Université Abou Bekr Belkaid
Tlemcen Algérie
جامعة أبي بكر بلقايد
تلمسان الجزائر

République Algérienne Démocratique et Populaire
Université Abou Bakr Belkaid– Tlemcen
Faculté des Sciences
Département d'Informatique

Mémoire de fin d'études

pour l'obtention du diplôme de Master en Informatique

Option : *Génie Logiciel (G.L)*

Thème

Réalisation d'un système de commande de plats en ligne auprès des restaurants

Réalisé par :

- **BESTAOUI YACINE FADL ALLAH**

Présenté le 27 Juin 2022 devant le jury composé de :

- *M. Benamar Abdelkrim* *(Président)*
- *Mme. Benmansour Fazilet* *(Examinatrice)*
- *M. Tadlaoui Mohamed* *(Encadrant)*
- *M. Bouabdellah Karim* *(Co-Encadrant)*

Année universitaire : 2021-2022

R e m e r c i e m e n t s

Louanges à Notre Seigneur le Tout puissant, le Miséricordieux de nous avoir donné la capacité, la force et la volonté de parvenir à réaliser ce modeste projet.

Mes premiers remerciements vont à ***Monsieur Tadlaoui Mohamed****, pour l'honneur qu'il m'a fait d'avoir été mon chef de projet, pour le choix du thème et de m'encadrer.*

Je lui sais gé d'avoir assuré la direction de mes recherches avec cette clairvoyance et cette franchise qui le caractérisent, pour la richesse de ses enseignements, ses encouragements constants et toujours bienveillants.

Sa mise en confiance dès les premiers instants, sa présence permanente, ses efforts, remarques et références, et ses orientations avisées m'ont permis de surmonter les nombreuses difficultés et de mieux orienter le choix méthodologique, enfin pour m'avoir guidé et poussé plus loin dans la réflexion et dans l'écriture.

Je lui adresse toute ma gratitude pour l'assurance qu'il m'a accordé quant à la proposition de stage, première immersion en entreprise au sein de ***Naltis*** *agence conseil en communication et pour ses encouragements pendant sa durée complète.*

Je tiens également à affirmer toute ma reconnaissance envers ***Naltis****, pour m'avoir accueillie en stage et qui m'a permis de l'accomplir dans de bonnes conditions. J'ai eu le privilège de découvrir une importante entreprise, bien structurée et en plein essor.*

Je témoigne ici toute ma reconnaissance à ***Monsieur Bouabdellah Karim*** *son Directeur Général, mon référent de stage pour son professionnalisme, son esprit de partage des connaissances, sa disponibilité, son soutien et sa patience tout au long de cette période. Cette proximité m'a permis de mener à bien la résolution de la problématique qui m'a été confiée.*

Ma sincère considération va également au Président du jury ***Monsieur Benamar Abdelkrim*** *et à l'Examinatrice* ***Madame Benmansour Fazilet*** *qui auront à lire, qui ont accepté d'examiner et d'évaluer ce travail.*

Pour terminer, merci à l'ensemble des enseignants du département informatique qui ont veillé à notre formation durant tout le cursus universitaire.

D é d i c a c e s

Ce travail n'aurait pas pu être réalisé sans l'appui et les efforts fournis par les personnes chères à mon cœur. J'ai nommé mes chers parents, c'est pour cela que je leur dédie en premier cette modeste recherche.

Je souhaite aussi le dédicacer à mes frères, mon beau-frère, ma très chère sœur, et mes belles-sœurs, pour leurs encouragements et leurs précieux conseils.

Aussi à tous mes amis de la faculté avec qui j'ai eu plaisir à étudier et à travailler tout au long de ma formation universitaire.

Et enfin à toutes celles et ceux qui ont participé de près ou de loin à la réalisation de ce projet.

Merci.

Table des matières

Liste des figures

Liste des tableaux

Table des abréviations

PDA	Personal Digital Assistant
RFID	Radio-identification
PHP	Hypertext Preprocessor
UML	Unified Modeling Language
TS	TypeScript
JS	JavaScript
API	Application Program Interface
MVC	Model-Vue-Controlleur
REST	REpresentational State Transfer
JSON	JavaScript Object Notation
BDD	Base de données
OAUTH	Open Authorization
OTP	One-Time Password
SCL	Système de commande en ligne
ORM	Mapping objet-relationnel
HTTP	Hypertext Transfer Protocol
URL	Uniform Resource Locator
I18N	Internationalisation
RTL	Right to Left
FCM	Firebase Cloud Messaging
SSL	Secure Socket Layer
XSS	Cross-site scripting
DOS	Denial of Service
CSRF	Cross Site RequestForgery
FTP	File Transfer Protocol

Introduction Générale

Depuis l'avènement d'internet, notre rapport à la nourriture a changé ; nous sommes devenus très influencés par ce qui nous est montré. La propagation des réseaux sociaux a permis aux gérants de restaurants d'être en contact permanent et au plus près de leur clientèle en partageant leurs services.

Dans le système traditionnel (directement de bouche à oreille), des problèmes de mauvaises communications entre les deux parties impactaient directement la commande et par là-même la renommée et l'image du restaurant (mauvais plat, mauvais supplément, quantité erronée, non-respect du détail de la commande, plat ou supplément non-disponibles, etc.) et le mécontentement du client.

Au vu de cette problématique, beaucoup d'entreprises ont cherché à développer des applications qui permettent aux consommateurs de commander d'une manière simple et efficace, et qui aident les restaurateurs à gérer leur restaurant en ligne pour recevoir les commandes en temps réel.

En accord avec le département informatique, un stage en entreprise a été effectué, plus particulièrement chez Naltis communication sise à Tlemcen. L'agence œuvre depuis de nombreuses années, avec un rayonnement national, à élaborer la communication interne et externe pour sa large gamme de clients.

Dans le cadre de son activité et grâce à la combinaison de son savoir-faire et des nouvelles technologies de l'information et de la communication, elle occupe une place majeure dans le développement de logiciels, standards ou personnalisés, en fonction des besoins des professionnels.

La particularité de Naltis réside dans sa capacité avérée à développer, sur les nouveaux supports de la communication comme les sites web ou les CD-ROM, des plateformes informatiques pouvant offrir de multiples services aux différents utilisateurs (managers, commerciaux, clients, etc.), en traitant l'information de manière poussée et efficace.

Le but du stage a consisté à réaliser un système pouvant offrir aux clients de passer des commandes à distances auprès de restaurateurs préalablement enregistrés dessus. De la sorte, cette procédure entièrement dématérialisée va mettre en relation deux protagonistes le client et le restaurateur. L'acheteur bénéficiera d'un mode de

restauration dit rapide dont la démarche est plus simple. Le vendeur va améliorer, sa productivité et par donc va satisfaire sa clientèle. Comme autres avantages, il y a :

- Une prestation plus performante des restaurants envers leurs consommateurs.
- Une gestion optimale des menus, des plats et des suppléments pour un restaurant.
- Un suivi des commandes en temps réel.
- Une visualisation des restaurants sur une Map.
- Effectuer une commande directement depuis l'application mobile.
- Etc.

C'est dans le cadre d'une nécessité de dématérialiser cette procédure que notre travail va s'inscrire en se basant sur les différents systèmes déjà existants.

Notre mémoire sera articulé comme suit :

Le premier chapitre va être réservé au système de commande en ligne, nous parlerons de la gestion de restaurant pour le restaurateur et de la procédure de commande en ligne pour un consommateur.

Le deuxième chapitre sera consacré à l'analyse comparative de l'existant des différents systèmes déjà existants, avec leurs points forts et leurs points faibles.

Enfin le troisième chapitre, sera axé à notre contribution au projet en montrant la phase de conception avec la spécification des besoins, et en évoquant l'architecture globale. Puis certains diagrammes avec les outils, le système et la technologie utilisés seront exposés. Pour terminer, une présentation de l'application sera effectuée en évoquant les problèmes rencontrés, plus brièvement le dépoilement et le test du système.

1. Systèmes de commande en ligne

1.1. Introduction :

Ce chapitre est composé d'une présentation du système traditionnel de commande suivi du système de commande en ligne auprès de restaurants, nous présentons les inconvénients des anciens systèmes et les avantages des SCL. Ainsi que quelques solutions d'automatisation du système développées au cours des années.

1.2. Système traditionnel de commande :

Le système traditionnel de commande est la procédure la plus courante, c'est celui encore utilisé mondialement et peut avoir deux formes :

- Commande sur place : le consommateur se présente à la réception du restaurant choisi où il est accueilli puis conduit vers sa table. Le serveur remet le manuscrit au client qui contient le menu avec la liste des divers mets qui composent le repas avec les prix. Un moment après, le serveur répond aux interrogations du client (composition du plat, ingrédients, type de cuisson, etc.), et inscrit la commande sur un carnet. Ensuite il remet la fiche en cuisine pour sa préparation, une fois prête elle est servie au client.

- Commande par téléphone : le consommateur appelle le restaurant sélectionné, demande les plats proposés ainsi que les prix (s'il ne les connait par), fait son choix et donne sa commande à son interlocuteur. Ce dernier enregistre la commande ainsi que le mode de récupération (enlèvement sur place ou livraison avec adresse) puis la remet au cuisinier pour exécution, pour terminer il renseigne le consommateur sur le délai d'attente.

1.3. Inconvénients du système traditionnel de commande :

Le système traditionnel de commande comporte de nombreux inconvénients qui peuvent nuire aux business du restaurateur, exemple :

- Difficulté à générer des rapports : renseigner manuellement les informations avant de générer des rapports.
- Erreur de calcul : pour établir une facture, les calculs sont faits manuellement, d'où les risques d'erreurs.
- Beaucoup de documents à gérer : le système existant nécessite une multitude de dossiers.

- Perte de documents : la perte d'un seul papier conduit à une situation embarrassante.
- Prend du temps : chaque tâche est effectuée manuellement, faire très attention à tout renseignement inscrit.
- Non-convivial : le système existant est ennuyeux et pénible car la récupération des données est très lente et les données ne sont pas gérées efficacement.
- Difficulté pour commander : pour passer des commandes les clients doivent soit téléphoner au restaurant, soit se rendre personnellement sur place pour connaître les plats proposés.
- Erreurs lors d'une commande : à maintes reprises, il a été déploré que des commandes passées par téléphone se mélangent, par une mauvaise interprétation, par un manque de communication, etc.

1.4. Communication consommateur/restaurateur :

La procédure de commande de plats implique habituellement une communication directe entre le consommateur et le restaurateur par le bais d'un appel téléphonique ou par un déplacement sur site.

Dans la majorité des cas, l'échange téléphonique est prisé par les clients habituels (ils connaissent déjà les plats proposés ainsi que leur prix), les autres préfèrent généralement se rendre sur place pour avoir une première idée sur le restaurant et découvrir le/les menus proposés avec que les prix.

Le chargé de commande écrit habituellement le contenu d'une commande sur un papier puis le remet au cuisinier pour lancer la préparation, une fois prête il la met à disposition pour enlèvement ou livraison.

1.5. Système de commande en ligne (SCL) :

Le but du SCL est de remplacer le mode traditionnel de prise de commande par un système informatisé, ce système apportera des rapports récapitulatifs de commandes, rapidement, dans un format correct et à tout moment, visant à être utilisé dans l'industrie de la restauration. Cette fonctionnalité aide les hôtels et les restaurants à augmenter leurs chiffres d'affaires en offrant de nouveaux services, les clients peuvent choisir parmi une large gamme de plats en quelques minutes et quelques clics seulement.

Depuis déjà plusieurs années, le SCL est devenu largement utilisé, particulièrement par la génération Z (personnes nées entre 1995 et 2010), connue pour être la plus familière avec les applications des technologies de l'information [1]. Cette génération a rapidement maitrisé son exploitation et sa manipulation. Elle a vite été la cible des entreprises qui cherchaient à faire plus de revenus en trouvant un nouveau public, c'est ainsi qu'est née l'idée de créer des systèmes de commande en ligne, ce procédé a commencé à avoir beaucoup d'intérêts et d'attraits.

Aujourd'hui le secteur du fast-food moderne est également capable de livrer rapidement et facilement chez un client. La principale raison de tout cela est que ce mode de consommation profite aux deux parties, et chacun y trouve satisfaction et intérêts [2].

1.6. Automatisation du processus de commande :

Un système informatisé de restauration est un système informatique intégré qui supervise, gère et facilite les opérations de planification en restaurant [3]. Avant l'apparition des systèmes d'automatisation, les commandes étaient gérées manuellement sur papier/registres/carnets. L'usage quotidien d'internet et la généralisation de l'informatique a permis le recours à de nouvelles technologies ; les chercheurs scientifiques ont commencé alors progressivement à mettre en place différents types de solutions pour automatiser les opérations exécutées au sein d'un restaurant.

Pour améliorer le système, de nombreux chercheurs se sont attachés à trouver des solutions pour un aspect, un problème spécifique ou récurrent auxquels sont confrontés les restaurateurs. Les solutions apportées peuvent être classées en différentes catégories, voici quelques-unes :

1.6.1. Système basé sur PDA :

PDA est un assistant numérique personnel dont l'usage est prévu originalement dans un but d'organisation (agenda, gestionnaire de tâche, carnet d'adresses), mais offrent également de nombreuses fonctions pour des smartphones.

Dans ce système, les PDA sont utilisés au sein d'un restaurant pour voir les menus et passer des commandes. Une fois les commandes validées par le client via PDA, le responsable peut les visionner sur ordinateur [4]. Les fonctions de commande de nourriture sont servies à la fois par des ordinateurs de bureau et des PDA sur un

réseau local intégré filaire/sans fil. Il y a d'autres variantes de ce système : pour l'une d'elle le serveur utilise le PDA afin de renseigner les commandes, pour d'autres le consommateur est muni d'un PDA afin de commander.

Limitations du système :

- Le restaurateur doit avoir un nombre suffisant de PDA pour servir une quantité conséquente de clients.
- L'hébergement et la maintenance d'un nombre suffisant de PDA peut entrainer l'augmentation des dépenses/coûts du restaurant.
- Le partage des mêmes PDA entre de nombreux clients risque de propager des bactéries et des maladies contagieuses [5].
- Les PDA ne sont plus aussi largement utilisés en raison de l'innovation de la technologie des téléphones portables.

1.6.2. Système basé sur RFID :

Ce système permet aux serveurs d'identifier immédiatement les clients via des cartes de membres basées sur la RFID, puis de suggérer activement les menus les plus appropriés à travers un système de recommandations de menus aux clients. L'adhérent sera identifié par cette carte basée sur la RFID et le menu d'aliments recommandés en fonction de son choix passé d'aliments [6]. Pour prendre les commandes, les serveurs utilisent des PDA connectés au système par le bais d'un réseau local (LAN).

Limitations du système :

- Le client doit se munir de sa carte au restaurant, sinon il ne peut pas utiliser toutes les installations du système.
- Ce système se concentre exclusivement sur les clients titulaires de la carte de membre ; mais tous n'ont pas ce service.
- La carte est uniquement utilisable dans un restaurant en particulier.

1.6.3. Système basé sur les applications web :

Le système repose sur le développement d'une application web pour automatiser le processus de commande auprès du restaurant et éliminer la paperasse [6]. Ce système a pu surmonter le problème de la dépendance à un appareil spécifique (PDA) ou à un système d'exploitation.

Limitations du système :

- L'application web est souvent difficile à utiliser pour des non familiers.
- Une mauvaise performance sur certains types et tailles d'appareils.
- Il est difficilement accessible pour les personnes ne connaissant pas l'adresse du site.

1.6.4. Système basé sur les applications mobile :

Les clients peuvent commander des plats à distance depuis une application mobile, la vitesse de traitement des commandes depuis une application mobile est plus rapide comparativement au système de commande basé sur PDA [7]. L'application est médiatrice entre le client, le gérant du restaurant et la cuisine.

Limitations du système :

- Les clients doivent installer l'application mobile du restaurant pour bénéficier des avantages du système de restaurant intelligent.
- Il n'est pas possible de commander des plats à partir d'un ordinateur avec ces applications, car les applications mobiles ne peuvent pas être parcourues via des ordinateurs..

1.7. Avantages du SCL :

On va énoncer quelques avantages du SCL :

- Les commandes sont passées par les clients sans avoir à se soucier des contraintes de lignes téléphoniques bloquées.
- Les restaurateurs peuvent désormais accepter plus de commandes qu'auparavant (par téléphones), réduisant ainsi la fiabilité des lignes téléphoniques et le coût des appels.
- Le système de commande en ligne permet aux clients de passer leurs commandes sans aucune perturbation de leurs horaires quotidiens.
- Les clients peuvent voir leur commande en ligne.
- Les plats sont organisés dans des menus de manière très systématique afin que les clients puissent y accéder facilement, sans tracas.
- Elimine le risque de confusion les commandes et aide le restaurateur à les recevoir formatées d'une façon simple et utilisable.

1.8. Conclusion :

Au cours de ce chapitre, après avoir décrit les systèmes traditionnels de commande en l'occurrence, les deux formes existantes, la commande sur place et la commande par téléphone, leurs avantages et leurs inconvénients ont pu être facilement identifiés. Après la désignation des nouveaux systèmes de remplacement à savoir les systèmes de commande en ligne, l'accent a été mis sur l'automatisation qu'ils ont connue (PDA, RFID, etc.). En dernier, le développement du SCL a offert des avantages concrets, particulièrement l'amélioration qualitative des échanges de communications entre consommateur et restaurateur et par voie de conséquence le perfectionnement de la prestation attendue.

Le chapitre suivant sera consacré à l'étude des systèmes similaires en présentant quelques applications existantes tout en comparant leurs fonctionnalités.

2. Étude des systèmes similaires

2.1. Introduction :

Dans ce chapitre sera analysé l'existant des applications mobiles utiles pour la gestion du restaurant et la commande de plats, elle sera étayée par une étude comparative entre leurs fonctionnalités pour dégager leurs points forts et leurs points faibles.

A défaut de pouvoir connaitre les technologies et les outils utilisés (domaine de l'architecte/développeur) qui ont permis de développer leurs applications. Parmi celles disponibles en niveau national nous avons retenus trois importantes applications : Jumia Food, YASSIR Express, Fast Delivery. Au niveau international, de grandes applications peuvent être citées comme Food Booking, Gloria Food, Wendy's, Uber Eats, pour les besoins de notre étude nous avons retenu Uber Eats.

2.2. Application mobile de gestion restaurant :

L'analyse de ces applications mobiles de gestion restaurant va nous permettre de dégager leurs diverses prestations offertes :

2.2.1. Jumia Food Vendor App :

Un client utilise l'application Jumia Food pour passer une commande dans un restaurant sans devoir se déplacer. Le vendeur cuisine le repas et Jumia Riders le récupérera et le livrera au client. Le payement se fera à la livraison ou par carte [8].L'application propose aussi de recevoir les avis des clients pour prendre note, ainsi qu'un rapport d'activité pour suivre les chiffres obtenus en fonction de la stratégie de vente adoptée. La **Figure *2-1*** montre l'écran de connexion de cette application :

Figure 2-1 : Exemple Jumia Food Vendor App

2.2.2. YASSIR Express Store App :

Le consommateur commande des plats sur l'application YASSIR Express. Une fois prêtes elles seront directement signalées sur l'application, les livreurs se chargeront de récupérer et livrer les commandes aux clients [9].La prise en charge des suppléments pour les différents plats proposés offre en plus une grande flexibilité lors d'une commande. Pour avoir accès à cette application le restaurateur doit s'inscrite sur un formulaire afin que les services de YASSIR Express puissent entrer en contact avec l'intéressé. La **Figure *2-2*** montre l'écran de connexion de cette application :

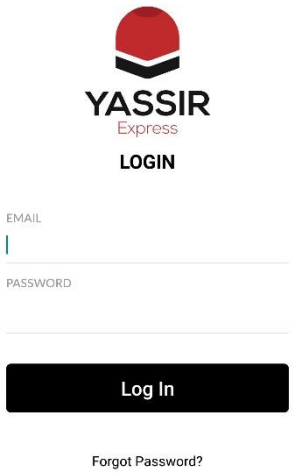

Figure 2-2 : Example YASSIR Express Store App

2.2.3. Fast Partner :

L'Application permet de gérer les commandes faites sur l'application Fast Delivery. Le restaurant doit être partenaire avec Fast Delivery pour pouvoir l'utiliser. La livraison est gérée sur Fast Delivery Coursier [10]. Pour élargir la commande du restaurant, l'application propose de vendre des produits de supermarché/épicerie pour toucher un plus large public. Il est à souligner qu'actuellement ce système n'est disponible que sur Alger. La **Figure *2-3*** montre l'écran de connexion de cette application :

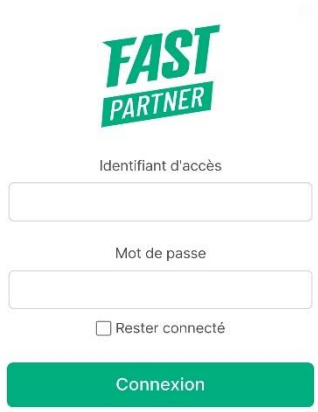

Figure 2-3 : Exemple Fast Partner

2.2.4. Uber Eats Manager :

Uber Eats Manager permet de gérer un restaurant depuis une plateforme unique. En utilisant un seul appareil dans un établissement ou une même application accessible à l'ensemble des employés depuis leur téléphone portable tel Uber Eats Order [11]. Un client passe une commande sur Uber Eat, puis il reçoit la commande sur les différentes applications, une fois la commande complétée un coursier utilisant la plateforme Uber prend en charge la commande pour livrer le client. La **Figure *2-4*** montre l'écran de connexion de cette application :

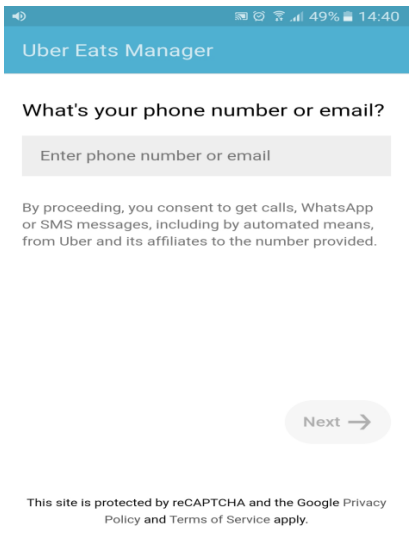

Figure 2-4 : Exemple Uber Eats Manager

2.3. Synthèse des applications mobiles de gestion restaurant :

Dans les applications ci-dessus, chaque solution a des avantages et des inconvénients différents, dont certains sont cités dans le **Tableau *2-1*** :

	Jumia Food Vendor App	**YASSIR Express Store App**	**Fast Partner**	**Uber Eats Manager**
Gestion commande	Oui	Oui	Oui	Oui
Historique commande	Oui	Oui	Oui	Oui
Recevoir avis consommateur	Oui	Non	Non	Oui
Notification pour nouvelle commande	Oui	Oui	Oui	Oui
Commission sur vente	Oui	Non précisé	Non précisé	Oui
Inscription directe	Non	Non	Non	Non
Prise en charge livraison	Oui (Avec Jumia Riders)	Oui (Avec YASSIR Express Partner)	Oui (Avec Fast Partner)	Oui (Avec Uber)
Gestion de plusieurs restaurants	Non	Non	Non	Oui
Gestion des suppléments	Non	Oui	Non	Oui
Rapport d'activité	Oui	Oui	Oui	Oui
Vendre produit de supermarchés / épiceries	Oui	Non	Oui	Oui
Synchroniser les appareils connectés	Non	Non	Non	Oui
Recevoir des alertes en cas de problème	Non	Non	Non	Oui

