs lorsque des remboursements de marchands sont crédités sur leur compte et en cas de charges dupliquées. Elle envoie également des rappels lorsque des paiements sont programmés.
- **Surveillance des charges répétitives :** Erica surveille les charges récurrentes et informe les utilisateurs des augmentations.
- **Rapports de dépenses et des transactions passées :** Les utilisateurs reçoivent des mises à jour hebdomadaires sur leurs dépenses mensuelles et peuvent localiser des transactions passées sur leurs différents comptes.
- **Consultation des soldes et informations de compte :** Erica permet de voir les soldes de tous les comptes et d'accéder aux numéros de routage et de compte.
- **Référencement à un spécialiste :** Erica peut référer les utilisateurs à un spécialiste pour discuter des opportunités potentielles.
- **Gestion des cartes :** Erica aide à remplacer les cartes perdues ou volées, vérifier les soldes et verrouiller ou déverrouiller temporairement les cartes de débit.

— HDFC Bank :

Eva : Le chatbot Eva de HDFC Bank fournit des conseils financiers personnalisés et des recommandations, en plus d'aider les clients avec plus de 200 tâches bancaires différentes. EVA offre une assistance bancaire en continu et prend en charge les langues anglaise et hindi.

- **Services de prêts :** Les utilisateurs peuvent obtenir des informations sur une variété de prêts, notamment les prêts personnels, les prêts aux

entreprises, les prêts sur cartes de crédit, les prêts pour deux-roues, les prêts immobiliers et les prêts auto.

- **Dépôts à terme :** Les clients peuvent ouvrir des dépôts à terme et accéder à des outils tels que le calculateur de DAT pour estimer leurs rendements.
- **Gestion des comptes bancaires :** EVA aide à ouvrir des comptes d'épargne et de salaire, fournit des informations sur les avantages et les fonctionnalités des comptes, et guide les clients tout au long du processus d'ouverture de compte.
- **Gestion des Cartes de crédit et débit :** Les utilisateurs peuvent obtenir des détails sur les cartes de crédit et de débit, consulter les relevés, effectuer des paiements de factures et demander de nouvelles cartes ou des mises à niveau.



FIGURE 10 – Logos des concurrents clés

3. Tableau comparatif

Le tableau comparatif suivant fournit une vue d'ensemble essentielle pour notre projet d'assistant vocal bancaire :

- L'analyse des fonctionnalités clés des concurrents, telles que la gestion des conversations et les paiements de factures, nous permet d'identifier des opportunités de distinction pour notre propre assistant.
- La compatibilité cross-platform et la réponse en temps réel sont des aspects cruciaux à intégrer pour assurer une expérience utilisateur fluide et réactive.
- L'accent mis sur l'interprétation précise des demandes et les réponses automatiques souligne l'importance de l'IA dans notre solution.
- Les commentaires positifs des clients sur la facilité d'utilisation, l'efficacité et l'utilité des assistants vocaux actuels mettent en lumière l'importance de l'expérience utilisateur dans la conception de notre propre assistant vocal bancaire.

En intégrant ces éléments, nous pouvons aspirer à offrir un service qui répond aux besoins et aux attentes des clients tout en restant compétitif sur le marché.

Produit Concurrent	Fonctionnalités clés	Cross-platform	Prix	Réponse en temps réel	Feedback des clients
Callbot - Société Générale	- Gestion des Conversations (4 millions de conversations par an) - Réponse Automatique (22% des demandes traitées) - Disponibilité 24/7 - Taux de Réponse Élevé (80%) - Interprétation Précise (90% de taux de confirmation) - Répondre aux FAQs Effectuer des Transactions Courantes - Prendre des Rendez-vous - Signaler des Problèmes - Suivre les Demandes - Obtenir des Conseils Personnalisés	Oui	Gratuit	Oui	Positif (facile à utiliser, utile, efficace)
Ally Assist - Ally Bank	- Paiement des Factures - Transferts d'Argent - Demandes d'Informations sur les Comptes - Analyse des Dépenses et Budgeting - Alertes de Paiement et Notifications en cas de Fraude et Activités Suspectes - Demande de Détails sur les Transactions - Prévisions et Suggestions Personnalisées - Voix et Texte	Oui	Gratuit	Oui	Très positif (innovant, pratique, excellent service client)
Eva Emirates - NBD	- Accès à tous les services bancaires de Emirates NBD - Assistant Virtuel Basé sur la Voix et le Chatbot - Self-Service et Redirection - Intégration Multicanal	Oui	Gratuit	Oui	Positif (rapide, efficace, utile)
Erica - Bank of America	- Interactions Multiples - Alertes Financières et Rappels de Paiement - Surveillance des Charges Répétitives - Rapports de Dépenses et des Transactions Passées - Consultation des Soldes et Informations de Compte - Disponibilité 24/7 - Référencement à un Spécialiste - Gestion des Cartes	Oui	Gratuit	Oui	Excellent (gain de temps, intuitif, fonctionnalités utiles)
Eva - HDFC Bank	- Assistance 24/7 et Multilingue - Services de Prêts - Dépôts à Terme - Gestion des Comptes Bancaires - Gestion des Cartes de Crédit et Débit	Oui	Gratuit	Oui	Bon (utile pour les transactions bancaires de base, interface conviviale)

TABLE 1 – Tableau récapitulatif de l'étude benchmarking des solutions des concurrents

1.3.3 Missions

Mon projet de fin d'études s'inscrit dans le cadre d'un projet visant à améliorer l'accessibilité des personnes non-voyantes dans le eBanking en exploitant des technologies de l'intelligence artificielle. Ma mission principale consiste à concevoir une solution complète, englobant l'ensemble du processus depuis la reconnaissance vocale jusqu'à la compréhension des intentions de l'utilisateur, en passant par l'extraction et le traitement des opérations bancaires demandées par l'utilisateur, pour enfin synthétiser les réponses en parole.

Plusieurs missions m'ont été confiées dans le cadre de ce projet :

- Analyse des besoins des utilisateurs non-voyants.
- Définition des fonctionnalités de l'application.

- Conception de l'architecture globale du système.
- Collecte et prétraitement des données.
- Développement et entraînement des modèles NLP pour la reconnaissance des intentions des utilisateurs , en explorant différentes architectures et différents algorithmes.
- Développement de l'API NLP qui exploite les modèles NLP avec mise en place d'une pipeline NLP pour la gestion automatique des FAQs.
- Développement et mise à l'essai d'un modèle de sécurité permettant la reconnaissance des utilisateurs par leur voix.
- Conception et mise en place de la base de données de l'API bancaire de simulation.
- Développement d'une API bancaire de simulation et d'une API d'authentification.
- Conception et développement de l'app mobile.
- Intégration des différents composants.
- Amélioration de l'UI.
- Tests et évaluation globale du système.

1.3.4 Objectif du projet

Le projet vise avant tout à créer un système de reconnaissance vocale (RV) dédié à l'interprétation des commandes vocales associées aux différentes fonctionnalités de l'application eBanking. Ces fonctionnalités couvrent des actions telles que la vérification du solde, les transferts d'argent, le paiement des factures, ou encore la consultation des transactions. L'objectif final est de rendre l'intégralité des opérations bancaires accessibles aux personnes non-voyantes, en proposant une interface vocale conviviale et intuitive.

Les résultats attendus comprennent :

- Sécurité et confidentialité des données à l'aide de l'authentification biométrique et multi-facteurs.
- Une compréhension précise des commandes vocales des utilisateurs.
- Navigation vocale intuitive avec rétroaction vocale informative , précise et en temps réel.
- Intégration transparente avec les fonctionnalités existantes et compatibilité avec les plateformes mobiles.

1.4 Plannification

1.4.1 Cycle de vie du projet

Étant donné la complexité et l'envergure de ce projet, nous avons opté pour une approche de développement en cycle de vie en V agile. Ce modèle comprend une planification initiale, tout en mettant l'accent sur l'adaptabilité et la réactivité aux changements, permettant ainsi une exécution itérative avec des phases de validation et de rétroaction

régulières à chaque étape du processus de développement.

Avantages du cycle de vie en V :

- Permet une planification et une conception en amont, ce qui réduit les risques.
- Favorise une approche itérative qui permet d'adapter le système en fonction des commentaires et des tests.
- Aide à garantir que le système répond aux exigences des utilisateurs.
- Facilite la maintenance et la mise à jour du système à long terme.

Cycle de vie du projet :

Pour notre projet, nous avons choisi un cycle de vie en V agile. L'agilité se manifeste principalement par une itération et une adaptation continues. Nous travaillons de manière itérative, en revisitant et en ajustant les phases précédentes au fur et à mesure que de nouvelles informations sont découvertes ou que de nouveaux défis sont rencontrés. Nous adoptons une approche flexible dans le développement des fonctionnalités, en priorisant celles qui ont le plus d'impact pour les utilisateurs. De plus, nous effectuons des tests continus et nous nous engageons dans une amélioration progressive tout au long du processus de développement.

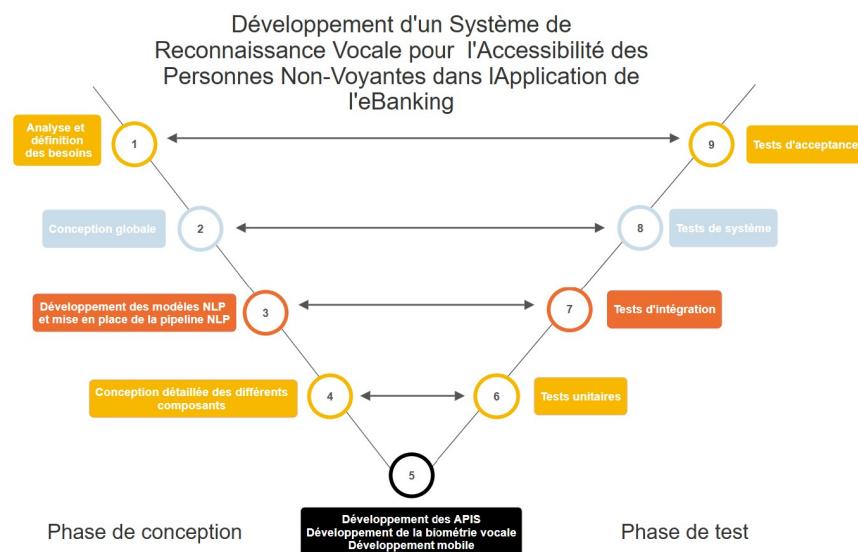


FIGURE 11 – Cycle de vie du projet

- 1. Analyse des besoins des utilisateurs et définition des fonctionnalités de l'application :** Comprendre les besoins des utilisateurs non-voyants en matière d'eBanking et à définir les fonctionnalités nécessaires pour répondre à ces besoins.
- 2. Conception de l'architecture globale du système :** Concevoir une architecture qui prend en compte les besoins des utilisateurs, en mettant l'accent sur la reconnaissance vocale, la compréhension des commandes de l'utilisateur, et les interactions avec l'API bancaire.

3. **Collecte et prétraitement des données** : Rassembler et prétraiter les données nécessaires pour entraîner les modèles.
4. **Développement et entraînement des modèles NLP, ainsi que la mise en place d'une pipeline NLP pour la FAQ** : Développer et entraîner des modèles NLP pour comprendre les demandes des utilisateurs, en explorant diverses architectures pour optimiser les performances. Ensuite, mettre en place une pipeline capable de répondre de manière très proche du langage humain aux questions fréquemment posées.
5. **Conception détaillée de tous les composants à développer** : Créer des schémas et des diagrammes pour représenter l'architecture du système, les flux de données et les interactions entre les modules tout en examinant les exigences fonctionnelles et non fonctionnelles ainsi que les spécifications techniques. .
6. **Développement des APIs, de l'application mobile et expérimentation du modèle de biométrie vocale** : Développer une API NLP pour reconnaître les intentions des utilisateurs et répondre automatiquement aux FAQs. Créer également une API bancaire de simulation et une API d'authentification, ainsi qu'une application mobile. Parallèlement, explorer un modèle siamois pour l'authentification biométrique basée sur la voix pour renforcer la sécurité.
7. **Tests unitaires, d'intégration système, d'acceptation et évaluation globale du système** : Réaliser des tests approfondis visant à évaluer le bon fonctionnement global du système. Nous effectuerons des tests unitaires pour chaque composant individuel, vérifiant leur fonctionnement isolé. Ensuite, nous procéderons aux tests d'intégration système pour garantir que ces composants fonctionnent correctement ensemble. Les tests d'acceptation seront également réalisés pour s'assurer que le système répond aux exigences et aux attentes des utilisateurs finaux.

1.4.2 Diagramme de Gantt

Pour garantir une organisation efficace de notre projet, nous avons opté pour l'utilisation d'un diagramme de Gantt. Ce précieux outil nous a permis de structurer l'ensemble des tâches inhérentes au développement de notre solution, tout en optimisant la gestion du temps.

Le lancement du projet en février a marqué le début d'un parcours traversant plusieurs étapes clés, débutant par « L'Analyse et Définition des Besoins » et se clôturant avec la « Finalisation et Rédaction de rapport ». Chacune de ces étapes est associée à une durée estimée, offrant ainsi une vue d'ensemble des délais et facilitant une coordination harmonieuse entre les divers aspects du projet.

Nous avons amorcé notre parcours par une description approfondie du travail, notamment avec « L'Analyse et Définition des Besoins », où les objectifs fondamentaux du

projet ont été définis. La planification est divisée en sept phases principales : analyse et définition, développement NLP, développement des API, développement de la biométrie vocale, développement mobile, tests et évaluation, et finalisation et préparation. Chaque phase comprend des tâches spécifiques avec des dates de début et de fin, et des pourcentages d'avancement indiquant la progression. Les tâches critiques, telles que l'entraînement des modèles NLP et le développement des APIs, sont bien avancées, assurant une progression cohérente et coordonnée du projet. Cette approche méthodique nous a permis de maintenir une vision claire du projet tout au long de son développement.



FIGURE 12 – Diagramme de GANTT

1.4.3 Les livrables

À la clôture de chaque phase du projet, un livrable est produit. Le tableau ci-dessous présente une liste détaillée des livrables pour chaque phase, accompagnée de leur description.

Phase	Livrables
Phase 1 : Analyse et définition	<ul style="list-style-type: none"> Etude du benchmarking et de l'état de l'art Spécifications des besoins fonctionnelles de l'application Architecture système globale Plan de collecte et de prétraitement des données
Phase 2 : Développement NLP	<ul style="list-style-type: none"> Modèles NLP entraînés pour la reconnaissance des intentions et l'extraction d'actions bancaires API NLP documentée Rapport d'expérimentation sur différentes architectures de modèles
Phase 3 : Développement des API	<ul style="list-style-type: none"> Base de données de l'API bancaire de simulation API Bancaire de simulation documentée Intégration de l'API NLP avec l'API bancaire
Phase 4 : Développement du modèle de reconnaissance de l'empreinte vocale	<ul style="list-style-type: none"> Modèle entraîné pour la reconnaissance des utilisateurs par leur voix Rapport de mise à l'essai du modèle
Phase 5 : Développement mobile	<ul style="list-style-type: none"> Application mobile native pour Android et iOS Intégration de l'API NLP avec l'application mobile Intégration de l'API bancaire avec l'application mobile
Phase 6 : Tests et évaluation	<ul style="list-style-type: none"> Évaluation de l'expérience utilisateur Recommandations d'amélioration

TABLE 2 – Livrables pour chaque phase du projet

Livrables supplémentaires :

- Rapport de stage
- Code source des modèles IA ,des APIs et de l'application
- Présentation du projet et démonstration de l'application

1.5 Conclusion

Au fil de ce chapitre, nous avons minutieusement exposé le contexte au sein duquel notre projet prend place, en mettant en lumière ses différents aspects et en détaillant ses

caractéristiques. Cette exploration nous a permis de mieux appréhender les enjeux et les besoins auxquels notre solution aspire à répondre. Le prochain chapitre nous conduira à une analyse approfondie de l'état de l'art dans le domaine de la reconnaissance vocale et du NLP, où nous examinerons les avancées et les pratiques actuelles pour situer notre projet dans le paysage de la recherche et de l'innovation.

Chapitre II : Étude de l'état de l'art

2 Étude de l'état de l'art

2.1 Introduction

Ce chapitre offre une analyse approfondie de l'état de l'art en présentant les modèles d'intelligence artificielle pertinents pour notre projet. En outre, il détaille la solution technique choisie, en exposant les conditions et contraintes qui ont orienté ce choix, ainsi que les technologies employées à chaque étape, en justifiant les décisions prises.

2.2 État de l'art

Cette section explore les technologies et approches de pointe en matière d'apprentissage automatique, d'apprentissage profond, de Transfert d'apprentissage, de traitement du langage naturel (NLP), de conversion Speech-to-Text (STT) et de Text-to-Speech (TTS) dans le développement de systèmes de reconnaissance vocale pour les personnes non-voyantes.

2.2.1 Apprentissage automatique (Machine Learning)

Le Machine Learning est une sous-discipline de l'intelligence artificielle (IA) qui englobe des techniques permettant aux ordinateurs de comprendre des choses à partir de données et de fournir des applications d'IA. Contrairement à la programmation traditionnelle, le ML permet aux systèmes de s'améliorer automatiquement avec l'expérience.

Techniques courantes dans le contexte dans notre projet :

1. **Support Vector Machine (SVM)** : Les SVM sont des algorithmes de classification supervisée utilisés pour séparer des ensembles de données en différentes classes avec un maximum de marge entre les classes. Ils utilisent des hyperplans pour effectuer la séparation et peuvent être appliqués à des problèmes de classification linéaire et non linéaire grâce à l'utilisation de fonctions kernel.

Efficacité et contribution à notre projet : Dans le cadre de notre assistant vocal, le SVM est un modèle pertinent pour la classification des intentions, permettant ainsi d'identifier avec précision l'intention de l'utilisateur à partir de sa commande vocale. Son efficacité se manifeste à plusieurs niveaux :

- **Précision élevée** : Les SVM démontrent une précision remarquable, surtout dans le cas de jeux de données de taille moyenne avec des marges de séparation bien définies.
- **Robustesse** : se révèlent robustes face aux données bruitées et aux valeurs aberrantes, garantissant ainsi une performance stable même dans des environnements variables.
- **Limitation** : Toutefois, la performance peut décroître avec des jeux de données très grands ou lorsque les classes sont très imbriquées.

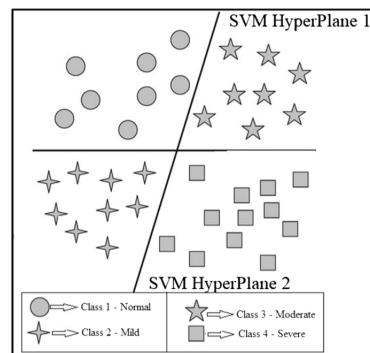


FIGURE 13 – Étude de l'état de l'art : SVM

2. Random Forest : Le Random Forest est un algorithme d'apprentissage supervisé qui utilise un ensemble d'arbres de décision pour améliorer la précision des prédictions et réduire le risque de surapprentissage (overfitting). Chaque arbre est construit à partir d'un échantillon aléatoire du jeu de données et les prédictions finales sont obtenues par la moyenne ou la majorité des votes des différents arbres.

Efficacité et contribution à notre projet : Le RF peut être déployé pour classifier les intentions, ce qui consiste à reconnaître l'objectif de l'utilisateur à partir de sa commande vocal. Nous intégrerons donc ce modèle dans notre future étude de benchmarking. Ce modèle se distingue par sa :

- **Précision robuste** : Très efficace pour une grande variété de tâches de classification et de régression.
- **Résilience aux outliers** : Les arbres de décision internes aident à minimiser l'effet des anomalies.
- **Scalabilité** : Peut gérer de grandes quantités de données et de caractéristiques.
- **Limitation** : Le modèle peut devenir complexe et coûteux en termes de temps de calcul et de mémoire.

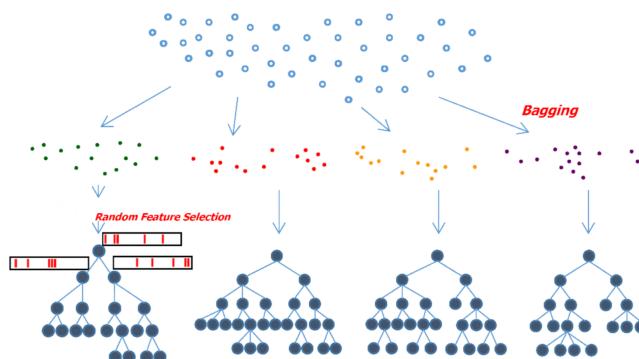


FIGURE 14 – Étude de l'état de l'art : Random Forest

3. XGBoost (Extreme Gradient Boosting) : est un algorithme de boosting de gradient qui combine plusieurs modèles faibles pour créer un modèle fort. Il utilise

une technique d'optimisation par gradient pour minimiser les erreurs et peut gérer des données avec des caractéristiques complexes.

Efficacité et contribution à notre projet : XGBoost peut servir de modèle de base pour notre étude de benchmarking sur la classification des intentions. Cet algorithme puissant nous permettra de discerner efficacement l'objectif de l'utilisateur à partir de sa commande vocale.

- **Performance supérieure :** Souvent en tête des compétitions de machine learning grâce à son efficacité et précision.
- **Vitesse :** Optimisé pour la vitesse et les ressources informatiques, il permet des calculs rapides même sur des jeux de données volumineux.
- **Flexibilité :** Peut être ajusté pour divers types de données et problèmes grâce à une large gamme d'hyperparamètres.
- **Limitation :** La complexité du modèle et le besoin d'un ajustement d'hyperparamètres précis peuvent rendre son utilisation plus difficile pour les débutants.

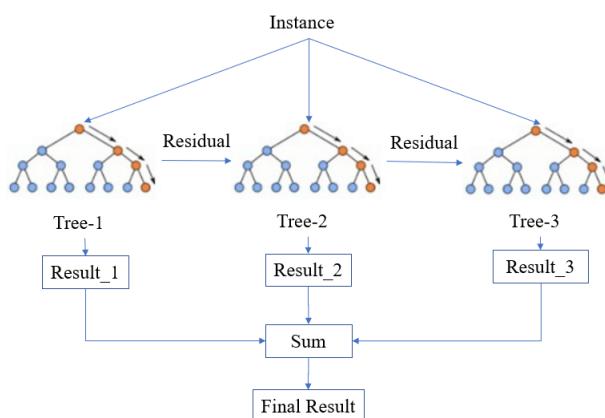


FIGURE 15 – Étude de l'état de l'art : XGBOOST

2.2.2 Apprentissage profond (Deep Learning)

Le Deep Learning est une branche du machine learning qui se concentre sur l'utilisation de réseaux de neurones artificiels pour apprendre des représentations de données complexes. Ces réseaux, appelés réseaux de neurones profonds en raison de leur architecture en couches multiples, sont capables d'apprendre automatiquement des caractéristiques hiérarchiques à partir de données non structurées ou brutes, ce qui leur permet de résoudre des tâches complexes telles que la reconnaissance vocale, la génération de texte et la synthèse de la parole.

Techniques courantes dans le contexte dans notre projet :

- **Réseaux de Neurones Convolutifs (CNN) :**
 1. **Rôle et Avantages des CNNs dans notre Projet :** Les *Convolutional Neural Networks (CNN)* sont une classe de réseaux de neurones spécialement

conçus pour traiter des données structurées en grille, comme les images. Toutefois, ils sont également puissants pour le traitement des signaux en raison de leur capacité à capturer des motifs locaux, à réduire la dimensionnalité et à généraliser à partir des données d'entraînement.

Le *CNN* sera employé pour extraire des caractéristiques des signaux audio, ce qui facilite la reconnaissance de motifs pertinents dans la voix de l'utilisateur. En particulier, pour les techniques d'authentification biométrique à l'aide de la voix.

2. Architecture : L'architecture typique des CNN comprend plusieurs types de couches qui sont agencées de manière séquentielle :

- **Couches de Convolution** : permet d'appliquer des filtres (ou kernels) sur l'entrée pour extraire des caractéristiques locales.
- **Couches de Activation (ReLU)** : introduire des non-linéarités dans le modèle en appliquant la fonction ReLu (Rectified Linear Unit).
- **Couches de Pooling** : permet de réduire la dimensionnalité des cartes de caractéristiques tout en conservant les informations importantes. Couches Complètement Connectées (Fully Connected) : intègre les caractéristiques extraites pour effectuer des prédictions finales.
- **Couches de Sortie** : produire la prédiction finale du modèle.

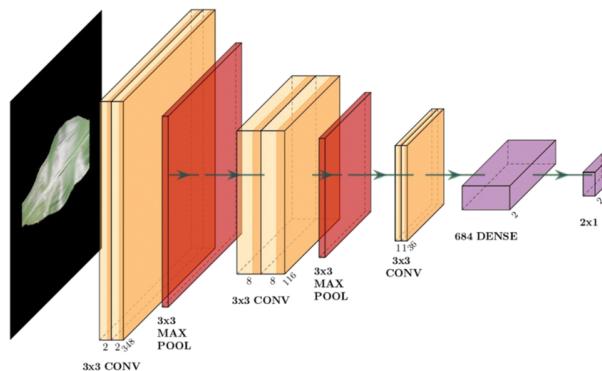


FIGURE 16 – Étude de l'état de l'art : Architecture du CNN

• **Réseaux de Neurones Récurrents (RNNs) :**

1. Rôle et Avantages pour notre projet : Les *RNNs* sont des algorithmes dotés d'une architecture d'apprentissage profond. Ils sont capables de prédire des séries temporelles ou des séquences de longueur variable, contrairement aux modèles précédents qui étaient limités à des entrées de taille fixe.

Les *RNNs* sont d'une grande utilité pour notre projet. Ils captent les dépendances contextuelles dans les séquences textuelles, ce qui aide à comprendre les intentions des utilisateurs et à extraire des entités clés , ils servent donc de modèle de base pour notre étude comparative des meilleurs modèles NLP.

2. Architecture : Un réseau de neurones récurrent est un type de réseau de neurones dans lequel les connexions entre les unités forment un graphe dirigé séquentiel. Cette architecture permet de capturer des dépendances temporelles en permettant un retour d'information dans le réseau, de la couche d'entrée à la couche de sortie.

L'architecture d'un RNN comprend une couche d'entrée, des couches cachées et une couche de sortie. À chaque étape temporelle t , chaque neurone récurrent reçoit le vecteur d'entrée $x(t)$ et le vecteur de sortie de l'étape temporelle précédente $y(t - 1)$.

Chaque neurone récurrent a deux types de poids :

- **Poids U** reliant les entrées à la sortie, similaires aux réseaux de neurones classiques.
- **Poids V** reliant la sortie de la couche à son entrée, formant les connexions récurrentes.

La sortie d'un neurone récurrent est une fonction de toutes les entrées des étapes précédentes, lui conférant une forme de mémoire. La **cellule de mémoire** est la partie du réseau qui conserve un état entre plusieurs étapes temporelles.

La fonction de la couche cachée à l'étape temporelle t est donnée par :

$$h(t) = f(h(t - 1), x(t))$$

où $h(t)$ représente l'état caché à l'étape t .

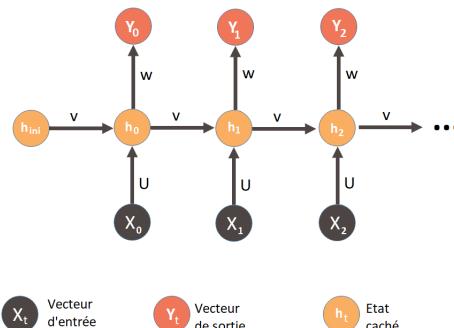


FIGURE 17 – Étude de l'état de l'art : Architecture des RNNs

• Long Short-Term Memory (LSTM)

1. Rôle et Avantages pour notre projet : Une cellule de mémoire à long terme et à court terme (LSTM) est un type de réseau de neurones récurrent (RNN), particulièrement efficace pour surmonter les problèmes de disparition et d'explosion du gradient, ainsi que les limitations de mémoire courte des RNNs classiques.

Nous avons choisi le **LSTM** comme modèle basé sur les embeddings de mots et les réseaux de neurones pour notre étude benchmarking, dans le but de déterminer le meilleur modèle de classification pour les intentions des utilisateurs. Nous avons opté pour le **LSTM** en raison de sa capacité à saisir les nuances contextuelles des textes grâce aux embeddings de mots, ainsi que de sa capacité à modéliser efficacement les dépendances dans les séquences de données longues à longueur variable à l'aide des RNNs.

2. Architecture : Au niveau de l'architecture, 4 couches sont intégralement connectées :

- une couche principale qui joue le rôle d'analyse des entrées courantes et de l'état précédent.
- 3 couches qui jouent le rôle de contrôleur des portes avec :
 - une porte d'oubli qui décide des parties de l'état à long terme qui doivent être effacées.
 - une porte d'entrée qui décide de l'addition des parties à ajouter à l'état long terme du réseau.
 - une porte de sortie qui sélectionne les parties de l'état à long terme qui doivent être lues.

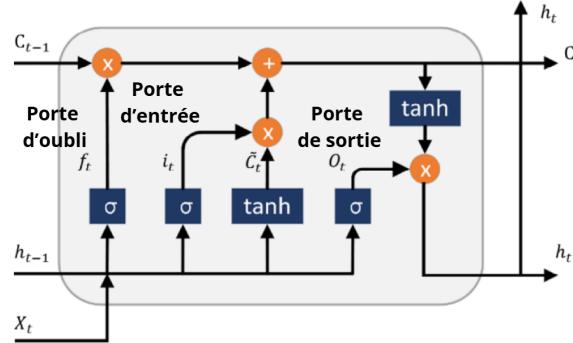


FIGURE 18 – État de l'art : Architecture LSTM

- **Gated Recurrent Unit GRU** La cellule d'unité récurrente à porte est similaire à LSTM mais avec une architecture simplifiée, facilitant l'entraînement tout en maintenant des performances élevées pour la modélisation des séquences. Son architecture comporte :

- une seule porte qui contrôle l'oubli et l'entrée.
- Pas de porte de sortie : l'état complet est émis à chaque étape temporelle.

Valeur ajoutée pour notre projet : Nous avons intégré GRU dans notre étude de benchmarking en raison de sa capacité à saisir efficacement les dépendances temporales dans les séquences de données, ce qui est crucial pour la compréhension des intentions des utilisateurs à partir de leurs requêtes. Son rôle principal était d'offrir

une analyse approfondie de sa performance comparativement à d'autres modèles de classification, en évaluant sa précision, sa robustesse et sa capacité à généraliser sur divers ensembles de données.

- **CamemBERT**

1. **Rôle et Avantages pour notre projet :** CamemBERT est une version du modèle **BERT** (Bidirectional Encoder Representations from Transformers) de Google AI, adaptée au français, entraînée sur un vaste corpus français pour capturer les particularités linguistiques et la structure de la langue. Utilisant une tokenization basée sur les sous-mots spécifiquement adaptée au français, il permet la modélisation du langage masqué (Masked Language Model) et la prédiction de la phrase suivante (NSP, Next Sentence Prediction).

CamemBERT a été choisi pour son excellence dans la compréhension du langage naturel, en particulier pour la classification des intentions des utilisateurs, grâce à son adaptation spécifique au français, garantissant une compréhension précise et contextuelle du texte dans cette langue. De plus, sa capacité à saisir les nuances linguistiques du français en fait un choix idéal pour l'extraction d'entités nommées, facilitant ainsi l'identification et la classification précises des entités dans le texte.

2. **Architecture :** CamemBERT conserve l'architecture de base de BERT mais est spécifiquement optimisé pour le français grâce à un entraînement sur un vaste corpus français et une tokenization adaptée.

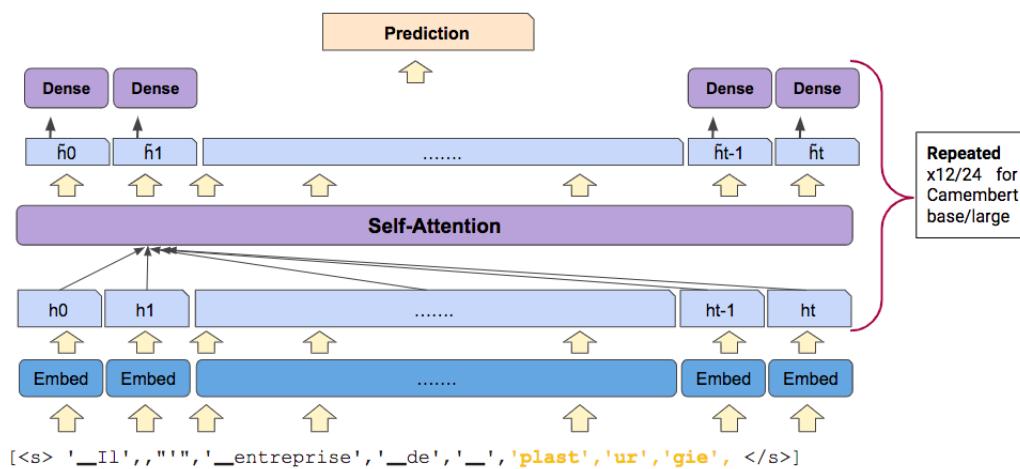


FIGURE 19 – Étude de l'état de l'art : Architecture CamemBERT

- **Entrée :**

- (a) **Tokenization :** Le texte d'entrée est tokenisé en sous-mots à l'aide de WordPiece. Chaque token est converti en un identifiant unique.
- (b) **Embedding :** Les tokens sont transformés en vecteurs d'intégration (embeddings) qui combinent des informations de position et de segment.

- (c) **Position Embeddings** : Pour capturer l'ordre des tokens, des embeddings de position sont ajoutés aux embeddings de tokens.
 - (d) **Segment Embeddings** : Pour différencier les paires de phrases (dans les tâches de classification de paires), des embeddings de segment sont ajoutés.
- **Transformers Encoders :**
 - (a) **Couches Empilées** : BERT utilise plusieurs couches d'encodeurs Transformers empilées (12 pour BERT-base, 24 pour BERT-large).
 - (b) **Self-Attention** : Chaque couche d'encodeur utilise des mécanismes d'attention bidirectionnelle, permettant au modèle de considérer le contexte de chaque mot dans les deux directions (gauche et droite).
 - (c) **Feed-Forward Layers** : Après le mécanisme d'attention, chaque couche passe par un réseau feed-forward entièrement connecté.

2.2.3 Apprentissage par Transfert (Transfer Learning)

Pour notre projet d'assistant vocal, nous envisageons d'utiliser les techniques de Transfer learning pour optimiser l'extraction des caractéristiques des signaux vocaux, ainsi que pour affiner les modèles afin de tirer parti des performances des modèles pré-entraînés et réduire les besoins en données. Le Transfer learning est un processus dans lequel un modèle formé sur un problème est utilisé pour résoudre un problème similaire, en exploitant les connaissances acquises. Dans le cadre de notre projet, nous mettrons en œuvre l'apprentissage par similarité, qui comprend deux aspects principaux :

- **Fine-tuning** : Cette étape implique l'ajustement des paramètres d'un modèle pré-entraîné sur un ensemble de données spécifique, soit pour la même tâche que celle pour laquelle le modèle a été initialement formé, soit pour une nouvelle tâche.
- **Extraction de caractéristiques** : Ici, nous utiliserons les représentations internes d'un modèle pré-entraîné comme entrée pour un nouveau modèle, afin de bénéficier de l'expertise capturée dans les données préalablement analysées.

2.2.4 Traitement du Langage Naturel (NLP)

Le Traitement du Langage Naturel (NLP) est une branche de l'IA. Il englobe la compréhension, l'analyse, la génération et la manipulation du langage naturel, permettant aux machines de traiter le texte de manière similaire à un être humain. Le NLP est une composante essentielle des assistants vocaux, facilitant la compréhension et la génération de texte en langage naturel. Cette technologie garantit une interaction fluide et naturelle avec les utilisateurs, offrant ainsi une expérience utilisateur optimale.

Capacités du NLP pour l'Assistant Bancaire Vocal

1. **Analyse Syntaxique** : Comprend la structure grammaticale des phrases pour saisir les demandes des utilisateurs.

2. **Analyse Sémantique** : Comprend le sens et le contexte des mots et des phrases pour des réponses plus précises.
3. **Réponse à des Questions** : Fournit des réponses précises aux questions des utilisateurs en extrayant des informations pertinentes.
4. **Extraction d'Entités Nommées (NER)** : Identifie des entités spécifiques comme les noms, les montants d'argent, etc.
5. **Reconnaissance des Intentions** : Identifie l'intention derrière une requête, comme vérifier le solde ou effectuer un virement.
6. **Génération de Réponses** : Produit des réponses appropriées en utilisant des règles ou des modèles pré-entraînés.

2.3 Approche de Solution et Stratégie Technique : Exploration des Pistes et Méthodologies

Pour notre projet de développement d'une solution de reconnaissance vocale pour l'accessibilité fluide des personnes à déficience visuelle dans l'application eBanking.

1. **Théoriquement, la solution doit ressembler principalement à :**

- **Prétraitement du signal** : Le signal vocal capturé est soumis à un prétraitement pour filtrer le bruit, normaliser le volume et optimiser la qualité du son, garantissant des données d'entrée propres et de haute qualité.
- **Modèle de Reconnaissance Vocale (Speech-to-Text)** : Le signal vocal prétraité est envoyé à un modèle de reconnaissance vocale, basé sur des réseaux neuronaux profonds, pour convertir l'audio en texte, assurant une transcription précise et rapide.
- **Modèle de compréhension du langage naturel (NLP)** : Le texte généré par le modèle de reconnaissance vocale est ensuite analysé par un modèle NLP avancé, qui comprend et interprète le langage naturel, permettant une compréhension contextuelle et sémantique précise.
- **Appel aux APIs et intégration des services externes** : Les entités et les actions extraites depuis le texte analysé seront utilisé pour interagir avec des APIs externes et des services bancaires.
- **Modèle de synthèse vocale (Text-to-Speech)** : Après le traitement et l'interaction avec les APIs, le texte résultant est envoyé à un modèle de synthèse vocale avancé, qui le convertit en une voix synthétique naturelle et expressive, assurant une expérience d'écoute agréable et immersive pour l'utilisateur.



