Ministry of Higher Education and Scientific Research

INTERSHIP REPORT

SPECIALIZATION: Mobile Application Development

Prepared By: motaz sammoud

Développement d'une application d'extraction de données pertinentes d'une carte visite

Carried out at

ESPRIT

Supervised by

MR. Salah Bousbia

Remerciements

Je tiens à exprimer ma profonde gratitude à Monsieur Salah Bousbia, mon encadrant à ESPRIT, pour son accompagnement attentif, sa disponibilité et ses conseils précieux tout au long de mon stage. Son encadrement m'a permis de structurer le travail et de mener à bien le développement de notre application mobile d'OCR, du backend NestJS et du module de reporting assisté par l'IA.

Je remercie également l'équipe pédagogique d'ESPRIT pour le cadre de travail fourni ainsi que mes camarades pour leurs tests, leurs retours constructifs et leur soutien tout au long du projet.

Ce stage, ponctué de neuf rapports d'avancement, m'a fait grandir sur les plans technique et professionnel, et m'a permis d'acquérir une expérience concrète du développement mobile et de l'intégration de services IA au sein d'une application métier.

Contenu

Liste des figures

Liste des tableaux

Introduction General

Le domaine de la gestion des contacts et des opportunités commerciales connaît une profonde mutation, portée par la complexification des échanges et le besoin de solutions plus efficaces et plus fiables. Alors que les organisations cherchent à développer leur réseau et convertir les contacts en partenariats concrets, les méthodes traditionnelles, fondées sur des saisies manuelles et des suivis dispersés, montrent leurs limites. Ces frictions ralentissent les flux de travail et impactent la productivité des équipes.

Dans ce contexte, notre application mobile « SCART » apporte une réponse concrète : numériser les cartes de visite via OCR, centraliser les informations extraites, générer des opportunités structurées et automatiser la prise de contact par e-mail. L'outil s'appuie sur un pipeline simple (new, contacted, qualified, won, lost) afin de fluidifier le suivi commercial et la collaboration.

Avant cette solution, la gestion reposait sur des notes éparses, des feuilles de calcul et des communications fragmentées entre intervenants. L'absence de vision unifiée entraînait des retards, des doublons et des décisions tardives, avec un risque réel de perte d'opportunités. La coordination entre parties prenantes restait difficile et le volume d'informations à traiter augmentait.

SCART unifie ces étapes : capture et lecture des cartes, création d'opportunités, mise à jour du statut et envoi d'e-mails contextualisés au contact extrait. Les données sont accessibles en temps réel et centralisées, ce qui accélère les décisions et réduit les tâches administratives. L'interface mobile (Flutter) s'appuie sur un backend NestJS et une base de données MongoDB pour garantir performance et évolutivité.

Un bénéfice clé réside dans l'automatisation des échanges et l'amélioration de la traçabilité : chaque opportunité est historisée, son statut est partagé entre membres de l'équipe et les actions clés (contact, qualification, relance) sont rendues visibles. L'expérience utilisateur s'en trouve simplifiée et plus transparente.

Enfin, un module de reporting IA (Gemini) synthétise les tendances, calcule des indicateurs clés (total, montants, taux de conversion), propose des actions et exporte les données au format CSV. Les informations sont stockées de manière sécurisée et facilement consultables, ce qui soutient la prise de décision et la planification à long terme.

Ce rapport est structuré comme suit : le premier chapitre présente l'entreprise d'accueil et situe le projet dans son contexte. Le deuxième chapitre expose les besoins fonctionnels et les acteurs concernés par la gestion des contacts et opportunités. Nous décrivons ensuite l'environnement de travail et les technologies utilisées (Flutter, NestJS, MongoDB, OCR, intégration e-mail, Gemini AI). Les chapitres suivants détaillent les versions de l'application, les cas d'usage, les interactions (diagrammes de séquence) et l'implémentation des écrans. Enfin, nous concluons par une synthèse et des perspectives d'évolution.

En s'attaquant aux limites des pratiques manuelles et en tirant parti de technologies modernes, SCART constitue une avancée notable dans la gestion du cycle opportunité : de la carte de visite scannée jusqu'au reporting IA, l'application renforce l'efficacité opérationnelle et soutient des décisions plus rapides et mieux informées.

Chapter 1

Le cadre général du projet

1.1 Introduction

Dans ce chapitre, nous commencerons par présenter l'organisme d'accueil et l'ensemble de ses activités. Nous exposerons ensuite la problématique ainsi que les objectifs fixés pour le développement de ce projet. Enfin, nous conclurons par un rappel des notions théoriques pertinentes pour notre étude.

1.2 Présentation de l'organisme hôte

Dans cette partie nous présenterons l'histoire d'ESPRIT.

1.2.1 Historiques

Historique d'ESPRIT

Fondée en 2003, l'École Supérieure Privée d'Ingénierie et de Technologies (ESPRIT) est un établissement privé d'enseignement supérieur basé à Ariana (Tunisie), accrédité par le Ministère de l'Enseignement Supérieur et de la Recherche Scientifique.

Depuis 2020, ESPRIT a rejoint le réseau Honoris United Universities, renforçant son ouverture internationale et ses partenariats académiques et industriels.

L'école propose un cycle préparatoire et plusieurs spécialités d'ingénierie: informatique, télécommunications, génie civil et génie électromécanique, ainsi qu'un pôle management via Esprit School of Business. Les enseignements sont dispensés en français et en anglais.

Figure 1.1: ESPRIT logo

1.2.2 Fiche d'identité

Nom de l'établissement : ESPRIT — École Supérieure Privée d'Ingénierie et de Technologie
Type : École d'ingénieurs privée (enseignement supérieur)
Année de création : 2003
Localisation : Pôle Technologique El Ghazala, Ariana — Tunisie
Langues d'enseignement : français et anglais
Site web : https://www.esprit.tn
Appartenance : Membre du réseau Honoris United Universities (depuis 2020)

1.2.3 Activités

ESPRIT forme des ingénieurs à travers un cycle préparatoire et plusieurs spécialités (informatique, télécommunications, génie civil, génie électromécanique), et dispose d'un pôle management via *Esprit School of Business*. L'établissement développe également des partenariats industriels favorisant l'employabilité et l'ouverture internationale des étudiants (réseau Honoris).

Figure 1.2: Notre Clients and Parteneres of ESPRIT

1.2.4 Organisation

ESPRIT est structuré autour de trois entités principales : l'ingénierie (ESPRIT School of Engineering), le cycle préparatoire intégré (ESPRIT Prépa) et l'école de management (ESB). Ces entités sont soutenues par des services supports tels que le centre de langues, la cellule relations entreprises, et les services aux étudiants. Implantée à Ariana avec une antenne à Monastir, ESPRIT bénéficie de partenariats académiques et industriels et fait partie du réseau Honoris United Universities depuis 2020, renforçant ainsi son ouverture internationale.

1.3 Étude préliminaire

Dans cette section, nous exposons d'abord la problématique rencontrée dans le contexte d'une application mobile pour la gestion des contacts et opportunités commerciales. Ensuite, nous présentons un état de l'art des solutions existantes (applications OCR, CRM mobiles, reporting assisté par l'IA). Enfin, nous décrivons la solution proposée : une application mobile Flutter associée à un backend NestJS, exploitant l'OCR pour extraire les données d'une carte de visite, structurer les opportunités et proposer un module de reporting intelligent via Gemini.

