République Algérienne et Démocratique Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université Abderrahmane Mira De Bejaïa
Faculté des Sciences Exactes
Département d'Informatique

جامعة بجاية
Tasdawit n Bgayet
Université de Béjaïa

Mémoire de fin cycle

En vue d'obtention du diplôme de Master Professionnel en informatique

Option : Génie Logiciel

Thème

Conception et Réalisation d'une application pour un magasin de pâtisserie

Réalisé par :

M[lle] Amokrane chanez

Encadré par :

M[r] OUZEGGANE Redouane

Membre de jury :

Président : M[r] MEHAOUED Kamel

Examinateur : M[r] BESSAAD Omar

Promotion 2019/2020

Remerciements

Je tiens tout d'abord à remercier DIEU le tout puissant et miséricordieux, qui m'a donné la force et la patience d'accomplir ce travail.

Je tiens à remercier mon encadrant ***M.OUZEGGUENE*** *pour avoir accepté de diriger ce travail. Son soutien, sa clairvoyance et ses compétences m'ont été d'une aide inestimable.*

Je tiens à remercier sincèrement les membres du jury qui me font le grand honneur d'évaluer ce travail.

Mes remerciements les plus chaleureux vont aussi à tous mes camarades pour leurs encouragements et pour l'ambiance agréable tout au long de ce stage et en particulier à ***BOUC****HBBAH et* ***BOUABBIDA*** *pour leur présence dans les moments difficiles et grâce a eux que j'ai passé d'excellents moments.*

Dédicaces

Ce travail est dédié :

A mes parents, pour leurs sacrifices déployés à mon égard, pour leur patience, leur amour et leur confiance. Qu'ils trouvent dans ce modeste travail, le témoignage de ma profonde affection et de mon attachement indéfectible. Nulle dédicace ne peut exprimer ce que je leur doit.

A mes sœurs et à mon frère

À mes amis.

Ahlem , kenza , wissame, celina ,sylia, lamia , juba

Je tiens à remercier chaleureusement, tous mes proches et tous ceux qui, de près ou de loin, m'ont apporté leurs sollicitudes pour accomplir ce travail.

Table Des Matieres

Liste des Figures

Liste des Tableaux

Liste des Abreviations

AJAX	**Asynchronous JavaScript And XML**
CSS	**Cascade Style Sheet**
HTML	**HyperText Markup Language**
HTTP	**Hypertext Transfer Protocol**
HTTPS	**HyperText Transfer Protocol Secure**
JQ	**JQuery**
JS	**JavaScript**
JSON	**JavaScript Object Notation**
NPM	**Npm Package Manager**
SGBD	**Système de Gestion de Base de Données**
SQL	**Structured Query Language**
TCP	**Transmission Control Protocol**
UML	**Unified Modeling Language**

Introduction Generale

Les nouvelles technologies d'information et de communication (TIC) jouent un rôle croissant dans notre société. À travers ces technologies, il est possible pour n'importe quel internaute ou organisation, de mettre à disposition un propre site ou application web, le plus souvent à but commercial.

Par conséquent, les systèmes d'informations des organismes et entreprises ont été ouvert au monde extérieure, grâce à l'internet et le Web, afin de répondre à un besoin important et grandissant de leurs clients, et ceci pour n'importe quel type d'organisation.

Dans cette optique d'ouverture et de commercialisation des services, les boutiques de pâtisserie veulent moderniser leurs gestions des produits finis et matières premières afin d'assurer un service rapide et accessible à une clientèle de plus en plus nombreuse et exigeante.

L'objectif de notre travail, dans le cadre de projet de fin de cycle de master, est de proposer une solution applicative aux boutiques de pâtisserie, pour cela, nous avons essayé de réaliser un stage, mais en vain. Pour remédier au problème de stage nous avons effectué des questionnaires et des interviews avec quelques gérants de ce type de magasin, afin de comprendre leurs méthodes de travail et les problèmes rencontrés quotidiennement.

En effet, après cette étude préliminaire, qui consiste juste à des questionnaires, nous avons décelé quelque problème de gestion, à savoir le suivi de stock des matières premières (ingrédients), aussi le suivi des produits finis (les gâteau), et difficulté d'établir des statistiques, spécialement, les statistiques hebdomadaire et mensuelles.

Pour bien présenter notre travail, ce présent mémoire est organisé autour de quatre chapitres.

Le premier est consacré à la présentation des technologies de développement web, classiques et modernes, spécialement après l'apparition de Node JS, qui a permis de révolutionner, à travers d'autres plateformes, la manière de développer les applications Web, Desktop et mobiles. Aussi nous avons présenté dans ce premier chapitre l'étude préalable et la problématique traitée, et nous l'avons conclu par un recueil des besoins fonctionnels et non fonctionnels.

Ces derniers feront l'objet d'une analyse, à travers quelques diagrammes de langage UML (Unified Modeling Language) et une démarche simplifiée, se basant sur l'identification des acteurs, leurs interactions avec le système, vu comme une boîte noire, à travers le diagramme de contexte dynamique. Par la suite, les fonctionnalités seront modélisées par des cas d'utilisation bien décrite et détaillée par des descriptions textuelles et des diagrammes de séquences système.

En troisième chapitre, nous entamons la phase cruciale de conception en éclatant le système à des objets, afin de spécifier comment les différents cas d'utilisation seront réalisés, sans entrer dans le détail technique. Pour cela, nous avons utilisé les diagrammes d'interaction qui utilise trois types d'objets : objets interface, objets contrôleur et objet entités. Ces derniers objets nous aiderons à établir le diagramme de

classe de domaine de notre application, suivi par le dictionnaire de données et le schéma relationnel.

Le dernier chapitre, consacré à la phase de réalisation, présente les différents outils, langages et environnements utilisé pour cette phase. Nous présenterons aussi le schéma physique de notre base de données, le diagramme de déploiement et quelques captures écrans de notre application.

Enfin, nous avons finalisé ce mémoire par une conclusion générale qui récapitule notre travail, suivie de quelques perspectives qui proposent la continuité de notre projet.

Chapitre I

Generalites sur les Technologies web et Etude Preliminaire.

I.1 Introduction

Vu l'importance et la puissance des technologies utilisé dans le développement Web, des techniques sont apparues afin d'exploiter ces technologies pour la programmation d'application Bureau (Desktop Application) et la programmation mobile multiplateforme (Android, IOS, Windows Mobile…). D'où l'utilisation du terme « Programmation Hybride ». Ces techniques sont apparues depuis les premières versions de Node JS (mai 2010) qui permet d'exécuter du code JavaScript en dehors du navigateur.

Dans ce chapitre, nous présenterons quelques technologies de la programmation Web, par la suite nous passons en revue Electron JS qui permet de créer des applications Desktop avec HTLM, CSS et JavaScript.A la fin de chapitre, nous effectuons une étude préliminaire du domaine de gestion de pâtisseries, en effectuons des questionnaires et des interviews avec quelques gérants de boutique de vente de pâtisserie. Cette étude nous permettra d'établir, sous forme d'un recueil de besoins, une liste de besoins fonctionnels et besoins non-fonctionnels.

I.2 Développement web

Le développement Web va de la création de pages en texte brut à des applications Web complexes, des applications de réseaux sociaux et des applications commerciales électroniques.

Les applications web ne sont pas conçues uniquement pour les Smartphone et tablette, mais pour tout navigateur internet ordinateur ou mobile.

I.3 Evolutions majeures application web

Le développement d'applications web a connu trois évolutions majeures

La première période se situe depuis la naissance du web jusqu'au milieu des années 2000. Il s'agissait de développer des applications classiques présentant des documents et des formulaires. Les technologies clientes utilisées se limitaient souvent au langage HTML et pour la fin de la période à l'utilisation de feuilles de style avec CSS. [1]

La deuxième période se situe entre 2005 et 2010. Cette période se distingue par l'arrivée des applications web riches (RIA : Rich Internet Application). La couche logique a été grandement déportée vers le serveur (grâce à la popularisation du langage côté serveur PHP). C'est aussi pendant cette période que le développement d'AJAX est apparu avec le style d'architecture REST (services web). De cette période, la technologie la plus utilisée a été la bibliothèque JQuery. Cette bibliothèque a facilité la manipulation de la partie de la structure HTML afin d'obtenir des applications qui réagissaient sans avoir à rafraichir l'intégralité d'une page complète. Malheureusement, c'est aussi à cette période que le nombre de navigateurs a explosé. Le développement était donc un peu chaotique, car il fallait supporter toutes les subtilités des navigateurs qui ne se conformaient pas forcément aux standards existants. Des initiatives via des bibliothèques

comme GWT sont apparues pour tenter de résoudre ce problème, mais cela restait quand même du bricolage. [1]

La troisième période se situe après 2010 jusqu'à nos jours. Nous sommes passés du terme applications web riches à applications web monopage (plus connu, en anglais sous le nom de SPA pour Single-Page application). Il s'agit désormais d'une application web où tout est accessible sur une page web unique afin de rendre l'expérience utilisateur la plus fluide possible. Pour obtenir ce résultat soit tout le contenu est chargé en avance soit à la demande, mais en tâche de fond. Il y a eu aussi un déport important de la logique vers le client afin de libérer des ressources côté serveur. Les effets de bord à ce transfert de code côté client a naturellement obligé les développeurs à structurer leur code et de facto à utiliser des outils adaptés (Grunt, Gulp, Yeoman, Webpack) pour réaliser des opérations sur cette masse de code (compilation, qualité de code,…). C'est ainsi que des Framework web JavaScript sont apparus.[1]

I.4 Développement d'application bureau

Lors de la recherche du meilleur cadre pour le développement d'applications de bureau, une classification assez simple peut être appliquée en fonction des tâches que notre future application de bureau devrait effectuer.

I.4.1 Développement d'applications Windows

Microsoft fournit à une communauté de développement une boîte à outils complète pour la création du puissant backend et du frontend brillant pour le developpement application windows

I.4.2 Développement application MacOS

Comme Microsoft, Apple encourage les développeurs à créer de belles versions du logiciel qui utilisent tous les avantages de l'architecture précise des ordinateurs Apple.

I.4.3 Application bureau multiplateformes

L'utilisation du cadre de bureau multiplateforme est le meilleur choix pour développer une application de bureau qui pourra ultérieurement être facilement transférée vers une autre plate-forme de bureau, des appareils mobiles et une application Web. Le principal avantage de travailler avec n'importe quel cadre de bureau multiplateforme est que vous créez la base de code unifiée qui peut plus tard être réutilisée pour la version SaaS du logiciel, augmentant ainsi considérablement le public cible.

I.5 Application client/serveur

Un system client /serveur fonctionne selon le schéma suivant :

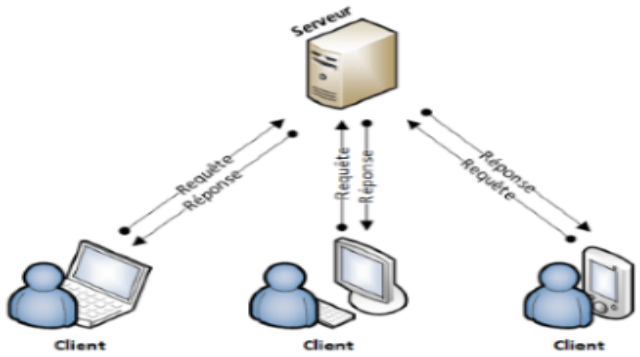


Figure I. 1 - Schéma représentant le fonctionnement d'une application client/serveur [3].

Client émet une requête vers le serveur grâce à son adresse IP et le port ; qui désigne un service particulier du serveur. Le serveur reçoit la demande et répond à l'aide de l'adresse de la machine client et son port [3]

Serveur web

Un « serveur web » peut faire référence à des composants logiciels (*software*) ou à des composants matériels (*hardware*) ou à des composants logiciels et matériels qui fonctionnent ensemble.

1. Au niveau des composants matériels, un serveur web est un ordinateur qui stocke les fichiers qui composent un site web (par exemple les documents HTML, les images, les feuilles de style CSS, les fichiers JavaScript) et qui les envoie à l'appareil de l'utilisateur qui visite le site. Cet ordinateur est connecté à Internet et est généralement accessible via un nom de domaine.

2. Au niveau des composants logiciels, un serveur web contient différents fragments qui contrôlent la façon dont les utilisateurs peuvent accéder aux fichiers hébergés.

> Chaque hébergeur propose des prestations et des solutions d'hébergement différentes selon le niveau de sécurité garanti, les tarifs appliqués, l'espace de stockage attribué à un site Web, les services apparentés, etc. [4]

I.6 L'architecture d'une application client/serveur

De nombreuses applications fonctionnent selon un environnement clients/serveur, cela signifie que des machines clientes (des machines faisant partie du réseau) contactent un serveur, une machine généralement très puissante en terme de capacités d'entrée-sortie, qui leur fournit des services.

En règle générale, une application est d´écopée en 3 niveaux (couches) d'abstraction :

• La couche présentation : c'est la partie de l'application visible par les utilisateurs (nous parlerons d'interface utilisateur). Dans notre cas, cette couche est un navigateur web, qui se présente sous forme de pages HTML, composée de formulaire et de bouton.

• La couche métier : correspond à la partie fonctionnelle de l'application, celle qui implémente la logique, et qui d´écrit les opérations que l'application opère sur les données, en fonction des requêtes d'un utilisateur effectué au travers de la couche présentation.

• La couche accès aux données : elle consiste en la partie gérant l'accès à la base de données du système [5]

Il existe différentes architectures pour une application web :

I.6.1 L'architecture 1 tiers

Dans une application un tiers, les trois couches applicatives sont intimement liées et s'exécutent sur le même ordinateur. On ne parle pas ici d'architecture client-serveur, mais d'informatique centralisée. Dans un contexte multiutilisateur, on peut rencontrer deux types d'architecture mettant en œuvre des applications un tiers :

Des applications sur site central, des applications réparties sur des machines indépendantes communiquant par partage de fichiers. [5]

I.6.2 L'architecture à 2 tiers

L'architecture s'appuie sur un poste central, qui envoie les données aux machines clientes:

• L'utilisateur émet une requête HTTP vers le serveur sur lequel est stockée la page HTML.

• Le serveur accède alors à ce fichier et le retourne au navigateur. Le navigateur interprète les balises HTML et affiche la page en résultat. [5]

Figure I. 2 - Architecture d'une application 2 tiers [5].

I.6.3 L'architecture à 3tiers

Le développement de sites interactifs, nécessitant de conserver des données sur les visiteurs, d'accéder à de grandes masses d'information ou de modifier régulièrement le contenu, repose aujourd'hui sur une architecture à 3 niveaux (ou architecture trois-tiers) entre serveur de données, serveur d'applications et client web

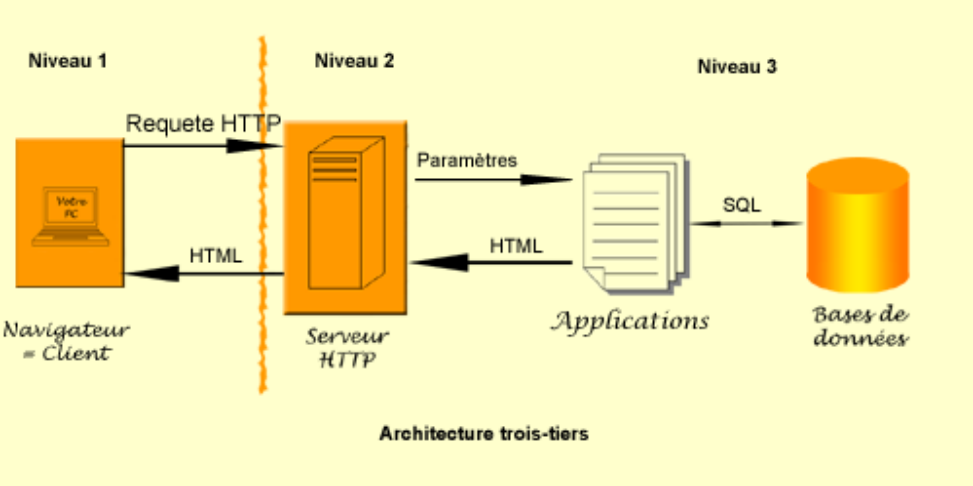


Figure I. 3 - Architecture d'une application 3 tiers [6].

