

Dédicaces

Je dédie ce modeste travail

A ma très chère mère, aucune dédicace ne saurait exprimer mon amour éternel et ma gratitude pour les sacrifices que vous avez consenti pour mon bien être.

A mon cher père, je vous remercie pour tout le soutien et l'amour que vous me portez et j'espère que votre bénédiction m'accompagne toujours.

A mes frères, je suis reconnaissant à votre compréhension et votre soutien morale, je vous souhaite une vie pleine de bonheur et de succès.

A mes amis Ismail, Maryem, Oussama, Assil, vous avez toujours cru en moi et je tenais à vous remercier.

A tous ceux qui m'aiment ...

Remerciements

En préambule, je souhaite adresser ici tous mes remerciements aux personnes qui m'ont apporté leur aide et qui ont contribué à l'élaboration de ce rapport.

Je tiens à remercier Monsieur **BHIRI SAMI**, mon encadrant à l'ISIMM, pour son soutien, pour son encouragement et motivation, et l'aide qu'il m'a apportée tout au long de mon travail.

Je remercie également **les membres du jury** pour avoir accepté d'examiner et de juger mon travail.

Je tiens aussi à exprimer mes sincères remerciements à Monsieur **HASSANI WALID**, mon encadrant à la société ONEDEV, pour son aide et son soutien.

Finalement, je tiens à exprimer ma reconnaissance à tous mes professeurs de l'ISIMM pour la formation qu'ils ont eu le soin de m'apporter tout au long de mon cursus universitaire.

TABLE DES MATIÈRES

Introduction Générale	1
-----------------------------	---

Chapitre I : Contexte et objectifs

Introduction.....	2
1 Présentation générale du projet.....	2
1.1 Cadre du projet	2
1.2 Organisme d'accueil.....	2
2. Motivation pour « Meal Prep »	3
3. Etude de l'existant.....	4
3.1 Solutions existantes	4
3.2 Analyse de l'existant	8
4. Travail à réaliser	9
5. Méthodologie de conception.....	9
5.1 Méthodologie en cascade	9
5.2 Processus Unifié (PU)	11
5.3 Choix de la méthodologie de conception	12
5.3.1 Rational Unified Process (RUP)	12
5.3.2 Two Track Unified Process (2TUP)	13
5.4 Méthodologie adoptée : 2TUP	13
6. Déroulement du projet	14
Conclusion	14

Chapitre II : Analyse et spécifications des besoins

Introduction :.....	15
1. Analyse des besoins fonctionnels	15
1.1 Identification des acteurs.....	15
1.2 Expression des besoins fonctionnels	16
2. Analyse et spécification des besoins non fonctionnels	17

2.1	Les besoins non fonctionnels	17
2.2	Les besoins techniques	18
3.	Spécification des besoins fonctionnels	21
3.1	Cas d'utilisation générale	21
3.2	Cas d'utilisation « Consulter historique commande »	23
3.3	Cas d'utilisation « Passer une commande »	24
3.4	Cas d'utilisation « Mettre à jour une commande »	27
3.5	Cas d'utilisation « Gérer compte traiteur »	29
	Conclusion	31

Chapitre III : Conception

	Introduction.....	32
1.	Architecture générale de l'application	32
1.1	L'architecture 3-tiers	32
1.2	Modèle MVC.....	33
1.3	Architecture adoptée	35
2.	Conception de la base de données	37
2.1	Modèle conceptuel de données (MCD).....	37
2.2	Modèle logique de données (MLD)	40
3.	Conception logicielle	40
3.1	Vue statique : diagramme de classe	40
3.2	Vue Dynamique : diagrammes de séquences de conception.....	43
3.2.1	Diagramme de séquence générale.....	43
3.2.2	Diagramme de séquence de C.U 'Authentification'	44
3.2.3	Diagramme de séquence de C.U 'Consulter repas'	47
3.3	Vue dynamique : diagramme d'état d'une commande.....	48
4.	Conception graphique	48
4.1	Diagramme de navigation	48
4.2	Maquettage	50
	Conclusion	54

Chapitre IV : Réalisation

	Introduction.....	55
--	-------------------	----

1. Environnement de développement.....	55
1.1 Environnement matériel	55
1.2 Environnement logiciel	56
2. Framework Symfony	57
2.1 Création d'un projet Symfony.....	58
2.2 Manipulation des données avec Symfony	59
2.3 Mapping objet relationnel avec Symfony	61
2.4 Calcul des besoins nutritifs	62
3. Framework Flutter	63
3.1 Structure du projet Flutter	64
3.2 Récupération des données	65
4. Travail réalisé.....	66
Conclusion	73
Conclusion générale.....	74

LISTE DES FIGURES

Figure 1-1. Page d'accueil du site Sunbasket.com	5
Figure 1-2 Page d'accueil du site Parsleybox.com.....	6
Figure 1-3 Page d'accueil du site Blueapron.com.....	7
Figure 1-4 Représentation générale de la méthodologie classique.....	10
Figure 1-5 Représentation du modèle de cycle en V	10
Figure 1-6 Les étapes fondamentales du PU.....	11
Figure 1-7 Représentation de cycle de développement du PU	12
Figure 1-8 Les étapes de développement de 2TUP.....	13
Figure 1-9 Chronogramme de déroulement du projet.....	14
Figure 2-1 Taux d'intérêt entre le Flutter et le React Native.....	20
Figure 2-2 Taux d'utilisation des différents SGBD.....	21
Figure 2-3 Diagramme de cas d'utilisation générale.....	22
Figure 2-4 Diagramme de C.U "Gérer profil".....	23
Figure 2-5 Diagramme de séquence "Gérer profil".....	23
Figure 2-6 Diagramme de C.U "Commander Repas".....	25
Figure 2-7 Diagramme de C.U "Commander Repas selon besoins".....	25
Figure 2-8 Diagramme de séquence "Commander Repas".....	26
Figure 2-9 Diagramme de C.U "Gérer commande".....	28
Figure 2-10 Diagramme de séquence "Gérer commande".....	28
Figure 2-11 Diagramme de C.U "Gérer compte traiteur".....	29
Figure 2-12 Diagramme de séquence "Gérer compte traiteur".....	30
Figure 3-1 Cinématique de l'architecture 3-tiers.....	33
Figure 3-2 Les parties fondamentales de modèle MVC.....	33
Figure 3-3 Echange d'informations entre les éléments du modèle MVC.....	34
Figure 3-4 Fonctionnement de modèle MVC.....	35
Figure 3-5 Architecture générale de mon système.....	35

Figure 3-6 Architecture détaillée de mon système.....	36
Figure 3-7 Modèle Conceptuel de Données (MCD).....	39
Figure 3-8 Diagramme de classe.....	42
Figure 3-9 Diagramme de séquence générale coté web.....	43
Figure 3-10 Diagramme de séquence générale coté mobile.....	44
Figure 3-11 Diagramme de séquence de C.U "Authentification" coté mobile.....	45
Figure 3-12 Diagramme de séquence de C.U "Authentification" coté web.....	46
Figure 3-13 Diagramme de séquence de C.U "Consulter repas" coté web.....	47
Figure 3-14 Diagramme d'état d'une commande.....	48
Figure 3-15 Diagramme de navigation.....	49
Figure 3-16 Maquettage de l'interface web "Calculer Mes Besoins"	50
Figure 3-17 Maquettage de l'interface mobile "Calculer Mes Besoins"	51
Figure 3-18 Maquettage de l'interface web "Consulter commandes"	52
Figure 3-19 Maquettage de l'interface mobile "Consulter commandes"	52
Figure 3-20 Maquettage de l'interface "Consulter mes commandes"	53
Figure 3-21 Maquettage de l'interface "Admin"	54
Figure 4-1 Fonctionnement générale du Framework Symfony.....	58
Figure 4-2 Squelette d'un projet Symfony.....	59
Figure 4-3 Capture d'écran de la configuration de la BD.....	59
Figure 4-4 Capture d'écran d'entité "Commande".....	60
Figure 4-5 Captures d'écran d'un extrait de migrations.....	61
Figure 4-6 Capture d'écran du classe CommandeRepository.....	62
Figure 4-7 Capture d'écran du classe Migration de l'entité Commande.....	62
Figure 4-8 Algorithme de calcul des besoins nutritifs.....	63
figure 4-9 Architecture du Framework Flutter.....	64
Figure 4-10 Structure de mon projet Flutter.....	65
Figure 4-11 Capture d'écran des bibliothèques invoqués dans le Framework Flutter.....	66
Figure 4-12 Capture d'écran de la fonction d'authentification via une requête HTTP POST.....	66
Figure 4-13 Interface "Accueil" de l'application Web.....	67
Figure 4-14 Interfaces "Accueil" et "Drawer" de l'application mobiles.....	67
Figure 4-15 Interface "Login" de l'application Web.....	68

Figure 4-16 Interfaces "Login User" et "Login Traiteur" de l'application mobile.....	68
Figure 4-17 Interface "Calculer Mes Besoins" de l'application web.....	69
Figure 4-18 Interface "Calculer Mes Besoins" de l'application mobile.....	69
Figure 4-19 Interface "Profil" de l'application web.....	70
Figure 4-20 Interface "Profil" de l'application mobile.....	70
Figure 4-21 Interface "Mes commandes" de l'application web d'utilisateur.....	71
Figure 4-22 Interface "Mes commandes" de l'application mobile d'utilisateur.....	71
Figure 4-23 Interface "Mes commandes" du traiteur.....	72
Figure 4-24 Interface "Admin"	72

LISTE DES TABLEAUX

Tableau 1-1 Détails de la société ONEDEV.....	3
Tableau 1-2 Analyse des applications existantes.....	8
Tableau 2-1 Description de C.U "Gérer profil".....	23
Tableau 2-2 Description de C.U "Consulter Repas".....	26
Tableau 2-3 Description de C.U "Gérer commande".....	27
Tableau 2-4 Description de C.U "Gérer compte Traiteur".....	29
Tableau 4-1 Environnement Matériel.....	54
Tableau 4-2 Environnement Logiciel.....	55

Introduction Générale

De nos jours, le temps fournit les moyens, le gain de temps et la simplification du quotidien sont les plus demandés par l'Homme. Donc, les services utilisateur permettant le gain de temps sont d'un intérêt primordial. Donc les services fournis aux utilisateurs leurs permettant un gain de temps, sont d'un intérêt majeur.

De plus, on trouve en Tunisie que les services et les plateformes de livraison à domicile sont devenus de plus en plus appréciés, en particulier, la livraison des Fast Food. Cependant ces services manquent de qualité, et livrent des repas malsains.

Le bon service, le gain du temps et une nourriture saine sont les trois principes de bases de l'idée de mon application, une application web et mobile destinées aux utilisateurs pour qu'ils puissent avoir et commander des repas sains. Mon application permet aux utilisateurs inscrits de calculer leurs besoins nutritifs et de commander des repas qui répondent à leurs besoins et les faire livrer.

Mon rapport de projet de fin d'études est scindé en quatre chapitres : Au niveau du premier chapitre, **Contexte et objectifs**, je présente le contexte de mon projet, l'idée émergente et le travail à réaliser. Dans le deuxième chapitre, **Analyse et spécification des besoins**, je spécifie les besoins fonctionnels et non-fonctionnels. Dans le troisième chapitre, **Conception**, je présente la conception de mon application en invoquant les diagrammes appropriés. Dans le quatrième chapitre, **Réalisation**, je décris l'environnement de développement matériel et logiciel et les différentes interfaces de mon application. Et finalement, je clôture ce rapport par une conclusion générale du travail accompli suivi par des perspectives de mon projet.

CONTEXTE ET OBJECTIFS

Introduction

Dans ce chapitre je présente le cadre du projet et l'organisme d'accueil. Ensuite, je motive pour le développement de l'application Meal Prep. Ainsi que j'effectue l'étude de l'existant qui m'a permis de raffiner de plus les fonctionnalités requises et éviter les lacunes des solutions existantes.

1 Présentation générale du projet

1.1 Cadre du projet

Le présent travail s'inscrit dans le cadre du projet de fin d'étude en vue de l'obtention du Diplôme de Licence Appliquée en Informatique de l'Institut Supérieur d'Informatique et Mathématique de Monastir.

Le stage a été effectué au sein de la société « OneDev » d'une durée de trois mois du 15 Février au 15 Mai, où j'ai identifié et implémenté le sujet du projet.

1.2 Organisme d'accueil

« OneDev » est une startup spécialisée dans le domaine d'informatique. Elle est implantée à Ksar Hellal Monastir depuis 2016 et elle a pour mission de développer et créer des logiciels. En effet, « OneDev » se spécialise aux technologies PC SOFT pour réaliser des applications Desktop par WinDev, des applications mobiles par WinDev Mobile, et des sites web par WebDev et PHP. De même, elle est spécialisée dans la vente de matériels informatique et l'installation réseau.

Adresse : Rue Hadj Ali Soua Imm. Le Palmyre Bureau 103 Etage 1, Ksar Hellal 5070

Téléphone : (00 216) 52985659

Site Web : www.onedev.com.tn

Logo :



Le tableau suivant donne plus de détails sur la société « OneDev ».

Raison sociale	ONEDEV
Forme juridique	SUARL
Capital	2600 DT
Domaine d'activité	Activité informatique

Tableau 1.1 - Détails sur la société OneDev

2. Motivation pour « Meal Prep »

Meal Prep est le concept de préparer des repas ou des plats entiers à l'avance. Le fait d'avoir des repas préparés à l'avance peut également réduire la taille des portions et vous aider à atteindre vos objectifs nutritionnels. De cette façon, vous éviterez les options malsaines comme les dîners télévisés ou les plats à emporter, surtout lorsque vous êtes épuisé ou occupé.

Le sujet du projet consiste à développer une application web et une application mobile compatible avec les deux systèmes d'exploitation Android 10 et iOS 13.3.1. Cette plateforme permet aux utilisateurs de calculer leurs besoins nutritifs et de préparer des repas qui répondent à leurs besoins et les faire livrer.

Le but de mon projet est de créer et développer une application web et mobile pour :

- Faciliter la procédure de livraison des repas.
- Garantir le bon service aux clients.
- Gagner de temps et atteindre la satisfaction.
- Calculer les besoins nutritifs d'un client en ordre de bien préparer son repas.

- Une application mobile et web est bien souvent plus ergonomique que l'utilisation des appels téléphoniques. Elle rend l'expérience de l'utilisateur bien meilleure grâce à la navigation.

3. Etude de l'existant

L'étude de l'existant a pour but de raffiner de plus l'idée de mon projet. Il m'aide de s'inspirer de l'existant pour dégager les principales fonctionnalités à enrichir mon application.

3.1 Solutions existantes

Afin de remplacer le travail classique du *Meal Prep*, plusieurs plateformes sont conçues et développées pour offrir un système plus moderne et efficace aux utilisateurs.

Dans ce que suit, je présente quelques solutions existantes dans le but de dégager les principales fonctionnalités à enrichir dans mon application.

Sunbasket.com¹ :

Sunbasket.com est un service basé sur *San Francisco- États-Unis* fondé en 2014 qui a embauché plus que 300 employés.

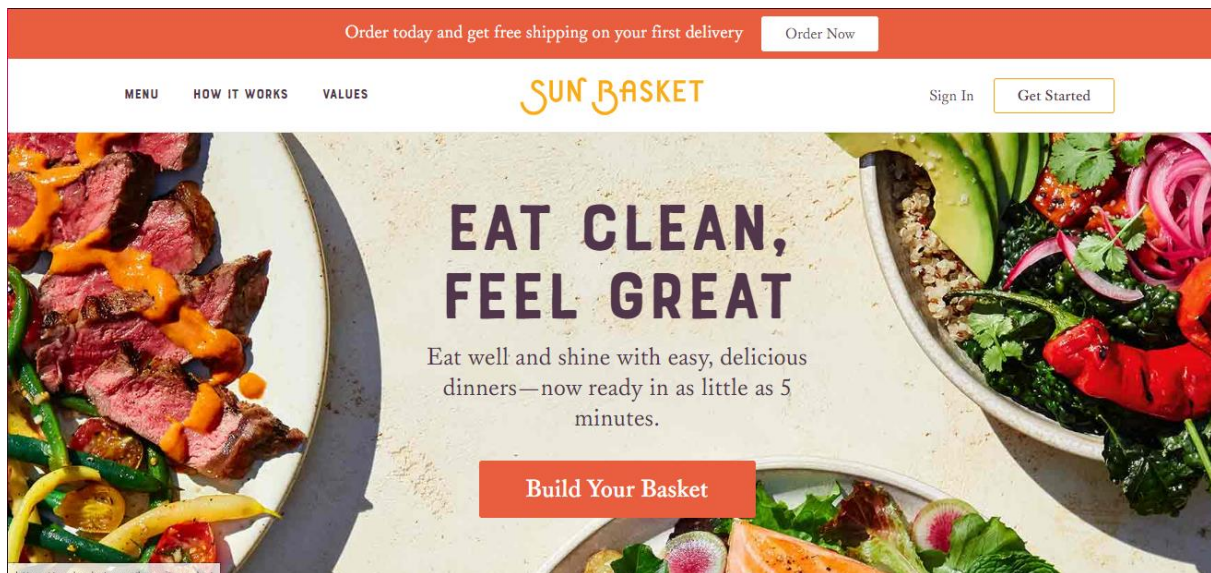


Figure 1-1. Page d'accueil du site Sunbasket.com

Ce site met à la disposition des clients un menu largement varié avec plusieurs des choix sur le type de repas désiré.

Sunbasket.com offre un ensemble des services comme :

- Faire des dons avec des repas.
- Choisir les repas selon leurs durées de préparation.
- Contacter l'admin pour faire des réclamations.
- Payer en ligne.

Parsleybox.com² :

Parsleybox.com est un service de *Meal Prep* basé sur *Edinburgh – Scotland* fondé en 2017 qui fonctionne dans tout le Royaume Uni.

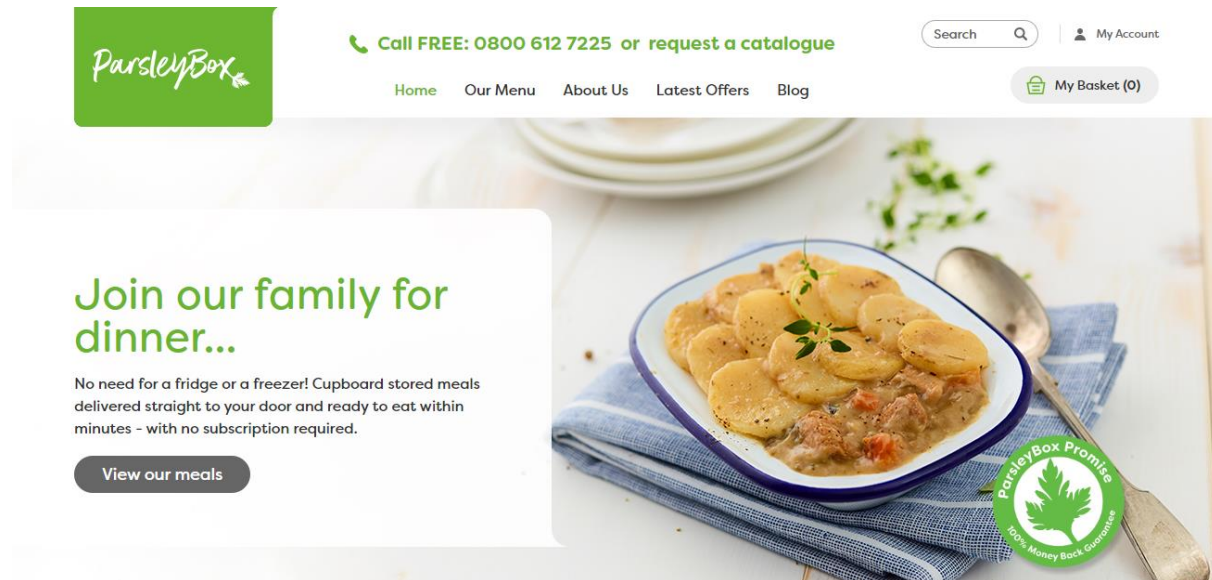


Figure 1-2 Page d'accueil du site Parsleybox.com

Parsleybox.com offre à ses clients une plateforme web qui sert à :

- Choisir un de ses paquets offerts.
- Saisir le nombre de repas désirés.
- Livrer le paquet choisi avec des repas cuits lentement.

Blueapron.com³ :

Blueapron.com est un service de *Meal Prep* qui opère à l'*États-Unis*, fondé en 2012 et en 2018 Blue Apron atteint un chiffre d'affaires de 667 millions USD (*United State Dollar*).

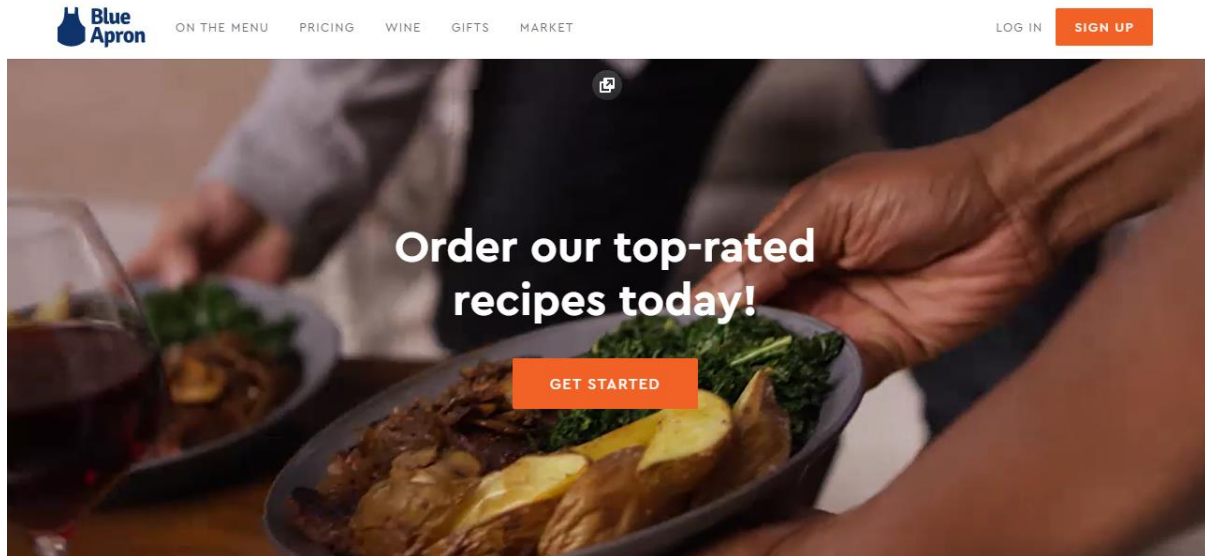


Figure 1-3 Page d'accueil du site Blueapron.com

Dans l'objectif de faire satisfaire ses clients, Blue Apron permet aux utilisateurs de :

- Choisir un plan de nutrition pour une semaine.
- Suivre le déroulement de la commande.
- Recevoir un paquet d'ingrédients avec les recettes de préparation.

3.2 Analyse de l'existant

Chaque plateforme mentionnée précédemment offre des services et des fonctionnalités spécifiques. J'ai dégagé les 9 principaux critères pris en considération dans le processus d'évaluation de ces applications dans le tableau suivant en ordre de développer ces derniers dans mon application.

	SunBasket	ParsleyBox	BlueApron
Application mobile	Oui	Non	Oui
Gestion du profil	Non	Non	Non
Calcule des besoins	Non	Non	Non
Détails du repas	Oui	Non	Oui
Livraison	Oui	Oui	Oui
Suivi du commande	Non	Oui	Oui
Actualités	Non	Oui	Oui
Choix de la catégorie des repas	Non	Oui	Non
Disponibilité à Tunisie	Non	Non	Non

Tableau 1.2 - Analyse des applications existantes

- Application mobile : Est-ce que ce site possède une application mobile associé à son site web ou non ?
- Gestion du profil : L'utilisateur possède un profil avec ses informations personnelles et qu'il peut les modifier.
- Calcule des besoins : Le calcul des besoins nutritionnels de l'utilisateur afin de préparer son repas.
- Détails du repas : L'utilisateur peut consulter les détails de tous les repas disponibles aux menus.
- Livraison : Le site livre les repas commandés.

- Suivi de commande : L'utilisateur peut suivre l'état de sa commande (En préparation, En route, Livrée).
- Actualités : Le site publie souvent d'actualités sur la nutrition saine.
- Choix de la catégorie des repas : L'utilisateur peut filtrer le menu selon une catégorie de repas choisie.
- Disponibilité à Tunisie : Le site fonctionne pour les utilisateurs localisés en Tunisie.

4. Travail à réaliser

L'étude de l'existant m'a permis d'inspirer de ces plateformes et de dégager plusieurs faiblesses dans les plateformes étudiés. Elles ne donnent pas assez d'importance à le fait que chaque personne a ses propres besoins nutritifs, et qu'il doit avoir son repas préparé selon ses besoins. De ce fait, il est primordial de proposer une nouvelle solution qui à la fois optimise les plateformes mentionnées précédemment, et surtout fonctionne en Tunisie.

