Titre professionnel Développeur Web et Web Mobile Niveau III

Dossier de projet

La Casa De Havrais

*« Le restaurant où se poser»*

JEAN LAURA

Remerciements

Tout d’abord, je tiens à remercier Anousone qui nous a encouragés tout au long de notre parcours pour nous dépasser et faire le mieux que l’on pouvait. Il a été derrière nous tous les jours pour savoir comment nous avancions, nous aider quand c’était nécessaire et surtout être à l’écoute de chacun pour nous rassurer tout du long. Grand merci à lui.

Ensuite, je tiens à remercier mes camarades pour avoir été patients et m’avoir aidé quand cela a été nécessaire. Particulièrement, Julien FOLLET, Yassine BENHAMOUDA, Mehdi ROUIS, Jonathan LEBOURG, aussi bien pour le back que le front. Grand merci à eux.

Je remercie également le fait qu’Anousone m’ait permis de prendre mon ordinateur portable personnel pour avoir mes logiciels et ainsi mieux travailler tout en continuant à faire la même chose que sur les pc fournis avec linux.

Résumé de projet

LA CASA DE HAVRAIS

La Casa De Havrais est un restaurant gastronomique de spécialité espagnol. J’ai eu l’idée de faire ce projet parce que mon petit ami est second de cuisine dans un restaurant gastronomique italien sur le Havre et il aimerait monter son restaurant dans quelques années. Pour ce faire, je lui propose une maquette de site pour qu’il ait une idée. Le concept principal du site est de pouvoir se renseigner sur l’ambiance du restaurant, voir la carte, apprécier les différents menus et pouvoir réserver une table à n’importe quel moment de la journée.

Dans le cadre du projet, j’ai dû choisir quelles fonctionnalités peuvent fidéliser le client et mettre en avant le restaurant car il n’y a pas de concept équivalent au Havre en ce moment.

De ce fait, les fonctionnalités sont :

* pouvoir mettre des plats dans le panier pour la vente à emporter,
* réserver une table à la place de son choix sur internet et par téléphone si plus de 10 personnes,
* le back office pour permettre au restaurateur d’ajouter un plat, de le modifier, de le supprimer mais également de pouvoir dire si le client qui a réservé une table est bien là ou pas.

Table des matières

[1.Cahier des charges 5](#_Toc1727650)

[1.1 Descriptif de la demande 5](#_Toc1727651)

[1.2 Architecture technique 5](#_Toc1727652)

[1.3 Spécifications fonctionnelles 6](#_Toc1727653)

[a) Description 6](#_Toc1727654)

[b) Utilisation 7](#_Toc1727655)

[1.4 Spécifications techniques 9](#_Toc1727656)

[a) Langages utilisés 9](#_Toc1727657)

[b) Bibliothèques et frameworks utilisés 9](#_Toc1727658)

[c) Outils utilisés 9](#_Toc1727659)

[d) Sécurité des systèmes d’information : Recommandations et bonnes pratiques 10](#_Toc1727660)

[2. Compétences du référentiel couvertes 11](#_Toc1727661)

[2.1 Tableau récapitulatif 11](#_Toc1727662)

[2.2 Maquetter une application 11](#_Toc1727663)

[2.3 Réaliser une interface utilisateur web statique et adaptable 13](#_Toc1727664)

[2.4 Développer une interface utilisateur web dynamique 14](#_Toc1727665)

[2.5 Développer une interface utilisateur avec une solution de gestion de contenu 14](#_Toc1727666)

[2.6 Créer une base de données 16](#_Toc1727667)

[2.7 Développer des composants d’accès aux données 19](#_Toc1727668)

[2.8 Développer la partie back-end d’une application web ou web mobile 20](#_Toc1727669)

[3. Réalisation 22](#_Toc1727670)

[3.1 Création d’un enregistrement qualité 22](#_Toc1727671)

[3.2 Affichage d’un plan d’action 27](#_Toc1727672)

[3.3 Modification d’un plan d’action 28](#_Toc1727673)

[3.4 Suppression d’un enregistrement qualité 30](#_Toc1727674)

[4. Anglais de rigueur : Une communauté importante orientée vers le partage 33](#_Toc1727675)

[5. Conclusion 33](#_Toc1727676)

# 1.Cahier des charges

## 1.1 Descriptif de la demande

L’application web doit permettre de réserver des tables dans le restaurant directement grâce à un formulaire qui ira jusqu’à 6 personnes. Au-delà, il faudra appeler le restaurant qui fera la réservation eux-mêmes grâce au formulaire dans la partie administration.

Elle permettra également d’insérer, modifier et supprimer des plats de la carte. Mais aussi de gérer les commandes de vente à emporter des clients. De plus, il y aura une partie recherche dans la partie administration pour les plats, les commandes et les réservations qui permettront de les retrouver plus facilement.

L’objectif étant de créer une interface intuitive, simple et rapide pour optimiser l’enregistrement des données relatives à la réservation, aux plats et aux commandes.

## 1.2 Architecture technique

L’application doit être accessible sur tous les types de navigateurs. Elle est principalement destinée à être déployée sur un serveur LAMP et/ou WAMP.

Le développement utilisera des langages orientés WEB : HTML, CSS, JavaScript et PHP. Il pourra également contenir différents FRAMEWORK tels que Jquery et Material Design for Bootstrap.

L’application sera responsive et structurée d’après l’architecture de développement Modèle-Vue-Contrôleur.

Les données seront stockées sur une base de données MySQL.

## 1.3 Spécifications fonctionnelles

### a) Description

Une réservation (lcdh\_reservation) est défini par :

* Un nom
* Un prénom
* Un numéro de téléphone
* Une adresse mail
* Une confirmation de mail
* Un sms de confirmation
* Une date de réservation
* Une heure de réservation
* Un nombre de personne
* Une heure d’arrivée
* Le numéro de table *(ciblé dans à l’aide de la table lcdh\_tables)*

Une table (lcdh\_tables) est défini par :

* Le numéro de table

Un client (lcdh\_customers) est défini par :

* Un nom
* Un prénom
* Un mail
* Un numéro de téléphone
* Une heure d’arrivée

Un plat (lcdh\_dishes) est défini par :

* Un nom
* Une description
* Un prix
* Une catégorie
* Une sous-catégorie

Une catégorie (lcdh\_categories) est défini par :

* Un nom

Une sous-catégorie (lcdh\_subCategories) est défini par :

* Un nom

Une carte (lcdh\_card) est défini par :

* Un nom

### b) Utilisation

L’utilisation de l’application respecte une hiérarchie de 2 niveaux :

* Restaurateur (Administrateur)
* Client (Client)

Création de réservation

L’application permet d’ajouter des réservations dans la base de données. Chaque réservation dispose d’une modification et d’une suppression disponible que par l’administrateur.

Aussi, à plus de 6 personnes dans la réservation, les clients sont obligés d’appeler le restaurateur pour qu’il puisse l’ajouter lui-même.

Affichages des informations de la réservation

Chaque client recevra un mail de confirmation à chaque réservation effectué avec leurs informations rappelées à l’intérieur.

L’administrateur à un espace dédié pour afficher la liste de toutes les réservations.

Modification des informations de la réservation

Seul l’administrateur est autorisé à modifier les réservations, si le client se rend compte qu’il s’est trompé, il faudra donc qu’il appelle en donnant soit son nom ou son numéro de réservation présent dans le mail pour permettre la modification.

Suppression de la réservation

L’application permettra de supprimer des réservations mais uniquement via l’espace administrateur. Le but est de contrôler les réservations.

Création / Affichage / Modification / Suppression d’une commande

Seul l’administrateur aura la main sur la gestion des commandes. Il aura donc pour rôle de donner les commandes au chef et de les avoirs en temps et en heure que le client aura indiqué.

Création d’un plat

Seul l’administrateur aura la possibilité de créer des plats à mettre dans la carte grâce à un formulaire présent dans l’espace dédié à l’administrateur.

Affichage d’un plat

L’application permettra à l’administrateur de voir les différents plats rangers suivant les catégories et sous-catégories pour pouvoir ensuite les modifier et supprimer.

Modification d’un plat

La modification se fait toujours par l’administrateur grâce à la liste qui regroupe tous les plats.

Suppression d’un enregistrement qualité

Tout comme la modification, seul l’administrateur aura la possibilité de supprimer les plats.

## 1.4 Spécifications techniques

### Langages utilisés

|  |  |
| --- | --- |
| Appellation | Description |
| HTML 5 | HTML5 (HyperText Markup Language 5) est la dernière révision majeure du HTML. C’est un format de données conçu pour représenter les pages web. C’est un langage de balisage permettant de l’hypertexte. |
| CSS 3 | Les feuilles de style en cascade, généralement appelées CSS de l'anglais Cascading Style Sheets, forment un langage informatique qui décrit la présentation des documents HTML et XML. |
| PHP 7 | PHP: Hypertext Preprocessor, est un langage de programmation libre, principalement utilisé pour produire des pages Web dynamiques via un serveur HTTP, mais pouvant également fonctionner comme n'importe quel langage interprété de façon locale. PHP est un langage impératif orienté objet. |
| JavaScript | JavaScript est un langage événementiel de programmation de scripts, principalement employé dans les pages web interactives. |
| MySQL | Langage de requête fonctionnant avec le système de gestion des bases de données relationnelles. Il permet la génération de la base de données, la lecture et la modification de ses tables. |

### Bibliothèques et frameworks utilisés

|  |  |
| --- | --- |
| Appellation | Description |
| Material Design for Bootstrap | Material Design for Bootstrap est un framework front-end adaptatif et moderne basé sur le Material Design. Son système de grille permet de rendre aisément une application web responsive. |
| JQuery | jQuery est une bibliothèque JavaScript open source et multiplateforme créée pour faciliter l'écriture de scripts côté client dans le code HTML des pages web. |

### Outils utilisés

|  |  |
| --- | --- |
| Appellation | Description |
| NetBeans | NetBeans est un environnement de développement intégré (EDI), placé en open source. En plus de Java, NetBeans permet la prise en charge native de divers langages tels le JavaScript, le XML, le PHP et le HTML. |
| Git | Git est un logiciel de gestion de versions décentralisé. Il permet de travailler avec des services d’hébergement spécifiques tel que GitHub et BitBucket. |
| Balsamiq | Balsamiq est un logiciel payant qui appartient à Adobe de création de maquettes typographiques. Il est utilisé afin de créer des diagrammes et des maquettes d'interface graphique de logiciels. |
| phpMyAdmin | phpMyAdmin (PMA) est une application Web de gestion pour les systèmes de gestion de base de données MySQL réalisée principalement en PHP. |
| JMerise | Logiciel d’aide à la conception des bases de données (Modèle Conceptuel de Données) qui permet la génération d’un Modèle Logique de Données et d’un script SQL pour plus de facilité. |

### Sécurité des systèmes d’information : Recommandations et bonnes pratiques

Pour sécuriser les informations qui sont entrées dans la base de données, il y a des regexs pour contrôler la saisie que les utilisateurs mettent dans les inputs.

