

# Rapport de projet

4eme année  
Ingénierie Informatique et Réseaux

---

Étude et développement d'un système de gestion  
de restaurant

---

**Réaliser par :**

AIT EL QADI Anas  
IZMAR Mohamed Taha  
HAMZA Hdidou

**Encadré par :**

MME.NOUHAILA BENSALAH



# Dédicace

À tous ceux qui ont contribué à la concrétisation de ce projet, nous avons ensemble traversé des épreuves et trouvé des inspirations, partagé des rires et des moments de réflexion.

Ce projet est bien plus qu'une simple réalisation ; il est le reflet de notre dévouement, de notre créativité et de notre esprit de collaboration. Chaque membre de notre groupe a apporté une touche unique, illuminant ce projet de manière spéciale et précieuse.

Que cette dédicace témoigne de notre travail acharné, de notre solidarité et de notre passion commune. Puissions-nous nous souvenir de ce projet comme d'une preuve indélébile de ce que nous sommes capables d'accomplir ensemble.

# **Remerciements**

Nous souhaitons exprimer notre profonde gratitude à tous ceux qui ont contribué à la réalisation de ce projet de fin d'études. Tout d'abord, nous remercions sincèrement Monsieur le Directeur de l'EMSI pour la formation de qualité qu'il nous a offerte. Grâce à son leadership et à sa vision, nous avons bénéficié d'un environnement académique exceptionnel, propice à notre épanouissement et à notre réussite. Nos sincères remerciements vont également à nos professeurs, dont l'expertise, la patience et l'engagement ont été essentiels à notre apprentissage. Leurs enseignements et leurs conseils nous ont guidés tout au long de notre parcours académique, nous permettant de développer les compétences nécessaires pour relever les défis de demain. À nos amis, merci pour votre soutien constant, vos encouragements et pour les moments de camaraderie qui ont rendu cette expérience encore plus mémorable. Votre présence à nos côtés a été une source de motivation et de réconfort. Enfin, nous souhaitons exprimer notre infinie reconnaissance à nos parents. Votre amour, votre soutien inconditionnel et vos sacrifices ont été le pilier sur lequel nous avons construit nos rêves. Sans vous, rien de tout cela n'aurait été possible. Ce projet est le fruit de la collaboration, du travail acharné et du soutien de chacun d'entre vous. Merci de nous avoir permis de réaliser cet accomplissement.

# Résumé

Ce rapport discute du processus de développement ainsi que des principales fonctionnalités associées à une application web de gestion de restaurant. L'objectif fondamental de l'application en question est de fournir une interface web permettant de gérer efficacement les activités d'un restaurant, notamment la gestion des menus, des réservations et des commandes clients.

L'implémentation de cette application s'est effectuée en quatre étapes : la planification, la construction, le test et le déploiement. Planification : création d'une interface utilisateur intuitive guidant les interactions avec l'application. Elle se compose de fonctionnalités telles que la gestion des menus, la saisie des réservations et la génération de rapports détaillés sur les performances du restaurant.

Pour réaliser l'application, des technologies web modernes, y compris HTML, CSS et JavaScript pour le frontend, ont été utilisées. De plus, des frameworks tels que Bootstrap ont été employés pour garantir une expérience utilisateur fluide et réactive. Des tests ont été effectués à chaque étape du cycle de vie du développement afin d'assurer la qualité et la fiabilité de l'application. Ces tests incluaient des tests unitaires, d'intégration et de régression pour identifier les erreurs ou les problèmes de performance.

En fin de compte, cette application de gestion de restaurant propose une solution efficace et conviviale pour optimiser les opérations et améliorer l'expérience client.

# Abstract

This report discusses the development process as well as the main features associated with a restaurant management web application. The fundamental objective of the application in question is to provide a web interface to efficiently manage the activities of a restaurant, including the management of menus, reservations and customer orders.

The implementation of this application was carried out in four stages : planning, building, testing and deployment. Planning : creating an intuitive user interface guiding interactions with the application. It consists of features such as menu management, booking reservations, and generating detailed reports on restaurant performance.

To realize the application, modern web technologies, including HTML, CSS and JavaScript for the frontend, were used. Additionally, frameworks such as Bootstrap have been employed to ensure a smooth and responsive user experience. Testing was carried out at each stage of the development lifecycle to ensure the quality and reliability of the application. This testing included unit, integration, and regression testing to identify errors or performance issues.

Ultimately, this restaurant management app offers an efficient and user-friendly solution to optimize operations and improve customer experience.

# Table des matières

<b>Dédicace</b>	<b>2</b>
<b>Remerciements</b>	<b>3</b>
<b>Résumé</b>	<b>4</b>
<b>Abstract</b>	<b>5</b>
<b>Liste des sigles et acronymes</b>	<b>10</b>
<b>Introduction générale</b>	<b>11</b>
<b>1 Contexte général du projet</b>	<b>12</b>
1.1 Présentation du projet . . . . .	13
1.1.1 Contexte et Objectifs . . . . .	13
1.1.2 Cahier des charges . . . . .	13
1.1.3 Explication des termes clés . . . . .	14
1.2 Conclusion . . . . .	15
<b>2 Analyse et conception</b>	<b>16</b>
2.1 Introduction . . . . .	17
2.2 Analyse des besoins . . . . .	17
2.2.1 Les besoins fonctionnels . . . . .	17
2.2.2 Les besoins non fonctionnels . . . . .	18
2.3 Langage de conception . . . . .	19
2.3.1 Langage UML . . . . .	19
2.3.2 Les différents types de diagrammes UML . . . . .	19
2.4 Conception . . . . .	20
2.4.1 Diagrammes utilisés . . . . .	20
2.5 Conclusion . . . . .	21
<b>3 Etude pratique</b>	<b>22</b>
3.1 Introduction . . . . .	23
3.2 Outils . . . . .	23
3.2.1 Html . . . . .	23
3.2.2 css . . . . .	25
3.2.3 bootstrap . . . . .	27
3.2.4 js . . . . .	28
3.2.5 Enterprise Architec . . . . .	29

3.2.6	ASP.NET Core . . . . .	30
3.2.7	Visual Studio . . . . .	31
3.2.8	SQL Server Management Studio . . . . .	31
3.2.9	Modèle MVC . . . . .	32
3.2.10	Langage C sharp . . . . .	33
3.3	Réalisation du projet . . . . .	33
3.4	Conclusion . . . . .	41
	<b>Conclusion générale</b>	<b>42</b>
	<b>Perspectives</b>	<b>43</b>
	<b>Webographie</b>	<b>44</b>

# Table des figures

2.1	diagramme de cas d'utilisation . . . . .	20
2.2	diagramme de classe . . . . .	21
3.1	Html logo . . . . .	23
3.2	Structure de base d'un document HTML . . . . .	24
3.3	css logo . . . . .	25
3.4	Structure de CSS . . . . .	26
3.5	bootstrap logo . . . . .	27
3.6	js logo . . . . .	28
3.7	enterprise architect logo . . . . .	29
3.8	ASP.net core logo . . . . .	30
3.9	Visual Studio logo . . . . .	31
3.10	ssms logo . . . . .	31
3.11	démarche du modele mvc . . . . .	32
3.12	C logo . . . . .	33
3.13	Page de connexion . . . . .	34
3.14	Page principale de l'admine . . . . .	35
3.15	admin controller . . . . .	35
3.16	Page de panier . . . . .	36
3.17	cart controller 1 . . . . .	37
3.18	cart controller 2 . . . . .	37
3.19	Page des commandes . . . . .	38
3.20	Commandes controller . . . . .	38
3.21	Page des employés . . . . .	39
3.22	employe controller . . . . .	39
3.23	Page de menu . . . . .	40
3.24	plats controller . . . . .	40

# Liste des tableaux

1	Tableau des sigles et acronymes . . . . .	10
---	---	----

# Liste des sigles et acronymes

Sigle	Description
HTML	HyperText Markup Language
CSS	Cascading Style Sheets
VS	Visual Studio
UML	Unified Modeling Language
SSMS	SQL Server Management Studio

TABLE 1 – Tableau des sigles et acronymes

# Introduction générale

Avec l'évolution rapide des menaces informatiques et l'importance croissante de la sécurité des systèmes d'information, la gestion des patches est devenue une nécessité cruciale pour toute organisation. La capacité à garantir que les composants du système sont à jour avec les derniers patches de sécurité est essentielle pour protéger les données sensibles et maintenir l'intégrité des opérations.

Notre projet, réalisé au cours de notre stage en tant qu'équipe, se concentre sur le développement d'une interface web dédiée à la vérification et à la gestion des patches de sécurité. Cette solution vise à fournir aux administrateurs système et aux responsables de la sécurité un outil efficace pour surveiller en temps réel l'état des mises à jour de sécurité, identifier les patches manquants et faciliter leur application.

