



Université de Montpellier

FACULTÉ DES SCIENCES DE MONTPELLIER

M1 IASD

Projet de programmation mobile

**Application pour la Gestion de boîte d'Intérim
"Paul Emploi"**



Rédigé par :

Rym ZEGGAR 21909615
Sofian ETTAHIRI 21901227
Yousef LABIAD 21710780

Encadrants :

Djamel SERIAI

Table des matières

1	Introduction	2
1.1	Contexte	2
1.2	Objectif du projet	2
2	Méthodologie et outils de développement	3
2.1	Choix des outils utilisé	3
2.1.1	Android studio	3
2.1.2	Firebase	4
2.2	Processus de développement	5
3	Architecture de l'application	5
3.1	Première vision de l'application	5
3.1.1	Page d'accueil	6
3.2	Application coté candidat	7
3.3	Application coté agence	7
3.4	Application coté gestionnaire	8
3.5	Gestion des données	9
3.5.1	Modèle UML	9
3.5.2	Structure de la base de données	10
3.5.3	Offre d'emploi	11
3.5.4	Admins ou Super-gestionnaire	11
3.5.5	Gestionnaire	12
3.5.6	Employeur et Agence Interim	12
3.5.7	Candidature	13
3.5.8	Candidats	13
3.6	Profils	14
4	Fonctionnalités	15
4.1	Démarrage de l'application	15
4.2	Menu de l'application	16
4.3	Fonctionnalités pour l'utilisateur anonyme	17
4.4	Fonctionnalités pour les demandeurs d'emploi (candidats)	18
4.5	Fonctionnalités pour les recruteurs (employeurs/agences)	18
4.6	Fonctionnalités pour les gestionnaires (Manager)	20
4.7	Fonctionnalités pour les administrateurs (Admin)	21
4.8	Les autres fonctionnalités notables	21

5 Bonne pratiques de conception :	21
6 Verrous et solutions	22
7 Conclusion	22
8 Annexe - Activités	23
8.1 Lien github et précision	23
8.2 Activités communes à tous profils d'utilisateurs	23
8.2.1 TermsActivity	23
8.3 Activités pour le profil "anonyme"	23
8.3.1 SplashActivity	23
8.3.2 WelcomeActivity	24
8.3.3 RegisterInitActivity	24
8.3.4 SigninInitActivity	25
8.3.5 GuestActivity	25
8.4 Activités pour le profil "candidat"	26
8.4.1 MainActivity	26
8.4.2 ScheduleActivity	27
8.4.3 MappingActivity	27
8.4.4 RegisterActivity	28
8.4.5 SigninActivity	28
8.4.6 ForgotPasswordUser	29
8.4.7 schedulerequest	29
8.4.8 myaccount	30
8.5 Activités pour le profil "recruteur"	30
8.5.1 RecruiterHomeActivity	30
8.5.2 CvActivity	31
8.5.3 ForgotPasswordRecruiter	32
8.5.4 PostJobActivity	32
8.5.5 RecruiterMyAccountActivity	33
8.5.6 RegisterActivityRecruiter	34
8.5.7 ScheduleRequestManagerActivity	34
8.5.8 SigninActivityRecruiter	35
8.5.9 UpdateAdActivity	36
8.6 Activités pour le profil "administrateur"	36
8.6.1 AdminHomeActivity	36
8.6.2 SigninActivityAdmin	37
8.6.3 CompaniesActivity	37

8.6.4	JobsActivity	38
8.6.5	SchedActivity	38
8.7	Activités pour le profil "gestionnaire"	39
8.7.1	ManagerHomeActivity	39
8.7.2	RegisterActivityManager	39
8.7.3	SigninActivityManager	40
8.7.4	ForgotPasswordManager	40
8.7.5	ManagerMyAccountActivity	41
9	Modules	41
9.1	ManagerModule	42
9.2	PostModule	42
9.3	ScheduleModule	43
9.4	UserModule	43
10	MyViewHolders	44
10.1	MyViewHolder_Company	44
10.2	MyViewHolder_Home	44
10.3	MyViewHolder_Manager	44
10.4	MyViewHolder_Schedule	45

Remerciements

Djamel, nous tenons à vous remercier pour vos connaissances approfondies en développement mobile. Vos explications claires et vos conseils avisés nous ont permis de mieux appréhender les concepts et de progresser dans nos compétences. Votre enthousiasme et votre disponibilité ont été une source de motivation et d'inspiration tout au long de ce projet.

Bachar, nous vous sommes reconnaissants pour votre accompagnement et vos retours constructifs. Vos suggestions et votre regard critique ont été d'une grande aide pour améliorer la qualité de notre rapport. Votre disponibilité et votre implication ont été très appréciées.

Grâce à vous deux, nous avons pu bénéficier d'un encadrement de qualité et relever les défis liés à notre projet avec confiance. Votre contribution a été déterminante dans notre apprentissage et notre réussite.

Nous souhaitons exprimer nos sincères remerciements à Djamel Seriai et Bachar Rima pour leur précieux soutien lors de la réalisation de notre rapport de projet. Leur expertise et leur disponibilité ont grandement contribué à notre réussite.

Djamel, nous tenons à vous remercier pour vos connaissances approfondies en développement mobile. Vos explications claires et vos conseils avisés nous ont permis de mieux appréhender les concepts et de progresser dans nos compétences. Votre enthousiasme et votre disponibilité ont été une source de motivation et d'inspiration tout au long de ce projet.

Bachar, nous vous sommes reconnaissants pour votre accompagnement et vos retours constructifs. Vos suggestions et votre regard critique ont été d'une grande aide pour améliorer la qualité de notre rapport. Votre disponibilité et votre implication ont été très appréciées.

Grâce à vous deux, nous avons pu bénéficier d'un encadrement de qualité et relever les défis liés à notre projet avec confiance. Votre contribution a été déterminante dans notre apprentissage et notre réussite.

1 Introduction

1.1 Contexte

Le secteur de l'emploi intérimaire joue un rôle essentiel dans le marché du travail, offrant des opportunités de recrutement et de placement de personnel temporaire pour répondre aux besoins des entreprises. Cependant, le processus traditionnel de recherche d'emplois intérimaires et de candidature peut souvent être complexe, fastidieux et peu efficace.

Dans ce contexte, notre projet vise à développer une application mobile pour les agences intérim qui simplifie et optimise les processus de recrutement. Cette application fournira une plateforme centralisée où les agences intérim pourront publier leurs offres d'emploi et les candidats pourront postuler facilement et rapidement. L'objectif est de créer un environnement numérique convivial, offrant une expérience utilisateur fluide tant pour les agences que pour les candidats.

1.2 Objectif du projet

Les principaux objectifs de notre projet sont les suivants :

- Développer une application mobile pour les agences intérim, permettant de publier et de gérer les offres d'emploi de manière efficace. Fournir une interface conviviale pour les candidats, leur permettant de rechercher et de postuler aux offres d'emploi correspondant à leurs compétences et à leurs préférences.
- Intégrer des fonctionnalités d'authentification sécurisée pour les agences intérim et les candidats, garantissant la confidentialité des informations et la protection des données personnelles.
- Implémenter des fonctionnalités de notifications push pour informer les agences intérim des nouvelles candidatures et des mises à jour importantes.
- Utiliser Firebase comme plateforme cloud pour gérer la base de données en temps réel, l'authentification des utilisateurs, le stockage des fichiers et les notifications push.

En mettant en œuvre ces objectifs, nous visons à améliorer l'efficacité des agences intérim dans le processus de recrutement, à simplifier la recherche d'emploi pour les candidats et à favoriser une meilleure correspondance entre les besoins des entreprises et les compétences des candidats intérimaires.

À travers ce rapport, nous détaillerons le processus de développement de l'application, les outils utilisés, l'architecture de l'application, les fonctionnalités implémentées, les tests effectués, ainsi que les conclusions et les perspectives pour l'amélioration future du projet.

Cette introduction au contexte du projet établit la base pour comprendre les problèmes rencontrés dans le domaine des agences intérim et l'objectif de notre application mobile. Les objectifs spécifiques du projet sont également clairement définis pour guider le lecteur à travers le reste du rapport.

2 Méthodologie et outils de développement

2.1 Choix des outils utilisé

Dans cette section, nous allons présenter les outils essentiels utilisés dans le cadre du développement de notre application mobile pour les agences intérim. Nous aborderons en détail l'utilisation d'Android Studio, l'environnement de développement intégré (IDE) pour la création d'applications Android, Firebase, la plateforme cloud de Google qui offre un ensemble de services pour faciliter le développement, le déploiement et la gestion de notre application, github une plateforme de développement collaboratif basée sur Git ainsi que Disocrd une plateforme de communication qui a facilité les échanges entre les membres du projets.

2.1.1 Android studio

Dans le cadre du développement de notre application mobile visant à faciliter les processus de recrutement des agences intérim, nous avons utilisé Android Studio, l'environnement de développement intégré (IDE) phare pour la création d'applications Android. Android Studio est principalement utilisé pour la création d'applications Android. Il est développé par Google et constitue l'outil de référence pour les développeurs d'applications Android. Dans le cadre de notre projet pour la faculté, nous avons utilisé Android Studio pour développer notre application mobile visant à faciliter les processus de recrutement des agences intérim.



FIGURE 1 – Logo Android studio

Voici quelques caractéristiques et fonctionnalités clés d'Android Studio qui ont été utiles pour notre projet :

- JAVA : language orienté objet, permet d'instancier des classes a Activity kjnfvjnzefvojvfojnfv...
- Émulateur Android : Android Studio est livré avec un émulateur Android intégré, ce qui nous a permis de tester notre application sur différents appareils virtuels sans avoir besoin de matériel physique. Cela nous a donné la possibilité de vérifier le fonctionnement de notre application sur différentes configurations, versions d'Android et tailles d'écran.
- Android Studio offre des fonctionnalités de débogage puissantes qui nous ont aidés à identifier et à résoudre rapidement les problèmes de notre application. Nous avons pu placer des points d'arrêt, examiner les variables en temps réel, suivre l'exécution pas à pas et utiliser d'autres fonctionnalités de débogage pour comprendre le comportement de notre application et corriger les erreurs.

- Intégration avec Gradle : Android Studio utilise Gradle comme système de build, ce qui nous a permis de gérer efficacement les dépendances et les configurations de notre projet. Nous avons pu ajouter des bibliothèques tierces, des plugins et personnaliser le processus de build de manière flexible grâce à l'intégration étroite entre Android Studio et Gradle.

2.1.2 Firebase

Nous avons également choisi d'utiliser Firebase, une plateforme cloud proposée par Google, offrant un ensemble d'outils et de services indispensables tels que la base de données en temps réel, l'authentification des utilisateurs, le stockage de fichiers et les notifications push, pour simplifier et optimiser le développement, le déploiement et la gestion de notre application.



FIGURE 2 – Logo Firebase

Dans notre projet pour la faculté, nous avons utilisé Firebase pour plusieurs aspects clés de notre application, notamment :

- Base de données en temps réel : Firebase Realtime Database est une base de données NoSQL hébergée dans le cloud qui permet de stocker et de synchroniser les données en temps réel. Nous avons utilisé cette fonctionnalité pour stocker les offres d'emploi des agences intérim, les informations des candidats, ainsi que d'autres données pertinentes. La mise à jour en temps réel de la base de données a permis d'assurer la cohérence des données et de synchroniser instantanément les informations entre les utilisateurs de l'application.
- Authentification des utilisateurs : Firebase Authentication propose une solution complète d'authentification des utilisateurs. Nous avons utilisé ce service pour gérer l'inscription et la connexion des agences intérim et des candidats à notre application. Grâce à Firebase Authentication, nous avons pu mettre en place un système d'authentification sécurisé et personnalisé, offrant différentes méthodes d'authentification telles que l'adresse e-mail et les comptes Google.
- Notifications push : Firebase Cloud Messaging (FCM) est un service de messagerie qui permet d'envoyer des notifications push aux utilisateurs de l'application. Nous avons utilisé FCM pour envoyer des notifications aux agences intérim lorsqu'un nouveau candidat postulait à une offre d'emploi ou lorsqu'il y avait des mises à jour importantes. Les notifications push ont amélioré l'engagement des utilisateurs en les tenant informés des activités pertinentes de l'application.
- Analytics et suivi des performances : Firebase Analytics fournit des fonctionnalités d'analyse et de suivi des performances de l'application. Nous avons utilisé cet outil pour collecter des données sur l'utilisation de notre application, les événements clés, les comportements des utilisateurs, etc. Ces informations nous ont permis d'obtenir des insights précieux sur l'engagement des utilisateurs et de prendre des décisions éclairées pour améliorer l'expérience utilisateur.

2.2 Processus de développement

Dans le cadre du développement de notre application mobile pour les agences intérim, nous avons opté pour une méthodologie agile afin de gérer efficacement les nombreux projets que nous avions en parallèle. Cette approche s'est avérée la plus appropriée pour notre équipe, compte tenu de la nature dynamique du projet, des délais serrés et de la nécessité de s'adapter rapidement aux changements. La planification initiale du projet s'est étendue sur une période de deux mois. Nous avons divisé cette période en sprints de deux semaines, ce qui nous a permis de travailler de manière itérative et de nous concentrer sur des objectifs spécifiques à chaque étape. Chaque sprint comprenait des tâches et des fonctionnalités spécifiques à réaliser.

La répartition des tâches s'est faite entre les membres de l'équipe de développement, à savoir Rym, Youcef et Sofian. Nous avons identifié les compétences et les intérêts de chaque membre afin de répartir les tâches de manière équilibrée et efficace. Rym s'est principalement concentrée sur la conception de l'interface utilisateur et l'expérience utilisateur, Youcef était responsable du développement de la logique de l'application, tandis que Sofian s'est occupé de la gestion de la base de données et de l'intégration des services Firebase.

Nous avons tenu des réunions régulières pour discuter de l'avancement du projet, partager des idées et des suggestions, et résoudre les problèmes éventuels. La communication transparente et la collaboration active entre les membres de l'équipe ont été essentielles pour assurer une coordination harmonieuse et la réalisation des objectifs fixés.

Pendant les sprints, nous avons utilisé la plateforme GitHub pour gérer notre code source et faciliter la collaboration. Nous avons créé des branches distinctes pour chaque fonctionnalité, ce qui nous a permis de travailler simultanément sans interférer avec le travail des autres membres de l'équipe. Grâce aux fonctionnalités de suivi des problèmes (issues) de GitHub, nous avons également pu signaler et résoudre rapidement les problèmes rencontrés lors du développement.

En ce qui concerne les tests et la validation, nous avons adopté une approche continue tout au long du processus de développement. Nous avons effectué des tests unitaires pour vérifier le bon fonctionnement de chaque composant, ainsi que des tests d'intégration pour s'assurer de la cohésion de l'application dans son ensemble. De plus, nous avons régulièrement sollicité les retours et les commentaires de notre équipe et de quelques utilisateurs cibles pour améliorer l'expérience utilisateur et corriger les éventuels problèmes.

3 Architecture de l'application

3.1 Première vision de l'application

Au début du projet, nous avons élaboré une première vision de notre application mobile pour les agences intérim. Cette vision initiale a servi de point de départ et de guide pour la conception et le développement ultérieurs. Notre objectif principal était de créer une plateforme conviviale et fonctionnelle qui permettrait aux agences intérim de publier des offres d'emploi et aux candidats de postuler facilement. Nous avons imaginé une interface intuitive, avec des fonctionnalités clés telles que la recherche d'offres d'emploi, la création de profils de candidats, la gestion des candidatures et la communication entre les agences et les candidats. Cette première vision nous a permis d'établir une base solide pour notre application et de définir les fonctionnalités essentielles à développer.

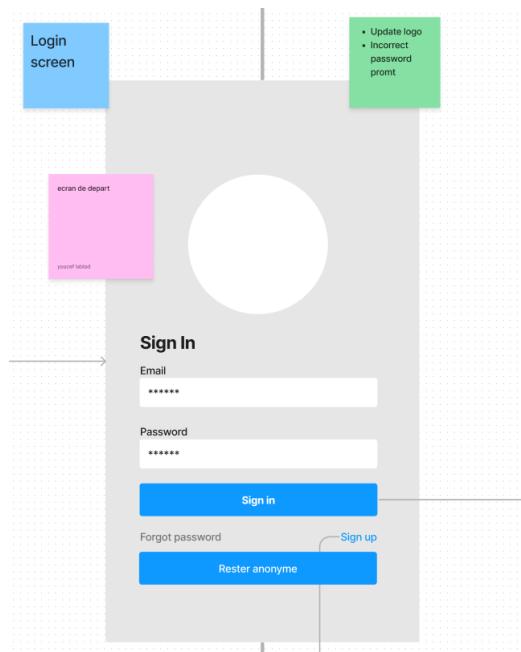


FIGURE 3 – Page d'accueil avec la login page

3.1.1 Page d'accueil

Dans notre conception initiale, nous avions envisagé d'avoir une page d'accueil présentée sous la forme d'un formulaire de connexion ou d'inscription. Cette approche permettrait à l'utilisateur d'accéder directement à son compte s'il en possède déjà un, ou de s'inscrire s'il n'en a pas. Nous n'avions pas encore pensé à diriger l'utilisateur vers le type de profil qui lui correspond. De plus, nous avons prévu une option permettant aux utilisateurs de rester anonymes et d'accéder au reste de l'application sans avoir à s'identifier. Cela offrirait une flexibilité supplémentaire aux utilisateurs qui souhaitent explorer l'application sans divulguer d'informations personnelles.

