

PROJET DE FIN D'ANNÉE

5ème Année en Ingénierie Informatique et Réseaux

Application de détection d'émotions faciales basée sur l'intelligence artificielle

Backend FastAPI et Frontend Flutter

Réalisé par :

Ismail Sadek Talimi

Zineb Nafil

Encadrant Pédagogique :

LARHLIMI Abderrahim

Année universitaire : [2025/2026]

Table des matières

Introduction générale	1
1 Présentation du cadre de projet	3
1.1 Introduction	3
1.2 Étude de l'existant	3
1.2.1 Description de l'existant	3
1.3 Choix de modèle de développement	6
1.4 Planning prévisionnel	6
1.5 Conclusion	6
2 Spécification des besoins	7
2.1 Introduction	7
2.2 Spécification des besoins fonctionnels	7
2.2.1 Besoin fonctionnel global 1 : Gestion des utilisateurs	7
2.2.2 Besoin fonctionnel global 2 : Analyse d'émotions	8
2.2.3 Besoin fonctionnel global 3 : Historique et rapport	8
2.3 Spécification des besoins non fonctionnels	8
2.4 Présentation des cas d'utilisation	11
2.4.1 Présentation des acteurs	11
2.4.2 Exemple de cas d'utilisation : analyser une émotion	11
2.5 Conclusion	11
3 Conception du système	12
3.1 Introduction	12
3.2 Modélisation dynamique	12
3.2.1 Diagrammes de séquences	12
3.2.2 Diagrammes de collaboration	13
3.2.3 Diagrammes d'états-transition	14
3.2.4 Diagrammes d'activité	16
3.3 Modélisation statique	18
3.3.1 Diagramme de classes	18

3.3.2	Modèle relationnel	19
3.3.3	Dictionnaire de données	19
3.3.4	Architecture de l'application	21
3.4	Conclusion	22
4	Réalisation du système	23
4.1	Introduction	23
4.2	Environnement de développement et technologies utilisées	23
4.2.1	Environnement matériel	23
4.2.2	Technologies backend	23
4.2.3	Technologies frontend	24
4.2.4	Outils de développement	24
4.3	Principales interfaces graphiques	25
4.4	Conclusion	30
	Conclusion générale	31

Table des figures

3.1	Diagramme de séquence de l'authentification	12
3.2	Diagramme de collaboration pour l'analyse d'une émotion	13
3.3	Diagramme d'états de la session utilisateur	14
3.4	Diagramme d'activité de l'authentification	16
3.5	Diagramme de classes simplifié du système	18
3.6	Architecture logicielle du système	21
3.7	Architecture matérielle (diagramme de déploiement)	22
4.1	Écran de connexion de l'application Flutter	25
4.2	Écran de de créer un compte Flutter	26
4.3	Écran d'analyse d'émotion	27
4.4	Les analyses précédentes de l'utilisateur	27
4.5	Historique	28
4.6	Rapport émotionnel	29
4.7	Rapport émotionnel	29

Liste des tableaux

1.1	Planning prévisionnel	6
2.1	Description du cas d'utilisation “Analyser une émotion”	11
3.1	Dictionnaire de données (extrait)	19
3.2	Dictionnaire de données	20

Introduction générale

Nous vivons aujourd’hui dans un contexte où les interactions homme-machine deviennent de plus en plus riches et personnalisées. Les systèmes capables de percevoir et d’interpréter les émotions humaines ouvrent la voie à de nombreuses applications : accompagnement psychologique, amélioration de l’expérience utilisateur, agents conversationnels empathiques, applications éducatives ou encore outils d’assistance au bien-être. Cependant, la mise en place d’une telle solution nécessite la combinaison de plusieurs briques technologiques : vision par ordinateur, modèles d’apprentissage profond, services de génération de texte et interfaces utilisateurs modernes.

Dans ce projet de fin d’année, nous nous proposons de concevoir et de réaliser une application complète de **détection d’émotions faciales** à partir d’images de visages. L’application repose sur un *backend* développé avec **FastAPI** (Python), qui exploite des modèles de type Transformers pour la classification d’expressions faciales ainsi qu’un modèle de génération de texte pour produire, à partir de l’émotion détectée, une description, des conseils personnalisés et un message empathique. En parallèle, un *frontend* moderne développé avec **Flutter** permet à l’utilisateur de capturer une image via la caméra, d’envoyer la requête au serveur et de visualiser les résultats dans une interface ergonomique. L’authentification est assurée localement via une base de données SQLite et des jetons JWT, sans dépendre de services externes payants.

La problématique principale à laquelle nous répondons peut être formulée comme suit : *comment concevoir et implémenter une application complète et locale de détection d’émotions, fiable et ergonomique, intégrant des modèles d’intelligence artificielle récents, tout en garantissant la simplicité de déploiement et la confidentialité des données utilisateur ?* Pour y répondre, nous avons adopté une démarche structurée allant de l’analyse de l’existant jusqu’à la mise en œuvre logicielle et aux tests.

Ce rapport est structuré en quatre chapitres. Dans le **premier chapitre**, nous présentons le cadre général du projet, l’étude de l’existant, ses limites, ainsi que la solution proposée et le modèle de développement retenu. Le **deuxième chapitre** est consacré à la spécification des besoins fonctionnels et non fonctionnels, ainsi qu’à la description des cas d’utilisation principaux de notre système. Le **troisième chapitre** traite de la conception du système en présentant la modélisation statique et dynamique à l’aide de diagrammes UML ainsi que l’architecture logicielle et matérielle de l’application. Enfin, le **quatrième**

chapitre décrit la réalisation du système, l'environnement de développement adopté ainsi que les principales interfaces graphiques mises en œuvre. Le rapport se termine par une conclusion générale, une bibliographie et des annexes.

Chapitre 1

Présentation du cadre de projet

1.1 Introduction

Dans ce chapitre, nous présentons le cadre général de notre projet de fin d'année. Nous décrivons dans un premier temps l'existant en matière de systèmes de détection d'émotions faciales et de solutions d'assistance émotionnelle. Nous analysons ensuite les limites de ces approches, ce qui nous amène à formuler notre proposition de solution. Nous terminons par le choix du modèle de développement logiciel et le planning prévisionnel du projet.

1.2 Étude de l'existant

1.2.1 Description de l'existant

Les familles de solutions en détection d'émotions : analyse Le domaine de la détection automatisée des émotions s'est structuré autour de plusieurs approches technologiques et méthodologiques distinctes, qui répondent à des besoins, des contraintes et des publics variés. Voici une analyse plus approfondie de chaque catégorie mentionnée.

a. Applications mobiles ou web basées sur le cloud Ces applications offrent une expérience utilisateur intégrée et grand public. L'utilisateur ouvre l'application, active sa caméra frontale, et obtient en quelques secondes une estimation de son état émotionnel, souvent accompagnée de visualisations (smileys, diagrammes) ou de conseils basés sur le résultat.

Fonctionnement : L'image capturée est généralement envoyée à un service cloud (par exemple, Microsoft Azure Cognitive Services, Google Cloud Vision API, ou Amazon Rekognition) qui exécute un modèle de reconnaissance faciale et d'analyse des expressions.

Avantages : Accessibilité, pas besoin de compétences techniques, interface souvent ludique ou pédagogique, mises à jour transparentes côté serveur.

Limites : Dépendance à une connexion Internet, questions de confidentialité (envoi d'images vers des serveurs tiers), parfois modèle « boîte noire » sans possibilité de personnalisation.

b. APIs des grands fournisseurs cloud (SaaS – Software as a Service) Ces services s'adressent davantage aux développeurs et aux entreprises qui souhaitent intégrer la détection d'émotions dans leurs propres produits (apps, robots, systèmes de CRM, analyse de contenu média, etc.).

Fonctionnement : Via une requête REST, on envoie une image ou un flux vidéo. L'API retourne un ensemble de scores normalisés (ex. : **joie**: 0.92, **tristesse**: 0.05, **colère**: 0.01) pour chaque visage détecté, parfois accompagnés de coordonnées de landmarks faciaux.