L'objectif de ce rapport est d'offrir une analyse approfondie du cycle de développement de cette interface web, en mettant en évidence les choix technologiques, les processus de conception et les caractéristiques fonctionnelles qui ont contribué à créer une solution conviviale et performante. Nous avons pris soin d'assurer une gestion simplifiée des mises à jour tout en garantissant la sécurité et l'efficacité du processus.

Au cours de ce projet, nous avons utilisé nos compétences en développement web, en conception d'interface utilisateur et en gestion de projet pour surmonter les défis liés à la création d'une solution complète et fiable. Ce rapport fournit un aperçu détaillé de notre approche de conception, de l'analyse des besoins initiaux à la mise en œuvre des fonctionnalités clés, en soulignant les leçons apprises et les succès obtenus tout au long du développement.

# Chapitre 1

## Contexte général du projet

## 1.1 Présentation du projet

### 1.1.1 Contexte et Objectifs

La gestion des activités d'un restaurant est un processus complexe qui nécessite une organisation efficace pour répondre aux attentes des clients et maximiser les performances de l'établissement. Parmi les principaux défis figurent la gestion des menus, des réservations, des commandes, et l'optimisation des ressources humaines et matérielles. Ces processus, souvent gérés manuellement, peuvent être source d'erreurs et de perte de temps.

Le projet consiste à développer une application web de gestion de restaurant permettant de centraliser et d'automatiser ces tâches essentielles. L'application offre une interface intuitive pour gérer les menus, les réservations, les commandes et les rapports d'activité. Elle vise à simplifier les opérations quotidiennes, améliorer l'efficacité et offrir une expérience client optimale.

#### Objectifs :

- Fournir un outil centralisé pour gérer les différentes activités d'un restaurant.
- Permettre aux administrateurs de personnaliser les menus et suivre les commandes en temps réel.
- Faciliter la gestion des réservations et minimiser les risques de double réservation.
- Générer des rapports détaillés pour analyser les performances du restaurant et prendre des décisions stratégiques.

### 1.1.2 Cahier des charges

#### Fonctionnalités principales :

Gestion des menus : L'application permettra d'ajouter, modifier et supprimer des éléments de menu avec des descriptions, prix et photos.

Réservations en ligne : Les clients pourront réserver des tables en précisant le nombre de personnes, la date et l'heure. Les administrateurs auront une vue globale des réservations pour mieux planifier.

Suivi des commandes : Les commandes, qu'elles soient effectuées sur place ou en ligne, seront enregistrées et suivies en temps réel. Une notification sera envoyée au personnel pour la préparation et au client pour le suivi.

Rapports et statistiques : L'application générera des rapports détaillés sur les ventes, les performances des plats, les heures de pointe, et les avis des clients.

#### Exigences techniques :

Interface utilisateur : L'application doit être simple à utiliser, avec une interface intuitive pour les administrateurs et les clients.

Base de données : Les données relatives aux menus, commandes et réservations seront stockées de manière sécurisée dans une base de données relationnelle.

Sécurité : L'application doit inclure des fonctionnalités de gestion des accès avec authentification, et les données sensibles doivent être protégées par chiffrement.

#### **Contraintes :**

Compatibilité : L'application doit être accessible depuis différents appareils (ordinateurs, tablettes, smartphones).

Performance : Elle doit gérer un grand nombre d'utilisateurs et de commandes simultanément tout en offrant une expérience fluide.

### **1.1.3 Explication des termes clés**

#### **Menu personnalisé :**

Un menu personnalisé est une liste de plats que les restaurants peuvent adapter en fonction des préférences des clients ou des spécificités de leur établissement. Par exemple, un menu peut inclure des options végétariennes, sans gluten, ou des plats spécifiques pour des occasions spéciales.

#### **Réservations en ligne :**

Ce processus permet aux clients de réserver une table via une plateforme web, en spécifiant des détails comme le nombre de convives, l'heure et la date. Cette fonctionnalité automatise la gestion des réservations, réduisant ainsi les erreurs et les conflits.

#### **Statistiques de vente :**

Les statistiques de vente représentent les données collectées sur les performances financières du restaurant, telles que le nombre de plats vendus, les périodes de forte activité et les plats les plus populaires. Ces données permettent d'optimiser les opérations et d'augmenter la rentabilité.

#### **Notifications en temps réel :**

Les notifications en temps réel permettent de tenir le personnel informé immédiatement des nouvelles commandes ou modifications, assurant ainsi une meilleure coordination entre la salle et la cuisine.

**Chiffrement des données :**

C'est une méthode de protection des données sensibles comme les informations personnelles des clients et les transactions en ligne. Cela garantit la confidentialité et la sécurité des informations stockées ou transmises.

## 1.2 Conclusion

Pour résumer, notre projet d'application web de gestion de restaurant vise à répondre aux besoins opérationnels des restaurateurs en leur offrant une solution moderne, intuitive et sécurisée. Ce chapitre a présenté le contexte, les objectifs et les fonctionnalités attendues, établissant ainsi les bases pour concevoir et implémenter une solution qui optimise les performances et améliore l'expérience client.

## Chapitre 2

### Analyse et conception

## 2.1 Introduction

La conception joue un rôle essentiel dans la réalisation de solutions innovantes. Que ce soit dans le domaine de l'ingénierie, de l'architecture, du design ou de la gestion, la conception est le processus clé qui permet de transformer une idée abstraite en une réalité tangible. Ce chapitre se concentre sur l'importance de la conception et explore les différentes étapes et approches nécessaires pour concevoir des solutions adaptées à la gestion d'un restaurant. En résumé, ce chapitre de conception nous permettra d'approfondir les fonctionnalités de l'application web de gestion de restaurant, en commençant par l'élaboration du diagramme de cas d'utilisation et en terminant par le diagramme de séquence.

## 2.2 Analyse des besoins

### 2.2.1 Les besoins fonctionnels

Les besoins fonctionnels décrivent les fonctionnalités principales de l'application. L'application de gestion de restaurant doit permettre de :

#### Gestion des réservations :

- Permettre aux clients de réserver une table en ligne via l'application.
- Fournir aux administrateurs un tableau de bord pour gérer les réservations (validation, modification, ou annulation).

#### Gestion des menus :

- Offrir une interface pour ajouter, modifier ou supprimer des plats du menu.
- Permettre de catégoriser les plats (entrées, plats principaux, desserts) et d'ajouter des détails comme le prix, les ingrédients, ou la disponibilité.

#### Suivi des commandes :

- Faciliter la prise des commandes depuis l'application (via les serveurs ou directement par les clients).
- Permettre de suivre en temps réel l'état des commandes (en attente, en cours, servi).

#### Gestion des stocks :

- Suivre les stocks des ingrédients utilisés pour préparer les plats.
- Alerter les administrateurs lorsque certains ingrédients atteignent un seuil critique.

#### Gestion des employés :

- Permettre aux administrateurs de gérer les informations des employés (nom, poste, horaires de travail, etc.).
- Offrir une fonctionnalité pour planifier les horaires des employés et attribuer des tâches spécifiques.

#### Rapports et statistiques :

- Générer des rapports sur les ventes quotidiennes, hebdomadaires ou mensuelles.
- Fournir des statistiques sur les plats les plus commandés, les périodes d'affluence, et les clients réguliers.

- Intégrer des rapports sur la performance et la productivité des employés.

#### **Interface utilisateur :**

- Offrir une interface intuitive et conviviale pour le personnel et les administrateurs.
- Fournir une section distincte pour les clients permettant de visualiser le menu, réserver des tables, ou passer des commandes.

### **2.2.2 Les besoins non fonctionnels**

Pour garantir une solution performante et fiable, l'application doit respecter les contraintes suivantes :

#### **Sécurité :**

- Protéger les données sensibles des clients (informations personnelles, réservations, paiements) et des employés (informations personnelles, rôles).
- Gérer les accès et les autorisations pour les différents utilisateurs (administrateurs, serveurs, clients).

#### **Performance :**

- Assurer un temps de réponse rapide même lors des périodes de forte activité.
- Supporter un grand nombre de commandes, de réservations et de données liées aux employés simultanément.

#### **Fiabilité et disponibilité :**

- Garantir un fonctionnement continu pour gérer les réservations, commandes et gestion du personnel en temps réel.
- Réduire au maximum les interruptions de service.

#### **Compatibilité :**

- Être compatible avec différents types de périphériques (PC, tablettes, smartphones).
- Supporter les navigateurs web modernes.

#### **Maintenance :**

- Permettre une mise à jour facile de l'application pour ajouter de nouvelles fonctionnalités ou corriger des bugs.
- Offrir un système de gestion des logs pour surveiller les éventuels incidents.