3.2 Application coté candidat

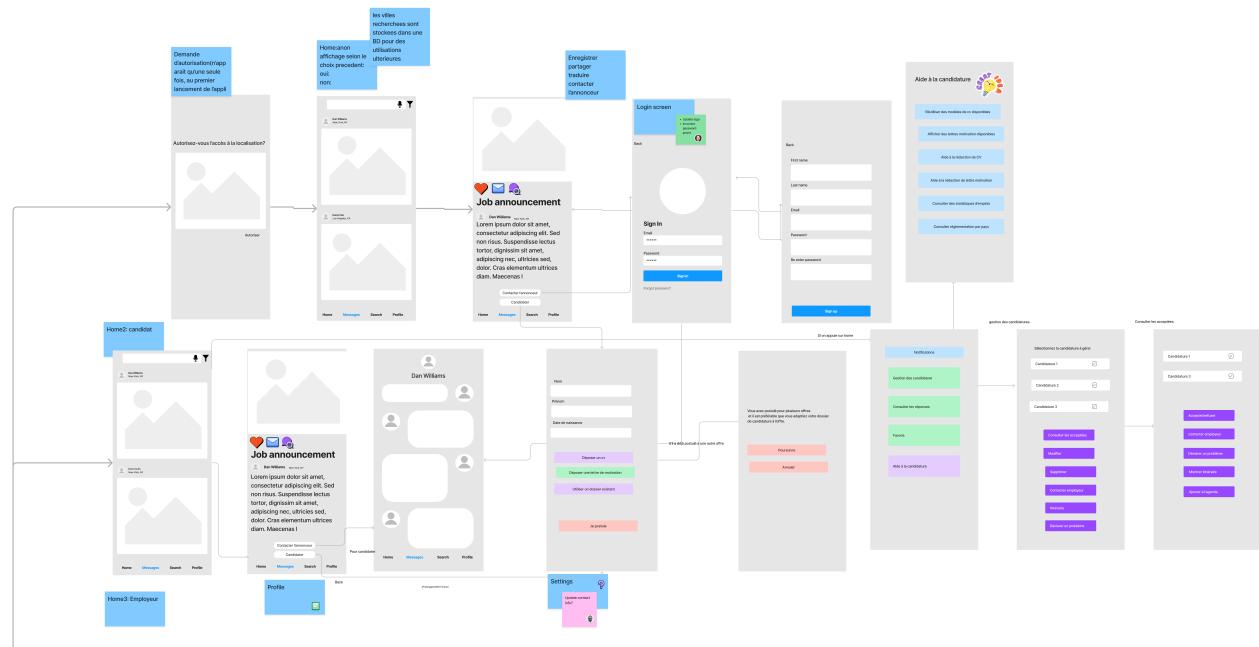


FIGURE 4 – Application coté candidat

Lorsque le candidat se connecte à l'application, il lui sera demandé d'autoriser l'accès à sa localisation. En fonction de cette demande, l'application affichera une liste d'offres d'emploi pertinentes. Le candidat pourra alors postuler à ces offres en fournissant ses informations personnelles, en attachant une lettre de motivation et un CV. De plus, les utilisateurs auront la possibilité de gérer leurs candidatures déjà soumises, leur permettant de suivre et de gérer leur progression dans le processus de recrutement. Dans notre vision initiale de l'application, nous avions également prévu de fournir à l'utilisateur une assistance pour la rédaction de son CV et de sa lettre de motivation.

3.3 Application coté agence

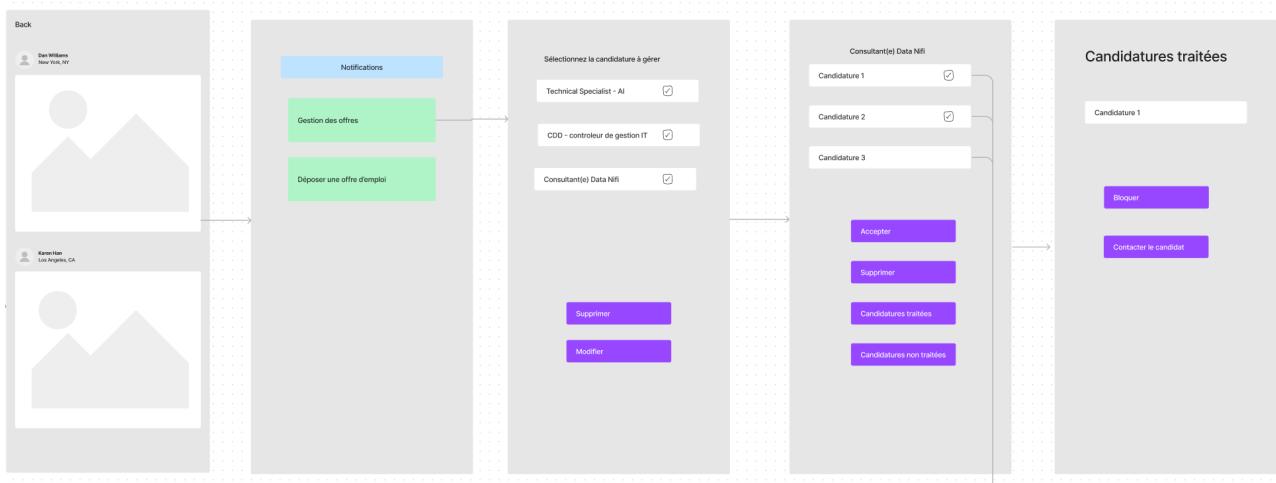


FIGURE 5 – Application coté candidat

Lorsque l'utilisateur décide de se connecter en tant qu'agence ou employeur, il sera dirigé vers une page d'accueil similaire à celle du candidat. Cependant, la différence réside dans le fait que les offres

d'emploi affichées seront spécifiques à l'utilisateur connecté. Cela signifie que l'utilisateur aura accès à ses propres offres d'emploi, lui permettant de les consulter et de les gérer. Il pourra également créer de nouvelles offres d'emploi en fournissant les détails pertinents tels que le titre, la description, les compétences requises et la localisation. Pour les offres d'emploi déjà créées, l'utilisateur aura la possibilité de gérer les candidatures reçues de la part des chercheurs d'emploi. Il pourra examiner les candidatures, les accepter si elles correspondent aux critères requis ou les refuser si elles ne conviennent pas. Cette fonctionnalité permet à l'utilisateur de gérer efficacement le processus de recrutement et de prendre des décisions éclairées concernant les candidatures reçues.

3.4 Application coté gestionnaire

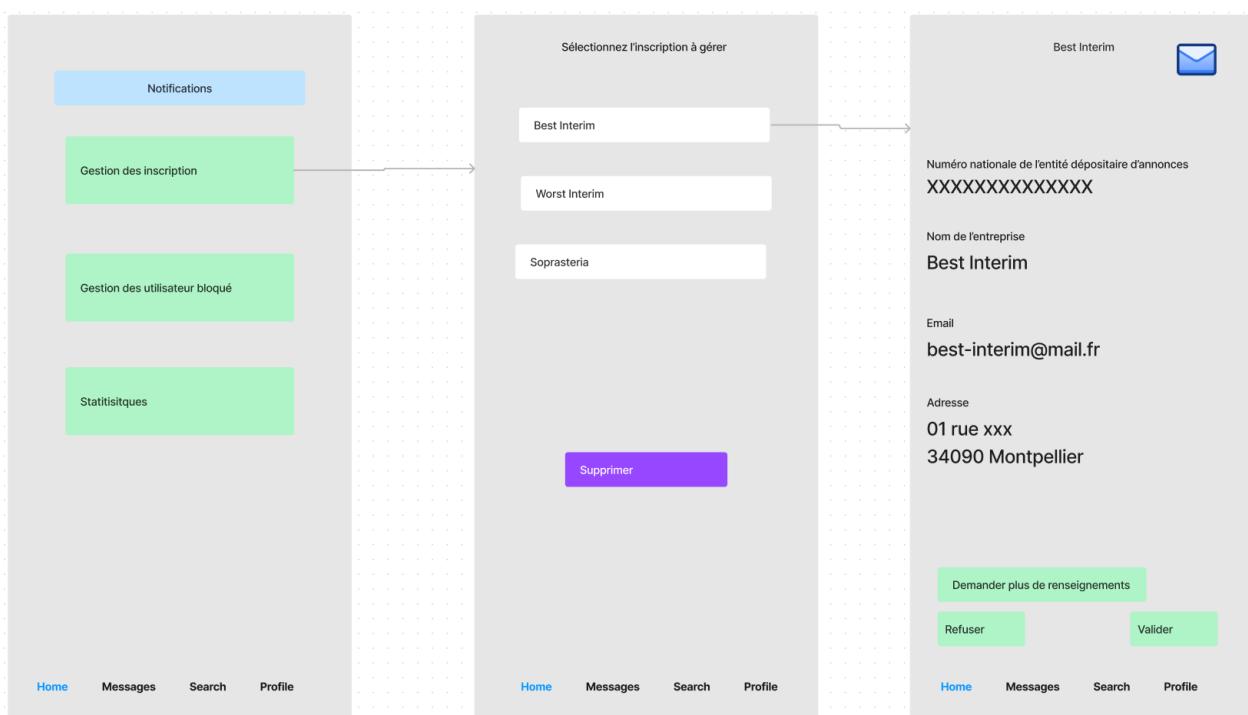


FIGURE 6 – Application coté gestionnaire

Les gestionnaires, quant à eux, ne seront pas redirigés vers la liste des offres d'emploi, mais vers une page qui leur est spécifique. Sur cette page dédiée, les gestionnaires auront accès à diverses fonctionnalités et outils pour assurer le bon fonctionnement de l'application.

Tout d'abord, ils trouveront un bouton leur permettant d'accéder à la gestion des inscriptions des agences et des employeurs. Cette fonctionnalité est essentielle pour vérifier et valider les comptes des agences et des employeurs avant qu'ils ne soient actifs sur la plateforme. Les gestionnaires examineront attentivement les informations fournies lors de l'inscription afin de s'assurer de l'authenticité des utilisateurs et d'éviter les comptes frauduleux, les arnaques et les faux profils.

Ensuite, les gestionnaires auront un autre bouton pour accéder à la gestion des utilisateurs bloqués. Cette fonctionnalité leur permettra de consulter la liste des utilisateurs qui ont été bloqués en raison de violations des conditions d'utilisation ou de comportements inappropriés. Les gestionnaires auront la possibilité de lever les blocages si nécessaire ou de prendre des mesures appropriées pour maintenir l'intégrité de la plateforme.

Enfin, une fonctionnalité de statistiques sera également mise à la disposition des gestionnaires. Cette fonctionnalité leur permettra de consulter des données et des informations statistiques sur l'utilisation de l'application, telles que le nombre total d'agences et d'employeurs inscrits, le nombre d'offres d'emploi publiées, les candidatures soumises, etc. Ces statistiques peuvent être utiles pour évaluer

les performances de l'application, identifier les tendances et prendre des décisions éclairées pour améliorer l'expérience des utilisateurs.

En fournissant ces outils spécifiques aux gestionnaires, l'application facilite la gestion et l'administration du système dans son ensemble, garantissant ainsi un environnement sûr, fiable et efficace pour les agences intérim, les employeurs et les candidats.

3.5 Gestion des données

3.5.1 Modèle UML

Dans le cadre de notre projet de développement d'une application mobile, nous avons élaboré un modèle UML exhaustif qui décrit de manière systématique l'architecture et les interactions de notre application. Ce modèle UML nous a permis de visualiser et de planifier efficacement les différentes composantes de notre application, en mettant particulièrement l'accent sur l'intégration de Firebase, une plateforme puissante et flexible, choisie pour ses fonctionnalités de stockage, d'authentification utilisateur et de gestion des données en temps réel. Grâce à cette approche méthodique et à l'utilisation judicieuse de Firebase, nous avons pu créer une application mobile robuste, évolutrice et hautement fonctionnelle.

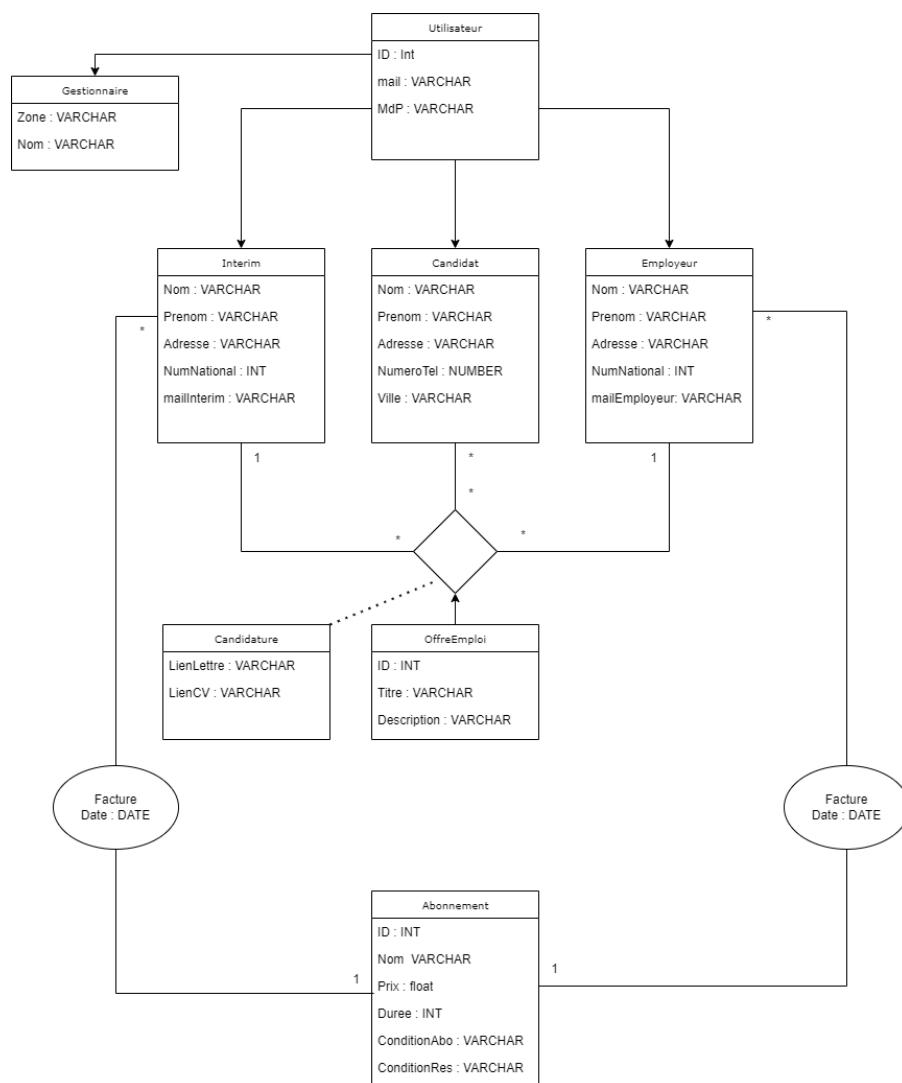


FIGURE 7 – 1ère structure de notre base de données

3.5.2 Structure de la base de données

Au commencement de notre projet, nous avons élaboré un diagramme UML afin d'avoir une première vision de la structure de notre base de données. Cependant, au fil de l'avancement de notre projet, nous avons effectué des ajustements et des modifications pour aboutir à une version différente. Nous avons revu et adapté la structure de la base de données en fonction des besoins et des exigences spécifiques de notre application mobile. Cette évolution nous a permis d'optimiser la manière dont les données sont organisées et stockées, afin d'améliorer la performance, la scalabilité et la maintenabilité de notre application. Voici ci dessous la nouvelle structure de notre base de données Firebase

<https://recruitment-management-s-cd04f-default.firebaseio.com/>

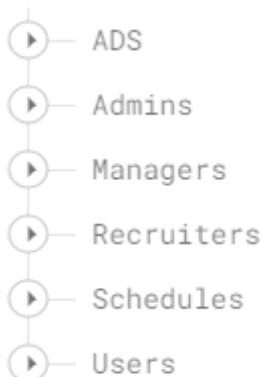


FIGURE 8 – Structre de notre base de donénes

On y trouve donc 5 tables de données :

1. ADS qui représente les information liées a une offre d'emploi
2. Admins qui représente les informations liées aux administrateurs
3. Manager qui représente les informations liées aux agence d'interim
4. Recruiters qui représente les informations liées aux employeurs
5. Schedules qui représentes les informations liées a la plannification d'un entretien
6. Uses qui représente les informations liées aux candidats

3.5.3 Offre d'emploi

On a donc la table suivante des offres d'emplois proposer par les employeurs et les agences appelé **ADS**

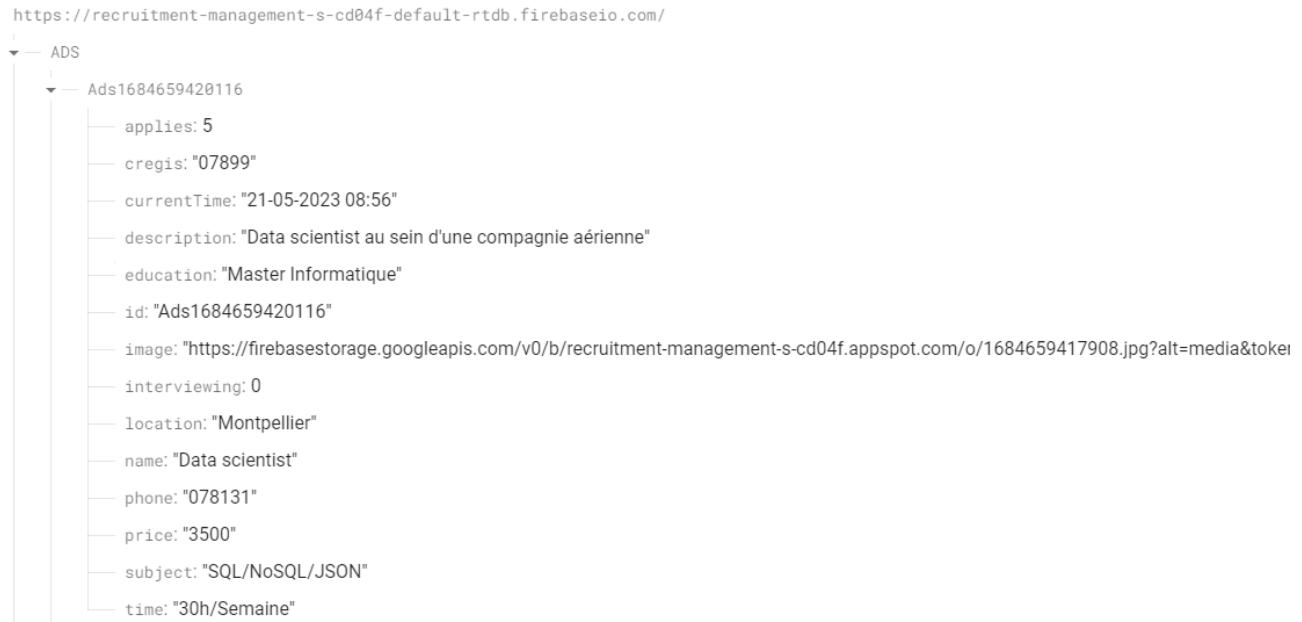


FIGURE 9 – Les offres d'emploi

Dans cette tables on va y trouver de nombreuses informations concernant les annonces d'emplois :