Principaux acteurs :

[noitemsep]

- Microsoft Azure Face API (inclus l'attribut « emotion »)
- Google Cloud Vision API (via `faceAnnotations`)
- Amazon Rekognition (attribut « Emotions »)
- IBM Watson Visual Recognition (désormais limité, mais historiquement présent)

Avantages : Modèles entraînés sur d'immenses jeux de données, maintenance et scaling gérés par le fournisseur, documentation riche, conformité RGPD possible via contrats.

Limites : Coût à l'usage, pas de contrôle sur le modèle, risques de biais culturels ou ethniques dans les prédictions, latence variable.

c. Projets open source avec modèles pré-entraînés Cette approche cible les chercheurs, les étudiants en machine learning, et les développeurs souhaitant expérimenter, personnaliser ou déployer localement des modèles sans dépendre du cloud.

Bases de données courantes pour l'entraînement :

[noitemsep]

- **FER2013** : Dataset classique de 7 émotions (colère, dégoût, peur, joie, tristesse, surprise, neutre), souvent utilisé comme benchmark académique.
- **AffectNet** : Beaucoup plus large et riche (plus d'un million d'images, 8 émotions de base + valence/arousal), permet des modèles plus robustes mais exige plus de ressources.

Frameworks et modèles populaires :

[noitemsep]

- **OpenCV avec DNN** : Chargement de modèles pré-entraînés (ex. : réseau CNN simple) pour la détection en temps réel sur appareil.
- **DeepFace (bibliothèque Python)** : Intègre plusieurs modèles état-de-l'art (VGG-Face, Facenet, etc.) et peut être utilisée pour l'émotion parmi d'autres tâches.
- **MediaPipe Face Landmarks (Google)** : Fournit des points faciaux en temps

réel, sur lesquels on peut greffer un classifieur d'émotions léger.

Avantages : Transparence, personnalisation possible (fine-tuning), fonctionnement hors ligne, coût marginal après développement.

Limites : Performance souvent inférieure aux solutions cloud (sauf si on utilise des modèles récents et optimisés), charge de déploiement et maintenance sur le développeur.

d. Systèmes de suivi d'humeur par questionnaires manuels Très répandus dans les applications de bien-être mental (ex. : Moodpath, Daylio, Youper), ces solutions ne reposent pas sur l'analyse automatique du visage, mais sur l'auto-évaluation via des échelles visuelles (smileys), des questions textuelles ou des sliders.

Fonctionnement : L'utilisateur sélectionne ou décrit son humeur à intervalles réguliers. L'application agrège les données pour produire des graphiques d'évolution, identifier des patterns et parfois proposer du contenu psychoéducatif.

Avantages : Respect de l'intimité (pas d'image), prise en compte du vécu subjectif, sensibilisation à l'introspection, pas de biais algorithmique sur l'expression faciale.

Limites : Subjectivité, dépend à la régularité et honnêteté de l'utilisateur, ne capture pas les émotions fugaces ou inconscientes, charge cognitive pour l'utilisateur.

-Critique de l'existant

Les principales limites que nous avons identifiées sont :

- la dépendance à des services cloud payants, avec des enjeux de coût, de latence et de confidentialité ;
- le manque de transparence sur les modèles utilisés et leurs performances réelles ;
- une intégration limitée entre la détection d'émotions, le conseil personnalisé et le suivi temporel ;
- une complexité de déploiement liée à la multiplicité des services externes.

-Solution proposée

Pour répondre à ces limites, nous proposons une solution composée de deux grandes parties :

- un **backend FastAPI** qui :
 - reçoit une image de visage en entrée ;
 - applique un modèle de classification d'expressions faciales pour déterminer l'émotion dominante ;
 - normalise cette émotion vers un ensemble standard (happy, sad, angry, neutral, surprise, fear, disgust) ;
 - génère une description, des conseils et un message empathique via un modèle de génération de texte ;
 - gère l'authentification locale et stocke l'historique émotionnel de l'utilisateur.
- une **application Flutter** qui :

- offre une interface moderne pour s'inscrire, se connecter et capturer une image via la caméra ;
- envoie les requêtes d'analyse au backend et affiche les résultats ;
- permet de consulter un historique simplifié et de télécharger un rapport sous forme de PDF.

1.3 Choix de modèle de développement

Nous avons opté pour un modèle de développement en V simplifié :

- analyse des besoins ;
- conception du système ;
- réalisation du backend et du frontend ;
- tests et validation.

1.4 Planning prévisionnel

Le planning prévisionnel du projet s'étend sur quatre mois (février à mai). Le tableau 1.1 illustre les grandes étapes.

TABLE 1.1 – Planning prévisionnel

Étape	Février	Mars	Avril	Mai
Étude préalable	X			
Conception		X	X	
Réalisation backend		X	X	
Réalisation frontend			X	X
Tests et validation			X	X
Rédaction du rapport				X

1.5 Conclusion

Dans ce premier chapitre, nous avons présenté le contexte général du projet, analysé l'existant et mis en évidence ses limites, puis décrit notre solution proposée ainsi que le modèle de développement et le planning adoptés. Le chapitre suivant s'intéresse à la spécification détaillée des besoins fonctionnels et non fonctionnels.

Chapitre 2

Spécification des besoins

2.1 Introduction

Ce chapitre a pour objectif de définir de manière claire les besoins auxquels notre application doit répondre. Nous distinguons d'une part les besoins fonctionnels, qui correspondent aux services attendus par l'utilisateur, et d'autre part les besoins non fonctionnels, qui concernent les contraintes de performance, de sécurité et d'ergonomie. Nous présentons également les cas d'utilisation principaux du système ainsi que les acteurs impliqués.

2.2 Spécification des besoins fonctionnels

2.2.1 Besoin fonctionnel global 1 : Gestion des utilisateurs

Inscription

L'application doit permettre à un nouvel utilisateur de créer un compte en fournissant une adresse e-mail et un mot de passe. Le mot de passe est haché et stocké dans la base de données.

Connexion

Un utilisateur enregistré doit pouvoir se connecter et obtenir un jeton JWT, utilisé ensuite pour accéder aux fonctionnalités protégées (historique, génération de rapport, etc.).

2.2.2 Besoin fonctionnel global 2 : Analyse d'émotions

Capture et envoi d'une image

L'utilisateur doit pouvoir capturer une image de son visage et l'envoyer au backend via l'application Flutter.

Détection et normalisation de l'émotion

Le backend doit appliquer un modèle de classification d'expressions faciales et renvoyer l'émotion dominante ainsi qu'un score de confiance.

Génération de conseils

À partir de l'émotion détectée, le backend doit générer une description, des conseils concrets et un message empathique.

2.2.3 Besoin fonctionnel global 3 : Historique et rapport

Enregistrement de l'historique

Pour chaque analyse, l'émotion et la confiance doivent être stockées pour les utilisateurs authentifiés, afin de permettre un suivi des émotions dans le temps.

Consultation et rapport

L'utilisateur doit pouvoir consulter un historique récent et télécharger un rapport PDF récapitulatif.

2.3 Spécification des besoins non fonctionnels

Les besoins non fonctionnels définissent les qualités du système et les contraintes qui s'appliquent à son fonctionnement. Ils complètent les exigences fonctionnelles en garantissant que l'application réponde aux attentes en termes de performance, sécurité, confidentialité et expérience utilisateur.

2.3.1 Performance raisonnable pour un usage interactif

L'application doit offrir une expérience utilisateur fluide et réactive, avec des temps de réponse compatibles avec une interaction naturelle et intuitive.

[leftmargin=*)

— **Temps de réponse :**

- Analyse d'émotion : < 3 secondes pour une analyse complète (capture + traitement + affichage)
- Navigation entre écrans : < 0.5 seconde
- Chargement de l'historique : < 2 secondes pour 50 entrées
- **Ressources système :**
 - Utilisation mémoire : < 150 MB en fonctionnement normal
 - Consommation CPU : optimisée pour éviter l'échauffement de l'appareil
 - Taille de l'application : < 100 MB (hors données utilisateur)
- **Expérience utilisateur :**
 - Interface maintenue à 60 FPS minimum
 - Feedback visuel immédiat pour chaque action utilisateur
 - Gestion optimale du cache pour réduire les chargements répétés

2.3.2 Sécurité des données

La protection des données utilisateurs et la sécurisation des échanges sont fondamentales pour garantir la confiance dans l'application.