FIGURE 20 – Solution Théorique Globale

2. En pratique, un large éventail d'approches s'offre à nous, que l'on peut regrouper en trois catégories principales :

(a) **Approches basées uniquement sur des APIs externes :**

Architecture Globale

Cette approche utilise des services cloud à chaque étape du traitement, de la reconnaissance vocale à la synthèse vocale. La commande vocale est captée par une API de reconnaissance vocale, suivie d'un prétraitement textuel incluant le nettoyage et la normalisation. La compréhension du langage naturel est réalisée à l'aide d'API de classification des intentions et d'extraction d'entités. L'interaction avec l'API bancaire est assurée par un middleware d'intégration sécurisé. Les réponses textuelles sont générées grâce à des APIs NLP de génération de texte, tandis que la synthèse vocale est effectuée via des APIs de synthèse vocale.

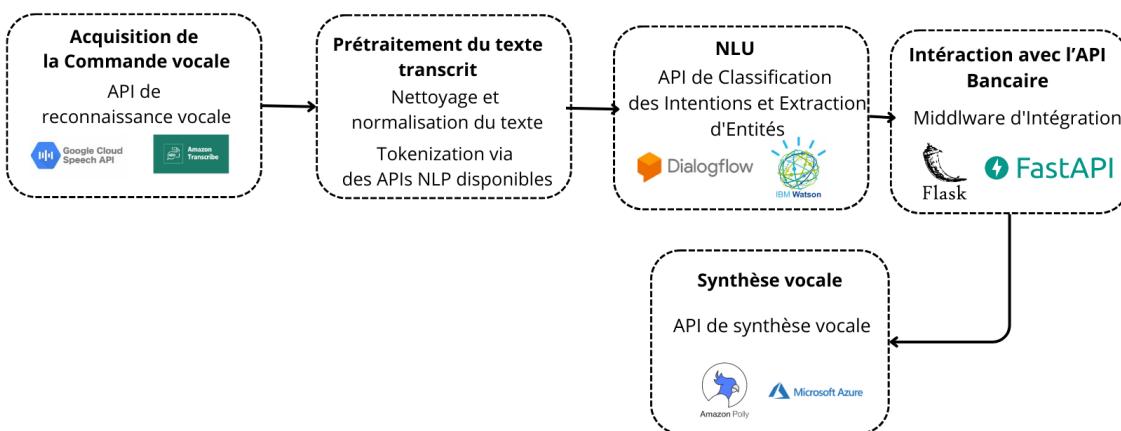


FIGURE 21 – Architecture globale du système : Approche basée sur des APIs uniquement

(b) **Approches basées uniquement sur des modèles développés en interne :**

Architecture Globale

Cette approche offre un contrôle total sur chaque étape du flux de traitement. Elle repose sur l'utilisation de modèles open-source pour la reconnaissance vocale, suivie d'un prétraitement du texte via des scripts internes pour le nettoyage et la tokenization. La compréhension du langage naturel est réalisée à l'aide de modèles internes pour la classification des intentions et l'extraction des entités. L'interaction avec l'API bancaire est gérée par un middleware interne. Les réponses textuelles et la synthèse vocale sont également générées par des modèles internes.

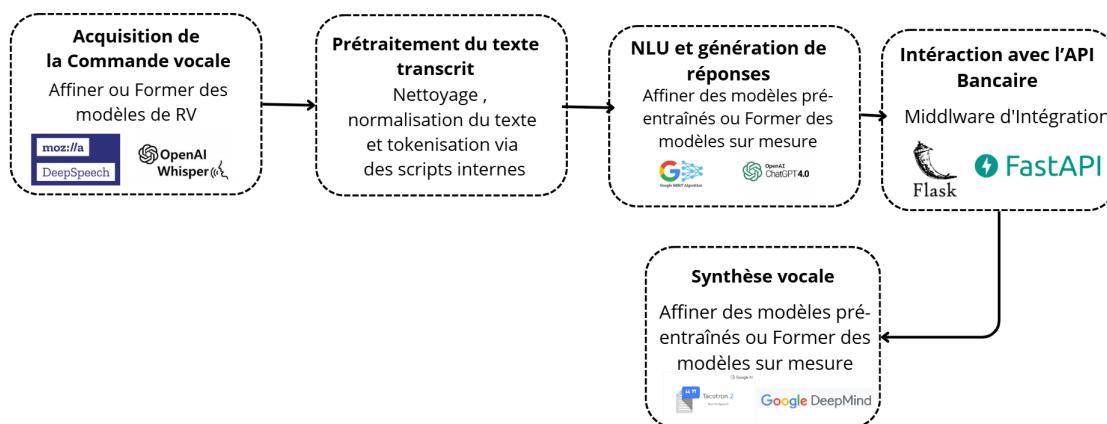


FIGURE 22 – Architecture globale du système : Approche basée uniquement sur des modèles développés en interne

(c) **Approches hybrides qui combinent des APIs externes et des modèles développés en interne :**

Architecture Globale

L'approche hybride combine des services cloud pour certaines fonctionnalités et des modèles internes pour d'autres. Cela implique l'utilisation d'APIs externes pour convertir la parole en texte, ainsi que des modèles internes pour classifier les intentions et extraire les entités. Un middleware interne assure l'interaction avec l'API bancaire. Les réponses sont générées à l'aide de modèles internes ou d'APIs externes, tandis que la synthèse vocale est effectuée à l'aide d'APIs externes.

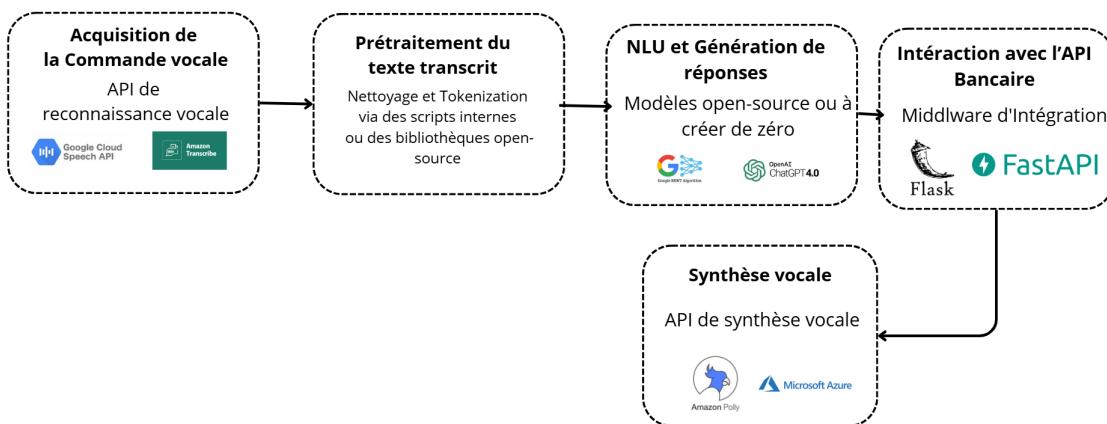


FIGURE 23 – Architecture globale du système vocal : Approche hybride

3. Comparaison des 3 approches :

Critères	Approches basées uniquement sur des APIs	Approches hybrides	Approches basées uniquement sur des modèles internes
Précision	Dépend de la qualité des APIs externes	Très bonne	Excellente
Personnalisation	Limitée	Moyenne	Haute
Besoin de données	Faible	Moyen	Élevé
Rapidité de mise en oeuvre	Rapide	Moyenne	Lente
Maintenance	Faible (dépendance externe)	Moyenne	Élevée
Coût	Variable	Modéré	Élevé (développement et maintenance)

TABLE 3 – Comparaison des différentes approches d’architecture globale

4. **Considérations et approche préviliégiée :** Lors de la sélection de l’architecture globale pour le traitement des commandes vocales, il est crucial de considérer que :

- ✓ les données d’entraînement bancaires conversationnels, qu’elles soient vocales ou textuelles, ne sont pas actuellement disponibles pour l’entreprise. Elles doivent donc être créées ou recherchées.
- ✓ la précision de la compréhension des commandes vocales doit être élevée.
- ✓ le système doit fournir des réponses en temps réel avec une faible latence.
- ✓ le temps de développement doit être réduit.
- ✓ le système doit être adéquat à la langue française.
- ✓ coût ne doit pas être élevé.
- ✓ système doit être facilement intégrable dans l’infrastructure existante.

En considération de ces contraintes ainsi que la comparaison des pistes possibles, nous avons opté pour l’approche reposant sur les interfaces de programmation (APIs) et les modèles de traitement du langage naturel (NLP). Nous projetons de concevoir une application mobile et d’utiliser les plugins du framework utilisé dans le développement mobile pour effectuer la conversion entre le texte et la parole, tout en élaborant un système sur mesure pour interpréter le texte produit par l’utilisateur , comme illustré sur la figure suivante.

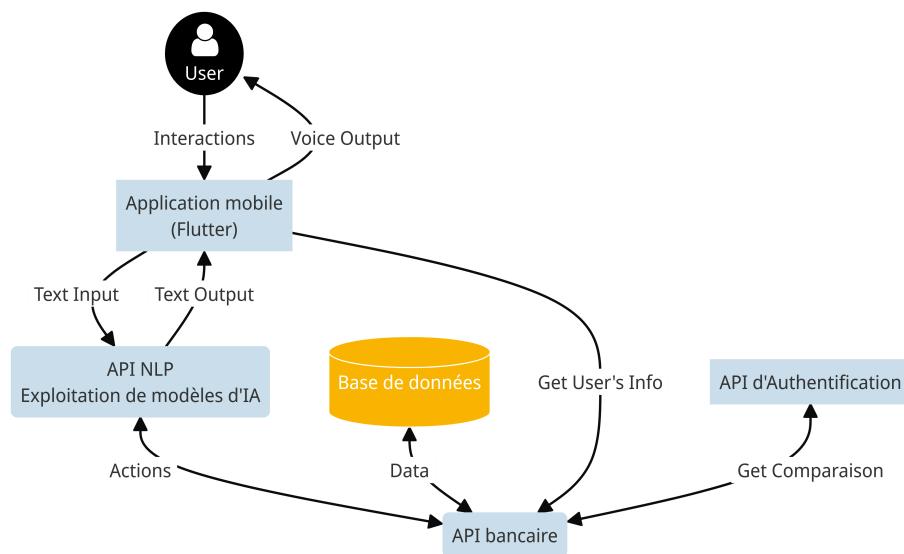


FIGURE 24 – Solution Pratique Proposée

2.4 Outils utilisés

2.4.1 Volet Data Science

Nous avons choisi **Google Colab** pour ses fonctionnalités avancées qui répondent parfaitement à nos besoins en matière de traitement, de visualisation et de préparation de données textuelles et audio, ainsi que pour l'entraînement de nos modèles. Son accès à des ressources informatiques puissantes nous permet d'exécuter nos tâches de manière efficace, tandis que sa facilité d'utilisation et sa compatibilité avec les bibliothèques populaires de machine learning en font un choix idéal pour notre projet.¹



FIGURE 25 – Logo Google Colab

Framework de développement des modèles :

Pour entraîner nos modèles, nous devions sélectionner le framework de développement approprié, tant pour le traitement du langage naturel que pour l'authentification biométrique basée sur la voix. Nous avons donc évalué deux principaux frameworks : PyTorch et TensorFlow. Bien que les deux soient réputés dans le domaine du NLP et de la biométrie vocale, notre cas d'utilisation spécifique privilégie PyTorch pour plusieurs raisons à savoir la grande précision, la réponse en temps réel et la grande flexibilité.

1. Il est également important de noter que les données utilisées ne sont pas sensibles, ce qui signifie que l'entraînement dans le cloud ne pose aucun problème de confidentialité ou de sécurité.



FIGURE 26 – Logo PyTorch et Tensorflow

Critères	Tensorflow	PyTorch
Précision	Généralement comparable à PyTorch, peut exceller dans certaines tâches spécifiques.	Souvent légèrement plus précis que TensorFlow, en particulier pour les tâches de vision par ordinateur et de traitement du langage naturel.
Temps réel	Moins adapté aux applications temps réel en raison de sa structure graphe statique et de son overhead de calcul.	Mieux adapté aux applications temps réel grâce à son graphe dynamique et à ses optimisations pour le déploiement sur des appareils mobiles et embarqués.
Personnalisation	Offre une grande flexibilité pour la personnalisation des modèles et des architectures de réseau neuronal.	Permet une personnalisation aisée des modèles et des architectures, avec un accent sur la simplicité et la rapidité de développement.
Compatibilité avec d'autres frameworks	Intégration native avec d'autres bibliothèques Google comme Keras et TPUs.	Prise en charge croissante de l'interopérabilité avec d'autres frameworks, mais moins mature que TensorFlow.
Facilité d'utilisation	Courbe d'apprentissage plus élevée en raison de sa syntaxe complexe et de son approche basée sur les graphes statiques.	Plus convivial pour les débutants grâce à sa syntaxe Python naturelle et à son approche dynamique des graphes.

TABLE 4 – Comparaison des différents Frameworks pour l'entraînement des modèles

Dans ce qui suit, nous allons présenter les bibliothèques que nous avons utilisées pour former les modèles.

Pour la manipulation de données :

- Nous avons opté pour **pandas** et **numpy** en raison de leurs performances optimisées, de leurs fonctionnalités de manipulation des données (charger les données à partir de diverses sources, les nettoyer, les filtrer, les regrouper, les agréger et les transformer à travers diverses manières), de leur compatibilité avec PyTorch et d'autres bibliothèques et de leur capacité à gérer efficacement les données manquantes et les valeurs aberrantes.



FIGURE 27 – Logos des bibliothèques pour la manipulation des données

Pour la visualisation de données

- **Seaborn** bibliothèque basée sur Matplotlib conçue pour créer des visualisations statistiques attrayantes et informatives. Elle offre une large gamme de styles et de palettes de couleurs prédéfinis, ce qui facilite la création de graphiques nets et élégants. **Seaborn** nous est particulièrement utile quant à la visualisation de données relationnelles et à la création de graphiques statistiques tels que des diagrammes à barres, les spectrogrammes d'audios et les matrices de confusion.
- **matplotlib** : bibliothèque de visualisation de données de base et polyvalente pour Python. Elle offre un contrôle précis sur la création de graphiques et permet de personnaliser presque tous les aspects d'un graphique. Nous l'avons employé pour tracer les courbes d'apprentissage , les courbes ROC-AUC et les matrices de confusion.
- **wordcloud** : bibliothèque pour générer des nuages de mots présents dans les données d'entraînement. wordcloud nous a été particulièrement utile pour analyser notre corpus de texte et identifier les mots les plus importants et les plus fréquents.



FIGURE 28 – Logos des bibliothèques de la visualisation de données

Prétraitement de texte et NLP

- **Prétraitement de texte :**

- Nous avons combiné les deux bibliothèques **nltk (Natural Language Toolkit)** et **Spacy** et l'outil **french_lefff_lemmatizer** pour une couverture complète des tâches de prétraitement pour le NLP. Spacy est reconnu pour sa rapidité et ses performances optimisées vu le gros volume du texte, NLTK offre des ressources linguistiques riches comme WordNet et french_lefff_lemmatizer spécialisé dans la lemmatisation du texte en français ce qui permet d'obtenir des formes lemmatisées correctes et uniformes pour le français.
- **transformers** : cette bibliothèque offre un ensemble puissant d'outils et de fonctionnalités qui simplifient l'utilisation des modèles PyTorch de Hugging

Face pour des tâches de NLP, notamment pour des modèles de traitement du langage naturel de pointe comme **CamemBERT** qu'on veut explorer.

- **TfidfVectorizer** : Pour notre tâche de classification textuelle, **TfidfVectorizer** est privilégié pour sa simplicité et son efficacité. Il offre une vectorisation rapide adaptée aux grands ensembles de données et met en avant les mots importants grâce à la pondération TF-IDF. De plus, il normalise le texte, le rendant robuste aux variations de style d'écriture. Son intégration facile avec scikit-learn en fait un choix polyvalent et compatible avec divers modèles de classification.



FIGURE 29 – Logos des bibliothèques NLP

- **Pipeline NLP pour les FAQs :**

- **LangChain** Une bibliothèque python utilisée pour notre pipeline FAQ, exploitant le modèle de langage Gemini pour créer des workflows complexes. Elle inclut des modules comme RetrievalQA, qui facilite la recherche vectorielle et la récupération de réponses pertinentes en identifiant des questions similaires dans une base de connaissances FAQ.
- **LangChain Community** Propose des modules axés sur la communauté, tels que HuggingFaceInstructEmbeddings. Ce module utilise les modèles de langage de Hugging Face pour offrir des embeddings vectoriels de haute qualité, capturant le sens des questions et des réponses, et permettant de choisir parmi une large sélection de modèles pré-entraînés.
- **LangChain Google GenAI** Intègre des fonctionnalités basées sur les modèles de langage de Google AI, comme le module GoogleGenerativeAI. Ce module est conçu pour la génération de texte, permettant de produire des réponses complètes et du contenu textuel de haute qualité, similaire au langage humain, enrichissant ainsi les capacités de LangChain en génération de contenu.



FIGURE 30 – Logos des bibliothèques pour la pipeline FAQ

Modélisation et machine learning

En plus de PyTorch nous avons employé également **Sklearn** :

- **Scikit-learn (sklearn)** se distingue par ses capacités étendues et sa simplicité en prétraitement, entraînement et évaluation de modèles de machine learning, particulièrement pour le NLP et les signaux.
 - **Prétraitement** scikit-learn propose une variété de transformateurs pour normaliser, standardiser et transformer les données, ce qui est crucial pour préparer les données de texte et de signaux audio de manière efficace.
 - **Model Selection** utilise des outils tels que **GridSearchCV** particulièrement utile pour l'optimisation des hyperparamètres, automatisant la recherche des meilleures configurations de modèles. La fonction **train_test_split** facilite la division des données en ensembles d'entraînement et de test, garantissant des évaluations de performances fiables.
 - **Métriques** scikit-learn fournit une gamme complète de mesures d'évaluation pour les modèles de classification. Des métriques telles que l'accuracy, la précision, le rappel et le F1-score, la matrice de confusion et le ROC-AUC permettent une évaluation détaillée des performances des modèles.
 - Nous aurons besoin également des méthodes d'ensemble comme **RandomForest** et **GradientBoosting** qui sont intégrées dans scikit-learn offrent des solutions puissantes pour améliorer les performances des modèles en combinant les prédictions de plusieurs modèles de base.



FIGURE 31 – Logos des bibliothèques pour entraînement des modèles

Traitement et analyse de signaux

Nous avons sélectionné Librosa et SciPy pour leur ensemble complet d'outils dédiés au prétraitement et à l'extraction de caractéristiques des signaux audio. Leurs API intuitives et flexibles facilitent la manipulation des signaux et permettent une personnalisation aisée. Ces bibliothèques, optimisées pour des performances efficaces, peuvent gérer de grands

ensembles de données et former des modèles complexes rapidement. De plus, leur intégration transparente avec des frameworks de machine learning comme PyTorch facilite la construction et l'entraînement de modèles profonds. Elles bénéficient également de communautés actives et de nombreuses ressources, offrant un soutien solide pour notre projet.



FIGURE 32 – Logo de bibliothèque pour traitement des signaux vocaux

2.4.2 Volet Développement Logiciel

1. Système de Gestion de Base de Données

Les données de notre solution sont structurées, ce qui signifie qu'elles sont organisées de manière cohérente et ordonnée. Cette structuration est essentielle car nous avons besoin de données qui soient facilement accessibles et interrogables pour notre projet. Ainsi, afin de sélectionner le meilleur SGBD, nous avons élaboré le tableau suivant :

Fonctionnalités	Postgresql	MySQL	MariaDB
Type de licence	Open source (GPL)	Open source (GPL)	Open source (GPL)
Structure des données	Relationnelle objet	Relationnelle	Relationnelle
Performances et évolutivité	Elevées	Bonnes	Bonnes
Fiabilité	Très élevée	Bonne	Bonne
Fonctionnalités	Riches, nombreux types de données et fonctionnalités avancées	Bonnes, fonctionnalités de base solides	Bonnes, fonctionnalités de base solides avec des ajouts de la communauté
Compatibilité	Bonne compatibilité avec les standards SQL	Bonne compatibilité avec les standards SQL	Bonne compatibilité avec les standards SQL, excellente compatibilité avec MySQL
Facilité d'utilisation	Assez complexe	Facile à utiliser	Facile à utiliser
Support de la communauté	Grande et active	Grande et active	Grande et active
Intégration Spring Boot	Bonne	Bonne	Bonne
Cas d'utilisation	Applications critiques, analyses complexes, Big Data	Applications web, petites et moyennes entreprises	Applications web, petites et moyennes entreprises, compatible avec les applications MySQL existantes

TABLE 5 – Comparaison des différents SGBD disponibles

En se basant sur le tableau comparatif ci-dessus, PostgreSQL est un excellent choix pour les applications qui nécessitent une base de données performante, fiable et riche en fonctionnalités, notamment les applications critiques qui exigent une haute disponibilité et une prévention des pertes de données et les applications web évolutives qui doivent gérer de gros volumes de données et des pics de trafic.



FIGURE 33 – Logo Postgresql

2. Outils de développement de l'API bancaire

(a) **Framework de développement :** Comme le montre le tableau comparatif

suivant, Spring se distingue par son adaptabilité aux backends bancaires complexes comportant de nombreuses fonctionnalités et intégrations, nécessitant une authentification et une autorisation strictes. De plus, Spring est reconnu pour être un framework stable, mature et fiable.

Fonctionnalités	Spring Boot	FaqtAPI	ASP.Net
Langage de programmation	Java	Python	C#
Philosophie du framework	Convention plutôt que configuration	Performance et simplicité	Orienté objet et complet
Maturité du framework	Mature et stable	En plein essor et dynamique	Mature et bien établi
Performances	Bonnes performances	Excellent performances	Bonnes performances
Documentation	Documentation complète et abondante	Documentation croissante et active	Documentation officielle et communautaire étendue
Communauté	Grande communauté active	Communauté en pleine expansion	Grande communauté active
Intégration à la base de données bancaire	Prise en charge JDBC native, frameworks ORM matures comme Hibernate	Accès aux bases de données via SQLAlchemy, Asyncpg et autres	Prise en charge native d'Entity Framework Core et d'autres LINQ to SQL
Sécurité	Sécurité robuste avec Spring Security	Intégration facile des bibliothèques de sécurité tierces	Sécurité basée sur .NET Framework et bibliothèques tierces

TABLE 6 – Comparaison des frameworks web disponibles pour l'API RESTful

Nous avons choisi donc le framework Spring Boot pour le développement de notre API RESTful.



FIGURE 34 – Logo Spring Boot

- (b) **Environnement de développement :** Le tableau suivant présente une comparaison détaillée des fonctionnalités entre les environnements de développement intégré (IDE) IntelliJ IDEA, Visual Studio Code et Eclipse. Il met en évidence des aspects tels que la plateforme prise en charge, le langage de

programmation, le support spécifique pour Spring Boot et PostgreSQL, les fonctionnalités de l'IDE, ainsi que les prix des différentes options.

Fonctionnalités	IntelliJ Idea	Visual Studio Code	Eclipse
Plateforme	Multiplateforme (Windows, macOS, Linux)	Multiplateforme (Windows, macOS, Linux)	Multiplateforme (Windows, macOS, Linux)
Langage de programmation	Java, Kotlin, Scala, C++, Python, JavaScript, etc.	JavaScript, TypeScript, Python, Java, C++, etc.	Java, C++, Python, PHP, JavaScript, etc.
Support Spring Boot	Excellent support avec des fonctionnalités spécifiques.	Extensions tierces disponibles pas d'intégration native complète	Extensions tierces disponibles pas d'intégration native complète
Support PostgreSQL	Excellent support pour PostgreSQL avec des fonctionnalités spécifiques	Extensions tierces disponibles pour le support PostgreSQL pas d'intégration native complète	Extensions tierces disponibles pour le support PostgreSQL, pas d'intégration native complète
Fonctionnalités IDE	Large éventail de fonctionnalités IDE avancées	Large éventail d'extensions disponibles pour étendre les fonctionnalités	Large éventail de plugins disponibles pour étendre les fonctionnalités
Prix	Édition communautaire gratuite et Licence pour d'autres options	Gratuit et open source	Gratuit et open source

TABLE 7 – Comparaison des environnements de développement pour API RESTful Spring Boot disponibles

IntelliJ IDEA est un choix idéal car il est un IDE complet et riche en fonctionnalités, avec un excellent support natif pour Spring Boot et PostgreSQL. Comparé à Visual Studio Code et Eclipse, IntelliJ IDEA offre une expérience utilisateur plus intégrée et cohérente. Ses outils de refactoring intelligents, sa navigation de code avancée et ses capacités de débogage améliorées surpassent celles de VS Code et Eclipse. De plus, IntelliJ IDEA fournit une gestion de projet plus robuste et une intégration plus fluide avec des technologies Java, ce qui en fait un environnement de développement particulièrement puissant pour les projets complexes d'entreprise comme le nôtre.

3. Outils de développement de l'app mobile

- (a) **Framework de développement :** Nous avons mené une étude comparative des différents frameworks de développement mobile et nous avons compilé les résultats dans le tableau ci-dessous :



FIGURE 35 – Logo IntelliJ IDEA

Critères	Flutter (Dart)	React Native (JavaScript)	Xamarin (C#)	Ionic (JavaScript/TypeScript)
Performance	Élevée	Variable	Élevée	Moyenne
Communauté	Grandissante, mais moins grande que React Native	Large et active	Moyenne	Large et active
Langage de programmation	Dart	JavaScript	C	JavaScript-TypeScript
Coût	Gratuit	Gratuit	Licence payante pour certaines fonctionnalités	Gratuit
Intégration des plugins IA	Bonne	Bonne	Bonne	Moyenne
Accessibilité	Bonne, avec des widgets natifs et des plugins	Bonne, avec des bibliothèques tierces	Bonne, avec des bibliothèques tierces	Bonne, avec des bibliothèques tierces

TABLE 8 – Comparaison des Frameworks mobile disponibles

Flutter s'impose comme le choix idéal pour le développement de notre application mobile bancaire en surpassant React Native, Xamarin et Ionic. Grâce à notre familiarité avec le framework, son modèle open-source, et son riche écosystème de plugins, Flutter permet un développement agile et efficace. Ses performances élevées et son hot reload facilitent les itérations rapides. Contrairement à React Native, Xamarin et Ionic, Flutter offre une expérience utilisateur fluide et native, soutenue par une communauté dynamique et un support actif de Google, garantissant un développement robuste et évolutif.



FIGURE 36 – Logo Flutter

(b) Environnement de développement :

Pour éclairer notre choix d'environnement de développement pour notre projet, nous avons dressé un tableau comparatif des fonctionnalités d'Android Studio, IntelliJ IDEA et VS Code.

Fonctionnalités	Android Studio	IntelliJ IDEA	VS Code
IDE officiel pour Flutter	Oui	Non	Non
Intégration avec les outils Android	Excellente	Bonne	Bonne
Fonctionnalités d'analyse de code	Bonnes	Excellentées	Bonnes
Fonctionnalités de refactoring	Bonnes	Excellentées	Bonnes
Fonctionnalités de débogage	Bonnes	Excellentées	Bonnes
Interface utilisateur	Personnalisable	Personnalisable	Personnalisable
Extensions	Large choix	Large choix	Large choix
Légereté	Moyen	Moyen	Léger
Gratuit et open-source	Oui	Certaines éditions	Oui

TABLE 9 – Comparaison des environnements de développement mobile disponibles

En somme, Android Studio se distingue comme le choix de prédilection pour le développement de notre application mobile Flutter, notamment pour cette application destinée à Android. Cette préférence découle de son intégration harmonieuse avec l'écosystème Android, de son environnement dédié spécifiquement à Flutter et du soutien officiel de Google. En comparaison, IntelliJ IDEA et VS Code, bien qu'ils offrent également des fonctionnalités de développement pour Flutter, ne proposent pas la même intégration native avec l'écosystème Android, ni le même niveau de support officiel de la part de Google.



FIGURE 37 – Logo Android Studio

4. Outils de développements de l'API NLP

- (a) **Framework de développement :** Le tableau ci-après compile les différences entre les frameworks de développement pour l'API NLP et l'API d'authentification biométrique pour mieux évaluer les options disponibles :

Fonctionnalités	FastAPI	Flask	Starlette
Performance	Rapide et performant, idéal pour les applications gourmandes en ressources	Léger et performant pour les projets de petite à moyenne envergure	Performant et modulaire, adapté aux applications évolutives
Architecture	Asynchrone basée sur ASGI	Structure MVC (Model-View-Controller)	Asynchrone basée sur ASGI
Fonctionnalités API	Validation de schéma, documentation automatique, prise en charge des dépendances injectables	Extensions tierces pour diverses fonctionnalités	Prise en charge native des dépendances injectables et des fonctionnalités avancées d'API RESTful
Déploiement de modèles PyTorch	Intégration transparente avec PyTorch	Extensions comme Flask-RESTful ou Flask-API	Intégration transparente avec PyTorch
Complexité	Plus complexe à prendre en main	Simple et flexible	Modérément complexe, équilibre entre performance et simplicité
Cas d'utilisation	Microservices, API complexes, applications gourmandes en ressources	Projets d'API web simples à intermédiaires	API modernes, applications évolutives

TABLE 10 – Comparaison des Frameworks de développement des APIs exploitant les modèles PyTorch

FastAPI se positionne donc comme le choix idéal pour le développement de nos APIs. Son architecture asynchrone basée sur ASGI garantit une performance et une évolutivité optimales, essentielles pour gérer les interactions fréquentes et gourmandes en ressources. Ses fonctionnalités API avancées simplifient le développement et la maintenance de l'API. L'intégration transparente avec PyTorch facilite le déploiement et l'exposition des modèles NLP et de speaker recognition. Enfin, ses fonctionnalités de sécurité intégrées et sa robustesse garantissent la protection et la fiabilité de l'API, des aspects cruciaux pour les applications bancaires.



FIGURE 38 – Logo FastAPI

(b) **Environnement de développement :**

Le tableau de comparaison ci-dessous vise à éclairer notre choix en mettant en lumière les principales différences entre les outils de développement PyCharm, VS Code et IntelliJ Idea pour les API FastAPI.

Fonctionnalités	PyCharm	VS Code	IntelliJ IDEA
Support natif pour FastAPI	Oui	Extension requise	Extension requise
Débogage et profilage	Intégré et puissant	Extensions disponibles	Intégré et puissant
Refactoring et auto-complétion	Avancé et personnalisable	Extensions disponibles	Avancé et personnalisable
Intégration Git	Intégrée et complète	Extensions disponibles	Intégrée et complète
Gestion des dépendances	Prise en charge intégrée pour pip et autres	Extensions disponibles	Prise en charge intégrée pour Maven, Gradle et autres
Extensions et personnalisation	Large éventail d'extensions disponibles	Marketplace d'extensions vaste	Large éventail d'extensions disponibles
Prix	Licence payante avec options communautaires et professionnelles	Gratuit avec extensions payantes pour certaines fonctionnalités	Licence payante avec options communautaires et ultimes

TABLE 11 – Comparaison des environnements de développement d'API FastAPI

D'après le tableau comparatif , PyCharm est l'outil de développement de premier plan pour les API FastAPI. Il offre un support complet, des fonctionnalités avancées telles que le débogage et le profilage, ainsi que des outils de refactoring et d'auto-complétion intelligents. L'intégration native de FastAPI simplifie le processus de développement, tandis que l'intégration Git et la gestion des dépendances garantissent une collaboration fluide et un environnement stable. Comparé à d'autres outils comme VS Code et IntelliJ IDEA, PyCharm se distingue par sa spécialisation pour Python, ses fonctionnalités avancées et son ergonomie pour le développement d'applications web basées sur ce langage.



FIGURE 39 – Logo PyCharm

2.4.3 Volet Test

Tests unitaires :

1. JUnit et Mockito

JUnit et **Mockito** sont deux outils essentiels pour tester notre API RESTful Spring Boot. **JUnit**, un framework de test Java largement utilisé, fournit des annotations et des assertions pour structurer et évaluer nos tests. Il est utilisé pour évaluer chaque composant de notre API de manière isolée, garantissant son bon fonctionnement. D'autre part, **Mockito**, une bibliothèque de mocking open-source, simplifie la création de mocks, des objets factices qui simulent le comportement d'objets réels dans nos tests. Grâce à **Mockito**, nous pouvons tester nos composants en isolant leurs dépendances et en simulant des scénarios de test précis. En combinant **JUnit** et **Mockito**, nous maintenons une haute qualité de code, détectons et corigeons rapidement les bugs, et nous assurons que notre API répond aux exigences fonctionnelles et de performance avant son déploiement.



FIGURE 40 – Logo JUnit-Mockito

2. Mocktail Flutter

Pour tester notre application mobile , nous avons employé la bibliothèque moqueuse Dart **Mocktail**. Spécifiquement conçue pour Flutter, elle simplifie le testing des widgets, services et providers. Avec Mocktail, créer des mocks pour les dépendances de l'application devient simple, permettant de simuler divers scénarios sans configurations complexes. Pour les tests de widgets, elle isole les composants de l'interface utilisateur, assurant leur bon fonctionnement indépendamment des autres parties de l'application. En ce qui concerne les services et les providers, Mocktail facilite le test des interactions et des réponses, garantissant leur bon fonctionnement et la maintenabilité des états. En utilisant Mocktail, nous améliorons la couverture des tests, détectons et corigeons les bugs plus tôt dans le cycle de développement, et assurons la robustesse et la fiabilité de notre application Flutter avant le déploiement.



FIGURE 41 – Logo Mocktail Flutter

3. Unittest et MagicMock Afin d'assurer la qualité et la fiabilité de nos API FastAPI, nous avons adopté une approche méticuleuse en mettant en place des tests unitaires exhaustifs. Pour ce faire, nous avons fait appel à des bibliothèques de test renommées telles que unittest et MagicMock. Grâce à ces outils, nous avons pu examiner chaque composant de nos API de manière approfondie, en les isolant et en vérifiant leur comportement dans une variété de scénarios. Cette démarche proactive a été essentielle pour identifier et corriger rapidement d'éventuels problèmes, garantissant ainsi la robustesse et la stabilité de nos API FastAPI.

Tests d'intégration :

PyTest est un outil de test Python reconnu pour sa simplicité, sa flexibilité et ses fonctionnalités avancées. Idéal pour tester nos API FastAPI, il simplifie la création et l'exécution de tests unitaires, fonctionnels et d'intégration. Grâce à ses capacités, PyTest nous permet d'écrire des tests clairs et maintenables, même pour des scénarios complexes. Pour les tests d'intégration, il assure une interaction harmonieuse entre les différentes parties de l'API, en simulant des requêtes HTTP et en vérifiant les réponses et les états du système. En utilisant des fixtures et des mocks, PyTest nous aide à configurer facilement des environnements de test et à simuler des interactions avec les APIs RESTful et les modèles, garantissant ainsi la fiabilité de notre API avant son déploiement.



FIGURE 42 – Logo PyTest

Tests d'API et documentation :

1. Postman :

Nous avons opté pour l'outil de développement API Postman afin de tester nos APIs

FastAPI et RESTful, en raison de sa richesse en fonctionnalités et de sa convivialité. Postman simplifie la conception, le test, la documentation et le partage des APIs en offrant une interface utilisateur intuitive pour créer et envoyer des requêtes HTTP, ainsi que pour analyser les réponses et automatiser les tests. Il prend en charge différents types de requêtes (GET, POST, PUT, DELETE, etc.) et permet de sauvegarder des collections de requêtes pour une utilisation répétée et organisée. Avec la possibilité de créer des requêtes complexes comprenant des paramètres, des en-têtes et des corps de requête personnalisés, Postman rend le processus de test plus simple et plus efficace. En résumé, c'est un choix idéal pour garantir le bon fonctionnement et la fiabilité de nos APIs.



FIGURE 43 – Logo Postman

2. **Swagger** : est un ensemble d'outils conçu pour simplifier la conception, la documentation et le test des APIs de manière efficace. Nous avons choisi Swagger pour nos APIs en raison de sa capacité à générer automatiquement une documentation claire et à jour à partir du code source de l'API. Son interface utilisateur conviviale, Swagger UI, permet une exploration facile des endpoints de l'API et facilite le processus de test. En somme, Swagger offre une solution complète pour concevoir, documenter et tester nos APIs, améliorant ainsi la compréhension et l'adoption de notre API par les développeurs et les utilisateurs.



FIGURE 44 – Logo Swagger

2.5 Conclusion

Dans ce chapitre, nous avons examiné l'état de l'art en matière de traitement de données textuelles et audio, mettant en lumière les avancées récentes et les meilleures pratiques dans le domaine. Nous avons détaillé ensuite notre approche de solution et stratégie technique, en explorant différentes pistes et méthodologies pour aborder notre problématique

de manière efficace et innovante Enfin, nous avons présenté les outils de travail sélectionnés pour notre projet, en justifiant leur choix par rapport à d'autres alternatives et leur compatibilité avec nos besoins spécifiques. L'ensemble de ces éléments constitue une base solide pour le développement et la mise en œuvre de notre solution, objet des prochains chapitres, garantissant une efficacité optimale et une performance élevée.

