Cimino Thomas

Studi

Graduate Developper Web Full Stack

DOSSIER DE PROJET

Site web Quai Antique

Studi
DIGITAL EDUCATION

Quai Antique

Remerciements

Premièrement, je tiens à remercier ma famille pour l'aide qu'elle m'a apporté afin de me permettre de préparer cette formation, et pour leurs encouragements.

Je tiens à remercier les personnes que j'ai croisé sur des salons Discord et qui m'ont beaucoup aidé à alimenter mon projet et à mieux comprendre certains aspects/fonctions du développement web ou mobile.

Je remercie mon maître de stage qui m'a permis de découvrir d'autres technologies de développement d'applications, tel que Flutter (kit de développement utile pour créer des applications mobiles, desktop et web) et Dart (langage de programmation optimisé pour les applications).

Je souhaite remercier mes amis et mes collègues qui m'ont aidé à éclaircir et à « alimenter » le sujet de la restauration et de l'expérience utilisateur.

Sommaire

Glossaire

API : Une interface de programmation d'application (application programming interface), est une interface logiciel qui permet de « connecter » un logiciel ou un service à un autre logiciel (service) afin d'échanger des données et des fonctionnalités.

Bundle : Un paquet d'éléments dématérialisés ou physiques, pour notre exemple ce sera un paquet de fichiers.

CRUD : L'acronyme informatique anglais CRUD (pour Create, Read, Update, Delete) (parfois appelé SCRUD avec un "S" pour Search) désigne les quatre opérations de base pour la persistance des données, en particulier le stockage d'informations en base de données.

Fenêtre modale : une fenêtre qui apparait directement à l'intérieur de la fenêtre courante du navigateur, au-dessus de la page web qui l'appelle. L'utilisateur devra forcément agir sur cette fenêtre avant de pouvoir agir sur la page courante.

Framework : Un ensemble d'outils et de composants logiciels organisés conformément à un plan d'architecture et de patterns.

Full stack : Un développeur Full Stack, est un développeur capable de tout faire, qui a les compétences pour faciliter le travail de ses collaborateurs.

Hachage : Une méthode de chiffrement qui transforme les enregistrements et les caractères, afin qu'ils ne soient pas stockés en clair dans une base de données.

Input : Un champ de formulaire, qui attendra une entrée de la part de l'utilisateur.

Métadonnées : Les métadonnées sont des informations structurée qui décrivent, expliquent, localisent ou facilitent autrement l'obtention, l'utilisation ou la gestion d'une ressource d'information.

POST : C'est la méthode que le navigateur utilise pour demander au serveur une réponse prenant en compte les données contenues dans le corps de la requête HTTP. Si un formulaire est envoyé avec cette méthode, les données sont ajoutées au corps de la requête HTTP.

Introduction

1. Compétences du référentiel couvertes par le projet

1.1 Développer la partie front-end d'une application web ou web mobile en intégrant les recommandations de sécurité

- Maquetter une application
- Réaliser une interface utilisateur web statique et adaptable
- Développer une interface utilisateur web dynamique

Une maquette graphique devait être réalisée afin qu'elle soit présentée au client. L'application web doit être disponible sur navigateur, d'où le choix de développer en PHP et en JavaScript. Elle doit pouvoir proposer une expérience utilisateur similaire quelle que soit la taille de l'écran. J'ai donc fait en sorte que l'interface s'adapte aux différentes tailles d'écrans (mobiles, ordinateurs, tablettes).

1.2 Développer la partie back-end d'une application web ou web mobile en intégrant les recommandations de sécurité

- Créer une base de données
- Développer les composants d'accès aux données
- Développer la partie back-end d'une application web ou web mobile

La partie back-end fait appel à une base de données relationnelles qui renvoie les données que j'ai pensées et développées. Concernant ces dernières, je les ai développées par le système de gestion MySQL, grâce à sa documentation. De plus, j'utilise l'extension PDO de PHP, une interface pour accéder à une base de données.

2. Résumé du projet

Dans le cadre de ma formation Graduate Developper Web Full Stack, je devais réaliser un projet d'évaluation qui consistait à créer un site pour un restaurant, avec pour nom « Quai antique ».

Le projet « Quai antique » est une application web ayant pour but d'ajouter de la visibilité au chef du restaurant, pour promouvoir les produits qu'il travaille et attirer de nouveaux clients. Les clients doivent trouver les coordonnées de l'établissement, pour pouvoir contacter facilement le restaurant. De plus, les horaires d'ouvertures doivent être consultables.

Cette clientèle pourra choisir une date et une heure précise de réservation, ajouter le nombre de personnes présentes, ce qui facilite la planification de leur visite. Cette fonctionnalité enverra également un email de confirmation comportant les informations de sa réservation au client. Pour accéder à cette fonctionnalité, le client devra avoir créer son compte et être connecté.

Le chef du restaurant souhaitait également un espace qui lui est réservé pour gérer le contenu des informations qu'il souhaite afficher à ses clients. Ce tableau de bord lui permet de mettre à jour les menus, les cartes, et les allergènes sur les aliments. Il pourra également gérer le planning et les images visibles sur la page d'accueil.

L'application devait être déployée pour le rendu de l'examen le 23 mai. Mon rôle au sein du projet est de le concevoir de A à Z. Pour cet examen, il n'y avait pas forcément de technologie imposée, juste une SGBDR (système de gestion de bases de données relationnelles) basé sur le langage SQL.

3. Environnement humain et technique

3.1 Environnement humain

Pendant la réalisation du projet, j'étais en contrat en intérim dans un autre domaine que la restauration ou le développement, donc il m'a fallu trouver des personnes « ressources » dans ce domaine spécifique, qui pouvaient m'aider à comprendre le métier de la restauration afin de cerner au mieux cet environnement et les besoins qu'attendaient le chef et les clients. Je me suis également adressé à ma famille et mes amis afin qu'ils puissent m'apporter leurs idées et ressentis sur l'appréhension, les accès, et l'expérience d'utilisation de ce site.

En exemple, j'ai appris que depuis le 1er mars 2022, l'affichage de l'élevage et l'origine des viandes est obligatoire dans les établissements, donc je devais faire de même sur le site.

3.2 Environnement technique

Je travaille sur un ordinateur portable, cela me permet de travailler où bon me semble. Mes échanges sur les technologies ou mes recherches pour l'avancée du projet, étaient principalement sur les forums/salons de discussion, ou mon entourage.

Certains sites internet comme **Grafikart**, **OpenClassrooms** ou **StackOverflow**, m'ont énormément aidé pour la mise en place de certaines fonctionnalités, à comprendre les erreurs de code que je réalisais, et à l'améliorer.

La réalisation du projet

1. Le cahier des charges

US1. Se connecter.

Utilisateurs concernés : Administrateur, Clients

Le compte administrateur sera créé pour un employé du restaurant en particulier : l'hôte d'accueil. C'est lui qui gérera les informations sur le site web.

Toutefois, un autre type de compte sera possible : le "client" (voir *US7 - Mentionner des allergies*).

Quel que soit le type d'utilisateur souhaitant se connecter, il pourra le faire grâce au même formulaire de connexion. Les identifiants à entrer seront l'adresse e-mail et un mot de passe sécurisé.

US2.Créer une galerie d'images

Utilisateurs concernés : Administrateur

Sur la page d'accueil, le chef Michant aimerait afficher les photographies de ses plats les plus appréciés afin de donner l'eau à la bouche à ses potentiels convives.

Toute photo devra pouvoir être ajoutée, modifiée ou supprimée sur la plateforme d'administration. Chaque photo aura aussi un titre. Il sera visible sur la page d'accueil lors du survol de son image.

Un bouton d'appel à l'action vers le module de réservation (Voir US6 - Réserver une table) devra être positionné juste après la galerie.

US3. Publier la carte.

Utilisateurs concernés : Administrateur

La carte du restaurant devra être présente sur une page dédiée.

Les plats seront listés dans des catégories (ex : Entrées, Desserts, Burgers, etc).

Les informations nécessaires pour chaque plat sont :

- Un titre
- Une description
- Un prix

US4. Présenter les menus.

Utilisateurs concernés : Administrateur

Le chef Michant propose des plats incontournables mais souhaiterais également mettre en place des menus et pouvoir les promouvoir avec son site web. Pour chaque menu, on aura :

- Un titre
- Une à plusieurs formules, ayant chacune un prix et une description.

US5. Définir les horaires d'ouverture.

Utilisateurs concernés : Administrateur

Pour un restaurant, cette information est CAPITALE.

Pour chaque jour de la semaine, les horaires d'ouverture devront donc être affichés dans le pied de chaque page du site.

De plus le Chef Michant souhaite avoir la main sur ces horaires pour permettre à l'administrateur de les modifier.

US6. Réserver une table.

Utilisateurs concernés : Visiteurs, Clients

Dans le menu, un bouton d'appel à l'action sera particulièrement mis en valeur : "**réserver**"

Au clic de ce dernier, le visiteur est redirigé sur un formulaire à remplir. Plusieurs champs seront nécessaires :

- Le nombre de couverts
- La date
- L'heure prévue
- La mention des allergies

Le visiteur doit savoir si des places sont disponibles sans rechargement de la page.