Tableau 2-1 : Comparatif des applications de gestion restaurant

2.4. Application mobile de commande de plats en ligne :

L'analyse de ces applications mobiles de commande de plats auprès de restaurants permet de dégager leurs diverses prestations offertes :

2.4.1. Jumia Food :

Jumia Food propose la livraison à domicile de repas, boissons, desserts, et de certain produit provenant de supermarché, pharmacie, etc [8]. Cette application propose un système de précommande qui permet de commander auprès de restaurants non-encore ouverts ; cette fonctionnalité est intéressante, elle préconise la présence d'un système de chat direct avec le restaurateur une fois la commande créée. La **Figure *2-5*** quelques captures d'écran de cette application :

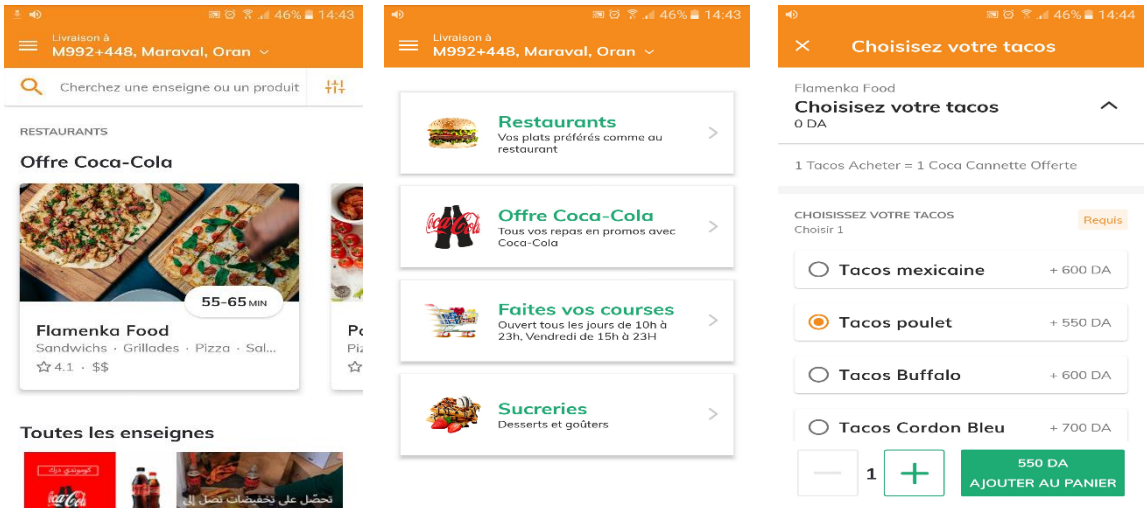

Figure 2-5 : Exemple Jumia Food

2.4.2. YASSIR Express :

YASSIR Express permet de choisir des plats, produit d'épicerie, fruits & légumes sans avoir à se déplacer pour commander et se faire livrer à domicile [9]. La livraison dans plusieurs villes et localisation représente un atout majeur contrairement aux autres applications qui ne proposent pas forcément la livraison partout, avec possibilité de payer par carte CIB qui est présent et opérationnel. La **Figure 2-6** montre quelques captures d'écran de cette application :

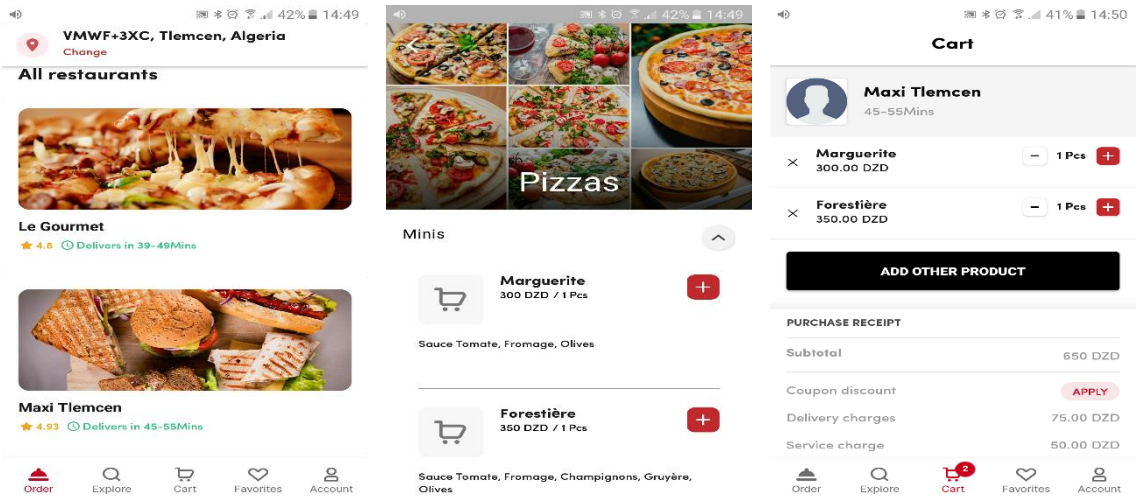

Figure 2-6 : Exemple YASSIR Express

2.4.3. Fast Delivery :

Fast Delivery est une application de livraison qui permet d'acheter et de se faire livrer n'importe quel produit à Alger, prise en charge que dans les restaurants et boutiques situé à Alger. Une fois la commande prête elle vous sera livrée avec la possibilité de suivre sa progression en temps réel [10]. Le consommateur a la possibilité d'acheter des produits de supermarché/épicerie en plus de la commande de plats, le payement par carte est présent mais non-utilisable car les restaurants ne l'autorisent pas pour le moment. La **Figure *2-7*** montre quelques captures d'écran de cette application :

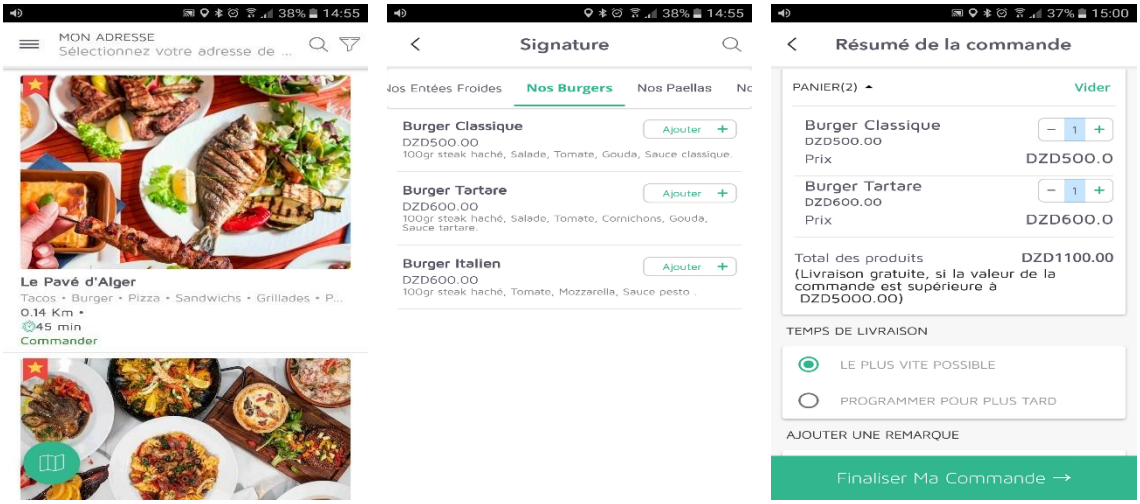

Figure 2-7 : Exemple Fast Delivery

2.4.4. Uber Eats :

Uber Eats regroupe des centaines de restaurants, vous rechercher un restaurant ou un type de cuisine, choisissez vos plats et confirmer la commande [12], ensuite

suivez la commande dans l'application (le restaurant accepte la commande, la commande est prête, le coursier la prend en charge, suivi de sa progression sur la carte, le coursier arrive à l'adresse et vous remet la commande). Uber Eats est présent dans beaucoup de pays (France, Etat unis, Belgique, etc.) pour le moment l'Algérie, le Maroc et la Tunisie ne sont pas encore pris en charge. La **Figure *2-8*** montre quelques captures d'écran de cette application :

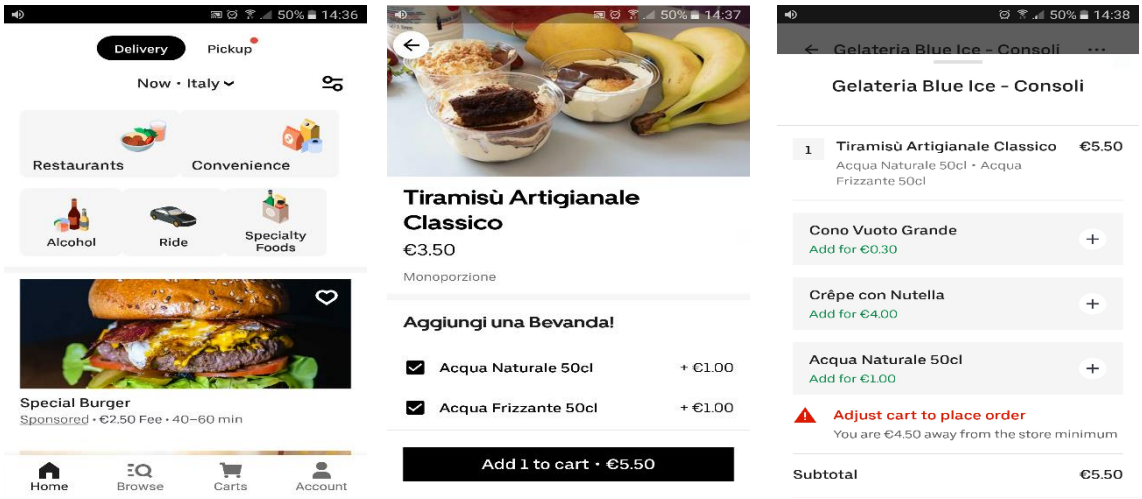

Figure 2-8 : Exemple Uber Eats

2.5. Synthèse des applications mobiles de commande de plats en ligne :

Dans les applications citées précédemment il existe des avantages et défauts différents pour chaque solution, dont certains sont notés dans le **Tableau *2-2*** :

	Jumia Food	**Yassir Express**	**Fast Delivery**	**Uber Eats**
Commande en ligne	Oui	Oui	Oui	Oui
Disponible sur plusieurs pays	Oui	Non	Non	Oui
Utilisation GPS	Oui	Oui	Oui	Oui
Livraison partout	Non (certaines ville et adresses)	Oui	Non (Alger seulement)	Oui
Payement par carte	Non (la grande majorité des restaurants ne le propose pas)	Oui (CIB)	Non (non-utilisable)	Oui
Payement à la livraison	Oui	Oui	Oui	Oui
Bon d'achat	Oui	Oui	Non	Oui
Choisir supplément pour un plat	Non	Oui	Non	Oui
Préciser détail commande (Allergie…)	Oui	Oui	Oui	Oui
Précommander	Oui	Non	Non	Oui
Annuler la commande manuellement	Non	Oui	Non	Oui
Chat en direct avec service	Oui	Non	Non	Oui
Visualisation sur Maps Intégrée	Non	Non	Non	Oui
Commande auprès de supermarchés / épiceries	Oui	Non	Oui	Oui
Souscrire à des packs spéciaux	Non	Non	Non	Oui

Tableau 2-2 : Comparatif des applications de commande de plats en ligne

2.6. Conclusion :

Durant ce chapitre, à la lumière de certaines applications sélectionnées, elles ont d'abord été présentées succinctement, ensuite une synthèse des fonctionnalités commerciales et attractives proposées par chacune d'elles a été dressée, ceci a alors permis de faire une analyse comparative de ces applications mobile.

Il en ressort que toutes les applications cherchent à répondre au mieux : à l'attente du restaurateur en ciblant la clientèle, en mettant en avant des critères parmi les plus attractifs (prix, photos, suppléments, notoriété, etc.), au souhait du client à découvrir de nouveaux plats, d'avoir la disponibilité et la sincérité de l'information, le choix et la qualité des produits achetés, la rapidité du service, la compétitivité des prix, etc.

Concernant les technologies de développement ayant servi à leur programmation, il est supposé qu'au vu de la large panoplie disponible et étant donné que le but recherché est identique, elles sont plus ou moins les mêmes.

3. Contribution

3.1. Introduction :

Apres avoir présenté le système, la problématique, suivis d'une analyse de résolutions semblables existantes. Nous allons proposer une solution informatique comme réponse à la problématique soulevée en s'aidant des comparatifs.

Ce chapitre présente le processus de réalisation suivi durant la réalisation du projet en présentant l'architecture générale puis de la conception élaborée. L'API réalisée sera présentée, suivie des applications mobiles et terminer par quelques tests effectués sur l'ensemble du système. Nous évoquerons également les différents systèmes et technologies utilisés pour sa réalisation.

3.2. Processus de réalisation du projet :

En première phase nous avons organisé de nombreuses réunions pour recueillir et comprendre les besoins fonctionnels et non-fonctionnels du SCL, après chaque séance le cahier de charge fonctionnel a été amélioré afin d'atteindre un état où les besoins étaient satisfaisants (suffisamment clairs, précis et en total accord avec le souhait du client). En parallèle les maquettes de nos applications mobiles ont été réalisées. Tout au long de cette partie nous nous sommes assurés en permanence de la faisabilité des besoins.

Suite à cela, la phase de conception a débuté avec la réalisation du cahier de charge technique en prenant en compte toutes les remarques et détails évoqués dans le cahier de charge fonctionnel pour réaliser : l'architecture globale du projet, les diagrammes (classe, cas d'utilisation, séquence). Le choix des technologies à utiliser et des systèmes à mettre en place ont été fixés.

La troisième phase a été consacrée au développement, réalisation de l'API RESTOZAD (centre de notre projet) car elle constitue l'interface de communication de nos applications, une documentation lui a été ajoutée, elle est nécessaire et importante sur le long terme. Elle est suivie par le développement des applications mobiles en s'appuyant sur les maquettes déjà réalisées auparavant.

Afin d'évaluer les aptitudes du programme, des tests ont été effectués sur quelques scénarios d'exécution pour voir le comportement et la réponse de l'API et des applications mobiles en cas de succès ou d'échec.

La **Figure *3-1*** présente une vue globale sur le processus de réalisation du projet, schématisé avec le diagramme de Gantt :

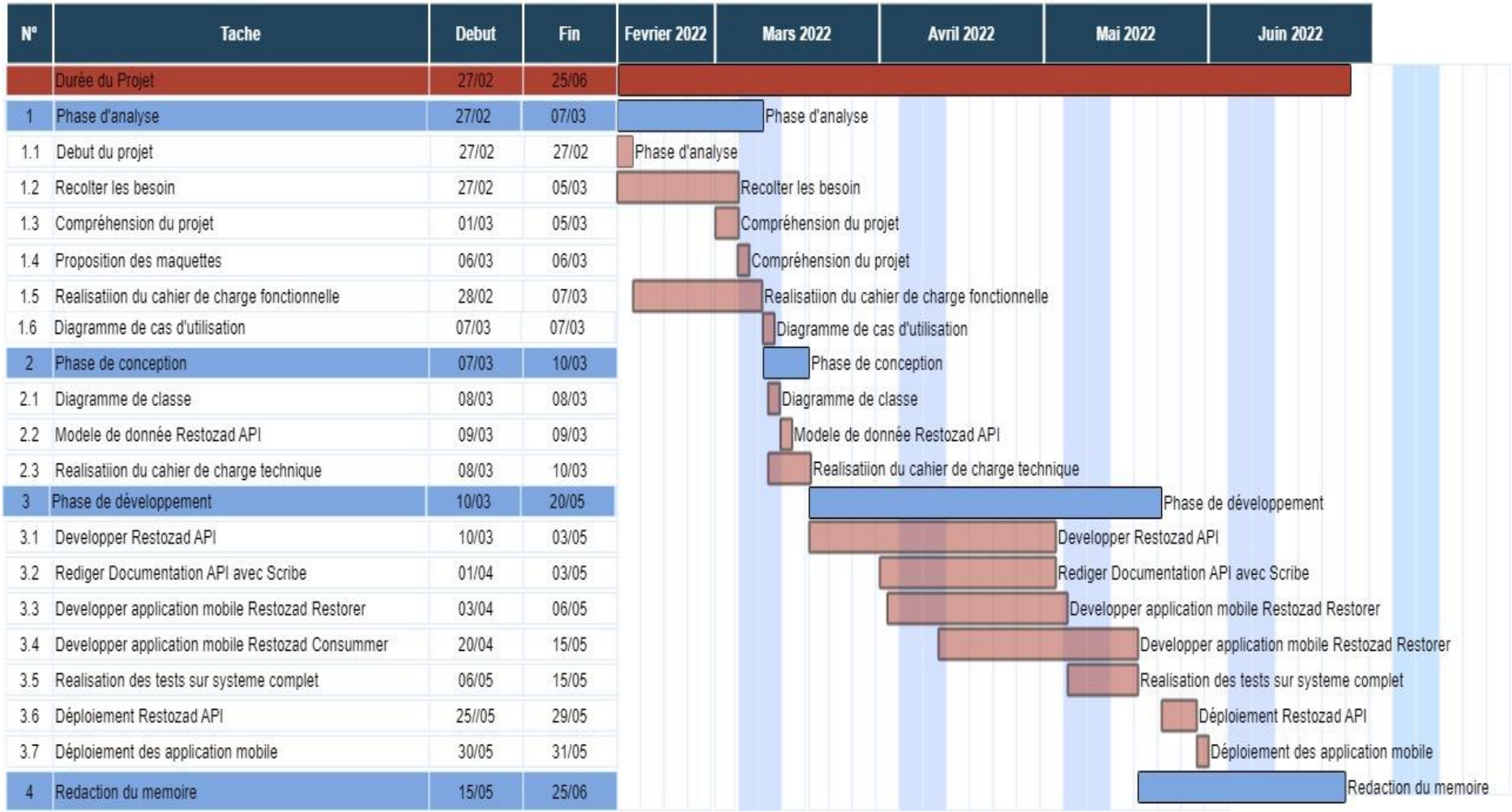

Figure 3-1 : Processus de réalisation

3.3. Spécification des exigences :

3.3.1. Identification des acteurs :

Les acteurs ci-dessus sont ceux qui interagissent avec le système :

- **Consommateur :**

  Celui qui passe des commandes sur l'application en choisissant des plats.

- **Restaurateur :**

  Cet acteur a la responsabilité de renseigner son restaurant en fournissant toutes ses informations nécessaires, il reçoit en temps réel les différentes commandes sollicitées par les consommateurs.

3.3.2. Spécification des besoins fonctionnels :

- **Besoin du consommateur :**

- Consulter les restaurants sur une carte Google Map Intégrée avec l'application.
- Consulter les menus des restaurants et le barème de prix des différents plats.
- Consulter les suppléments des plats.
- Consulter les informations relatives au restaurant.
- Sélectionner différents plats avec ou sans supplément dans un restaurant.
- Passer une commande après avoir sélectionné des plats avec ou sans supplément et en remplissant les informations nécessaires pour passer une commande (préciser adresse de livraison, etc.).
- Consulter les informations de la commande.
- Annulée une commande avant la prise en charge par le restaurant.
- Créer un compte (manuel, Google, Facebook) et remplir les informations personnelles.
- Consulter l'historique des commandes.
- Préciser des détails par rapport au plat commandé (sans salade, avec sauce, etc.).
- Recevoir une notification après la mise à jour d'une commande par le restaurateur.

- **Besoin du restaurateur :**

- Créer un compte (manuel, Google, Facebook) et remplir les informations personnelles.
- Gérer son ou ses restaurants (ajout, modification, suppression).

- Gérer les horaires d'ouverture fermeture d'un restaurant (ajout, modification, suppression).
- Gérer les menus du restaurant (ajout, modification, suppression).
- Gérer les plats d'un menu (ajout, modification, suppression).
- Gérer les suppléments d'un plat (ajout, modification, suppression).
- Recevoir une notification dès que la commande est prête.
- Gérer les commandes (renseigner le délai d'attente pour une nouvelle commande, annulée une commande).
- Consulter les détails d'une commande.
- Consulter l'historique des commandes.

3.3.3. Spécification des besoins non-fonctionnels :

Les besoins non-fonctionnels spécifient les propriétés du système telles que les contraintes d'environnement et d'implémentation, la performance, la maintenance, la sécurité, l'extensibilité et la flexibilité.

- **Contraintes ergonomiques :**

Les contraintes ergonomiques sont les contraintes liées à l'adaptation entre les fonctionnalités de l'application, leurs interfaces et leur utilisation, elles doivent obéir aux contraintes suivantes :

- Permettre un accès rapide à l'information.
- L'interface doit être simple et compréhensible.
- L'organisation des rubriques, des onglets, (etc.) doivent être cohérentes.
- L'application doit guider le client ou le visiteur pour avoir sa demande, en d'autres termes elle doit être développée avec un langage compréhensible par l'utilisateur, et présenter les informations d'une façon simple et claire, faire apparaître les choix ou les saisies du client.

- **Contraintes techniques :**

Toute l'interface de l'application doit être homogène, les différentes pages doivent suivre le même modèle de représentation (couleurs, images, textes défilants, etc.). Le code doit être extensible et maintenable pour faciliter toute opération d'amélioration ou d'optimisation en utilisant des patrons de conception.

- **Contraintes de matériel :**

L'application sera installée sur un téléphone mobile Android et iOS.

- **Contraintes de déploiement :**

Les applications clients doivent être téléchargeables à partir du Play Store pour Android et de l'Apple Store pour iOS.

- **Sécurité du SCL :**

La sécurité du SCL est importante pour le consommateur et le restaurateur. Elle assure que les commandes créées sont valides et permettent aux restaurateurs de ne pas se soucier de détails (calcul du prix total, gérer la disponibilité des plats et suppléments en temps réel, etc.) ce qui donne une intégrité des données et rassure le client sur la fiabilité des commandes en ligne.

- **Intégrité :** L'intégrité des données fait référence à la fiabilité et à la crédibilité des données durant tout leur cycle de vie.
- **Fiabilité :** Assurer que les données sont de bonne qualité en : empêchant les erreurs de saisie, privilégiant le recours aux API, triant les données collectées.
- **Disponibilité :** Assurer que les données sont disponibles à tout moment, et sont uniquement accessibles par les utilisateurs qui ont les droits d'accès nécessaires.
- **Confidentialité :** Les données doivent être stockées en toute sécurité et transitent sur un réseau sécurisé et chiffré.

3.3.4. Diagramme de cas d'utilisation :

Les diagrammes de cas d'utilisation sont des diagrammes UML utilisés pour une représentation du comportement fonctionnel d'un système logiciel. Ils sont utiles pour des présentations auprès de la direction ou des acteurs d'un projet [13].