- **Applies** : C'est le nombre de candidature effectué pour cette offre
- **Cregis** : C'est l'ID de la personne ayant créer l'offre
- **currentTime** : L'heure ou l'annonce a été déposé
- **education** : Le niveau d'étude requis
- **id** : L'ID de l'offre en question
- **image** : Le lien ou est stocké l'image qui représente l'offre
- **interviewing** : Le nombre de personne qui ont eu un entretien pour cette offre
- **location** : Le lieu de travail
- **phone** : Le numéro à contacter du déposeur d'annonce
- **price** : Le salaire proposé
- **subject** : Information liée à l'emploi
- **time** : Le temps de travail

3.5.4 Admins ou Super-gestionnaire

On a donc la table des gestionnaire qu'on trouve ci-dessous :

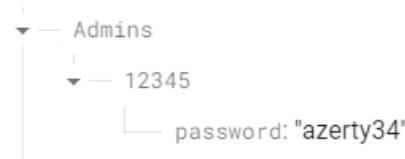


FIGURE 10 – Les super-gestionnaire

Dans cette tables on va y trouver les informations nécessaire à l'identification concernant les gestionnaires :

- **ID** : Un ID nécessaire à différencier les gestionnaire et leurs permettent de se connecter
- **password** : Un mot de passe pour pouvoir se connecter

3.5.5 Gestionnaire

Les gestionnaires sont inscrits dans la table manager suivante : Dans cette dernière on va trouver

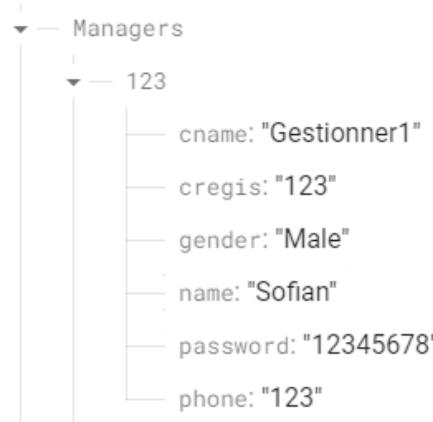


FIGURE 11 – Les gestionnaires

des informations personnelles sur le gestionnaire ainsi que ses identifiants pour ce connecter :

- **cname** : Le nom de gestionnaire
- **cregis** : L'identifiant ou numéro du gestionnaire
- **gender** : Le genre de la personne
- **name** : Le nom de la personne
- **password** : Le mot de passe de l'utilisateur
- **phone** : Le numéro de téléphone de l'utilisateur

3.5.6 Employeur et Agence Interim

Comme les employeurs et les agences interim ont les même fonctionnalités, ils se retrouvent dans la même table *Recruiter* :

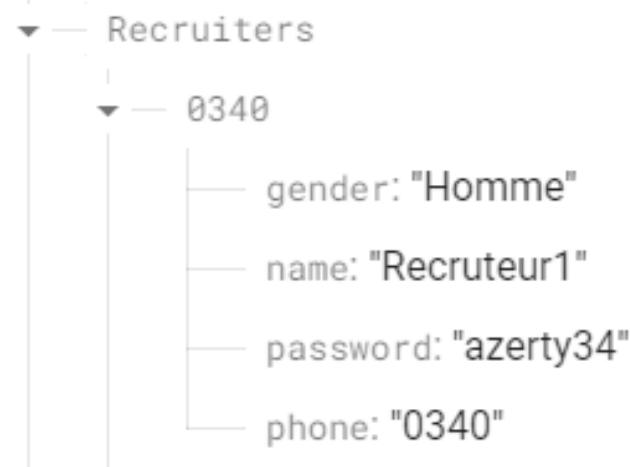


FIGURE 12 – Table Recruiter

On y trouve les informations suivantes sur les recruteurs :

- **Gender** : Le genre de la personne
- **Name** : Le nom de la personne
- **Password** : Le mot de passe utile à l'identification
- **phone** : Le numéro de téléphone également utile à l'identification

3.5.7 Candidature

On a également décidé de créer une table *Schedules* pour les candidatures et entretiens afin qu'elle soit stocker et ainsi y avoir toujours accès : On y trouve les informations suivantes :

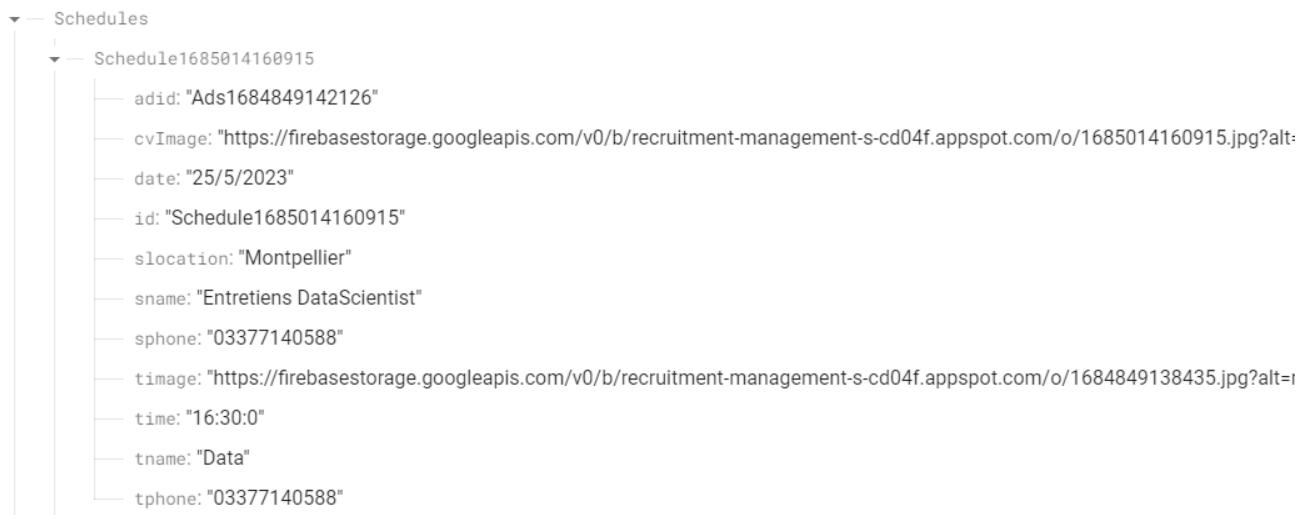


FIGURE 13 – Table *Schedules*

- **adid** : ID de l'offre d'emploi en question
- **cvImage** : Lien du CV
- **date** : Date de l'entretien
- **id** : ID de la demande d'entretiens
- **slocation** : Lieu de l'entretien
- **sname** : Nom du candidat
- **sphone** : Numéro du candidat
- **timage** : Image descriptive de l'offre d'emploi
- **time** : Heure de l'entretien
- **tname** : Nom de l'offre d'emploi
- **tphone** : Numéro du créateur de l'offre d'emploi

3.5.8 Candidats

On va donc avoir ici une table *User* pour pouvoir stocker les informations utiles de nos candidats Les données des utilisateurs suivantes y sont stocké :

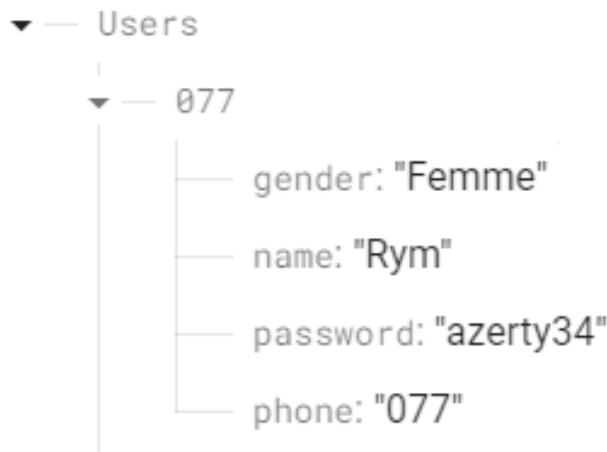


FIGURE 14 – Table User

- **Gender** : Le genre de la personne
- **Name** : Le nom de la personne
- **Password** : Le mot de passe utile à l’identification
- **Phone** : Le numéro de téléphone utile pour les offreurs d’emploi et l’identification

3.6 Profils

Nous comptons dans notre application cinq profils que nous avons dû créer.

- Profil Anonyme : Ce profil est destiné à tous les utilisateurs qui installent notre application. Ils peuvent profiter des services de base offerts par l’application sans avoir besoin de s’inscrire. En tant qu’utilisateurs anonymes, ils peuvent explorer les fonctionnalités disponibles, accéder aux informations générales et obtenir un aperçu des offres d’intérim disponibles.
- Profil Candidat Inscrit/Connecté : Ce profil est conçu pour les utilisateurs qui se sont inscrits à notre application en tant que candidats à la recherche d’opportunités d’intérim. En s’inscrivant, ils peuvent accéder à des fonctionnalités supplémentaires telles que la création d’un profil, la soumission de candidatures, la gestion des offres d’emploi et la communication avec les employeurs.
- Profil Recruteur (employeur et agence) : Ce profil est conçu pour les employeurs et les agences en même temps, les utilisateurs qui se sont inscrits en tant qu’employeurs souhaitant proposer des offres d’intérim et utilisateurs qui représentent des agences d’intérim. En utilisant ce profil, les employeurs peuvent publier des offres d’emploi, consulter les candidatures reçues, interagir avec les candidats potentiels et gérer le processus de recrutement. Les agences d’intérim inscrites peuvent bénéficier de fonctionnalités spécifiques de notre application, telles que la gestion centralisée des offres d’emploi pour leurs clients, la sélection des candidats appropriés, le suivi des missions en cours et la coordination avec les employeurs.
- Profil Gestionnaire : Ce profil est réservé aux utilisateurs qui se sont inscrits en tant que gestionnaires, tels que des administrateurs de l’application. Les gestionnaires bénéficient d’un accès privilégié et de fonctionnalités avancées leur permettant de superviser l’ensemble du système, de gérer les utilisateurs, de surveiller les activités, de générer des rapports et de maintenir la sécurité et la performance de l’application.

4 Fonctionnalités

4.1 Démarrage de l'application

Lors du démarrage de l'application nous tombons directement sur une page de chargement de l'application où le nom de l'application est affiché avec une bar de progression .Cette page est affichée grâce à l'activité SupplashActivity.

Elle vérifie les préférences partagées pour déterminer l'état de connexion de l'utilisateur. Selon l'état, l'application redirige l'utilisateur vers différentes activités telles que MainActivity (pour le candidat), AdminHomeActivity(pour l'admin), RecruiterHomeActivity (pour le recruteur) ou ManagerHomeActivity (pour le gestionnaire). Si aucun utilisateur n'est connecté, l'application vérifie également les préférences partagées pour l'introduction. Si l'introduction a déjà été affichée, l'utilisateur est redirigé vers GuestActivity, sinon, vers WelcomeActivity. Enfin, le thread est démarré pour exécuter les actions planifiées.

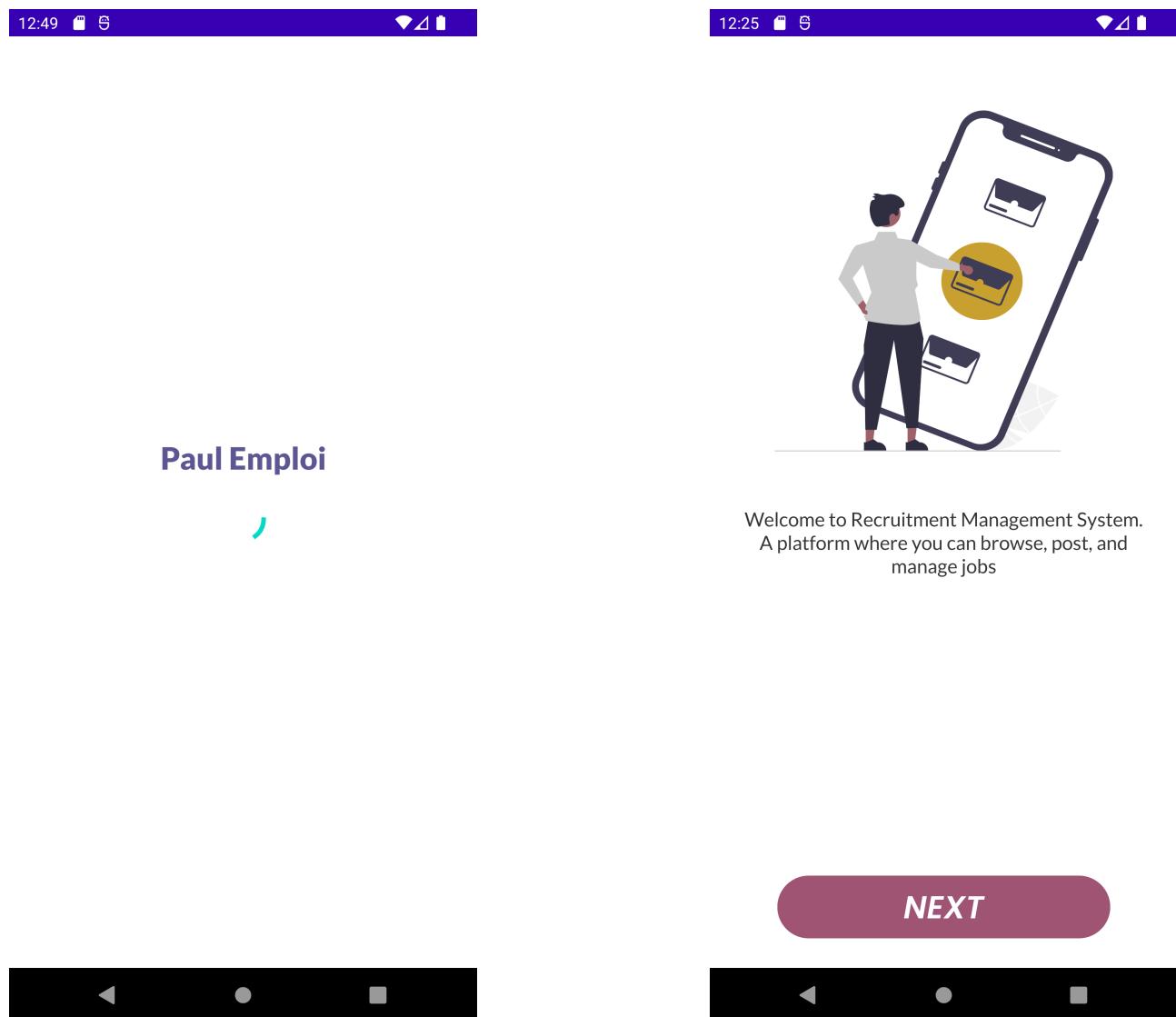


FIGURE 15 – Ecran de chargement

FIGURE 16 – Ecran de bienvenue

L'activité WelcomeActivity représente l'écran de bienvenue de l'application Android. Lorsque l'utilisateur clique sur le bouton "Next" de l'écran de bienvenue, l'activité enregistre que l'introduction a été affichée une fois en utilisant les préférences partagées. Ensuite, l'activité GuestActivity est

démarrée et l'activité WelcomeActivity est terminée, permettant à l'utilisateur de passer à l'écran suivant de l'application.

4.2 Menu de l'application

Avant de plonger dans les fonctionnalités spécifiques à chaque type d'utilisateur, nous allons d'abord vous présenter le menu, une fonctionnalité commune à tous les utilisateurs.

Le menu peut être accessible en cliquant sur le bouton situé dans le coin supérieur gauche de la page d'accueil. Ce bouton est présent dans chaque activité principale des différents utilisateurs, que nous appelons les "homeActivity". En accédant au menu, vous trouverez plusieurs options, notamment :

- Les termes et conditions : Ce bouton permet d'accéder aux termes et conditions de l'application.
- Partager l'application : Ce bouton facilite le partage de l'application en copiant un lien fictif vers le Google PlayStore.
- Noter l'application : Ce bouton vous redirige vers une page fictive pour évaluer et donner une note à l'application.
- Mes candidatures : Ce bouton vous permet de consulter les candidatures que vous avez soumises si vous êtes un candidat ou les demandes candidatures que vous avez reçues si vous êtes un employeur.
- Se connecter ou se déconnecter : Ce bouton vous offre la possibilité de vous connecter à votre compte utilisateur ou de vous déconnecter si vous êtes déjà connecté.



FIGURE 17 – Menu de l'application

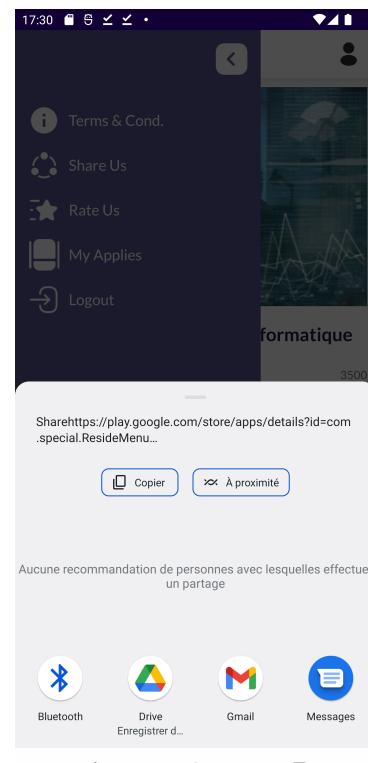


FIGURE 18 – Partage de l'application

Le menu est donc un élément central dans l'expérience utilisateur de l'application, offrant diverses options et fonctionnalités accessibles depuis n'importe quelle page principale de l'application.

4.3 Fonctionnalités pour l'utilisateur anonyme

Par la suite , l'utilisateur n'est pas connecté, il arrive sur une page avec des offres d'emploi sans filtres. Il peut spécifier sa localisation pour trouver des offres dans son secteur et chercher à l'aide de la barre de recherches les postes de son choix.