On pourra sélectionner un horaire par tranche de 15 minutes entre l'ouverture et la fermeture du restaurant.

La dernière heure avant la fermeture, aucun convive n'est accepté.

Exemple : si le restaurant ouvre de 12h à 15h, alors on aura

12h00 / 12h15 / 12h30 / 12h45 / 13h00 / 13h15 / 13h30 / 13h45 / 14h00

Attention! Le restaurant est limité en nombre de convives. Pour éviter toute déconvenue sur place, il faudra refuser les réservations au-delà d'un certain seuil. Ce seuil de convives maximum pourra être précisé dans le panel d'administration.

US7. Mentionner des allergies.

Utilisateurs concernés : Visiteurs, Clients

Lors de la réservation d'une table, le visiteur peut indiquer si une personne qui l'accompagne a des allergies.

Si le visiteur vient régulièrement dans ce restaurant, il peut aussi créer un compte client et donc gagner du temps lors de la complétion du formulaire.

Quand le visiteur créera son compte, on lui proposera d'entrer une adresse email, un mot de passe sécurisé, un nombre de convives par défaut ainsi que la mention des allergies.

Dorénavant, si le visiteur se connecte avant de remplir le formulaire de réservation d'une table, le nombre de convives et les allergies seront remplis automatiquement avec les réglages du client.

2. Spécifications techniques

Voici les différentes technologies que j'ai utilisé pour réaliser ce projet :

- Environnement de travail :
    - L'éditeur de code **Visual Studio Code**, grâce à son terminal et son extension **PHP** me permet d'exécuter une commande qui démarre un serveur web pour avoir le résultat de mon code.
    - L'outil de versioning **Git** et la plateforme web **GitHub** me donne la possibilité de stocker mon code dessus et donc d'avoir un historique de version.
    - L'outil **Docker** 4.2 peut exécuter plusieurs applications sur mon ordinateur.
    - Le logiciel **Looping** 4.0 m'offre la possibilité de créer mon plan UML.
    - Le site **Figma**, permet de créer la maquette, la charte graphique et la feuille de style.
- Partie Front :
    - Le langage **HTML** 5 structure mon application web et le langage **CSS** pour mettre en forme les pages de mon application.
    - Le framework **Bootstrap** 5.2 : Cette boite à outils me propose des classes toutes configurées pour styliser mes pages, avec des fonctionnalités utiles pour l'adaptabilité des tailles d'écran.
    - Le langage **JavaScript** ES6 dynamise les pages de mon site
- Partie back :
    - Le langage **PHP** 8.2 : le code est exécuté sur le serveur donc invisible par l'utilisateur et il permet d'intégrer facilement du HTML
    - L'environnement bas niveau **Node.js** permet l'exécution de JavaScript coté serveur et l'utilisation de son outil **npm**.

3. Détails des outils techniques

Dans l'optique d'avoir une meilleure compréhension de mon environnement de travail ainsi que du projet, j'ai pris le soin de détailler deux outils qui selon moi, m'ont permis de travailler efficacement et proactivement tout au long de ce projet.

Dans cette liste on retrouve donc Docker, qui comme je l'ai mentionné, est un outil permettant de créer un environnement d'exécution multi-conteneur puis Git, l'outil miracle permettant de disposer de ce qu'on appelle le *versioning* qui permet d'avoir un suivi de l'évolution de nos ajouts et suppressions et pour finir GitHub le lieu où mon travail sera stocké.

Docker

Docker est une plateforme open-source révolutionnaire simplifiant le développement et le déploiement d'applications. Il est né en 2013 développé par Solomon Hykes. Grâce à ses conteneurs isolés, appelés "images Docker", tout ce dont une application a besoin est encapsulé, garantissant portabilité et reproductibilité. Cela élimine les problèmes de compatibilité, facilite le travail en équipe et améliore l'efficacité du processus de développement et de déploiement.

Git

Git est donc un système de contrôle de version qui a été inventé et développé par Linus Torvalds développé en 2005, Git est aujourd'hui utilisé dans le monde entier et permet aux développeurs de travailler sur les mêmes projets en simultané, tout en évitant les erreurs entres les différents opérateurs.

De plus, il permet d'accéder aux versions antérieures si le besoin en est.

GitHub

GitHub repose sur le logiciel de versioning Git, mais va plus loin que celui-ci en offrant la possibilité de partager son code au monde entier, tout le monde peut avoir un visuel sur notre code si le propriétaire d'un projet le permet.

4. Conception de la base de données

Pour appréhender la création du site, j'ai listé toutes les données que le chef du restaurant aurait besoin de voir apparaître.

Par exemple, les « menus », les « cartes », les « aliments », les « allergènes », les « horaires d'ouverture ». Pour mes menus, j'ai comme propriété une id, un titre de menu, un prix, une description.

Une fois toutes mes entités définies, j'ai réalisé mon plan **UML** (langage de modélisation unifié) sur le logiciel **Looping**. Cette étape consiste à réaliser un diagramme de classes représentant mes entités (table) avec leur cardinalité (relations) et leurs propriétés. En restant sur l'exemple du menu, la table contiendra une clé primaire (PRIMARY KEY) afin d'être liée à une autre table.

Du fait de la cardinalité définie pour le menu et ses plats, qui est de 1,n – *,n, une table sera créée contenant 2 champs qui font référence à la clé primaire de chaque table, créant une relation entre elles. En ce qui concerne des cardinalités *,1 (1,1) – *,n la clé primaire de la table coté *,n devient la clé étrangère de la table coté *,1 ou 1,1, créant une relation entre les tables.

Veuillez retrouver ci-dessous mon plan UML dans son entièreté :

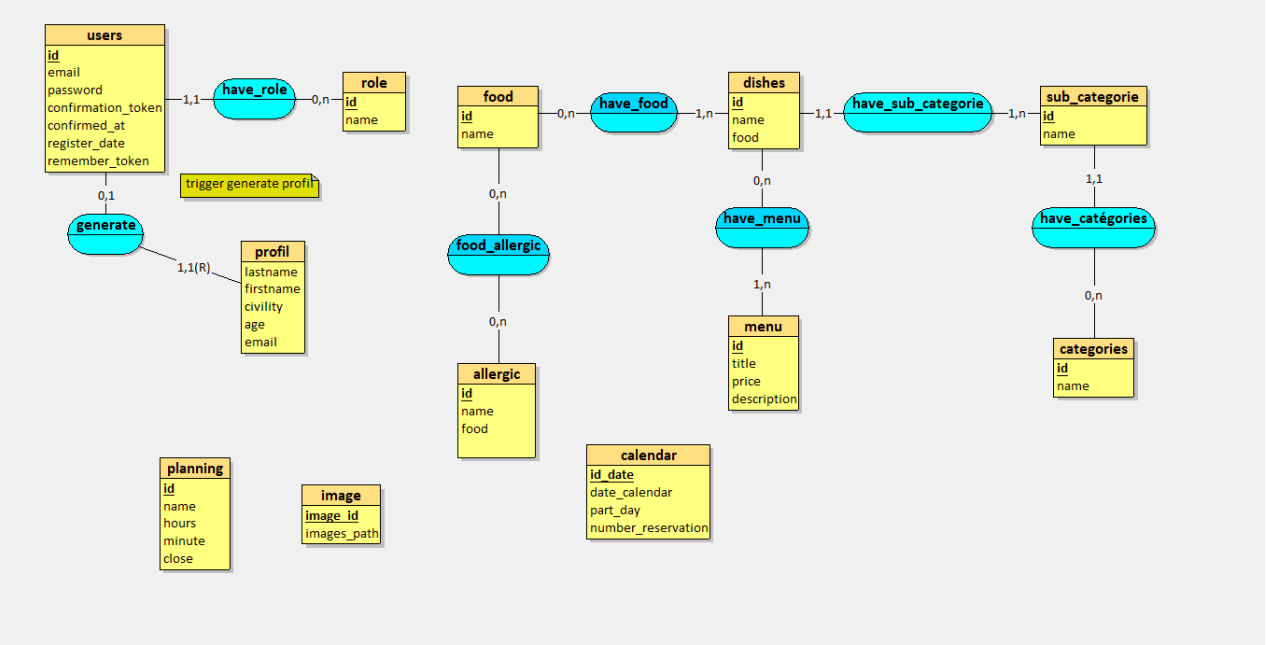


Figure 1 - Plan UML

5. Développement de l'API

5.1 Initialisation du projet et versioning

Pour débuter mon projet, j'ai créé un fichier README.md afin de pouvoir écrire une documentation pour le client, et le guider pour la prise en main du site.

J'ai commencé par relier mon projet à une plate-forme d'hébergement de code pour avoir un contrôle de version dessus, mon choix s'est porté sur **GitHub** comme énoncé précédemment, dans le but de pouvoir partager rapidement mon code à d'autres, pour qu'ils puissent m'aider, ou simplement de pouvoir revenir sur une version précédente pour de multiples raisons. Pour ceci, j'initialise le projet avec la commande *git init*. Une fois que j'ai développé des fonctionnalités, j'enregistre les modifications avec la commande *git commit*.

