



Département de Mathématiques et Informatique

application pour travaux à domicile

Projet de Fin d'Etudes
Pour Obtenir la Licence en
Sciences **M**athématiques et **I**nformatique

*Conception et réalisation d'une
application pour travaux à domicile*

Réalisé et soutenu par

CHAMAKH Zakaria

ANASRI Anas

OTMANI IDRISI Mokhtar

Membres du Jury

FSAC Président

FSAC Examinateur

FSAC Encadrant

Dédicaces

À nos très chers parents

Avec un énorme plaisir, un cœur ouvert et une immense joie, que nous dédions notre travail à nos très chères, respectueux et magnifiques parents qui nous ont soutenus tout au long de notre vie, dont leurs mérites, leurs sacrifices, leurs qualités humaines nous ont permis de vivre ce jour : les mots nous manquent pour exprimer toute la reconnaissance, la fierté et le profond amour que nous portons pour les sacrifices qu'ils nous ont consenti pour notre réussite, qu'ils trouvent ici le témoignage de notre attachement notre reconnaissance, gratitude et respect, que Dieu leur préservent bonne santé et longue vie. Tous nos sentiments de reconnaissance pour vous.

Nous espérons atteindre le seuil de vos espérances. Que ce travail soit l'expression de notre profonde affection. Je vous remercie pour le soutien moral et l'encouragement que vous nous avez accordés. Nous vous souhaitons tout le bonheur que vous méritez. En leur souhaitant un brillant avenir.

Résumé :

Le présent projet,est la conception et la réalisation d'une application de bricolage qui fonctionne sous le système Android/ios/web dans le cadre de la préparation du projet de fin d'étude présenté en vue de l'obtention du diplôme de licence fondamentale en science Mathématique et informatique à la faculté des sciences Ain Chock pour l'année universitaire 2021/2022

La disponibilité de la croissance des mobiles présente un avantage pour les usagers comme pour les développeurs des applications. Le développement des applications mobiles se positionne actuellement comme une activité majeure du génie logiciel.

Le secret des appareils mobiles repose sur les systèmes d'exploitation qu'ils utilisent, ils fournissent des caractéristiques favorables à un développement réussi des différentes applications sur mobiles.

Pour la développer, nous avons utilisé plusieurs technologies tel que le langage Dart,javascript .par ailleurs ,nous avons utilisé la base de donnée Postgres 12

Mot clés:

Dart,javascript,flutter,postgresql,Uml

Abstract :

This project, is the design and implementation of an application of handymen that runs under the Android/ios/web systems., is carried out as part of the preparation of the project of the end of study presented in order to obtain the diploma of fundamental bachelor in science Mathematics and computer science in the Faculty of Science Ain Chock for the academic year 2020/2021

The increasing availability of mobile is an advantage for users as for developers of applications, the development of mobile applications is currently positioned as a major activity of software engineering.

The secret of mobile devices is based on the operating systems they use, they provide favorable characteristics for successful development of different applications on mobile.

We used several technologies such as Dart and javascript in programming languages and Postgres as a database .

Keywords:

Dart,javascript,flutter,postgresql,Uml

Remerciements

Ce mémoire n'aurait pas pu être confectionné si DIEU le tout puissant ne nous avait pas doté d'une santé physique et morale chaque instant ; c'est pourquoi, nous le remercions à l'infini pour ce don inestimable dont il nous a gratifié.

Nous tenons bien entendre remercier particulièrement, notre cher encadreur, en l'occurrence Mr : NAHHAL Tarik, de l'Université de Science Ain Chock Hassan II qui, par son encadrement ses précieux conseils, sa patience, sa générosité et enfin sa disponibilité ont fait que notre œuvre a été largement facilité : nous ne saurions l'oublier.

Nos remerciements s'adressent également à tous les enseignants qui ont participé à notre formation tout au long de notre cursus.

Nous ne pouvons pas oublier de remercier tous les membres de nos familles pour leurs soutiens et leurs encouragements et particulièrement nos très chers parents.

Merci à tous ceux qui ont contribué, de près ou de loin, à la réalisation de ce travail.

Table des matières

Introduction générale	1
I Applications mobiles	2
1.1 Introduction	2
1.2 Application mobiles	2
1.2.1 Définition	2
1.2.2 Objectifs	2
1.2.3 Supports et outils de la mobilité	3
1.2.3.1 Terminaux mobiles	3
1.2.3.2 Assistants numériques (PDA)	3
1.2.3.3 Smartphone	4
1.2.3.4 Terminaux converges	4
1.2.4 Types d'application mobiles	4
1.2.4.1 Les Applications natives (Natives App)	4
1.2.4.2 Les applications web (Web App)	4
1.2.4.3 Les applications de type hybride ou Hybrid App	5
1.3 Systèmes d'exploitation mobiles	5
1.3.1 Caractéristique des systèmes d'exploitation mobiles	5
1.3.2 Types systèmes d'exploitation mobiles	5
1.4 Présentation d'Android	6
1.4.1 Le choix d'Android	6
1.4.2 Les avantages de la plateforme Android	7
1.4.3 Cycle de vie d'une activité sous Android	7
1.5 Présentation d'IOS	10
1.5.1 Le choix d'IOS	10
1.5.2 Les avantages de la plateforme IOS	10
1.5.3 Cycle de vie d'une activité sous IOS	11
1.6 Conclusion	11
II Etude de l'existant	12
2.1 Introduction	12
2.2 SWOT(Méthode d'analyse)	12

2.3 Définitions	13
2.3.1 ELMAWKEF	13
2.3.2 Travaux/Travaux de Bricolage	13
2.4 Analyse de l'existant	13
2.4.1 Présentation des applications Saweblia et Bricall	14
2.4.2 Exemple d'autre sites internationaux	16
2.5.4 Analyse concurrentielle	19
2.6 Conclusion	19
III Conception	20
3.1 Introduction	20
3.2 Spécification des besoins	20
3.2.1 Les besoins attendus de l'application	20
3.2.1.1 Les besoins fonctionnels	21
3.2.1.2 Les besoins non fonctionnels	21
3.2.2 Le langage UML	21
3.2.3 L'intérêt d'UML	22
3.2.4 Définition d'un modèle	22
3.2.5 Les différents types diagrammes d'UML	22
3.3 Identification des acteurs	23
3.4 Diagramme de contexte du système réaliser	23
3.5 Les Diagrammes des cas d'utilisation	23
3.5.1 Le diagramme global des cas d'utilisations	24
3.5.2 Les différents cas d'utilisation	25
3.5.3 Le cas d'utilisation <>Recherche>>	25
3.5.4 Le cas d'utilisation <>Consultation des catégories >>	26
3.5.5 Le cas d'utilisation <>Préférence>>	27
3.5.6 Le cas d'utilisation <>gestion profil>>	28
3.5.7 Le cas d'utilisation <>consultation des services >>	29
3.5.8 Le cas d'utilisation <>modification de profil bricoleur>>	30
3.5.9 Le cas d'utilisation <>gestion des profils par admin>>	31
3.5.10 Le cas d'utilisation <>Gestion des catégories>>	32
3.6 Diagrammes de séquence	33
3.6.1 Définition d'un diagramme de séquence	33
3.6.2 Diagramme de séquence du cas d'utilisation Authentification	34
3.6.3 Diagramme de séquence du cas d'utilisation consultation des catégories	35
3.6.4 Diagramme de séquence du cas d'utilisation consultation des services.	36

3.6.5	Diagramme de séquence du cas d'utilisation recherche service.	37
3.6.6	Diagramme de séquence du cas d'utilisation Gestion de profil	38
3.6.7	Diagramme de séquence du cas d'utilisation Choisir thème/langue	39
3.7	Diagramme de classe	40
3.8	Modèle conceptuel de données (MCD)	41
3.9	Modèle logique des données (MLD)	42
3.10	Entity Relationship Diagram ERD	43
3.8	Conclusion	43
IV	Gestion du projet	44
4.1	Planning initial du projet	44
4.1.1	Scrum and Agile	45
4.1.1.1	les étiquettes	46
4.1.1.2	user story	47
4.1.1.3	backlog	47
4.1.1.4	sprint	48
4.2	Conduite du projet	57
4.2.1	Méthodologies de développement	57
4.3	Cycle de vie	65
4.3.1	Planification	65
4.3.2	Création	66
4.3.3	Test	67
4.3.4	Versionnage	68
4.3.5	Déploiement	69
4.3.6	Exploitation	71
4.3.7	Supervision	71
V	Réalisation	72
5.1	Introduction	72
5.2	Outils de développement	72
5.2.1	Android Studio	72
5.2.2	IntelliJ IDEA	73
5.2.3	Visual Studio Code	74
5.2.4	Postman	74
5.2.5	DigitalOcean	75
5.2.6	NameCheap	75
5.2.7	Github	76

5.2.8	Figma	77
5.2.9	Trello	77
5.2.10	Javascript	77
5.2.11	NodeJs	78
5.2.12	Plantuml	78
5.2.13	Postgresql	79
5.2.14	Agile/Scrum	79
5.2.15	Strapi	80
5.2.16	Html	81
5.2.17	Css	81
5.2.18	Flutter	81
5.2.19	Dart	81
5.2.20	Getx	82
5.2.21	Firebase	83
5.2.22	Json	83
5.2.23	Yaml	84
5.3	Présentation des interfaces de l'application	84
5.3.1	Interface d'inscription	85
5.3.2	Interface d'accueil	86
5.3.3	Interface pour chercher le profil du professionnel	87
5.3.4	Interface profil	88
5.3.5	Option dark / light	89
5.3.6	Option de la langue	89
5.3.7	Ajout des services	91
5.3.8	Option du partage	92
5.3.9	Option d'évaluation	93
4.3.10	Option nous contacter	94
5.3.11	Option se déconnecter	95
5.4	Conclusion	96

TABLE DES FIGURES

I Applications mobiles	2
Figure 1.1 Cycle de vie d'une activité sous Android.	8
Figure 1.2 Cycle de vie d'une activité sous IOS.	11
II Etude de l'existant	12
Figure 2.1 Logo de ELMAWKEF .	13
Figure 2.2 La page d'accueil de l'application Bricall.	14
Figure 2.3 SWOT de l'application Bricall.	14
Figure 2.4 La page d'accueil de l'application Saweblia.	15
Figure 2.5 SWOT de l'application Saweblia.	15
Figure 2.6 La page d'accueil du site HomeAdvisor.	16
Figure 2.7 La page d'accueil du site Handy.	17
Figure 2.8 La page d'accueil du site Thumbtack.	18
III Conception	20
Figure 3.1 les diagrammes UML.	22
Figure 3.2 Diagramme de contexte .	23
Figure 3.3 Diagramme global des cas d'utilisations associés aux acteurs.	24
Figure 3.4 Diagramme de cas d'utilisation <>Recherche>>	25
Figure 3.5 Diagramme de cas d'utilisation <>Consultation des catégories >>	26
Figure 3.6 Diagramme de cas d'utilisation <>Préférence>>	27
Figure 3.7 Diagramme de cas d'utilisation <>gestion profil>>	28
Figure 3.8 Diagramme de cas d'utilisation <>consultation des services >>	29
Figure 3.9 Diagramme de cas d'utilisation <>modification de profil bricoleur>>	30
Figure 3.10 Diagramme de cas d'utilisation <>gestion des profils par admin>>	31
Figure 3.11 Diagramme de cas d'utilisation <>Gestion des catégories>>	32
Figure 3.12 Diagramme de séquence du cas d'utilisation Authentification .	34
Figure 3.13 Diagramme de séquence du cas d'utilisation consultation catégories .	35
Figure 3.14 Diagramme de séquence du cas d'utilisation Consultation des services .	36
Figure 3.15 Diagramme de séquence du cas d'utilisation pour la recherche d'un service.	37
Figure 3.16 Diagramme de séquence du cas d'utilisation gestion du profil.	38

Figure 3.17	Diagramme de séquence du cas d'utilisation choisir thème /langue.	39
Figure 3.18	Diagramme de classe.	40
Figure 3.19	MCD.	41
Figure 3.20	MLDR.	42
Figure 3.21	ERD.	43
IV Gestion du projet		44
Figure 4.1	scrum framework.	45
Figure 4.2	étiquettes.	46
Figure 4.3	structure des dossiers.	47
Figure 4.4	structure des dossiers.	47
Figure 4.5	sprint.	48
Figure 4.6	facturation de server.	49
Figure 4.7	l'interface de configuration.	50
Figure 4.8	variable d'environnement de server.	51
Figure 4.9	base donne type de utilisateur.	51
Figure 4.10	base donne détails de connexion.	52
Figure 4.11	Structure de Dossier.	53
Figure 4.12	Deploye line.	54
Figure 4.13	Monitoring.	55
Figure 4.14	Server component.	56
Figure 4.15	Log base donnée.	56
Figure 4.16	DevOps.	57
Figure 4.17	json catégorie.	66
Figure 4.18	Github branche..	67
Figure 4.19	Postman.	68
Figure 4.20	Play Stor.	70
V Réalisation		72

Figure 5.1	Android Studio.	72
Figure 5.2	IntelliJ IDEA.	73
Figure 5.3	Visual Studio Code.	74
Figure 5.4	Postman.	74
Figure 5.5	DigitalOcean.	75
Figure 5.6	NameCheap.	75
Figure 5.7	Github.	76
Figure 5.8	Figma.	76
Figure 5.9	Trello.	77
Figure 5.10	Javascript.	77

Figure 5.11	NodeJs.	78
Figure 5.12	Plantuml.	78
Figure 5.13	Postgresql.	79
Figure 5.14	Agile/Scrum.	79
Figure 5.15	Strapi.	80
Figure 5.16	Html.	80
Figure 5.17	Css.	81
Figure 5.18	Flutter.	81
Figure 5.19	Dart.	81
Figure 5.20	Getx.	82
Figure 5.21	Firebase.	82
Figure 5.22	Json.	83
Figure 5.23	Yaml .	83
Figure 5.24	Interface d'inscription.	85
Figure 5.25	Page d'accueil .	86
Figure 5.26	chercher le profil .	87
Figure 5.27	Interface profil .	88
Figure 4.28	Option dark / light .	89
Figure 5.29	Option de la langue .	90
Figure 5.30	Ajout des services .	91
Figure 5.31	Option du partage .	92
Figure 5.32	option d'évaluation .	93
Figure 5.33	Option nous contacter .	94
Figure 5.34	Option se déconnecter.	95

Liste des tableaux

Diagramme de cas d'utilisation <<Recherche>>	25
Diagramme de cas d'utilisation <<Consultation des catégories >>	26
Diagramme de cas d'utilisation <<Préférence>>	27
Diagramme de cas d'utilisation <<gestion profil>>	28
Diagramme de cas d'utilisation <<consultation des services >>	29
Diagramme de cas d'utilisation <<modification de profil bricoleur>>	30
Diagramme de cas d'utilisation <<gestion des profils par admin>>	31
Diagramme de cas d'utilisation <<Gestion des catégories>>	32

Liste des abréviations

- **API** Application Programming Interface.
- **CSS** Cascading Style Sheets.
- **GPS** Global Positioning System.
- **HTML** HyperText Markup Language.
- **IOS** Iphone Operating System.
- **JSON** Java Script Object Notation.
- **OS** Operating System.
- **PDA** Personal Digital Assistant.
- **PHP** Hypertext Preprocessor.
- **SDK** Software Development Kit.
- **SGBD** Système de Gestion de Base de Données.
- **SQL** Structured Query Language.
- **SWOT** Strengths, Weaknesses, Opportunities and Threats analysis.
- **UML** Unified Modeling Language.
- **XML** EXtended Markup Lang.

Introduction générale

Aujourd’hui, le monde connaît une avancée considérable dans l’utilisation des applications téléphoniques portables (mobiles), ces dernières qui sont capables de satisfaire les besoins actuels des utilisateurs avec de nombreuses fonctionnalités et en offrant plusieurs services.

De nouveaux usages sont apparus tels que les jeux, la lecture audio, etc. sur un plan plus pratique, le téléphone mobile dispose des variétés d’utilisations et l’appareil est devenu plus utile que jamais. L’essor du mobile a connu une évolution considérable. Les réseaux de télécommunication mobiles sont également en expansion : le nombre d’utilisateurs de mobile est en constante progression et la couverture territoriale est largement répandue.

Notre projet s’inscrit dans un cadre général du développement et de la réalisation d’une application de bricolage qui fonctionne sous Android / Ios / Web qui permet à plusieurs utilisateurs de trouver un bricoleur suite à leur besoin.

Notre projet est structuré en quatre chapitres, le premier chapitre porte sur les applications mobiles et leurs objectifs, ensuite, on va citer les différents supports et outils de la mobilité, les types d’application mobiles et les différents systèmes d’exploitation et une présentation détaillée du système Android.

Le deuxième chapitre sera dédié à l’étude de l’existant, on parlera de quelque site web qui offre le même service.

Le troisième chapitre sera consacré à la conception qui est une étude préalable de notre projet, elle est une étape primordiale et un passage obligatoire, car il s’agit de décrire les besoins de notre système, on modélise avec le langage UML.

La réalisation de notre application fera l’objet du chapitre quatre, ou nous allons présenter les outils ainsi que les langages de programmations utilisés, on va expliquer le fonctionnement de notre application en utilisant quelques captures d’écran.

Nous conclurons notre travail par exposer l’ensemble des connaissances acquises au cours de la réalisation de notre projet, et nous exposerons quelques perspectives.

1 Applications mobiles

1.1 Introduction

Aujourd’hui, les applications mobiles ont pris une place importante dans notre vie quotidienne, et prennent de plus en plus d’espace dans l’utilisation de nos terminaux mobiles, elles sont conçues pour des plateformes mobiles et utilisées pour des services de l’information, médias sociaux, jeux etc.

Dans ce chapitre nous abordons la notion de l’application mobile et ses objectifs, ses types, les différents systèmes d’exploitation mobiles, en nous allons faire une présentation détaillée sur le système Android.

1.2 Application mobiles

Les applications mobiles sont apparues dans les années 1990, elles sont liées aux développements d’Internet et des télécommunications, des réseaux sans fil et des technologies agents, et l’apparition et la démocratisation des terminaux mobiles : Smartphones, tablettes ...etc.

1.2.1 Définition

Une application mobile est un logiciel applicatif développé pour un appareil électronique mobile, tel qu’un assistant personnel, un téléphone portable, un smartphone, un baladeur numérique, une tablette tactile, ou encore certains ordinateurs fonctionnant avec le système d’exploitation Windows Phone.

1.2.2 Objectifs

Elles visaient d’abord la productivité et faciliter la récupération d’informations telles que courrier électronique, calendrier électronique, contacts, marché boursier et informations météorologiques. La demande du public et la disponibilité d’outils de développement ont conduit une expansion rapide dans d’autres domaines, comme : les jeux mobiles, les

automatismes industriels, le GPS et les services basés sur la localisation, les opérations bancaires, les suivis des commandes, l'achat de billets, des applications médicales mobiles, la réalité virtuelle.

1.2.3 Supports et outils de la mobilité

Les supports du multimédia mobile peuvent être regroupés en 4 grandes catégories :

1.2.3.1 Terminaux mobiles

De par sa diversité, l'univers de terminaux est sans doute le révélateur de la dynamique du monde mobile. Jusqu'à une époque récente, on connaissait comme seuls objets communicants les téléphones et les ordinateurs. Mais de plus en plus, la miniaturisation des puces, des mémoires, des antennes et des batteries permet aujourd'hui d'ajouter des capacités d'acquisition, de traitement, de stockage, de communication et même d'action un nombre croissant d'objets rendus intelligents. Nous nous limitons ici à un inventaire sommaire des outils de mobilité les plus populaires qui permettent nombre d'utilisateurs d'avoir une connexion de type multimédia mobile.

-PC portable et Tablet PC (e-books) : Le Pocket PC de Microsoft est l'exemple type de l'ordinateur de poche. Doté d'un système d'exploitation, Un Pocket PC rappelle un PC avec son menu Démarrer et la suite Pocket Office : Pocket Internet Explorer, Outlook, Pocket Word et Pocket Excel. Le Tablet PC, lui, est un ultra-portable avec toutefois des caractéristiques spéciales : son écran est tactile, il reconnaît votre écriture grâce à un stylet, et la transforme en texte. L'unique charnière centrale de son écran permet à ce dernier de pivoter et se rabattre sur le clavier.

1.2.3.2 Assistants numériques (PDA)

Le PDA (Personal Digital Assistant) ou assistant numérique personnel est un ordinateur de poche faisant office d'assistant personnel. Un assistant numérique personnel ou PDA - est à l'origine un agenda électronique destiné à la prise de rendez-vous, la planification des tâches et au transport de données personnelles. Il s'est, depuis, beaucoup enrichi en fonctionnalités communicantes (push mail, carte 3G, connexions Bluetooth, etc.).

1.2.3.3 Smartphone

Les Smartphones sont des téléphones dits intelligents qui assurent en priorité les fonctions de communication et de navigation sur internet. Ces terminaux proposés par les constructeurs informatiques et les équipementiers télécoms associent les fonctions des assistants personnels et celles des téléphones mobiles. Un utilisateur de Smartphone consomme deux fois plus de voix, et dix fois plus de data , c'est- -dire de données (mail, accès Internet).

1.2.3.4 Terminaux converges

Il y a une convergence des PDA, Pocket PC, smartphone et autres produits, car tous les professionnels sont demandeurs d'un produit simple rassemblant toutes les applications mobiles avec Bluetooth, Wi-Fi, etc. Ces terminaux offrent des services voix ou données et sont capables de synchroniser des informations personnelles et/ou des courriels avec des serveurs, des postes de travail ou des ordinateurs portables. Ils sont dotés d'un système d'exploitation évolué comme Palm OS, Windows Mobile 5.0, ainsi que la plateforme Symbian.

1.2.4 Types d'application mobiles

Actuellement, il existe 3 principaux types d'applications mobiles :

1.2.4.1 Les Applications natives (Natives App)

Celles-ci correspondent à des logiciels créés uniquement pour une plateforme mobile. Le développement de ces logiciels se fait au travers du SDK ou software développement kit de la plateforme mobile choisie. Le nom de ces applications vient du fait qu'elles sont développées avec l'utilisation de langages natifs comme par exemple le langage JAVA ou le langage Objective-C. Les natives App sont téléchargées à partir d'une plateforme de téléchargement qui est souvent une App Store.

1.2.4.2 Les applications web (Web App)

Ces applications correspondent à des sites web qui sont conçus spécialement pour un affichage sur mobile optimisé. Pour accéder à ces sites web, on utilise le navigateur internet disponible sur le mobile. Ces applications mobiles sont développées principalement à partir de technologies web comme HTML5 ou encore CSS3. Grâce au support HTML5 il est possible d'accéder à environ 80% des fonctions présentes sur le mobile. Par exemple, cela permet

d'accéder à la géolocalisation, à l'accéléromètre, gérer la fonction multitouche ou encore permettre la synchronisation offline lorsque le mobile perd et retrouve sa connexion.

1.2.4.3 Les applications de type hybride ou Hybrid App

Ces applications sont considérées comme un mix à la fois entre les applications natives et les applications web. En effet, elles sont compatibles avec toutes les plateformes mobiles. Ces applications sont principalement développées à l'aide d'HTML5 aujourd'hui qui est très performant mais utilisent aussi d'autres langages web comme CSS et JavaScript.

1.3 Systèmes d'exploitation mobiles

Un système d'exploitation mobile est un système d'exploitation conçu pour fonctionner sur un appareil mobile. Ce type de système d'exploitation se concentre entre autres sur la gestion de la connectivité sans fil et celle des différents types d'interface.

1.3.1 Caractéristique des systèmes d'exploitation mobiles

Un système d'exploitation mobile regroupe un ensemble des fonctionnalités dont :

- La gestion de la mémoire.
- La gestion des microprocesseurs et l'ordonnancement.
- La gestion de système de fichiers.
- La gestion des Entrées/Sorties.
- La gestion de sécurité.
- La gestion de fonctionnalités multimédia.

1.3.2 Types systèmes d'exploitation mobiles

Les principaux systèmes d'exploitation mobile sont : Android, Bada, BlackBerry OS, iOS, OpenMoko, Palm OS, HP webOS, Symbian OS, Windows CE, Windows Mobile, Windows Phone 7.

-ios : est un système d'exploitation mobile créé et développé par la société américaine

Apple exclusivement pour ses produits. Il alimente de nombreux appareils tels que l'iPhone, l'iPod touch et sur les iPad

-**Android** : est un système d'exploitation open source utilisant le noyau Linux, pour smartphones, PDA et terminaux mobiles conçu par Android, une startup rachetée par Google, et annoncé officiellement le 5 novembre 2007. D'autres types d'appareils possédant ce système d'exploitation existent, par exemple des téléviseurs et des tablettes.

-**Bada** : est le système d'exploitation pour smartphone haut de gamme et milieu de gamme de Samsung basé sur le système d'exploitation propriétaire SHP OS. Tous les téléphones sous Bada ont un nom commençant par Samsung Wave.

-**BlackBerry OS** : est un système d'exploitation qui fonctionne sur le smartphone BlackBerry. Il permet aux développeurs de mettre en place des applications en utilisant les APIs BlackBerry, mais toute application doit être signé numériquement par le compte RIM du développeur.

-**Symbian OS** : est un système d'exploitation le plus utilisé pour des Smartphones et PDA, qui a été conçu par Symbian Ltd. Symbian OS fournit les fonctionnalités essentielles du système d'exploitation, notamment le cœur du système, ainsi que les API communes et une interface utilisateur de référence. Il a été adopté par différents fabricants de téléphones portables de 2G et 3G (Nokia, Sony Ericsson, Motorola, Samsung, etc.). Les principales interfaces utilisateur sont S60 série (pour clavier numérique), UIQ (pour Écran tactile), S80 série (pour clavier alphanumérique).

1.4 Présentation d'Android

Il existe plusieurs types de systèmes d'exploitation pour le téléphone mobile, chaque système a ses avantages et ses limites. Il est donc très difficile de choisir la plateforme répondant bien à l'objectif d'une société ou d'une personne.

Dans cette partie nous allons présenter la raison du choix d'Android, nous aurons aussi à présenter ce système et les outils pour créer une application dans la pratique.

1.4.1 Le choix d'Android

Android est un système d'exploitation open Source pour smartphones, PDA et autres terminaux mobiles, conçu par Android, une start-up rachetée par Google en juillet 2005. Il est

donc gratuit et librement modifiable, ce qui explique d'ailleurs le nombre de mobiles qui l'utilise et ce indépendamment de leur fabricant. Cet aspect lui permet également d'être présent sur d'autres types d'appareils possédant ce système d'exploitation tels que les téléviseurs et les tablettes. Ceci peut d'ailleurs constituer une opportunité de se détacher du seul cadre du téléphone mobile.