L'activité GuestActivity est l'écran d'accueil pour les utilisateurs invités dans l'application de système de recrutement. Elle affiche divers éléments de la mise en page, tels que les boutons de navigation, les boutons de compte, les boutons d'actions spécifiques, et une liste d'annonces d'emploi à l'aide d'un RecyclerView. Les interactions avec les boutons permettent de démarrer des activités spécifiques. De plus, la gestion de la fermeture de l'application est également prise en compte pour assurer une expérience utilisateur fluide.

1. Préciser sa localisation, enregistrer le lieu dans les préférences de l'utilisateur et afficher les annonces en lien avec cette localisation.
La MappingActivity lui permet de pointer sa localisation.
2. Pour chaque annonce d'offre affiché, quelque que soit l'utilisateur (anonyme ou connecté), un certain nombre de possibilité sont offertes aux utilisateurs :
 - Candidater : Pour chaque annonce/offre affichée, un bouton est disponible pour candidater à cette offre. Il est à noter que quand un utilisateur anonyme veut candidater pour une offre donnée, l'application lui demande de s'authentifier (s'il est déjà inscrit) ou de s'inscrire.
 - Mettre en favori les offres d'emploi.
3. Chercher les annonces/offres grâce à la barre de recherche.
4. S'inscrire/s'authentifier :
Quand le bouton login-btn est cliqué, l'activité SigninInitActivity est lancée. Si elle détecte des preferences partagées, elle redirige l'utilisateur automatiquement vers une page d'accueil appropriée (ManagerHomeActivity par ex). Sinon, elle lance l'activité RegisterInitActivity, qui est responsable de l'écran d'inscription initial où les utilisateurs peuvent choisir entre les rôles de candidat, de recruteur, de gestionnaire ou d'administrateur, puis lance à son tour une autre activité selon le profil souhaité, qui permet à un utilisateur anonyme de s'identifier (par exemple RegisterActivityManager si "gestionnaire" est choisi).
L'activité SigninInitActivity offre aussi la possibilité de s'inscrire. Pour cela, il faut cliquer le bouton register-btn pour accéder à l'activité RegisterInitActivity.
5. Changement de langue de l'application



FIGURE 19 – Page home

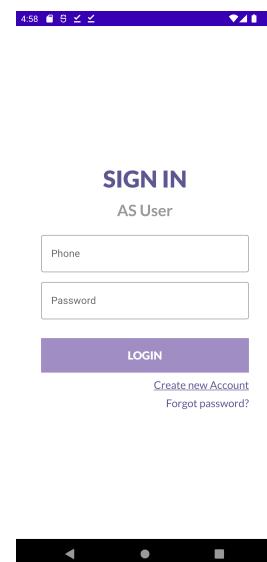


FIGURE 20 – Page de connexion

4.4 Fonctionnalités pour les demandeurs d'emploi (candidats)

1. Se connecter/S'inscrire. L'activité SigninActivity permet de se connecter et l'activité RegisterActivity lui permet de s'inscrire.
2. Spécifier sa localisation grâce à l'activité MappingActivity.
3. Parcourir et rechercher des offres d'emploi en fonction du titre du poste, du lieu ou des mots-clés. L'activité MainActivity (qui correspond à l'écran d'accueil des candidats) affiche une liste d'annonces d'emploi en utilisant un RecyclerView et un adaptateur FirebaseRecyclerViewAdapter. Les données sont récupérées depuis la base de données Firebase, liées aux vues dans le ViewHolder, et affichées dans la liste. Cela permet aux utilisateurs de consulter les annonces d'emploi disponibles dans l'application de système de recrutement.
4. Soumettre des CV et postuler à des opportunités d'emploi souhaitées. L'activité ScheduleActivity permet de postuler à des offres en entrant ses informations personnelles et sélectionner une image dans la galerie de son appareil pour soumettre son CV.
5. Planifier des entretiens avec des recruteurs et gérer les demandes d'entretien. L'activité ScheduleActivity permet aux utilisateurs de sélectionner une date et une heure pour le rendez-vous prévu. En cliquant sur le bouton "date_btn", une boîte de dialogue de sélection de date est affichée, et en cliquant sur le bouton "time_btn", une boîte de dialogue de sélection de l'heure est affichée. Si tous les champs requis sont remplis, y compris la date et l'heure sélectionnées, le processus de téléchargement de l'image est lancé. L'image sélectionnée est téléchargée sur Firebase Storage, et après un téléchargement réussi, l'URL de téléchargement est obtenue. Les détails du rendez-vous, y compris l'URL de l'image, sont stockés dans la base de données en temps réel de Firebase sous le noeud "Schedules". Pendant le processus de téléchargement, une barre de progression est affichée, et après la planification réussie, un message de succès est affiché, et l'activité se termine.
6. Modifier, mettre à jour ses informations. L'activité UserMyAccountActivity fonctionne pour afficher et mettre à jour les informations du compte utilisateur.

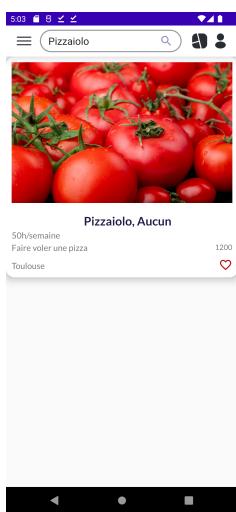


FIGURE 21 – Recherche

FIGURE 22 – Candidature

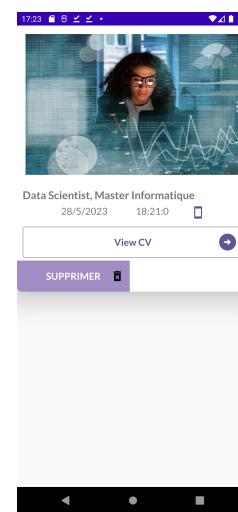


FIGURE 23 – Mes candidatures

4.5 Fonctionnalités pour les recruteurs (employeurs/agences)

1. Se connecter et s'inscrire. La classe SigninActivityRecruiter permet aux recruteurs de se connecter à l'application à

l'aide de leur numéro de téléphone et de leur mot de passe. Ils peuvent également choisir la langue de l'application et passer à l'activité d'inscription s'ils n'ont pas encore de compte. L'activité RegisterActivityRecruiter permet à un recruteur de s'inscrire. L'utilisateur remplit un formulaire avec les informations suivantes :

- Numéro de téléphone
- nom
- Mot de passe
- Gender
- Le nom de l'entreprise
- Le numéro SIREN

Lorsqu'il appuie sur le bouton d'inscription, la présence des renseignements demandées est vérifiée, et les données sont enregistrées dans une base de données Firebase, à condition qu'aucun autre recruteur n'ait déjà utilisé le même numéro de téléphone. Si l'enregistrement est réussi, l'utilisateur est redirigé vers l'écran de connexion. En revanche, s'il y a des erreurs, des messages appropriés sont affichés pour guider l'utilisateur. L'activité utilise des vues et des écouteurs d'événements pour interagir avec l'interface utilisateur et la base de données. Lorsque l'utilisateur clique sur le bouton "Sign Up Here", l'activité RegisterActivityRecruiter est lancée.

Lorsque l'utilisateur clique sur le bouton "Login", le numéro de téléphone et le mot de passe saisis sont vérifiés. Ensuite, une requête est effectuée dans la base de données Firebase pour vérifier si un recruteur est enregistré avec ce numéro de téléphone

2. Publier des annonces d'emploi avec des descriptions détaillées.

L'activité PostJobActivity permet aux recruteurs de publier des offres d'emploi. Elle permet aux utilisateurs de saisir les détails de l'offre d'emploi :

- Le titre
- Un numéro de téléphone
- La localisation du poste
- Les compétences requises
- Le niveau d'éducation
- Le montant de salaire
- Les horaires de travail
- La description du poste

Les recruteurs peuvent également sélectionner une image pour accompagner l'offre d'emploi. Une fois les informations saisies et l'image sélectionnée, l'offre d'emploi est ajoutée à la base de données Firebase.

3. Supprimer ou modifier les annonces d'emploi publiées.

UpdateAdActivity est l'activité qui permet aux recruteurs de mettre à jour les détails d'une annonce d'emploi existante, y compris l'image associée.

4. Afficher les publications du recruteur.

La classe "RecruiterHomeActivity" représente l'activité principale des recruteurs dans l'application. Elle affiche les offres d'emploi publiées par le recruteur, permet la suppression des offres, et offre des fonctionnalités supplémentaires telles que la navigation et le partage.

5. Examiner et gérer les candidatures et les CV des candidats.

L'activité ScheduleRequestManagerActivity permet aux recruteurs de gérer les demandes de planification dans l'application. Elle affiche les demandes de planification dans une liste, où chaque demande est représentée par des informations telles que l'image, le nom, la date, l'adresse, l'heure, etc. Les recruteurs peuvent consulter les CV grâce à l'activité CvActivity .

6. Accepter ou décliner les demandes d'entretien des candidats.

L'activité ScheduleRequestManagerActivity permet également de supprimer une demande ou

d'en accepter.

Par ailleurs, lorsque les recruteurs choisissent de supprimer ou d'accepter une demande, une boîte de dialogue de confirmation s'affiche pour s'assurer de leur intention avant de procéder à l'action.

7. Communiquer directement avec les candidats par téléphone.

ScheduleRequestManagerActivity permet en outre d'appeler directement les candidats.

8. Modifier les informations personnelles du compte.

La classe "RecruiterMyAccountActivity" permet aux recruteurs de gérer leur compte et d'édition leurs informations personnelles telles que le nom, le numéro de téléphone, l'e-mail et le genre. Elle utilise la classe SharedPreferences pour accéder aux préférences partagées et stocker les données de l'utilisateur. L'activité récupère les données de l'utilisateur à partir de la base de données Firebase et les affiche dans les champs de texte correspondants.

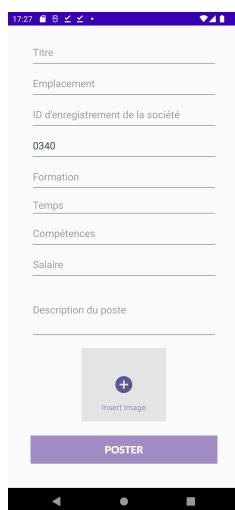


FIGURE 24 – Création d'offre

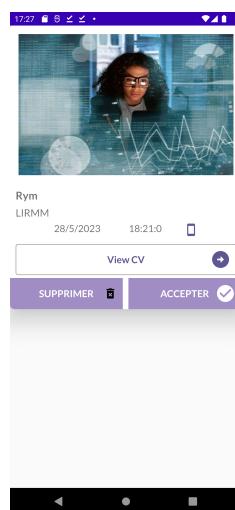


FIGURE 25 – Candidats

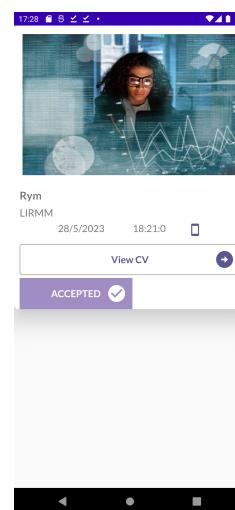


FIGURE 26 – Candidats accepté

4.6 Fonctionnalités pour les gestionnaires (Manager)

1. Se connecter/S'inscrire. La connexion se fait avec SigninActivityManager, et l'inscription avec RegisterActivityManager. Les informations requises sont :
 - Numéro de téléphone
 - Nom
 - Mot de passe
 - Genre
2. Surveiller et superviser les publications d'emplois et les candidatures au sein de leur organisation.
3. Accéder à des statistiques complètes sur les emplois publiés, les candidatures totales et les entretiens. Cela se fait à travers ManagerHomeActivity.
4. Suivre l'avancement des activités de recrutement et prendre des décisions éclairées.
Comme on l'a dit précédemment, l'activité "ManagerHomeActivity" permet aux gestionnaires d'accéder et de visualiser les statistiques liées aux annonces. Grâce à l'utilisation de Firebase Realtime Database et d'un RecyclerView, les données sont récupérées et affichées de manière conviviale, permettant aux gestionnaires de prendre des décisions informées.

4.7 Fonctionnalités pour les administrateurs (Admin)

1. Se connecter/S'inscrire. La connexion se fait avec SigninActivityManager. Les informations requises sont :
 - Numéro de téléphone
 - Mot de passe
2. Obtenir un accès administratif à l'ensemble de l'application(entreprises, annonces, plannings) depuis AdminHomeActivity.
3. Visualiser toutes les entreprises publant des annonces d'emploi et gérer leurs comptes, grâce à l'activité CompaniesActivity.
4. Surveiller et modérer toutes les annonces d'emploi et les plannings, grâce à l'activité JobsActivity, et SchedActivity respectivement.

4.8 Les autres fonctionnalités notables

de l'application **Paul Emploi** comprennent :

1. Des capacités de recherche avancées permettant aux utilisateurs d'affiner leur recherche d'emploi en fonction de divers critères.
2. Des recommandations d'emploi personnalisées basées sur les préférences des utilisateurs et leurs activités passées.
3. Une interface conviviale pour une navigation et une interaction faciles.
4. Des fonctionnalités de gestion de compte permettant aux utilisateurs, aux recruteurs et aux gestionnaires de modifier leurs informations de compte.Pour chaque utilisateur , on a une activité forgotPassword pour gérer l'oubli de mot de passe. On a essayé de travailler sur l'envoi de mail pour réinitialiser le mot de passe. Mais en dépit du temps mis , nous n'avons pas pu la mettre en place en vue de privilégier le temps restant à implémenter un maximum de fonctionnalités. Lorsqu'un utilisateur appuie sur mot de passe oublié. On le dirige vers une page où il note son identifiant de nouveau et inscrit son nouveau mot de passe en vue qu'il soit enregistré dans la base de données.
5. Intégration avec l'API de géolocalisation de Google pour afficher des informations sur la localisation des utilisateurs.

5 Bonne pratiques de conception :

Plusieurs bonnes pratiques de conception ont été suivies dans le développement de l'application Paul Emploi afin d'assurer une expérience utilisateur positive. Tout d'abord, l'application a été conçue avec une interface utilisateur claire et intuitive, avec des fonctionnalités de recherche et de planification faciles à utiliser, ainsi qu'une mise en page simple. Cela permet aux utilisateurs de naviguer facilement et de trouver les informations dont ils ont besoin. Deuxièmement, l'application a été développée avec un design adaptatif, garantissant qu'elle fonctionne bien sur différents appareils et tailles d'écran. Cela permet aux utilisateurs d'accéder à l'application depuis n'importe quel appareil, ce qui la rend pratique à utiliser en déplacement. Enfin, l'application a été développée en tenant compte de la sécurité, en utilisant des serveurs sécurisés et un stockage de données chiffré pour protéger les informations des utilisateurs. Cela contribue à garantir que les données des utilisateurs sont conservées de manière privée et sécurisée. Dans l'ensemble, l'application Paul Emploi a été conçue en gardant à l'esprit l'expérience utilisateur, offrant une solution pratique et conviviale pour trouver et se connecter avec des recruteurs.

6 Verrous et solutions

Dans le cadre du projet de création d'agences d'intérim au sein de notre application mobile, nous avons rencontré différents verrous techniques et organisationnels. Parmi ces verrous, l'un des défis majeurs était la mise en place de la fonctionnalité de localisation pour permettre aux utilisateurs de trouver rapidement les agences d'intérim les plus proches de leur emplacement.

Pour surmonter ce défi, nous avons adopté une approche basée sur l'intégration de services de géolocalisation tels que Google Maps. Nous avons utilisé le service FusedLocationProviderClient de Google Play Services pour obtenir la dernière position connue de l'appareil de l'utilisateur. Cette solution nous a permis d'offrir une fonctionnalité de localisation précise et en temps réel, améliorant ainsi l'expérience des utilisateurs en facilitant la recherche et la sélection des agences d'intérim.

Comme évoqué plus tôt, nous avons fait face à une contrainte technique majeure qui nous a empêchés d'envoyer des e-mails automatiques aux utilisateurs lorsqu'ils demandent une réinitialisation de leur mot de passe. Bien que nous reconnaissions l'importance de cette fonctionnalité pour offrir une expérience utilisateur optimale, nous avons dû trouver une solution alternative pour pallier cette limitation et être les plus productifs possible.

En effet, nous pallions à ce problème en demandant à l'utilisateur d'écrire son identifiant (son numéro de téléphone) et de changer le mot de passe. Cela engendre un problème de sécurité, mais on a préféré trouver une solution que d'abandonner cette fonctionnalité.

7 Conclusion

En conclusion, ce projet a abouti au développement d'une application mobile destinée à la gestion des offres et candidatures pour le travail en intérim. L'application offre une gamme de services aux différents utilisateurs, tels que les candidats inscrits, les employeurs, les agences d'intérim, les gestionnaires et les administrateurs.

Les fonctionnalités mises en œuvre visent à rationaliser le processus de recherche d'emploi et de recrutement, à faciliter la communication efficace entre les demandeurs d'emploi et les recruteurs, ainsi qu'à offrir une supervision et une administration complètes aux gestionnaires et aux administrateurs. Cette synthèse met en évidence la mise en œuvre réussie de l'application Paul Emploi, couvrant un large éventail de fonctionnalités pour répondre aux besoins diversifiés des utilisateurs, des recruteurs, des gestionnaires et des administrateurs.

Nous avons réussi à accomplir une grande partie du projet avec succès, mais malheureusement, certaines fonctionnalités n'ont pas pu être implémentées. Nous nous sommes investis pleinement dans ce travail, en mettant tout notre cœur pour affiner notre application. Nous espérons sincèrement que vous, en tant que professeur, serez satisfait de nos réalisations et ressentirez une certaine fierté quant à l'application de vos enseignements. Si nous avions disposé de plus de temps, nous aurions pu aller encore plus loin dans notre démarche.

Dans ce qui suit, nous allons expliquer quelques concepts en détail, pour aller plus loin dans notre présentation du projet, en commençant par les activités associés à chaque profil d'utilisation.

8 Annexes - Activités

8.1 Lien github et précision

Veuillez trouver l'ensemble de notre projet dans le lien github suivant [Projet Mobile](https://github.com/SofianEtt/ProjetMobile)(<https://github.com/SofianEtt/ProjetMobile>), en plus du projet le dépôt est également composé d'une présentation ainsi que de deux vidéo démonstratives de notre application.