Les 3 niveaux s'articulent dès lors de la manière suivante :

- Le premier niveau s'occupe de l'interface avec l'utilisateur depuis le navigateur.

- Le second héberge le serveur web qui est complété par le serveur d'application qui exécutent les traitements demandés lors de l'appel HTTP d'une page. Le serveur HTTP, aussi appelé middleware, est donc à la fois serveur et client. Serveur vis-à-vis du navigateur et client par rapport au serveur d'applications à qui il envoie une requête et dont il attend en retour le résultat. Une fois reçus, le serveur HTTP les compose dans un format assimilable par le navigateur client.

- Le troisième niveau assure la gestion des données au sein d'un SGBD (Système de Gestion de Bases de Données) et répond aux requêtes du serveur HTTP. [6]

I.6.4 L'architecture à n-tiers

Cette évolution des architectures trois tiers met en œuvre une approche objet pour offrir une plus grande souplesse d'implémentation et faciliter la réutilisation des développements.

L'appellation 'n-tiers'pourrait faire penser que cette architecture met en œuvre un nombre indéterminé de niveaux de service, alors que ces derniers sont au maximum trois (les trois niveaux d'une application informatique). En fait, l'architecture n-tiers qualifie la distribution d'application entre de multiples services et non la multiplication des niveaux de service. [6]

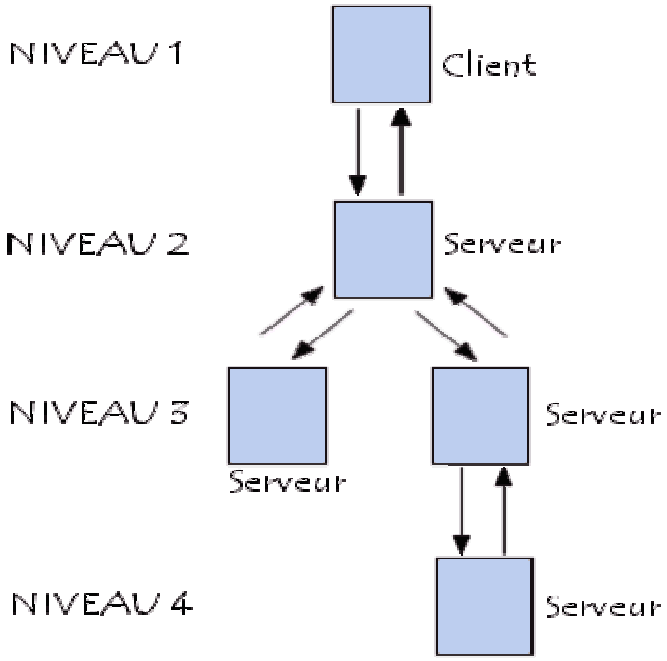


Figure I. 4 - Architecture d'une application n tiers [6].

I.7 Etude préalable

Avant la réalisation de notre travail, dans le cadre de projet de fin de cycle de Master en informatique, qui consiste à la conception et réalisation d'une application pour la gestion d'un magasin de pâtisserie, nous avons planifié de réaliser un stage. Malheureusement, et à cause de la pandémie, notre stage a été annulée. Par conséquent, nous avons basculé vers le travail par questionnaires et entretiens avec quelques gérants de ce type de magasin, afin de comprendre la méthode de travail et de gestion, ainsi que les problèmes important à résoudre pour ces gérants.

I.7.1 Objectif du projet

Notre projet s'inscrit dans le cadre de la gestion commerciale de magasin de pâtisserie, ceci pour simplifier la gestion des activités sous la responsabilité d'un gérant et de toute son équipe. Cela peut se faire grâce à des fonctionnalités automatisées telles que le suivi des ventes à distance. Cette dernière est essentielle pour gérer plusieurs magasins en même temps. L'application assure également le calcul des marges pour établir des bilans réguliers. Il est alors possible d'analyser le fonctionnement de la structure et d'adopter une meilleure stratégie si besoin.

Ainsi, nous aurons d'abord à structurer le système d'information existant impliqué d'une manière directe dans la gestion commerciale et nous aboutirons à un nouveau système qui permettra une gestion d'information efficace, ainsi qu'à l'automatisation de certaines tâches programmables, parmi ces tâches, nous pouvons citer l'enregistrement de différentes informations relatives à l'entrée et à la sortie des marchandises en stock, les informations relatives l'évolution de vente de chaque produit par période et par catégorie.

I.7.2 Problématique

La majorité des magasins pour lesquels nous avons effectué un entretien ne possèdent pas de solution informatique pour la gestion de leurs ventes. Ceci engendre beaucoup de problèmes :

- Manque d'une traçabilité des ventes effectuer (du jour, de la semaine …).
- Complication de calculer les recettes journalière hebdomadaire.
- Pour les magasins avec plusieurs vendeurs, difficulté de connaître le volume d'activité, en termes de nombre de ventes, pour chacun d'eux.

• complexité de posséder les mouvements du stock de la matière première et des produits fini (pièce de gâteau).

• Dans le cas de gérants possédant plusieurs magasins, difficulté de gérée ces dernier a partir de même endroit.

I.8 Recueil des besoins

Dans cette section, nous présenterons les différents besoins que l'application doit répondre. Nous avons deux types de besoins, besoins fonctionnels et besoins non-fonctionnels (besoins techniques). Les besoins fonctionnels sont les fonctionnalités que l'application doit assurer pour le client, et les besoins non-fonctionnels (aussi dits : besoins techniques) qui ont le trait sur les aspects de sécurité, ergonomie de l'IHM, fonctionnement sous réseau, …

Pour notre application, nous avons recensé les besoins spécifiés dans les deux sections ci-dessous.

I.8.1 Besoin fonctionnel

Notre application doit rependre aux besoins fonctionnels suivants :

- L'authentification : accès sécurisé par un identifiant et un mot de passe.
- La gestion des gâteaux pouvoir ajouter, supprimer, modifier et consulter les gâteaux
- La gestion des catégories pouvoir ajouter, supprimer, modifier et consulter les catégories

- La gestion des ventes pouvoir ajouter, supprimer, modifier et consulter les ventes

- La gestion de la production journalières pouvoir ajouter, supprimer, modifier et consulter les gâteaux de comptoir

- La gestion des clients pouvoir ajouter, supprimer, modifier et consulter les clients

- La gestion des stocks pouvoir ajouter, supprimer, modifier et consulter les stocks

- La gestion des fournisseurs pouvoir ajouter, supprimer, modifier et consulter les fournisseurs

- La gestion des lots pouvoir ajouter, supprimer, modifier et consulter les lots

I.8.2 Besoin non fonctionnel

Pour le développement de cette application, plusieurs besoins non fonctionnels s'imposent :

- L'ergonomie des interfaces et la facilité d'utilisation, l'application doit offrir des interfaces simples et conviviales pour l'interaction avec l'utilisateur.

- L'extensibilité : l'application doit pouvoir être extensible afin de pouvoir introduire d'autres fonctionnalités.

- Sécurité : Protéger l'accès à l'application par un identifiant et un mot de passe.

- Le temps de réponse de l'application doit être acceptable.

I.9 Méthode de conception et démarche de développement

Pour réaliser une application, il ne convient pas de se lancer tête baissée dans l'écriture du code source il faut tout d'abord bien comprendre les besoins, les documenter, puis définir les étapes et les phases de la réalisation. C'est cette démarche entièrement à l'écriture que l'on appelle modélisation. Pour ce faire, on utilisera le langage UML (Unifed Modeling Language) et une méthode simple et générique qui se situe a mi-chemin entre UP (Unified Process), qui constitue un cadre générale très complet de processus de développement, et XP (eXterme Programming) qui est une approche minimale a la mode contré sur le code.

I.10 Conclusion

Dans ce premier chapitre nous avons commencé à cite quelques notions de base sur le développement web et bureau par la suite nous avons établi une étude préalable ou nous avons posé la problématique et tracé quelque objectifs, et la solution proposé. Nous avons conclu ce chapitre par un recueil de besoins fonctionnels et non-fonctionnels et la méthode de conception et démarche suivi. Dans le chapitre suivant, nous allons entamer la phase d'analyse des besoins.

Chapitre II

Analyse des Besoins

II.1 Introduction

Dans ce chapitre nous allons présenter les différents acteurs de notre système, leurs rôles, les différentes interactions ainsi que les besoins qui seront modélisés par un diagramme de cas d'utilisation pour chaque entité, la description textuelle de ces cas d'utilisation et des diagrammes de séquence système.

II.2 Identification des acteurs

Un acteur représente un rôle joué par une entité externe (utilisateur humain, dispositif matériel ou autre système) qui interagit directement avec le système étudié. Un acteur peut consulter et /ou modifier directement l'état du système, en émettant et /ou en recevant des messages susceptibles d'être porteur de données [7]. En ce qui concerne notre système, nous avons pu identifier les acteurs suivants :

L'administrateur Joue un rôle primordial dans l'assurance du bon fonctionnement du système. C'est la personne qui prend en charge :

- La gestion de catégories

- La gestion de gâteaux
- Gestion clients
- Gestion de production journalière
- Gestion de vente

Operateur : c'est un acteur qui prend en charge la gestion de vente

En plus de ces deux acteurs, nous ajoutons l'acteur générique « Utilisateur » qui représente toute entité non encore identifié par le système. Ainsi pour cet acteur, la seule fonctionnalité accessible est l'authentification. Le diagramme suivant illustre la relation entre les trois acteurs :

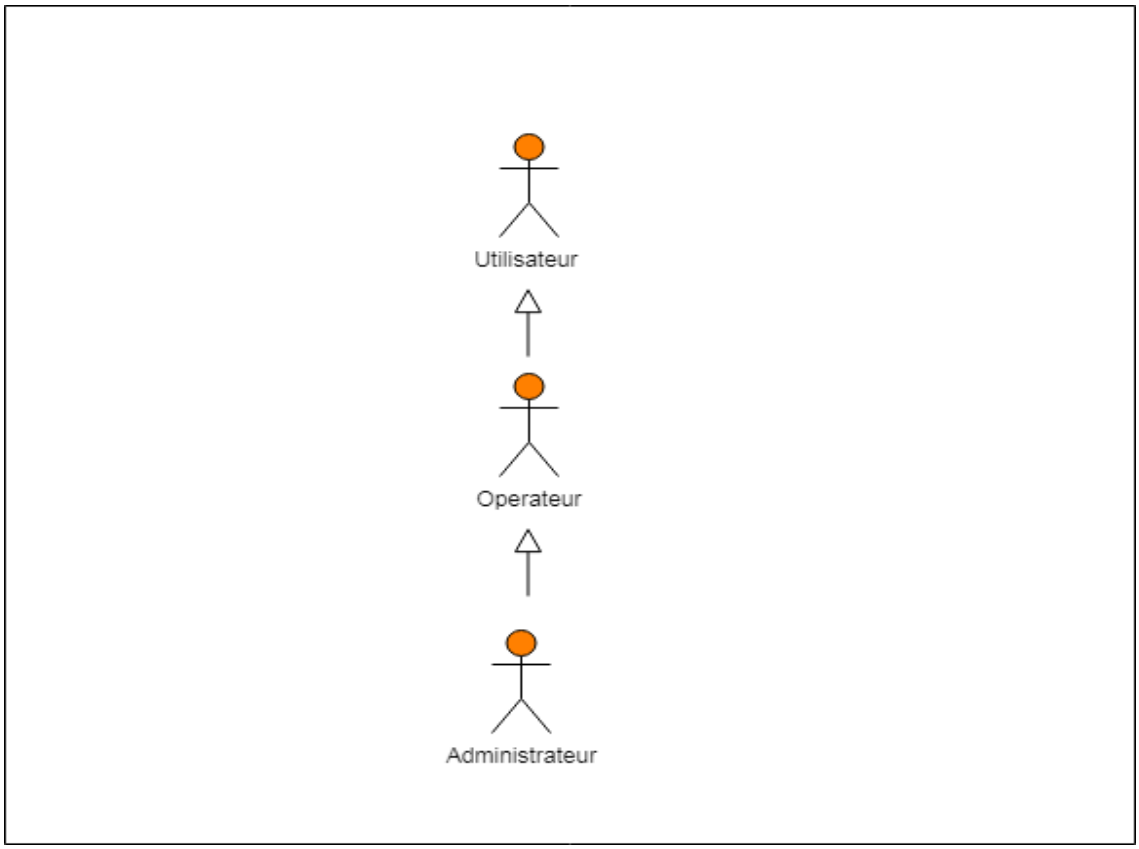


Figure II. 1 - Relation entre les acteurs du système.

II.3 Diagramme de contexte dynamique

Dans le processus d'analyse, le diagramme de contexte se situe au début. Son objectif est simple. Il doit présenter le système à modéliser on générale sous la forme d'une boite noire et les différents acteurs qui interagissent avec le système

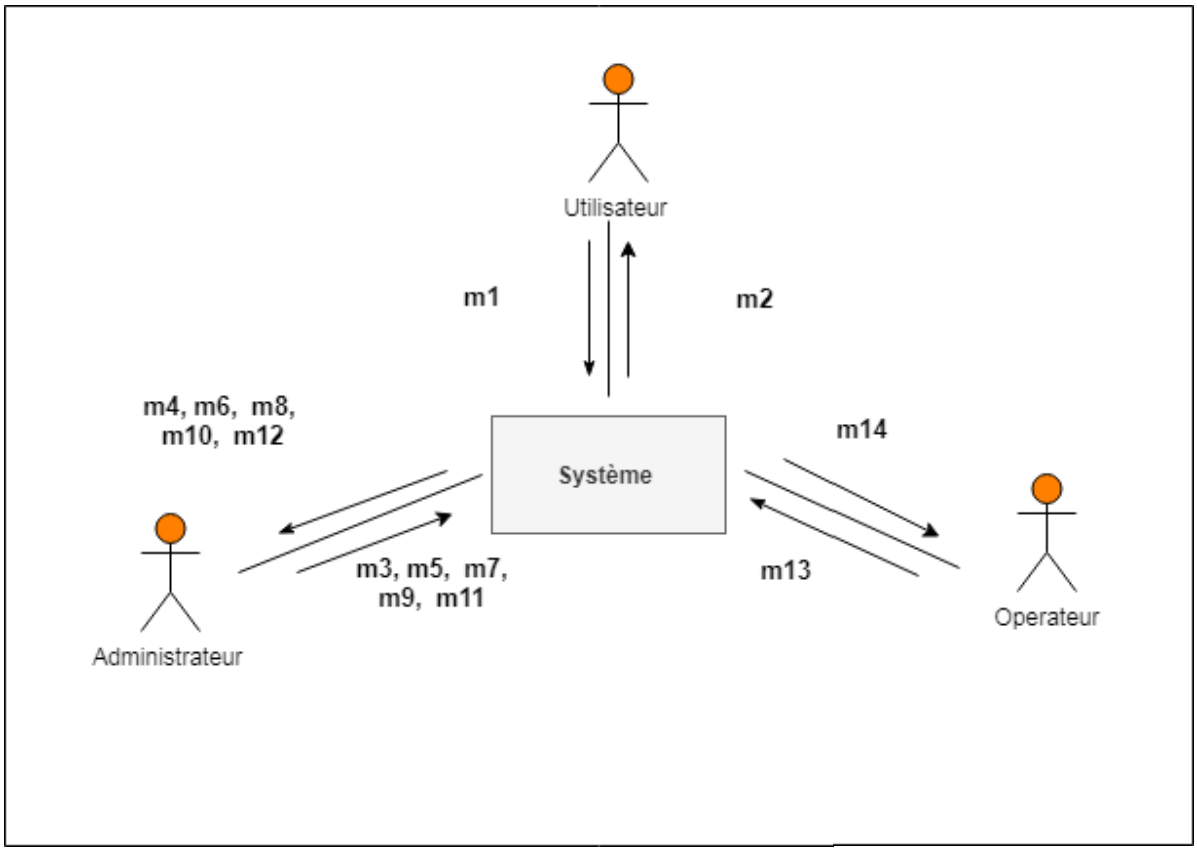


Figure II. 2 - Diagramme de Contexte.

