# Département informatique et réseaux

Filière génie informatique et intelligence artificielle

# *Création d'une application mobile Android de détection des dangers et de demande d'aide*


**Réalisé par :**

AJGAGAL Asma

CHKIROU Aya

MESFAR Salma

YADAN Nohayla

**Encadré par :**

M. CHAHBOUNI Othmane



Année universitaire: 2024 / 2025

# Table de figures

# Table de matières

# *Dédicaces*

*À Dieu, pour sa guidance inébranlable tout au long de ce voyage. Nous Le remercions pour sa lumière qui a éclairé notre chemin et pour Sa grâce qui a rendu possible notre réussite. À nos chers parents : vos encouragements incessants, vos sacrifices et votre confiance en nous ont été les fondations sur lesquelles nous avons construit nos rêves. À nos professeurs et mentors, pour leur dévouement sans faille à nourrir nos esprits et à élargir nos horizons. Nous vous sommes reconnaissants pour votre précieuse influence sur nos vies. Que cette dédicace soit le reflet de notre profonde gratitude envers tous ceux qui ont contribué à notre succès. Avec tout notre amour et notre reconnaissance*

# *Remerciements*

*Nous souhaitons exprimer notre profonde gratitude à Monsieur* Othmane CHAHBOUNI *pour son soutien inébranlable et ses précieux conseils, qui ont été une source d'inspiration tout au long de ce projet. Sa bienveillance et son expertise ont été des piliers essentiels de notre succès. Son savoir approfondi et son dévouement ont été des atouts majeurs dans notre parcours académique. À nos familles et amis, nous adressons notre reconnaissance pour leur soutien constant et leurs encouragements sans faille. Cette page de remerciements témoigne de notre profonde gratitude envers tous ceux qui ont contribué à la réussite de ce projet*

# *Résumé*

Ce projet vise à développer une application mobile de sécurité permettant aux utilisateurs de partager instantanément leur position avec leurs proches en cas de danger, grâce à deux mécanismes d'activation intuitifs. En utilisant les capteurs du smartphone, l'application détecte un mouvement brusque (secousse) ou reconnaît des mots-clés vocaux prédéfinis pour déclencher automatiquement l'envoi des coordonnées GPS aux contacts de confiance préalablement enregistrés. Conçue avec Kotlin et Firebase, la solution intègre une interface minimaliste pour une prise en main immédiate, tout en garantissant la protection des données grâce à un système d'authentification sécurisé et un chiffrement des échanges. L'application s'adresse particulièrement aux personnes souhaitant disposer d'un outil discret et efficace pour alerter leur entourage en situation critique, combinant réactivité technologique et simplicité d'utilisation. Des améliorations futures pourraient inclure l'ajout d'un mode veille géolocalisé ou l'intégration avec les services d'urgence locaux pour une assistance renforcée.

Mots-clés : application mobile, sécurité, partage de localisation, Kotlin, Firebase, reconnaissance vocale, détection de mouvement, contacts d'urgence.

# *Introduction générale*

Les technologies mobiles jouent aujourd'hui un rôle essentiel dans notre quotidien, offrant des solutions innovantes pour répondre à des besoins urgents, notamment en matière de sécurité personnelle. Dans ce contexte, notre projet consiste à développer une application mobile dédiée à la protection des utilisateurs en leur permettant de partager instantanément leur localisation avec leurs proches en cas de situation critique. Cette application, développée en Kotlin avec Firebase, se distingue par son approche intuitive et discrète : elle s'active soit par un simple geste (secousse du téléphone), soit par une commande vocale prédéfinie, déclenchant automatiquement l'envoi des coordonnées GPS aux contacts de confiance enregistrés. L'objectif est de proposer un outil accessible, fiable et sécurisé, combinant réactivité technologique et simplicité d'utilisation pour rassurer les utilisateurs dans des circonstances délicates.

Ce rapport détaille les différentes étapes de conception et de réalisation du projet, structuré en quatre chapitres clés. Le premier chapitre présente une vue d'ensemble du projet, incluant ses objectifs, ses enjeux et les technologies mobilisées. Le deuxième chapitre aborde l'analyse des besoins et la conception technique, garantissant une architecture optimale. Le troisième chapitre décrit la mise en œuvre concrète des modèles de l'intelligence artificielle, le quatrième chapitre se concentre sur les technologies utilisées et le dernier chapitre se contente à représenter la réalisation de la plateforme.

# Chapitre 1

# Présentation générale sur le projet

## 1.1 Introduction approfondie

Dans un contexte où la sécurité personnelle devient une préoccupation majeure, notre projet d'application mobile se positionne comme une solution innovante et technologiquement avancée. L'idée centrale est de fournir aux utilisateurs un moyen discret et fiable de signaler une situation de danger en exploitant pleinement les capacités des smartphones modernes. Contrairement aux applications existantes qui nécessitent souvent une action manuelle explicite, notre approche combine intelligemment deux modes d'activation automatique : la détection de mouvement par accéléromètre et la reconnaissance de commandes vocales prédéfinies. Cette redondance calculée permet de couvrir un large éventail de scénarios d'urgence, depuis les agressions physiques jusqu'aux situations où la victime ne peut pas parler librement. Le développement s'appuie sur les dernières avancées en matière de traitement du signal mobile, d'intelligence artificielle embarquée et de technologies cloud, le tout intégré dans une interface utilisateur volontairement épurée pour garantir une accessibilité maximale. L'application est particulièrement destinée aux publics vulnérables (étudiants, travailleurs nocturnes, personnes vivant seules) mais aussi à tous ceux qui souhaitent disposer d'une solution de sécurité discrète et efficace au quotidien.

## 1.2 Objectifs techniques détaillés

Le cœur technique du projet repose sur trois piliers fondamentaux qui en garantissent la fiabilité et l'efficacité. Premièrement, le module de détection de mouvement intègre des algorithmes sophistiqués de traitement du signal capable de différencier une secousse intentionnelle des mouvements quotidiens normaux, avec une sensibilité réglable selon le profil utilisateur. Deuxièmement, le système de reconnaissance vocale utilise des modèles de machine learning optimisés pour fonctionner en temps réel sur mobile, même dans des environnements bruyants, tout en maintenant une faible consommation énergétique. Troisièmement, le mécanisme de partage de localisation implémente une synchronisation

intelligente avec Firebase permettant d'actualiser la position à intervalles variables selon la situation (mode haute fréquence en cas d'urgence détectée, mode économe en veille). Ces composants techniques sont complétés par des fonctionnalités secondaires mais essentielles comme le chiffrement bout-en-bout des communications, la journalisation sécurisée des événements, et un système de notification multi-canal (SMS, email, push) pour maximiser les chances que l'alerte soit reçue par les contacts désignés. L'ensemble est conçu pour maintenir un impact minimal sur l'autonomie de la batterie tout en garantissant une disponibilité permanente du service.

## 1.3 Contexte technique et marché

L'analyse du marché révèle une demande croissante pour des solutions de sécurité mobiles, avec des attentes particulières en matière de discrétion et de fiabilité. Notre étude comparative montre que les solutions existantes présentent souvent des lacunes soit dans la simplicité d'utilisation, soit dans les fonctionnalités offertes, soit dans leur modèle économique. Techniquement, le projet s'appuie sur un stack moderne combinant Kotlin pour le client mobile (avec une architecture modulaire favorisant la maintenance), Firebase comme backend serverless (offrant scalabilité et temps de développement réduit), et diverses API Android natives pour l'accès aux capteurs. Le diagramme de Gantt détaillé prévoit cinq phases majeures sur une période de six mois, depuis la recherche et conception jusqu'au déploiement progressif, avec des points de validation technique à chaque étape. Le plan de tests rigoureux inclut non seulement des tests unitaires et d'intégration classiques, mais aussi des simulations en conditions réelles avec des groupes d'utilisateurs tests, permettant de valider tous les scénarios d'usage dans des environnements variés (urbains, ruraux, intérieur/extérieur). Les métriques de performance suivies incluent non seulement des indicateurs techniques (latence, précision) mais aussi des mesures d'expérience utilisateur (facilité d'activation en situation de stress, clarté des feedbacks visuels).

## 1.4 Conclusion

Ce chapitre a posé les bases de notre application mobile de sécurité, en définissant ses objectifs principaux : offrir une solution fiable et discrète pour partager sa position en cas d'urgence via des déclencheurs intuitifs (mouvement et voix). Nous avons présenté l'architecture technique combinant Kotlin et Firebase, tout en soulignant l'importance accordée à la sécurité des données et à l'expérience utilisateur. La planification détaillée assure une mise en œuvre méthodique du projet. Ces éléments fondamentaux nous permettent désormais d'aborder sereinement les phases de conception et de développement approfondies qui feront l'objet des chapitres suivants.