Chapitre III : Analyse et Conception

3 Analyse et Conception

3.1 Introduction

Dans ce chapitre, nous abordons la phase d'analyse et de conception du système, qui se décompose en deux étapes principales. Tout d'abord, nous amorçons la spécification des besoins en employant des diagrammes UML tels que les diagrammes de cas d'utilisation. Cette approche nous permet de définir de manière claire et précise les fonctionnalités attendues de l'application. Ensuite, nous procérons à la conception générale en utilisant des diagrammes de séquence, de classes et d'activité. Ces diagrammes nous aident à décrire les interactions entre les différents composants du système, à modéliser les structures de données et à visualiser le flux des activités.

3.2 Analyse de besoins et modélisation des cas d'utilisation

3.2.1 Capture des Besoins

Pour capturer les besoins des utilisateurs non-voyants dans l'application e-banking, nous avons mené une étude de benchmarking et exploré les solutions existantes. Nous avons constaté que la plupart des solutions existantes proposent un large éventail de fonctionnalités, mais pour répondre aux besoins spécifiques des utilisateurs non-voyants, nous devons sélectionner les opérations bancaires les plus demandées par ces clients.

1. Les besoins fonctionnelles à répondre comprennent :

— Consultation des Informations de Comptes, de Cartes et des opérations :

- Consultation de Solde.
- Infos de Cartes y compris les numéros de carte, les codes pin, les dates d'expiration et les types de cartes.
- Infos d'Opérations Passées y compris les dates, les montants et les bénéficiaires des transactions passées.

— Gestion des Cartes et Leur Configuration :

- Activation et Désactivation de Services.
- Augmenter ou Diminuer le Plafond de Certains Services.
- Opposition de Carte.

— Paiement de Factures :

- Payer les Factures courantes.

— Gestion des Bénéficiaires de Transactions :

- Ajout de Bénéficiaires.
- Mettre à jour les informations des bénéficiaires existants.
- Retirer des bénéficiaires de la liste de contacts pour les transactions.

— **Virement vers des Comptes Attijariwafa et Confrères.**

— **Accès à l'Assistance et la FAQ :**

- **Assistance** : Contacter le service client pour obtenir de l'aide sur divers sujets.
- **FAQ** : Accéder à une base de connaissances pour trouver des réponses aux questions fréquentes.

— **Accès à la Géolocalisation des Agences :**

- Trouver et obtenir des directions vers les agences bancaires les plus proches.

2. **Les besoins non fonctionnelles** : Les besoins non fonctionnels concernent des aspects qui ne sont pas directement liés aux fonctionnalités métier mais qui sont essentiels pour le bon fonctionnement et la qualité globale du système. Voici les principaux besoins non fonctionnels de notre système :

- (a) **Sécurité** : Garantir la disponibilité, la confidentialité et l'intégrité de l'application.
- (b) **Performance** : Assurer la précision du traitement des demandes et la réponse en temps réduit.
- (c) **Convivialité** : Offrir une interface utilisateur intuitive et accessible pour garantir une expérience utilisateur agréable et facile à utiliser.
- (d) **Maintenabilité** : Assurer la facilité de maintenance pour permettre des mises à jour et des corrections rapides.

3.2.2 Diagramme des cas d'utilisation

En focalisant notre attention sur ces fonctionnalités, nous sommes en mesure de concevoir une expérience utilisateur adaptée aux utilisateurs non-voyants, en mettant en avant l'accessibilité, la simplicité d'utilisation et la pertinence des fonctionnalités pour répondre à leurs besoins particuliers. En conséquence, nous avons élaboré le diagramme des cas d'utilisation suivant qui met en évidence les fonctionnalités choisies , ainsi que les interactions l'utilisateur et le système :

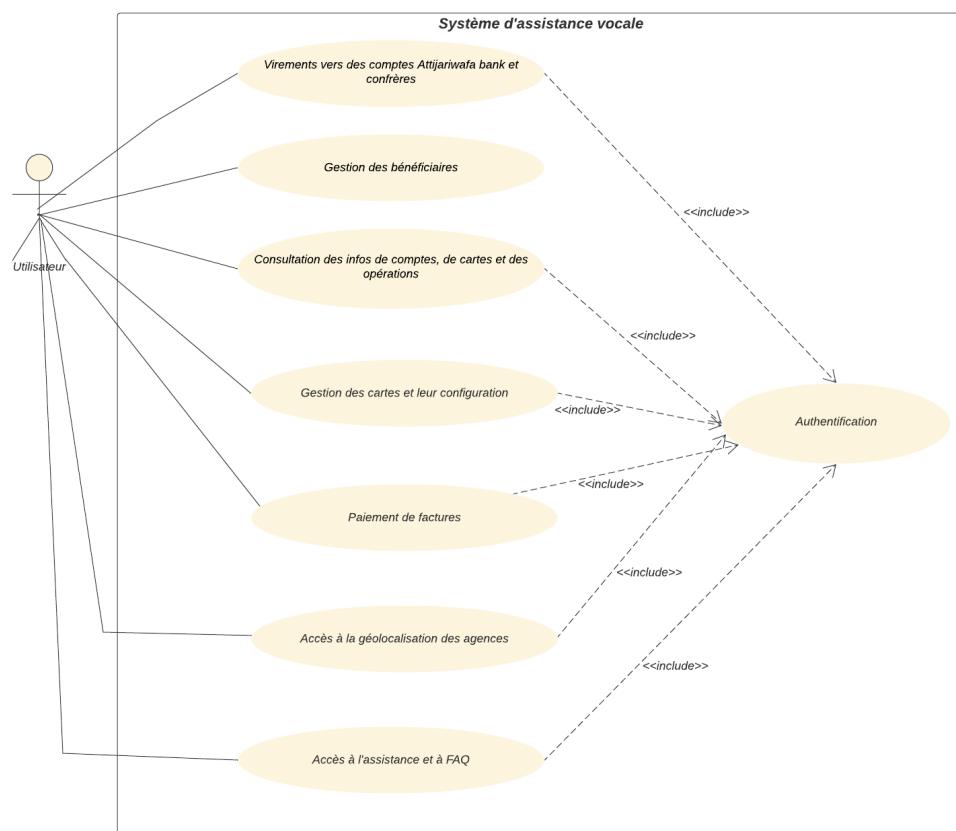


FIGURE 45 – Diagramme des cas d'utilisation

3.2.3 Description des cas d'utilisation

Les fiches suivantes décrivent en détail les cas d'utilisation.

Nom du Cas d'Utilisation	Accès à l'Assistance et la FAQ
Acteur Principal	Utilisateur final
Précondition	L'utilisateur doit être authentifié dans le système.
Résumé	Ce cas d'utilisation permet aux utilisateurs de contacter le service client pour obtenir de l'aide sur divers sujets et de répondre à leurs questions fréquentes. Cette fonctionnalité est accessible via une interface vocale.
Enchaînement Principal	<ol style="list-style-type: none"> 1. L'utilisateur lance la commande vocale pour accéder à l'assistance ou à la FAQ. 2. Le système convertit la commande vocale en texte. 3. Le système identifie l'intention de l'utilisateur (accéder à l'assistance ou à la FAQ). 4. Le système extrait les informations pertinentes de la base de données. 5. Le système génère une réponse vocale contenant les informations demandées.

TABLE 12 – Fiche descriptive du cas d'utilisation : Accès à l'Assistance et la FAQ

Nom du cas d'utilisation	Accès à la Géolocalisation des Agences
Acteur Principal	Utilisateur final
Précondition	<ul style="list-style-type: none"> L'utilisateur doit être authentifié dans le système. La fonctionnalité de géolocalisation doit être activée et disponible.
Résumé	Ce cas d'utilisation permet aux utilisateurs de trouver et d'obtenir des directions vers les agences bancaires les plus proches. Cette fonctionnalité est accessible via une interface vocale.
Enchaînement Principal	<ol style="list-style-type: none"> L'utilisateur lance la commande vocale pour accéder à la géolocalisation des agences. Le système convertit la commande vocale en texte. Le système identifie l'intention de l'utilisateur (trouver des agences bancaires proches). Le système utilise les services de géolocalisation pour trouver les agences les plus proches. Le système génère une réponse vocale contenant les directions vers les agences et peut diriger l'utilisateur vers la plus proche agence.

TABLE 13 – Fiche descriptive du cas d'utilisation : Accès à la Géolocalisation des Agences

Nom du Cas d'Utilisation	Consultation des Informations de Comptes, de Cartes et des Opérations
Acteur Principal	Utilisateur final
Précondition	L'utilisateur doit être authentifié dans le système.
Résumé	Ce cas d'utilisation permet aux utilisateurs de consulter des informations détaillées sur leurs comptes bancaires y compris leur solde, leurs cartes à savoir les codes pin , les dates d'expiration..., ainsi que l'historique de leurs transactions.
Enchaînement Principal	<ol style="list-style-type: none"> L'utilisateur lance la commande vocale pour accéder aux informations de son compte. Le système convertit la commande vocale en texte. Le système identifie l'intention de l'utilisateur dans ce cas consultation des informations de compte, de cartes ou des opérations à travers le texte transcrit. Le système extrait les informations pertinentes de la base de données. Le système génère une réponse vocale contenant les informations demandées.

TABLE 14 – Fiche descriptive du cas d'utilisation : Consultation des Informations de Comptes, de Cartes et des Opérations

Nom du Cas d'Utilisation	Gestion des Cartes et Leur Configuration
Acteur Principal	Utilisateur final
Précondition	L'utilisateur doit être authentifié dans le système.
Résumé	Ce cas d'utilisation permet aux utilisateurs de gérer et de configurer leurs cartes bancaires via une interface vocale. Les utilisateurs peuvent activer ou désactiver des services comme les paiements en ligne ou les retraits à l'étranger, ajuster les plafonds de dépenses ou de retrait, et opposer leurs cartes en cas de perte ou de vol.
Enchaînement Principal	<ol style="list-style-type: none"> 1. L'utilisateur initie la commande vocale pour gérer sa carte. 2. Le système convertit la commande vocale en texte. 3. Le système identifie l'intention de l'utilisateur (activation, désactivation de services, ajustement des plafonds, opposition de carte). 4. Le système exécute la demande de l'utilisateur en interagissant avec la base de données et les services bancaires, puis demande à l'utilisateur de confirmer l'action demandée. 5. L'utilisateur reçoit l'état de l'action via la synthèse vocale en cas de confirmation.

TABLE 15 – Fiche descriptive du cas d'utilisation : Gestion des Cartes et Leur Configuration

Nom du Cas d'Utilisation	Paiement de Factures
Acteur Principal	Utilisateur final
Précondition	<ul style="list-style-type: none"> • L'utilisateur doit être authentifié dans le système. • L'utilisateur doit avoir des fonds suffisants pour effectuer le paiement.
Résumé	Ce cas d'utilisation permet aux utilisateurs de payer leurs factures courantes, telles que les factures d'électricité, d'école, etc.. en utilisant une interface vocale.
Enchaînement Principal	<ol style="list-style-type: none"> 1. L'utilisateur lance la commande vocale pour payer une facture. 2. Le système convertit la commande vocale en texte. 3. Le système identifie l'intention de l'utilisateur (paiement de facture) à travers le texte transcrit. 4. L'utilisateur sélectionne la facture à payer (électricité, école, etc.) via une interaction vocale. 5. Le système demande une confirmation du montant et du bénéficiaire. 6. Le paiement est effectué après confirmation. Le système génère une réponse vocale confirmant le paiement.

TABLE 16 – Fiche descriptive du cas d'utilisation : Paiement de Factures

Nom du Cas d'Utilisation	Gestion des Bénéficiaires de Transactions
Acteur Principal	Utilisateur final
Précondition	<ul style="list-style-type: none"> • L'utilisateur doit être authentifié dans le système.
Résumé	Ce cas d'utilisation permet aux utilisateurs d'ajouter, de modifier et de supprimer les bénéficiaires de leurs transactions bancaires via une interface vocale accessible.
Enchaînement Principal	<ol style="list-style-type: none"> 1. L'utilisateur lance la commande vocale pour gérer les bénéficiaires. 2. Le système convertit la commande vocale en texte, puis identifie l'intention de l'utilisateur (ajout, modification ou suppression de bénéficiaires) à travers le texte transcrit. 3. Le système extrait les informations pertinentes de la base de données. 4. Le système demande à l'utilisateur de confirmer sa demande. 5. Après confirmation, le système effectue les opérations de gestion des bénéficiaires, puis génère une réponse vocale de l'état de l'opération.

TABLE 17 – Fiche descriptive du cas d'utilisation : Gestion des Bénéficiaires de Transactions

Nom du Cas d'Utilisation	Virement vers des Comptes Attijariwafa et Confrères
Acteur Principal	Utilisateur final
Précondition	<ul style="list-style-type: none"> • L'utilisateur doit être authentifié dans le système. • L'utilisateur doit avoir des fonds suffisants pour effectuer le virement.
Résumé	Ce cas d'utilisation permet aux utilisateurs d'effectuer des virements bancaires vers des comptes Attijariwafa et confrères, en utilisant une interface vocale.
Enchaînement Principal	<ol style="list-style-type: none"> 1. L'utilisateur lance la commande vocale pour initier un virement. 2. Le système convertit la commande vocale en texte, puis identifie l'intention de l'utilisateur d'effectuer un virement. 3. L'utilisateur fournit les détails du virement, tels que le montant et le bénéficiaire. 4. Le système demande à l'utilisateur de confirmer les infors de virement après avoir vérifier les fonds disponibles et les détails fournis. 5. Le système effectue le virement et envoie une confirmation à l'utilisateur.

TABLE 18 – Fiche descriptive du cas d'utilisation : Virement vers des Comptes Attijariwafa et Confrères

3.3 Diagrammes de séquence pour les cas d'utilisation clés

Afin de bien visualiser le flux d'exécution des opérations et de clarifier les responsabilités des différents composants du système, nous avons élaboré un ensemble de diagrammes de séquence pour les cas d'utilisation clés de notre solution. Ces diagrammes permettent de représenter de manière séquentielle les interactions entre les différents composants du système. Ils décrivent comment les messages sont échangés entre l'acteur principal et les objets du système, illustrant ainsi le déroulement des scénarios d'utilisation.

3.3.1 Diagramme de séquence cas d'utilisation : « Accès à l'assistance et à la FAQS »

Ce diagramme de séquence illustre le processus d'accès à la FAQ via notre chatbot bancaire vocal.

1. Ouverture de l'application :

- L'utilisateur lance l'application mobile d'ebanking.
- L'application mobile s'ouvre et demande à l'utilisateur de procéder à l'authentification biométrique en lançant l'assistant vocal.

2. Authentification biométrique :

- L'utilisateur enregistre un audio en répétant la phrase demandée par l'assistant vocal pour capturer ses empreintes vocales.
- L'application mobile envoie ces empreintes à l'API bancaire pour vérification.
- L'API bancaire sollicite l'API d'authentification pour comparer les empreintes.
- Si la comparaison est réussie, l'API bancaire renvoie l'identifiant de l'utilisateur à l'application mobile, qui redirige alors l'utilisateur vers la page principale.
- Si la comparaison échoue, l'API bancaire informe l'application mobile de l'échec, et l'utilisateur est redirigé vers la page d'authentification biométrique.

3. Interaction avec l'assistant vocal :

- Si l'utilisateur a utilisé l'application dans la dernière heure, il peut poser directement ses questions d'assistance.
- L'API NLP identifie l'intention de l'utilisateur et appelle la pipeline FAQ pour générer les réponses, qui sont renvoyées à l'application mobile et fournies à l'utilisateur de manière vocale.

4. Vérification d'identité si inactivité prolongée :

- Si l'utilisateur n'a pas interagi avec l'application depuis plus d'une heure, il doit répondre à une question aléatoire générique de sécurité.
- L'application mobile pose cette question via la synthèse vocale.
- L'utilisateur répond, et cette réponse est envoyée à l'API NLP pour vérification.

- L'API NLP demande à l'API bancaire les données nécessaires pour la vérification.
- Si l'identité est vérifiée, l'utilisateur peut continuer à poser des questions, et le système fonctionne de manière similaire à l'étape précédente.
- Si l'identité ne peut pas être vérifiée, l'application mobile informe l'utilisateur de l'échec.



FIGURE 46 – Diagramme de séquence du cas d'utilisation : « Accès à l'assistance et à la FAQS »

3.3.2 Diagramme de séquence cas d'utilisation : « Transaction Virement »

Ce diagramme de séquence décrit le processus de passage d'un virement bancaire via le chatbot vocal de notre application bancaire. Étant donné que la phase d'authentification à deux facteurs a déjà été détaillée dans le diagramme d'accès à la FAQ, nous nous concentrerons ici sur les étapes spécifiques au virement bancaire

1. Ouverture de l'application et authentification :

- Tout comme le diagramme de séquence précédent, l'utilisateur ouvre l'application, enregistre son audio, et le processus d'authentification demeure le même.

2. Demande de virement bancaire :

- Si l'utilisateur a interagi avec l'application dans l'heure écoulée et qu'il est authentifié, il peut directement demander un virement bancaire en interagissant avec l'assistant vocal.
- L'API NLP identifie l'intention de l'utilisateur et vérifie la présence des informations requises pour le virement.
- Si toutes les informations nécessaires sont présentes, l'API NLP demande à l'application mobile de solliciter la confirmation de l'utilisateur.
- L'utilisateur confirme ou annule la transaction via une réponse vocale.

3. Exécution de la transaction :

- En cas de confirmation, l'API NLP transmet la demande à l'API bancaire pour exécution.
- L'API bancaire renvoie l'état de la transaction, et l'application mobile informe l'utilisateur de l'état de la transaction via une réponse vocale.
- En cas de données manquantes, l'application mobile informe l'utilisateur des données manquantes.
- En cas d'annulation , l'utilisateur peut demander une autre action ou un autre virement.

Vérification d'identité si inaktivité prolongée :

- Le processus et les étapes de vérification se poursuivent comme décrit précédemment pour le diagramme de séquence de FAQ pour valider l'identité de l'utilisateur.

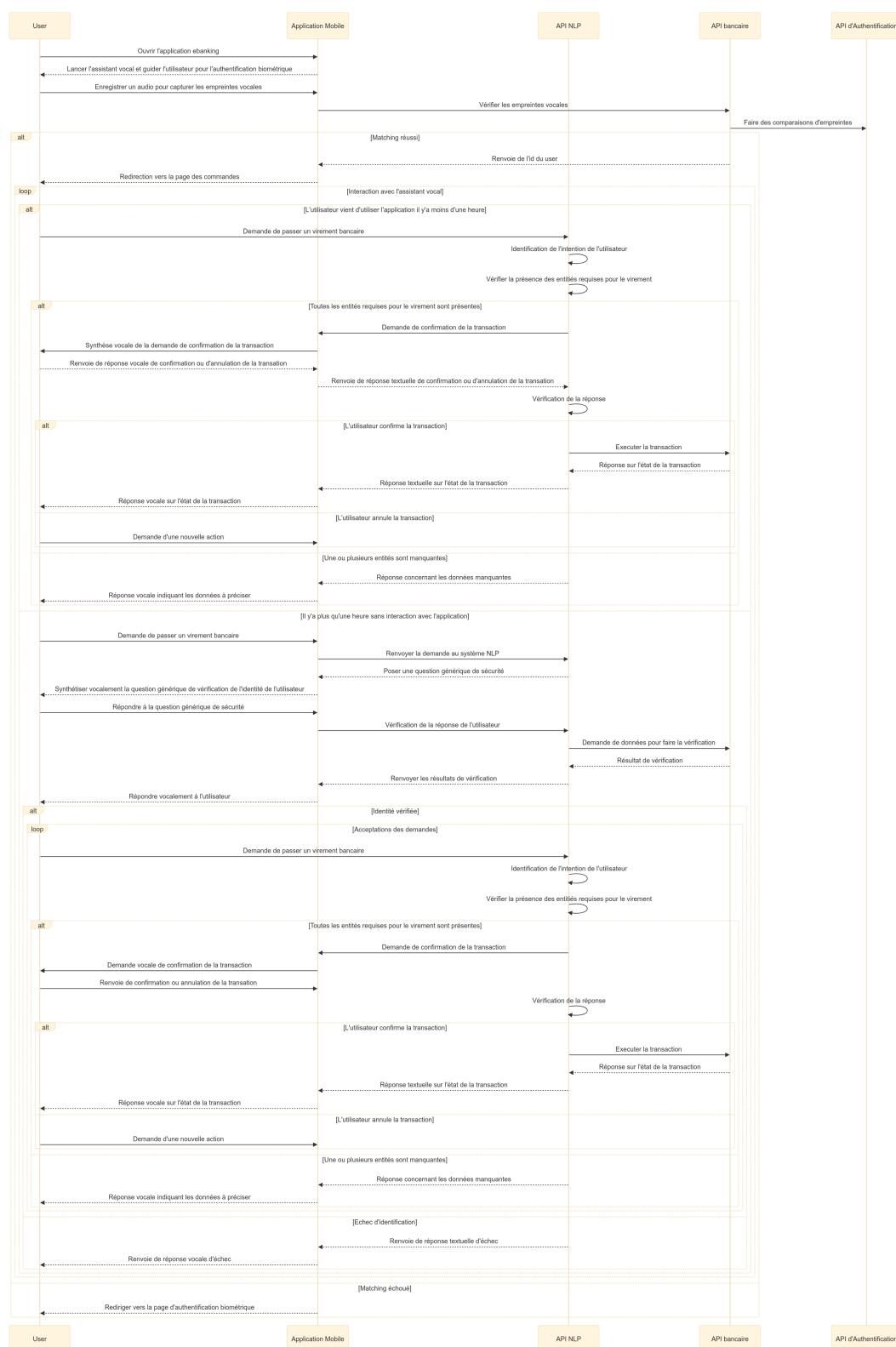


FIGURE 47 – Diagramme de séquence du cas d'utilisation : « Transaction Virement »

3.3.3 Diagramme de séquence cas d'utilisation : « Consultation des Inofs de comptes, de cartes et des opérations »

Le diagramme de séquence ci-dessous illustre le scénario d'utilisation de la consultation des informations de comptes, de cartes et des opérations.

- 1. Authentification à 2 facteurs comme les diagrammes de séquence précédents :** Tout d'abord, l'utilisateur ouvre l'application mobile d'e-banking et lance l'assistant vocal, qui guide l'utilisateur à travers le processus d'authentification à deux facteurs. Ce processus combine l'authentification biométrique et la vérification de l'identité de l'utilisateur à travers des questions de sécurité générées aléatoirement, comme décrit dans les diagrammes de séquence précédents.
- 2. Execution de la demande de consultation :** Si la vérification réussit, l'application redirige l'utilisateur vers la page des commandes, où il peut interagir avec l'assistant vocal. S'il a récemment interactué avec l'application, il peut directement demander la consultation de son solde, de ses cartes ou de ses opérations. L'API NLP identifie l'intention de l'utilisateur et demande les données nécessaires à l'API bancaire, qui les renvoie à l'API NLP. Ensuite, cette dernière envoie la réponse textuelle à l'application mobile, qui la synthétise vocalement pour l'utilisateur.

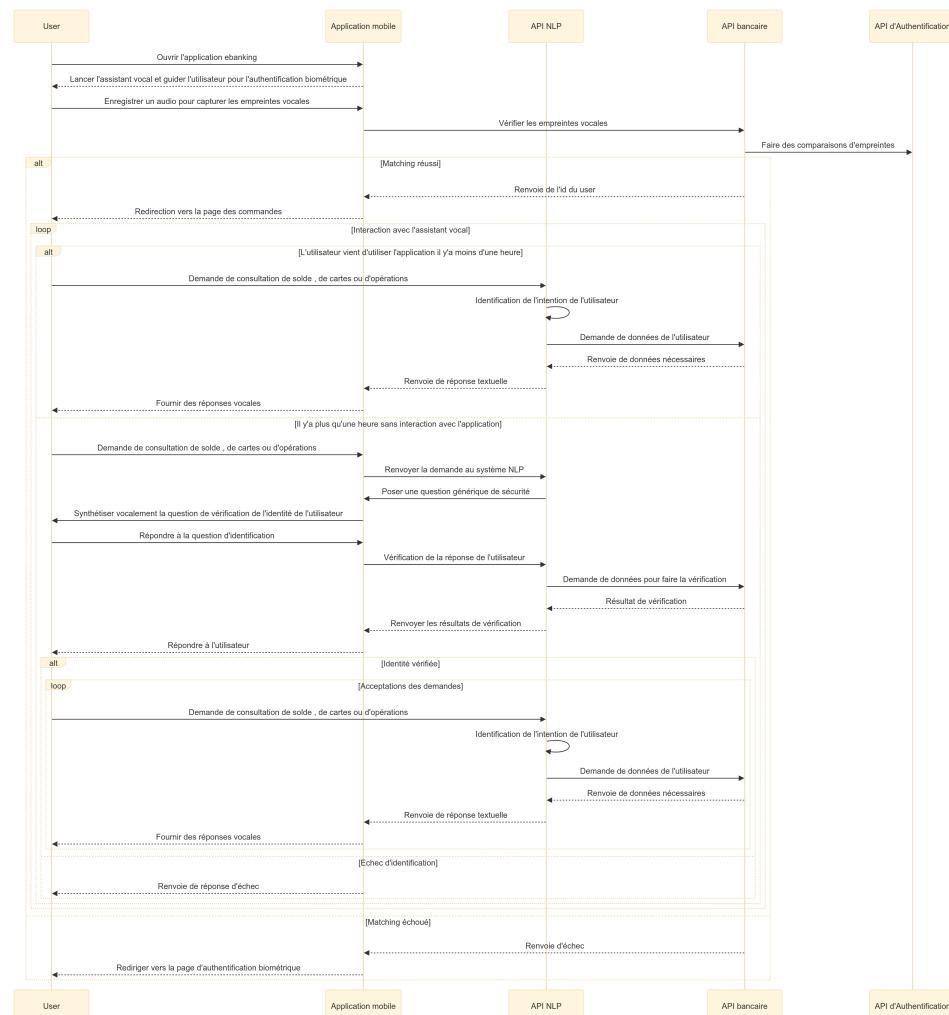


FIGURE 48 – Diagramme de séquence du cas d'utilisation : « Consultation des Infos de comptes, de cartes et des opérations »

3.4 Diagramme de classes

Les diagrammes de classes sont parmi les types de diagrammes UML les plus utiles, car ils décrivent de manière claire la structure d'un système en modélisant ses classes, attributs, opérations et les relations entre les objets. Contrairement au diagramme de cas d'utilisation, qui présente le système du point de vue des acteurs, le diagramme de classes en révèle la structure interne. Il offre une représentation abstraite des objets du système et de leurs interactions pour réaliser les cas d'utilisation.

Descriptif des classes :

Classe	Description	Attributs	Opérations
Utilisateur	Représente un utilisateur du système	username, password, email, telephone, voiceFeatures	Récupérer, Authentifier
Client	Représente un client de la banque	cin, nom, prenom, dateNaissance, adresse, email, telephone	Récupérer
Compte	Représente un compte bancaire	numeroCompte, typeCompte, solde, dateOuverture, tauxInterets, statutCompte	Récupérer
Carte	Représente une carte bancaire	typeCarte, numeroCarte, dateExpiration, codePIN, dateActivation, dateOpposition, CVV, statutCarte, services	Récupérer, Mettre à jour, Opposer
Plafond	Représente les plafonds de transaction	typePlafond, montantPlafond, periode, estPermanent, dateDebut, dateExpiration	Récupérer, Mettre à jour
Operation	Représente une opération bancaire	dateOperation, typeOperation, montant, motif, categorieOperation	Récupérer, Ajouter
Beneficiaire	Représente un bénéficiaire des transactions	nom, prenom, RIB, typeBeneficiaire	Ajouter, Récupérer, Mettre à jour, Supprimer
Agence	Représente une agence bancaire	nomAgence, adresse, telephone, horairesOuverture, servicesDisponibles	Récupérer
Facture	Représente une facture	numeroFacture, montant, dateEmission, dateEcheance, statutPaiement	Récupérer, Mettre à jour
PaiementFacture	Représente le paiement d'une facture	datePaiement	Récupérer, Ajouter

TABLE 19 – Description des classes et opérations du système bancaire

La création d'un diagramme de classes repose sur un ensemble de règles de gestion du système, que nous listons ci-dessous :

1. **Opération et Compte** : Un compte peut être lié à une ou plusieurs opérations, mais chaque opération est associée à un seul compte.
2. **Compte et Carte** : Un compte peut posséder une ou plusieurs cartes, tandis qu'une carte est liée à un seul compte.
3. **Compte et Paiement de Facture** : Un compte peut réaliser un ou plusieurs paiements de factures, mais chaque paiement de facture est associé à un seul compte.
4. **Facture et Paiement de Facture** : Une facture peut être associée à un ou plusieurs paiements de factures, mais chaque paiement de facture est rattaché à une seule facture.
5. **Carte et Plafond** : Une carte peut avoir un ou plusieurs plafonds de services, tandis qu'un plafond peut être associé à une ou plusieurs cartes.
6. **Opération et Bénéficiaire** : Un bénéficiaire peut être lié à une ou plusieurs opérations, mais chaque opération est associée à un seul bénéficiaire.
7. **Facture et Bénéficiaire** : Un bénéficiaire peut être associé à un ou plusieurs paiements de factures, mais chaque paiement de facture est rattaché à un seul bénéficiaire.
8. **Facture et Client** : Un client peut posséder une ou plusieurs factures, mais chaque facture est liée à un seul client.
9. **Client et Bénéficiaire** : Un client peut avoir un ou plusieurs bénéficiaires, tandis qu'un bénéficiaire est associé à un seul client.
10. **Compte et Client** : Un client peut être affilié à un ou plusieurs comptes, mais chaque compte est lié à un seul client.
11. **Agence et Compte** : Une agence est affectée à un seul compte, mais une agence peut gérer plusieurs comptes.
12. **Utilisateur et Client** : Un client hérite d'un utilisateur.

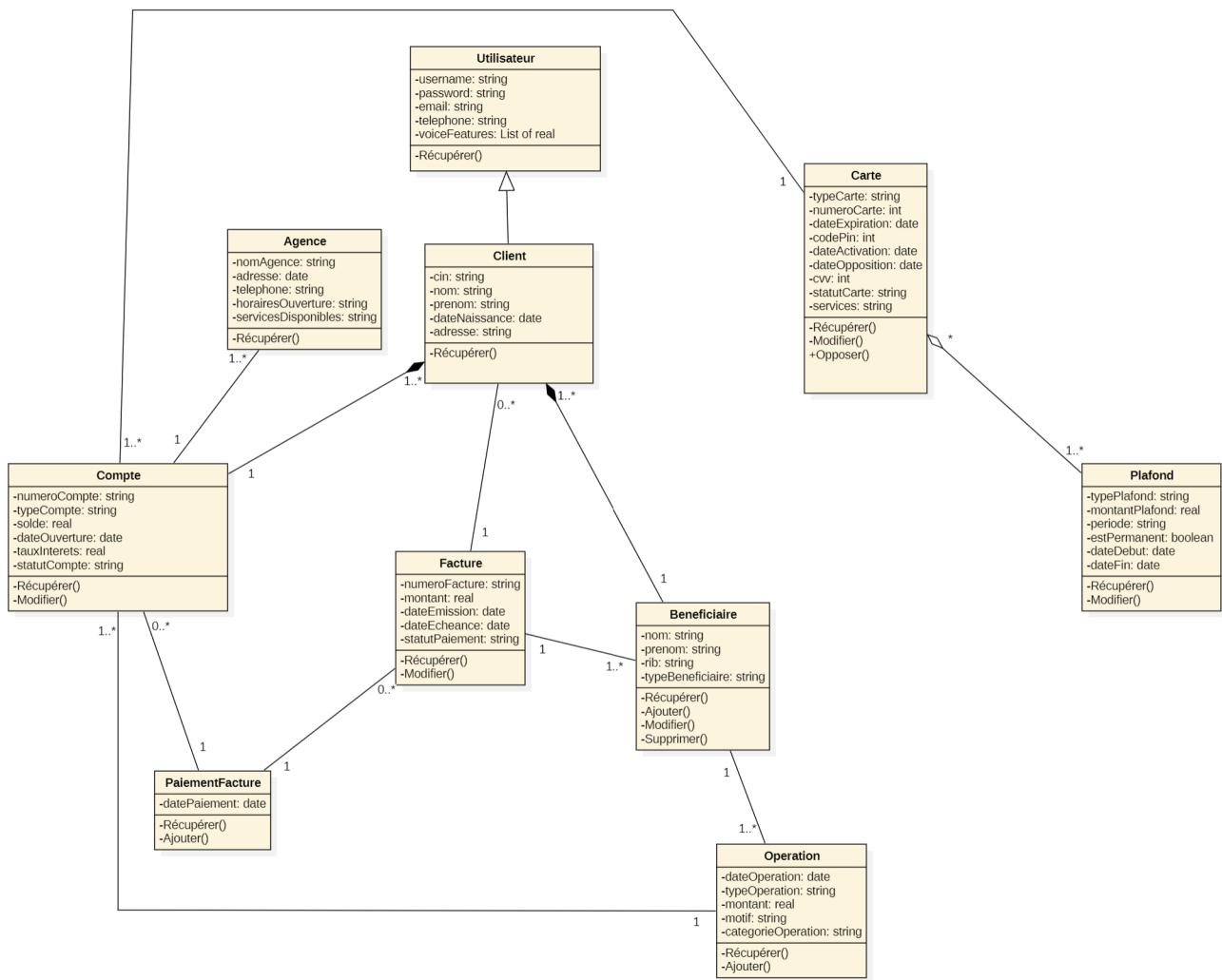


FIGURE 49 – Diagramme de Classe

3.5 Diagrammes d'activité pour la conception du système

3.5.1 Diagramme d'activité : Système

Le diagramme d'activité illustre le processus d'interaction de l'utilisateur avec le système bancaire via l'application mobile. L'utilisateur commence par ouvrir l'application et enregistrer sa voix, qui est ensuite envoyée à l'API bancaire pour extraire les caractéristiques vocales uniques et les comparer à celles enregistrées dans la base de données. Si des correspondances sont trouvées, l'application récupère l'identifiant de l'utilisateur et le redirige vers la page principale.

Une fois authentifié, l'utilisateur peut demander une action. La demande vocale est transcrise par l'application mobile et envoyée à l'API NLP, qui vérifie la date de la dernière connexion. Si celle-ci remonte à plus d'une heure, l'utilisateur doit répondre à une question de sécurité dont la réponse sera validée par l'API NLP en sollicitant l'API bancaire.

Après confirmation de la sécurité, la demande vocale est transcrise et traitée par l'API NLP, et une réponse est envoyée à l'utilisateur. En cas d'erreur ou de réponse incorrecte, l'utilisateur est invité à réessayer et reçoit un message d'échec. Si la dernière connexion

date de moins d'une heure, la demande de l'utilisateur est traitée directement et une réponse vocale lui est renvoyée.

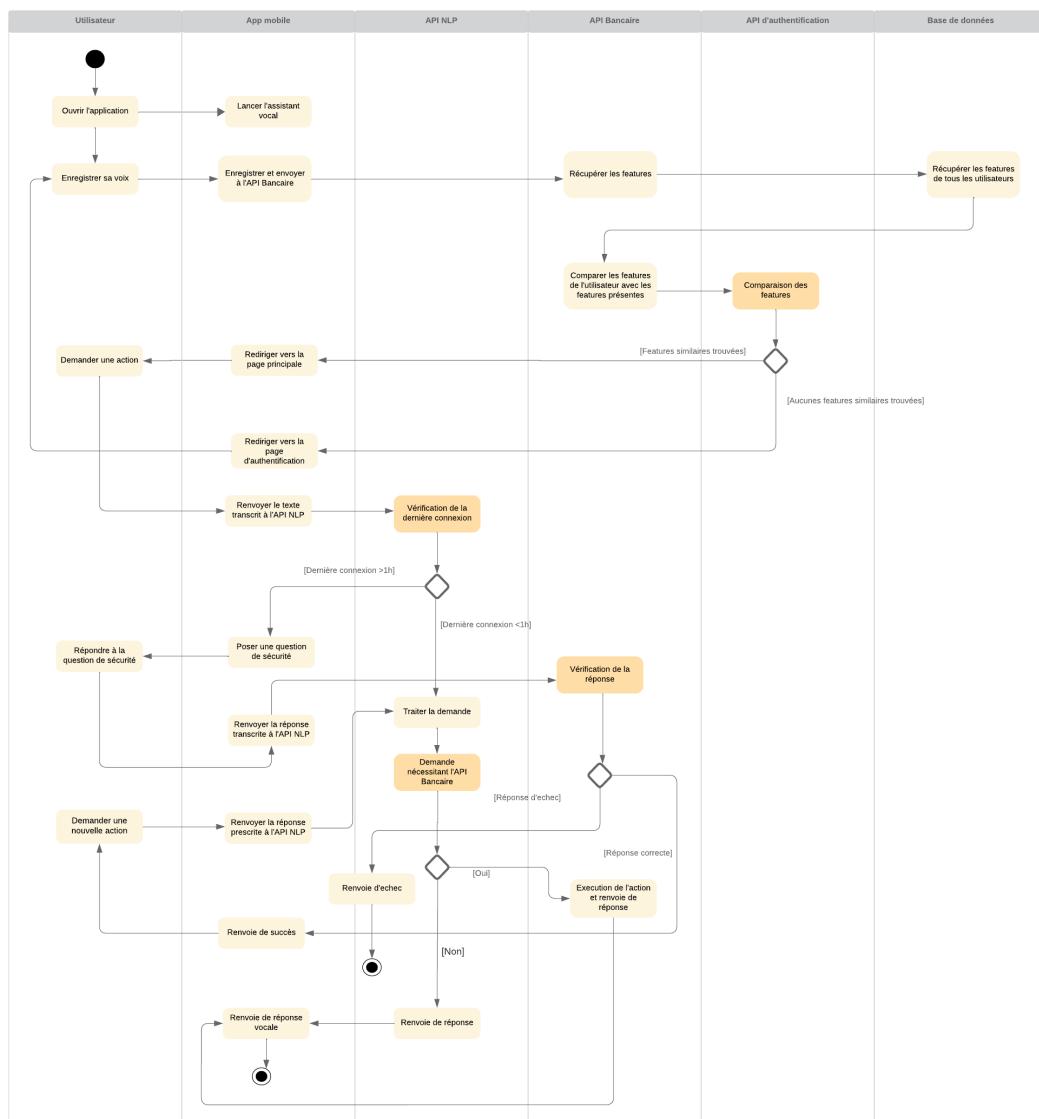


FIGURE 50 – Diagramme d'activité du système

3.5.2 Diagramme d'activité : Sécurité

Le diagramme d'activité décrit le processus d'authentification à deux facteurs dans le système. L'utilisateur commence par ouvrir l'application et enregistrer sa voix, laquelle est envoyée à l'API bancaire. Les caractéristiques vocales sont comparées aux données de la base de données. Si une correspondance est trouvée, l'utilisateur est redirigé vers la page principale ; sinon, il est dirigé vers la page d'authentification.