Ensuite, j'ai créé un dépôt de code, que j'ai relié grâce à la ligne de commande *git remote add master git@github.com:Oda-7/quai_antique.git*. Je n'ai plus qu'à pousser sur le dépôt pour enregistrer mon code avec la commande *git push*.

L'outil **Git** permet également le versioning en local.

5.2 Développement de la base de données

Une fois ce plan réalisé, j'ai souhaité utiliser **Docker** pour exécuter une base de données en local. Pour ceci, j'ai dû créer une image qui contenait une base de données, grâce à la ligne de commande *docker run unless-stopped mysql :8*, depuis le terminal du projet. Je peux utiliser la commande **docker** grâce à l'installation du logiciel **Docker**.

Ensuite j'ai créé un fichier docker-compose.yml, où j'ai défini le service de base de données « mysql » avec l'image utilisée, le nom de l'hôte, le nom du conteneur (une instance de l'image), les ports et ses variables d'environnement qui définiront l'utilisateur de la base de données.

```yaml
version: '3.7'
services:
  mysql:
    image: mysql:8.0
    hostname: mysql
    container_name: quai_antique_bd
    restart: unless-stopped
    ports:
      - 3306:3306
    volumes:
      - ./mysql:/var/lib/mysql
    env_file:
      - database.env

volumes:
  mysql:
    name: quai_antique
```

Figure 2 - Extrait du fichier docker-compose.yml

```
MYSQL_ROOT_PASSWORD=root
MYSQL_DATABASE=mysql
MYSQL_USER=admin
MYSQL_PASSWORD=quaiantique
```

Figure 3 - Extrait du fichier database.env

Je n'ai plus qu'à exécuter tout ça grâce à la ligne de commande *docker compose up*, pour créer ma base de données en local.

Il me reste à connecter mon projet à la base de données, ainsi qu'à créer les tables et leurs relations qui seront nécessaires pour stocker mes données. Pour réaliser tout ça, j'ai utilisé le langage de programmation **PHP** et son extension **PDO** (une interface pour accéder à une base de données) afin de me connecter à la base de données. Ensuite les tables de ma base de données seront créées lors de l'exécution en local ou en production, du fait d'avoir préparé les requêtes dans des variables qui seront exécutées avec l'extension **PDO**.

Mes tables contiennent également une vérification afin d'éviter la duplication de mes tables.

```php
if ($_SERVER['SERVER_NAME'] == "localhost") {
    // Information a modifier pour une utilisation local
    $host_name = '127.0.0.1';
    $database = 'quaiantique';
    $user_name = 'admin';
    $password = 'quaiantique';
} else {
    $host_name = 'db5013125203.hosting-data.io';
    $database = 'dbs11016128';
    $user_name = 'dbu1243084';
    $password = 'quai.AntiqueArnaud3';
}
```

Figure 4 - Extrait du fichier Info_db.php

```php
$createAllergicTable = 'CREATE TABLE IF NOT EXISTS allergic(
    allergic_id INT AUTO_INCREMENT,
    allergic_name VARCHAR(50) NOT NULL,
    allergic_food TEXT NOT NULL,
    PRIMARY KEY(allergic_id)
);';
```

```php
include './sys/info_db.php';

try {
    $pdo = new PDO("mysql:host=$host_name; dbname=$database;", $user_name, $password);
    $pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
    $pdo->setAttribute(PDO::ATTR_DEFAULT_FETCH_MODE, PDO::FETCH_OBJ);

    $pdo->exec($createAllergicTable);
    $pdo->exec($createFoodTable);
    $pdo->exec($createFoodAllergicTable);
    $pdo->exec($createCategorieFoodTable);
    $pdo->exec($createSubcategorieFoodTable);
    $pdo->exec($createDischesTable);
    $pdo->exec($createHaveFoodTable);
    $pdo->exec($createMenuTable);
    $pdo->exec($createHaveMenuTable);
    $pdo->exec($createTableImage);
    $pdo->exec($createTableCalendar);

    $pdo->exec($createRoleTable);
    $pdo->exec($createUsersTable);
    $pdo->exec($createProfilTable);
    $pdo->exec($createTriggerUserToProfil);

    $pdo->exec($createPlanning);
} catch (PDOException $e) {
    echo "Connection échoué: " . $e->getMessage();
}
```

Figure 5 - Extrait du fichier db.php

Maintenant que ceci est réalisé, je pourrais effectuer mes tests quand la partie back-end sera développée.

5.2 Création d'un CRUD

D'après le cahier des charges, le client souhaitait pouvoir gérer les données qu'il afficherait sur les pages de son site. Pour réaliser ceci, j'ai créé un back office uniquement visible par le chef constitué de son compte administrateur avec des droits d'accès plus élevés, en utilisant son adresse mail.

Pour que la page du back office soit uniquement accessible par l'administrateur, j'ai réalisé une vérification dans la barre de navigation du site, qui vérifie le rôle du compte connecté avec une requête **SELECT**, cette requête prend en données l'id du rôle enregistré sur la table de l'utilisateur pour connaitre les informations du rôle qui lui est lié. Si le nom du rôle correspond à admin alors il pourra accéder au panel administrateur.

```php
if (isset($_SESSION['auth']->user_email)) : ?>
    <a id="button_header" href="./panel_user.php" class="nav-link" style="color: #e8eddf;">Panel</a>
<?php endif;
$reqSelectRole = $pdo->prepare('SELECT * FROM role WHERE role_id = ?');
$reqSelectRole->execute([$_SESSION['auth']->user_role_id]);
$selectRole = $reqSelectRole->fetch();

if ($selectRole->role_name == 'admin') : ?>
    <a id="button_header" href="./panel.php" class="nav-link" style="color: #e8eddf;">Gestion</a>
<?php endif; ?>
```

Figure 6 - Extrait de header.php

Une fois ceci fait, j'ai développé le **CRUD** (Create, Read, Update & Delete) qui permettra la gestion des données à partir du back office. L'exemple suivant concerne les données des aliments. Pour commencer, il me fallait un visuel des données des aliments sur la page du back office, afin de réaliser ceci, j'ai créé un fichier food.php et je l'ai inclus sur la page.

```php
<div class="col-md-6 col-lg-5 d-flex flex-column align-items-center flex-wrap">
    <h3 class="py-3">Aliments</h3>
    <?php include './sys/food/food.php'
    ?>
</div>
```

Figure 7 - Extrait du fichier panel.php

READ :

Ensuite, j'ai créé un tableau pour afficher les aliments qui seront ajoutés dans ma base de données, j'y ajoute une condition : s'il n'y a aucune donnée enregistrée, alors cela affiche un message de non-présence de cette donnée.

Afin d'avoir une architecture **MVC** (Model, View, Controller), je commence par la View (Vue) par récupérer les données de mes aliments, j'effectue une requête **SELECT** (pour lire les données sélectionnées) sur la table aliment (food) avec un tri par nom (food_name).

```php
$reqFoodList = $pdo->prepare('SELECT * FROM food ORDER BY food_name');
$reqFoodList->execute();
$foodList = $reqFoodList->fetchAll();
```

Figure 8 - Extrait du fichier db_food.php

J'inclus cette requête dans mon fichier food.php, avec la fonction *include()*, afin de pouvoir boucler sur les éléments (mes aliments) du tableau que j'ai récupéré dans la variable **$foodList**. Ce fichier sera mon Model (modèle).

```php
include './sys/food/db_food.php';
```

```php
<form method="post" class="">
    <div class="border border-3 border-dark rounded-3" style="background-color: white ;">
        <div class=" d-flex flex-row  p-2 gap-1" style="background-color: #242423 ;color:white;">
            <p scope="col" class="pt-2">Nom des aliments</p>
            <p scope="col" class="pt-2 pe-3">Origine</p>
            <p scope="col" class="pt-2">Condition de vie</p>
            <p scope="col" class="pt-2">Allergènes</p>
            <p scope="col" class=""></p>
            <p scope="col" class=""> </p>
        </div>
        <div class="border border-1 d-flex flex-column flex-wrap" tabindex="0" style="overflow-x: scroll; height: 500px;">
            <?php
            foreach ($foodList as $food) {
                $reqFoodAllergic = $pdo->prepare('SELECT allergic_id FROM food_allergic WHERE food_id = ?');
                $reqFoodAllergic->execute([$food->food_id]);
                $foodAllergic = $reqFoodAllergic->fetch();
                if ($foodAllergic) {···
                }

                echo '<div class="d-flex flex-row align-items-center justify-content-between border p-1 px-2">
                 <p class="pt-3 px-1">' . $food->food_name . '</p>';
                if ($food->food_origin) {···
                } else {···
                }

                if ($food->food_breeding) {···
                } else {···
                }

                if ($foodAllergic) {···
                } else {···
                }
                echo '<p class="pt-3 px-1"><input type="checkbox" name="checkbox_delete_food[]" value="' . $food->food_id . '"></p>
                </div>';
                unset($allergicName);
            }
            ?>
        </div>
    </div>
    <div class="d-flex flex-wrap gap-2 py-2 justify-content-center">
        <input class="btn" style="background-color: #242423 ;color:white;" type="submit" name="add_food" value="Ajouter">
        <input class="btn" style="background-color: #242423 ;color:white;" type="submit" name="delete_food" value="Supprimer">
    </div>
</form>
```