En ce qui concerne la partie back-end, seul les administrateurs y ont accès. Une page de connexion est pour cela nécessaire et primordial. Un mot de passe est demandé quand on arrive sur la page, il prend en compte les différents règles mis en ligne par la CNIL pour éviter de se faire hacker. Caractère spéciaux, chiffres, lettres en majuscule et minuscule sont de vigueur.

# 2. Compétences du référentiel couvertes

## 2.1 Tableau récapitulatif



Légendes :

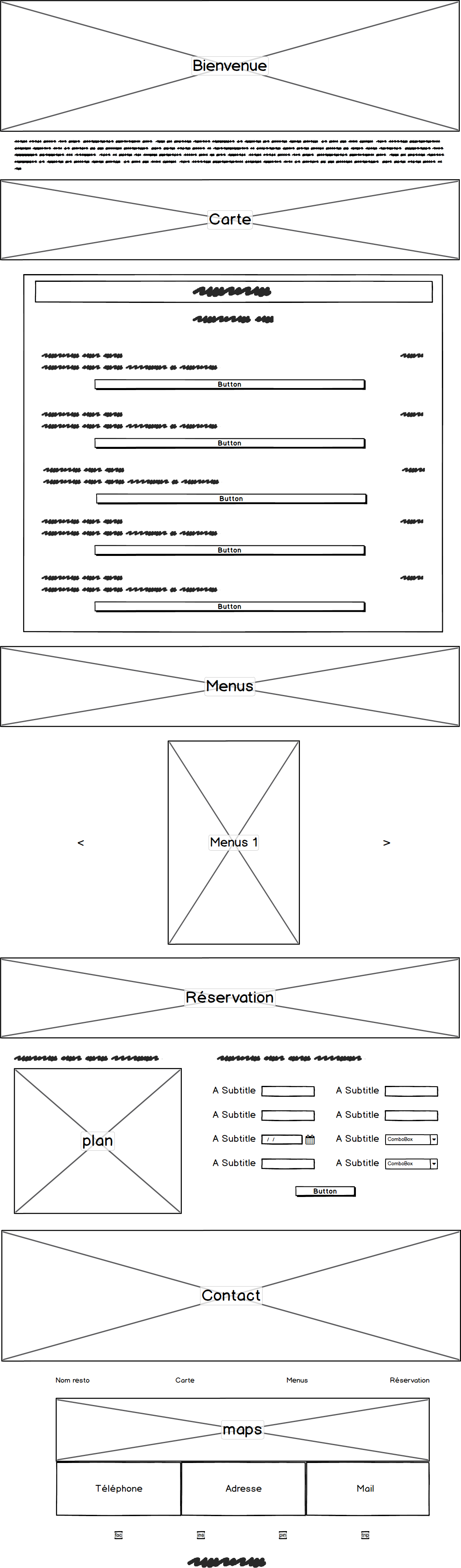
|  |
| --- |
| Compétences couvertes |
| Compétences non couvertes |

## 2.2 Maquetter une application

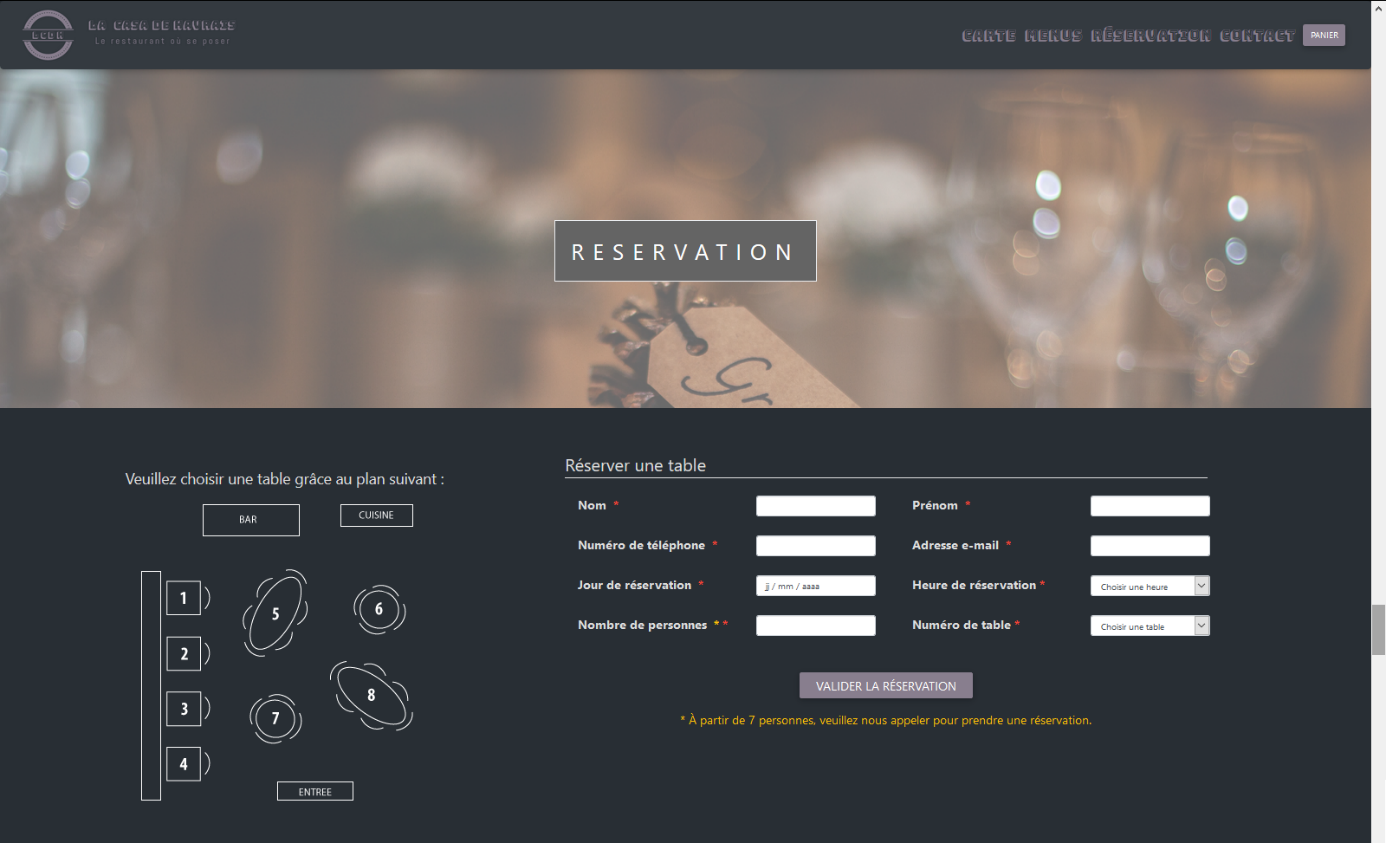
Une fois le cahier des charges défini, j’ai commencé mon projet par l’élaboration d’une maquette qui me permettait d’avoir un visuel des différents éléments et des différentes fonctions de l’application. Le but de mon maquettage était de matérialiser les différentes idées et avoir un fil conducteur tant sur le plan visuel que conceptuel.

J’ai alors utilisé le logiciel « Balsamiq » qui permet des créer rapidement une interface et de créer une rapide navigation de la future application.

Version maquettée sous Balsamiq



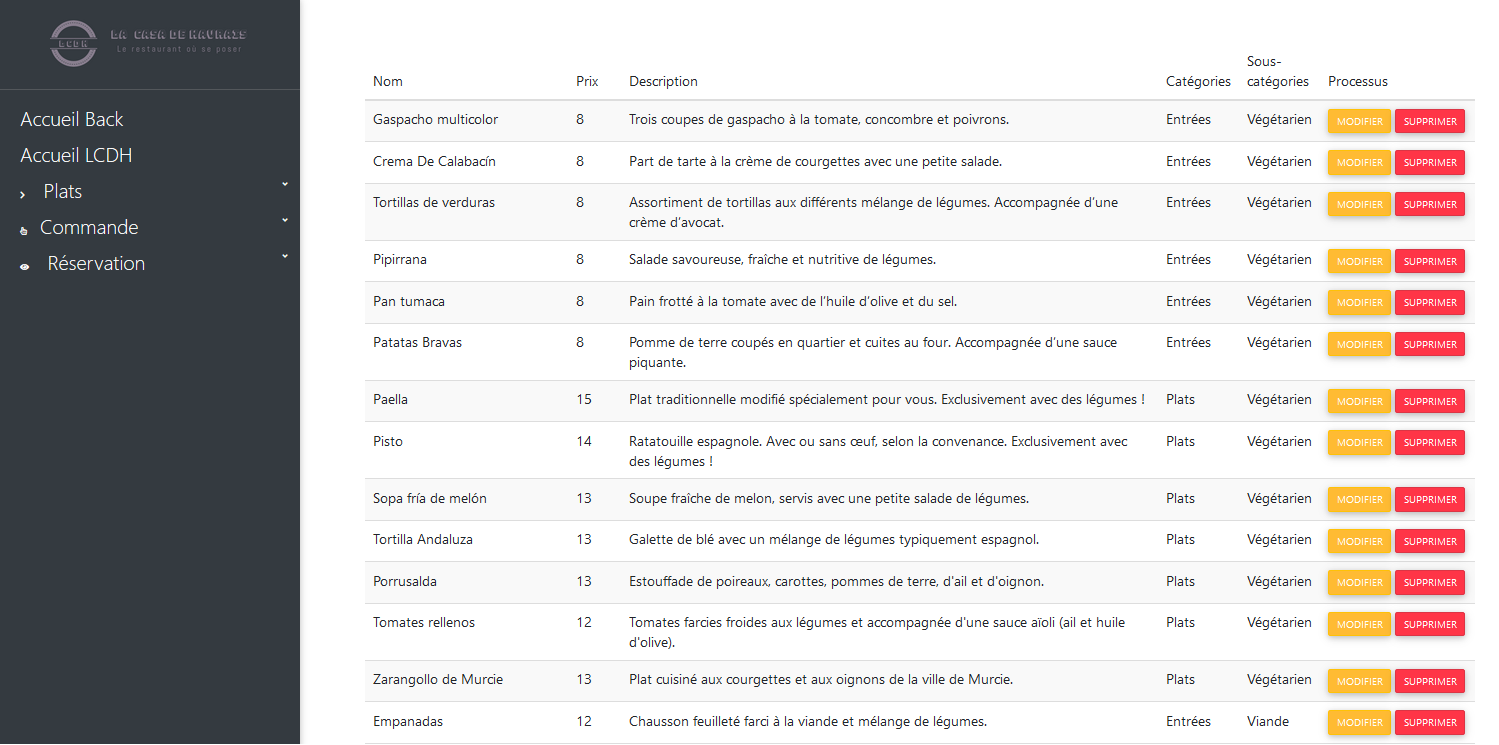
Version finale de l’interface dans l’application web, globalement très peu de changements