#### **Conformité :**

- Respecter les normes légales et réglementaires en matière de protection des données et de sécurité.

#### **Facilité d'utilisation :**

- Fournir une interface simple à utiliser pour le personnel du restaurant et les administrateurs.
- Inclure des instructions claires pour les clients lors de l'utilisation de l'application.

## 2.3 Langage de conception

### 2.3.1 Langage UML

L'UML (Unified Modeling Language ou Langage de modélisation unifiée en français) est un langage graphique de modélisation informatique. Ce langage est désormais la référence en modélisation objet, ou programmation orientée objet. Cette dernière consiste à modéliser des éléments du monde réel (immeuble, ingrédients, personne, logos, organes du corps...) ou virtuel (temps, prix, compétence...) en un ensemble d'entités informatiques appelées « objet ». L'UML est constitué de diagrammes qui servent à visualiser et décrire la structure et le comportement des objets qui se trouvent dans un système. Il permet de présenter des systèmes logiciels complexes de manière plus simple et compréhensible qu'avec du code informatique. L'UML a des applications dans le développement logiciel, mais aussi dans l'industrie (pour modéliser les flux de processus par exemple), dans l'ingénierie ou le marketing.

### 2.3.2 Les différents types de diagrammes UML

L'UML définit 14 types de diagrammes divisés en deux catégories.

1. Les diagrammes de structure représentent les éléments du système, leurs propriétés et leurs relations entre eux :

Diagramme de classes  
Diagramme d'objets  
Diagramme de composants  
Diagramme de structure composite  
Diagramme d'ensemble  
Diagramme de déploiement  
Diagramme de profil

2. Les diagrammes de comportement représentent les processus et les interactions entre les objets

Diagramme de cas d'utilisation  
Diagramme d'activité  
Diagramme d'état-transition  
Diagramme de séquence  
Diagramme de communication  
Diagramme de temps  
Diagramme d'aperçu d'interaction

## 2.4 Conception

### 2.4.1 Diagrammes utilisés

Pour commencer, nous allons élaborer le diagramme de cas d'utilisation, qui représente les différentes interactions entre les utilisateurs et le système. Ce diagramme nous permettra d'identifier les fonctionnalités essentielles de l'application et de définir les actions principales que les utilisateurs pourront effectuer.

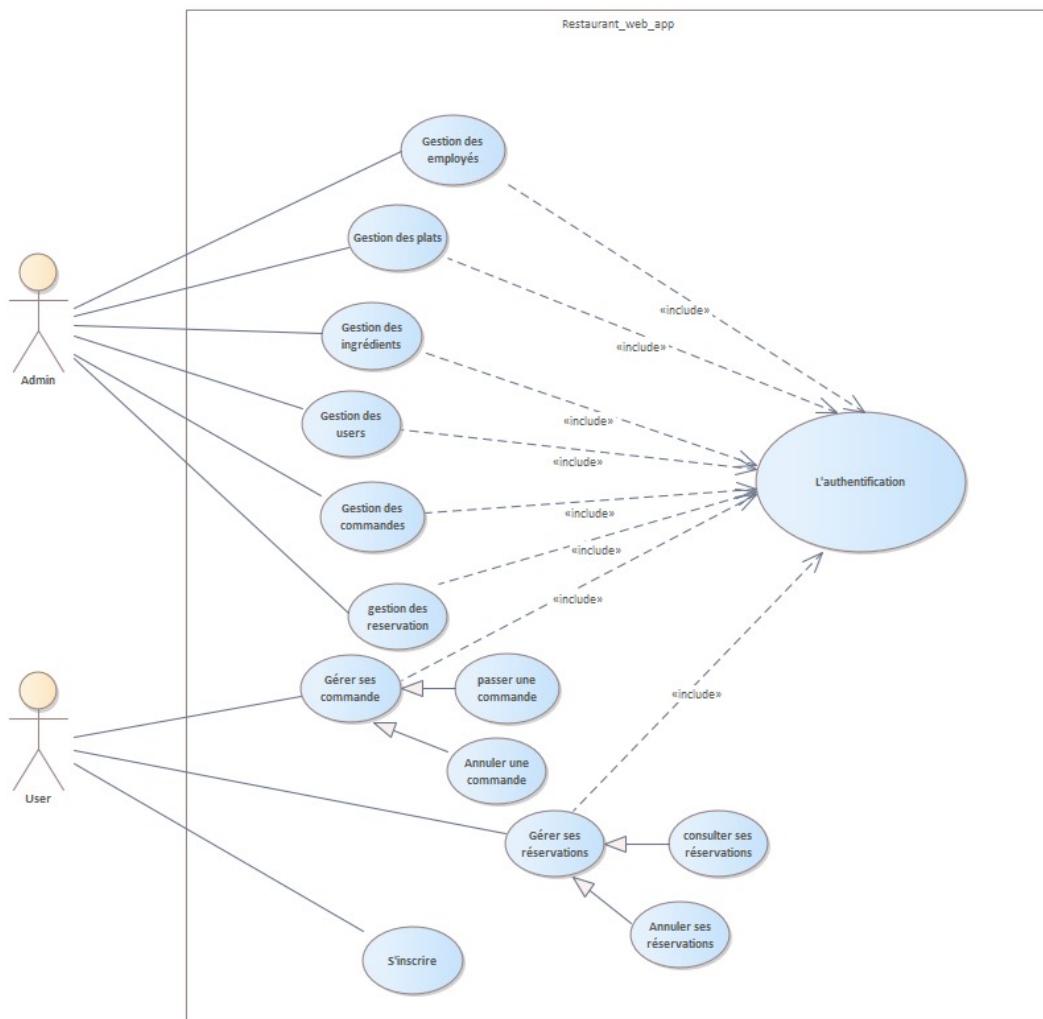


FIGURE 2.1 – diagramme de cas d'utilisation

Après cela, nous procéderons à la création du diagramme de classes, qui illustre la structure statique du système en modélisant les classes, leurs attributs, méthodes et interrelations. Ce schéma sera utile pour que nous puissions comprendre l'interaction entre les différentes parties de l'application et concevoir une architecture cohérente et extensible.

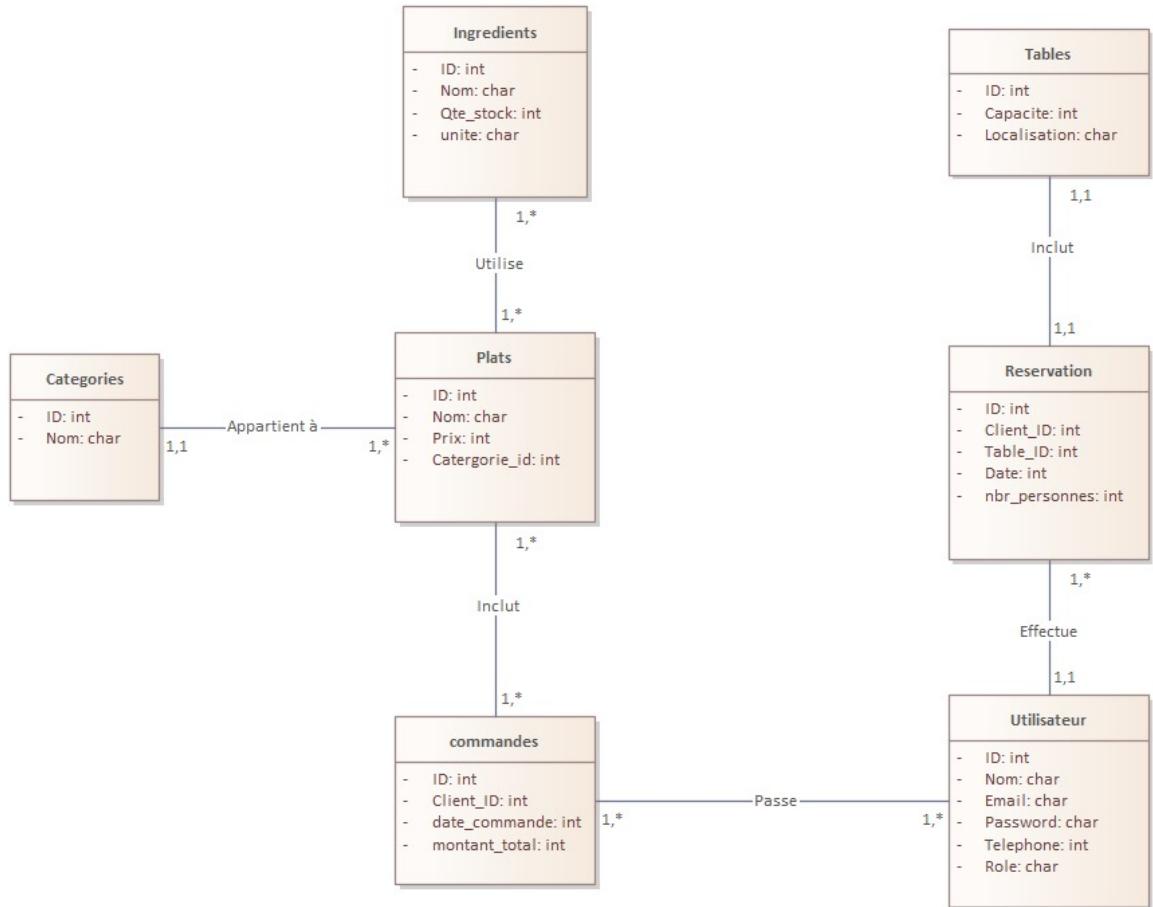


FIGURE 2.2 – diagramme de classe

## 2.5 Conclusion

Nous avons structuré notre application autour de compétences architecturales avec une distinction nette, et l'aide fut fournie par des diagrammes de conception pour donner le développement clair et consistant. A travers plusieurs schémas de cahier de charges, séquences de classe et autres diagrammes, nous avons pu tracer des fondations solides pour démarrer et réussir notre projet.