# Chapitre 2

# Analyse et conception

## 2.1 Introduction

Après avoir défini les objectifs de ShakeUp et identifié les défis techniques liés à la détection des dangers et à l'envoi d'alertes automatisées, ce chapitre se consacre à la conception détaillée de notre solution. Cette étape est cruciale, car elle établit les fondations techniques et fonctionnelles sur lesquelles reposera le développement de l'application. Dans ce chapitre, nous allons présenter les diagrammes de cas d'utilisation et de classe en UML, afin d'expliquer les fonctionnalités que les utilisateurs peuvent avoir dans l'application ShakeUp et de décrire la base de données.

## 2.2 La modélisation en UML :

Avant de prolonger dans les détails de la conception de notre site web, il est important d'introduire d'abord l'outil qu'on a utilisé pour modéliser notre projet : L'UML (en anglais 'Unified Modeling Language' ou 'langage de modélisation unifié') est un langage de modélisation graphique. Ce langage est désormais la référence en modélisation objet, ou programmation orientée objet. Cette dernière consiste à modéliser des éléments du monde réel (immeuble, ingrédients, personne, logos, organes du corps...) ou virtuel (temps, prix, compétence...) en un ensemble d'entités informatiques appelées « objet ». Il est constitué de diagrammes qui servent à visualiser et décrire la structure et le comportement des objets qui se trouvent dans un système.

## 2.3 Diagramme de cas d'utilisation :

Le diagramme de cas d'utilisation est un outil de modélisation UML qui permet de représenter les interactions entre les acteurs et un système. Il est utilisé pour décrire les différentes actions que les acteurs peuvent effectuer sur le système et comment le système répond à ces actions. Le diagramme de cas d'utilisation est particulièrement utile pour les analystes et les concepteurs qui cherchent à comprendre les exigences du système et à concevoir son architecture.

Ce diagramme de cas d'utilisation illustre les interactions principales entre l'utilisateur et l'application de sécurité. L'utilisateur peut créer un compte, gérer son profil, consulter l'historique des dangers, choisir un mot SOS personnalisé ou gérer ses contacts de confiance. Une fois authentifié, il peut déclencher une alerte de danger par détection vocale (mot SOS) ou par secousse du téléphone. Ce déclenchement d'alerte peut être suivi, de manière conditionnelle (<<extend>>), par une confirmation de l'utilisateur indiquant s'il est réellement en danger ou non. En l'absence de réponse dans un délai défini, une réponse automatique est envoyée pour garantir la réactivité du système. Ce diagramme met ainsi en évidence les fonctionnalités essentielles liées à la sécurité proactive de l'utilisateur.

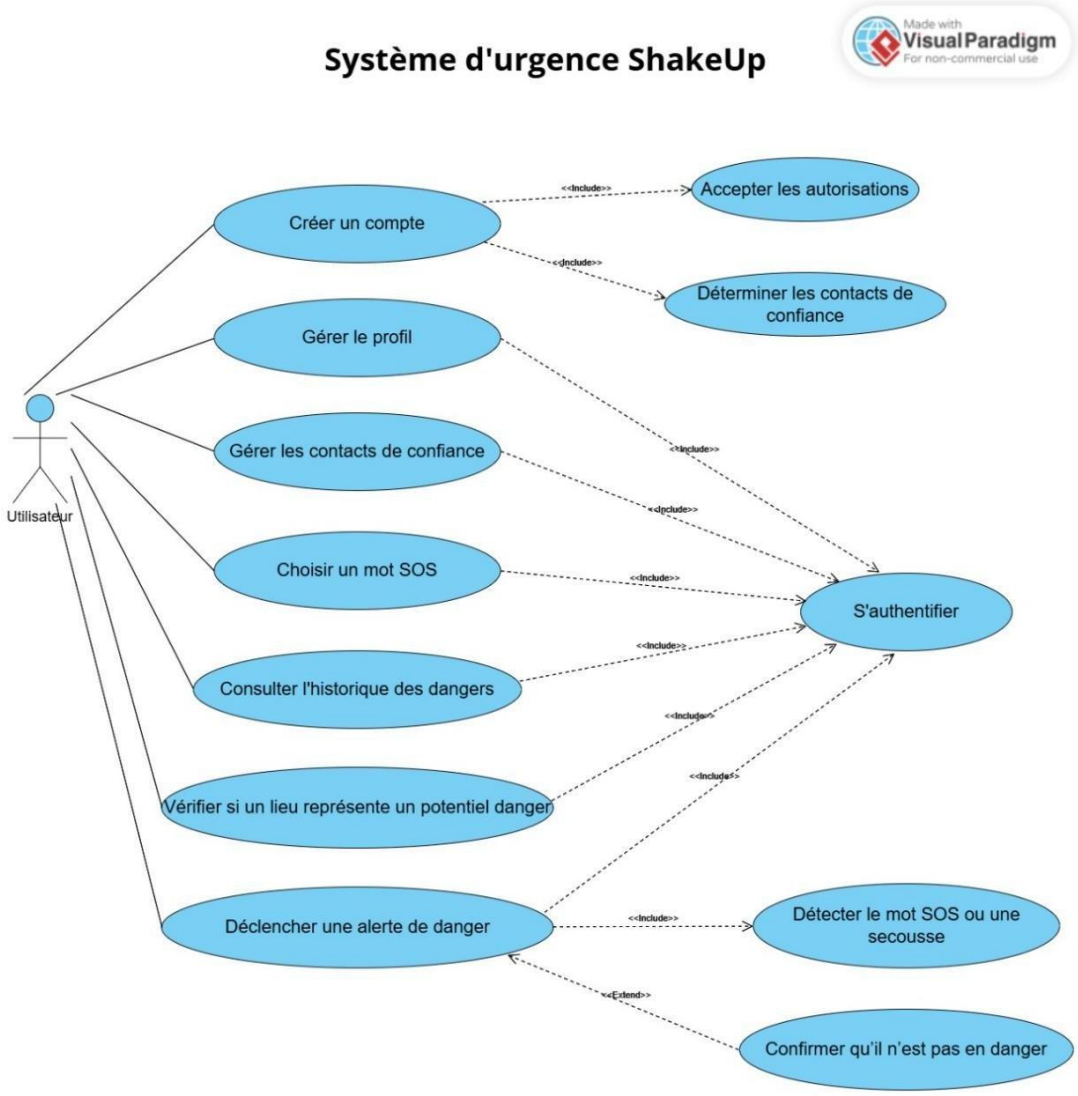



*Figure 1: Diagramme de cas d'utilisation*

## 2.4 Diagramme de classe :

Le diagramme de classe est un outil de modélisation UML qui permet de représenter les classes, les interfaces, les attributs et les relations entre les différents éléments d'un système. Il est utilisé pour décrire la structure du système et la manière dont les différents composants interagissent entre eux.

Ce diagramme de classe présente la structure générale des principales données gérées dans l'application. Chaque utilisateur est identifié par un identifiant unique et peut ajouter plusieurs contacts de confiance, nécessaires en cas d'alerte. Il peut également configurer un mot SOS personnalisé, limité à un seul mot actif par utilisateur. L'utilisateur peut également consulter un historique des dangers enregistrés, qui contient des informations associées à chaque situation détectée. Ce modèle relationnel permet d'organiser les données de manière cohérente afin de faciliter leur exploitation par les différentes fonctionnalités de l'application.

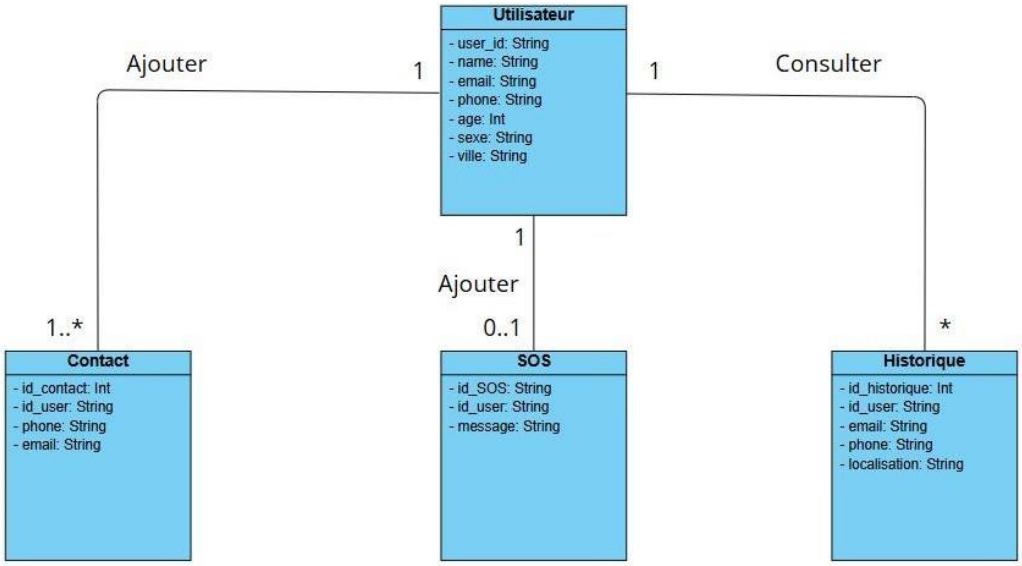



Figure 2: Diagramme de classe

## 2.5 Conclusion :

Ce chapitre a permis d'établir les bases conceptuelles de ShakeUp à travers une modélisation UML complète (diagrammes de cas d'utilisation et de classes), une conception détaillée des interfaces utilisateur et une architecture technique optimisée. Ces éléments structurants garantissent la

cohérence du système avant l'intégration des modèles d'intelligence artificielle, qui fera l'objet du prochain chapitre, où nous détaillerons leur implémentation, leur entraînement et leur optimisation pour répondre aux besoins spécifiques de détection de dangers.