L'API NLP calcule ensuite la durée depuis la dernière connexion. Si celle-ci est supérieure à une heure, une question de sécurité est posée à l'utilisateur. Si la réponse est correcte, la demande de l'utilisateur est traitée par l'API NLP et l'API bancaire, avec

un renvoi de succès, et l'utilisateur peut demander une nouvelle action. En cas de réponse incorrecte, un message d'échec est renvoyé et l'utilisateur est invité à réessayer. Si la dernière connexion date de moins d'une heure, la demande est directement traitée, une réponse de succès est envoyée, et l'utilisateur peut demander une nouvelle action.

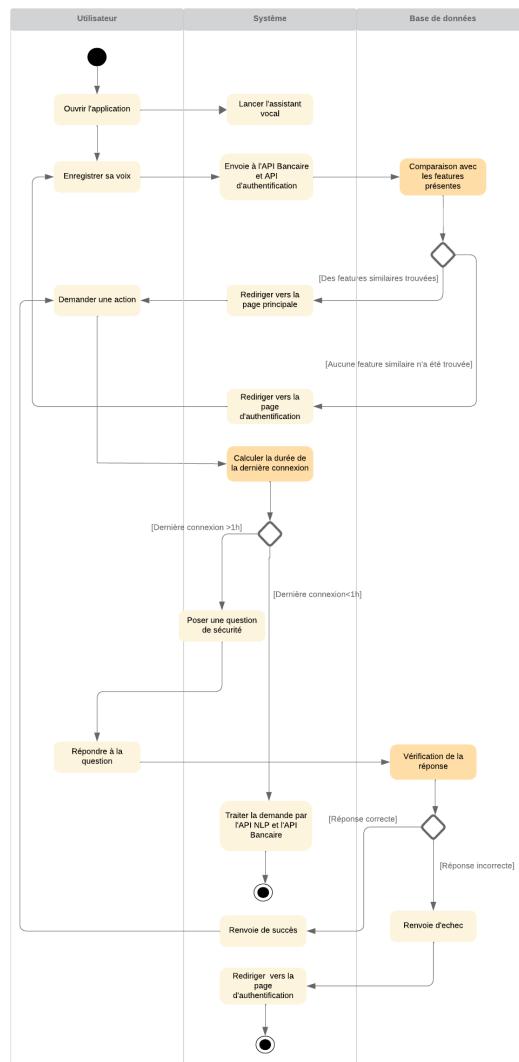


FIGURE 51 – Diagramme d'activité du volet sécurité

3.5.3 Diagramme d'activité : Système NLP

Le diagramme d'activité ci-dessous dépeint le processus de gestion des requêtes des utilisateurs par l'API NLP. Le texte de l'utilisateur est d'abord reçu par l'API. Ensuite, elle vérifie le contexte de la demande.

Si c'est une première demande, une question de sécurité est posée. Si c'est une réponse à une question de sécurité, l'API NLP vérifie la réponse en collaboration avec l'API bancaire, qui exécute les requêtes nécessaires pour récupérer les données, vérifier la réponse et renvoyer un succès ou un message d'erreur en cas d'échec.

Pour une requête ordinaire de l'utilisateur (par exemple, une réponse à une entité manquante, une demande d'action, ou une réponse à une question de confirmation d'action), l'API NLP extrait le contexte de la demande en utilisant notre modèle de classification pour s'assurer qu'il n'y a pas eu de changement de contexte. Si la prédiction est suffisamment fiable, l'API NLP extrait les entités correspondantes à l'action demandée par l'utilisateur à l'aide de notre modèle NER et des expressions régulières. Ensuite, on vérifie si l'action requiert des entités supplémentaires pour être accomplie. Si des entités ou des données supplémentaires sont nécessaires, elles sont demandées à l'utilisateur.

Si l'action ne requiert pas d'entités supplémentaires pour être exécutée, l'action correspondante à l'intention de l'utilisateur est exécutée soit en sollicitant l'API bancaire, soit en déclenchant le pipeline FAQ.

Enfin, les données sont modifiées et nettoyées périodiquement, et des réponses appropriées sont envoyées à l'utilisateur, clôturant ainsi le cycle d'interaction.

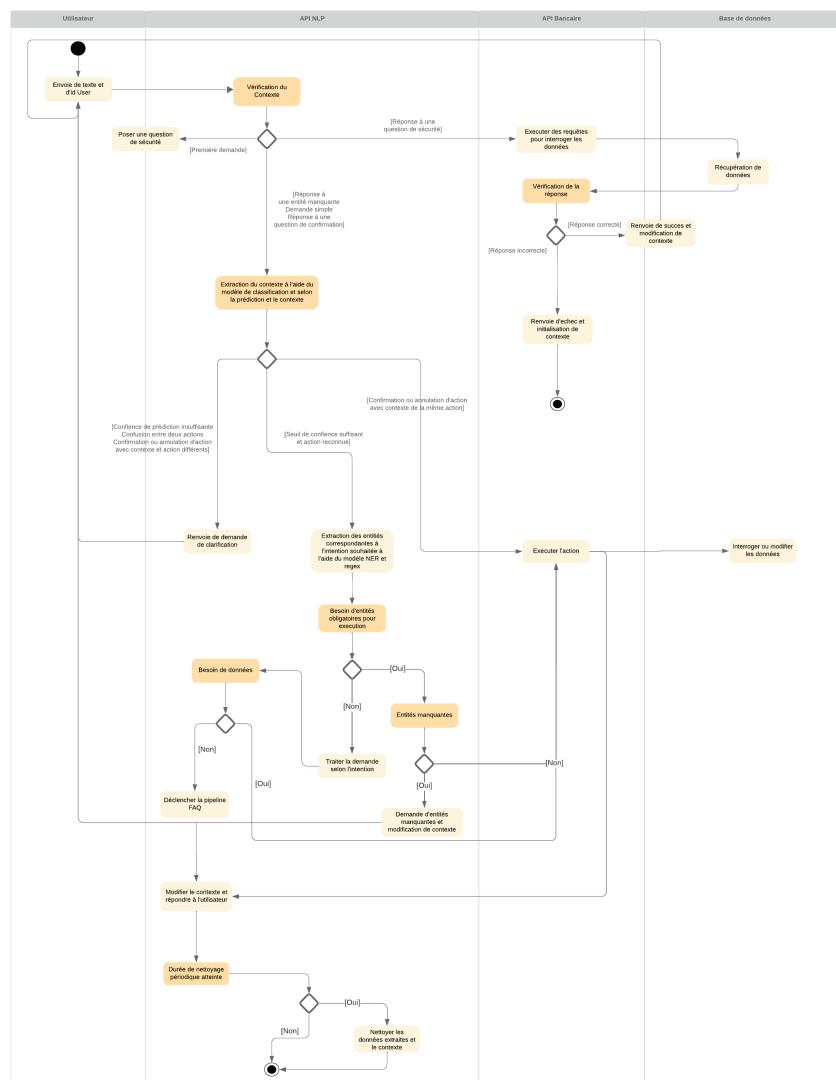


FIGURE 52 – Diagramme d'activité du volet NLP

3.6 Réalisation d'un protocole de l'application

Dans le domaine des applications, un prototype se présente comme une illustration pratique de l'application envisagée. Son objectif est de présenter aux utilisateurs potentiels et/ou aux investisseurs l'apparence attendue de l'application et d'aider à évaluer si notre produit répond aux exigences prévues. Nous avons efficacement conçu toutes les interfaces et parcouru sans heurts l'application en utilisant Figma, un outil en ligne collaboratif conçu pour créer des interfaces.

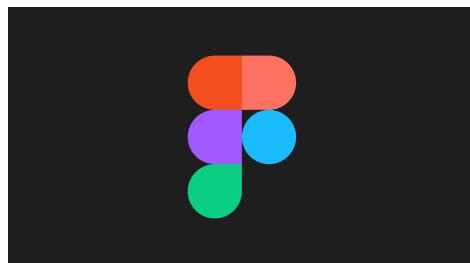


FIGURE 53 – Logo de la plateforme Figma

- **Page d'authentification :** La page d'authentification propose une méthode basée sur les empreintes vocales, permettant à l'utilisateur de cliquer sur l'écran pour enregistrer sa voix en répétant une phrase demandée par l'assistant.
- **Page des commandes :** permet à l'utilisateur d'enregistrer sa commande vocale concernant les opérations d'eBanking souhaitées en cliquant sur l'écran, puis de recevoir la réponse vocale de l'assistant.
- **Page Google Maps :** La page Google Maps offre la possibilité de repérer les agences les plus proches de la position de l'utilisateur et de démarrer la navigation vers ces agences.

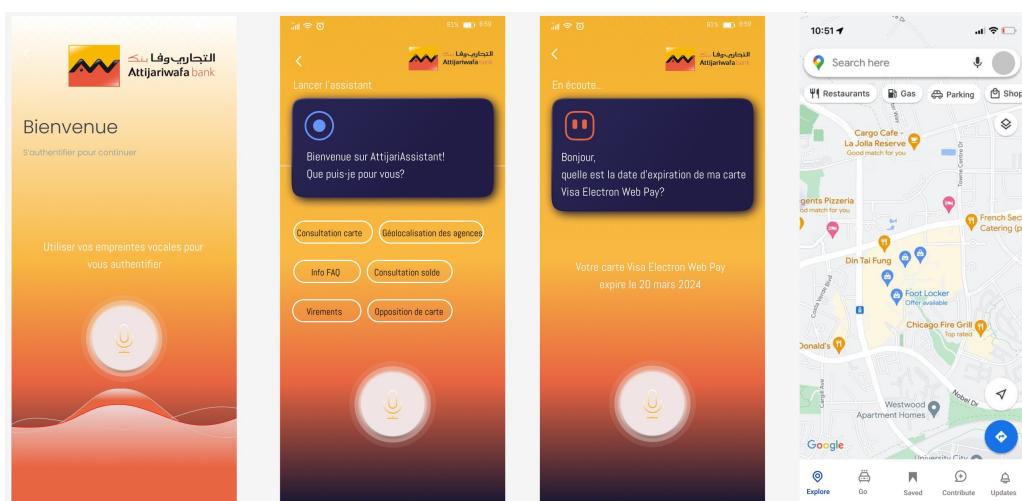


FIGURE 54 – Prototype de l'application

3.7 Conclusion

Dans ce chapitre, notre attention s'est portée sur la conception de notre système, avec la création de tous les diagrammes associés. Ces diagrammes nous ont offert une vision claire et structurée des diverses fonctionnalités et interactions de notre système. De plus, nous avons présenté les interfaces éventuelles que nous avons conçues à l'aide de Figma. Dans les sections suivantes, nous aborderons la phase de réalisation, au cours de laquelle nous concrétiserons notre conception en développant effectivement les composants de notre système.

Chapitre IV : Stratégies d'Authentification

4 Stratégies d'Authentification

4.1 Introduction

Dans ce chapitre, nous explorerons les stratégies d'authentification pour sécuriser notre application dédiée aux utilisateurs non-voyants. Nous étudierons deux approches principales : l'authentification biométrique, basée sur des caractéristiques physiologiques uniques comme les empreintes digitales, le visage et la voix, ainsi que l'authentification classique, utilisant des méthodes telles que les mots de passe ou les codes PIN en les dictant vocalement. Notre analyse de ces approches nous permettra de choisir la solution optimale répondant à nos exigences spécifiques et assurant une expérience utilisateur fluide.

4.2 Exploration des Méthodes d'Authentification

Pour mettre en place l'authentification biométrique avec les empreintes digitales ou le visage, il est crucial d'évaluer le processus d'enrôlement et de stockage des données biométriques, susceptible de susciter des préoccupations liées à la sécurité, à la protection de la vie privée, aux coûts et à la complexité. Dans le cadre de notre projet, ces solutions ne sont pas viables en raison de leur coût élevé, des risques potentiels de refus des utilisateurs et des préoccupations de sécurité associées. Ainsi, nous devons rechercher d'autres solutions pour garantir la sécurité et l'accessibilité de notre système d'authentification, notamment pour les utilisateurs non-voyants.

Dans cette perspective, nous envisageons de combiner la biométrie embarquée, qui utilise les capteurs intégrés au téléphone lui-même, avec l'authentification classique, comme l'utilisation de mots de passe. Cette approche nous permettra d'associer l'utilisateur à ses données dans notre base de données, tout en assurant une authentification sécurisée et conforme à nos contraintes techniques actuelles.

4.2.1 Comparaison des différentes approches

Nous avons compilé l'ensemble des résultats de comparaison dans le tableau suivant :

Méthode d'Authentification	Convivialité	Précision	Degré de sécurité	Disponibilité de la biométrie embarquée
Authentification classique et Empreintes digitales	Moyenne	Élevée	Élevé	Oui
Visage et Authentification classique	Faible	Moyenne	Moyen	Oui
Voix seule	Haute	Élevée	Élevé	Oui
Authentification classique	Haute	Moyenne	Faible	N/A

TABLE 20 – Comparaison des différentes méthodes d'authentification

- **En termes de convivialité pour les non-voyants, l'authentification vocale seule et l'authentification classique seule en dictant le mot de passe vocallement se distinguent.** En effet, ces méthodes s'appuient uniquement sur la voix, un sens préservé chez les utilisateurs non-voyants, pour l'authentification.
- **L'empreinte digitale et la reconnaissance faciale, bien que disponibles sur la plupart des smartphones modernes, présentent des défis d'accessibilité pour les utilisateurs non-voyants.** La manipulation du capteur d'empreintes digitales et la reconnaissance précise du visage peuvent s'avérer difficiles pour certains utilisateurs.
- **Concernant la précision, l'empreinte digitale et la voix seule ressortent comme les plus précises.** Ces méthodes offrent un taux de faux positifs relativement faible, garantissant une authentification fiable des utilisateurs.
- **Du point de vue de la sécurité, l'empreinte digitale et la voix seule offrent également un niveau élevé de sécurité.** La difficulté à falsifier une empreinte digitale ou une voix authentique renforce la protection contre les accès non autorisés.
- **Enfin, la disponibilité de la biométrie embarquée est un facteur important à considérer.** Seules les méthodes d'authentification basées sur des technologies biométriques présentes sur le smartphone, comme l'empreinte digitale et la reconnaissance faciale, peuvent bénéficier de cette fonctionnalité.

En conclusion, notre choix se porte sur l'authentification vocale, à condition que nous puissions mettre en place un modèle fiable pour authentifier les utilisateurs. Dans le cas contraire, nous opterons pour l'authentification biométrique par empreintes digitales, combinée à l'authentification classique.

4.2.2 Approche basée sur la biométrie vocale

Nous avons choisi une architecture d'authentification typique, où le processus d'authentification par la voix se déroule généralement en deux étapes principales :

1. Enrôlement :

- L'utilisateur prononce des phrases ou des mots spécifiques dans un microphone.
- Ces échantillons vocaux sont extraits à l'aide du modèle que nous allons former, puis ils sont enregistrés et analysés pour créer un "modèle vocal" unique pour l'utilisateur.
- Le modèle vocal est stocké de manière sécurisée dans notre base de données.

2. Vérification :

- L'utilisateur prononce à nouveau des phrases ou des mots spécifiques dans un microphone.
- Les caractéristiques vocales sont extraites puis comparées au modèle vocal stocké dans la base de données.

- Si la correspondance est suffisamment proche, l'utilisateur est authentifié et autorisé à accéder au système.

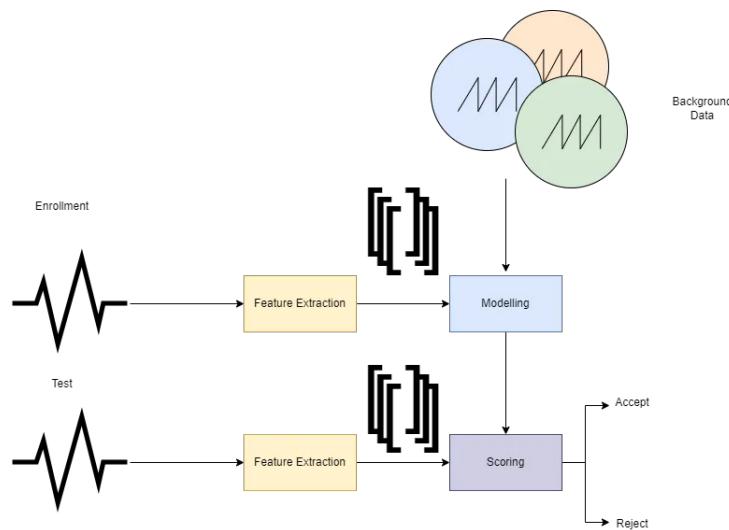


FIGURE 55 – Processus d'authentification via la biométrie vocale

Dans la suite, nous expliquerons comment nous avons entraîné le modèle.

4.3 Acquisition de Données et prétraitement pour la Biométrique Vocale

Le jeu de données utilisé pour entraîner et tester notre modèle est le jeu de données **LibriSpeech**. Il s'agit d'un corpus d'environ 1000 heures de parole anglaise à 16 kHz, fourni par Vassil Panayotov avec l'aide de Daniel Povey. Les données sont issues de la lecture de livres audio du projet LibriVox et ont été soigneusement segmentées et alignées. Le corpus est librement disponible sous la licence très permissive CC BY 4.0. Chaque échantillon est enregistré sous forme de fichier FLAC, qui signifie Free Lossless Audio Codec. Il s'agit d'un codec audio gratuit avec compression sans perte. Cela signifie que l'audio est compressé sans perdre en qualité contrairement à la compression avec perte telle que le MP3 ou le AAC. Ce processus ne supprime pas d'informations dans le flux audio.

4.3.1 Analyse Exploratoire des Données (EDA)

1. **Acquisition de données à travers le Site "<https://www.openslr.org/12/>" sous forme de dataframe :** Les données choisies contiennent 251 locuteurs. Elles sont chargées à l'aide d'un dataframe qui contient le chemin vers l'audio, le nom du locuteur, son genre et la durée de l'audio.

```

audio_path speaker_name gender \
0 dataset-speaker-recognition-librispeech/LibriS... Kate Adams F
1 dataset-speaker-recognition-librispeech/LibriS... Kate Adams F
2 dataset-speaker-recognition-librispeech/LibriS... Kate Adams F
3 dataset-speaker-recognition-librispeech/LibriS... Kate Adams F
4 dataset-speaker-recognition-librispeech/LibriS... Kate Adams F

duration
0 25.07
1 25.07
2 25.07
3 25.07
4 25.07

```

len(df['speaker_name'].unique())

251

(a) DataFrame des Données Audios :

LibriSpeech

(b) Le nombre de locuteurs

FIGURE 56 – Acquisition du dataset d’audios : LibriSpeech

2. La distribution de la longueur des audios dans le dataset : La distribution de la longueur des audios dans le dataset présente une forme asymétrique avec une majorité d’audios courts et une queue pour les audios plus longs. On observe un pic de fréquence autour de 3 secondes indiquant que la plupart des audios du dataset durent environ 3 seconds. La présence d’audios longs indique que le dataset n’est pas homogène en termes de durée et qu’il peut inclure des enregistrements plus longs, comme des discours ou des interviews. Il est donc nécessaire d’appliquer une normalisation pour garantir la cohérence des données.

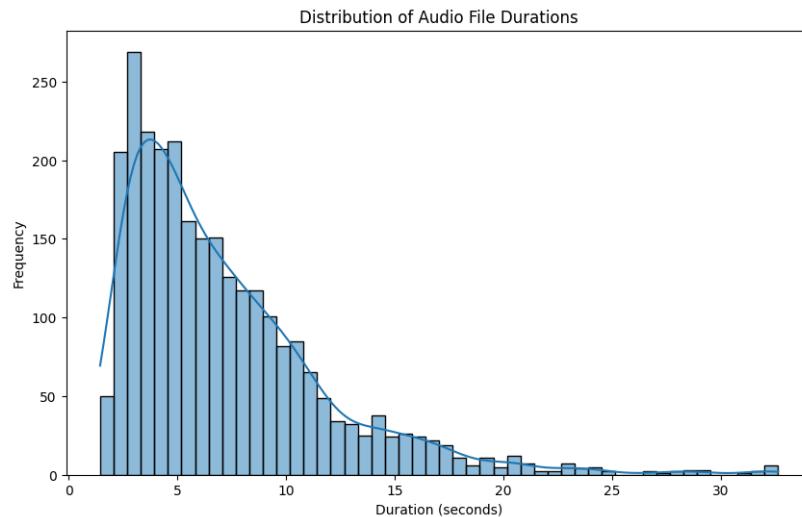


FIGURE 57 – Distribution de la longueur des audios dans le Dataset : LibriSpeech

3. Analyse statistique de la distribution des fichiers audio par locuteur dans l’ensemble de données :

- **Moyenne du nombre de fichiers audio :** En moyenne chaque locuteur possède environ 113,7 fichiers audio dans notre ensemble de données.
- **Écart-type des fichiers audio :** Cet écart type de 15,21 suggère à quel point les nombres de fichiers audio peuvent varier parmi les locuteurs. Cela signifie

que la distribution des données n'est pas extrêmement concentrée autour de la moyenne, mais qu'il y a une certaine dispersion.

- **Coefficient de variation** : permet de mesurer la dispersion relative des données par rapport à leur moyenne. Une valeur basse, comme celle-ci (0.13), indique une faible dispersion relative des données par rapport à leur moyenne.

```

Mean number of audio files: 113.7011952191235
Standard deviation of audio files: 15.21165206562841
Coefficient of variation: 0.133786210745742

```

FIGURE 58 – Mesures statistiques de la distribution des audios par locuteur :
LibriSpeech

4. **Histogramme de la distribution des fichiers Audio par locuteur** : La distribution semble être approximativement normale, en forme de cloche, avec la plupart des locuteurs ayant entre 80 et 140 fichiers audio. Le pic de la distribution se situe autour de 110 fichiers audio, ce qui indique que c'est le nombre le plus courant de fichiers audio par locuteur. Il y a moins de locuteurs avec un nombre très faible (20-70) ou très élevé (140-160) de fichiers audio. La distribution est légèrement asymétrique vers la droite, car quelques locuteurs ont un nombre plus élevé de fichiers audio allant jusqu'à environ 160. La majorité des données se regroupe autour du centre, entre 100 et 120 fichiers audio, avec un déclin graduel de chaque côté. Il y a également des valeurs aberrantes, avec quelques locuteurs possédant un nombre de fichiers audio inhabituellement bas (20-60) et inhabituellement élevé (150-160).

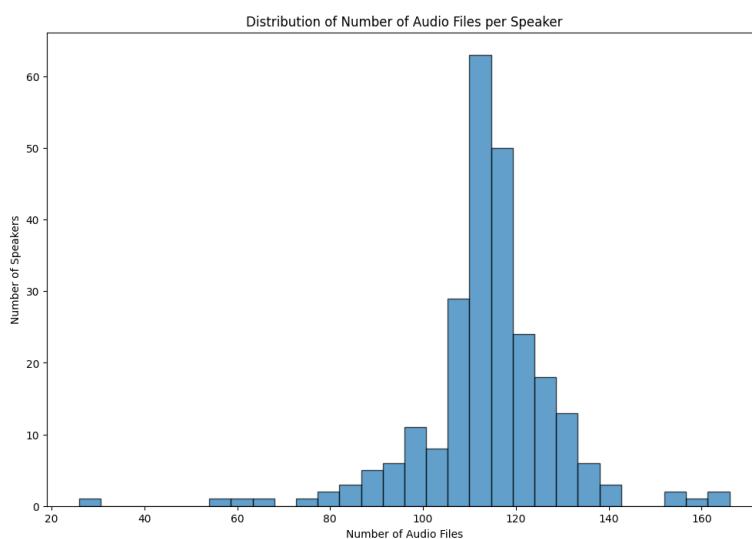


FIGURE 59 – Histogramme de la distribution du nombre d'audios par locuteur :
LibriSpeech

5. **Boîte à moustaches de la distribution des fichiers Audio par locuteur** : Le diagramme en boîte montre la distribution du nombre de fichiers audio par locuteur. La boîte principale indique que la majorité des locuteurs possède entre environ

100 et 125 fichiers audio, avec une médiane située autour de 115 fichiers audio. Les moustaches s'étendent de 80 à 140 fichiers audio, suggérant une dispersion modérée autour de la médiane. Les points situés en dehors des moustaches représentent des valeurs aberrantes, avec des locuteurs ayant un nombre de fichiers audio inhabituellement bas (20-80) ou élevé (140-160). Ces valeurs aberrantes sont visibles sous forme de cercles isolés de part et d'autre du diagramme. Ce diagramme confirme que la majorité des locuteurs se situe dans une plage restreinte de nombre de fichiers audio, tout en mettant en évidence la présence de quelques exceptions notables.

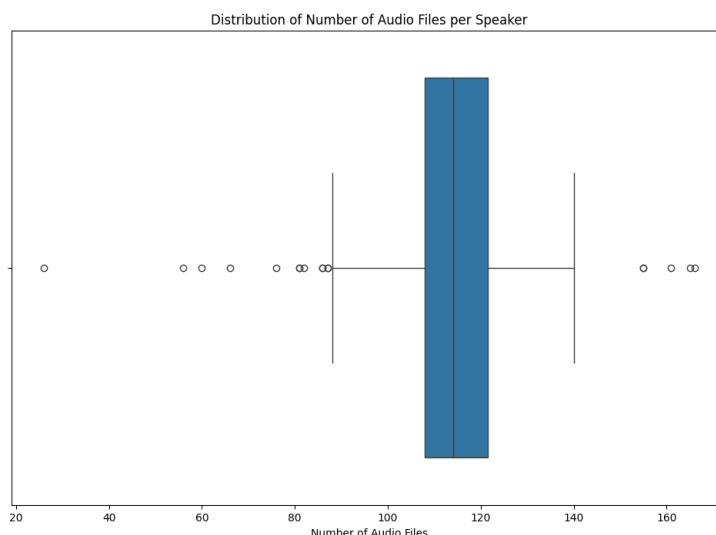


FIGURE 60 – Boîte à moustaches de la distribution des fichiers audio par locuteur : LibriSpeech

6. **Distribution des audios selon le genre :** Il y a légèrement plus de fichiers audio associés aux locuteurs féminins (F) qu'aux locuteurs masculins (M). Le nombre de fichiers audio pour chaque genre est très proche, avec environ 14 500 fichiers pour les femmes et un peu moins pour les hommes, ce qui indique une répartition relativement équilibrée entre les deux genres.

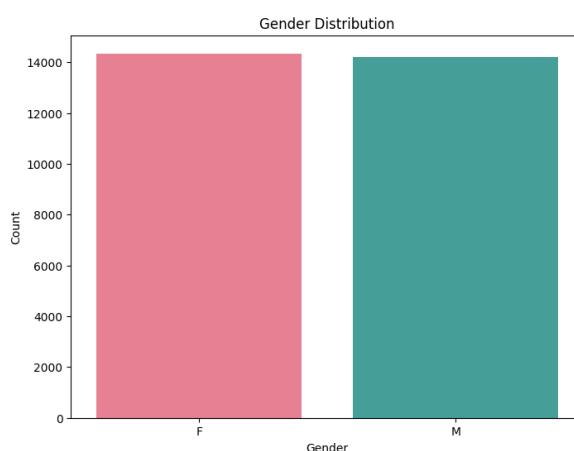


FIGURE 61 – Distribution des fichiers audio par genre : LibriSpeech

7. Distribution de la longueur des audios par locuteur : La plupart des locuteurs ont une durée totale de fichiers audio proche de 25 heures. Il y a une diminution progressive de la durée totale des fichiers audio au fur et à mesure que l'on passe de gauche à droite² sur le graphique. Certains locuteurs, vers la droite du diagramme, ont des durées totales beaucoup plus faibles, allant jusqu'à environ 5 heures. Cela indique que la majorité des locuteurs a une durée totale de fichiers audio élevée, autour de 25 heures, avec une diminution progressive pour certains locuteurs, indiquant une variabilité dans la quantité de données audio entre les locuteurs.

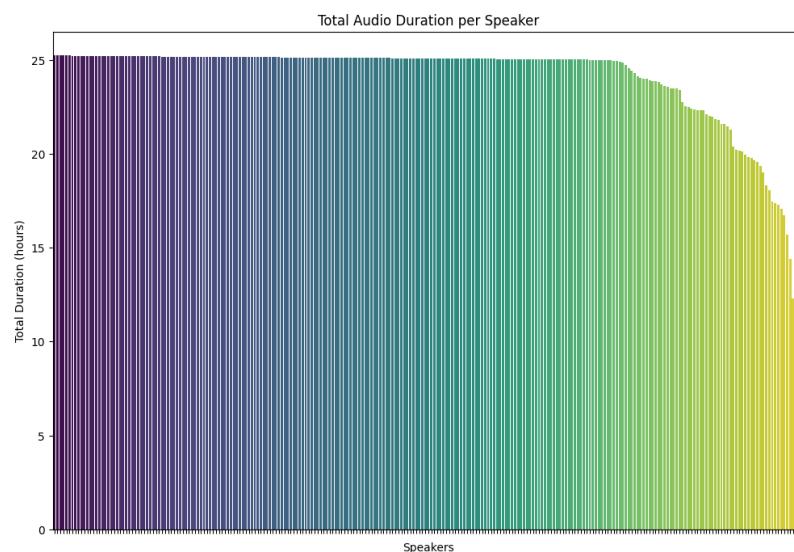


FIGURE 62 – Distribution de la longueur des audios par locuteur : LibriSpeech

8. Visualisation des données : Les visualisations sont cruciales pour la préparation des données et la sélection des caractéristiques. En traitant efficacement des éléments tels que les MFCCs et les spectrogrammes, nous pouvons construire un modèle robuste pour évaluer la similarité entre les fichiers audio. Les graphiques présentés ci-dessous mettront en lumière différents aspects des audios.

(a) **Waveform (Formes d'ondes)** : Les formes d'onde révèlent une variabilité dans la durée et l'intensité des segments sonores, indiquant que les audios du dataset possèdent une richesse temporelle et dynamique précieuse pour la formation du modèle. Les silences et pauses fréquents peuvent être exploités pour segmenter l'audio en unités plus petites, facilitant ainsi l'extraction de caractéristiques et la comparaison. Les variations d'amplitude montrent que le dataset inclut des audios avec des dynamiques variées, ce qui peut nous aider à entraîner le modèle à reconnaître les similarités et les différences basées sur ces variations.

2. Vu le nombre élevé des locuteurs (251) nous avons distingué les noms par les couleurs dans le diagramme.

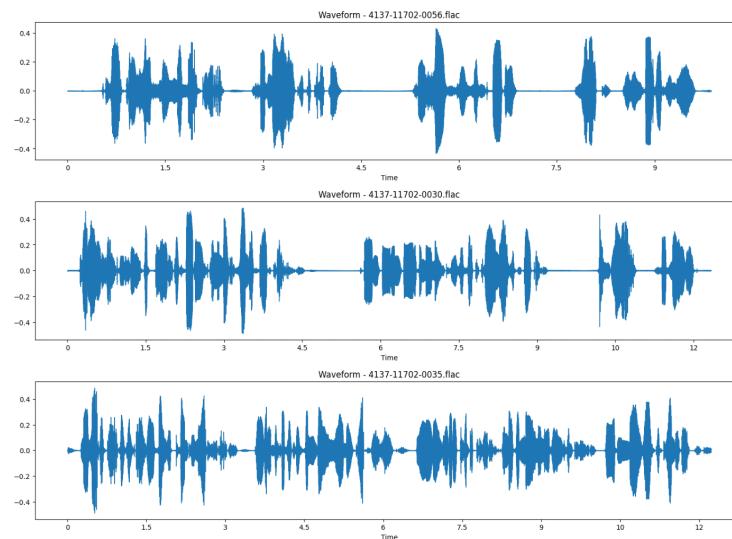


FIGURE 63 – Formes d’ondes de certains échantillons d’audios : LibriSpeech

(b) **Spectrogram (spectrogramme)** : Les spectrogrammes montrent les variations en fréquence et en intensité des trois échantillons audio au fil du temps, révélant des motifs récurrents et des variations dynamiques. Les bandes de fréquences et les intensités similaires à travers les trois audios indiquent des caractéristiques acoustiques communes, suggérant une certaine similarité entre eux. Les zones sombres représentent des silences ou des pauses, utiles pour segmenter les audios en unités analytiques plus petites. Ces visualisations permettent d’identifier les caractéristiques spectrales et temporelles essentielles pour former notre modèle.

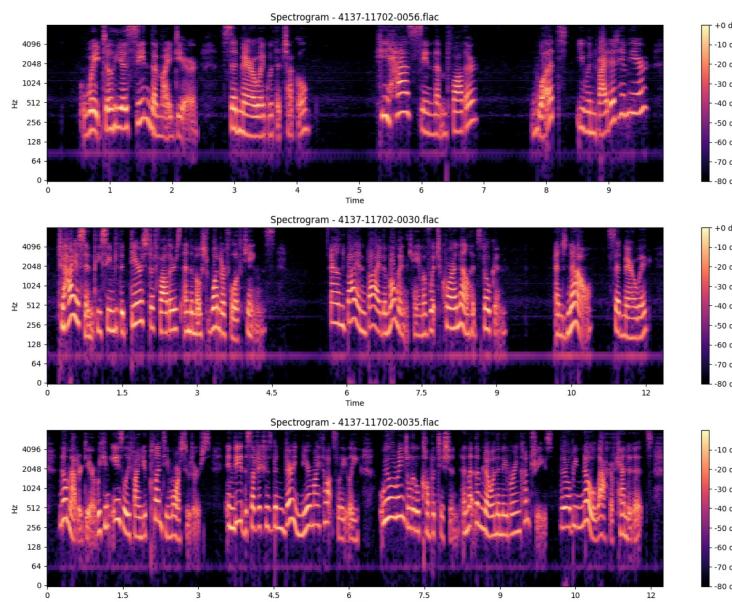


FIGURE 64 – Spectrogrammes de certains échantillons d’audios : LibriSpeech

(c) **Coefficients cepstraux de fréquence de Mel (MFCC)** Les visualisations des MFCC révèlent des motifs récurrents et des variations spécifiques,

cruciaux pour la préparation des données. Elles guident l'extraction, la normalisation des caractéristiques et la formation des paires d'échantillons pour maximiser la performance du modèle. L'analyse des échantillons montre que l'échantillon 1 présente des bandes de haute énergie dispersées (rouge), indiquant des segments vocaux riches, tandis que les bandes bleues correspondent à des transitions ou des silences. L'échantillon 2 affiche des motifs récurrents similaires avec des variations d'intensité et des segments de haute énergie plus concentrés, suggérant des éléments sonores prolongés ou intenses. L'échantillon 3 présente des motifs MFCC similaires avec des variations distinctes de basse énergie et des transitions plus fréquentes, révélant des changements rapides dans le contenu audio.