- **Cas d'utilisation du consommateur :**

  Le consommateur doit d'abord s'authentifier (manuellement, avec Google, avec Facebook) sur le système, pour pouvoir consulter (restaurant, menu, plat, supplément, commande, liste des commandes), passer une commande en choisissant des plats en ajoutant éventuellement des suppléments. Après confirmation il sera soit notifié du délai d'attente pour recevoir sa commande, ou d'une annulation de sa commande par le restaurant. Le

consommateur peut aussi annuler sa commande avant la prise en charge par le restaurateur. La **Figure 3-2** montre les cas d'utilisation du consommateur.

- **Cas d'utilisation du restaurateur :**

  Le restaurateur doit d'abord s'authentifier (manuellement, avec Google, avec Facebook) sur le système, pour pouvoir gérer les restaurants, les menus, les plats et leur supplément (ajout, modification et suppression), afin que le consommateur puisse les consulter et passer une commande en ligne. Il a aussi la capacité de gérer des commandes par (la communication des délais d'attente, l'annulation d'une commande pas encore prise en charge), de consulter liste des commandes, de recevoir toute notification de nouvelle commande ou d'annulation d'une commande par le consommateur. Il a en plus la capacité de mettre à jour la disponibilité du restaurant, de même mettre à jour la disponibilité des plats et suppléments. La **Figure 3-2** montre les cas d'utilisation du restaurateur.

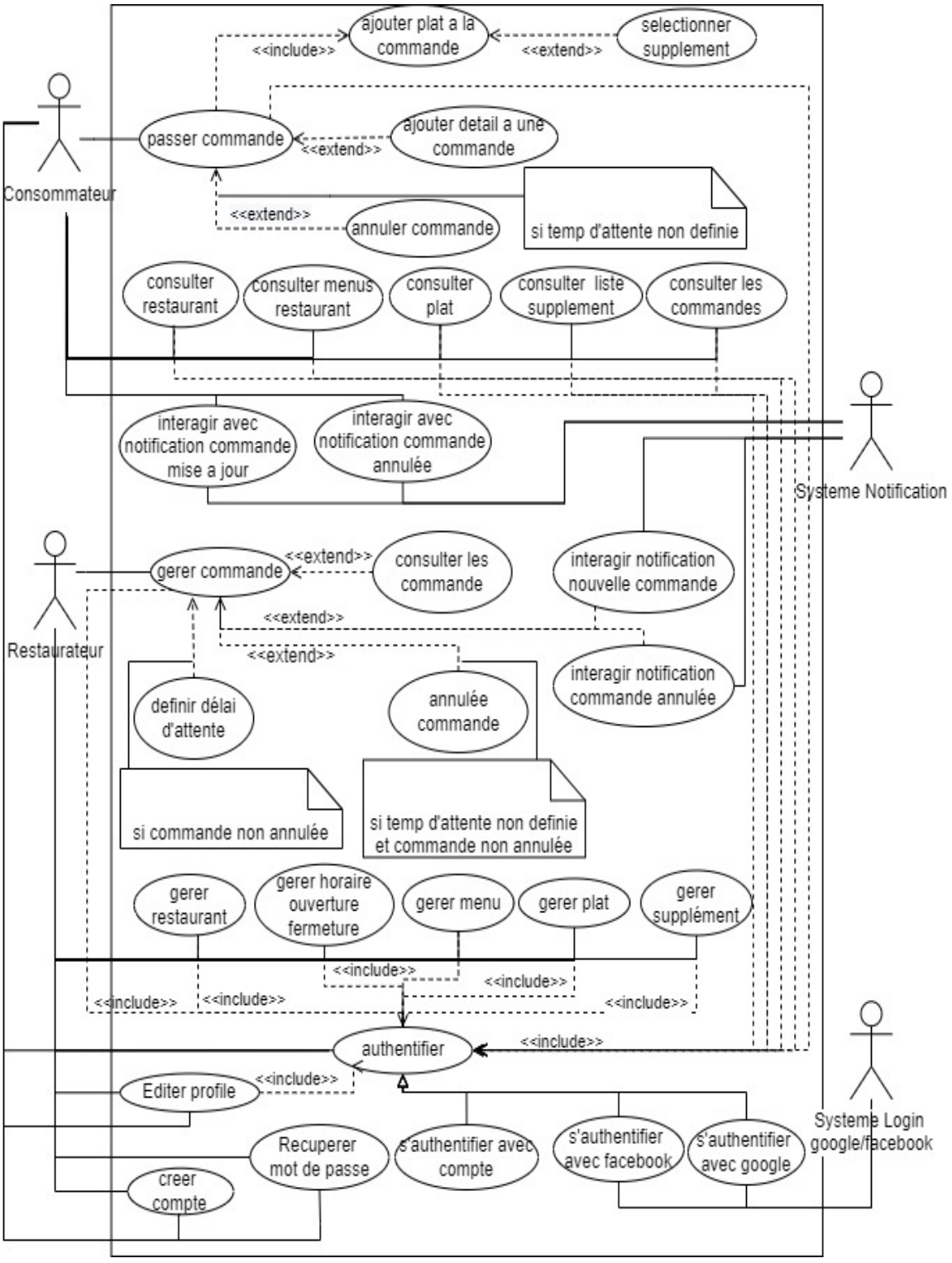

Figure 3-2 : Cas d'utilisation du restaurateur et consommateur

3.3.5. Diagramme de séquence :

On présente dans la **Figure 3-3** le diagramme de séquence pour le chargement de la liste des commandes d'un restaurant lors de la connexion d'un restaurateur sur l'application mobile RESTOZAD Restorer.

Dans le diagramme suivant le restaurateur se connecte sur l'application mobile avec l'email et password, après vérification le système génère un Token unique pour l'utilisation lors de la communication avec l'API. Une fois sur l'application les restaurants du restaurateur sont chargés (depuis le cache sinon la BDD), dès que le restaurant est sélectionné une requête et envoyée à l'API pour récupérer la liste des commandes.

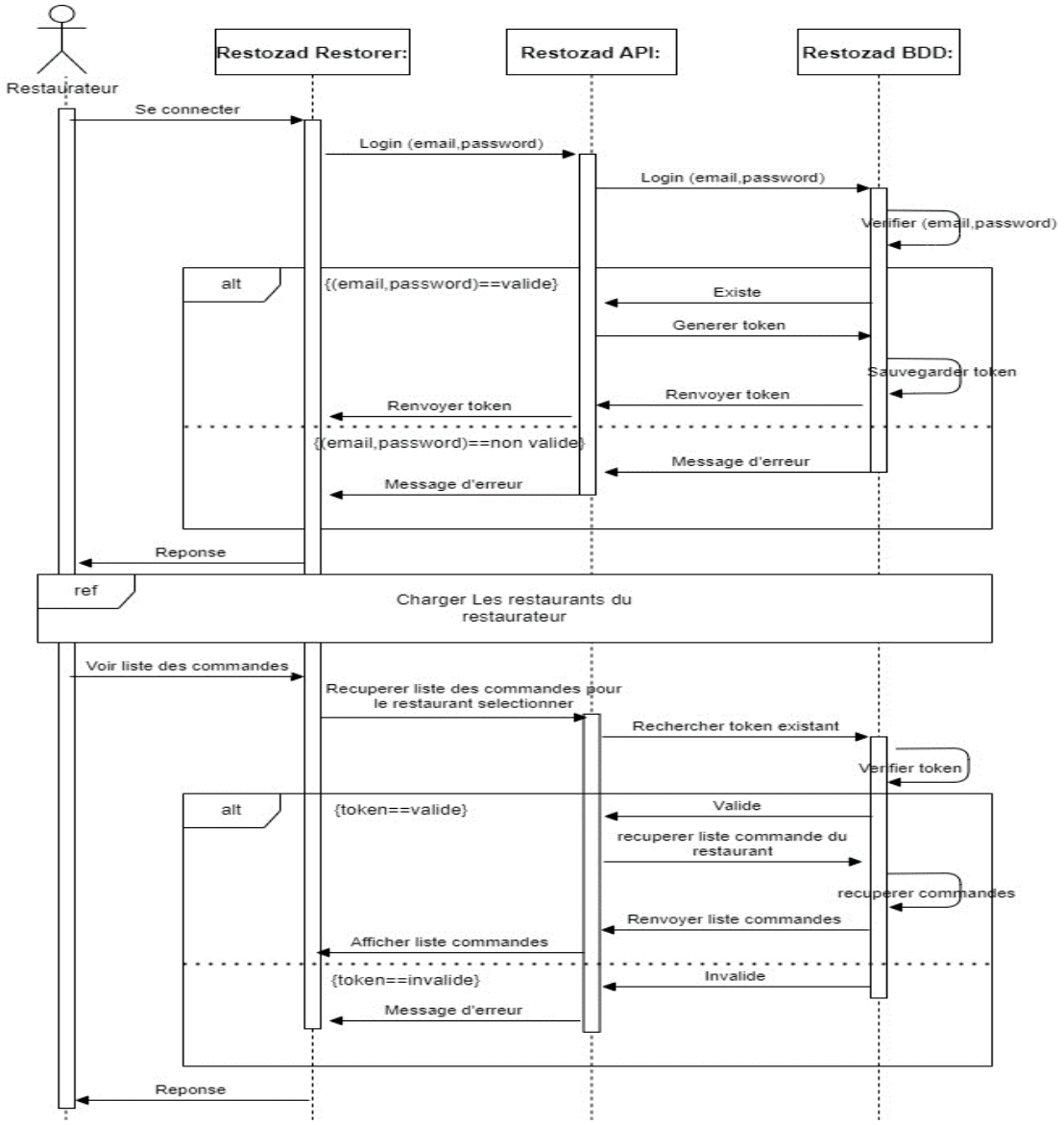

Figure 3-3 : Diagramme de séquence pour la liste des commandes

3.3.6. Public Cible :

Deux types de public sont visés, un public professionnel c'est généralement des restaurateurs qui souhaitent gérer les commandes de leurs plats en ligne sur une application mobile, et l'autre public client/consommateur qui passera des commandes auprès de restaurants.

3.3.7. Maquette :

Un ensemble de maquettes a été représenté pour les deux applications, qui met en évidence les différents besoins. Pour ce faire, nous nous sommes inspirés d'applications déjà existantes afin d'avoir des maquettes assez réalistes et proches du design final.

- **Maquette, le consommateur/restaurateur n'est pas authentifié :**

La **Figure 3-4** est une maquette qui montre les interfaces et interactions possibles pour consommateur/restaurateur non-connectés sur notre application ceci pour plusieurs vues, exemple : home, création de compte, connexion Google, etc.

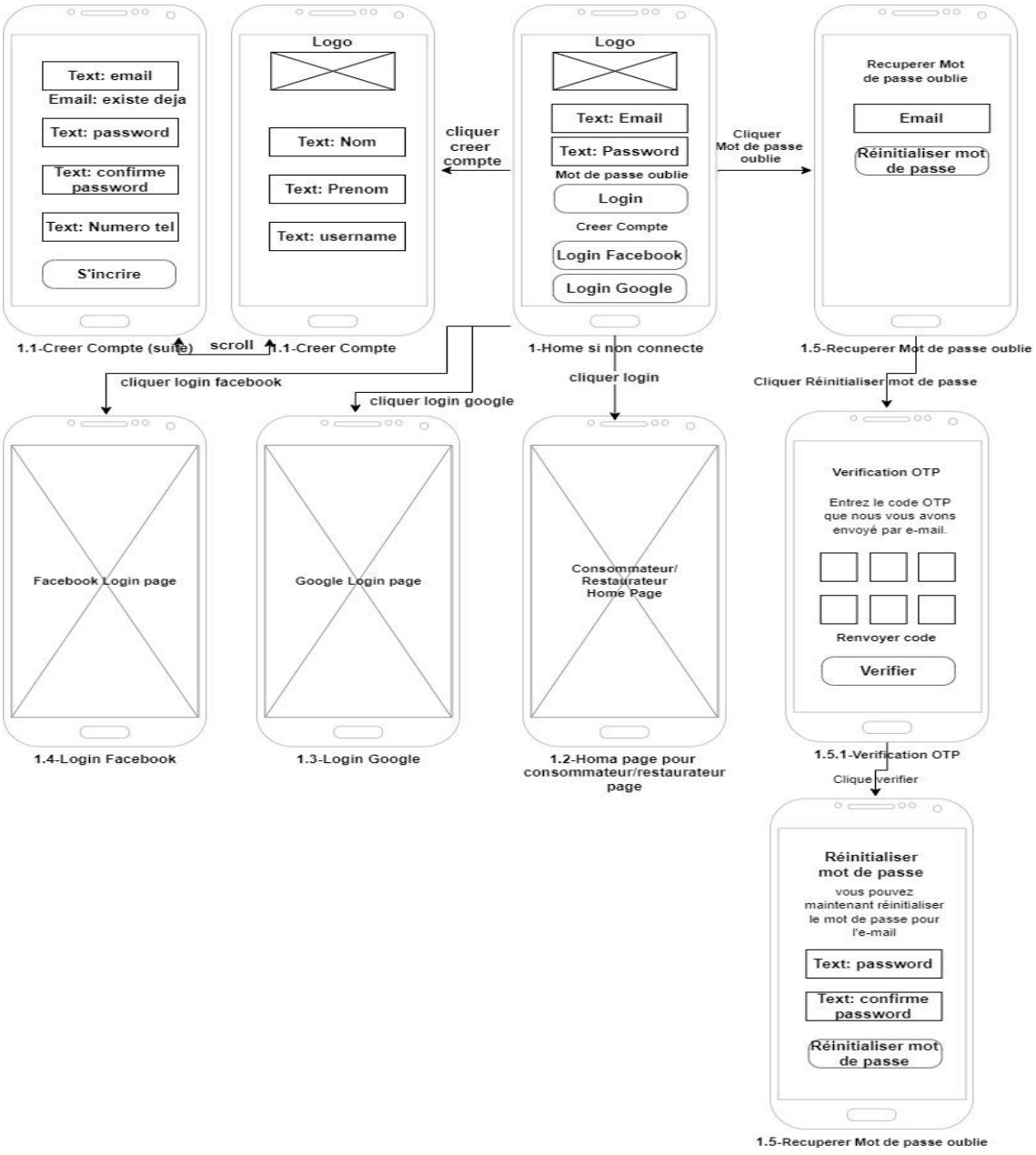

Figure 3-4 : Maquette pour utilisateur non-authentifié

- **Maquette, le consommateur est authentifié :**

La **Figure 3-5** quant à elle présente les interfaces et interactions possibles mais pour un consommateur connecté sur l'application et cela pour plusieurs vues, exemple : map, détail restaurant, détail plat, panier, liste des commandes, etc.

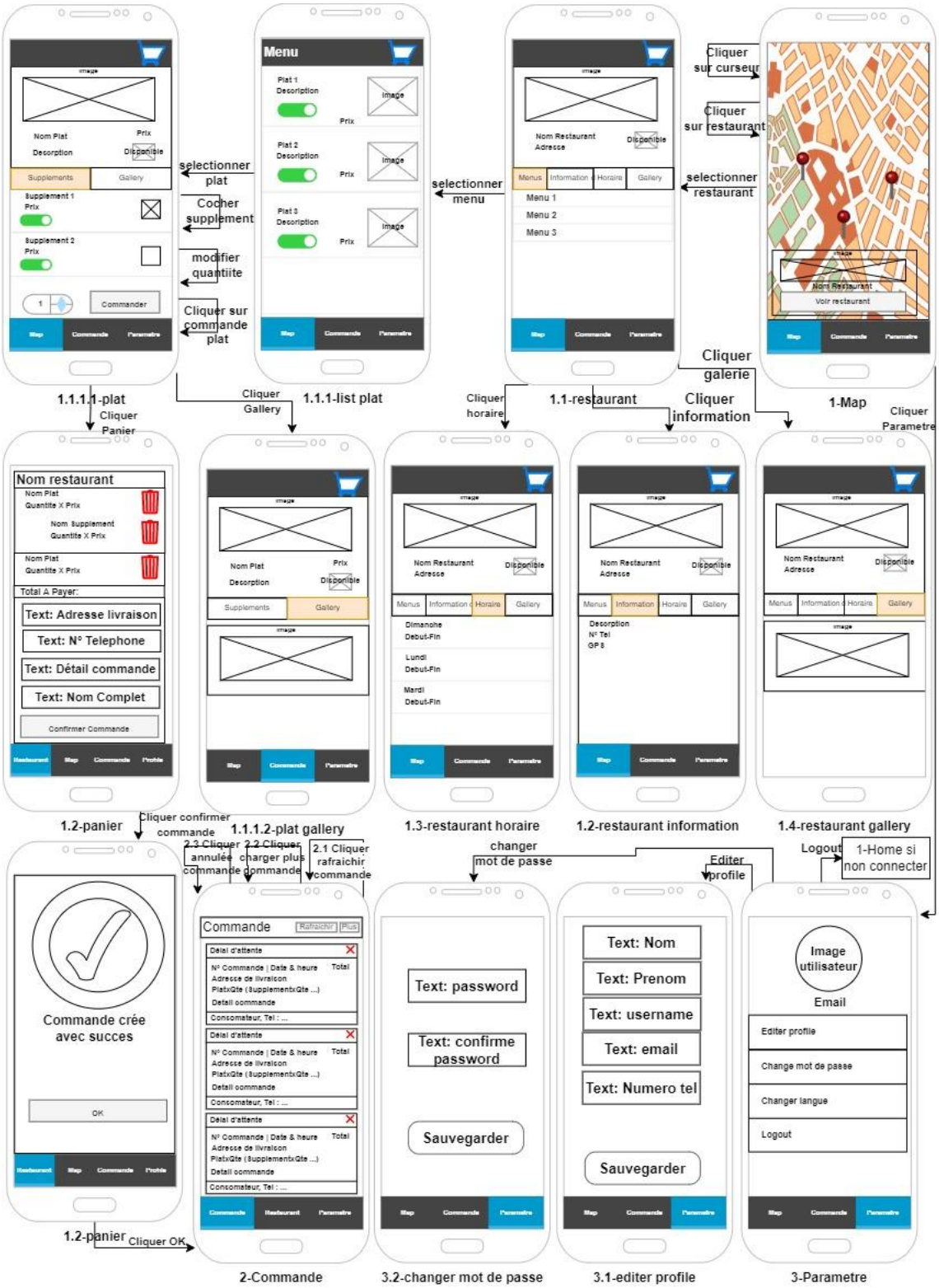

Figure 3-5 : Maquette pour consommateur authentifié

- **Maquette, le restaurateur est authentifié :**

Ici la **Figure 3-6** présente les interfaces et interactions possibles mais pour un restaurateur connecté sur l'application toujours pour plusieurs vues, comme : gestion des commandes, gestion des plats, gestion des suppléments, etc.

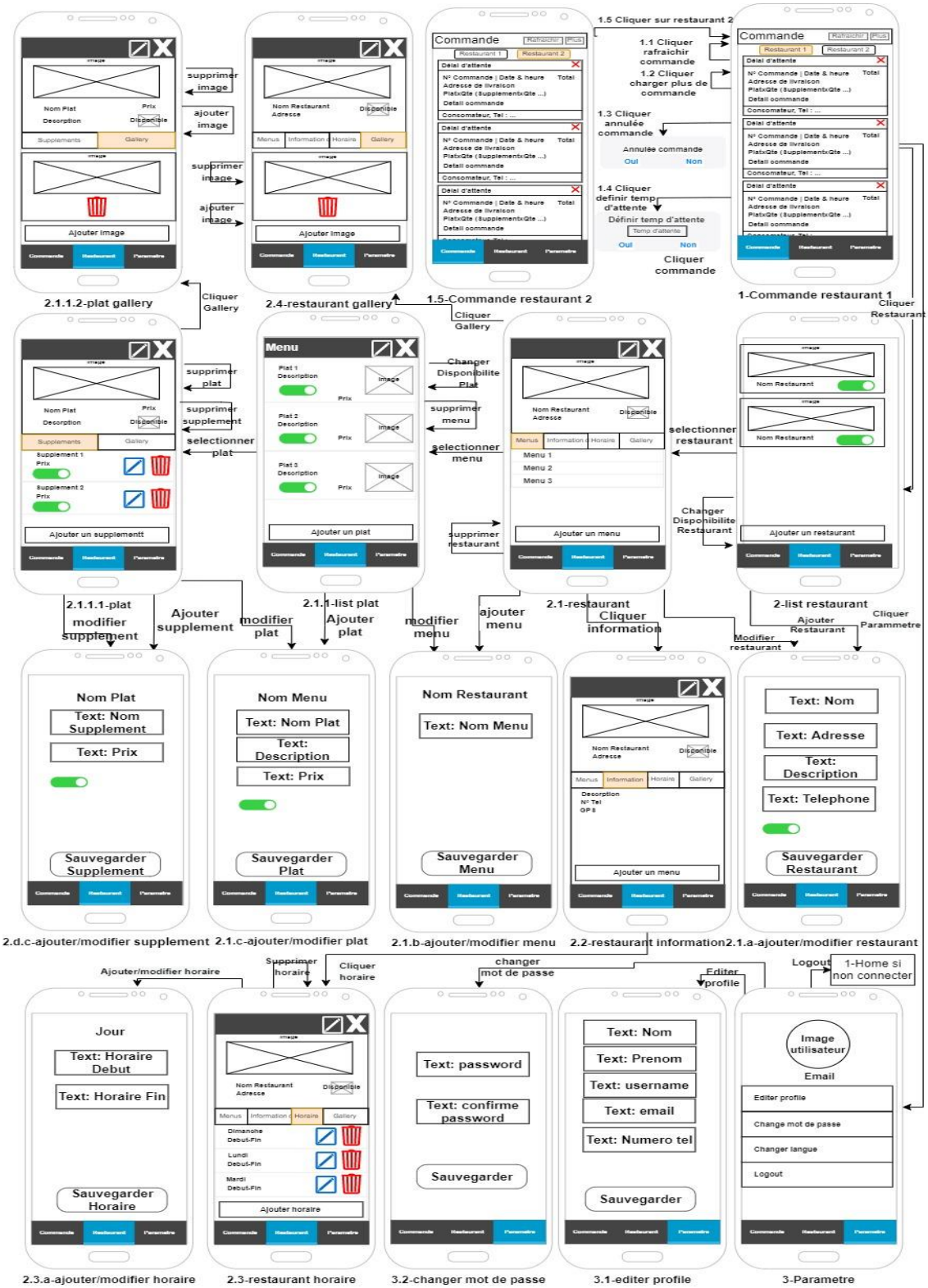

Figure 3-6 : Maquette pour restaurateur authentifié

3.4. Conception :

Ici, la réalisation des diagrammes qui présentent la conception du SCL avec recours à l'outil de conception UML, suivi de l'architecture globale du système.

3.4.1. Architecture système :

On préconise de créer une architecture qui sera composée de deux applications, une pour les consommateurs l'autre pour les restaurateurs, d'un service Backend et d'une base de données SQL.