# Chapitre 3

## Etude pratique

## 3.1 Introduction

Idée d'amélioration pour faciliter la vérification des composants du système d'information, la création d'une application web dédiée à la vérification et à la gestion des patches de sécurité est une solution efficace. solution, afin de rendre les processus de sécurité automatisés et simplifiés. L'application il est il est possible de la développer en utilisant des technologies informatiques appropriées. Parmi les exemples figurent l'HTML et à CSS (cascading style sheets).

## 3.2 Outils

### 3.2.1 Html



FIGURE 3.1 – Html logo

Développé par le W3C (World Wide Web Consortium) et le WHATWG (Web Hypertext Application Technology Working Group), le format ou langage HTML est apparu dans les années 1990. Il a progressivement subi des modifications et propose depuis 2014 une version HTML5 plus aboutie.

L'HTML est ce qui permet à un créateur de sites Web de gérer la manière dont le contenu de ses pages Web va s'afficher sur un écran, via le navigateur. Il repose sur un système de balises permettant de titrer, sous-titrer, mettre en gras, etc., du texte et d'introduire des éléments interactifs comme des images, des liens, des vidéos... L'HTML est plus facilement compris des robots de crawl des moteurs de recherche que le language JavaScript, aussi utilisé pour rendre les pages plus interactives.

## Structure de base d'un document HTML

Voici un exemple

```
<!DOCTYPE html>
<html>

    <head>
        <title> Title here </title>
    </head>

    <body>
        Web page content goes here.
    </body>

</html>
```

FIGURE 3.2 – Structure de base d'un document HTML

## Le rôle du HTML

est donc crucial puisqu'il va être notre langage privilégié pour indiquer aux navigateurs ce quoi est constituée chaque page et ce qu'ils doivent afficher. Grâce au HTML, on va par exemple pourvoir indiquer que tel contenu est un texte qui n'est qu'un paragraphe, que tel autre contenu est un texte qui est un titre de niveau 1 dans notre page, que tel autre contenu est une liste, un lien, etc. En plus de cela, le HTML va également nous permettre d'insérer différents médias (images, vidéos, etc.) dans nos pages web en indiquant au navigateur » à cette place- là dans ma page, je veux que s'affiche cette image « . Notez que dans ce cas précis, pour que le navigateur affiche la bonne image, on va lui fournir l'adresse de l'image dans le code HTML.

### 3.2.2 css



FIGURE 3.3 – css logo

Le terme CSS est l'acronyme anglais de Cascading Style Sheets qui peut se traduire par "feuilles de style en cascade". Le CSS est un langage informatique utilisé sur l'internet pour mettre en forme les fichiers HTML ou XML puisque leur options stylistiques d'un document sont quelque peu limitées. Ainsi, les feuilles de style, aussi appelé les fichiers CSS, comprennent du code qui permet de gérer le design d'une page en HTML.

#### À quoi sert le CSS

Le CSS permet donc de définir l'esthétique d'un site web. Sans CSS, une page web ne serait qu'une succession d'éléments noirs sur fond blanc les uns à la suite des autres.

C'est avec du code CSS que l'on définit par exemple :

la police d'un texte et son aspect (couleur, taille, etc.)

les marges et le rembourrage (padding) entourant un élément

l'apparence d'un menu

la création d'éléments géométriques

la mise en place de grilles d'images

l'ajout de bordure

l'application de différents effets ou animations simples

le changement de présentation d'un site selon l'appareil utilisé par l'utilisateur (téléphone

portable, ordinateur, tablette)

la manière dont on navigue dans un site internet

### La structure d'une déclaration CSS

Vous pouvez utiliser une déclaration CSS pour déterminer quels éléments de votre document électronique doivent prendre quelles valeurs ou propriétés. Dans sa structure de base, la déclaration se compose d'un sélecteur et d'accolades. Dans ces parenthèses, vous énumérez les déclarations de propriétés, qui sont séparées par des points-virgules. Chaque directive se compose du nom de la propriété, de deux points et d'une valeur spécifique. Après la dernière déclaration de propriété et avant la parenthèse finale, il est possible de placer un autre point-virgule, mais il n'est pas obligatoire. La déclaration CSS dans le diagramme ci-dessous permet de s'assurer que le titre h1 est affiché dans la couleur bleue et la taille de police 12 :

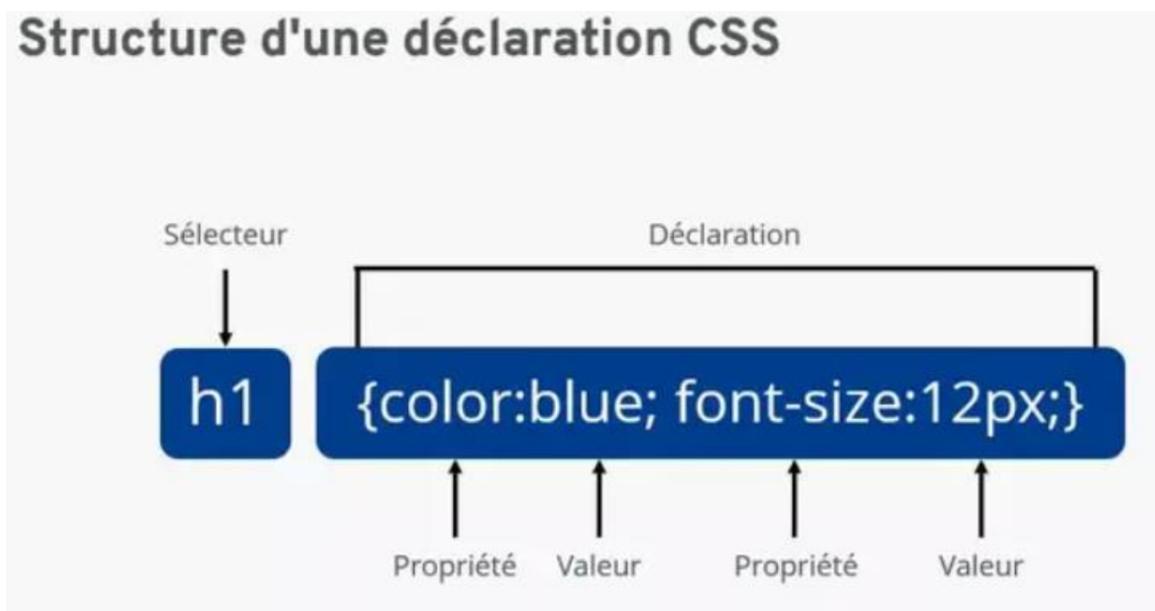


FIGURE 3.4 – Structure de CSS

### 3.2.3 bootstrap



FIGURE 3.5 – bootstrap logo

Bootstrap est un framework de développement front-end gratuit et open source pour la création de sites Web et d'applications Web. Conçu pour permettre le développement réactif de sites Web destinés aux mobiles, Bootstrap fournit une collection de syntaxes pour la conception de modèles.

En tant que framework, Bootstrap inclut les bases du développement Web réactif, les développeurs n'ont donc qu'à insérer le code dans un système de grille prédéfini. Le framework Bootstrap est construit sur Hypertext Markup Language (HTML), des feuilles de style en cascade (CSS) et JavaScript. Les développeurs Web utilisant Bootstrap peuvent créer des sites Web beaucoup plus rapidement sans perdre de temps à se soucier des commandes et fonctions de base.