1.3.1 Problématique

Le secteur commercial et RH fait face à des défis croissants d'efficacité et d'innovation dans la gestion des contacts et des opportunités. Les données issues des cartes de visite sont souvent saisies manuellement, provoquant pertes d'information, erreurs et délais, et rendant difficile le suivi du pipeline (recherche, filtres, états). Il existe un besoin d'une solution mobile capable de scanner (OCR), d'extraire les champs clés (nom, société, email, téléphone, etc.), de créer l'opportunité et d'automatiser le reporting (insights IA, exports), les outils actuels étant peu fl

1.3.2 Revue de l'état actuel

Dans le domaine commercial, la gestion des contacts et opportunités s'appuie encore sur des méthodes traditionnelles : feuilles Excel, échanges d'emails, ou outils CRM datés. Ces pratiques posent des limites notables en termes de temps, de fiabilité et de partage. La collaboration multi-acteurs reste fragmentée et peu traçable.

La saisie manuelle des données issues des cartes de visite est chronophage et source d'erreurs, ce qui dégrade la qualité du suivi et la réactivité commerciale au quotidien. Les informations clés (coordonnées, société, fonction, email, téléphone) restent éparses et difficilement accessibles en mobilité, ralentissant l'avancement du pipeline. Les décisions se prennent avec retard faute de visibilité consolidée.

Côté collaboration, il est difficile d'organiser et de partager l'information entre commerciaux, managers et support, ce qui génère des doublons, pertes et malentendus. L'absence d'un historique fiable nuit au pilotage et à l'amélioration continue.

En synthèse, les approches manuelles et non intégrées limitent la productivité, la fiabilité des données et la collaboration. Une solution mobile et centralisée est requise pour structurer le pipeline, accélérer la recherche et fluidifier les échanges. L'OCR et l'automatisation permettent de fiabiliser la capture et la mise à jour. L'IA de reporting apporte une vision synthétique et actionnable des opportunités.

Le projet propose une application mobile Flutter avec backend NestJS et MongoDB. La fonctionnalité centrale est le scan de cartes (OCR), l'extraction des champs clés (nom, société, email, téléphone, adresse, site, fonction), et la création d'opportunités. Le suivi inclut filtres par pipeline, recherche, détail et mise à jour de statut, suppression. Une page "Reporting Exports (Gemini)" fournit des insights JSON et l'export CSV. La clé API Gemini est gérée côté serveur ; l'ensemble vise autonomie, traçabilité et délais.

1.3.3 Solution proposée

Pour répondre aux défis de la prospection et du suivi, la solution proposée est le développement d'une application Flutter avec backend NestJS et MongoDB, capable de scanner des cartes de visite (OCR) et d'extraire les champs clés pour créer des opportunités fiables, tout en améliorant efficacité, précision et collaboration.

La plateforme centralise la capture, la recherche et les filtres par pipeline, ainsi que le détail, la mise à jour de statut, la suppression et un reporting IA (Gemini) unifié

1.4 Metho dology adopted

1.4.1 SCR UM

La m'ethode de gestion SCRUM [5] est bas'ee sur des d'eveloppements it'eratifs 'a un rythme consta
g'en'eralement compris entre 2 et 4 semaines (sprints). Parmi les avantages de l'alignement de la
m'ethodologie Agile avec SCRUM, on note une r'eduction de la documentation au minimum,
ce qui am'eliore la productivit'e. Cette approche permet de conserver l'historique des
d'ecisions prises sur le projet du client et facilite les interventions futures sur la solution
logicielle. SCRUM est consid'er'ee comme simple, pragmatique et transparente, ce qui la rend
particuli'erement adapt'ee 'a ces objectifs.

1.4.2 T eam

La méthodologie Scrum se compose de trois rôles principaux. Ci-dessous,
un tableau présente les membres de notre équipe Scrum, leurs rôles respectifs et
une brève description de leurs responsabilités :

Name	Role	Description
TAHAR BEN LAKHDAR	Product Owner	
SALAH BOUSBIA	Scrum Master	S'assure que les principes et pratiques Scrum sont respectés.
Motaz SAMMOUD, Amal ammar	Developer	Créer et implémenter des fonctionnalités produit conformément aux exigences définies par le Scrum Master.

T able 1.1: Scrum T eam Roles and Resp onsibilities

Detailing

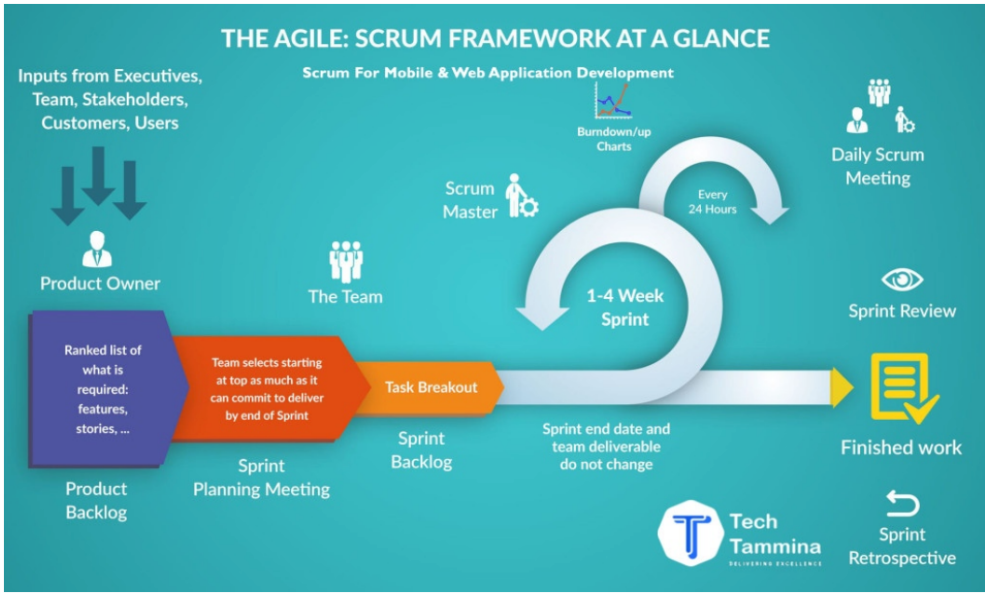

Figure 1.4: Scrum methodology

1.5 Conclusion

Dans ce chapitre introductif, nous avons pr'esent'e une vue d'ensemble compl'ete de notre projet. Nous avons commenc'e par pr'esenter l'organisme d'accueil, en mettant en avant sa structure et ses principaux d'epartements. Ensuite, nous avons approfondi la probl'ematique de notre projet et pr'esent'e la solution propos'ee. Ce chapitre initial pr'epare le terrain pour les sections suivantes, o'u nous examinerons plus en d'etail les aspects sp'ecifiques de notre projet, notamment l'analyse des besoins, la conception conceptuelle, la mise en œuvre et d'autres 'el'ements Les bases pos'ees dans ce chapitre nous guideront vers une compr'ehension d'etaill'ee des objectifs et des m'ethodologies de notre projet.