# Chapitre 3

# Modèles d’intelligence artificielle

## 3.1 Introduction :

Dans ce chapitre, nous présentons les mécanismes analytiques du système intelligent de ShakeUp, là où la technologie rencontre les besoins concrets de sécurité. Nous avons développé deux mécanismes clés qui fonctionnent en tandem : un module d'analyse vocale et un système d'évaluation géolocalisée.

## 3.2 Modèle 1 : Reconnaissance Vocale (Audio → Texte)

Pour identifier des mots-clés comme « aidez-moi, help, danger » en cas d'urgence, nous avons choisi le modèle Whisper-small de la bibliothèque Hugging Face transformers. Ce modèle pré-entraîné par OpenAI, est spécialisé dans la reconnaissance vocale automatique (Automatic Speech Recognition, ASR).

Parmi les différentes versions disponibles, nous avons exclu les modèles les plus puissants comme le modèle Whisper-large-v3-turbo – bien qu'offrant une précision supérieure – pour privilégier un équilibre optimal entre performance, rapidité et légèreté, essentiel pour une application mobile.

Comparaison des modèles et choix technique :

| Critère | Whisper-small | Whisper-large-v3-turbo |
|---|---|---|
| Taille du modèle | ~1.5 Go | ~10 Go |
| Vitesse | Rapide(temps réel sur mobile) | Lente (requiert GPU) |
| Précision | Suffisante pour mots-clés | Excellence(transcription complexes) |
| Ressources nécessaires | Faible consommation CPU/GPU | Nécessite des serveurs dédiés |

*Schéma d'Intégration :*

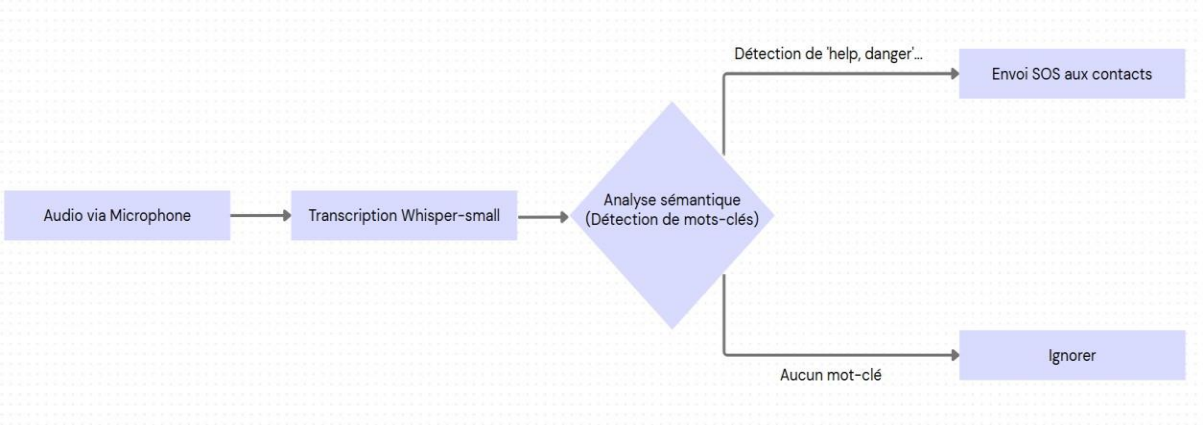



Figure 3: Schéma d'intégration du modèle Whisper-small

*Fonctionnement dans ShakeUp :*

1. **Capture audio :** Le microphone du smartphone enregistre l'environnement lorsque l'utilisateur active manuellement la fonction ou lors de vibrations violentes (détectées par l'accéléromètre).
2. **Transcription :** Whisper convertit l'audio en texte.

3. **Détection de mots-clés :** Un filtre simple identifie les termes prédéfinis (aide, danger, etc).
4. **Action :** Si un mot-clé est détecté, l'application envoie automatiquement un message SOS aux contacts sélectionnés, incluant la localisation GPS.

## 3.3 Modèle 2 : Détection de Lieux Dangereux

*Objectif :*

Ce modèle vise à prédire si une zone géographique est dangereuse (1) ou sûre (0) en fonction :

- Localisation (LAT, LON)
- Heure de la journée
- Âge et genre de l'utilisateur

*Approche et Méthodologie :*

### 1. Préparation des Données :

**Source des données :** Base de données simulée manuellement avec des cas réels (ex : Agadir, Safi, Essaouira …).
Variables utilisées :

**Variables indépendantes (X) :**

- Vict Age (Âge)
- Vict Sex (Genre, encodé en M=1, F=0)
- Heure (Heure de la journée, 0-23)

- LAT, LON (Coordonnées GPS)

**Variable cible (y) :** danger (0 = sûr, 1 = dangereux)

**Nettoyage :**

- Suppression des colonnes non pertinentes (jour, mois, LOCATION).

## 2. Séparation des Données :

Pour évaluer objectivement les performances du modèle, on divise les données en deux parties :

- Jeu d'entraînement (70%) : Pour apprendre les patterns.
- Jeu de test (30%) : Pour simuler des nouvelles données et vérifier si le modèle généralise bien.

```python
import pandas as pd
from sklearn.model_selection import train_test_split

# Charger ton dataset
df = pd.read_csv("geo_safety_data.csv")

# Supprimer les colonnes inutiles
df.drop(columns=['jour', 'mois', 'LOCATION'], inplace=True)

# Filtrer les lignes avec une heure valide
df = df[(df['heure'] >= 0) & (df['heure'] <= 23)]

X = df[['Vict Age', 'Vict Sex', 'heure', 'LAT', 'LON']]
y = df['danger']

# Encoder la variable 'Vict Sex' en numérique
X['Vict Sex'] = X['Vict Sex'].map({'M': 1, 'F': 0})

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Figure 4: Préparation des données pour le modèle de prédiction de danger

## 3. Méthodologie de test :

Plusieurs algorithmes d'apprentissage automatique ont été évalués pour déterminer celui offrant les meilleures performances dans la prédiction des zones dangereuses. Les modèles ont été entraînés et testés sur le même jeu de données prétraitées, avec une répartition de 80% pour l'entraînement et 20% pour le test.

Le tableau suivant présente les performances (précision) des différents modèles testés :

| Modèle | Accuracy (Précision) |
|---|---|
| Random Forest | 95.5% |
| KNN (K-Nearest Neighbors) | 83.8% |
| Régression Logistique | 70.3% |
| SVM (Machine à Vecteurs de Support) | 64.4% |

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
model_rf = RandomForestClassifier(random_state=42)
model_rf.fit(X_train, y_train)
y_pred_rf = model_rf.predict(X_test)
print("Accuracy (Random Forest):", accuracy_score(y_test, y_pred_rf))
```

```
Accuracy (Random Forest): 0.954954954954955
```

Figure 5: Implémentation et évaluation du modèle Random Forest

Au terme de cette analyse comparative, le modèle Random Forest a été retenu pour sa performance supérieure (95.5% de précision) et sa robustesse face aux données géolocalisées, offrant ainsi un équilibre optimal entre fiabilité et efficacité pour la détection des zones dangereuses.