### 3.2.4 js

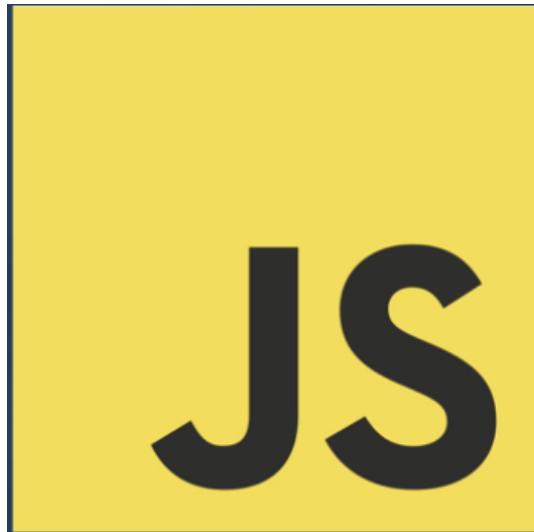


FIGURE 3.6 – js logo

#### Caractéristiques de JS

1. Langage de Script Côté Client : JavaScript est principalement employé côté navigateur pour manipuler le contenu HTML/CSS et interagir avec l'utilisateur, évitant ainsi de recharger la page.
2. Polyvalent et Dynamique : JavaScript permet l'utilisation de différents paradigmes de programmation, tels que la programmation orientée objet, fonctionnelle et impérative. Il convient parfaitement à la création de applications web complexes, jeux vidéo, applications mobiles et bien d'autres.
3. Syntaxe Simple : La syntaxe simple et expressive de JavaScript en fait un langage apprécié, accessible aux débutants mais proposant également des fonctionnalités avancées pour les développeurs expérimentés.
4. Modèle d'Objet Document (DOM) : En JavaScript, il est possible de manipuler le DOM qui représente les éléments HTML d'une page sous forme d'arborescence. Cela permet de faciliter la modification dynamique de l'interface utilisateur ainsi que la gestion des événements utilisateurs.
5. Large Écosystème : JavaScript est doté d'une large gamme de frameworks et de bibliothèques tels que React.js, Vue.js et AngularJS facilitent la création d'applications web modernes et réactives.
6. Exécution Rapide : JavaScript étant un langage interprété, il peut s'exécuter directement dans le navigateur. Cela offre aux utilisateurs des interactions instantanées et des mises à jour en temps réel, sans devoir recharger la page.

### Utilisation de JS :

JavaScript est couramment utilisé pour améliorer l'interactivité des pages web en y ajoutant des fonctionnalités comme des animations, des formulaires dynamiques, des effets visuels et même la création de jeux ou d'applications web en temps réel. Il est également possible de développer des applications mobiles hybrides avec JavaScript. JavaScript est un élément essentiel du développement front-end moderne, se combinant souvent avec HTML et CSS pour former une base solide.

#### 3.2.5 Enterprise Architec



FIGURE 3.7 – enterprise architect logo

Enterprise Architect est un logiciel de modélisation UML (Unified Modeling Language) utilisé pour concevoir des architectures de systèmes d'information complexes. Il permet de créer des diagrammes pour modéliser les processus, les systèmes, et les structures d'entreprise, ce qui facilite la compréhension et la gestion des projets informatiques à grande échelle.

### 3.2.6 ASP.NET Core



FIGURE 3.8 – ASP.net core logo

ASP.NET Core est un framework de développement web moderne et open-source développé par Microsoft. Il est conçu pour créer des applications performantes, évolutives et multiplateformes, pouvant être exécutées sur Windows, macOS et Linux. Grâce à son architecture modulaire, ASP.NET Core permet une meilleure flexibilité et une personnalisation accrue par rapport à son prédecesseur, ASP.NET.

Ce framework est particulièrement apprécié pour son intégration avec des outils modernes comme Entity Framework Core pour la gestion des bases de données et son support natif pour les API REST.

De plus, il offre des performances optimisées grâce à son moteur de serveur web intégré, Kestrel, et à son support pour le paradigme asynchrone. En combinant sécurité, compatibilité multiplateforme et un écosystème riche, ASP.NET Core constitue une solution idéale pour développer des applications web robustes et modernes, répondant aux besoins des entreprises et des utilisateurs.

### 3.2.7 Visual Studio



FIGURE 3.9 – Visual Studio logo

Visual Studio est un environnement de développement intégré (IDE) puissant et polyvalent développé par Microsoft. Il est conçu pour le développement d'applications sur une large gamme de plateformes, notamment le web, le mobile, le bureau, et le cloud.

Visual Studio prend en charge plusieurs langages de programmation, tels que C, VB.NET, Python, et JavaScript, et offre des outils robustes pour le débogage, les tests et la collaboration en équipe. Avec des fonctionnalités avancées comme IntelliSense, qui suggère des complétions de code en temps réel, et Git intégré pour le contrôle de version, Visual Studio améliore considérablement la productivité des développeurs.

Il propose également un ensemble d'extensions pour personnaliser l'IDE selon les besoins du projet, ce qui en fait un choix idéal pour des projets de toutes tailles.

### 3.2.8 SQL Server Management Studio



Microsoft®  
**SQL Server**  
**Manager**

FIGURE 3.10 – ssms logo

SQL Server Management Studio (SSMS) est un outil essentiel pour la gestion et l'admi-

nistration des bases de données SQL Server. Développé par Microsoft, SSMS offre une interface utilisateur intuitive pour effectuer des tâches variées, telles que la création, la modification, et la suppression de bases de données, ainsi que l'exécution de requêtes SQL.

Cet outil inclut également des fonctionnalités avancées pour la surveillance des performances des bases de données, la gestion des utilisateurs et des permissions, et l'automatisation des tâches via des scripts SQL ou des plans de maintenance.

Grâce à ses outils graphiques intuitifs et à son éditeur SQL performant, SSMS simplifie le travail des développeurs et des administrateurs de bases de données, tout en assurant une gestion efficace et sécurisée des données.

### 3.2.9 Modèle MVC

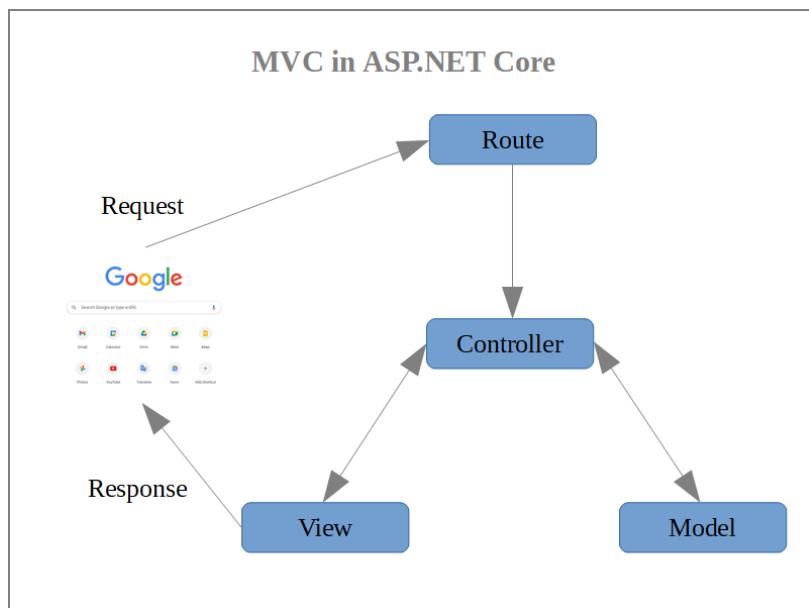


FIGURE 3.11 – démarche du modèle mvc

Le modèle MVC (Model-View-Controller) dans ASP.NET Core est une architecture logicielle qui divise une application web en trois composants principaux :

**Modèle (Model)** : Représente les données et la logique métier de l'application. Il interagit avec la base de données via des entités, des règles de validation et utilise souvent Entity Framework Core.

**Vue (View)** : Responsable de la présentation des données à l'utilisateur. Elle utilise Razor pour afficher dynamiquement les informations et permet l'intégration de C# dans des pages HTML.

**Contrôleur (Controller)** : Gère les interactions entre le modèle et la vue. Il reçoit les requêtes des utilisateurs, traite la logique métier et retourne la vue appropriée.

Avantages :

Séparation des responsabilités pour un code plus structuré et maintenable.

Testabilité facilitée avec des tests unitaires.

Flexibilité et extensibilité des composants.

Rendu dynamique des pages via Razor.

Le modèle MVC dans ASP.NET Core permet de créer des applications web performantes, maintenables et évolutives, en suivant des pratiques de développement modulaire.

### 3.2.10 Langage C sharp



FIGURE 3.12 – C logo

Le langage C (prononcé "C sharp") est un langage de programmation moderne, orienté objet, développé par Microsoft dans le cadre de la plateforme .NET.