Chapter 2

Évaluation des besoins

2.1 Introduction

Dans ce chapitre, nous examinons d'abord de manière exhaustive les exigences et les objectifs du projet. La première partie porte sur l'évaluation des besoins, où nous clarifions les exigences fonctionnelles et non fonctionnelles. La section 2 traite ensuite de la modélisation de ces exigences fonctionnelles. Ce chapitre prépare le terrain pour comprendre le périmètre et la finalité du projet, et pour la discussion et l'analyse détaillées dans les chapitres suivants.

2.2 Spécification des exigences

Dans cette section, nous décrirons les exigences fonctionnelles et non fonctionnelles.

2.2.1 Identification de l'acteur

Commercial (utilisateur principal)
Manager commercial (supervision/validation)
Administrateur (droits d'accès, paramètres, pipelines)

2.2.2 Exigences fonctionnelles

Administrateur (droits d'accès, paramètres, pipelines)

Manager commercial (supervision/validation)

Administrateur (droits d'accès, paramètres, pipelines)

Acteur	Cas d'utilisation
Commercial	- Authentification / Réinitialisation mot de passe. - Modifier profil. - Scanner carte de visite (OCR). - Valider / corriger champs. - Créer une opportunité. - Rechercher / filtrer pipeline. - Consulter détail opportunité. - Mettre à jour statut. - Envoyer email via mailto. - Supprimer opportunité / historique / notifications.

Table 2.1: Cas d'utilisation pour l'acteur Commercial

Acteur	Cas d'utilisation
Manager commercial	- Authentification / Réinitialisation. - Visualiser pipeline global / KPI. - Rechercher / filtrer (commercial, étape, période). - Valider / recadrer opportunité. - Consulter insights IA (Gemini), top sociétés. - Export CSV / relances emails.

Table 2.2: Cas d'utilisation pour l'acteur Manager commercial

Acteur	Cas d'utilisation
Admin	- Gérer utilisateurs / rôles. - Configurer pipelines / statuts. - Paramétrer clé Gemini / variables serveur. - Superviser journaux / OCR. - Gérer rétention / suppression.

Table 2.3: Cas d'utilisation pour l'acteur Administrateur

2.2.3 Non-Functional Requirements

Performance:

- L'application doit rester fluide et rapide même sous forte charge.

Sécurité:

- Les données (utilisateurs, opportunités) doivent être stockées et chiffrées.

Utilisabilité et Expérience utilisateur

- L'interface doit être intuitive et conviviale, nécessitant une formation minimale pour tous les rôles utilisateurs.
- Un design cohérent et une navigation claire doivent améliorer l'expérience globale des utilisateurs.

Fiabilité et Disponibilité

- L'application doit garantir une haute disponibilité, avec un temps d'arrêt minimal pour la maintenance.
- Des sauvegardes régulières doivent être effectuées afin d'assurer la restauration des données en cas d'incident.

Documentation

- Une documentation complète, incluant guides utilisateurs et documentation technique, doit être fournie pour aider les utilisateurs et administrateurs.
- La documentation doit être régulièrement mise à jour pour refléter les changements et améliorations de l'application.

Chapitre 3

Étude conceptuelle

3.1 Introduction

Dans ce chapitre, nous allons définir l'architecture de notre application puis présenter un diagramme de packages ainsi qu'un diagramme entité–relation (ER).

3.2 Architecture de l'application

Afin d'assurer une base de code bien structurée et maintenable, nous avons adopté le modèle architectural **MVVM (Model – View – ViewModel)** avec Provider comme gestionnaire d'état. Ce modèle offre une séparation claire des responsabilités entre la logique métier (Model), la gestion de l'état (Provider) et l'interface utilisateur (View).
L'architecture MVVM est particulièrement adaptée au développement , mobile avec Flutter, car elle facilite la modularité ,la maintenabilité, le test unitaire et l'évolutivité.

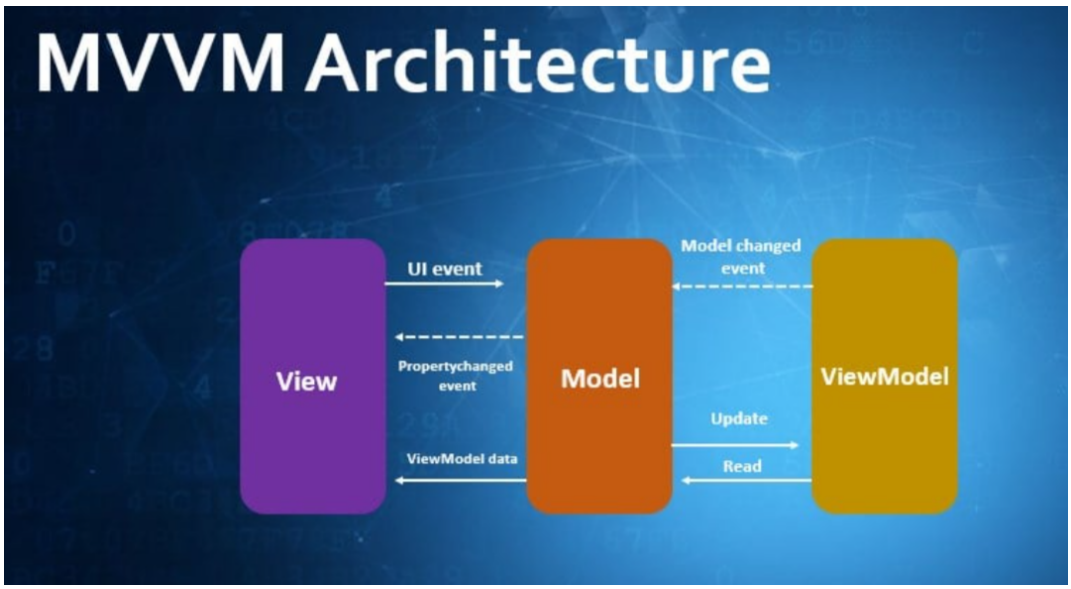

Figure 3.1: Architecture MVVM

3.3 Architecture MVVM

1. **Model** : représente les données et la logique métier de l'application. Il gère la persistance, les règles métiers et la communication avec la base de données ou les services distants (API).

2. **View** : correspond à l'interface utilisateur. Elle affiche les informations fournies par le ViewModel et transmet les actions de l'utilisateur (clics, saisie, navigation). Elle ne contient pas de logique métier.

3. **ViewModel** : fait le lien entre la View et le Model. Il expose les données sous une forme directement exploitable par l'interface, gère l'état de l'application et orchestre les appels au Model.

Avantages de l'architecture MVVM :

1. **Séparation des responsabilités** : chaque couche a un rôle clair (données, logique, interface), ce qui améliore l'organisation du code.

2. **Testabilité** : la logique étant isolée dans le ViewModel et le Model, il est plus simple d'écrire des tests unitaires.

3. **Réutilisabilité et évolutivité** : l'architecture facilite l'ajout de nouvelles fonctionnalités sans impacter les autres couches.

4. **Adaptée à Flutter** : MVVM s'intègre bien avec le pattern `Provider` ou Riverpod pour la gestion d'état, ce qui simplifie le développement mobile.