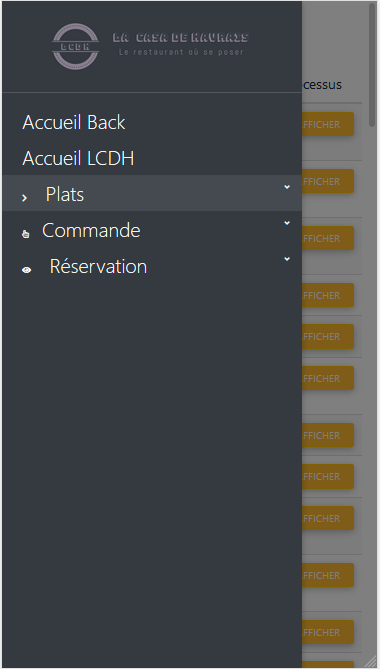
## 2.3 Réaliser une interface utilisateur web statique et adaptable

Afin de gagner de la mobilité, j’ai fait en sorte que mon application soit responsive en m’appuyant sur un framework appelé Material Design for Bootstrap. Il m’a permis d’utiliser un système de grille pour une meilleure structure ainsi que des classes spéciales pour que mon application puisse s’adapter parfaitement aux différents supports et tailles d’écran.

Ex : Vue administrateur en grande résolution

**

Vue administrateur sur la résolution d’un mobile : La « navbar » s’est transformée en « burger menu » et la navigation s’est adaptée au support

**

## 2.4 Développer une interface administrateur web dynamique

* Menu sur la gauche quand on se connecte à la partie administrateur
* Accès aux ajouts, listes, mises à jour, suppression et recherches des données
* Permets d’accéder au site vu par les clients facilement grâce à un raccourci dans le menu

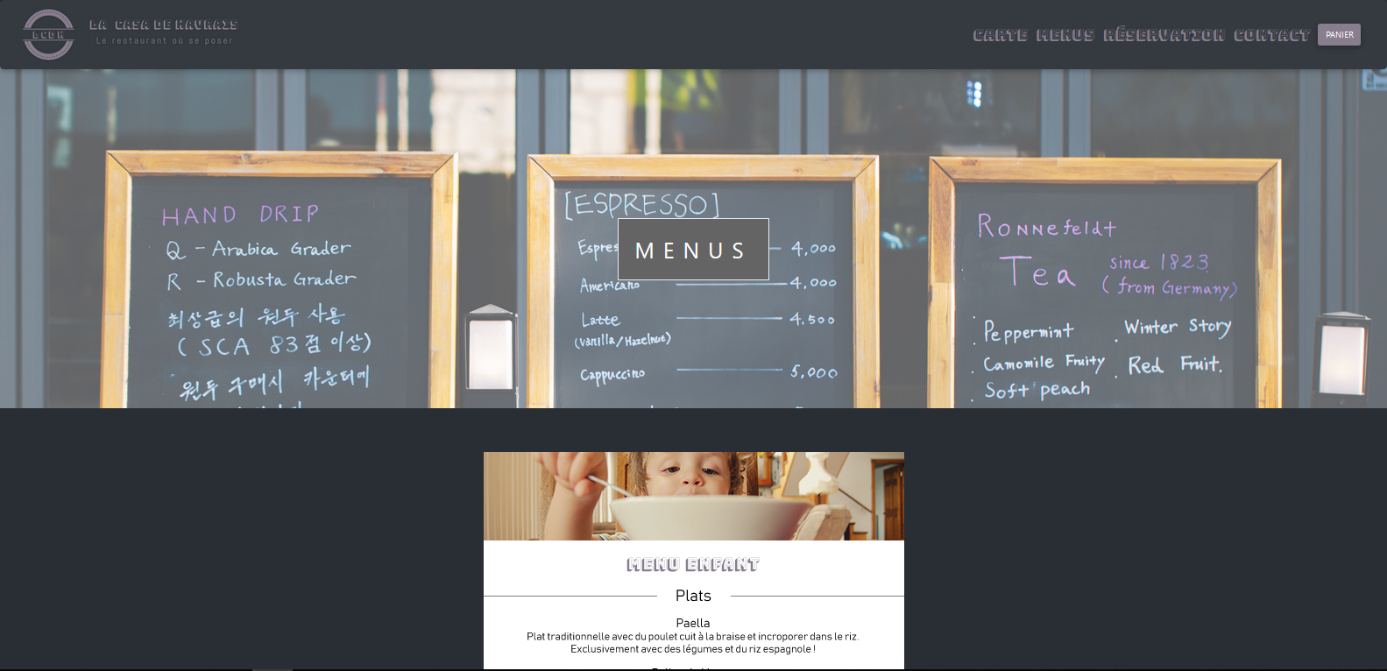
## 2.5 Développer une interface utilisateur avec une solution de gestion de contenu

Ayant opté pour une architecture Modèle-Vue-Contrôleur (MVC), une interface administrateur était indispensable pour afficher les données de mes modèles via mes différents contrôleurs. Ce type d’architecture permet également de faciliter la Programmation Orientée Objet (POO).

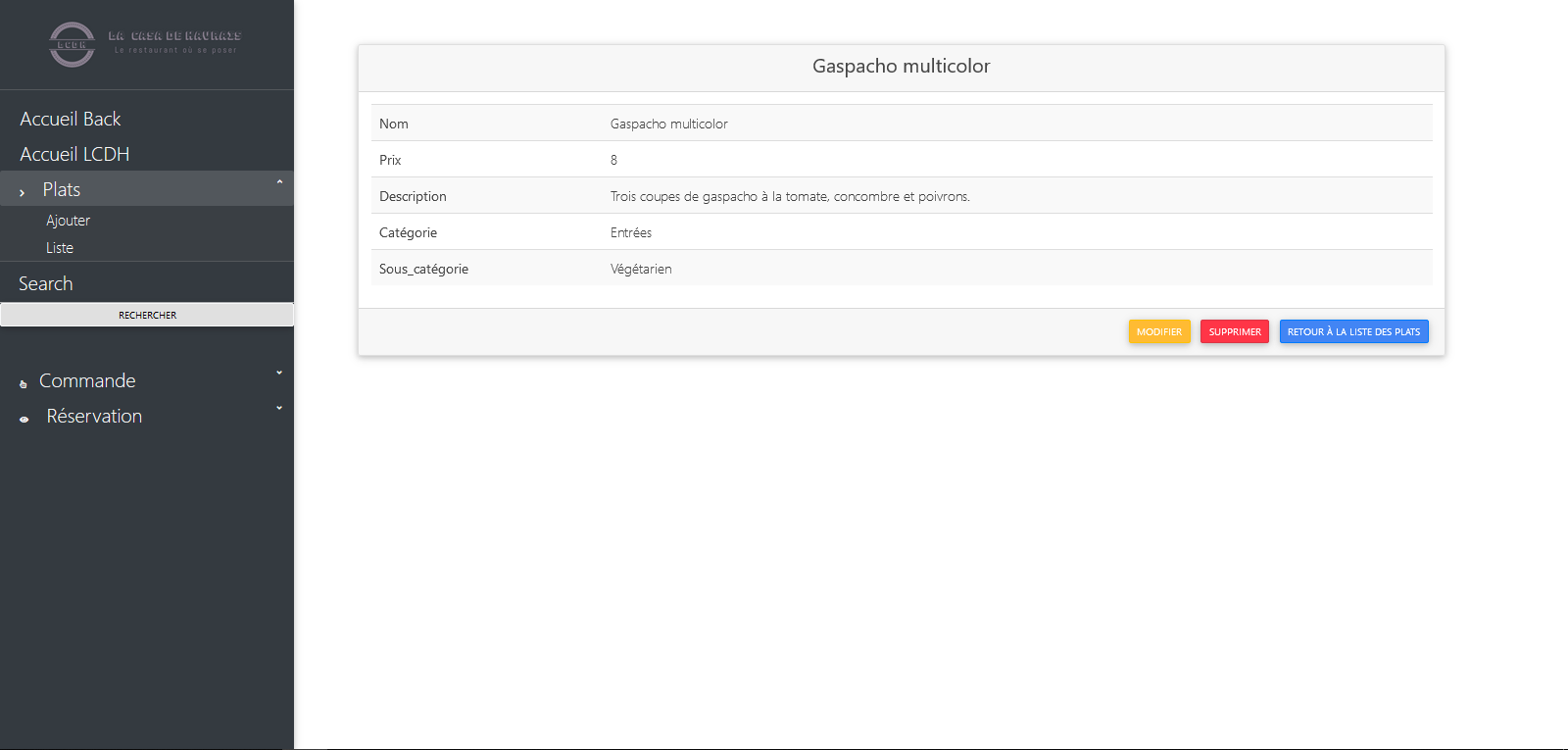
J’ai donc maquetté l’ensemble de mon application en suivant ce leitmotiv : « Simple, clair et intuitive ».

Ex : L’application web permet d’accéder aux différentes parties (administrateur et clients) pour s’adapter aux profils.