Les différents messages sont explicités dans le tableau suivant :

Acteur	Message Acteur -> Système		Message Système -> Acteur	
Utilisateur	**M1**	L'authentification (identifiant et mot de passe)	**m2**	Réponse du système (soit erreur soit l'affichage de l'interface d'accueil selon le rôle de l'acteur
Administrateur	**m3**	Accès à la gestion des catégories	**m4**	Affichage de la liste des catégories, avec possibilité d'ajout, de suppression et de modification.
	m5	Accès à la gestion des gâteaux	**m6**	Affichage de la liste des gâteaux, avec possibilité d'ajout, de suppression et de modification.
	m7	Accès à la gestion des clients.	**m8**	Affichage de La liste des clients avec possibilité d'ajout, de suppression et de modification.
	m9	Accès à la gestion ventes	**m10**	Affichage de la liste des ventes avec la possibilité d'ajout, de suppression, de modification
	m11	Accès à la gestion de la production journalière	**m12**	interface de mise à jour des produits journaliers
Operateur	**M13**	Accès à la gestion ventes	**M14**	Interface de mise à jour des ventes.

Tableau II. 1 – Messages entre Acteurs et Systèmes.

II.4Cas d'utilisation

D'après **[07]** (*p. 62*), pour un acteur particulier du système, un cas d'utilisation représente une séquence d'actions qui seront réalisées par le système afin de répondre à un besoin métier et produire un résultat observable et intéressant. Il permet de décrire *ce que* le futur système doit faire, sans spécifier *comment* il le fera.

II.4.1Identification des cas d'utilisation

Dans ce qui suit, nous allons énumérer les différents cas d'utilisation pour chaque acteur du système. Pour mieux présenter ces cas d'utilisation, nous avons opté pour une structure tabulaire Le tableau suivant présente les différents cas d'utilisation identifié pour notre système :

N	Cas d'utilisation		Acteur
1	S'authentifier	Connexion	Utilisateur
		Déconnexion	
3	Gérer les catégories	Consulter	Administrateur
		Ajouter	
		Modifier	
		Supprimer	
4	Gérer les Gâteaux	Consulter	Administrateur
		Ajouter	
		Modifier	
		Supprimer	
5	Gérer les clients	Consulter	Administrateur
		Ajouter	
		Modifier	
		Supprimer	
9	Gérer la production journalière	Consulter	Administrateur
		Ajouter	
		Modifier	
		Supprimer	
10	Gérer les ventes	Consulter	Administrateur
		Ajouter	
		Modifier	
		Supprimer	
		Exporter	
11	Gérer les ventes	Consulter	Operateur
		Ajouter	

Tableau II. 2 – Les cas d'utilisation du système.

II.4.2 Description textuelle des cas d'utilisation

La description des cas d'utilisations est libre. Cependant, cette description prend souvent une forme rédigée qui convient mieux à la communication avec les utilisateurs.

Des règles de structuration doivent être appliquées pour en faciliter l'expression, la compréhension et la cohérence.

Pour exprimer les cas d'utilisations de notre système, nous avons choisi le formalisme décrit dans le tableau [Tableau II. 3]

Description textuelle de cas d'utilisation se Connecter

Cas d'utilisation N 1	Se Connecter
Acteur	Tous les acteurs
Pré condition	Application accessible S'authentifié
Scénario alternatif	Les informations fournies sont incorrectes ; Le système réaffiche le formulaire d'authentification et attend que l'utilisateur ressaisisse ses informations.
Scénario d'exception	Compte inexistant.

Tableau II. 3 – Description du cas « S'authentifier ».

Description textuelle de cas d'utilisation gestion des gâteaux

Cas d'utilisation N 4		Gérer les gâteaux
Ajouter	Objectif	Ajouter a la liste des gâteaux un nouveau gâteaux
	Acteur	Administrateur
	Pré condition	Application accessible S'authentifié
	Scénario nominal	[Début] L'administrateur demande l'interface de mise à jour de liste des gâteaux ; Le système affiche la page ; L'administrateur choisit l'ajout ; Le système affiche le formulaire d'ajout ; L'administrateur saisit les données et confirme ; Le système confirme l'ajout ; [Fin]
	Scénario alternatif	En cas de champ invalide le système affiche un message d'erreur ;
Modifier	Objectif	Modifier les informations d'un gâteau existant ;
	Acteur	Administrateur
	Pré condition	Application accessible S'authentifié
	Scénario nominal	[Début] L'administrateur demande l'interface de mis a jour de liste des gâteaux ; Le système affiche la page ; L'administrateur sélectionne et modifie les informations souhaiter et valide ; Le système confirme la modification ; [Fin]
	Scénario alternatif	En cas de saisit de données non compatible aux champs le

		système affiche un message d'erreur
Supprimer	Objectif	Supprimer un gâteau ;
	Acteur	Administrateurs
	Pré condition	Application accessible S'authentifié
	Scénario nominal	[Début] L'administrateur demande l'interface de mis a jour de liste des gâteaux ; Le système affiche la page ; L'Administrateur sélectionne gâteaux à supprimer ; Le système envoie une boite dialogue pour la confirmation de suppression L'administrateur confirme la suppression ; Le système supprime le gâteau indiqué [Fin]
	Scénario alternatif	Aucun

Tableau II. 4 – Description du cas « gérer les gâteaux ».

II.4.3 Diagramme de cas d'utilisation

Le diagramme de cas d'utilisation est un diagramme UML utilisé pour donner une vision globale du comportement fonctionnel d'un système logiciel. Un cas d'utilisation représente une unité discrète d'interaction entre un utilisateur (humain ou Machine) et un system. [8]

Diagramme de cas d'utilisation pour Operateur

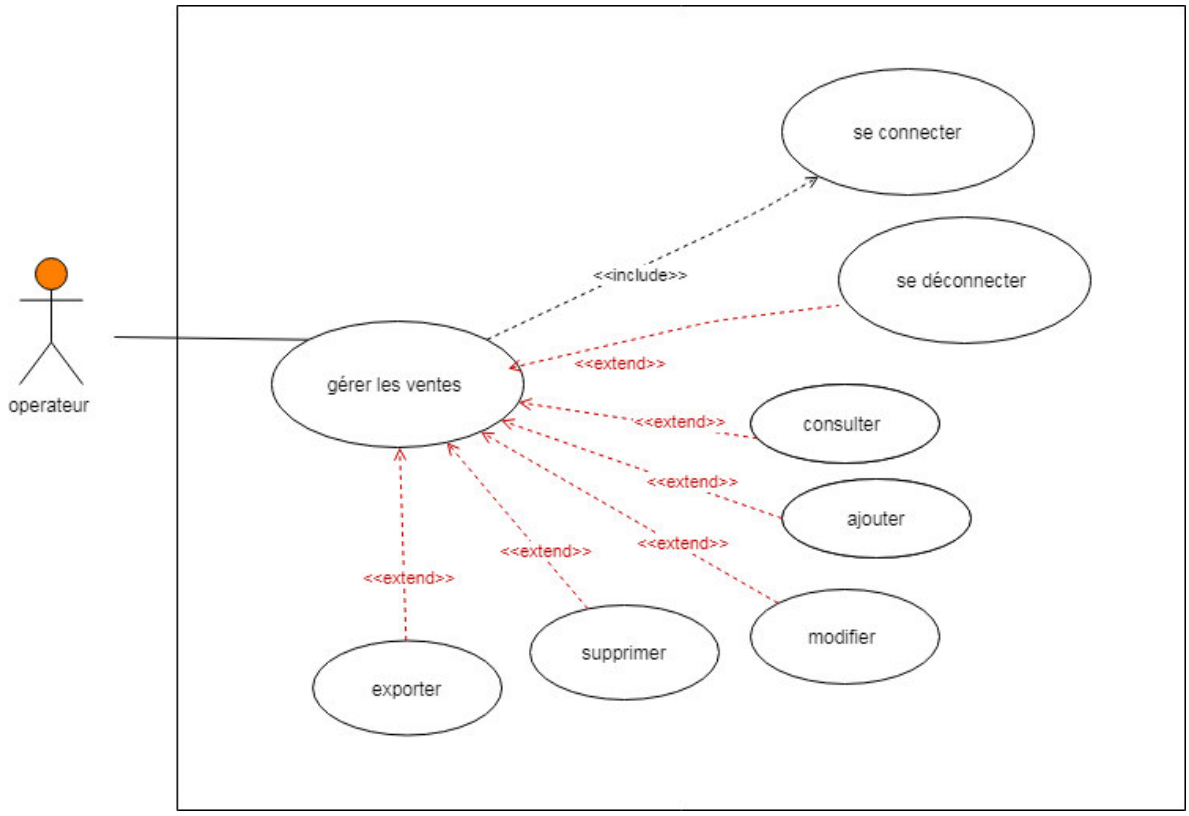


Figure II. 3 - Diagramme des cas d'utilisation pour operateur.

Diagramme de cas d'utilisation pour Administrateur

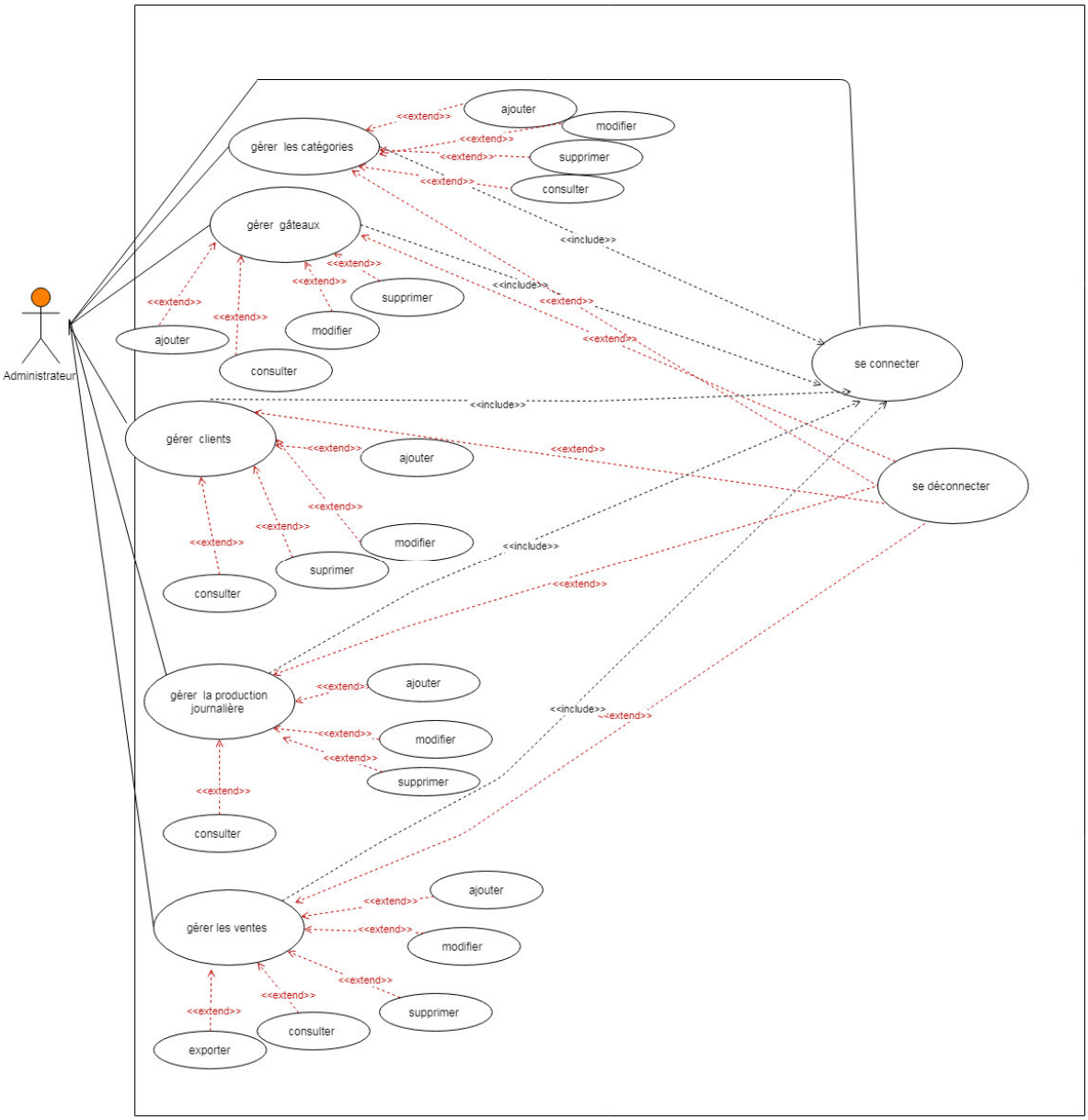


Figure II. 4 - Diagramme des cas d'utilisation pour administrateur.

Diagramme de cas d'utilisation globale

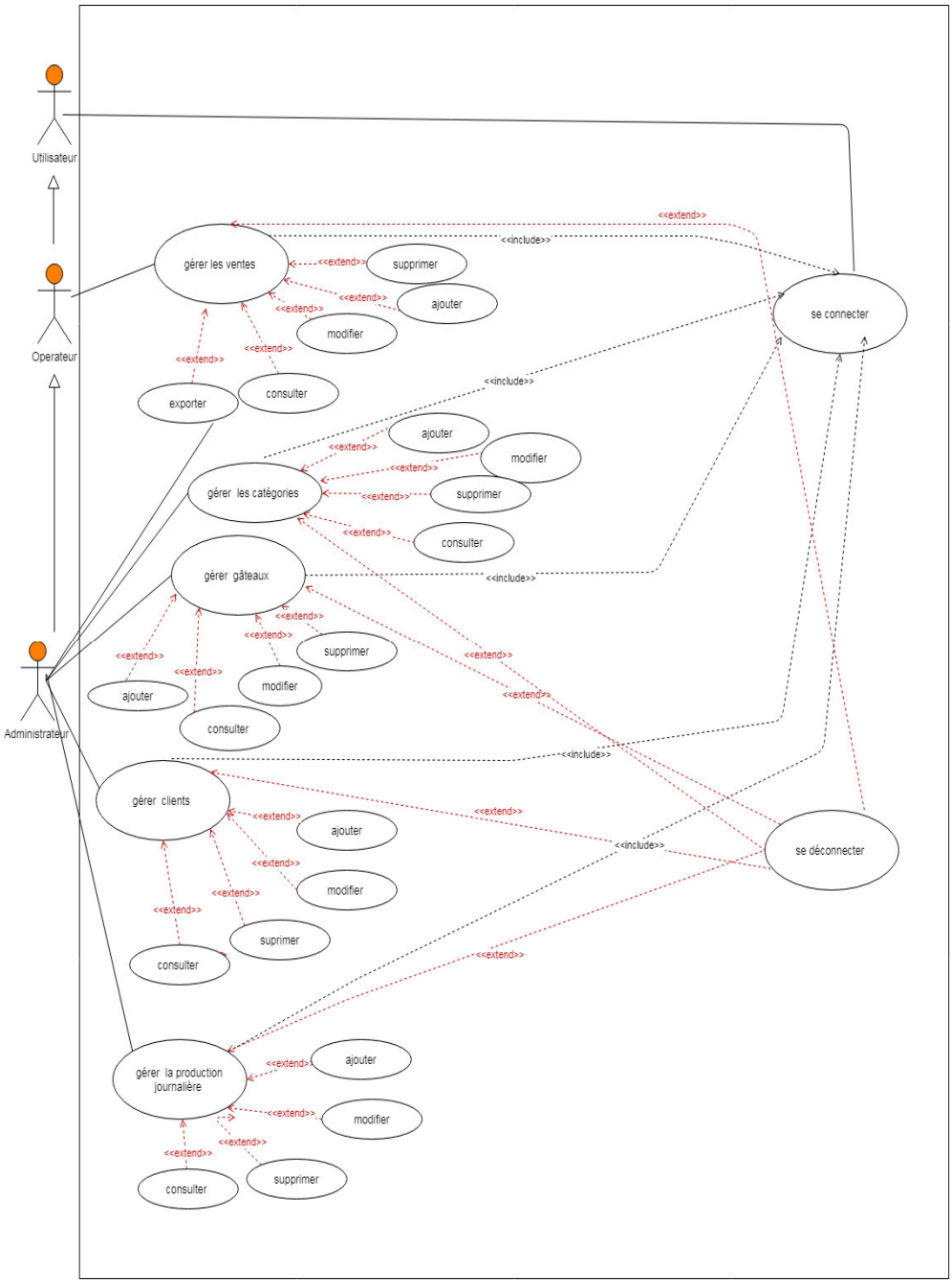


Figure II. 5 - Diagramme des cas d'utilisation globale.