Figure 9 - Extrait du fichier food.php

J'ai ensuite ajouté un formulaire avec la méthode POST pour qu'il puisse transmettre les données que je souhaite ajouter au serveur, qui seront récupérées dans la partie back-end.

```php
if (isset($_POST['add_food'])) {
    if (isset($_GET)) {
        unset($_GET['food']);
    }
    echo '<form method="post" class="d-flex gap-2 flex-column align-items-center">
        <input class="form-control" type="text" name="add_food_name" placeholder="Aliment">
        <input class="form-control" type="text" name="add_food_origin" placeholder="Traçabilités des produits">
        <select class="form-select" name="add_food_breeding">
            <option value="null" selected>Condition de vie</option>
            <option value="1">Élevage</option>
            <option value="2">Sauvage</option>
        </select>
        <select class="form-select" name="select_food_allergic">
            <option value="null">Allergènes</option>';
    foreach ($allergicListBd as $allergic) {
        echo '<option value="' . $allergic->allergic_id . '">' . $allergic->allergic_name . '</option>';
    }
    echo '</select>
        <input class="btn mb-2" style="background-color: #242423 ;color:white;" type="submit" name="validate_add_food">
        <input class="btn mb-2" style="background-color: #242423 ;color:white;" type="submit" id="close_post_food" value="Annuler">
    </form>';
}
```

CREATE :

Ces informations seront traitées grâce au langage **PHP** afin de les envoyer dans ma base de données. Une vérification sera effectuée sur les données (vérification de champ vide) pour enregistrer celles-ci dans la base de données, avec la requête **INSERT**. Ce fichier sera mon Controller (contrôleur).

```php
if (isset($_POST["validate_add_food"])) {
    if (!empty($_POST['add_food_name'])) {
        $reqVerifyFood = $pdo->prepare('SELECT * FROM food WHERE food_name = ?');
        $reqVerifyFood->execute([ucfirst($_POST['add_food_name'])]);
        $verifyFood = $reqVerifyFood->fetch();

        if (!$verifyFood) {
            if (!empty($_POST["add_food_origin"]) && $_POST["add_food_breeding"] != 'null') {
                $reqAddFood = $pdo->prepare('INSERT INTO food SET food_name = ?, food_origin = ?, food_breeding = ?');
                $reqAddFood->execute([ucfirst($_POST['add_food_name']), ucfirst($_POST["add_food_origin"]), $_POST["add_food_breeding"]]);
            } elseif (!empty($_POST["add_food_origin"])) {
                $reqAddFood = $pdo->prepare('INSERT INTO food SET food_name = ?, food_origin = ? ');
                $reqAddFood->execute([ucfirst($_POST['add_food_name']), ucfirst($_POST["add_food_origin"])]);
            } else {
                $reqAddFood = $pdo->prepare('INSERT INTO food SET food_name = ? ');
                $reqAddFood->execute([ucfirst($_POST['add_food_name'])]);
            }
            // header('location: panel.php');
            $urlLogin = "panel.php";
            echo '<script type="text/javascript">window.location.href="' . $urlLogin . '";</script>';
        } else {
            $errors['verify_food'] = "L'aliment existe déjà";
        }

        if ($_POST["select_food_allergic"] != 'null') {
            $idFood = $pdo->lastInsertId();
            $reqInsertFoodAllergic = $pdo->prepare('INSERT INTO food_allergic SET food_id = ?, allergic_id = ?');
            $reqInsertFoodAllergic->execute([$idFood, $_POST["select_food_allergic"]]);
        }
    } else {
        $errors['form_add_food'] = "Vous n'avez pas donner de nom d'aliment";
    }
}
```

Figure 10 - Extrait du fichier food_action.php

Il me faut ensuite ajouter les opérations de modification, suppression des aliments afin de correspondre au CRUD et avoir une gestion complète de mes aliments.

UPDATE :

Mise à jour des aliments grâce à la requête UPDATE. Les données des aliments seront modifiées après soumission du formulaire ayant pour attribut la méthode POST, après comparaison avec les anciennes données et vérifications sur les champs pour envoyer de bonnes données.

```php
if (isset($_POST['modify_food'])) {
    $reqVerifyFoodAllergic = $pdo->prepare('SELECT * FROM food_allergic WHERE food_id = ?');
    $reqVerifyFoodAllergic->execute([$_GET['food']]);
    $verifyFoodAllergic = $reqVerifyFoodAllergic->fetch();

    if ($_POST["modify_food_allergic"] != "null" && $verifyFoodAllergic) {
        //modify foodAllergic
        $reqUpdateFoodAllergic = $pdo->prepare('UPDATE food_allergic SET allergic_id = ? WHERE food_id = ?');
        $reqUpdateFoodAllergic->execute([$_POST["modify_food_allergic"], $_GET['food']]);
    } elseif ($_POST["modify_food_allergic"] != "null") {
        //insert foodAllergic
        $reqInsertFoodAllergic = $pdo->prepare('INSERT INTO food_allergic SET food_id = ?, allergic_id = ?');
        $reqInsertFoodAllergic->execute([$_GET['food'], $_POST["modify_food_allergic"]]);
    } else {
        $reqDeleteFoodAllergic = $pdo->prepare("DELETE FROM food_allergic WHERE food_id = ?")->execute([$_GET['food']]);
    }

    if (isset($_POST["modify_food_origin"]) && $_POST["modify_food_breeding"] != 'null') {
        $reqUpdateFood = $pdo->prepare('UPDATE food SET food_name = ?, food_origin = ?, food_breeding = ? WHERE food_id = ?');
        $reqUpdateFood->execute([ucfirst($_POST["modify_food_name"]), ucfirst($_POST["modify_food_origin"]), $_POST["modify_food_breeding"], $_GET['food']]);
    } elseif (isset($_POST["modify_food_origin"])) {
        $reqUpdateFood = $pdo->prepare('UPDATE food SET food_name = ?, food_origin = ? WHERE food_id = ?');
        $reqUpdateFood->execute([ucfirst($_POST["modify_food_name"]), ucfirst($_POST["modify_food_origin"]), $_GET['food']]);
    } else {
        $reqUpdateFood = $pdo->prepare('UPDATE food SET food_name = ? WHERE food_id = ?');
        $reqUpdateFood->execute([ucfirst($_POST["modify_food_name"]), $_GET['food']]);
    }
    if (isset($_GET)) {
        // unset($_GET['food']);
        // header('location: panel.php', true);
        $urlLogin = "panel.php";
        echo '<script type="text/javascript">window.location.href="' . $urlLogin . '";</script>';
    }
}
```

Figure 11 - Extrait du fichier food_action.php

DELETE :

Suppression des données de l'aliment grâce à la requête DELETE.

Pour supprimer les aliments, j'ai mis en place un bouton *checkbox* qui permettra au chef du restaurant de supprimer plusieurs aliments en même temps. Avec un message de confirmation de suppression avec le nom de l'aliment supprimé.

```php
if (isset($_POST['delete_food'])) {
    if (!empty($_POST["checkbox_delete_food"])) {
        foreach ($_POST["checkbox_delete_food"] as $deleteFood) {
            $reqHaveFood = $pdo->prepare("SELECT * FROM have_food WHERE food_id = ?");
            $reqHaveFood->execute([$deleteFood]);
            $haveFood = $reqHaveFood->fetch();

            $reqNameDishes = $pdo->prepare("SELECT * FROM dishes WHERE dishes_id = ?");
            $reqNameDishes->execute([$haveFood->dishes_id]);
            $nameDishes = $reqNameDishes->fetch();
            $reqNameFood = $pdo->prepare('SELECT * FROM food WHERE food_id = ?');
            $reqNameFood->execute([$deleteFood]);
            $nameFood = $reqNameFood->fetch();
            if (!$haveFood) {
                $reqDeleteFoodAllergic = $pdo->prepare('DELETE FROM food_allergic WHERE food_id = ?')->execute([$deleteFood]);
                $reqDeleteFood = $pdo->prepare('DELETE FROM food WHERE food_id = ?')->execute([$deleteFood]);
                $_SESSION['flash']['danger'] .= "L'aliment " . $nameFood->food_name . " a été supprimé <br>";
            } else {
                $errors['no_delete_food'] .= "Vous ne pouvais pas supprimer l'aliment <b>" . ucfirst($nameFood->food_name) . "<
            }
        }
    } else {
        $errors['delete_food'] = "Vous n'avez pas sélectionné d'aliment(s) a supprimer";
    }
    if (isset($_GET)) {
        unset($_GET['food']);
    }
}
```

Figure 12 - Extrait du fichier food_action.php

Ce **CRUD** permettra au chef du restaurant la gestion des allergènes sur les aliments, les aliments pourront être utilisés pour les plats, qui eux seront affichés sur une page visible par les clients.