Profil clients :



Profil administrateur :

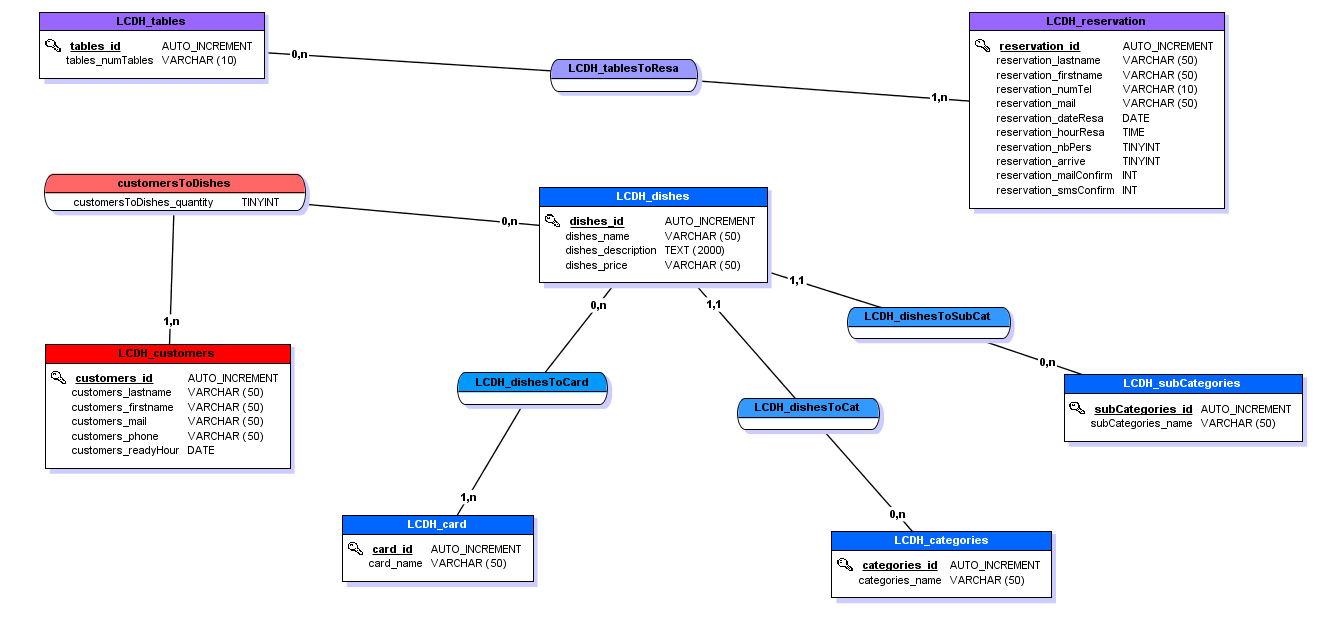


## 2.6 Créer une base de données

Une fois les différentes fonctionnalités définies et le maquettage terminé, il a été primordiale d’établir la base de données car c’est elle qui stockera toutes les données de l’application.

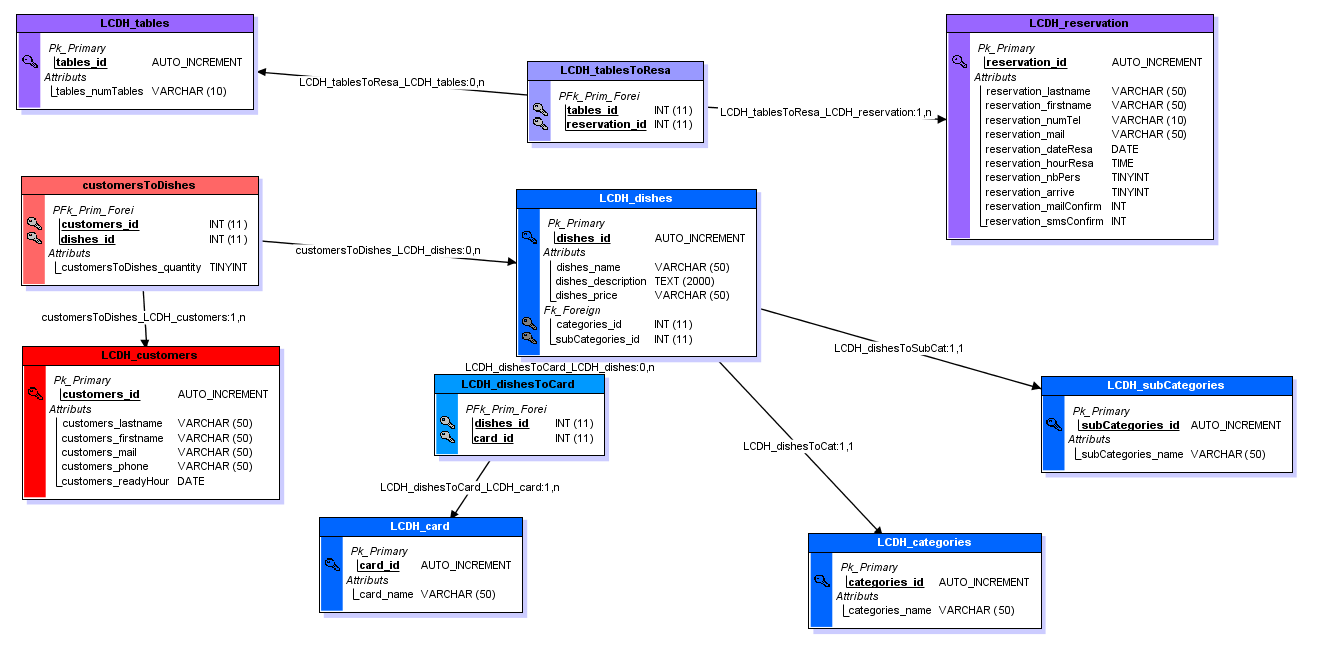
J’ai donc utilisé le logiciel JMerise pour modéliser mes tables, mes champs ainsi que mes cardinalités. Dans un premier temps, j’ai obtenu un MCD (Modèle Conceptuel de Données) que j’ai modifié selon mes besoins tout au long de mon projet.

MCD :

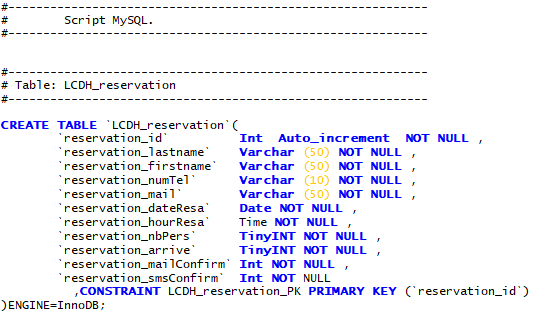
**

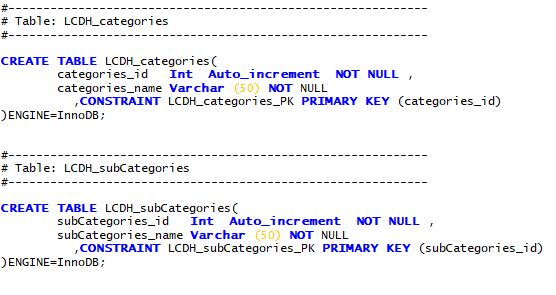
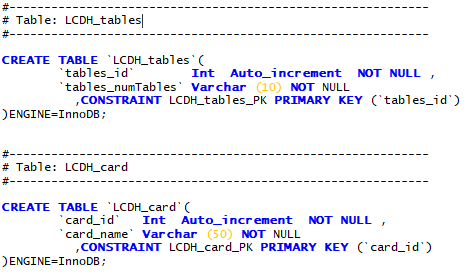
Puis dans un second temps, une fois mon MCD modélisé, j’ai utilisé une fonctionnalité de JMerise pour générer automatiquement mon MLD : Modèle Logique de Données. L’outil se base sur les clefs primaires et les cardinalités renseignées pour créer les clefs étrangères dans les différentes tables.

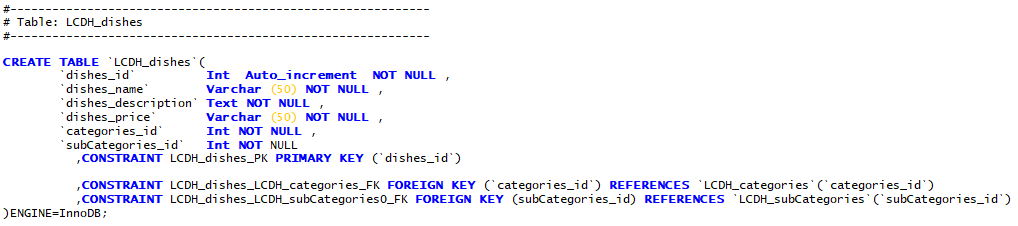
MLD :

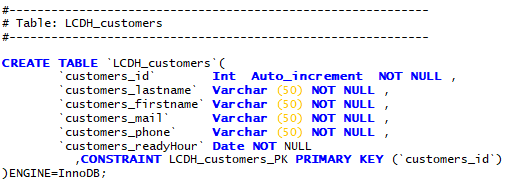


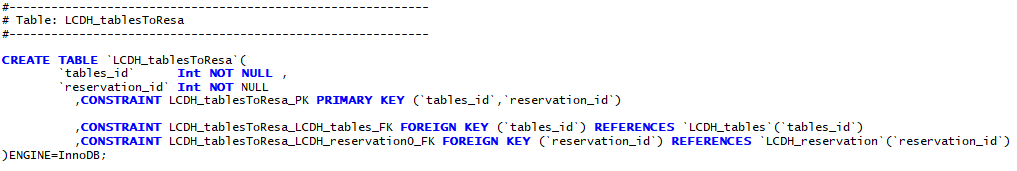
Une fois mon MCD et mon MLD de créés, je me suis appuyée sur une fonction de JMerise pour me générer automatiquement le script SQL correspondant à mon merise.