Conçu pour être simple à apprendre tout en étant puissant, C permet de développer une large gamme d'applications, allant des applications web et desktop aux jeux vidéo, en passant par les applications mobiles.

Il offre une syntaxe claire et concise, une gestion automatique de la mémoire grâce au ramasse-miettes (garbage collection), et une forte intégration avec l'écosystème .NET, facilitant ainsi le développement de solutions robustes et performantes.

C prend en charge des fonctionnalités avancées comme les expressions lambda, LINQ (Language Integrated Query), ainsi que les tâches asynchrones et parallèles, ce qui en fait un langage flexible et adapté aux exigences modernes de programmation.

## 3.3 Réalisation du projet

Cette application contient 2 espaces principaux :

-L'interface du client .

-L'interface de l'administrateur .

## Page de connexion

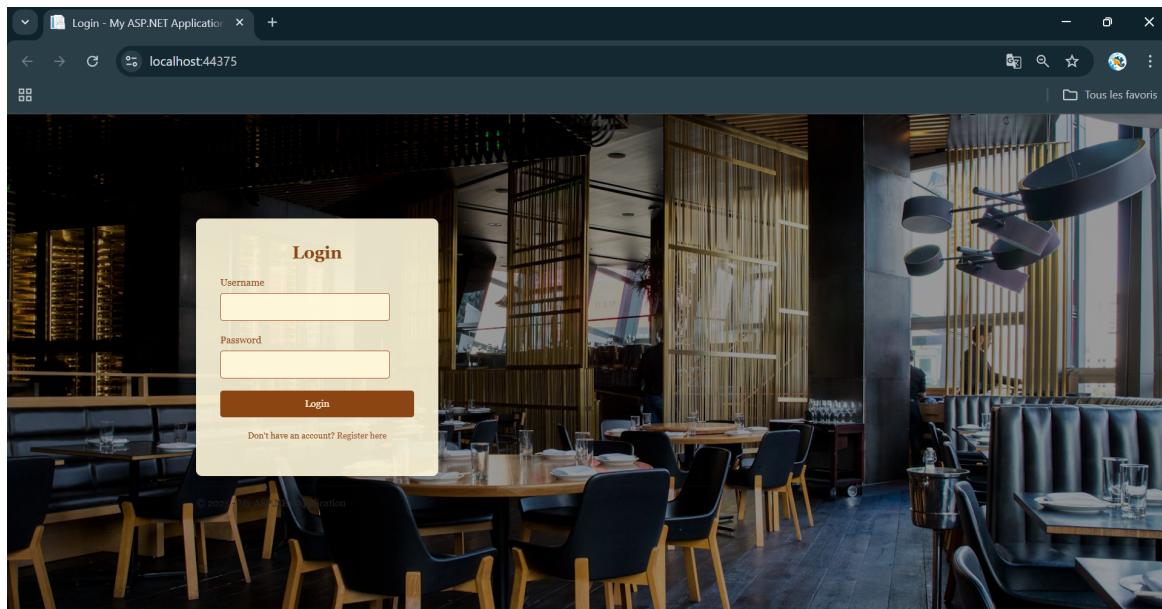


FIGURE 3.13 – Page de connexion

## Page principale de l'admine :

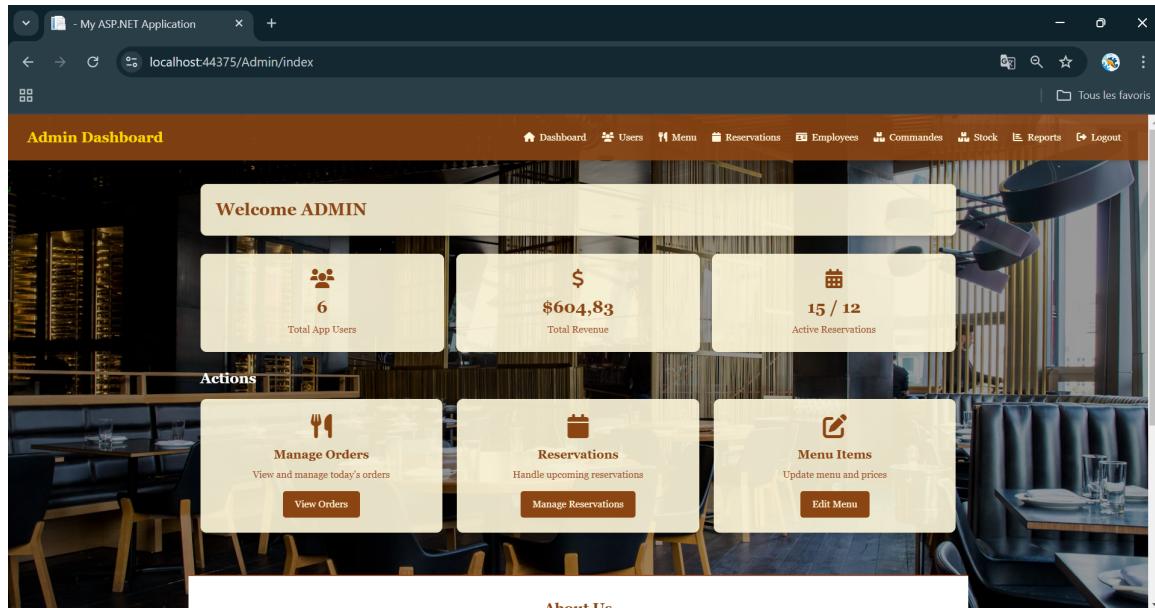


FIGURE 3.14 – Page principale de l'admine

```

11
12
13 [SessionCheck]
14 [SessionCheckAdmin]
15 0 références
16 public class AdminController : Controller
17 {
18     private DB_RestoEntities db = new DB_RestoEntities();
19     // GET: Admin
20     0 références
21     public ActionResult Index()
22     {
23         var userId = Session["UserId"] as int?;
24         int userCount = db.Users.Count();
25         decimal totalSalary = (decimal)db.Ventes.Sum(u => u.montant_total);
26         int tableCount = db.Tables_Restaurant.Count();
27         int reservationCount = db.Reservations.Count();
28
29         var user = db.Users.FirstOrDefault(u => u.id == userId);
30
31         ViewBag.UserCount = userCount;
32         ViewBag.TotalSalary = totalSalary;
33         ViewBag.TableCount = tableCount;
34         ViewBag.ReservationCount = reservationCount;
35         ViewBag.User = user;
36
37         return View();
    }
}