8.2 Activités communes à tous profils d'utilisateurs

8.2.1 TermsActivity

L'activité "TermsActivity" affiche les termes et conditions d'utilisation. Lorsque cette activité est lancée, elle affiche une interface utilisateur contenant un WebView, qui est un composant permettant d'afficher du contenu Web.

La WebView est configurée pour activer JavaScript et charger une URL spécifique qui contient les termes et conditions, dans ce cas, l'URL est "<https://www.privacypolicygenerator.info/live.php?token=EBWsDbEAR09vCtKR5de9wOwxedHAFcSt>".

L'activité comprend également un bouton "Retour" représenté par un ImageView, qui permet à l'utilisateur de revenir à l'écran précédent ou de quitter cette activité et revenir à l'activité parente. Lorsque l'utilisateur appuie sur le bouton "Retour", l'activité se termine.

Pour assurer une expérience utilisateur fluide, un WebClient est configuré pour la WebView. Cela permet de contrôler le chargement des URL dans la WebView elle-même, au lieu d'ouvrir les liens dans un navigateur externe. Ainsi, lorsque l'utilisateur clique sur un lien dans les termes et conditions, la WebView ouvre ce lien à l'intérieur de la WebView, permettant à l'utilisateur de naviguer dans le contenu sans quitter l'application.

8.3 Activités pour le profil "anonyme"

Ces activités sont définies dans le dossier "Init".

8.3.1 SplashActivity

Le module "SplashActivity" est responsable de l'écran de lancement de l'application. Il gère les différentes redirections en fonction de l'état de connexion de l'utilisateur.

Lorsque l'activité est créée, elle charge le contenu de l'interface utilisateur à partir du fichier "activity_splash.xml".

Dans le module SplashActivity, un thread est utilisé pour vérifier l'état de connexion de l'utilisateur en consultant les préférences partagées.

Si les préférences partagées contiennent une clé "phone", cela signifie qu'un utilisateur est déjà connecté. En fonction de la valeur de la clé "status" dans les préférences partagées, l'application redirige l'utilisateur vers l'activité correspondante. Par exemple, si le statut est "user", l'application

redirige vers MainActivity. Si le statut est "admin", l'application redirige vers AdminHomeActivity, et ainsi de suite.

Si les préférences partagées ne contiennent pas de clé "phone", cela signifie que l'utilisateur n'est pas connecté. L'application vérifie ensuite si les préférences partagées contiennent une clé "intro". Si c'est le cas, cela signifie que l'activité Welcome a déjà été affichée, et l'application redirige vers GuestActivity. Sinon, l'application redirige vers WelcomeActivity pour afficher l'introduction.

8.3.2 WelcomeActivity

Cette classe est responsable de l'affichage de l'introduction de l'application. Lorsque l'activité est créée, elle charge le contenu de l'interface utilisateur à partir du fichier "activity_welcome.xml". Dans le module WelcomeActivity, un bouton appelé "next_btn" est initialisé en utilisant findViewById pour le lier à l'élément correspondant de l'interface utilisateur. Un écouteur d'événements est défini pour le bouton "next_btn". Lorsque l'utilisateur appuie sur le bouton, l'écouteur d'événements est déclenché et les actions suivantes sont effectuées :

1. Une instance de SharedPreferences.Editor est créée pour modifier les préférences partagées avec le nom "INTRO".
2. La clé "intro" est définie sur la valeur "true" dans les préférences partagées pour indiquer que l'introduction a été affichée.
3. Les modifications sont appliquées à l'éditeur des préférences partagées.
4. Une nouvelle intention (Intent) est créée pour lancer l'activité GuestActivity.
5. L'intention est lancée à l'aide de la méthode startActivity.
6. L'activité WelcomeActivity se termine avec la méthode finish.

8.3.3 RegisterInitActivity

Le module "RegisterInitActivity" gère l'écran d'inscription initial de l'application. Il permet à l'utilisateur de choisir entre trois types d'utilisateurs différents : candidat, recruteur et gestionnaire. De plus, il offre également la possibilité de passer à l'écran de connexion.

Lorsque l'activité est créée, elle charge le contenu de l'interface utilisateur à partir du fichier "activity_register_init.xml".

Dans le module RegisterInitActivity, les éléments de l'interface utilisateur, tels que les boutons et le texte, sont initialisés en utilisant findViewById pour les lier aux éléments correspondants de l'interface utilisateur.

Des écouteurs d'événements sont définis pour chaque bouton et le texte "login_tv". Lorsque l'utilisateur clique sur l'un des boutons, l'écouteur d'événements correspondant est déclenché et les actions suivantes sont effectuées :

1. Une nouvelle intention (Intent) est créée pour lancer l'activité correspondante à l'utilisateur sélectionné. Par exemple, si l'utilisateur choisit le bouton "candidate_btn", l'intention lancera l'activité RegisterActivity.
2. L'intention est lancée à l'aide de la méthode startActivity.
3. L'activité RegisterInitActivity se termine avec la méthode finish.

Le même processus est suivi pour les boutons "recruiter_btn" et "manager_btn" afin de lancer les activités correspondantes : RegisterActivityRecruiter et RegisterActivityManager.

De plus, lorsque l'utilisateur clique sur le texte "login_tv", une intention est créée pour lancer l'activité SigninInitActivity, qui représente l'écran de connexion. L'intention est lancée et l'activité RegisterInitActivity se termine également.

8.3.4 SigninInitActivity

Le module "SigninInitActivity" gère l'écran d'authentification initial de l'application. Il permet à l'utilisateur de choisir entre quatre types d'utilisateurs différents : candidat, recruteur, gestionnaire et administrateur. Il offre également la possibilité de passer à l'écran d'inscription et de sélectionner la langue de l'application.

Lorsque l'activité est créée, elle vérifie d'abord si l'utilisateur est déjà connecté en vérifiant les préférences partagées "USER". Si les informations d'utilisateur sont présentes, l'activité redirige automatiquement vers l'écran d'accueil approprié en fonction du statut de l'utilisateur.

Ensuite, le contenu de l'interface utilisateur est chargé à partir du fichier "activity_signin_init.xml".

Dans le module SigninInitActivity, les éléments de l'interface utilisateur, tels que les boutons et le texte, sont initialisés en utilisant findViewById pour les lier aux éléments correspondants de l'interface utilisateur.

Des écouteurs d'événements sont définis pour chaque bouton et le texte "register_tv". Lorsque l'utilisateur clique sur l'un des boutons, l'écouteur d'événements correspondant est déclenché et les actions suivantes sont effectuées :

1. Une nouvelle intention (Intent) est créée pour lancer l'activité correspondante à l'utilisateur sélectionné. Par exemple, si l'utilisateur choisit le bouton "candidate_btn", l'intention lancera l'activité SigninActivity.
2. L'intention est lancée à l'aide de la méthode startActivity.
3. L'activité SigninInitActivity se termine avec la méthode finish.

Le même processus est suivi pour les boutons "recruiter_btn", "manager_btn" et "admin_btn" afin de lancer les activités correspondantes : SigninActivityRecruiter, SigninActivityManager et SigninActivityAdmin.

De plus, lorsque l'utilisateur clique sur le texte "register_tv", une intention est créée pour lancer l'activité RegisterInitActivity, qui représente l'écran d'inscription. L'intention est lancée et l'activité SigninInitActivity se termine également.

Enfin, un sélecteur de langue est ajouté à l'activité à l'aide d'un Spinner. Lorsque l'utilisateur sélectionne une langue, l'écouteur d'événements correspondant est déclenché et les actions suivantes sont effectuées :

1. La langue sélectionnée est récupérée à partir du Spinner.
2. En fonction de la langue sélectionnée, la méthode setlanguage est appelée pour mettre à jour la configuration de la ressource avec la nouvelle langue.
3. Une nouvelle intention est créée pour relancer l'activité SigninInitActivity afin d'appliquer les modifications de langue.
4. L'intention est lancée et l'activité SigninInitActivity se termine.

En résumé, le module SigninInitActivity gère l'écran d'authentification initial de l'application. Il permet à l'utilisateur de choisir entre différents types d'utilisateurs, de passer à l'écran d'inscription et de sélectionner la langue de l'application.

8.3.5 GuestActivity

Le premier module de l'application est appelé "GuestActivity". Ce module gère l'interface utilisateur pour les utilisateurs invités ou non connectés.

Lorsque l'application démarre, l'utilisateur anonyme est dirigé vers l'activité GuestActivity. Cette activité affiche différentes fonctionnalités et options pour les utilisateurs invités.

L'interface utilisateur de l'activité GuestActivity est composée des éléments suivants :

- Bouton de navigation : Lorsque l'utilisateur clique sur ce bouton, un menu de navigation apparaît, offrant des options supplémentaires à l'utilisateur.
- Bouton "Mon compte" : Ce bouton permet à l'utilisateur d'accéder à la page de connexion ou d'inscription pour créer un compte ou se connecter.
- RecyclerView (Liste déroulante) : Cette vue affiche une liste d'éléments, dans ce cas, des offres d'emploi. Chaque élément de la liste contient des informations telles que l'image, le titre, le niveau d'éducation recherché, l'emplacement du poste,, le salaire, etc.
- menu : Il y a plusieurs boutons supplémentaires pour effectuer différentes actions. Ces boutons comprennent les boutons "Conditions", "Partager" et "Évaluer". Le bouton "Conditions" permet à l'utilisateur de consulter les conditions d'utilisation de l'application. Le bouton "Partager" permet à l'utilisateur de partager l'application avec d'autres personnes. Le bouton "Évaluer" permet à l'utilisateur de donner une évaluation à l'application.
- Bouton de fermeture de menu : Ce bouton permet à l'utilisateur de fermer le menu de navigation.

8.4 Activités pour le profil "candidat"

Ces activités sont définies dans le dossier "User"

8.4.1 MainActivity

Cette classe est responsable de l'affichage de l'écran principal du candidat et de la gestion des fonctionnalités associées.

Lorsque l'activité est créée, elle commence par charger l'interface utilisateur correspondante à partir du fichier de mise en page "activity_main.xml". Cela comprend plusieurs éléments tels que des boutons, une barre de recherche, une vue de liste déroulante, des images, etc.

La classe MainActivity étend la classe AppCompatActivity, qui fournit des fonctionnalités pour la compatibilité avec les versions antérieures d'Android. Cela garantit que l'application fonctionne correctement sur différentes versions du système d'exploitation Android.

Le module MainActivity utilise la bibliothèque FirebaseUI pour interagir avec la base de données Firebase. Il y a une référence à la base de données Firebase créée à l'aide de la classe DatabaseReference.

L'activité MainActivity utilise également un adaptateur FirebaseRecyclerAdapter pour afficher les offres d'emploi récupérées de la base de données Firebase dans la vue de liste déroulante. Cet adaptateur gère la liaison des données aux éléments de l'interface utilisateur et permet leur affichage dans une disposition de liste.

L'utilisateur peut effectuer différentes actions à partir de l'écran principal de l'application. Par exemple, en utilisant la barre de recherche, l'utilisateur peut entrer du texte et effectuer une recherche dans la base de données Firebase pour trouver des éléments correspondants.

Il y a plusieurs boutons dans l'interface utilisateur, tels que le bouton de navigation, le bouton "Mon compte", le bouton "Mes candidatures", etc. Lorsque l'utilisateur clique sur ces boutons, des actions spécifiques sont déclenchées. Par exemple, en cliquant sur le bouton "Mon compte", l'activité MainActivity démarre une nouvelle activité appelée UserMyAccountActivity, qui affiche les informations du compte de l'utilisateur.

Lorsque l'utilisateur clique sur un élément de la liste déroulante, l'activité MainActivity lance une autre activité appelée ScheduleActivity pour afficher les détails de cet élément spécifique. Les informations nécessaires telles que l'image associée à l'offre, le titre, la description...sont transmises à cette activité via des extras dans l'intention.

De plus, l'activité MainActivity gère également des fonctionnalités supplémentaires telles que le partage de l'application et l'évaluation de l'application.

Enfin, l'activité MainActivity implémente les fonctionnalités du menu latéral d'crité précédemment (dans la section GuestActivity).

8.4.2 ScheduleActivity

Cette classe est responsable de la planification des entretiens d'embauche.

Lorsque le candidat accède à l'écran de planification, il voit plusieurs champs à remplir, tels que le nom, l'adresse, le numéro de téléphone, ainsi que des boutons pour sélectionner la date et l'heure. Le candidat peut également ajouter un CV et/ou une lettre de motivation sous format image en cliquant sur un bouton correspondant.

Une fois que tous les champs requis sont remplis, l'utilisateur peut confirmer la planification en cliquant sur un bouton de confirmation. Avant de confirmer, l'application vérifie si toutes les informations nécessaires ont été fournies. Si des informations manquent, un message d'erreur est affiché demandant à l'utilisateur de remplir tous les champs.

Lorsque l'utilisateur confirme la planification, l'application enregistre les informations saisies dans une base de données Firebase. Les données enregistrées comprennent le nom, le numéro de téléphone, l'adresse, la date et l'heure de l'événement. De plus, l'application génère un identifiant unique pour cette planification.

L'application permet également au candidat de sélectionner un CV/lettre de motivation depuis son répertoire d'images. Lorsqu'il clique sur le bouton d'ajout d'image, une galerie d'images s'ouvre et lui permet de choisir une image à partir de là. L'image sélectionnée est ensuite téléchargée sur Firebase Storage et son URL de téléchargement est également enregistrée dans la base de données Firebase.

Une fois la planification enregistrée avec succès, l'application notifie l'utilisateur en affichant un message de succès et en terminant l'écran de planification. En outre, l'application met à jour les informations dans une autre partie de la base de données Firebase pour refléter le nombre de candidatures effectuées pour cet événement spécifique.

8.4.3 MappingActivity

La classe "MappingActivity" est une activité dans une application Android qui permet de récupérer et d'afficher la position géographique de l'utilisateur sur une carte Google Maps. Lorsque l'activité est créée, elle utilise le service FusedLocationProviderClient de Google Play Services pour obtenir la dernière position connue de l'appareil de l'utilisateur.

Si l'application est exécutée sur un appareil avec Android Marshmallow (version 6.0) ou une version ultérieure, l'activité vérifie d'abord si l'autorisation d'accès à la localisation a été accordée par l'utilisateur. Si l'autorisation n'a pas été accordée, l'activité demande à l'utilisateur de l'accorder en affichant une demande de permission.

Une fois que l'autorisation est accordée, l'activité utilise fusedLocationProviderClient.getLastLocation() pour récupérer la dernière position connue de l'appareil. Si une position valide est disponible, les coordonnées de latitude et de longitude sont enregistrées dans des variables.

L'activité affiche ensuite un message Toast pour indiquer que la localisation a été obtenue avec succès. Elle utilise un SupportMapFragment pour afficher la carte Google Maps dans l'interface utilisateur. Lorsque la carte est prête, c'est-à-dire lorsque la méthode onMapReady est appelée, l'activité crée un marqueur sur la carte en utilisant les coordonnées de latitude et de longitude récupérées. Le titre du marqueur est défini à l'aide d'une valeur provenant des préférences partagées de l'application.

Enfin, la caméra de la carte est animée pour afficher le marqueur et zoomer sur sa position, permettant ainsi à l'utilisateur de visualiser sa position géographique sur la carte.

8.4.4 RegisterActivity

Le module "RegisterActivity" de l'application est responsable de l'enregistrement des nouveaux candidats. Lorsque l'utilisateur accède à l'activité d'enregistrement, il est invité à saisir ses informations personnelles telles que son nom, son numéro de téléphone, son mot de passe et son genre.

L'activité comporte des champs de saisie pour le numéro de téléphone, le nom, le mot de passe et la confirmation du mot de passe. Il y a également un menu déroulant permettant de sélectionner le genre. L'utilisateur doit remplir tous les champs obligatoires et choisir un genre valide avant de pouvoir continuer.

Une fois que l'utilisateur a rempli tous les champs et appuyé sur le bouton "Continuer", les informations saisies sont vérifiées. Si toutes les informations sont valides, une vérification est effectuée pour s'assurer qu'il n'existe pas déjà un utilisateur enregistré avec le même numéro de téléphone. Si l'utilisateur existe déjà, un message d'erreur est affiché. Sinon, les informations de l'utilisateur sont enregistrées dans la base de données Firebase.

Si l'enregistrement est réussi, l'utilisateur est redirigé vers l'activité de connexion. Sinon, un message d'erreur approprié est affiché pour indiquer la raison de l'échec de l'enregistrement.

La classe offre aussi la possibilité de basculer vers l'écran de connexion des candidats.

8.4.5 SigninActivity

Le module "SigninActivity" de l'application gère l'authentification des utilisateurs. Lorsque les utilisateurs accèdent à l'activité de connexion, ils sont présentés avec une interface utilisateur contenant plusieurs composants pour entrer leurs informations de connexion.

Les composants de l'interface utilisateur dans l'activité "SigninActivity" sont les suivants

- Champ de saisie du numéro de téléphone : C'est un champ texte où l'utilisateur doit entrer son numéro de téléphone.
- Champ de saisie du mot de passe : C'est un champ texte où l'utilisateur doit entrer son mot de passe.
- Bouton de connexion : C'est un bouton qui permet à l'utilisateur de se connecter une fois qu'il a saisi ses informations de connexion.
- Texte "Créer un nouveau compte" : C'est un texte cliquable qui redirige l'utilisateur vers l'activité d'inscription s'il clique dessus.
- Texte "Mot de passe oublié ?" : C'est un texte cliquable qui redirige l'utilisateur vers l'activité de réinitialisation du mot de passe s'il clique dessus.
- Menu déroulant de la langue : C'est un menu déroulant qui affiche les options de langue disponibles (par exemple, anglais, français). L'utilisateur peut sélectionner sa langue préférée à partir de ce menu.

Une fois que le candidat a saisi son numéro de téléphone et son mot de passe, il peut appuyer sur le bouton de connexion. Les informations de connexion sont vérifiées en interrogeant la base de données Firebase pour trouver un utilisateur correspondant au numéro de téléphone saisi. Si le candidat est trouvé et que le mot de passe correspond, il est connecté avec succès et redirigé vers l'activité principale de l'application. Sinon, des messages d'erreur appropriés sont affichés pour indiquer les problèmes de connexion.