3.4 Diagramme de packages

Notre application est divisée en deux couches principales :

- **Frontend (Flutter – MVVM)** : L'application mobile est développée avec le framework Flutter en adoptant le modèle architectural MVVM. La couche View gère l'interface utilisateur, le ViewModel assure la gestion de l'état et la logique de présentation, tandis que le Model traite les données et communique avec l'API.
- **Backend (NestJS – API REST)** : Le backend est implémenté avec le framework NestJS. Il expose des services REST pour la gestion des utilisateurs, de model OCR , DetectionOcr , des opportunités commerciales, des relances et du reporting. Cette couche est responsable de la logique métier, de la persistance des données et de la sécurité.

3.5 Conclusion

En conclusion, ce chapitre a permis de définir la structure conceptuelle de notre application. L'adoption du modèle MVVM pour le frontend Flutter et l'utilisation de NestJS pour le backend garantissent une application modulaire, maintenable, scalable et adaptée aux besoins du projet.

Chapitre 4

Réalisation

4.1 Introduction

Dans ce chapitre, nous présentons les choix techniques et l'environnement de travail utilisés pour développer notre application. Nous détaillons les décisions prises, les outils adoptés ainsi que le matériel employé.

4.2 Choix techniques

Notre projet repose sur une architecture moderne combinant un frontend en Flutter et un backend en NestJS. Ces choix technologiques garantissent une application performante, maintenable et évolutive.

Flutter : Utilisé pour le développement du frontend mobile et web. Flutter offre une interface réactive, moderne et multi-plateformes grâce à son moteur de rendu.

NestJS : Adopté pour le backend, ce framework Node.js fournit une API REST robuste, sécurisée et extensible, facilitant la gestion des données et l'intégration avec Flutter.

4.3 Environnement et Outils

4.3.1 Environnement matériel

Machine	Spécifications
Système	Windows 11
RAM	16GB
Processeur	Intel i5 10ème génération quad-core
CarteG	GeForce GTX1650Ti Max-Q Design

4.3.2 Environnement logiciel

Outils utilisés

Visual Studio Code

Figure 4.1: Visual Studio Code

Dans l'environnement logiciel de notre projet, nous avons principalement utilisé Visual Studio Code (VS Code) comme environnement de développement intégré (IDE). VS Code est un éditeur de code léger, extensible et polyvalent qui offre un riche ensemble de fonctionnalités pour le développement d'applications Flutter.
Ci-dessous, nous décrivons les principaux aspects de notre environnement logiciel :

- Description : Visual Studio Code est un éditeur de code gratuit et open-source développé par Microsoft. Il offre un environnement léger mais puissant pour coder, déboguer et gérer des projets Flutter.
- Fonctionnalités : VS Code est reconnu pour sa large gamme d'extensions et de plugins qui améliorent l'expérience de développement. Ces extensions incluent Flutter et Dart, qui offrent des outils spécialisés comme l'autocomplétion, le débogage et la gestion de projet.
- Gestion de version : VS Code s'intègre parfaitement avec les systèmes de gestion de version comme Git, permettant une collaboration efficace et une gestion optimale du code source.
- Extensions : Pour accroître notre productivité, nous avons installé des extensions spécifiques à Flutter. Elles offrent des fonctionnalités supplémentaires et facilitent des tâches comme la création de widgets, la gestion d'état et le formatage du code.
- Personnalisation : VS Code permet une personnalisation poussée via les paramètres, thèmes et raccourcis clavier. Cela permet d'adapter l'IDE aux préférences et au flux de travail du développeur.
- Multi-plateforme : Visual Studio Code est disponible sous Windows, macOS et Linux, ce qui le rend accessible aux développeurs quel que soit leur OS.

- Communauté : La communauté VS Code est active et dynamique, offrant des ressources précieuses, des tutoriels et des forums de support pour les développeurs. Cette approche orientée communauté favorise la collaboration et la résolution de problèmes.

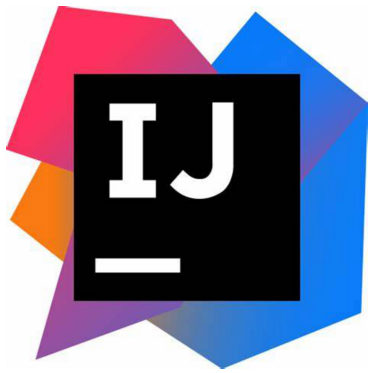

Figure 4.2: IntelliJ IDEA

Dans l'environnement logiciel de notre projet, nous avons principalement utilisé IntelliJ IDEA comme environnement de développement intégré (IDE). IntelliJ IDEA est un IDE robuste et riche en fonctionnalités qui fournit des outils complets pour le développement backend avec NestJS et Node.js. Ci-dessous, nous décrivons les principaux aspects de notre environnement logiciel :

- Description : IntelliJ IDEA est un IDE puissant développé par JetBrains, conçu pour le développement professionnel. Il offre un large éventail de fonctionnalités facilitant le codage, le débogage et la gestion de projets backend, notamment pour NestJS.
- Fonctionnalités : IntelliJ IDEA est réputé pour son autocomplétion intelligente, ses outils de refactoring et ses outils intégrés. Il offre un support spécialisé pour Node.js et NestJS, avec intégration de générateurs, gestion des dépendances et configuration automatique.
- Gestion de version : IntelliJ IDEA s'intègre parfaitement avec Git, permettant une collaboration efficace et une gestion optimisée du code source.
- Plugins : Pour améliorer notre productivité, nous avons utilisé des plugins spécifiques adaptés au développement NestJS et Node.js. Ces plugins offrent des fonctionnalités supplémentaires comme l'exécution de services REST, la visualisation de la base de données et la gestion des tests.
- Personnalisation : IntelliJ IDEA permet une personnalisation poussée via les paramètres, thèmes et raccourcis clavier, afin d'adapter l'environnement aux préférences et aux besoins du développeur.
- Multi-plateforme : IntelliJ IDEA est disponible sur Windows, macOS et Linux, le rendant accessible aux développeurs quel que soit leur système.

- Communauté : La communauté IntelliJ IDEA est active et dynamique, offrant des ressources précieuses, des tutoriels et des forums de support aux développeurs. Cette approche favorise la collaboration et la résolution de problèmes.

Google Colab

Figure 4.3: Google Colab

Google Colab est une plateforme de développement collaborative basée sur le cloud, permettant d'exécuter du code directement dans un environnement Jupyter Notebook. Il offre un support natif pour Python et l'intégration de bibliothèques de data science et de machine learning. Grâce à Colab, les développeurs peuvent tester, partager et exécuter leurs projets sans configuration locale, avec un accès gratuit à des GPU/TPU.

Flutter

Figure 4.4: Flutter Logo

Pour le frontend de notre projet, nous avons utilisé Flutter. Flutter est un framework open-source développé par Google pour créer des applications mobiles, web et desktop à partir d'un seul code source.

- Description : Flutter utilise le langage Dart et un moteur de rendu performant, permettant de concevoir des interfaces modernes et réactives. Il est conçu pour être multi-plateforme et adapté aussi bien aux petites parties d'un projet qu'à des applications complexes.
- Fonctionnalités : Flutter propose un système de widgets réactifs, une architecture basée sur les composants et une API intuitive. Sa simplicité permet aux développeurs de construire rapidement des interfaces dynamiques et réactives.