[leftmargin=*)

- **Authentification :**
 - Hachage des mots de passe avec algorithme sécurisé (bcrypt ou Argon2)
 - Utilisation de jetons JWT (JSON Web Tokens) pour les sessions
 - Expiration des tokens après 24 heures d'inactivité
 - Régénération du token de rafraîchissement
- **Protection des données :**
 - Chiffrement des communications avec TLS 1.3
 - Chiffrement des données sensibles au repos (base de données locale)
 - Validation et sanitisation de toutes les entrées utilisateur
 - Protection contre les attaques CSRF et XSS
- **Gestion des accès :**
 - Principe du moindre privilège pour les droits d'accès
 - Audit des logs d'authentification
 - Limitation des tentatives de connexion (5 tentatives maximum)

2.3.3 Confidentialité et protection de la vie privée

Le respect de la vie privée des utilisateurs est une exigence prioritaire, particulièrement sensible étant donné la nature des données traitées (émotions, images faciales).

[leftmargin=*)

- **Traitement en local :**
 - Possibilité d'exécuter l'analyse d'émotion entièrement sur l'appareil

- Option de stockage exclusivement local des images et résultats
- Consentement explicite requis pour tout envoi vers le cloud
- **Gestion des données personnelles :**
 - Conformité au RGPD pour les utilisateurs européens
 - Politique de confidentialité transparente et accessible
 - Droit à l'effacement des données facilement exercé
 - Anonymisation des données utilisées pour l'amélioration du modèle
- **Transparence :**
 - Indication claire quand l'analyse est effectuée localement vs. cloud
 - Explication des données collectées et de leur utilisation
 - Pas de collecte de données sans consentement explicite

2.3.4 Ergonomie de l'interface Flutter

L'interface doit être intuitive, accessible et adaptée aux différentes plateformes et contextes d'utilisation.

[leftmargin=*)

- **Expérience utilisateur :**
 - Design Material 3 conforme aux guidelines d'Android et iOS
 - Navigation intuitive avec une barre de navigation inférieure fixe
 - Support natif des gestes (swipe, pinch, etc.)
 - Animations fluides et cohérentes
- **Accessibilité :**
 - Support de VoiceOver/TalkBack avec descriptions alternatives
 - Contraste des couleurs conforme au WCAG 2.1 (ratio 4.5 :1 minimum)
 - Taille de police adaptable aux préférences système
 - Éléments interactifs d'au moins 44x44 points
- **Adaptabilité :**
 - Design responsive s'adaptant aux smartphones et tablettes
 - Support des orientations portrait et paysage
 - Thèmes clair/sombre suivant les préférences système
 - Interface fonctionnelle en mode hors ligne
- **Internationalisation :**
 - Support du français et de l'anglais (au minimum)
 - Dates et nombres formatés selon les locales
 - Textes externalisés pour faciliter l'ajout de langues

2.3.5 Autres contraintes techniques

[leftmargin=*)

- **Compatibilité** : Support d’Android 10+ et iOS 13+
- **Maintenabilité** : Code modulaire avec tests unitaires ($> 80\%$ de couverture)
- **Évolutivité** : Architecture permettant l’ajout de nouvelles fonctionnalités d’analyse
- **Portabilité** : Codebase unique pour Android et iOS via Flutter
- **Documentation** : Documentation technique et utilisateur complète

2.4 Présentation des cas d’utilisation

2.4.1 Présentation des acteurs

Les acteurs principaux sont :

- utilisateur non authentifié ;
- utilisateur authentifié ;
- système backend.

2.4.2 Exemple de cas d’utilisation : analyser une émotion

TABLE 2.1 – Description du cas d’utilisation “Analyser une émotion”

Cas n°	UC_01
Acteur(s)	Utilisateur authentifié
Objectif	Obtenir l’émotion dominante à partir d’une image de visage, avec conseils associés.
Pré-conditions	L’utilisateur est connecté et le backend est disponible.
Post-conditions	L’émotion est renvoyée au frontend et éventuellement enregistrée dans l’historique.
Scénario nominal	<ol style="list-style-type: none"> 1. L’utilisateur ouvre l’écran d’analyse. 2. Il capture une image. 3. L’image est envoyée au backend. 4. Le backend calcule l’émotion et les conseils. 5. Les résultats sont affichés.

2.5 Conclusion

Dans ce chapitre, nous avons formalisé les besoins fonctionnels et non fonctionnels ainsi que les cas d’utilisation principaux. Ces spécifications servent de base à la conception du système présentée dans le chapitre suivant.

Chapitre 3

Conception du système

3.1 Introduction

Dans ce chapitre, nous présentons la solution conceptuelle proposée pour notre application de détection d'émotions faciales. Nous modélisons d'abord le comportement dynamique du système à l'aide de différents diagrammes UML (séquences, activités, états, collaboration), puis nous décrivons la structure statique à travers un diagramme de classes, un modèle relationnel et un dictionnaire de données. Enfin, nous présentons l'architecture logicielle et matérielle de l'application.

3.2 Modélisation dynamique

3.2.1 Diagrammes de séquences

Séquence d'authentification

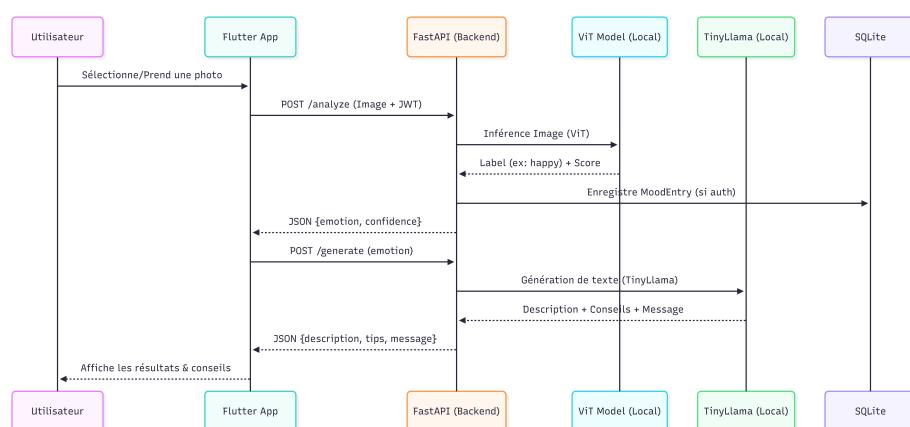


FIGURE 3.1 – Diagramme de séquence de l'authentification

Ce diagramme montre les échanges lors de la connexion : saisie des identifiants dans l'interface Flutter, envoi d'une requête POST /auth/login au backend, vérification des identifiants, génération d'un jeton JWT et retour de la réponse vers le frontend.