6. Développement de la partie front-end

6.1 Création de la maquette, charte graphique et la feuille de style

Afin de fournir au client un aperçu du projet, une charte graphique et pour définir la feuille de style du site, j'ai utilisé **Figma** (un logiciel de design en ligne) pour les concevoir.

DES EXEMPLES SERONT PRESENTES DANS LES ANNEXES PAGE 28

6.2 Mise en place de l'environnement de développement

Pour démarrer mes fichiers **HTML**, j'utilise la balise *<!DOCTYPE html>* et la balise <html lang='fr'> pour préciser la langue par défaut. Ensuite dans la balise <head>, je vais y mettre toutes les métadonnées, lien des script ou feuille de style, titre et description du site. Exemple de métadonnées : l'encodage des caractères en **utf-8**, ou le ratio entre la taille de l'écran et la zone d'affichage.

J'ai choisi le framework CSS Bootstrap pour styliser mon projet, car il fournit des classes CSS préfètes, intuitives et accès responsive (adaptable à toute tailles d'écrans), ainsi que de nombreux plug-ins JavaScript. Ce framework m'a permis de gagner beaucoup de temps pour agencer et styliser les éléments mon site.

Pour l'utiliser, j'ai choisi de l'installer sur le projet afin que le site ne rencontre pas de problème de version. Afin de l'installer, j'ai utilisé la commande *npm install bootstrap@5.2.0,* que je lierai à mon HTML avec la balise <link> qui se situera dans la balise <head>, pour le CSS, et avec la balise <script> pour le bundle JS, qui elle se situera dans ma balise <footer> afin d'être exécuté en dernier.

6.3 La navigation au sein du site

La navigation au sein du site se fera via la barre de navigation, située en en-tête du site qui dirigera l'utilisateur sur les pages nécessaires au restaurant. Celle-ci contiendra les liens vers les pages nécessaires, ces liens seront dans une balise HTML <a>.

Sur la page de gestion, disponible uniquement par l'administrateur, une barre de menu latérale sera disponible afin de naviguer sur les différents menus de gestion. Cette barre se modifiera selon l'appareil sur lequel on se rend sur le site, le bouton de sous menu fera disparaitre les noms du sous menu, afin de rendre l'expérience utilisateur plus agréable.

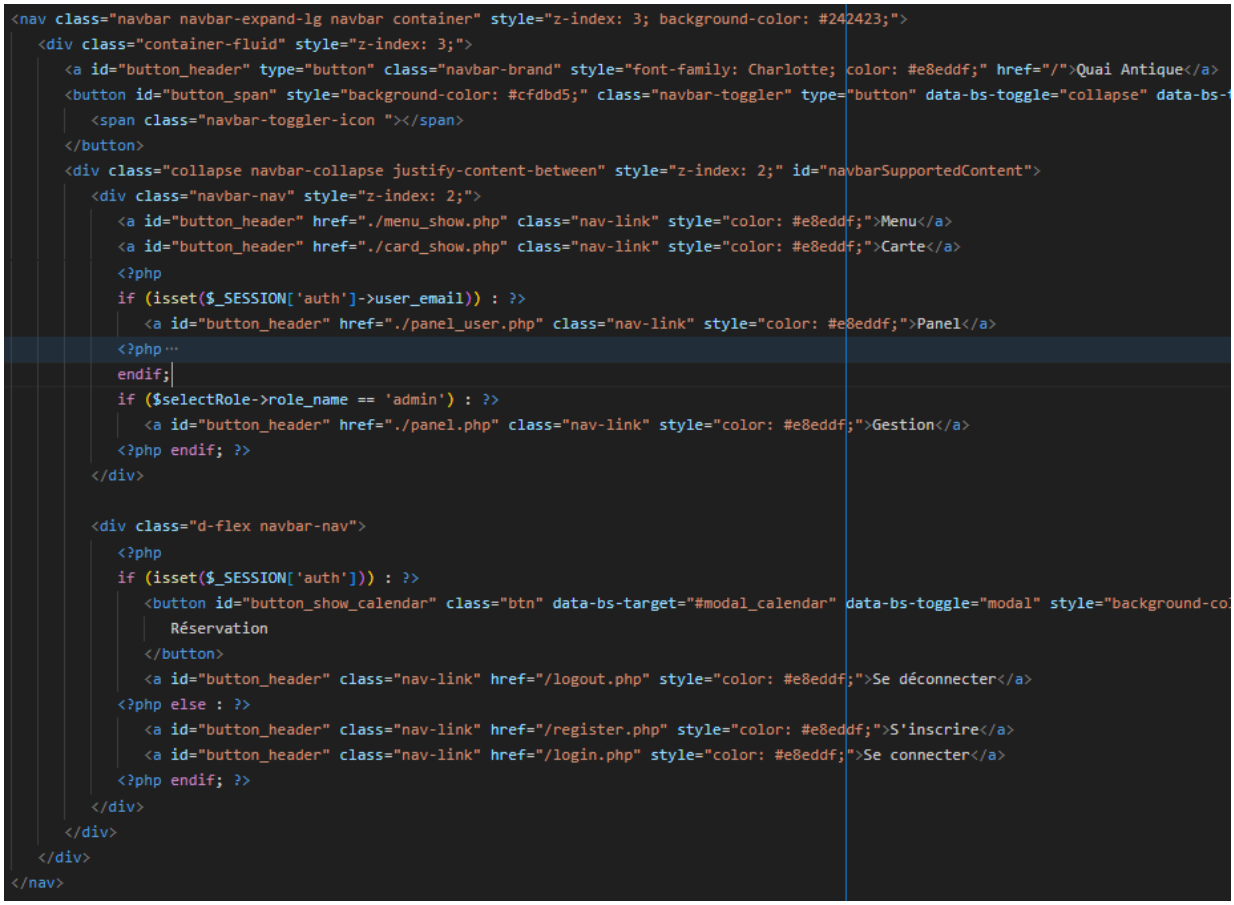


Figure 13 - Extrait du fichier header.php

6.4 Les fonts

Pour la sélection de la police d'écriture, j'ai sélectionné le style sur le site dafont.com que j'ai téléchargé et mis sur le projet. Cette sélection était faite à partir des attentes du cahier des charges.

```css
@font-face {
    font-family: Charlotte;
    src: url(../fonts/charlotte.ttf);
    font-weight: regular;
    font-display: swap;
    /* color: #e8eddf; */
}
```

J'ai ensuite créé mon fichier *font.css* afin de pouvoir l'utiliser comme style CSS dans mes fichiers.

Je n'ai plus qu'à lier ma feuille de style dans la balise <head> avec la balise <link>, en remplissant ses attributs, tel que *rel='stylesheet'* pour définir que c'est une feuille de style et *href=''* pour indiquer le chemin du fichier. Afin de l'utiliser sur du texte dans le <body>, en utilisant l'attribut style d'une balise et la déclaration CSS *font-family*.

6.5 Formulaire de réservation

Afin de répondre au cahier des charges pour la réservation des clients, je devais créer un formulaire de réservation, accessible depuis n'importe quelle page du site. Pour cela, j'ai souhaité le rendre dynamique, j'ai donc utilisé le langage de programmation **JavaScript**.

Je me suis posé beaucoup de questions afin de récupérer des dates précises sur du long terme (calendrier à jour en fonction des années), afin que le client n'ait pas de problème dans les années à venir. Je me suis interrogé sur la possibilité de faire des calculs sur une date sachant que le nombre de jours total d'une année est différent d'une autre.

Avec ces recherches, j'ai pris connaissance du calendrier Julien et du calendrier Grégorien, ce qui a enrichi ma culture.

Cela ne m'a pas aidé pour autant à réaliser mon code, pour y arriver la question suivante était :

- *Est-ce que je me crée une classe qui contiendra mes fonctions pour récupérer toutes les informations d'un mois, pour pouvoir les insérer dans ma base de données ?*

Au final, j'ai utilisé le champ <input> de type date, qui permet de rentrer une date avec le clavier ou de la sélection à partir d'un calendrier. J'ai pu rentrer des paramètres pour que l'utilisateur ne sélectionne que les jours qui seront insérés en base de données.

J'ai mis en place un planning avec le nombre de réservation disponible par tranche d'horaires dans une base de données, pour pouvoir récupérer le nombre disponible lors de la réservation, afin d'effectuer des opérations dessus (exemple : lui soustraire le nombre souhaité par le client).

Pour réaliser ceci, j'ai également utilisé le framework **Bootstrap** pour définir ma fenêtre modale, avec les balises et les classes qu'il fournissait, car l'agencement qu'il proposait me convenait. Je n'ai plus qu'à le rendre accessible sur toutes les pages en ajoutant un bouton sur la barre de navigation, qui sera disponible sur chacune des pages.