4.4.1 Module Utilisateur – Login

L'interface de connexion permet à l'utilisateur de s'authentifier en saisissant son email et son mot de

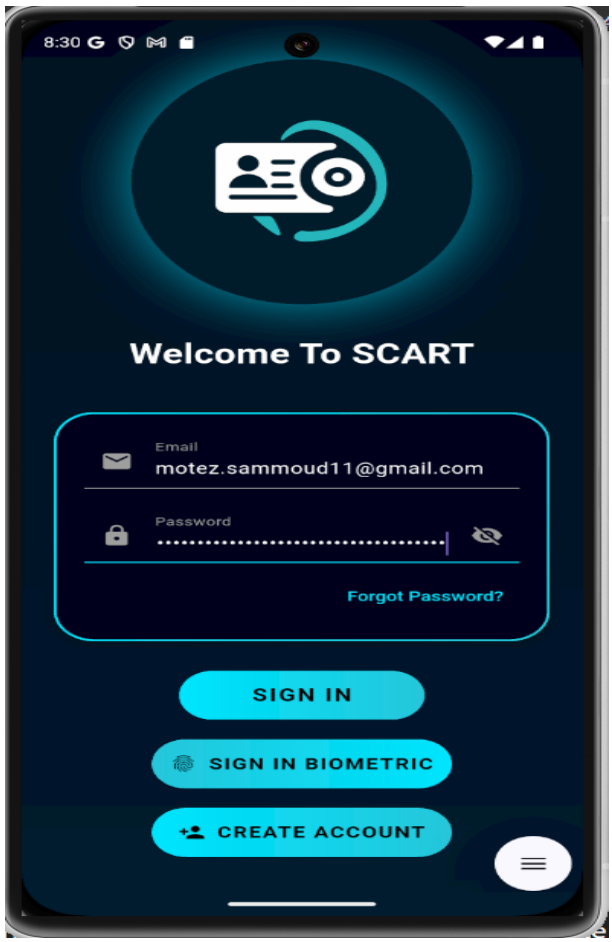

Figure 4.5: Interface de connexion (Login)

4.4.2 Module Utilisateur – Inscription

L'interface d'inscription permet aux nouveaux utilisateurs de créer un compte en saisissant leurs informations personnelles. Les données sont envoyées au backend pour validation et stockage.

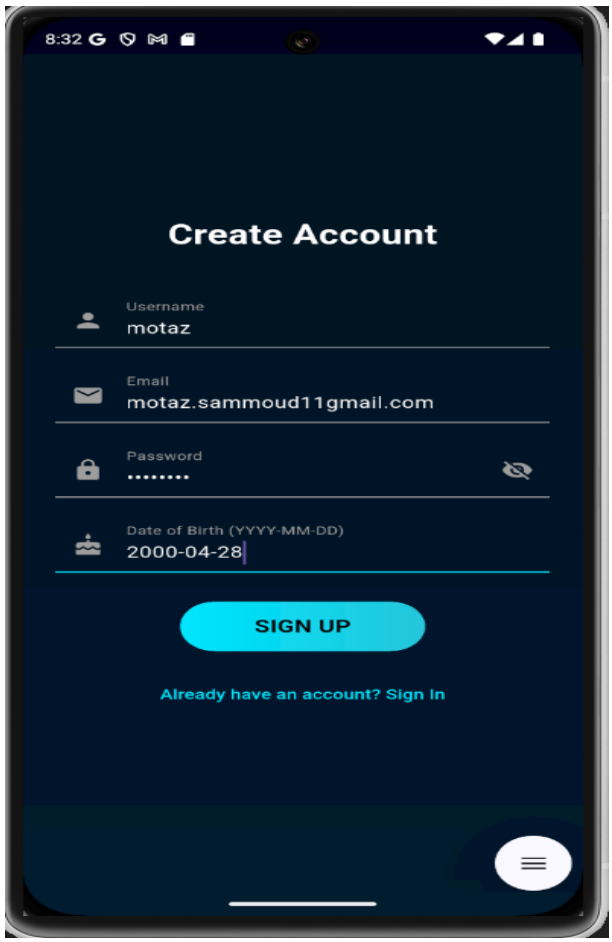

Figure 4.6: Interface d'inscription (Signup)

4.4.3 Module Utilisateur – Profil connecté et Déconnexion

L'interface de profil permet à l'utilisateur connecté de consulter ses informations personnelles (nom, email, photo de profil). Elle inclut également un bouton de déconnexion (Logout) qui permet de terminer la session en supprimant le jeton d'authentification.

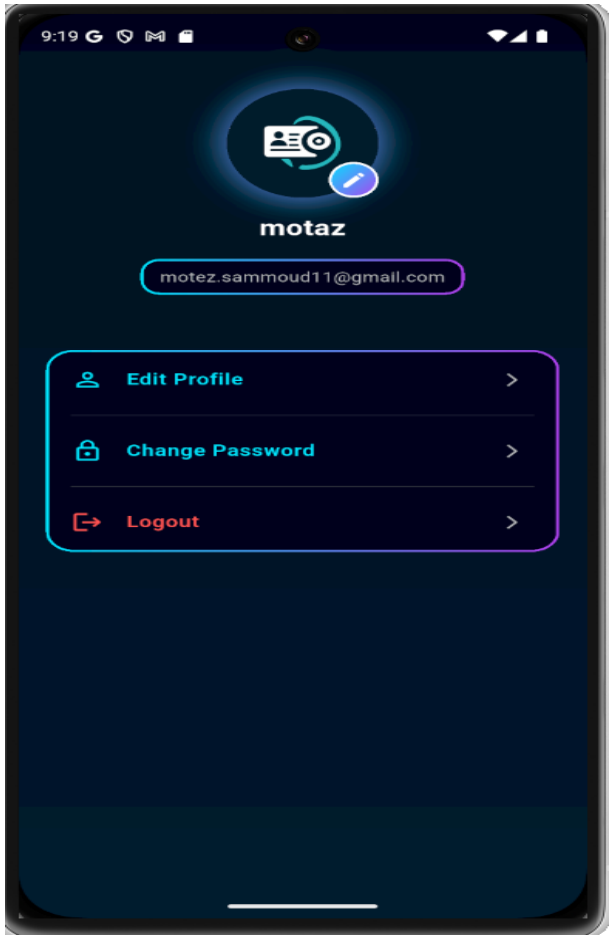

Figure 4.7: Interface du profil utilisateur et déconnexion

4.4.4 Module Utilisateur – Modifier le profil

L'interface permet à l'utilisateur de mettre à jour ses informations personnelles (nom, email, photo de profil). Les données modifiées sont envoyées au backend NestJS qui met à jour la base de données.