II.5 Diagramme de Séquence Système

Un diagramme de séquence est un diagramme UML (Unified Modeling Language) qui représente la séquence de messages entre les objets au cours d'une interaction. Un diagramme de séquence comprend un groupe d'objets, représentés par des lignes de vie, et les messages que ces objets échangent lors de l'interaction.

Les diagrammes de séquence représentent la séquence de messages transmis entre des objets. Ils peuvent également représenter les structures de contrôle entre des objets. Par exemple, les lignes de vie dans un diagramme de séquence pour un scénario de banque peuvent représenter un client, un guichetier ou un responsable d'agence. Les communications entre le client, le guichetier et le responsable sont représentés par les messages entre ces derniers. Le diagramme de séquence représente les objets et les messages entre ces objets. [9]

Nous allons maintenant présenter quelques diagrammes de séquence qui sont les plus utiles pour la suite

Diagrammes de séquence le cas d'utilisation « Se connecter ».

L'authentification consiste a assurer la confidentialité des données, elle se base sur la vérification des informations associées a un utilisateur (un identifiant et un mot de passe dans notre application).

Ce diagramme décrit le scénario du cas d'utilisation Se Connecter. Dans un premier lieu une page d'authentification s'affiche. L'utilisateur saisie l'identifiant et le mot de passe puis valide ses données en clique sur le bouton de s'authentifier. Ensuite le système vérifie s'il s'agit bien d'un membre ou pas. La figure suivante illustre une description détaillée du ce scénario.

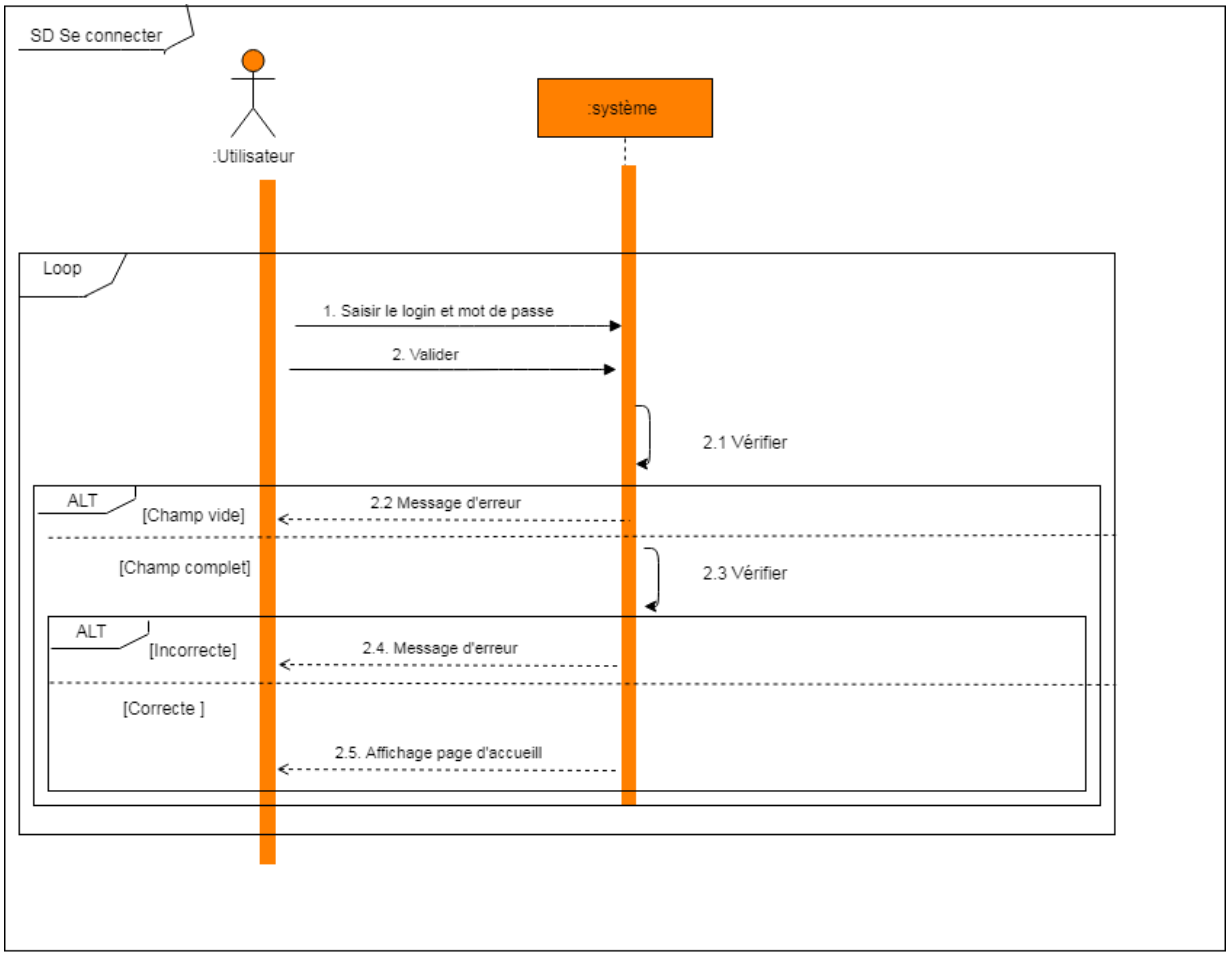


Figure II. 6 - Diagramme de Séquence Système « Se connecter ».

Diagrammes de séquence le cas d'utilisation Ajouter un gâteau

La figure suivante (Figure III.7) montre l'interaction entre l'acteur Administrateur et le système afin de réaliser l'opération « ajouter un gâteau ».

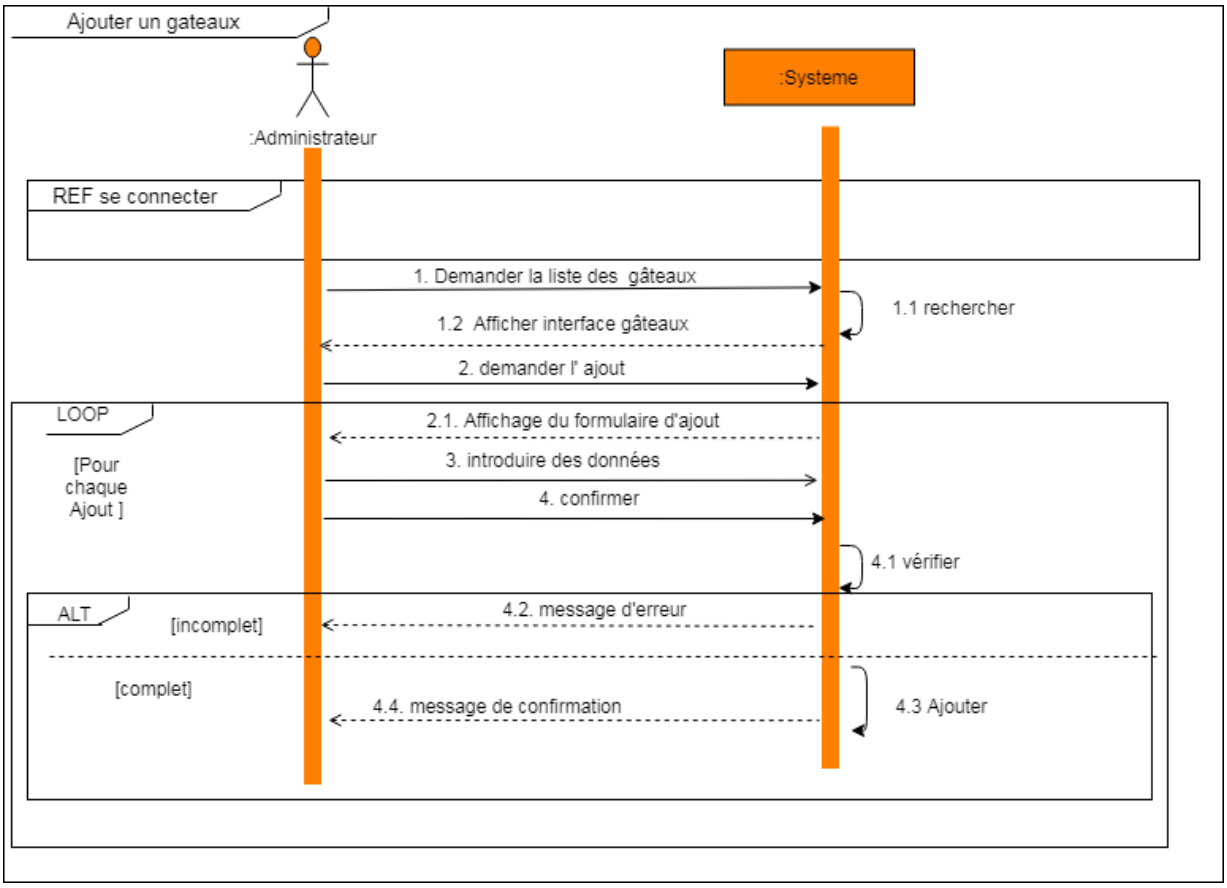


Figure II. 7 - Diagramme de Séquence Système « Ajouter un gâteau ».

Diagrammes de séquence le cas d'utilisation Supprimer un gâteau.

La figure suivante (Figure III.8) montre l'interaction entre l'acteur Administrateur et le système afin de réaliser l'opération « Supprimer un gâteau ».

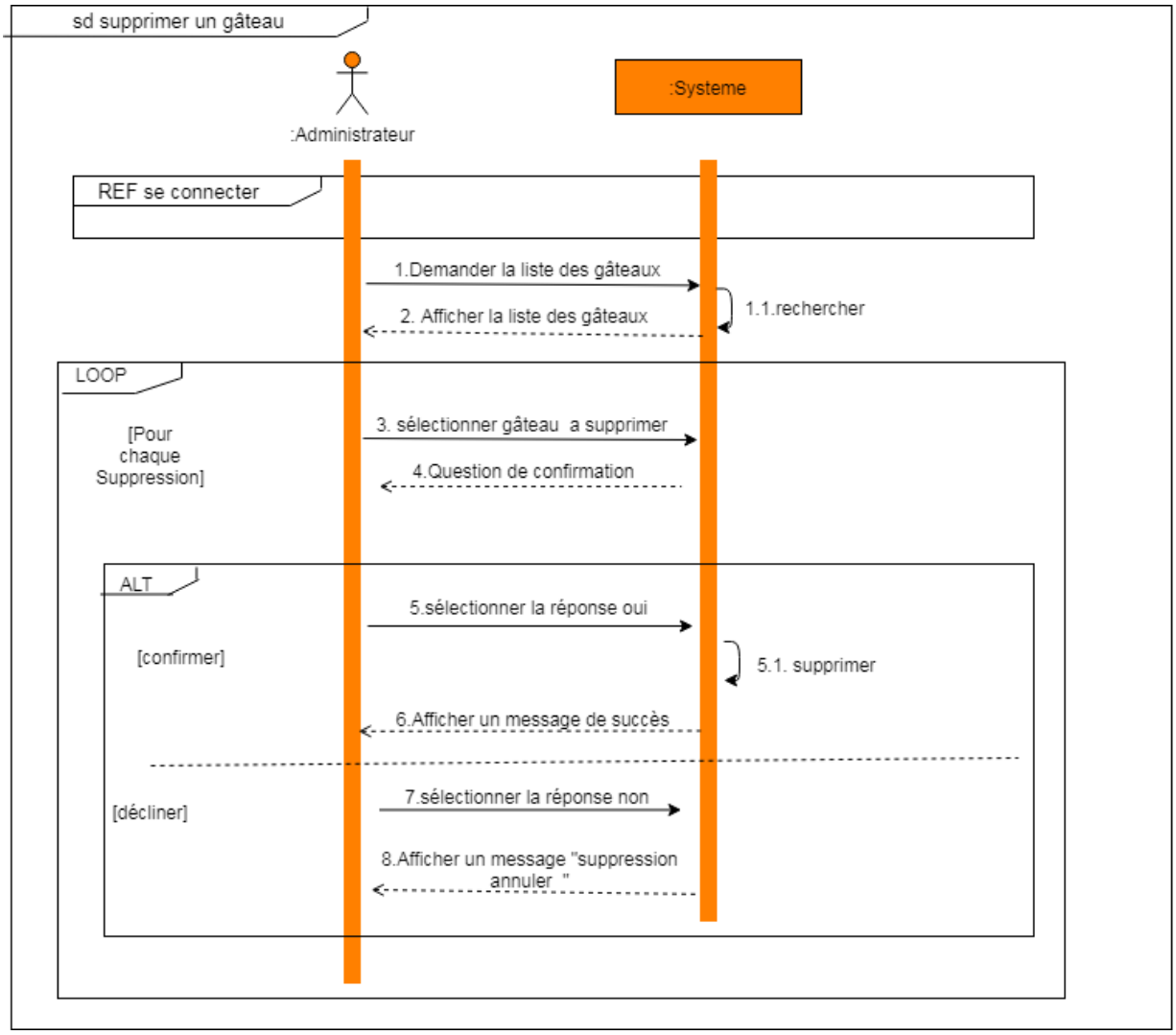


Figure II. 8 - Diagramme de Séquence Système « Supprimer un gâteau ».

Diagrammes de séquence le cas d'utilisation modifier un gâteau

La figure suivante (Figure III.9) montre l'interaction entre l'acteur Administrateur et le système afin de réaliser l'opération « Modifier un gâteau ».