De plus, l'activité "SigninActivity" comporte également des fonctionnalités supplémentaires, telles que la possibilité de changer la langue de l'application. Lorsque l'utilisateur sélectionne une langue dans le menu déroulant de la langue, la langue de l'application est mise à jour en conséquence à l'aide de la méthode "setlanguage". Cela permet d'offrir une expérience utilisateur multilingue.

8.4.6 ForgotPasswordUser

Le module "ForgotPasswordUser" de l'application gère la fonctionnalité de réinitialisation du mot de passe pour les utilisateurs candidats. Lorsque les utilisateurs accèdent à l'activité "ForgotPasswordUser", ils sont présentés avec une interface utilisateur contenant plusieurs composants pour entrer les informations nécessaires à la réinitialisation du mot de passe.

Les composants de l'interface utilisateur dans l'activité "ForgotPasswordUser" sont les suivants :

- Champ de saisie du numéro de téléphone : C'est un champ texte où l'utilisateur doit entrer son numéro de téléphone.
- Champ de saisie du nouveau mot de passe : C'est un champ texte où l'utilisateur doit entrer son nouveau mot de passe.
- Champ de saisie de confirmation du mot de passe : C'est un champ texte où l'utilisateur doit entrer à nouveau le nouveau mot de passe pour confirmer.
- Bouton de validation : C'est un bouton qui permet à l'utilisateur de valider la réinitialisation du mot de passe une fois qu'il a saisi les informations requises.

Une fois que l'utilisateur a saisi son numéro de téléphone, son nouveau mot de passe et la confirmation du mot de passe, il peut appuyer sur le bouton de validation. Les informations sont vérifiées en interrogeant la base de données Firebase pour trouver un utilisateur correspondant au numéro de téléphone saisi. Si l'utilisateur est trouvé, le mot de passe de cet utilisateur est mis à jour avec le nouveau mot de passe saisi. Un message de succès est affiché pour indiquer que le mot de passe a été modifié avec succès.

Si aucun utilisateur n'est enregistré avec le numéro de téléphone saisi, un message d'erreur approprié est affiché pour indiquer qu'aucun utilisateur n'est enregistré avec ce numéro.

8.4.7 schedulerequest

Le module "ScheduleRequestActivity" de l'application gère l'affichage des demandes de planification pour les candidats. Lorsque les utilisateurs accèdent à l'activité "ScheduleRequestActivity", une liste de demandes de planification associées à leur numéro de téléphone est affichée.

L'activité "ScheduleRequestActivity" contient les composants suivants :

- RecyclerView (rv) : C'est un composant qui affiche les demandes de planification sous forme de liste défilante. Les demandes de planification sont chargées à partir de la base de données Firebase et sont affichées dans le RecyclerView à l'aide d'un adaptateur.
- ImageView (back_btn) : C'est une icône de flèche arrière qui permet à l'utilisateur de revenir en arrière et de quitter l'activité "ScheduleRequestActivity".

Lorsque l'activité est créée, le RecyclerView est configuré avec un gestionnaire de disposition linéaire et une taille fixe. Ensuite, les données des demandes de planification sont récupérées de la

base de données Firebase en fonction du numéro de téléphone de l'utilisateur. Un adaptateur FirebaseRecyclerAdapter est utilisé pour afficher les demandes de planification dans le RecyclerView.

Chaque élément de demande de planification dans le RecyclerView est affiché à l'aide de l'adaptateur MyViewHolder_Schedule. Les détails de chaque demande, tels que l'image de l'enseignant, le nom, la date et l'heure de la demande, sont extraits de l'objet ScheduleModule correspondant. L'image de l'enseignant est chargée à l'aide de Picasso et affichée dans ImageView. Le nom de l'enseignant, la date et l'heure de la demande sont affichés dans les TextView correspondants.

De plus, chaque élément de demande de planification comporte les fonctionnalités suivantes :

- Lorsque l'utilisateur appuie sur le bouton d'appel (call_btn), une intention d'appel est déclenchée pour composer le numéro de téléphone de l'enseignant associé à la demande.
- Lorsque l'utilisateur appuie sur le bouton de suppression (remove_btn), une boîte de dialogue de confirmation apparaît pour confirmer la suppression de la demande de planification. Si l'utilisateur choisit de supprimer la demande, l'entrée correspondante est supprimée de la base de données Firebase.

8.4.8 myaccount

Le module "UserMyAccountActivity" de l'application gère les fonctionnalités liées au compte utilisateur d'un candidat. Lorsque un candidat accède à l'activité "UserMyAccountActivity", les informations de son compte, telles que le nom, le numéro de téléphone, l'e-mail et le genre, sont affichées.

L'activité "UserMyAccountActivity" contient les composants suivants :

- TextInputEditText (editTextPhone, email_box, name_box) : Ce sont des champs de saisie de texte permettant à l'utilisateur de modifier les informations de son compte, telles que le nom et l'e-mail.
- FrameLayout (edit_btn, okay_btn) : Ce sont des boutons permettant à l'utilisateur d'éditionner les informations de son compte et de valider les modifications. Lorsque l'utilisateur appuie sur le bouton "Edit" (edit_btn), les champs de saisie de texte deviennent modifiables et le bouton "Edit" est masqué, tandis que le bouton "OK" (okay_btn) pour valider les modifications devient visible.
- Spinner (gender_sp) : C'est une liste déroulante permettant à l'utilisateur de sélectionner son genre, tel que "Male" ou "Female".

Lorsque l'activité est créée, les informations du compte utilisateur sont récupérées à partir de la base de données Firebase en utilisant le numéro de téléphone de l'utilisateur. Les informations sont ensuite affichées dans les champs de saisie de texte correspondants et le genre sélectionné est affiché dans le Spinner.

L'utilisateur peut appuyer sur le bouton "Edit" pour activer la modification des informations du compte. Les champs de saisie de texte deviennent modifiables et le bouton "Edit" est masqué, tandis que le bouton "OK" devient visible. Lorsque l'utilisateur appuie sur le bouton "OK", les modifications sont validées et les nouvelles informations du compte utilisateur sont enregistrées dans la base de données Firebase.

8.5 Activités pour le profil "recruteur"

8.5.1 RecruiterHomeActivity

Le premier module de l'application est la classe "RecruiterHomeActivity". Voici les composants visuels importants de cette classe :

- ImageView : nav_btn, myAccount_btn

- RecyclerView : rv
- FloatingActionButton : fab

La classe "RecruiterHomeActivity" est une activité qui représente l'écran d'accueil pour les recruteurs de l'application. Elle affiche une liste d'annonces publiées par les recruteurs.

Lorsque l'activité est créée, les composants visuels sont initialisés et les listeners sont définis pour répondre aux actions de l'utilisateur.

- Lorsque l'utilisateur clique sur le bouton fab, une intention est créée pour démarrer l'activité "PostJobActivity". Cela permet aux recruteurs de publier de nouvelles offres d'emploi.
- Le RecyclerView "rv" est configuré avec un LinearLayoutManager pour afficher la liste des annonces. Les annonces sont récupérées à partir de la base de données Firebase et affichées à l'aide d'un adaptateur FirebaseRecyclerViewAdapter personnalisé.
- Le bouton "myApplies_btn" permet de passer à l'activité "ScheduleRequestManagerActivity" pour afficher les demandes d'emploi reçues par les recruteurs.
- Le bouton "myAccount_btn" permet de passer à l'activité "RecruiterMyAccountActivity" pour afficher et modifier les informations du compte du recruteur.
- Le bouton "nav_btn" est utilisé pour afficher un menu de navigation latéral.
- Les boutons "terms_btn", "share_btn", "rate_btn" et "logout_btn" sont utilisés pour afficher des fonctionnalités supplémentaires telles que les conditions d'utilisation, le partage de l'application, l'évaluation de l'application et la déconnexion du compte du recruteur.
- Lorsque l'utilisateur clique sur le bouton "menu_close_btn", le menu de navigation latéral est fermé en animant sa translationX et son alpha.
- Lorsque l'utilisateur clique sur le bouton "nav_btn", le menu de navigation latéral est ouvert en animant sa translationX et son alpha.
- Les annonces affichées dans le RecyclerView peuvent être supprimées en cliquant sur le bouton de suppression correspondant à chaque annonce. Une boîte de dialogue de confirmation est affichée pour confirmer la suppression de l'annonce.
- Les annonces affichées peuvent également être modifiées en cliquant sur le bouton d'édition correspondant à chaque annonce. Cela ouvre l'activité "UpdateAdActivity" pour permettre la modification des détails de l'annonce.

8.5.2 CvActivity

Le rôle principal de cette classe est d'afficher le CV d'un candidat sélectionné.

Lorsque l'activité est créée, elle charge le contenu de l'interface utilisateur à partir du fichier "activity_cv.xml". Cette interface utilisateur contient notamment un composant ImageView, qui sera utilisé pour afficher l'image du CV.

Le module CvActivity est responsable de récupérer l'image du CV à afficher. Il reçoit cette information grâce à une intention (Intent) provenant d'une autre partie de l'application. L'intention contient l'URL de l'image du CV.

Pour afficher l'image du CV, le module utilise la bibliothèque Picasso. Picasso est une bibliothèque Android populaire pour le chargement et la mise en cache d'images. Il prend en charge le chargement d'images à partir d'URLs, de fichiers locaux, etc. Dans ce cas, il charge l'image à partir de l'URL spécifiée dans l'Intent.

Avant de charger l'image, le module affiche un logo temporaire (R.drawable.round_logo) dans l'ImageView pour indiquer que l'image est en cours de chargement. Ensuite, il utilise la méthode "load" de Picasso pour charger l'image à partir de l'URL et l'afficher dans l'ImageView.

En résumé, le module CvActivity est responsable de l'affichage du CV d'un candidat sélectionné. Il récupère l'URL de l'image du CV à partir d'une intention, utilise la bibliothèque Picasso pour charger

et afficher cette image dans un composant ImageView.

8.5.3 ForgotPasswordRecruiter

La classe "ForgotPasswordRecruiter" affiche un écran pour les recruteurs afin de les permettre de reinitialiser leur mot de passe en cas d'oubli.

Dans le layout, il y a plusieurs vues, notamment des TextInputEditText avec les identifiants "phone_box", "pwd_box" et "cpwd_box", ainsi qu'un FrameLayout avec l'identifiant "okay_btn". Ces vues sont référencées dans le code de l'activité en utilisant la méthode "findViewById".

Une référence à la base de données Firebase est obtenue en utilisant la méthode "FirebaseDatabase.getInstance().getReference()".

Un écouteur d'événement est attaché au bouton "okay_btn" en utilisant la méthode "setOnClickListener". Lorsque le bouton est cliqué, le code à l'intérieur de la méthode "onClick" est exécuté.

Les valeurs saisies dans les TextInputEditText sont récupérées et stockées dans des variables locales. Ensuite, des vérifications sont effectuées sur les valeurs saisies. Si le champ du numéro de téléphone est vide, une erreur est affichée. Si le champ du mot de passe est vide, un message d'erreur est affiché. Si les mots de passe saisis ne correspondent pas, un message d'erreur est affiché. Si la longueur du mot de passe est inférieure à 8 caractères, un message d'erreur est affiché.

Ensuite, une requête est effectuée sur la base de données Firebase pour vérifier si un recruteur est enregistré avec le numéro de téléphone donné. Si un enregistrement correspondant est trouvé, le mot de passe du recruteur est mis à jour avec le nouveau mot de passe saisi. Sinon, un message d'erreur est affiché.

Finalement, lors de la réussite de la mise à jour du mot de passe, un message de confirmation est affiché et l'activité se termine.

8.5.4 PostJobActivity

La classe "PostJobActivity" affiche un écran pour les recruteurs pour publier une offre d'emploi.

Dans le layout, il y a plusieurs vues, notamment des EditText avec les identifiants "name_box", "timing_box", "salary_box", "skills_box", "phone_box", "location_box", "education_box", "desc_box" et "cregis_box", ainsi que des FrameLayout avec les identifiants "post_btn" et "addImage_btn". Ces vues sont référencées dans le code de l'activité en utilisant la méthode "findViewById".

Il y a aussi une ImageView avec l'identifiant "imageView" pour afficher l'image sélectionnée et une ProgressBar avec l'identifiant "progressBar" pour afficher la progression du téléchargement de l'image.

Une référence à la base de données Firebase est obtenue en utilisant la méthode "FirebaseDatabase.getInstance().getReference()". Une référence au stockage Firebase est obtenue en utilisant la méthode "FirebaseStorage.getInstance().getReference()".

Dans la méthode "onCreate", un écouteur d'événement est attaché au bouton "back_btn" pour revenir à l'activité précédente lorsque le bouton est cliqué.

Un autre écouteur d'événement est attaché au bouton "addImage_btn". Lorsque le bouton est cliqué, une intention est créée pour sélectionner une image à partir de la galerie de photos du téléphone. L'image sélectionnée est ensuite affichée dans l'ImageView "imageView".

Les données de l'utilisateur sont récupérées à partir des préférences partagées. Le numéro de téléphone de l'utilisateur est utilisé pour remplir le champ correspondant dans l'interface utilisateur.

Un autre écouteur d'événement est attaché au bouton "post_btn". Lorsque le bouton est cliqué, les valeurs saisies dans les champs sont récupérées et stockées dans des variables locales.

Ensuite, des vérifications sont effectuées sur les valeurs saisies. Si certains champs sont vides, des messages d'erreur sont affichés.

Si une image n'a pas été sélectionnée, un message d'erreur est affiché.

Ensuite, un objet SimpleDateFormat est utilisé pour obtenir la date et l'heure actuelles, qui sont stockées dans la variable "currentTime".

Ensuite, l'image sélectionnée est téléchargée dans le stockage Firebase à l'aide de la méthode "putFile" de la référence du stockage. Lorsque le téléchargement est terminé avec succès, l'URL de téléchargement est obtenue et utilisée pour créer un objet "PostModule" contenant les données de l'offre d'emploi. Cet objet est ensuite stocké dans la base de données Firebase en utilisant la méthode "setValue" de la référence de la base de données.

Finalement, lors de la réussite de l'ajout de l'offre d'emploi, un message de confirmation est affiché, la barre de progression est masquée et l'activité se termine.

La méthode "onActivityResult" est utilisée pour gérer le résultat de l'intention de sélection d'image. Lorsqu'une image est sélectionnée avec succès, son chemin d'accès est obtenu à partir des données de l'intention et l'image est convertie en un objet Bitmap. Ensuite, l'image est affichée dans l'ImageView "imageView".

La méthode privée "getFileExtension" est utilisée pour obtenir l'extension du fichier à partir de l'URI de l'image sélectionnée. Elle utilise le ContentResolver pour obtenir le type de contenu de l'URI, puis utilise la classe MimeTypeMap pour obtenir l'extension du fichier correspondant au type de contenu.

En résumé, cette activité permet aux recruteurs de publier des offres d'emploi. Elle permet aux utilisateurs de saisir les détails de l'offre d'emploi tels que le nom, le numéro de téléphone, la localisation, les compétences requises, l'éducation, le salaire, les horaires, la description, etc. Les recruteurs peuvent également sélectionner une image pour accompagner l'offre d'emploi. Une fois les informations saisies et l'image sélectionnée, l'offre d'emploi est ajoutée à la base de données Firebase.

8.5.5 RecruiterMyAccountActivity

Les variables de classe :

- "editTextPhone" est un objet de la classe "TextInputEditText" qui représente le champ de texte pour le numéro de téléphone.
- "edit_btn" et "okay_btn" sont des objets de la classe "FrameLayout" qui représentent les boutons d'édition et de validation.
- "name_box" et "email_box" sont des objets de la classe "TextInputEditText" qui représentent les champs de texte pour le nom et l'e-mail.
- "gender_sp" est un objet de la classe "Spinner" qui représente le sélecteur de genre.
- "gender" est une variable de type String qui stocke la valeur du genre sélectionné.
- "databaseReference" est un objet de la classe "DatabaseReference" qui sera utilisé pour accéder à la base de données Firebase.

Les méthodes :

La méthode "onCreate" : La méthode "onCreate" est une méthode spéciale de l'activité qui est appelée lors de la création de l'activité. La méthode commence par appeler la méthode "super.onCreate" pour exécuter le code de la classe de base. Ensuite, le layout de l'activité est défini en utilisant la méthode "setContentView" et le fichier de layout "activity_my_account.xml". Différentes vues du layout sont initialisées en utilisant la méthode "findViewById". Un adaptateur de tableau est créé pour le spinner "gender_sp" en utilisant les ressources définies dans le fichier "strings.xml". Un écouteur est défini sur le spinner pour récupérer la valeur sélectionnée dans la variable "gender". Les préférences partagées sont récupérées pour obtenir le numéro de téléphone de l'utilisateur. Un écouteur est ajouté à la référence de la base de données Firebase pour récupérer les données de l'utilisateur à partir du nœud "Recruiters". Les

champs de texte sont remplis avec les données récupérées de la base de données Firebase. Un écouteur est défini sur le bouton "edit_btn" pour activer l'édition des champs de texte et masquer le bouton d'édition. Un écouteur est défini sur le bouton "okay_btn" pour valider les modifications des champs de texte, vérifier les valeurs saisies et mettre à jour les données dans la base de données Firebase si les conditions sont remplies.

Cette activité est conçue pour permettre aux recruteurs de gérer leur compte, d'éditer leurs informations personnelles telles que le nom, le numéro de téléphone, l'e-mail et le genre, et de sauvegarder ces modifications dans la base de données Firebase.

8.5.6 RegisterActivityRecruiter

La classe "RegisterActivityRecruiter" permet aux recruteurs de s'inscrire dans l'application de recrutement. Elle contient des champs de saisie de texte, des boutons et un spinner (liste déroulante). Ces vues sont utilisées pour collecter les informations nécessaires à l'inscription du recruteur.

Lors de la création de l'activité (dans la méthode onCreate), les éléments de l'interface utilisateur sont initialisés en utilisant leurs identifiants respectifs définis dans le fichier XML de mise en page. Par exemple, les champs de saisie de texte sont associés à des objets de la classe TextInputEditText, le spinner est associé à un objet de la classe Spinner, et ainsi de suite.

Certains éléments de l'interface utilisateur ont des gestionnaires d'événements attachés. Par exemple, le TextView "loginhere_tv" a un gestionnaire d'événements qui redirige l'utilisateur vers l'activité de connexion lorsqu'il est cliqué.