Dans ma fenêtre modale, j'ai créé un formulaire <form> avec un <input> de type *date*, avec l'id « calendar » et un bouton avec l'id « validate_calendar ». Avec le langage **JavaScript** et son API **DOM** (Document Object Model), je vais enregistrer l'input avec l'id « calendar » grâce à la méthode *getElementById()* dans une variable, afin d'y récupérer sa valeur dans l'étape suivante. J'ai ajouté un écouteur d'évènement sur le bouton grâce à la méthode *addEventListener()*, récupéré dans une variable grâce à la méthode présentée précédemment, qui déclenchera une fonction après chaque clic du bouton, afin de récupérer la valeur du calendrier souhaité par l'utilisateur. Cette fonction ajoutera également les tranches horaires attendu par le cahier des charges.

```
const displayCalendar = document.getElementById('display_button')
const validateCalendar = document.getElementById('validate_calendar')
const inputDate = document.getElementById('calendar')

validateCalendar.addEventListener('click', (event) => {
    const inputCalendar = inputDate.value
    displayCalendar.classList.replace('d-none', 'd-flex')
    returnNumberReservationPart1(inputCalendar)
    returnNumberReservationPart2(inputCalendar)
})
```

```
function returnNumberReservationPart1(dateCalendar) {
    partSelect1 = 1
    xhrPart1 = new XMLHttpRequest()
    if (dateCalendar != "") {
        xhrPart1.onreadystatechange = function() {
            if (this.readyState == 4 && this.status == 200) {
                let test = xhrPart1.responseText
                document.getElementById('number_reservation1').textContent = "Nombre de repas restant : " + test
                const inputGet1 = document.createElement('input')
                inputGet1.type = "hidden",
                    inputGet1.value = test,
                    inputGet1.id = "input_get_number1"
                const firstPart = document.getElementById('first_part')
                firstPart.append(inputGet1)
            }
        }
        xhrPart1.open("POST", "./sys/calendar/calendar_get.php", true)
        xhrPart1.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");
        xhrPart1.send("date=" + dateCalendar + "&part=" + partSelect1)
    }
    firstPart.insertAdjacentElement('beforeend', numberPart1)
}
```

Après la sélection de la tranche horaire souhaitée par l'utilisateur, un formulaire apparaitra avec les données sur le nombre de réservation, si l'utilisateur l'a renseigné, sinon il devra le remplir lui-même pour donner un nombre de place qu'il souhaite réserver. J'ai également créé une fonction qui récupère le nombre de place disponible selon le jour et le reste des disponibilités sans rechargement de page.

La fonction pour soustraire le nombre de réservation voulu par l'utilisateur, s'exécutera sans rechargement de la page grâce à la classe *XMLHttpRequest()* qui interagira avec ma partie back end. Cette classe exécutera une fonction lors du changement d'état de la requête qui se déroulera lors de la validation de réservation par l'utilisateur et renverra le résultat du nombre restant. Cette fonction est donc asynchrone, elle ne s'exécute pas en même temps que le fichier et ne bloque pas le reste des actions sur le site.

```
function insertData1(int, dateSelect, partSelect, emailUser, hoursSelect) {
    xhr = new XMLHttpRequest()
    if (int != "") {
        xhr.onreadystatechange = function() {
            if (this.readyState == 4 && this.status == 200) {
                let test = xhr.responseText
                document.getElementById('number_reservation1').textContent = "Nombre de repas restant : " + test
            }
        }
        xhr.open("POST", "./sys/calendar/calendar_post.php", true)
        xhr.setRequestHeader("Content-Type", "application/x-www-form-urlencoded")
        xhr.send("number_reservation=" + int + "&date=" + dateSelect + "&part=" + partSelect + "&email=" + emailUser + "&hours=" + hoursSelect)
    }
}
```

Suite à la validation de tous ces renseignements, le formulaire et les boutons disparaitront pour laisser place à un message de validation, et un email sera envoyé aux clients pour confirmer que la réservation a bien été prise en compte, avec les informations sur l'horaire et la date.

Cet envoie d'email est disponible grâce à la classe PHPMailer que j'ai installé sur mon projet avec la commande *composer require phpmailer/phpmailer*. Cette classe m'apporte de nombreuses possibilités, par exemple : gérer mon service mail que je souhaite lier, formater le contenu de l'email.

Pour utiliser cette classe, j'ai créé une adresse email au chef du restaurant que j'ai relié au site, afin que le site puisse envoyer des emails. Il m'a fallu également renseigner les informations SMTP du serveur mail choisit (l'hôte, le protocole et le ports), préremplir le texte à envoyer et ajouter les informations de l'envoyeur récupérées sur la session de connexion.

```php
try {
    $date = new DateTime();
    $dt = $date->createFromFormat('Y-m-d', $_POST['date']);
    $secret_m = 'byzloaokvucfvloe';
    $username_m = 'arnaud.michant@gmail.com';
    $username = 'Arnaud Michant';
    $subject = "Confirmation de reservation pour " . $dt->format('d-m-Y') . " à " . $_POST['hours'];
    $mailBody = "Bonjour, votre réservation a bien été enregistré nous somme heureux de vous acceuillir le " . $dt->format
    Merci de vous munir de la réservation sur votre téléphone.';

    //Server settings
    $mailReservation->SMTPDebug = 0; // debogage: 1 = Erreurs et messages, 2 = messages seulement
    $mailReservation->isSMTP();                                            //Send using SMTP
    $mailReservation->Host       = 'smtp.gmail.com';                     //Set the SMTP server to send through
    $mailReservation->SMTPAuth   = true;                                   //Enable SMTP authentication
    $mailReservation->Username   = $username_m;                             //SMTP username
    $mailReservation->Password   = $secret_m;                               //SMTP password
    $mailReservation->SMTPSecure = 'TLS';      // STARTTLS pour outlook      //Enable implicit TLS encryption
    $mailReservation->Port       = 587;                                    //TCP port to connect to; use 587 if you have s

    $mailReservation->smtpConnect();
    //Recipients
    $mailReservation->setFrom($username_m, $username);
    $mailReservation->addAddress(trim($_POST['email']));     //Add a recipient

    //Content
    $mailReservation->isHTML(true);    //Set email format to HTML
    $mailReservation->Subject = $subject;
    $mailReservation->Body = $mailBody;


    $mailReservation->send();
    $_SESSION['flash']['success'] = "Un email de confirmation vous a été envoyer";
} catch (Exception $e) {
    echo "Le mail n'a pas été envoyer !";
    $errorMail = $mailReservation->ErrorInfo; //Envoyer le mail au propriétaire ou l'enregistrer dans un fichier de log
    $_SESSION['flash']['danger'] = "Le mail n'a pas été envoyer, avertir le chef Arnaud Michant";
}
```

DES EXEMPLES SERONT PRESENTES DANS LES ANNEXES PAGE 28

6.6 Écrans développés

Afin de rendre accessible le site dans un format adapté pour chaque écran, j'ai utilisé les classes utilitaires avec les différents **breakpoints** (points d'arrêt) de Bootstrap, afin de pouvoir travailler sur chaque élément que je souhaite adapter. Pour réaliser ceci, j'ai pensé le style « mobile first » afin de travailler en premier sur le rendu mobile et de finir sur le rendu ordinateur.

Et voici le détail de quelques écrans développés pour le restaurant.

DES CAPTURES D'ECRAN SERONT DISPONIBLES EN ANNEXE PAGE 28.

- La Page d'accueil :

La page d'accueil contiendra une présentation du restaurant, des images qui afficheront les créations du chef, ainsi que les coordonnées du restaurant. Ces images seront modifiables par le chef du restaurant. Elle aura également une barre de navigation, afin de pouvoir se rendre sur les autres pages.

- Login :

Il contiendra une page pour s'inscrire comme utilisateur du restaurant afin de pouvoir réserver et une page pour se connecter.

  - Inscription :

    La page d'inscription demande des informations sur l'utilisateur avec un email et un mot de passe. Une confirmation de création de compte sera envoyée par email.

  - Connexion :

    Une fois que le client aura confirmé son inscription via le lien qui lui est envoyé par email, il pourra se connecter sur la page de connexion pour accéder aux fonctionnalités disponibles une fois connecté.

- Panel utilisateur :

Ce panel contiendra toutes les données de l'utilisateur, afin qu'il puisse les modifier s'il le souhaite, et ajouter un nombre par défaut de place pour réserver lors d'une prochaine réservation.

- Panel administrateur :

Ce panel permettra à l'administrateur de gérer toutes les fonctionnalités attendues dans le cahier des charges, par exemple : le gestionnaire d'image pour la page d'accueil, gestion des menus et cartes, gestion des aliments et des allergènes, gestion du planning.

  - Accueil :

    C'est dans ce sous menu que le chef arrive quand il se rend sur la page de gestion. Ce sous menu gère les horaires d'ouverture du restaurant, ces horaires s'afficheront dans le pied de page.

- Menu et cartes :

  Ce sous menu servira à la gestion de la partie alimentaire et la partie allergènes, qui sera visible par le client.