Ainsi, j'ai identifié les principales fonctionnalités que mon application doit offrir :

- Une application mobile sous Android et iOS.
- Une application web.
- La gestion du profil personnel.
- Le calcul des besoins nutritifs.
- Les détails de chaque repas.
- Le service de livraison.
- Le suivi des commandes.
- Le publie fréquent d'actualités.
- Une base de données qui conserve tous les détails des clients.

5. Méthodologie de conception

Dans cette partie, je vais spécifier la méthodologie de conception adopté afin de déterminer la démarche de conception. On y trouve une comparaison entre les méthodologies classiques et le processus unifié.

5.1 Méthodologie en cascade

La méthodologie classique est une succession de phases qui se déverse les unes dans les autres et on ne peut pas commencer une phase de l'approche classique tant que la précédente n'est pas terminée. Alors le plan d'un projet se faire par étape.

La figure 1.4 présente les étapes fondamentales de la méthodologie classique.

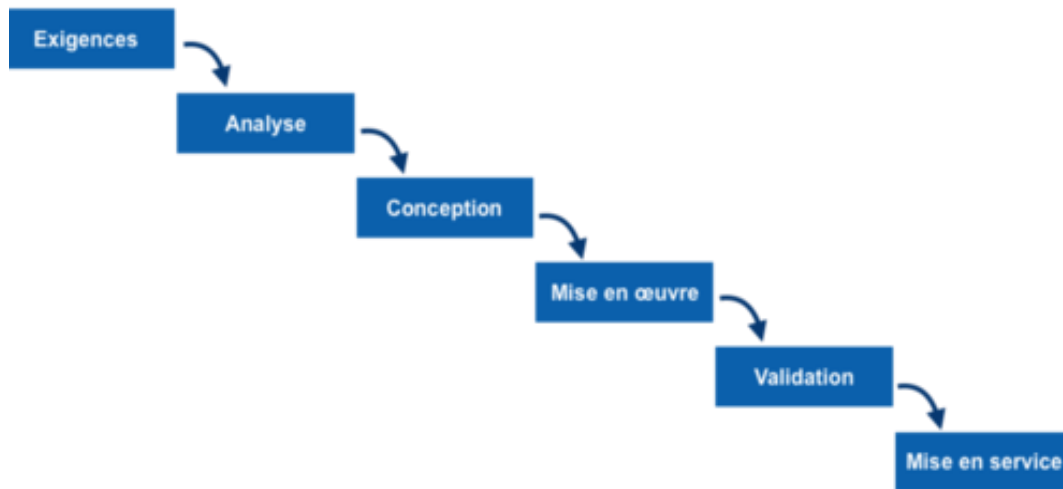


Figure 1-4 Représentation générale de la méthodologie classique

Parmi ces méthodologies classiques, on y trouve le modèle de cycle en V. La figure 1.5 présente le cycle de développement en V⁴.

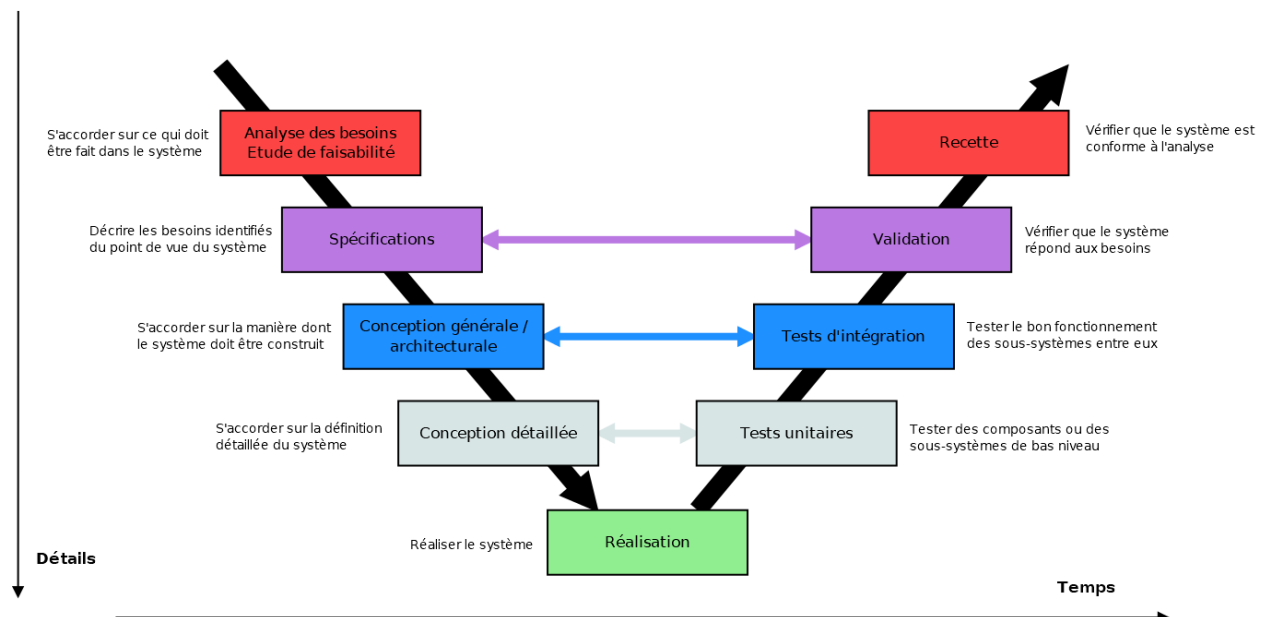


Figure 1-5 Représentation du modèle de cycle en V

La figure 1.4 et 1.5 montre que la méthodologie classique limite le retour aux étapes précédentes.

5.2 Processus Unifié (PU)

Le processus unifié (PU)⁵ est une famille de méthodes de développement de logiciels orientés objets. Elle se caractérise par une démarche itérative et incrémentale, pilotée par les cas d'utilisation, et centrée sur l'architecture et les modèles UML. Elle définit un processus intégrant toutes les activités de conception et de réalisation au sein de cycles de développement composés d'une phase de création, d'une phase d'élaboration, d'une phase de construction et d'une phase de transition, comprenant chacune plusieurs itérations. Les figures 1.6 et 1.7 présente les phases fondamentales et le cycle de développement du processus unifié.

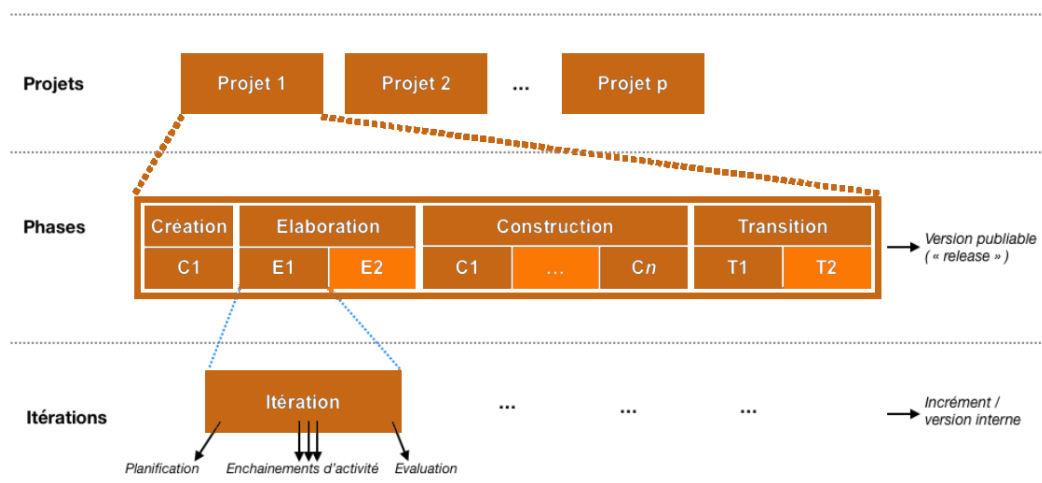


Figure 1-6 Les étapes fondamentales du PU

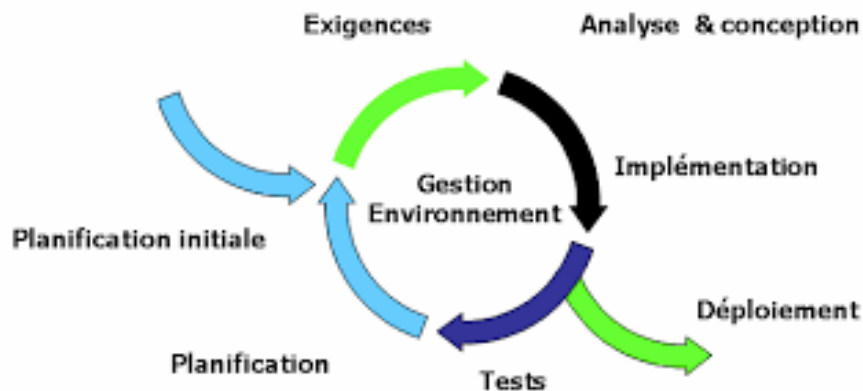


Figure 1-7 Représentation de cycle de développement du PU

5.3 Choix de la méthodologie de conception

Après une étude comparative des méthodologies classiques et du processus unifié, je considère que le processus unifié (PU) est le plus adaptable à mon projet. En ce qui suit, je fais une comparaison entre les méthodologies les plus utilisés du processus unifié afin de choisir la méthodologie la plus ajustée.

5.3.1 Rational Unified Process (RUP)

Rational Unified Process (RUP) est à l'origine du processus unifié et est à ce titre l'implémentation la plus connue. La méthode est livrée clés en main et est accompagnée d'outils pour guider les équipes dans l'adaptation et exécution du processus.

Le cadre au développement logiciel proposé par RUP répond aux caractéristiques du processus unifié : il s'agit d'une méthode de développement guidée par les besoins des utilisateurs, centrée sur l'architecture logicielle, itérative et incrémentale, mettant en œuvre UML pour la modélisation.

RUP étend les enchainements d'activités pour couvrir également :

- La modélisation métier ou modélisation d'affaires (« Business modeling » en anglais) afin de différencier au niveau des exigences les impératifs organisationnels propre à l'environnement métier des utilisateurs d'une part, et les exigences propre au système d'autre part. Cette modélisation des activités d'entreprise est également basée sur UML.
- La gestion de projets, pour prendre en compte les particularités du développement logiciel tout en assurant une conformité avec les normes existantes dans ce domaine.

5.3.2 Two Track Unified Process (2TUP)

2TUP est un processus de développement logiciel qui implémente le processus unifié. Chacune des étapes du cycle découle des précédentes. Il préconise un cycle de vie en Y, et s'apparente à un cycle de développement en cascade, par ailleurs elle est incrémentale. À partir de la capture des besoins fonctionnels, on définit plusieurs cas d'utilisation représentant chacun un incrément du cycle de développement. Elle favorise des formes de recherche de qualité et de performance intéressante telle que le service réutilisation. Ainsi, 2TUP fait une large place à la technologie et la gestion des risques, et s'articule autour de l'architecture du système, c'est pourquoi cette méthodologie convient parfaitement avec la conception de mon application. La figure 1.8 montre la méthode de 2TUP (*Two Track Unified Process*)

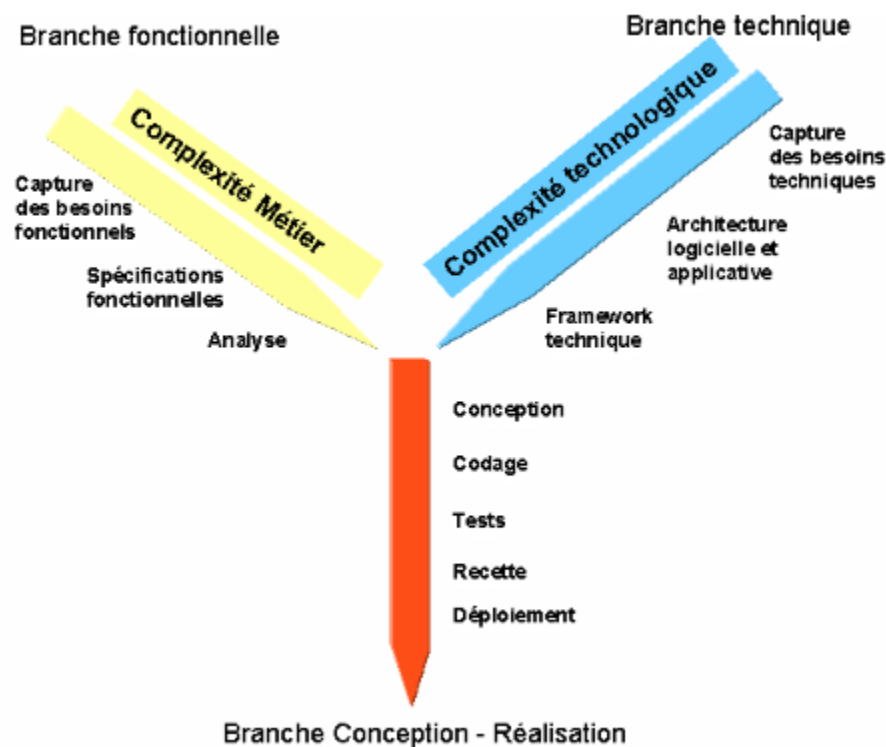


Figure 1.8 - Les étapes de développement de 2TUP

5.4 Méthodologie adoptée : 2TUP

Suite à cette étude comparative de RUP et 2TUP et afin d'éviter les risques et de mener à bon terme ce projet, j'ai décidé d'adopter le processus 2TUP. En effet, mon projet est basé sur un

processus de développement qui commence par la détermination des besoins fonctionnels et techniques, qui s'arrête à la conception, et qui finalise par la recette finale. C'est pour cela, j'ai besoin d'une méthodologie qui dissocie les aspects techniques et fonctionnels tout en commençant par une étude préalable. Donc, mon choix pour la méthodologie est porté vers la méthode 2TUP vu qu'elle est désignée par une approche qui convient avec le cadre du projet.

6. Déroulement du projet

Pour le bon déroulement du mon projet, j'essaie de suivre le chronogramme ci-dessous qui donne une idée approximative sur le temps alloué pour chaque tâche.

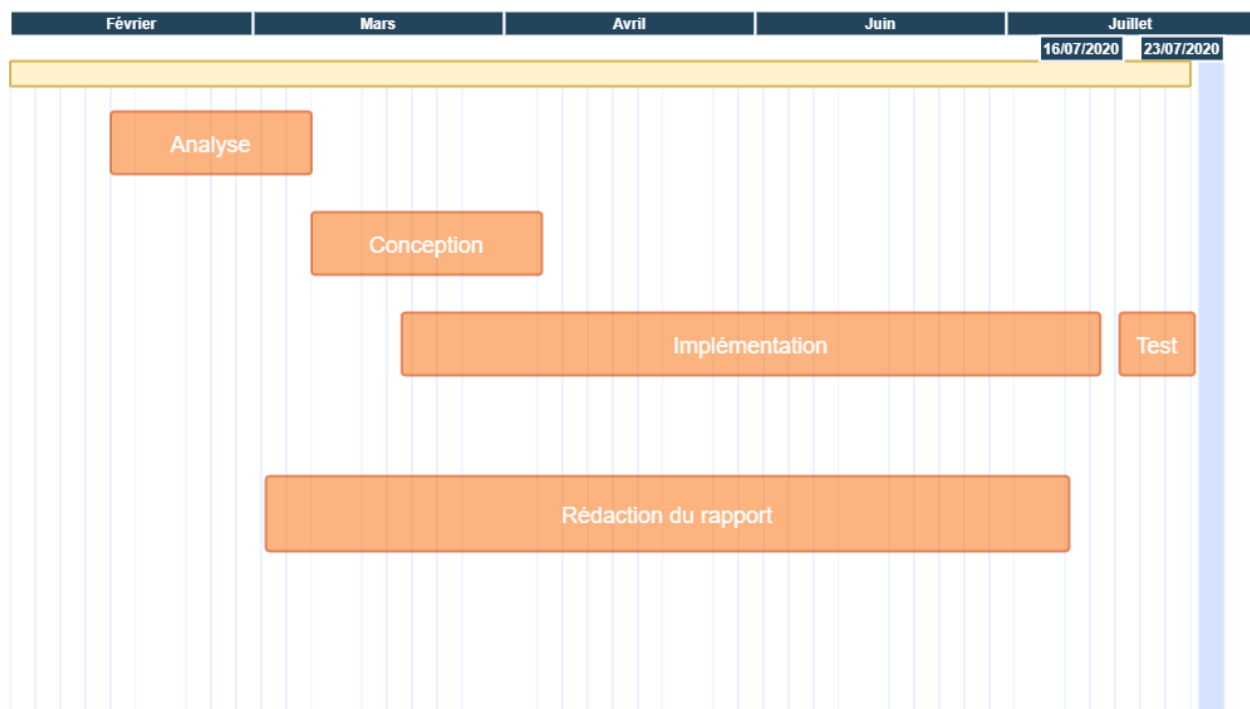


Figure 1.9 – Chronogramme de déroulement du projet

Conclusion

Dans ce chapitre, j'ai mis mon projet dans son contexte général. J'ai commencé par la présentation de la société « OneDev », ensuite j'ai présenté le cadre de mon projet avec une étude sur les solutions existantes, afin de raffiner de plus le sujet de mon projet et identifier les fonctionnalités attendues. Enfin, j'ai présenté la méthodologie de développement que j'ai adopté.

ANALYSE ET SPECIFICATION DES BESOINS

Introduction :

La partie analyse et spécification des besoins est la partie la plus importante dans l'implémentation de n'importe quel projet. Dans ce chapitre, je vais identifier les acteurs du système et on va extraire les besoins fonctionnels et non-fonctionnels. Par la suite, je vais présenter les diagrammes de cas d'utilisation et les diagrammes de séquence accompagnés avec des descriptions détaillées. Je clôture ce chapitre par une explication de mon choix concernant les besoins techniques.

1. Analyse des besoins fonctionnels

J'analyserai les différents besoins fonctionnels de mon application commençant par l'identification des acteurs.

1.1 Identification des acteurs

Un acteur représente un rôle joué par une entité externe (Utilisateur humain, dispositif matériel, ou autre système) qui interagit avec le système. Mon projet dispose quatre acteurs :

- **Administrateur** : C'est le rôle primordial dans l'assurances de bon fonctionnement du système, c'est la personne qui prend en charge la gestion des comptes clients et des comptes traiteurs.
- **Traiteur** : C'est le rôle joué par les traiteurs pour surveiller le processus de livraison.
- **Membre** : C'est le rôle central joué par les membres qui inscrivent dans mon application.

- **Utilisateur :** C'est le rôle joué par les utilisateurs qui ne sont pas inscrits dans mon application.

1.2 Expression des besoins fonctionnels

Mon système doit satisfaire les besoins fonctionnels suivants :

- Fonctions orientées aux utilisateurs et membres :

Consulter l'actualité du régime : Chaque utilisateur sans avoir créer un compte ou s'authentifier, a l'accès à les articles du régime qui sont publiés dans mon application.

Consulter les recettes des repas sains : L'utilisateur peut accéder chaque jour à des nouvelles délicieuses recettes.

Consulter la liste des repas préparés : Les utilisateurs peuvent consulter le menu de repas avec leurs valeurs nutritionnels mais sans faire des commandes (pour les utilisateurs).

Faire une demande de partenariat : L'utilisateur peut remplir un formulaire s'il est un traiteur ou bien un propriétaire d'un restaurant en ordre de devenir un partenaire.

- Fonctions orientées aux membres :

Gérer profil : Le membre peut gérer son profil tout en modifiant ses informations personnelles.

Consulter ses commandes : Le membre peut consulter la liste de toutes les commandes qu'il a passé et celles qui sont déjà en cours.

Calculer ses besoins nutritifs : Le membre peut calculer ses besoins nutritifs en fonction de ses données personnelles (âge, sexe, taille, poids, niveau d'activité, objectif) afin d'avoir une liste des repas qui conviennent avec ses besoins.

Faire une commande : Le membre est apte à commander n'importe quel repas.

Contacter l'admin : Les membres peuvent contacter l'admin pour une enquête ou pour réclamer à un problème.

- Fonctions orientées aux traiteurs :

Gérer les commandes : Le traiteur peut consulter la liste des commandes tout en modifiant le statut de déroulement de chaque commande.

Ajouter un repas : Le traiteur est autorisé à ajouter un repas dans son menu.

Contacteur l'admin : Les traiteurs peuvent contacter l'admin pour toutes réclamations.

▪ Fonctions orientées au admin :

Consulter toutes les commandes : L'admin est apte à consulter toutes les commandes passées par les utilisateurs.

Gérer les comptes traiteur : L'admin est autorisé à gérer les comptes traiteur, par ajouter un nouveau traiteur, supprimer un traiteur, ou modifier les données d'un traiteur.

Gérer les comptes membre : L'admin peut consulter la liste des comptes membre, tout en supprimant un ou plusieurs membres.

Télécharger un article : L'admin peut télécharger un article et le publier sur la plateforme.

2. Analyse et spécification des besoins non fonctionnels

Dans cette partie, j'analyserai les différents besoins non fonctionnels comme les règles à respecter pendant l'implémentation de l'application, et les besoins techniques nécessaires.

2.1 Les besoins non fonctionnels

Les besoins non fonctionnels représentent les règles à respecter afin d'assurer une bonne qualité d'application à achever, et le bon fonctionnement du futur système. Quant aux besoins non fonctionnels, ils se récapitulent en :

- **Sécurité :** La sécurité de mon système est un must, cela signifie que les données des clients, des traiteurs et même du serveur doivent être bien protégées. Pour cela, tous les mots de passe vont être cryptés, et l'admin peut passer à son espace administratif avec un url personnalisé.
- **Performance et fiabilité :** Pour rendre le système plus fiable et plus performant, il doit être en mesure d'exécuter ses fonctions rapidement, ainsi que le temps de réponse, les délais de rafraîchissement et le chargement de l'application, que ce soit mobile ou web.

- **Ergonomie** : L'ergonomie est l'adaptation d'un environnement de travail aux besoins de l'utilisateur, pour qu'il peut apprendre à exploiter et interpréter le système, et de se familiariser rapidement avec le contenu. D'autre part, pour assurer la convivialité du système, il dispose des messages d'erreurs, des messages informatifs qui aident l'utilisateur et des interfaces graphiques bien formés et bien lisibles.
- **Disponibilité** : J'ai assuré que le système sera disponible pour les utilisateurs en trois versions, la version web, la version Android, et la version iOS. Ainsi qu'il sera extensible de sorte qu'il pourra y avoir des modifications.

2.2 Les besoins techniques

Afin d'implémenter une application qui sera disponible sur les plateformes web, Android et iOS, j'ai ciblé les technologies les plus performantes en termes de rapidité, sécurité et simplicité. Et pour cela, j'ai fixé les contraintes techniques suivantes :

- Développer une application web en utilisant le Framework Symfony.
- Développer une application mobile multi-plateformes en utilisant le Framework Flutter.
- Utiliser la base de données MySQL.
- Les fonctionnalités du Back-End sont fournis par des services REST

Dans la suite, je justifie mon choix pour ces besoins techniques.

• Symfony

En informatique, l'utilisation d'un Framework a pour objectif d'aider à implémenter des applications plus rapidement. En web, le Framework PHP est généralement livré avec des composants et modules génériques qui peuvent être réutilisés pour rendre le développement d'applications web plus facile et plus rapide. Parmi les Frameworks les plus plébiscités par les développeurs aujourd'hui, Symfony 4 semble être particulièrement apprécié pour ses performances et sa simplicité d'utilisation.

Symfony [6] est un ensemble de composants ou bibliothèques PHP réutilisables qui facilitent le développement web en réduisant de façon considérable le temps et l'effort requis pour créer des composants génériques. Il même utilise des projets open source PHP existants dans le cadre du Framework, y compris :

- ORM Doctrine : Pour manipuler les bases de données (Orienté Objet)
- Twig : Moteur de modèles (*Views*)
- Swift Mailer : Bibliothèque d'E-mails
- PHPUnit : Framework de test unitaire

J'ai considéré l'utilisation de Symfony 4 comme choix technique dans l'implémentation de l'application web pour les raisons suivantes :

- **Flexible** : Symfony est complètement configurable. Il est d'ailleurs souvent considéré comme le meilleur Framework pour la création d'applications web hautement sécurisées.
- **Performant et facile à utiliser** : Le Framework Symfony offre une grande flexibilité pour les utilisateurs débutants et avancés. La documentation, les forums et le soutien de la communauté le rendent très facile à utiliser.
- **Facile à tester** : Avec PHPUnit (langage informatique sur lequel se base Symfony), Symfony offre la première couche de test fonctionnel qui stimule les requêtes HTTP et examine les résultats sans avoir à écrire de script à l'aide d'outils de test.