Le noyau Linux lui fournit une grande mémoire, la gestion de processus, le modèle de sécurité, le soutien des bibliothèques partagées...etc. Le SDK de l'Android offre complètement les APIs, avec un accès facile pour développer l'application.

Une API, ou interface de programmation en français, est un ensemble de règles à suivre pour pouvoir dialoguer avec d'autres applications. Dans le cas de Google API, il permet en particulier de communiquer avec Google Maps.

Android est un système d'exploitation puissant et moderne, qui se caractérise par la simplicité et la flexibilité; cela signifie que le système est développé avec un langage Java, et il s'adapte à beaucoup de structures différentes.

1.4.2 Les avantages de la plateforme Android

- + Pas de Licence à obtenir, pas de dépense pour la distribution et le développement.
- + Développer des applications location-based en utilisant le GPS.
- + Utiliser des cartes géographiques avec Google Maps.
- + Recevoir et émettre des SMS, envoyer et recevoir des données sur le réseau mobile.
 - + Enregistrer et lire image, son et vidéo.
 - + Des outils de stockage de données partagés (SQLite en version Sandbox).
 - + Une accélération matérielle pour la 2D et la 3D.
- + Des services et des applications qui peuvent tourner en tâche de fond : qui peuvent réagir au cours d'une action, votre position dans la ville, l'heure qu'il est, suivant l'identité de l'appelant.
- + Une plateforme de développement qui favorise la réutilisation de composants logiciels et le remplacement des applications fournies.

1.4.3 Cycle de vie d'une activité sous Android

Voici un diagramme qui explique le cycle de vie d'une activité sous Android

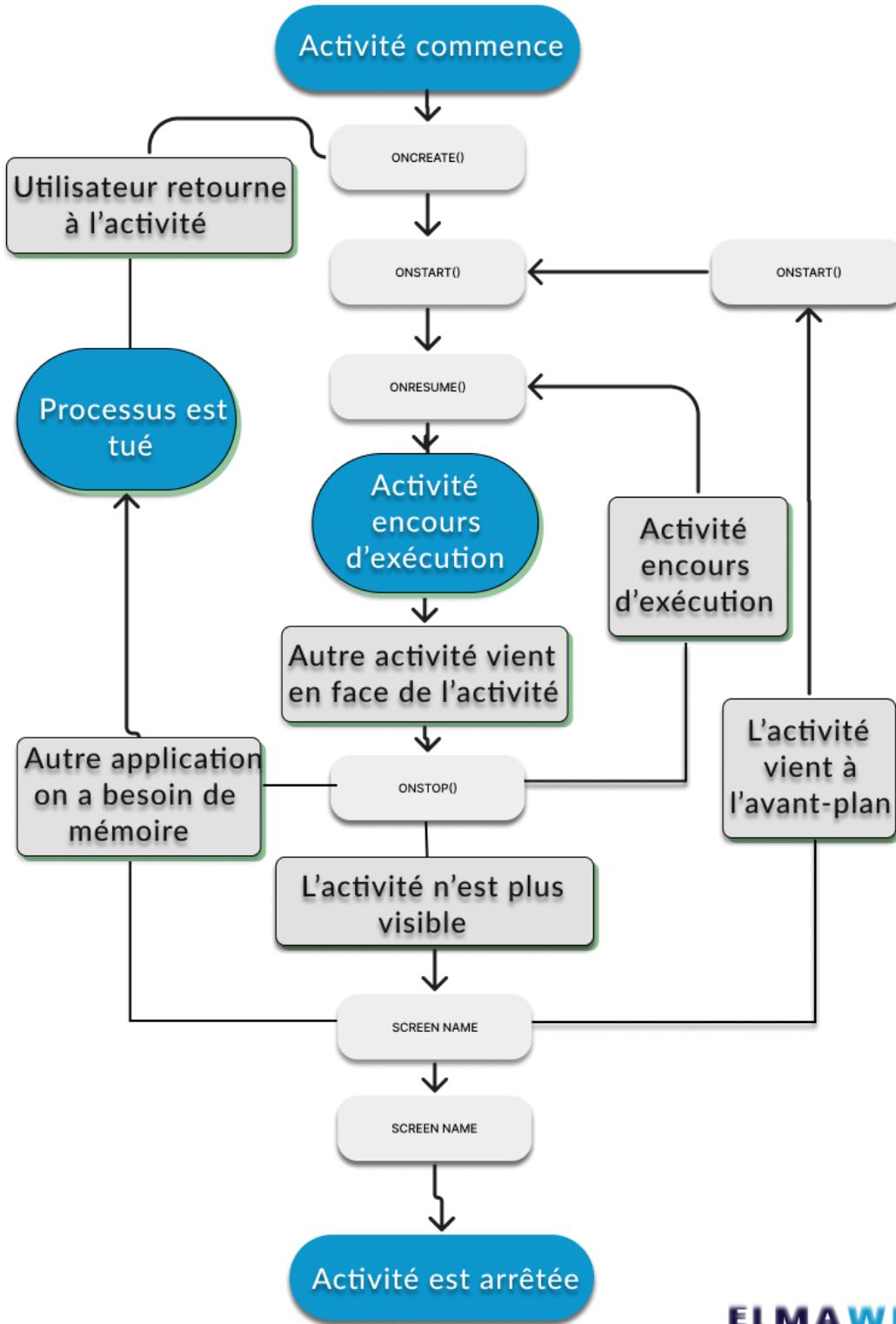


Figure 1.1 Cycle de vie d'une activité sous Android.

OnCreate

Cette méthode est appelée la création de l'activité (Activity). Elle sert à initialiser votre activité ainsi que toutes les données nécessaires à cette dernière. Quand la méthode OnCreate est appelée, on lui passe un Bundle en argument. Ce Bundle contient l'état de sauvegarde enregistré lors de la dernière exécution de votre activité.

OnStart

Cette méthode est appelée dans le cas où votre application est en arrière-plan et qu'elle repasse en avant-plan. Si votre activité ne peut pas aller en avant-plan quelle que soit la raison, l'activité sera transférée OnStop.

OnResume

Cette méthode est appelée après OnStart au moment où l'application repasse en avant-plan ou en arrière-plan à cause d'une autre application.

OnPause

Appelée juste avant qu'une autre activité passe en OnResume. À ce stade, notre activité n'a plus accès à l'écran, vous devez arrêter de faire toute action en rapport avec l'interaction utilisateur. Vous pouvez par contre continuer à exécuter des algorithmes nécessaires mais qui ne consomment pas trop de CPU.

OnStop

Appelée quand l'activité n'est plus visible quelle que soit la raison.

OnDestroy

Appelée quand l'application est totalement fermé (Processus terminé)

1.5 Présentation d'IOS

Il s'agit du deuxième système d'exploitation mobile le plus installé au monde, après Android. Il constitue la base de trois autres systèmes d'exploitation : iPadOS , tvOS et watchOS. Il s'agit d'un logiciel propriétaire, bien que certaines parties soient en open source sous la licence Apple Public Source License et d'autres licences.

1.5.1 Le choix d'IOS

Lorsque l'on parle de iOS, on fait référence au système d'exploitation (Operating System) propre à la marque Apple et qui concerne donc toutes les gammes d'appareils de type iPhone, iPod, l'iPad et les périphériques compatibles. Il est utilisé par les générations successives d'iPhone depuis la sortie du premier modèle en 2007 puis par l'iPad depuis 2010. Les puristes seront curieux de savoir si les applications développées pour l'iOS sont codées en Swift ou bien en Objective-C.

1.5.2 Les avantages de la plateforme IOS

- +Tous les avantages qu'Android peut offrir et plus.
- +La recherche intégrée, qui permet d'effectuer des recherches à la fois dans des fichiers, des objets multimédias, des applications et des e-mails.
- +La reconnaissance des gestes,
- +Le navigateur mobile Safari.
- +Un accès direct au catalogue Apple Store : applications, musique, podcasts, émissions de télévision et films
- +La compatibilité avec iCloud, le service Cloud d'Apple.

1.5.3 Cycle de vie d'une activité sous IOS

Voici un diagramme qui explique le cycle de vie d'une activité sous IOS :

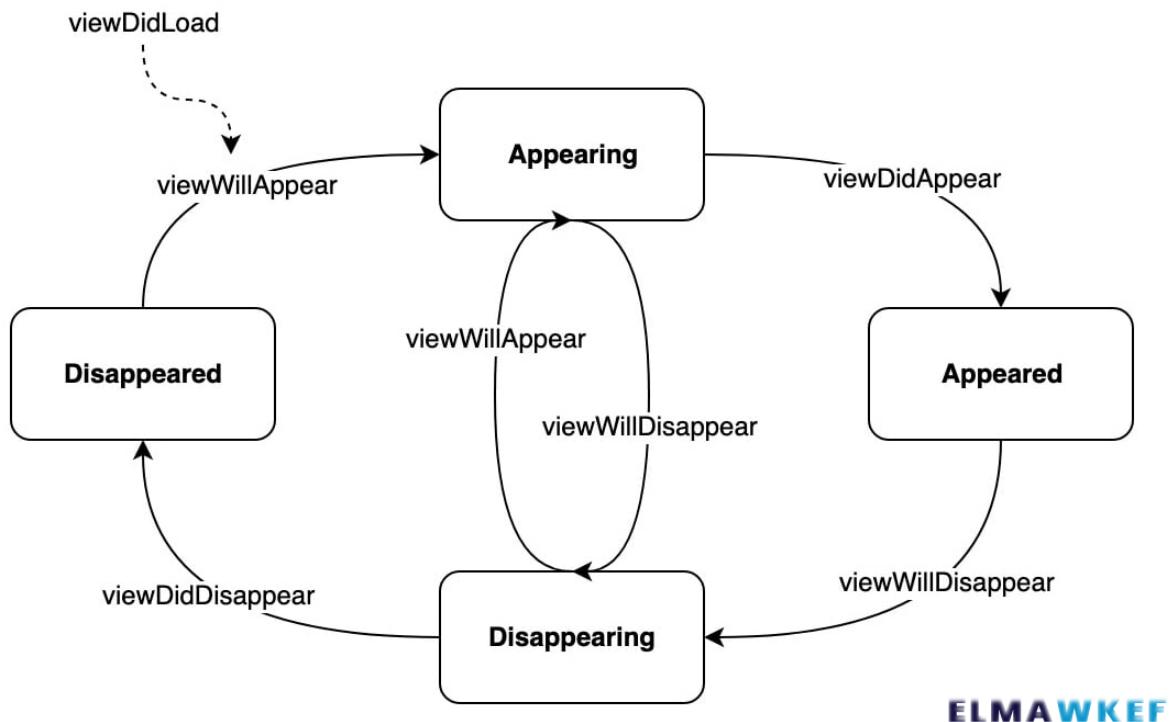


Figure 1.2 Cycle de vie d'une activité sous IOS.

1.6 Conclusion

Dans ce chapitre, nous avons donné un petit aperçu sur les applications mobiles, ainsi que ses différents objectifs. Et puis on a présenté les différents supports et outils de la mobilité et on a énuméré les différents types d'application mobile, puis on a défini les systèmes d'exploitation mobiles ainsi leurs caractéristiques et leurs types. A la fin on a fait une présentation d'Android et IOS dont sa description, ses avantages, ses fonctionnalités et son architecture.

2 Etude de l'existant

2.1 Introduction

Il existe plusieurs moyens pour trouver un type de services pour vous fournir une solution à vos travaux ou à vos pannes à domicile, dont chacune d'elle présente ses propres avantages et inconvénients par rapport aux autres.

Nous présenterons dans ce chapitre, l'étude de l'existant qui doit être élaboré avant d'entamer la spécification des besoins et la conception de notre application, et pour cela nous allons donner une présentation générale de ses méthodes et des services qui propose des solutions similaires

2.2 SWOT(Méthode d'analyse)

L'**analyse SWOT**, matrice **SWOT** ou **synthèse SWOT** est un outil de stratégie d'entreprise permettant de déterminer les options offertes dans un domaine d'activité stratégique. Il vise à préciser les objectifs de l'entreprise ou du projet et à identifier les facteurs internes et externes favorables et défavorables à la réalisation de ces objectifs.

L'analyse SWOT a été décrite comme l'outil éprouvé de l'analyse stratégique 1. Les forces et les faiblesses sont souvent d'ordre interne, tandis que les opportunités et les menaces se concentrent généralement sur l'environnement extérieur. Le nom est un acronyme pour les quatre paramètres examinés par la technique :

- **Strengths** (Forces) : caractéristiques de l'entreprise ou du projet qui lui donnent un avantage sur les autres.
- **Weaknesses** (Faiblesses) : caractéristiques de l'entreprise qui désavantagent l'entreprise ou le projet par rapport aux autres.
- **Opportunities** (Opportunités) : éléments de l'environnement que l'entreprise ou le projet pourrait exploiter à son avantage.
- **Threats** (Menaces) : éléments de l'environnement qui pourraient causer des problèmes à l'entreprise ou au projet.

2.3 Définitions

2.3.1 ELMAWKEF

C'est un terme marocain et qui fait référence à un emplacement où on peut trouver des bricoleurs dans différents domaines qui cherchent du travail.



Figure 2.1 Logo de ELMAWKEF .

2.3.2 Travaux/Travaux de Bricolage

Le bricolage est une activité manuelle, souvent amateur, visant à réparer, entretenir, améliorer ou fabriquer de petits objets. On dit d'une personne habile de ses mains qu'il est un « bon bricoleur ». À l'inverse, l'expression « bricoleur du dimanche » est plus péjorative.

2.4 Analyse de l'existant

Dans cette section, nous présentons les sites BRICALL et SAWEBLIA. Ensuite nous procéderons à une analyse concurrentielle.

2.4.1 Présentation des applications Saweblia et Bricall

● BRICALL

Bricall est une plateforme de mise en relation entre particuliers et artisans de différentes spécialités.

Créé par DIGISERV, société de développement de services digitaux, Bricall a pour vocation de faciliter la vie des marocains dans la réalisation de leurs petits travaux.



Figure 2.2 La page d'accueil de l'application Bricall.



Figure 2.3 SWOT de l'application Bricall.

● Saweblia

Saweblia est une équipe opérant dans 3 villes à travers le Maroc. Ils s'efforcent de créer les meilleurs services pour aider leurs clients à être à l'aise au quotidien .



Figure 2.4 La page d'accueil de l'application Saweblia.



Figure 2.5 SWOT de l'application Saweblia.

2.4.2 Exemple d'autre sites internationaux

● HomeAdvisor

HomeAdvisor est une marche digitale connue sous le nom de ServiceMagic A pour but de connecter le propriétaire de maison avec des professionnelle pour leur problème

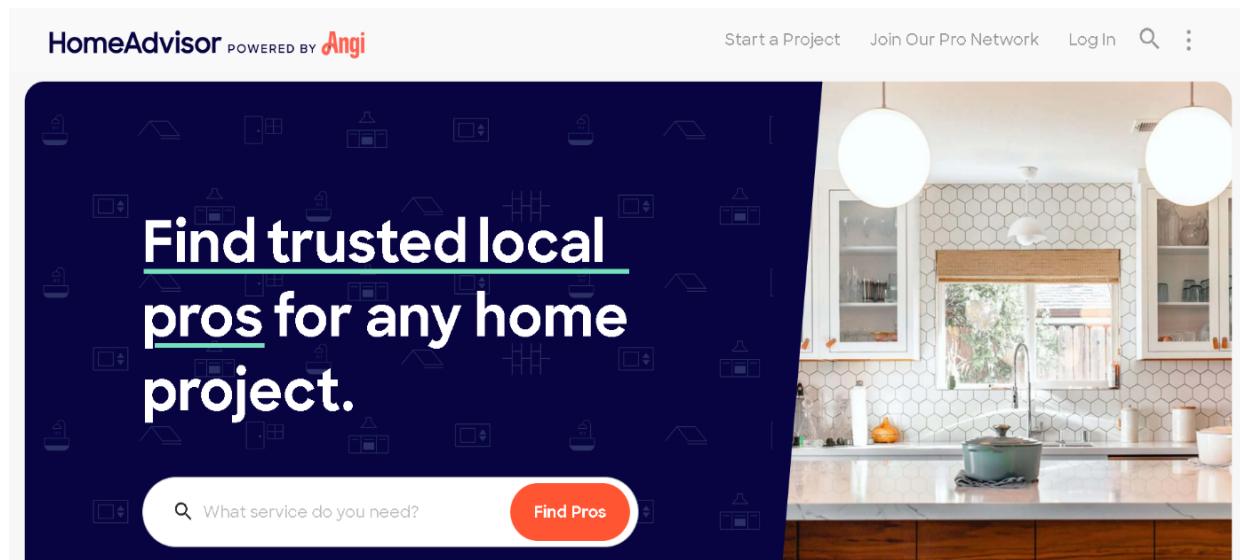


Figure 2.6 La page d'accueil du site HomeAdvisor.

- **Handy**

Handy est la principale plate-forme de mise en relation des particuliers à la recherche de services ménagers avec des professionnels de services indépendants présélectionnés et de qualité supérieure. Du nettoyage à domicile aux services de bricolage, Handy met en relation des milliers de clients chaque semaine avec des professionnels de premier plan dans des villes du monde entier. Avec un processus de réservation transparent de 60 secondes, un paiement sécurisé et soutenu par la garantie Handy Happiness, Handy est le moyen le plus simple et le plus pratique de réserver des services à domicile.

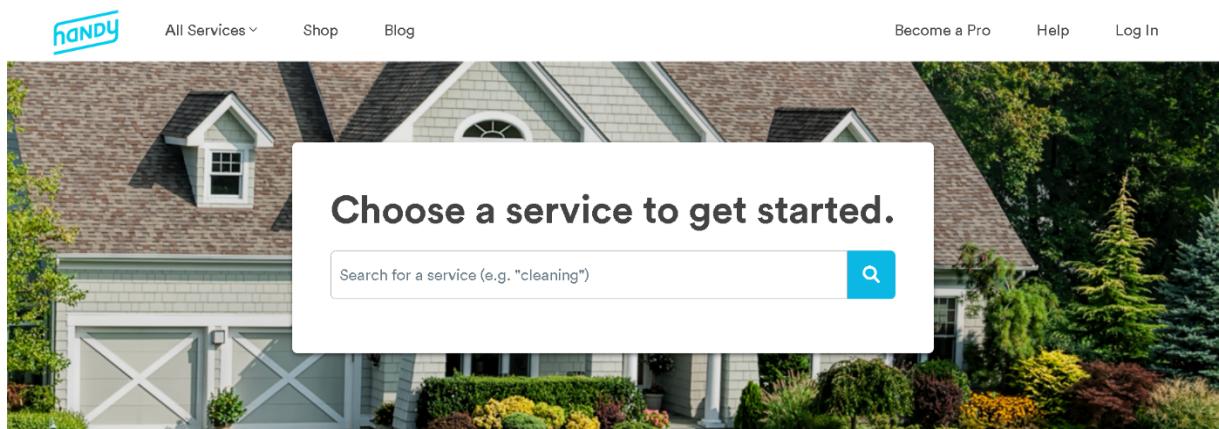


Figure 2.7 La page d'accueil du site Handy.

● Thumbtack

Thumbtack est un site Web américain de services à domicile. Il s'agit d'un répertoire en ligne qui permet aux utilisateurs de rechercher, d'évaluer et d'embaucher des fournisseurs de services locaux pour travailler sur une variété de projets personnels, y compris la rénovation domiciliaire, les services financiers et juridiques et la planification d'événements.

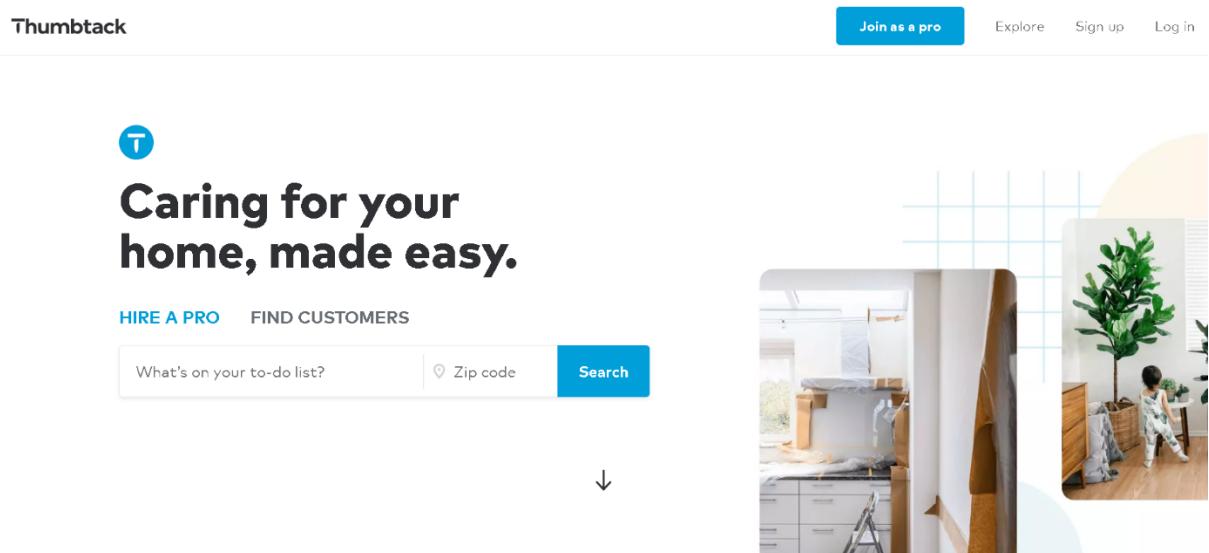


Figure 2.8 La page d'accueil du site Thumbtack.

2.5.4 Analyse concurrentielle

Toutes ces applications sont destinées aux utilisateurs disposants d'accès internet, elles offrent plusieurs services du domaine de bricolage : trouver un professionnel pour votre panne, prendre un rendez-vous avec un professionnel, postuler les réalisations pour les professionnels dont chacune ses propres avantages par rapport aux autres.

BRICALL :

- Possibilité de créer un profile
- Voir le profil du professionnel
- Trouver un professionnel par sélection d'une catégorie
- Donner un avis sur le travail d'un professionnel
- Prendre un rendez-vous avec un professionnel
- Postuler les réalisations pour les professionnels

SAWEBLIA :

- La possibilité du suivi de la demande
- Choix de la catégorie
- La possibilité de demande de devis

HomeAdvisor/Handy/Thumbtack :

- Création du profil simple ou professionnel
- Recherche par mot clé
- Choix d'une catégorie
- Recherche par code postal

2.6 Conclusion

Dans ce chapitre, on a présenté quelques définitions liées au service bricolage, son évolution et son fonctionnement, enfin on a énuméré ces différentes fonctionnalités offertes qui seront utiles dans le prochain chapitre. Enfin on a fait une présentation de quelques services concurrentiels sous formes d'applications/site web, suite à une analyse concurrentielle dont on a cité les avantages de chacune par rapport aux autres.

3 Conception

3.1 Introduction

La phase de conception est la première étape dans la réalisation d'un projet, elle doit décrire de manière non ambiguë le fonctionnement futur du système, afin d'en faciliter la réalisation. Pour cela, différentes méthodes existent permettant de formaliser les étapes préliminaires du développement.

Dans ce chapitre, nous présentons les objectifs de notre application, ce qui nous amène à identifier les possibilités du système et les besoins des utilisateurs que nous essayons de projeter dans des diagrammes de cas d'utilisations globaux et détails.

3.2 Spécification des besoins

3.2.1 Les besoins attendus de l'application

L'application envisagée doit satisfaire les besoins fonctionnels qui seront exécutés par le système et les besoins non fonctionnels qui perfectionnent la qualité logicielle du système.

3.2.1.1 Les besoins fonctionnels

Les besoins fonctionnels ou besoins métiers représentent les actions que le système doit exécuter, il ne devient opérationnel que s'il les satisfait. Cette application doit couvrir principalement les besoins fonctionnels suivants :

- Inscription des utilisateurs.
- Choix des catégories.
- Recherche du service souhaité.
- Enregistrement des profils des bricoleurs pour prochain accès.
- Accès direct à l'application téléphonique pour une rapidité optimale.
- Possibilité de poster des photos concernant les échantillons du travail qui concerne bricoleur.
- Possibilité de créer un profil professionnel pour bricoleur

3.2.1.2 Les besoins non fonctionnels

Ce sont des exigences qui ne concernent pas spécifiquement le comportement du système mais plutôt identifient des contraintes internes et externes du système. Les principaux besoins non fonctionnels de notre application se résument comme suit :

Le code doit être clair pour permettre de futures évolutions ou améliorations.

L'ergonomie : l'application offre une interface conviviale et facile à utiliser.

La sécurité : l'application doit respecter la confidentialité des données.

Garantir l'intégrité et la cohérence des données chaque mise jour et à chaque insertion

3.2.2 Le langage UML

UML (Unified Modeling Language), se définit comme un langage de modélisation graphique et textuel destiné à comprendre et définir des besoins, spécifier et documenter des systèmes, esquisser des architectures logicielles, concevoir des solutions et communiquer des points de vue. UML modélise l'ensemble des données et des traitements en élaborant différents diagrammes.

En clair, il ne faut pas designer UML en tant que méthode (Il y manque la démarche) mais plutôt comme une boîte d'outils qui sert à améliorer les méthodes de travail.

3.2.3 L'intérêt d'UML

UML est un langage semi-formel et normalisé :

- Gain de précision.
- Gagne de stabilité.
- Encourage l'utilisation d'outils.

UML est un support de communication performant :

- Il cadre l'analyse
- Il facilite la compréhension de représentations abstraites complexes.
- Son caractère polyvalent et sa souplesse en font un langage universel.

3.2.4 Définition d'un modèle

Un modèle est une vue subjective mais pertinente de la réalité. Un modèle définit une frontière entre la réalité et la perspective de l'observateur. Ce n'est pas "la réalité", mais une vue très subjective de la réalité. Bien qu'un modèle ne représente pas une réalité absolue, un modèle reflète des aspects importants de la réalité, il en donne donc une vue juste et pertinente.