## 4. Limites du modèle de détection des lieux dangereux :

Bien que le modèle Random Forest ait démontré une haute précision (95.5%), certaines limitations doivent être mentionnées :

- **Problème de généralisation :**

Le modèle peut produire des prédictions imprécises pour :

- Les lieux absents du dataset d'entraînement (ex: nouvelles zones urbaines non répertoriées)
- Les cas où les caractéristiques (heure, localisation) ne correspondent pas aux motifs appris

- **Biais temporel :**

Une tendance a été observée :

- Le modèle associe systématiquement la nuit (après 22h) à un risque élevé

Même pour des zones sécurisées non couvertes par les données d'entraînement

- **Solutions envisagées :**

Pour atténuer ces limites il faut :

- Collecter continue de données terrain

- Ajouter d'un système de feedback utilisateur

## 3.4 Conclusion :

Ce chapitre a permis de mettre en place les mécanismes intelligents au cœur de ShakeUp, avec un focus particulier sur leurs forces et limites opérationnelles. Le module de reconnaissance vocale (Whisper-small) et le système de prédiction géolocalisée (Random Forest) offrent ensemble une détection fiable des situations à risque, tout en nécessitant des améliorations continues pour mieux gérer les cas limites et les nouvelles données.

Le chapitre suivant présentera l'écosystème technologique qui soutient ces fonctionnalités - outils de développement, frameworks et infrastructures cloud - essentiels pour une application mobile robuste et évolutive.

# Chapitre 4

# Les technologies utilisées

## 4.1 Introduction :

Dans ce chapitre, nous présentons l'écosystème technologique qui a permis de concrétiser l'application ShakeUp. Le choix des langages de programmation, frameworks et outils d'intégration a été déterminant pour répondre aux exigences spécifiques de notre système de détection d'urgences. Nous détaillerons ici chaque composant clé de notre stack technique, en expliquant comment ces technologies s'articulent pour créer une application mobile à la fois performante et sécurisée.

## 4.2 Les langages et les Frameworks :

### *4.2.1 Kotlin:*

Kotlin est un langage de programmation moderne, statiquement typé et conçu pour être entièrement interopérable avec Java. Développé par JetBrains, il vise à offrir une expérience de codage améliorée avec un code concis, expressif et sûr. C'est le langage privilégié pour le développement d'applications Android. Nous allons utiliser kotlin pour le développement de l'application mobile.

Figure 6: Kotlin

### *4.2.2 Python :*

Python est un langage de programmation interprété, open source, multiplateformes et orienté objet. Grâce à des bibliothèques spécialisées, Python s'utilise pour de nombreuses situations comme le développement logiciel, l'analyse de données, la gestion d'infrastructures, l'intelligence artificielle et l'apprentissage automatique.

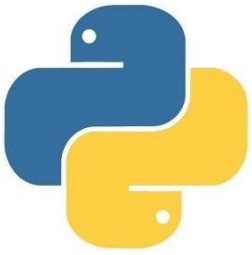


Figure 7: Python

### *4.2.3 FastAPI :*

Figure 8: FastAPI

FastAPI est un framework web Python moderne et performant, spécialisé dans la création d'APIs REST. Il combine vitesse élevée (grâce à ASGI), validation automatique des données via Pydantic, et génération de documentation interactive (OpenAPI/Swagger), ce qui en fait un outil idéal pour déployer rapidement des services backend fiables - comme nous l'avons utilisé pour exposer nos modèles d'analyse vocale et géolocalisée dans ShakeUp.

## 4.3 Les Logiciels utilisés :

### *4.3.1 Visual Studio Code :*

Figure 9: Visual Studio Code

Visual Studio Code est développé par Microsoft et est un éditeur de code extensible compatible avec les systèmes d'exploitation Windows, Linux et macOS3. Cet outil offre une multitude de fonctionnalités pratiques telles que le débogage, la coloration syntaxique, l'auto complétion intelligente du code (IntelliSense4.), les extraits de code, la refonte du code, ainsi que l'intégration de Git.
Les utilisateurs ont la possibilité de personnaliser le thème, les raccourcis clavier, les préférences, et peuvent également installer des extensions qui ajoutent des fonctionnalités supplémentaires à l'éditeur.

### *4.3.2 Android Studio :*

Figure 10: Android Studio

Android Studio est l'environnement de développement intégré (IDE) officiel des applications Android. Basé sur le puissant outil de développement et d'édition de code d'IntelliJ IDEA, Android Studio offre encore plus de fonctionnalités qui améliorent votre productivité lorsque vous créez des applications Android.

## 4.4 Services Cloud :

### *4.4.1 Firebase :*

Figure 11: Firebase

Firebase est un ensemble d'outils pour l'hébergement et le développement d'applications mobiles et web, qui permet l'envoi de notifications et de publicités, la remontée des erreurs et des clics effectués dans l'application. Firebase est le cloud utilisé pour le stockage des données.

## 4.5 Conclusion:

Dans ce chapitre, nous avons exploré les technologies essentielles qui ont soutenu le développement de notre projet. Nous avons détaillé les langages de programmation et frameworks utilisés. Nous avons également couvert les outils et logiciels.

# Chapitre 5

# Mise en œuvre

## 5.1 Introduction :

Ce chapitre présente la réalisation concrète de ShakeUp, illustrant comment les concepts théoriques et les choix techniques se matérialisent dans l'application finale. À travers des captures d'écran commentées, nous détaillons le fonctionnement des principales fonctionnalités : la détection d'urgence (par secousse ou voix), l'envoi d'alertes automatisées, et l'évaluation des zones à risque. Ces éléments démontrent l'adéquation entre notre solution technique et les besoins réels de sécurité des utilisateurs, validant ainsi l'ensemble des étapes de conception et de développement précédentes.

## 5.2 Activity Splash :

C'est la première interface affichée lors du lancement de l'application. Elle joue un rôle esthétique et introductif, en présentant le logo et le nom de l'application de manière centrée à l'écran.


Figure 12: Activité Splash

## 5.2 Activity Affichage :

Cette interface offre une brève introduction à la finalité et aux fonctionnalités clés de l'application.

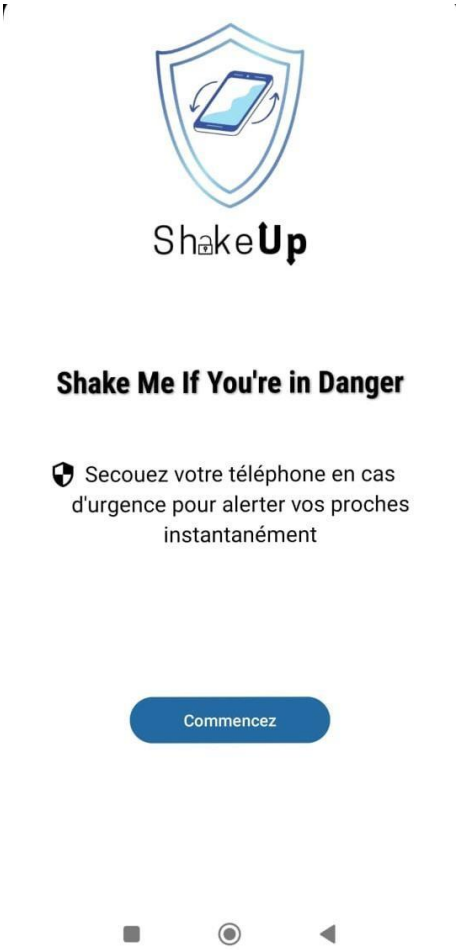



Figure 13: Activité affichage

## 5.3 Activity Regiter :

Cette activité permet à l'utilisateur de créer un compte en suivant un processus en 3 étapes successives. Chaque étape doit être complétée avant de passer à la suivante, garantissant ainsi la validité et la complétude des informations fournies.

### *5.3.1 Etape 1 :*

L'utilisateur saisit ses informations de base telles que le nom, le prénom et l'adresse email, etc. Ces données sont essentielles pour créer un profil unique.