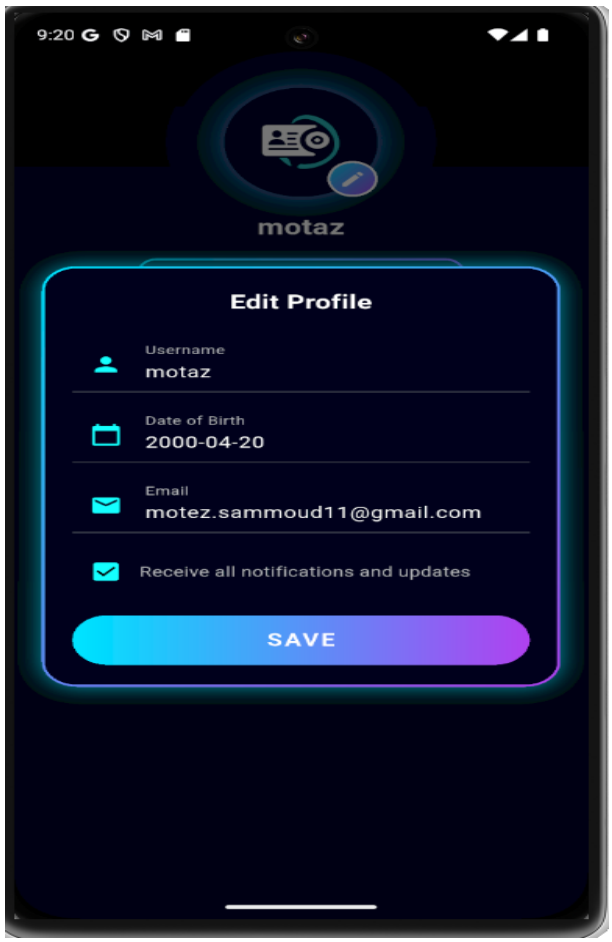

Figure 4.8: Interface de modification du profil utilisateur

4.4.5 Module Utilisateur – Changer le mot de passe

Cette interface permet à l'utilisateur de modifier son mot de passe en saisissant l'ancien mot de passe et le nouveau. Le backend NestJS valide la requête et met à jour la sécurité du compte.

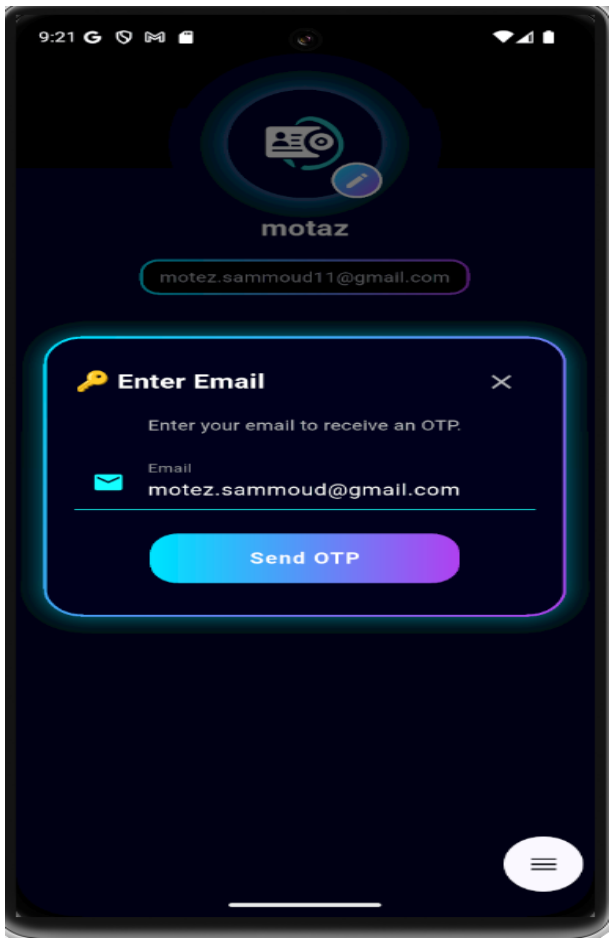

Figure 4.9: Interface de changement du mot de passe

4.4.3 Module OCR – Scan Carte de visite

Ce module ouvre la caméra et permet de scanner une carte de visite. L'image capturée est envoyée a

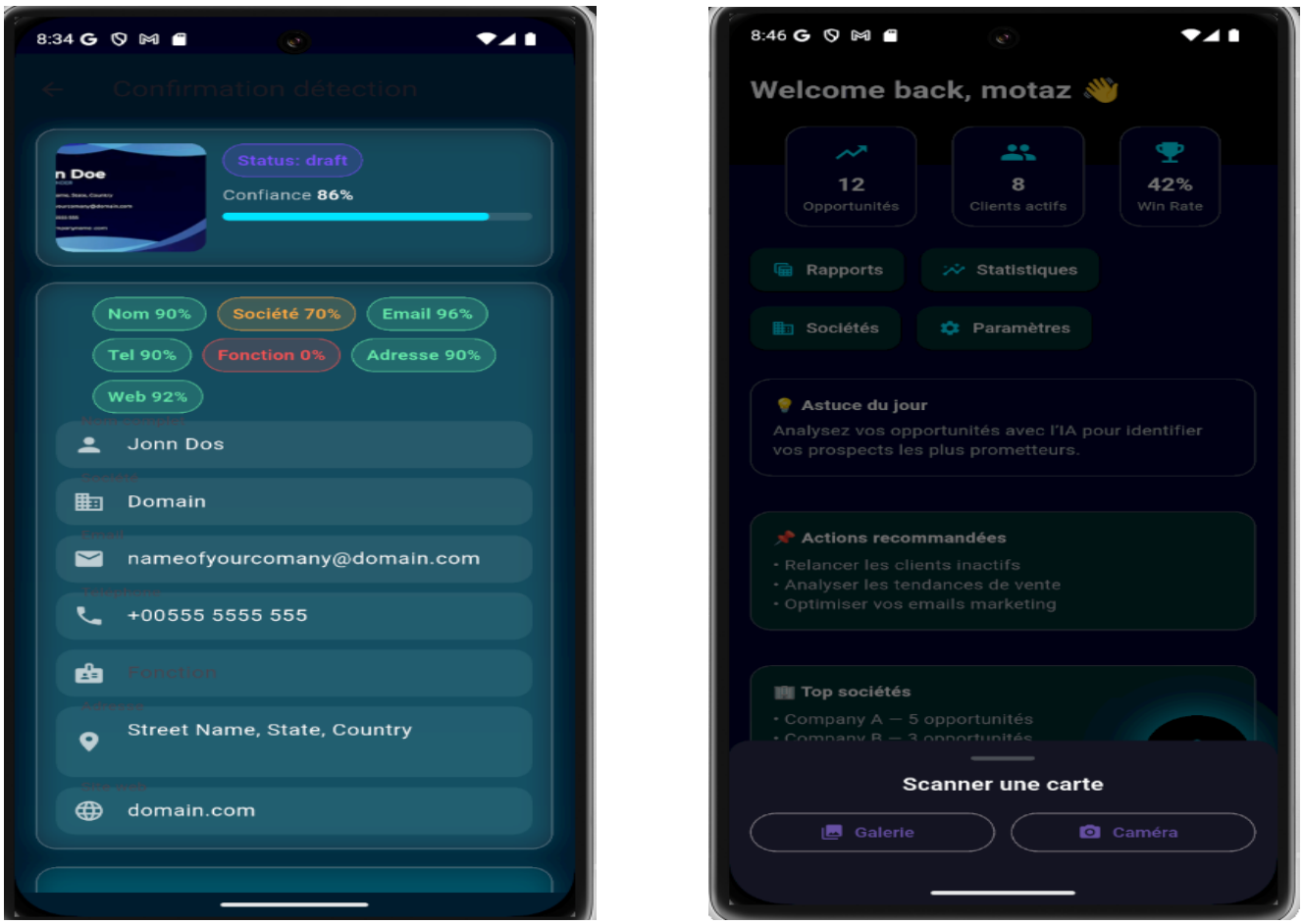

Figure 4.7: Module OCR – Capture caméra et extraction

4.4.4 Module OCR Detection – Liste des cartes détectées

Ce module affiche toutes les cartes détectées avec possibilité de voir : l'adresse sur Google Maps, passer un appel téléphonique, ou envoyer un email directement depuis la carte.