3.2.2 Diagrammes de collaboration

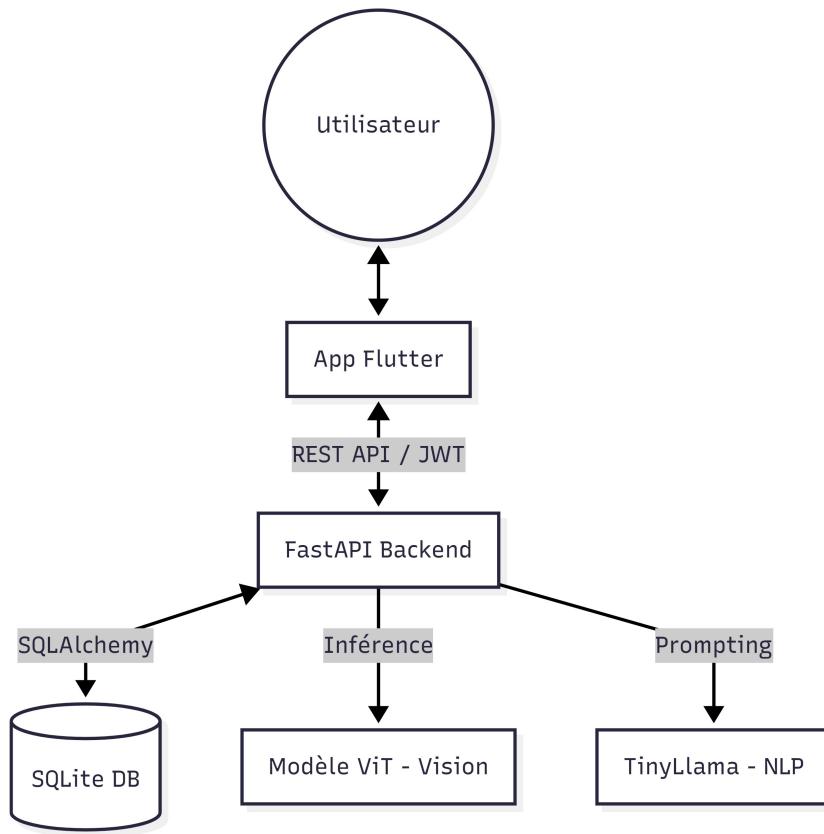


FIGURE 3.2 – Diagramme de collaboration pour l'analyse d'une émotion

Ce diagramme de collaboration met en évidence les objets qui interviennent lors de l'analyse (utilisateur, interface Flutter, contrôleur, API FastAPI, service d'émotion et base SQLite) ainsi que les messages échangés entre eux.

3.2.3 Diagrammes d'états-transition

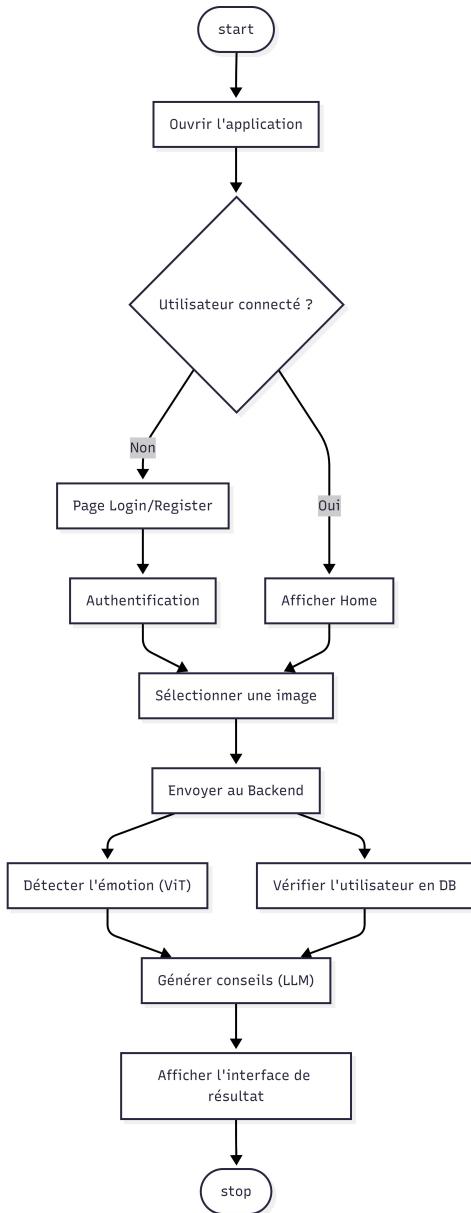


FIGURE 3.3 – Diagramme d'états de la session utilisateur

Ce diagramme d'états-transitions décrit le cycle de vie de la session utilisateur : passage de l'état « non connecté » à « connecté » après authentification réussie, puis retour à « non connecté » en cas de déconnexion ou d'expiration du jeton JWT.

3.2.4 Diagrammes d'activité

Activité d'authentification

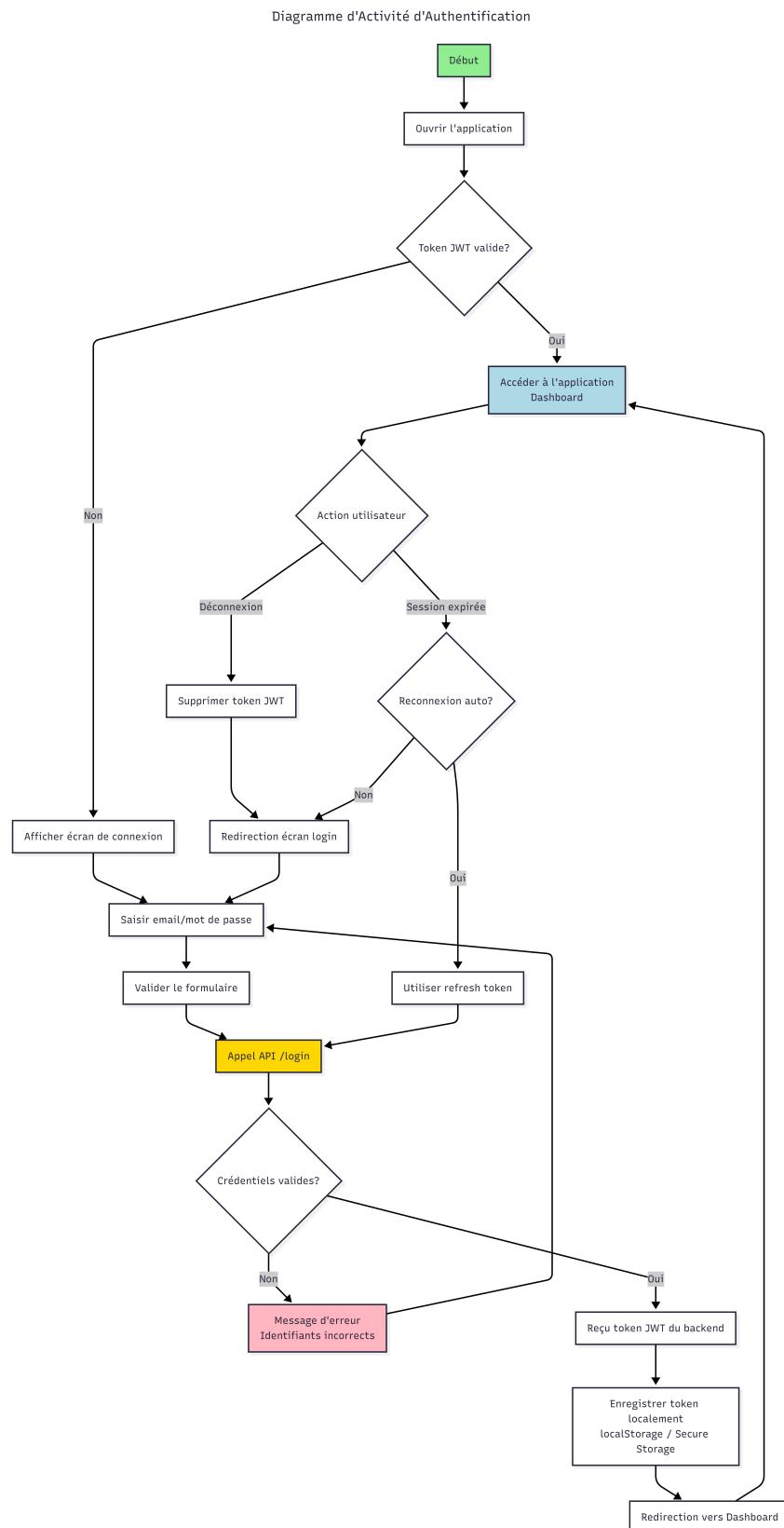


FIGURE 3.4 – Diagramme d'activité de l'authentification

Ce diagramme illustre le flux complet du processus d'authentification dans l'application EmotiScan, depuis l'ouverture de l'application jusqu'à la gestion de la session utilisateur. Il met en évidence les différentes étapes, décisions et chemins possibles lors de la connexion et de la gestion de l'accès sécurisé.