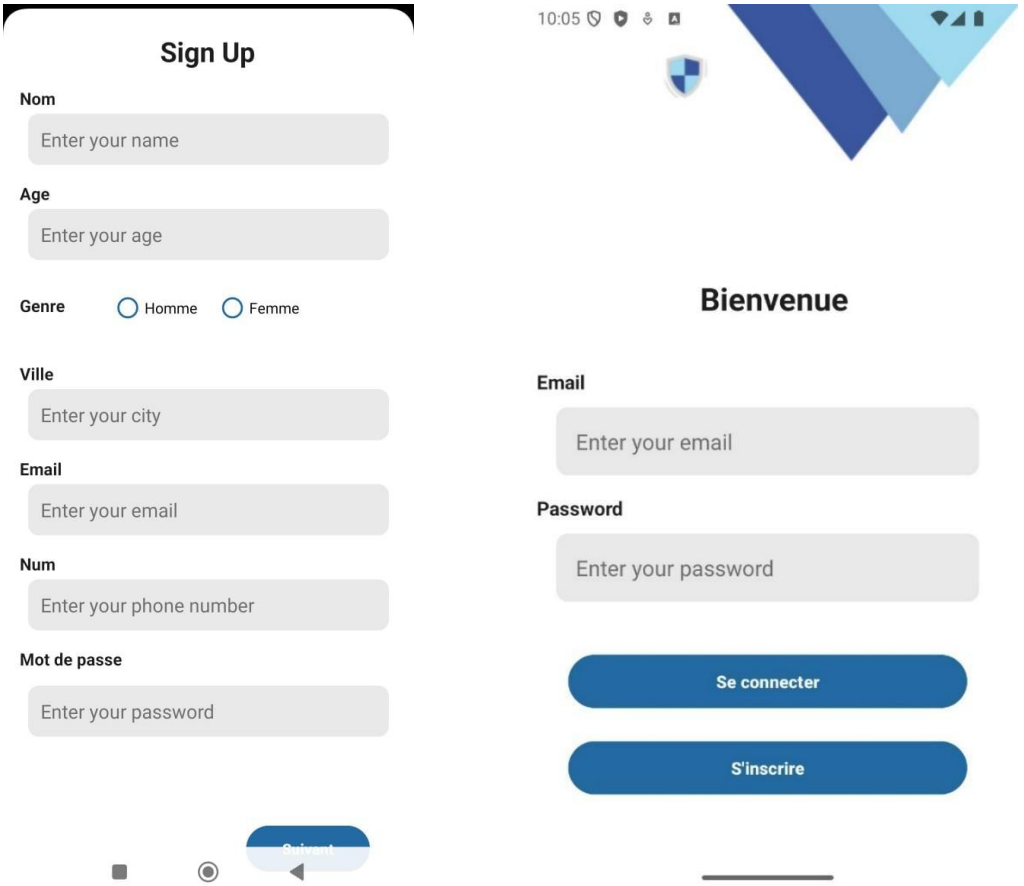



Figure 14: Register et login

Ainsi que l'utilisateur peut après avoir s'inscrit il peut se connecter automatiquement en garant sa session.

Cette étape est cruciale car elle initialise les données critiques de chaque utilisateur elle doit être assurer par une sécurité qui permettra à l'utilisateur de nous confier et utiliser notre application pour assurer aussi sa sécurité.

### *5.3.2 Etape 2 :*

Dans cette étape, l'utilisateur doit ajouter un ou plusieurs contacts de confiance. Ces contacts seront utilisés dans les situations d'urgence pour envoyer automatiquement des SMS SOS si l'utilisateur est en danger.

Cela permet à l'application de réagir rapidement et de prévenir des proches.

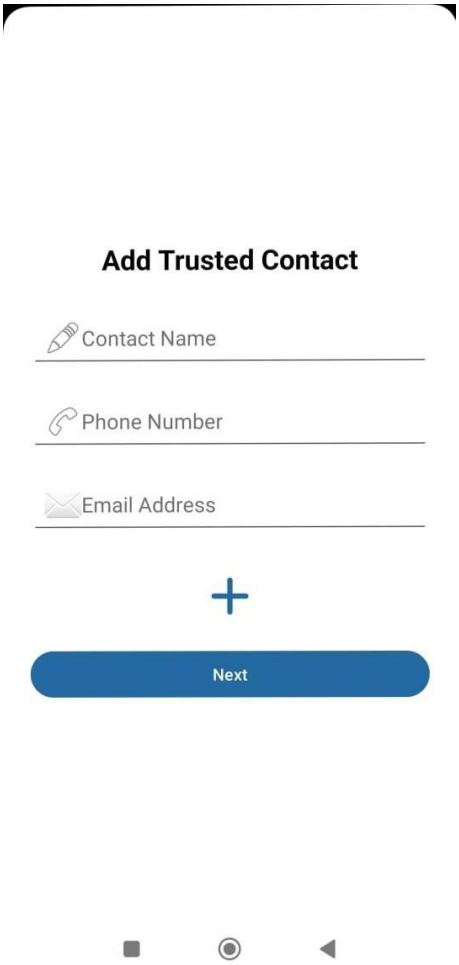



Figure 15: Ajouter contacts de confiance

L'utilisateur ici doit forcément ne pas rater cette étape car c'est par le biais de cette dernière les contacts de confiance c'est eux après avoir réagir auront la clé pour avoir aidé la personne en danger n'importe qu'il est.

### *5.3.3 Etape 3 :*

Dans cette dernière étape, l'utilisateur est invité à accorder les autorisations nécessaires au bon fonctionnement de l'application. Ces autorisations permettent d'assurer une réactivité optimale en cas de situation d'urgence.


Figure 16: Page d'autorisation

## 5.4 DialogFragement SOS :

Ce composant est une boîte de dialogue qui permet à l'utilisateur de sélectionner un mot-clé vocal parmi une liste prédéfinie pour déclencher une alerte en cas de danger ; le mot choisi est enregistré et peut être modifié ultérieurement.

Pour bien clarifier c.-à-d. que lorsqu'il enregistre le mot clé il doit créer un bruit pour que le modèle détecte une voix après 6s il doit dire le mot pour que la localisation s'envoie.

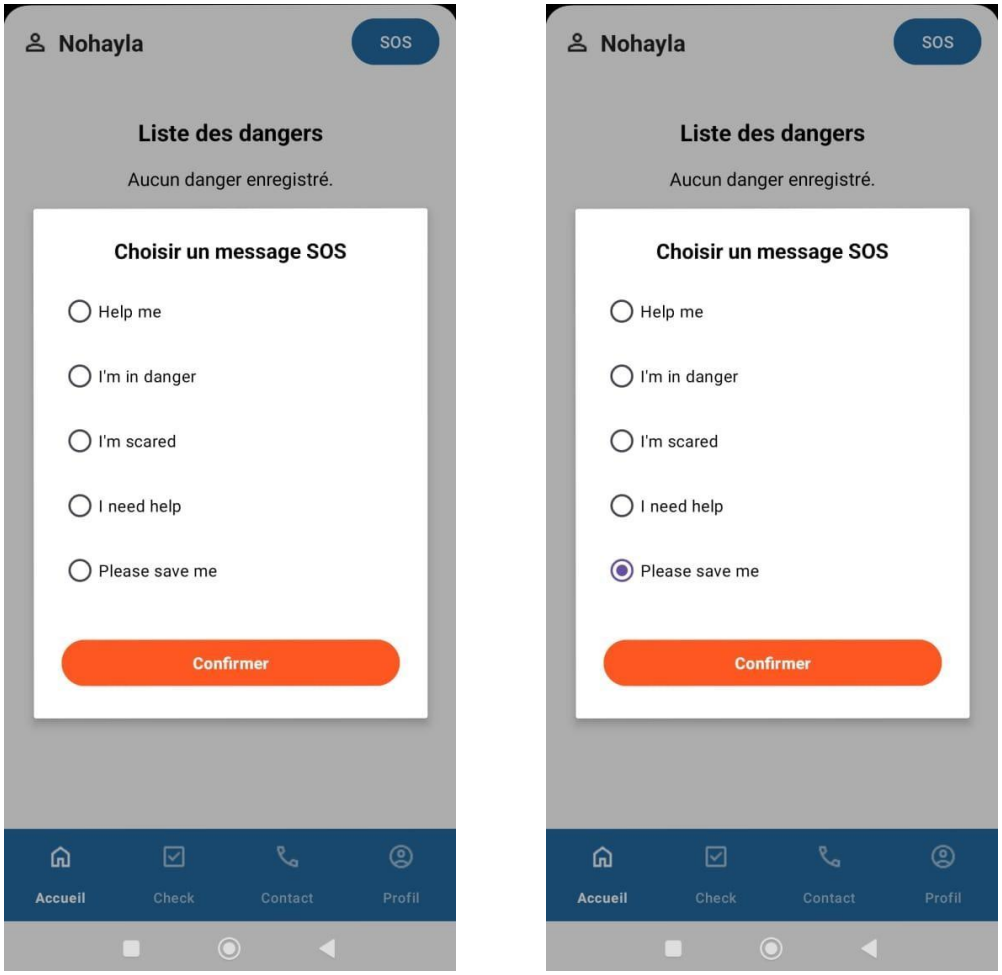
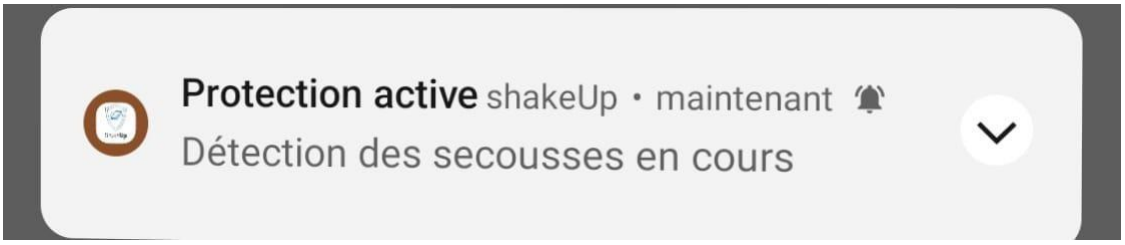



Figure 17: Fragment SOS

## 5.5 Fragment Historique

Après connexion, le service d'accéléromètre s'active pour suivre les mouvements de l'utilisateur et déclencher une alerte confirmant l'absence de problème et le bon fonctionnement du capteur.