3.2.5 Les différents types diagrammes d'UML

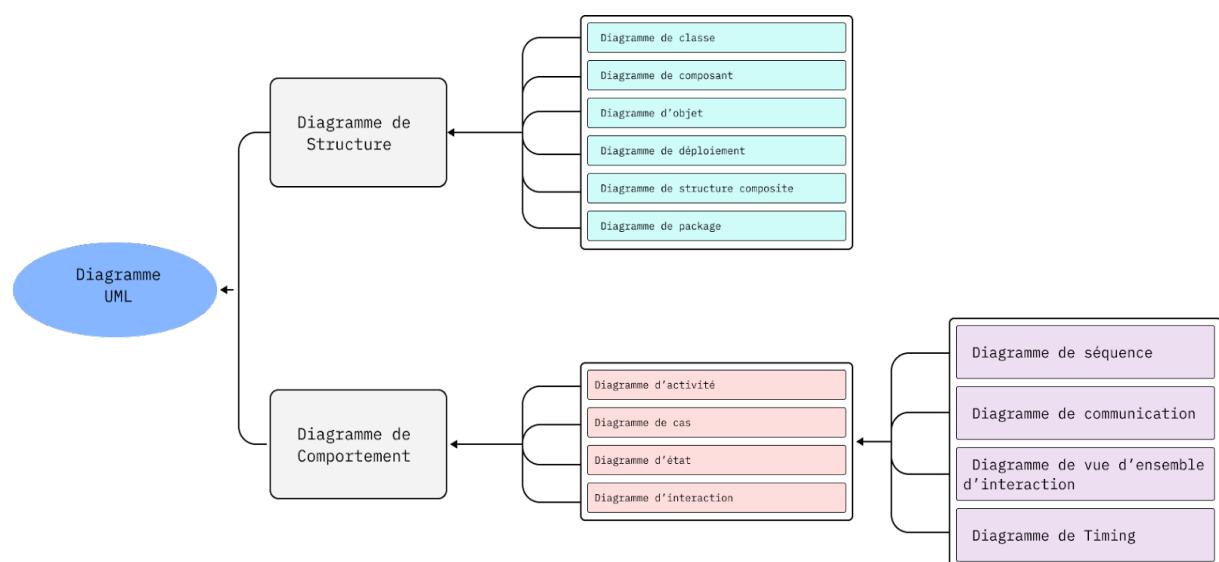


Figure 3.1 les diagrammes UML.

3.3 Identification des acteurs

Un acteur représente un rôle joué par un utilisateur humain ou un autre système qui interagit directement avec le système étudié. Un acteur participe à au moins un cas d'utilisation.

3.4 Diagramme de contexte du système réaliser

La figure ci-dessous montre l'interaction entre un acteur qui est l'utilisateur/Bricoleur/Admin et le système que nous allons mettre en place :

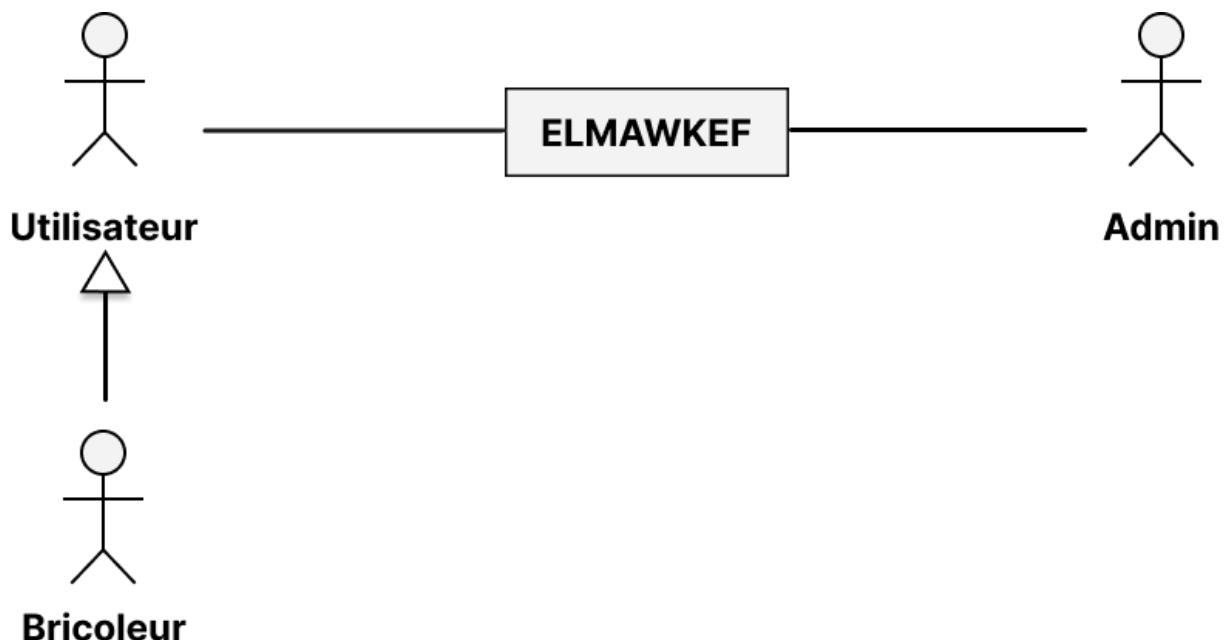


Figure 3.2 Diagramme de contexte .

3.5 Les Diagrammes des cas d'utilisation

Le diagramme de cas d'utilisation représente la structure des grandes fonctionnalités nécessaires aux utilisateurs du système. C'est le premier diagramme du modèle UML, celui où s'assure la relation entre l'utilisateur et les objets que le système met en œuvre.

3.5.1 Le diagramme global des cas d'utilisations

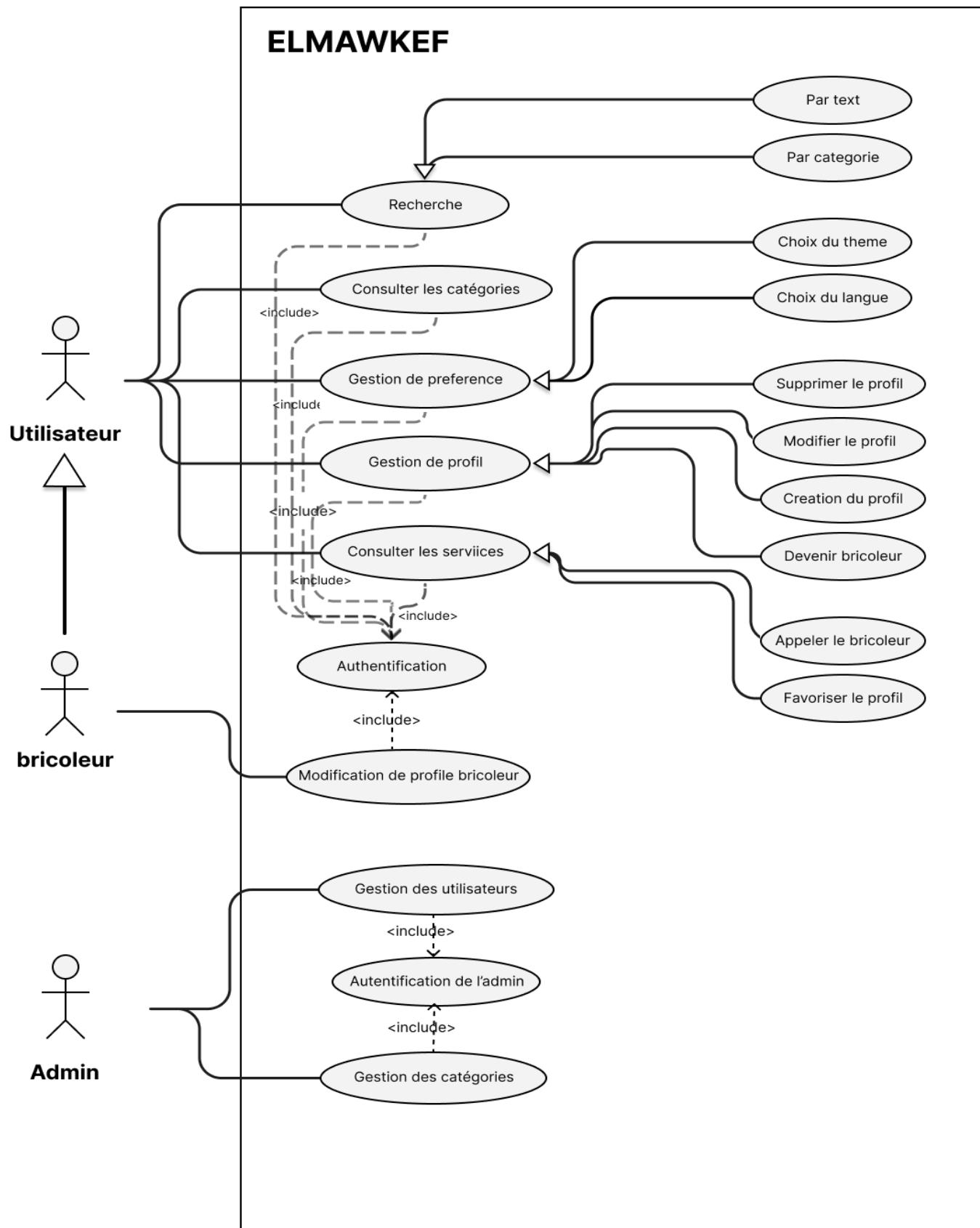


Figure 3.3 Diagramme global des cas d'utilisations associés aux acteurs.

3.5.2 Les différents cas d'utilisation

L'étude de cas d'utilisation a pour objectif de déterminer ce que chaque utilisateur attend du système. La détermination du besoin est basée sur la représentation de l'interaction entre l'acteur et le système.

Chaque cas d'utilisation doit être associée une description textuelle des interactions entre l'acteur et le système et les actions que le système doit réaliser en vue de produire les résultats attendus par les acteurs. Pour exprimer les cas d'utilisations de notre système, nous avons choisi le formalisme suivant :

3.5.3 Le cas d'utilisation <>Recherche>>

Le cas d'utilisation N° 1	Recherche
Résumé	Recherche d'un service ou bricoleur
Acteurs	Utilisateur
Precondition	L'utilisateur doit être dans l'application et authentifié
Scénario nominal	1. L'utilisateur recherche un service souhaité 2. Le système affiche les résultats basés sur le nom du service ou catégorie recherchée.

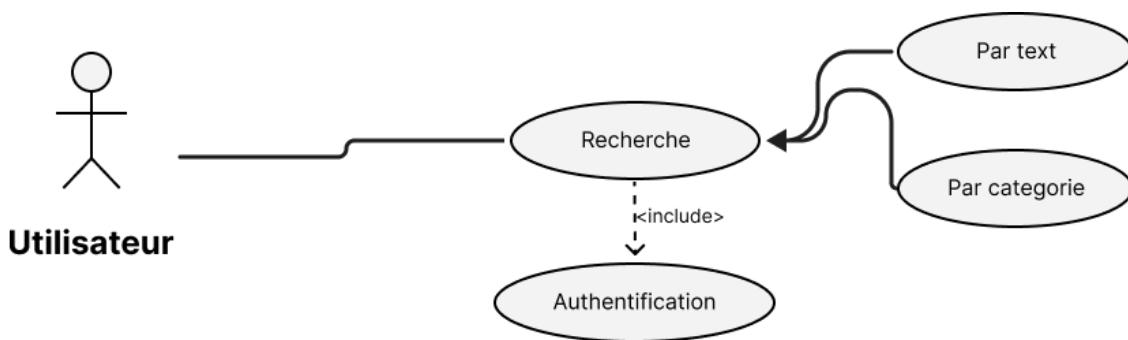


Figure 3.4 Diagramme de cas d'utilisation <>Recherche>>

3.5.4 Le cas d'utilisation <<Consultation des catégories >>

Le cas d'utilisation N° 1	Consultation des catégories
Résumé	Consultation des catégories des services proposés
Acteurs	Utilisateur
Precondition	L'utilisateur doit être dans l'application et authentifié
Scénario nominal	1. choix des catégorie listé dans la section catégorie 2. Le système affichera les services liés à la catégorie choisie.

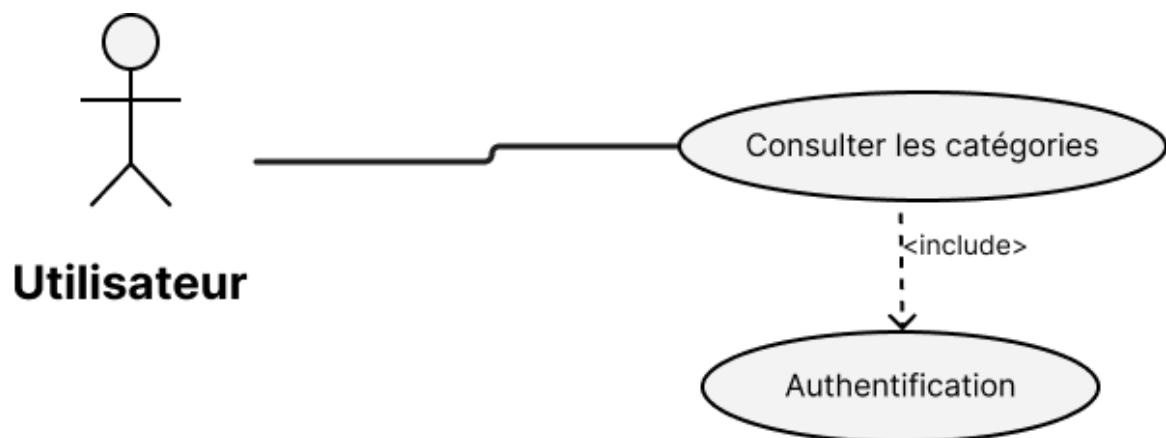


Figure 3.5 Diagramme de cas d'utilisation <<Consultation des catégories >>

3.5.5 Le cas d'utilisation <<Préférence>>

Le cas d'utilisation N° 1	Préférence
Résumé	choix des préférence thème et langue
Acteurs	Utilisateur
Precondition	L'utilisateur doit être dans l'application et authentifié
Scénario nominal	Les préférences sont par défaut déterminées suivant les paramètres du téléphone mais l'utilisateur pourra toujours les modifier dans la section du menu ,une fois sélectionné le système fournira les préférences choisies.

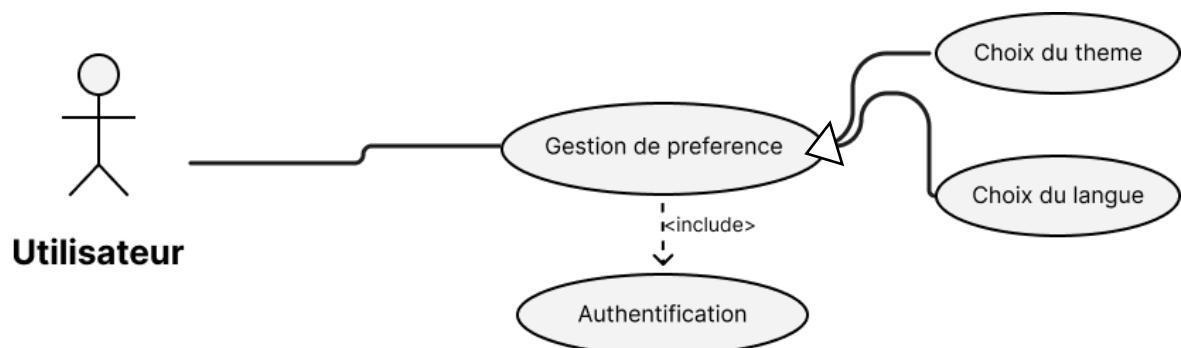


Figure 3.6 Diagramme de cas d'utilisation <<Préférence>>

3.5.6 Le cas d'utilisation <>gestion profil>>

Le cas d'utilisation Nº 1	gestion profil
Résumé	la gestion du profil de l'utilisateur
Acteurs	utilisateur
Precondition	L'utilisateur doit être dans l'application et authentifié
Scénario nominal	<ol style="list-style-type: none"> 1. la création du profil de l'utilisateur . 2.ajouter des informations. 3.modification des informations. 4.suppression des informations et du profil.

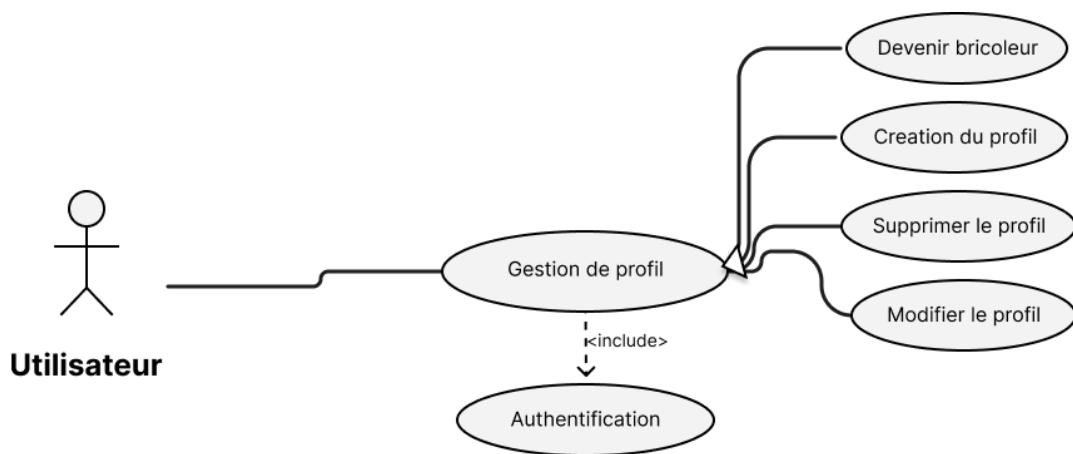


Figure 3.7 Diagramme de cas d'utilisation <>gestion profil>>

3.5.7 Le cas d'utilisation <<consultation des services >>

Le cas d'utilisation N° 1	consultation des services
Résumé	la consultation des services proposés
Acteurs	utilisateur
Precondition	L'utilisateur doit être dans l'application et authentifié
Scénario nominal	une fois l'utilisateur à cliquer sur le service à consulter ,il aura comme option d'appeler le bricoleur ou favoriser le profil.

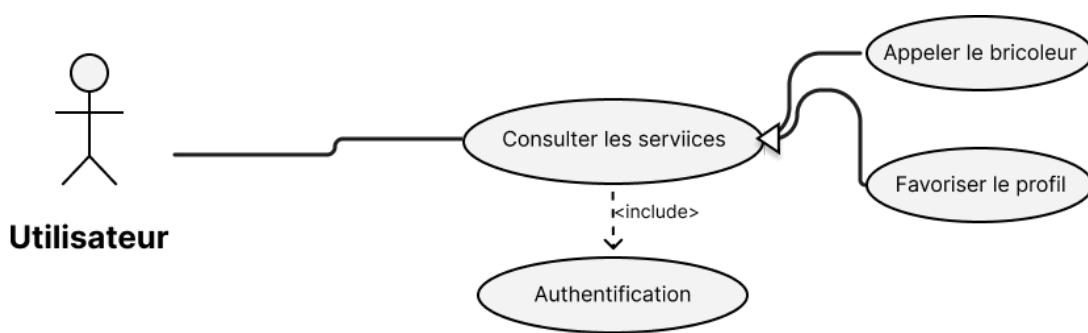


Figure 3.8 Diagramme de cas d'utilisation <<consultation des services >>

3.5.8 Le cas d'utilisation <<modification de profil bricoleur>>

Le cas d'utilisation Nº 1	modification de profil bricoleur
Résumé	modification des informations du profil de bricoleur
Acteurs	Bricoleur
Precondition	Le bricoleur doit être dans l'application et authentifié
Scénario nominal	<ol style="list-style-type: none"> 1.Ajouter des informations. 2.Modification des informations. 3.retour au profil normal. 4.Suppression des informations et du profil.

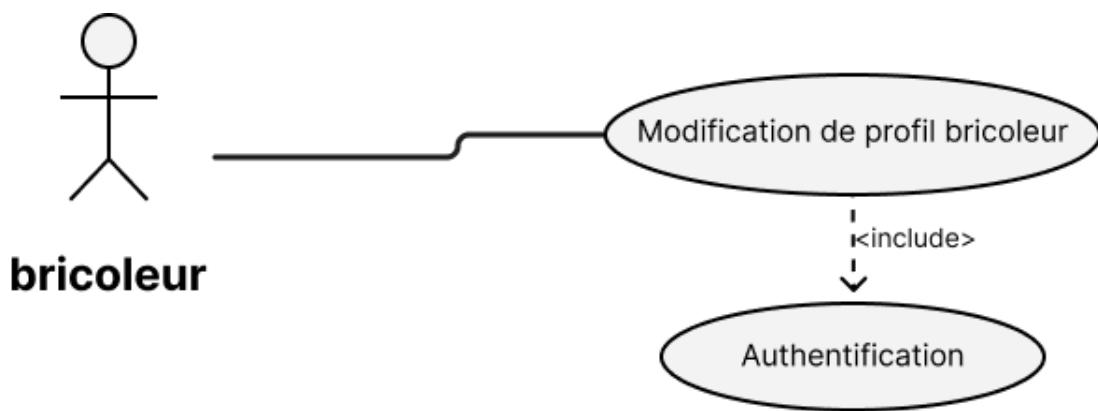


Figure 3.9 Diagramme de cas d'utilisation <<modification de profil bricoleur>>

3.5.9 Le cas d'utilisation <>gestion des profils par admin>>

Le cas d'utilisation N° 1	gestion des profils par admin
Résumé	la gestion du profil de l'utilisateur et bricoleur par l'administrateur .
Acteurs	Admin
Precondition	Admin doit être authentifié .
Scénario nominal	Admin a pour privilège d'ajouter modifier et supprimer les profils des utilisateurs et bricoleur.

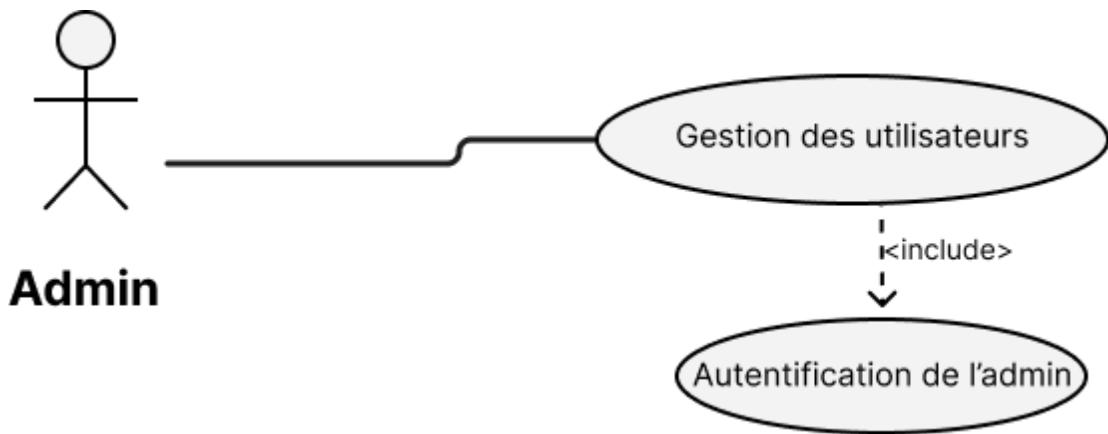


Figure 3.10 Diagramme de cas d'utilisation <>gestion des profils par admin>>

3.5.10 Le cas d'utilisation <>Gestion des catégories>>

Le cas d'utilisation N° 1	Gestion des catégories
Résumé	Gestion des catégories par admin
Acteurs	Admin
Precondition	Admin doit être authentifié .
Scénario nominal	Admin a pour privilège d'ajouter, modifier et supprimer les catégories des services.

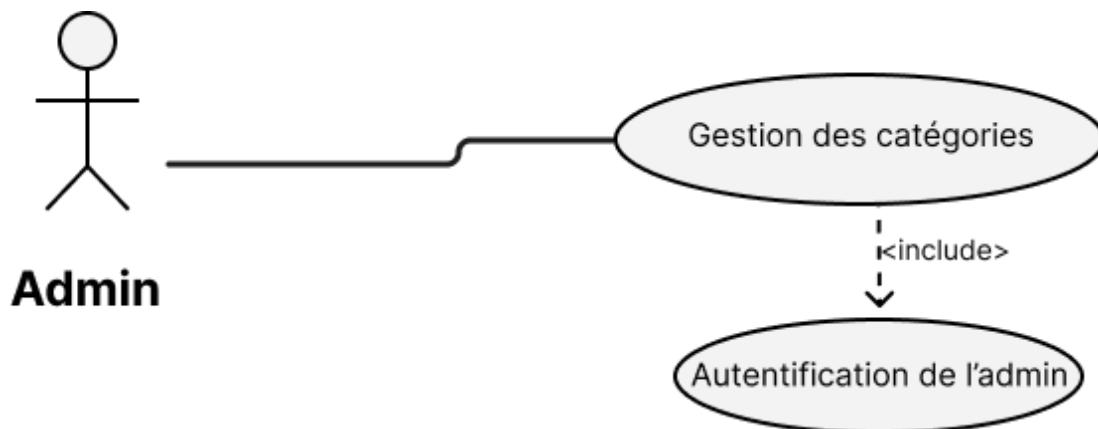


Figure 3.11 Diagramme de cas d'utilisation <>Gestion des catégories>>

3.6 Diagrammes de séquence

3.6.1 Définition d'un diagramme de séquence

Un diagramme de séquences est un diagramme d'interaction qui expose en détail la façon dont les opérations sont effectuées : quels messages sont envoyés et quand ils le sont.

Les diagrammes de séquences sont organisés en fonction du temps qui s'écoule au fur et mesure que nous parcourons la page.

Les objets impliqués dans l'opération sont répertoriés de gauche à droite en fonction du moment où ils prennent part dans la séquence.

3.6.2 Diagramme de séquence du cas d'utilisation Authentification

Le diagramme de séquence suivant illustre les interactions nécessaires pour l'authentification de l'utilisateur.