Initialisation et Vérification de Session Début : Le processus démarre lorsque l'utilisateur ouvre l'application. Vérification du Token JWT : Immédiatement après le lancement, le système vérifie si un token JWT (JSON Web Token) valide est présent dans le stockage local de l'appareil. Si valide : L'utilisateur est automatiquement redirigé vers le Dashboard principal de l'application, sans besoin de ressaisir ses identifiants. Si invalide ou absent : Le système affiche l'écran de connexion pour demander les identifiants.

Processus de Connexion Manuelle Lorsque l'utilisateur doit s'authentifier :

Saisie des identifiants : L'utilisateur entre son adresse email et son mot de passe.

Validation du formulaire : Une validation côté client vérifie le format des données (email valide, mot de passe non vide).

Appel API /login : Les identifiants sont envoyés sécurisé au backend via une requête HTTPS.

Vérification des crédentiels : Le backend authentifie l'utilisateur contre la base de données.

Si invalides : Un message d'erreur "Identifiants incorrects" est affiché, et l'utilisateur peut réessayer.

Si valides : Le backend génère et retourne un nouveau token JWT.

3.3 Modélisation statique

3.3.1 Diagramme de classes

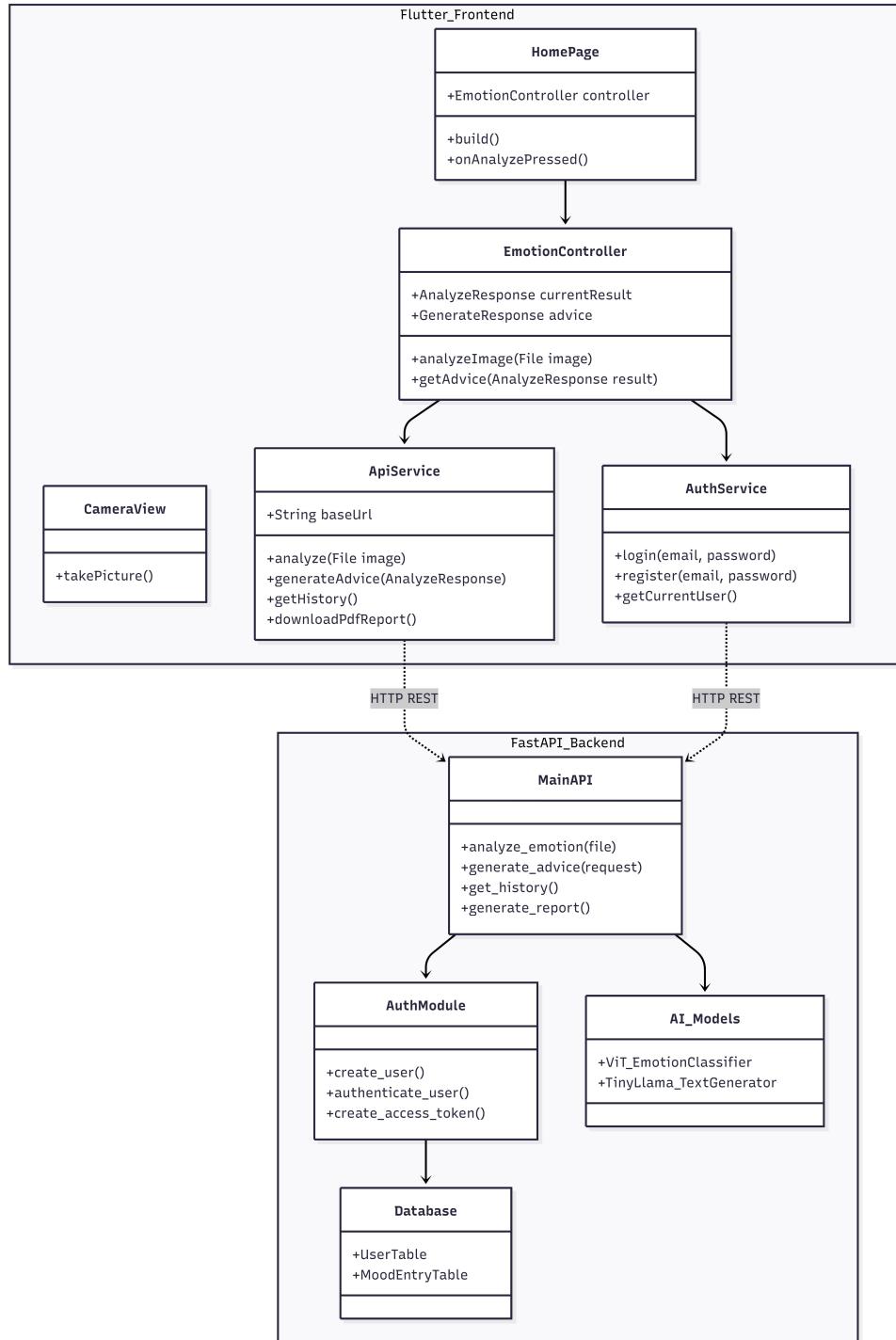


FIGURE 3.5 – Diagramme de classes simplifié du système

Le diagramme de classes présente les principales entités du backend : la classe User qui représente un utilisateur de l'application, la classe MoodEntry qui représente une

entrée d'historique émotionnel, ainsi que les services `AuthService`, `EmotionService` et `ReportService` qui encapsulent la logique métier (authentification, analyse d'image, génération de rapports).

3.3.2 Modèle relationnel

À partir du diagramme de classes, nous dérivons le modèle relationnel de la base de données SQLite. Les tables principales sont :

- **users**(*id*, email, hashed_password, created_at) ;
- **mood_entries**(*id*, user_id, emotion, confidence, created_at).

3.3.3 Dictionnaire de données

TABLE 3.1 – Dictionnaire de données (extrait)

Nom	Type	Taille	Oblig.	Défaut	Valeurs	Clé	Table
<i>id</i>	Integer		Oui	Auto-inc		PK	users
email	String	255	Oui		email	Unique	users
hashed_pwd	String	255	Oui				users
created_at	DateTime		Oui	Now()			users
emotion	String	50	Oui		happy, sad...		moods
confidence	Float		Non		[0,1]		moods
user_id	Integer		Oui			FK	moods

TABLE 3.2 – Dictionnaire de données

Table	Champ	Type	Description
users	id	Integer (PK)	Identifiant unique de l'utilisateur
	email	String	Email unique (identifiant de connexion)
	hashed_password	String	Mot de passe sécurisé (Bcrypt)
mood_entries	id	Integer (PK)	Identifiant de l'analyse
	user_id	Integer (FK)	Référence à l'utilisateur
	emotion	String	Label de l'émotion détectée
	confidence	Float	Score de confiance (0,0 à 1,0)

Ce tableau constitue le dictionnaire de données, une documentation essentielle qui décrit la structure et l'organisation des informations stockées dans la base de données de l'application. Il présente les deux tables principales du système : la table users, qui gère l'authentification et les comptes des utilisateurs avec des champs pour l'identifiant, l'email et le mot de passe chiffré ; et la table mood_entries, qui enregistre les résultats des analyses d'émotions faciales effectuées par le modèle d'intelligence artificielle, incluant l'émotion détectée, le score de confiance et une référence à l'utilisateur concerné. Ce document sert de référence technique pour les développeurs, garantissant une compréhension commune de l'architecture des données et facilitant la maintenance, l'évolution du système et la communication au sein de l'équipe de projet.