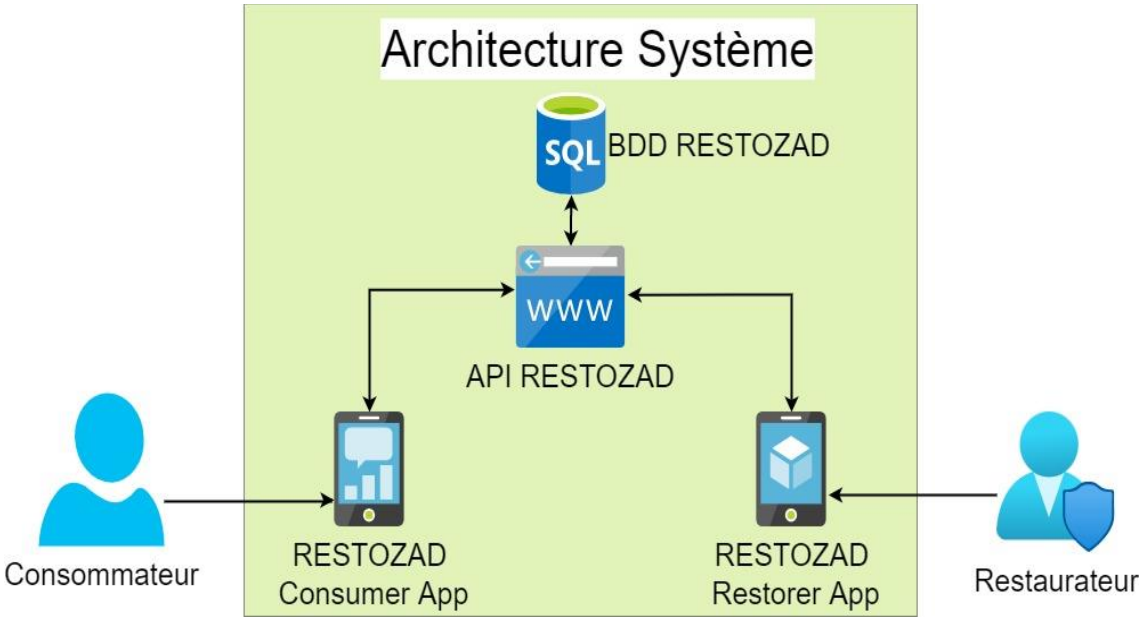

Figure 3-7 : Architecture système

La **Figure 3-7**, montre la répartition de l'architecture du projet :

- **RESTOZAD API :** contient toute la logique métier de notre système, les deux applications mobiles communiqueront avec cette API.
- **RESTOZAD Consumer App :** c'est l'application pour le consommateur, elle va lui permettre de commander des plats auprès de restaurants en ligne.
- **RESTOZAD Restorer App :** c'est **l'**application pour gérer les commandes faites pour les consommateurs, avec la gestion du restaurant (menu, plat, suppléments).

3.4.2. Architecture modulaire :

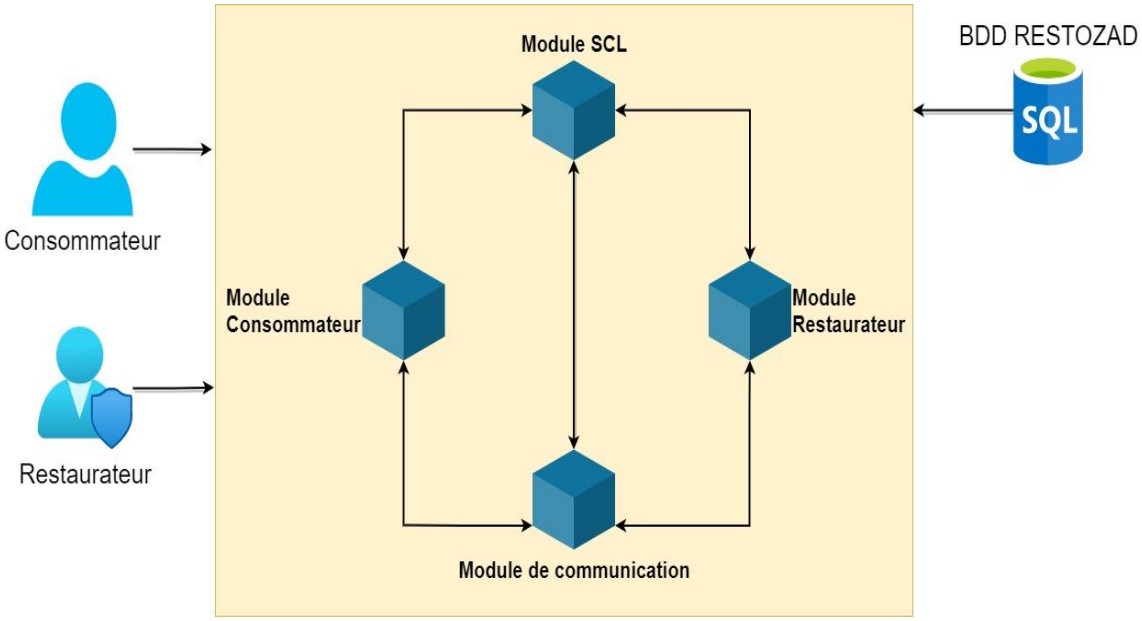

Figure 3-8 : Details du Module pour l'API

La **Figure 3-8** présente l'architecture modulaire de notre système, répartie en quatre modules intitulés :

- **Module SCL :** permet la gestion des restaurants, menus, plats, suppléments commande.
- **Module Consommateur :** gestion et administration des droits d'accès au module SCL pour le consommateur (compte consommateur, profil, etc.).
- **Module Restaurateur :** gestion et administration des droits d'accès au module SCL pour le restaurateur (compte restaurateur, profil, etc.).
- **Module de communication :** permet d'annoter les différentes rubriques du SCL.

3.4.3. Diagramme de classe :

Le diagramme de classe présent dans la **Figure 3-9** montre le model MVC utilisé pour l'API RESTOZAD pour la création d'une commande. En cliquant sur le bouton pour voir un plat les éléments du MVC interagissent :

- Le chargement de la vue (PlatScreen) montre tous les éléments graphique (texte, image, couleur), elle écoute les interactions faites avec ces éléments et affiche le plat avec sa liste de supplément.

- Le contrôleur (PlatControlleur) charge les éléments affichés sur la vue et gère les interactions de la vue actuelle, il permet de communiquer avec les autres vues et contrôleurs.
- Le modèle (PlatModel) représente l'entité avec ses différentes méthodes (all, findById, etc) qui est utilisée par le contrôleur et la vue.

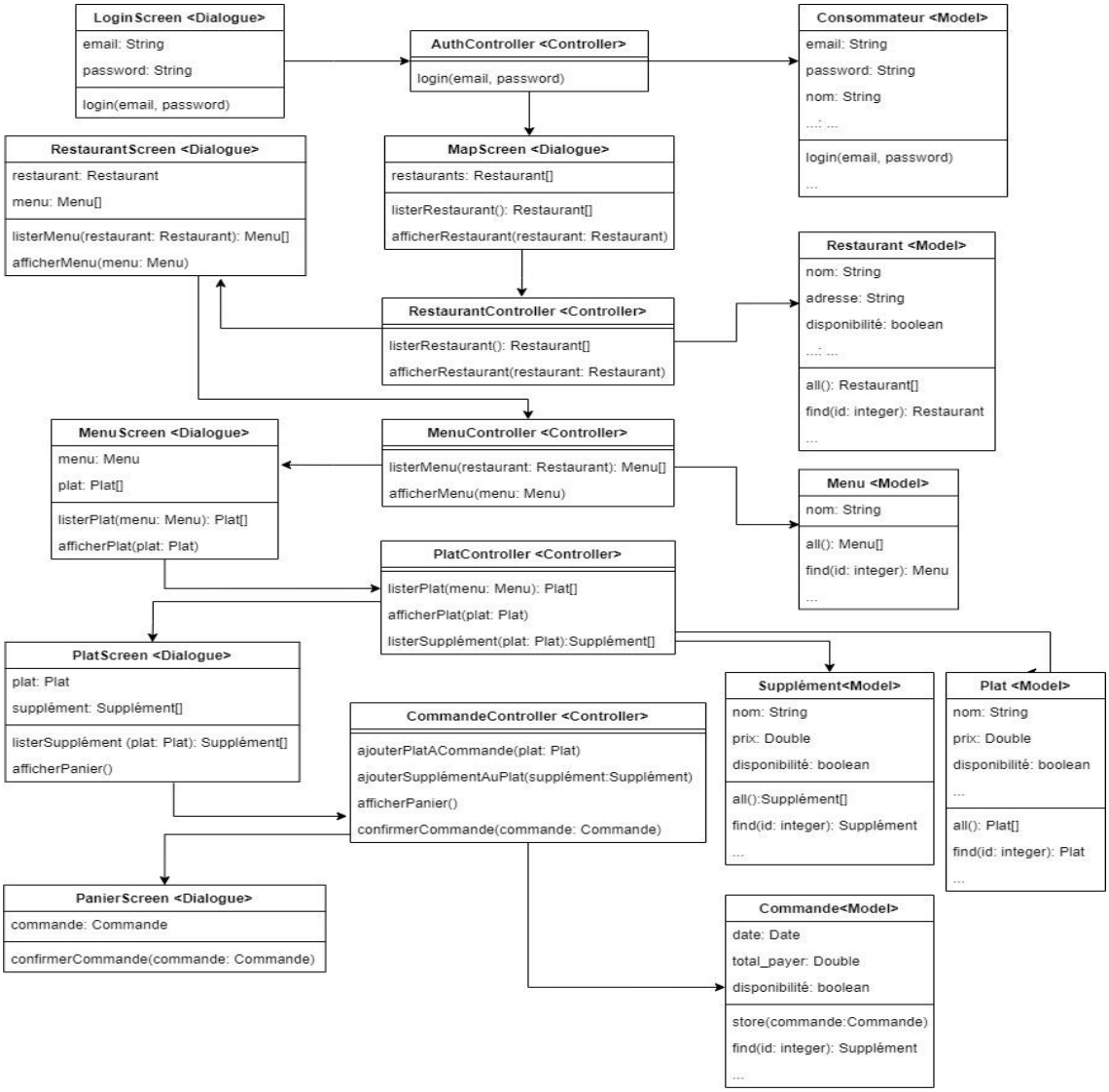

Figure 3-9 : Diagramme de classe participante pour la création de commande

3.4.4. Modèle de données :

On présente dans la **Figure *3-10*** la manière dont nos données sont structurées et représentées par des tables dans notre base de données SQL. Elles sont reparties en tables qui possèdent des attributs nommées et typées avec une clé primaire et des clés étrangères. La visualisation des interrelations entre les tables par le référencement d'un attribut d'une table pour une autre. Ce modelé de données est pour l'ensemble de notre système. Le modelé de donnée nous permet de comprendre qu'un menu contient un ou

plusieurs plats sachant que chaque plat a un nom, une description, un prix, une disponibilité, un numéro ID, appartient à un seul menu, et que chaque plat lui aussi peut avoir plusieurs images.

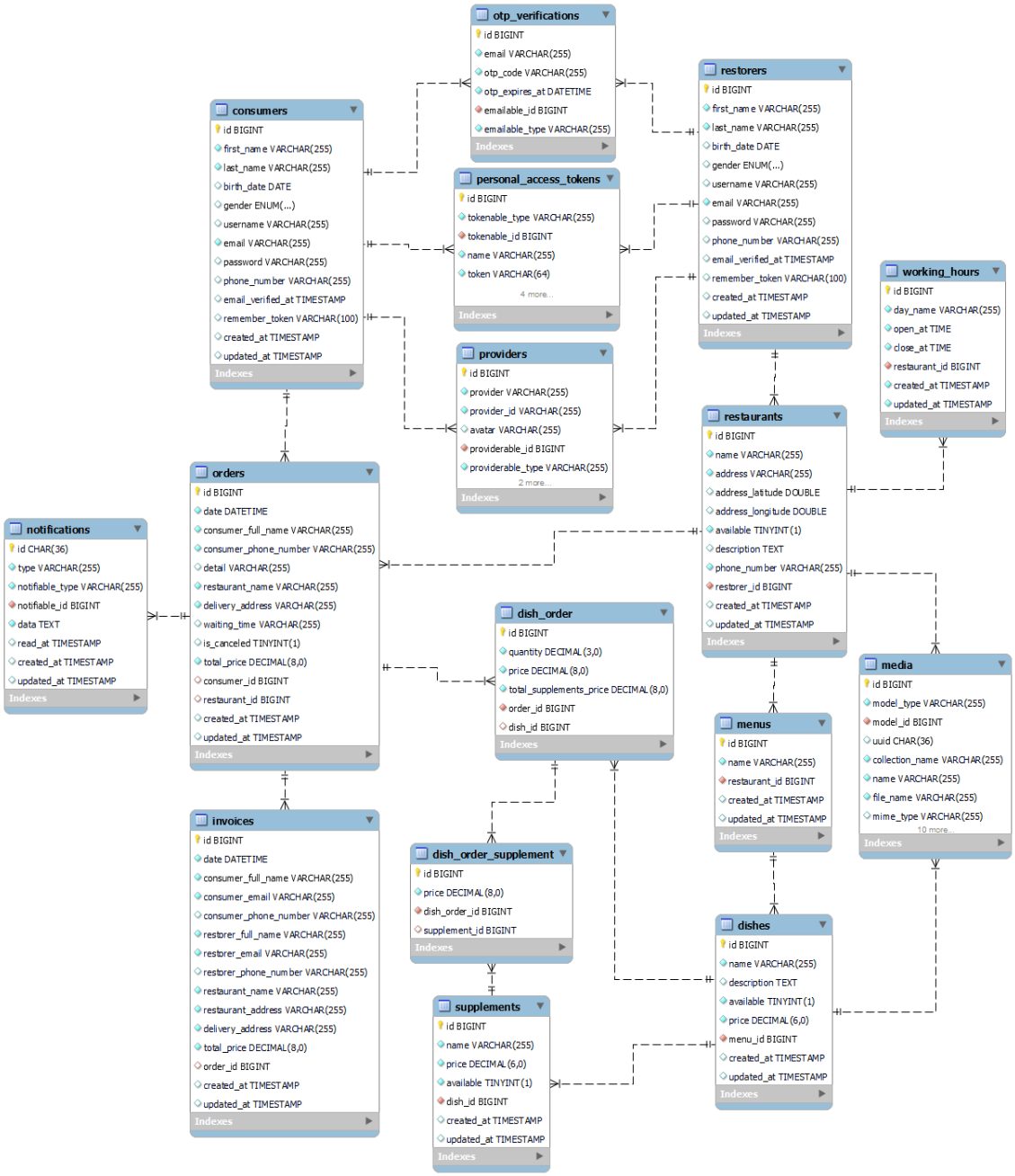

Figure 3-10 : Modèle de données SCL

3.5. Implémentation :

Ce paragraphe est dédié aux technologies et outils utilisés pour l'implémentation du SCL, il sera suivi d'une explication du Backend, Frontend et du système existant utilisé. En terminant par la présentation de l'ensemble du système tout en évoquant les problèmes rencontrés et la sécurité de nos applications.

3.5.1. Outils utilisés :

- **Environnement de développement :**

  Laragon : est un environnement de développement web qui regroupe les serveurs (Apache HTTP Server, MySQL), il apporte également des outils pour gérer une application web (PHP, NodeJS, etc.) Cet outil est similaire aux logiciels XAMPP et WAMP [14].

  Visual Studio Code : est un éditeur de code extensible pour Windows, Linux et macOS1.Les fonctionnalités incluent la prise en charge du débogage, la mise en évidence de la syntaxe, la complétion intelligente du code, (etc.) [15].

  Android Studio : est un environnement de développement pour développer des applications mobiles Android. Il est basé sur IntelliJ IDEA [16].

- **Serveur web :**

  Apache : est un logiciel de serveur web gratuit et open-source qui alimente environ 46% des sites web à travers le monde. Il permet aux propriétaires de sites web de servir du contenu sur le web d'une manière sécurisée [17].

- **Administration de base de données :**

  Heidi SQL : est un outil d'administration de base de données qui possède un éditeur SQL et un constructeur de requête. Il a été développé et optimisé pour être utilisé avec le SGBD relationnel MySQL [18].

- **Testeur d'API :**

  Postman : est un logiciel qui aide les développeurs et les entreprises à développer des flux de travail API. C'est efficace sur Chrome pour tester, développer, documenter des API et créer des requêtes complexes, remonter le temps et visualiser les résultats de manière magnifique [19].

  Thunder Client : est une extension client API légère pour Visual Studio Code avec un design simple et épuré, représentant une alternative à Postman [20].

- **Contrôleur de version :**

  Git : est un système de contrôle de version, il s'agit d'un outil de développement qui aide une équipe de développeurs à gérer les changements apportés au code source au fil du temps, en gardent une trace de chaque changement apporté [21].

  GitHub : est un logiciel très utilisé, généralement utilisé pour le contrôle de version. Il est utile lorsque plusieurs personnes travaillent sur un projet. Lorsqu'une équipe de développeurs souhaite créer un site Web, tout le monde peut mettre à jour les codes simultanément tout en travaillant sur le projet [22].

- **Dessin graphique, navigateur, émulateur Android :**

  Draw.io : est un logiciel de dessin graphique multi plateforme utilisée pour créer des diagrammes tels que des organigrammes, des structures filaires, des diagrammes UML, des organigrammes et des diagrammes de réseau [23].

  Chrome Developers : Google Chrome pour les développeurs a été conçu pour le Web ouvert, tester des API avancées de plates-formes Web et des outils pour les développeurs mis à disposition [24].

  AVD Manager : le gestionnaire AVD fournit une interface utilisateur graphique dans laquelle il créait et gère des appareils virtuels Android (AVD), qui sont requis par l'émulateur Android [25].

3.5.2. Technologies utilisées pour Backend :

- **Framework et langage de programmation :**

  PHP : PHP Hypertext Preprocessor, est un langage de programmation libre, principalement utilisé pour produire des pages web dynamiques via un serveur HTTP. PHP est un langage impératif orienté objet [26].

  Laravel : est un Framework web écrit en PHP respectant le principe modèle-vue-contrôleur et entièrement développé en programmation orientée objet [27].

  Composer : est un gestionnaire de dépendances libre écrit en PHP. Il permet à ses utilisateurs de déclarer et d'installer des bibliothèques dans un projet [28].

- **Déboguer d'application et générateur de documentation :**

  Laravel Telescope : Telescope est un assistant pour déboguer une application Laravel. Il donne accès à une foule d'informations sur les requêtes qui entrent dans l'application, les exceptions, les requêtes à la base de données, (etc.) [29].

  Scribe documentation : Scribe aide à générer une documentation API pour les humains à partir d'une base de code Laravel/Lumen/Dingo [30].

- **Langage de requête :**

  SQL : (langage de requête structurée) est un langage informatique normalisé servant à exploiter des bases de données relationnelles. La partie langage de manipulation des données permet de rechercher, d'ajouter, de modifier ou de supprimer des données dans les bases de données relationnelles [31].

  Redis : est une structure de données en mémoire de stockage utilisé comme base de données, cache, courtier de messages et moteur de diffusion, il fait partie de la mouvance NoSQL et vise à fournir des performances plus élevées [32].

- **Automatisation des scripts :**

  Cron : est un programme qui permet aux utilisateurs des systèmes Unix d'exécuter automatiquement des scripts, des commandes ou des logiciels à une date et une heure spécifiée à l'avance, ou selon un cycle défini à l'avance [33].

- **API et style architecturale :**

  API : est un ensemble normalisé de classes, de méthodes, de fonctions et de constantes qui sert de façade par laquelle un logiciel offre des services à d'autres logiciels. Elle est offerte par une bibliothèque logicielle ou un service web, le plus souvent accompagnée d'une description [34].

  RESTFUL : une API est RESTful quand elle respecte le principe d'architecture REST. Ce principe d'architecture s'applique aux services web. La particularité principale de cette architecture est que la partie serveur (API) et la partie client communiquent sans que le client ne connaisse la structure et le contenu des informations stockées sur le serveur [35].

- **Sécurisation et vérification :**

  OAUTH : est un protocole libre qui permet d'autoriser un site web, un logiciel ou une application à utiliser l'API sécurisée d'un autre site web pour le compte d'un utilisateur. OAuth n'est pas un protocole d'authentification, mais de délégation d'autorisation [36].

  OTP : un mot de passe à usage unique siglé OTP est un mot de passe qui n'est valable que pour une session ou une transaction. Il permet de combler certaines lacunes associées aux traditionnels mots de passe statiques [37].

3.5.3. Technologies utilisées pour Frontend :

- **Framework et langage de programmation :**

  JSON : JavaScript Object Notation (JSON) est un format de données textuelles dérivé de la notation des objets du langage JavaScript. Il permet de représenter de l'information structurée comme le permet XML [38].

  Javascript : est un langage de programmation de scripts principalement employé dans les pages web interactives et à ce titre il est une partie essentielle des applications web [39].

  Typescript : est un langage de programmation libre et open-source qui a pour but d'améliorer et de sécuriser la production de code JavaScript. Il s'agit d'un sur-ensemble syntaxique strict de JavaScript. Le code Typescript est transcompilé en JavaScript, il et interprété par ou moteur JavaScript [40].

  React : est une bibliothèque JavaScript libre développée par Facebook depuis 2013. Le but principal de cette bibliothèque est de faciliter la création d'application web mono page, via la création de composants dépendant d'un état et générant une page HTML à chaque changement d'état [41].

  React Native : est un Framework d'applications mobiles, il est utilisé pour développer des applications pour Android et iOS en permettant aux développeurs d'utiliser React avec les fonctionnalités natives [42].

  Expo : c'est à la fois un Framework et une plateforme qui simplifient la création et le déploiement d'applications mobiles avec React Native. Expo embarque de nombreux outils utiles et des librairies natives pour React Native [43].

- **Consommateur d'API :**

  Axios : est un client HTTP basé sur les Promesses largement utilisé pour effectuer des appels d'API REST, Il facilite l'envoi de requêtes HTTP asynchrones aux points de terminaison REST et l'exécution d'opérations CRUD. Il peut être utilisé en JavaScript simple ou avec une bibliothèque [44].

- **Gestionnaire de paquets :**

  Yarn : est un système de packaging logiciel pour l'environnement d'exécution JavaScript Node.js. Yarn a été créé pour résoudre les problèmes de cohérence, de sécurité et de performances avec de grandes bases de code [45].

- **Notification :**

  Expo Notification : fournit une API pour récupérer les jetons de notification push et pour présenter, planifier, recevoir et répondre aux notifications [46].

  Push Notification : les notifications Push ressemblent à des SMS et à des alertes mobiles, mais elles n'atteignent que les utilisateurs qui ont installé l'application. Son objectif est de déclencher une session utilisateur ou d'apporter une information instantanée [47].

3.5.4. RESTOZAD API :

Notre API a été réalisée avec le Framework Laravel qui par défaut utilise essentiellement un modèle architectural (MVC) mais qui nous permet de l'adapter pour le développement d'une API. Le style architectural REST a été utilisé pour que l'API soit RESTFUL et ainsi avoir ces avantages (la séparation du client et du serveur, API Statless pour que chaque requête et réponse soit déterminée et compréhensible, la mise en cache du côté client, l'utilisation d'interface uniforme, les systèmes en couches, etc.).

L'utilisation et implémentation de certains patrons de conception déjà présents avec Laravel ou qu'on a dû implémenter nous-même. La combinaison de ces patrons a permis de réaliser un système bien structuré, résolvant des problématiques rencontrées durant le développement.