Cette interface enregistre la date et la position de l'utilisateur dès qu'un danger est signalé ; si aucune alerte n'a encore été déclenchée, elle affiche "aucun danger enregistré".


Figure 18: Page d'historique


Une fois l'accéléromètre détecte un mouvement conforme aux seuils définis, une notification interroge l'utilisateur : "Êtes-vous en danger ?" — si oui, un SMS standard avec sa position GPS est immédiatement envoyé aux contacts de confiance ; si non, rien ne se passe ; sans réponse au bout de 1 minute, l'utilisateur est considéré en danger et le SMS est envoyé automatiquement.

Lorsqu'on clique sur « Oui », la date et la localisation (affichables en lien) sont enregistrées dans une table.

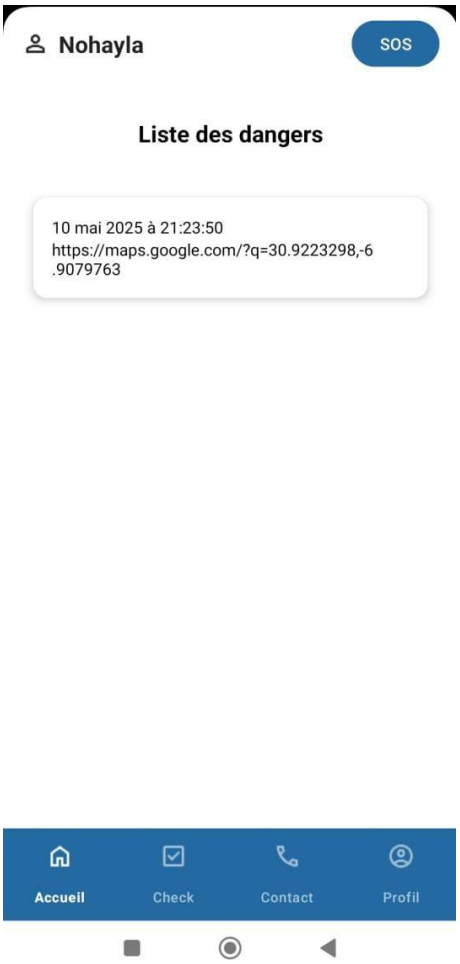



Figure 19: Liste des dangers

## 5.6 Fragment Check :

L'utilisateur a la possibilité de vérifier si un lieu représente un potentiel danger, il choisit la localisation dans une map, puis il choisit l'heure de visite et clique sur le bouton check safety. Les données nécessaires sont alors envoyées au model AI comme des entrées, la réponse du model sera soit que c'est un lieu danger ou non, et la réponse s'affiche à l'utilisateur.

Lorsque l'utilisateur sélectionne un emplacement sur la carte et spécifie une plage horaire, ces données sont transmises à un modèle machine learning hébergé sur Firebase. Ce modèle, préalablement entraîné sur des datasets d'incidents historiques géolocalisés et de données contextuelles (éclairage public, densité de population, jours fériés), analyse plusieurs facteurs de risque : le modèle extrait d'abord les caractéristiques spatiales (quartier, type de zone) et temporelles (heure, jour de semaine), puis les croise avec des indicateurs de sécurité (taux de criminalité local, proximité de services d'urgence) pour générer une prédiction.

Concernant le modèle de voix, si l'utilisateur accepte l'accès de notre application à son microphone, une notification permanente s'affiche pour indiquer que le service de détection de voix est actif, c'est une manière d'indiquer à l'utilisateur que son microphone est en cours d'écoute (Safety monitoring active). L'application ShakeUp n'enregistre pas le son tous le temps, elle détecte quand l'utilisateur est en cours de parler puis elle enregistre un audio de 6 secondes et l'envoie avec le mot SOS choisis par l'utilisateur à fastAPI, l'intermédiaire entre ShakeUp et le modèle de transcription de voix vers texte, qui ensuite transforme l'audio en texte et le compare avec le mot SOS indiqué, si l'utilisateur a bien dit son mot SOS, une notification s'affiche chez l'utilisateur pour confirmer s'il est en danger ou non, après 30 secondes si l'utilisateur ne répond pas une auto-réponse s'envoie pour confirmer qu'il est en danger, et des SMS avec son localisation s'envoie à leurs contacts de confiance.