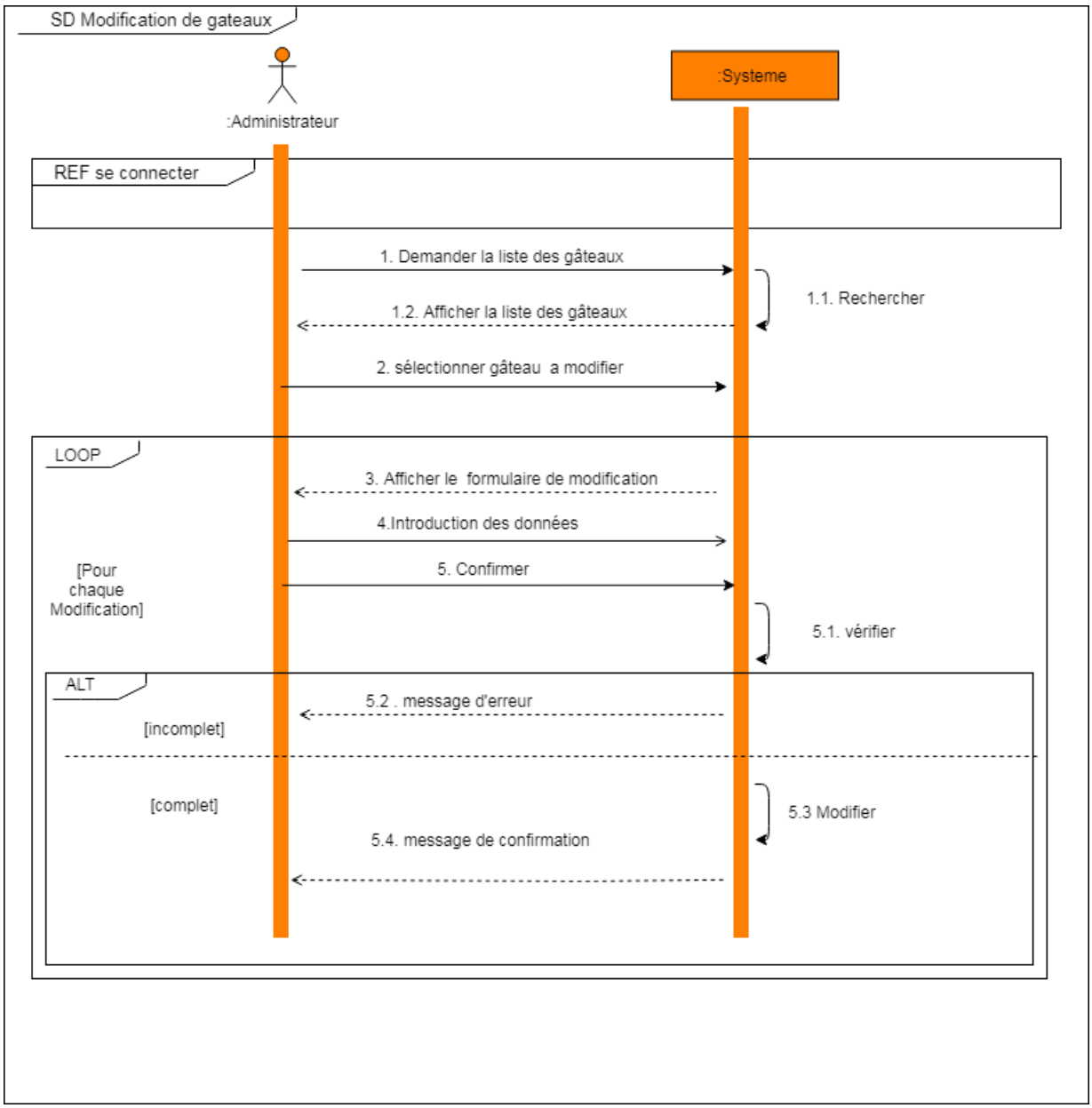


Figure II. 9 - Diagramme de Séquence Système « Modifier un gâteau ».

II.6 Conclusion

Ce chapitre nous permis de détailler les spécifications du projet, d'identifier les acteurs et leurs cas d'utilisation et d'élaborer les diagrammes correspondants. Ces derniers ont été décrits textuelle et par diagramme de séquences. A présent nous sommes prêts à passer à l'étape de conception dans le chapitre suivant.

Chapitre III

Conception et elaboration de Schema Relationnel

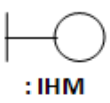
III.1 Introduction

Après avoir tracé les grandes lignes de phase de spécification de besoin mettons l'accent maintenant sur une phase fondamentale dans le cycle de vie de logiciel, la phase de conception. Cette phase a pour objectif de déduire la spécification de l'architecture de système. En tant que concepteurs nous allons élaborer le modèle de conception qui va donner une image prête à coder de notre solution.

III.2 Diagramme de séquence d'interaction

Pour chaque diagramme de séquence système, nous établirons le diagramme de séquence d'interaction, dans lequel le système est remplacé par les objets qui interviennent pour réaliser le cas d'utilisation concerné. Pour ce type de diagramme, nous avons trois types de classes (voir **[07]** *page 171*) :

Classes d'interface (boundary) : des classes qui permettent l'interaction entre l'application et ses utilisateurs. Pour chaque cas d'utilisation, il y a au moins une classe d'interface. Ce type de classe est schématisé comme suit :



Classes de Contrôle (Control) : Ce sont des classes qui contiennent les traitements et la cinématique de l'application. Elles font la transition entre les classes d'interface et les classes entité. Elles sont schématisées comme suit :

Classes entité (entity) : Elles représentent les objets métiers, et ce sont très souvent des entités persistantes, c'est-à-dire qui vont garder leurs informations (données) après l'exécution d'un cas d'utilisation particulier. En général, elles sont enregistrées dans une base de données. Leur schématisation se fait grâce à ce stéréotype :

III.2.1 Diagramme d'interaction pour le cas d'utilisation « Se Connecter »

Ce diagramme représente le diagramme de séquence détaillée cas d'utilisation « Se Connecter »

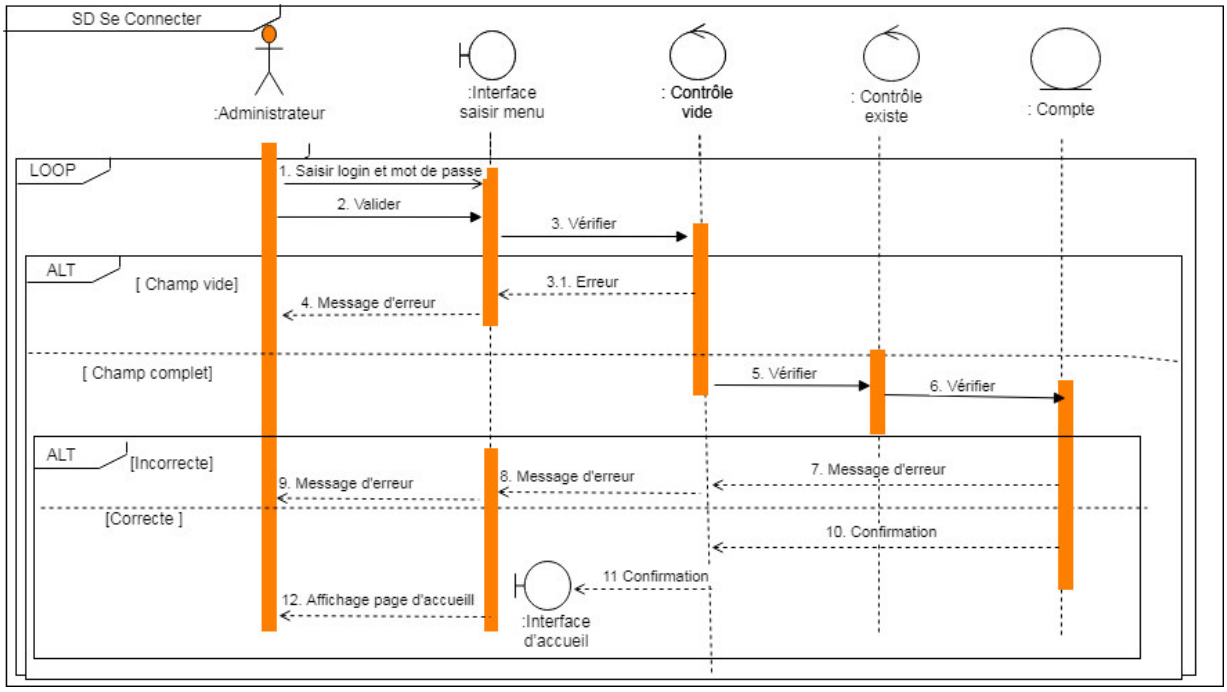


Figure III. 1 - Diagramme interaction « Se Connecter ».

III.2.2 Diagramme d'interaction pour le cas d'utilisation « Ajouter un gâteau »

Ce diagramme représente le diagramme de séquence détaillée cas d'utilisation « Ajouter un gâteau »

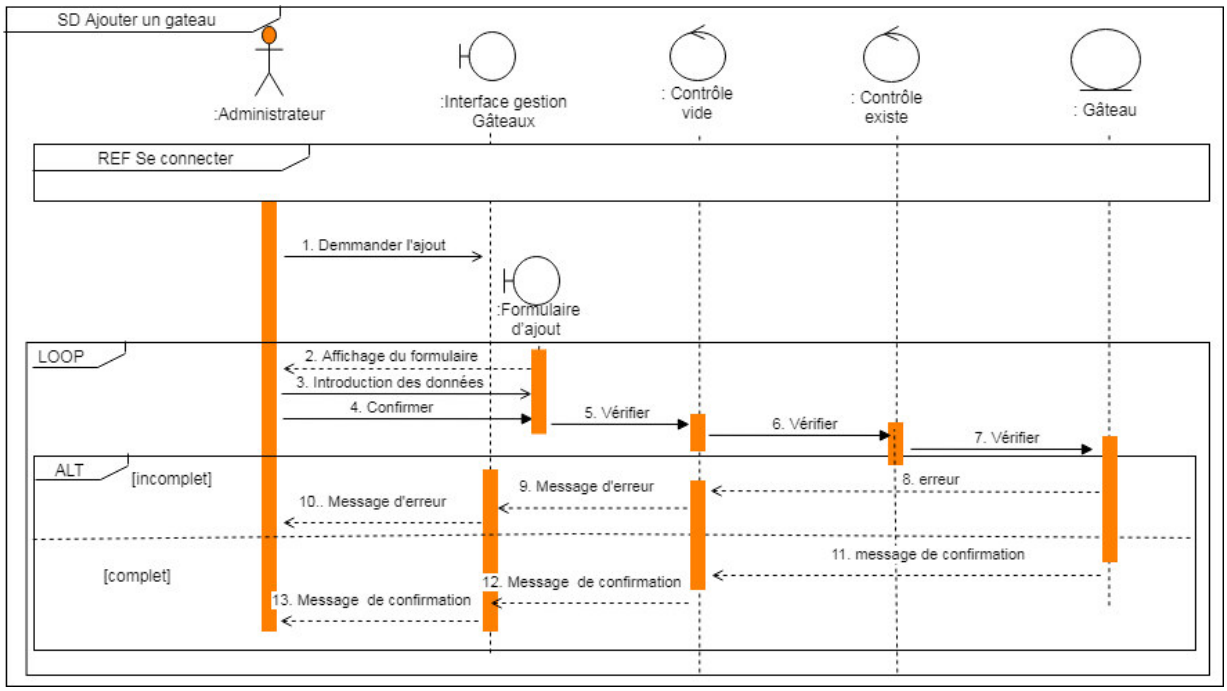


Figure III. 2 - Diagramme interaction « Ajouter un gâteau ».

III.3 Diagramme de Classes du Domaine

III.3.1 Diagramme de classe

Le diagramme de classes est considéré comme le plus important de la modélisation orientée objet, il est le seul obligatoire lors d'une telle modélisation.

Il s'agit d'une vue statique, car on ne tient pas compte du facteur temporel dans le comportement du système. Le diagramme de classes modélise les concepts du domaine d'application ainsi que les concepts internes créés de toutes pièces dans le cadre de

l'implémentation d'une application. Chaque langage de Programmation orienté objet donne un moyen spécifique d'implémenter le paradigme objet (pointeurs ou pas, héritage multiple ou pas, etc.), mais le diagramme de classes permet de modéliser les classes du système et leurs relations indépendamment d'un langage de programmation particulier.

Les principaux éléments de cette vue statique sont les classes et leurs relations : association, généralisation et plusieurs types de dépendances, telles que la réalisation et l'utilisation. [10].

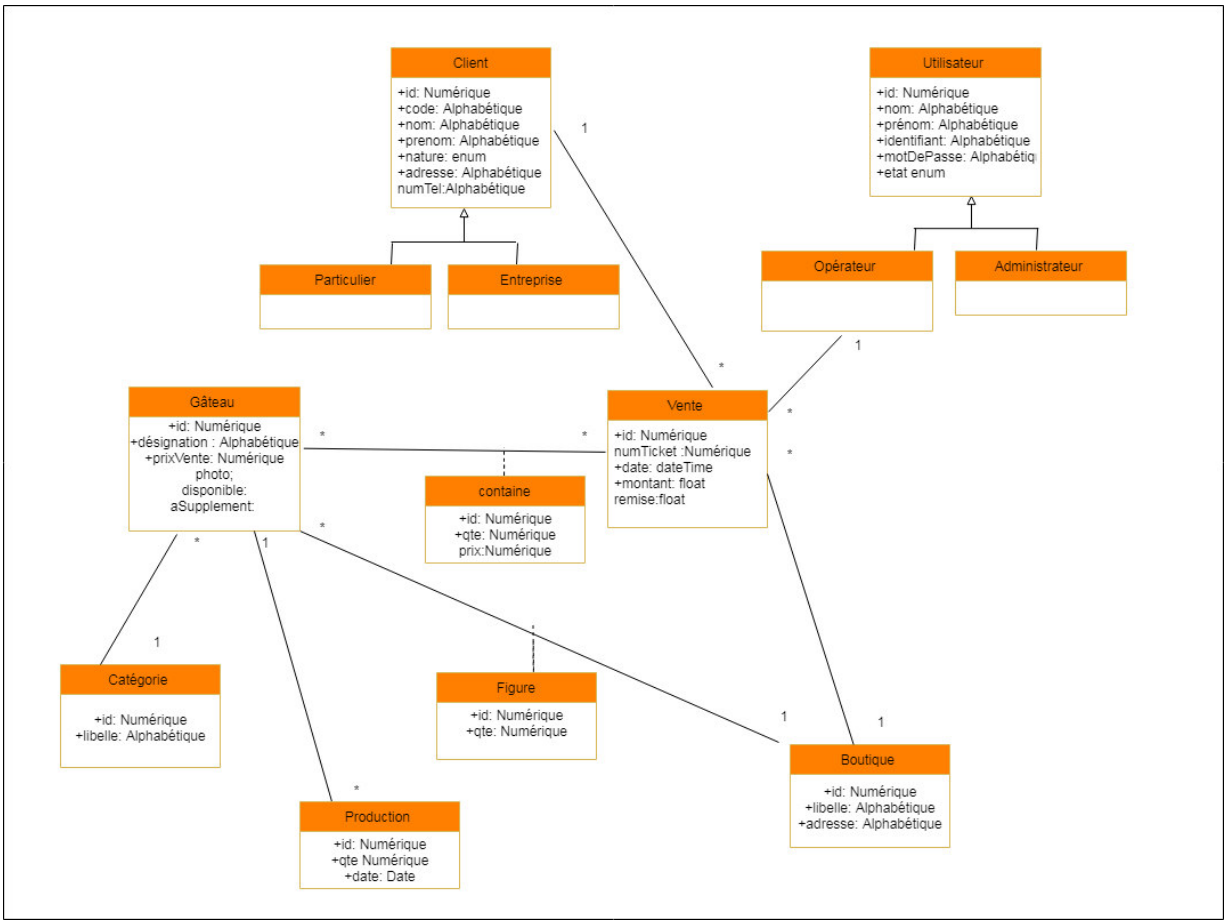


Figure III. 3 - Diagramme de classe de domaine.