Quelques exemples de patrons :

- **Factory Pattern :** le patron de conception Factory est un patron de création qui utilise des méthodes de fabrique pour traiter le problème de la création d'objets en spécifiant leurs classes concrètes [48]. Ce pattern déjà existant dans Laravel a permis de simplifier la création d'objet durant le développement du code.
- **Repository Pattern :** le patron de conception repository dans une application Laravel est un lien entre les modèles et les contrôleurs, le modèle ne doit pas être responsable de la connexion ou de la récupération des données de la base de données [48].Laravel par défaut implémenté, ce pattern grâce à Eloquent qui est un ORM permet de mapper les tables de la base de données à des modèles éloquents particuliers.
- **Facade Pattern :** le patron de conception Facade fournit une interface combinée à un ensemble d'interfaces dans un sous-système, une façade définit une interface de niveau supérieur qui facilite l'utilisation du sous-système [48]. Ce patron est beaucoup utilisé dans les packages développés par Laravel ou par sa communauté pour les rendre utilisables pour les autres développeurs.
- **Service Pattern :** le patron de conception Service est un patron architectural, appliqué dans le cadre du paradigme de conception orienté service, qui vise à organiser les services, au sein d'un inventaire de services, en un ensemble de couches logiques [49]. Ce patron n'est pas utilisé par default dans Laravel mais son implémentation reste assez simple et très utile pour de grand projet, il permet d'avoir un code propre et très bien structuré avec une architecture orientée service.

En plus des patrons de conceptions et le respect des 5 principes de conception orientées objet SOLID qui permettent d'avoir des architectures logicielles plus compréhensibles (voir **Figure *3-11***), flexibles et maintenables. SOLID signifie :

- **S (single responsibility principle) :** il ne devrait jamais y avoir plus d'une raison pour qu'une classe change. Autrement dit, chaque classe ne devrait avoir qu'une seule responsabilité.
- **O (open/closed principle) :** les objets ou entités devraient être ouverts à l'extension mais fermés à la modification.
- **L (liskov substitution principle) :** les classes dérivées doivent pouvoir être remplacées par leurs classes de base, les classes dérivées doivent avoir la même

intention mais une implémentation différente. Cela peut être vu comme une conception par contrat.

- **I (interface segregation principle) :** un client ne doit jamais être forcé à installer une interface qu'il n'utilise pas et les clients ne doivent pas être forcés à dépendre de méthodes qu'ils n'utilisent pas.
- **D (dependency inversion principle) :** les entités doivent dépendre des abstractions, pas des implémentations. Il indique que le module de haut niveau ne doit pas dépendre du module de bas niveau, mais qu'ils doivent dépendre des abstractions.

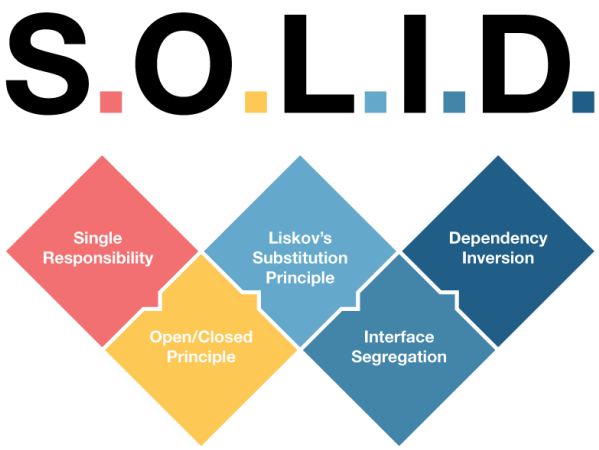

Figure 3-11 : S.O.L.I.D

- **Test et débogage de l'API :**

Lors du développement nous avons testé nos API avec Postman (voir **Figure *3-12***) et Thunder Client (voir **Figure *3-13***) qui permettent de visualiser la réponse, body, header et HTTP status code de la requête émise. En plus, les requêtes arrivantes sur l'application Laravel ont été déboguées grâce à Laravel Telescope (voir **Figure *3-14***) pour débogue requête, exceptions, log, requêtes de base de données, tâches en file d'attente, mail, notifications, opérations de cache, tâches planifiées.

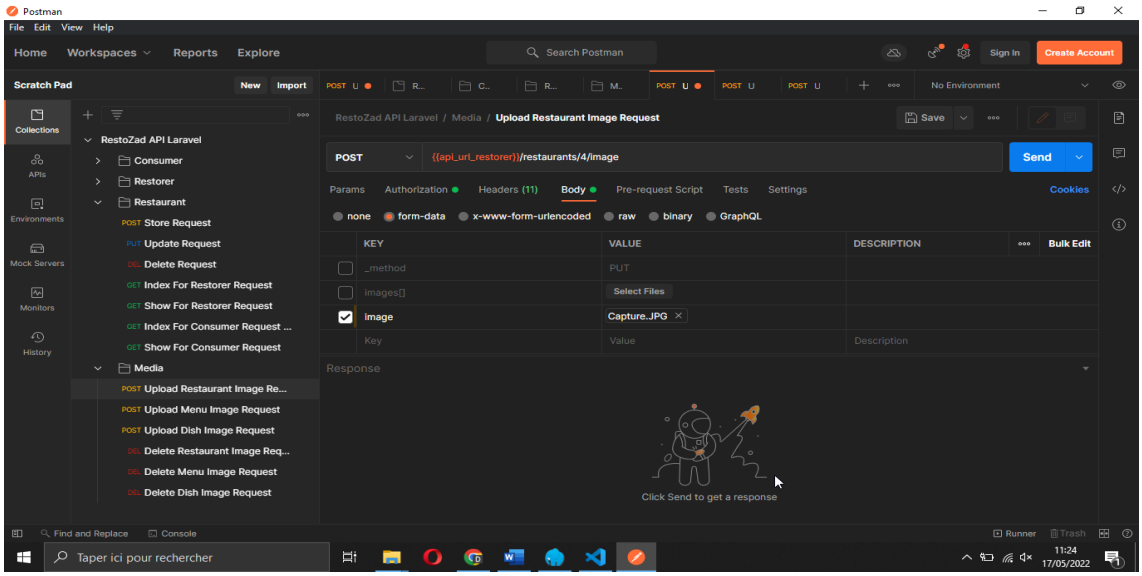

Figure 3-12 : Exemple Postman

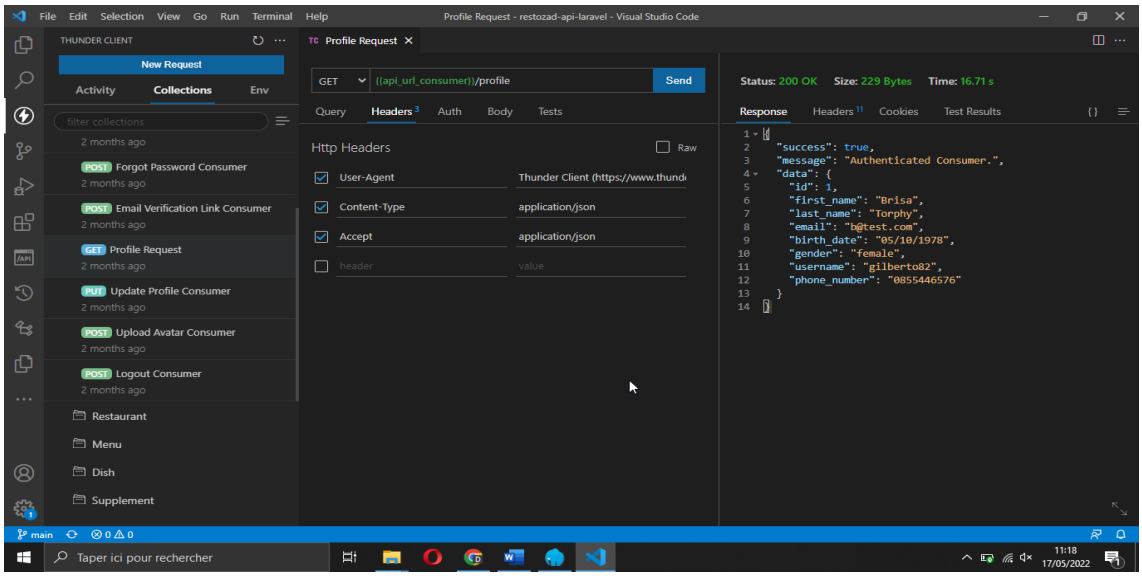

Figure 3-13 : Exemple Thunder Client API

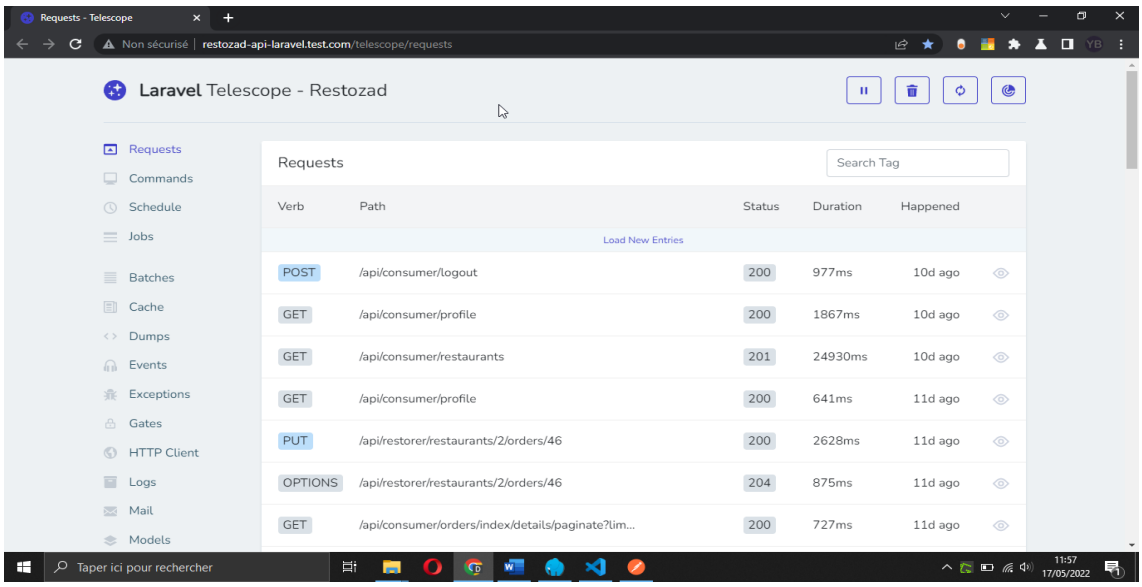

Figure 3-14 : Laravel Telescope pour RESTOZAD API

- **Documentation de l'API :**

En parallèle une documentation a été réalisée pour l'API avec Scribe (voir **Figure *3-15*** et **Figure *3-16***), chaque route possible et utilisable est documentée pour permettre aux utilisateurs d'avoir suffisamment d'informations à une utilisation optimale. La documentation contient :

- Nom : définit un nom pour la route.
- Description : donne une description pour cette route.
- Type de requête HTTP : precise le type: GET, POST, PUT, PATCH, DELETE.
- Code de réponse HTTP : indique le code de réponse, en cas de succès (200, 201, 202…), en cas d'erreur (400, 404, 401…).
- Réponse HTTP : retourne un message et une API Ressources en cas de succès, sinon un message est un objet contenant l'erreur ou l'exceptions 'affiche. Le retour dépend du type de la requête émise.
- Grouper Requête : groupe les requêtes dans des groupes.
- Authentification status : précise si la requête nécessite l'authentification (Token, BearerToken…).
- URL paramètres : précise paramètre URL nom, type, description et exemple.
- Query paramètres : précise paramètre Query nom, type, description, paramètre exigé ou pas et exemple.
- Body paramètres : précise paramètre Body nom, type, description, exige ou pas.
- Règles de validation : fixe une description pour la validation de Query ou Body paramètres.

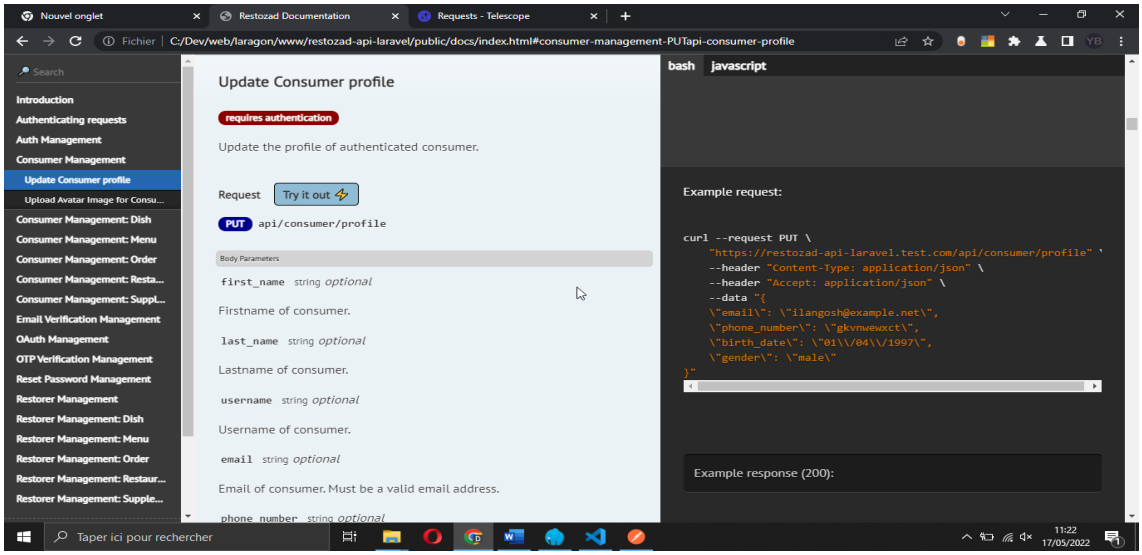

Figure 3-15 : Exemple 1 documentation RESTOZAD API

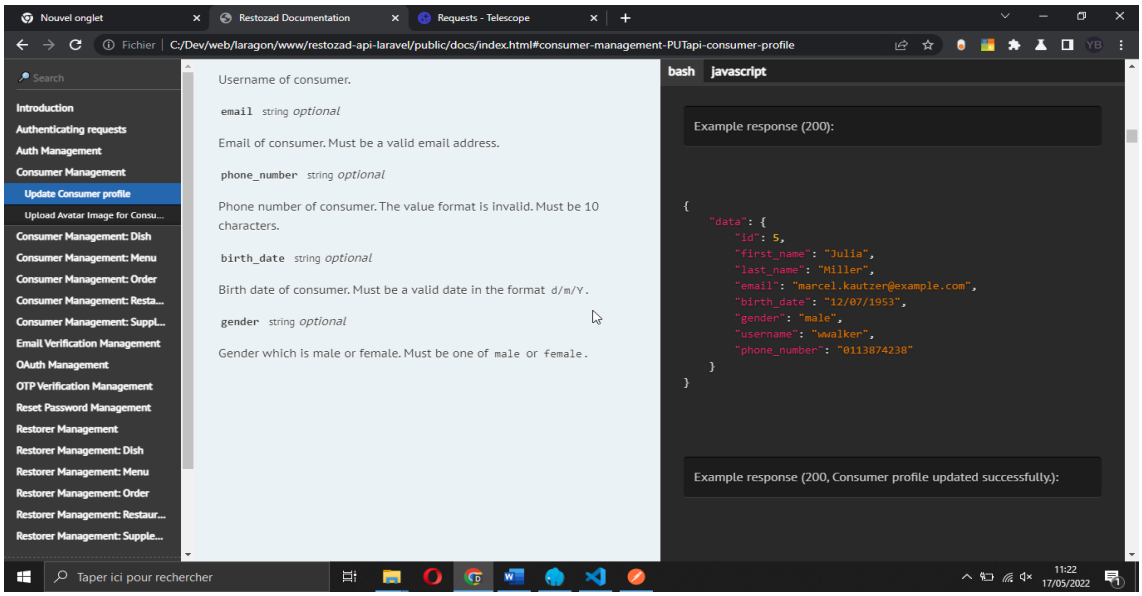

Figure 3-16 : Exemple 2 documentation RESTOZAD API

- **Localisation pour l'API :**

Pour répondre aux besoins du client, plusieurs langues ont été prises en charge dans notre application (Français, Anglais, Arabe). Ceci a été fait grâce à Laravel localisation qui permet de récupérer des chaînes de caractères dans différentes langues définies dans des fichiers spécifiques (JSON, PHP). La langue souhaitée est envoyée comme paramètre header dans une requête, ensuite la chaine de caractères est récupérée dans la langue souhaitée sinon dans la langue par défaut.

- **Cache Redis pour l'API :**

Un système de cache Redis a été utilisé pour le stockage temporaire des données afin d'accélérer le processus d'exécution des requêtes de base de données. Dans le cas d'une lecture, les données en cache (s'ils existent déjà) sont récupérées directement sinon elles sont récupérées depuis la BDD, pour les cas d'ajout, modification ou suppression le cache est vidé.

3.5.5. Nos applications mobiles :

Nos applications mobiles ont été réalisées avec le Framework React Native afin d'être disponibles pour Android et Ios. En règle générale le développement en React Native est fait en Javascript mais nous avons choisi d'utiliser Typescript. Ce dernier est un langage qui étend JavaScript en ajoutant des définitions de type pour ainsi avoir : un code plus robuste, l'auto-documentation grâce aux typages, ajouter une structure pour les grandes applications, simplifier débogage, (etc.).

Expo permet d'accélérer la création et le test d'application React Native, il fournit un ensemble d'outils qui simplifient le développement des composants de l'interface utilisateur et des services. Expo permet de publier un projet sur notre compte pour tester directement sur un smartphone ou uniquement en scannant le code QR. La **Figure *3-17*** montre nos deux applications publier et disponible sur notre compte Expo.

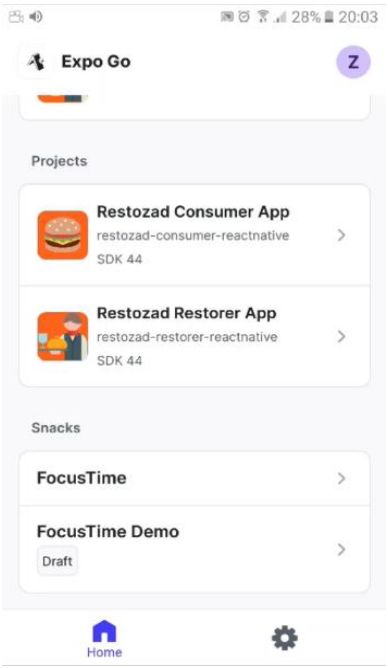

Figure 3-17 : EXPO Go projet publier

RESTOZAD Restorer et RESTOZAD Consumer vont communiquer avec l'API en utilisant le protocole HTTP, pour réaliser cette communication le recours à un consommateur d'API (Axios ou Fetch API) s'est avéré nécessaire, il va permettre de demander des données à une API sur le réseau. Les différents paramètres seront ajoutés dans le header des requêtes en utilisant un intercepteur de requêtes pour insérer BearerToken, Langage, (etc.). Les données reçues seront en JSON, ce qui facilitera leur lecture et leur utilisation dans React Native.

- **État des applications :**

On a choisi de gérer l'état de notre application avec Context API parce qu'elle fournit un moyen de transmettre des données via l'arborescence des composants sans avoir à transmettre manuellement les propriétés à chaque niveau [50]. Ceci nous permet de répondre à la complexité de passage des données via les propriétés entre plusieurs couches de composants.

- **Localisation des applications :**

Pour que les applications soient multilingues (Français, Anglais, Arabe) l'utilisation d'un système de localisation est alors nécessaire (React Native I18N). I18N est le processus d'adaptation d'une application pour travailler avec différentes langues et régions [51]. Les chaînes de caractères des différentes langues sont définies dans des fichiers JSON spécifiques. Pour changer de langue, l'application doit être redémarrée pour sa prise en charge. Dans le cas de la langue arabe, RTL sera activée pour les différents composants et styles utilisés, le redémarrage est alors obligatoire pour sa prise d'effet.

- **Stockage Local :**

Un système de stockage local a été utilisé pour enregistrer des données localement sur l'appareil de l'utilisateur. Notre choix s'est tourné vers SecureStore car il fournit un moyen de chiffrer et de stocker en toute sécurité les paires clé-valeur localement. Seules les données nécessaires pour notre application sont stockées (Token, langue, thème, (etc.).

3.5.6. Sécurité d'authentification :

On va parler de quelque systèmes implémentés dans notre projet qui nous ont permis de sécuriser notre système d'authentification, ils sont généralement implémentés de manière différente suivant la technologie utilisée :

1) Système d'authentification par Token :

L'authentification est un processus permettant à un système informatique de s'assurer de la légitimité de la demande d'accès faite par une entité (être humain ou un autre système), afin d'autoriser son accès à des ressources du système (système d'exploitation, réseaux, applications…) et ce conformément au paramétrage du contrôle d'accès [52].

Un Token d'authentification généralement appelé Bearer Token est un schéma d'authentification HTTP qui est utilisé pour accéder à une ressource à accès restreint. Le Token contient des données créées par le serveur et utilisées pour confirmer l'identité d'un utilisateur spécifique. Il doit être attaché à toute requête pour vérifier à chaque demande si l'utilisateur est autorisé à accéder à la ressource protégée sinon un code d'erreur HTTP (401) est retourné. Ce Token est généré par le serveur en réponse à une

demande de connexion et ne doit être utilisé que sur HTTPS (SSL) ceci pour des raisons de sécurité. Ci-dessous la **Figure *3-18*** présente l'authentification par token pour nos applications mobiles.

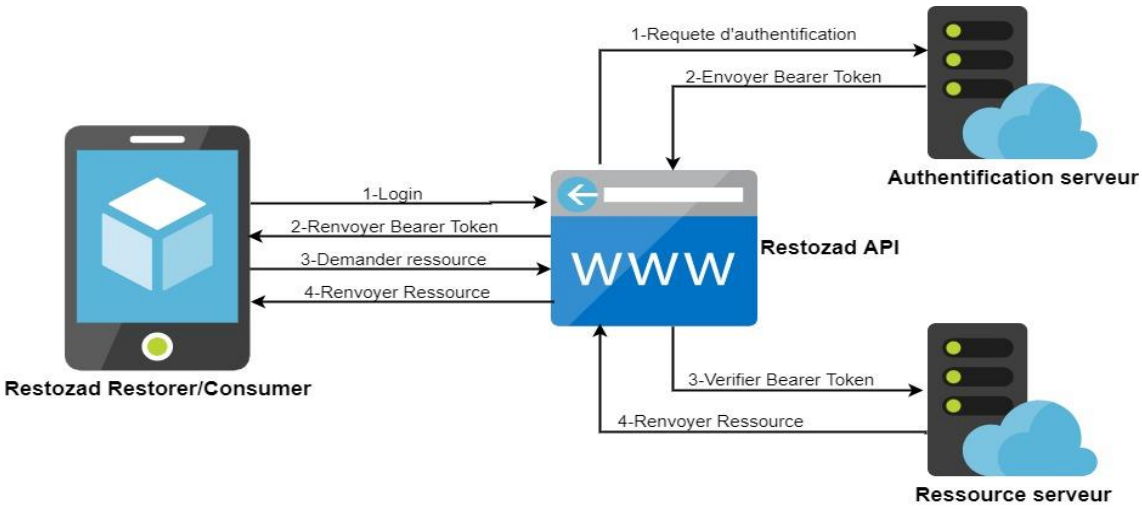

Figure 3-18 : Authentification par Token

Dans notre système c'est Laravel Sanctum qui est utilisé sur le Backend afin de générer un Bearer Token stocké pour l'utilisateur concerné, avec renvoi de Token comme réponse de succès. Les applications mobiles récupèrent alors le Token et l'attachent ensuite à chaque requête pour accéder aux ressources protégées.