3.3.4 Architecture de l'application

Architecture logicielle

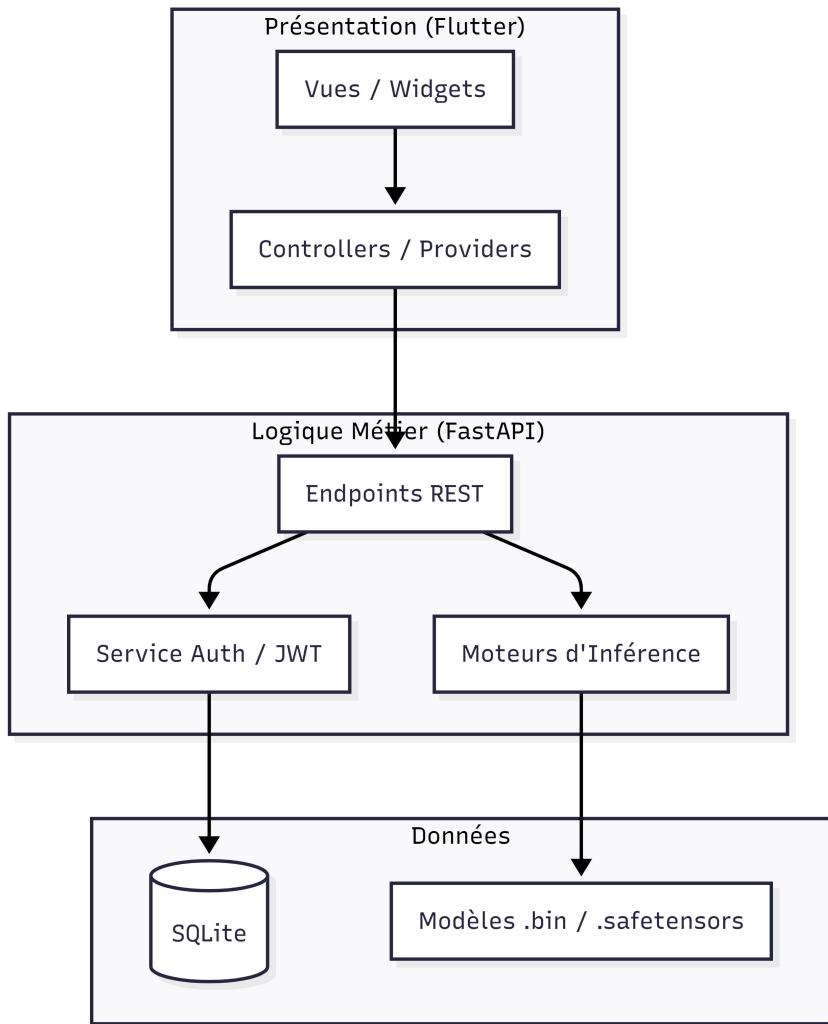


FIGURE 3.6 – Architecture logicielle du système

L'architecture logicielle repose sur une séparation claire entre le frontend Flutter, le backend FastAPI et la base de données SQLite. Le frontend communique avec le backend via des appels HTTP REST, tandis que le backend interagit avec la base de données via SQLAlchemy.

Architecture matérielle

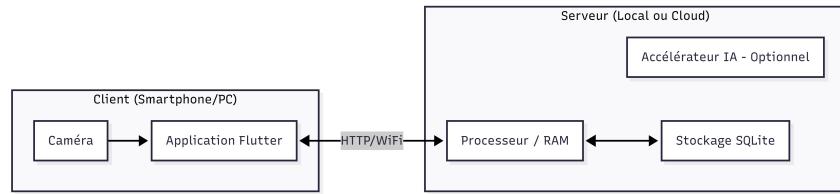


FIGURE 3.7 – Architecture matérielle (diagramme de déploiement)

L'architecture matérielle cible repose sur un poste utilisateur unique : l'application Flutter, le serveur FastAPI et la base SQLite sont exécutés localement et communiquent via HTTP sur `localhost`. Ce schéma peut être étendu ultérieurement à un déploiement sur serveur distant.

3.4 Conclusion

Dans ce chapitre, nous avons présenté la conception détaillée de notre système à l'aide de diagrammes UML dynamiques et statiques, ainsi que les architectures logicielle et matérielle retenues. Ces modèles constituent la base de la réalisation et de l'implémentation décrites dans le chapitre suivant.

Chapitre 4

Réalisation du système

4.1 Introduction

Dans ce chapitre, nous présentons la mise en œuvre concrète de l’application. Nous décrivons d’abord l’environnement de développement, les technologies et outils utilisés pour le backend et le frontend, puis nous illustrons la solution par les principales interfaces graphiques représentatives.

4.2 Environnement de développement et technologies utilisées

4.2.1 Environnement matériel

Le développement a été réalisé sur un poste de travail de type PC sous Windows, disposant d’un processeur moderne, de suffisamment de mémoire vive et d’un espace disque permettant d’installer les dépendances nécessaires (Python, Flutter, bibliothèques d’IA, etc.).

4.2.2 Technologies backend

Le backend de l’application repose principalement sur :

- **Python 3.x**;
- **FastAPI** pour l’API REST ;
- **Uvicorn** comme serveur ASGI ;
- **Transformers**, **Torch**, **Pillow** pour la partie IA et le traitement d’images ;
- **SQLAlchemy** et **SQLite** pour la base de données ;
- **Passlib[bcrypt]** et **python-jose** pour l’authentification (hachage + JWT) ;
- **FPDF** pour la génération de rapports PDF.

4.2.3 Technologies frontend

Le frontend est développé avec :

- **Flutter** (interfaces, navigation, widgets) ;
- le package **http** pour la communication avec le backend ; The backend of EmotiScan is powered by Firebase, providing a robust, scalable, and secure infrastructure to support the app's real-time emotional analysis features. Firebase Authentication securely manages user sign-up, login, and session handling, ensuring that each user's data remains private and accessible only to them. User-generated content—such as uploaded images, scan results, and emotional history—is stored in Cloud Firestore, a NoSQL database that enables fast, synchronized data access across devices and supports complex queries for generating weekly reports and historical trends. Additionally, Firebase Cloud Storage is used to securely store high-resolution images and generated PDF reports, while Firebase Cloud Functions can automate backend processes—such as triggering AI model analysis, generating insights, or compiling PDFs—without requiring client-side computation. This serverless architecture ensures low latency, real-time updates, and reliable performance, allowing EmotiScan to deliver a seamless, responsive experience while maintaining data integrity and user privacy
- **flutter_secure_storage** pour le stockage sécurisé du jeton JWT ;
- les plugins nécessaires pour l'accès à la caméra.

4.2.4 Outils de développement

- Un IDE (par exemple VS Code) ;
- **Git** pour la gestion de versions ;
- **PlantUML** pour la génération des diagrammes UML ;
- **LaTeX/Overleaf** pour la rédaction du rapport.

4.3 Principales interfaces graphiques

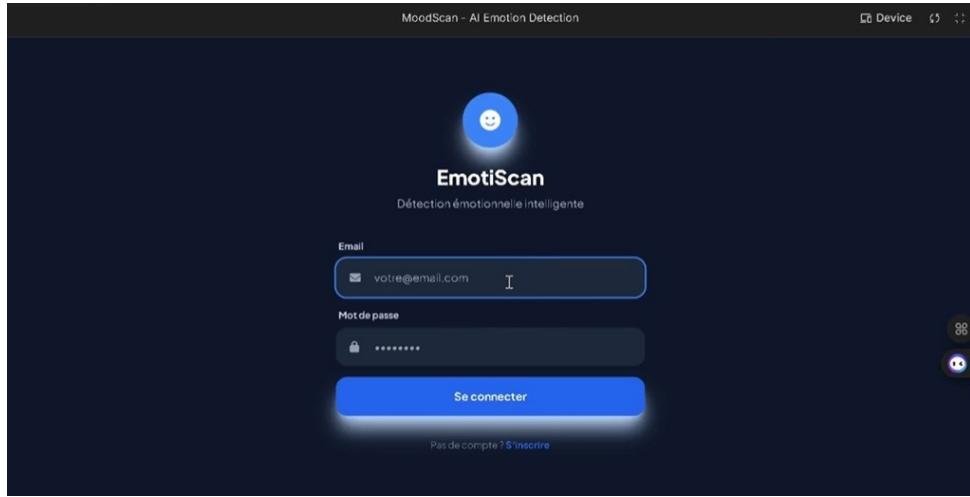


FIGURE 4.1 – Écran de connexion de l’application Flutter

Cette interface constitue le point d’entrée principal de l’application EmotiScan. Elle propose une expérience d’authentification épurée et intuitive, centrée sur la simplicité d’accès. L’utilisateur est invité à saisir son identifiant (une adresse e-mail pré-remplie « voice@email.com » est présentée à titre d’exemple) et son mot de passe masqué pour des raisons de sécurité. Le bouton « Se connecter » permet de valider les identifiants et d’accéder à l’espace personnel. Une mention « Pas de compte ? S’inscrire », placée stratégiquement sous le formulaire, guide les nouveaux utilisateurs vers le processus d’inscription, assurant ainsi une prise en main fluide quelle que soit leur familiarité avec l’application. Le titre « EmotiScan » et son sous-titre « Détection émotionnelle intelligente » positionnent clairement la promesse centrale du service : une analyse avancée des émotions via une technologie innovante.