- **Flutter**

Pour le développement natif, le problème majeur des développeurs est d'implémenter des applications mobiles de qualité et multiplateformes. Flutter a changé la donne.

Flutter [7] est un Framework open source développé par Google. Il s'appuie sur le langage de programmation DART, et il est capable de faire du développement multiplateforme de manière rapide et simplifiée. D'autre manière, le même code est exécutable sur Android et iOS.

Le choix de l'utilisation de Flutter est pris à la suite de ses avantages :

- **Interface utilisateur** : Flutter est le seul Framework qui offre les outils d'implémenter une interface utilisateur de très bonnes performances sur les deux plateformes, avec une conception design bien structurée.
- **Temps de codage réduit** : Non seulement que Flutter offre le développement multiplateforme qui déjà réduit le temps de codage, mais il offre la fonctionnalité « Hot

Reload », qui en fait permet au développeur de voir les changements appliqués instantanément, sans perdre l'état actuel de l'application.

- **Une maintenance accélérée et optimisée :** Les corrections de bugs sont rapides et régulières.

La figure 2.1 présente une comparaison de taux d'intérêts entre le Flutter et le React Native présentés par Yojji.io (<https://yojji.io/blog/flutter-vs-react-native>).

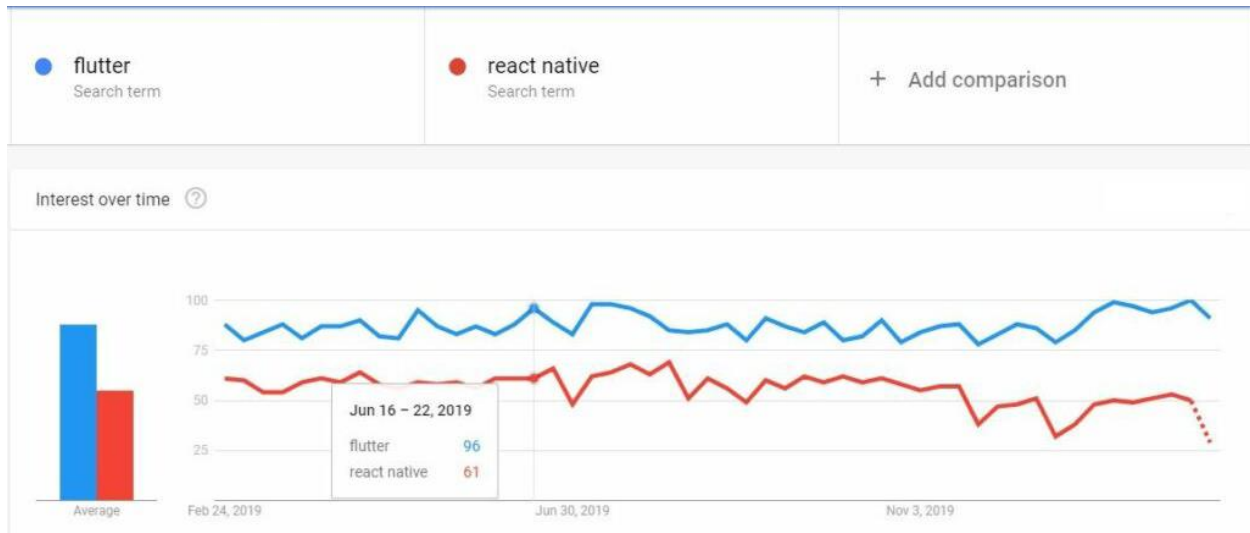


Figure 2.1 - Taux d'intérêt entre le Flutter et le React Native

• MySQL

MySQL [8] est la base de données open source la plus populaire au monde (Voir figure 2.2). Bien qu'elle soit avant tout connue pour son utilisation par des sociétés Web, MySQL est également une base de données embarquée très populaire.

J'ai considéré l'utilisation de MySQL pour les raisons suivantes :

- **Facilement interrogeable via PHP et Flutter**
- **Facilité d'administration avec phpMyAdmin**
- **Modèle Relationnel**
- **Léger, portable, gratuit**

La figure 2.2 présente une comparaison entre les 10 plus reconnus base de données en termes d'utilisation en Juin 2020 présentés par Digore.com (<https://www.digora.com/fr/blog/TOP-10-des-bases-de-donnees>).

Rank			DBMS	Database Model	Score		
Jun 2020	May 2020	Jun 2019			Jun 2020	May 2020	Jun 2019
1.	1.	1.	Oracle +	Relational, Multi-model	1343.59	-1.85	+44.37
2.	2.	2.	MySQL +	Relational, Multi-model	1277.89	-4.75	+54.26
3.	3.	3.	Microsoft SQL Server +	Relational, Multi-model	1067.31	-10.99	-20.45
4.	4.	4.	PostgreSQL +	Relational, Multi-model	522.99	+8.19	+46.36
5.	5.	5.	MongoDB +	Document, Multi-model	437.08	-1.92	+33.17
6.	6.	6.	IBM Db2 +	Relational, Multi-model	161.81	-0.83	-10.39
7.	7.	7.	Elasticsearch +	Search engine, Multi-model	149.69	+0.56	+0.86
8.	8.	8.	Redis +	Key-value, Multi-model	145.64	+2.17	-0.48
9.	9.	11.	SQLite +	Relational	124.82	+1.78	-0.07
10.	11.	10.	Cassandra +	Wide column	119.01	-0.15	-6.17

▪ Figure 2.2 - Taux d'utilisation des différents SGBD en Juin 2020

3. Spécification des besoins fonctionnels

Dans cette partie, je vais détailler les différents cas d'utilisations majeurs de mon application par des diagrammes de cas d'utilisations et des diagrammes de séquence, qui sont des diagrammes UML utilisés pour donner une vision globale du comportement fonctionnel du futur système.

3.1 Cas d'utilisation générale

D'abord, je détaille les cas d'utilisation de chaque acteur et les interactions entre eux dans le diagramme de cas d'utilisation générale présenté par la figure 2.3.

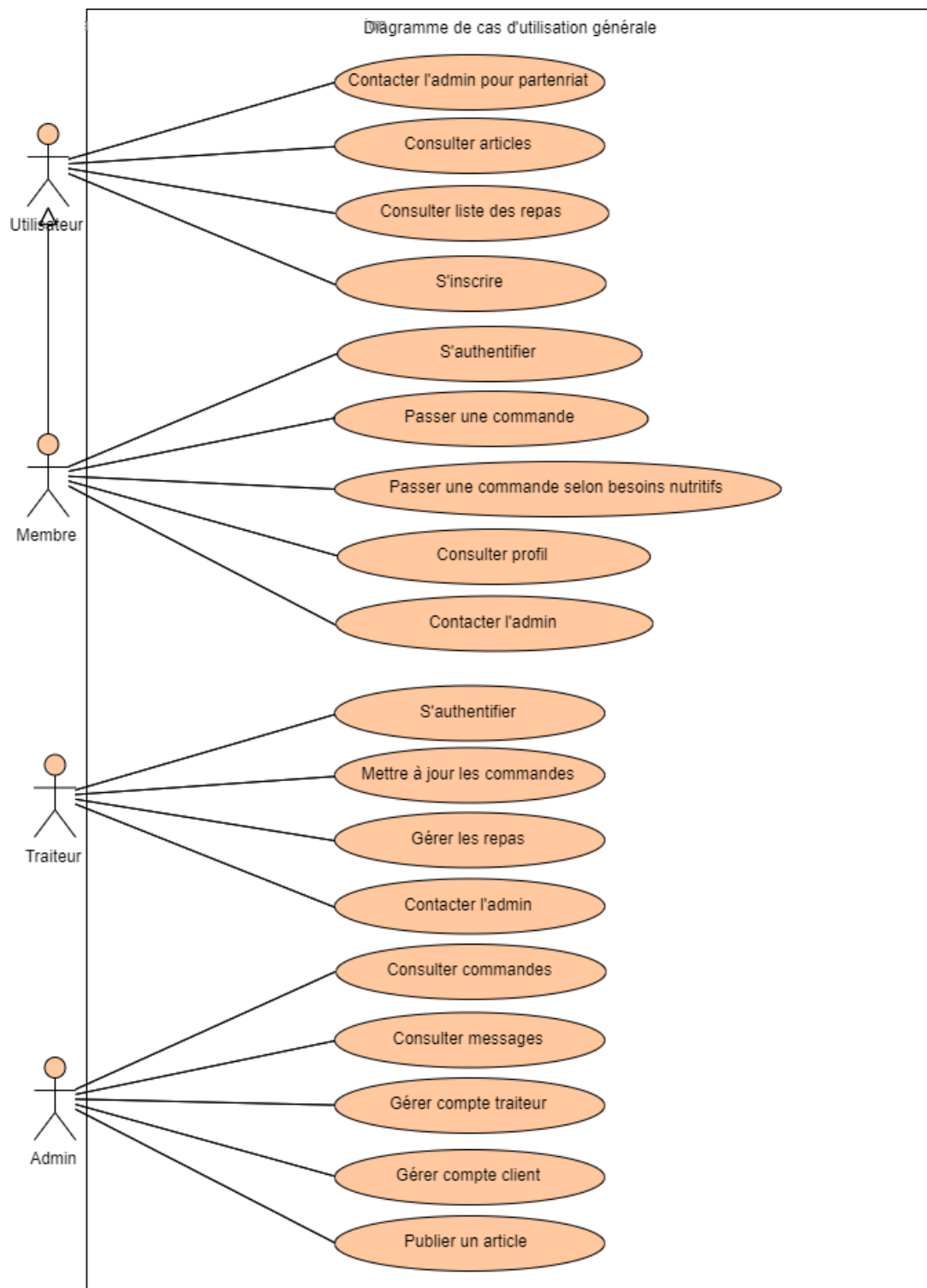


Figure 2.3 – Diagramme de cas d'utilisation générale

3.2 Cas d'utilisation « Consulter historique commande »

- **Diagramme de cas d'utilisation**

Un membre peut consulter son profil où il peut modifier ses informations et consulter l'historique de ses commandes.

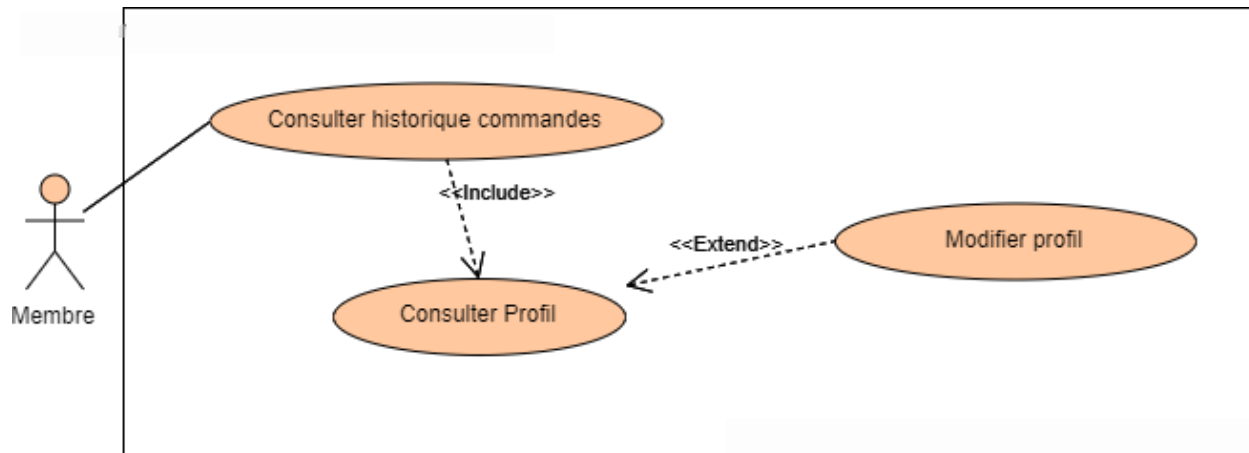


Figure 2.4 – Diagramme de CU "Consulter historique commande"

- **Diagramme de séquence**

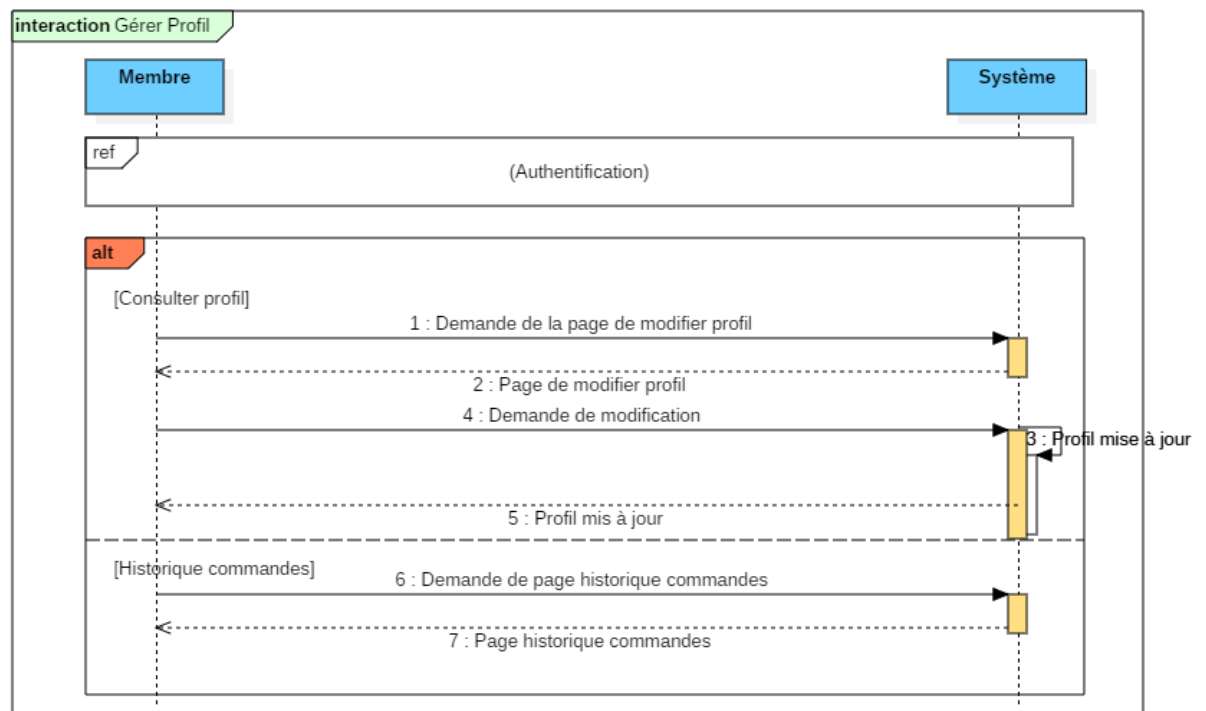


Figure 2.5 – Diagramme de séquence "Consulter historique commande"

Cas d'utilisation	CONSULTER HISTORIQUE COMMANDE
Acteurs	Membre
Pré condition	Membre authentifié
Description du scénario principal	1-Consulter profil -Le membre demande la page de modifier profil. -Le système renvoie la page de modifier profil. -Le membre demande de modifier son profil. -Le système renvoie le profil mis à jour. 2-Historique commandes -Le membre demande la page historique commandes. -Le système renvoie la page historique commandes.
Post condition	Historique commande consulté
Exception	-S'il y a une erreur de saisie le système affiche un message d'erreur

Tableau 2.1- Description de CU "Consulter historique commande"

3.3 Cas d'utilisation « Passer une commande »

- **Diagramme de cas d'utilisation** « Passer une commande selon besoins »

Un membre peut calculer ses besoins nutritifs en fonctions de ses données, afin d'accéder à la liste des repas qui conviennent avec ses besoins et commander ceux qu'il préfère.

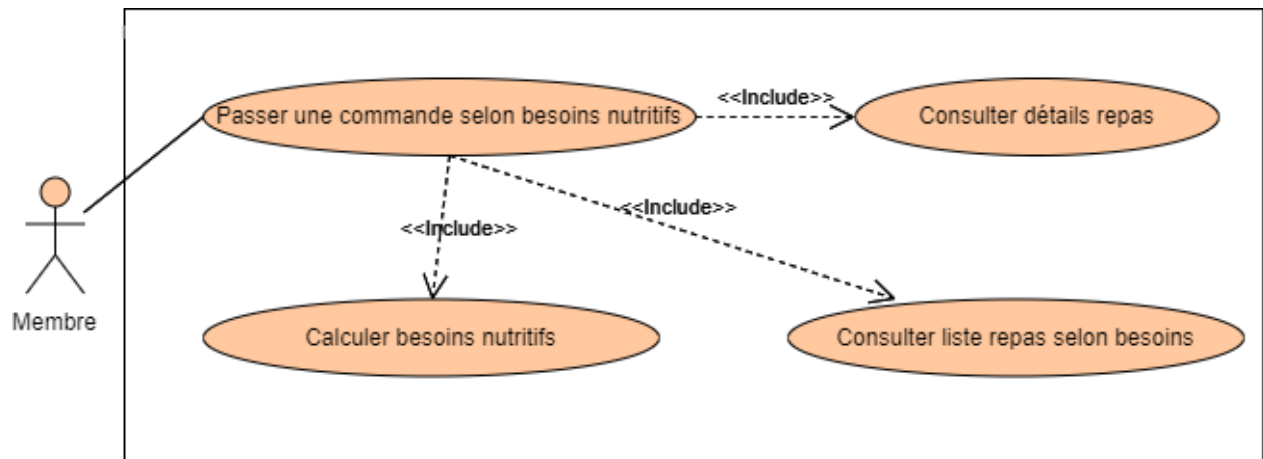


Figure 2.6 – Diagramme de CU "Passer une commande selon besoins"

▪ Diagramme de cas d'utilisation « Passer une commande »

Un membre peut consulter la liste des repas préparés afin de commander ceux qu'il préfère.

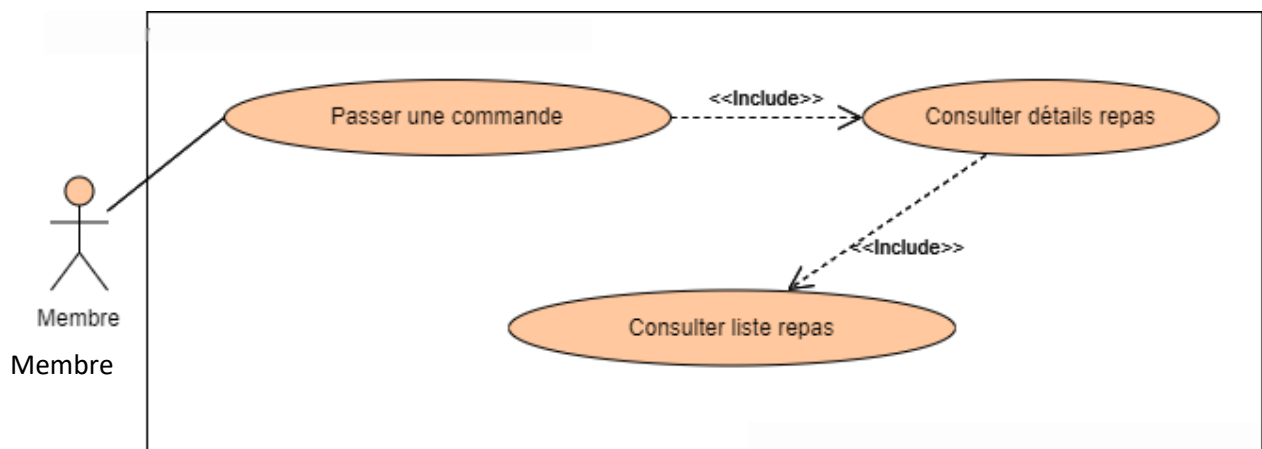


Figure 2.7 – Diagramme de CU "Passer une commande"

• Diagramme de séquence

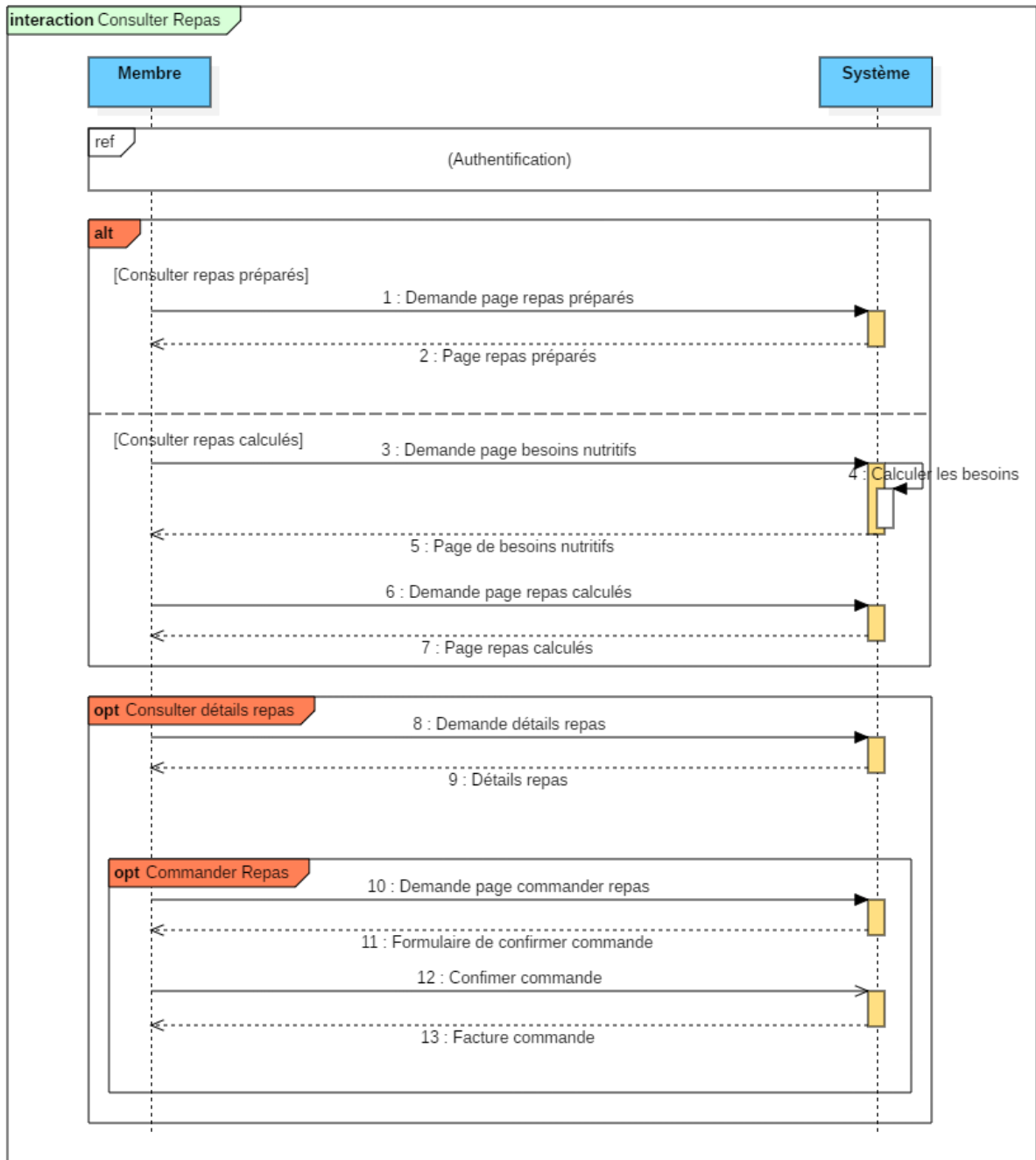


Figure 2.8 – Diagramme de séquence "Passer une commande"

Cas d'utilisation	Passer une commande
Acteurs	Membre
Pré condition	Membre authentifié
Description du scénario principal	<p>1-Consulter repas préparés</p> <ul style="list-style-type: none"> -Le membre demande la page de repas préparés. -Le système renvoie la page de repas préparés. <p>2-Consulter repas calculés</p> <ul style="list-style-type: none"> -Le membre demande la page besoins nutritifs. -Le système calcul les besoins et renvoie la page de besoins nutritifs. <p>Extension : Consulter détails repas</p> <ul style="list-style-type: none"> -Le membre demande les détails du repas. -Le système renvoie les détails du repas. <p>Extension : Commander repas</p> <ul style="list-style-type: none"> -Le membre demande la page commander repas. -Le système renvoie un formulaire de confirmer commande. -Le membre confirme sa commande. -Le système renvoie la facture de la commande.
Post condition	Commande passée
Exception	<ul style="list-style-type: none"> -S'il y a une erreur de saisie le système affiche un message d'erreur. -En cas d'erreur le système affiche un message d'erreur.

Tableau 2.2- Description de CU "Passer une commande"

3.4 Cas d'utilisation « Mettre à jour une commande »

- **Diagramme de cas d'utilisation**

Un traiteur peut gérer ses commandes en mettre à jour le statut de chaque commande.