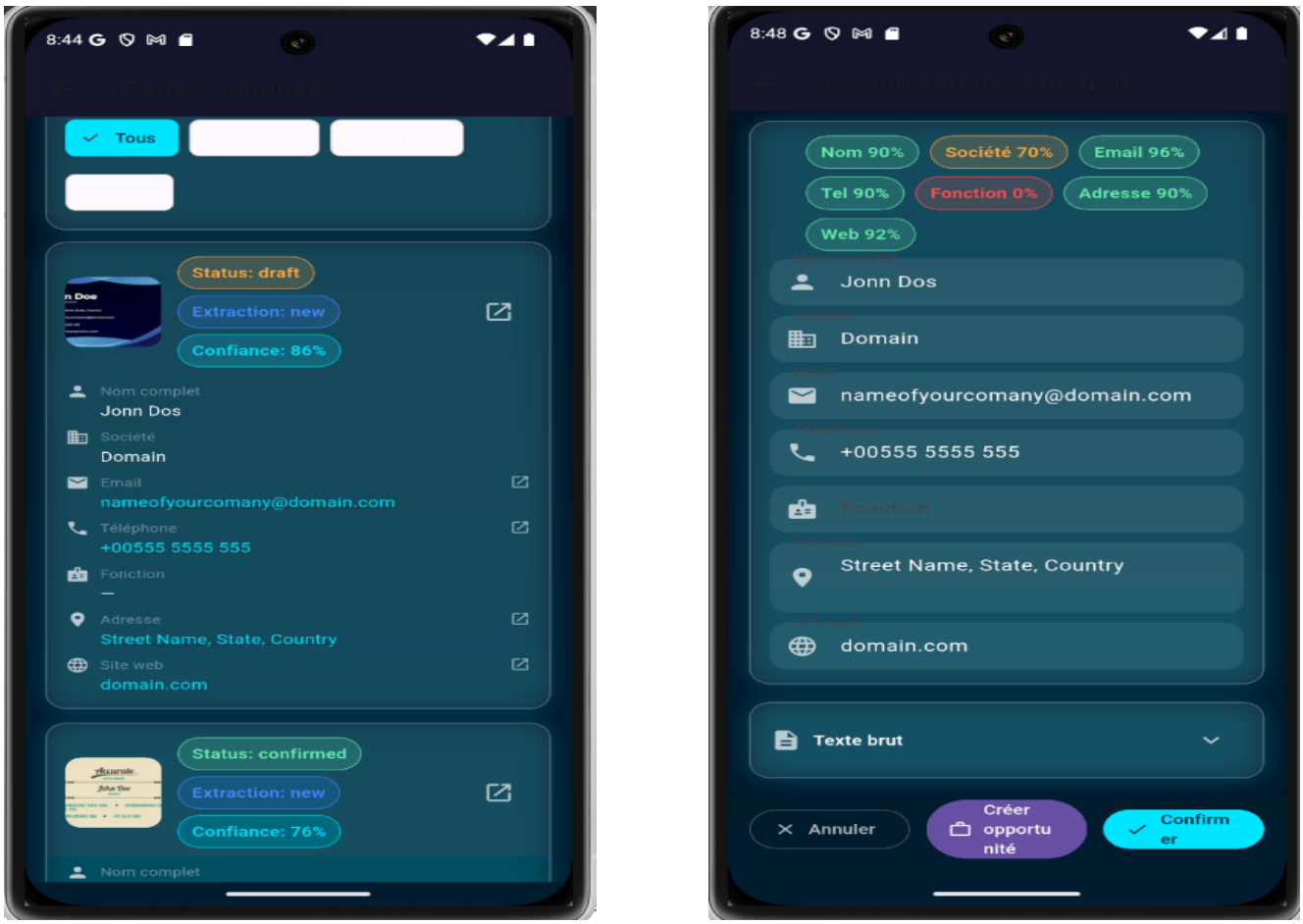

Figure 4.8: Module OCR Detection – Liste et intégrations

4.4.5 Module Opportunités – Liste et détails

Ce module gère les opportunités commerciales. L'utilisateur peut afficher la liste de toutes les opportunités ainsi que les détails de chacune (contact, statut, historique).