La classe utilise également un ArrayAdapter pour afficher les options disponibles dans le spinner. Les options sont extraites à partir de la ressource "gender" définie dans le fichier XML des ressources.

Lorsque l'utilisateur clique sur le bouton de continuation ("buttonContinue"), les informations saisies par l'utilisateur, telles que le numéro de téléphone, le nom, le mot de passe, etc., sont extraites des champs de saisie de texte.

Ensuite, plusieurs vérifications sont effectuées pour s'assurer que les informations fournies sont valides. Par exemple, le nom ne doit pas être vide, les mots de passe doivent être identiques et d'au moins 8 caractères, etc. Si une erreur est détectée, un message d'erreur est affiché à l'utilisateur.

Si toutes les vérifications sont réussies, la classe interagit avec la base de données Firebase pour vérifier si un recruteur avec le même numéro de téléphone existe déjà. Si tel est le cas, un message d'erreur est affiché. Sinon, un nouvel objet UserModule est créé avec les informations du recruteur et stocké dans la base de données Firebase sous le nœud "Recruiters".

Enfin, si l'inscription est réussie, l'utilisateur est redirigé vers l'activité de connexion.

8.5.7 ScheduleRequestManagerActivity

La classe "ScheduleRequestManagerActivity" permet aux gestionnaires de gérer les demandes de planning pour les entretiens d'embauche.

L'activité affiche une liste de demandes de planning d'entretiens d'embauche dans un RecyclerView. Elle utilise un adaptateur FirebaseRecyclerView pour récupérer les données à partir de la base de données Firebase et les afficher dans le RecyclerView.

Lors de la création de l'activité (dans la méthode onCreate), les éléments de l'interface utilisateur sont initialisés en utilisant leurs identifiants respectifs définis dans le fichier XML de mise en page. Par exemple, le bouton de retour est associé à un objet de la classe ImageView et le RecyclerView est associé à un objet de la classe RecyclerView.

L'activité utilise également une référence à la base de données Firebase pour récupérer les données relatives aux demandes de planning. Les données sont filtrées en fonction du numéro de téléphone de l'utilisateur actuel, qui est stocké dans les préférences partagées. Cela permet de récupérer uniquement les demandes de planning associées à l'utilisateur connecté.

La méthode "load_orderAdapter" est utilisée pour charger les données des demandes de planning à partir de la base de données Firebase et les afficher dans le RecyclerView. Elle utilise la référence à la base de données et une requête pour récupérer les demandes de planning associées à l'utilisateur actuel.

L'adaptateur FirebaseRecyclerAdapter est configuré pour mapper les données récupérées aux éléments de la vue dans le RecyclerView. Chaque demande de planning est représentée par un objet de la classe ScheduleModule, qui contient des informations telles que l'image du candidat, le nom, la date, l'adresse, etc.

L'adaptateur utilise également la bibliothèque Picasso pour charger et afficher l'image du candidat à l'aide de son URL.

Chaque élément de la liste dans le RecyclerView contient des boutons et des gestionnaires d'événements associés. Par exemple, le bouton d'appel permet à l'utilisateur de composer le numéro de téléphone du candidat. Le bouton de vue permet à l'utilisateur de consulter le CV du candidat. Le bouton de suppression permet à l'utilisateur de supprimer la demande de planning.

Lorsque l'utilisateur clique sur le bouton de suppression, une boîte de dialogue de confirmation s'affiche pour confirmer la suppression de la demande de planning. Si l'utilisateur confirme la suppression, la demande de planning est supprimée de la base de données Firebase.

Lorsque l'utilisateur clique sur le bouton d'acceptation, une boîte de dialogue de confirmation s'affiche pour confirmer l'acceptation de la demande de planning. Si l'utilisateur confirme l'acceptation, la base de données Firebase est mise à jour pour indiquer que la demande de planning a été acceptée.

8.5.8 SigninActivityRecruiter

La classe "SigninActivityRecruiter" représente l'écran de connexion pour les recruteurs. Elle contient des éléments de l'interface utilisateur tels que des champs de saisie pour le numéro de téléphone et le mot de passe, un bouton de connexion, des textes d'invitation, un Spinner pour sélectionner la langue et un lien pour créer un nouveau compte.

Lors de la création de l'activité, les éléments de l'interface utilisateur sont initialisés en utilisant leurs identifiants respectifs définis dans le fichier XML de mise en page. Par exemple, les champs de saisie du numéro de téléphone et du mot de passe sont associés à des objets de la classe TextInputEditText, le bouton de connexion est associé à un objet de la classe FrameLayout, etc.

La classe contient également une référence à la base de données Firebase, qui est utilisée pour vérifier les informations de connexion fournies par les recruteurs.

Un adaptateur ArrayAdapter est utilisé pour remplir le Spinner avec une liste de langues disponibles. Lorsque l'utilisateur sélectionne une langue, la méthode "setlanguage" est appelée pour mettre à jour la configuration de la langue de l'application en fonction de la sélection.

Lorsque l'utilisateur clique sur le bouton de connexion, les informations de connexion saisies sont récupérées. Ensuite, une requête est effectuée sur la base de données Firebase pour vérifier si un recruteur correspondant existe avec les informations fournies. Si c'est le cas, l'utilisateur est authentifié et redirigé vers l'écran d'accueil du recruteur. Sinon, un message d'erreur approprié est affiché.

L'activité contient également des gestionnaires d'événements pour d'autres actions, telles que la création d'un nouveau compte ou la récupération du mot de passe oublié. Lorsque l'utilisateur clique sur le lien de création d'un nouveau compte, il est redirigé vers l'écran d'inscription pour les recruteurs. Lorsque l'utilisateur clique sur le lien de récupération du mot de passe oublié, il est redirigé vers l'écran de récupération du mot de passe pour les recruteurs.

8.5.9 UpdateAdActivity

La classe 'UpdateAdActivity' comporte plusieurs champs pour la saisie d'informations, des boutons et des méthodes pour gérer les interactions avec l'utilisateur.

Les champs de saisie comprennent 'name_box', 'timing_box', 'salary_box', 'skills_box', 'phone_box', 'location_box', 'education_box', 'desc_box', et 'cregis_box'. Ce sont des zones de texte où l'utilisateur peut entrer les informations nécessaires pour mettre à jour une annonce.

Il y a aussi un bouton 'post_btn' qui déclenche l'action de mise à jour de l'annonce. Lorsque ce bouton est cliqué, les valeurs saisies dans les champs de saisie sont récupérées et validées. Si toutes les valeurs sont correctes, l'image sélectionnée est téléchargée dans le stockage Firebase, et les informations de l'annonce sont mises à jour dans la base de données Firebase.

Il y a également un bouton 'addImage_btn' qui permet à l'utilisateur de sélectionner une image à associer à l'annonce. Lorsque ce bouton est cliqué, l'utilisateur peut choisir une image dans la galerie de son appareil.

La classe utilise également des références Firebase pour accéder à la base de données et au stockage Firebase. Elle utilise la classe 'SharedPreferences' pour récupérer les informations de l'utilisateur connecté.

Lorsque l'activité est créée, elle récupère les données de l'annonce à partir de l'intent qui l'a lancée et les affiche dans les champs de saisie correspondants. L'utilisateur peut modifier ces valeurs et cliquer sur le bouton de mise à jour pour effectuer les modifications.

8.6 Activités pour le profil "administrateur"

8.6.1 AdminHomeActivity

La classe 'AdminHomeActivity' correspond à l'écran d'accueil de l'administrateur. Les éléments de l'interface comprennent les boutons 'logout_btn', 'companies_btn', 'ads_btn' et 'schedules_btn'. Lorsque l'utilisateur clique sur le bouton de déconnexion ('logout_btn'), une boîte de dialogue de confirmation s'affiche pour demander à l'utilisateur s'il souhaite vraiment se déconnecter. Si l'utilisateur choisit de se déconnecter, les informations d'identification de l'utilisateur sont supprimées des préférences partagées, et l'activité de connexion initiale ('SignInInitActivity') est démarrée.

Lorsque l'utilisateur clique sur le bouton 'companies_btn', l'activité 'CompaniesActivity' est démarrée, qui affiche la liste des entreprises enregistrées dans le système de recrutement.

Lorsque l'utilisateur clique sur le bouton 'ads_btn', l'activité 'JobsActivity' est démarrée, qui affiche la liste des annonces d'emploi enregistrées dans le système de recrutement.

Lorsque l'utilisateur clique sur le bouton 'schedules_btn', l'activité 'SchedActivity' est démarrée, qui affiche les plannings des entretiens d'embauche dans le système de recrutement.

En résumé, la classe 'AdminHomeActivity' est l'écran d'accueil de l'administrateur dans le système de recrutement. Elle permet à l'administrateur de se déconnecter, d'accéder à la liste des entreprises, aux annonces d'emploi et aux plannings d'entretiens d'embauche.

8.6.2 SigninActivityAdmin

La classe ‘SigninActivityAdmin’ est responsable de la gestion de l’authentification et la connexion des administrateurs . Elle comporte des éléments d’interface utilisateur tels que ‘phone_box’ (un champ de texte pour le numéro de téléphone), ‘pwd_box’ (un champ de texte pour le mot de passe) et ‘login_btn’ (un bouton de connexion).

Dans la méthode ‘onCreate’, on vérifie d’abord si l’utilisateur est déjà connecté en vérifiant la présence d’un numéro de téléphone enregistré dans les préférences partagées (‘SharedPreferences’). Si l’utilisateur est déjà connecté, il est redirigé vers l’activité principale (‘MainActivity’).

Ensuite, la méthode ‘setContentView’ est appelée pour définir la mise en page de l’activité (‘R.layout.activity_signin’).

Les éléments d’interface utilisateur (‘phone_box’, ‘pwd_box’, ‘login_btn’) sont initialisés en utilisant les identifiants correspondants de la mise en page.

La référence à la base de données Firebase (‘databaseReference’) est initialisée en utilisant ‘FirebaseDatabase.getInstance().getReference()’.

Lorsque le bouton de connexion (‘login_btn’) est cliqué, les valeurs des champs de texte (‘phone_box’ et ‘pwd_box’) sont extraites. Ensuite, des validations sont effectuées pour s’assurer que les données saisies sont valides (par exemple, un numéro de téléphone valide et un mot de passe d’au moins 8 caractères).

Ensuite, une écouteur de valeur unique (‘addListenerForSingleValueEvent’) est ajouté à la référence “Admins” dans la base de données Firebase. Cela permet de vérifier si un administrateur avec le numéro de téléphone donné existe dans la base de données.

Dans la méthode ‘onDataChange’, on vérifie si le snapshot contient l’administrateur correspondant au numéro de téléphone donné. Si tel est le cas, on compare le mot de passe saisi avec le mot de passe stocké dans la base de données. Si les mots de passe correspondent, l’utilisateur est connecté avec succès et les informations de l’utilisateur sont enregistrées à l’aide de la méthode ‘LoginUserToApp’. Ensuite, l’utilisateur est redirigé vers l’activité ‘AdminHomeActivity’.

Si le numéro de téléphone ou le mot de passe est incorrect, des messages d’erreur appropriés sont affichés à l’utilisateur.

8.6.3 CompaniesActivity

Cette classe est utilisée pour afficher la liste des entreprises dans la base de données, en utilisant un RecyclerView.

La méthode ‘load_orderAdapter’ est utilisée pour configurer et charger l’adaptateur ‘order_Adapter’ qui affiche la liste des entreprises. Elle initialise une référence à la base de données Firebase (‘databaseReference’) pour récupérer les données des entreprises à partir du noeud “Managers”.

L’adaptateur ‘order_Adapter’ est configuré avec les options ‘orders_options’ qui spécifient la requête pour récupérer les données de l’entreprise à partir de la référence de la base de données.

La méthode ‘onBindViewHolder’ est utilisée pour lier les données de chaque entreprise (‘ManagerModule’) à un élément de vue (‘MyViewHolder_Company’) dans l’adaptateur. Dans cet exemple, le nom de l’entreprise est affiché dans un ‘TextView’ (‘name_tv’) et le numéro d’enregistrement de l’entreprise est affiché dans un autre ‘TextView’ (‘id_tv’).

La méthode ‘onCreateViewHolder’ est utilisée pour créer de nouveaux objets ‘MyViewHolder_Company’ qui représentent les éléments de vue dans l’adaptateur. Elle utilise un inflater de mise en page (‘LayoutInflater’) pour gonfler la mise en page spécifiée (‘company_adapter_layout’) et créer une instance de ‘MyViewHolder_Company’.

Enfin, l'adaptateur 'order_Adapter' est démarré ('startListening') et attaché au 'RecyclerView' ('rv') pour afficher la liste des entreprises.

8.6.4 JobsActivity

Cette Android appelée "JobsActivity" est responsable de l'affichage de toutes les offres d'emploi dans la base de données. L'activité affiche une liste d'annonces d'emploi à l'aide d'un RecyclerView, qui est représenté par la variable "rv" (RecyclerView). La méthode "load_orderAdapter" est appelée pour charger les données des annonces d'emploi dans le RecyclerView. Dans la méthode "load_orderAdapter", la référence à la base de données Firebase est obtenue en utilisant "FirebaseDatabase.getInstance().getReference().child("ADS")". Cela fait référence à la collection "ADS" dans la base de données Firebase, qui stocke les annonces d'emploi. Les options pour l'adaptateur FirebaseRecyclerAdapter sont configurées à l'aide de "FirebaseRecyclerOptions.Builder". Ici, la requête est définie pour récupérer les données de la référence de base de données et le modèle "PostModule" est spécifié. Un nouvel adaptateur FirebaseRecyclerAdapter<MyViewHolder_Home> est créé en utilisant les options et une implémentation de méthodes abstraites. La méthode "onBindViewHolder" est utilisée pour lier les données du modèle "PostModule" aux vues dans le ViewHolder "MyViewHolder_Home". Dans cet exemple, les attributs du modèle, tels que "image", "name", "education", "location", "subject", "price" et "time", sont utilisés pour définir les valeurs des vues correspondantes dans le ViewHolder. Par exemple, l'image est chargée à l'aide de la bibliothèque Picasso et affichée dans "iv_productImage". La méthode "onCreateViewHolder" est utilisée pour créer une instance du ViewHolder "MyViewHolder_Home" en inflatant la mise en page du fichier "home_adapter_layout.xml". L'adaptateur est démarré avec la méthode "startListening" pour commencer à écouter les modifications de données Firebase. Enfin, l'adaptateur est défini sur le RecyclerView à l'aide de la méthode "setAdapter".

8.6.5 SchedActivity

Cette activité est responsable de l'affichage de tous les plannings d'entretiens d'embauche de la base de données. L'activité affiche une liste d'horaires d'événements à l'aide d'un RecyclerView, qui est représenté par la variable "rv" (RecyclerView). Dans la méthode "onCreate", l'activité définit le contenu de la mise en page en utilisant le fichier "activity_companies.xml" et initialise le RecyclerView. La méthode "load_orderAdapter" est utilisée pour charger les données des horaires d'événements dans le RecyclerView. Dans la méthode "load_orderAdapter", la référence à la base de données Firebase est obtenue en utilisant "FirebaseDatabase.getInstance().getReference().child("Schedules")". Cela fait référence à la collection "Schedules" dans la base de données Firebase, qui stocke les horaires d'événements. Les options pour l'adaptateur FirebaseRecyclerAdapter sont configurées à l'aide de "FirebaseRecyclerOptions.Builder". Ici, la requête est définie pour récupérer les données de la référence de base de données et le modèle "ScheduleModule" est spécifié. Un nouvel adaptateur FirebaseRecyclerAdapter<MyViewHolder_Schedule> est créé en utilisant les options et une implémentation de méthodes abstraites. La méthode "onBindViewHolder" est utilisée pour lier les données du modèle "ScheduleModule" aux vues dans le ViewHolder "MyViewHolder_Schedule". Dans cet exemple, les attributs du modèle, tels que "timage" (image de l'événement), "sname" (nom de l'événement), "date" (date de l'événement), "slocation" (lieu de l'événement) et "time" (heure de l'événement), sont utilisés pour définir les valeurs des vues correspondantes dans le ViewHolder. Par exemple, l'image est chargée à l'aide de la bibliothèque Picasso et affichée dans "imageView". La méthode "onCreateViewHolder" est utilisée pour créer une instance du ViewHolder "MyViewHolder_Schedule" en inflatant la mise en page du fichier "schedule_adapter_layout.xml". L'adaptateur est démarré avec la méthode "startListening" pour commencer à écouter les modifications de données Firebase. Enfin, l'adaptateur est défini sur le RecyclerView à l'aide de la méthode "setAdapter".

8.7 Activités pour le profil "gestionnaire"

Ces activités sont définies dans le dossier "Manager"

8.7.1 ManagerHomeActivity

Cette activité, appelée "ManagerHomeActivity", représente l'écran d'accueil de l'application pour les gestionnaires. Lors de son lancement, la mise en page est définie à l'aide de la méthode setContentView(), qui insuffle le fichier de mise en page XML activity_manager_home.xml. La mise en page comprend plusieurs vues, notamment des boutons de navigation, une image et un RecyclerView pour afficher les offres d'emploi. L'activité intègre la bibliothèque ResideMenu pour créer un tiroir de navigation. Elle initialise l'objet ResideMenu et définit les valeurs de fond et d'échelle. Différents objets ResideMenuItem sont ajoutés au tiroir de navigation, tels que "Mon compte", "Conditions générales", "Partager", "Évaluer" et "Déconnexion". Des écouteurs d'événements sont mis en place pour gérer les interactions de l'utilisateur avec les boutons de navigation et les éléments de menu. En cliquant sur le bouton de navigation, le ResideMenu s'ouvre, permettant aux utilisateurs d'accéder à différents écrans. En cliquant sur l'élément de menu "Mon compte" ou sur le bouton "Mon compte", l'écran ManagerMyAccountActivity s'ouvre. En cliquant sur l'élément de menu "Conditions générales", l'écran TermsActivity s'ouvre. ManagerHomeActivity inclut également la fonctionnalité de déconnexion. Lorsque l'utilisateur clique sur l'élément de menu ou le bouton "Déconnexion", une boîte de dialogue AlertDialog s'affiche pour confirmer l'action de déconnexion. Si la déconnexion est confirmée, les préférences partagées de l'utilisateur sont effacées, et il est redirigé vers l'écran SigninInitActivity.