- Gestionnaire d'images :

  Ce sous menu, quant à lui, sera utilisé pour la gestion des images. Les images téléchargées sur le projet seront affichées et le chef pourra les afficher sur la page d'accueil du site.

- **Menus et cartes :**

Le site a une page pour les plats à la carte, il a également une page affichant les menus. Ces deux pages seront remplies par le chef du restaurant dans le back office.

Fonctionnalité la plus représentative : L'authentification

1. Jeu d'essai

Afin que l'utilisateur accède à toutes les fonctionnalités que le site propose, j'ai réalisé un système d'inscription et de connexion. J'ai donc réalisé une page d'inscription, et une page de connexion, et un système de déconnexion.

Sur les données passées au serveur, je m'assure de vérifier chaque entrée transmise par les utilisateurs, car il faut toujours se méfier des entrées utilisateur.

INSCRIPTION :

Sur cette page, une redirection est faite si l'utilisateur est déjà connecté. Pour la page d'inscription, j'ai ajouté un formulaire avec la méthode POST, avec plusieurs champs à rentrer. Par exemple : le prénom, le nom, l'email, le mot de passe, etc.

Afin de ne pas rencontrer de problème sur des faille XSS, je vérifie si mes champs ne sont pas vides et s'ils correspondent à l'expression régulière attendue. Il y a une vérification pour voir si l'email n'est pas déjà relié à un utilisateur. J'ai ensuite transformé les données des champs du formulaire grâce à plusieurs fonctions. Par exemple : htmlspecialchars() convertit les caractère spéciaux en entités HTML, stripslashes() supprime les antislashs. Ses fonctions sont toutes appliquées à la donnée qui doit être traitée, via la fonction que j'ai créée *validateData()* qui prend en arguments $data (la données à traiter).

Une fois que les données auront passées ces étapes de vérification, elles seront transmises à la base de données. Pour ce qui est du mot de passe, il n'est pas envoyé en clair, il est haché avant l'envoi vers la base de données, grâce à la fonction *password_hash().* Cette fonction hache le mot de passe grâce à l'algorithme bcrypt(), et elle génère un *salt* (grain de sel) aléatoire pour renforcer le hachage du mot de passe. Dans les données de l'utilisateur, il y a un token (jeton) de confirmation de compte qui empêchera l'utilisateur de se connecter s'il ne le confirme pas avec son adresse email.

Les données de l'utilisateur s'enregistreront en 2 étapes, les informations de connexion, les tokens, les informations de confirmation de compte s'enregistreront dans une table appelée *users*. Tandis que les données personnelles, tel que le prénom, le nom, l'âge, la civilité, seront quant à elles enregistrées sur une table appelée *profil.*

Une fois ces étapes réalisées, le site enverra un mail pour que l'utilisateur confirme la création du compte grâce à la classe **PHPMailer**, avec les données d'email renseignées par l'utilisateur. L'utilisateur n'aura plus qu'à se rendre sur le lien contenant le token de vérification, qui lui a était envoyé et le compte sera confirmé. Il pourra ensuite se connecter via la page de connexion.

DES EXEMPLES SERONT PRESENTES DANS LES ANNEXES PAGE 28

CONNEXION :

Sur cette page, une redirection est faite si l'utilisateur est déjà connecté. Sur la page de connexion, j'utilise un formulaire avec la méthode POST avec un champ email, un champ mot de passe, un **reCAPTCHA v2**, et une case à cocher pour rester identifié si on quitte le site web.

Lors de la soumission du formulaire, j'applique une vérification pour voir si les champs sont vides, afin de renvoyer un message à l'utilisateur, si c'est le cas. Si ce n'est pas le cas je vérifie si l'email est correct grâce la fonction filter_var() qui utilise le filtre FILTER_VALIDATE_EMAIL. Si l'email est correct, je passe à la vérification suivante.

Je vérifie également que le reCAPTCHA me renvoie bien une réponse valide. Pour obtenir un traitement de cette donnée, j'ai créé une classe (un objet) Recaptcha qui va réaliser les traitements sur le reCAPTCHA et me retourner si la réponse est valide ou non.

Pour que les traitements soient effectués, la classe aura besoin d'une clé secrète fournie par le service, lors de la création du reCAPTCHA pour le site. Cette clé secrète, je la passe dans le constructeur de ma classe afin que la classe contienne la donnée nécessaire pour le reste des opérations, et j'instancie l'objet, en l'enregistrant dans une variable, par exemple : « *$recaptcha = new Recaptcha($secret)* ». Ensuite, je vais récupérer le code que me fournira le reCAPTCHA à chaque soumission, grâce à la superglobal $_POST['g-recaptcha-response']. Grâce à ces deux données je crée une opération pour lire l'url, qui me retournera un objet JSON que je décoderai avec la fonction *json_decode()*, afin de pouvoir la lire dans ma partie back-end, cette étape sera traitée lorsque j'appellerai la méthode $recaptcha->*checkCode($code)*. Je n'ai plus qu'à ajouter des conditions sur l'objet, qui m'est retourné afin de vérifier si l'étape du reCAPTCHA est bien valide (**true**).

```php
class Recaptcha
{
    private $secret;

    public function __construct($secret)
    {
        $this->secret = $secret;
    }

    public function checkCode($code)
    {
        if (empty($code)) {
            return false;
        }

        $url = "https://www.google.com/recaptcha/api/siteverify?secret=$this->secret&response=$code";
        $response = file_get_contents($url);

        if (empty($response) || is_null($response)) {
            return false;
        }

        $json = json_decode($response);
        return $json->success;
    }
}
```

Pour l'étape suivante, il me faudra vérifier que le mot de passe donné par l'utilisateur pour se connecter est bien valide. Vu que les mots de passe ne sont pas enregistrés en claire, et qu'ils sont hachés, je vais devoir utiliser la fonction *password_verify()*, qui vérifiera que le mot de passe enregistré en base de données et celui fournit par l'utilisateur, sont bien compatibles. Toutes les informations afin de vérifier le hachage, sont comprises dans l'algorithme, la méthode à utiliser pour comprendre le hachage ainsi que le *salt.*

Une fois toute ces étapes validées, l'utilisateur pourra enfin se connecter au site. Pour que l'utilisateur reste connecté sur toutes les pages du site, je vais stocker ses informations dans la superglobale **$_SESSION** comme ceci $_SESSION['auth] = $user. Cette variable sera liée au cookie de session, qui identifie l'utilisateur lors des requêtes au serveur. Pour charger les informations de l'utilisateur stockées dans mon fichier de session, j'initialiserai la session à chaque début de page avec la fonction *session_start()*.

Afin de rendre la connexion plus rapide pour l'utilisateur, j'ai créé un cookie grâce à la fonction *setcookie()* sur le navigateur, contenant un token. Ce token sera lui aussi enregistré en base de données, dans les informations de l'utilisateur. Lorsque l'utilisateur se rendra à nouveau sur le site, une fonction sera automatiquement lancée, si le cookie existe, afin de comparer le cookie et le token enregistré. Si le cookie et le token sont identiques, l'utilisateur sera de nouveau connecté.

DES EXEMPLES SERONT PRESENTES DANS LES ANNEXES PAGE 29

2. Recherches anglophones sur le reCAPTCHA

"Ayant abordé le reCAPTCHA rapidement et la sécurité qu'il apporte, je me suis documenté pour connaître l'aide qu'il peut apporter pour la sécurité des données des futurs clients.

To start using reCAPTCHA, you need to sign up for an API key pair for your site. The key pair consists of a site key and secret key. The site key is used to invoke reCAPTCHA service on your site or mobile application. The secret key authorizes communication between your application backend and the reCAPTCHA server to verify the user's response. The secret key needs to be kept safe for security purposes.

Each reCAPTCHA user response token is valid for two minutes and can only be verified once to prevent replay attacks. If you need a new token, you can re-run the reCAPTCHA verification.

After you get the response token, you need to verify it within two minutes with reCAPTCHA using the following API to ensure the token is valid. URL API:

- https://www.google.com/recaptcha/api/siteverify?secret=$secret&response=$response

The response is a JSON object. If the success key of the JSON object is valid, the verification of the reCAPTCHA will be valid as well."

« Pour commencer à utiliser le service de reCAPTCHA, il faut vous inscrire afin d'obtenir une clef API liée à votre site internet. La clef est constituée de deux parties : La clef du site en lui-même et une clef secrète. La clef du site est utilisée pour obtenir le service de reCAPTCHA sur votre site internet ou sur votre application mobile. La clef secrète, elle, autorise la communication entre le "backend" de votre application et le serveur du reCAPTCHA afin de vérifier la réponse de l'utilisateur. La clef secrète doit être protégée pour des raisons de sécurité.

Chaque jeton réponse des utilisateurs sur le reCAPTCHA n'est valide que deux minutes et ne peut être vérifie qu'une fois afin de prévenir d'attaques répétitives. Si vous avez besoin d'un autre jeton réponse, vous pouvez relancer la vérification reCAPTCHA.