```

FIGURE 3.15 – admin controller

## Page de panier

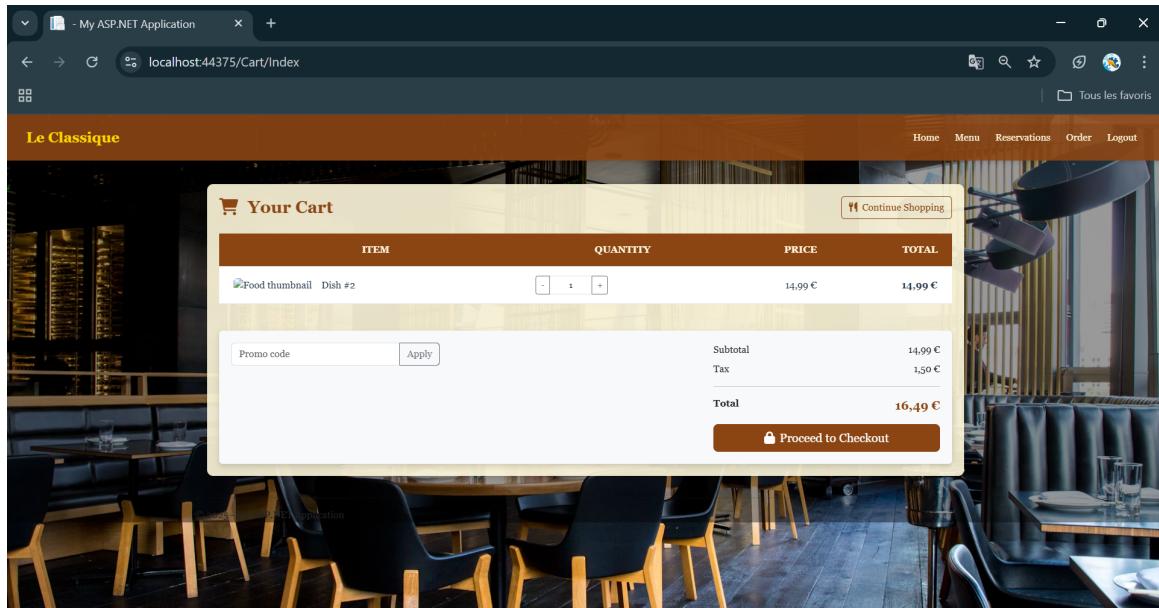


FIGURE 3.16 – Page de panier

```

7
8     [SessionCheck]
9      0 références
10     <public class CartController : Controller
11     {
12         private readonly DB_RestoEntities db = new DB_RestoEntities();
13
14         // GET: Cart
15         0 références
16         public ActionResult Index()
17         {
18             // Récupérer le panier depuis la session
19             var cart = Session["Cart"] as List<CartItem> ?? new List<CartItem>();
20             return View(cart);
21         }
22
23         // POST: Cart/Checkout
24         0 références
25         public ActionResult Checkout()
26         {
27             var cart = Session["Cart"] as List<CartItem>;
28             if (cart == null || !cart.Any())
29             {
30                 return RedirectToAction("Index", "Home");
31             }
32
33             var clientId = Session["UserId"] as int?;
34             if (clientId == null)
35             {
36                 return RedirectToAction("Login", "Account");
37             }
38
39             // Calculer le montant total de la commande
40             var totalAmount = cart.Sum(item => item.Price * item.Quantity);
41
42             // Créer une nouvelle commande
43             var commande = new Commande
44             {
45                 client_id = clientId.Value,
46                 date = DateTime.Now,
47                 montant_total = totalAmount
48             };
49
50             // Ajouter la commande à la base de données
51             db.Commandes.Add(commande);
52             db.SaveChanges(); // Sauvegarder pour obtenir l'ID de la commande
53             // Ajouter les détails de la commande
54             foreach (var item in cart)
55             {
56                 // Ajouter une ligne de commande
57                 var ligneCommande = new Lignes_Commande
58                 {
59                     plat_id = item.PlatId,
60                     quantite = item.Quantity,
61                     prix_apres_remise = item.Price,
62                     commande_id = commande.id
63                 };
64                 db.Lignes_Commande.Add(ligneCommande);
65
66                 // Mettre à jour le stock des ingrédients
67                 var ingredients = db.Plats.Ingredients
68                     .Where(pi => pi.pplat_id == item.PlatId)
69                     .ToList();
70                 foreach (var ingredient in ingredients)
71                 {
72                     var stockIngredient = db.Ingredients.Find(ingredient.ingredient_id);
73                     if (stockIngredient != null)
74                     {
75                         // Soustraire la quantité nécessaire en fonction de la quantité commandée
76                         stockIngredient.quantite_stock -= item.Quantity * ingredient.quantite;
77
78                         // Vérifier si le stock est suffisant
79                         if (stockIngredient.quantite_stock < 0)
80                         {
81                             ModelState.AddModelError("", $"Stock insuffisant pour l'ingrédient : {stockIngredient.nom}");
82                             return View("Error");
83                         }
84                     }
85
86                     // Ajouter une entrée dans la table des ventes
87                     var vente = new Vente
88                     {
89                         plat_id = item.PlatId,
90                         quantite = item.Quantity,
91                         prix_unitaire = item.Price,
92                         date = DateTime.Now
93                     };
94                     db.Ventes.Add(vente);
95
96                     // Sauvegarder toutes les modifications
97                     db.SaveChanges();
98                     // Vider le panier après la commande
99                     Session["Cart"] = null;
100                    // Rediriger vers la page d'accueil
101                    return RedirectToAction("Index", "Home");
102                }
103            }
104        }
105    }
106
```

FIGURE 3.17 – cart controller 1

```

51         foreach (var item in cart)
52         {
53             // Ajouter une ligne de commande
54             var ligneCommande = new Lignes_Commande
55             {
56                 plat_id = item.PlatId,
57                 quantite = item.Quantity,
58                 prix_apres_remise = item.Price,
59                 commande_id = commande.id
60             };
61             db.Lignes_Commande.Add(ligneCommande);
62
63             // Mettre à jour le stock des ingrédients
64             var ingredients = db.Plats.Ingredients
65                 .Where(pi => pi.pplat_id == item.PlatId)
66                 .ToList();
67             foreach (var ingredient in ingredients)
68             {
69                 var stockIngredient = db.Ingredients.Find(ingredient.ingredient_id);
70                 if (stockIngredient != null)
71                 {
72                     // Soustraire la quantité nécessaire en fonction de la quantité commandée
73                     stockIngredient.quantite_stock -= item.Quantity * ingredient.quantite;
74
75                     // Vérifier si le stock est suffisant
76                     if (stockIngredient.quantite_stock < 0)
77                     {
78                         ModelState.AddModelError("", $"Stock insuffisant pour l'ingrédient : {stockIngredient.nom}");
79                         return View("Error");
80                     }
81                 }
82
83                 // Ajouter une entrée dans la table des ventes
84                 var vente = new Vente
85                 {
86                     plat_id = item.PlatId,
87                     quantite = item.Quantity,
88                     prix_unitaire = item.Price,
89                     date = DateTime.Now
90                 };
91                 db.Ventes.Add(vente);
92
93                 // Sauvegarder toutes les modifications
94                 db.SaveChanges();
95                 // Vider le panier après la commande
96                 Session["Cart"] = null;
97                 // Rediriger vers la page d'accueil
98                 return RedirectToAction("Index", "Home");
99             }
100         }
101     }
102 }
```

FIGURE 3.18 – cart controller 2

## Page des commandes

date	montant_total	nom	Actions
12/01/2025 01:34:34	12,99	yassine	<a href="#">Edit</a> <a href="#">Delete</a> <a href="#">Details</a>
12/01/2025 01:36:15	12,99	Karim	<a href="#">Edit</a> <a href="#">Delete</a> <a href="#">Details</a>
12/01/2025 01:37:18	21,98	yassine	<a href="#">Edit</a> <a href="#">Delete</a> <a href="#">Details</a>
12/01/2025 02:01:56	120,00	Imad	<a href="#">Edit</a> <a href="#">Delete</a> <a href="#">Details</a>
13/01/2025 01:35:55	41,98	Imad	<a href="#">Edit</a> <a href="#">Delete</a> <a href="#">Details</a>
13/01/2025 01:39:17	16,99	Karim	<a href="#">Edit</a> <a href="#">Delete</a> <a href="#">Details</a>
13/01/2025 01:39:44	16,99	Karim	<a href="#">Edit</a> <a href="#">Delete</a> <a href="#">Details</a>
13/01/2025 01:40:45	30,00	Imad	<a href="#">Edit</a> <a href="#">Delete</a> <a href="#">Details</a>
13/01/2025 01:41:29	16,99	yassine	<a href="#">Edit</a> <a href="#">Delete</a> <a href="#">Details</a>

FIGURE 3.19 – Page des commandes

```

13
14
15 [SessionCheck]
16 0 références
17 public class CommandesController : Controller
18 {
19     private DB_RestoEntities db = new DB_RestoEntities();
20
21     // GET: Commandes
22     [SessionCheckAdmin]
23 0 références
24     public ActionResult Index()
25     {
26         var commandes = db.Commandes.Include(c => c.User);
27         return View(commandes.ToList());
28     }
29
30     // GET: Commandes/Details/5
31     [SessionCheckAdmin]
32 0 références
33     public ActionResult Details(int? id)
34     {
35         if (id == null)
36         {
37             return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
38         }
39         Commande commande = db.Commandes.Find(id);
40         if (commande == null)
41         {
42             return HttpNotFound();
43         }
44         return View(commande);
45     }