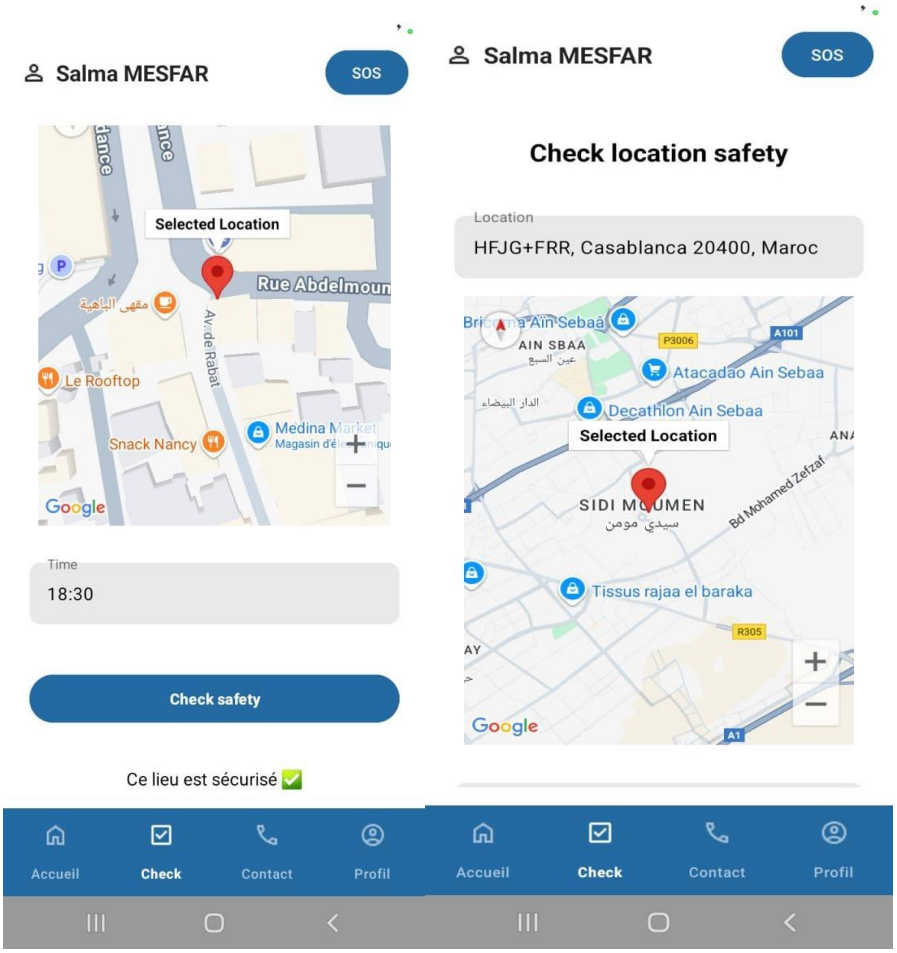



Figure 20: Check safety : lieu sécurisé

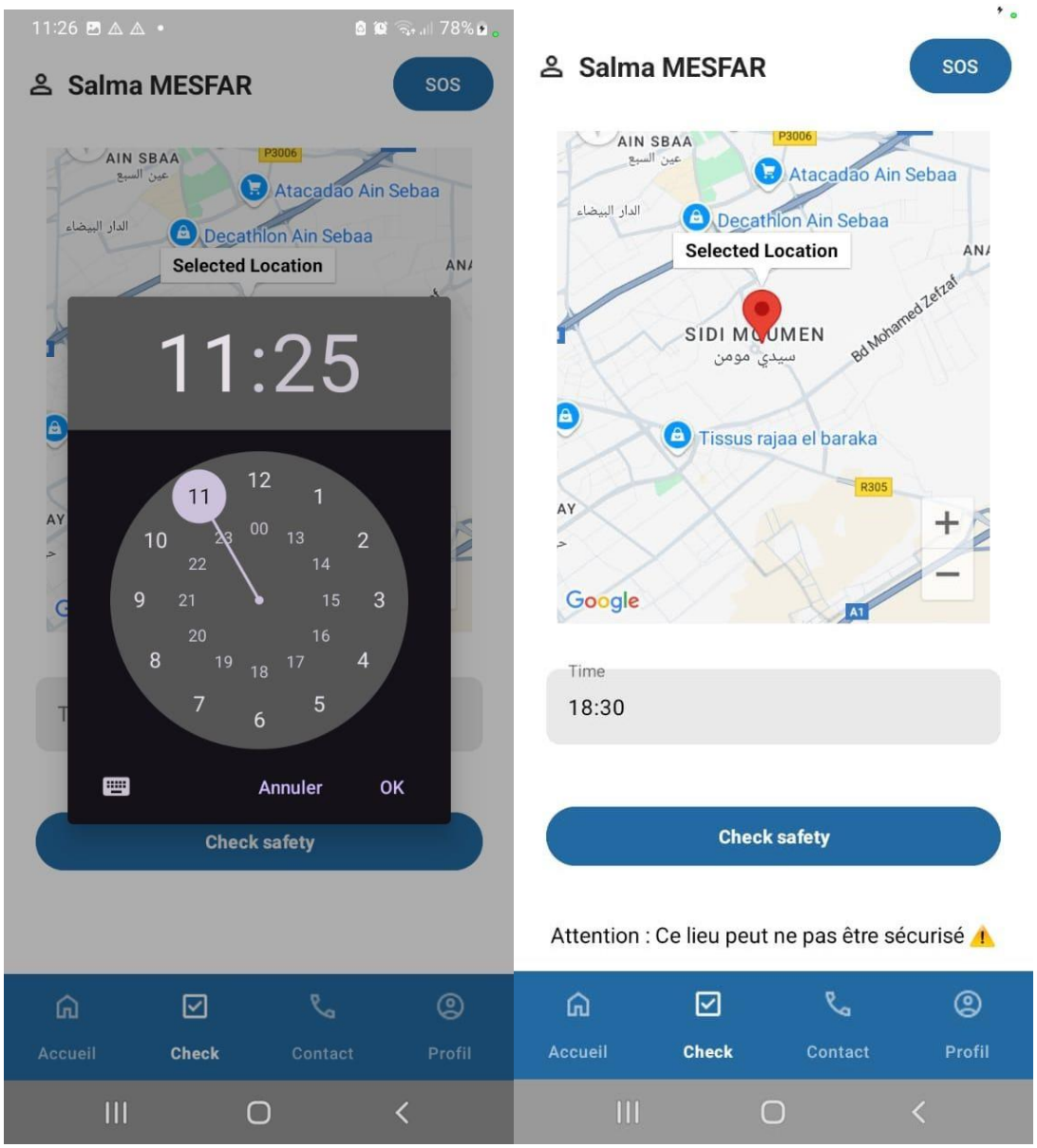



Figure 21: Check safety: danger

Le résultat, retourné sous forme de score de dangerosité (ex: "85% de risque entre 20h et 22h"), est affiché à l'utilisateur via une interface intuitive utilisant un code couleur (vert/orange/rouge) et des recommandations personnalisées ("Évitez ce parc la nuit, privilégiez le côté nord mieux éclairé"). Pour garantir des prédictions actualisées, le modèle interroge en temps réel des APIs de données urbaines tout en protégeant la vie privée via l'anonymisation des requêtes. Une journalisation des recherches permet par ailleurs d'améliorer continuellement l'algorithme grâce à l'apprentissage automatique.

## 5.7 Fragment Contact :

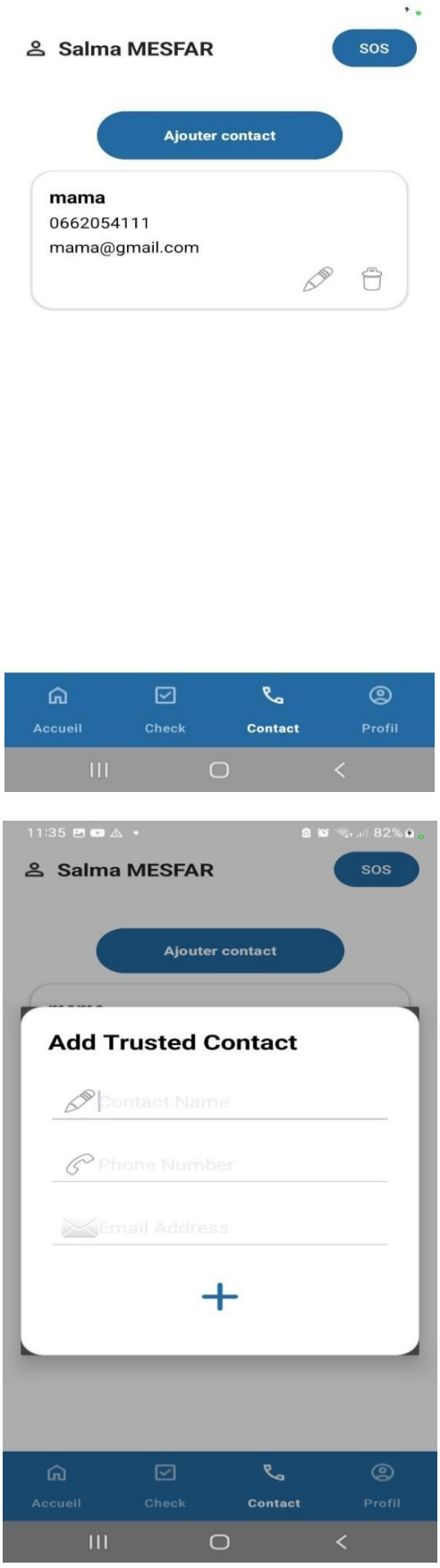
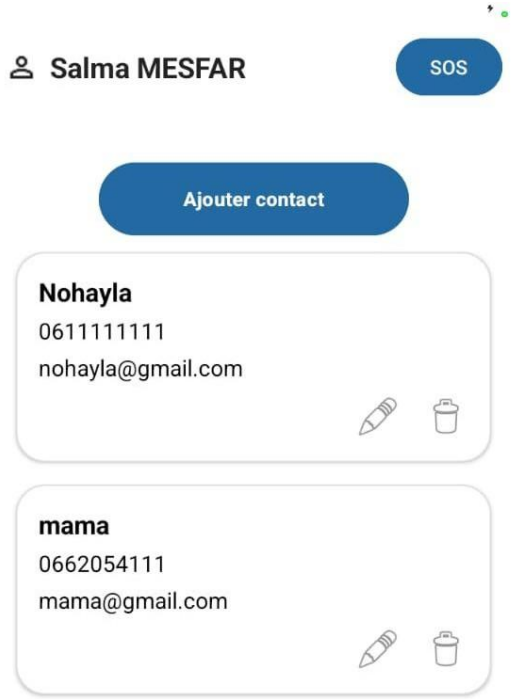
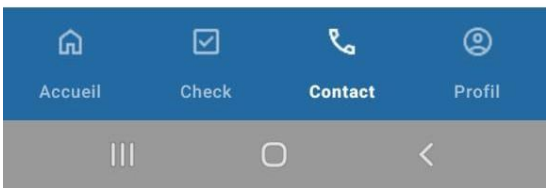



Figure 22: Fragment contact

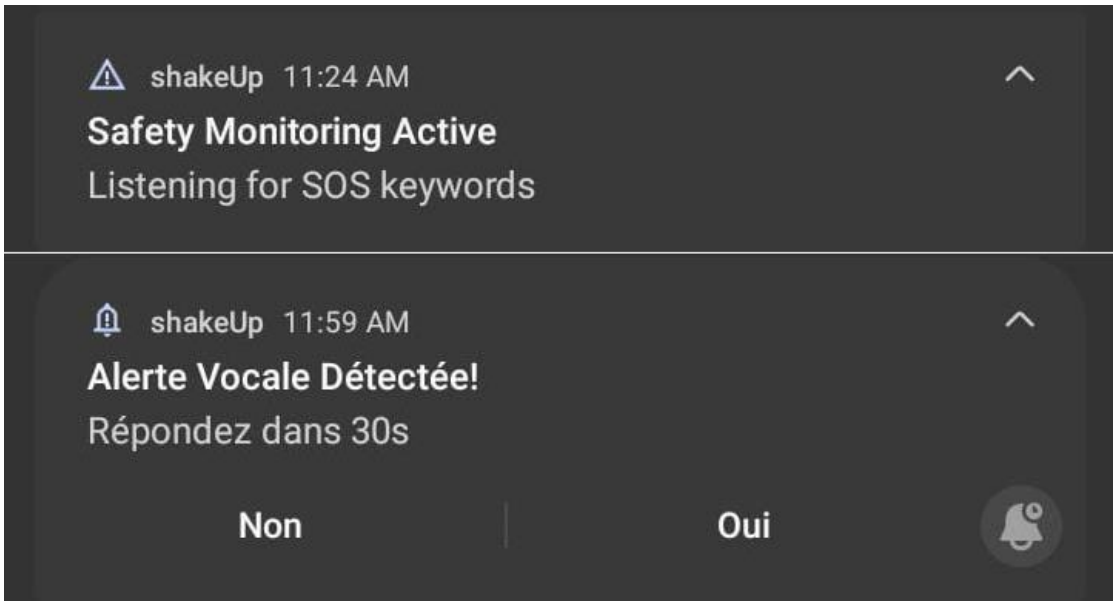
Après l'inscription, l'utilisateur doit configurer au moins un contact de confiance pour activer les fonctionnalités de sécurité. Dans la section dédiée, accessible via le menu principal, il peut ajouter des contacts en saisissant leurs informations (nom, numéro de téléphone, relation) ou en les sélectionnant directement depuis son répertoire. Le système impose un minimum d'un contact validé pour garantir l'efficacité du service d'urgence.