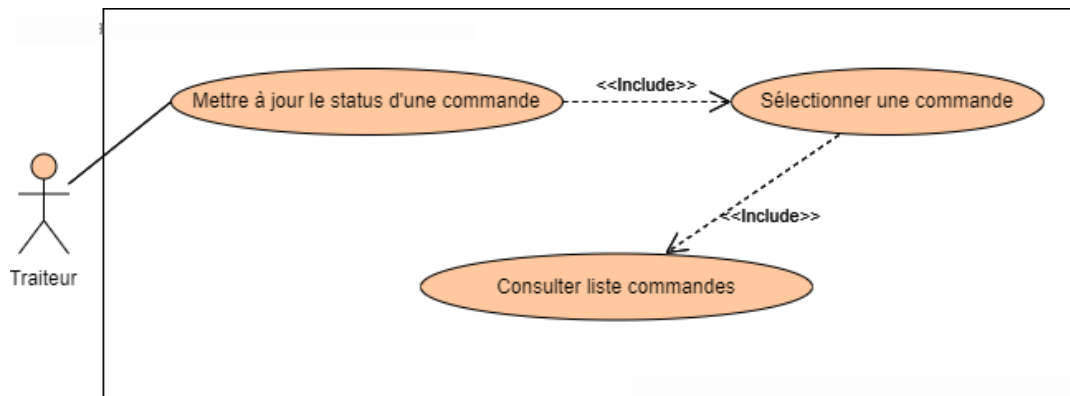


Figure 2.9 – Diagramme de CU "Mettre à jour une commande"

- **Diagramme de séquence**

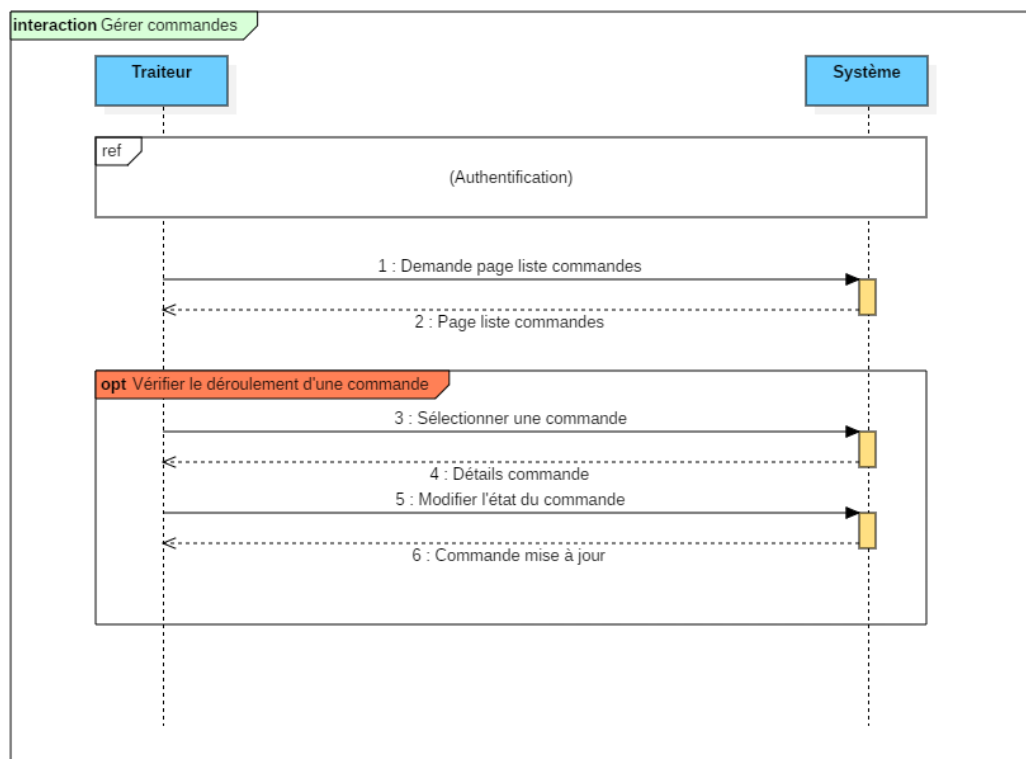


Figure 2.10 – Diagramme de séquence "Mettre à jour une commande"

Cas d'utilisation	Mettre à jour une commande
Acteurs	Traiteur
Pré condition	Traiteur authentifié
Description du scénario principal	<p>-Le traiteur demande la page de liste commandes.</p> <p>-Le système renvoie la page de liste commandes.</p> <p>Extension : Vérifier le déroulement d'une commande</p> <p>-Le traiteur sélectionne une commande.</p> <p>-Le système renvoie les détails de la commande.</p> <p>-Le traiteur modifier l'état de commande.</p> <p>-Le système renvoie la commande mise à jour.</p>
Post condition	Commande mis à jour
Exception	-En cas d'erreur le système affiche un message d'erreur

Tableau 2.3- Description de CU "Mettre à jour une commande"

3.5 Cas d'utilisation « Gérer compte traiteur »

- **Diagramme de cas d'utilisation**

L'admin peut gérer les comptes traiteur tout en ajoutant nouveau traiteur, supprimant un traiteur déjà existant, ou bien modifier les données d'un traiteur.

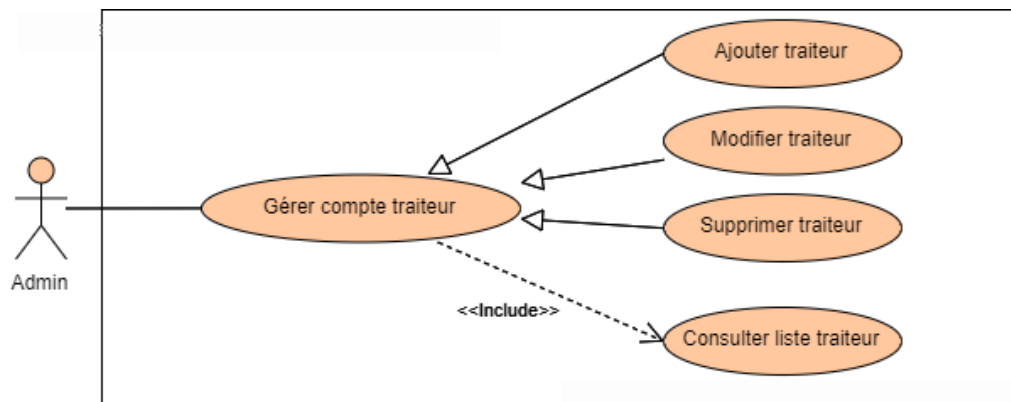
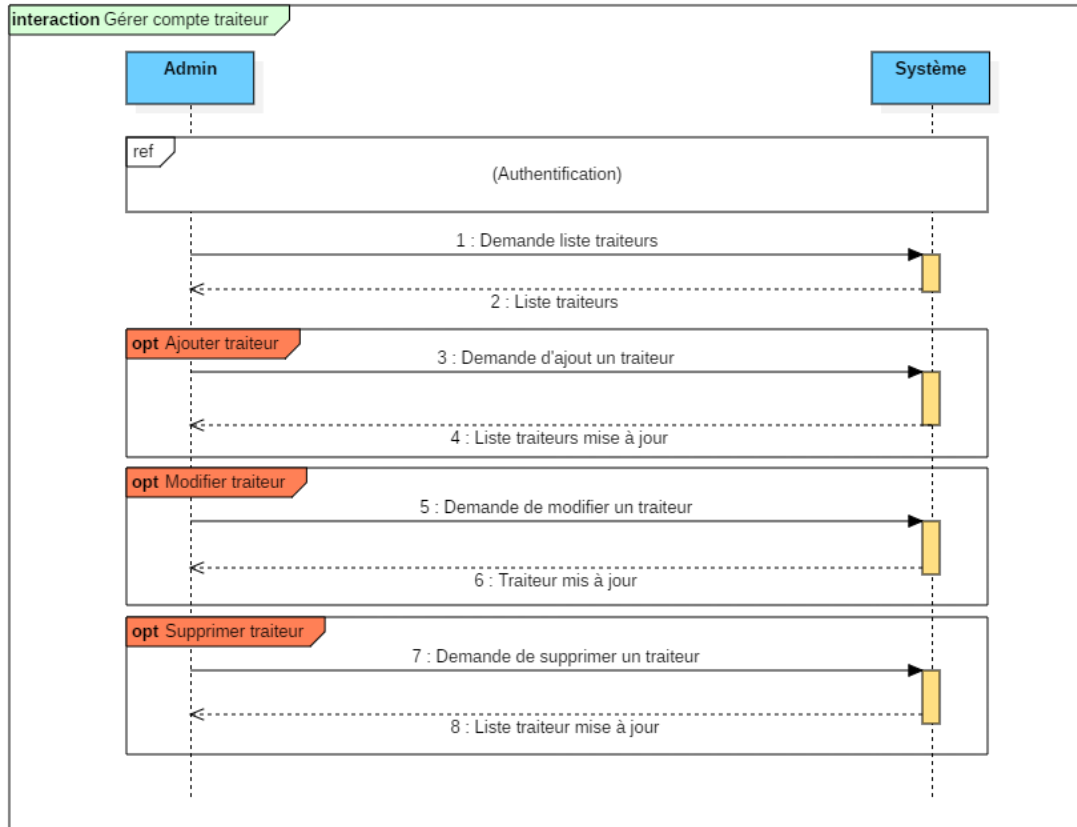


Figure 2.11 – Diagramme de CU "Gérer compte traiteur"

- **Diagramme de séquence**



- *Figure 2.12 – Diagramme de séquence "Gérer compte traiteur"*

Cas d'utilisation	Gérer compte traiteur
Acteurs	Admin
Pré condition	Admin authentifié
Description du scénario principal	<p>-L'admin demande la page de liste traiteurs. -Le système renvoie la page de liste traiteurs.</p> <p>Extension : Ajouter traiteur</p> <p>-L'admin demande d'ajouter un traiteur. -Le système renvoie la liste traiteurs mise à jour.</p> <p>Extension : Modifier traiteur</p> <p>-L'admin demande de modifier un traiteur.</p>

	<ul style="list-style-type: none"> -Le système renvoie le traiteur mis à jour. <p>Extension : Supprimer traiteur</p> <ul style="list-style-type: none"> -L'admin demande de supprimer un traiteur. -Le système renvoie la liste traiteurs mise à jour.
Post condition	Comptes traiteur gérés
Exception	<ul style="list-style-type: none"> -En cas d'erreur le système affiche un message d'erreur. -S'il y a une erreur de saisie le système affiche un message d'erreur.

Tableau 2.4- Description de CU "Gérer compte traiteur"

Conclusion

Dans ce chapitre, j'ai cité les besoins fonctionnels et non-fonctionnels de mon système. J'ai détaillé les besoins fonctionnels via des diagrammes de cas d'utilisation et de séquence acteur système. J'ai aussi présenté les besoins techniques qui conviennent avec mon projet.

Dans le chapitre suivant, je vais aborder la conception de mon application.

CONCEPTION

Introduction

Au cours de ce chapitre, je vais présenter l'architecture globale de mon application. Ensuite, je vais détailler la conception de la base de données. Je vais mieux présenter la conception logicielle via les deux vues statique et dynamique, ainsi que la conception graphique.

1. Architecture générale de l'application

Après avoir fait le choix de la méthodologie 2TUP, le processus de conception sera en adéquation sur l'architecture de l'application.

1.1 L'architecture 3-tiers

L'architecture 3-tiers est un modèle logique d'architecture applicatif qui vise à séparer très nettement trois couches logicielles au sein d'une même application. L'application est modélisée comme un empilement de trois couches dont le rôle est clairement défini :

Présentation des données : Correspondant à l'affichage, la restitution sur le poste de travail et le dialogue avec l'utilisateur.

Traitement métier des données : Correspondant à la mise en œuvre de l'ensemble des règles de gestion et de la logique applicatif.

Accès aux données persistantes : Correspondant aux données qui sont destinées à être conservées sur la durée, voire de manière définitive.

Avec ce modèle, chaque couche dialogue avec une autre au travers d'un contrat d'interface. Par convention, la couche de données est la couche la plus basse et la couche de présentation est la couche la plus haute. Chaque couche expose alors des services à la couche supérieure.

Cinématique

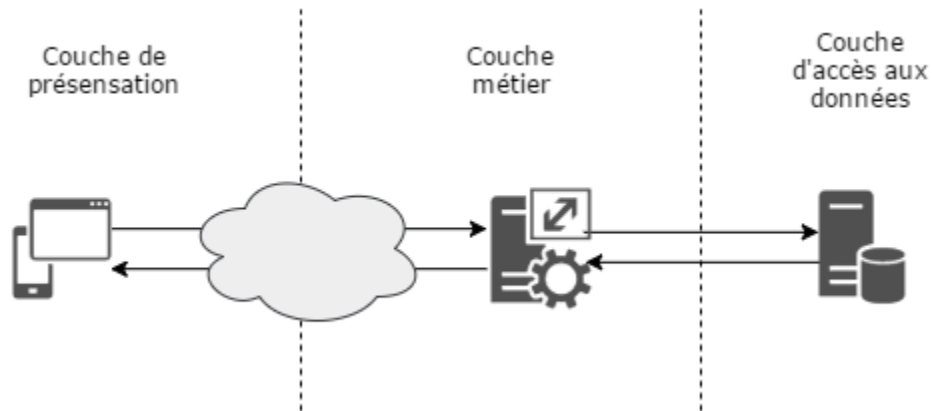


Figure 3.1 - Cinématique de l'architecture 3-tiers

Ce découpage permet de bien séparer les responsabilités. Chaque couche peut ainsi évoluer sans impacter les autres. En revanche, l'implémentation d'une nouvelle fonctionnalité peut toucher plusieurs couches.

1.2 Modèle MVC

Le Modèle-Vue-Contrôleur [9] est un *design pattern* qui impose la séparation entre les données, la présentation et les traitements, ce qui donne trois parties fondamentales dans l'application finale : le modèle, la vue et le contrôleur.



Figure 3.2 - Les parties fondamentales de modèle MVC

Modèle : Cette partie gère les données de votre site. Son rôle est d'aller récupérer les informations « brutes » dans la base de données, de les organiser et de les assembler pour qu'elles puissent ensuite être traitées par le contrôleur.

Vue : Cette partie se concentre sur l'affichage. Elle ne fait presque aucun calcul et se contente de récupérer des variables pour savoir ce qu'elle doit afficher. On y trouve pour l'application web essentiellement du code TWIG mais aussi quelques boucles et conditions PHP et Javascript très simples, pour afficher par exemple une liste de messages ou bien une animation. Et pour l'application mobile, on y trouve du code Dart pure.

Contrôleur : Cette partie gère la logique du code qui prend des décisions. C'est en quelque sorte l'intermédiaire entre le modèle et la vue : le contrôleur va demander au modèle les données, les analyser, prendre des décisions et renvoyer le texte à afficher à la vue. Le contrôleur contient exclusivement du PHP. C'est notamment lui qui détermine si le visiteur a le droit de voir la page ou non (gestion des droits d'accès).

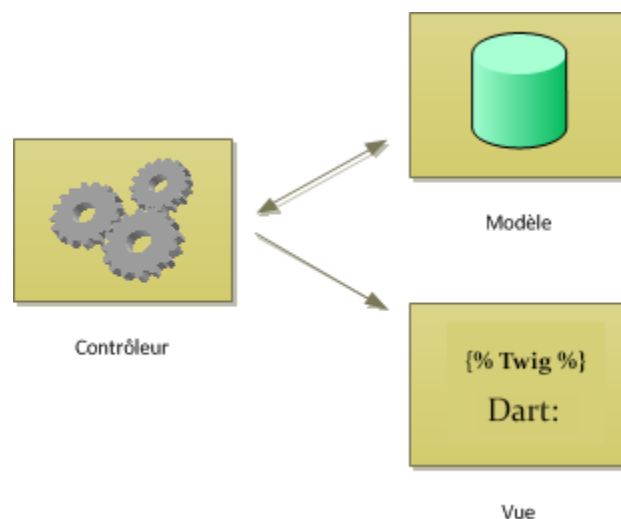


Figure 3.3 – Echange d'informations entre les éléments du modèle MVC

Il est bien évident que le contrôleur est le chef d'orchestre. C'est lui qui reçoit la requête du visiteur et qui contacte d'autres fichiers (le modèle et la vue) pour échanger des informations avec eux.

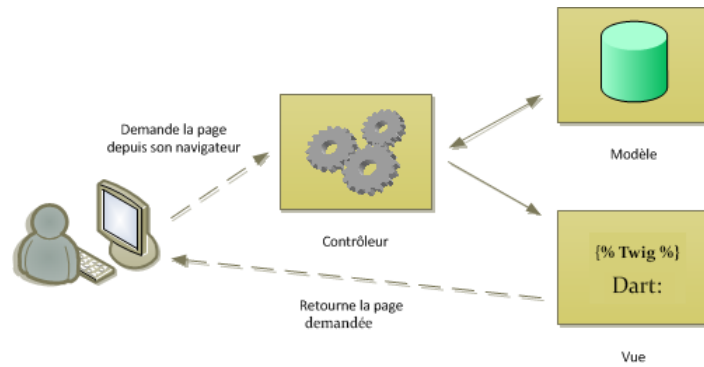


Figure 3.4 - Fonctionnement de modèle MVC

1.3 Architecture adoptée

J'ai adopté dans mon application une architecture 3-tiers. Côté client, mon application dispose d'un navigateur web ou bien une interface mobile affichant les différentes vues et assurant les interactions avec le serveur d'application. Côté serveur, on prévoit qu'il englobe la couche contrôleur et modèle et qui interagit avec la dernière couche base de données. La figure 3.5 présente l'architecture générale de mon application.

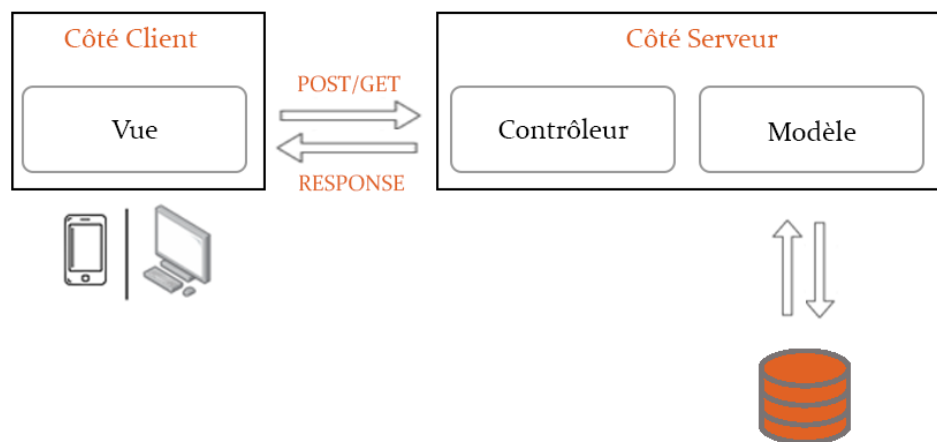


Figure 3.5 - Architecture générale de mon application

De manière plus détaillée, l'architecture de mon application est présentée par la figure 3.6.

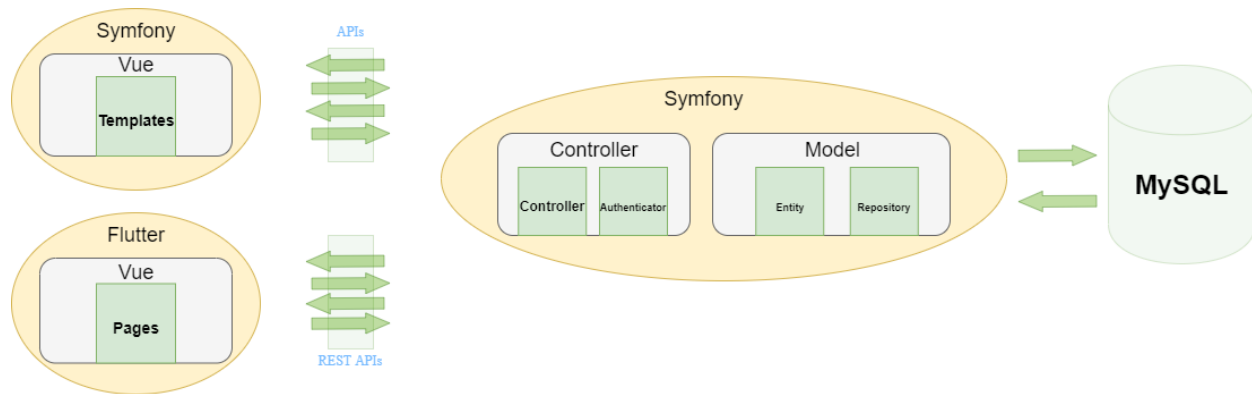


Figure 3.6 - Architecture détaillée de mon application

En adoptant l'architecture MVC, j'ai distingué les parties suivantes de mon application :

Partie Modèle :

Cette partie comporte deux couches : la couche « Entity » et la couche « Repository »

- Couche « Entity » : Ce sont les entités de mon système représentant les données qui vont être stockées ou récupérées de la base de données.
- Couche « Repository » : Le repository sert à construire les requêtes dans un fichier centralisé. Cela a comme gros avantage de ne pas réécrire une requête dans chaque action d'un contrôleur. Cette couche est aussi responsable du mapping objet relationnel, dans le cas de mon projet, le Framework Symfony utilise le composant ORM.

Partie Vue :

Cette partie comporte la couche « Templates » au niveau du Framework Symfony et la couche « Pages » au niveau du Framework Flutter.

- Couche « Templates » | Couche « Pages » : Correspond à une interface graphique qui affiche des données à l'utilisateur, interagit avec les événements générés par l'utilisateur et déclenche des actions au niveau de contrôleur selon ces événements.

Partie Contrôleur :

Cette partie comporte deux couches : la couche « Controller » et la couche « Authenticator ».

- Couche « Controller » : Cette couche est le centre de l'application, elle est responsable sur la logique du code qui prend des décisions, et ceci avec le reçu des requêtes et la production des réponses.
- Couche « Authenticator » : Correspond à un pare-feu de sécurité, cette couche est responsable à l'authentification des utilisateurs et les mémoriser.

2. Conception de la base de données

Dans cette section, je présente la conception détaillée de la base de données qui sera stockée sur un serveur base de données. Cette conception sera formulée par le modèle conceptuel de données et le modèle physique de données.

2.1 Modèle conceptuel de données (MCD)

Un modèle conceptuel de données cible à identifier les entités métiers de mon application ainsi que les relations entre elles. La figure 3.7 présente le modèle Entité / Association de ma base de données.

J'ai identifié les entités suivantes :

- Entité « Utilisateur » : modélise tous les membres de la plateforme. Chaque membre doit posséder principalement un email, un mot de passe, nom, sexe, âge, taille, poids, objectif et un niveau d'activité.
- Entité « Repas » : modélise tous les repas présentés dans la plateforme. Chaque repas doit avoir ses valeurs nutritives, un prix, un nom et une image.
- Entité « Commande » : représente les commandes faites par les utilisateurs. Chaque commande doit posséder la date, le temps et l'adresse de livraison, et même le statut de commande.
- Entité « Traiteur » : cette entité représente le deuxième acteur fondamental de mon application. Chaque traiteur possède un login et un mot de passe en plus de son numéro de téléphone, son nom et son email.
- Entité « Besoins » : modélise les besoins nutritifs de chaque utilisateur. L'entité Besoins est caractérisée par les valeurs de calories, de protéine, de graisses et de carbohydrates.
- Entité « Message » : représente un message envoyé par soit un utilisateur, soit un membre ou un traiteur. Cette entité est caractérisée par un type, un contenu et le date d'envoi.

- Entité « Article » : Cette entité est créée par l'administrateur à chaque fois il publie d'actualité. Elle est caractérisée par un titre, un en-tête, un contenu et une image.
- Entité « Recette » : Cette entité représente les recettes saines publiés dans la plateforme. Elle est caractérisée par un titre, un contenu et une image.