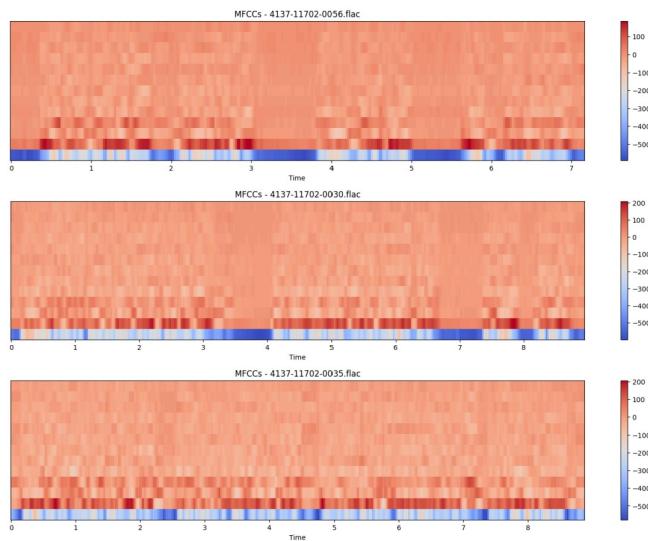


FIGURE 65 – Les coefficients MFCC de certains échantillons d'audios

4.3.2 Prétraitement des Données

1. Nettoyage de données (Data Cleaning)

- (a) **Gestion des valeurs manquantes** : nous constatons qu'il n'y a aucune donnée manquante.

```
df.isnull().sum()
```

audio_path	0
speaker_name	0
gender	0
duration	0
dtype:	int64

FIGURE 66 – Gestion des valeurs manquantes : LibriSpeech

- (b) **Suppression des doublons** : les doublons ne sont pas présents dans le dataset comme le montre la figure suivante :

```
# Identifier les doublons
duplicates = df.duplicated()

# Afficher les lignes en double (index)
print(df[duplicates])

Empty DataFrame
Columns: [audio_path, speaker_name, gender, duration]
Index: []
```

FIGURE 67 – Gestion des doublons : LibriSpeech

(c) Gestion des valeurs aberrantes :

- Après une analyse approfondie de nos données, nous avons repéré des valeurs aberrantes en examinant les mesures, l'histogramme de distribution des audios par locuteur ainsi que la boîte à moustaches. Ces observations ont mis en évidence un déséquilibre dans le nombre d'audios associés à chaque locuteur, certains ayant un nombre excessivement élevé ou faible d'enregistrements par rapport aux autres.
- Pour remédier à cette disparité, nous avons utilisé des techniques d'oversampling et d'undersampling afin de rééquilibrer le nombre d'audios par locuteur. Nous avons défini un seuil minimal de 90 audios et un seuil maximal de 130 audios par locuteur.
- La décision sur les seuils découle de notre observation antérieure de valeurs aberrantes, allant parfois aussi bas que 30 et aussi haut que 160. Notre objectif était de préserver la diversité des enregistrements tout en évitant une duplication excessive des données.

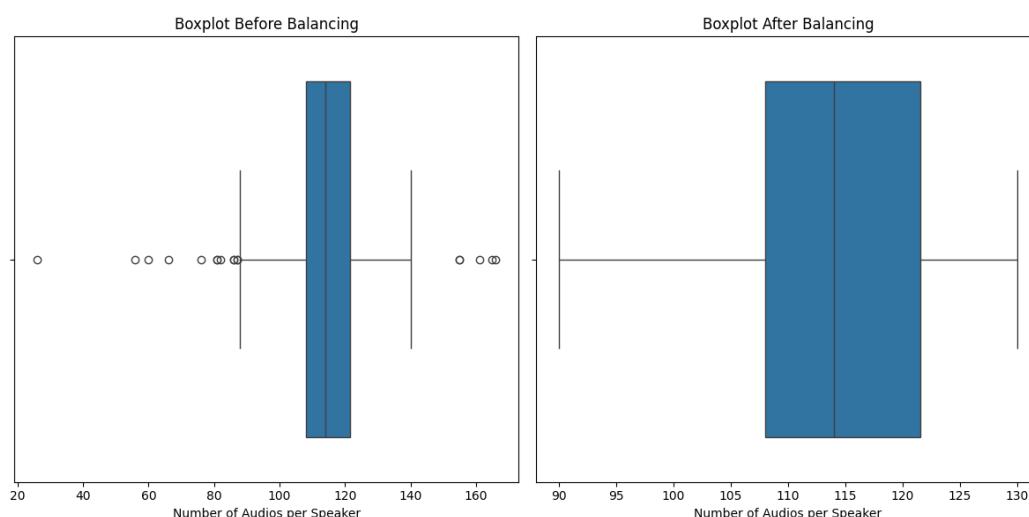


FIGURE 68 – Équilibrage des audios par locuteur :LibriSpeech

2. Prétraitement des audios (Audio Preprocessing)

- (a) Chargement et Découpage des Audios :** Nous utilisons la bibliothèque `soundfile` qui nous permet de lire les fichiers audios au format .flac. Ensuite,

pour garantir la cohérence des données, nous procédons à **l'échantillonnage** de tous les audios à un taux fixe de 16 kHz. Enfin, nous appliquons la technique de **segmentation** en découplant chaque audio en segments de durée fixe de 1,4 secondes. Cette approche assure que chaque entrée dans le modèle a une taille constante, facilitant ainsi l'entraînement et améliorant la performance du modèle. De plus, cette segmentation nous permet de capturer les différentes pauses, variations d'intensité et autres nuances dans la voix contribuant ainsi à une meilleure représentation des variations vocales d'un locuteur.

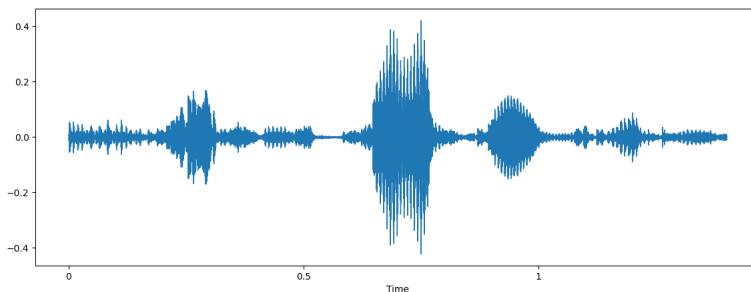


FIGURE 69 – Visualisation de la forme d'onde d'un audio prétraité : LibriSpeech

- (b) **Extraction des Caractéristiques (MFCC)** : Nous extrayons les caractéristiques des audios en utilisant les coefficients MFCC (Mel-Frequency Cepstral Coefficients). Pour chaque clip audio, nous calculons 13 coefficients MFCC en utilisant des paramètres comme `n_fft = 2048` et `hop_length = 512`. Ces caractéristiques sont ensuite normalisées pour standardiser les entrées du modèle. Les **MFCC** sont choisis pour leur capacité à capturer les caractéristiques spectrales des signaux audio, ce qui est essentiel pour la tâche de reconnaissance de locuteur. La **normalisation** des caractéristiques MFCC contribue à améliorer la convergence et la performance du modèle.

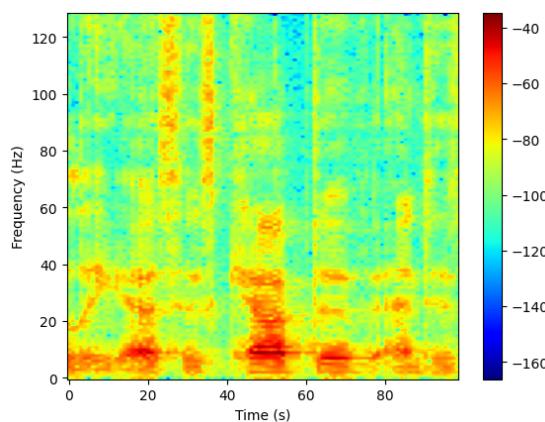


FIGURE 70 – Visualisation du spectrogramme de l'audio prétraité : LibriSpeech

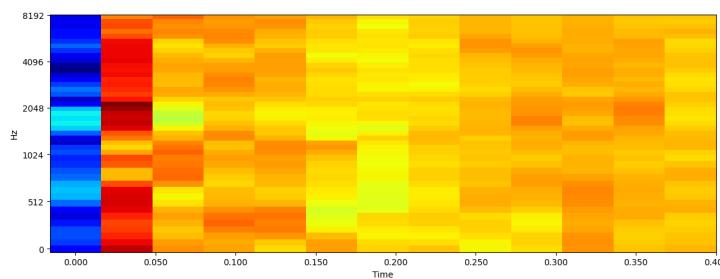


FIGURE 71 – Visualisation du MFCC de l’audio prétraité : LibriSpeech

- (c) **Conversion en Tenseurs PyTorch** : Enfin, pour faciliter l’entraînement et l’évaluation avec PyTorch, nous convertissons les caractéristiques MFCC normalisées en tenseurs PyTorch. Cette conversion ajoute les dimensions de lot et de canal nécessaires pour l’entrée du modèle. Convertir les données en tenseurs permet une intégration transparente avec notre modèle PyTorch, ce qui simplifie le processus d’entraînement et d’évaluation.
- 3. Préparation des paires d’audios (Pairs Preparation)** Tout d’abord les étiquettes sont convertit en un format séquentiel. Ensuite, les paires positives (paires d’audios appartenant à la même classe) et négatives (paires d’audios appartenant à des classes différentes) sont générées. Pour les paires positives, tous les indices de la même classe sont combinés, tandis que pour les paires négatives, des indices de différentes classes sont choisis aléatoirement en assurant un nombre équilibré de paires positives et négatives. Ces paires sont ensuite mélangées pour garantir un apprentissage non biaisé. Le processus finalise en convertissant ces paires en tenseurs PyTorch, prêts pour l’entraînement du modèle, tout en vérifiant et affichant le nombre de paires positives et négatives créées.

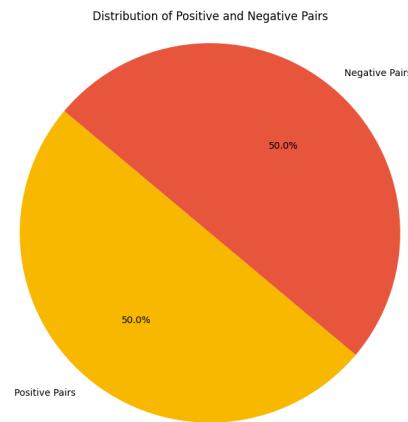


FIGURE 72 – Graphique de distribution des paires positives et négatives : LibriSpeech

- 4. Division des données (Dataset Splitting)** L’ensemble de paires a été divisé aléatoirement en trois sous-ensembles distincts à l’aide de la fonction `train_test_split` de la bibliothèque scikit-learn :

- **Entraînement (Train)** : Représentant 80% de l'ensemble de données, ce sous-ensemble est utilisé pour l'apprentissage des paramètres du modèle.
- **Validation (Validation)** : Utilisé pour évaluer les performances du modèle pendant l'apprentissage et ajuster les paramètres, ce sous-ensemble constitue 10% de l'ensemble de données.
- **Test (Test)** : Composant 10% de l'ensemble de données, ce sous-ensemble, non utilisé pendant l'apprentissage, sert à l'évaluation finale des performances du modèle sur des données inédites.

Ensuite, les données ont été chargées par lots de taille 64 pour exploiter efficacement les ressources matérielles et accélérer le processus d'apprentissage en traitant simultanément les données.

4.4 Modèle de Biométrie Vocale

4.4.1 Architecture du modèle

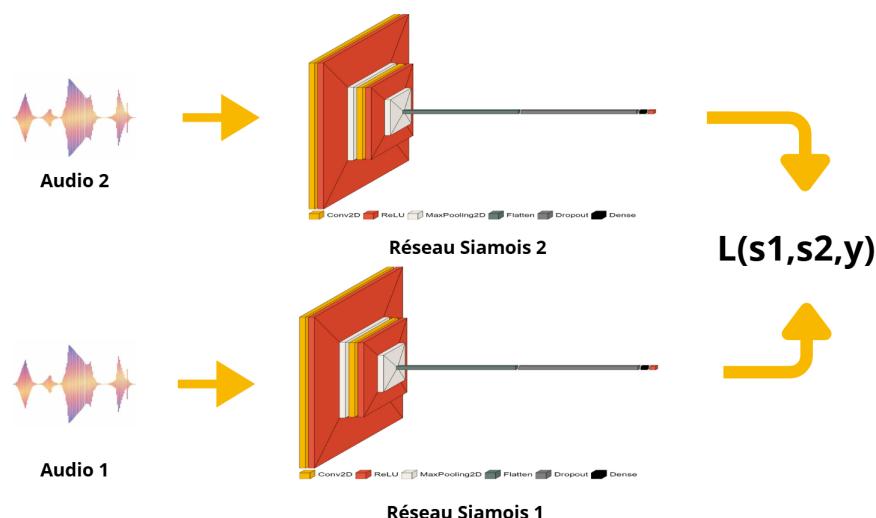


FIGURE 73 – Architecture du modèle Siamois pour la biométrie vocale

Nous avons opté pour un modèle siamois , en raison de son architecture particulièrement adaptée à la mesure de la similarité entre des données, notamment des audios. Cette architecture se distingue par sa capacité à capturer des représentations profondes et discriminantes de données d'entrée, tout en s'affranchissant de la nécessité d'une comparaison explicite, elle est moins sensible aux variations de bruit , de tempo ou de timbre, tout en offrant une certaine interprétabilité grâce aux vecteurs d'embeddings produits par les réseaux identiques.

Le modèle opère en apprenant deux réseaux neuronaux identiques , chacun traitant indépendamment un des éléments de la paire d'audios à comparer à l'aide de la similarité cosinus. L'architecture du modèle Siamois se compose de :

1. **Deux Réseau Siamois identiques qui partagent les mêmes poids et se composent de :**

- Deux couches convolutives (Conv2D) avec des fonctions d'activation ReLU.
- La première couche a 32 filtres de taille 3x3 et la seconde a 64 filtres de taille 3x3.
- Des couches de pooling (MaxPool2D) avec une taille de noyau de 2x2 sont utilisées après chaque couche convective pour réduire la dimensionnalité des sorties et capturer des caractéristiques à différentes échelles temporelles.
- Une couche Flatten aplatis la sortie des couches convolutives en un vecteur unidimensionnel.
- Une couche Dropout (0.5) est utilisée pour éviter le sur-apprentissage.
- Une couche linéaire (Linear) projette le vecteur dans un espace d'embedding de dimension 16. Cette couche apprend à représenter l'entrée audio par un vecteur compact qui capture son contenu informationnel important.
- Une fonction d'activation ReLU est appliquée pour introduire de la non-linéarité dans la sortie.

2. Couche d'intégration des deux réseaux :

- Le modèle siamois final combine les réseaux siamois définis précédemment avec une couche de sortie.
- La couche de sortie (fc) est une couche linéaire qui prend la sortie de la similarité cosinus (entre les embeddings des deux entrées) et la projette vers une valeur unique.
- La fonction d'activation sigmoïde (sigmoid) est appliquée à la sortie finale pour normaliser la valeur entre 0 et 1. Cela permet d'interpréter la sortie comme une probabilité de similarité entre les deux entrées audio.

4.4.2 Entraînement du modèle

Le processus d'entraînement de notre modèle siamois s'est déroulé sur 30 époques. À chaque époque, nous avons itéré sur les données d'entraînement et de validation en utilisant des lots (batches). Pour chaque lot, nous avons effectué une passe avant pour obtenir les prédictions du modèle, calculé la perte par rapport aux étiquettes cibles, et mis à jour les poids du modèle via rétropropagation. Nous avons choisi d'utiliser la fonction de perte binaire **BCELoss**, bien adaptée à notre tâche de classification binaire, plutôt que **MSE-Loss**. L'optimiseur Adam a été sélectionné pour son efficacité et sa capacité à s'adapter aux gradients, avec un taux d'apprentissage initial de **0.0001**. Après chaque époque, les performances du modèle sur les ensembles d'entraînement et de validation ont été évaluées et sauvegardées, permettant un suivi de l'amélioration du modèle. Pour éviter l'overfitting, une technique d'arrêt précoce a été implémentée, interrompant l'entraînement lorsque les performances sur l'ensemble de validation cessaient de s'améliorer.

Les métriques de perte et de précision enregistrées pendant l'entraînement et la validation sont évaluées en traçant les courbes d'apprentissage montrant l'évolution de la

perte et de la précision pour les ensembles d'entraînement et de validation sur 30 époques, ce qui permet de surveiller l'apprentissage du modèle.

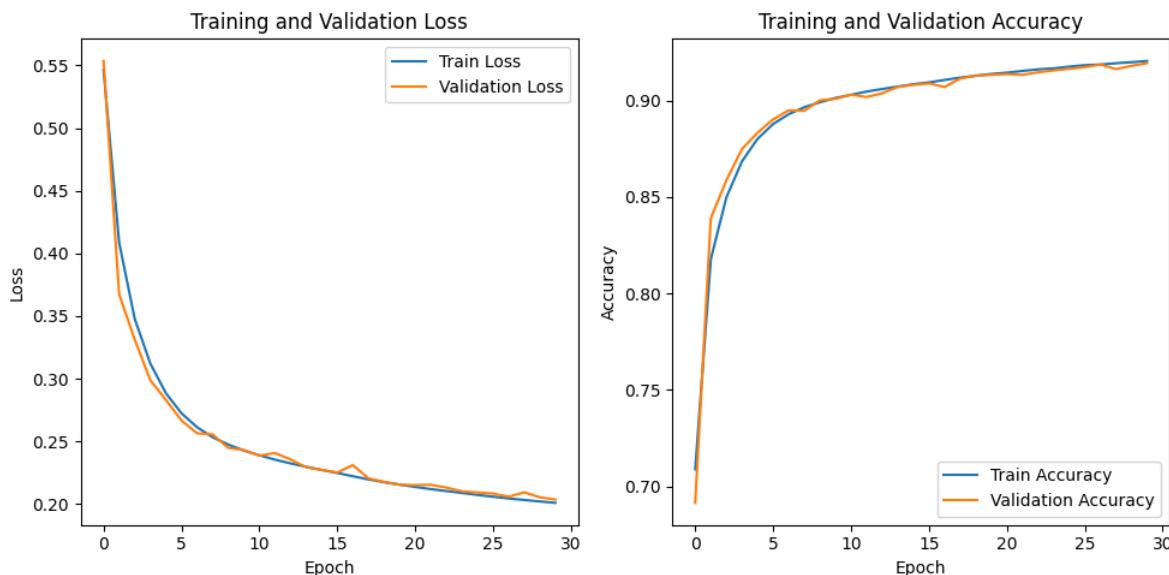


FIGURE 74 – Les courbes d'apprentissage du modèle siamois pour la biométrie vocale

- La courbe de perte montre une diminution régulière de la perte tant pour l'entraînement que pour la validation, convergeant vers une valeur basse similaire. Cela indique que le modèle s'améliore et que l'ajustement des paramètres est efficace, avec une faible probabilité d'overfitting étant donné la cohérence entre les courbes d'entraînement et de validation.
- Tandis que la courbe de précision montre une augmentation rapide de la précision au début, qui se stabilise autour de 90% pour les deux ensembles d'entraînement et de validation. Cette convergence et la proximité des courbes suggèrent que le modèle généralise bien aux données non vues et qu'il est bien entraîné.

En conclusion, les courbes indiquent que le modèle apprend efficacement et atteint une bonne performance sans surapprentissage, car les métriques d'entraînement et de validation sont étroitement alignées et se stabilisent à des valeurs favorables.

4.4.3 Résultats et Évaluation

Pour évaluer notre modèle, nous avons employé un ensemble complet de mesures de performance. Cela inclut le rapport de classification, le F1-score, la précision, le rappel, l'accuracy, la matrice de confusion et la courbe ROC-AUC. Ces outils nous permettent d'avoir une vision globale et approfondie des capacités et des limites de notre modèle. Nous présenterons ci-dessous une présentation détaillée de ses performances spécifiques :

- Rapport de classification et mesures de performances globales :**
 - Accuracy (Exactitude) :** Le modèle a correctement classé près de 91.53% des exemples de test.

- **Precision (Précision)** : 92,38 % des exemples similaires ont été correctement identifiés par le modèle parmi tous les exemples qu'il a classés comme similaires.
- **Recall (Rappel)** : 90.79% d'exemples similaires que le modèle a correctement identifiés parmi tous les exemples qui sont réellement similaires.
- **F1 Score** : Le F1-score, qui est la moyenne harmonique de la précision et du rappel, est de 91.58%, indiquant un bon équilibre entre les deux.

Le rapport de classification nous donne des informations détaillées sur la performance du modèle pour chaque classe individuelle. Il montre une bonne performance pour les deux classes, avec une précision et un rappel d'environ 91% pour la classe des paires différentes et de 92% pour la classe des paires similaires.

Accuracy:	0.9152987524622456
Precision:	0.9237739029658115
Recall:	0.9079424251838045
F1 Score:	0.9157897485442723
Classification Report:	
	precision recall f1-score support
0.0	0.91 0.92 0.91 18761
1.0	0.92 0.91 0.92 19314
accuracy	0.92
macro avg	0.92 0.92 0.92 38075
weighted avg	0.92 0.92 0.92 38075

FIGURE 75 – Rapport de classification et métriques de performances du modèle siamois pour la biométrie vocale

En résumé, les mesures indiquent que notre modèle est équilibré et efficace pour classifier correctement les instances positives et négatives, démontrant une performance globale solide.

2. Matrice de confusion :

Si les métriques globales comme l'accuracy, la précision, le rappel et le F1-score fournissent une vision d'ensemble utile des performances du modèle, la matrice de confusion offre une analyse bien plus détaillée. En effet, l'examen de cette matrice dévoile des informations cruciales sur le comportement de notre modèle et sa capacité à identifier correctement les paires d'audio similaires et non similaires.

Le modèle a montré une performance globale solide avec 92,38% d'exactitude, identifiant correctement 57 379 paires d'audio similaires (rappel de 87,54%) et 54 682 paires d'audio non similaires (précision de 93,26%). Cependant, il a commis 8 245 faux positifs et 5 194 faux négatifs, indiquant un potentiel biais à sous-estimer la similarité entre les paires d'audio.

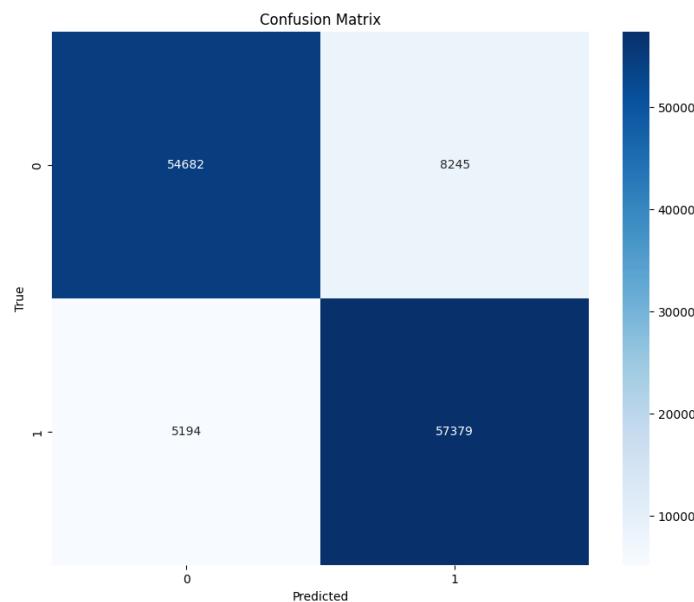


FIGURE 76 – Matrice de confusion du modèle siamois

3. **Courbe de la fonction d'efficacité du récepteur (Receiver Operating Characteristic Roc) :** L'analyse de la courbe ROC-AUC, avec une AUC de 0,89, confirme que le modèle siamois utilisé pour la biométrie vocale est très performant pour distinguer les paires d'audio similaires des non similaires. Cette performance est surtout notable à bas taux de faux positifs (FPR), ce qui signifie que le modèle fait peu d'erreurs en classant des paires d'audio non similaires comme similaires. En outre, le modèle maintient une bonne performance à des taux élevés de vrais positifs (TPR), ce qui montre sa capacité à identifier correctement un grand nombre de paires d'audio similaires. Cette haute précision et faible taux de faux positifs sont cruciaux pour garantir la sécurité et la fiabilité du système de biométrie vocale.

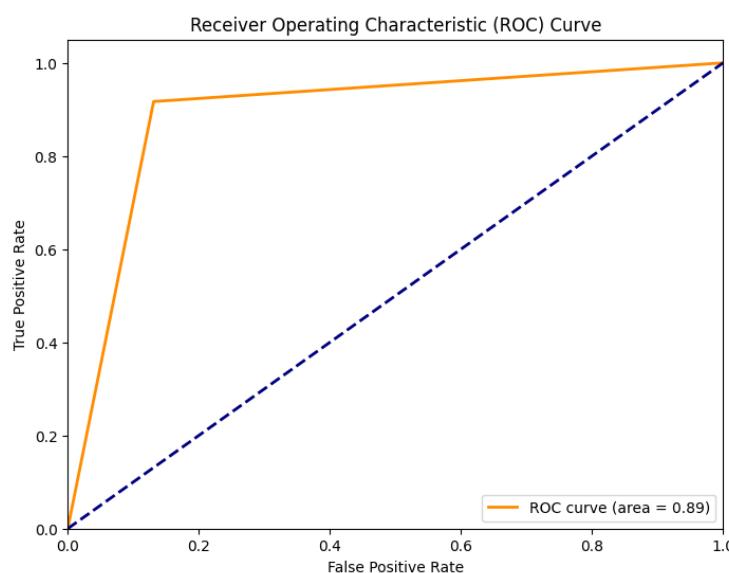


FIGURE 77 – Courbe ROC du modèle siamois

4.5 Conclusion

En résumé, après une analyse détaillée des différentes techniques d'authentification et une évaluation approfondie des performances du modèle siamois pour la biométrie vocale, nous pouvons affirmer que l'authentification vocale constitue la solution la plus optimale pour notre application destinée aux utilisateurs non-voyants. Cette approche présente de nombreux avantages en termes de convivialité et d'accessibilité accrues pour les utilisateurs non-voyants, tout en garantissant une authentification fiable et sécurisée grâce à la précision remarquable du modèle, avec un taux de précision de 92,38% et un score F1 de 91,58%. Dans le chapitre suivant, nous explorerons la mise en oeuvre de la partie compréhension des commandes à l'aide des techniques du NLP.

Chapitre V : Système de Traitement du Langage Naturel

5 Système de Traitement du Langage Naturel

5.1 Introduction

Dans ce chapitre, nous examinons la mise en œuvre de la partie dédiée à la compréhension des commandes textuelles de l'utilisateur. Nous menons une étude de benchmarking pour déterminer le meilleur modèle de reconnaissance d'intention de l'utilisateur, en couvrant toutes les étapes, depuis la création des données et leur analyse exploratoire jusqu'au prétraitement, l'entraînement et la comparaison des résultats. Nous présentons également l'architecture de la pipeline FAQ, conçue pour gérer intelligemment les réponses aux questions fréquentes, ainsi que le processus d'extraction des entités nommées.

5.2 Acquisition de données et prétraitement

5.2.1 Analyse exploratoire des données (EDA)

- Collecte de données :** Notre ensemble de données contient 27 135 phrases, réparties en 18 classes qui représentent les diverses intentions des utilisateurs, couvrant des actions telles que *la consultation des opérations, la gestion des cartes pour ajuster les plafonds, l'activation ou la désactivation, ainsi que des transactions telles que les paiements de factures et les virements*. Il inclut également des fonctionnalités d'assistance clientèle telles que les FAQ. Puisque l'entreprise ne dispose pas de données historiques conversationnelles et qu'un dataset personnalisé de qualité est nécessaire, nous avons utilisé le modèle de langage **ChatGPT** en lui fournissant des prompts adaptés pour générer des phrases pour chaque classe, constituant ainsi notre ensemble de données.

tag	
Consultation_Operations	1952
Gestion_Cartes_Augmenter_Plafond	1796
Consultation_Solde	1738
Confirmation_Action	1707
Transaction_PaiementFacture	1701
Annulation_Action	1664
Gestion_Cartes_Diminuer_Plafond	1624
Gestion_Cartes_Désactivation	1560
Gestion_Cartes_Activation	1560
Gestion_Cartes_Opposition	1554
Transaction_Virement	1549
Gestion_Bénéficiaires_Ajout	1500
Info_Assistance	1474
Consultation_Cartes	1326
Info_Geolocalisation	1170
Info_FAQ	1135
Gestion_Bénéficiaires_Suppression	1111
Gestion_Bénéficiaires_Modification	1014
Name: count, dtype: int64	

df.shape

(27135, 2)

(a) Taille du dataset

(b) Nombre de phrases par classe

FIGURE 78 – Taille du dataset

- NUAGE DE MOTS PAR PHRASE :** Le graphe indique une tendance générale montrant que, globalement, à mesure que le nombre de phrases augmente, le nombre de mots augmente également, bien que cette relation ne soit pas linéaire. Certaines variations et dispersions sont visibles, ce qui suggère que la longueur des phrases varie

considérablement entre différents textes. Ainsi, certains textes avec un nombre de phrases élevé ont un nombre de mots disproportionnellement élevé, ce qui indique l'utilisation de phrases plus longues et complexes.

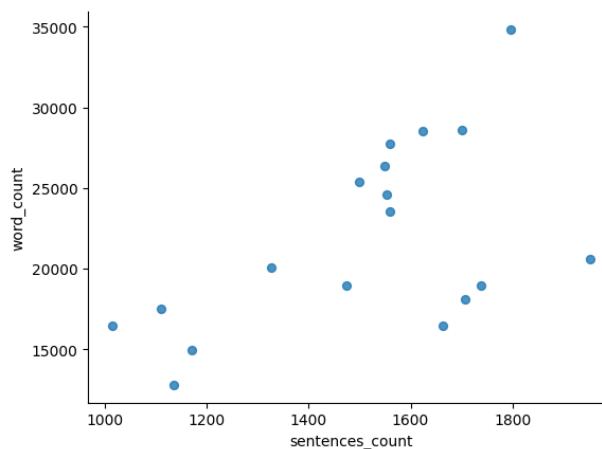


FIGURE 79 – Distribution des mots par phrase

3. **La distribution du nombre de phrases par classe :** On observe à travers le graphe ci-dessous que l'intention **Consultation_Opérations** possède le plus grand nombre de phrases (1952), suivie de **Consultation_Solde** (1738) et **Transaction_PaiementFacture** (1701). Les intentions **Gestion_Cartes_Augmenter_Plond** et **Info_Assistance** présentent également un nombre de phrases élevé (1796 et 1474 respectivement). En revanche, les intentions **Gestion_Bénéficiaires_Modification** et **Gestion_Bénéficiaires_Suppression** ont un nombre de phrases plus faible (1014 et 1111 respectivement). Le graphique nous a permis de révéler une distribution variée des phrases, montrant quelles intentions sont les plus fréquentes dans le corpus de données.

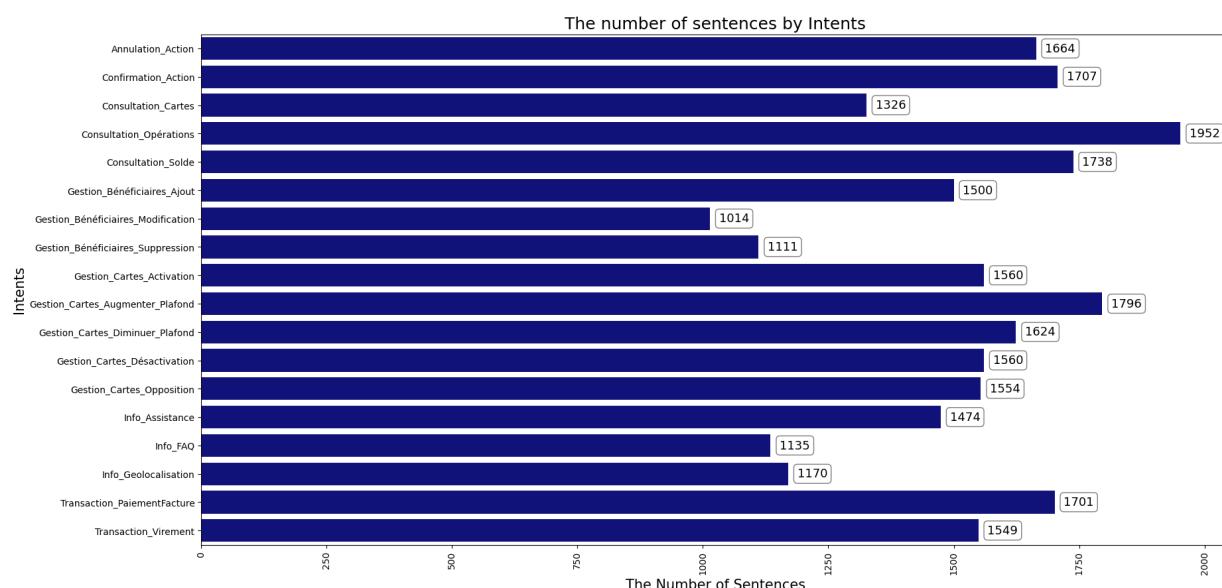


FIGURE 80 – Distribution du nombre de phrases par classe

4. Proportion de chaque classe dans l'ensemble des classes : Pour mieux appréhender l'équilibre entre les différentes classes, nous disposons du diagramme suivant illustrant la répartition de chaque intention. La **consultation des opérations** est l'intention la plus prédominante, constituant 7,2% du total. Elle est suivie de près par **l'augmentation du plafond des cartes** (6,6%), la **consultation de solde** (6,4%), la **confirmation d'action** (6,3%) et le **paiement de factures** (6,1%). D'autres intentions telles que la **diminution du plafond des cartes**, la **désactivation et l'activation des cartes** représentent chacune 5,7%. Les intentions moins représentées incluent les **informations FAQ** (4,2%) et la **suppression de bénéficiaires** (4,1%). Ce graphique révèle une répartition relativement équilibrée des intentions, avec certaines légèrement plus fréquentes que d'autres.

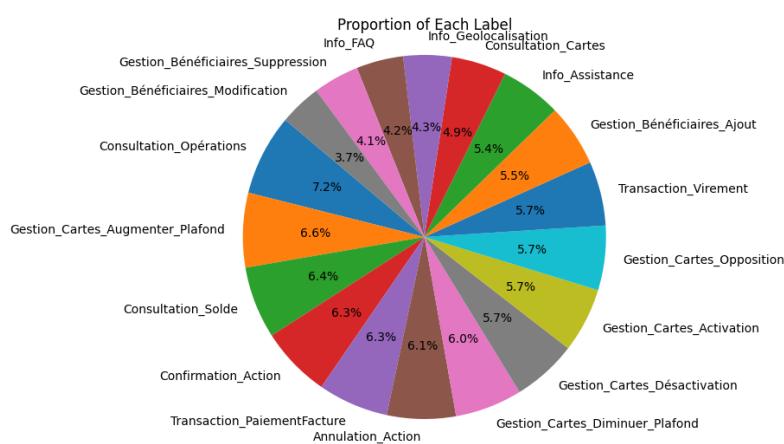


FIGURE 81 – Répartition des intentions

5. Distribution de la longueur des phrases par classe : La distribution des longueurs de phrases pour différentes classes nous est utile pour mieux comprendre la répartition des longueurs de phrases dans notre corpus et de décider de la nécessité de normaliser ou de tronquer les phrases afin d'équilibrer leur représentation. D'après le graphique ,on observe que la majorité des phrases se situe entre **5 et 20 mots**, avec un pic notable autour de **10 mots**. La fréquence des phrases diminue progressivement au-delà de **20 mots**, avec très peu de phrases dépassant les **30 mots**. Certaines classes, telles que **Consultation_Solde** et **Consultation_Operations**, ont des pics plus élevés autour de **10 mots**, tandis que d'autres montrent une plus grande variabilité en termes de longueur de phrases. Globalement, le graphique illustre la diversité et la concentration des longueurs de phrases parmi les classes, indiquant une prépondérance de phrases courtes à moyennes dans l'ensemble des données.

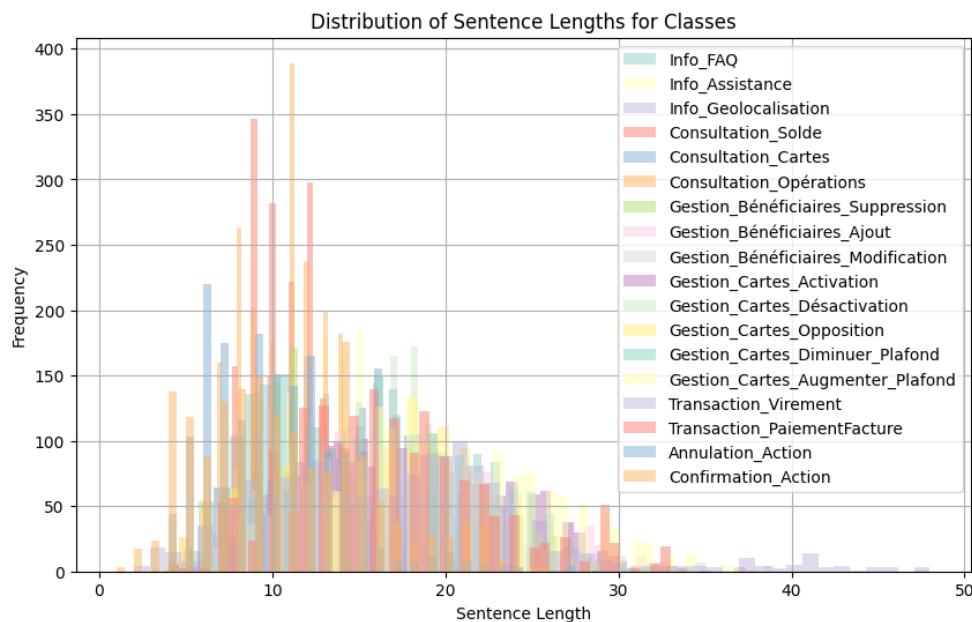


FIGURE 82 – Distribution de la longueur des phrases par classe

6. Nuage de mots fréquents par classe : En utilisant la bibliothèque wordcloud, nous avons créé les graphiques ci-dessous pour visualiser les mots les plus fréquemment utilisés dans chaque classe du corpus textuel. Cette représentation graphique nous permet de repérer rapidement les tendances et les motifs spécifiques à chaque classe du corpus.