Chaque ajout nécessite une confirmation par SMS pour vérifier l'accord du contact. L'utilisateur peut à tout moment modifier les informations (mettre à jour un numéro) ou supprimer un contact, sauf le dernier enregistré - une sécurité pour maintenir une protection minimale.

L'interface intuitive présente les contacts sous forme de liste claire avec icônes d'édition/suppression, et un rappel visuel signale quand le nombre minimum n'est pas atteint. Pour renforcer la sécurité, chaque modification de la liste déclenche une notification aux contacts concernés, et un historique des changements est conservé dans le cloud.




L'application analyse votre environnement pour détecter le mot-clé d'urgence ('Alerte' ou autre mot configuré). Vous avez **6 secondes** pour prononcer la commande avant annulation automatique.

→ **Parlez clairement maintenant**

## 5.8 Conclusion du Chapitre

Ce chapitre a présenté l'ensemble du système de détection et d'envoi d'alertes, depuis les interfaces utilisateur jusqu'aux mécanismes de transmission des données. Nous avons décrit le processus complet qui permet à l'application d'identifier une situation de danger grâce à la détection vocale ou gestuelle, puis de partager automatiquement la localisation avec les contacts préalablement enregistrés. Les différentes interfaces ont été conçues pour rester simples et intuitives, tout en offrant une expérience fluide et réactive. La gestion des contacts de confiance a été intégrée de manière sécurisée, avec des validations systématiques pour garantir la fiabilité du système. L'ensemble de ces fonctionnalités forme un dispositif cohérent qui répond au besoin initial : fournir une solution discrète et efficace pour alerter ses proches en cas d'urgence. Les prochaines étapes consisteront à approfondir les aspects techniques liés

à la sécurisation des données et à l'optimisation des performances pour une expérience utilisateur encore plus robuste.

# *Conclusion générale*

Ce projet a permis de concevoir et développer une application mobile de sécurité innovante, offrant aux utilisateurs une solution fiable et discrète pour alerter leurs proches en cas de danger. À travers les différents chapitres, nous avons abordé l'ensemble des aspects techniques et fonctionnels, depuis la détection des situations d'urgence (via les capteurs du smartphone et la reconnaissance vocale) jusqu'à l'envoi sécurisé de la localisation aux contacts de confiance.

L'application repose sur une architecture robuste, combinant Kotlin pour l'interface utilisateur et Firebase pour le stockage et la gestion des données en temps réel. Les mécanismes de déclenchement (geste de secousse ou mot-clé vocal) ont été optimisés pour garantir une réactivité immédiate, tout en minimisant les risques de faux positifs. La gestion des contacts, avec son système de validation par SMS, assure que seules les personnes autorisées reçoivent les alertes.

Les interfaces, pensées pour être intuitives et accessibles, guident l'utilisateur à chaque étape, que ce soit pour configurer ses paramètres de sécurité ou pour déclencher une alerte en situation critique. La priorité a été donnée à la protection des données personnelles, avec un chiffrement des informations et des règles strictes de confidentialité.

En somme, ce projet répond à un besoin concret de sécurité, en proposant un outil simple mais efficace, capable de s'adapter à différentes situations d'urgence. Les perspectives d'évolution, comme l'intégration avec les services de secours ou l'ajout de fonctionnalités avancées (détection de chute, mode veille géolocalisé), ouvrent la voie à des améliorations futures pour renforcer encore son utilité.

En conclusion, cette application représente une avancée significative dans le domaine de la sécurité mobile, alliant technologie moderne et simplicité d'usage, pour offrir aux utilisateurs une tranquillité d'esprit au quotidien.

# Annexe

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
model_rf = RandomForestClassifier(random_state=42)
model_rf.fit(X_train, y_train)
y_pred_rf = model_rf.predict(X_test)
print("Accuracy (Random Forest):", accuracy_score(y_test, y_pred_rf))
```

```
Accuracy (Random Forest): 0.954954954954955
```

```
import joblib
joblib.dump(model_rf, "model_danger2.pkl")
```

```
['model_danger2.pkl']
```

```
import pandas as pd
from sklearn.model_selection import train_test_split

# Charger ton dataset
df = pd.read_csv("geo_safety_data.csv")

# Supprimer les colonnes inutiles
df.drop(columns=['jour', 'mois', 'LOCATION'], inplace=True)

# Filtrer les lignes avec une heure valide
df = df[(df['heure'] >= 0) & (df['heure'] <= 23)]

X = df[['Vict Age', 'Vict Sex', 'heure', 'LAT', 'LON']]
y = df['danger']

# Encoder la variable 'Vict Sex' en numérique
X['Vict Sex'] = X['Vict Sex'].map({'M': 1, 'F': 0})

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```python
from fastapi import FastAPI, UploadFile, File, Form
from fastapi import HTTPException
from fastapi.responses import JSONResponse
from pydantic import BaseModel
import pandas as pd
import joblib
import whisper
import tempfile
import shutil
import os
import subprocess
import time

# Load both models
safety_model = joblib.load("model_danger2.pkl")
whisper_model = whisper.load_model("small")

app = FastAPI()

# === Endpoint 1: Safety Prediction ===
class SafetyRequest(BaseModel):
    latitude: float
    longitude: float
    hour: int
    gender: str
    age: str
```

```python
    try:
        # Save uploaded file
        with tempfile.NamedTemporaryFile(suffix=".mp4", delete=False) as tmp:
            shutil.copyfileobj(file.file, tmp)
            tmp_path = tmp.name

        # Convert to WAV with optimized parameters
        wav_path = tmp_path + ".wav"
        cmd = [
            "ffmpeg",
            "-y",  # Overwrite without prompting
            "-i", tmp_path,
            "-acodec", "pcm_s16le",
            "-ar", "16000",  # Target 16kHz sample rate
            "-ac", "1",      # Mono audio
            "-hide_banner",  # Suppress FFmpeg logs
            "-loglevel", "error",
            wav_path
        ]
        subprocess.run(cmd, check=True)
```

```python
@app.post("/safety-check/")
def check_safety(request: SafetyRequest):
    df = pd.DataFrame([{
        "Vict Age": request.age,
        "Vict Sex": request.gender,
        "heure": request.hour,
        "LAT": request.latitude,
        "LON": request.longitude
    }])
    prediction = safety_model.predict(df)[0]
    return {"safe": int(prediction)}

# === Endpoint 2: Voice Transcription ===
# In FastAPI code
@app.post("/transcribe/")
async def transcribe_audio(
    file: UploadFile = File(...),
    sos_word: str = Form(...)
):
    if not file.filename.lower().endswith(('.mp4', '.m4a')):
        raise HTTPException(400, "Unsupported file format")
    start_time = time.time()
    try:
        # Save uploaded file
        with tempfile.NamedTemporaryFile(suffix=".mp4", delete=False) as tmp:
```

```python
        # Transcribe with Whisper
        result = whisper_model.transcribe(wav_path, language="en")
        text = result["text"].strip().lower()
        sos_word = sos_word.strip().lower()

        end_time = time.time()
        print(f"Total processing time: {end_time - start_time:.2f}s")
        return {"text": text, "match": sos_word in text}

    except subprocess.CalledProcessError as e:
        return JSONResponse(
            status_code=500,
            content={"error": f"FFmpeg conversion failed: {str(e)}"}
        )
    finally:
        for path in [tmp_path, wav_path]:
            if path and os.path.exists(path):
                try:
                    os.unlink(path)
                except Exception as e:
                    print(f"Error deleting {path}: {e}")
```