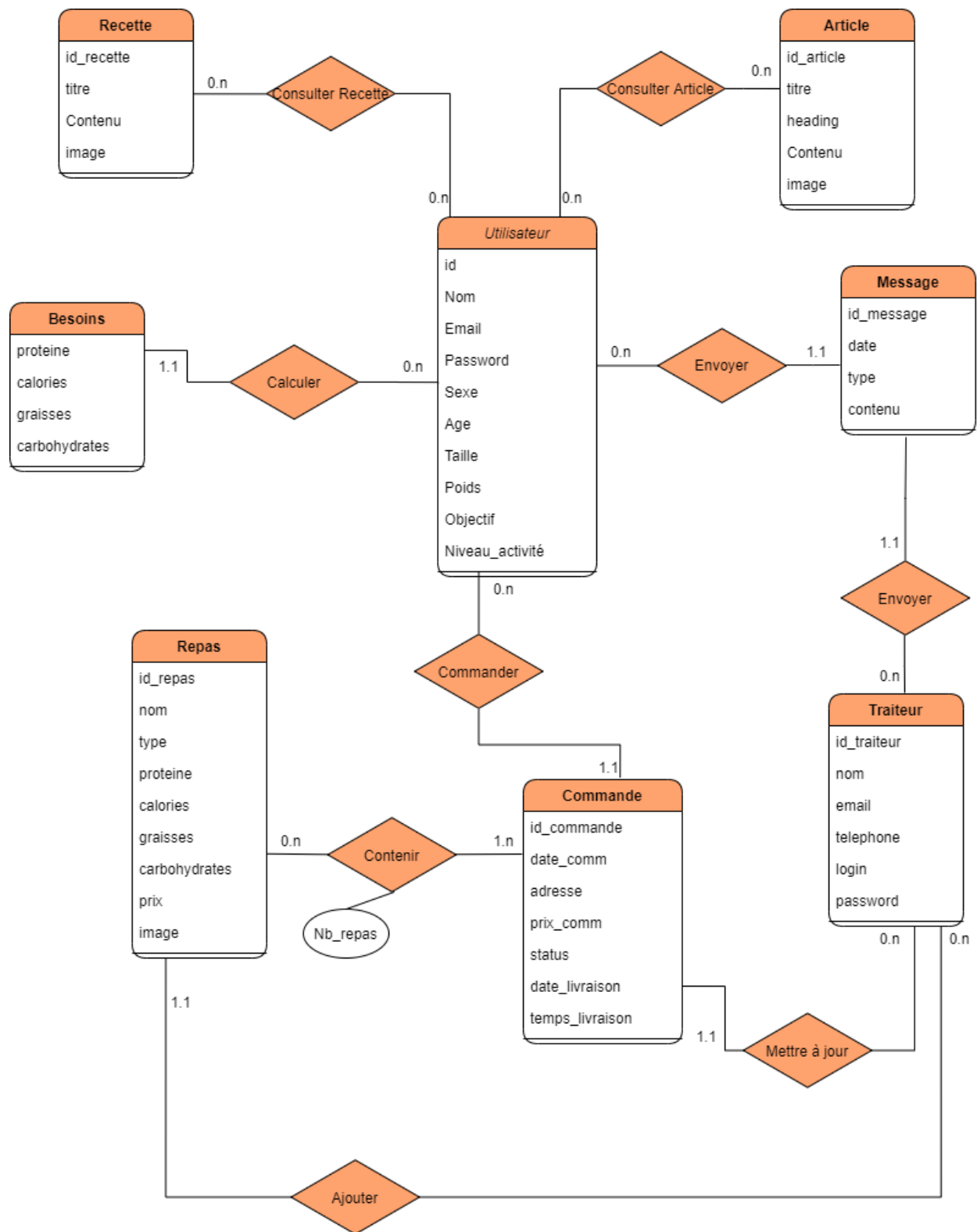


Figure 3.7 – Modèle Conceptuel de données (MCD)

2.2 Modèle logique de données (MLD)

Dans le but de stocker les données de mon application, j'étais amené à utiliser la base de données MySQL. Et afin de réaliser ce stockage, j'ai besoin de convertir le modèle conceptuel de données en un schéma relationnel. En appliquant les règles de transformation du diagramme Entité/Association en schéma relationnel, j'ai obtenu 11 tables dont 8 sont des entités présentés dans le diagramme précédent et 3 associations de type n..n

- Utilisateur (id, nom, email, password, sexe, âge, taille, poids, objectif, niveau_activité)
- Repas (id_repas, nom, type, protéine, calories, graisses, carbohydrates, prix, image, #id_traiteur)
- Commande (id_commande, date_comm, adresse, prix_comm, status, date_livraison, temps_livraison, #id_user, #id_repas, #id_traiteur)
- Traiteur (id_traiteur, nom, email, telephone, login, password)
- Besoins (#id_user, protéine, calories, graisses, carbohydrates)
- Message (id_message, date, type, contenu, #email, #nom)
- Article (id_article, titre, heading, contenu, image)
- Recette (id_recette, titre, contenu, image)
- Consulter Article (#id_user, #id_article)
- Consulter Recette (#id_user, #id_recette)
- Contenir (#id_repas, #id_commande)

3. Conception logicielle

Cette partie présente la conception logicielle de mon application en se basant sur le langage UML. Tout en respectant les contraintes du modèle MVC, je vais construire une vue statique sous forme de diagramme de classe et une vue dynamique sous forme de diagramme de séquences de conception et un diagramme d'état.

3.1 Vue statique : diagramme de classe

Pour cette section, je vais illustrer l'aspect statique de mon application en présentant le diagramme de classe. Puisque j'ai adopté l'architecture MVC dans la section 1.3, l'explication du diagramme de classe sera faite en trois parties.

Première partie : Modèle

Cette partie est composée de deux couches : la couche Entity et la couche Repository. Couche 'Entity' : c'est la représentation objet de la base de données. Couche 'Repository' : c'est la couche responsable du mapping objet relationnel, Elle est composée des fichiers PHP et à chaque entité créée, un fichier Repository associé est généré automatiquement. Cette couche est constituée de ces objets « CommandeRepository », « UserRepository », « RepasRepository », « RecetteRepository », « TraiteurRepository », « ArticleRepository », « MessageRepository », « BesoinsRepository ».

Deuxième partie : Contrôleur

Cette couche est le chef d'orchestre de l'application, elle va gérer le lien entre la partie vue et la partie modèle. Les classes de cette partie implémentent les traitements qui vont être invoqués par le navigateur, elle gère les requêtes et les réponses entre l'utilisateur et le serveur comme les demandes des pages, des routes, des formulaires et elle assure l'envoi des données à l'application mobile. Cette couche est constituée de classes suivantes : « DefaultController » est le contrôleur principal de la plateforme, « RegistrationController » et « SecurityController » sont les contrôleurs qui gèrent l'authentification, la registration et la déconnexion d'un utilisateur.

Troisième partie : Vue

L'implémentation de la partie vue sera faite sur les deux plateformes web et mobile. Pour la plateforme mobile, le Framework Flutter offre les outils de développer une application mobile sur Android et iOS avec le même code Dart, et Dart offre les outils de la réalisation d'une interface graphique dynamique sans l'utilisation d'aucun autre langage. Et pour la plateforme web, la partie vue est composée sur deux types :

- Fichier CSS pour la mise en forme des pages.
- Fichiers TWIG pour la définition du contenu et de la structure de la page.

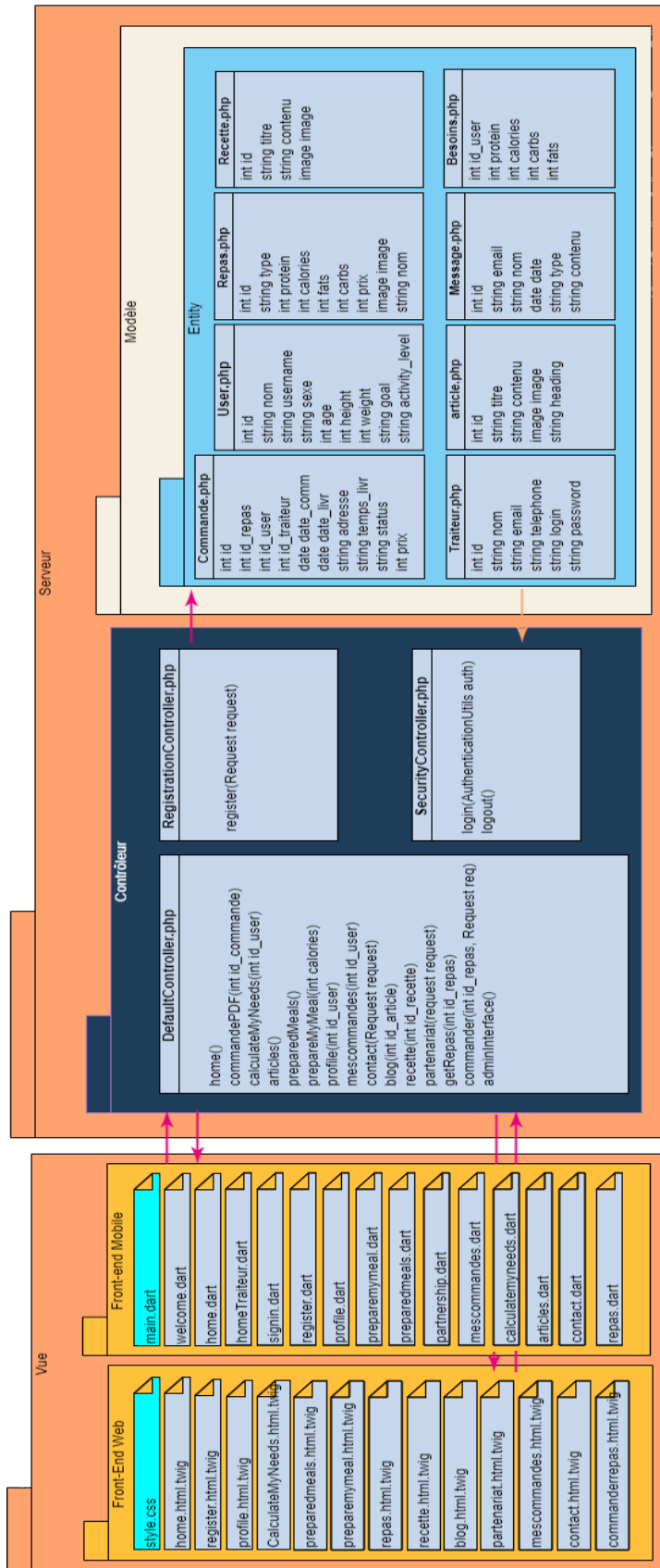


Figure 3.8 – Diagramme de classe

3.2 Vue Dynamique : diagrammes de séquences de conception

Les diagrammes de vue dynamiques permettent de comprendre et de décrire le comportement des objets et leurs interactions. Ces modèles offrent une vision microscopique du fonctionnement du système. Ils servent à mettre en évidence les relations temporelles entre les objets. Pour cela, je vais représenter les aspects dynamiques entre les classes de mon application à l'aide des diagrammes des séquences des plus importantes cas d'utilisations.

3.2.1 Diagramme de séquence générale

La figure 3.9 et 3.10 présentent les diagrammes de séquences qui représentent le modèle d'interaction générale entre les différentes couches de l'architecture de système coté web et coté mobile respectivement. Les diagrammes de séquences qui suivent, adoptent etinstancient ces modèles. Ils montrent l'interaction entre les couches templates et pages avec les contrôleurs et les modèles d'une façon générale.

On note par : `_template.twig` dénote un objet de la couche templates, `_page.dart` dénote un objet de la couche pages, `_controller.php` dénote un objet de la couche Controler et `_model.php` dénote un objet de la couche Model.

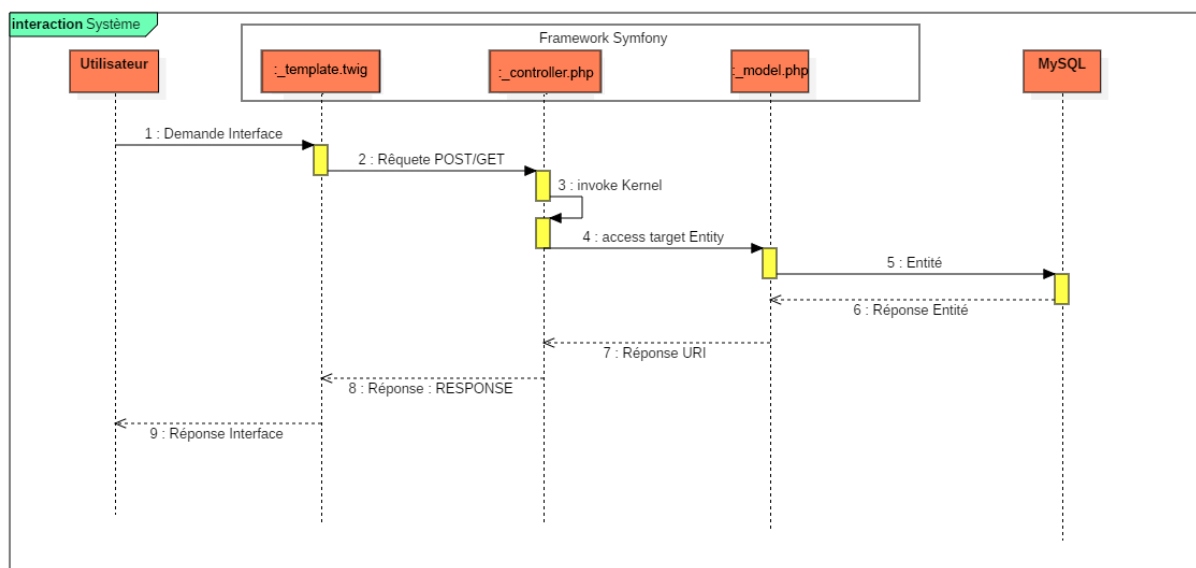


Figure 3.9 – Diagramme de séquence générale coté web

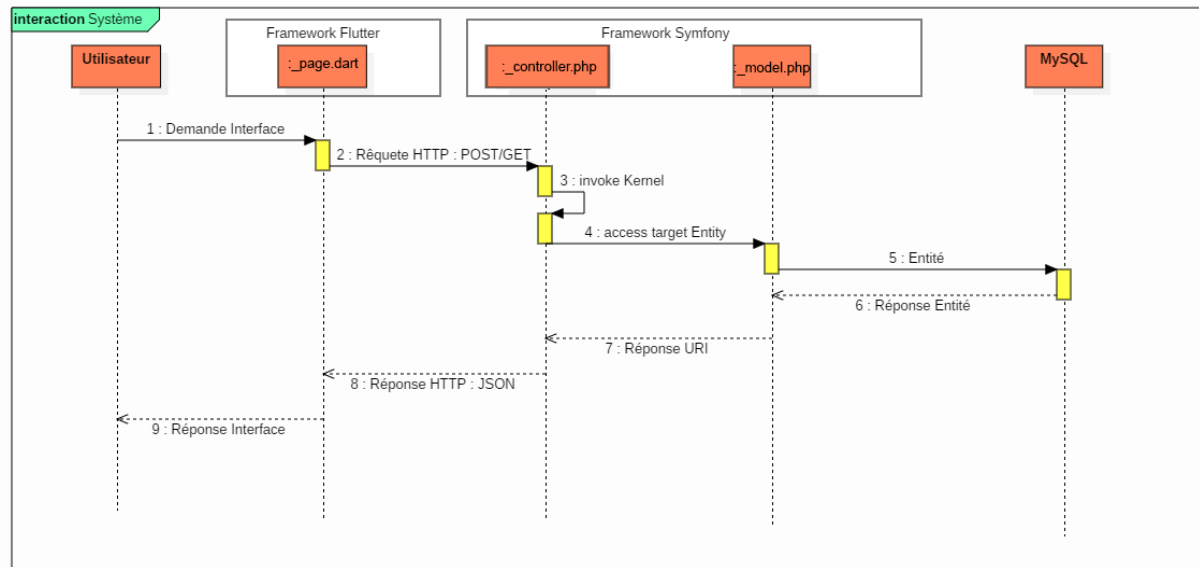


Figure 3.10 – Diagramme de séquence générale coté mobile

3.2.2 Diagramme de séquence de C.U ‘Authentification’

Afin d’avoir une expérience complète dans la plateforme, l’utilisateur doit s’authentifier, c’est vrai qu’un utilisateur a un accès à l’application sans authentification mais cet accès est limité, donc c’est mieux d’identifier. Cette identification est importante car elle permet le système d’accéder à les données de l’utilisateur pour générer ses besoins nutritifs. L’utilisateur saisit dans un formulaire son email et son mot de passe compatibles avec les données existantes dans la base de données.

- Si le compte existe et les données saisis sont valides, le système rediriger l’utilisateur à la page d’accueil.
- Sinon, le système affiche un message d’erreur pour que l’utilisateur ressaye.

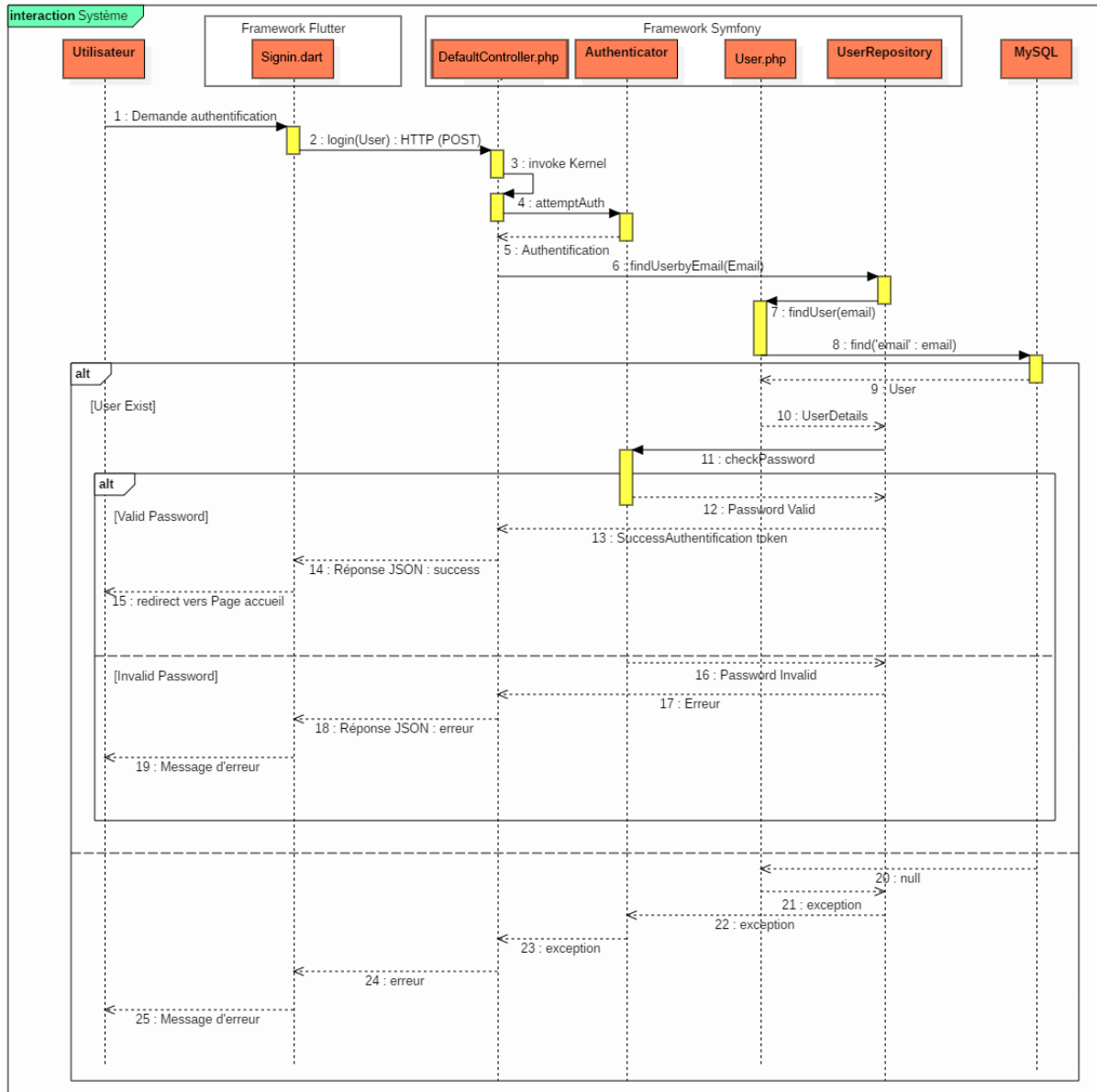


Figure 3.11 – Diagramme de séquence de C.U 'Authentication' coté mobile

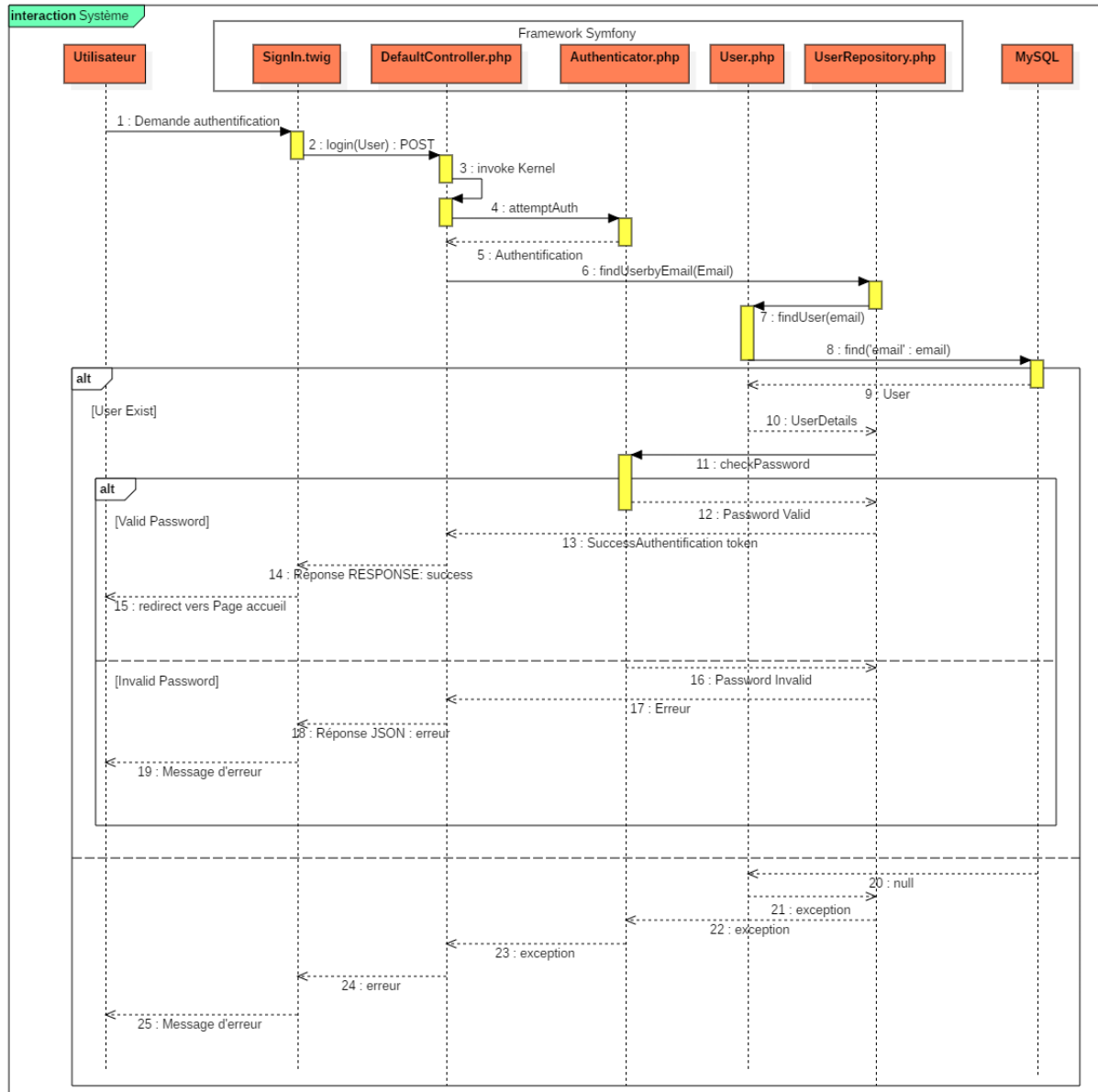


Figure 3.12 – Diagramme de séquence de C.U 'Authentification' coté web

3.2.3 Diagramme de séquence de C.U ‘Consulter repas’

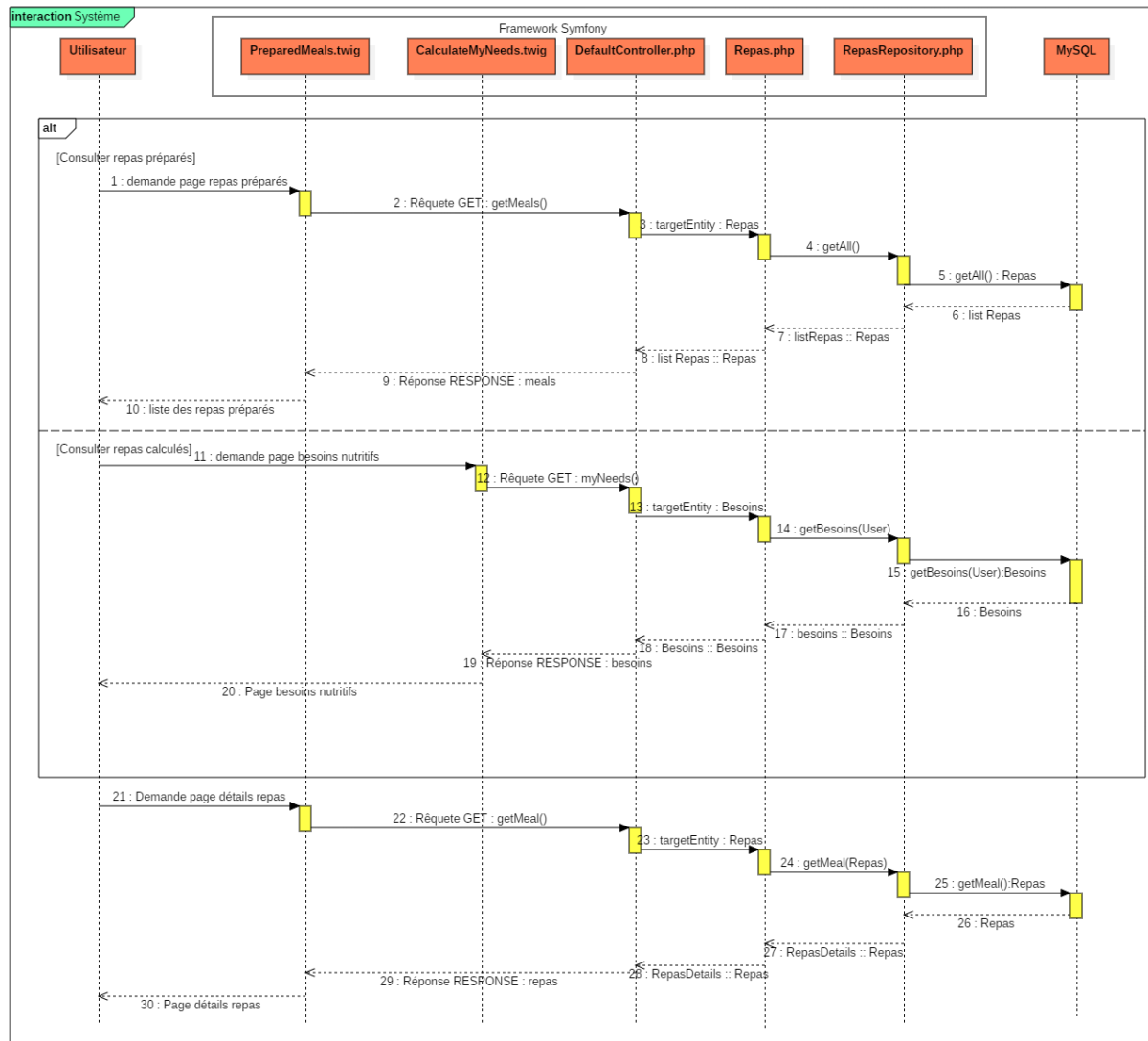


Figure 3.13 – Diagramme de séquence de C.U ‘Consulter Repas’ coté web

Coté mobile, c’est presque le même déroulement mais la partie ‘Vue’ est transmis vers le Framework Flutter et les requêtes GET et POST vont être changés en des requêtes HTTP.