FIGURE 83 – Nuage de mots fréquents par classe

5.2.2 Prétraitement et préparation des données

1. Nettoyage des données (Data Cleaning) :

- (a) **Gestion des valeurs manquantes :** Ce processus implique la détection et la suppression des valeurs manquantes. Cependant, étant donné que nous avons créé le jeu de données nous-mêmes, nous n'avons pas de valeurs manquantes à traiter.
- (b) **Suppression des doublons :** Supprimer les lignes en double dans l'ensemble de données pour éviter tout biais potentiel.

```
df_no_duplicates = df.drop_duplicates()

print(df_no_duplicates)

request      0           tag
0   Comment récupérer mon identifiant ?  Info_FAQ
1   Comment récupérer mon mot de passe?  Info_FAQ
2   Comment bénéficier du service Attijari Mobile ?  Info_FAQ
3   Comment souscrire au service Attijarinet?  Info_FAQ
4   Comment accéder à mon espace sécurisé Attijari...  Info_FAQ
...
27039  J'accepte la transaction pour l'achat en ligne... Confirmation_Action
27043  Je confirme le retrait d'argent pour mes dépens... Confirmation_Action
27044  Mon consentement est donné pour le transfert v... Confirmation_Action
27049  J'approuve le transfert vers le compte de ma s... Confirmation_Action
27055  J'approuve le transfert vers le compte de mon ... Confirmation_Action
[23101 rows x 2 columns]
```

(a) Valeurs manquantes

(b) Suppression des valeurs dupliquées

FIGURE 84 – Nettoyage du dataset

- 2. **Équilibrage des données (Data Balancing) :** Après notre analyse EDA, nous avons constaté un léger déséquilibre entre les classes, ce qui pourrait affecter les performances des modèles en introduisant des biais. Pour remédier à cela, nous avons appliqué la technique de suréchantillonnage (oversampling), qui consiste à augmenter artificiellement le nombre d'exemples dans les classes minoritaires en créant des copies aléatoires des échantillons de ces classes. La visualisation ci-dessous illustre le nombre total d'exemples par classe avant et après l'oversampling.

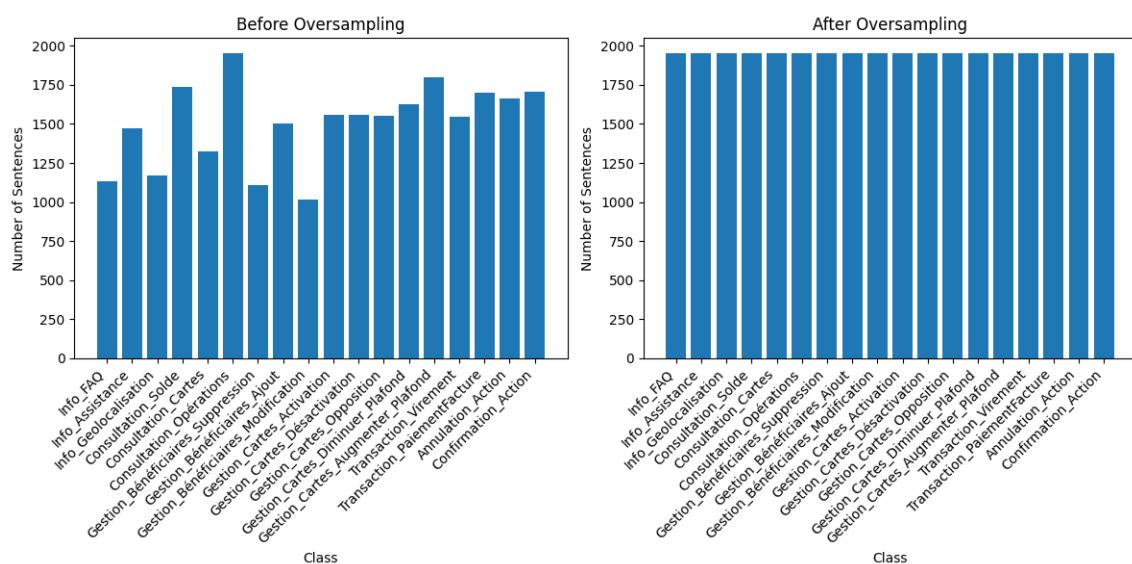


FIGURE 85 – Visualisation du nombre d'échantillons par classe avant et après l'oversampling

3. Transformation des données (Data Transformation) :

- (a) **Encodage des étiquettes des classes** : L'encodage des étiquettes s'est fait avec le **one-hot encoding**. Cette méthode représente chaque classe par un vecteur binaire où chaque position correspond à une classe spécifique, évitant ainsi toute relation d'ordre implicite entre les classes. Le one-hot encoding offre une interprétation claire des prédictions du modèle et est compatible avec une variété d'algorithmes de classification.
- (b) **Prétraitement de texte** :
- Prétraitement commun** : Cette étape comprenait la conversion des textes en minuscules, la suppression de la ponctuation, l'étiquetage des parties du discours (POS-Tagging), la lemmatisation et l'élimination des mots vides (stopwords). En utilisant un modèle de langue française de spaCy, nous avons assuré une précision optimale dans l'analyse linguistique, ce qui a permis de normaliser les textes et de se concentrer sur les mots pertinents et de comprendre leur rôles grammaticaux. La combinaison de **NLTK** pour sa robustesse, **spaCy** pour sa rapidité et précision, et **FrenchLefffLemmatizer** pour sa capacité à normaliser les textes français, nous a permis de réaliser un prétraitement complet et précis des données textuelles.
 - Pour les méthodes de machine learning** : Nous avons utilisé le **TFIDFVectorizer** pour convertir les textes en une matrice de termes en utilisant la fréquence de terme inverse de document (TF-IDF). Cela permet de gérer efficacement les variations de longueur des textes en capturant l'importance relative des mots dans les documents, ce qui est crucial pour des modèles comme les SVM, Random Forest, et XGBoost.
 - Pour les réseaux de neurones profonds** : Pour nos modèles RNN (LSTM et GRU), nous avons choisi les embeddings de **CamemBERT** pour leur capacité à capturer des informations sémantiques pertinentes en français. Nous avons donc utilisé son tokenizer spécifique pour prétraiter notre corpus textuel, assurant ainsi la compatibilité et l'optimisation des embeddings de mots. Les textes ont été convertis en tokens avec des masques d'attention créés pour chaque séquence. Pour le fine-tuning de CamemBERT, nous avons chargé ces embeddings à l'aide des DataLoader de PyTorch et préparé les données pour une itération efficace lors de l'entraînement.

4. Division des données (Dataset Splitting) :

Notre jeu de données a été divisé ensuite en trois segments distincts : **un ensemble d'entraînement**, représentant **80%** du jeu de données, est utilisé pour construire le modèle ; **un ensemble de validation**, constitué de **10%** des données, sert à ajuster les paramètres du modèle et évaluer ses performances pendant l'entraînement pour éviter le surapprentissage ;

enfin, **un ensemble de test**, également de **10%**, est dédié à l'évaluation des performances finales du modèle. Cette démarche permet d'évaluer la capacité de généralisation du modèle sur des données qu'il n'a pas rencontrées lors de l'entraînement.

5.3 Modèles de reconnaissance d'intention

5.3.1 Exploration d'architectures des modèles entraînés

À la base de notre étude de l'état de l'art, nous avons sélectionné plusieurs modèles pour notre étude comparative visant à déterminer le meilleur modèle pour comprendre le contexte. Nous avons opté pour :

1. **Des modèles de machine learning : SVM, Random Forest, et XGBoost** en raison de leur robustesse, de leur performance, de leur capacité à gérer des données de grande dimension, de leur flexibilité et de leur adaptabilité, ainsi que de leur capacité à offrir une certaine interprétabilité. Nous avons utilisé les modèles de la bibliothèque **scikit-learn** (**sklearn**) pour leur facilité d'utilisation et leur efficacité. Nous avons ensuite procédé à une recherche exhaustive des hyperparamètres optimaux pour chaque modèle.
2. **Des RNN à base des embeddings de CamemBERT : LSTM et GRU** en raison de leur capacité à traiter des séquences de données, capturer les dépendances séquentielles à long terme, s'adapter à des entrées de taille variable, et avoir démontré leur efficacité et leur flexibilité dans la modélisation de la structure séquentielle des textes. L'architecture personnalisée de ces modèles **RNNs** avec **CamemBERT** se compose de :
 - **Embedding Layer** : Nous avons choisi de l'initialiser à partir des poids du modèle CamemBERT pré-entraîné, performant en français. Cette couche transforme les indices de mots en vecteurs denses de dimension fixe.
 - **Dropout Layer** : Après la couche d'embedding, un dropout de 0.5 est appliqué pour régulariser le modèle et éviter le surapprentissage.
 - **LSTM (GRU) Layer** : Ensuite, une couche LSTM (GRU) est utilisée pour capturer les dépendances séquentielles dans les données textuelles. La couche LSTM (GRU) a une taille d'entrée égale à la dimension des embeddings (correspondant à la taille des embeddings CamemBERT), une taille cachée de 50, une seule couche et un dropout de 0.5.
 - **Fully Connected (FC) Layer** : Enfin, une couche entièrement connectée est utilisée pour produire les scores de sortie pour chaque classe de sortie. Cette couche a une taille d'entrée de 50 (correspondant à la taille cachée de la dernière couche LSTM (GRU)) et une taille de sortie égale au nombre de classes de sortie. La fonction softmax est appliquée sur la sortie pour obtenir des probabilités de classe.

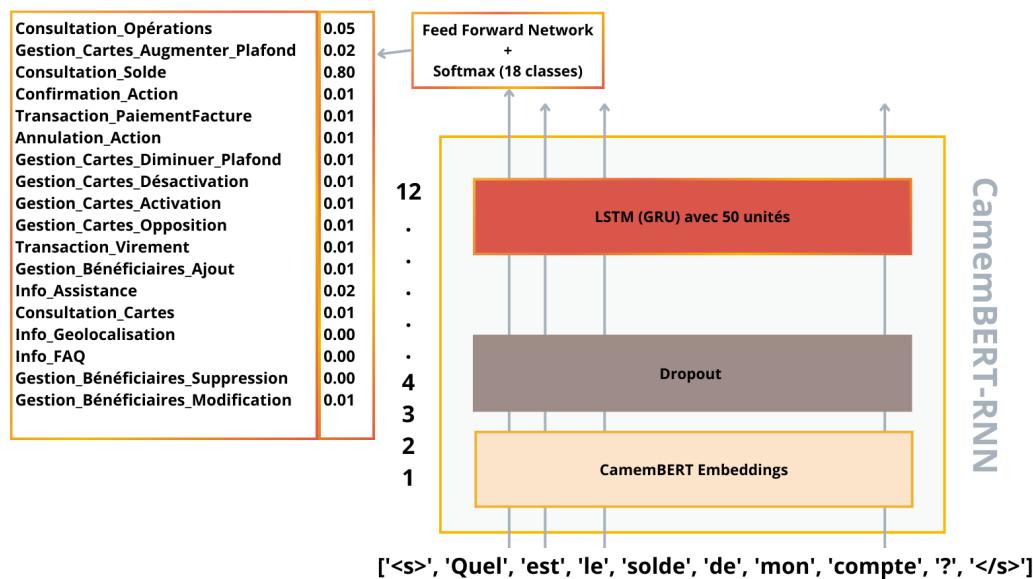


FIGURE 86 – Architecture du modèle CamemBERT-RNN pour la reconnaissance d'intention

Le modèle **CamemBERT-LSTM** utilise une couche LSTM pour la récurrence, tandis que le modèle **CamemBERT-GRU** utilise une couche GRU.

3. **Le fine-tuning de CamemBERT-base** car il saisit les subtilités du langage, gère diverses tâches de traitement du langage, bénéficie du transfert d'apprentissage pour une adaptation rapide, et est facilement accessible via la bibliothèque Hugging Face Transformers.

(a) **L'architecture du modèle de fine-tuning CamemBERT avec le classificateur par défaut :** à l'aide de la classe '**CamemBERTForSequenceClassification**' qui fournit une structure optimisée et simplifiée pour le fine-tuning, réduisant ainsi le temps de développement et augmentant la précision des modèles sur la classification. L'architecture comprend deux composants principaux :

i. **Le modèle CamemBERT sous-jacent :** similaire à RoBERTa, avec des embeddings pour les tokens, les positions et les types de tokens pour saisir les informations contextuelles des séquences de texte. Ensuite, il comprend 12 couches d'attention auto-régressive pour analyser les relations complexes entre les tokens. Chaque couche comprend des sous-couches d'attention, de transformation intermédiaire et de normalisation, ce qui lui permet de capturer efficacement la structure et le sens des phrases.

ii. **Le classificateur par défaut :** ajoute une couche dense, une couche dropout avec une probabilité de 0.1, et une couche de sortie linéaire ajustée au nombre de classes (18).

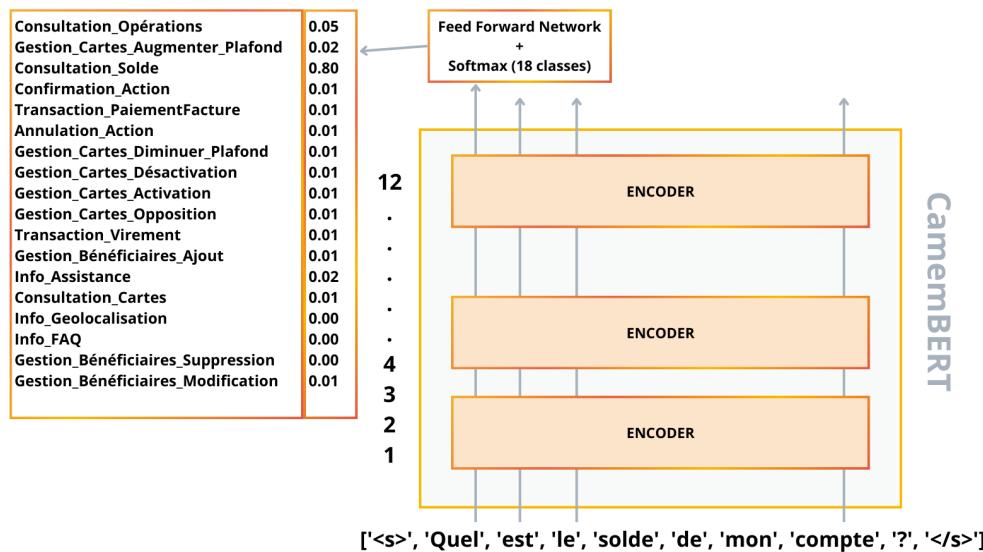


FIGURE 87 – Architecture du modèle CamemBERT avec le classificateur par défaut pour la reconnaissance des intentions

(b) **L'architecture du modèle CamemBERT avec le classificateur personnalisé :** combine la puissance du modèle pré-entraîné CamemBERT avec un classificateur que nous avons personnalisé pour s'adapter à notre besoin. Les embeddings et les couches d'attention de CamemBERT fournissent des représentations riches des séquences textuelles, tandis que le classificateur personnalisé affine ces représentations pour prédire les classes cibles. Le classificateur se compose de plusieurs couches linéaires et de dropout, suivies de fonctions d'activation ReLU, permettant d'améliorer la généralisation et la performance du modèle.

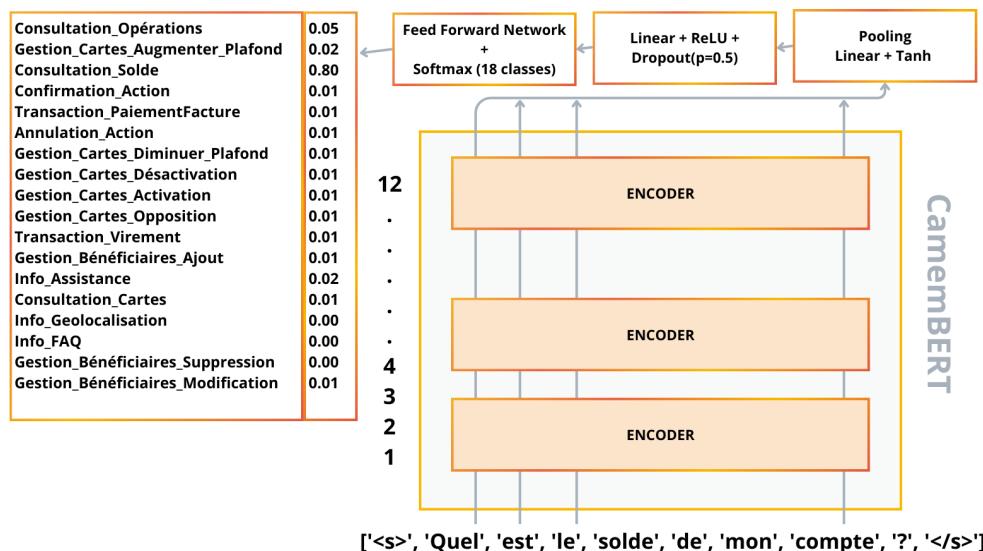


FIGURE 88 – Architecture du modèle CamemBERT avec le classificateur personnalisé pour la reconnaissance des intentions

5.3.2 Entraînement des Modèles

1. Pour les méthodes traditionnelles de ML : Pour entraîner nos modèles, nous avons utilisé plusieurs techniques pour optimiser les hyperparamètres, éviter l'overfitting et sélectionner les meilleurs paramètres pour des performances optimales :
 - (a) **Validation Croisée** : pour estimer la performance du modèle de manière robuste et éviter le surapprentissage.
 - (b) **Hyperparameter Tuning** : pour chaque modèle, la **GridSearchCV** a été utilisé pour rechercher systématiquement les meilleurs hyperparamètres. Par exemple, pour **SVM**, nous avons optimisé les paramètres du noyau tels que C et gamma. Pour **Random Forest**, nous avons ajusté des paramètres tels que le nombre d'arbres (n_estimators), la profondeur maximale des arbres (max_depth). De même, pour **XGBoost**, nous avons optimisé le nombre d'arbres (n_estimators), la profondeur des arbres (max_depth), et le taux d'apprentissage (learning_rate).
 - (c) **Regularisation** : nous avons appliqué des techniques de régularisation telles que le paramètre C dans SVM et les paramètres de régularisation dans XGBoost pour limiter la complexité du modèle et éviter le surapprentissage.
 - (d) **Early Stopping** : nous avons utilisé l'arrêt précoce dans les modèles qui le supportent, comme XGBoost, pour arrêter l'entraînement lorsque les performances sur le jeu de validation cessent de s'améliorer, ce qui aide à prévenir l'overfitting.

Ce tableau présente les meilleurs paramètres (best_params) et le meilleur score (best_score) pour chaque modèle trouvés à l'aide du GridSearch :

Model	best_params	best_score
SVM	'C' : 1, 'gamma' : 'scale', 'kernel' : 'linear'	0.995725
Random Forest	'max_depth' : 10, 'max_features' : 'sqrt', 'max_leaf_nodes' : 30, 'min_samples_leaf' : 1, 'min_samples_split' : 10, 'n_estimators' : 150	0.895784
XGBoost	'learning_rate' : 0.1, 'max_depth' : 5, 'n_estimators' : 200	0.9875

TABLE 21 – Meilleurs paramètres et score pour chaque modèle

2. Pour CamemBERT-LSTM et CamemBERT-RNN : Les modèles sont entraînés sur 10 epochs en utilisant des données converties en tensors PyTorch et exécutées sur GPU. Un DataLoader ensuite gère le chargement des mini-batchs pour une accélération matérielle. Pendant l'entraînement, le modèle utilise des passes avant et arrière pour ajuster les paramètres avec l'optimiseur Adam, en minimisant la perte de cross-entropy. L'early stopping surveille la perte de validation pour arrêter l'en-

traînement si la performance ne s'améliore pas, évitant ainsi l'overfitting et réduisant le temps de calcul.

- (a) **Pour le CamemBERT-LSTM :** Les courbes de perte et de précision montrent que le modèle CamemBERT-LSTM apprend efficacement et généralise bien sans surapprentissage notable. La perte d'entraînement diminue rapidement de 2.6 à 2.1 dans les premières epochs, puis plus lentement jusqu'à 2.0, tandis que la perte de validation suit une tendance similaire, descendant de 2.2 à 2.0. La précision d'entraînement augmente rapidement de 0.5 à 0.8, puis atteint 0.98, et la précision de validation passe de 0.8 à 0.99. Ces résultats indiquent une bonne généralisation du modèle sur les données de validation.

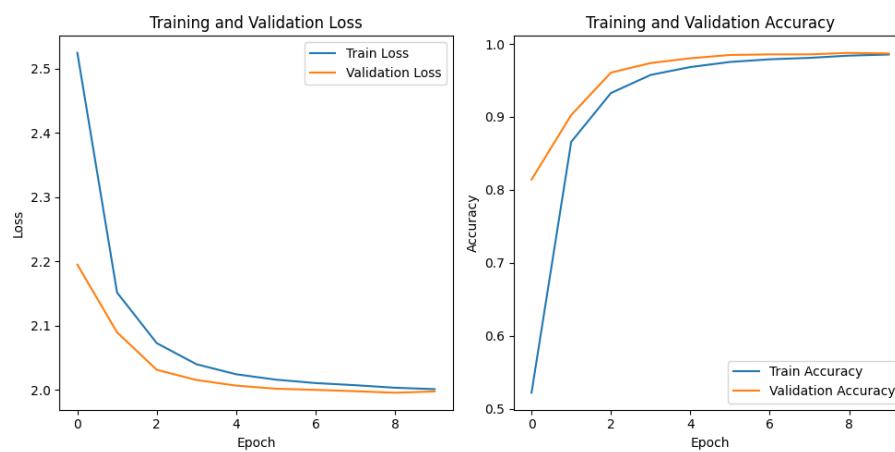


FIGURE 89 – Les courbes d'apprentissage de CamemBERT-LSTM

- (b) **Pour le CamemBERT-GRU :** Le modèle CamemBERT-GRU démontre une efficacité remarquable et une bonne généralisation sur les données de validation. Les courbes de perte témoignent d'un apprentissage cohérent et convergent, avec des pertes d'entraînement et de validation se stabilisant autour de 2.0 après 10 epochs, indiquant un bon apprentissage sans surapprentissage. De même, les courbes de précision révèlent une augmentation rapide de la précision d'entraînement et de validation, atteignant environ 0.98, confirmant ainsi la capacité du modèle à généraliser efficacement sur les données de validation.

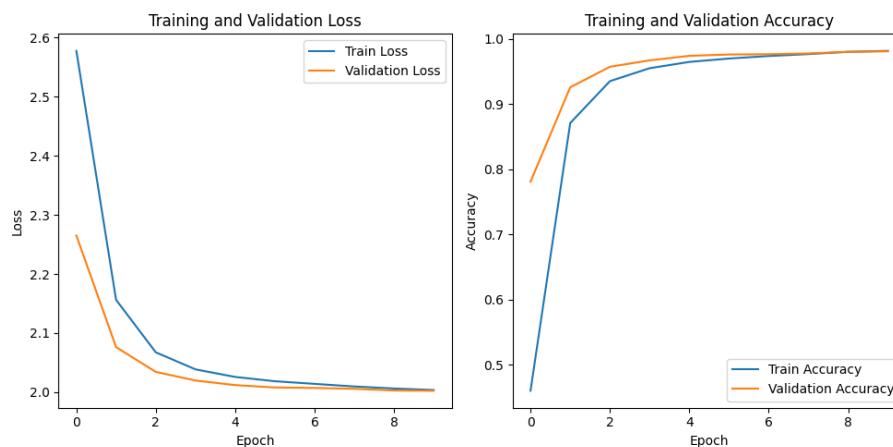


FIGURE 90 – Les courbes d'apprentissage de CamemBERT-GRU

3. Pour le fine-tuning de CamemBERT : Nous avons utilisé l'optimiseur AdamW, qui combine l'adaptation du taux d'apprentissage de 2e-5 d'Adam avec une régularisation L2 pour prévenir le surajustement en pénalisant les grandes valeurs de paramètres. Pour calculer l'erreur , nous avons utilisé la fonction de perte de cross-entropy , adaptée à la classification. Le modèle est entraîné pendant 5 epochs pour le classificateur par défaut et 3 pour le classificateur personnalisé, avec des données divisées en mini-batchs pour une gestion efficace de la mémoire et un calcul rapide sur GPU. À chaque batch, les gradients sont réinitialisés, une passe avant et arrière est effectuée pour mettre à jour les paramètres, et la précision et la perte sont suivies. Après chaque epoch, le modèle est évalué sur un ensemble de validation en mini-batchs.

(a) **Fine-tuning de CamemBERT-base avec le classificateur par défaut :** montre une excellente performance. La perte d'entraînement chute rapidement de 0,6 à environ 0,05, tandis que la perte de validation reste stable et proche de 0, indiquant une bonne généralisation. La précision d'entraînement atteint près de 100% après 2 epochs, avec une légère baisse vers la cinquième epoch. La précision de validation est élevée dès le début (99%) et atteint pratiquement 100% à la fin des 5 epochs. Ces résultats suggèrent que le modèle CamemBERT fine-tuné est très performant pour notre tâche, avec des pertes faibles et des précisions élevées.

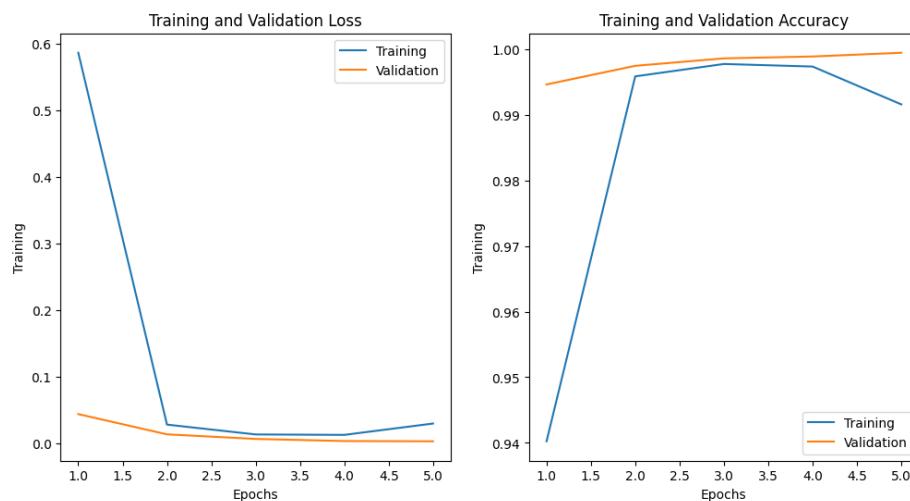


FIGURE 91 – Les courbes d'apprentissage du fine-tuning de CamemBERT avec le classificateur par défaut

(b) **Fine-tuning de CamemBERT-base avec le classificateur personnalisé :** La perte d'entraînement diminue rapidement de plus de 1 à presque 0 après deux époques, tandis que la perte de validation diminue légèrement et se stabilise, indiquant une bonne généralisation. La précision d'entraînement atteint presque 100% après deux époques, et la précision de validation est extrêmement élevée dès le début, passant de 99,77% à 99,94% vers la fin du troisième epoch. Globalement, le modèle fine-tuné avec le classificateur personnalisé montre des performances exceptionnelles avec des pertes très faibles et des précisions presque parfaites.

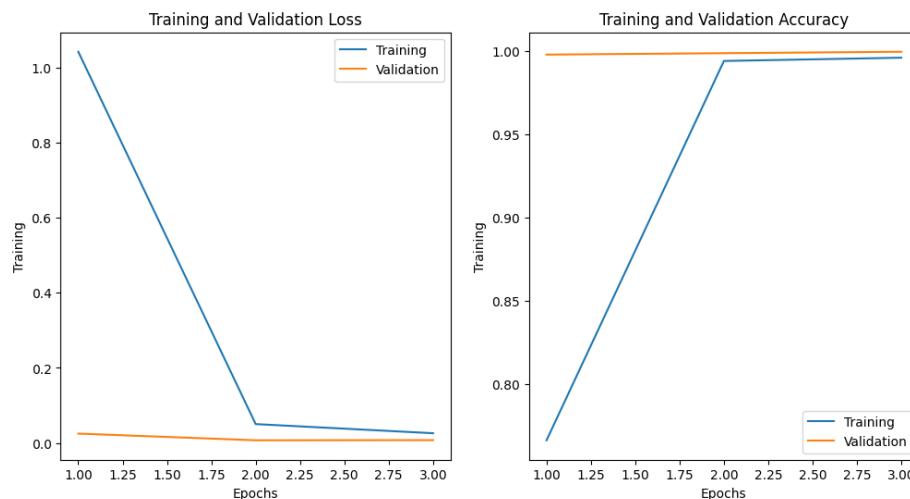


FIGURE 92 – Les courbes d'apprentissage du fine-tuning de CamemBERT avec le classificateur personnalisé

5.3.3 Résultats et Comparaison

Dans notre étude comparative, nous avons évalué les performances des sept modèles de classification sur les données de test. Les résultats sont regroupés dans le tableau ci-dessous :

Modèle	Accuracy	Précision	Rappel	F1-Score
SVM	0.9957	0.9957	0.9957	0.9957
Random Forest	0.8958	0.9031	0.8958	0.8876
XGBoost	0.9875	0.9876	0.9875	0.9875
LSTM	0.9869	0.9869	0.9869	0.9868
GRU	0.9812	0.9813	0.9812	0.9810
CamemBERT (default)	0.9989	0.9989	0.9989	0.9989
CamemBERT (custom)	0.9977	0.9977	0.9977	0.9977

TABLE 22 – Performance des différents modèles de classification

- Les résultats montrent que le modèle **CamemBERT** fine-tuné avec le classificateur par défaut obtient les meilleures performances globales, avec une précision, un rappel et un F1-score de **0.9989**. Le modèle CamemBERT avec le classificateur personnalisé suit de près avec des scores de **0.9977** dans toutes les métriques.
- Ces performances exceptionnelles des modèles CamemBERT peuvent être attribuées à plusieurs facteurs. Premièrement, les modèles de type transformateur, comme CamemBERT, sont capables de capturer des dépendances contextuelles complexes dans les données textuelles, ce qui améliore la précision de la classification. De plus, le fine-tuning permet d'adapter précisément le modèle pré-entraîné aux caractéristiques spécifiques du jeu de données utilisé, ce qui augmente encore la performance.
- Les modèles basés sur les réseaux de neurones récurrents (LSTM et GRU) montrent également de bonnes performances, avec des scores de précision et de rappel proches de 0.98. Ces modèles sont efficaces pour traiter les données séquentielles, et leur capacité à conserver des informations contextuelles sur de longues séquences leur permet de bien performer dans les tâches de classification de texte.
- SVM affiche une accuracy de 0.9957, ce qui en fait une alternative solide parmi les modèles d'apprentissage automatique classiques. SVM est connu pour ses bonnes performances dans les tâches de classification grâce à sa capacité à trouver des hyperplans optimaux pour séparer les classes, surtout dans des espaces de haute dimension.
- Le modèle XGBoost obtient également de très bons résultats avec une précision de 0.9875. XGBoost est un algorithme de boosting efficace qui améliore les performances en combinant plusieurs modèles faibles pour créer un modèle fort. Sa robustesse et sa capacité à gérer les données déséquilibrées contribuent à ses bons résultats.

- En revanche, le modèle Random Forest a les performances les plus faibles dans cette comparaison, avec une accuracy de 0.8958 et un F1-score de 0.8876. Bien que Random Forest soit généralement performant pour une variété de tâches, il semble moins adapté pour cette tâche de classification spécifique, peut-être en raison de sa moindre capacité à capturer des dépendances complexes dans les données textuelles par rapport aux modèles de type transformateur et récurrents.
- Ces résultats soulignent la supériorité des modèles fine-tunés de CamemBERT pour notre tâche de classification, surpassant à la fois les modèles d'apprentissage automatique traditionnels et les modèles basés sur les réseaux de neurones récurrents.

Performances du meilleur modèle : Pour tous les modèles, nous avons évalué les performances à l'aide du rapport de classification, du F1-score, de la précision, du rappel, de l'accuracy, de la matrice de confusion et des courbes ROC-AUC. Étant donné que le meilleur modèle est le CamemBERT fine-tuné avec le classificateur par défaut, nous présenterons ses performances exclusives dans ce qui suit :

1. Rapport de classification et métriques de performances :

- **Accuracy (précision globale)** : 99.89%. Cela signifie que le modèle a correctement classé près de 99.89% des exemples de test.
- **Precision (précision)** : 99.89%. Cela indique la proportion d'exemples positifs identifiés correctement par le modèle parmi tous les exemples qu'il a classés comme positifs.
- **Recall (rappel)** : 99.89%. Cela représente la proportion d'exemples positifs que le modèle a correctement identifiés parmi tous les exemples positifs réels.
- **F1 Score** : 99.89%. C'est une mesure qui combine à la fois la précision et le rappel en une seule métrique. Il est utile lorsque les classes sont déséquilibrées.

Le rapport de classification fournit des détails supplémentaires sur la performance du modèle pour chaque classe individuelle. Il montre que le modèle a atteint des scores élevés pour toutes les classes, avec une précision, un rappel et un F1-score de 100% pour la plupart des classes. Cela suggère une capacité exceptionnelle du modèle à généraliser et à identifier correctement les différentes classes dans les données de test.

```

Accuracy: 0.9988616960728515
Precision: 0.9988645830227372
Recall: 0.9988616960728515
F1 Score: 0.9988617057846698

Classification Report:
precision    recall   f1-score   support
          0       0.99     1.00     1.00      184
          1       1.00     1.00     1.00      190
          2       1.00     1.00     1.00      199
          3       1.00     1.00     1.00      195
          4       1.00     1.00     1.00      197
          5       0.99     0.99     0.99      178
          6       1.00     1.00     1.00      206
          7       1.00     1.00     1.00      185
          8       1.00     1.00     1.00      198
          9       1.00     1.00     1.00      206
         10      1.00     1.00     1.00      183
         11      1.00     1.00     1.00      199
         12      1.00     1.00     1.00      178
         13      1.00     0.99     1.00      194
         14      1.00     1.00     1.00      210
         15      1.00     1.00     1.00      197
         16      1.00     1.00     1.00      206
         17      1.00     1.00     1.00      209

accuracy           1.00      1.00      1.00      3514
macro avg        1.00      1.00      1.00      3514
weighted avg     1.00      1.00      1.00      3514

```

FIGURE 93 – Rapport de classification et métriques de performances du meilleur modèle

- Matrice de confusion :** Bien que des métriques globales comme l'accuracy, la précision, le rappel et le F1-score donnent une vue d'ensemble des performances du modèle, la matrice de confusion permet une analyse plus détaillée. Elle montre les prédictions correctes et incorrectes pour chaque classe, aidant à identifier les forces et faiblesses spécifiques du modèle. Celle-ci affiche une haute précision, avec presque toutes les prédictions correctes, comme l'indiquent les valeurs élevées sur la diagonale. Les rares erreurs de prédiction, principalement entre "Info_Assistance" et "Info_FAQ", ainsi qu'entre les classes de gestion de bénéficiaires pour ajout et modification, quand il s'agit de phrases très ambiguës.

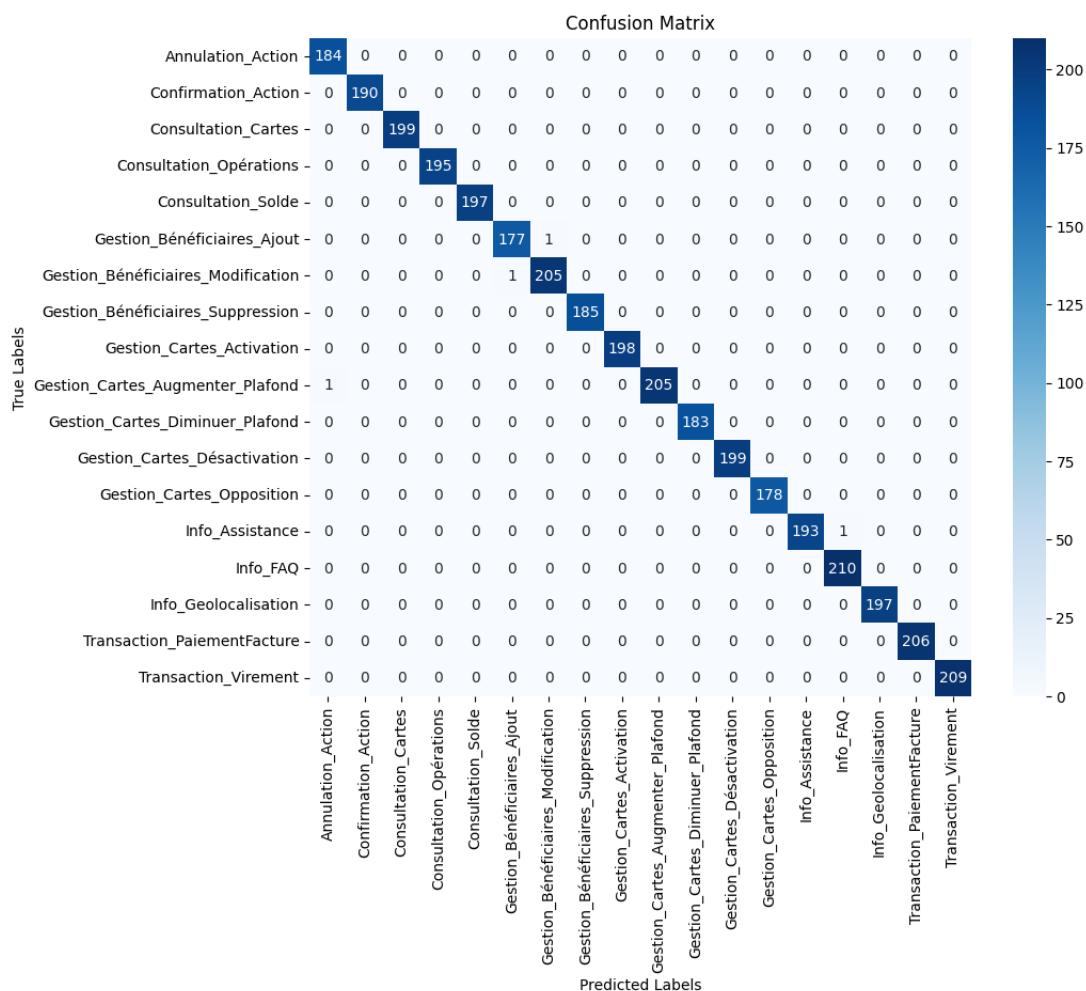


FIGURE 94 – Matrice de confusion du meilleur modèle

5.4 Extraction d'Entités Nommées (NER)

Dans notre projet, l'extraction d'entités nommées (NER) revêt d'une grande importance pour récupérer des informations clés comme les montants, les dates, les numéros de compte, etc., à partir des commandes des utilisateurs. Cela permet au chatbot de comprendre et traiter précisément leurs requêtes et de leur fournir des réponses pertinentes.