III.3.2 Classes, attributs et responsabilités

Dans ce qui suit, nous allons décrire les différentes classes schématisées dans la figure III.3. Cette description sera présentée sous forme d'un tableau, comme suit :

Classe	**Responsabilité**	**Attributs**		
		Désignation	**Définition**	**Type**
Client	Représente les clients de la boutique .	**id**	Identifiant	Numérique
		code	Code du client	Alphanumérique
		nom	nom du client	
		prénom	prénom du client	
		adresse	Adresse de client	Alphanumérique
		numTel	N° du Téléphone	Numérique.
Utilisateur	Représente les Utilisateurs de l'application	**id**	Identifiant.	Numérique.
		nom	Nom de utilisateur	Alphabétique.
		Prénom	Prenom de utilisateur	Alphanumérique.
		Identifiant	L'identifiant d'accès	Chaine de caractère.
		Etat	Description de l'état de utilisateur (suspendu ,actif)	Enuméré.
		mot Depasse	Mot de passe d'accès	Date
Vente	Représente les ventes	id	Identifiant	Numérique
		numTicket	Numéro de ticket de client	Numérique
		date	La date de vente	Date

		montant	Montant totale de la vente	Numérique
		remise	Remise de vente	Numérique
Gâteaux	Représente les gâteaux	**id**	Identifiant.	Numérique.
		designation	Désignation du gâteaux	Alphanumérique.
		prixVente	Prix de vente de gâteaux	Numérique.
		photo	Photo de gâteau	
		disponible	Disponibilité de gâteaux	Alphanumérique.
Production	Représente les production	Id	Identifiant	Numérique
		Qte	Quantités de production	Numérique
		Date	Date de production	Date
Boutique	Représente les boutiques	Id	Identifiants	Numérique
		Libelle	Libelle de la boutique	Alphanumérique.
		Adresse	Adresse de la boutique	Alphanumérique.
Comptoir	Représente les comptoir	Id	Identifiant	Numérique
Catégorie	Représente les catégories	Id	Identifiant	Numérique
		Libelle	Libelle de la catégorie	Alphanumérique.
Containe		Id	Identifiant	Numérique
		Qte	Quantité de gâteaux vendue	Numérique
		Prix	Prix de la vente de gâteaux	Numérique
Figure		Id	Identifiant	Numérique
		Qte	Quantité de gâteau dans un comptoir	Numérique

Tableau III. 1 - Description des classes et leurs attributs.

A partir d'un diagramme de classe, nous obtiendrons le schéma relationnel (modèle logique de données), et ceci en appliquant les règles de passage suivantes :

Règle 01 : Chaque classe du diagramme de classe devient une relation.

Règle 02 : Association Unique-Multiple (1/0..1 - *), dans ce cas, nous ajouter une clé étrangère dans la relation issue de la classe qui participe une seul fois dans l'association.

Règle 03 : Association Multiple-Multiple (* - *), l'association sera transformée à une relation, dont la clé primaire est la concaténation des clés primaires des associations issues des classes qui participent dans cette association.

Règle 04 : Héritage, ici, nous avons trois méthodes pour transformer l'héritage :

- la méthode ascendante : Supprimer les relations issues des sous-classes et faire migrer tous les attributs dans la relation issue de la classe mère et ajouter un nouveau attribut qui détermine le type d'une instance de cette relation ;
- la méthode descendante : Supprimer la relation issue de la superclasse et faire migrer ses attributs vers les relations issues des sous-classes ;
- la méthode distincte : ne pas supprimer aucune relation, ni celle issue de la superclasse, ni celles issues des sous-classes, et ajouter une clé étrangère dans les relations issues des sous classes vers la clé primaire de la relation issue de la superclasse.

Après l'application des règles ci-dessus citées, nous obtiendrons le schéma relationnel suivant :

III.4 Schéma relationnel

Client(**idClient**, code, nom, nature, adresse , numTel)

Utilisateur (**idUtilisateur**, nom, prenom, identifiant, mot_de_passe, etat, role)

Boutique(**idBoutique**, libelle, adresse)

Gâteaux (**idGateau**, designation, prixVente, photo, disponible, #idCategorie)

Vente (idVente, numTicket, date, montant, remise, #idboutique, #idClient, #idUtilisateur)

Contain(#idVente, #idGateaux, qte, prix)

Figure (**#idBoutique, #idGateaux**, qte)

Production (**idProduction**, qte,date ,#idgateau)

Categorie (**idCategorie**, libelle)

III.5 Conclusion

Dans ce chapitre nous avons présenté les diagrammes UML nécessaires à la phase de conception, ce qui nous permet d'entamer le développement en ayant une idée claire et détaillée sur l'aspect statique et dynamique de la solution. Il reste à coder ce qu'on a conçu. Le chapitre suivant présentera l'environnement de travail à savoir les outils et les méthodes utilisés lors du développement.

Chapitre IV

Realisation et Test

IV.1 Introduction

A ce stade du processus, les cas d'utilisation sont bien cernés, le problème a été analysé, et nous avons défini une conception appropriée aux besoins de l'application. Nous pouvons alors entreprendre l'implémentation.

Premièrement dans ce chapitre, nous illustrerons comment nous aurons pu déployer, valider notre application, ensuite, nous survolerons quelques langages que nous avons utilisé pour la réalisation. Ensuite, nous allons nous intéresser à l'IHM et aux interfaces que nous aurons pu réaliser.

IV.2 Environnement de programmation et bibliothèques

IV.2.1 SEMANTIC UI

Semantic-ui sont des groupes de CSS, de polices, d'images et de JavaScript qui constituent un seul élément. Contrairement aux autres bibliothèques JavaScript, les éléments sémantiques de l'interface utilisateur sont autonomes et ne nécessitent que leurs propres ressources pour fonctionner correctement. [11]

IV.2.2 Html

HTML signifie « *HyperText Markup Language* » qu'on peut traduire par « langage de balises pour l'hypertexte ». Il est utilisé afin de créer et de représenter le contenu d'une page web et sa structure [15]

IV.2.3 CSS

Les CSS (*Cascading Style Sheets* en anglais, ou « feuilles de style en cascade ») sont le code utilisé pour mettre en forme une page web [16]

IV.2.4 Javascript

IV.2.4.1 JavaScripts cote client

JavaScript est un langage de script ou de programmation qui permet d'implémenter des programmes complexes sur les pages web, ces scripts peuvent être écrits directement sur une page web HTML et exécutés automatiquement quand la page se charge.

- **JavaScript (ES6)**

ES6 fait référence à la version 6 du langage de programmation ECMA Script. ECMA Script est le nom standardisé de JavaScript, et la version 6 est la prochaine version après la version 5, qui a été publiée en 2011. Il s'agit d'une amélioration majeure du langage JavaScript et ajoute de nombreuses autres fonctionnalités destinées à faciliter le développement de logiciels à grande échelle. [12]

IV.2.4.2 JavaScript côté serveur

JavaScript a connu son essor dans le navigateur. Cependant, il ne s'agit que d'un contexte, qui vous dit ce que vous pouvez faire avec le langage, mais pas ce que le

langage lui-même peut faire. JavaScript est un langage complet ; il peut être utilisé dans différents contextes et peut réaliser les mêmes choses que tout autre langage.

- **Node JS**

Node.js est un environnement open-source permettant de développer des applications multiplateformes. Basé sur le V8 de Google, le moteur d'exécution JavaScript utilisé dans Chrome, Node.js traite donc les données en JavaScript.

Node.js vous permet d'utiliser JavaScript côté serveur, en dehors du navigateur avant de pouvoir exécuter votre JavaScript sur le serveur, le code a besoin d'être interprété. C'est précisément ce que fait Node.js en utilisant le moteur V8 de Google, celui qui est utilisé par le navigateur Chrome. D'autre part, Node.js intègre un nombre important de modules utiles, ainsi, vous n'avez pas à écrire tout le code à partir de zéro Ainsi, Node.js comprend deux éléments distincts : un contexte d'exécution et une bibliothèque.[13]

- **Node Package Manager (NPM)**

NPM (Node Package Manager) comme son nom l'indique est le "package manager" officiel de l'univers JavaScript (frontend / backend). Il est installé automatiquement lors de l'installation de NodeJS[16]

IV.2.5 JQuery

jQuery, est une bibliothèque JavaScript gratuite, libre et multiplateforme. Compatible avec l'ensemble des navigateurs Web (Internet Explorer, Safari, Chrome, Firefox, etc.), elle a été conçue et développée en 2006 pour faciliter l'écriture de scripts. Il s'agit du framework JavaScript le plus connu et le plus utilisé. Il permet d'agir sur les codes HTML, CSS, JavaScript et AJAX et s'exécute essentiellement côté client. [15]

IV.2.5 JSON

JSON (JavaScript Object notation) : un format léger qui permet de décrire des objets en JavaScript, il est principalement utilisé pour faire des échanges entre JS et un serveur web, JSON se base sur deux structures :

Une collection de couple nom/valeur : {nom : valeur ;}

Un tableau qui est une collection de valeurs ordonnées [valeur, valeur]

JSON est un format de données. Autrement dit, c'est une façon de stocker des informations, un peu comme une base de données, il est maintenant lié à JavaScript qui inclut un objet JSON, et de nombreux développeurs l'incorporent quasiment comme un sous-ensemble du langage. [16]

IV.2.6 AJAX

Le terme AJAX désigne une technologie qui s'est popularisée dans le domaine de la création de sites internet. Elle est principalement utilisée pour apporter de l'interactivité au sein des pages d'un site web tout en économisant les ressources serveur.

En effet, AJAX permet de communiquer avec le serveur à l'aide de code JavaScript en arrière-plan pendant que la page est affichée à l'écran. Ainsi le contenu de la page peut être modifié sans qu'il soit nécessaire de faire transiter et afficher la page en entier.[15]

IV.2.7 ELECTRON JS

Electron Js est un module Node.js qui permet de créer des applications (logiciels) desktop cross-plateforme (Win, Mac, Linux) en utilisant des technologies web telles que :

HTML5, CSS et JavaScript côté client pour l'interface graphique du Logiciel (front-end)

Node.js pour la partie back-end (opérations de système, etc.)

Electron est un projet open-source maintenu par GitHub.

Le module Electron n'est pas compatible avec Windows XP et Windows Vista, parce qu'il appele les touch APIs specifiques qui ne sont pas presentent dans ces versions de Windows. Electron est compatible avec les versions Windows 7+. [17]

- **Avantages de Electrons JS**

Il repose entièrement sur des normes Web que presque tous les développeurs Web connaissent déjà, ce qui leur permet d'écrire des logiciels de bureau.

Pour les applications de bureau, il fournit diverses fonctionnalités de base telles que la mise à jour automatique, le reporter de crash, le créateur du programme d'installation et des fonctionnalités spécifiques au système

- **Inconvénients**

Aucun MVC intégré n'est fourni par Electron et les plates-formes pour Chrome ne sont pas encore entièrement prises en charge

IV.3 Serveur d'application

IV.3.1 WAMP SERVER

WampServer est une plate-forme de développement Web sous Windows pour des applications Web dynamiques à l'aide du serveur Apache2, du langage de scripts PHP et

d'une base de données MySQL. Il possède également PHPMyAdmin pour gérer plus facilement vos bases de données.[18]

IV.3.2 APACHE

Apache est un logiciel de serveur web gratuit et open-source qui **alimente environ 46% des sites web** à travers le monde. Le nom officiel est **Serveur Apache HTTP** et il est maintenu et développé par Apache Software Fondation .Il permet aux propriétaires de sites web de servir du contenu sur le web – d'où le nom « serveur web » -. C'est l'un des serveurs web les plus anciens et les plus fiables avec une première version sortie il y a plus de 20 ans, en 1995. [18]

IV.3.3 MySQL

Le terme **MySQL**, pour My Structured Query Language, désigne un serveur de base de données distribué sous licence libre GNU (General Public License). Dans la pratique, le serveur **MySQL** peut se résumer à un lieu de stockage et d'enregistrement des données, que celles-ci soient ou non cryptées. [19]

IV.3.4 SQL

Le langage **SQL** (Structured Query Language) est un langage informatique utilisé pour exploiter des bases de données. Il permet de façon générale la définition, la manipulation et le contrôle de sécurité de données.

Dans la pratique, le langage **SQL** est utilisé pour créer des tables, ajouter des enregistrements sous forme de lignes, interroger une base de données, la mettre à jour, ou encore gérer les droits d'utilisateurs de cette base de données. Il est bien supporté par la très grande majorité des systèmes de gestion de base de données (SGBD).[19]

IV.4 Schéma Physique de la Base de donnée

La base de données a été implantée en utilisant l'application Web PhpMyAdmin (outil intégré dans wamp server), ce dernier nous permet de schématiser les tables et leurs relations (clé étrangère) comme indiqué à la *figure IV.1.*

Dans le schéma physique, nous indiquons les types des attributs de chaque table, les clés primaires, les clés étrangères ainsi que les champs références

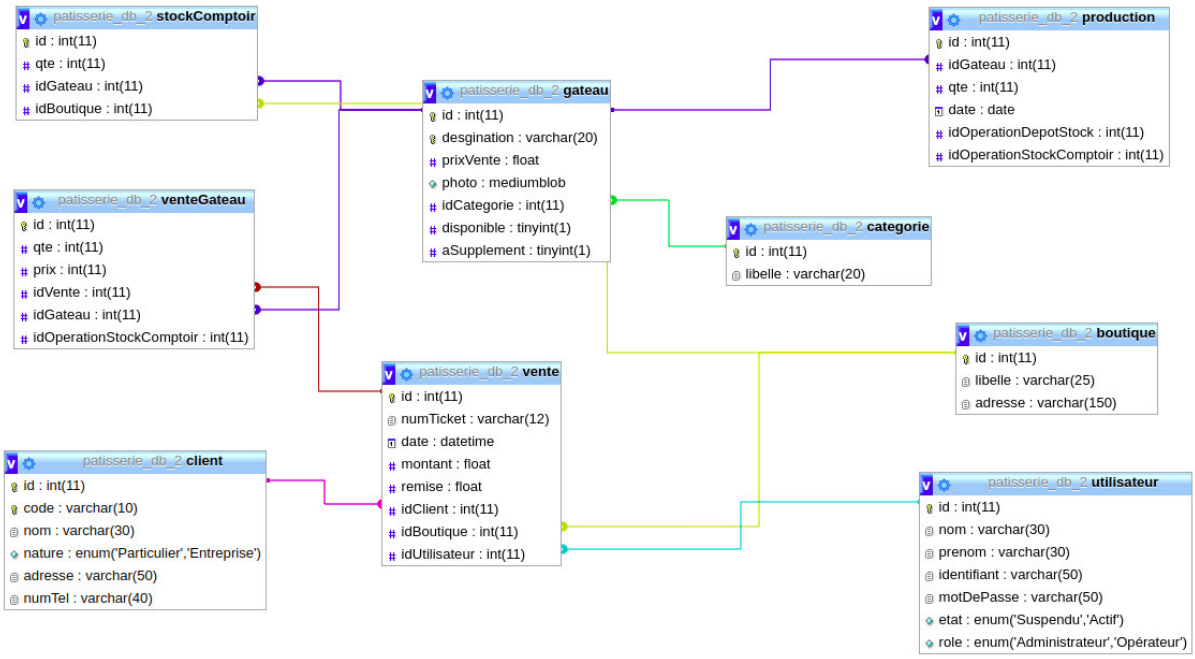


Figure IV. 1 - Schéma Physique de la Base de données.

IV.5 Architecture de l'Application

Après avoir présenté la partie de données de notre application, nous allons expliquer l'architecture de l'Application en termes de déploiement des modules qui la constituent sur les différentes machines (Serveur, Ordinateur de travail). En plus de ça, nous présentons la structuration de notre code source (Partie client et serveur).