3.3 Vue dynamique : diagramme d'état d'une commande

L'entité Commande est une entité principale dans mon application. Ainsi j'ai jugé utile et intéressant de schématiser le déroulement de son cycle de vie avec un diagramme d'états.

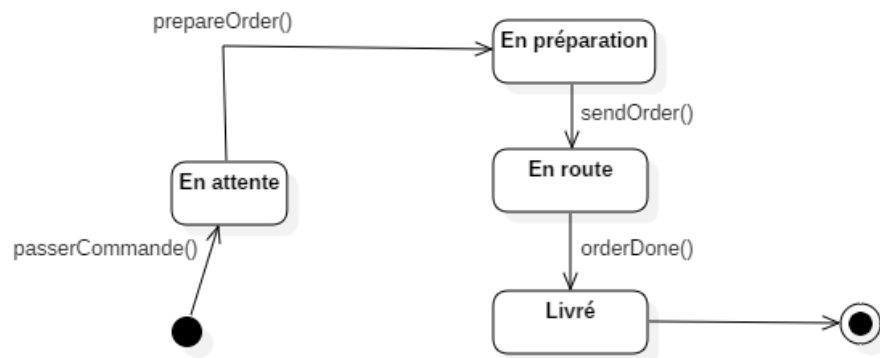


Figure 3.14 – Diagramme d'état d'une commande

4. Conception graphique

4.1 Diagramme de navigation

Cette section possède le diagramme de navigation entre les interfaces utilisateurs. Ce diagramme présenté par la figure 3.15 liste les différentes interfaces graphiques de l'application et les liens de navigations entre elles. Les différents scénarios et liens de navigation sont les mêmes dans les deux plateformes (web et mobile).

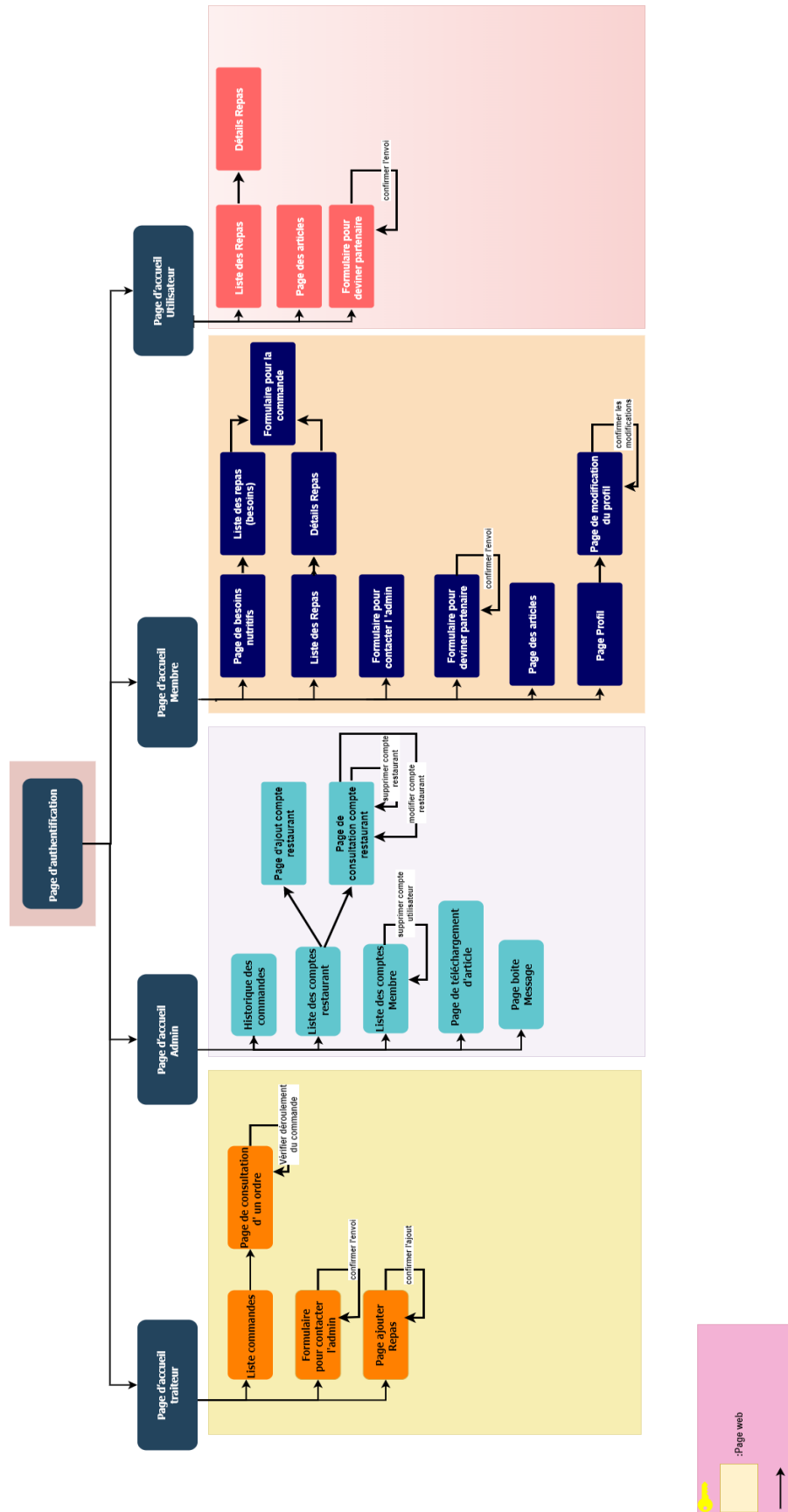


Figure 3.15 – Diagramme de navigation

4.2 Maquettage

L'étape finale juste avant le développement technique de mon application, est la création d'une application sur mesure : Maquettage (Wireframe). Les interfaces graphiques donnent une vision claire de l'application avant sa réalisation. Dans ce qui suit, je présente quelques exemples d'interfaces que j'ai choisi parmi les interfaces établies.

Interface de C.U « Calculer Mes Besoins »

Cette interface est dédiée aux membres qui ont déjà authentifiés pour consulter leurs besoins nutritifs et avoir les repas qui conviennent avec leurs conditions physiques et leurs niveaux d'activité.

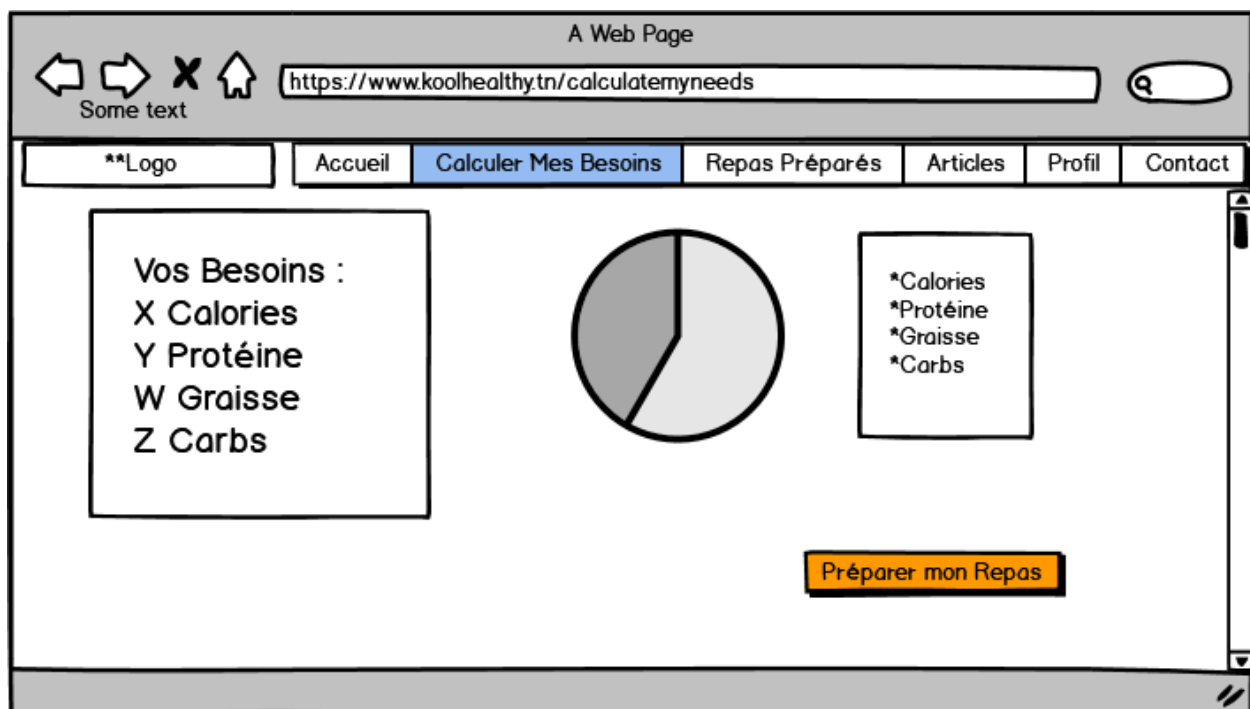


Figure 3.16 – Maquettage de l'interface Web « Calculer mes besoins »

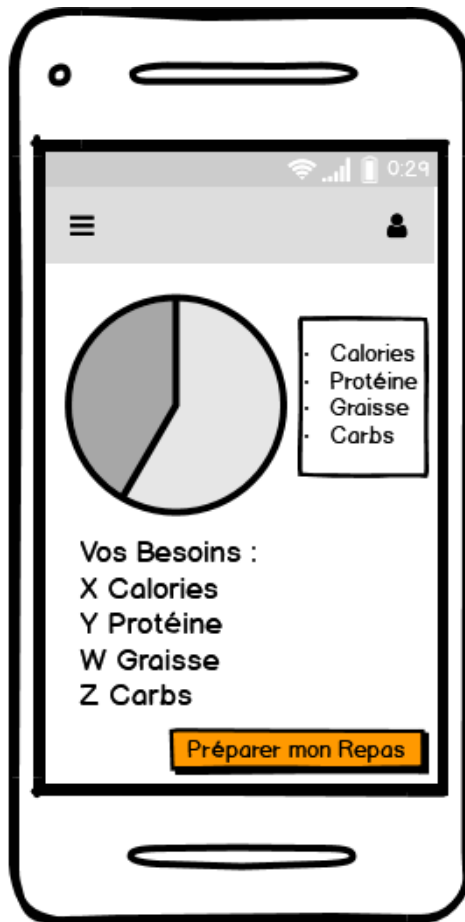


Figure 3.17 – Maquettage de l'interface Mobile « Calculer mes besoins »

Interface de C.U « Consulter commandes »

Un membre peut consulter les commandes qu'il a déjà passé et il peut également poursuivre le statut de chaque commande.

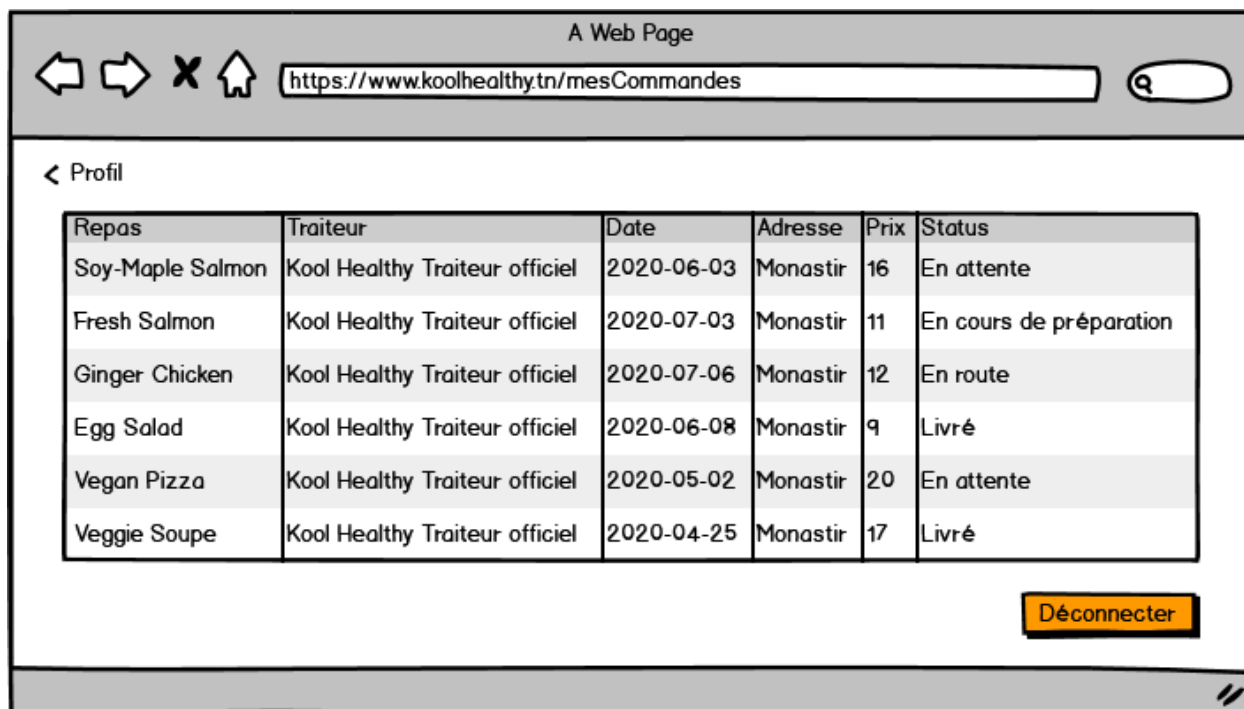


Figure 3.18 – Maquettage de l'interface web « Consulter commandes »

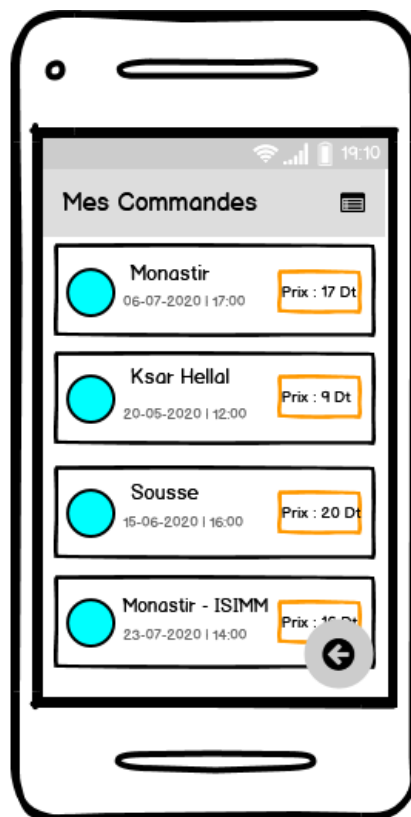


Figure 3.19 – Maquettage de l'interface mobile « Consulter commandes »

Interface de C.U « Consulter mes commandes »

Un traiteur peut consulter ses commandes afin de vérifier le déroulement de chaque commande.

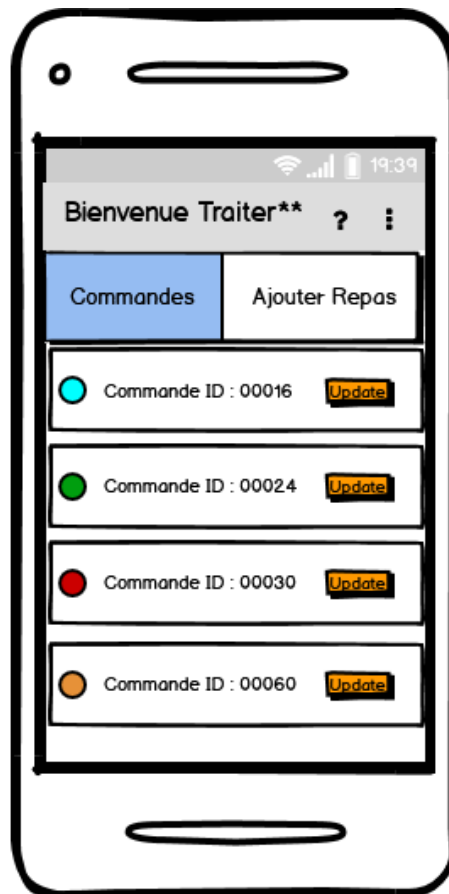


Figure 3.20 – Maquettage de l'interface « Consulter mes commandes »

Interface Admin

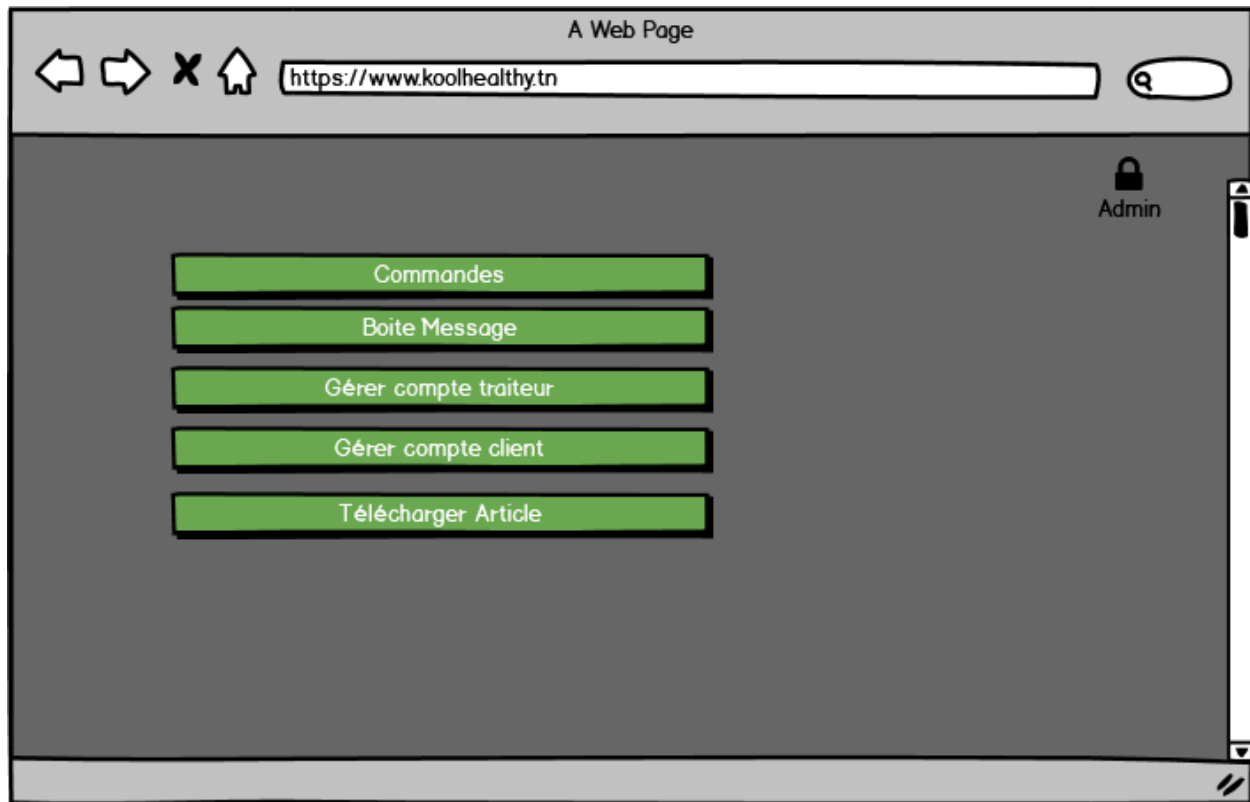


Figure 3.21 – Maquettage de l'interface « Admin »

Conclusion

Dans ce chapitre, j'ai commencé par la présentation de l'architecture globale de mon application. Puis j'ai présenté la conception détaillée de ma base de données. Ensuite, j'ai décrit la conception logicielle à travers une vue statique via un diagramme de classe et une vue dynamique via les diagrammes de séquences. Et finalement, j'ai présenté la conception logicielle à travers un diagramme de navigation et des maquettes.

RÉALISATION

Introduction

Après avoir terminé la phase de conception, j'aborde le chapitre de réalisation, qui est le dernier volet de ce rapport. Ce chapitre a comme objectif d'illustrer l'environnement du développement, détailler les différentes technologies utilisées et à la description de quelques fonctionnalités de la plateforme à l'aide des captures écrans représentatives.

1. Environnement de développement

Cette section décrit l'environnement matériel et logiciel du mon projet qui a permis l'implémentation de mon application.

1.1 Environnement matériel

J'ai utilisé comme environnement matériel un ordinateur ASUS qui possède les caractéristiques suivantes :

Ordinateur	ASUS
Processeur	i7 9ème génération
Mémoire vive (RAM)	8 GO
Disque dur	1 TO
Système d'exploitation	Windows 10 Education

Table 4.1 – Environnement matériel

1.2 Environnement logiciel

Dans cette section, je vais détailler les logiciels utilisés lors de la phase de conception et la phase de développement. Ces logiciels sont illustrés par le tableau 4.2.

Outil	Description	Utilisation
Start UML	Logiciel de modélisation UML	Création des diagrammes UML comme les diagrammes de cas d'utilisation et de séquence
Visual Paradigm	Outil Web pour la modélisation UML et la modélisation Business Process Modeling	Création des diagrammes comme le MCD, MPD et la création des différentes figures qui sont présentes dans le rapport
Balsamiq Mockup	Outil de maquettage et de wireframing qui permet de créer des designs graphiques d'interface utilisateur (UI)	Réaliser les maquettes de mon application
Microsoft Word	Logiciel de traitement de texte	Créer, traiter et mettre en page le rapport de mon projet
Adobe Photoshop	Outil de traitement d'images	Créer et modifier des designs de certaines interfaces graphiques
XAMPP	Ensemble de logiciels permettant de mettre en place un serveur Web local, un serveur FTP et un serveur de messagerie électronique.	Connecter au SGBDR MySQL
PhpMyAdmin	Outil d'administration et de gestion de bases de données MYSQL	Créer et administrer la base de données du projet

Visual Studio Code	Environnement intégré de développement (IDE) pour le langage PHP, CSS, HTML (et d'autres langages)	Environnement intégré de développement (IDE) pour le Framework Symfony
Android Studio	Environnement intégré de développement (IDE) pour le langage Dart (et d'autres langages)	Environnement intégré de développement (IDE) pour le Framework Flutter
TWIG	Moteur de templates pour le langage PHP. Il peut être inclus dans un bloc de code HTML	Création de la partie front-end de mon site web
CSS	Langage de style qui définit la présentation des documents HTML	Gestion de l'apparence de la partie front-end de mon site web
PHP	Langage de programmation généralement utilisé pour la partie back-end des applications	Gestion du côté serveur de la plateforme (les contrôleurs et les modèles)
Dart	Langage de programmation web, mobile et desktop	Création de l'interface mobile.