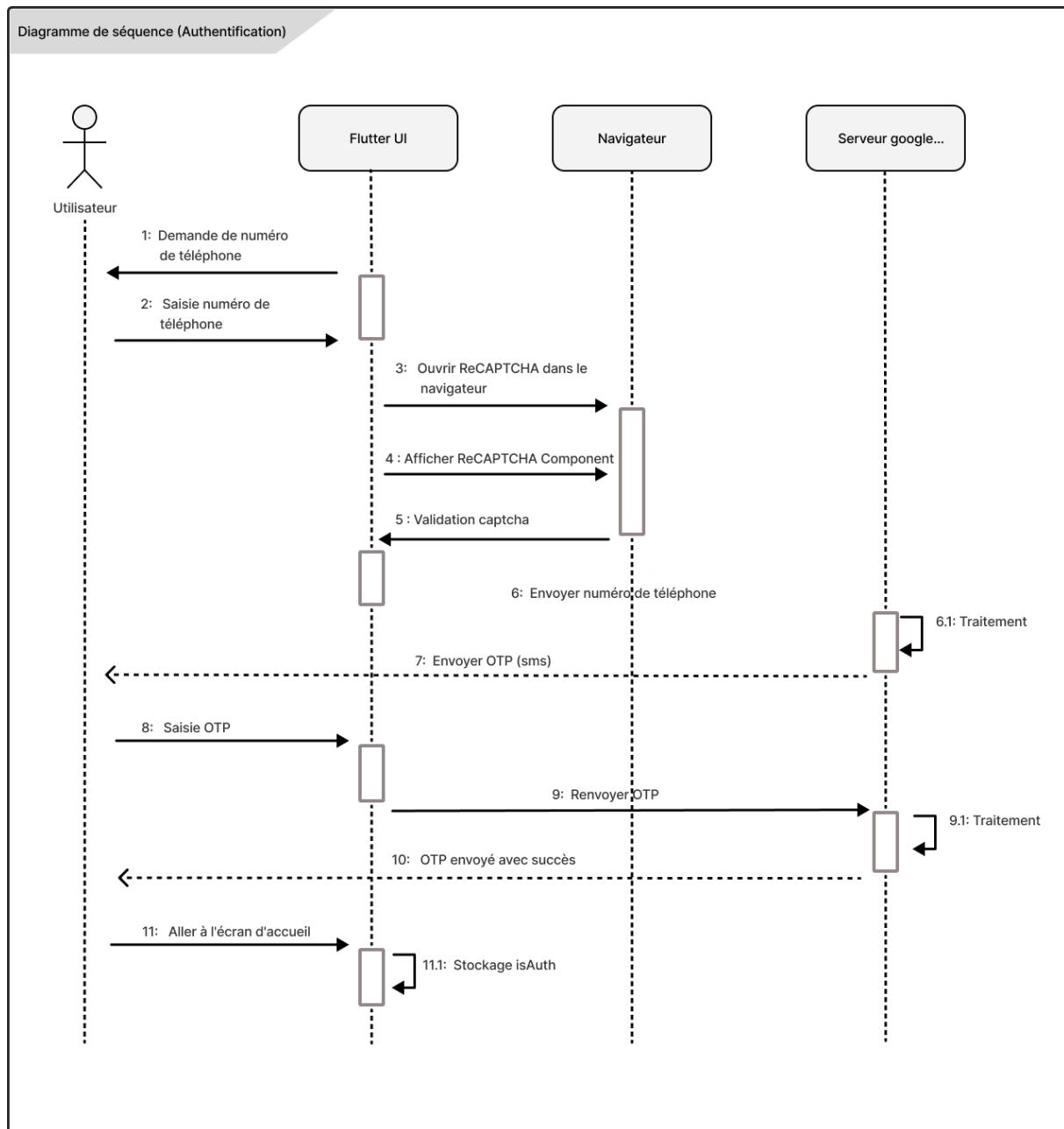


Figure 3.12 Diagramme de séquence du cas d'utilisation Authentification .

3.6.3 Diagramme de séquence du cas d'utilisation consultation des catégories

Le diagramme de séquence suivant illustre les interactions nécessaires pour consulter les catégories.

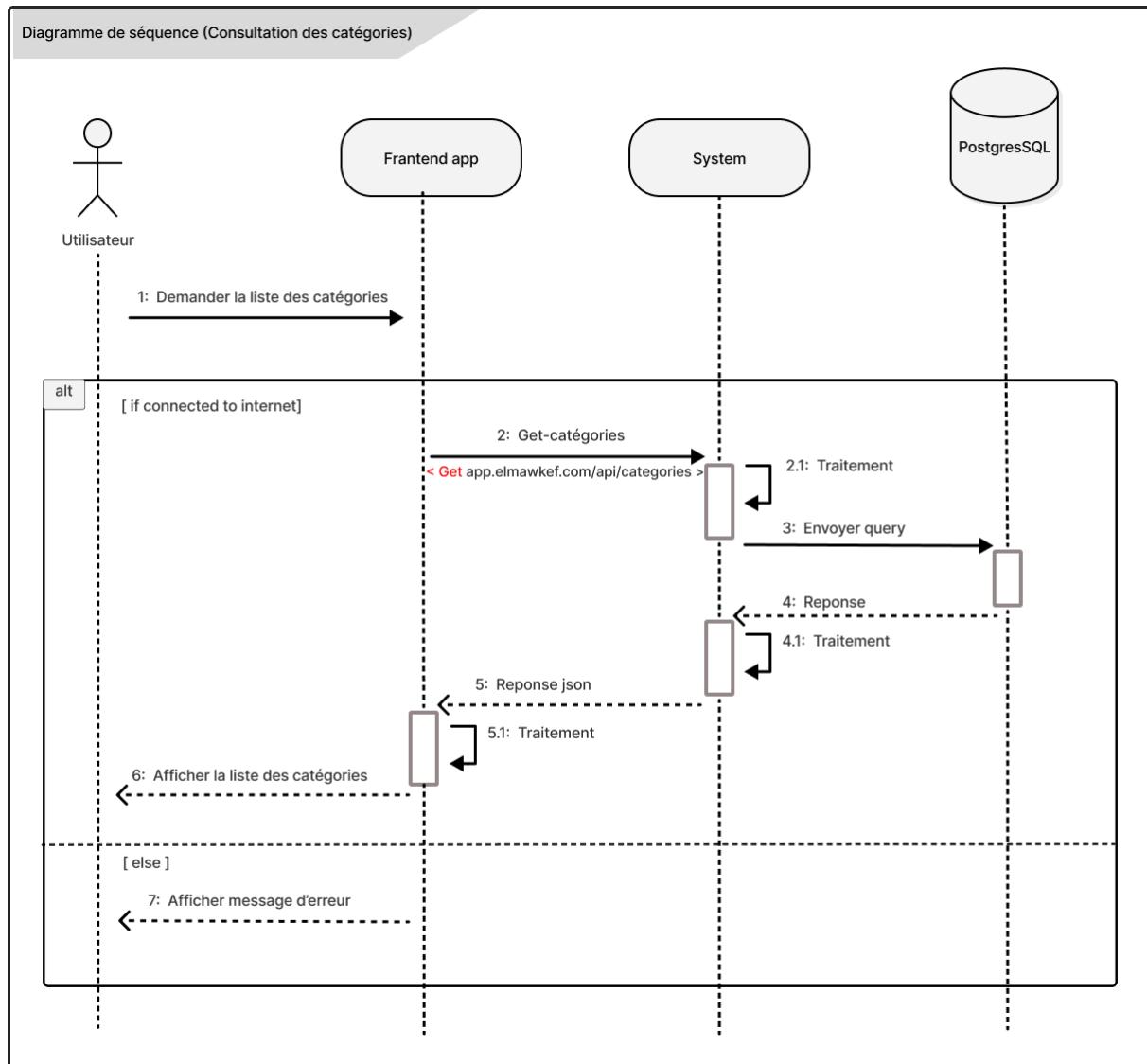


Figure 3.13 Diagramme de séquence du cas d'utilisation consultation catégories .

3.6.4 Diagramme de séquence du cas d'utilisation consultation des services.

Le diagramme de séquence suivant illustre les interactions nécessaires pour consulter les services.

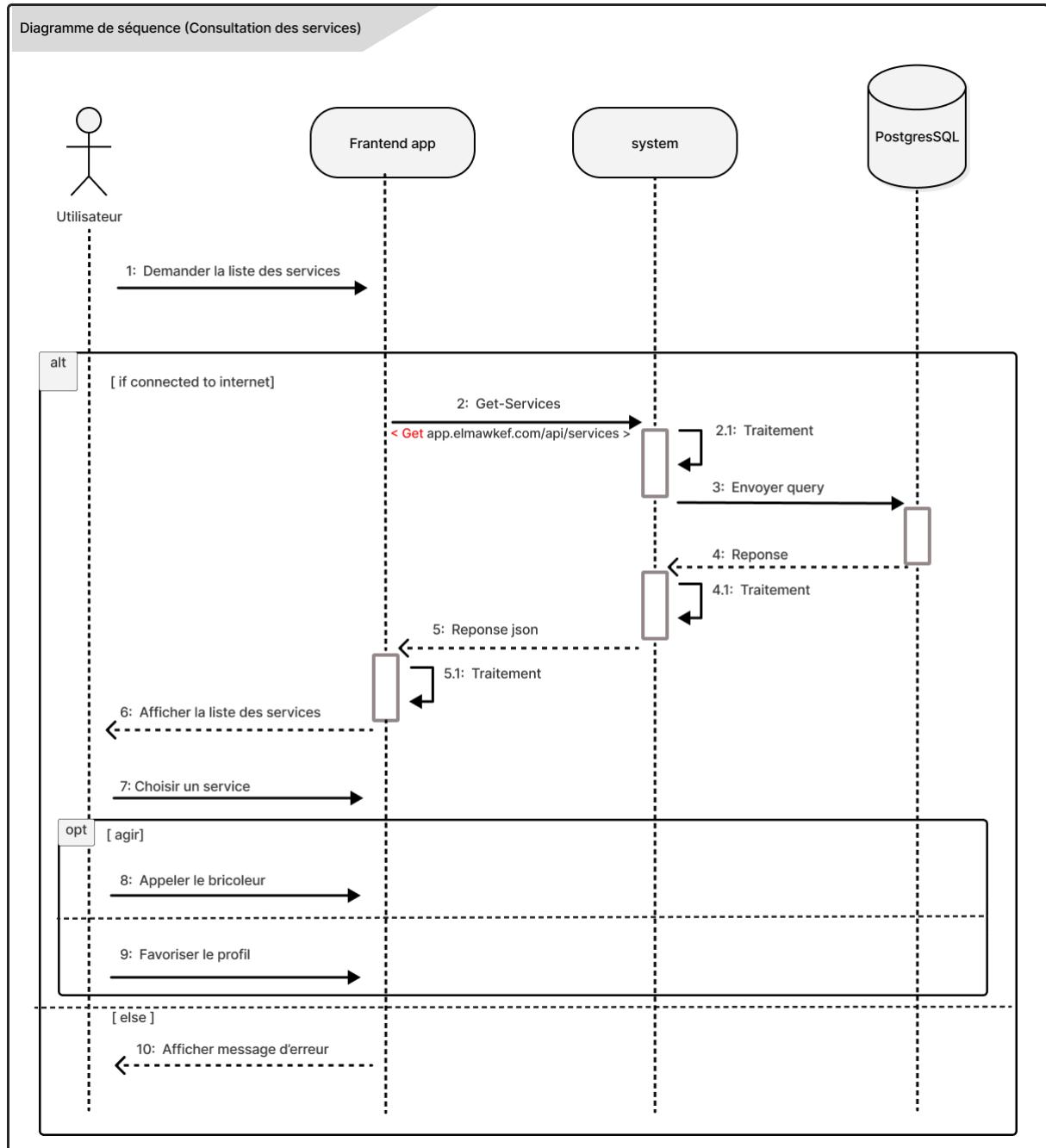


Figure 3.14 Diagramme de séquence du cas d'utilisation Consultation des services .

3.6.5 Diagramme de séquence du cas d'utilisation recherche service.

Le diagramme de séquence suivant illustre les interactions nécessaires pour la recherche d'un service.

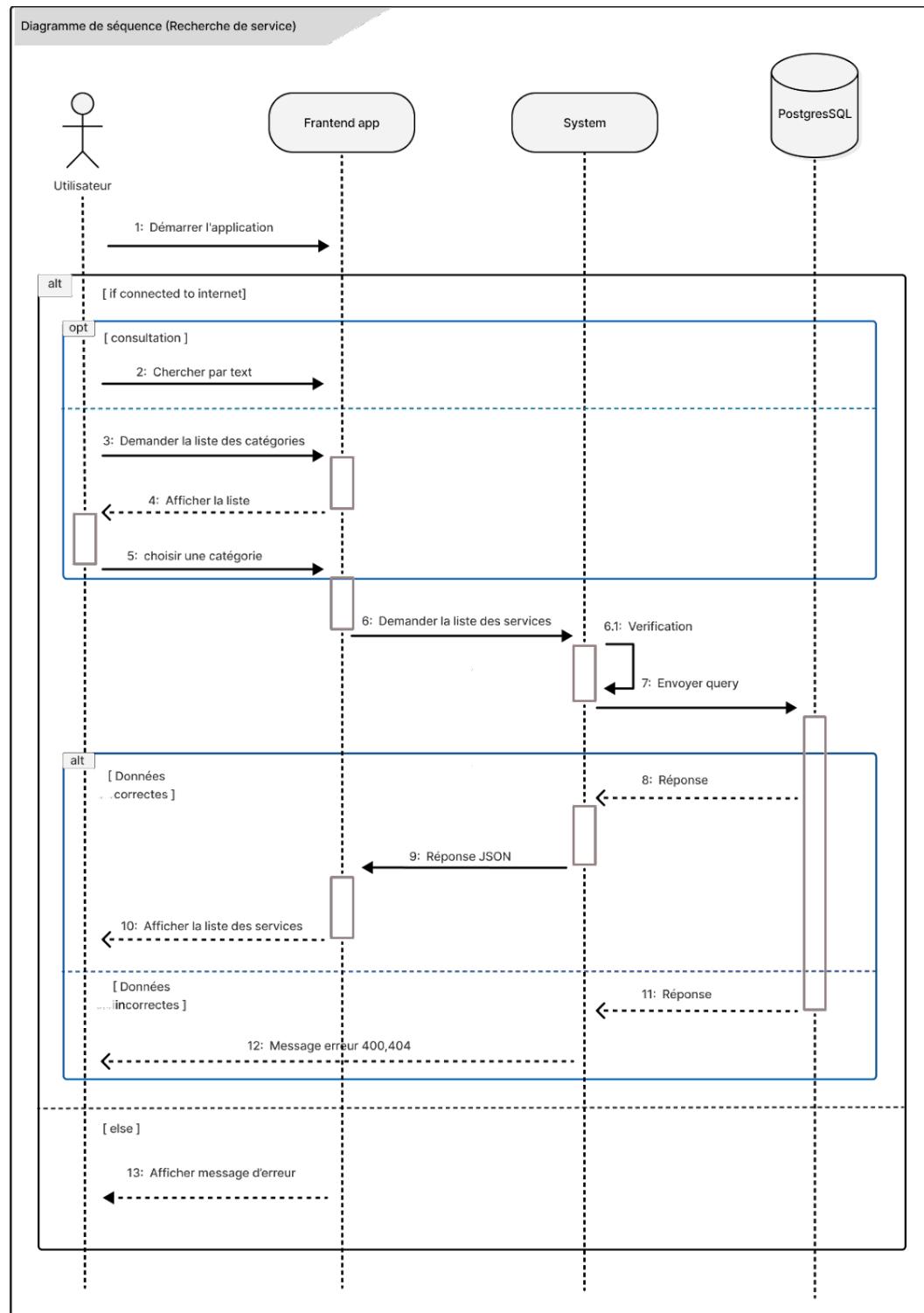


Figure 3.15 Diagramme de séquence du cas d'utilisation pour la recherche d'un service .

3.6.6 Diagramme de séquence du cas d'utilisation Gestion de profil

Le diagramme de séquence suivant illustre les interactions nécessaires pour la gestion du profil.

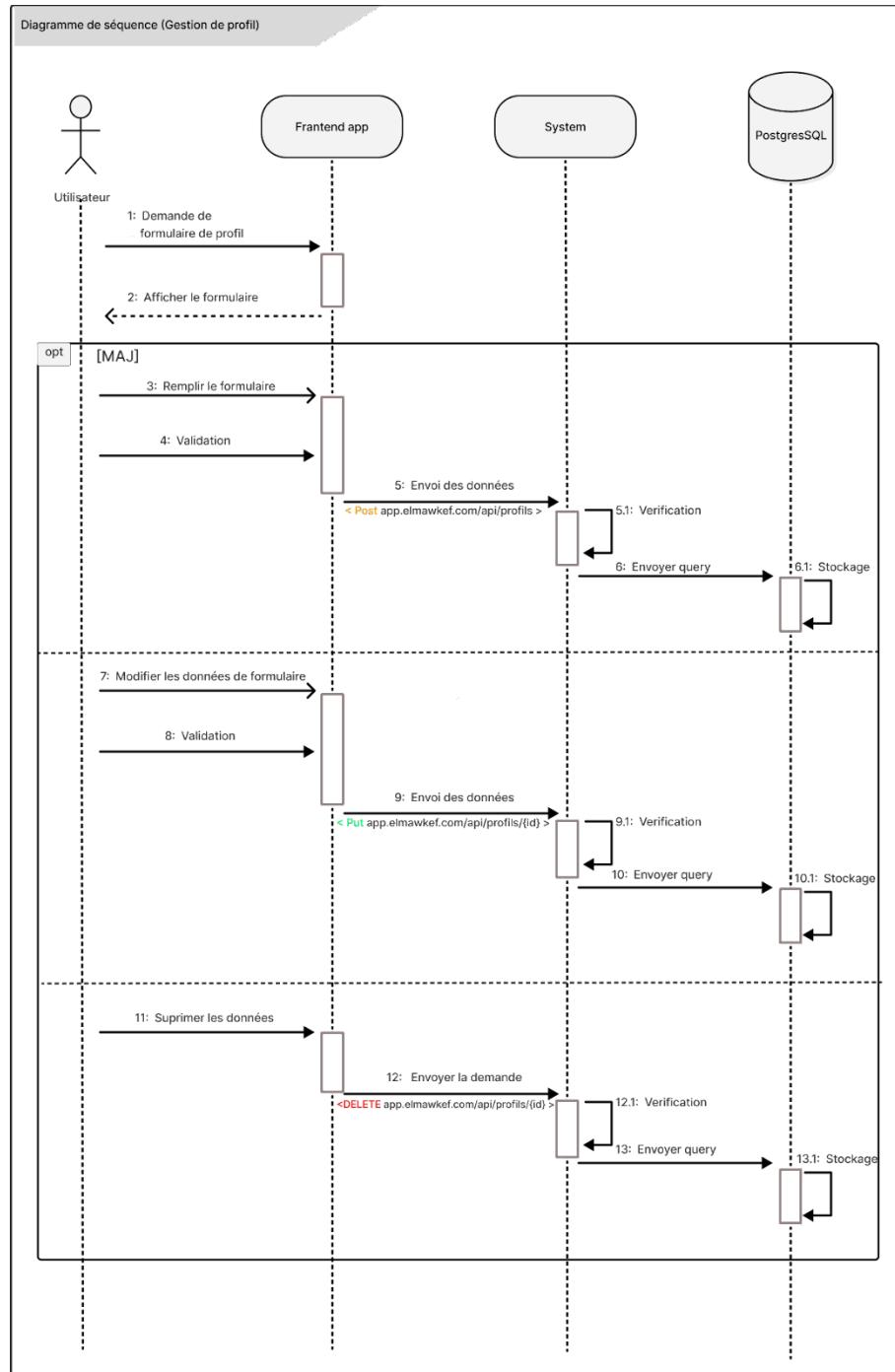


Figure 3.16 Diagramme de séquence du cas d'utilisation gestion du profil

3.6.7 Diagramme de séquence du cas d'utilisation Choisir thème/langue

Le diagramme de séquence suivant illustre les interactions nécessaires pour changer le thème ou la langue.

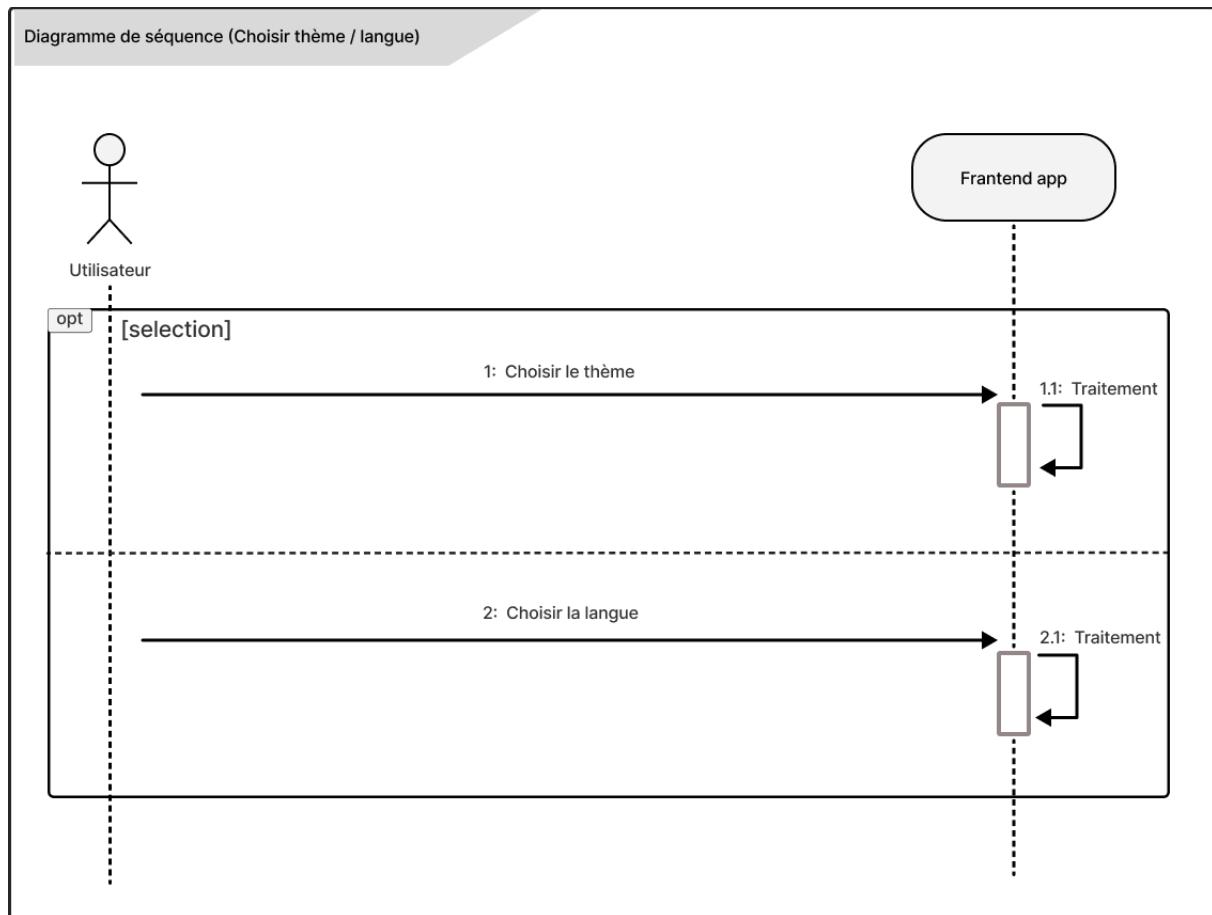


Figure 3.17 Diagramme de séquence du cas d'utilisation choisir thème /langue .

3.7 Diagramme de classe

Le diagramme de classes est considéré comme le plus important de la modélisation orientée objet, il est le seul obligatoire lors d'une telle modélisation. Il montre la structure interne. Il permet de fournir une représentation abstraite des objets du système qui vont interagir pour réaliser les cas d'utilisation. Il est important de noter qu'un même objet peut très bien intervenir dans la réalisation de plusieurs cas d'utilisation. Les cas d'utilisation ne réalisent donc pas une partition des classes du diagramme de classes. Un diagramme de classes n'est donc pas adapté (sauf cas particulier) pour détailler, décomposer, ou illustrer la réalisation d'un cas d'utilisation particulier.

Il s'agit d'une vue statique, car on ne tient pas compte du facteur temporel dans le comportement du système. Le diagramme de classes modélise les concepts du domaine d'application ainsi que les concepts internes créés de toutes pièces dans le cadre de l'implémentation d'une application. Chaque langage de Programmation orienté objet donne un moyen spécifique d'implémenter le paradigme objet (pointeurs ou pas, héritage multiple ou pas, etc.), mais le diagramme de classes permet de modéliser les classes du système et leurs relations indépendamment d'un langage de programmation particulier.

Les principaux éléments de cette vue statique sont les classes et leurs relations : association, généralisation et plusieurs types de dépendances, telles que la réalisation et l'utilisation.

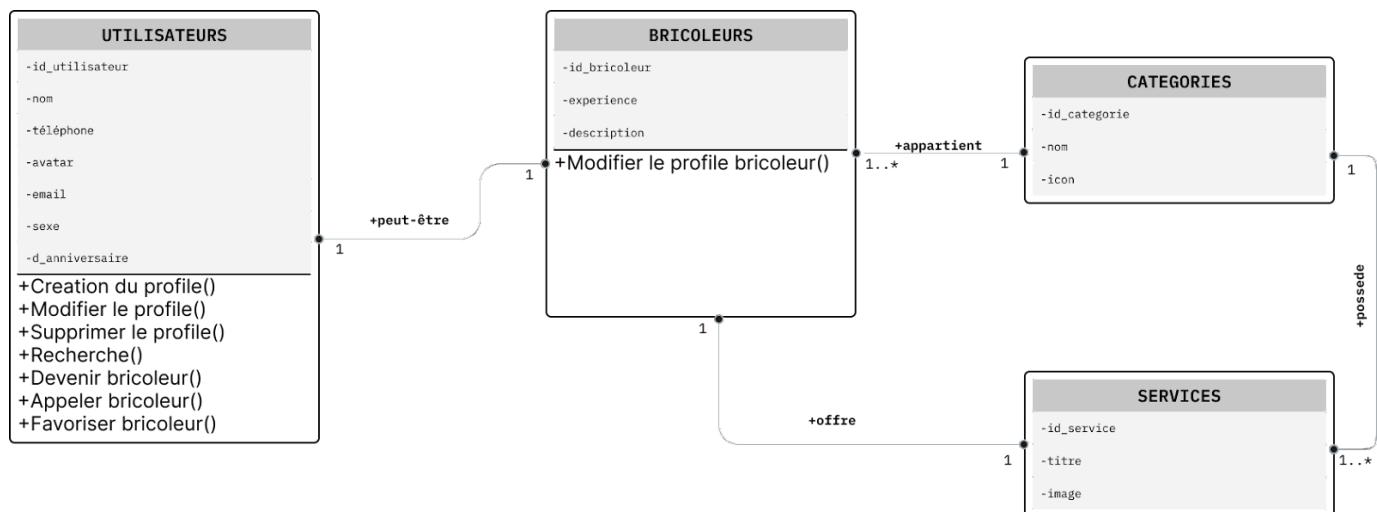


Figure 3.18 Diagramme de classe.

3.8 Modèle conceptuel de données (MCD)

Le **MCD** est une représentation graphique de haut niveau qui permet facilement et simplement de comprendre comment les différents éléments sont liés entre eux à l'aide de diagrammes codifiés.