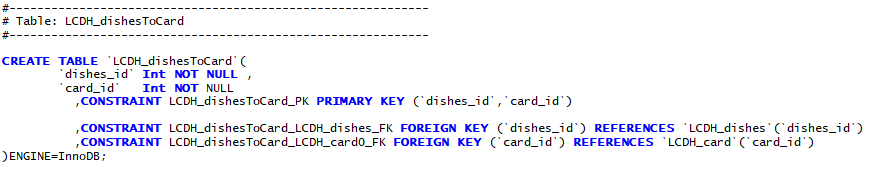


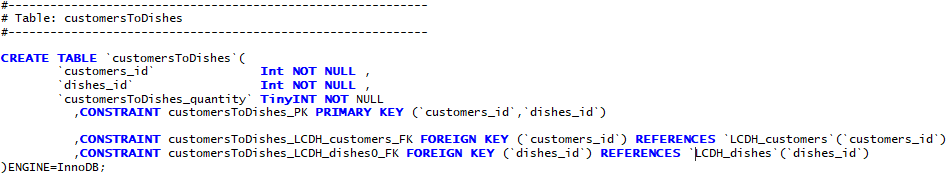












## 2.7 Développer des composants d’accès aux données

Dans une architecture MVC, chaque modèle représente une table de la base de données. Ils doivent pouvoir s’y connecter à chaque exécution d’une méthode afin d’accéder aux différentes données.

J’ai donc créé une classe spécifique regroupant tous les éléments nécessaires pour se connecter à la base de données.

**<?php**

/\*\*

\* Classe permettant la connexion avec la base de données

\* @return Methods

\*/

class database {

public $database;

// Ce qui va être exécuté en premier lors de l'instanciation de la classe

public function \_\_construct() {

try {

// variables stockant les paramètres de connexions à la base

$db\_config = array();

$db\_config['SGBD'] = 'mysql';

$db\_config['HOST'] = 'localhost'; // le chemin vers le serveur

$db\_config['DB\_NAME'] = 'resto\_lacasadehavrais'; // le nom de votre base de données

$db\_config['USER'] = 'root'; // nom d'utilisateur pour se connecter

$db\_config['PASSWORD'] = ''; // mot de passe de l'utilisateur pour se connecter

$db\_config['OPTIONS'] = array(

PDO::ATTR\_ERRMODE => PDO::ERRMODE\_EXCEPTION,

PDO::ATTR\_ERRMODE => PDO::ERRMODE\_WARNING

); // Affiche les erreurs SQL

// Mise en relation avec la base de donnée

$this->database = new PDO($db\_config['SGBD'].':host='.$db\_config['HOST'].';dbname='.$db\_config['DB\_NAME'].';charset=UTF8', $db\_config['USER'], $db\_config['PASSWORD']);

unset($db\_config);

} catch (Exception $error) {

// récupération des erreurs

die('Erreur : '.$error->getMessage());

}

}

// Ce qui va être exécuté en dernier lors de l'instanciation de la classe

public function \_\_destruct() {

$this->database = NULL;

}

}

**?>**

La classe « database » sera commune à toutes les autres classes nécessitant une connexion à la base de données, ex : « reservation »

**<?php**

class reservation extends database {

// champs de la table patients

public $reservation\_id;

public $reservation\_lastname;

public $reservation\_firstname;

public $reservation\_numTel;

public $reservation\_mail;

public $reservation\_dateResa;

public $reservation\_hourResa;

public $reservation\_nbPers;

public $reservation\_arrive;

public $tables\_id;

public $reservation\_mailConfirm;

public $reservation\_smsConfirm;

**?>**

## 2.8 Développer la partie back-end d’une application web ou web mobile

Comme indiqué précédemment, dans une architecture MVC, un Modèle correspond à une table de la base de données. Dans notre application, les Modèles utilisent la classe « database » pour se connecter et disposent de diverses méthodes pour créer, lire, modifier et supprimer les données respectives.

Pour pouvoir les utiliser, les Modèles sont directement instanciés dans les Contrôleurs des différentes Vue.

En effet, les Vues permettent d’afficher des données et interagir avec les clients. Pour cela, chaque Vue dispose d’un contrôleur respectif.

*Ex : Affichage d’une liste de clients (customers) selon les données de la table reservation.*

**Vue** : On inclut les Modèles et le Contrôleur via un « require\_once»

**<?php**

require\_once '../controllers/controllerSupprimer-reservation.php';

include 'header\_back.php';

**?>**

<!-- CONTENT PAGE -->

<div class="content-wrap marginTop" id="margSuppResa">

<div class="container-fluid">

<div class="row">

**<?php**

if ($success){

**?>**

<p id="pResaSupp">Votre réservation à bien été supprimé !</p>

</div>

<div class="row">

<div class="col-md-6" id="suppResaDiv6">

<div class="col-md-4" id="suppResaDiv4">

<a href="listReservation.php" class="btn btn-primary btnResaBack">Retour à la liste des réservation</a>

</div>

<div class="col-md-4" align="center">

<a href="index.php" class="btn btn-default btnResaBack">Accueil</a>

</div>

<div class="col-md-4">

<a href="ajout-reservation.php" class="btn btn-info btnResaBack">Ajouter une réservation</a>

</div>

</div>

**<?php**

}else{

echo 'Dommage pour toi, ça n\'a pas fonctionné.';

}

**?>**

</div>

</div>

</div>

**<?php** include 'footer\_back.php'; **?>**

**Contrôleur** : J’instancie un nouvel objet à l’aide du Modèle respectif et j’appelle les méthodes, ici je l’appel la méthode « deleteResa (). »

**<?php**

require\_once '../models/database.php';

require\_once '../models/modelResa.php';

// Instanciation de l'objet Réservation contenant les méthodes utilisées

$resaOBJ = new reservation();

$resa = $resaOBJ->listResa(); //Tableau qui reprend la liste des réservations

$success = true;

if (isset($\_GET['idResa'])) {

$resaOBJ->reservation\_id = $\_GET['idResa']; //Récupère id initialisé comme idResa

if($resaOBJ->deleteResa()){//Suppression de la réservation

$success;

}

}

**?>**

**Modèle** : Connexion à la base de données via la classe « database » + création d’une méthode pour effectuer une requête dans la table « reservation »

**<?php**

class reservation extends database {

// champs de la table patients

public $reservation\_id;

public $reservation\_lastname;

public $reservation\_firstname;

public $reservation\_numTel;

public $reservation\_mail;

public $reservation\_dateResa;

public $reservation\_hourResa;

public $reservation\_nbPers;

public $reservation\_arrive;

public $tables\_id;

public $reservation\_mailConfirm;

public $reservation\_smsConfirm;

/\*\*

\* Méthode permettant d'ajouter une réservation

\* @return exécute la requête pour ajouter une réservation

\*/

//Suppression d'une réservation

function deleteResa() {

$sql = $this->database->prepare('DELETE FROM `lcdh\_reservation` WHERE `reservation\_id` = :idResa');

$sql->bindValue(':idResa', $this->reservation\_id, PDO::PARAM\_INT);

return $sql->execute();

}

}

**?>**

# 3. Réalisation

## 3.1 Création d’une réservation

Création d’une Vue avec un formulaire pour recueillir les différentes informations



**<?php**

session\_start();

$\_SESSION['totalCart'] = count($\_SESSION['cart']);

require 'controllers/controllerListe-plats.php';

require 'controllers/controller-reservation.php';

require 'header.php';

**?>**

<div class="bgimg-3">

<div class="caption">

<span class="border bgimg-3-reservation">RESERVATION</span>

</div>

</div>

<div id="divReservation">

<div id="reservation">

<div class="container-fluid">

<div class="row" id="pageWebOrdinateur">

<div class="col-12 col-md-6" align="center" id="planWidth">

<h3 id="pPlanResa">Veuillez choisir une table grâce au plan suivant :</h3>

<img class="imgPlan img-responsive" id="imgPlanResa" src="img/plan\_resto2.png" alt="">

</div>

<div class="col-12 col-md-6" id="formWidth">

**<?php** if ($addResaSuccess){

**?>**

<p>Réservation n°**<?=** $reservation->reservation\_id **?>** enregistrer au nom de **<?=** $reservation->reservation\_lastname **?>** **<?=** $reservation->reservation\_firstname **?>** le **<?=** $reservation->reservation\_dateResa **?>** à **<?=** $reservation->reservation\_hourResa **?>**.</p>

<p>

Un mail de confirmation vous a été envoyés à l'adresse que vous avez indiqué : **<?=** $reservation->reservation\_mail **?>**.

</p>

<p>

Nous pouvons vous joindre au numéro suivant : **<?=** $reservation->reservation\_numTel **?>**.

</p>

<p>

À bientôt dans notre restaurant.

</p>

**<?php**

} else { **?>**

<form class="formResa" id="formResa" name="formResa" method="POST" action="">

<fieldset>

<legend>Réserver une table</legend>

<div class="row" id="resaLigne1">

<p>

<div class="col-12 col-md-6">

<div class="col-md-6" id="labelLastnameWidth">

<label for="inputLastnameResa">Nom&nbsp;

<span class="red-text">\* **<?=** isset($arrayError['lastnameResaErr']) ? $arrayError['lastnameResaErr'] : ''; **?>**</span>

</label>

</div>

<div class="col-md-6" id="inputLastnameWidth">

<input type="text" name="inputLastnameResa" class="form-control" id="inputLastnameResa" value="**<?=** count($arrayError) != 0 ? $resaOBJ->reservation\_lastname : ''; **?>**">

</div>

</div>

<div class="col-12 col-md-6">

<div class="col-12 col-md-6" id="labelFirstnameWidth">

<label for="inputFirstnameResa">Prénom&nbsp;

<span class="red-text">\* **<?=** isset($arrayError['firstnameResaErr']) ? $arrayError['firstnameResaErr'] : ''; **?>**</span>

</label>

</div>

<div class="col-12 col-md-6" id="inputFirstnameWidth">

<input type="text" name="inputFirstnameResa" class="form-control" id="inputFirstnameResa" value="**<?=** count($arrayError) != 0 ? $resaOBJ->reservation\_firstname : ''; **?>**">

</div>

</div>

</p>

</div>

<div class="row" id="resaLigne2">

<p>

<div class="col-12 col-md-6">

<div class="col-12 col-md-6" id="labelTelResaWidth">

<label for="inputTelResa">Numéro&nbsp;de&nbsp;téléphone&nbsp;

<span class="red-text">\* **<?=** isset($arrayError['telResaErr']) ? $arrayError['telResaErr'] : ''; **?>**</span>

</label>

</div>

<div class="col-12 col-md-6" id="inputTelResaWidth">

<input type="text" name="inputTelResa" class="form-control" id="inputTelResa" value="**<?=** count($arrayError) != 0 ? $resaOBJ->reservation\_numTel : ''; **?>**">