```

FIGURE 3.20 – Commandes controller

## Page des employés

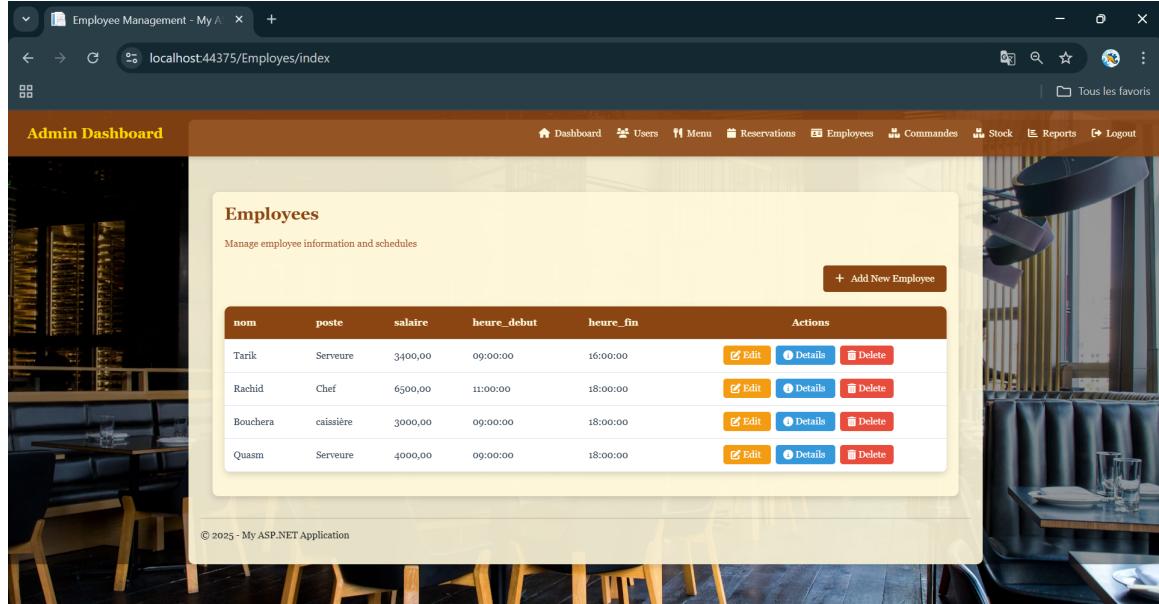


FIGURE 3.21 – Page des employés

```
15 [SessionCheckAdmin]
16 public class EmployesController : Controller
17 {
18     private DB_RestaEntities db = new DB_RestaEntities();
19
20     // GET: Employes
21     public ActionResult Index()
22     {
23         return View(db.Employes.ToList());
24     }
25
26     // GET: Employes/Details/5
27     public ActionResult Details(int? id)
28     {
29         if (id == null)
30         {
31             return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
32         }
33         Employe employe = db.Employes.Find(id);
34         if (employe == null)
35         {
36             return HttpNotFound();
37         }
38         return View(employe);
39     }
40 }
```

FIGURE 3.22 – employe controller

## Page de menu

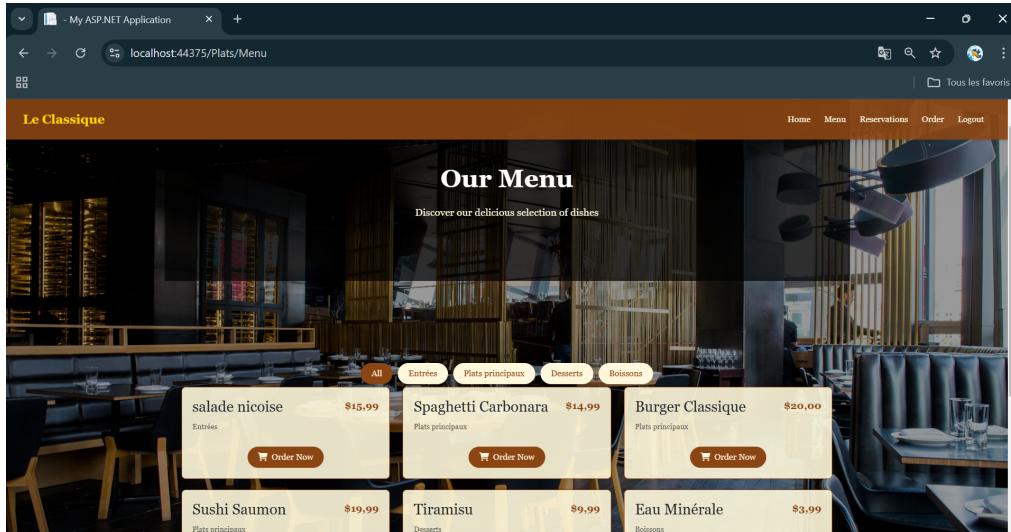


FIGURE 3.23 – Page de menu

```

13  [SessionCheck]
14  public class PlatsController : Controller
15  {
16      private DB_RestoEntities db = new DB_RestoEntities();
17
18      // GET: Plats
19      [SessionCheckAdmin]
20      public ActionResult Index()
21      {
22          var plats = db.Plats.Include(p => p.Category);
23          return View(plats.ToList());
24      }
25
26      // Récupérer le plat depuis la base de données
27      [SessionCheckAdmin]
28      public ActionResult Menu()
29      {
30          var plats = db.Plats.Include(p => p.Category);
31          return View(plats.ToList());
32      }
33
34      [HttpPost]
35      public ActionResult OrderDish(int platId, int qte)
36      {
37          var clientId = Session["UserId"] as int?;
38          if (clientId == null)
39          {
40              return RedirectToAction("Login", "Account");
41          }
42
43          // Récupérer le plat depuis la base de données
44          var plat = db.Plats.SingleOrDefault(p => p.id == platId);
45          if (plat == null)
46          {
47              return HttpNotFound("Plat non trouvé");
48          }
49
50          var prix = plat.prix;
51
52          // Récupérer ou initialiser le panier
53          var cart = Session["Cart"] as List<CartItem> ?? new List<CartItem>();
54
55          // Vérifier si l'article existe déjà dans le panier
56          var existingItem = cart.FirstOrDefault(item => item.PlatId == platId);
57          if (existingItem != null)
58          {
59              // Mettre à jour la quantité
60          }
61
62          // Ajouter le plat au panier
63          CartItem newItem = new CartItem
64          {
65              PlatId = platId,
66              Quantite = qte,
67              Prix = prix
68          };
69
70          cart.Add(newItem);
71
72          Session["Cart"] = cart;
73
74          return RedirectToAction("Index");
75      }
    
```

FIGURE 3.24 – plats controller

### 3.4 Conclusion

Pour conclure cette section consacrée à l'étude pratique de notre projet, nous avons minutieusement analysé les outils de développement utilisés et la séquence des interfaces. Nos éléments sont essentiels afin de garantir la solidité technique et l'ergonomie de notre application . Le choix méticuleux des outils et la planification minutieuse des interfaces représentent deux étapes cruciales pour réaliser notre vision.

# Conclusion générale

En conclusion, notre application web dédiée à la gestion de restaurant représente un grand pas en avant dans la gestion des opérations quotidiennes d'un restaurant. Nous avons développé une plateforme facile à utiliser et sécurisée, permettant aux restaurateurs et au personnel de gérer efficacement les commandes, les stocks et la planification des services. Grâce à cette application, les équipes peuvent travailler de manière plus fluide et offrir un meilleur service aux clients.

Nous avons mis en place des fonctionnalités essentielles telles que la gestion des menus, la prise de commande en ligne et la génération de rapports sur les ventes et les inventaires. Ces outils aident les restaurateurs à suivre en temps réel l'état de leur activité, à optimiser la gestion des stocks et à améliorer l'efficacité des opérations.

Ce projet a mis en lumière l'importance d'une interface simple et intuitive, ainsi que d'une infrastructure solide pour garantir une expérience optimale. Les défis rencontrés tout au long du développement nous ont permis de renforcer nos compétences en développement web, design UX/UI et gestion de projet. Ces expériences ont contribué à notre capacité à proposer des solutions fiables et adaptées aux besoins des professionnels de la restauration.

Nous sommes convaincus que cette application apportera une réelle valeur ajoutée aux restaurants, en leur offrant les outils nécessaires pour améliorer leur gestion et leur efficacité, tout en garantissant une expérience client de qualité.

# Perspectives

Afin de continuer à améliorer notre application de gestion de restaurant, plusieurs évolutions sont envisagées pour enrichir l'expérience utilisateur et répondre aux besoins croissants du secteur.

Tout d'abord, nous prévoyons d'ajouter des méthodes de paiement variées. Cela permettra aux clients de régler leurs commandes en ligne ou sur place via des options comme les cartes bancaires, les paiements mobiles et autres solutions numériques.

Ensuite, nous allons intégrer une barre de recherche pour permettre aux utilisateurs de trouver rapidement les plats ou ingrédients qu'ils recherchent dans le menu, améliorant ainsi la navigation et l'expérience utilisateur.

Nous envisageons également d'inclure des photos pour chaque plat afin de rendre le menu plus attractif et d'aider les clients à faire leur choix plus facilement. Les images enrichiront l'expérience visuelle et faciliteront la prise de décision.

Enfin, nous prévoyons de continuer à améliorer la partie backend de l'application, notamment en développant une architecture plus robuste et évolutive, capable de supporter un plus grand nombre d'utilisateurs et d'offrir une meilleure gestion des données.

Ces améliorations permettront d'optimiser l'efficacité de l'application et de répondre aux attentes des restaurateurs, tout en améliorant l'expérience globale des utilisateurs.

# Webographie

1. Pour entreprise architecte : <https://sparxsystems.fr/>
2. Pour bootstrap : <https://getbootstrap.com/docs/5.3/getting-started/introduction/>
3. Pour l'éditeur VS : <https://visualstudio.microsoft.com/fr/>
4. Pour w3schools (guide por HTML/CSS/Javascript) : <https://www.w3schools.com/>
5. Pour ASP.NET core : <https://dotnet.microsoft.com/en-us/apps/aspnet>
6. Pour C sharp : <https://learn.microsoft.com/fr-fr/dotnet/csharp/tour-of-csharp/>