Etapes pour avoir un Token :

- Créer un compte sur notre application.
- Se connecter par email grâce à un password.
- Un Token est généré sur le Backend et est sauvegardé, ensuite il est retourné vers l'application client pour le garder en cache ou le sauvegarder localement.
- A chaque requête envoyée le Token est récupéré (stockage local, cache) et attaché dans les headers en tant que « Authorization : Bearer ».
- Le serveur vérifie pour les routes qui ont besoin d'un Token qu'il est présent et valide dans le header.

Etapes pour supprimer un Token :

- Se déconnecter de l'application.
- Supprimer le Token qui est sauvegardé localement.
- Le serveur supprime le Token attribué à l'utilisateur.

2) Système d'authentification OAuth2 :

OAuth2 est utilisé pour permettre aux internautes d'autoriser des sites web ou des applications à accéder à leurs informations sur d'autres sites web, mais sans leur donner les mots de passe. La fonctionnalité la plus connue est celle de permettre à un utilisateur de se connecter avec son Compte Google, Facebook, etc. Donc sans avoir à passer par l'étape de création de compte habituel. Les implémentations les plus courantes d'OAuth utilisent l'un de ces Token ou les deux :

- Access Token : envoyé comme une clé API, il permet à l'application d'accéder aux données d'un utilisateur ; éventuellement, les access Token peuvent expirer.
- Refresh Token : faisant éventuellement partie d'un flux OAuth, les refresh Token récupèrent un nouveau access Token s'ils ont expirés.

Ci-dessous la **Figure *3-19*** présente l'authentification OAuth2 pour mobile dans notre système :

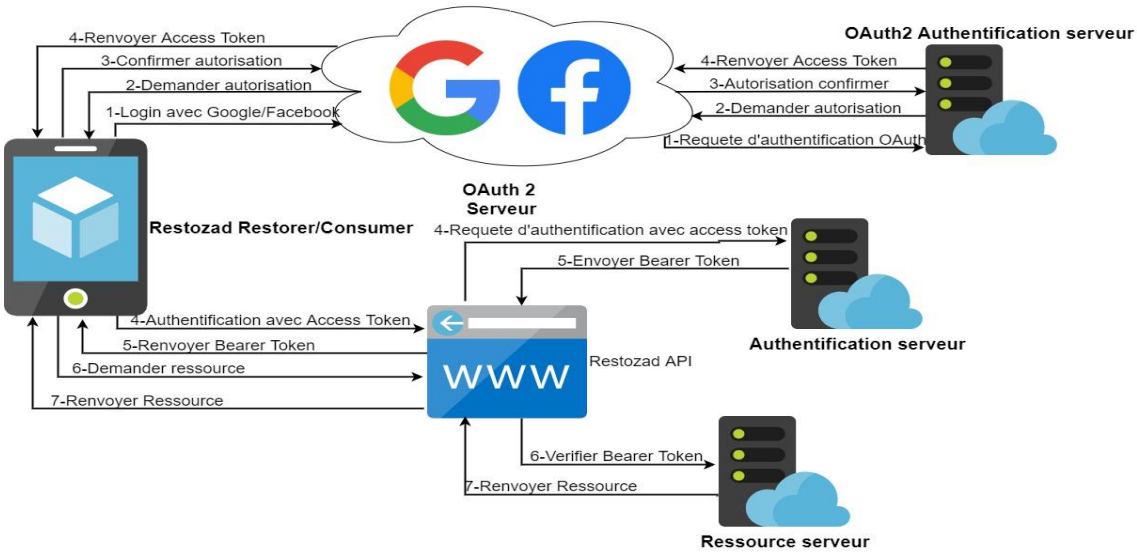

Figure 3-19 : Authentification OAuth2 pour mobile

Dans le système c'est Laravel Socialite qui est utilisé sur le Backend comme moyen de s'authentifier auprès des fournisseurs OAuth ; Socialite prend actuellement en charge l'authentification via Facebook, Google, (etc.), l'étape d'authentification se déroule comme suit :

- L'utilisateur va être dirigé vers la page de connexion du fournisseur OAuth (Google, Facebook…).

- L'utilisateur se connecte sur son compte d'une manière sécurisée sur le site du fournisseur.
- On demande le droit d'accéder à certaines informations (email, username…).
- Apres confirmation une réponse est renvoyée, elle contient les données et l'access Token au Backend concerné (RESTOZAD API).
- On crée le compte de l'utilisateur en utilisant l'access Token et les données récupérées puis les informations du fournisseur sont sauvegardées.
- Socialite va générer à son tour un Bearer Token en utilisant Laravel Sanctum.
- Le Bearer Token et renvoyé comme réponse.

Concernant les applications mobiles réalisées en React Native, l'authentification avec OAuth peut être faite avec différents packages mais certains sont à éviter car déclarés comme obsolète. Expo Auth Session est utilisé car étant le package recommandé. Ce package réalise l'authentification comme suit :

- L'utilisateur va être dirigé vers le navigateur ensuite il va ouvrir la page de connexion du fournisseur OAuth (Google, Facebook…).
- L'utilisateur se connecte à son compte (d'une manière sécurisée) sur le site du fournisseur.
- On demande le droit d'accéder à certaines informations (email, username…).
- Apres confirmation, une réponse est renvoyée autorisant l'access Token à l'application mobile.
- L'application mobile envoie l'access Token au Backend concerné.
- L'API effectue l'authentification OAuth comme indiqué précédemment puis renvoie à la fin Token à l'application mobile.

3) Système de vérification OTP :

Un système de vérification est implémenté par l'utilisation d'OTP, c'est un mot de passe à usage unique qui expire au bout d'un certain temps, il est envoyé par (adresse mail, sms ou notification) mais tout dépend du besoin et du niveau de sécurité souhaité. Ce système peut être implémenté d'une manière simple et basique ou en utilisant des packages qui proposent des fonctionnalités complexes. Les OTP sont devenus largement utilisables parce qu'ils proposent une sécurité supérieure au mode d'identification de base identifiant/mot de passe. Enfin, OTP permet de vérifier pour des actions risquées que l'utilisateur soit bien le titulaire du compte.

Ce système a uniquement implémenté l'idée de la vérification OTP en générant un code à 6 chiffres qui est sauvegardé et ensuite envoyé sur email. Après, l'utilisateur vérifie son mail pour obtenir le code est l'utiliser sur l'application mobile. Cette vérification par OTP est utilisée pour : la vérification du compte, la demande de modification d'un mot de passe oublié.

Dans la **Figure *3-20*** nous montrons un exemple d'OTP dans l'une des applications mobiles :

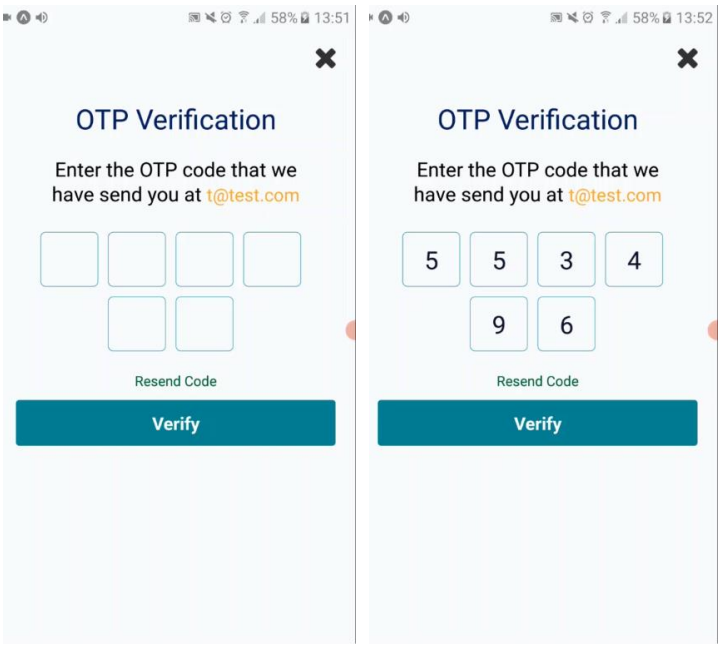

Figure 3-20 : Exemple OTP

3.5.7. Système de Push Notification :

Une notification push correspond à un message envoyé pour une application mobile qui s'affiche sur un smartphone. Les éditeurs d'applications peuvent les expédier à tout moment ; les utilisateurs n'ont nul besoin d'ouvrir l'application ou d'être en train d'utiliser leur appareil pour les recevoir [47]. Les notifications Push ne sont envoyées qu'aux utilisateurs ayant installés l'application, certaines plateformes mobiles prennent en charge les notifications push iOS, Android, Windows, (etc.), elles ont toutes leurs propres services.

Ci-dessous la **Figure *3-21*** montre l'enregistrement de notre smartphone auprès d'Expo pour recevoir les push notification sur le smartphone :

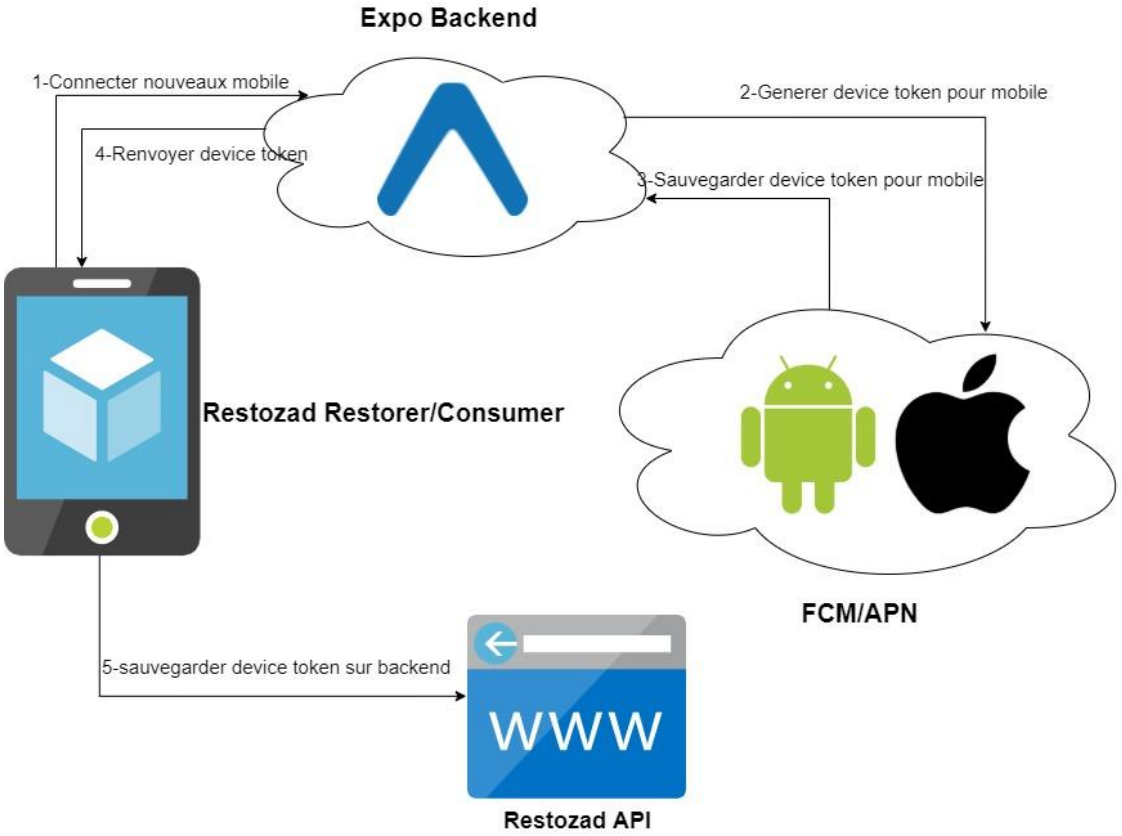

Figure 3-21 : Push Notification enregistrement d'un smartphone

Expo Push Notification est utilisé pour les notifications push dans une application React Native, l'application doit être préalablement enregistrée pour obtenir un Token de notification push (voir **Figure *3-22***). Ce dernier est une longue chaîne qui identifie de manière unique chaque appareil. Il est stocké dans une base de données sur le serveur, une notification est reçue pour accéder au contenu.

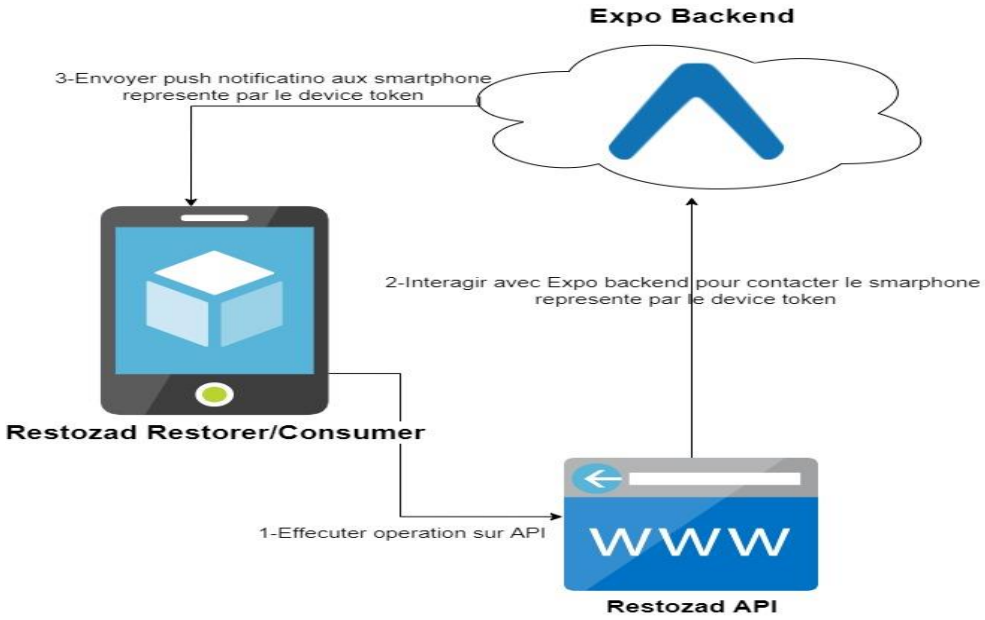

Figure 3-22 : Push Notification envoyé notification

Dans notre système l'application mobile utilise le système de Push Notification comme suit :

- S'enregistrer d'abord pour obtenir un Token de notification push.
- Ce Token est enregistré sur le serveur d'Expo mais aussi sur notre API.
- Suite à la commande d'un plat ou la modification du délai d'attente d'une commande, l'API va envoyer une requête contenant toutes les informations de la notification et le Token de l'utilisateur concerné au serveur d'Expo.
- Le serveur d'Expo va envoyer un push notification au smartphone ayant le Token qu'on a reçu.
- Le smartphone concerné reçoit une notification.

3.5.8. Système de planification de tâche :

Laravel propose un planificateur de commandes ; c'est une nouvelle approche de gestion des tâches planifiées sur un serveur. Le planificateur permet de définir de manière expressive la planification de commandes dans une seule entrée Cron [53].

Dans notre système les tâches Cron sont utilisées pour : changer la disponibilité du restaurant selon les horaires d'ouverture et fermeture, supprimer les codes OTP expirés sur notre BDD.

3.5.9. Présentation de RESTOZAD :

- **Création de compte :** création d'un compte pour consommateur/restaurateur le même design a été utilisé dans les deux applications.

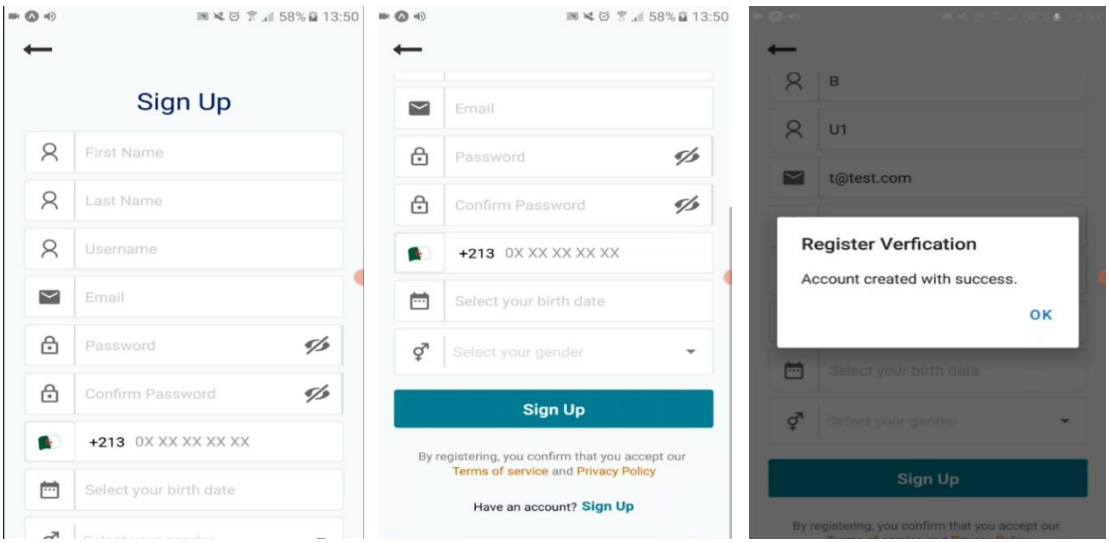

Figure 3-23 : Création de compte

- **Connexion restaurateur/consommateur :** connexion d'un restaurateur/ consommateur avec succès.

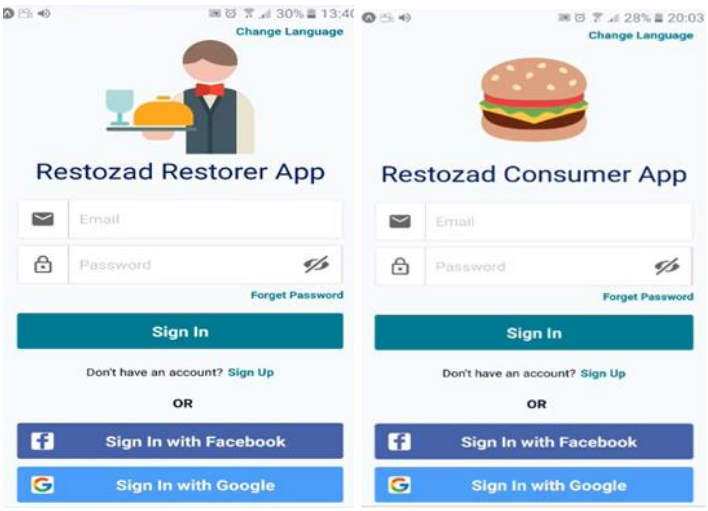

Figure 3-24 : Connexion restaurateur/consommateur

- **Vérification restaurateur :** vérification du compte d'un restaurateur avec OTP.

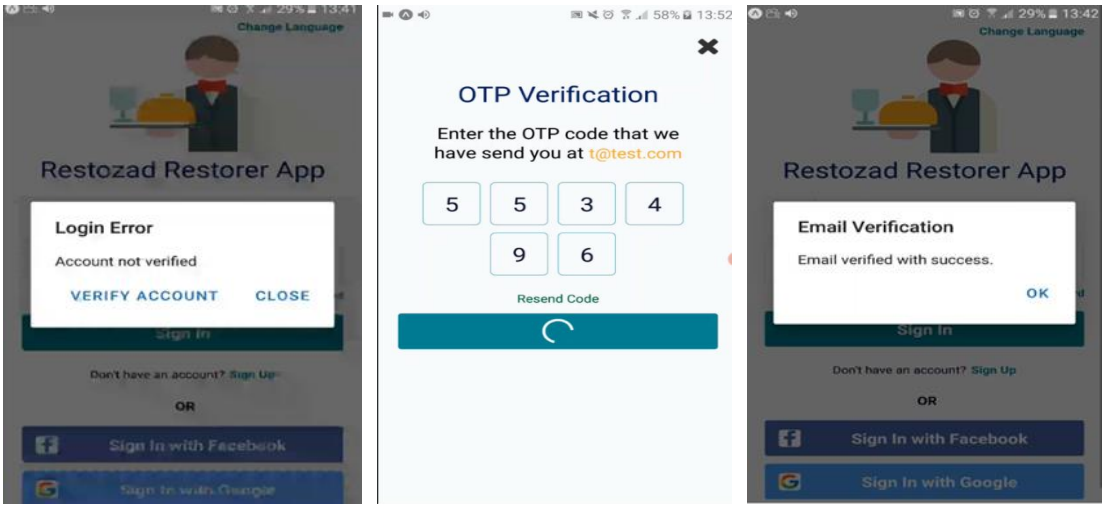

Figure 3-25 : Vérification restaurateur

- **Récupérer un mot de passe oublié :** récupérer le mot de passe d'une adresse en utilisant la vérification OTP. Le même design est utilisé pour les deux applications.

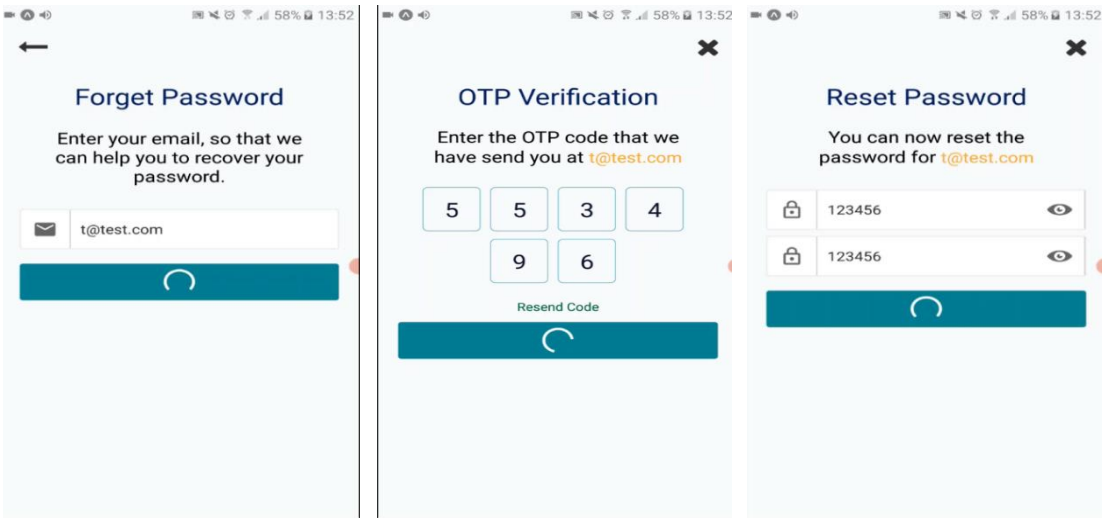

Figure 3-26 : Récupérer mot de passe oublié

- **Connexion OAuth Google :** connexion OAuth avec Google, redirige vers la page de Google.