- Comprendre les entités à extraire :** Il est crucial de différencier les entités génériques des entités spécifiques à notre entreprise. Cette distinction nous permettra de choisir les meilleures approches de NLP pour chaque type d'entité. Les entités génériques telles que les montants, les dates et heures, les libellés des opérations, les services des cartes, ainsi que les noms et prénoms des bénéficiaires sont couramment utilisées en NLP et n'exigent pas d'adaptation spécifique. En revanche, des entités métier comme les types de compte, les numéros de compte, les numéros de facture, les RIB, les types de carte, et les types de virement nécessitent une attention particulière et une adaptation précise à nos besoins bancaires.
- Choix des techniques NLP pour l'extraction :** Face à la diversité des choix possibles en matière de techniques de NLP pour l'extraction d'entités, deux ap-

proximes ont été retenues pour leur efficacité et leur adaptabilité :

- (a) **Expressions régulières (Regex)** : Les expressions régulières sont utilisées pour détecter les entités spécifiques dans le texte, tels que les numéros de compte, les montants d'argent et les types de cartes. Elles offrent une méthode rapide et précise pour l'extraction d'entités avec des règles personnalisables.
- (b) **Modèle pré-entraîné CamemBERT-ner-with-dates de Hugging Face** : spécialisé dans la reconnaissance d'entités nommées, avec une capacité étendue à identifier les dates. Entraîné sur un large ensemble de données en français, ce modèle offre une précision élevée dans l'extraction d'entités complexes. Cependant, en raison de contraintes de données et du processus laborieux d'annotation requis pour le fine-tuning, nous avons choisi d'utiliser directement ce modèle pré-entraîné pour notre projet. Cette approche garantit une extraction précise des entités, y compris les noms et les dates implicites, comme "hier" ou "la semaine dernière".

Métrique	Précision	Rappel	Score F1	Support	Description
Globale	0.928	0.928	0.928	N/A	Performance globale du modèle
LOC (Localisation)	0.929	0.932	0.931	9510	Entités représentant des emplacements géographiques tels que des villes, des pays, des rues, etc.
PER (Personne)	0.952	0.965	0.959	9399	Entités représentant des noms de personnes incluant des prénoms, des noms de famille, des titres, etc.
MISC (Divers)	0.878	0.844	0.860	5364	Entités regroupant divers types tels que des noms d'organisations, des événements, des produits, etc.
ORG (Organisation)	0.848	0.883	0.865	2299	Entités représentant des noms d'organisations telles que des entreprises, des institutions, des partis politiques, etc.
Date	Non pertinent en raison de la méthode utilisée pour ajouter les balises de date dans le jeu de données Wiki-NER (F1 estimé 90%)				Entités représentant des références temporelles telles que des dates précises, des périodes de temps, etc.

TABLE 23 – Performance du modèle NER

5.5 Pipeline de FAQs

1. **Besoin** : Nous visons à créer un assistant bancaire intelligent qui excelle dans la précision, l'automatisation et la cohérence des réponses, même face à des questions reformulées ou multiples. Ce chatbot sera capable de comprendre le langage naturel, d'extraire des informations pertinentes et de fournir des réponses de manière similaire à un humain.

2. Architecture et fonctionnement :

- (a) **Chargement des Données** : Le processus démarre en extrayant les paires de questions et de réponses des FAQ d'un fichier CSV à l'aide de CSVLoader de LangChain. Ce composant est spécialement conçu pour structurer ces données, facilitant ainsi leur intégration et leur gestion efficace.
- (b) **Encodage** : Ensuite, les questions sont converties en vecteurs denses par le biais du modèle de traitement du langage naturel CamemBERT de Hugging Face. Conçu spécifiquement pour le français, CamemBERT améliore la précision et la compréhension des réponses dans cette langue.
- (c) **Stockage** : Les vecteurs ainsi générés sont ensuite stockés dans une base de données vectorielle FAISS de Meta, optimisée pour la recherche de similarités. Cette architecture permet de retrouver rapidement les réponses les plus pertinentes en se basant sur la similarité des vecteurs.
- (d) **Requête Utilisateur** : Lorsqu'un utilisateur pose une question, celle-ci est encodée en vecteur et comparée aux vecteurs stockés dans FAISS à l'aide de la fonction RetrievalQA. Cela garantit que les réponses proposées correspondent étroitement aux questions initiales.
- (e) **Récupération et Génération de Réponse** : Les réponses les plus adaptées sont récupérées et soumises au modèle LLM de Google AI. Celui-ci les reformule de manière cohérente et naturelle. Le choix de Google AI s'explique par sa gratuité, contrairement à Meta qui nécessite le téléchargement d'un modèle volumineux pour l'inférence.
- (f) **Réponse Utilisateur** : Enfin, l'API NLP fournit une réponse cohérente et précise à l'utilisateur, garantissant ainsi une interaction fluide.

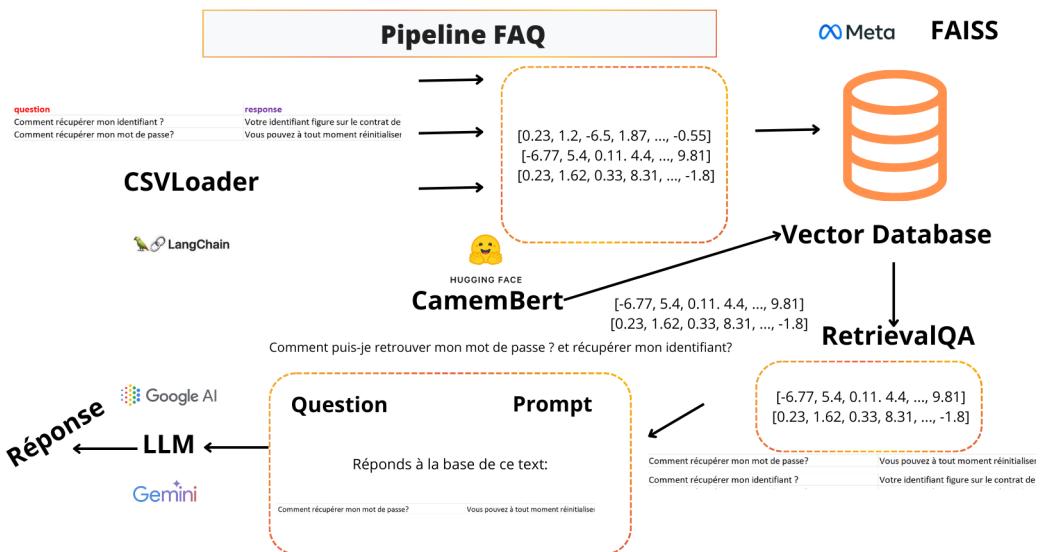


FIGURE 95 – Architecture de la pipeline de la gestion des FAQ

5.6 Conclusion

Dans ce chapitre, nous avons examiné en détail les éléments clés pour construire le système NLP. Nous avons débuté par une analyse approfondie des étapes nécessaires à la compréhension des commandes textuelles de l'utilisateur, incluant une étude comparative des modèles de reconnaissance d'intention. En parallèle, nous avons élaboré une architecture pour gérer les questions fréquentes, intégrant des techniques avancées d'extraction des entités nommées. Ce chapitre a souligné l'importance d'une approche méthodique pour garantir des résultats précis et pertinents, depuis l'analyse des données jusqu'à l'entraînement des modèles.

Chapitre VI : Mise en oeuvre

6 Mise en oeuvre

6.1 Introduction

Ce chapitre détaille la mise en œuvre de la partie développement logiciel de notre projet. Il couvre le développement, les tests et la documentation de l'API bancaire de simulation, de l'API de biométrie vocale et de l'API NLP. Il inclut également le développement de l'application mobile. Enfin, le chapitre décrit les intégrations réalisées entre les différentes API et l'application mobile pour garantir une mise en œuvre efficace et intégrée du projet.

6.2 Développement des APIs

6.2.1 API bancaire de simulation

Cette API gère les données bancaires des clients, permettant l'authentification des utilisateurs, l'interrogation et la manipulation des données pour répondre aux besoins des clients en matière de consultation de solde, d'opérations, de gestion des cartes, ainsi que pour les transactions, paiements de factures, et autres actions bancaires.

1. Composants de l'architecture

(a) Composants matériels :

- L'API repose sur une **machine physique** pour son exécution, exploitant ainsi les ressources matérielles disponibles. **IntelliJ IDEA**, utilisé également pour le développement, fournissant un environnement intégré pour la conception de l'application.
- Le **serveur d'application**, développé avec **Spring Boot**, assure le traitement des requêtes HTTP et la gestion de l'état de l'application, grâce à sa facilité de configuration et sa capacité à créer des applications autonomes prêtes pour la production.
- En ce qui concerne le stockage des données, un **serveur de base de données PostgreSQL** est sélectionné pour sa fiabilité et ses fonctionnalités avancées de gestion de bases de données relationnelles.

(b) Composants logiciels :

L'architecture repose sur un schéma MVC avec des tokens JWT pour l'authentification et l'autorisation. Les requêtes HTTP émises par le client, tel que Postman, sont traitées par un filtre JWT qui valide le token et met à jour le SecurityContextHolder. Si le token est valide, la requête est dirigée vers le contrôleur via le DispatcherServlet. Ce dernier utilise les services pour appliquer la logique métier et accéder aux données via les Repository Classes, qui interagissent avec la base de données PostgreSQL. Enfin, le contrôleur renvoie une réponse JSON au client. L'API s'appuie sur un **serveur HTTP**

(Apache Tomcat) intégré pour gérer les requêtes et la communication client-serveur, et sur Spring MVC pour structurer le développement en séparant les préoccupations de présentation, contrôle et modèle. Spring Data JPA facilite l'accès aux bases de données en utilisant JPA, réduisant la quantité de code nécessaire pour les opérations CRUD et permettant de définir des requêtes complexes de manière déclarative. Enfin, Spring Security garantit la sécurité de l'application en gérant l'authentification et l'autorisation des utilisateurs, créant un environnement sécurisé pour l'API.

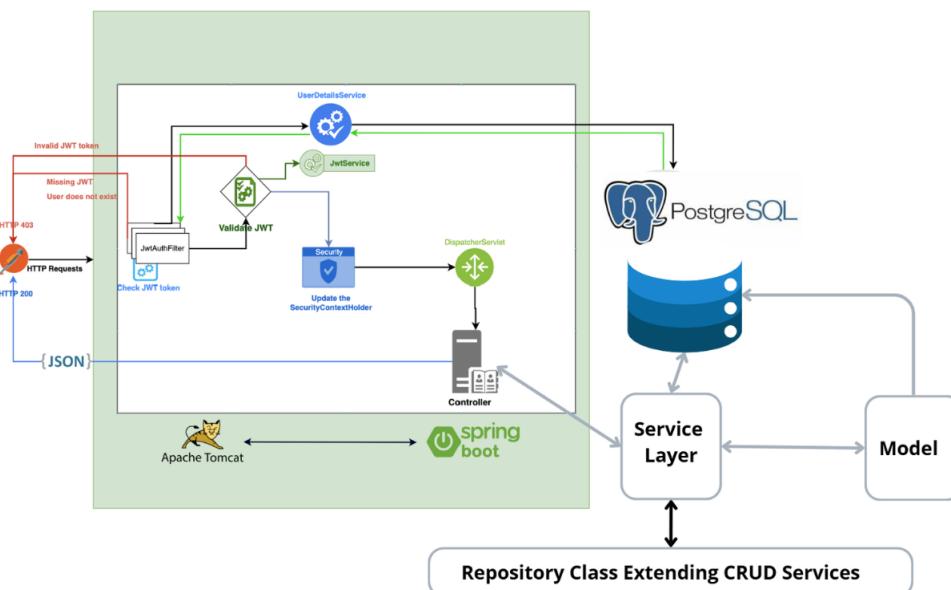


FIGURE 96 – Architecture Logicielle API Bancaire

Cette architecture utilise des composants logiciels à savoir :

2. Flux de données

(a) Requête HTTP :

- Le client envoie une requête HTTP avec un token JWT dans l'en-tête.

(b) Vérification du Token JWT :

- Le JwtAuthFilter intercepte la requête et vérifie la présence du token JWT.
- Le JwtService valide le token JWT. Si le token est valide, les informations de l'utilisateur sont récupérées via UserDetailsService.

(c) Mise à jour du SecurityContextHolder :

- Le SecurityContextHolder est mis à jour avec les détails de l'utilisateur authentifié.

(d) Traitement de la Requête par le Contrôleur :

- Le DispatcherServlet redirige la requête vers le contrôleur approprié.

(e) **Logique Métier :**

- Le contrôleur appelle les services nécessaires pour effectuer les opérations métier.

(f) **Accès aux Données :**

- Les services utilisent les Repository Classes pour interagir avec la base de données PostgreSQL.

(g) **Réponse HTTP :**

- Le contrôleur renvoie une réponse HTTP (JSON) au client.

3. Structure du projet

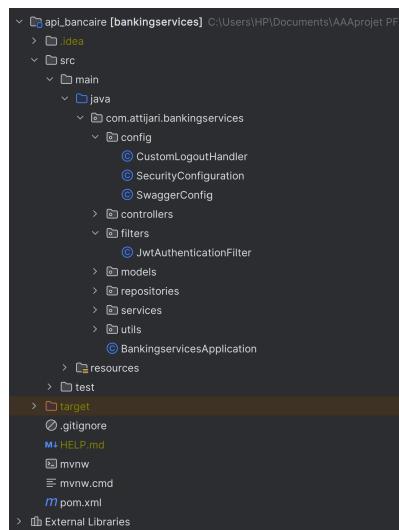


FIGURE 97 – Structure du projet de l'API bancaire

- config :** Ce dossier contient les fichiers de gestion de la configuration de sécurité **JWT** de l'application, ainsi que la configuration de **Swagger** pour générer une documentation **OpenAPI**. Ces fichiers définissent les règles d'autorisation et d'authentification, configurent les filtres de sécurité, spécifient la gestion des exceptions et des déconnexions, et fournissent les informations de base de la documentation de l'API.
- controllers :** Ce dossier contient les classes de contrôleurs de l'API pour la gestion et le traitement des requêtes HTTP entrantes en coordination avec les services appropriés.
- filters :** Ce dossier contient un filtre qui assure que les requêtes entrantes sont authentifiées à l'aide de jetons JWT valides avant d'être autorisées à accéder aux ressources protégées de l'API.
- models :** Ce dossier contient les classes de modèles de données de l'API. Ces classes représentent les entités métier de l'application et sont utilisées pour mapper les données entre la base de données et l'API.

- (e) **repositories** : Ce dossier contient les interfaces de repository de l'API , utilisées pour interagir avec la base de données et effectuer des opérations de lecture et d'écriture sur les données.
- (f) **services** contient les interfaces de service de l'API. Les services sont responsables de la logique métier de l'application. Il encapsule également le sous-dossier **implementations** contient les implémentations des services. Ces classes implémentent les interfaces de service et contiennent la logique réelle pour traiter les requêtes de l'API.
- (g) **utils** : Ce dossier contient des classes utilitaires qui fournissent des fonctionnalités réutilisables à travers l'application.
- (h) **BankingservicesApplication.java** : Ce fichier contient la classe principale de l'application Spring Boot. C'est à partir de cette classe que l'application est démarrée.
- (i) **tests** : Ce dossier contient les tests de l'application, y compris les tests unitaires et les tests d'intégration.
- (j) **Resources** : Ce répertoire héberge le fichier **application.properties**, qui stocke les configurations essentielles de l'application. Il contient divers paramètres, notamment les informations de connexion à la base de données, les configurations JPA, les paramètres de Swagger et les configurations de sécurité.

4. Tests et documentation

- (a) **Tests unitaires et d'intégration** : Le testing des services offre une couverture exhaustive des fonctionnalités de l'API, assurant une meilleure qualité et résilience du code. Les tests unitaires isolent les parties individuelles du code pour garantir leur bon fonctionnement indépendamment. Nous avons simulé les dépendances des services avec **Mockito** pour des tests isolés et répétables. Les tests d'intégration vérifient les interactions entre les différentes parties du système, comme les services REST et la base de données. **JUnit** et **Mockito** sont choisis pour leur simplicité, leur intégration avec Spring, et leur capacité à automatiser les tests, simplifiant ainsi le développement et la maintenance du code. L'ensemble de ces tests est regroupé dans le dossier **test** du projet.

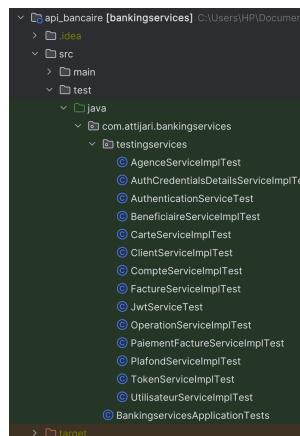


FIGURE 98 – Tests unitaires et d'intégrations de l'API Spring Boot

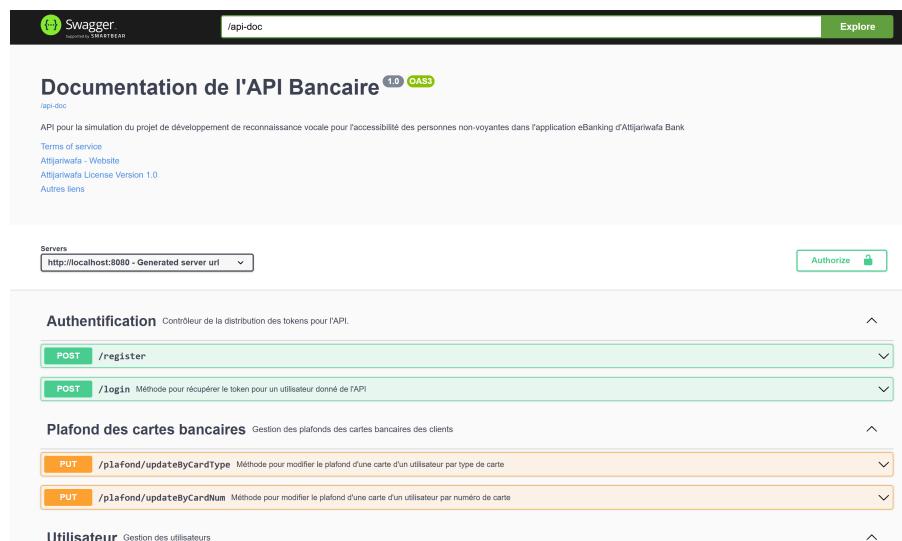
(b) **Tests Fonctionnels :** Nous avons validé les fonctionnalités de l'API en testant les endpoints pour garantir qu'ils répondent correctement aux requêtes et produisent les réponses attendues, en utilisant **Postman**. Ces tests ont inclus la vérification des statuts de réponse, du contenu des réponses, des performances, ainsi que des comportements spécifiques à différents scénarios d'utilisation.

```

1 [
2   {
3     "idCarte": 2,
4     "compte": {
5       "idCompte": 2,
6       "numeroCompte": "000987L50987654",
7       "solde": 100000.00,
8       "typeCompte": "épargne",
9       "client": {
10         ...
11       },
12       "agence": {
13         ...
14       },
15       "dateOuverture": "2024-04-18T16:36:39.439+00:00",
16       "tauxInterets": 0.10,
17       "statutCompte": "actif"
18     }
19   }
20 ]
  
```

FIGURE 99 – Test fonctionnel de l'API Bancaire

(c) **Documentation :** La documentation est générée automatiquement en utilisant l'URL spécifiée dans la configuration Swagger, <http://localhost:8080/swagger-ui/index.html>. Cette documentation interactive offre une vue claire et détaillée des endpoints de l'API, de leurs paramètres et réponses associés, ainsi que des exemples d'utilisation. Elle permet de comprendre rapidement le fonctionnement de l'API, d'explorer ses fonctionnalités , de les tester et de démarrer facilement son intégration dans d'autres applications.



The screenshot shows the Swagger UI for the "Documentation de l'API Bancaire". At the top, there's a navigation bar with "Swagger", "api-doc", and a "Explore" button. Below it, the title "Documentation de l'API Bancaire 1.0 OAS3" is displayed. A brief description follows: "API pour la simulation du projet de développement de reconnaissance vocale pour l'accessibilité des personnes non-voyantes dans l'application eBanking d'Attijariwafa Bank". Under "Terms of service", links to "Attijariwafa - Website", "Attijariwafa License Version 1.0", and "Autres liens" are provided. The main content area is divided into sections: "Authentification", "Plafond des cartes bancaires", and "Utilisateur". Each section contains one or more API endpoints with their methods (e.g., POST, PUT) and descriptions.

FIGURE 100 – Documentation de l'API Bancaire

6.2.2 API NLP

L'API NLP est conçue pour comprendre et traiter les demandes bancaires textuelles des utilisateurs. Les demandes, initialement exprimées par la voix via l'application mobile, sont transcrites en texte grâce à un plugin de reconnaissance vocale. Ensuite, l'API NLP prend en charge l'exécution des demandes en collaboration avec l'API bancaire et renvoie les résultats à l'application mobile.

1. Composants de l'architecture

(a) **Composants matériels :** La machine physique, exécutera le serveur Unicorn, qui traitera les requêtes HTTP et exécutera les fonctions Python nécessaires. Cette machine stockera également les données temporaires des utilisateurs et gérera l'état de l'application. Les composants matériels incluent le CPU pour le traitement, la mémoire RAM pour le stockage des données temporaires, et le disque dur pour tout stockage nécessaire.

(b) Composants logiciels :

L'architecture de l'API NLP utilise Unicorn et FastAPI pour exploiter les modèles NLP afin de gérer les conversations et les commandes bancaires des utilisateurs. **Unicorn**, un serveur HTTP asynchrone, reçoit les requêtes HTTP des utilisateurs, tandis que **FastAPI** les traite, prédit le contexte et les entités nommées à l'aide de modèle de reconnaissance d'intention que nous avons affiné pour nos besoins et de modèle NER de Hugging Face. À la base de ces prédictions, FastAPI interroge l'API bancaire pour traiter les demandes des utilisateurs, puis elle compile les résultats et renvoie des réponses rapides et contextuelles. Unicorn et FastAPI sont choisis pour leurs performances élevées, leur support des requêtes asynchrones et leur intégration facile avec les modèles PyTorch.

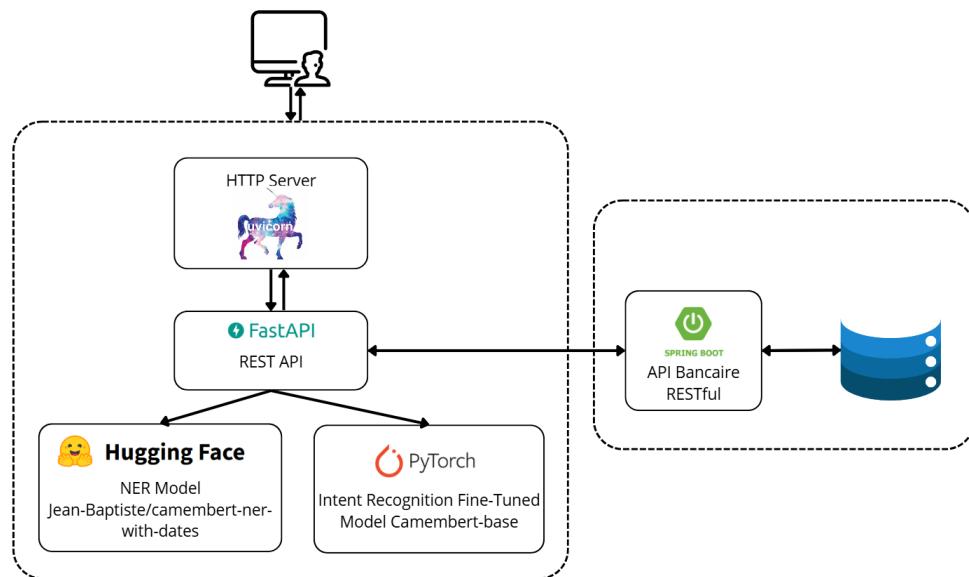


FIGURE 101 – Architecture logicielle de l'API NLP

2. Flux de données

- Réception des requêtes** : Le serveur HTTP asynchrone Uvicorn reçoit les requêtes HTTP des utilisateurs.
- Traitement initial des requêtes** : Les requêtes sont transmises à FastAPI pour le traitement ultérieur.
- Analyse du contexte utilisateur** : FastAPI vérifie le contexte de la requête et, si nécessaire, pose des questions de sécurité pour valider l'identité de l'utilisateur. La réponse de l'utilisateur est ensuite vérifiée à l'aide de l'API bancaire.
- Prédiction contextuelle** : FastAPI utilise notre modèle de reconnaissance d'intention pour extraire le contexte et prédire l'intention de l'utilisateur.
- Gestion des actions** : En fonction de la prédiction, le système peut exécuter une action, annuler une action précédente, demander des clarifications ou modifier le contexte.
- Extraction des entités** : L'API extrait les entités à l'aide du modèle NER et des expressions régulières en fonction de l'intention de l'utilisateur.
- Exécution de l'action** : Une fois toutes les informations nécessaires recueillies, l'action est exécutée, ce qui peut impliquer une interrogation ou une mise à jour de la base de données.
- Gestion des données manquantes** : Si des données sont manquantes, l'API NLP demande à l'utilisateur de les fournir.
- Assistance et FAQ** : En cas de demande de FAQ, FastAPI déclenche le pipeline FAQ pour fournir des réponses appropriées.
- Envoi de la réponse** : Le résultat final est renvoyé à l'utilisateur après un éventuel nettoyage périodique des données et du contexte pour maintenir la

confidentialité et l'intégrité des informations.

3. Structure du projet

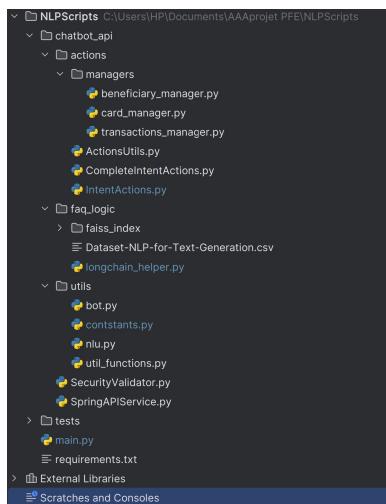


FIGURE 102 – Structure logicielle de l'API NLP

Pour la structure de l'API, et comme montre la figure ci-dessus, l'architecture contient les dossiers et fichiers suivants :

- **Dossier *tests*** : Ce dossier contient les différents tests des modules du projet.
- **La fonction *main.py*** : représente le point d'accès principal à l'API. Elle gère les différentes requêtes selon un contexte global. C'est là que les requêtes des utilisateurs sont reçues et dirigées vers les composants appropriés pour traitement.
- **Dossier *chatbot_api*** : ce dossier constitue le cœur du chatbot, qui est responsable du traitement et de la compréhension des commandes des utilisateurs. Il est subdivisé en sous-dossiers pour une meilleure organisation :
 - **Dossier *faq_logic*** : Contient l'implémentation de la pipeline question-réponse expliquée dans le chapitre 5. Le fichier *langchain_helper* charge les données à partir du fichier CSV présent aussi dans le dossier, les transforme en une base de données de vecteurs pour la recherche de réponses similaires, et utilise Gemini pour générer des réponses cohérentes basées sur le contexte et la question posée.
 - **Dossier *utils*** : regroupe des fichiers et des classes utilitaires. "*nlu.py*" utilise nos modèles NLP pour déterminer les intentions des utilisateurs et extraire les entités nommées. "*bot.py*" contrôle les requêtes utilisateur en déclenchant les actions appropriées. "*util_functions*" propose des fonctions pour formater les dates, filtrer les opérations bancaires et générer des messages informatifs. "*constants.py*" configure l'API avec des variables d'environnement et des réponses prédéfinies. En outre, d'autres classes sont incluses pour interagir avec l'API bancaire, valider les ré-

ponces aux questions de sécurité, et interroger et manipuler les données des utilisateurs.

- **Dossier *actions*** : contient des classes pour le traitement des actions bancaires. Il inclut la classe "***IntentActions.py***" qui gère diverses intentions telles que la localisation d'agences, la consultation de soldes et la gestion de cartes. La classe "***CompleteIntentActions.py***" finalise les actions bancaires, notamment celles nécessitant une confirmation de l'utilisateur. Le dossier "***managers***" propose des classes modulaires pour simplifier l'interaction avec l'API bancaire pour des intentions spécifiques.

4. Tests et documentation

- (a) **Tests unitaires et d'intégration** : Pour les tests unitaires de tous les composants de l'API FastAPI, nous avons utilisé **unittest** et **MagicMock** pour isoler et vérifier le comportement des différentes parties de l'application. En complément, des tests d'intégration avec **pytest** ont été mis en place pour s'assurer du bon fonctionnement et de l'interaction correcte entre les différents composants de l'API. L'ensemble de ces tests est regroupé dans le dossier *tests* du projet.

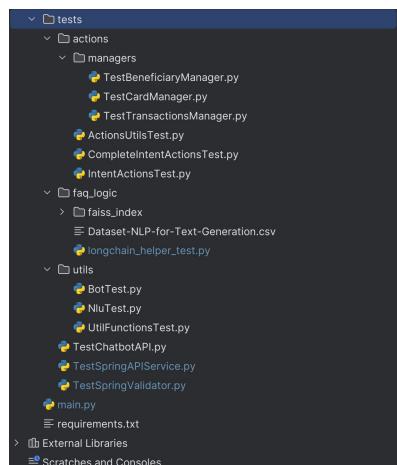


FIGURE 103 – Tests unitaires et d'intégrations de l'API NLP

- (b) **Tests Fonctionnels** : Nous avons ensuite vérifié que les fonctionnalités de l'API répondent correctement aux requêtes et fournissent les réponses attendues à l'aide de Postman. Les tests incluaient des vérifications des statuts de réponse, des contenus de réponse, des performances et des comportements spécifiques à différents scénarios d'utilisation.

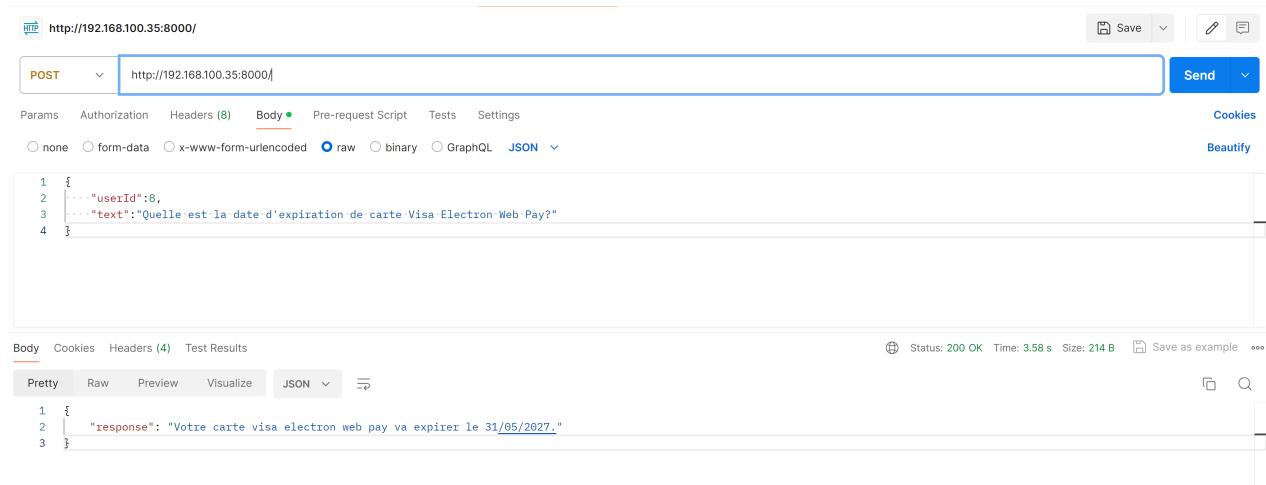
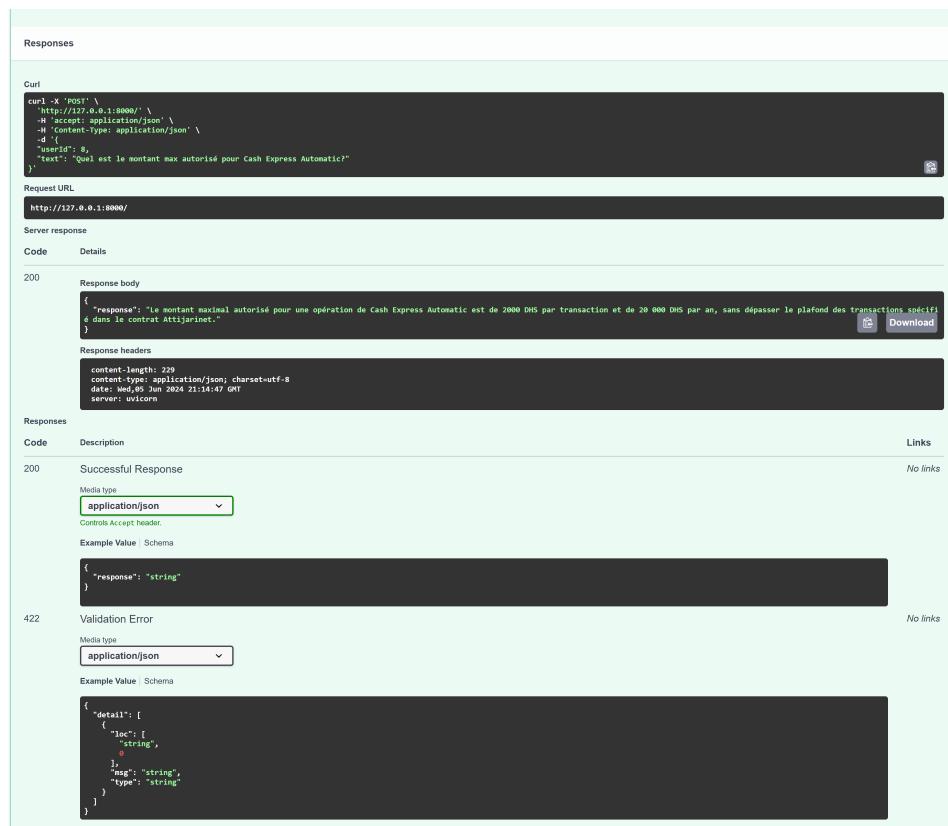


FIGURE 104 – Test fonctionnel de l’API NLP

- (c) **Documentation** : La documentation de l’API FastAPI est accessible via l’interface interactive Swagger UI, qui peut être consultée en naviguant vers <http://127.0.0.1:8000/docs> dans un navigateur web. Cette interface conviviale offre une vue complète de l’endpoint de l’API, de ses paramètres, ainsi que de ses schémas de requête et de réponse. Elle permet également d’effectuer des requêtes directement depuis le navigateur, en fournissant des exemples de données et en affichant les réponses en temps réel. Grâce à cette documentation interactive, nous pouvons facilement explorer et tester les fonctionnalités de l’API sans avoir besoin d’utiliser des outils externes.



The screenshot shows the Swagger UI interface for testing an API. It displays two main sections: 'Responses' and 'Validation Error'.

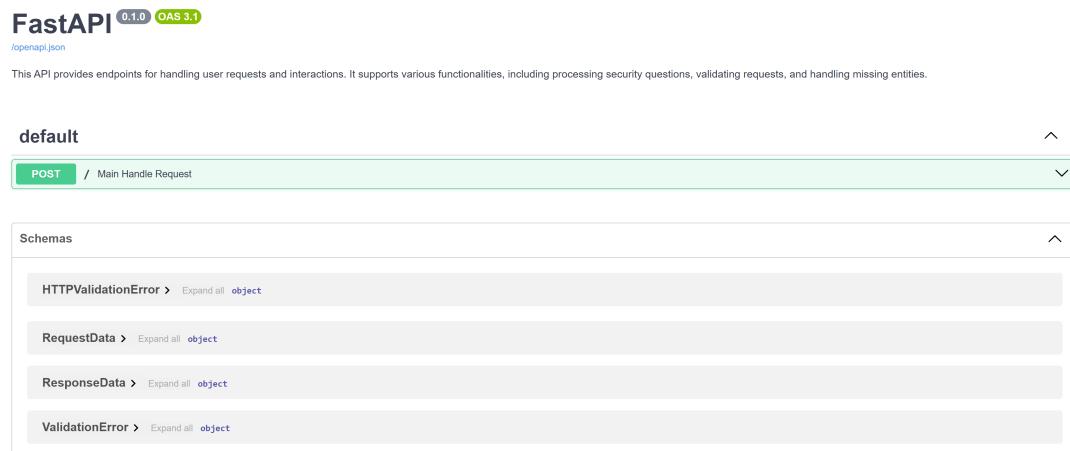
Responses:

- Curl:** A command-line tool example for sending a POST request to `http://127.0.0.1:8000/` with JSON content type and a specific payload.
- Request URL:** `http://127.0.0.1:8000/`
- Server response:**
 - Code:** 200
 - Response body:** A JSON object with a single key 'response' containing the string 'Le montant maximal autorisé pour une opération de Cash Express Automatic est de 2000 DHS par transaction et de 20 000 DHS par an, sans dépasser le plafond des transactions spécifiées dans le contrat Attijariwafa.'
 - Download:** A button to download the response body.
- Response headers:** Headers include content-length: 229, content-type: application/json; charset=utf-8, date: Wed, 05 Jan 2023 21:30:47 GMT, and server: uvicorn.