Table 4.2 – Environnement logiciel

2. Framework Symfony

Symfony est un ensemble de composants PHP ainsi qu'un Framework MVC libre écrit en PHP. Il fournit des fonctionnalités modulables et adaptables qui permettent de faciliter et d'accélérer le développement d'un site web.

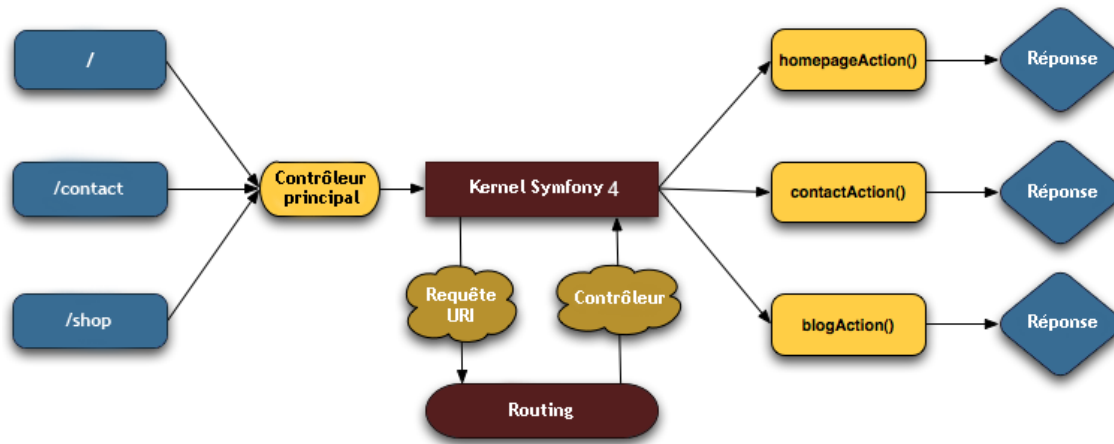


Figure 4.1 – Fonctionnement général du Framework Symfony

La figure 4.1 présente le fonctionnement du Framework Symfony, et elle montre comment le contrôleur principal gère les événements par la réception des routes demandés, il l'envoie à le Kernel de Symfony qui ce dernier envoie une requête URI au composant Routing et reçoit le contrôleur correspondant pour la route demandée. Ensuite le Kernel exécute la fonction du contrôleur reçue pour générer une réponse à l'utilisateur.

Dans le cas de mon projet, j'ai utilisé le Framework Symfony pour l'implémentation des :

- Contrôleurs de l'application
- Modèles de l'application
- Front-End de l'application web

2.1 Création d'un projet Symfony

Afin de créer un projet Symfony, le logiciel Composer doit être fourni afin de gérer les dépendances de PHP et requérir les bibliothèques nécessaires. La création du projet se fait par une commande sur le terminal qui permet de créer le squelette d'arborescence d'un projet Symfony.

```
> composer create-project symfony/skeleton projetPFE
```

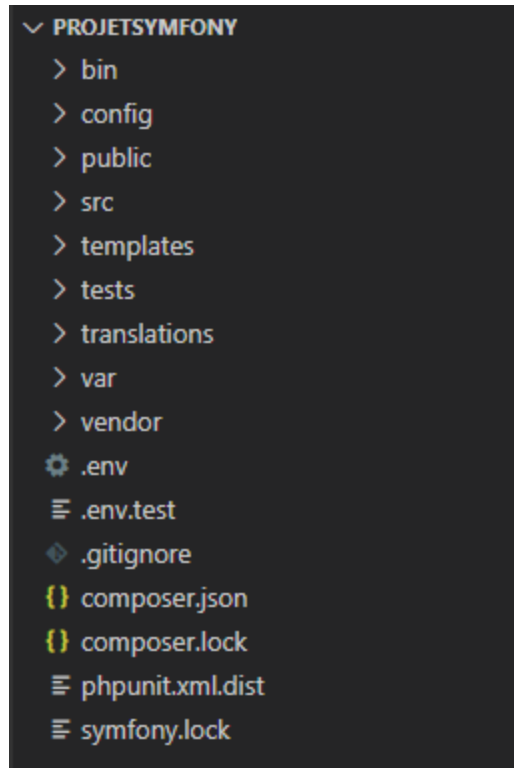


Figure 4.2 – Squelette d'un projet Symfony

Le squelette d'un projet Symfony est organisé d'une manière qui facilite le développement et améliore la performance du système.

- Dossier **bin** : Stocke les scripts exécutables.
- Dossier **src** : Stocke la partie Back-End de l'application.
- Dossier **templates** : Stocke la partie Front-End de l'application.
- Dossier **public** : Stocke les fichiers 'publiques' comme les fichiers CSS, Js, images, fonts...
- Fichier **.env** : responsable de la connexion à la base de données.

2.2 Manipulation des données avec Symfony

Symfony offre un fichier de configuration **.env** qui contient les données sensibles de la base de données comme le mot de passe, les ports et le nom de la base de données... La figure 4.3 présente une capture d'écran de la configuration de la base de données dans le fichier **.env**

```
# IMPORTANT: You MUST configure your server version, either here or in config/packages/doctrine.yaml
DATABASE_URL=mysql://root:@127.0.0.1:3306/projetpfe?serverVersion=5.7
###< doctrine/doctrine-bundle ###
```

Figure 4.3 – Capture d'écran de la configuration de la BD

ORM

L'ORM est un outil de mappage des modèles aux éléments de la base de données relationnelle principale pour récupérer et conserver les modèles de manière sûre et efficace. Symfony fournit un bundle séparé, DoctrineBundle, qui intègre Symfony à l'outil ORM de base de données PHP tiers, **Doctrine**.

Création de la base de données :

```
> php bin/console doctrine:database:create
```

Création d'une entité :

```
> php bin/console make:entity
```

Migrer les données à la base de données MySQL :

```
> php bin/console doctrine:migrations:migrate
```

```
> php bin/console doctrine:migrations:diff
```

Exemple d'entité 'Commande : avec l'association 'OneToOne'

La figure 4.4 représente une capture d'écran d'un aperçu de l'entité Commande avec l'association OneToOne qui définit l'attribut `id_repas` comme un clé étranger de l'entité Repas.

```
/**
 * @ORM\Entity(repositoryClass=CommandeRepository::class)
 */
class Commande
{
    /**
     * @ORM\Id()
     * @ORM\GeneratedValue()
     * @ORM\Column(type="integer")
     */
    private $id;

    /**
     * @ORM\OneToOne(targetEntity="Repas")
     * @ORM\Column(type="integer")
     */
    private $id_repas;
```

Figure 4.4 – Capture d'écran d'entité Commande

Migrations

Les migrations sont le moyen utilisé par Symfony pour propager les modifications que j’ai apporté aux modèles (ajouter un attribut, supprimer une entité, etc.) dans mon schéma de base de données.

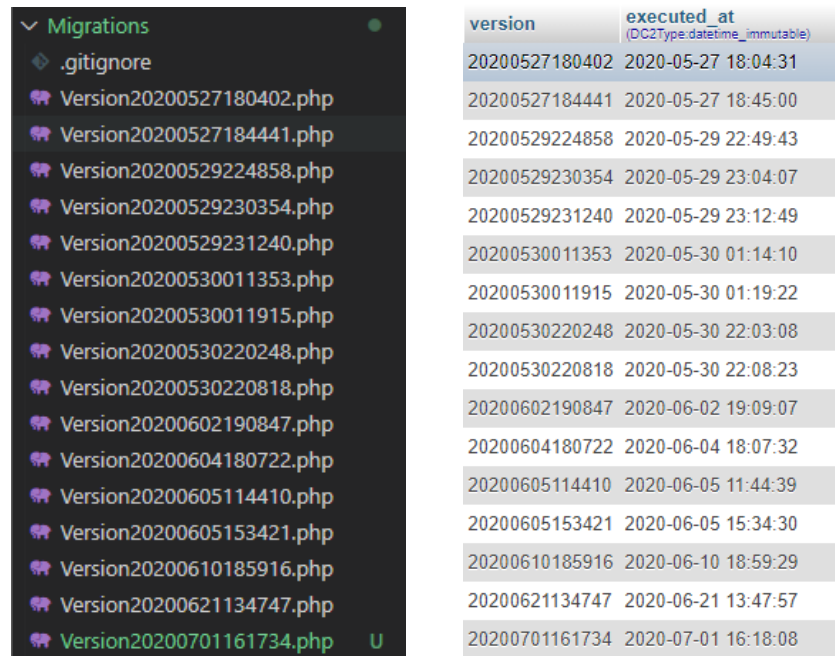
The image shows two side-by-side screenshots from a Symfony application. The left screenshot is a file explorer view of the 'Migrations' directory, showing a list of migration files named 'Version20200527180402.php' through 'Version20200701161734.php'. The right screenshot is a table showing the execution status of these migrations, with columns for 'version' and 'executed_at'.

Figure 4.5 – Captures d’écran d’un extrait de migrations

2.3 Mapping objet relationnel avec Symfony

Les fichiers responsables de mapping objet relationnel avec Symfony sont les ‘*Repositories*’. Pour chaque entité créée avec Symfony, un fichier Repository est généré automatiquement qui va être responsable sur le mapping objet.

La figure 4.6 représente la classe CommandeRepository assurant le mapping objet relationnel

```

use App\Entity\Commande;
use Doctrine\Bundle\DoctrineBundle\Repository\ServiceEntityRepository;
use Doctrine\Persistence\ManagerRegistry;

/**
 * @method Commande|null find($id, $lockMode = null, $lockVersion = null)
 * @method Commande|null findOneBy(array $criteria, array $orderBy = null)
 * @method Commande[]    findAll()
 * @method Commande[]    findBy(array $criteria, array $orderBy = null, $limit = null, $offset = null)
 */
class CommandeRepository extends ServiceEntityRepository
{
    public function __construct(ManagerRegistry $registry)
    {
        parent::__construct($registry, Commande::class);
    }

    // /**
    //  * @return Commande[] Returns an array of Commande objects
    //  */

```

Figure 4.6 – Capture d’écran du classe CommandeRepository

Les annotations de la classe CommandeRepository permet de récupérer les données de l’entité Commande via des méthodes générées dont l’id ou bien un critère sont passés en paramètre. On trouve aussi la méthode findAll() qui récupère toutes les instances de l’entité Commande.

```

final class Version20200530220818 extends AbstractMigration
{
    public function getDescription() : string
    {
        return '';
    }

    public function up(Schema $schema) : void
    {
        // this up() migration is auto-generated, please modify it to your needs
        $this->abortIf($this->connection->getDatabasePlatform()->getName() !== 'mysql', 'Migration can only be executed safely on \'mysql\'');

        $this->addSql('CREATE TABLE commande (id INT AUTO_INCREMENT NOT NULL, id_repas INT NOT NULL, date_comm DATE NOT NULL, adresse VARCHAR(255) NOT NULL);');
    }

    public function down(Schema $schema) : void
    {
        // this down() migration is auto-generated, please modify it to your needs
        $this->abortIf($this->connection->getDatabasePlatform()->getName() !== 'mysql', 'Migration can only be executed safely on \'mysql\'');

        $this->addSql('DROP TABLE commande');
    }
}

```

Figure 4.7 – Capture d’écran du classe Migration de l’entité Commande

2.4 Calcul des besoins nutritifs

Le calcul des besoins nutritifs est la fonctionnalité de base de mon application. Tout un algorithme se fait afin d’avoir les valeurs exactes des protéines, des calories, des graisses et des carbohydrates d’un membre, ce calcul se fait à partir des données saisies par l’utilisateur dès son inscription.

Méthode de calcul :

Les données nécessaires pour le calcul des besoins nutritifs qui s'expriment par le nombre de calories et la quantité de protéine, graisses et carbohydrates sont : le poids, la taille, le sexe, l'âge, le niveau d'activité ['Pas actif', 'Peu actif', 'Actif', 'Super actif'] et l'objectif ['Prendre du poids', 'Maintenir le poids', 'Perdre du poids']. La figure 4.8 résume l'algorithme de calcul des besoins nutritifs.

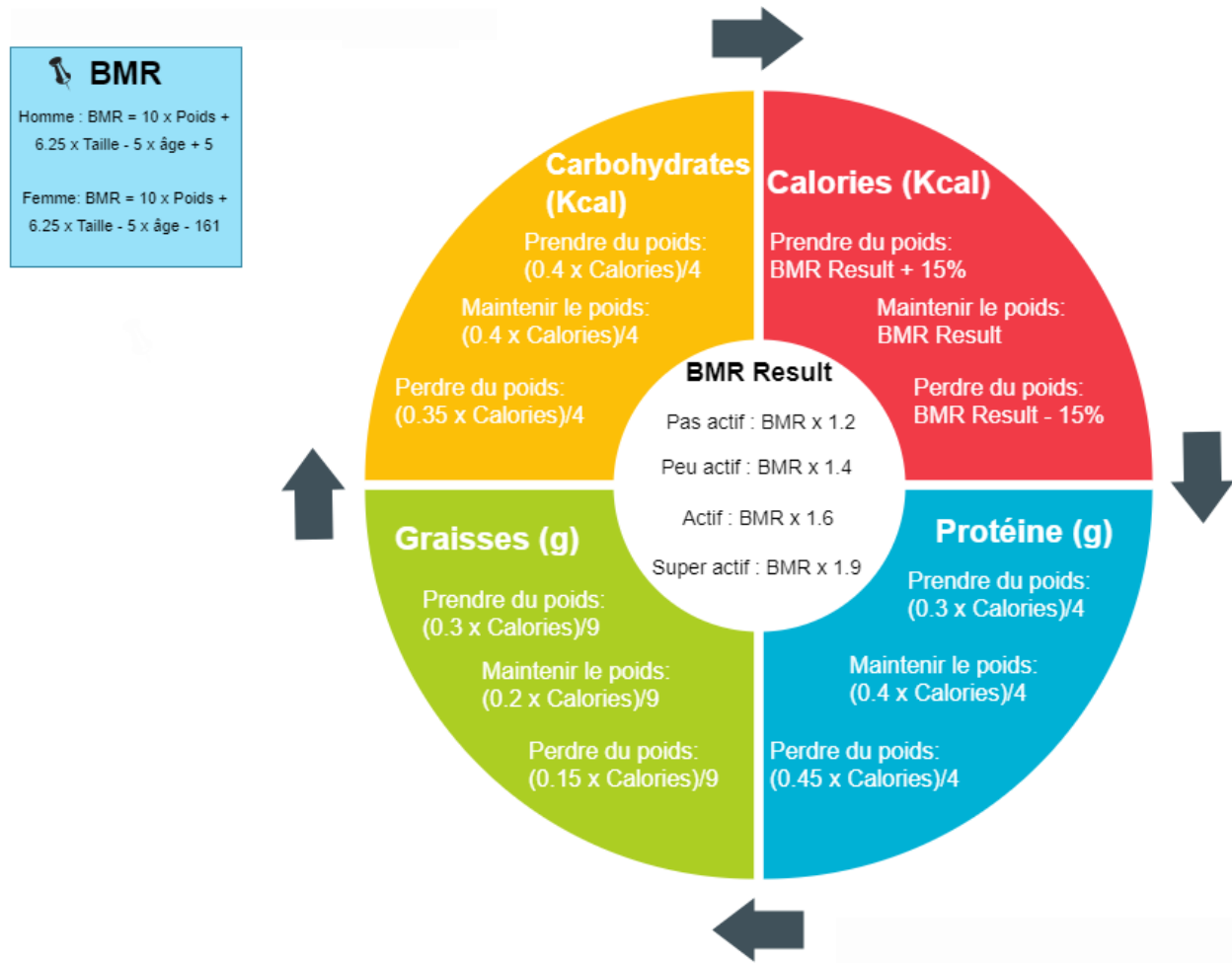


Figure 4.8 – Algorithme de calcul des besoins nutritifs

3. Framework Flutter

Flutter est un SDK (Software Development Kit) d'application permettant de créer des applications performantes pour iOS et Android à partir d'une base de code unique. En adoptant

Dart comme le langage fondamental du Framework, des outils d'implémentation des interfaces graphique dynamique de haute qualité sont fournis. Dart est connu par les différents 'widgets' qu'il offre, ce qui rend l'interface mobile plus harmonieux que des autres langages pour le développement mobile. D'ailleurs, les applications développées en utilisant les SDKs natifs sont spécifiques à la plateforme, par exemple Java / Kotlin pour Android et Swift / Objective-C pour iOS. Mais, ce qui rend Flutter intéressant, c'est que contrairement à d'autres Frameworks qui utilisent des widgets OEM, Flutter utilise ses propres ensembles de widgets riches et personnalisables. Ces widgets sont créés selon les directives de style Cupertino (iOS) et Material Design (Android) pour correspondre à l'aspect et à la convivialité des widgets natifs, mais avec moins de travail. La figure 4.9 représente l'architecture du Framework Flutter présentée par Yaftamobile.com (<https://blog.yaftamobile.com/en/flutter-a-game-changer-for-mobile-app-development/>)

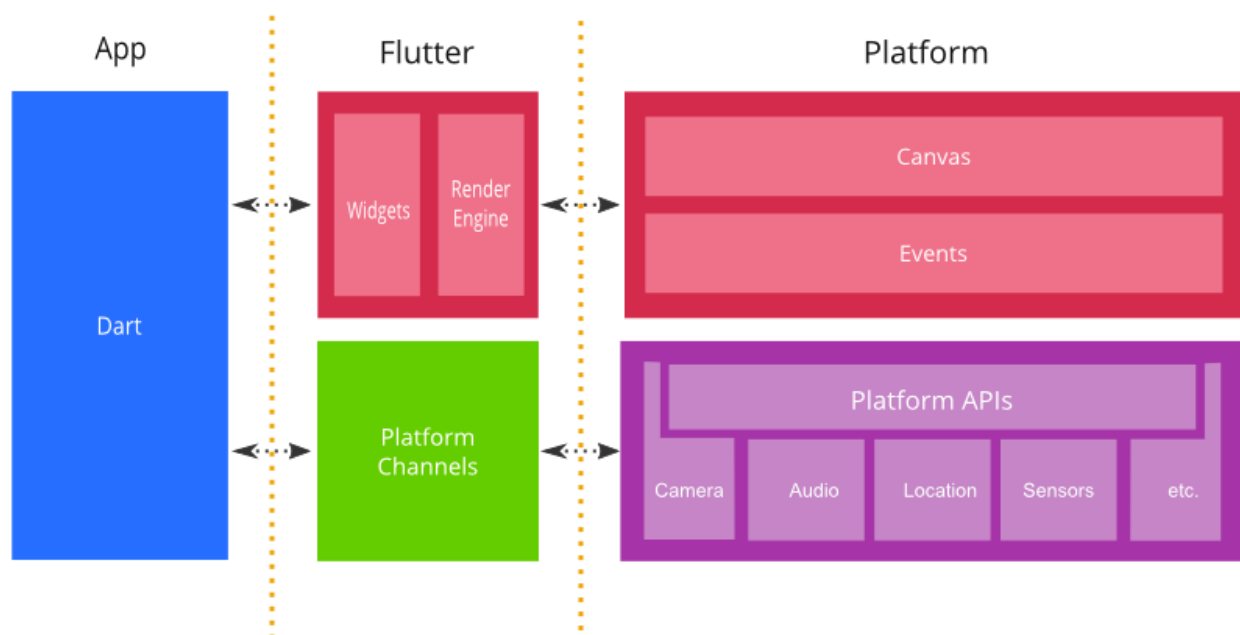


Figure 4.9 – Architecture du Framework Flutter

J'ai utilisé le Framework Flutter pour la réalisation du Front-End de l'application mobile.

3.1 Structure d'un projet Flutter

Dans cette partie, je présente la structure de mon projet Flutter afin de créer une application mobile avec une interface utilisateur dynamique.

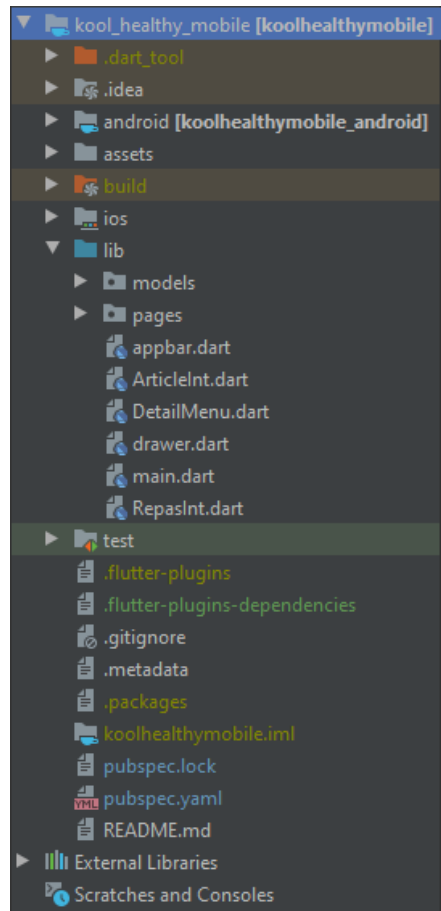


Figure 4.10 – Structure de mon projet Flutter

Vu que j’ai utilisé le Framework Flutter seulement pour la réalisation d’une interface mobile, je n’ai pas utilisé tous les composants de ce Framework. En fait, j’ai implémenté mon code seulement dans le répertoire **bin** qui stocke le contenu des interfaces utilisateurs avec un sous-dossier **pages** qui stocke les différentes interfaces et le sous-dossier **model** qui stocke les types des entités pour que Flutter reconnaisse les entités récupérées de la base de données.

3.2 Récupération des données

Vue que j’ai utilisé le Framework Flutter seulement pour la partie Front-End, j’ai besoin de faire une liaison avec le serveur pour la récupération et l’utilisation des données stockés dans la base de données. Donc, j’ai invoqué un REST API dans le code de mon application mobile.

REST API (REpresentational State Transfer)

REST (Representational State Transfer) est un style d'architecture logicielle définissant un ensemble de contraintes à utiliser pour créer des services web. Les services web conformes au style d'architecture REST, aussi appelés services web RESTful, établissent une interopérabilité entre les machines sur Internet. Dans le cas de mon projet, les deux machines (La machine locale et la machine mobile virtuelle) se communiquent via des requêtes HTTP locales.

L'invocation de la bibliothèque HTTP est donc nécessaire au niveau du Framework Flutter pour établir la connexion avec le serveur. Elle se fait dans le fichier **pubspec.yaml**.

```
environment:
  sdk: ">=2.2.0 <3.0.0"

dependencies:
  flutter:
    sdk: flutter
  http: ^0.12.0
```

Figure 4.11 – Capture d'écran des bibliothèques invoquées dans le Framework Flutter

La figure 4.12 représente un exemple de requête HTTP POST au niveau d'authentification d'utilisateur.

```
Future<List> _loginUser() async{
  final response = await http.post("http://10.0.2.2/projetpfe/loginUser.php", body: {
    "username" : emailC.text,
    "password" : passwordC.text,
  });
  var dataUser = json.decode(response.body);
  if(dataUser.length == 0){
    setState(() {
      error=true;
    });
  }else{
    Navigator.pushReplacement(
      context,
      MaterialPageRoute(
        builder: (context) => Home(
          connected: true,
          user: new User(int.parse(dataUser[0]['id']),dataUser[0]['nom'],dataUser[0]['username'],dataUser[0]['password'],dataUser[0]['sexe'],int.
        ), // MaterialPageRoute
      );
  }
}
```

Figure 4.12 – Capture d'écran de la fonction d'authentification via une requête HTTP POST

4. Travail réalisé

Dans cette partie, je présente quelques interfaces de mon application.

Interface ‘Accueil’

Les figure 4.13 et 4.14 illustrent l’interface d’accueil de mon application.