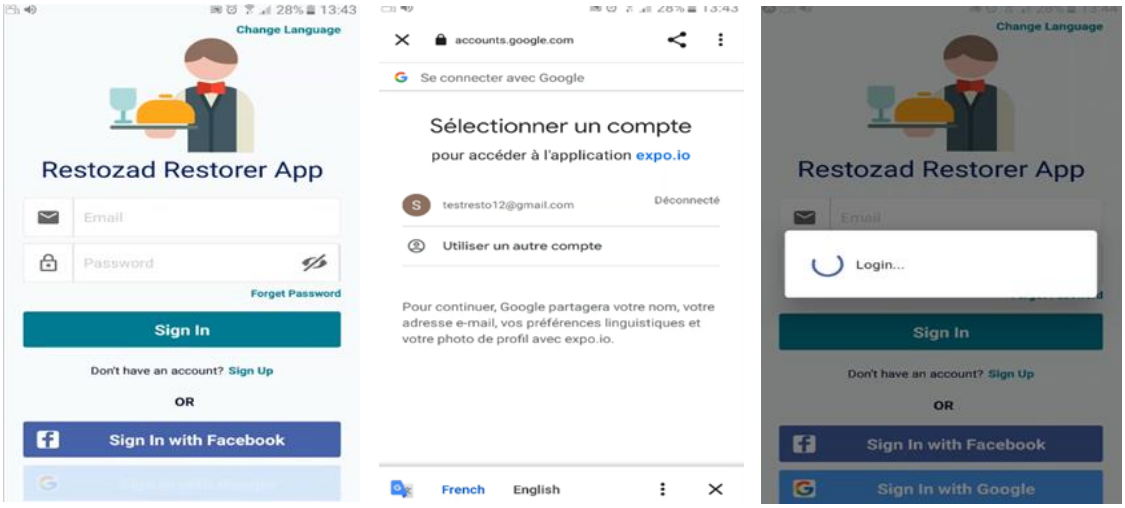

Figure 3-27 : Connexion OAuth Google

- **Gestion des commandes :** la gestion des commandes pour restaurateur.

- **Liste des commandes :** afficher la liste des commandes pour les restaurants.

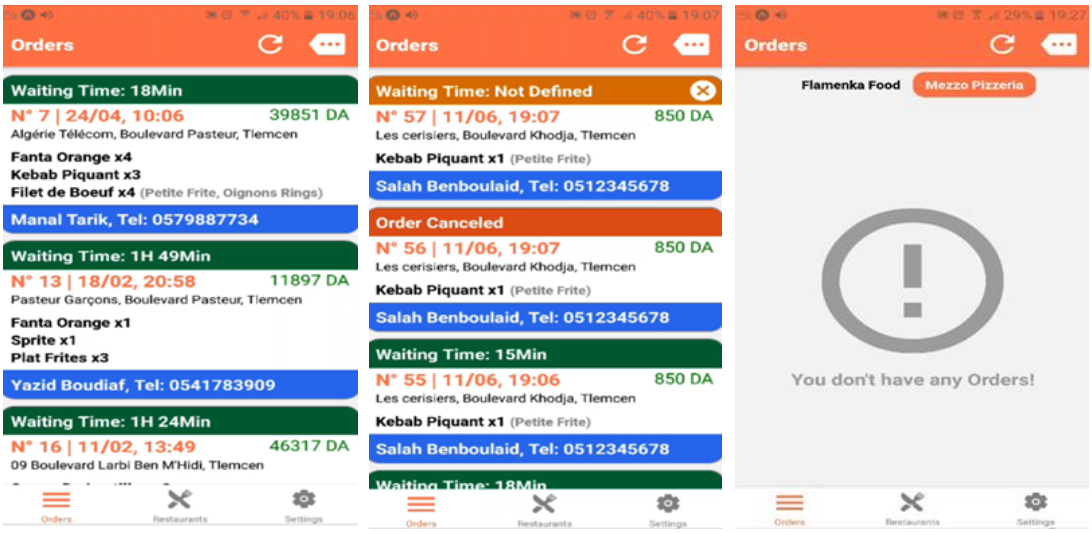

Figure 3-28 : Liste des commandes

- **Nouvelle commande et modification délai d'attente :** notification d'une nouvelle commande pour restaurant et modification du délai d'attente d'une commande.

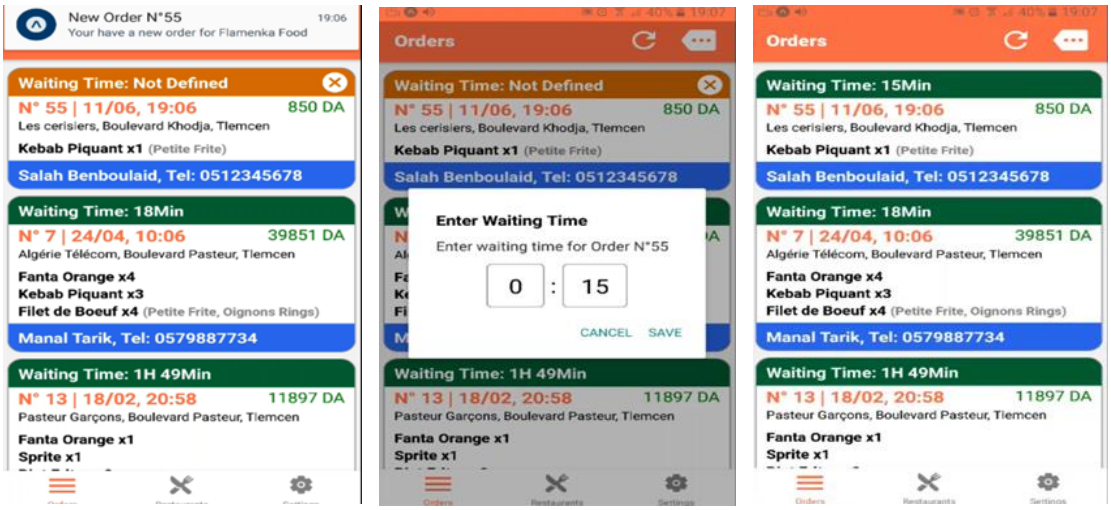

Figure 3-29 : Nouvelle commande et modification délai d'attente d'une commande

- **Gestion restaurant :** la gestion d'un restaurant pour le restaurateur, le restaurateur doit pouvoir : lister restaurant, ajouter/modifier/supprimer restaurant, changer disponibilité restaurant, détail restaurant (détail, description, position GPS), ajouter/modifier/supprimer horaires d'ouverture/fermeture, liste horaires d'ouverture/fermeture, galerie images.

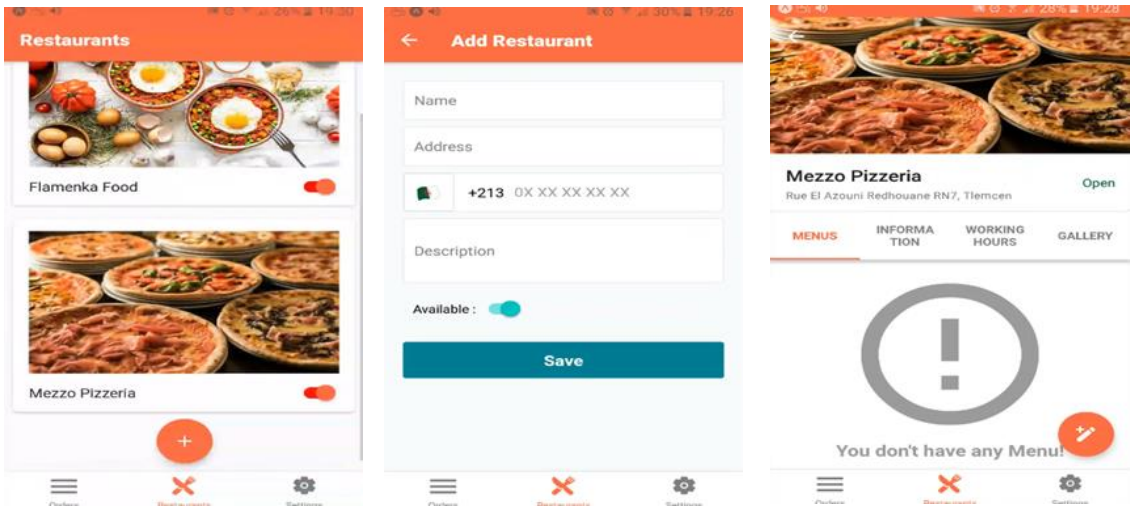

Figure 3-30 : Gestion restaurant partie 1

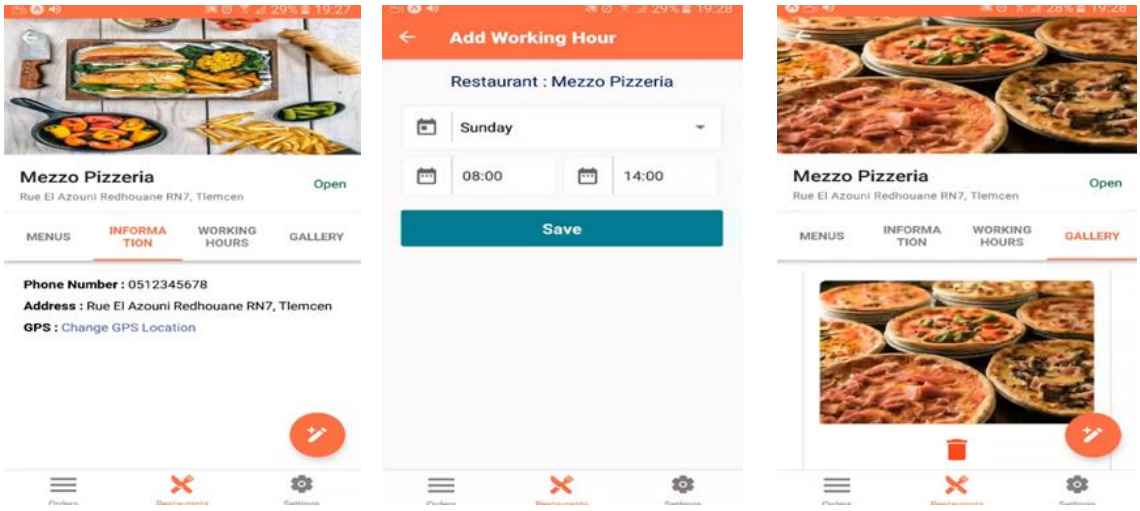

Figure 3-31 : Gestion restaurant partie 2

- **Gestion menu :**la gestion de menu pour le restaurateur, il doit pouvoir : lister le menu pour le restaurant, ajouter/modifier/supprimer menu.

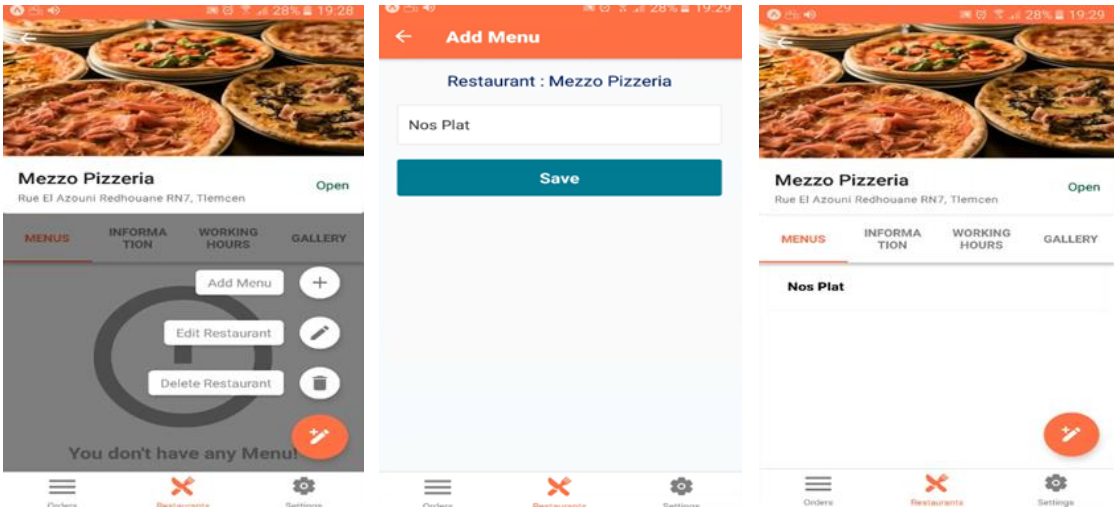

Figure 3-32 : Gestion menu

- **Gestion plat :**la gestion d'un plat pour le restaurateur, il doit pouvoir : consulter liste des plats pour menu, ajouter/modifier/supprimer plat, gérer galerie images.

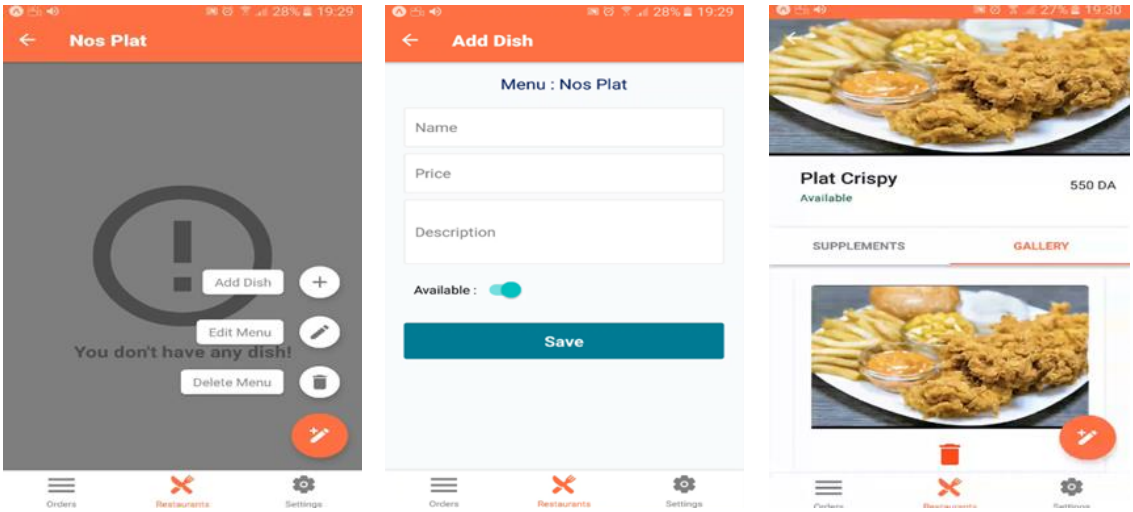

Figure 3-33 : Gestion plat

- **Gestion supplément :** la gestion d'un supplément pour le plat, le restaurateur doit pouvoir : consulter liste des suppléments pour le plat, ajouter/modifier/supprimer supplément.

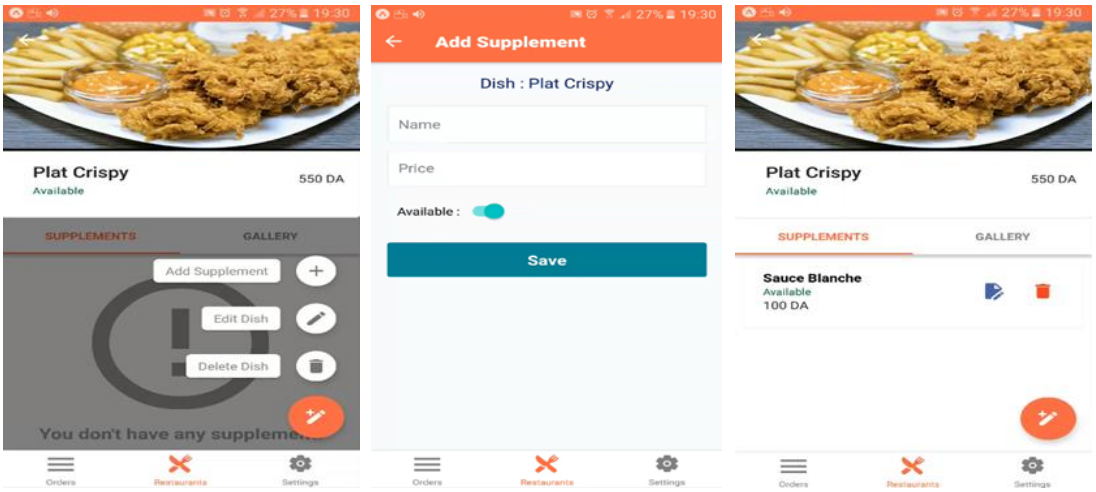

Figure 3-34 : Gestion supplément

- **Paramètres :** éditer profil, modifier mot de passe, uploader avatar image, changement de langue, déconnexion. Le même design est utilisé pour les deux applications.

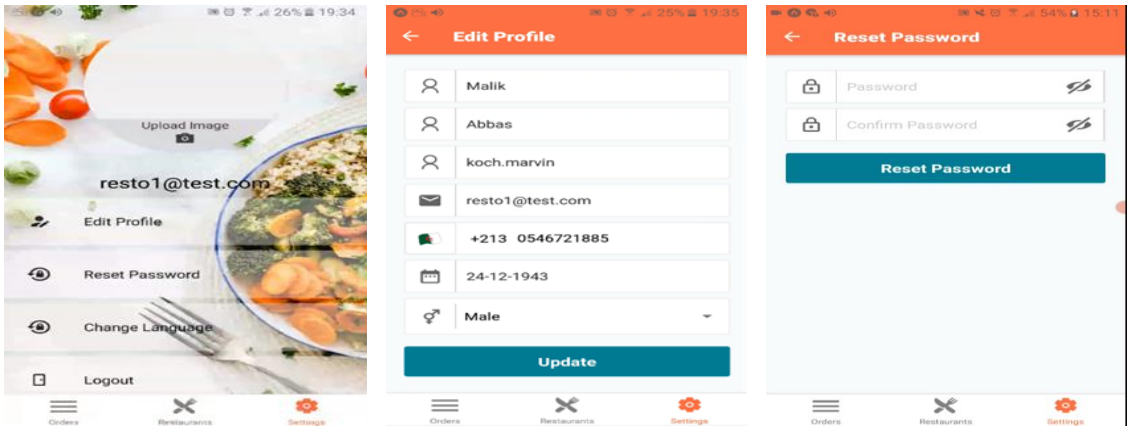

Figure 3-35 : Paramètres

- **Map restaurant :** voir les différents restaurants sur une MAP.

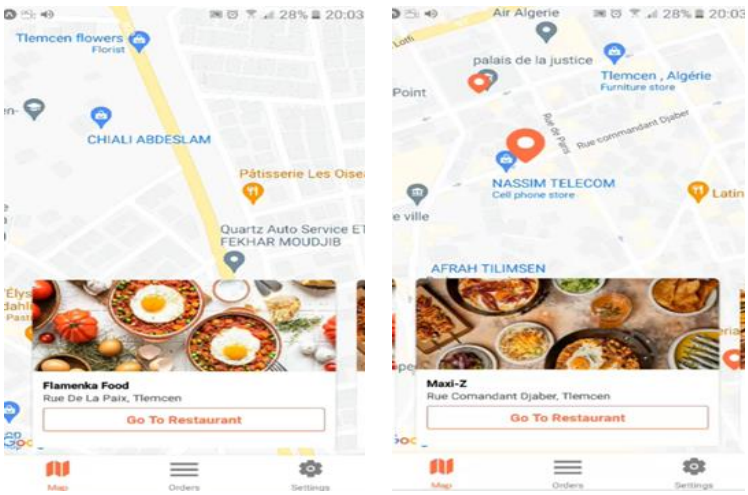

Figure 3-36 : Map restaurant

- **Détail restaurant, menu, plat, supplément :** voir le détail d'un restaurant avec les menus, plat, supplément et toutes les informations les concernant.

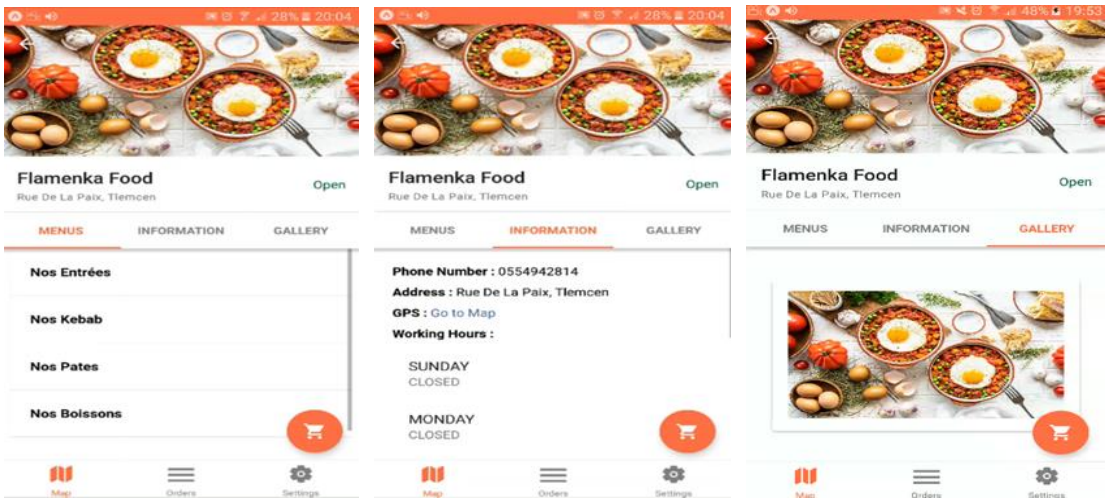

Figure 3-37 : Détail restaurant avec liste menu

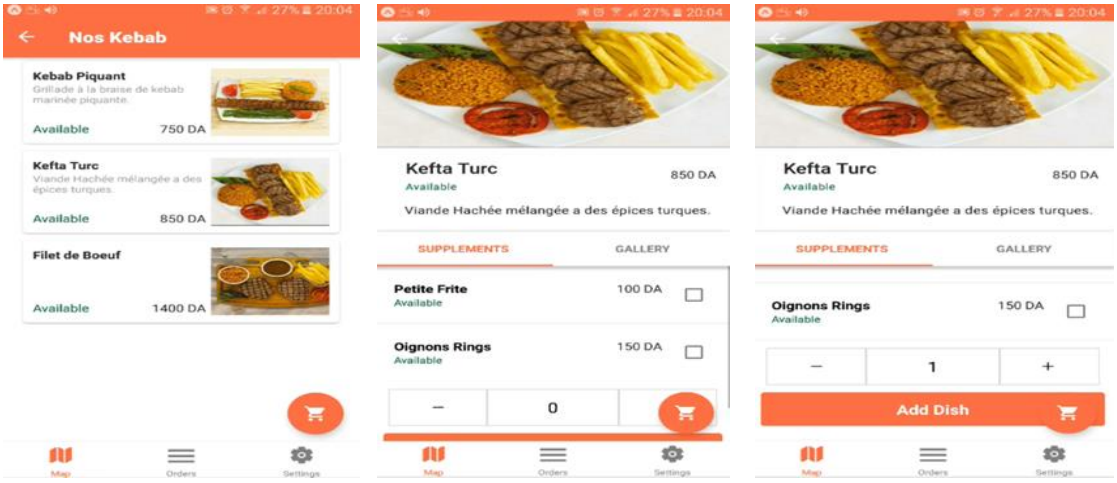

Figure 3-38 : Liste plat, détail plat avec supplément

- **Ajouter plat au panier :** ajouter plat au panier avec ou sans supplément.