Le MCD est représenté ci-dessous dans la figure suivante :

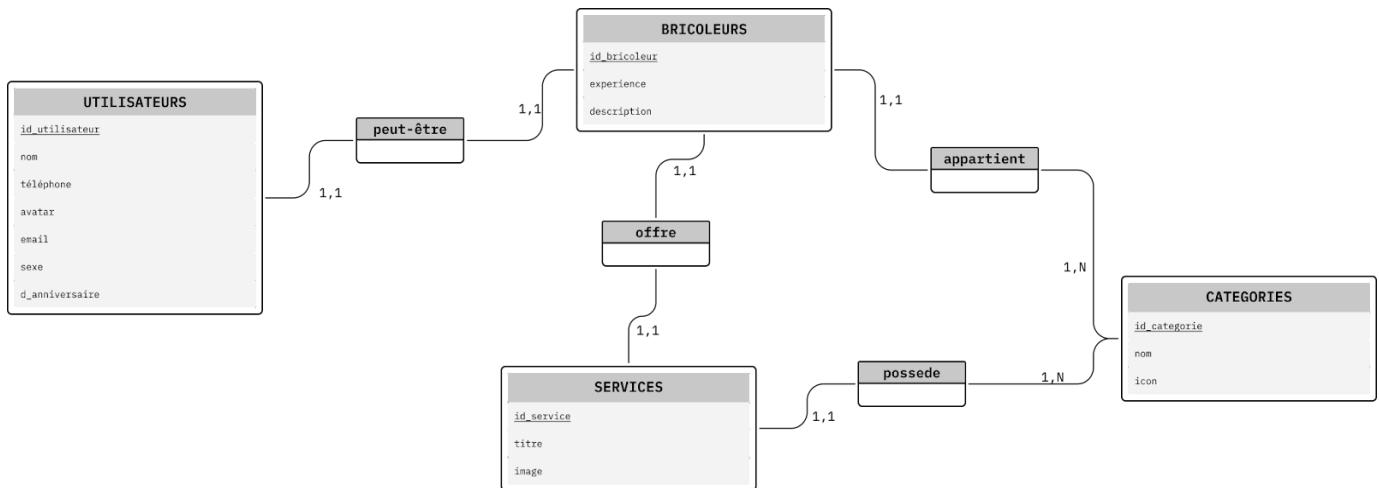


Figure 3.19 : MCD.

3.9 Modèle logique des données (MLD)

Le modèle logique de données (MLD) est une suite normale du processus MERISE dont l'objectif est de nous rapprocher du modèle physique. Pour cela, nous partons d'un modèle conceptuel des données en supprimant les relations, pas aléatoirement mais en respectant certaines règles.

**UTILISATEURS (id_utilisateur , nom , téléphone , avatar , email , sexe ,
d_anniversaire ,#id_bricoleur)**

id_utilisateur : clé primaire de la table UTILISATEURS

#id_bricoleur : clé étrangère de la table BRICOLEURS

**BRICOLEURS (id_bricoleur , experience , description , #id_utilisateur , #id_categorie
, #id_service)**

id_bricoleur : clé primaire de la table BRICOLEURS

#id_utilisateur : clé étrangère de la table UTILISATEURS

#id_categorie : clé étrangère de la table CATEGORIES

#id_service : clé étrangère de la table SERVICES

CATEGORIES(id_categorie , nom , icon)

id_categorie : clé primaire de la table CATEGORIES

SERVICES (id_service , titre , image , #id_bricoleur , #id_categorie)

id_service : clé primaire de la table SERVICES

#id_bricoleur : clé étrangère de la table BRICOLEURS

#id_categorie : clé étrangère de la table CATEGORIES

Figure 3.20 : MLDR

3.10 Entity Relationship Diagram ERD

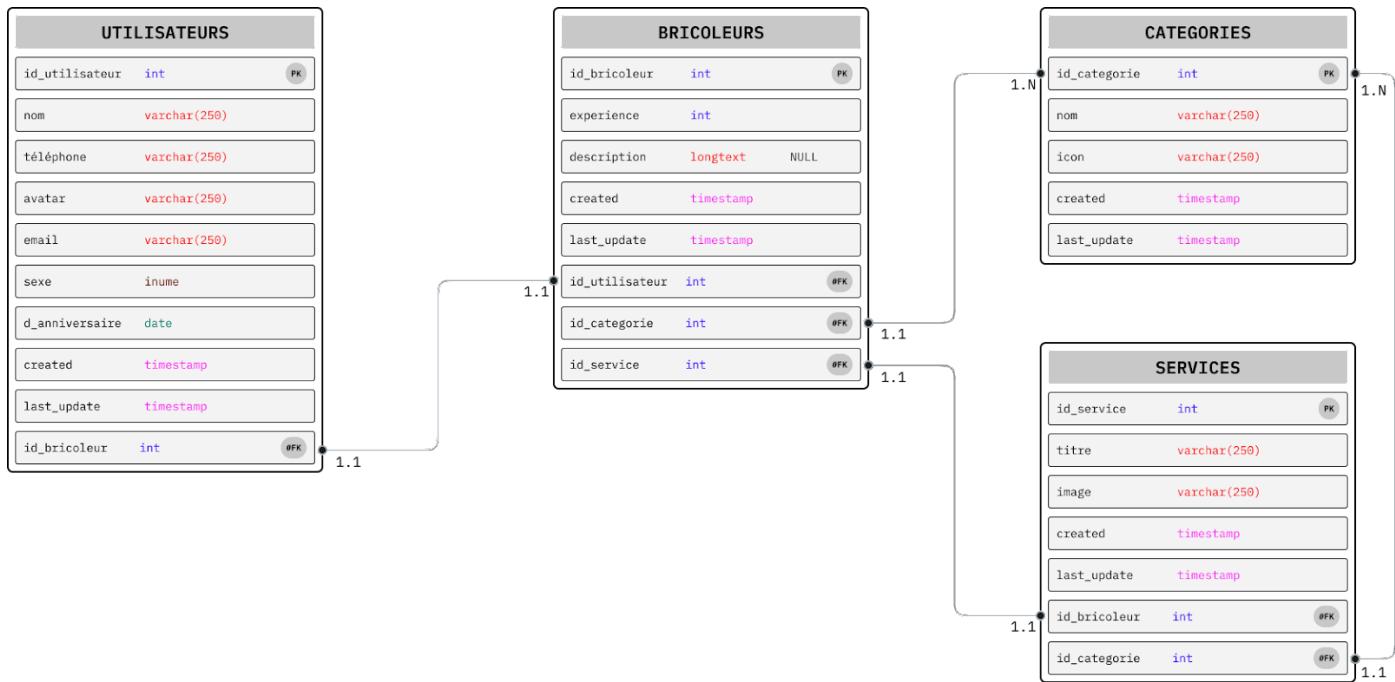


Figure 3.21 ERD

3.8 Conclusion

Ce chapitre nous a permis de présenter le langage de modélisation UML et l'approche devops. Ensuite on a déterminé le cadre du projet de réalisation de l'application et de définir les besoins, en identifiant toutes les entités internes qui vont interagir avec le système (acteurs). Nous avons représenté l'ensemble de séquence d'actions, en identifiant les cas d'utilisation de tous les acteurs du système. Ensuite, nous avons recensé la description graphique des cas d'utilisation, en réalisant des diagrammes de séquence décrivant les scénarios nominaux des cas d'utilisations essentiels.

Enfin, nous avons réalisé la conception appropriée de notre application selon les concepts de base de diagramme de classe ainsi que les règles de modélisation. Ensuite, nous avons recensé les règles de passage de diagramme de classe vers le modèle relationnel qui nous permet d'avoir le schéma de la base de données de l'application réaliser. Cela fait la base pour la phase de réalisation telle qu'on va garantir la fiabilité et l'efficacité.

4 Gestion du projet

Le présent chapitre a pour but définir l'essentiel du travail d'identification et planification du projet ainsi que l'organisation et le choix méthodologique adoptés qui est le devops.

4.1 Planning initial du projet

L'objectif de ce planning est de déterminer les étapes du projet et le timing. Ce planning joue un rôle primordial pour la réalisation et le suivi du projet, il est établi dans le début de chaque projet afin de suivre le bon déroulement de chaque tâche.

4.1.1 Scrum and Agile

Afin de savoir quoi développer, il est nécessaire d'avoir à disposition un outil permettant la collaboration entre les développeurs. Cet outil permettra notamment de gérer les différentes releases et toutes les fonctionnalités, de garantir la priorité du backlog, etc.

Intervenant tout au long du projet, la collaboration de toute l'équipe est nécessaire pour assurer la planification du projet. Cette planification est étroitement liée à la méthodologie Scrum. Elle a pour but de découper le projet en petites tâches à réaliser par toute l'équipe.

Sur notre chaîne d'assemblage, la planification représente la répartition des tâches de chaque collaborateur, ou encore le brief du matin destiné à définir les tâches à faire durant la journée.

après la préparation et le paiement des comptes des services qui vont être utilisés.

On a procédé à la configuration de notre server et notre base de donne

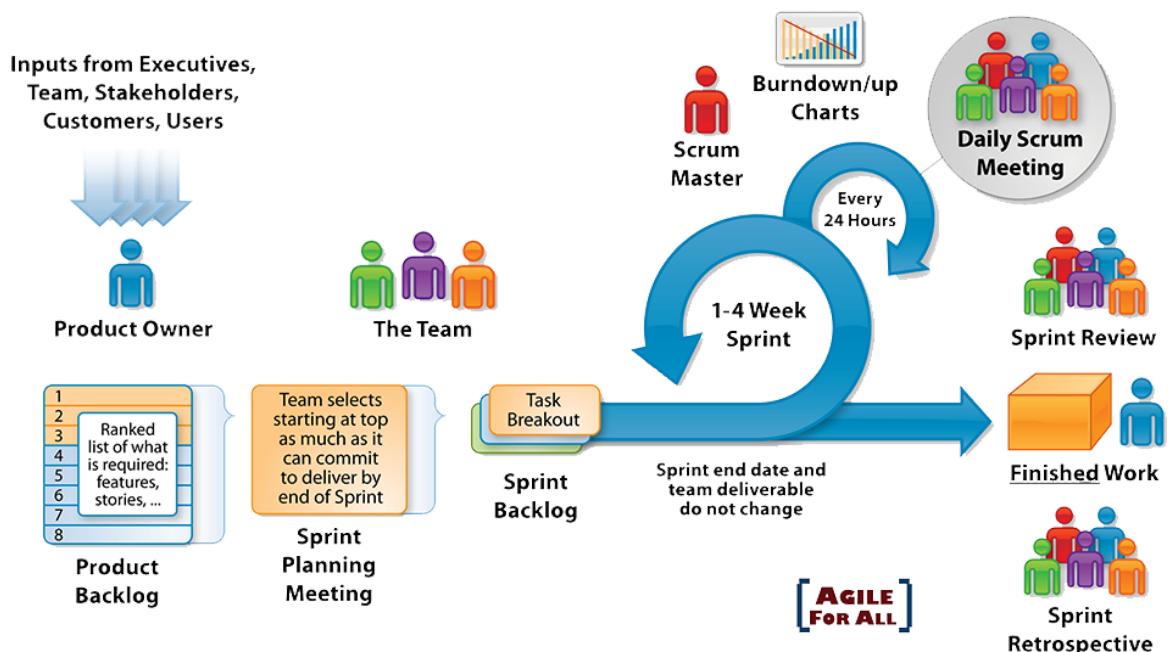


Figure 4.1 scrum framework

4.1.1.1 les étiquettes

Ce board sera notre board par défaut durant tout notre projet, afin de voir son avancement et de pouvoir catégoriser les tâches que nous allons ouvrir.

The screenshot shows a Trello board titled "Labels Documentation". The board has a cover image of a neon sign that says "LABELS". On the left, there's a sidebar with a "Description" section and an "Activity" section. The main board area contains several cards, each with a title and a detailed description. To the right of the cards is a vertical sidebar with various actions and settings.

Action	Description
Suggested	<input checked="" type="checkbox"/> Join
Add to card	<input checked="" type="checkbox"/> Members
Labels	<input checked="" type="checkbox"/> Checklist
Dates	<input checked="" type="checkbox"/> Attachment
Attachment	<input checked="" type="checkbox"/> Custom Fields
Custom Fields	<input checked="" type="checkbox"/> Power-Ups
Power-Ups	<input checked="" type="checkbox"/> Add Power-Ups
Automation	<input checked="" type="checkbox"/> Add button
Add button	<input checked="" type="checkbox"/> Actions
Actions	<input checked="" type="checkbox"/> Move
Move	<input checked="" type="checkbox"/> Copy
Copy	<input checked="" type="checkbox"/> Make template
Make template	<input checked="" type="checkbox"/> Watch
Watch	<input checked="" type="checkbox"/> Archive
Archive	<input checked="" type="checkbox"/> Share
Share	

Que sont les étiquettes ?

Les étiquettes sont d'excellents moyens de différencier le type de tâches de votre équipe.

Définition:

- (vert) **UML**: La modélisation est l'étape la plus importante du cycle du développement d'application. Elle se base essentiellement sur la bonne spécification et l'analyse des besoins.
- (mauve) **Analyse de l'existant**: C'est la phase du projet pendant laquelle le Business Analyst va auditer les processus et les solutions informatiques existants. Elle est réalisée avant l'initialisation du changement. Elle permet de préparer l'analyse des besoins de la solution cible et de réaliser l'analyse des écarts.
- (rouge) **Design et UI**: C'est l'une des étapes les plus importantes où on exige à offrir à l'utilisateur la meilleure expérience possible en termes de design et interaction de notre application.
- (bleu) **déploiement**: Le déploiement aussi appelé déploiement continu, en complément du processus de distribution continue, automatise la publication d'une version prête pour la production dans un référentiel de code.
- (jaune) **Utilisateur authentifié**: l'utilisateur doit être authentifié pour qu'il ait accès aux fonctionnalités que notre système a à offrir .
- (noir) **backend** : c'est toute la partie que l'utilisateur ne voit pas, mais qui lui permet de réaliser des actions sur un site ou une application.

Activity

Show details

Write a comment...

Figure 4.2 étiquettes

4.1.1.2 user story

Un récit utilisateur, ou « user story » en anglais, est une description simple d'un besoin ou d'une attente exprimée par un utilisateur et utilisée dans le domaine du développement de logiciels et de la conception de nouveaux produits pour déterminer les fonctionnalités à développer.

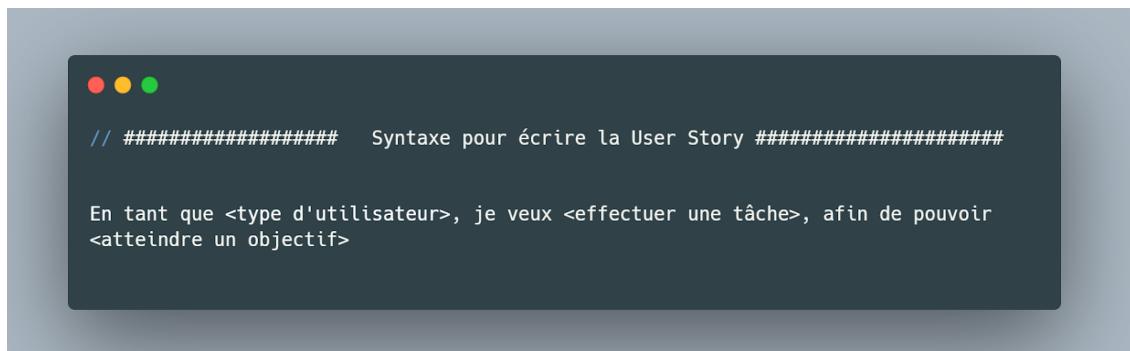


Figure 4.3 structure des dossiers

4.1.1.3 backlog

Backlog est une liste ordonnée et émergente de ce qui est nécessaire pour améliorer le produit.

C'est l'unique source du travail entrepris par la Scrum Team

The screenshot shows a digital project management board with the following sections:

- Project Resources:** 8 cards. Includes a card for "#18 Labels Documentation" featuring a neon-style "LABEL" sign.
- Product Backlog:** 11 cards (total 131). Includes cards like "#j'aimerai bien choisir le thème et la langue" (8 points), "#7 (21) création component de catégorie", "#8 (21) écran de création de la catégorie de la liste", "#9 (13) création de tableau des catégories", "#10 (34) creation restful API category", and "#11 (13) test restful API category".
- En cours / En attente:** 2 cards (total 55). Includes cards like "#34 En tant que Utilisateur authentifié, je veux créer un service" (55 points) and "#35 modifier l'algorithme de recherche".

At the bottom left, there are cards for "#10 Syntax to write the Story" and "#11 Scrum Workflow".

Figure 4.4 structure des dossiers

4.1.1.4 sprint

Ce sont des événements d'une durée fixe, pour créer une cohérence. Un nouveau Sprint commence immédiatement après la fin du précédent.

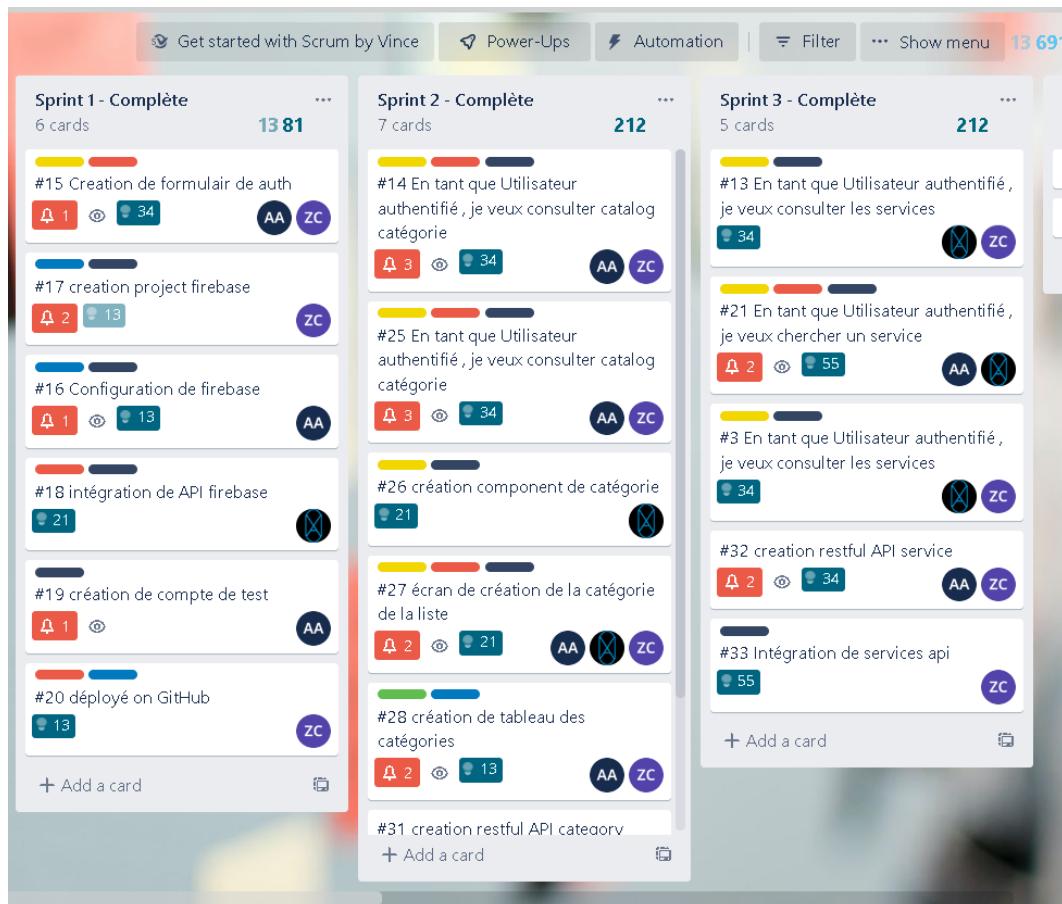


Figure 4.5 sprint

 DigitalOcean				
Month-to-date summary June 1 - 22, 2022				THIS IS NOT A FINALIZED INVOICE
For				
My Team <ananasanasi3030@gmail.com> SALMIA1 RUE2 N26 CD CASABLANCA CASABLANCA Casablanca 20200 MOROCCO				
Summary				
Total usage charges*				\$55.28
DigitalOcean Credit Applied - PaaS**				-\$25.15
DigitalOcean Credit Applied - IaaS**				-\$30.13
Current total*				\$0.00
*Your final total may vary based on your actual usage at the close of this billing period.				
**Credits and discounts are applied towards your final balance at the close of the billing period.				
<hr/>				
Product usage charges				
<i>Detailed usage information is available via the API or can be downloaded from the billing section of your account</i>				
<hr/>				
App Platform				\$25.15
<hr/>				
elmawkef (professional)		Hours	Start	End
backend (1x professional-m)		0	06-07 22:03	06-07 22:07
Primary Node - Development Database 0.5 GB / 1 vCPU / 1 GB Disk		0	06-07 21:56	06-07 22:01
<hr/>				
elmawkef (professional)		Hours	Start	End
backend (1x professional-m)		337	06-07 22:22	06-22 00:00
Primary Node - Development Database 0.5 GB / 1 vCPU / 1 GB Disk		0	06-07 22:17	06-07 22:28
<hr/>				
Database Clusters				\$30.13
<hr/>				
app-82fd2dec-c55e-4c60-b219-f23f6d43287b (PostgreSQL)		Hours	Start	End
Primary Node - Basic 4 GB / 2 vCPU / 38 GB Disk		337	06-07 22:28	06-22 00:00
<hr/>				
app-a63137fa-de53-488f-95fd-f6035ad2f7b8 (PostgreSQL)		Hours	Start	End
Primary Node - Basic 2 GB / 1 vCPU / 25 GB Disk		0	06-07 22:01	06-07 22:06
<hr/>				
Page 1 of 1				

Figure 4.6 facturation de server

The screenshot shows the Heroku Settings page for the 'backend' component of an application. The top navigation bar includes links for Overview, Insights, Activity, Runtime Logs, Console, and Settings, with Settings being the active tab. Below the navigation is a 'Components' section showing three items: App, backend (selected), and db. The main content area is titled 'Component Settings: backend'. It contains several configuration sections:

- Info**: Shows the component name as 'backend' with a question mark icon and 'Web Service'.
- Scaling**: Displays a plan of '\$20.00/mo – Basic | 2 GB RAM | 1 vCPU x 1' with an 'Edit' link.
- Source**: Shows the GitHub repository URL 'https://github.com/zakaria200dh/backend' with a fork icon, branch 'master', source directory '/', autodeploy status 'On', and a note about a 'GitHub authentication error'.
- Alert Policies**: Shows 0 alert policies enabled with an 'Edit' link.
- Environment Variables**: Shows 11 environment variables with an 'Edit' link.
- HTTP Request Routes**: Shows 1 route with an 'Edit' link.
- CORS Policy**: Shows 'Not Configured' with an 'Edit' link.
- Health Checks**: Shows 'TCP' with an 'Edit' link.
- Commands**: Shows build command 'yarn build' and run command 'yarn start' with an 'Edit' link.
- Log Forwarding**: Shows 0 destinations with an 'Edit' link.
- HTTP Port**: Shows port '1337' with an 'Edit' link.
- Destroy**: Contains a red button labeled 'Destroy Component' with a trash icon.

Figure 4.7 l'interface de configuration

App Settings		
Info	elmawkef 🇬🇧 London  elmawkef	Edit
Plan	Basic Plan \$20/mo	Edit
Alert Policies	0 Alert Policies enabled	Edit
Domains	elmawkef-6k84n.ondigitalocean.app + 1 other domain	Edit
App-Level Environment Variables	0 environment variables	Edit
BETA Features	Help us test and integrate new features before release	Edit
Log Forwarding	0 Destinations	Edit
App Spec	Your app topology is defined by the app spec and can be used to edit advanced settings. Learn more ↗	Edit
Destroy	Destroy app and associated components	<button>Destroy</button>

Figure 4.8 variable d'environnement de server

Users		
Username	Password	
db 	AVNS_n4TjJX3YQIAKYJ  	...
doadmin 	AVNS_irCTILid0R7BkjZ  	...
Username*		
<input type="text" value="Add new user"/>	<button>Save</button>	

Databases		
Name		
db 		...
defaultdb 		...
Database Name*		
<input type="text" value="Add new database"/>	<button>Save</button>	

Figure 4.9 base donne type de utilisateur

tout en configurant la connection à notre base de donnée pour que notre code accède

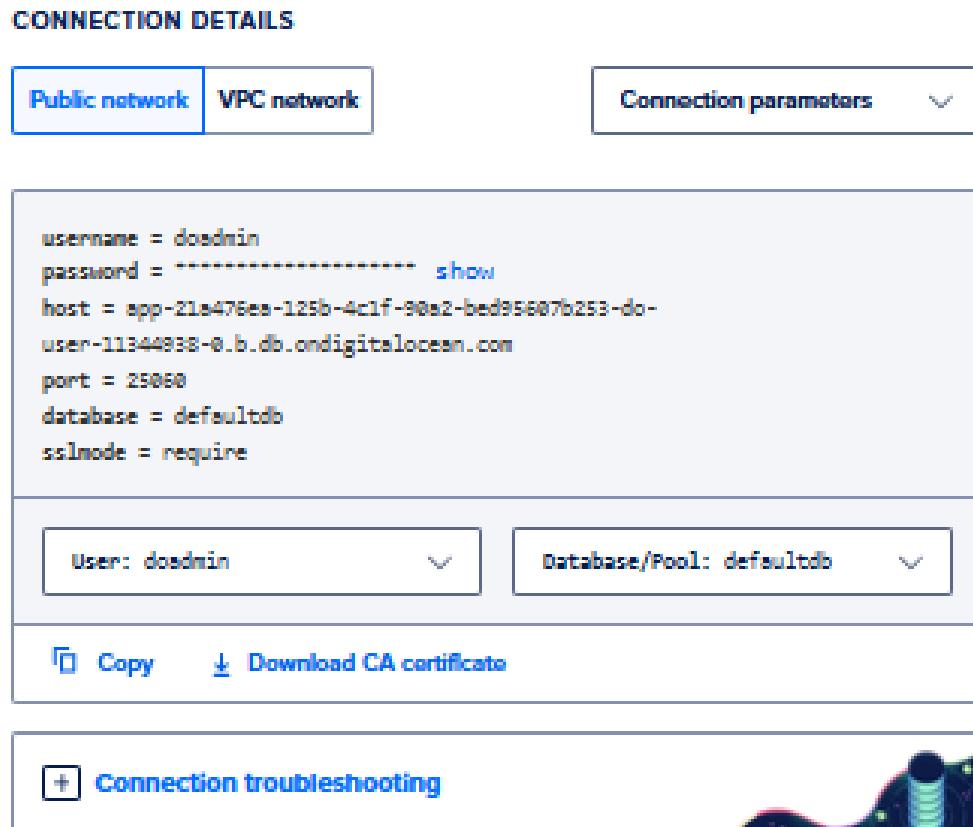


Figure 4.10 base donne détails de connexion

ensuite la structurer des fichiers fallait qu'elle soit optimal pour future intégration ou modification puisque le framework flutter n'offre pas ce genre d'option

```
.  
├── app.dart  
├── config  
│   └── README.md  
├── contracts  
│   └── README.md  
├── controllers  
│   └── README.md  
├── exceptions  
│   └── README.md  
├── models  
│   └── README.md  
├── router  
│   ├── README.md  
│   ├── pages.dart  
│   └── routers.dart  
├── services  
│   ├── README.md  
│   ├── locale  
│   └── remote  
├── themes  
│   ├── README.md  
│   ├── app_colors.dart  
│   ├── dark  
│   │   └── v1  
│   └── light  
│       └── v1  
├── translations  
│   └── README.md  
└── views  
    ├── README.md  
    ├── components  
    └── screens  
        ├── auth  
        ├── shared  
        └── splash_onboarding
```

Figure 4.11 Structure de Dossier

après chaque intégration on suivait l'état de notre serveur et les Runtimes log

Timeline

RECENT

May 19 2022

LIVE App Platform's deployment went live

- Trigger: App Platform deployed due to maintenance
02:01:04 PM • [?]