IV.6 Diagramme de Déploiement

La figure suivante illustre les composants de notre application leur exécution par les différentes machines :

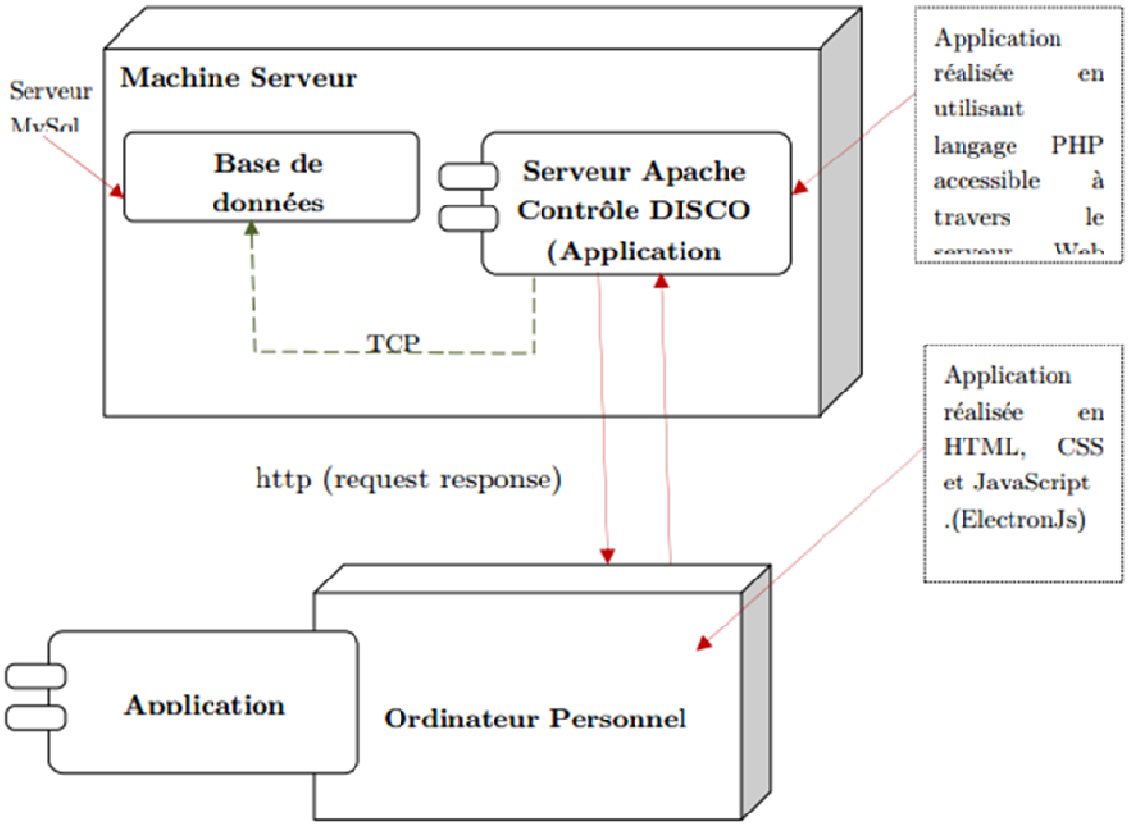


Figure IV. 2 - Diagramme de Déploiement.

IV.7 Structuration du Code Source

Dans cette partie, nous allons discuter de l'architecture de notre application en termes de structure de code source. Ceci permet d'ajouter un éclaircissement sur le diagramme de déploiement ci-dessus présenté.

Notre application est basée sur l'architecture Client/serveur à trois niveaux. C'est niveaux sont comme suit :

Niveau 1 – font end de application (les interfaces de application).

Niveau 2 – Serveur Web Apache.

Niveau 3 – Serveur base de donnée, pour notre cas, nous avons utilisé MySQL.

La figure suivante, présente sommairement la structuration de notre code source :

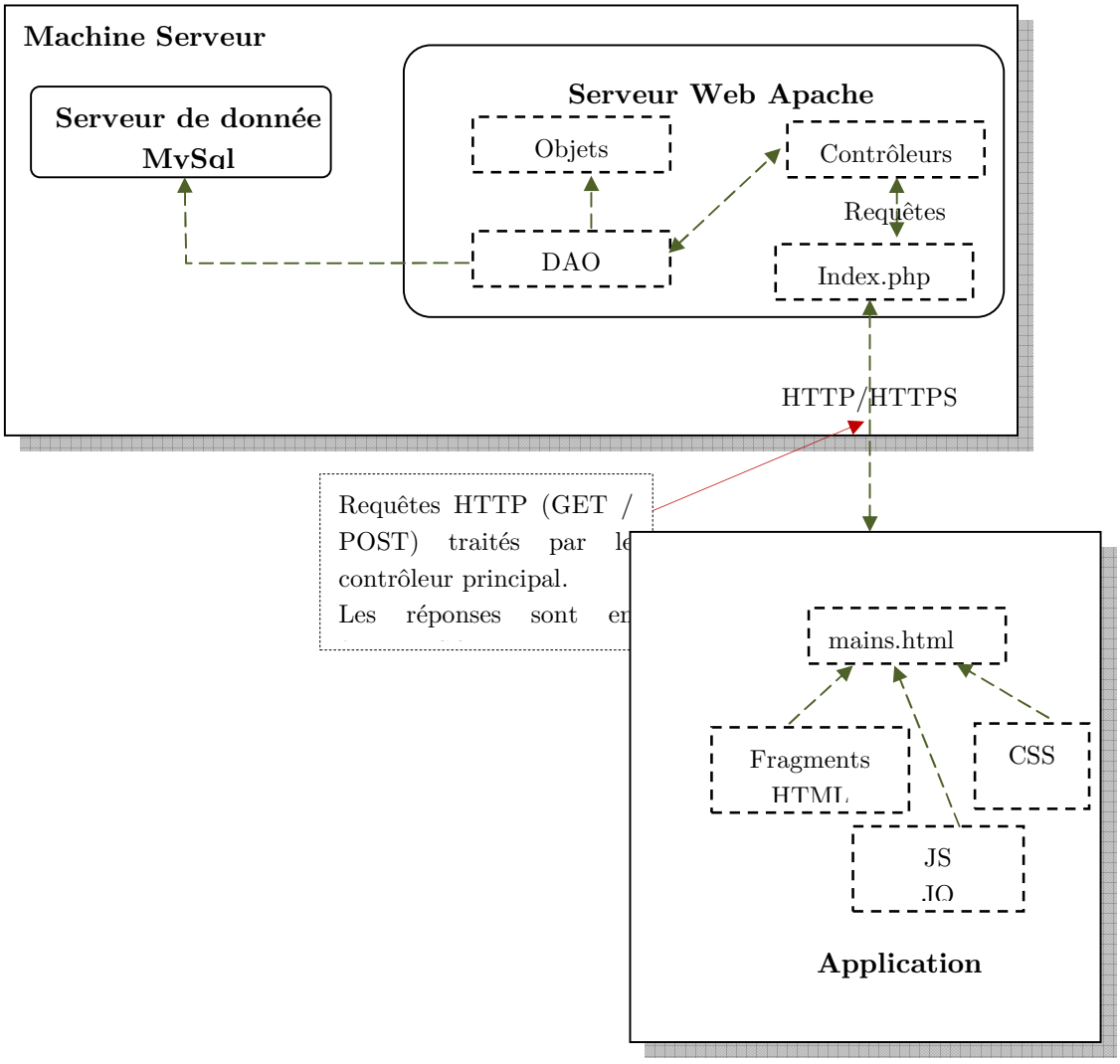


Figure IV. 3 - Structure de code source.

IV.8 Interfaces de l'application

Dans cette section, quelques vues de l'interface Homme-Machine de notre application seront illustrées et expliquées.

- Interface d'authentification

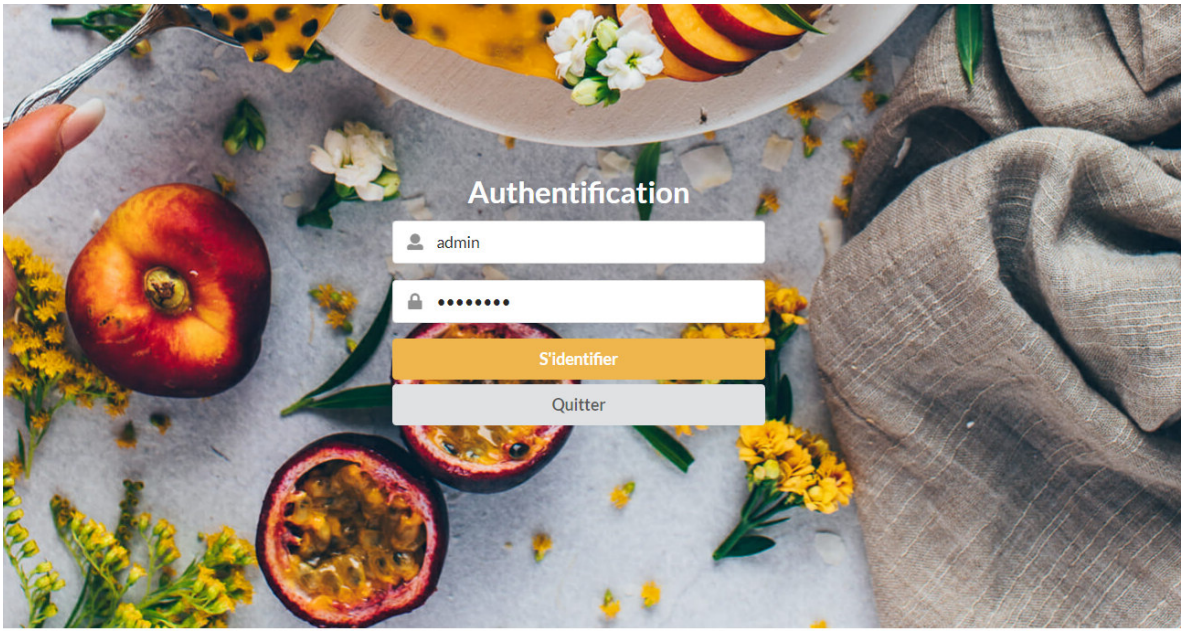


Figure IV. 4 - Formulaire d'Authentification.

➢ Interface d'accueil pour l'administrateur

La figure ci-dessus montre la première interface qui sera affichée pour l'administrateur. Une fois ce dernier s'authentifie, il sera redirigé vers interface accueil.

Le clic sur le bouton du Menu affichera ceci :

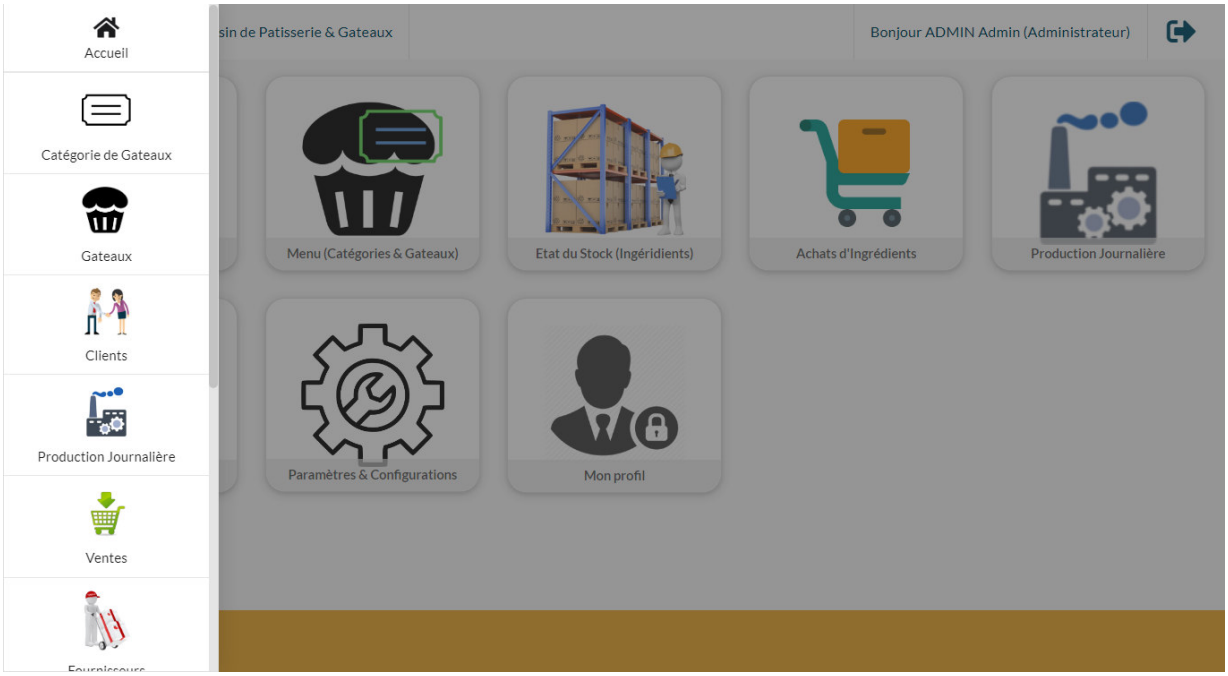


Figure IV. 5 - Le menu de l'administrateur.

➢ Interface gestion catégories

Le click sur une catégorie permet d'afficher l'interface suivante :

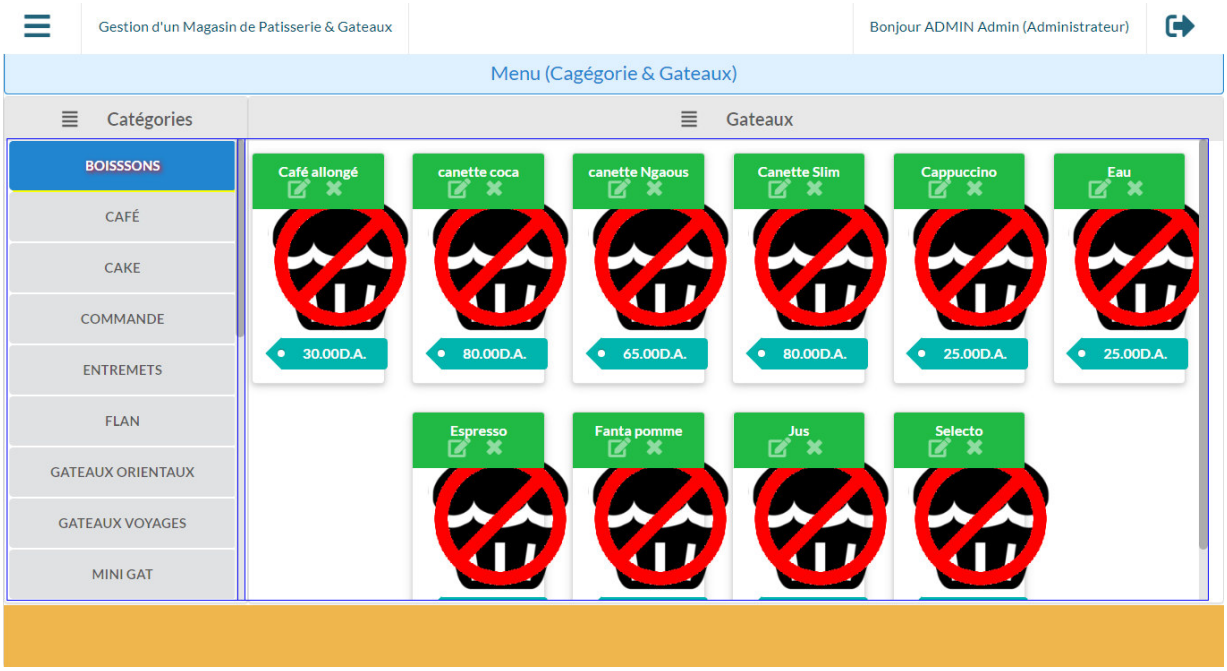


Figure IV. 6 - Interface de gestion des catégories.