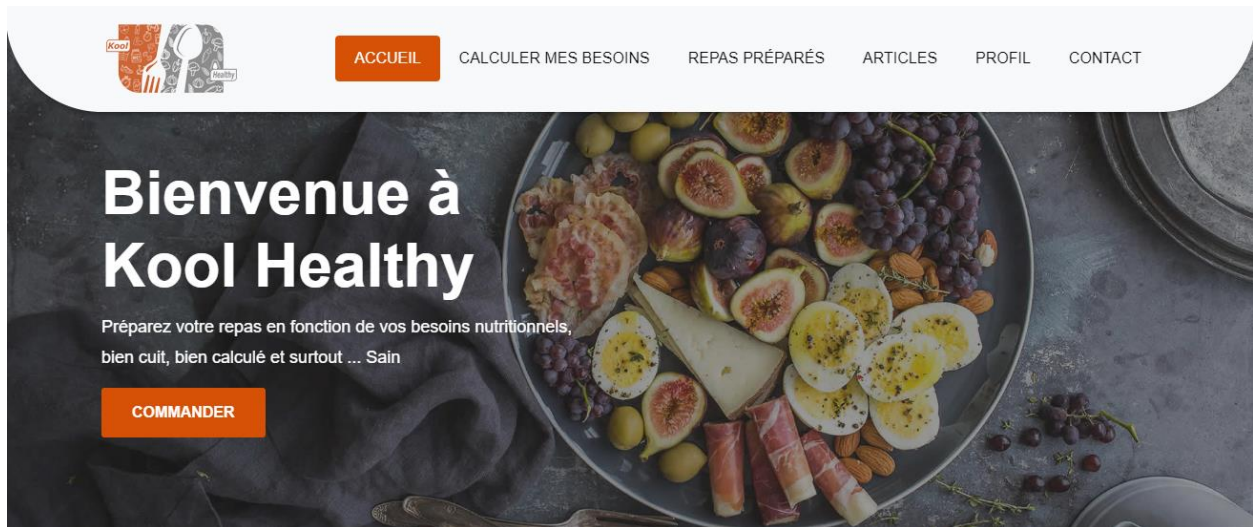


Figure 4.13 – Interface ‘Accueil’ de l’application web

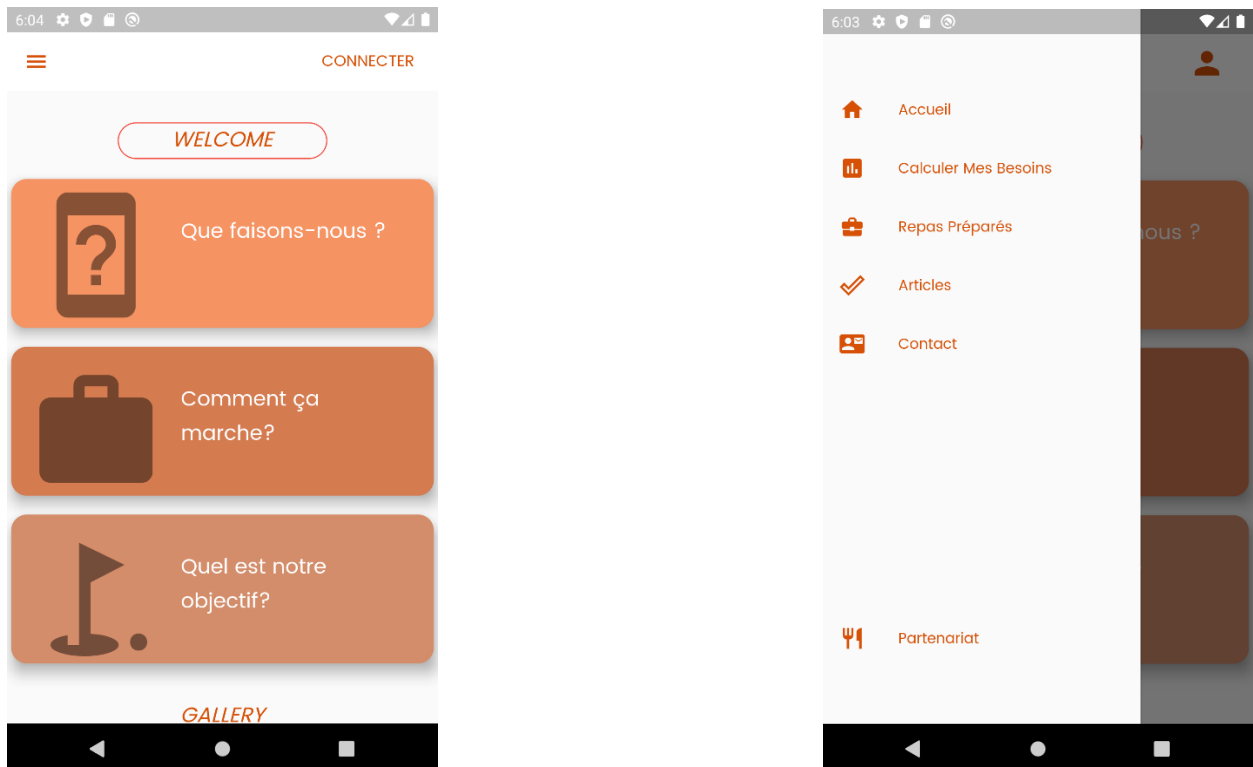


Figure 4.14 – Interfaces ‘Accueil’ et ‘Drawer’ de l’application mobile

Interface 'Login'

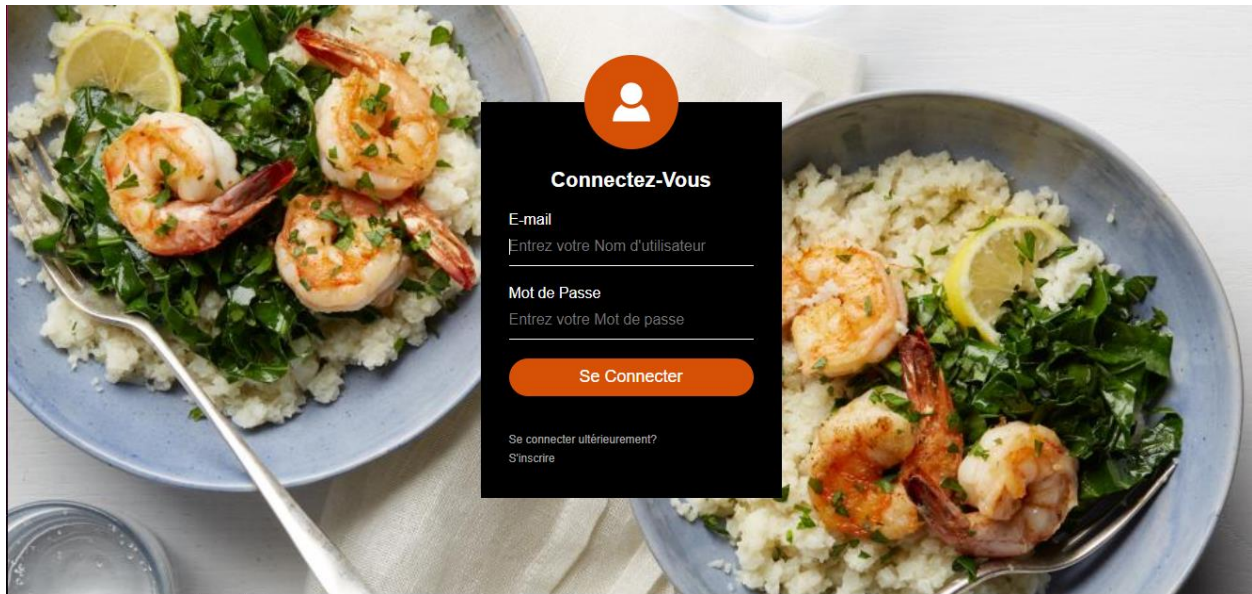


Figure 4.15 – Interface 'Login' de l'application web

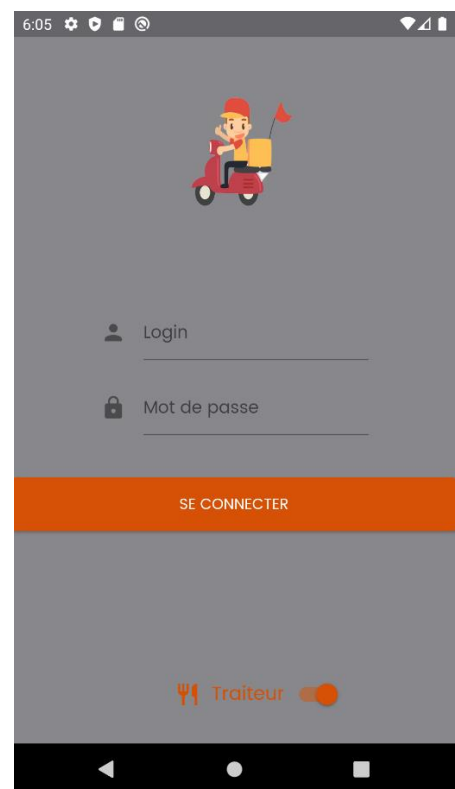
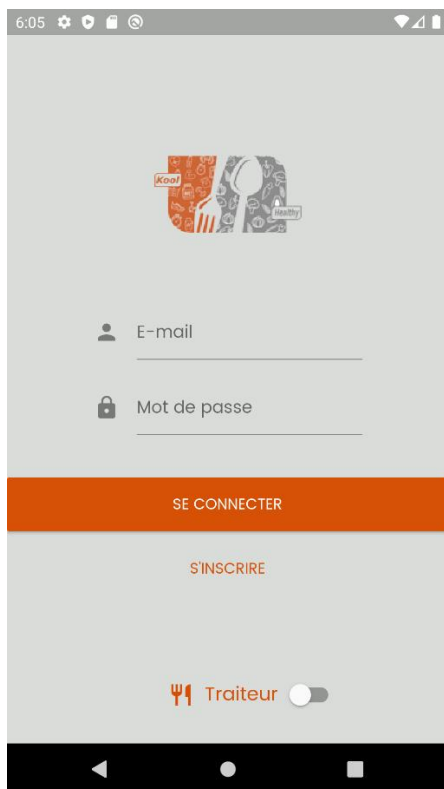


Figure 4.16 – Interfaces 'Login User' et 'login Traiteur' de l'application mobile

Interface 'Calculer Mes Besoins'

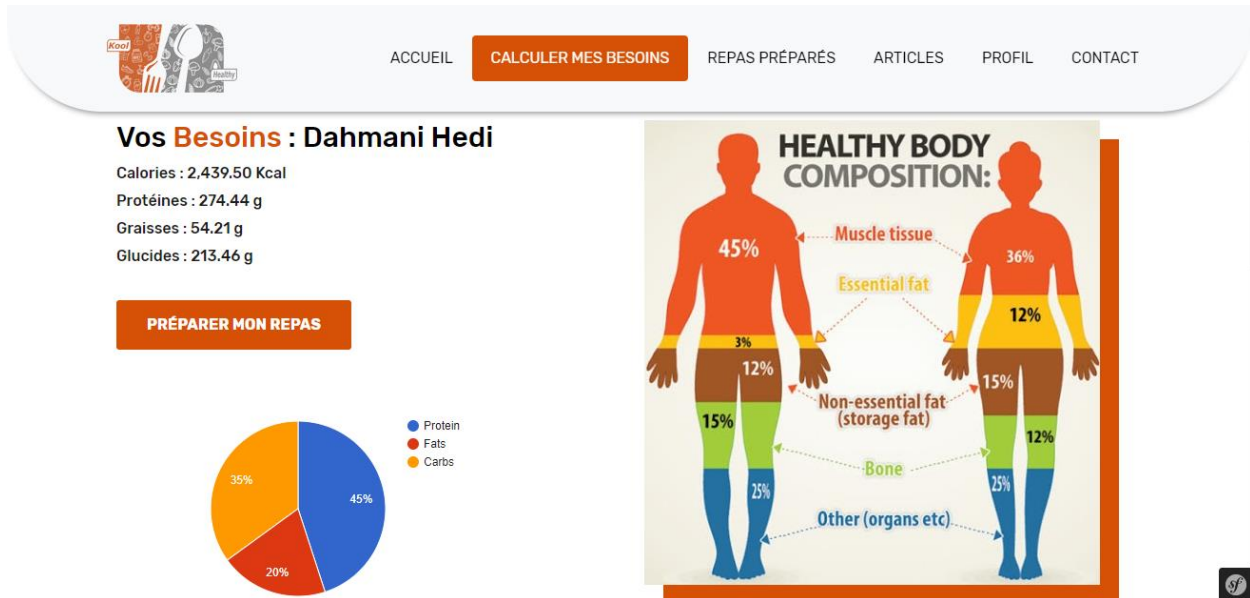


Figure 4.17 – Interface 'Calculer Mes Besoins' de l'application web

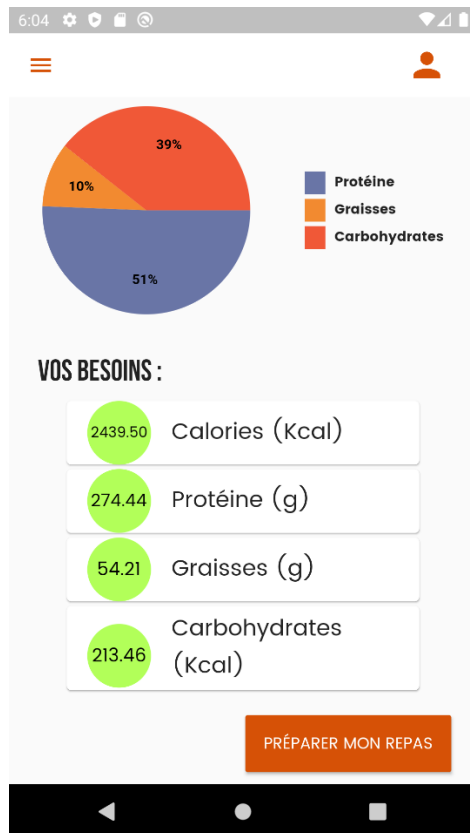


Figure 4.18 – Interface 'Calculer Mes Besoins' de l'application mobile

Interface 'Profil'

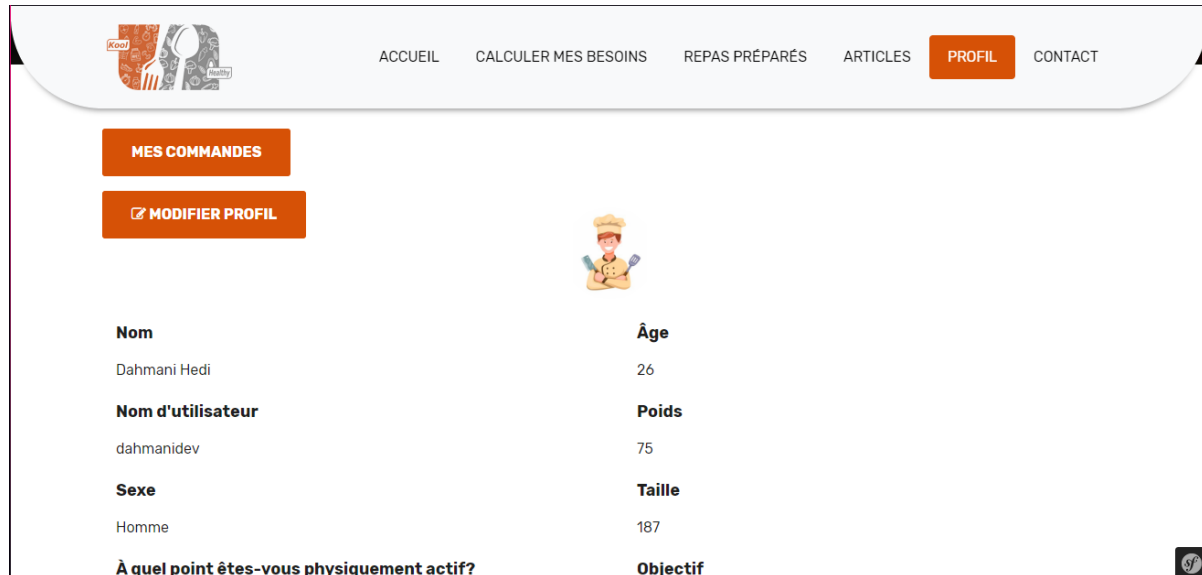


Figure 4.19 – Interface 'Profil' de l'application web



Figure 4.20 – Interface 'Profil' de l'application mobile

Interface 'Mes Commandes' : Utilisateur

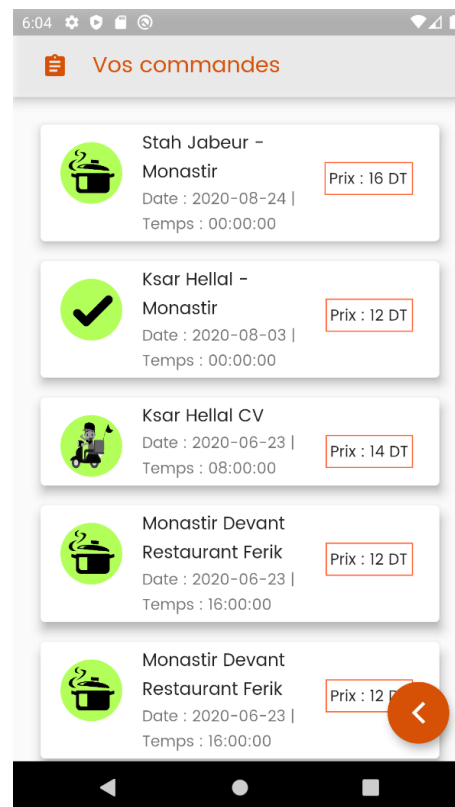


RETOUR

Vos commandes

Repas	Traiteur	Date	Adresse	Prix	Status
Soy-Maple Salmon	Mix Max	2020-06-04	Stah Jabeur - Monastir	16	En cours de préparation
Ginger Chicken Thighs	Mix Max	2020-06-07	Ksar Hellal - Monastir	12	Livré
Fresh salmon with thai noodle salad	Mix Max	2020-06-22	Ksar Hellal CV	14	En route

Figure 4.21 – Interface 'Mes commandes' de l'application web d'utilisateur



Vos commandes






- 
Stah Jabeur - Monastir
 Date : 2020-08-24 | Temps : 00:00:00
 Prix : 16 DT
- 
Ksar Hellal - Monastir
 Date : 2020-08-03 | Temps : 00:00:00
 Prix : 12 DT
- 
Ksar Hellal CV
 Date : 2020-06-23 | Temps : 08:00:00
 Prix : 14 DT
- 
Monastir Devant Restaurant Ferik
 Date : 2020-06-23 | Temps : 16:00:00
 Prix : 12 DT
- 
Monastir Devant Restaurant Ferik
 Date : 2020-06-23 | Temps : 16:00:00
 Prix : 12 DT

Figure 4.22 – Interface 'Mes commandes' de l'application mobile d'utilisateur

Interface 'Mes Commandes' : Traiteur

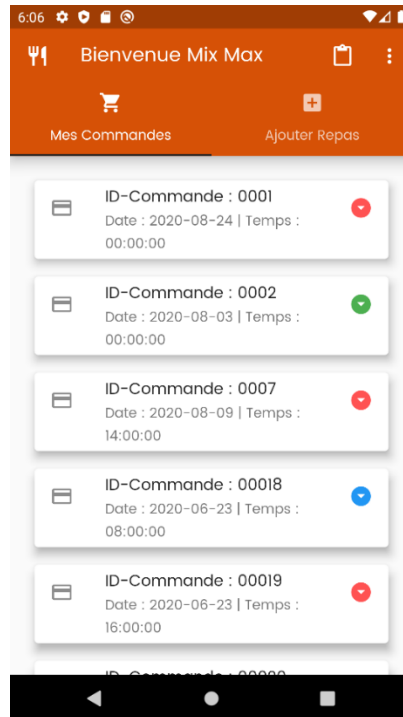


Figure 4.23 – Interface 'Mes commandes' du traiteur

Interface 'Admin'

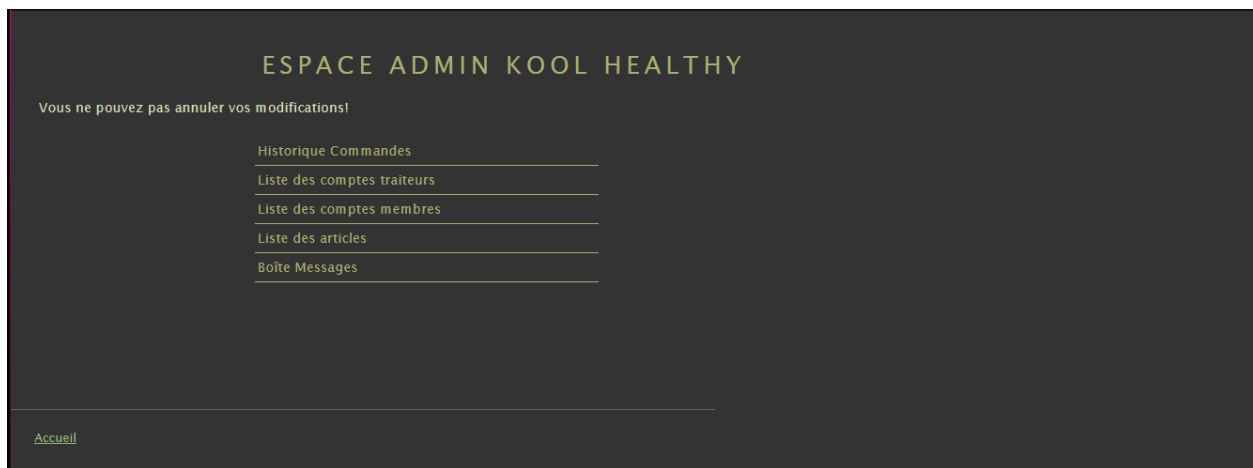


Figure 4.24 – Interface 'Admin'

Conclusion

Dans ce chapitre, j'ai donné une brève description de l'environnement matériel et logiciel du travail. Ensuite, j'ai présenté les différents Framework que j'ai utilisés pour l'implémentation de ce projet. Finalement, j'ai clôturé ce chapitre par des captures d'écran de quelques interfaces de mon application.

Conclusion générale

Ce rapport présente les travaux achevés dans le cadre de mon projet de fin d'étude au sein de la société « ONEDEV ». Ce projet était inspiré d'une idée de groupe 'Kool Healthy'. Durant ce projet, j'ai développé une plateforme web et mobile de Meal Prep. Les fonctionnalités prévues de mon application ont été implémenté à savoir le calcul des besoins nutritifs d'utilisateur, l'offre d'un menu de repas sains qui conviennent aux les valeurs nutritives calculées, le service de commande de repas, le suivi des commandes et la publication fréquents d'articles et de recettes. Il reste l'implémentation de la fonctionnalité qui permet à un utilisateur de commander plusieurs repas par commande.

Sur le plan pratique, j'ai étudié l'existant tout d'abord afin de s'inspirer des solutions existantes, éviter les lacunes et raffiner de plus le sujet de mon projet. J'ai justifié le choix de la méthodologie 2TUP. Ensuite, j'ai analysé et spécifié les besoins fonctionnels et non fonctionnels de mon projet. Dans la suite, j'ai passé à la partie de conception où j'ai présenté l'architecture de mon système, la modélisation de la base de données et la conception logicielle. Enfin, j'ai décrit l'environnement de développement, les technologies utilisés et j'ai présenté les résultats de travail réalisé à travers des captures d'écrans.

Durant la période de stage, j'ai acquis des connaissances en le langage UML durant la phase d'analyse et de conception. La prise en main des Frameworks Symfony et Flutter tout au long de la phase de réalisation qui me seront certainement utiles dans ma vie professionnelle, m'a permis de mesurer l'importance des qualités attendues d'un développeur. Ainsi, en pratiquant le stage chez « ONEDEV » et en travaillant avec le groupe 'Kool Healthy' j'ai eu l'occasion de développer mes capacités interpersonnelles de communication et d'intégration dans le monde professionnel.

En perspective, je suis conscient que ce projet n'est qu'un premier aperçu de la vie professionnelle. D'ailleurs, ce projet va évoluer d'ici Octobre 2020 en lançant l'entreprise 'Kool Healthy' avec plusieurs autres fonctionnalités comme un chat-bot, le service de paiement en ligne et un espace de E-coaching.

WEBLIOGRAPHIE

- [1] Site officiel SunBasket : <https://www.sunbasket.com> (dernière consultation 17/07/2020)
- [2] Site officiel ParsleyBox : <https://www.parsleybox.com> (dernière consultation 17/07/2020)
- [3] Site officiel BlueApron : <https://www.blueapron.com> (dernière consultation 17/07/2020)
- [4] Cycle de développement en V : <https://www.supinfo.com/articles/single/5131-imites-methodes-classiques> (dernière consultation 17/07/2020)
- [5] Processus Unifié : https://fr.wikipedia.org/wiki/Processus_unifié (dernière consultation 17/07/2020)
- [6] Symfony 4.4 : <https://symfony.com/doc/current/index.html> (dernière consultation 17/07/2020)
- [7] Flutter : <https://flutter.dev/docs> (dernière consultation 17/07/2020)
- [8] MySQL : <https://dev.mysql.com/doc/> (dernière consultation 17/07/2020)
- [9] Modèle MVC : <https://fr.wikipedia.org/wiki/Modèle-vue-contrôleur> (dernière consultation 17/07/2020)