✓ zakaria200dh's deployment went live

- Trigger: zakaria200dh configured global app settings
02:00:40 PM • Rollback

✓ zakaria200dh's deployment went live

- Trigger: zakaria200dh configured backend settings
01:53:28 PM • [?]

✓ zakaria200dh's deployment went live

- Trigger: zakaria200dh configured db settings
01:52:53 PM • [?]

✓ zakaria200dh's deployment went live

- Trigger: zakaria200dh configured backend settings
01:44:33 PM • [?]

⚠ zakaria200dh's deployment failed during deploy phase

- Trigger: zakaria200dh created a new app
01:32:53 PM • [?]

>You created a new app called elmawkef — this is where the fun begins.

01:32:54 PM

Figure 4.12 Deploye line



Figure 4.13 Monitoring

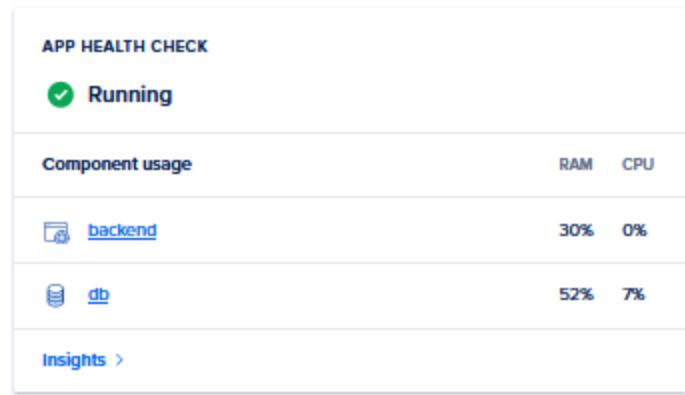


Figure 4.14 Server component

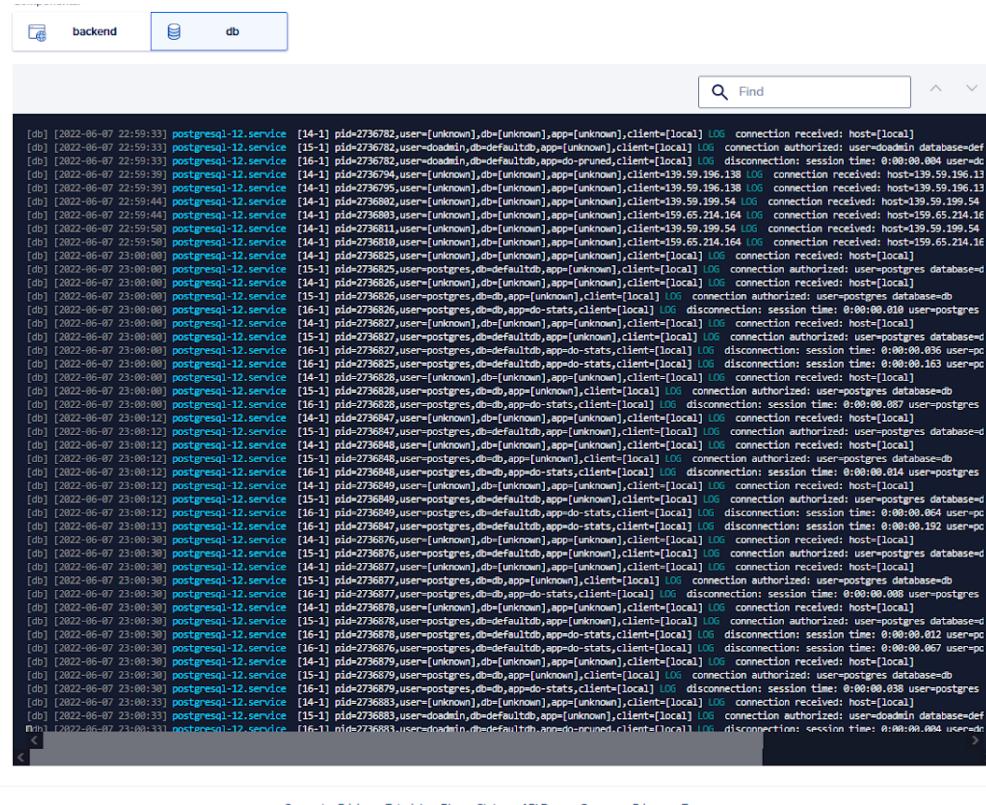


Figure 4.15 Log base donnée

4.2 Conduite du projet

4.2.1 Méthodologies de développement

DevOps

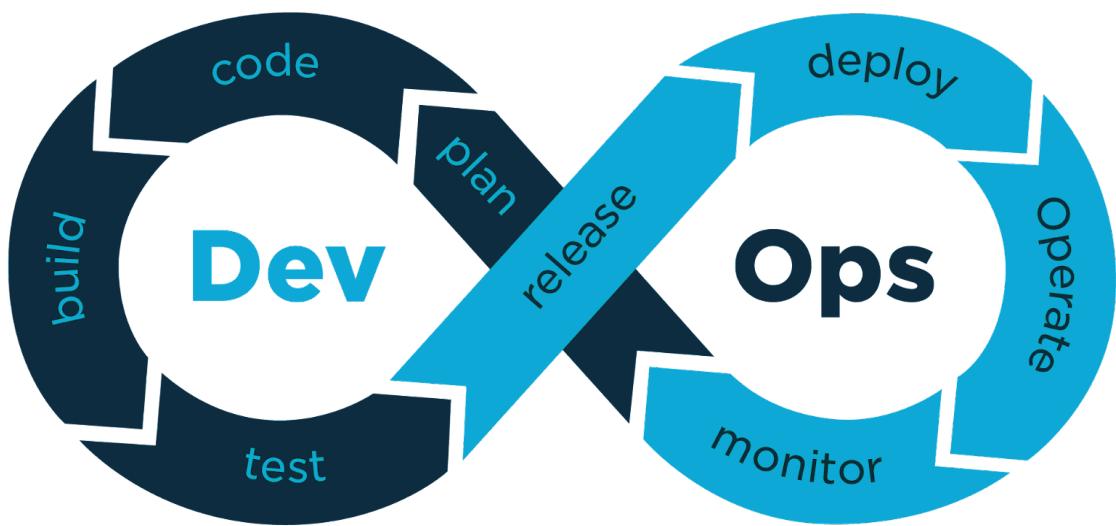


Figure 4.16 DevOps.

C'est la contraction de Développeur (Dev) et de Opérationnel (Ops). Ce terme est apparu en 2008 lors de la conférence de Shafter et s'est largement diffusé en 2013 grâce au livre "The Phoenix project". Historiquement les équipes de développeurs et les équipes opérationnelles ont des objectifs antagonistes, on parle du "wall of confusion". Les Devs doivent produire de nouvelles fonctionnalités et les Ops s'assurent de la fiabilité du site ou de l'application.

La méthodologie DevOps consiste donc à faire communiquer et collaborer les développeurs et les ops sur vos projets pour :

- gagner du temps sur la résolution des problèmes,

- lancer de nouvelles features plus rapidement
- tout en réduisant les risques grâce à l'automatisation des processus,
- et ainsi augmenter la satisfaction de vos clients.

L'intégration continue est une manière de réduire les risques et de lancer des nouvelles features plus rapidement. C'est un bon moyen de rendre les Devs et le Ops complémentaires, pour ainsi casser les silos entre eux.

L'intégration continue et le déploiement continu

L'intégration continue est un ensemble de pratiques qui consistent à tester automatiquement chaque modification du code avant la mise en production. Autrement dit, lorsqu'un développeur révise son code, il est testé de manière automatique. Si une erreur est détectée, il est notifié et s'il n'y a pas d'erreur, le code est mis en production automatiquement. L'étape de la mise en production automatique est couramment appelée le déploiement continu.

Le but de l'intégration continue est simple : garantir un code de qualité rapidement. Les équipes de développement ont un retour immédiat, plus besoin d'attendre plusieurs semaines pour identifier les erreurs et les corriger. Ce qui impacte positivement la satisfaction des utilisateurs finaux, qui voient de nouvelles features sortir régulièrement, tout en conservant la même fluidité de navigation.

- **l'intégration continue va se faire en 5 étapes:**

- 1. Planifiez votre développement:**

Afin de savoir quoi développer, il est nécessaire d'avoir à disposition un outil permettant la collaboration entre les développeurs. Cet outil permettra notamment de gérer les différentes releases et toutes les fonctionnalités, de garantir la priorité du backlog, etc

Intervenant tout au long du projet, la collaboration de toute l'équipe est nécessaire pour assurer la planification du projet. Cette planification est étroitement liée à la méthodologie Scrum. Elle a pour but de découper le projet en petites tâches à réaliser par toute l'équipe.

Sur notre chaîne d'assemblage, la planification représente la répartition des tâches de chaque collaborateur, ou encore le brief du matin destiné à définir les tâches à faire durant la journée.

2. Compilez et intégrez votre code:

Le contrôle de code source

Le code source se doit d'être disponible à chaque instant sur un dépôt central. Chaque développement doit faire l'objet d'un suivi de révision. Le code doit être compilable à partir d'une récupération fraîche, et ne faire l'objet d'aucune dépendance externe. Même s'il existe des notions de branche, la création d'une branche doit être évitée le plus possible, privilégiant le développement sur la branche principale ; cela évite de maintenir plusieurs versions en parallèle. Ce genre de pratique est appelé trunk-based development.

L'orchestrateur

Ensuite, toutes les étapes doivent être automatisées par un orchestrateur, qui saura reproduire ces étapes et gérer les dépendances entre elles. De plus, l'utilisation d'un orchestrateur permet de donner accès à tous, et à tout moment, à un tableau de bord qui donnera l'état de santé des étapes d'intégration continue. Ainsi, les développeurs ont au plus tôt la boucle de feedback nécessaire, afin de garantir que l'application soit prête à tout moment. De plus, l'orchestrateur permettra d'aller plus loin dans la livraison continue, comme on le verra dans la suite du cours.

La première étape, et celle qui paraît la plus évidente, est de compiler le code de manière continue. En effet, sans cette étape, le code est compilé manuellement sur le poste du développeur, afin que ce dernier s'assure que son code compile.

La mise en place d'une première étape de compilation dans un processus d'intégration continue permet justement de ne plus se soucier si des modifications de code cassent la compilation. Le développeur doit alors s'assurer de bien envoyer son code source sur le dépôt central. En faisant cela, il déclenche une première étape de compilation, avec

toutes les modifications des autres développeurs. Si la compilation ne se fait pas, le code est alors rejeté, et le développeur doit corriger ses erreurs.

Après cette première étape, le code devient plus sûr, et le dépôt de code source garantit qu'à chaque instant, un développeur récupère un code qui compile. Dans cette étape, les tests ne sont pas encore exécutés. Le code peut donc être de mauvaise qualité.

3. Testez votre code:

Les tests unitaires

Dans cette étape, l'orchestrateur se charge de lancer les tests unitaires à la suite de la compilation. Ces tests unitaires, généralement avec un framework associé, garantissent que le code respecte un certain niveau de qualité.

Plus il y a d'unitaires, plus le code est garanti sûr. Évidemment, l'orchestrateur ne peut lancer que les tests qui ont été codés par les développeurs, et ne peut pas inventer de nouveaux cas de tests.

Ces tests doivent s'exécuter de la manière la plus rapide possible, afin d'avoir un feedback le plus rapide lui aussi. Pour arriver à ce niveau, il est nécessaire que les tests unitaires n'aient aucune dépendance vis-à-vis de systèmes externes, comme par exemple une base de données, ou même le système de fichiers de la machine.

Les tests unitaires apportent 3 atouts à la production :

- trouver les erreurs plus facilement. Les tests sont exécutés durant tout le développement, permettant de visualiser si le code fraîchement écrit correspond au besoin
- sécuriser la maintenance. Lors d'une modification d'un programme, les tests unitaires signalent les éventuelles régressions. En effet, certains tests peuvent échouer à la suite d'une modification, il faut donc soit réécrire le test pour le faire correspondre aux nouvelles attentes, soit corriger l'erreur se situant dans le code ;

- documenter le code. Les tests unitaires peuvent servir de complément à la documentation ; il est très utile de lire les tests pour comprendre comment s'utilise une méthode. De plus, il est possible que la documentation ne soit plus à jour, mais les tests, eux, correspondent à la réalité de l'application.

L'ensemble des tests unitaires doivent être relancés après une modification du code, afin de vérifier qu'il n'y ait pas de régressions (l'apparition de nouveaux dysfonctionnements).

La multiplicité des tests unitaires oblige à les maintenir dans le temps, au fur et à mesure que le développement avance.

4. Mesurez la qualité de votre code:

Maintenant que les tests unitaires sont écrits et exécutés, nous commençons à avoir une meilleure qualité de code, et à être rassurés sur la fiabilité et la robustesse de l'application. Grâce à la compilation et aux tests unitaires, nous pouvons maintenant mesurer la qualité du code. Tout ceci permet aux développeurs de maintenir dans le temps un code de très bonne qualité, alertant l'équipe en cas de dérive des bonnes pratiques de tests.

Lors de l'étape de qualité de code, nous cherchons à assurer la plus petite dette technique possible de notre application. La dette technique est le temps nécessaire à la correction de bugs ou à l'ajout de nouvelles fonctionnalités, lorsque nous ne respectons pas les règles de coding. La dette est exprimée en heures de correction. Plus cette dette est élevée, plus le code sera difficile à maintenir et à faire évoluer.

L'étape de qualité de code mesure aussi d'autres métriques, comme le nombre de vulnérabilités au sein du code, la couverture de test, mais aussi les code smells (qui sont des mauvaises pratiques à ne pas implémenter), la complexité cyclomatique (complexité du code applicatif) ou la duplication de code. C'est le rôle du développeur de respecter les normes définies et de corriger au fur et à mesure son code.

Afin de renforcer la qualité du code et de ne pas autoriser le déploiement d'un code de mauvaise qualité, nous pouvons implémenter un arrêt complet du pipeline d'intégration continue, si le code n'atteint pas la qualité requise.

5. Gérez les livrables de votre application:

Le code, une fois compilé, doit être déployé dans un dépôt de livrables, et versionné. Les binaires produits sont appelés artefacts. Ces artefacts doivent être accessibles à toutes les parties prenantes de l'application, afin de pouvoir les déployer et lancer les tests autres qu'unitaires (test de performance, test de bout en bout, etc.). Ces artefacts sont disponibles dans un stockage, centralisé et organisé, de données. Ce peut être une ou plusieurs bases de données où les artefacts sont localisés en vue de leur distribution sur le réseau, ou bien un endroit directement accessible aux utilisateurs.

Le Méthode DevOps

Comme vu précédemment, la culture DevOps repose sur l'idée de casser les silos entre les Devs et les Ops. Il n'est donc plus question de chercher un coupable, de se renvoyer la faute, et de mettre 24h à réparer un bug qui pourrait l'être en 10 minutes.

Laisser les ops sur le banc de touche n'est pas envisageable dans la méthodologie DevOps car la communication est le nerf de la guerre. Pour aligner les Devs et les Ops il faut communiquer et cela passe par cinq actions :

1. Créer une équipe fonctionnelle pour tous vos projets. Historiquement l'équipe Produit est constituée d'un product owner, d'un ou plusieurs développeur(s) et d'un UX. L'équipe Fonctionnelle réunit l'équipe produit et les Ops sur un même plateau, autour de boards et d'objectifs communs.
2. Dédier du temps Ops au soutien des équipes de Devs. Il faut que chacun comprenne bien les priorités des autres et puisse consacrer du temps à l'imprévu.
3. En finir avec les demandes "informelles". Demander à un collègue de lui donner un coup de main autour de la machine à café c'est bien, mais il ne faut pas oublier d'en faire un ticket pour évaluer la charge de travail réelle de chacun.
4. Impliquer les Ops dans les événements des Devs. Si les projets, les boards et les objectifs sont communs, ça signifie que les Ops doivent être inclus dans toutes les discussions, réunions, etc.
5. Définir des objectifs communs et célébrer les succès ensemble. Cela permet d'amener une meilleure implication au sein de l'équipe.

Adaptation avec méthodologie agile

Le principal levier de la méthodologie agile est de raccourcir la durée des cycles de développement et donc la fréquence entre les différentes mises en production.

Pour rappel, la culture DevOps consiste à lisser les relations entre les équipes de développeurs et les ops afin de faire des mises en production plus rapides et de meilleure qualité.

À ce stade, les deux méthodologies semblent se rejoindre puisqu'elles ont un but commun. Pourtant il y a bien une divergence. La méthodologie agile est par définition agile, flexible. Elle permet d'adapter les priorités des développements à des imprévus ou des changements de direction. Or, dans la méthodologie DevOps les imprévus ne sont pas faciles à faire rentrer dans le schéma traditionnel de priorisation des tâches.

Il suffit de bien prendre en compte les problématiques liées aux métiers de Dev et d'Ops. Cela passe par trois actions :

Formuler les demandes sous forme de tickets. Ainsi les charges de travail sont bien établies. Ne pas oublier de créer un ticket pour laisser du temps aux imprévus, chez Padok nous appelons ça le buffer.

Participer aux événements des uns et des autres. Eh oui, la communication reste encore et toujours la clé ! Pour mettre en place une démarche d'amélioration continue il faut prévoir des discussions où chacun expose ses points bloquants pour avancer sur le projet.

Sensibiliser les équipes de Dev et Ops. Chaque équipe doit connaître l'impact du travail de son équipe sur le business et les utilisateurs. Si l'on reprend le premier exemple du t-shirt et des soldes, un site e-commerce peut perdre jusqu'à 30 000 euros sur une matinée de soldes si le site ne répond plus. En ayant cette information, les Devs valoriseront mieux le travail des équipes opérationnelles.

Pour aligner vos Devs et vos Ops et adapter la méthodologie agile, la communication est primordiale, mais vous aurez aussi besoin d'outils pour tout mettre en place.

Les outils DevOps

Le DevOps est donc une culture de collaboration et d'automatisation des processus entre les équipes de développement et les opérationnels. Pour collaborer et automatiser vous allez avoir besoin d'outils et il en existe un très grand nombre. Padok vous donne sa sélection pour vous aider à y voir plus clair.

Les outils de gestion de code source font partie des fondamentaux du développement mais il est important de les étendre et de les partager avec les opérationnels pour une meilleure communication. Cela permet de connaître les différentes modifications du code et les auteurs de celles-ci. Git et Github permettent notamment de créer un historique pour les fichiers code source.

Il vous faudra ensuite un outil de tests d'intégration continue et de déploiement continu (CI/CD). Comme expliqué plus haut, ces outils automatisent les tests des révisions du code source et lancent automatiquement le déploiement d'environnement de tests, voire de production. Chez Padok, nous utilisons GitlabCI mais il en existe un grand nombre, comme par exemple Jenkins (open source), Google, AWS, etc.

Pour éviter les mauvaises surprises au moment de la mise en production, nous vous conseillons d'utiliser des conteneurs. Vous pourrez isoler une application avec l'ensemble des éléments dont elle a besoin pour fonctionner. Les deux principaux outils sont Docker et RKT. Afin de simplifier le déploiement et la gestion des conteneurs nous vous conseillons d'utiliser un orchestrateur de conteneurs tel que Kubernetes.

Pour montrer les clusters Kubernetes, Prometheus (outil open source) et Grafana sont efficaces. Et pour les analyses de logs et d'agrégation, la suite ELK (Elasticsearch, Logstash et Kibana) est répandue.

Il vous faut impérativement un outil de gestion de projet soit de type agile comme Jira ou soit généraliste tel que Trello.

Il n'existe pas d'outils parfaits en DevOps, tout dépend de vos projets. Le mieux est de les tester pour voir s'ils répondent à vos besoins. D'autant plus que tous les outils cités plus haut sont facilement disponibles et intégrables dans le cloud à des fins de tests et de production.

Le DevOps et le Cloud

On entend de plus en plus parler de Cloud mais qu'est-ce que ça change pour le DevOps et qui sont les Cloud providers du marché ? Les Cloud providers sont des solutions d'hébergement distant, c'est-à-dire que les données et les outils sont stockés dans des data centers en dehors de l'entreprise.

Le Cloud et le DevOps sont indépendants mais sont des stratégies complémentaires car pour rester compétitif sur le marché, les entreprises doivent revoir leur approche du travail.

Les cycles de production doivent être réduits pour apporter des améliorations constantes auprès des utilisateurs finaux. Le Cloud répond directement à ce besoin d'agilité.

l'Ops peut se concentrer sur la mise en production des nouvelles fonctionnalités au lieu de penser à la coupure de serveurs physiques. En effet, les problématiques de load balancing et de redondance par exemple, sont notamment prises en compte dans ce contexte.

Il existe trois principaux Cloud providers sur le marché :Google Cloud Platform (GCP), Azure et AWS. Ils proposent tous les outils mentionnés précédemment et bien d'autres.

4.3 Cycle de vie

Le cycle de vie d'un logiciel désigne toutes les étapes du développement du projet , de sa conception à sa disparition. L'objectif d'un tel découpage est de permettre de définir des jalons intermédiaires permettant la validation du développement, c'est-à-dire la conformité du produit avec les besoins exprimés, et la vérification du processus de développement, c'est-à-dire l'adéquation des méthodes mises en œuvre.

DevOps impose différentes étapes à suivre tout au long de ses projets:

Ces étapes sont à suivre dans un ordre bien défini et se répètent continuellement afin de suivre les évolutions du projet dans le temps.

Dans cette section nous détaillons chaque étape une à une.

4.3.1 Planification

La planification permet de définir la valeur commerciale et les exigences attendues avant de lancer les développements.

Cette phase permet ainsi d'identifier et de suivre les travaux dans le temps. L'idée est d'avoir une vision claire des actions à réaliser pour atteindre les objectifs business fixés.

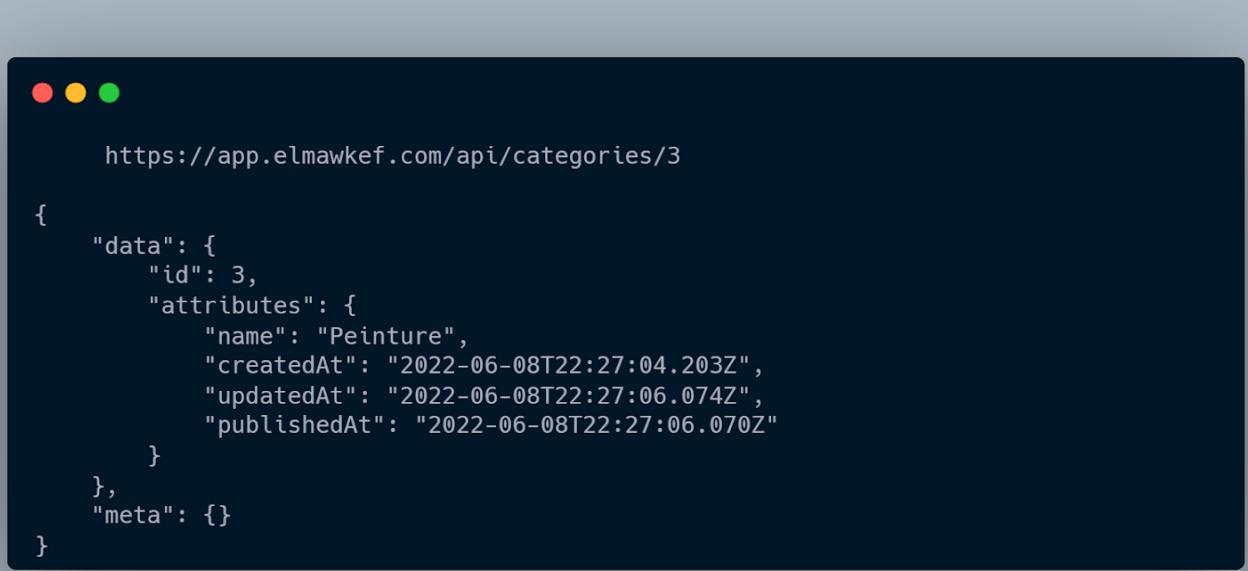
Les entreprises appliquant cette phase s'appuient fréquemment sur des pratiques Agiles. Des outils comme Jira, Trello ou ClickUp sont de bons supports pour assurer la sérénité de la phase dans le temps

4.3.2 Création

La création inclut la conception logicielle et la création de code. En d'autres mots, c'est la phase de développement du produit.

Durant cette phase les développeurs conçoivent le produit en équipe et s'appuient sur des outils de gestion de code tels que GitHub, GitLab ou Bitbucket. Des good practices de gestion de code existent comme les principes de GitFlow.