</div>

</div>

<div class="col-12 col-md-6">

<div class="col-12 col-md-6" id="labelMailWidth">

<label for="inputMailResa">Adresse&nbsp;e-mail&nbsp;

<span class="red-text">\* **<?=** isset($arrayError['mailResaErr']) ? $arrayError['mailResaErr'] : ''; **?>**</span>

</label>

</div>

<div class="col-12 col-md-6" id="inputMailWidth">

<input type="text" name="inputMailResa" class="form-control" id="inputMailResa" value="**<?=** count($arrayError) != 0 ? $resaOBJ->reservation\_mail : ''; **?>**">

</div>

</div>

</p>

</div>

<div class="row" id="resaLigne3">

<p>

<div class="col-12 col-md-6">

<div class="col-12 col-md-6" id="labelDateResaWidth">

<label for="inputDateResa">Jour&nbsp;de&nbsp;réservation&nbsp;

<span class="red-text">\* **<?=** isset($arrayError['dateResaErr']) ? $arrayError['dateResaErr'] : ''; **?>**</span>

</label>

</div>

<div class="col-12 col-md-6" id="inputDateResaWidth">

<input class="form-control" type="date" name="inputDateResa" id="inputDateResa" min="**<?=** $today **?>**" max="**<?=** $oneDateLater **?>**" value="**<?=** isset($\_POST['inputDateResa']) ? $\_POST['inputDateResa'] : '' **?>**" />

</div>

</div>

<div class="col-12 col-md-6">

<div class="col-12 col-md-6" id="labelHourResaWidth">

<label for="selectHourResa">Heure&nbsp;de&nbsp;réservation

<span class="red-text">\* **<?=** isset($arrayError['hourResaErr']) ? $arrayError['hourResaErr'] : ''; **?>**</span>

</label>

</div>

<div class="col-12 col-md-6" id="selectHourResaWidth">

<select class="form-control" id="selectHourResa" name="selectHourResa">

<option value="" disabled selected>Choisir une heure</option>

<option value="MIDI" disabled>MIDI</option>

<option value="11:30:00">11h30</option>

<option value="12:00:00">12h00</option>

<option value="12:30:00">12h30</option>

<option value="13:00:00">13h00</option>

<option value="13:30:00">13h30</option>

<option value="14:00:00">14h00</option>

<option value="14:30:00">14h30</option>

<option value="15:00:00">15h00</option>

<option value="SOIR" disabled>SOIR</option>

<option value="18:00:00">18h00</option>

<option value="18:30:00">18h30</option>

<option value="19:00:00">19h00</option>

<option value="19:30:00">19h30</option>

<option value="20:00:00">20h00</option>

<option value="20:30:00">20h30</option>

<option value="21:00:00">21h00</option>

<option value="21:30:00">21h30</option>

<option value="22:00:00">22h00</option>

</select>

</div>

</div>

</p>

</div>

<div class="row" id="resaLigne4">

<p>

<div class="col-12 col-md-6">

<div class="col-12 col-md-6" id="labelNbPersResaWidth">

<label for="inputNbPersResa">Nombre&nbsp;de&nbsp;personnes&nbsp;

<span class="amber-text">\*</span>

<span class="red-text">\* **<?=** isset($arrayError['nbPersResaErr']) ? $arrayError['nbPersResaErr'] : ''; **?>**</span>

</label>

</div>

<div class="col-12 col-md-6" id="inputNbPersResaWidth">

<input type="text" name="inputNbPersResa" class="form-control" id="inputNbPersResa" value="**<?=** count($arrayError) != 0 ? $resaOBJ->reservation\_hourResa : ''; **?>**">

</div>

</div>

<div class="col-12 col-md-6">

<div class="col-12 col-md-6" id="labelNumTableWidth">

<label for="selectNumTable">Numéro&nbsp;de&nbsp;table

<span class="red-text">\* **<?=** isset($arrayError['numTableErr']) ? $arrayError['numTableErr'] : ''; **?>**</span>

</label>

</div>

<div class="col-12 col-md-6" id="selectNumTableWidth">

<select class="form-control" id="selectNumTable" name="selectNumTable">

<option value="" disabled selected>Choisir une table</option>

<option value="1">Table n°1</option>

<option value="2">Table n°2</option>

<option value="3">Table n°3</option>

<option value="4">Table n°4</option>

<option value="5">Table n°5</option>

<option value="6">Table n°6</option>

<option value="7">Table n°7</option>

<option value="8">Table n°8</option>

</select>

</div>

</div>

</p>

</div>

<div class="col-12 col-md-12" align="center" id="btnSubmitResa2">

<p>

<button type="submit" class="btn btn-blue-grey" id="submitResaResponsive" name="submitResaResponsive" style="display:none;">Valider la<br>réservation</button>

<button type="submit" class="btn btn-blue-grey" id="submitResa2" name="submitResa2">Valider la réservation</button>

</p>

<p><span class="amber-text">\*&nbsp;À partir de 7 personnes, veuillez nous appeler pour prendre une réservation.</span></p>

</div>

</fieldset>

</form>

**<?php** } **?>**

</div>

</div>

</div>

<!---->

</div>

</div>

**<?php** require 'footer.php' **?>**

Création d’un contrôleur pour effectuer l’enregistrement des inputs en effectuant des contrôles de conditions et de regex

**<?php**

require\_once '../models/database.php';

require\_once '../models/modelResa.php';

// Instanciation de l'objet Réservation contenant les méthodes utilisées

$resaOBJ = new reservation();

$addResaSuccess = false;

$today = date('Y-m-d'); // variable servant à initialiser le calendrier à la date du jour

$oneDateLater = date('Y-m-d', strtotime(date('Y-m-d') . ' +1 month')); // variable servant à délimiter la fin du calendrier

// variable de récupération d'erreurs

$arrayError = [];

// Test des champs obligatoires

if ($\_SERVER['REQUEST\_METHOD'] == 'POST') {

// regex utilisées pour le contrôle de saisie

$patternTxt = '/^[a-zA-ZÀ-ÿ œ\'-.,]\*$/';

//$patternPrice = '/^[0-9€0-9]\*$/';

$patternPhone = '/^0[0-9]([ .-]?[0-9]{2}){4}$/';

$patternDate = '/^(0?[1-9]|[12][0-9]|3[01])[\/\-](0?[1-9]|1[012])[\/\-]\d{4}$/';

$patternHour = '/^([01]?[0-9]|2[0-3]):[0-5][0-9]$/';

$patternNbPers = '/^[0-9][0-9]$/';

$patternMail = '/^[a-zA-Z0-9.!#$%&\'\*+\/=?^\_`{|}~-]+@[a-zA-Z0-9](?:[a-zA-Z0-9-]{0,61}[a-zA-Z0-9])?(?:\.[a-zA-Z0-9](?:[a-zA-Z0-9-]{0,61}[a-zA-Z0-9])?)\*$/';

// contrôle de saisie

// NOM

if (empty($\_POST['inputLastnameResa'])) {

$arrayError['lastnameResaErr'] = 'Le nom pour la réservation est requis.';

} else {

$resaOBJ->reservation\_lastname = test\_input($\_POST['inputLastnameResa']);

// vérifie si le champs contient des lettres et de la ponctuation

if (!preg\_match($patternTxt, $resaOBJ->reservation\_lastname)) {

$arrayError['lastnameResaErr'] = 'Caractères incorrects, ex : DOE.';

} else {

unset($arrayError['lastnameResaErr']);

}

}

// PRENOM

if (empty($\_POST['inputFirstnameResa'])) {

$arrayError['firstnameResaErr'] = 'Le prénom pour la réservation est requis.';

} else {

$resaOBJ->reservation\_firstname = test\_input($\_POST['inputFirstnameResa']);

// vérifie si le champs contient des lettres et de la ponctuation

if (!preg\_match($patternTxt, $resaOBJ->reservation\_firstname)) {

$arrayError['firstnameResaErr'] = 'Caractères incorrects, ex : Jane.';

} else {

unset($arrayError['firstnameResaErr']);

}

}

// VALIDER

if (isset($\_POST['submitResa2']) && count($arrayError) == 0) {

$resaOBJ->addResa();

$addResaSuccess = true;

}

}

// fonction de sécurisation de la saisie, injection de code, espaces et antislashs

function test\_input($data) {

$data = trim($data);

$data = stripslashes($data);

$data = htmlspecialchars($data);

return $data;

}

//Récupérer les infos de la réservation

$reservation = $resaOBJ->displayResaInformations();

**?>**

Dans le modèle de la table « reservation », je crée une méthode pour rajouter une réservation une fois toutes les informations validées.

**<?php**

class reservation extends database {

// champs de la table réservation

public $reservation\_id;

public $reservation\_lastname;

public $reservation\_firstname;

public $reservation\_numTel;

public $reservation\_mail;

public $reservation\_dateResa;

public $reservation\_hourResa;

public $reservation\_nbPers;

public $reservation\_arrive;

public $tables\_id;

public $reservation\_mailConfirm;

public $reservation\_smsConfirm;

/\*\*

\* Méthode permettant d'ajouter une réservation

\* @return exécute la requête pour ajouter une réservation

\*/

function addResa() {

// Ajouter une réservation

$sql = $this->database->prepare('

INSERT INTO `lcdh\_reservation` (`reservation\_lastname`, `reservation\_firstname`, `reservation\_numTel`, `reservation\_mail`, `reservation\_dateResa`, `reservation\_hourResa`, `reservation\_nbPers`, `tables\_id`)

VALUES (:lastnameResa, :firstnameResa, :numTelResa, :mailResa, :dateResa, :hourResa, :nbPersResa, :tables\_id)

');

$sql->bindValue(':lastnameResa',$this->reservation\_lastname,PDO::PARAM\_STR);

$sql->bindValue(':firstnameResa',$this->reservation\_firstname,PDO::PARAM\_STR);

$sql->bindValue(':numTelResa',$this->reservation\_numTel,PDO::PARAM\_STR);

$sql->bindValue(':mailResa',$this->reservation\_mail,PDO::PARAM\_STR);

$sql->bindValue(':dateResa',$this->reservation\_dateResa,PDO::PARAM\_STR);

$sql->bindValue(':hourResa',$this->reservation\_hourResa,PDO::PARAM\_STR);

$sql->bindValue(':nbPersResa',$this->reservation\_nbPers,PDO::PARAM\_INT);

$sql->bindValue(':tables\_id',$this->tables\_id,PDO::PARAM\_INT);

return $sql->execute();