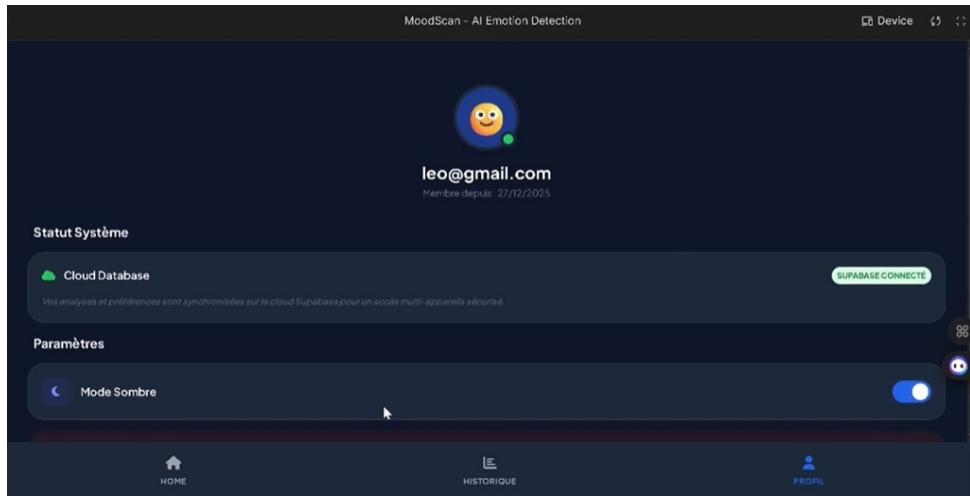


FIGURE 4.2 – Écran de de créer un compte Flutter

Cet écran offre une vue complète sur l'infrastructure technique et les préférences de l'application. Il confirme que toutes les données d'analyse et les profils utilisateurs sont synchronisés de manière sécurisée sur une base de données cloud (Supabase), garantissant ainsi une accessibilité multi-appareils et une sauvegarde fiable. La section « Paramètres » inclut des options personnalisables telles que l'activation du « Mode Sombre », permettant d'adapter l'interface visuelle au confort de l'utilisateur. En bas de l'écran, des informations de compte (adresse e-mail et date de début d'utilisation) ainsi qu'un indicateur visuel « SUPABASE CONNECTÉ » renforcent la transparence et la confiance quant à la sécurité et la disponibilité des données.

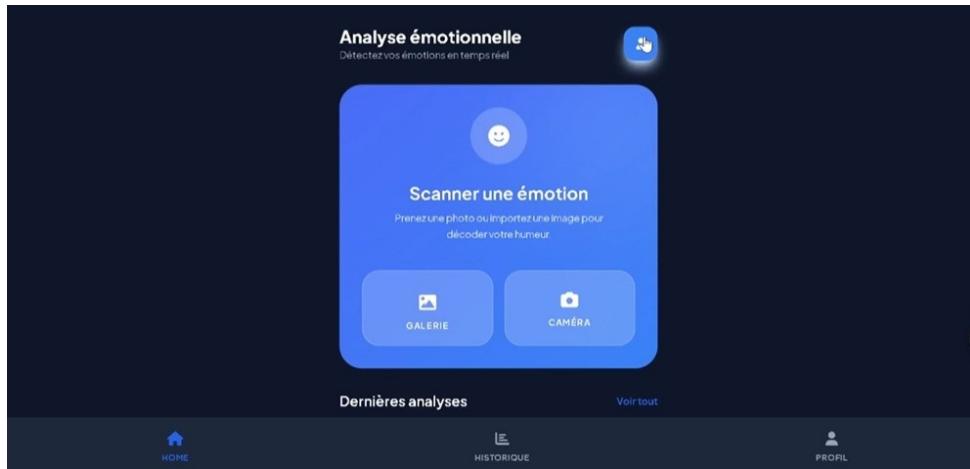


FIGURE 4.3 – Écran d’analyse d’émotion

Cette page est le cœur fonctionnel de l’application, dédiée à l’analyse immédiate des émotions. L’utilisateur est invité à « Scanner une émotion » en choisissant entre deux sources : importer une image depuis la « GALERIE » ou capturer une photo en direct via la « CAMÉRA ». L’interface met en avant la simplicité et la rapidité du processus, avec des boutons clairs et accessibles. La section « Dernières analyses » offre un aperçu rapide des scans précédents, et le bouton « Voir tout » redirige vers l’historique complet. Une barre de navigation fixe en bas de l’écran (HOME, HISTORIQUE, PROFIL) permet une circulation aisée entre les différentes sections de l’application.

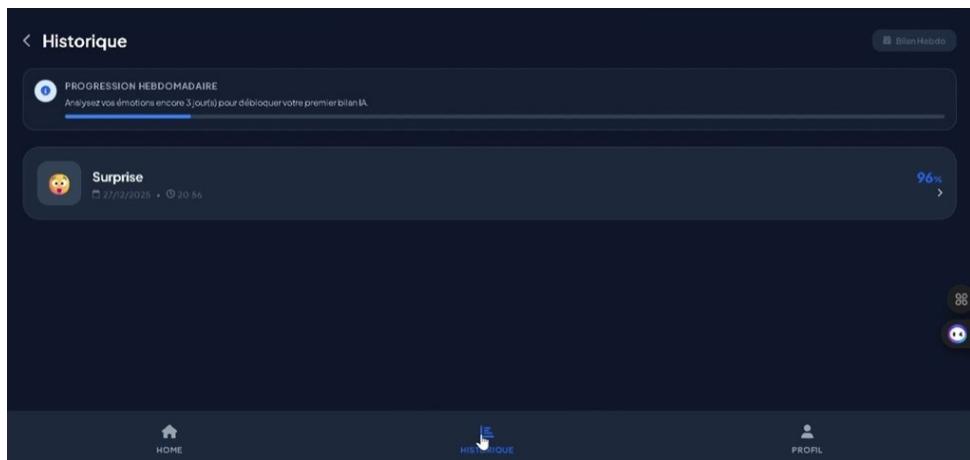


FIGURE 4.4 – Les analyses précédentes de l’utilisateur

L’écran d’historique fournit une perspective temporelle sur le parcours émotionnel de l’utilisateur. Il présente une « PROGRESSION HEBDOMADAIRE » sous forme de barre de complétion, avec un objectif incitatif : réaliser environ 3 analyses pour débloquer un premier bilan IA personnalisé. Un exemple concret d’analyse passée est affiché (l’émotion « Surprise » détectée le 2/12/2015 avec un score de confiance de 30), illustrant le format des données conservées. En complément, un « Bilan Hebdo » synthétise les tendances

émotionnelles sous forme de pourcentage (96) et de compteur (38 analyses), offrant ainsi une vue agrégée de l'état émotionnel sur la semaine.