Ces good practices assurent la pérennité du code dans le temps et facilitent le travail en équipe.



```
https://app.elmawkef.com/api/categories/3

{
  "data": {
    "id": 3,
    "attributes": {
      "name": "Peinture",
      "createdAt": "2022-06-08T22:27:04.203Z",
      "updatedAt": "2022-06-08T22:27:06.074Z",
      "publishedAt": "2022-06-08T22:27:06.070Z"
    }
  },
  "meta": {}
}
```

Figure 4.17 json catégorie

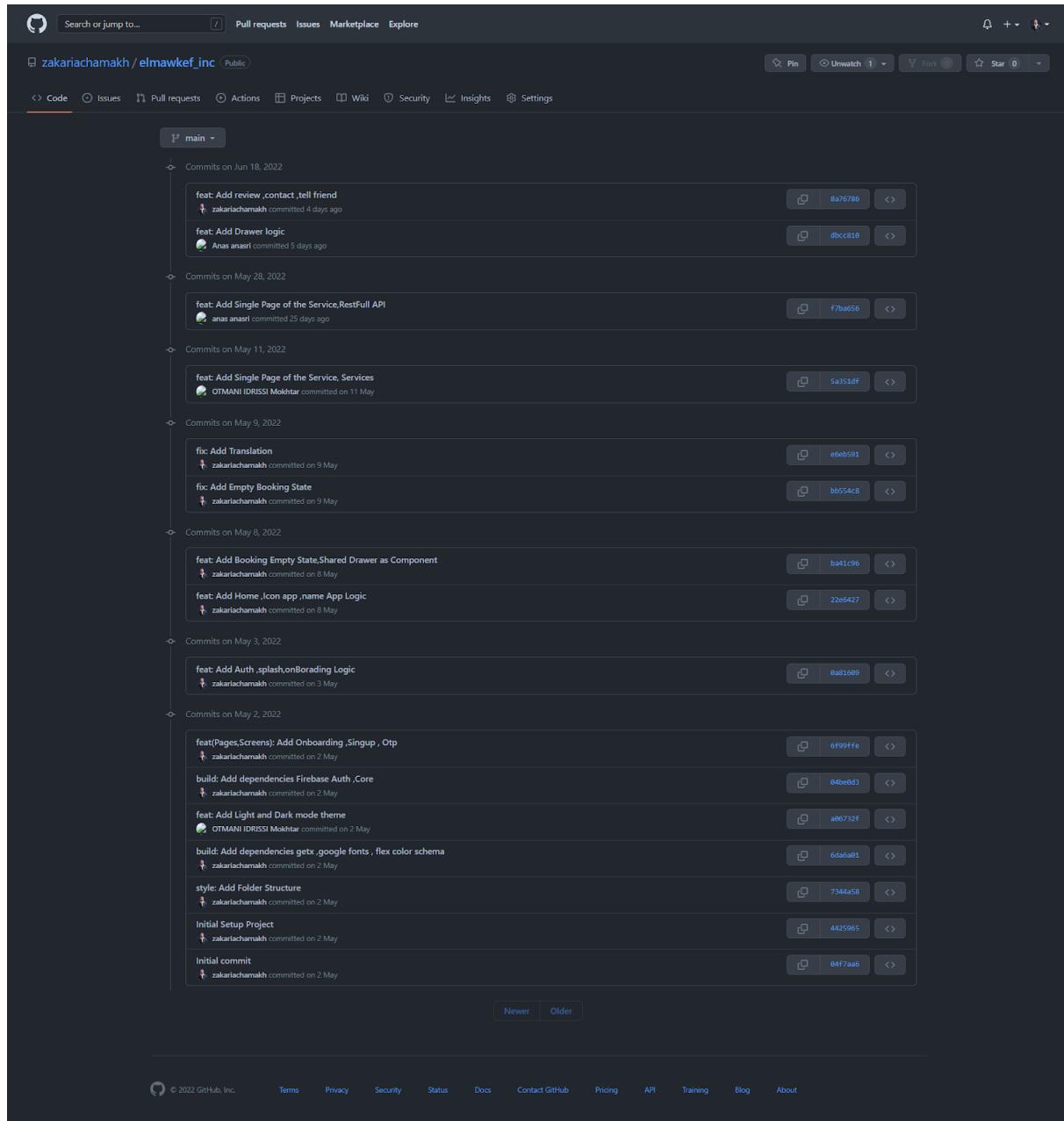


Figure 4.18 Github branche

4.3.3 Test

La phase de test consiste à vérifier le bon fonctionnement et faire la recette des développements réalisés durant l'étape de création.

Il existe différents types de test:

- Les tests unitaires : Ces tests vérifient le bon fonctionnement d'une partie précise d'un logiciel ou d'une portion d'un programme
- Les tests d'intégration : Ces tests vérifient que chacun des modules indépendants du logiciel est assemblé et fonctionne bien dans l'ensemble.
- Les tests de validation : Ces tests vérifient si toutes les exigences client, décrites dans le document de spécification du logiciel sont respectées.

Dans la culture DevOps, il est fortement recommandé d'automatiser ces tests au maximum avec des outils tels que Selenium, Karaté...

Cette phase souvent négligée est le socle d'une production de qualité. Des méthodologies de développement comme le TDD permettent d'assurer cette phase.

Réaliser des tests est une façon de détecter rapidement les bogues et d'éviter tout impact business non désiré.

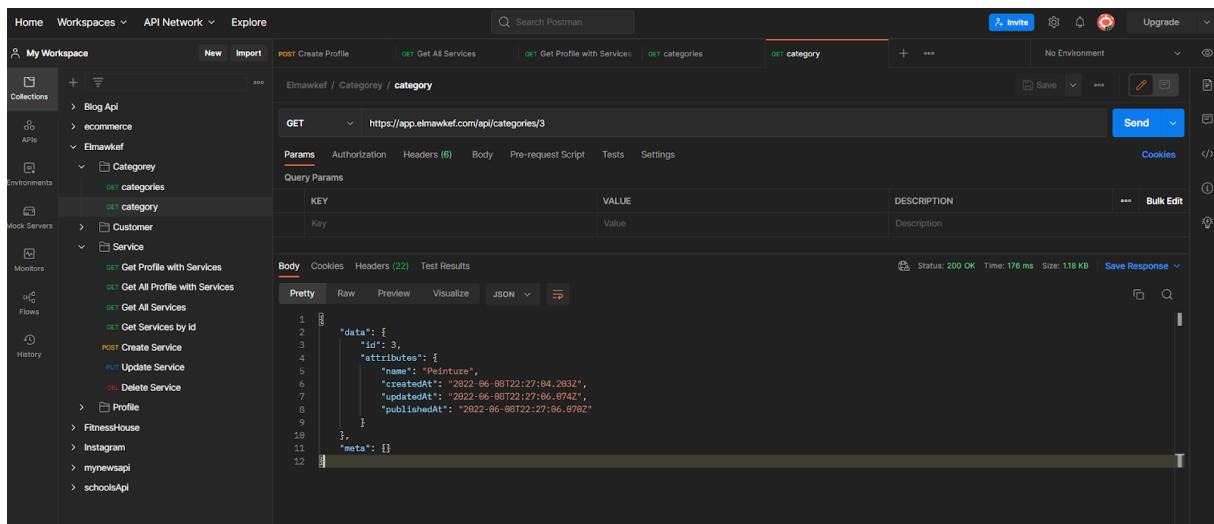


Figure 4.19 Postman

4.3.4 Versionnage

Le versionnage aussi appelé distribution continu consiste à publier le code validé dans un référentiel de façon continue.

Cette phase permet ainsi de gérer les versions logicielles et à exploiter des outils automatisés pour compiler et intégrer le code en vue de sa mise en production.

Des référentiels de code source ou de package « empaquettent » aussi l'infrastructure requise pour la livraison du produit à l'aide des logiciels Docker, Ansible, Puppet, Chef, Gradle, Maven ou JFrog Artifactory, par exemple.

Aussi, pour garantir l'efficacité du processus de distribution continue, il faut d'abord introduire le processus d'intégration continue (Création + Test) dans le pipeline de développement. La distribution continue permet de disposer d'une base de code toujours prête à être déployée dans un environnement de production.

Dans le cadre du versionnage, chaque étape (de la fusion des modifications de code jusqu'à la distribution des versions prêtes pour la production) implique l'automatisation des processus de test et de publication du code. À la fin de ce processus, l'équipe d'exploitation est en mesure de déployer facilement et rapidement une application dans un environnement de production.

4.3.5 Déploiement

Le déploiement aussi appelé déploiement continu, en complément du processus de distribution continue, automatise la publication d'une version prête pour la production dans un référentiel de code.

Le déploiement continu automatisé le lancement d'une application dans un environnement de production.

En l'absence de passerelle manuelle entre la production et l'étape précédente du pipeline, le déploiement continu dépend surtout de la conception de l'automatisation des processus de test.

Dans le cadre du déploiement continu, une modification apportée par un développeur à une application peut être publiée quelques minutes seulement après la rédaction du code en question (en supposant qu'elle passe les tests automatisés). Il est ainsi beaucoup plus facile de recevoir et d'intégrer en continu les retours des utilisateurs.

Combiné avec le test et le versionnage, le déploiement continu réduit les risques liés au déploiement des applications, puisqu'il est plus simple de publier des modifications par petites touches qu'en un seul bloc.

Cette approche nécessite néanmoins un investissement de départ considérable, car les tests automatisés devront être rédigés de manière à s'adapter à un large éventail d'étapes de test et de lancement dans le pipeline CI/CD.

Des outils de planification et d'automatisation existent pour réaliser cette phase comme: Ansible, Puppet, Chef, Keltio, Jenkins, Kubernetes, OpenShift, OpenStack, Docker, par exemple.

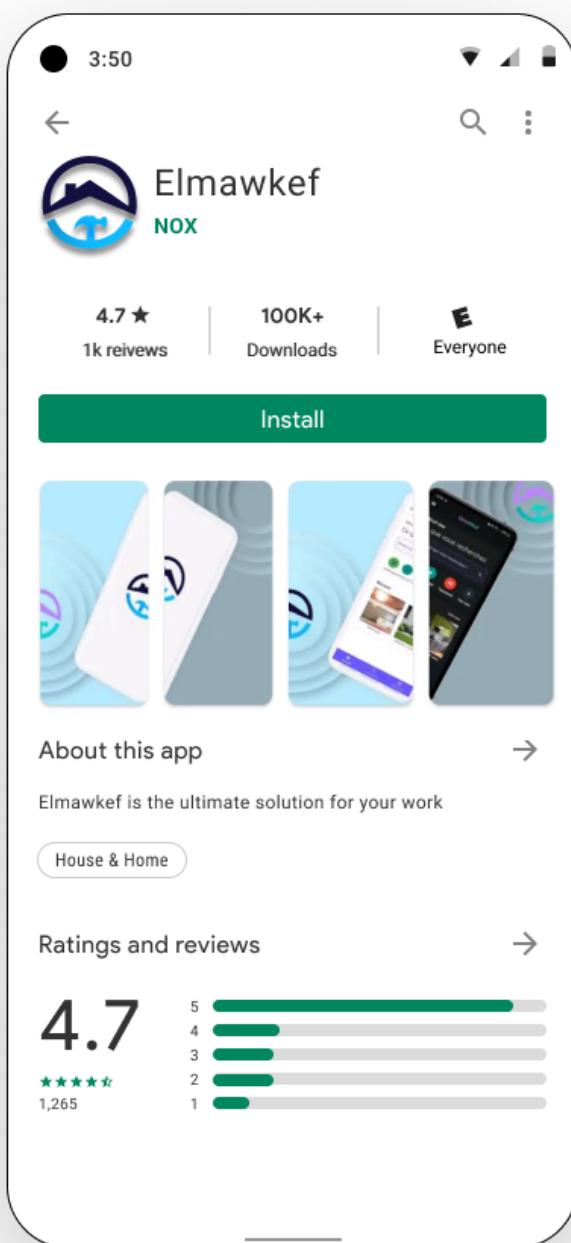


Figure 4.20 Play Stor :<https://play.google.com/store/apps/details?id=com.elmawkef.pfe>

4.3.6 Exploitation

L'exploitation intervient une fois le logiciel déployé. La phase d'exploitation implique la maintenance et le dépannage des applications dans les environnements de production.

En adoptant les pratiques DevOps, les équipes veillent à assurer la fiabilité, la haute disponibilité du système, et visent à éliminer les temps d'arrêt tout en renforçant la sécurité et la gouvernance. Les équipes DevOps entendent identifier les problèmes avant qu'ils n'affectent l'expérience client, et à les atténuer rapidement lorsqu'ils se surviennent.

Des solutions comme Ansible, Puppet, PowerShell, Chef, Keltio, existent pour opérer sur les produits de façon automatisée.

4.3.7 Supervision

Les produits ont, vis-à-vis de leurs utilisateurs, un SLA (Service Level Agreement), de ce fait, ils doivent être supervisés continuellement afin de détecter tout problème. C'est la dernière phase du cycle DevOps.

Elle consiste à moniter l'activité des produits et de définir des actions à réaliser par les équipes support ou produit (incident, feedback...)

Maintenir cette vigilance nécessite une riche télémétrie, des alertes exploitables et une visibilité totale sur les applications et le système sous-jacent.

Des solutions de monitoring et de supervision comme Grafana, Prometheus, Elasticsearch peuvent être utilisées dans cette phase.

5 Réalisation

5.1 Introduction

A ce stade du processus, les cas d'utilisation sont terminés. Le problème a été analysé en profondeur ; nous avons défini une conception mieux appropriée aux besoins de l'application.

Ce chapitre est consacré la réalisation et la mise en œuvre de notre application, nous allons présenter les outils de développement adoptés, soit l'environnement utilisé , ainsi que les langages de programmation (dart, javascript ,html ,css ,nodejs), et nous allons présenter le système de gestion de base de données postgresql ,ainsi le langage de manipulation de bases de données SQL , et enfin nous allons montrer les interfaces principaux et fenêtres de l'application.

5.2 Outils de développement

5.2.1 Android Studio



Figure 5.1 Android Studio.

Android Studio est un environnement de développement pour développer des applications mobiles Android. Il est basé sur IntelliJ IDEA et utilise le moteur de production Gradle. Il peut être téléchargé sous les systèmes d'exploitation Windows, macOS, Chrome OS et Linux.

Android Studio permet principalement d'éditer les fichiers Java/Kotlin et les fichiers de configuration XML d'une application Android.

Il propose entre autres des outils pour gérer le développement d'applications multilingues et permet de visualiser rapidement la mise en page des écrans sur des écrans de résolutions variées simultanément. Il intègre par ailleurs un émulateur permettant de faire tourner un système Android virtuel sur un ordinateur.

5.2.2 IntelliJ IDEA



Figure 5.2 IntelliJ IDEA.

IntelliJ IDEA également appelé « IntelliJ », « IDEA » ou « IDJ » est un environnement de développement intégré (en anglais Integrated Development Environment - IDE) destiné au développement de logiciels informatiques reposant sur la technologie Java. Il est développé par JetBrains (anciennement « IntelliJ ») et disponible en deux versions, l'une communautaire, open source, sous licence Apache 2 et l'autre propriétaire, protégée par une licence commerciale. Tous deux supportent les langages de programmation Java, Kotlin, Groovy et Scala.

5.2.3 Visual Studio Code



Figure 5.3 Visual Studio Code.

Visual Studio Code est un éditeur de code extensible développé par Microsoft pour Windows, Linux et macOS. Les fonctionnalités incluent la prise en charge du débogage, la mise en évidence de la syntaxe, la complétion intelligente du code, les snippets, la refactorisation du code et Git intégré. Les utilisateurs peuvent modifier le thème, les raccourcis clavier, les préférences et installer des extensions qui ajoutent des fonctionnalités supplémentaires. Le code source de Visual Studio Code provient du projet logiciel libre et open source VSCode de Microsoft publié sous la licence MIT permissive, mais les binaires compilés constituent un freeware, c'est-à-dire un logiciel gratuit pour toute utilisation mais privé. Dans le sondage auprès des développeurs réalisé par Stack Overflow en 2021, Visual Studio Code a été classé comme l'outil d'environnement de développement le plus populaire, avec 71,06 % des 82 277 répondants déclarant l'utiliser.

5.2.4 Postman



Figure 5.4 Postman.

Postman est une application permettant de tester des API, créée en 2012 par Abhinav Asthana, Ankit Sobti et Abhijit Kane2 à Bangalore pour répondre à une problématique de test d'API partageable. D'abord module complémentaire de Google Chrome, puis client lourd, et finalement client léger, elle est à présent utilisée par plus de 500 000 entreprises dans le monde et a son siège à San Francisco

5.2.5 DigitalOcean



Figure 5.5 DigitalOcean.

DigitalOcean, Inc. est un fournisseur américain d'infrastructure cloud dont le siège est à New York et qui possède des centres de données dans le monde entier. DigitalOcean fournit aux développeurs, aux startups et aux PME des plates-formes d'infrastructure cloud en tant que service.

5.2.6 NameCheap



Figure 5.6 NameCheap.

Namecheap est un bureau d'enregistrement de noms de domaine accrédité par l'ICANN fournissant l'enregistrement de noms de domaine et l'hébergement Web basé à Phoenix,

Arizona, États-Unis. Namecheap est un fournisseur d'hébergement économique avec 11 millions d'utilisateurs enregistrés et 10 millions de domaines

5.2.7 Github



Figure 5.7 Github.

Github est une entreprise de développement et services logiciels sise aux États-Unis. Github développe notamment la plateforme Github, l'éditeur de texte Atom ou encore la structure Electron . Le 4 juin 2018, Microsoft annonce l'acquisition de l'entreprise pour la somme de 7,5 milliards de dollars américains GitHub est un site web et un service de cloud qui aide les développeurs à stocker et à gérer leur code, ainsi qu'à suivre et contrôler les modifications qui lui sont apportées

5.2.8 Figma

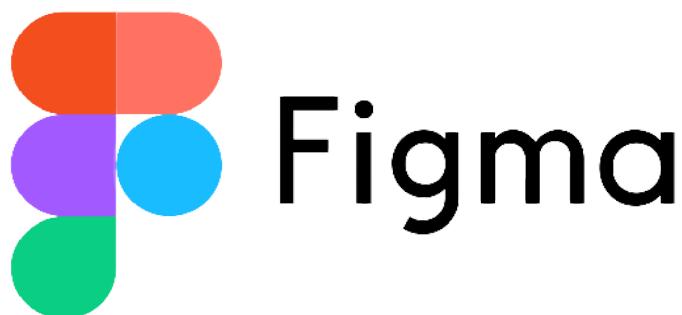


Figure 5.8 Figma.

Le Figma est un éditeur de graphiques vectoriels et un outil de prototypage. Il est principalement basé sur le web, avec des fonctionnalités hors ligne supplémentaires

activées par des applications de bureau pour macOS et Windows. Les Figma Mirror companion apps pour Android et iOS permettent de visualiser des prototypes Figma sur des appareils mobiles. L'ensemble des fonctionnalités de Figma est axé sur l'utilisation dans la conception de l'interface utilisateur et de l'expérience utilisateur, en mettant l'accent sur la collaboration en temps réel.

5.2.9 Trello



Figure 5.9 Trello.

Trello est un outil de gestion de projet en ligne, lancé en septembre 2011 et inspiré par la méthode Kanban de Toyota. Il repose sur une organisation des projets en planches listant des cartes, chacune représentant des tâches.

5.2.10 Javascript

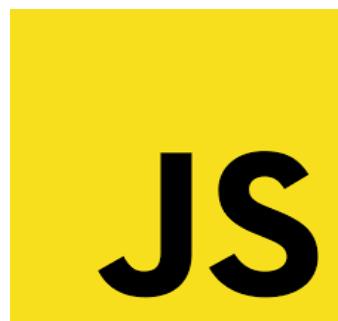


Figure 5.10 Javascript.

JavaScript est un langage de programmation de scripts principalement employé dans les pages web interactives et à ce titre est une partie essentielle des applications web. Avec les langages HTML et CSS, JavaScript est au cœur des langages utilisés par les développeurs web.

5.2.11 NodeJs



Figure 5.11 NodeJs.

Node.js est une plateforme logicielle libre en JavaScript, orientée vers les applications réseau évènementielles hautement concurrentes qui doivent pouvoir monter en charge. Elle utilise la machine virtuelle V8, la librairie libuv pour sa boucle d'évènements, et implémente sous licence MIT les spécifications CommonJS

5.2.12 Plantuml



Figure 5.12 Plantuml.

PlantUML est un outil open source permettant aux utilisateurs de créer des diagrammes à partir d'un langage de texte brut. Outre divers diagrammes UML, PlantUML prend en charge divers autres formats liés au développement de logiciels, ainsi que la visualisation des fichiers JSON et YAML

5.2.13 Postgresql



Figure 5.13 Postgresql.

PostgreSQL est un système de gestion de base de données relationnelle et objet. C'est un outil libre disponible selon les termes d'une licence de type BSD. Ce système est comparable à d'autres systèmes de gestion de base de données, qu'ils soient libres, ou propriétaires.

5.2.14 Agile/Scrum



Figure 5.14 Agile/Scrum.

En ingénierie logicielle, les pratiques agiles mettent en avant la collaboration entre des équipes auto-organisées et pluridisciplinaires et leurs clients. Elles s'appuient sur l'utilisation d'un cadre méthodologique léger mais suffisant centré sur l'humain et la communication

Afin de savoir quoi développer, il est nécessaire d'avoir à disposition un outil permettant la collaboration entre les développeurs. Cet outil permettra notamment de gérer les différentes releases et toutes les fonctionnalités, de garantir la priorité du backlog, etc.

Intervenant tout au long du projet, la collaboration de toute l'équipe est nécessaire pour assurer la planification du projet. Cette planification est étroitement liée à la méthodologie Scrum. Elle a pour but de découper le projet en petites tâches à réaliser par toute l'équipe.

Sur notre chaîne d'assemblage, la planification représente la répartition des tâches de chaque collaborateur, ou encore le brief du matin destiné à définir les tâches à faire durant la journée.

5.2.15 Strapi



Figure 5.15 Strapi.

Strapi est un CMS headless c'est-à-dire qu'il ne permet de gérer que la partie back-end d'un projet. Pour la partie front-end, vous pouvez relier un framework front-end en utilisant une API.

5.2.16 Html



Figure 5.16 Html.

Le HyperText Markup Language, généralement abrégé HTML ou, dans sa dernière version, HTML5, est le langage de balisage conçu pour représenter les pages web. Ce langage permet : d'écrire de l'hypertexte, d'où son nom, de structurer sémantiquement la page, de mettre en forme le contenu,

5.2.17 Css**Figure 5.17** Css.

Les feuilles de style en cascade, généralement appelées CSS de l'anglais Cascading Style Sheets, forment un langage informatique qui décrit la présentation des documents HTML et XML. Les standards définissant CSS sont publiés par le World Wide Web Consortium.

5.2.18 Flutter**Figure 5.18** Flutter.

Flutter est un kit de développement logiciel d'interface utilisateur open-source créé par Google. Il est utilisé pour développer des applications pour Android, iOS, Linux, Mac, Windows, Google Fuchsia et le web à partir d'une seule base de code

5.2.19 Dart**Figure 5.19** Dart.

Dart est un langage de programmation optimisé pour les applications sur plusieurs plateformes. Il est développé par Google et est utilisé pour créer des applications mobiles, de bureau, de serveur et web. Dart est un langage orienté objet à ramasse-miettes avec une syntaxe de type C++.

5.2.20 Getx



Figure 5.20 Getx.

Get ou GetX est un logiciel rapide, stable, extra- cadre léger pour la création d'applications Flutter. GetX est livré prêt à l'emploi avec une gestion d'état haute performance, une injection de dépendances intelligente et une gestion des itinéraires d'une manière simpliste et pratique

5.2.21 Firebase



Figure 5.21 Firebase.

Firebase est un ensemble de services d'hébergement pour n'importe quel type d'application. Il propose d'héberger en NoSQL et en temps réel des bases de données, du contenu, de

l'authentification sociale, et des notifications, ou encore des services, tel que par exemple un serveur de communication temps réel.

5.2.22 Json



Figure 5.22 Json.

JSON (JavaScript Objet Notation) est un langage léger d'échange de données textuelles. Pour les ordinateurs, ce format se génère et s'analyse facilement. Pour les humains, il est pratique à écrire et à lire grâce à une syntaxe simple et à une structure en arborescence

5.2.23 Yaml



Figure 5.23 Yaml .

YAML est un langage de sérialisation des données qui est souvent utilisé pour coder des fichiers de configuration.

5.3 Présentation des interfaces de l'application

Dans ce qui suit, nous allons présenter les interfaces de notre application.

5.3.1 Interface d'inscription

Cette figure offre un aperçu de l'interface de création d'un nouveau compte.

Sur cette interface, l'utilisateur doit saisir son nom (identifiant), son numéro de téléphone et cliquer sur le bouton valider pour attendre un code de vérification.

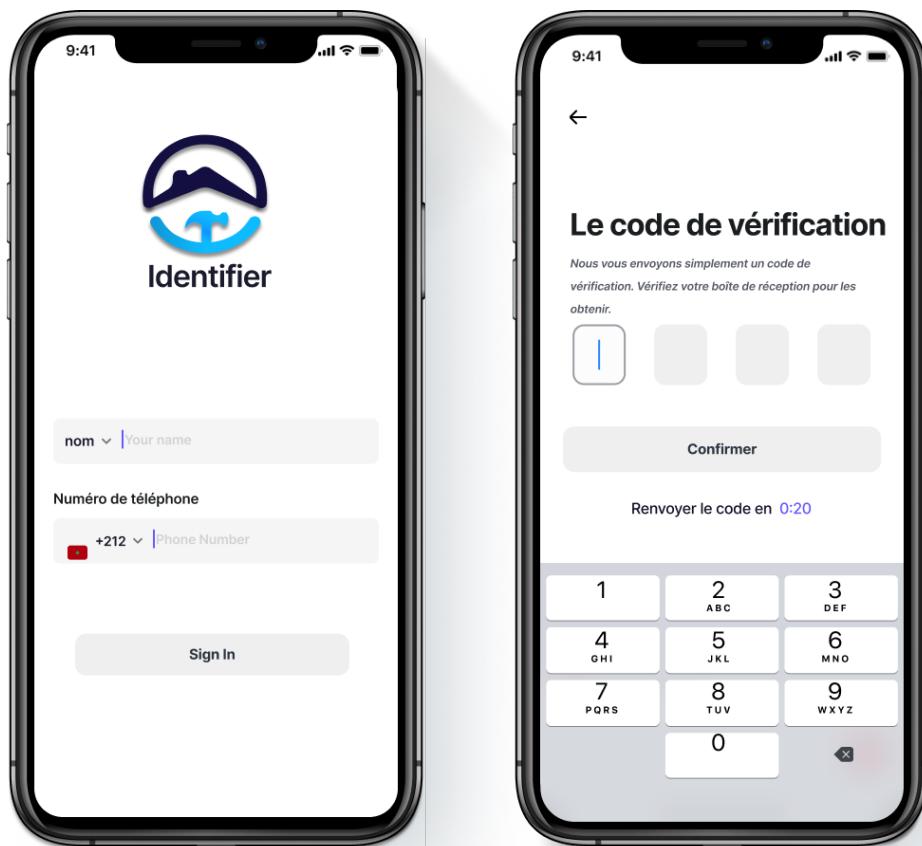


Figure 5.24 Interface d'inscription.