Validation Error:

- Code:** 422
- Media type:** application/json
- Example Value:** A JSON object with a 'detail' field containing an array of validation errors, each with a 'loc' field (containing 'string') and a 'msg' field (containing 'string').
- Schema:** A JSON schema for the validation error object.
- No links:** A link section indicating no links are available.

(a) Test de l'API à l'aide de l'interface de Swagger



The screenshot shows the FastAPI documentation interface. It features a top navigation bar with the title 'FastAPI 0.1.0 OAS 3.1 /openapi.json'. Below the title, a brief description states: 'This API provides endpoints for handling user requests and interactions. It supports various functionalities, including processing security questions, validating requests, and handling missing entities.'

default

POST / Main Handle Request

Schemas:

- HTTPValidationError > Expand all object
- RequestData > Expand all object
- ResponseData > Expand all object
- ValidationError > Expand all object

(b) Interface de la documentation Swagger

FIGURE 105 – Documentation de l'API NLP à l'aide de Swagger

6.2.3 API de la biométrie vocale

Cette API est conçue pour fournir à l'API bancaire les niveaux de similitude entre les caractéristiques vocales de divers utilisateurs dans la base de données et celles de l'utilisateur cherchant à s'authentifier. Ceci permet d'identifier l'identité de l'utilisateur

en se basant sur le modèle siamois pré-entraîné.

1. Composants de l'architecture

- (a) **Composants physique** : Tout comme pour les autres API, cette API nécessite une machine physique qui exécutera le serveur Unicorn. Ce serveur traitera les requêtes HTTP et exécutera les fonctions Python requises. En outre, cette machine supervisera l'état global de l'application. Les composants matériels essentiels comprennent le CPU pour le traitement des opérations, la mémoire RAM pour stocker les données temporaires, et le disque dur pour toute forme de stockage permanente nécessaire.
- (b) **Composants logiciels** : Tout comme l'API NLP, cette API utilise le serveur HTTP Unicorn pour gérer les requêtes et exécuter l'API Python, grâce à sa faible latence et ses performances élevées. FastAPI est utilisé pour développer l'API REST qui exploite le modèle siamois pré-entraîné avec PyTorch, en raison de sa simplicité, de sa rapidité de développement et de ses performances exceptionnelles.

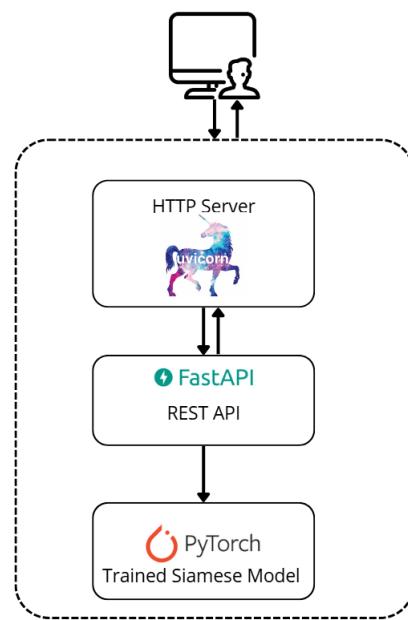


FIGURE 106 – Architecture logicielle de l'API de la biométrie vocale

2. Flux de données

- (a) **Requête de l'utilisateur** : Un utilisateur envoie une requête POST au serveur HTTP. La requête contient les caractéristiques de voix d'un utilisateur envoyées par l'API bancaire et un fichier audio de l'utilisateur qui veut s'authentifier.
- (b) **Serveur HTTP (Uvicorn)** :
Le serveur HTTP Unicorn reçoit la requête et la transmet à l'API REST FastAPI.

(c) API REST (FastAPI) :

FastAPI gère la logique de l'API, y compris la validation des données d'entrée (caractéristiques de voix et fichier audio).

(d) Traitement du modèle (PyTorch) :

- FastAPI passe les données (caractéristiques de voix de l'utilisateur et le fichier audio de notre utilisateur) au modèle siamois pré-entraîné avec PyTorch.
- Le modèle siamois traite le fichier audio de l'utilisateur pour extraire les caractéristiques de sa voix.

(e) Comparaison des caractéristiques : Une fois les caractéristiques de la voix de l'utilisateur extraites, on compare ces caractéristiques avec celles de l'utilisateur envoyées par l'API bancaire en utilisant la similarité cosinus.

(f) Détermination de l'identité : En fonction du résultat de la comparaison, l'API détermine si l'utilisateur est la même personne que celle identifiée par l'API bancaire ou s'il s'agit d'une autre personne. Cette information est ensuite envoyée en réponse à la requête initiale de l'utilisateur.

3. Structure du projet

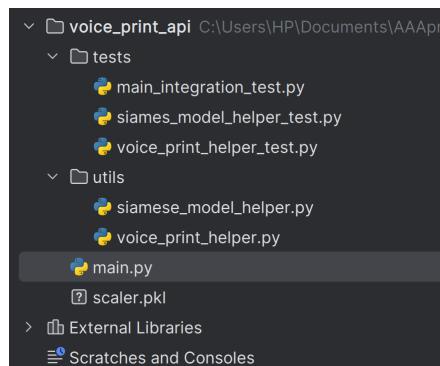


FIGURE 107 – Structure du projet de l'API de la biométrie vocale

L'implémentation de cette api contient les dossiers et fichiers suivants :

- Fichier *main.py* :** le fonction principale qui sert de point d'entrée de notre api et qui gère les requêtes HTTP.
- Dossier *tests* :** contient les tests des différents composants de l'API.
- Dossier *utils* :** contient des fichiers essentiels pour le fonctionnement de l'API, notamment la définition du réseau siamois avec PyTorch, des fonctions de prétraitement audio et de prédiction.

4. Tests et documentation

- Tests unitaires et d'intégration :** Nous avons utilisé **pytest** pour effectuer les tests unitaires de chaque composant de l'API en les isolant du reste du code, ainsi que des tests d'intégration pour vérifier l'interaction entre ces composants.

- (b) **Tests fonctionnels** : Comme pour toutes les API, nous avons testé les endpoints avec Postman pour nous assurer de leur bon fonctionnement.

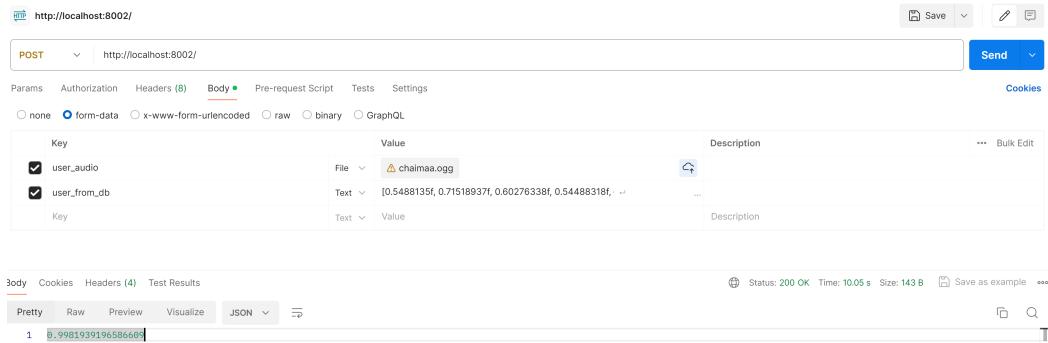


FIGURE 108 – Test fonctionnel de l’API de biométrie vocale

- (c) **Documentation** : La documentation de l’API FastAPI est accessible via l’interface interactive Swagger UI, à travers le lien suivant `http://127.0.0.1:8002/docs` dans un navigateur web.

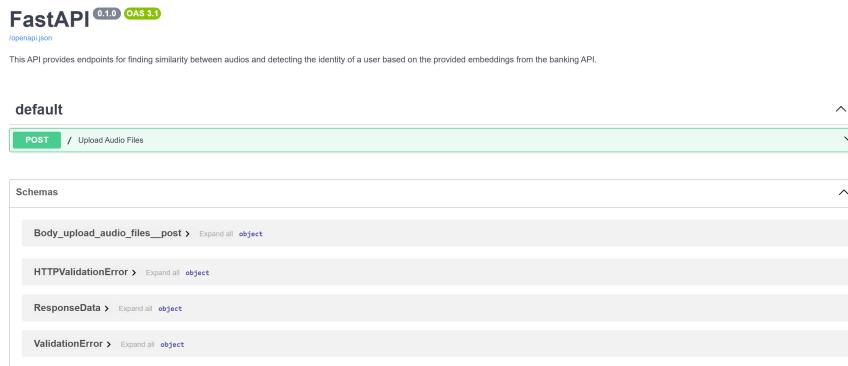


FIGURE 109 – Documentation de l’API de biométrie vocale

6.3 Développement de l’application mobile

6.3.1 Architecture physique de l’application

L’architecture physique utilise les composants suivants :

- Machine physique** : Pour développer avec **Android Studio** et **Flutter**, une machine puissante est nécessaire, avec un processeur multicœur rapide, une grande quantité de RAM et un espace de stockage suffisant. Ces outils sont préférés car ils sont officiellement supportés par Google pour le développement Android, offrant une intégration transparente avec les fonctionnalités natives et une efficacité de développement grâce à Dart.
- Mobile Device Simulator** : Android Studio et Flutter fournissent des émulateurs et des outils de simulation intégrés pour tester et déboguer des applications sur une gamme de périphériques virtuels ou bien utiliser son propre téléphone portable.

3. Communication réseau : L'application mobile dépend de la communication réseau pour accéder à Internet, synchroniser les données et permettre des interactions en ligne. Android Studio et Flutter offrent une variété d'options pour gérer la communication réseau.

6.3.2 Architecture logicielle de l'application

L'architecture logicielle du projet adopte une conception bien structurée qui assure l'organisation efficace de l'ensemble de l'application. Pour y parvenir, l'architecture MVC (Modèle-Vue-Contrôleur) avec services de données externes pour un modèle unique a été adoptée. Cette approche permet de séparer différentes responsabilités et assure une gestion cohérente des différents composants de l'application.

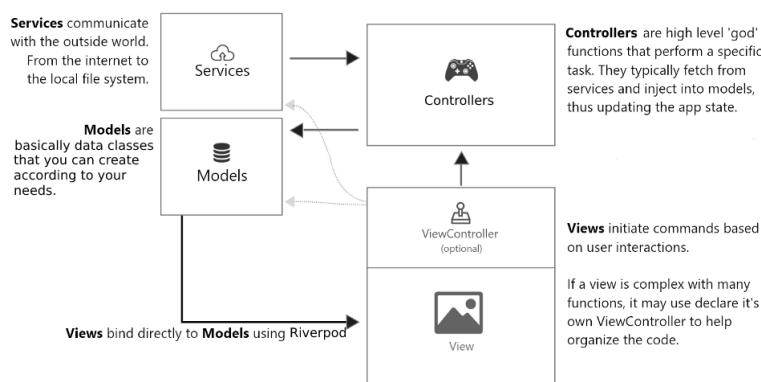


FIGURE 110 – Architecture logicielle de l'application mobile

L'architecture choisie divise l'application en trois couches principales :

- Modèle** : Cette couche gère les données et la logique métier. Les modèles représentent les structures de données utilisées dans l'application, principalement les utilisateurs. Toutes les opérations impliquant la manipulation des données et les règles métier sont gérées dans cette couche.
- Vue** : La couche Vue est responsable de l'interface utilisateur. Elle affiche les données aux utilisateurs et capture leurs interactions. Les éléments visuels et sonores, tels que les widgets et les composants, sont conçus et organisés ici pour garantir une expérience utilisateur fluide et intuitive.
- Contrôleur** : Cette couche agit comme un intermédiaire entre le Modèle et la Vue. Les contrôleurs gèrent les interactions utilisateur, traitent les demandes, mettent à jour le modèle en conséquence et rafraîchissent la vue pour refléter les changements de manière transparente.
- Services** : Cette partie supplémentaire de l'architecture est responsable de la logique métier qui n'est pas spécifiquement liée à un modèle ou à une vue particulière. Les services fournissent des fonctionnalités transversales à l'ensemble de l'application, telles que l'authentification des utilisateurs, la gestion des sessions et la com-

munication avec l'API NLP. Ils sont conçus pour être réutilisables et indépendants de toute interface utilisateur spécifique.

6.3.3 Implémentation

La mise en œuvre de l'architecture en couches est minutieusement structurée pour garantir une organisation cohérente des différents composants du projet, favorisant la clarté, la maintenabilité et des processus de développement efficaces. La figure suivante met en évidence comment chaque composant contribue à l'architecture globale :

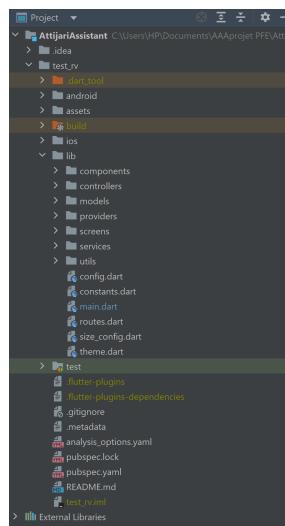


FIGURE 111 – Struture du projet de l'application mobile

1. **Dossier de ressources** : Le dossier *assets > audios/icons/images* contient les ressources statiques de l'application, comme les fichiers audio, les icônes, et les images utilisés dans l'interface utilisateur.
2. **Dossiers de code source** :
 - **lib > components** : Contient les éléments d'interface utilisateur réutilisables, tels que des boutons, des barres personnalisées et des widgets de description de données.
 - **lib > controllers** : Héberge les classes qui gèrent la logique de l'application et contrôlent l'interaction entre les vues et les modèles. Ils reçoivent les requêtes des utilisateurs, les traitent, récupèrent les données nécessaires auprès des services et mettent à jour les vues en conséquence.
 - **lib > models** : Contient le modèle unique de l'application, en l'occurrence la classe représentant les utilisateurs.
 - **lib > services** : Regroupe les services utilisés pour effectuer des opérations spécifiques, tels que l'authentification, les appels API et la communication avec l'API NLP.
 - **lib > providers** : Contient les classes de providers pour gérer l'état global de l'application, en fournissant des instances de services aux contrôleurs,

- **lib > screens** : Contient les différentes interfaces de l'application, comme l'écran de connexion, l'écran d'accueil, ou l'écran de commandes vocales.
- **lib > utils** : Héberge les utilitaires et les fonctions d'aide qui peuvent être utilisés dans différentes parties de l'application pour des tâches courantes, comme des codes utils pour la synthèse vocale et la conversion de parole en texte.

3. Fichiers de configuration :

- **config.dart** : Contient les configurations générales de l'application, comme le logo de l'application et d'autres paramètres de configuration globaux.
- **constants.dart** : Regroupe les constantes utilisées dans l'application, comme les couleurs, les styles de texte, les marges et les chaînes de caractères statiques.
- **main.dart** : Le point d'entrée de l'application, où l'application est initialisée et la fonction `runApp` est appelée pour démarrer l'application Flutter.
- **routes.dart** : Définit les routes de navigation de l'application, associant les noms de routes aux widgets correspondants pour une navigation structurée.
- **size_config.dart** : Contient des utilitaires pour gérer les tailles d'écran et les dimensions de manière réactive, permettant de créer une interface utilisateur adaptative.
- **theme.dart** : Définit le thème global de l'application, incluant les couleurs, les polices, et les styles de widgets par défaut.

4. Tests :

Le dossier **test** contient les tests unitaires et d'intégration pour vérifier le bon fonctionnement de différentes parties de l'application, assurant ainsi la qualité et la fiabilité du code.

5. Fichiers de configuration du projet :

- **pubspec.yaml/pubspec.lock** : Le fichier `pubspec.yaml` définit les dépendances du projet, les assets, et les configurations de l'application Flutter. Le fichier `pubspec.lock` enregistre les versions spécifiques des dépendances installées.

6.3.4 Tests

1. Test d'unitaires et d'intégration :

Les tests de l'application mobile ont été menés à bien grâce aux packages **mocktail** et **flutter_test**, choisis pour leur simplicité, leur efficacité et leur compatibilité avec Flutter. Cette approche exhaustive a permis de tester minutieusement tous les aspects de l'application, des écrans aux providers en passant par les services. Les **tests unitaires** ont évalué le bon fonctionnement des classes individuelles, tandis que les **tests d'intégration** ont validé l'interaction harmonieuse entre les différents modules. Enfin, un **test final d'intégration** a confirmé la cohérence et la fluidité de l'expérience utilisateur. Cette démarche proactive a permis de détecter et de résoudre les problèmes dès les premières étapes

du développement, garantissant ainsi la qualité et la stabilité de l'application finale.

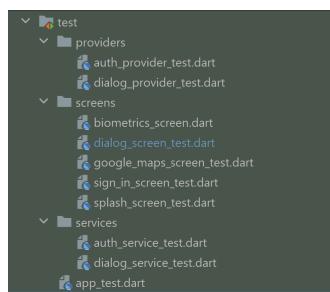


FIGURE 112 – Implémentation des tests de l'application mobile

2. **Test de simulation :** Le test de simulation avec le téléphone a enrichi le processus de test en permettant une validation approfondie du comportement de l'application dans un environnement réel. En exécutant l'application sur notre propre appareil, nous avons évalué des aspects spécifiques tels que l'interface utilisateur, les performances et la réactivité, impossibles à évaluer pleinement dans un environnement de test simulé. Cette approche a renforcé notre confiance dans la qualité et la stabilité de l'application en détectant et résolvant les problèmes potentiels qui pourraient survenir lors d'une utilisation quotidienne par les utilisateurs.

6.4 Intégrations

Ce passage détaille l'intégration entre les divers composants du système. Chaque sous-section expose minutieusement le processus d'intégration entre deux composants spécifiques, mettant en lumière les méthodes utilisées et les fonctionnalités impliquées dans chaque interaction.

6.4.1 Intégration de l'API bancaire avec l'API d'authentification

L'intégration entre l'API bancaire et l'API d'authentification a été mise en place à travers l'implémentation du service d'interface **Utilisateur** de l'API bancaire RESTful Spring Boot. Ce service permet de transmettre l'audio de l'utilisateur désirant s'authentifier, ainsi que les caractéristiques vocales d'un utilisateur sélectionné dans la base de données. Pour ce faire, le service parcourt l'ensemble des caractéristiques vocales de tous les utilisateurs enregistrés dans la base de données à l'aide d'une boucle for.

6.4.2 Intégration de l'API bancaire avec l'application mobile

L'intégration entre l'API bancaire et l'application mobile a été réalisée via le service **VoiceBiometricService**, spécifiquement défini dans l'application mobile. Ce service utilise l'API bancaire pour transmettre l'audio de l'utilisateur, enregistré lors de la répétition de la phrase "Voice Assistant", à l'API d'authentification. L'audio est ensuite traité

à l'aide de notre modèle siamois pour extraire les caractéristiques vocales de l'utilisateur souhaitant s'authentifier, puis les comparer aux caractéristiques stockées dans la base de données.

6.4.3 Intégration de l'API bancaire avec l'API NLP

L'API NLP communique avec l'API Spring Boot en fonction des intentions des utilisateurs via la classe **IntentActions** et les classes du dossier **managers** grâce à des requêtes HTTP de type PUT, POST, DELETE et GET. Ces requêtes impliquent souvent des données utilisateur et servent à interroger, manipuler et mettre à jour les données lors de différentes opérations telles que la gestion de bénéficiaires, les transactions de paiement de factures, les virements, et autres.

6.4.4 Intégration de l'API NLP avec l'application mobile

L'intégration entre l'API NLP et l'application mobile a été réalisée via le service **DialogService**, défini dans l'application mobile. Ce service fait appel à l'API NLP pour traiter les requêtes vocales des utilisateurs, lesquelles sont préalablement converties en texte grâce au plugin speech_to_text.

6.5 Conclusion

Ce chapitre a couvert deux aspects majeurs : le développement des APIs et celui de l'application mobile. Pour les APIs, nous avons examiné en détail l'architecture comprenant les composants logiciels et matériels, le flux de données, les tests et la documentation de chaque API, notamment l'API bancaire de simulation, l'API NLP et l'API d'authentification. Concernant le développement mobile, nous avons défini l'architecture, tant sur le plan physique que logiciel, et nous avons fourni les détails d'implémentation et de test. En somme, ce chapitre a mis en lumière les étapes cruciales du développement de la partie logicielle du projet.

Chapitre VII : Démonstration et Défis

7 Démonstration et Défis

7.1 Introduction

Ce chapitre se concentre sur la démonstration pratique de notre solution et les défis rencontrés au cours de sa mise en oeuvre. Nous présenterons d'abord une démonstration finale du système, illustrant les principales fonctionnalités et les interfaces utilisateur. Ensuite, nous discuterons des défis techniques et méthodologiques rencontrés, ainsi que des perspectives futures pour améliorer et étendre notre travail. Cette section vise à fournir une vue d'ensemble complète des performances de notre solution tout en mettant en lumière les obstacles surmontés et les pistes d'amélioration envisagées.

7.2 Démonstration Finale

7.2.1 Interface d'accueil immersive

L'interface d'accueil comprend un *splash screen* engageant qui immerge l'utilisateur dans une expérience multisensorielle. Une musique entraînante en arrière-plan dynamise l'expérience, tandis que l'assistant vocal se lance pour saluer l'utilisateur et indiquer que l'application est prête à être utilisée. Après le *splash screen*, l'utilisateur est automatiquement redirigé vers l'écran d'authentification.



FIGURE 113 – Interface d'accueil de l'assistant vocal

7.2.2 Interface d'Authentification

Une fois redirigé vers l'interface d'authentification, l'assistant vocal demande à l'utilisateur d'enregistrer sa voix en appuyant n'importe où sur l'écran, puis de réappuyer pour arrêter l'enregistrement. Après quelques secondes, l'assistant informe l'utilisateur de l'état

de l'authentification. Si l'authentification est réussie, l'utilisateur est redirigé vers l'écran des commandes ; sinon, un message d'erreur lui est communiqué.

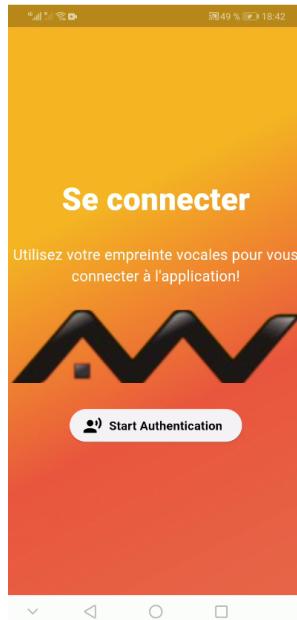


FIGURE 114 – Interface d'authentification de l'assistant vocal

7.2.3 Interface principale de commandes

L'assistant vocal informe l'utilisateur qu'il peut appuyer n'importe où sur l'écran pour énoncer les commandes bancaires souhaitées. La reconnaissance vocale transcrit la commande en temps réel et l'affiche à l'écran en quelques secondes. L'assistant vocal répond alors aux besoins de l'utilisateur en matière d'e-banking, avec le texte de réponse également affiché à l'écran. Cela permet aux utilisateurs très mal-voyants de recevoir à la fois une réponse auditive et visuelle.



FIGURE 115 – Interface principale de commandes de l'assistant vocal

7.2.4 Interface Google Maps

L'interface Google Maps permet à l'utilisateur de démarrer un itinéraire vers les agences bancaires les plus proches, récupérées par l'assistant vocal si l'utilisateur en a besoin.



FIGURE 116 – Interface Google Maps

7.3 Défis et perspectives

7.3.1 Défis rencontrés

1. **Sécurité et Confidentialité** : Trouver un équilibre entre sécurité et facilité d'utilisation, tout en garantissant la précision de la reconnaissance vocale pour des mesures de sécurité fiables.
2. **Collecte et Qualité des Données** : La création d'un corpus de données bancaires adapté est laborieuse, tout comme l'assurance de la qualité des données collectées pour garantir des performances élevées des modèles NLP.
3. **Développement des Modèles NLP** : Trouver des modèles performants tout en évitant l'overfitting et en assurant la reconnaissance des entités nommées représente un défi, surtout en raison des besoins en terme de données inexistantes et de ressources matérielles nécessaires pour l'entraînement.
4. **Accessibilité et Expérience Utilisateur** : Garantir une accessibilité complète pour les non-voyants tout en gérant les limitations des permissions d'accès au Micro et à Google Maps, en assurant une faible latence et une reconnaissance vocale précise sont des défis importants à relever.
5. **Maintenance et Évolutivité** : Concevoir un système évolutif et maintenable pour l'ajout de nouvelles fonctionnalités, impliquant un fine-tuning des modèles et un contrôle rigoureux de l'API NLP.

7.3.2 Perspectives

1. **Amélioration du Modèle de Biométrie Vocale** : Continuer à développer et à affiner le modèle de biométrie vocale pour améliorer la précision et la fiabilité de la reconnaissance des utilisateurs, en explorant différentes architectures de modèles.
2. **Personnalisation du Modèle de Reconnaissance Vocale** : Développer un modèle de reconnaissance vocale personnalisé à l'aide des enregistrements de conversations historiques générées après le déploiement, pour améliorer la précision de la compréhension des commandes vocales.
3. **Ajout d'Autres Langues, y Compris l'Arabe** : Élargir le support linguistique de l'application en ajoutant d'autres langues, notamment l'arabe, pour permettre à un plus grand nombre d'utilisateurs de bénéficier des fonctionnalités de l'application.
4. **Personnalisation du Modèle d'Extraction d'Entités Nommées (NER)** : Développer un modèle NER personnalisé pour reconnaître et extraire des entités spécifiques au domaine bancaire, en tenant compte des particularités linguistiques et des besoins spécifiques des utilisateurs.
5. **Ajout de Nouvelles Fonctionnalités** : Intégrer de nouvelles fonctionnalités pour enrichir l'expérience utilisateur, telles que des fonctionnalités de recommandation personnalisée basées sur les habitudes de dépenses, des alertes et notifications en

temps réel pour les transactions importantes, ainsi que d'autres services bancaires innovants.

6. **Optimisation de la Performance :** Poursuivre l'optimisation de la performance du système IA à l'aide des données générées par la première version de l'application pour garantir des réponses précises, rapides et une faible latence.
7. **Expérimentation avec de Nouvelles Technologies :** Explorer de nouvelles technologies émergentes telles que l'intelligence artificielle conversationnelle avancée et l'apprentissage par renforcement pour enrichir les fonctionnalités de l'application et offrir des services bancaires plus innovants et sécurisés.

7.4 Conclusion

Ce chapitre a mis en lumière les fonctionnalités de notre solution tout en mettant en évidence les défis techniques, tels que la sécurité, la qualité des données, le développement des modèles NLP, l'accessibilité et la maintenance. La démonstration finale a également souligné l'importance de l'expérience utilisateur, de l'accueil à l'utilisation des services bancaires. Pour l'avenir, nous envisageons d'améliorer les modèles de reconnaissance vocale, d'ajouter de nouvelles langues et de développer des fonctionnalités personnalisées. Ces initiatives visent à rendre notre application plus performante, accessible et utile pour nos utilisateurs.

Conclusion générale

En guise de conclusion, ce rapport synthétise les efforts et les réalisations de mon projet de stage effectué au sein du groupe Attijariwafa Bank (AWB), au sein des équipes du Digital Center du pôle Systèmes d'Information Groupe, partie intégrante du Pôle Transformation, Innovation, Technologies et Opérations. Le développement d'un système de reconnaissance vocale pour l'accessibilité des personnes non-voyantes dans l'application e-banking d'Attijariwafa Bank marque une avancée majeure vers l'inclusion numérique et l'amélioration de l'expérience utilisateur pour les personnes ayant des besoins spécifiques. En intégrant une technologie de pointe, nous avons conçu une solution qui allie innovation, sécurité et efficacité, répondant ainsi aux exigences complexes d'un environnement bancaire moderne.

Ce projet a été jalonné de défis variés, de la spécification des besoins à la mise en œuvre technique. Après une analyse minutieuse de l'état de l'art et une planification méticuleuse, nous avons pris la décision stratégique de développer une application mobile accessible pour les non-voyants, offrant ainsi une autonomie accrue dans la gestion de leurs finances et promouvant l'inclusion numérique. Les choix technologiques, tels que l'utilisation du framework Flutter pour la conversion speech to text et la synthèse vocale, ont été guidés par la nécessité d'offrir une solution précise et sécurisée dans des délais serrés.

Dans le domaine du NLP, notre sélection du modèle CamemBERT fine-tuné s'est avérée pertinente, démontrant les meilleures performances en reconnaissance des intentions grâce à notre étude comparative sur différents algorithmes et modèles NLP. En associant ce modèle avec des expressions régulières personnalisées pour l'extraction d'entités nommées, nous avons obtenu des résultats solides. La mise en place d'une pipeline FAQ, utilisant les embeddings de questions, a également amélioré la précision et la capacité à répondre aux différentes reformulations de questions. La combinaison de la biométrie vocale avec des questions de sécurité génériques pour l'authentification a renforcé la sécurité tout en préservant la convivialité de l'application. De même, notre approche pour le contrôle du flux conversationnel, basée sur la programmation procédurale et l'utilisation de Python et FastAPI, a facilité la gestion efficace des interactions complexes.

La mise en œuvre de ce projet a présenté plusieurs défis. Parmi les obstacles surmontés, nous avons dû trouver un équilibre entre sécurité et facilité d'utilisation, assurer la qualité des données collectées, et développer des modèles NLP performants tout en garantissant la généralisation. La garantie d'une accessibilité complète pour les non-voyants et la maintenance du système ont également représenté des défis significatifs.

Pour l'avenir, notre projet ouvre la voie à de nombreuses perspectives d'amélioration, telles que l'optimisation continue du modèle de biométrie vocale, l'ajout de nouvelles

langues pour une portée plus large, et la personnalisation du modèle NER pour répondre aux besoins bancaires spécifiques. En continuant à intégrer de nouvelles fonctionnalités et à optimiser les performances de l'application, nous pouvons améliorer encore l'expérience utilisateur et contribuer à l'avancement de l'inclusion numérique dans le secteur bancaire.

En somme, ce projet a démontré la faisabilité et l'efficacité de l'intégration d'un système de reconnaissance vocale dans une application bancaire pour les personnes non-voyantes. La solution développée offre une accessibilité accrue et renforce la sécurité des utilisateurs, tout en ouvrant la voie à des innovations futures dans le domaine des services bancaires numériques. Ce travail illustre l'engagement d'Attijariwafa Bank envers l'inclusion et l'innovation, et pose les bases pour des améliorations continues et des développements futurs dans le domaine de l'accessibilité numérique.

Références

- [1] **Documentation de FastAPI** : Consulté le 01 mars 2024. Disponible à l'adresse :
<https://fastapi.tiangolo.com/>.
- [2] **Documentation Flutter** : Consulté le 11 avril 2024. Disponible à l'adresse :
<https://docs.flutter.dev/>.
- [3] **Documentation Android Studio** : Consulté le 21 avril 2024. Disponible à l'adresse :
<https://developer.android.com/develop?hl=fr>.
- [4] **Documentation PyTorch** : Consulté le 01 mars 2024. Disponible à l'adresse :
<https://pytorch.org/docs/stable/index.html>.
- [5] **Réseau de neurones récurrents** : Consulté le 21 mars 2024. Disponible à l'adresse :
https://fr.wikipedia.org/wiki/R%C3%A9seau_de_neurones_r%C3%A9currents.
- [6] **Transformateur** : Consulté le 21 mars 2024. Disponible à l'adresse : <https://fr.wikipedia.org/wiki/Transformeur>.
- [7] **Attention is All you Need** : Consulté le 21 mars 2024. Disponible à l'adresse :
<https://arxiv.org/abs/1706.03762>.
- [8] **FlauBERT vs. CamemBERT** : Consulté le 21 mars 2024. Disponible à l'adresse :
<https://pubmed.ncbi.nlm.nih.gov/35430035/>.
- [9] **A study of transformer-based end-to-end speech recognition system** : Consulté le 08 avril 2024. Disponible à l'adresse : <https://www.nature.com/articles/s41598-022-12260-y>.
- [10] **Performance en classification de données textuelles des passages aux urgences des modèles BERT** : Consulté le 08 avril 2024. Disponible à l'adresse :
<https://inria.hal.science/hal-03276129/document>.
- [11] **Long Short Term Memory (LSTM)** : Consulté le 08 avril 2024. Disponible à l'adresse :
https://en.wikipedia.org/wiki/Long_short-term_memory.
- [12] **Top 20 Most Powerful Large Language Models For NLP Tasks Transfer Learning In 2024** : Consulté le 11 avril 2024. Disponible à l'adresse :
https://spotintelligence.com/2023/04/18/large-language-models-nlp/#Top_20_best_NLP_models.
- [13] **Siamese Network Keras for Image and Text similarity** : Consulté le 11 avril 2024. Disponible à l'adresse :
<https://medium.com/@prabhnoor0212/siamese-network-keras-31a3a8f37d04>.
- [14] **What You Need to Know About Voice Biometrics** : Consulté le 26 avril 2024. Disponible à l'adresse :
<https://www.aware.com/blog-what-you-need-to-know-about-voice-biometrics/#:~:text=Voice%20biometrics%20is%20the%20science,to%20virtual%20and%20physical%20spaces>.

- [15] **13 Best Free Speech-to-Text Open Source Engines, APIs, and AI Models :** Consulté le 26 avril 2024. Disponible à l'adresse : <https://www.notta.ai/en/blog/speech-to-text-open-source>.
- [16] **The top free Speech-to-Text APIs, AI Models, and Open Source Engines :** Consulté le 26 avril 2024. Disponible à l'adresse : <https://www.assemblyai.com/blog/the-top-free-speech-to-text-apis-and-open-source>
- [17] **Disponible à l'adresse :** Consulté le 26 avril 2024. Disponible à l'adresse : <https://deepgram.com/learn/best-speech-to-text-apis#what-are-the-most-important-apis>
- [18] **Société Générale : DÉPLOIEMENT DE LA DATA ET DE L'INTELLIGENCE ARTIFICIELLE :** Consulté le 01 juin 2024. Disponible à l'adresse : <https://www.societegenerale.com/sites/default/files/documents/2023-05/sg-deploiement-de-la-data-et-de-l-intelligence-artificielle.pdf>.
- [19] **Documentation sklearn :** Consulté le 06 mai 2024. Disponible à l'adresse : <https://scikit-learn.org/stable/>.
- [20] **Documentation Mermaid :** Consulté le 11 mai 2024. Disponible à l'adresse : <https://mermaid.js.org/intro/>.
- [21] **Documentation StarUML :** Consulté le 11 mai 2024. Disponible à l'adresse : <https://docs.staruml.io/>.
- [22] **Documentation des développeurs Meta :** Consulté le 01 juin 2024. Disponible à l'adresse : <https://developers.meta.com/>.
- [23] **Journée mondiale des aveugles et malvoyants :** Consulté le 15 mai 2024. Disponible à l'adresse : <https://www.lepharmacienmanager.com/journee-mondiale-des-aveugles-et-malvoyants#:~:text=Au%20Maroc%2C%20on%20estime%20%C3%A0,200%20000%20personnes%20non-voyantes>
- [24] **Statistiques des aveugles au Maroc selon l'IAPB (The International Agency For The Prevention Of Blindness) :** <https://www.iapb.org/fr/learn/vision-atlas/magnitude-and-projections/countries/morocco/>.
- [25] **Mot du Président :** Consulté le 25 mai 2024. Disponible à l'adresse : <https://www.attijariwafabank.com/fr/notre-groupe-bancaire/mot-du-president#:~:text=CELLES%20sont%20prises%20en,appellation%20%3A%20%C2%AB%20Energies%202020%20%C2%BB>.
- [26] **Rapport Digital 2019 :** Consulté le 05 juin 2024. Disponible à l'adresse : <https://www.attijariwafabank.com/fr/rapport-digital-2019/une-dynamique-digitale->
- [27] **Missions :** Consulté le 10 mai 2024. Disponible à l'adresse : <https://corporate.attijariwafa.net/a-propos-de-nous/missions/>.
- [28] **Gouvernance et Organisation :** Consulté le 20 mai 2024. Disponible à l'adresse : <https://www.attijariwafabank.com/fr/notre-groupe-bancaire/gouvernance-organisation>
- [29] **Activité et Marques du Groupe :** Consulté le 30 mai 2024. Disponible à l'adresse : <https://www.attijariwafabank.com/fr/notre-groupe-bancaire/activite-et-marques-du>

-
- [30] **Plaquette institutionnelle du Groupe :** Consulté le 08 juin 2024. Disponible à l'adresse : https://www.attijariwafabank.com/sites/default/files/plaquette_instit-groupe_awb_vf_2019.pdf.
 - [31] **Organigramme 2020 du Groupe :** Consulté le 12 juin 2024. Disponible à l'adresse : https://www.attijariwafabank.com/sites/default/files/2020-09/18-00942-organigramme_2020_vf3.pdf.