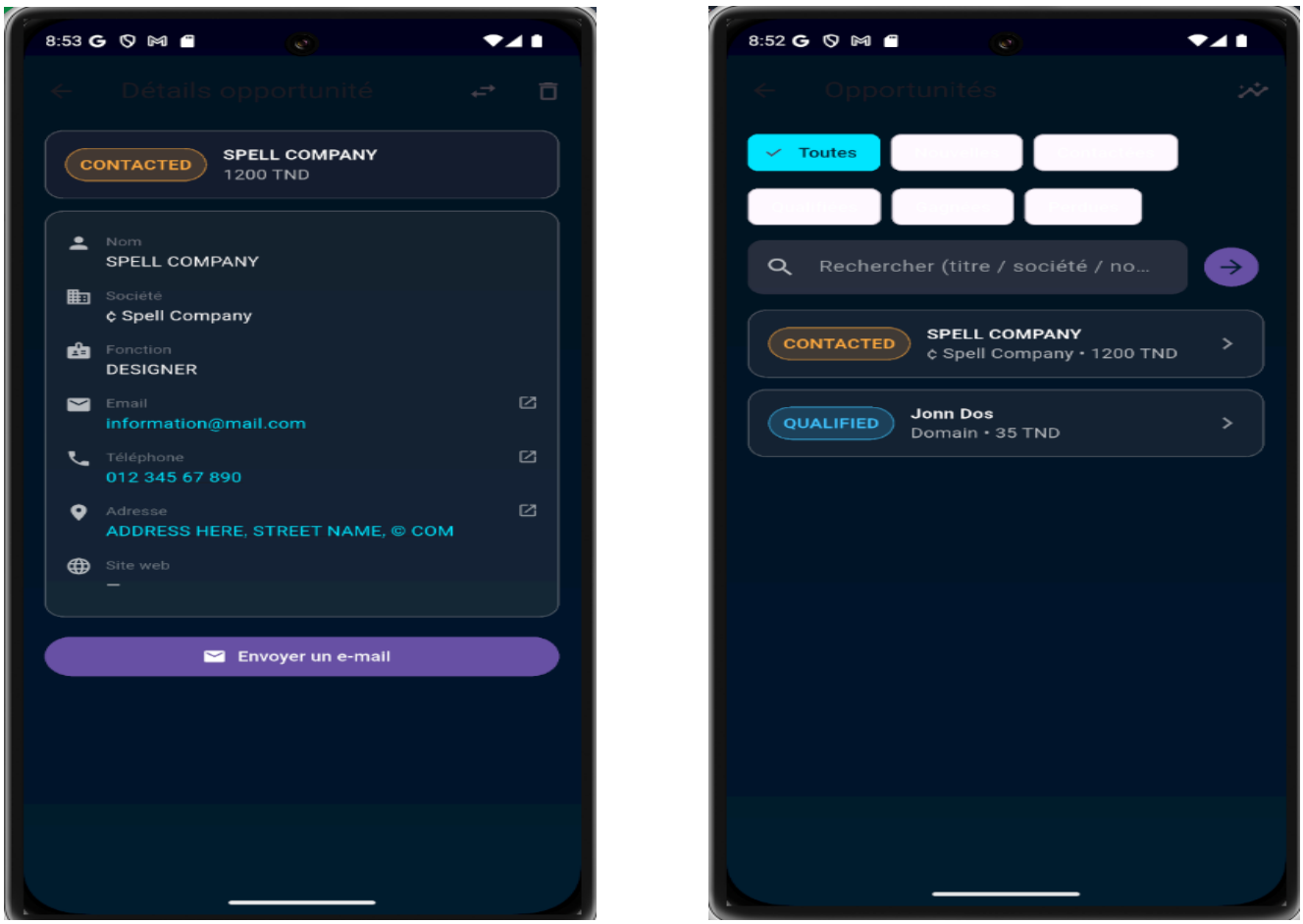

Figure 4.9: Module Opportunités – Liste et détails

4.4.6 Module Reporting – IA et Export

Le module de reporting permet de générer automatiquement un rapport grâce à l'IA et de l'exporter en PDF ou CSV. L'utilisateur peut aussi prévisualiser le rapport avant téléchargement.

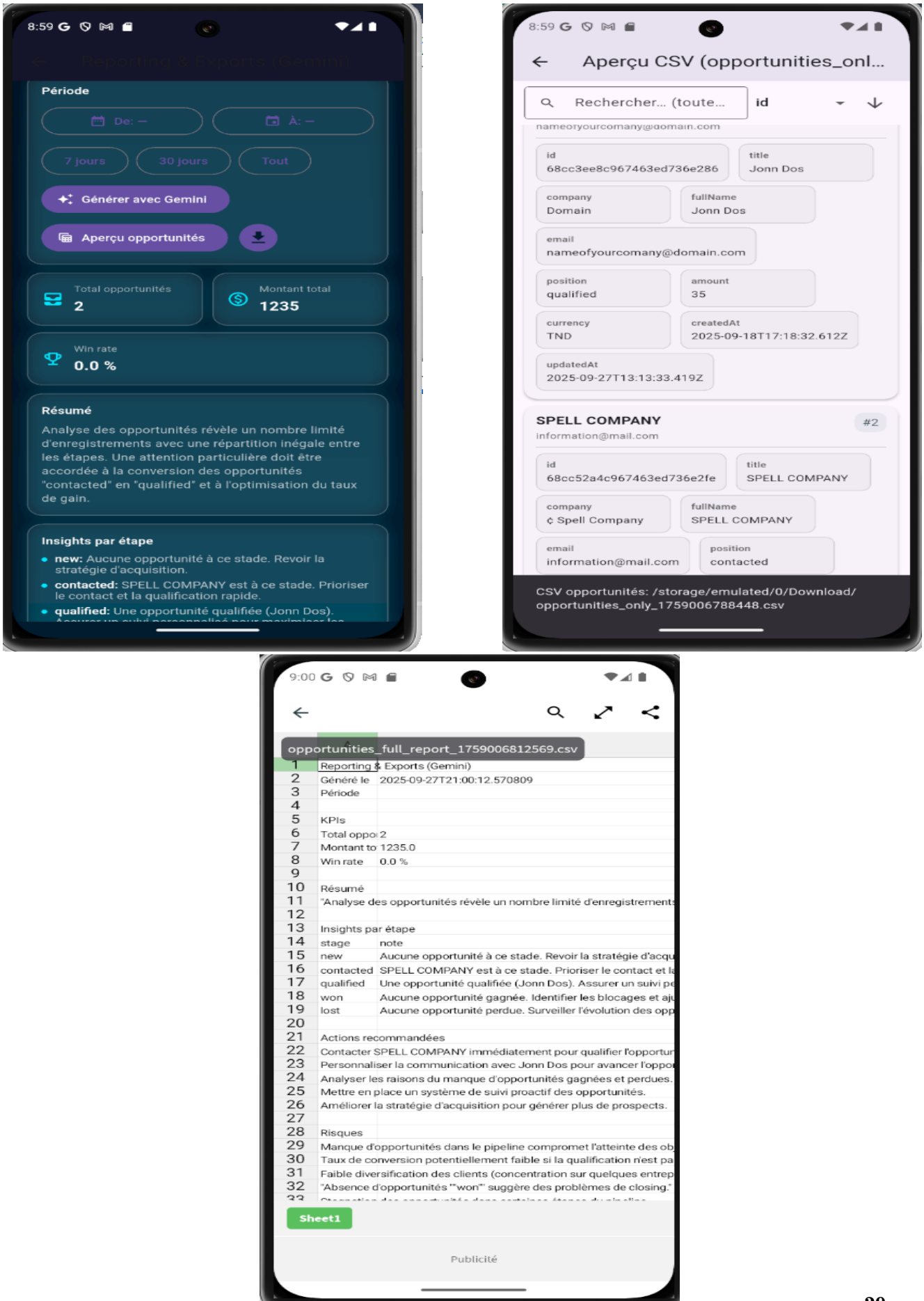

Figure 4.10: Module Reporting – Génération IA et export

4.5 Conclusion

En résumé, le chapitre de réalisation illustre la mise en œuvre réussie des fonctionnalités essentielles et des interfaces qui renforcent considérablement la gestion et l'efficacité de notre application. L'intégration de Flutter pour le frontend et de NestJS pour le backend s'est révélée très efficace, permettant des interactions fluides entre les utilisateurs et le système.

Conclusion Générale

Ce rapport conclut mon stage ingenieur réalisé à l'ESPRIT au sein de l'entreprise, où nous avons mis en pratique nos connaissances académiques et les compétences acquises durant le stage. L'objectif principal était de développer une application complète destinée à optimiser le processus de gestion et de communication commerciale. Pour ce faire, nous avons utilisé Flutter pour le développement du frontend et NestJS pour le backend,

créant ainsi une solution robuste et performante pour la gestion des opportunités et du suivi. Le choix de Flutter nous a permis de construire une interface dynamique et intuitive, tandis que NestJS a facilité la gestion efficace du backend et du traitement des données. L'approche agile, et plus particulièrement la méthodologie Scrum, nous a permis de suivre

un processus itératif, améliorant à la fois la rapidité et la qualité du développement, tout en assurant que l'application réponde aux besoins des utilisateurs et aux exigences du projet. Cette expérience a été très enrichissante pour renforcer nos compétences techniques et notre compréhension de l'importance d'aligner la logique métier avec des solutions

techniques adaptées. Les connaissances acquises nous seront d'un grand apport pour nos futures carrières en tant qu'ingénieurs informaticiens.

En conclusion, le projet a atteint ses objectifs en optimisant la gestion des processus, en améliorant le suivi des opportunités et en facilitant la communication interne. L'application constitue une preuve de notre engagement en faveur de l'innovation et de l'efficacité opérationnelle.