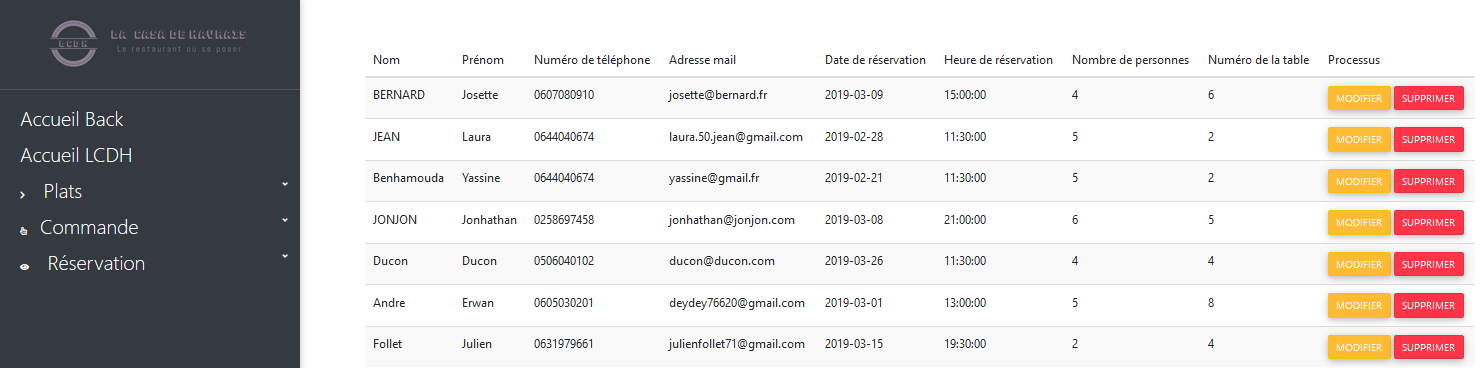
}

}

**?>**

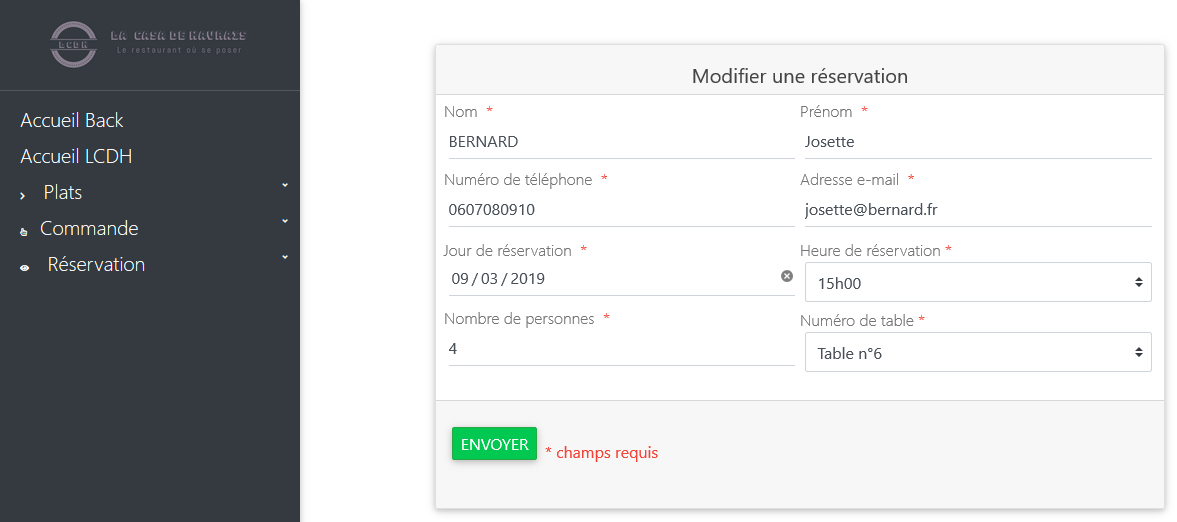
## 3.2 Affichage d’une réservation

Dans la Vue correspondante, je crée un tableau pour pouvoir afficher toutes les réservations de créés selon l’id du client, il aura également la possibilité d’afficher une réservation en particulier.



## 3.3 Modification d’une réservation

Dans la Vue respective, je crée un formulaire pour afficher les données préalablement renseignées et je permets à l’utilisateur de modifier celles-ci pour qu’il puisse effectuer une modification / mise à jour.



Je crée un contrôleur pour enregistrer les différentes interactions avec le client (Modification, envoi de formulaire, etc. …) et je m’assure que les informations respectent les prérequis (ex. Regex) avant d’utiliser une méthode pour mettre à jour la table respective.

**<?php**

require\_once '../models/database.php';

require\_once '../models/modelResa.php';

// Instanciation de l'objet Réservation contenant les méthodes utilisées

$resaOBJ = new reservation();

if (isset($\_GET['idResa'])) {

$resaOBJ->reservation\_id = $\_GET['idResa'];

$resa = $resaOBJ->displayResa();

}

//MODIFICATION

$arrayHour = $resaOBJ->listHour(); //Tableau qui reprend la liste des catégories

$arrayNumTable = $resaOBJ->listTable(); //Tableau qui reprend la liste des sous-catégories

$updateResaSuccess = false;

$today = date('Y-m-d'); // variable servant à initialiser le calendrier à la date du jour

$oneDateLater = date('Y-m-d', strtotime(date('Y-m-d') . ' +1 month')); // variable servant à délimiter la fin du calendrier

// variable de récupération d'erreurs

$arrayError = [];

// Test des champs obligatoires

if ($\_SERVER['REQUEST\_METHOD'] == 'POST') {

// regex utilisées pour le contrôle de saisie

$patternTxt = '/^[a-zA-ZÀ-ÿ œ\'-.,]\*$/';

$patternPhone = '/^0[0-9]([ .-]?[0-9]{2}){4}$/';

$patternDate = '/^(0?[1-9]|[12][0-9]|3[01])[\/\-](0?[1-9]|1[012])[\/\-]\d{4}$/';

$patternHour = '/^([01]?[0-9]|2[0-3]):[0-5][0-9]$/';

$patternNbPers = '/^[0-9][0-9]$/';

$patternMail = '/^[a-zA-Z0-9.!#$%&\'\*+\/=?^\_`{|}~-]+@[a-zA-Z0-9](?:[a-zA-Z0-9-]{0,61}[a-zA-Z0-9])?(?:\.[a-zA-Z0-9](?:[a-zA-Z0-9-]{0,61}[a-zA-Z0-9])?)\*$/';

// contrôle de saisie

// MAIL

if (empty($\_POST['inputMailResa'])) {

$arrayError['mailResaErr'] = 'Le mail est requis.';

} else {

$resaOBJ->reservation\_mail = test\_input($\_POST['inputMailResa']);

// vérifie si le champs contient un email

if (!preg\_match($patternMail, $resaOBJ->reservation\_mail)) {

$arrayError['mailResaErr'] = 'Format de mail invalide, ex : mail@mail.com.';

} else {

unset($arrayError['mailResaErr']);

}

}

// DATE

if (empty($\_POST['inputDateResa']) || $\_POST['inputDateResa'] < $today) {

$arrayError['dateResaErr'] = 'Le jour de réservation est requis.';

} else {

$resaOBJ->reservation\_dateResa = test\_input($\_POST['inputDateResa']);

// vérifie si le champs contient une date

if (preg\_match($patternDate, $resaOBJ->reservation\_dateResa)) {

unset($arrayError['dateResaErr']);

}

}

// VALIDER

if (isset($\_REQUEST['submit']) && count($arrayError) == 0) {

$resaOBJ->ModifyResa();

$updateResaSuccess = true;

}

}

// fonction de sécurisation de la saisie, injection de code, espaces et antislashs

function test\_input($data) {

$data = trim($data);

$data = stripslashes($data);

$data = htmlspecialchars($data);

return $data;

}

**?>**

Dans le modèle correspondant, je crée une méthode pour mettre à jour selon l’id de la réservation.

**<?php**

class reservation extends database {

// champs de la table patients

public $reservation\_id;

public $reservation\_lastname;

public $reservation\_firstname;

public $reservation\_numTel;

public $reservation\_mail;

public $reservation\_dateResa;

public $reservation\_hourResa;

public $reservation\_nbPers;

public $reservation\_arrive;

public $tables\_id;

public $reservation\_mailConfirm;

public $reservation\_smsConfirm;

// Modifier une réservation

function ModifyResa() {

$sql = $this->database->prepare('

UPDATE `lcdh\_reservation`

SET `reservation\_lastname`=:lastnameResa, `reservation\_firstname`=:firstnameResa, `reservation\_numTel`=:numTelResa, `reservation\_mail`=:mailResa, `reservation\_dateResa`=:dateResa, `reservation\_hourResa`=:hourResa, `reservation\_nbPers`=:nbPersResa, `tables\_id`=:tables\_id

WHERE `reservation\_id` = :idResa

');

$sql->bindValue(':idResa', $this->reservation\_id, PDO::PARAM\_INT);

$sql->bindValue(':lastnameResa',$this->reservation\_lastname,PDO::PARAM\_STR);

$sql->bindValue(':firstnameResa',$this->reservation\_firstname,PDO::PARAM\_STR);

$sql->bindValue(':numTelResa',$this->reservation\_numTel,PDO::PARAM\_STR);

$sql->bindValue(':mailResa',$this->reservation\_mail,PDO::PARAM\_STR);

$sql->bindValue(':dateResa',$this->reservation\_dateResa,PDO::PARAM\_STR);

$sql->bindValue(':hourResa',$this->reservation\_hourResa,PDO::PARAM\_STR);

$sql->bindValue(':nbPersResa',$this->reservation\_nbPers,PDO::PARAM\_INT);

$sql->bindValue(':tables\_id',$this->tables\_id,PDO::PARAM\_INT);

return $sql->execute();

}

}

**?>**

## 3.4 Suppression d’une réservation

Pour effectuer la suppression d’une réservation, j’ai mis en place un affichage en fonction de la réussite dans la Vue pour confirmer la suppression.