8.7.2 RegisterActivityManager

L'activité permet à un gestionnaire (manager) de s'inscrire en fournissant ses informations personnelles.

L'activité contient des éléments d'interface utilisateur tels que des champs de texte (EditText), des boutons (FrameLayout), un sélecteur de genre (Spinner) et un texte (TextView).

Lorsque l'activité est créée, les éléments d'interface utilisateur sont initialisés en récupérant leurs références à partir de la vue inflatée.

Un adaptateur est utilisé pour configurer le sélecteur de genre (gender_sp) en utilisant les valeurs définies dans le tableau "gender" dans les ressources de l'application.

L'utilisateur peut sélectionner un genre à partir du sélecteur de genre. Lorsqu'un élément est sélectionné, la valeur correspondante est extraite et stockée dans la variable gender.

L'utilisateur peut également cliquer sur le texte "Login Here" (loginhere_tv) pour être redirigé vers l'activité de connexion (SigninActivityManager).

Lorsque l'utilisateur clique sur le bouton "Sign Up" (buttonContinue), les valeurs des différents champs de texte sont extraites, y compris le numéro de téléphone, le nom, le mot de passe, la confirmation du mot de passe, le nom de l'entreprise et le numéro d'enregistrement de l'entreprise.

Des validations sont effectuées pour s'assurer que tous les champs obligatoires sont remplis correctement et que les mots de passe correspondent et respectent les critères de longueur.

Si toutes les validations réussissent, une référence à la base de données Firebase est obtenue, et il est vérifié si un gestionnaire avec le même numéro de téléphone existe déjà dans la base de données.

Si le numéro de téléphone n'est pas déjà utilisé par un autre gestionnaire, les informations du gestionnaire sont enregistrées dans la base de données Firebase, sous la référence "Managers".

Une fois l'enregistrement réussi, l'utilisateur est redirigé vers l'activité de connexion (SigninActivityManager).

8.7.3 SigninActivityManager

L'activité permet à un gestionnaire (manager) de se connecter en utilisant son numéro de téléphone et son mot de passe.

L'activité contient des éléments d'interface utilisateur tels que des champs de texte (TextInputEditText), un bouton (FrameLayout) et des textes (TextView).

Lorsque l'activité est créée, les éléments d'interface utilisateur sont initialisés en récupérant leurs références à partir de la vue inflatée.

Un adaptateur est utilisé pour configurer le sélecteur de langue (language_sp) en utilisant les valeurs définies dans le tableau "lang" dans les ressources de l'application.

L'utilisateur peut sélectionner une langue à partir du sélecteur de langue. Lorsqu'un élément est sélectionné, la valeur correspondante est extraite et stockée dans la variable language.

Si l'utilisateur sélectionne une langue différente de "Select Language", "English" ou "French", aucune action n'est effectuée.

Si l'utilisateur sélectionne "English" comme langue, la méthode setlanguage est appelée pour mettre à jour la configuration de la ressource en utilisant la langue "en" (anglais). Ensuite, l'activité "SigninActivityManager" est redémarrée pour appliquer la nouvelle langue.

Si l'utilisateur sélectionne "French" comme langue, la méthode setlanguage est appelée pour mettre à jour la configuration de la ressource en utilisant la langue "po" (français). Ensuite, l'activité "SigninActivityManager" est redémarrée pour appliquer la nouvelle langue.

L'utilisateur peut également cliquer sur le texte "Sign Up Here" (newAccount_tv) pour être redirigé vers l'activité d'inscription (RegisterActivityManager).

Lorsque l'utilisateur clique sur le bouton "Login" (login_btn), les valeurs des champs de texte du numéro de téléphone et du mot de passe sont extraites.

Des validations sont effectuées pour s'assurer que le numéro de téléphone est valide et que le mot de passe a au moins 8 caractères.

Ensuite, une vérification est effectuée dans la base de données Firebase pour voir si un gestionnaire avec le même numéro de téléphone existe et si le mot de passe correspond.

Si les informations d'identification sont valides, l'utilisateur est connecté et redirigé vers l'activité principale du gestionnaire (ManagerHomeActivity).

Si les informations d'identification sont incorrectes ou si aucun utilisateur n'est enregistré avec le numéro de téléphone donné, un message d'erreur approprié est affiché à l'utilisateur.

La méthode setlanguage est utilisée pour mettre à jour la configuration de la ressource et changer la langue de l'application en fonction de la sélection de l'utilisateur.

8.7.4 ForgotPasswordManager

La classe 'ForgotPasswordManager' permet aux gestionnaires de réinitialiser leur mot de passe en cas d'oubli. Elle comporte des éléments d'interface utilisateur tels que 'phone_box' (un champ de texte pour le numéro de téléphone), 'pwd_box' (un champ de texte pour le nouveau mot de passe), 'cPwd_box' (un champ de texte pour confirmer le nouveau mot de passe) et 'okay_btn' (un bouton pour confirmer la réinitialisation du mot de passe).

Dans la méthode ‘onCreate’, on définit la mise en page de l’activité en utilisant la méthode ‘setContentView’ et on initialise les éléments d’interface utilisateur en utilisant les identifiants correspondants de la mise en page.

La référence à la base de données Firebase (‘databaseReference’) est initialisée en utilisant ‘FirebaseDatabase.getInstance().getReference()’.

Lorsque le bouton de confirmation (‘okay_btn’) est cliqué, les valeurs des champs de texte (‘phone_box’, ‘pwd_box’ et ‘cPwd_box’) sont extraites. Ensuite, des validations sont effectuées pour s’assurer que les données saisies sont valides (par exemple, un numéro de téléphone valide, la correspondance entre les deux mots de passe et la longueur minimale du mot de passe).

Ensuite, une écouteur de valeur unique (‘addListenerForSingleValueEvent’) est ajouté à la référence “Managers” dans la base de données Firebase. Cela permet de vérifier si un gestionnaire avec le numéro de téléphone donné existe dans la base de données.

Dans la méthode ‘onDataChange’, on vérifie si le snapshot contient un gestionnaire correspondant au numéro de téléphone donné. Si tel est le cas, le nouveau mot de passe est enregistré dans la base de données Firebase en utilisant ‘setValue’ pour le nœud correspondant au gestionnaire. Une fois la réinitialisation du mot de passe effectuée avec succès, un message est affiché à l’utilisateur et l’activité est terminée.

Si aucun gestionnaire n’est enregistré avec le numéro de téléphone donné, un message d’erreur approprié est affiché à l’utilisateur.

8.7.5 ManagerMyAccountActivity

L’activité permet à un utilisateur d’accéder à son compte de gestionnaire (manager) et de modifier ses informations personnelles.

Voici une description des éléments clés du code :

L’activité contient des éléments d’interface utilisateur tels que des champs de texte (TextInputEditText), des boutons (FrameLayout) et un sélecteur de genre (Spinner).

Lorsque l’activité est créée, elle récupère le numéro de téléphone de l’utilisateur à partir des préférences partagées (SharedPreferences) et utilise ce numéro pour récupérer les informations du gestionnaire associé à partir de la base de données Firebase en temps réel.

Les informations récupérées, telles que le nom, le numéro de téléphone, l’e-mail et le genre, sont affichées dans les champs de texte correspondants.

L’utilisateur a la possibilité de modifier ses informations personnelles en cliquant sur le bouton "Edit" (edit_btn). Cela active les champs de texte pour permettre l’édition.

Une fois que l’utilisateur a terminé la modification, il peut cliquer sur le bouton "Okay" (okay_btn) pour enregistrer les modifications. Les nouvelles valeurs des champs de texte sont extraites, et des validations sont effectuées pour s’assurer que le nom, l’e-mail et le genre sont correctement renseignés.

Si toutes les validations réussissent, les nouvelles informations du gestionnaire sont mises à jour dans la base de données Firebase en temps réel.

L’activité se termine une fois que les modifications ont été enregistrées avec succès.

9 Modules

La partie "Modules" de l’application constitue une composante essentielle qui définit la structure et le fonctionnement du système. Les modules sont des classes spécifiques conçues pour représenter différentes entités et informations au sein de l’application. Chaque classe de module est responsable

de la gestion et de la manipulation des données liées à son domaine d'activité spécifique. Ces modules servent de blocs de construction pour le bon fonctionnement de l'application, en permettant le stockage, l'accès et la modification des informations pertinentes. Ils facilitent également la communication et l'échange de données entre différentes parties de l'application, contribuant ainsi à une expérience utilisateur fluide et efficace. Dans la suite du rapport, nous présenterons en détail les différents modules de l'application, notamment le ManagerModule, le PostModule, le ScheduleModule et le UserModule, en expliquant leur structure, leurs fonctionnalités et leur rôle dans le système global.

9.1 ManagerModule

Cette classe est utilisée pour représenter les informations d'un gestionnaire (manager).

La classe "ManagerModule" est une classe simple qui utilise des variables publiques pour stocker les informations du gestionnaire telles que le nom (name), le numéro de téléphone (phone), le mot de passe (password), le genre (gender), le nom de l'entreprise (cname), et le numéro d'enregistrement de l'entreprise (cregis).

Pour chaque variable, il existe des méthodes d'accès (getters) et des méthodes de modification (setters) pour récupérer et modifier les valeurs.

La classe dispose d'un constructeur par défaut (public ManagerModule()) qui ne fait rien, ainsi qu'un constructeur paramétré (public ManagerModule(String name, String phone, String password, String gender, String cname, String cregis)) qui permet de définir toutes les valeurs des variables lors de la création d'une instance de la classe.

Les getters et setters sont utilisés pour accéder aux variables de la classe à partir d'autres parties du code.

La classe ne contient aucune logique métier spécifique, elle est simplement utilisée pour encapsuler les informations du gestionnaire et faciliter leur manipulation et leur transmission entre différentes parties du code.

9.2 PostModule

Ce code représente la classe "PostModule" dans une application Android. Cette classe est utilisée pour représenter les informations d'une publication de poste (job posting).

Voici une explication du code :

La classe "PostModule" est une classe simple qui utilise des variables pour stocker les informations d'une publication de poste. Les informations incluent le nom (name), le numéro de téléphone (phone), le sujet (subject), l'emplacement (location), l'éducation (education), l'heure (time), le prix (price), l'identifiant (id), l'image (image), l'heure actuelle (currentTime), le numéro d'enregistrement de l'entreprise (cregis), la description (description), le nombre de candidatures (applies), et le nombre d'entretiens (interviewing).

Pour chaque variable, il existe des méthodes d'accès (getters) et des méthodes de modification (setters) pour récupérer et modifier les valeurs.

La classe dispose d'un constructeur par défaut (public PostModule()) qui ne fait rien, ainsi qu'un constructeur paramétré (public PostModule(String name, String phone, String subject, String location, String education, String time, String price, String id, String image, String currentTime, String cregis, String description, long applies, long interviewing)) qui permet de définir toutes les valeurs des variables lors de la création d'une instance de la classe.

Les getters et setters sont utilisés pour accéder aux variables de la classe à partir d'autres parties du code.

La classe ne contient aucune logique métier spécifique, elle est simplement utilisée pour encapsuler les informations d'une publication de poste et faciliter leur manipulation et leur transmission entre différentes parties du code.

9.3 ScheduleModule

Cette classe est utilisée pour représenter les informations d'un calendrier ou d'un horaire.

Voici une explication du code :

La classe "ScheduleModule" est une classe simple qui utilise des variables pour stocker les informations d'un horaire. Les informations incluent le nom de l'enseignant (tname), le numéro de téléphone de l'enseignant (tphone), l'image de l'enseignant (timage), le nom de l'étudiant (sname), le numéro de téléphone de l'étudiant (sphone), l'emplacement (slocation), la date (date), l'heure (time), l'identifiant (id), l'identifiant de l'annonce (adid), et l'image du CV (cvImage).

Pour chaque variable, il existe des méthodes d'accès (getters) et des méthodes de modification (setters) pour récupérer et modifier les valeurs.

La classe dispose d'un constructeur par défaut (public ScheduleModule()) qui ne fait rien, ainsi qu'un constructeur paramétré (public ScheduleModule(String tname, String tphone, String timage, String sname, String sphone, String slocation, String date, String time, String id, String adid, String cvImage)) qui permet de définir toutes les valeurs des variables lors de la création d'une instance de la classe.

Les getters et setters sont utilisés pour accéder aux variables de la classe à partir d'autres parties du code.

La classe ne contient aucune logique métier spécifique, elle est simplement utilisée pour encapsuler les informations d'un horaire ou d'un rendez-vous et faciliter leur manipulation et leur transmission entre différentes parties du code.

9.4 UserModule

Cette classe est utilisée pour représenter les informations d'un utilisateur.

Voici une explication du code :

La classe "UserModule" est une classe simple qui utilise des variables pour stocker les informations d'un utilisateur. Les informations incluent le nom de l'utilisateur (name), le numéro de téléphone de l'utilisateur (phone), l'adresse e-mail de l'utilisateur (email), le mot de passe de l'utilisateur (password) et le genre de l'utilisateur (gender).

Pour chaque variable, il existe des méthodes d'accès (getters) et des méthodes de modification (setters) pour récupérer et modifier les valeurs.

La classe dispose de deux constructeurs. Le premier constructeur (public UserModule(String name, String email, String gender)) est utilisé pour créer une instance de la classe avec les informations essentielles de l'utilisateur, c'est-à-dire le nom, l'e-mail et le genre. Le deuxième constructeur (public UserModule(String name, String phone, String email, String password, String gender)) est utilisé pour créer une instance de la classe avec toutes les informations de l'utilisateur, y compris le nom, le numéro de téléphone, l'e-mail, le mot de passe et le genre.

Les getters et setters sont utilisés pour accéder aux variables de la classe à partir d'autres parties du code.

La classe ne contient aucune logique métier spécifique, elle est simplement utilisée pour encapsuler les informations d'un utilisateur et faciliter leur manipulation et leur transmission entre différentes parties du code.

10 MyViewHolders

Les MyViewHolders sont des classes utilisées dans le contexte d'un RecyclerView, une composante d'Android, pour afficher une liste d'éléments de manière efficace. Chaque classe MyViewHolder est responsable de maintenir les références aux vues qui représentent les éléments individuels de la liste. Chaque MyViewHolder est une extension de la classe RecyclerView.ViewHolder, ce qui lui permet de bénéficier des fonctionnalités de base fournies par cette classe.

Chaque MyViewHolder est spécifique à un type d'élément dans la liste et contient des attributs publics qui représentent les vues nécessaires pour afficher les informations de cet élément. Ces vues peuvent inclure des TextView, des ImageView, des boutons, des cadres, des mises en page linéaires, etc.

10.1 MyViewHolder_Company

La classe appelée "MyViewHolder_Company" étend la classe "RecyclerView.ViewHolder". Cela signifie que cette classe est utilisée comme un conteneur pour afficher les éléments d'une liste dans un RecyclerView, spécifiquement pour les éléments d'une entreprise.

La classe contient deux attributs publics de type TextView appelés "name_tv" et "id_tv", qui sont utilisés pour afficher le nom et l'ID de l'entreprise respectivement.

Dans le constructeur de la classe, la vue est passée en tant que paramètre. La vue est ensuite transmise à la classe de base (RecyclerView.ViewHolder) à l'aide de l'instruction "super(view)". Ensuite, les attributs "name_tv" et "id_tv" sont initialisés en recherchant les vues correspondantes dans la vue passée, en utilisant les identifiants R.id.name_tv et R.id.id_tv respectivement.

10.2 MyViewHolder_Home

La classe appelée "MyViewHolder_Home" qui étend la classe "RecyclerView.ViewHolder" est utilisée comme conteneur pour afficher les éléments d'une liste dans un RecyclerView, spécifiquement pour les éléments d'une activité ou d'un article.

La classe contient plusieurs attributs publics, notamment des ImageView (iv_productImage, fav_btn), des TextView (name_tv, location_tv, subject_tv, price_tv, time_tv) et un LinearLayout (card). Il y a également deux ImageView supplémentaires (delete_btn, edit_btn).

Dans le constructeur de la classe, une vue est passée en tant que paramètre. La vue est ensuite transmise à la classe de base (RecyclerView.ViewHolder) à l'aide de l'instruction "super(view)". Ensuite, les attributs sont initialisés en recherchant les vues correspondantes dans la vue passée, en utilisant les identifiants R.id correspondants.

10.3 MyViewHolder_Manager

Ce code décrit une classe appelée "MyViewHolder_Manager" qui étend la classe "RecyclerView.ViewHolder". Cette classe est utilisée comme conteneur pour afficher les éléments d'une liste dans un RecyclerView, spécifiquement pour les éléments destinés à être affichés dans une vue de gestionnaire.

La classe contient plusieurs attributs publics, notamment un ImageView (iv_productImage) et plusieurs TextView (name_tv, subject_tv, price_tv, applies_tv, employes_tv). Ces attributs représentent les différentes vues qui seront utilisées pour afficher les informations relatives à un gestionnaire.

Dans le constructeur de la classe, une vue est passée en tant que paramètre. La vue est ensuite transmise à la classe de base (RecyclerView.ViewHolder) à l'aide de l'instruction "super(view)". Ensuite,

les attributs sont initialisés en recherchant les vues correspondantes dans la vue passée, en utilisant les identifiants R.id correspondants.

10.4 MyViewHolder_Schedule

La classe "MyViewHolder_Schedule" est utilisée comme conteneur pour afficher les éléments d'une liste dans un RecyclerView, spécifiquement pour les éléments liés à un emploi du temps ou à une planification.

La classe contient plusieurs attributs publics, notamment un ImageView (imageView) et plusieurs TextView (name_tv, date_tv, time_tv, address_tv, accept_tv). Il y a également des boutons et des vues supplémentaires, tels qu'un FrameLayout (remove_btn, accept_btn, view_btn) et un ImageView (call_btn). De plus, un LinearLayout (card) est présent pour contenir les vues.

Dans le constructeur de la classe, une vue est passée en tant que paramètre. La vue est ensuite transmise à la classe de base (RecyclerView.ViewHolder) à l'aide de l'instruction "super(view)". Ensuite, les attributs sont initialisés en recherchant les vues correspondantes dans la vue passée, en utilisant les identifiants R.id correspondants.