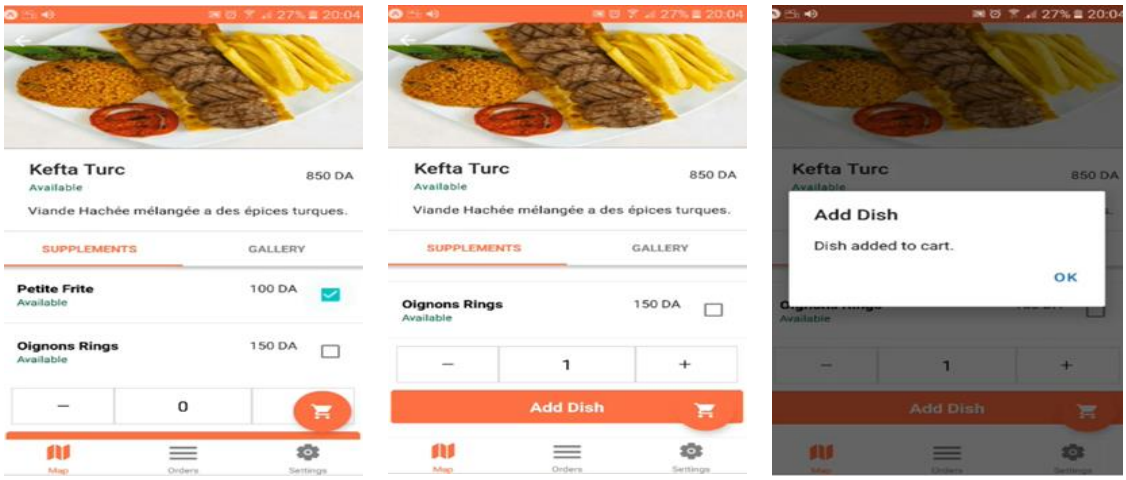

Figure 3-39 : Ajouter plat au panier

- **Commander des plats présents dans le panier :** gérer panier et confirmer une commande.

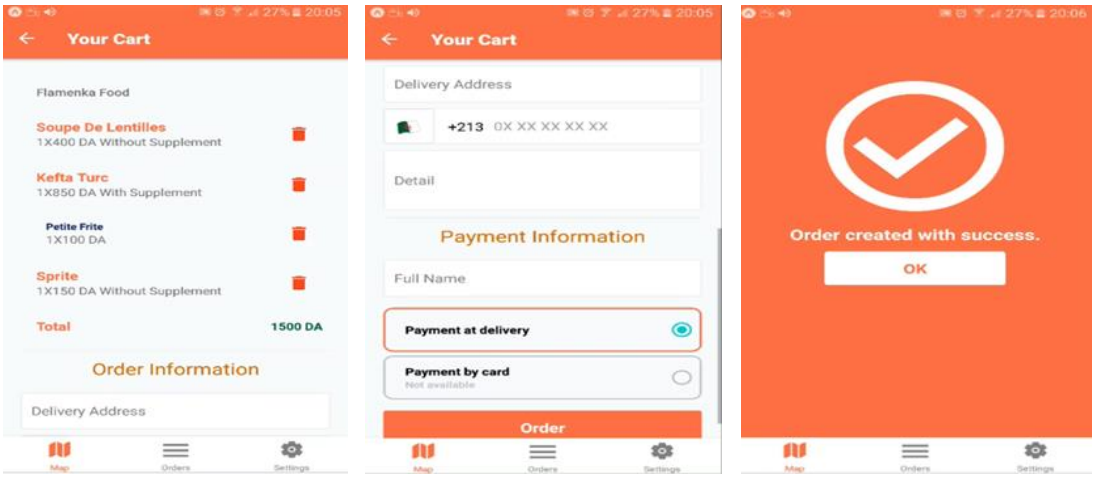

Figure 3-40 : Commande des plats présents dans le panier

- **Annulée commandes :** annulée commandes par le consommateur.

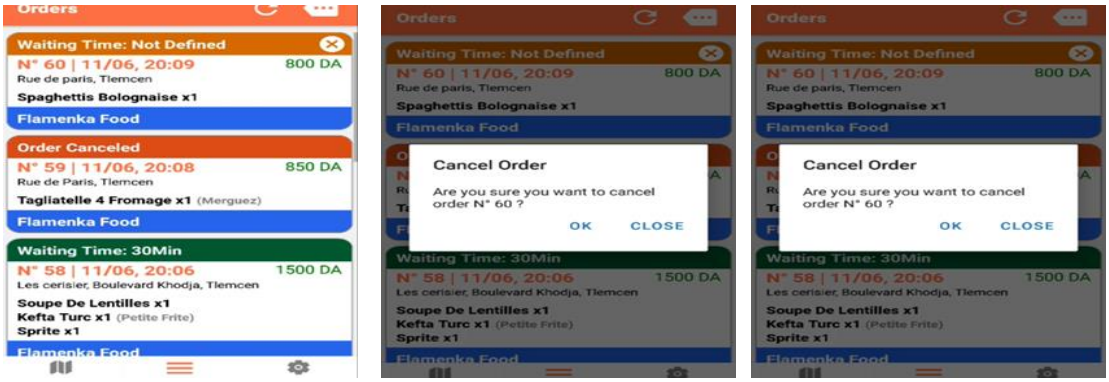

Figure 3-41 : Annulée une commande

3.5.10. Problèmes rencontrés :

Durant la phase de développement et d'implémentation divers obstacles ont été rencontrés, dus à différentes causes et qui ont impactés la réalisation du SCL.

On peut commencer par la difficulté à bien appréhender le besoin du client sans trop sortir du contexte. Le SCL se concentre principalement sur la commande de plats auprès de restaurant mais ce n'est pas un système de gestion de restaurant à part entière pour le restaurateur même s'il nécessite certaines fonctionnalités de gestion de la restauration. Ce système ne gère par l'aspect livraison des commandes mais qui peut être régi par un autre système. Aussi, il faut bien se mettre d'accord sur ce que le SCL représente vraiment.

La difficulté à implémenter certaines fonctionnalités a été rencontrée, parce que les technologies utilisées obligent à suivre leur implémentation, d'où l'obligation de devoir se réadapter aux technologies, certains cas rencontrés :

- Planification du changement de disponibilité d'un restaurant selon les horaires d'ouverture fermeture. Pour ce faire, le système doit en permanence vérifier et comparer l'horaire actuel avec les horaires prédéfinis d'où l'impossibilité de planifier la modification selon des horaires dynamiques et définis par l'utilisateur.
- Authentification OAuth, elle a représenté un grand problème car la méthode d'implémentation change d'un système à l'autre. La difficulté à réaliser l'OAuth sur l'application mobile en laissant la gestion des Token et utilisateur sur notre API ce qui n'as pas été facile à faire. Généralement l'authentification est faîte entièrement avec des services tel que Firebase ce qui réduit la difficulté mais dans notre cas il fallait combiner plusieurs services pour réaliser cela.

Un des obstacles le plus souvent rencontré s'explique par le fait que certains packages (principalement ceux utilisés dans React Native) étaient devenus obsolètes et leur alternative contenait des bugs non-fixés et qui sont encore présents. Cela était laborieux car chaque cas devenait difficile à résoudre ce qui eut pour conséquence de ralentir fortement le développement, ce problème a été rencontré surtout lors le développement Frontend avec React Native.

3.5.11. Sécurité des applications :

La sécurisation d'applications est une partie majeure de tout développement de système informatique, elle permet de se protéger des failles de sécurité et des attaques pirates malveillantes et nuisibles à un système. Laravel et React Native proposent des techniques et mécanismes de protection des attaques les plus connues, exemple :

- Protection contre les attaques SQL par la validation des requêtes reçues par Laravel.
- Protection contre les injections XSS aussi bien existantes sur Laravel et React Native.
- Vérifier l'authenticité entre client et serveur avec CSRF Token.
- Authentification sécurisée par l'utilisation de Token avec une date expirée.
- La gestion des droits d'accès pour les Token dans Laravel.

- Protection contre l'attaque DOS sur Laravel en définissant une limite de requête.
- Utilisation de stockage local sécurisé sur React Native au moyen de SecureStore.
- Recourir au package non-obsolète et remplacer ceux contenant des failles de sécurité par d'autres alternatives.
- Communication avec des API RESTFUL pour des raisons de cohérence.
- Echange avec des serveurs dotés d'un certificat SSL.

3.6. Déploiement :

Dans cette phase de déploiement, nous mettrons en œuvre le processus de mise en service du SCL.

L'API RESTOZAD sur le serveur FTP du client est déployée en ajoutant les url des différents services nécessaires pour le bon fonctionnement (Redis, MySQL, (etc.). Les clés API sont créées et indexées aux fichiers d'environnement. Grâce à l'accès SSH fournit, les tâches Cron et certaines commandes dont une pour nettoyer le cache sont exécutées. Les outils de test utilisés (Laravel Telescope, etc.) ont dû être protégés.

Les applications mobiles sont déployées sur une machine distante (sous Windows 10), et les fichiers d'environnement sont mis à jour pour utiliser la nouvelle url de l'API. Les clés API sont créées et indexées aux fichiers d'environnement pour finalement publier les applications mobiles sur EXPO afin d'avoir accès et les tester directement.

Pour le nom de domaine nous avons utilisé « www.restozad.naltis.dz » pour l'application l'api.

Côté messagerie, une adresse mail « restozad@burotek.com » est créée pour pouvoir contacter « l'administration » au sujet des informations générales sur le projet.

3.7. Test :

Quelques fiches de tests d'intégration ont été rédigées dans le **Tableau *3-1*** afin de de vulgariser le contenu d'une fiche de test à savoir : le numéro de fiche, l'entité ou l'objet concerné par le test, le nom de la fonctionnalité à tester, le scénario à suivre pour réaliser l'objectif du test, et enfin le résultat attendu.

NF	Objet	Nom	Scénario	Résultat attendu
1	Authentification du Restaurateur	Connexion à l'application mobile RESTOZAD Restorer	1-Ouvrir l'application mobile RESTOZAD Restorer. 2- entrez email et password. 3-cliquer sur connexion.	La partie gestion des commandes de restaurant doit apparaitre
2	Commande	Consommateur créer une commande	1-naviguer vers la page d'un restaurant 2-choisir un plat. 3- ajouter le plat au panier en sélectionnant la quantité et le supplément souhaité. 6-cliquer sur bouton du panier. 7-confirmer commande en remplissant les informations restantes.	Page de succès de commande réussie avec succès. Une notification est reçue sur l'application du restaurateur.
3	Commande	Restaurateur Met à jour le délai d'attente d'une commande	1-cliquer bouton pour mettre à jour le délai d'attente d'une commande. 2- inscrire le délai. 3-confirmer modification.	La liste des commandes est mise à jour. Une notification est reçue sur l'application du consommateur.

Tableau 3-1 : Un exemple de fiche de test

3.8. Conclusion :

Dans ce chapitre, notre contribution a suivi la méthodologie suivante, tout d'abord présenter et montrer le processus de réalisation du projet, puis parler de la spécification des exigences, enfin décrire notre conception (cas d'utilisation, classe, séquence). Pour ce faire, des diagrammes significatifs ont dû être élaborés ainsi que la construction de l'architecture système et modulaire.

Au terme de ces développements, l'énumération des principaux outils et technologies utilisés ainsi le volet sécurisation du système ont étés mis en avant. Pour conclure notre travail a été étayé et enrichi par la présentation nos deux applications mobiles.

Conclusion Générale

Notre travail vise à accompagner les relations restaurateurs/consommateurs et ainsi à simplifier, faciliter, et accélérer la commande de plats auprès de prestataires. Dans ce travail nous avons conçu une API et deux applications :

RESTOZAD API représente le Backend de notre système, elle contient tout le code métier du projet et les deux applications mobiles développées communiqueront avec elle. Cette API a été développé de façon à être complétement détachée du client qui va l'utiliser qu'il soit une application mobile, une application web, etc. L'API représente en réalité le centre de notre projet sans elle les applications mobiles ne peuvent rien réaliser.

Une application mobile pour le restaurateur appelée RESTOZAD Restorer pour la gestion de commande. Elle peut recevoir la notification d'une nouvelle commande, contrôler la disponibilité des plats et supplément, régir de disponibilité du restaurant manuellement ou automatiquement, définir des horaires d'ouverture et de fermeture, gérer des menus, plats et suppléments proposés, sur un ou sur une chaine de restaurants.

Une autre application mobile pour le consommateur dénommée RESTOZAD Consumer pour commander des plats auprès de restaurants et dans laquelle il peut visionner les restaurants sur la Map, voir en détail un restaurant, son menu, ses plats et ses suppléments, ajouter des plats au panier, confirmer la commande, renseigner les détails de la commande, recevoir une mise à jour de la commande.

A la lumière du développement de notre système et de l'étude des systèmes concurrents, de futures perspectives pour l'évolution et l'amélioration du SCL peuvent être envisagées.

Comme première perspective, améliorer les applications en ajoutant un système de chat direct entre restaurateur et consommateur lors d'une commande.

En deuxième perspective, ajouter un système de livraison en créant une application spécifique pour les coursiers et ainsi ne plus dépendre de particuliers ou de tierces personnes pour la livraison. Il a l'avantage de pouvoir suivre en temps réel l'emplacement de la commande depuis sa récupération jusqu'à l'arrivée à destination.

En troisième perspective, donner la possibilité aux utilisateurs de noter et signaler leur avis sur l'opération, ainsi que de définir des paramètres de niveau de satisfaction, conformité produits, qualité, rapidité, (etc.), cela confèrera aux restaurateurs des statistiques décrivant les critères évoqués.

Comme dernière amélioration, créer un fil d'actualités avec différentes offres, promotions ou activités projetées par les restaurateurs aux consommateurs, ainsi que la proposition de l'utilisation de coupons ou de tickets de réduction.

Bibliographies

[1] **H. Akhmadi, A. R. Alfathah, and Susanawati, "Generation Z consumer's preferences for online food ordering application: a study of gofood and grabfood," *E3S Web Conf.*, vol. 316, p. 01011, 2021, doi: 10.1051/e3sconf/202131601011.**

[2] **K. Dhiman, "Online Food Ordering Management System," *Int. J. Res. Appl. Sci. Eng. Technol.*, vol. 9, no. VII, pp. 2096–2107, 2021, doi: 10.22214/ijraset.2021.36835.**

[3] **C. L. Tan, "Implementing a Web-Based Computerized Restaurant System," pp. 1–177, 2013, [Online]. Available: https://studentnet.cs.manchester.ac.uk/resources/library/thesis_abstracts/MSc13/FullText/Tan-ChinLoong-fulltext.pdf**

[4] **X. Hongzhen, T. Bin, and S. Wenlin, "Wireless Food Ordering System Based on Web Services," in *2009 Second International Conference on Intelligent Computation Technology and Automation*, 2009, vol. 4, pp. 475–478. doi: 10.1109/ICICTA.2009.830.**

[5] **N. A. Samsudin, S. K. Ahmad Khalid, M. F. A. Mohd Kohar, Z. Senin, and M. N. Ihkasan, "A customizable wireless food ordering system with realtime customer feedback," in *2011 IEEE Symposium on Wireless Technology and Applications (ISWTA)*, 2011, pp. 186–191. doi: 10.1109/ISWTA.2011.6089405.**

[6] **T.-H. Tan, C.-S. Chang, and Y.-F. Chen, "Developing an Intelligent e-Restaurant With a Menu Recommender for Customer-Centric Service," *IEEE Trans. Syst. Man, Cybern. Part C (Applications Rev.*, vol. 42, no. 5, pp. 775–787, 2012, doi: 10.1109/TSMCC.2011.2168560.**

[7] **R. Shinde, B. B. Ambedkar, P. Thakare, N. Dhomne, and S. Sarkar, "Design and Implementation of Digital dining in Restaurants using Android," 2014.**

Webographie

[8] **"Jumia Food." https://food.jumia.co.ke/. Dernière visite : 22/06/2022.**

[9] **"YASSIR Express." https://express.yassir.io/. Dernière visite : 22/06/2022.**

[10] **"Fast Delivery." https://www.fastdelivery.dz/. Dernière visite : 22/06/2022.**

[11] **"Ubereats Merchants." https://merchants.ubereats.com/fr/fr/s/signup/. Dernière visite : 22/06/2022.**

[12] **"Ubereats." https://www.ubereats.com/fr. Dernière visite : 22/06/2022.**

[13] **"Diagramme de cas d'utilisation." https://fr.wikipedia.org/wiki/Diagramme_de_cas_d%27utilisation. Dernière visite : 22/06/2022.**

[14] **"Laragon." https://laragon.org/. Dernière visite : 22/06/2022.**

[15] **"vsc." https://code.visualstudio.com/. Dernière visite : 22/06/2022.**

[16] **"Android Studio." https://developer.android.com/studio. Dernière visite : 22/06/2022.**

[17] **"Apache." https://httpd.apache.org/. Dernière visite : 22/06/2022.**

[18] **"HeidiSQL." https://www.heidisql.com/. Dernière visite : 22/06/2022.**

[19] **"Postman." https://www.postman.com/. Dernière visite : 22/06/2022.**

[20] **"Thunder Client." https://www.thunderclient.com/. Dernière visite : 22/06/2022.**

[21] **"Git." https://git-scm.com/. Dernière visite : 22/06/2022.**

[22] **"Github." https://github.com/. Dernière visite : 22/06/2022.**

[23] **"draw.io." https://www.draw.io/. Dernière visite : 22/06/2022.**

[24] **"Chrome Dev." https://developer.chrome.com/. Dernière visite : 22/06/2022.**

[25] **"AVD Manager." http://android.cn-mirrors.com. Dernière visite : 22/06/2022.**

[26] **"PHP." https://www.php.net/. Dernière visite : 22/06/2022.**

[27] **"Laravel." https://laravel.com/. Dernière visite : 22/06/2022.**

[28] **"Composer." https://getcomposer.org/. Dernière visite : 22/06/2022.**

[29] **"Laravel Telescope." https://github.com/laravel/telescope. Dernière visite : 22/06/2022.**

[30] **"Scribe." https://scribe.readthedocs.io/. Dernière visite : 22/06/2022.**

[31] **"SQL." https://sql.sh/. Dernière visite : 22/06/2022.**

[32] **"Redis." https://redis.io/docs/about/. Dernière visite : 22/06/2022.**

[33] **"Cron." https://fr.wikipedia.org/wiki/Cron. Dernière visite : 22/06/2022.**

[34] **"API." https://fr.wikipedia.org/wiki/Interface_de_programmation. Dernière visite : 22/06/2022.**

[35] **"RESTFUL." https://www.journaldunet.fr/web-tech/developpement/1203011-qu-est-ce-que-restful/. Dernière visite : 22/06/2022.**

[36] "OAUTH." https://fr.wikipedia.org/wiki/OAuth. Dernière visite : 22/06/2022.

[37] "OTP." https://fr.wikipedia.org/wiki/Mot_de_passe_à_usage_unique. Dernière visite : 22/06/2022.

[38] "JSON." https://fr.wikipedia.org/wiki/JavaScript_Object_Notation. Dernière visite : 22/06/2022.

[39] "JS." https://www.javascript.com/. Dernière visite : 22/06/2022.

[40] "TS." https://www.typescriptlang.org/. Dernière visite : 22/06/2022.

[41] "React." https://fr.reactjs.org/. Dernière visite : 22/06/2022.

[42] "React Native." https://reactnative.dev/. Dernière visite : 22/06/2022.

[43] "EXPO." https://expo.dev/. Dernière visite : 22/06/2022.

[44] "Axios." https://www.geeksforgeeks.org/axios-in-react-native/. Dernière visite : 22/06/2022.

[45] "YARN." https://yarnpkg.com/. Dernière visite : 22/06/2022.

[46] "EXPO PUSH." https://jobphoning.com/dictionnaire/notification-push. Dernière visite : 22/06/2022.

[47] "Push notification." https://www.airship.com/fr/ressources/definition/notifications-push/. Dernière visite : 22/06/2022.

[48] "Pattern." https://codesource.io/brief-overview-of-design-pattern-used-in-laravel/. Dernière visite : 22/06/2022.

[49] "service pattern." https://en.wikipedia.org/wiki/Service_layer_pattern. Dernière visite : 22/06/2022.

[50] "Context API." https://blog.devgenius.io/react-native-state-management-with-context-api-61f63f5b099. Dernière visite : 22/06/2022.

[51] "React Native Localisation." https://enappd.com/blog/how-to-translate-in-react-native-app-globalization-internationalization-and-text-to-speech/120/. Dernière visite : 22/06/2022.

[52] "Authentification." https://fr.wikipedia.org/wiki/Authentification. Dernière visite : 22/06/2022.

[53] "Laravel Cron." https://laravel.com/docs/9.x/scheduling. Dernière visite : 22/06/2022.

Résumé:

La commande de plats en ligne représente une avancée significative et prometteuse visant à remplacer le mode traditionnel de prise de commande par un système informatisé automatisé qui apportera tous les avantages des technologies actuelles et ainsi faciliter et améliorer l'échange entre restaurateur et consommateur.

Notre contribution a consisté à réaliser deux applications mobiles : une pour le restaurateur afin de pouvoir gérer ses restaurants et leurs commandes, une deuxième pour les consommateurs qui pourront naviguer sur les restaurants disponibles et effectuer une commande en choisissant des plats. Notre travail vise à accompagner les relations restaurateurs/consommateurs pour répondre aux problèmes et difficultés de communication entre eux et ainsi à simplifier, faciliter, et accélérer la commande de plats auprès de prestataires.

Mots clés: Commande de plats en ligne, système informatisé automatisé, restaurateur, consommateur, restaurants, commande, plat, application mobile.

Abstract:

Online meal ordering represents a significant and promising step forward aimed at replacing the traditional method of ordering by using an automated computerized system that will bring all the advantages of current technologies and therefore facilitates and improve the exchange between restorer and consumer.

Our contribution consisted in creating two mobile applications: one for the restorer in order to be able to manage his restaurants and their orders; the second for the consumers who would be able to browse the available restaurants and place an order by choosing dishes. Our work aims to support their relations to respond to communication problems and difficulties between them and therefore to simplify, facilitate and speed up the meal ordering from service providers.

Keywords: Online meal ordering, automated computerized system, restorer, consumer, restaurants, order, dish, mobile application.

ملخص :

يمثل طلب الوجبات عبر الإنترنت خطوة مهمة وواعدة إلى الأمام تهدف إلى استبدال الطريقة التقليدية لأخذ الطلبات بنظام محوسب آلي يجلب جميع مزايا التقنيات الحالية وبالتالي يسهل ويحسن التبادل بين صاحب المطعم والمستهلك.

تتمثل مساهمتنا في إنشاء تطبيقين للجوال: تطبيق لصاحب المطعم حتى يتمكن من إدارة مطاعمه وطلباتهم، والثاني للمستهلكين الذين سيتمكنون من تصفح المطاعم المتاحة وتقديم طلب باختيار الأطباق .يهدف عملنا إلى دعم العلاقات بين أصحاب المطاعم والمستهلكين للاستجابة لصعوبات الاتصال بينهم وبالتالي تسريع طلب الأطباق من مقدمي الخدمات.

لكلمات الدالة: طلب وجبات عبر لإنترنت، نظام آلي محوسب، صاحب مطعم، مستهلك، مطاعم، طلب، طبق، تطبيق جوال.