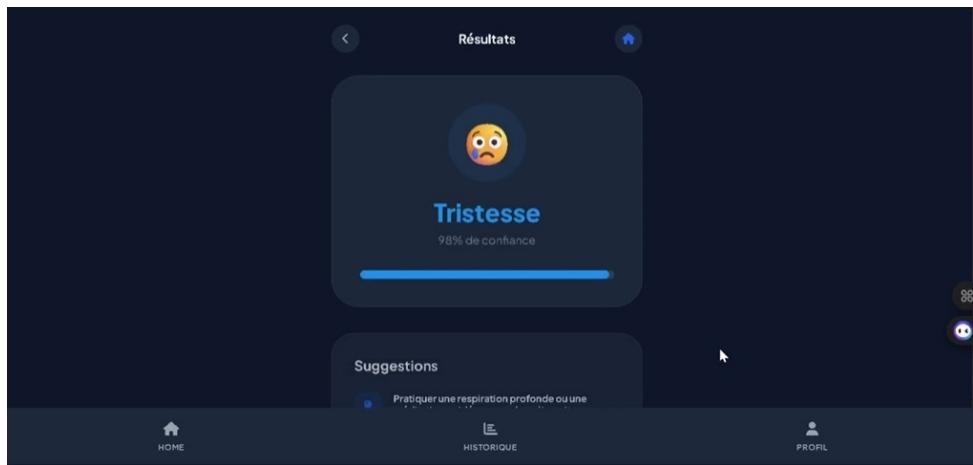


FIGURE 4.5 – Historique

Cet écran détaille les résultats d'une analyse émotionnelle précise, ici focalisée sur l'émotion « Tristesse » avec un niveau de confiance très élevé (98%). Il sert de point de départ vers des ressources d'accompagnement, en proposant immédiatement des « Suggestions » pour aider l'utilisateur à gérer cette émotion. Le design sépare clairement le résultat de l'analyse des recommandations, et la barre de navigation persistante permet de basculer rapidement vers l'historique ou le profil, ou de revenir à l'accueil. L'écran joue ainsi un double rôle : informatif (présentation du résultat) et orienté vers l'action (accès à des conseils pratiques).

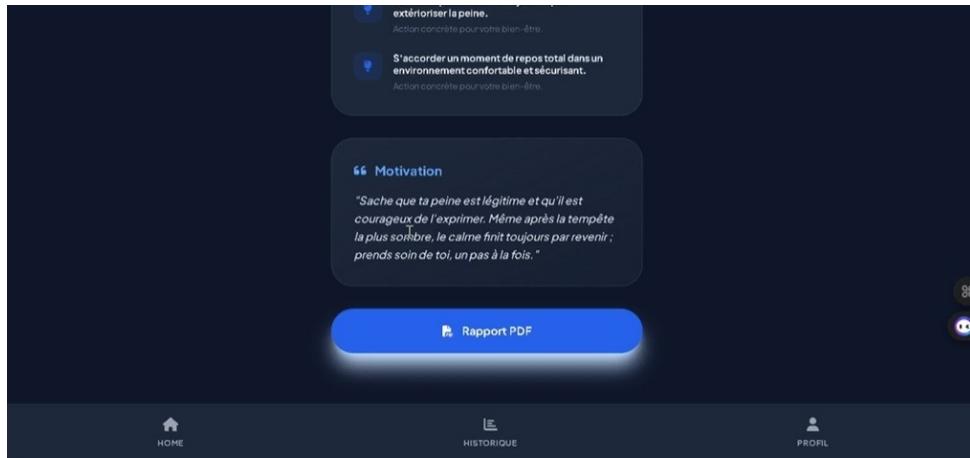


FIGURE 4.6 – Rapport émotionnel

Cet écran approfondit les recommandations amorcées précédemment en listant des actions concrètes pour favoriser le bien-être émotionnel. Les suggestions, comme « Pratiquer une respiration profonde » ou « Écrire ses pensées dans un journal », sont formulées de manière bienveillante et opérationnelle. Il s'enrichit surtout d'un message de soutien émotionnel, une « Motivation » écrite pour valider les sentiments de l'utilisateur (« ta peine est légitime ») et l'encourager avec bienveillance (« prends soin de toi, un pas à la fois »). La présence d'un bouton « Rapport PDF » en bas de l'écran permet de sauvegarder ou partager ce contenu sous un format structuré et portable.

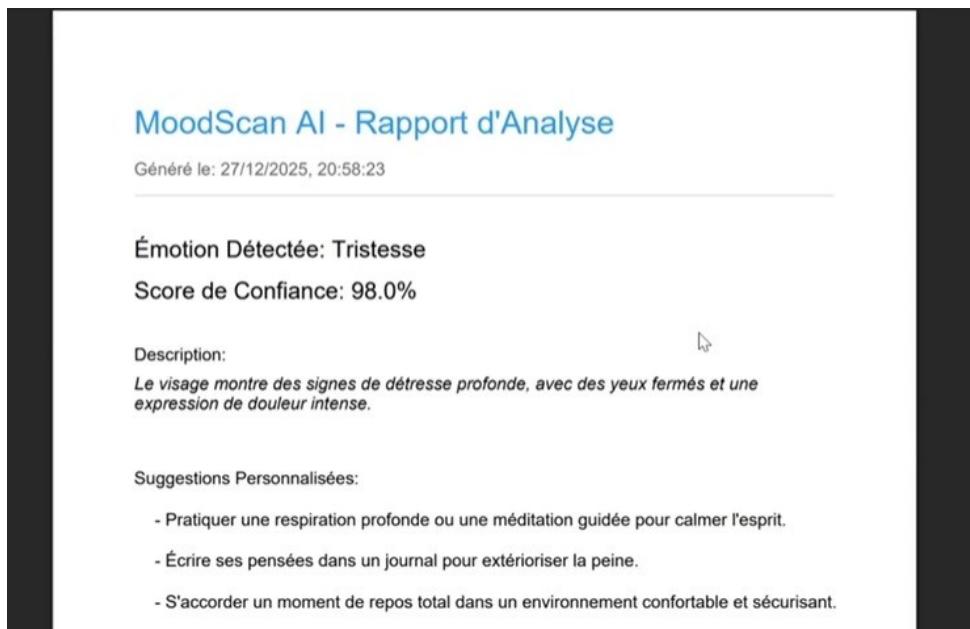


FIGURE 4.7 – Rapport émotionnel

Cette interface présente un exemple concret du rapport PDF généré par l'application, synthétisant de manière professionnelle une session d'analyse. Le rapport inclut des métadonnées précises (date et heure de génération), l'émotion détectée (« Tristesse ») avec

son score de confiance (98,0), et une description analytique du visage analysé. Les suggestions personnalisées sont reprises sous une forme listée et structurée, transformant les données de l'analyse en un document clair, organisé et potentiellement partageable avec un professionnel de santé ou conservé pour un suivi personnel. Il matérialise ainsi la valeur ajoutée d'EmotiScan : transformer une analyse instantanée en un outil de réflexion et de suivi à plus long terme.

4.4 Conclusion

Ce chapitre a présenté de manière détaillée l'environnement de développement ainsi que les principaux aspects de la réalisation logicielle de l'application. Nous avons décrit les technologies, les outils et les frameworks utilisés pour le développement du backend et du frontend, en mettant en évidence leur rôle dans la mise en œuvre de la solution proposée. À travers cette présentation, nous avons montré comment les choix de conception définis lors des phases précédentes ont été concrètement traduits en une application fonctionnelle, cohérente et conforme aux besoins exprimés. Cette étape de réalisation constitue ainsi un lien essentiel entre la conception théorique du système et son implémentation pratique.

Conclusion générale

Dans ce projet de fin d'année, nous avons conçu et réalisé une application de détection d'émotions faciales reposant sur un backend FastAPI et un frontend Flutter. Après avoir analysé l'existant et défini les besoins, nous avons proposé une architecture intégrant un modèle de classification d'expressions faciales, un module de génération de texte et un système d'authentification locale basé sur SQLite et JWT.

La solution obtenue permet à un utilisateur de capturer une image de son visage, d'obtenir une estimation de son émotion principale, ainsi que des conseils personnalisés et un message empathique. L'historique des analyses peut être consulté et exporté sous forme de rapport. Ce travail ouvre la voie à plusieurs perspectives, telles que l'amélioration des modèles d'IA utilisés, l'enrichissement des visualisations d'humeur ou encore le déploiement de l'application sur d'autres plateformes.

Bibliographie

- [1] Pascal Roques, *UML 2 par la pratique : Études de cas et exercices corrigés*, 5^e édition, 2006.
- [2] Site officiel FastAPI, <https://fastapi.tiangolo.com/>
- [3] Site officiel Flutter, <https://flutter.dev/>