➢ Interface ajouter une catégorie

Pour ajouter une catégorie, l'administrateur doit cliquer sur le bouton menu une liste déroulante apparaisse il choisis ajouter catégorie la figure suivante montre l'interface adéquate.

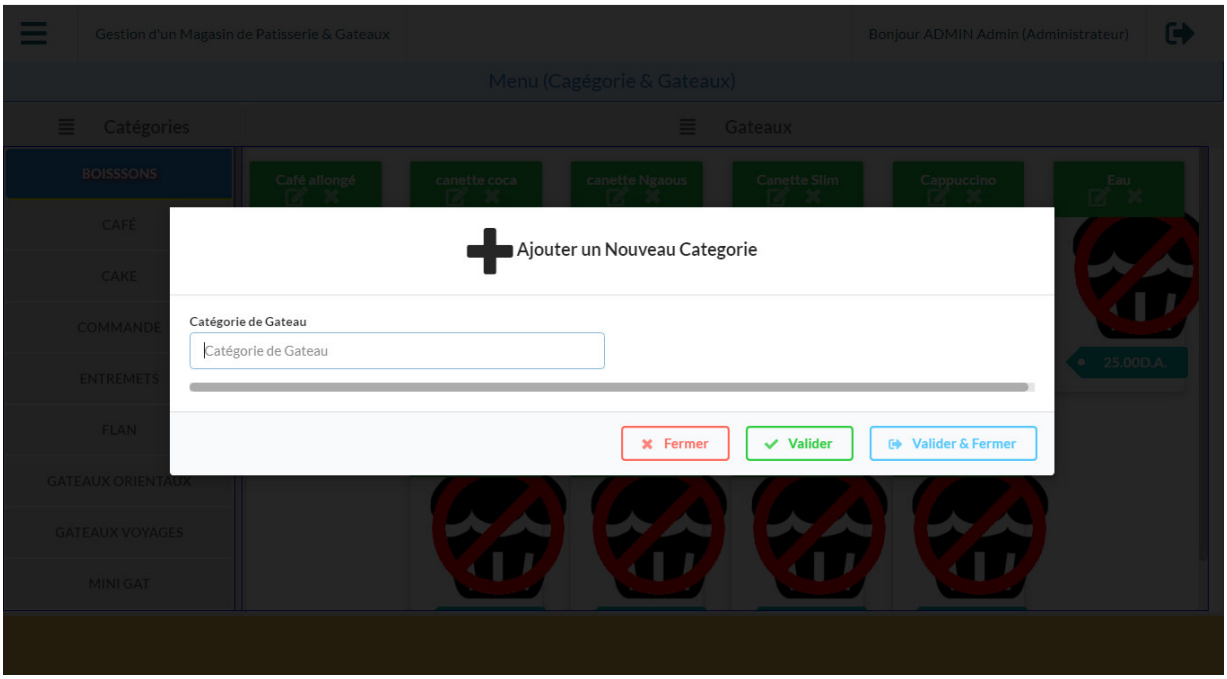


Figure IV. 7 - Ajouter une catégorie.

➢ Interface gestion vente

Cette figure illustre l'interface de gestion de vente

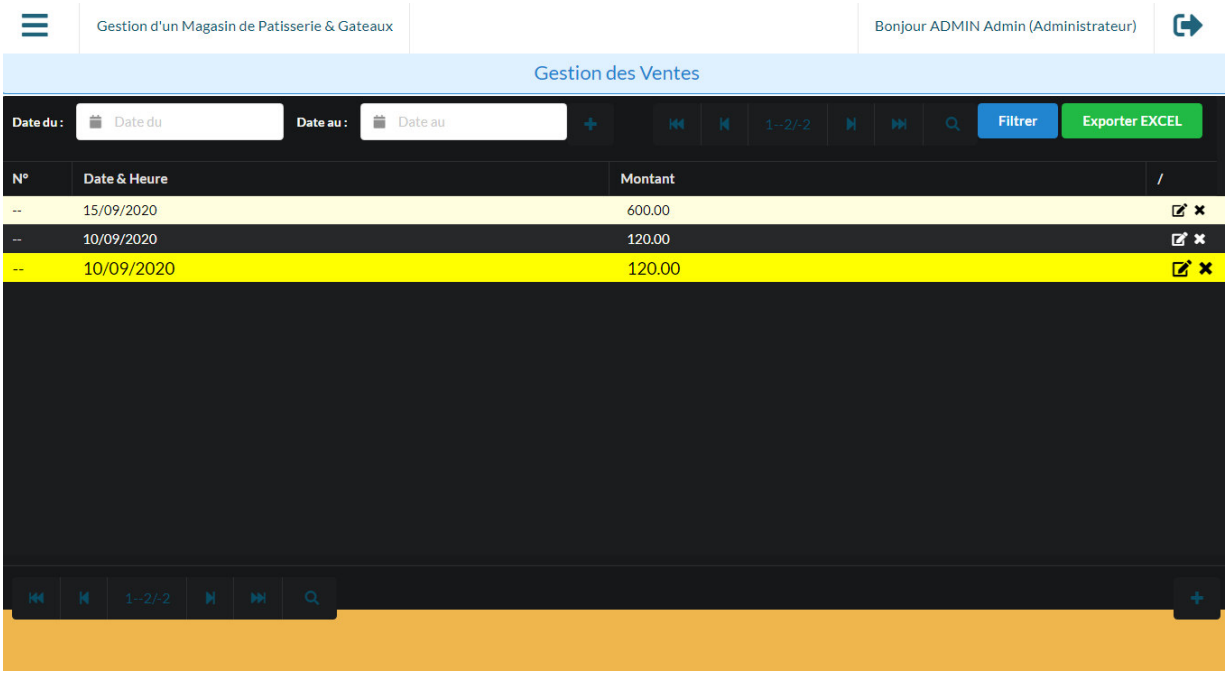


Figure IV. 8 - Interface de gestion de vente.

➢ Interface d'accueil pour l'opérateur

La figure ci-dessus montre la première interface qui sera affichée pour l'operateur. Une fois ce dernier s'authentifie, il sera redirigé vers l'interface gestion vente:

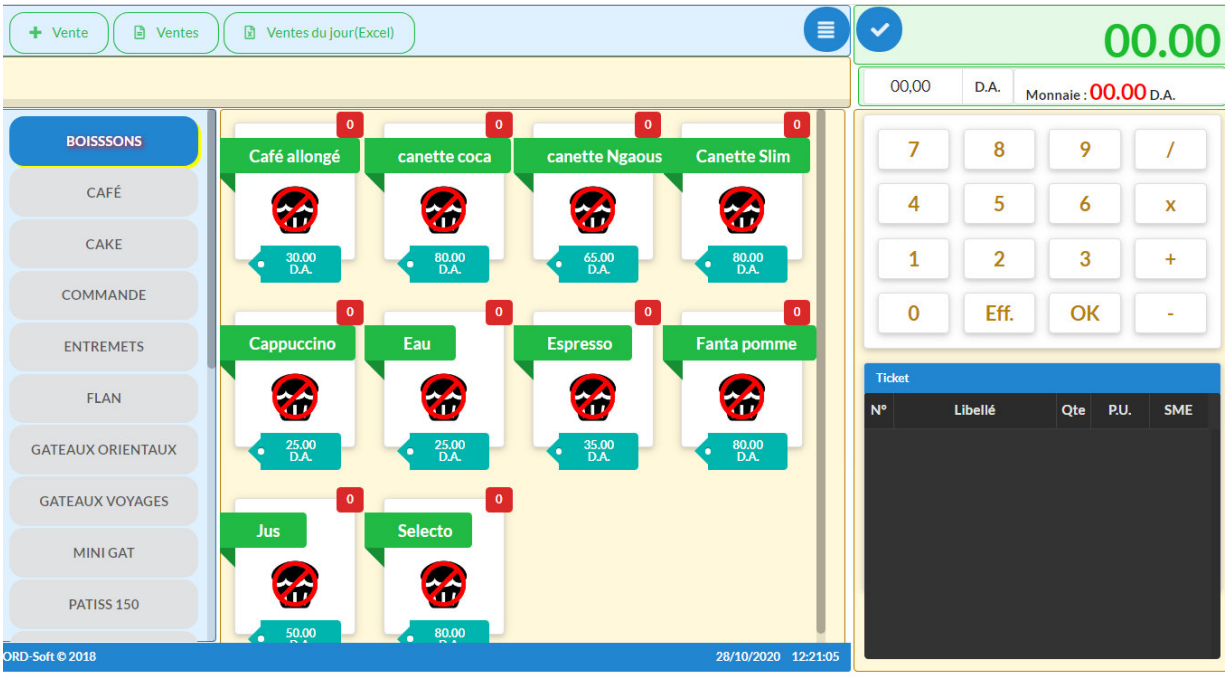


Figure IV. 9 - La page d'accueil de l'operateur.

IV.9 Conclusion

Dans ce dernier chapitre, qui représente la phase finale du processus de développement logiciel, nous avons présenté les différents outils et langages informatiques utilisés pour développer notre application. Par la suite, nous avons montré la structure physique de notre base de données, en montrant les tables et leurs interrelations. Par la suite, le diagramme de déploiement a été présenté pour bien

illustrer les différentes parties de notre application et leur affectation à différentes machines (Serveur, ordinateur personnel et téléphone intelligent).

Nous avons clôturé ce chapitre par la présentation de quelques interfaces des utilisateurs. En ce qui concerne la phase des tests, le travail est toujours en cours de développements et attente d'une validation de la part de l'organisme d'accueil.

Conclusion Generale

Le domaine de développement des applications, d'une façon générale, et les applications de gestion d'une façon particulière, ne cesse d'évoluer et de se développer pour mettre à disposition des réalisateurs de logiciels des technologies et outils qui facilitent et améliore la qualité de leurs travails.

Parmi ces évolution, le faite de pourvoir utiliser le langage JavaScript, qui gagne de plus en plus de puissances et de communauté, dans différents types de programmation : WEB, Applications lourdes et aussi le développement Mobile. Ceci, est dû à l'apparition, en 2009, de Node JS, qui représente un environnement d'exécution de code écrit en JS (JavaScript).

Dans ce contexte d'évolution du lange JS, nous avons utilisé le Framework ElectronJS afin de développer une application BUREAU de gestion de pâtisserie. Ce framework nous a permet de réalisé la partie frontend (Interface) de l'application en se basant sur les technologies WEB : HTML, CSS et JS. En ce qui concerne la partie backend, nous avons utilisé le langage PHP qui interagit facilement avec notre base de données implémenté sous le SGBD MySQL.

À l'issue de ce projet, notre application est réalisée dans sa première version. Elle restera toujours ouverte à des perspectives d'amélioration. Nous envisageons de prendre

en charge la gestion de stock qui permet le suivi de la matière première et des produits fini (pièce de gâteau), et d'ajouter de nouvelles fonctionnalités selon les recommandations des utilisateurs.

Bibliographie

[1] Mikael BARON . '*developer une application web avec Vue.js'.* page web : URL :https://mbaron.developpez.com/tutoriels/vuejs/developper-application-web-vuejs-vuecli-generalites/#LI . Consulter le 23/06/2020.

[2] Victor Osetskyi, '*Meilleurs cadres pour le développement d'applications de bureau'* . page web : URL : https://dzone.com/articles/best-frameworks-for-desktop-application-development, Consulter le 23/06/2020

[3] Jean-François pilou. '*Application de base de données,principe et exemple'* page web : URL :https://stph.scenari-community.org/bdd/lap2/co/webUC002archi.html. Consulter le 10/06/2020

[4] H.S. Oluwatosin. *Client-server model.* IOSR Journal of Computer Engineering (IOSR-JCE), VOL. 16, Issue 1, pp. 2278-8727. Feb 2014.

[5] Rémi leblond. '*vers une architecture n-tiers'.* Oral probatoire 2002

[6]Frencoiscorbizier. '*architecture web'.* Page web : URL : https://www2.ulb.ac.be/cours/acohen/travaux_2004_infodoc/projettechnologiesweb/html/architecturesweb.html#:~:text=L%27architecture%20de%20base%20pour,Enterprise%20Server%20...).&text=L%27utilisateur%20%C3%A9met%20une%20requ%C3%AAte,est%20stock%C3%A9e%20la%20page%20HTML. 2006. Consulter le 11/07/2020

[7] Roy GILLES. *'UML2 en action, de l'analyse des besoins en action'.* Presses de l'Université de Québec, 2009, première édition.

[8] The Apache Software Foundation. '*Apache Tomcat 7 User Guide'. Corporation*, 2011.

[9]Joseph Gabay, David Gabay. '*UML2 Analyse et conception'.* DUNOD, 2008. 1ere edition.

[10] Laurent Audibert. '*UML2 De l'apprentissage à la pratique'.* Site web : URL :https://laurent-audibert.developpez.com/Cours-UML/?page=diagramme-classes . 2003. Consulter le 2/06/2020.

[11] 'Semantic-UI Guide | Seùantic-UI'. site web : URL : https://legacy.semantic-ui.com/introduction/definitions.html. Consulter le 12/08/2020.

[12] Jay Sridhar. '*whatjavascriptprogrammersneedstooknows'.* page web : URL : https://www.makeuseof.com/tag/es6-javascript-programmers-need-know/#:~:text=ES6%20refers%20to%20version%206,Script%20(Javascript)%20programming%20language.&text=It%20is%20a%20major%20enhancement,large%2Dscale%20software%20development%20easier.2017. consulter le 10/06/2020.

[13] 'Node.js Guide | Node.js'. site web : URL : https://nodejs.org . consulter le 22/08/2020

[14]Younes JAAIDI. '*Le guide angular'.* page web : URL : https://guide-angular.wishtack.io/tools/npm#:~:text=NPM%20permet%20entre%20autres%20%3A,fichier%20de%20description%20de%20d%C3%A9pendances. Consulter le 3/09/2020.

[15]Bouchebbah Dina et al. '*Conception et réalisation d'une application web de gestion de stock cas d'étude « SONELGAZ » De Bejaia'.* Mémoire de Master. Université ABDERAHMANE MIRA. 2017/2018.

[16] [AIT HATRIT Fatima et all, Conception et réalisation d'un site web dynamique pour l'aéroport « ABANE RAMDANE » De Bejaia, Mémoire de licence, université ABDERAHMANE MIRA, 2014/2015.].

[17]'ElectronJs Guide | ElectronJs'. Page web : URL : https://www.electronjs.org/ . consulter le 10/09/2020.

[18] 'Wamp server Guide | Wamp server'. Page web : URL : https://www.wampserver.com/. Consulter le 15/09/2020.

[19] The Apache Software Foundation. '*Apache Tomcat 7 User Guide'. Corporation*, 2011.

RÉSUMÉ

L'objectif de notre projet est la conception et la réalisation d'une application bureau pour gestion d'un magasin de pâtisserie. Pour ce faire, nous avons utilisé comme langage de modélisation UML, qui est un formalisme simple, riche et performant en matière de conception Quant à sa mise en œuvre, elle a été réalisé sous l'environnement de développement Visual studio code Afin de réaliser notre application, nous avons utilisé Apache comme serveur de base de données. Le Framework multiplateforme électron Js (HTML, CSS, JavaScript) pour le front end.PHP pour le back end, finalement le format HTTP nous a servi d'intermédiaire entre eux.

Mots clés : UML, PHP, Apache, CSS, HTML, JAVASCRIPT

ABSTRACT

The objective of our project is the design and production of a desktop application for managing a pastry shop.To do this, we used as the modeling language UML, which is a simple, rich and powerful formalism in terms of design. Implementation, it was carried out under the Visual studio code development environment In order to realize our application, we have used Apache as the database server. The electronJs cross-platform Framework (HTML, CSS, JavaScript) for the front end. PHP for the back end, finally the HTTP format served us Intermediary between them.

Keywords: UML, PHP, Apache, CSS, HTML, JAVASCRIPT