5.3.2 Interface d'accueil

Cette interface offre un aperçu de la page d'accueil de l'application. Elle offre aux utilisateurs un service d'inscription et d'identification.

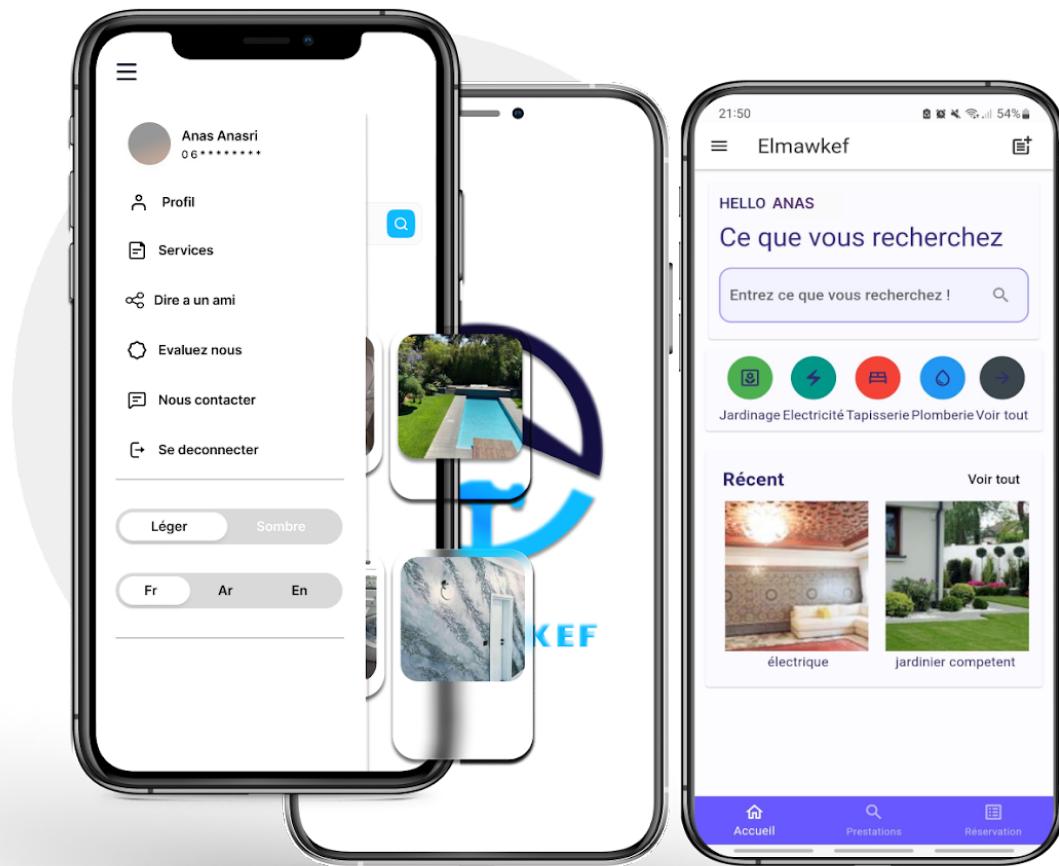


Figure 5.25 Page d'accueil .

5.3.3 Interface pour chercher le profil du professionnel

Pour trouver un pour le service voulu l'utilisateur pourra faire une recherche par texte ou par catégorie

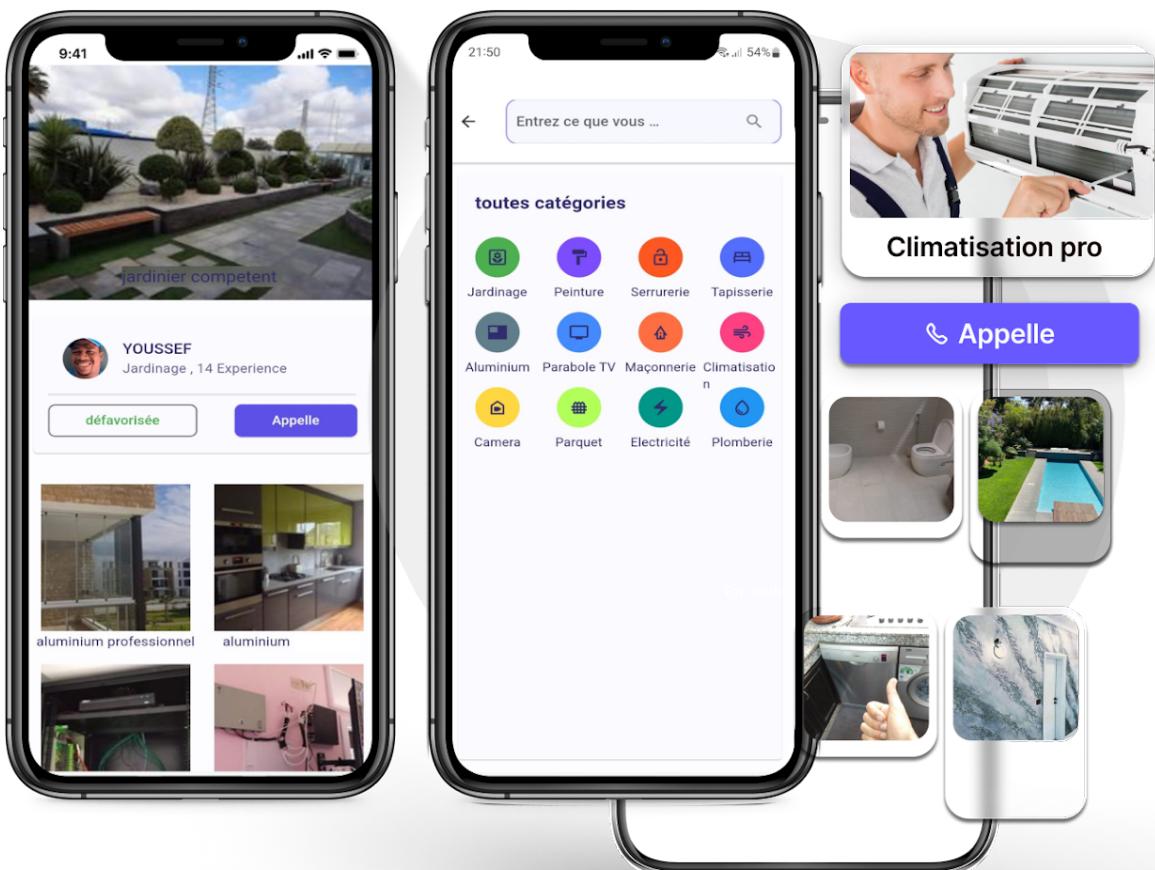


Figure 5.26 chercher le profil .

5.3.4 Interface profil

Sur cette interface, l'utilisateur aura un accès supplémentaire pour l'ajout d'autres informations sur son profil .

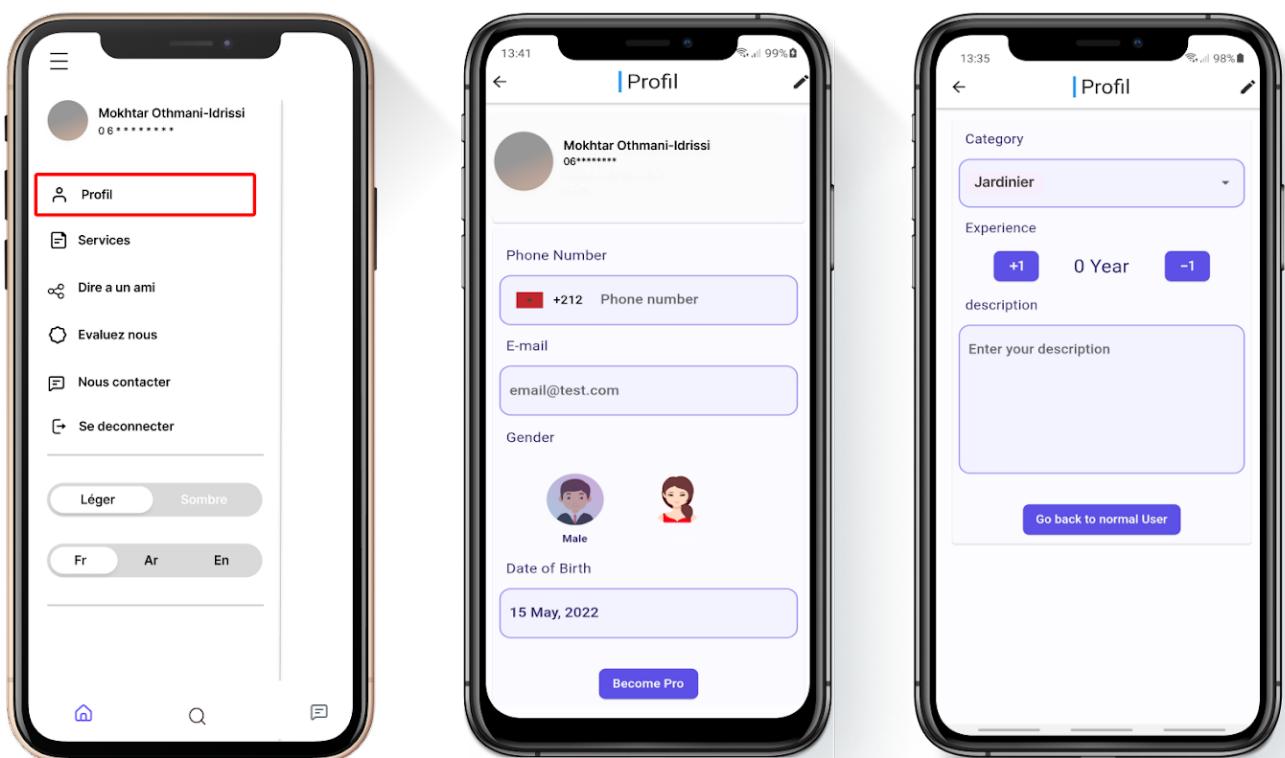


Figure 5.27 Interface profil .

5.3.5 Option dark / light

Cette option est sélectionnée par défaut suite au thème sélectionné sur le téléphone, mais l'utilisateur aura la possibilité de la changer manuellement



Figure 4.28 Option dark / light .

5.3.6 Option de la langue

Cette option est sélectionnée par défaut suite à la langue sélectionné sur le téléphone, mais l'utilisateur aura la possibilité de la changer manuellement

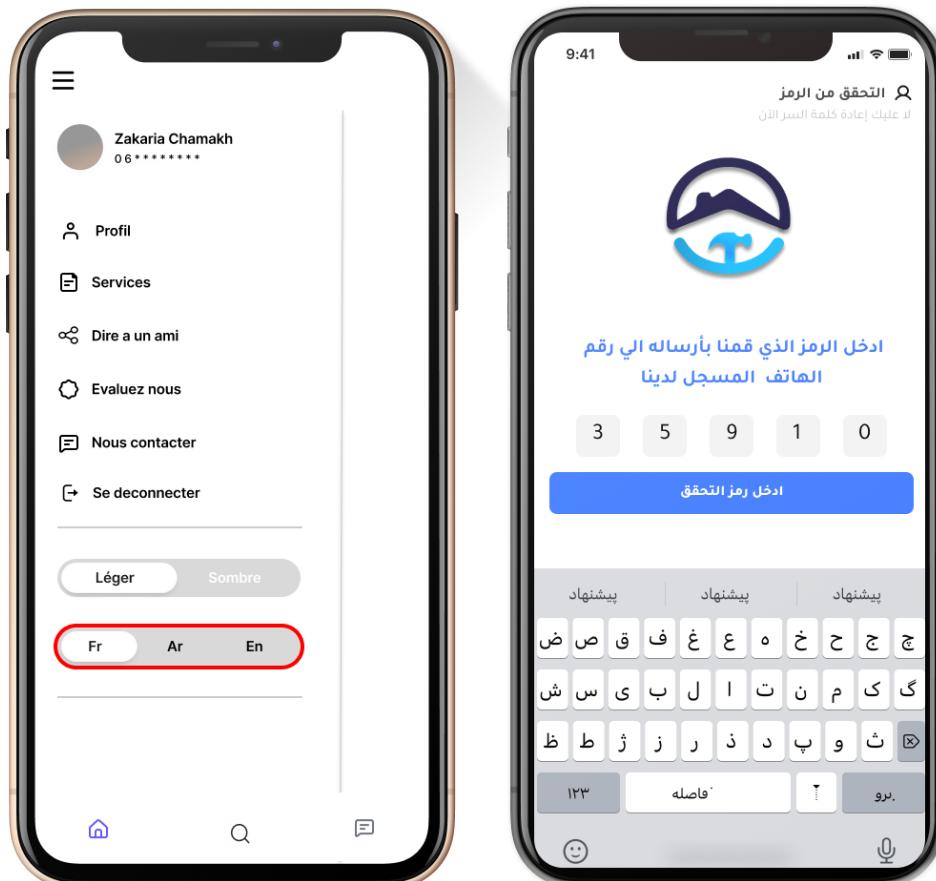


Figure 5.29 Option de la langue .

5.3.7 Ajout des services

cette option permet au bricoleur d'ajouter les services qu'il propose sur son profil .

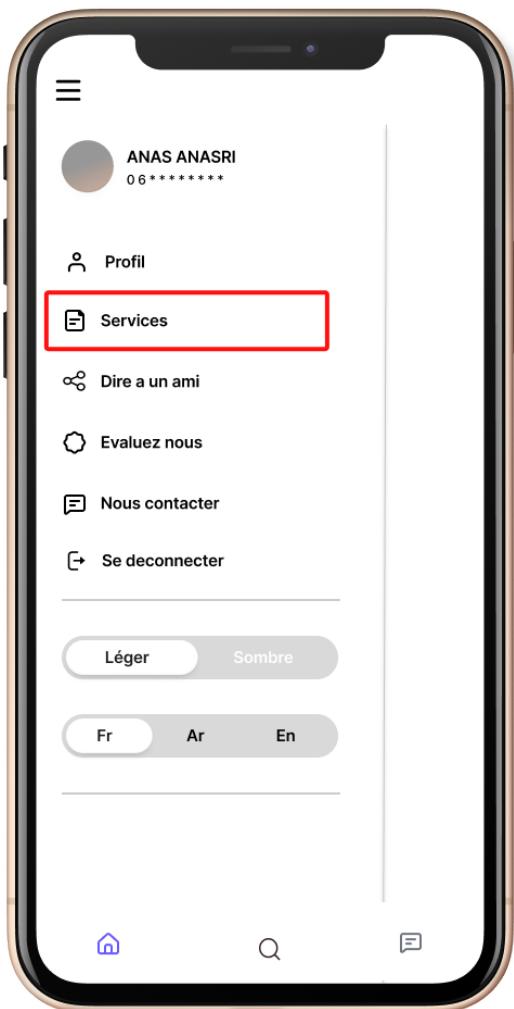


Figure 5.30 Ajout des services .

5.3.8 Option du partage

Cette option va rediriger l'utilisateur vers le site web où plusieurs informations sont présentées à propos de notre projet et il va pouvoir partager notre projet .



Figure 5.31 Option du partage .

5.3.9 Option d'évaluation

Cette option va permettre à l'utilisateur d'évaluer notre projet .

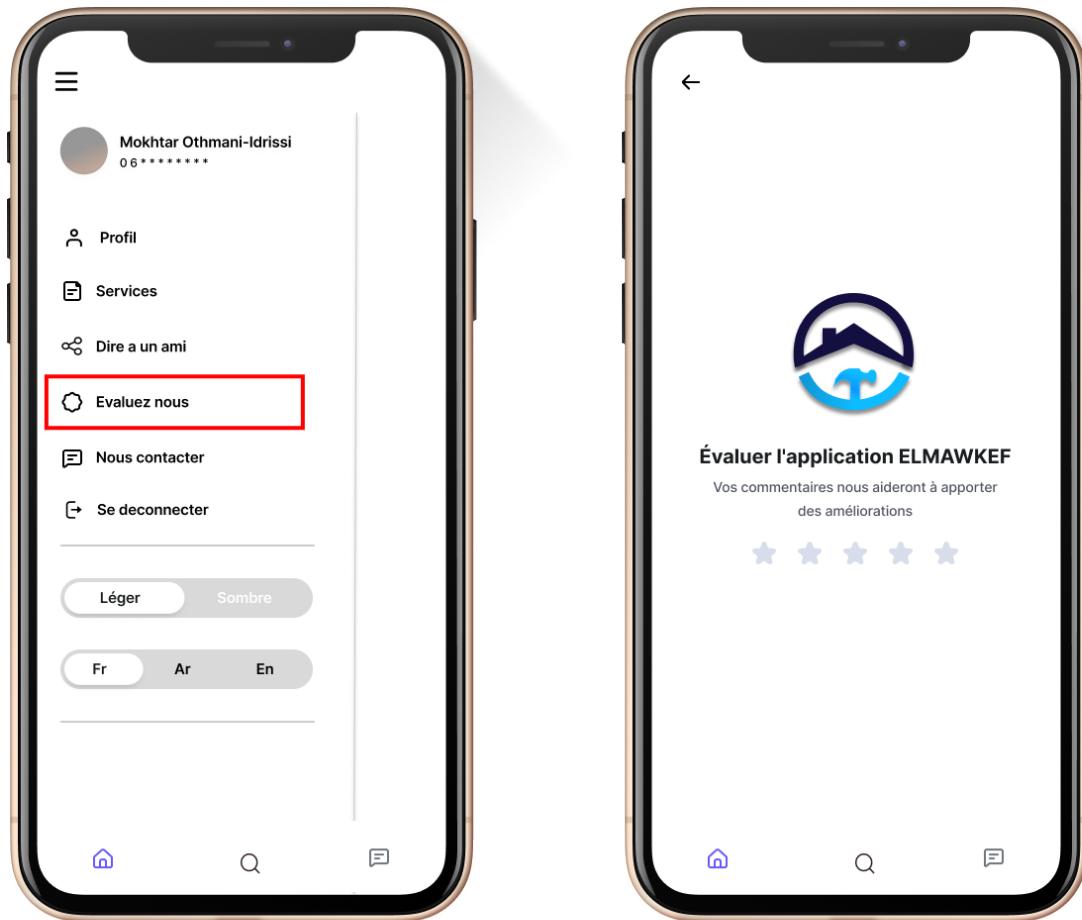


Figure 5.32 option d'évaluation .

4.3.10 Option nous contacter

Cette option va permettre à l'utilisateur de créer un ticket afin de nous contacter à propos de leur problème.

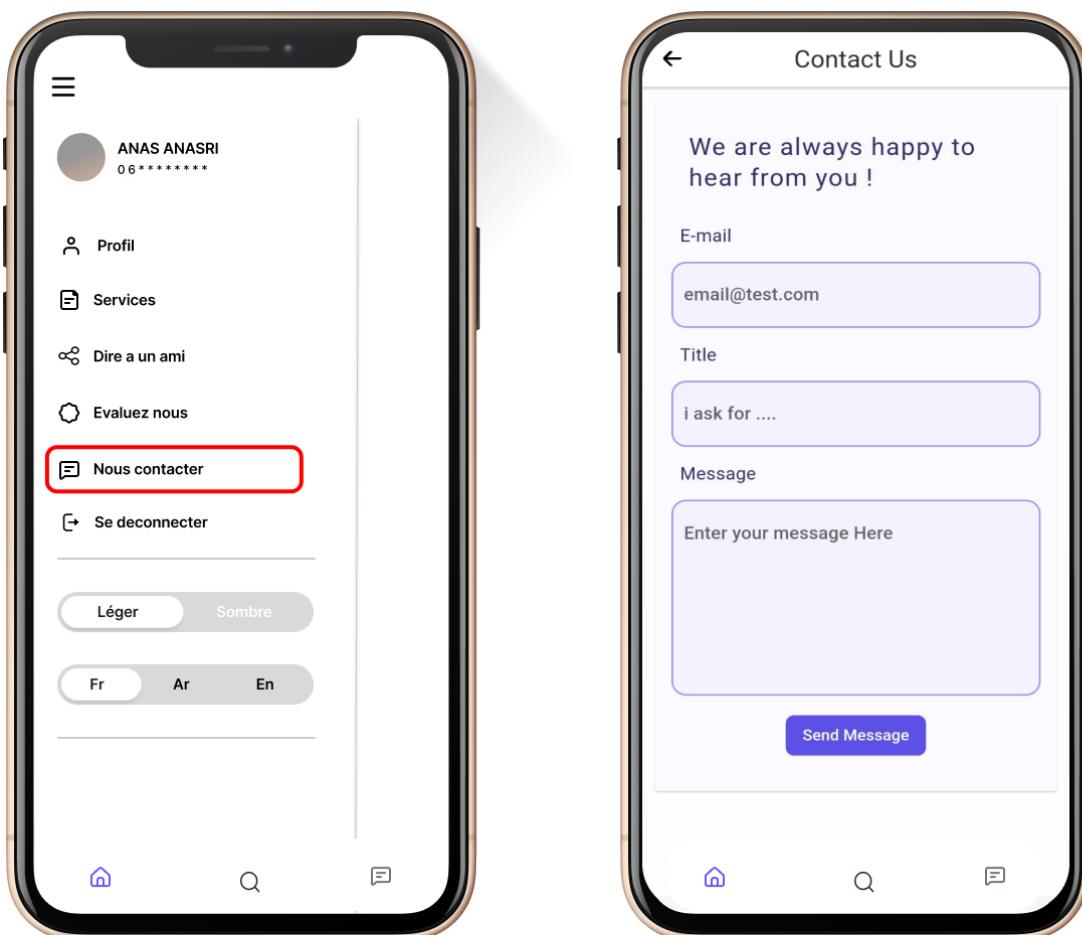


Figure 5.33 Option nous contacter .

5.3.11 Option se déconnecter

Cette option va déconnecter l'utilisateur de l'application .

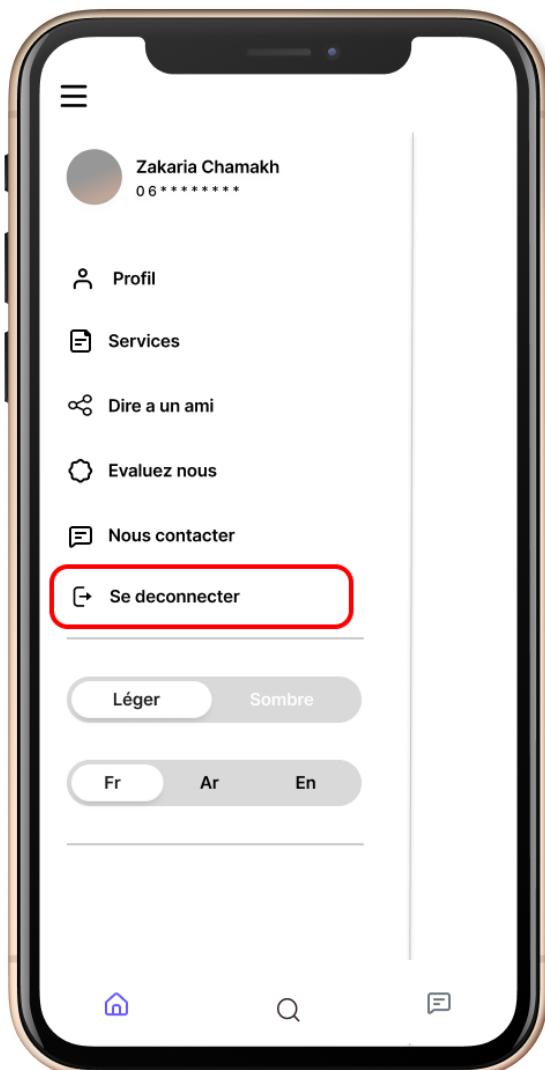


Figure 5.34 Option se déconnecter.

5.4 Conclusion

La phase réalisation est l'étape la plus importante dans le cycle de vie d'une application. Dans ce chapitre, nous avons décrit brièvement le processus de réalisation de notre application en spécifiant l'environnement de développement associé à notre système. En effet, nous avons achevé l'implémentation tout en respectant la conception élaborée. En d'autres termes, nous détenons la version finale de l'application, installée dans notre environnement.

Conclusion générale et Perspectives

Nous sommes parvenus, par le biais de ce projet, à réaliser une application qui fonctionne sous Android/Ios/web qui permet aux utilisateurs de trouver un bricoleur dans n'importe quel domaine.

Au cours de la phase d'analyse nous avons structuré et défini les besoins du système. Il s'agit de formuler, d'affiner et d'analyser la plupart des cas d'utilisations par les diagrammes UML.

La phase de conception suit immédiatement la phase d'analyse, il s'agit alors d'étendre la représentation effectuée au niveau de l'analyse en y intégrant les aspects techniques les plus proches des préoccupations des besoins techniques. L'élément principal à livrer au terme de cette phase est le diagramme de classe ainsi que le schéma relationnel.

Enfin, nous avons entamé la réalisation en utilisant les outils d'implémentation appropriés à l'intégration du contenu, la gestion de la base de données (MySQL), et présenter les différentes interfaces de notre application.

Ce travail nous a permis d'apprendre énormément de choses concernant le développement sous Android, et ce mémoire présente notre première expérience pour la mise en oeuvre d'une application mobile, également nous avons appris manipuler toute une panoplie d'outils :Android Studio and IntelliJ,Visual Studio Code,Flutter Framework,Getx Digitalocean,Github,Firebase et quelques langages de programmation telles que JavaSCRIPT,Nodejs, Strapi,Dart,postgresql,html/css, Json,yaml .

Ce fut une occasion pour nous de compléter de manière transversale nos compétences en informatique, d'élargir et d'approfondir nos connaissances et les apprécier aux diverses réalités du terrain. Cependant des perspectives d'améliorations de notre application restent envisageables pour être enrichies par des fonctionnalités avancées .

Webographiques

- [1] <https://docs.flutter.dev/development/tools/android-studio>
- [2] <https://docs.flutter.dev/development/tools/vs-code>
- [3] <https://www.postman.com/product/tools/>
- [4] <https://www.digitalocean.com/>
- [5] <https://www.namecheap.com/>
- [6] <https://github.com/>
- [7] <https://www.figma.com/>
- [8] <http://trello.com/https://plantuml.com/>
- [9] <https://strapi.io/>
- [10] <https://flutter.dev/>
- [11] <https://docs.flutter.dev/development/ui/layout/tutorial>
- [12] <https://docs.flutter.dev/development/ui/layout>
- [13] <https://dart.dev/>
- [14] <https://github.com/jonataslaw/getx/blob/master/README-fr.md>