**<?php**

require\_once '../controllers/controllerSupprimer-reservation.php';

include 'header\_back.php';

**?>**

<!-- CONTENT PAGE -->

<div class="content-wrap marginTop" id="margSuppResa">

<div class="container-fluid">

<div class="row">

**<?php**

if ($success){

**?>**

<p id="pResaSupp">Votre réservation à bien été supprimé !</p>

</div>

<div class="row">

<div class="col-md-6" id="suppResaDiv6">

<div class="col-md-4" id="suppResaDiv4">

<a href="listReservation.php" class="btn btn-primary btnResaBack">Retour à la liste des réservation</a>

</div>

<div class="col-md-4" align="center">

<a href="index.php" class="btn btn-default btnResaBack">Accueil</a>

</div>

<div class="col-md-4">

<a href="ajout-reservation.php" class="btn btn-info btnResaBack">Ajouter une réservation</a>

</div>

</div>

**<?php**

}else{

echo 'Dommage pour toi, ça n\'a pas fonctionné.';

}

**?>**

</div>

</div>

</div>

**<?php** include 'footer\_back.php'; **?>**

Dans le contrôleur de la Vue, je récupère le choix de l’administrateur afin de supprimer la réservation.

**<?php**

require\_once '../models/database.php';

require\_once '../models/modelResa.php';

// Instanciation de l'objet Réservation contenant les méthodes utilisées

$resaOBJ = new reservation();

$resa = $resaOBJ->listResa(); //Tableau qui reprend la liste des réservations

$success = true;

if (isset($\_GET['idResa'])) {

$resaOBJ->reservation\_id = $\_GET['idResa']; //Récupère id initialisé comme idResa

if($resaOBJ->deleteResa()){//Suppression de la réservation

$success;

}

}

**?>**

Je crée également une recherche dans la « reservation » pour permettre à l’administrateur de pouvoir trouver plus vite la personne qu’il veut et ainsi l’afficher, modifier ou supprimer à sa guise.

**Vue** : On inclut les Modèles et le Contrôleur via un « require\_once»

<li id="liResa">

<a class="collapsible-header waves-effect arrow-r">

<i class="fas fa-eye"></i> Réservation<i class="fas fa-angle-down rotate-icon"></i>

</a>

<div class="collapsible-body" id="divMenusResa">

<ul>

<li><a href="ajout-reservation.php" class="waves-effect">Ajouter</a></li>

<li><a href="listReservation.php" class="waves-effect">Liste</a></li>

<li>

<form class="search-form" role="search" id="searchNavAdminResa">

<div class="form-group md-form mt-0 pt-1 waves-light waves-effect waves-light">

<input type="text" class="form-control" id="inputSearchNavAdminResa" placeholder="Search">

<button type="submit" name="submit" class="btn btn-light validate btn-block">Rechercher</button>

</div>

</form>

</li>

</ul>

</div>

</li>

**Contrôleur** : J’instancie un nouvel objet à l’aide du Modèle respectif et j’appelle les méthodes, ici je l’appelle la méthode « searchResa($search) »

**<?php**

require\_once '../models/database.php';

require\_once '../models/modelResa.php';

// Instanciation de l'objet Réservation contenant les méthodes utilisées

$resaOBJ = new reservation();

//TOUS

if(isset($\_POST['search'])){

$arrayResa = $resaOBJ->searchResa($\_POST['inputSearchNavAdminDishes']);

}else{

$arrayResa = $resaOBJ->listResa();

}

**?>**

**Modèle** : Connexion à la base de données via la classe « database » + création d’une méthode pour effectuer une requête dans la table « reservation »

**<?php**

class reservation extends database {

// champs de la table patients

public $reservation\_id;

public $reservation\_lastname;

public $reservation\_firstname;

public $reservation\_numTel;

public $reservation\_mail;

public $reservation\_dateResa;

public $reservation\_hourResa;

public $reservation\_nbPers;

public $reservation\_arrive;

public $tables\_id;

public $reservation\_mailConfirm;

public $reservation\_smsConfirm;

//Recherche des réservations

function searchResa($search) {

$sql = $this->database->prepare('SELECT \* FROM `lcdh\_reservation` WHERE `lastnameResa` LIKE :search OR `mailResa` LIKE :search OR `tables\_id` LIKE :search');

$sql->bindValue(':search', '%'.$search.'%', PDO::PARAM\_INT);

$sql->execute();

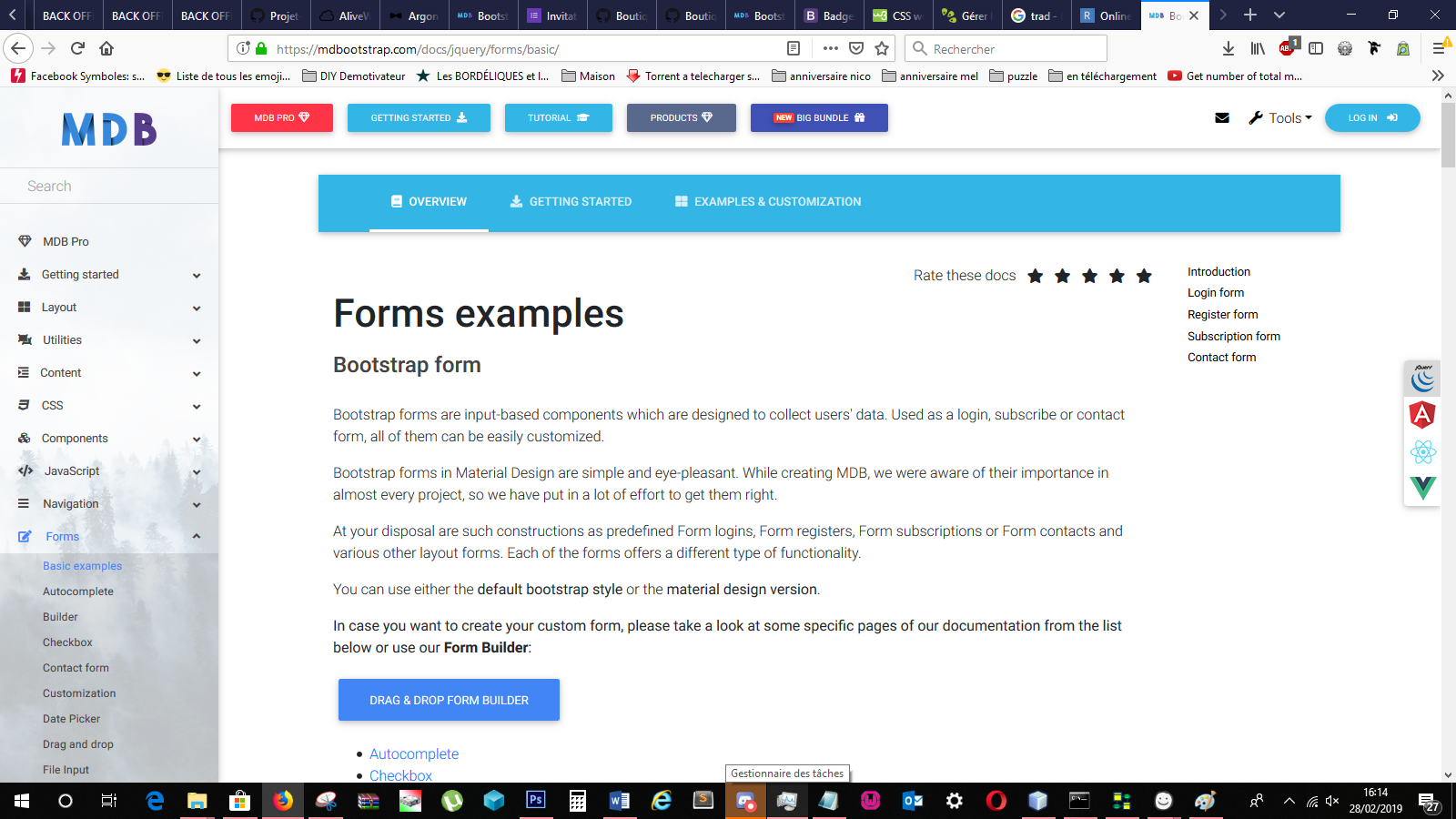
return $sql->fetchAll(PDO::FETCH\_OBJ);

}

}

**?>**

# 4. Anglais de rigueur : Une communauté importante orientée vers le partage



Les formulaires Bootstrap sont basés sur la composition des entrées de données sur lequel est désigné la collecte des données utilisateurs. Utilisé comme une connexion, une souscription ou un formulaire de contact, ils peuvent tous être aisément customiser.

Les formulaires Bootstrap dans Material Design sont simple et plaisant à l’œil. Pendant la création de Material Design for Bootstrap, nous sommes conscients de l’importance dans la plupart de chaque projet, donc nous avons mis beaucoup d’effort pour qu’ils soient bien.

Sont à votre disposition des formulaires construit prédéfini tel que les formulaires de connexion, de souscription ou de contact et varies selon leur forme. Chacun des formulaires offres différentes formes de fonctionnalité.

Vous pouvez utiliser aussi bien le bootstrap style par défaut ou la version de Material Design for Bootstrap.

Au cas où vous voudriez créer votre formulaire personnalisé, merci de regarder les pages spécifiques à notre documentation de la liste au-dessous ou d’utilisé notre constructeur de formulaires.

# 5. Conclusion

Mon projet La Casa De Havrais n’est qu’à ses débuts. L’enregistrement des Réservations est pleinement opérationnel. Tout comme celle de la mise en place des plats. Quelques détails sont à améliorer mais la base est présente pour effectuer les tests et donner une idée aux clients.

Faute de temps, je n’ai pas pu mettre en œuvre l’enregistrement du panier dans la base de données. Même si celui-ci est opérationnel dans les faits, commander des plats n’est toujours pas possible.

Après cette première version, j’ai pour projet de continuer à développer La Casa De Havrais pour apporter des idées innovantes dans le système de réservation et ainsi aider les clients à se démarquer.

Ces nouvelles fonctionnalités seront apportées bientôt pour permettre de servir de tests et de book pour aider à trouver un job en relation avec le projet. L’ergonomie et le responsive sera revu sur quelques pages pour permettre aux utilisateurs une navigation optimale.

A travers ce projet, j’ai pu me rendre compte que de développer une application demande du temps et un très fort investissement que ce soit de la conception, à la réalisation et aux séances de débogage.

Je vous remercie de m’avoir lu et écouté. J’espère vous avoir donné envie d’en connaitre plus dans le domaine de la restauration ou tout simplement pour déguster ses plats merveilleux qui verront peut-être le jour dans ce dit restaurant.

Merci.