Une fois que vous avez reçu le jeton réponse, vous devez vérifier en un maximum de deux minutes - avec le reCAPTCHA - en utilisant le clef API suivante afin de vérifier sa validité : https://www.google.com/recaptcha/api/siteverify?secret=$secret&response=$response

La réponse est un objet JSON. A savoir que si la clef success de l'objet JSON est valide, les vérifications du reCAPTCHA seront valides également. »

Conclusion

Concernant le projet, les fonctionnalités attendues sont présentes afin de permettre au chef du restaurant le bon fonctionnement de son site. Celui-ci est également déployé et fonctionnel, comportant quelques erreurs que je vais m'empresser de résoudre.

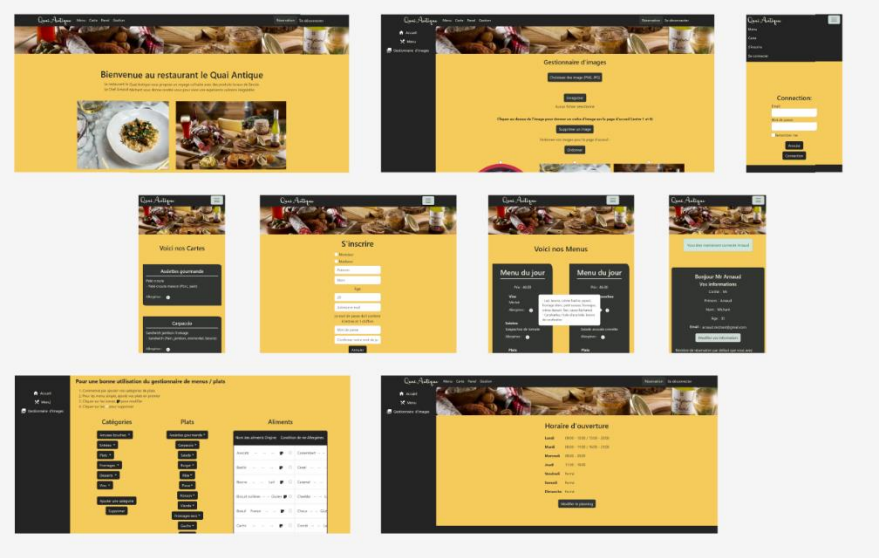
Sans connaissance du développement, et en participant à ce projet, j'ai pu tester mes compétences acquises tout au long de ma formation GRADUATE DEVELOPPER WEB FULL STACK (commencée en décembre 2021 avec une participation CPF-Compte Professionnelle de Formation et une prime de France Relance), afin de vérifier mes acquis et d'améliorer mon code informatique. J'ai apprécié d'apporter des fonctionnalités intuitives et dynamiques pour l'utilisateur, en souhaitant répondre aux attentes du client.

J'ai pris du plaisir, au fur et à mesure, de par la diversité des éléments, la complexité des recherches et la compréhension du métier ciblé, à développer ce site, afin d'être au plus juste du projet initial et de la demande des futurs clients.

Chaque partie technique, que ce soit la création de design d'interface, les fonctionnalités cachées, la conception de A à Z d'un site, que j'ai pu apprendre tout au long de cette formation, me conforte dans le choix de ce nouveau métier et de mon avenir dans le milieu informatique.

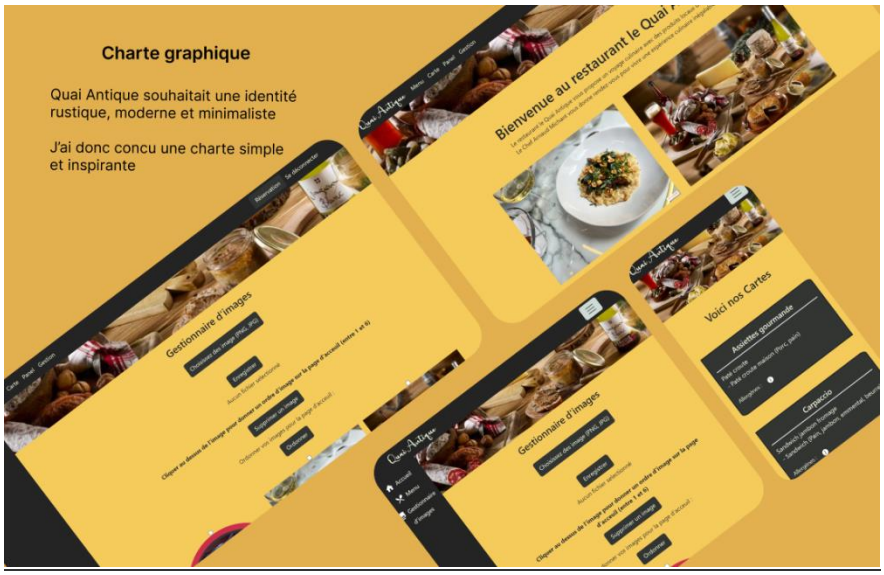
Annexes

Maquette :

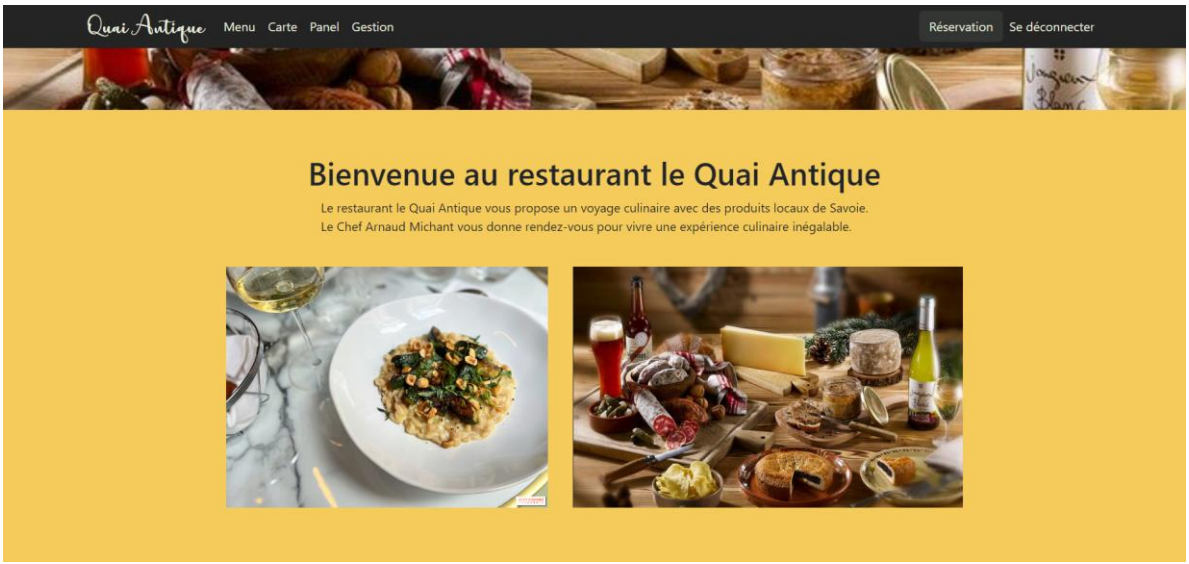


Feuille de style :

Charte graphique :

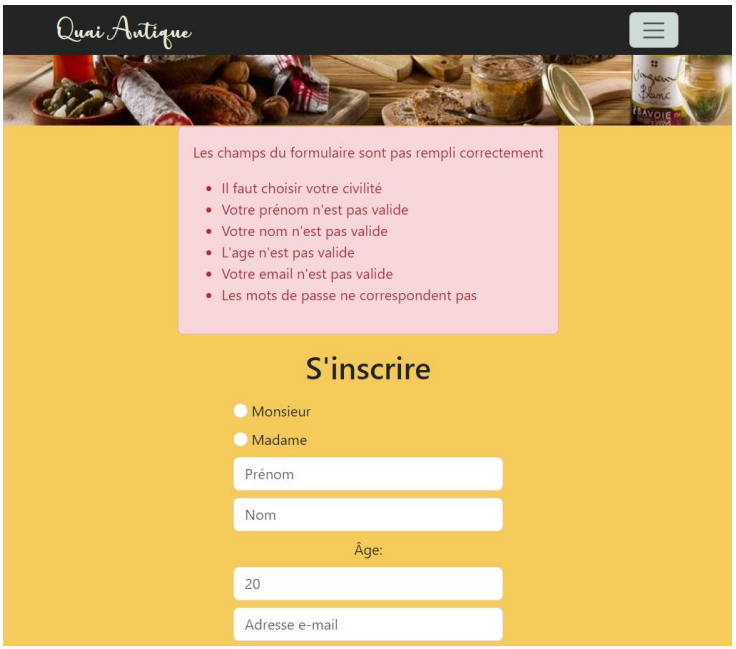


Page d'accueil :



Page d'inscription :

Page d'inscription erronée :



Page de connexion :

Page de connexion erronée :

Tableau de bord de l'utilisateur :

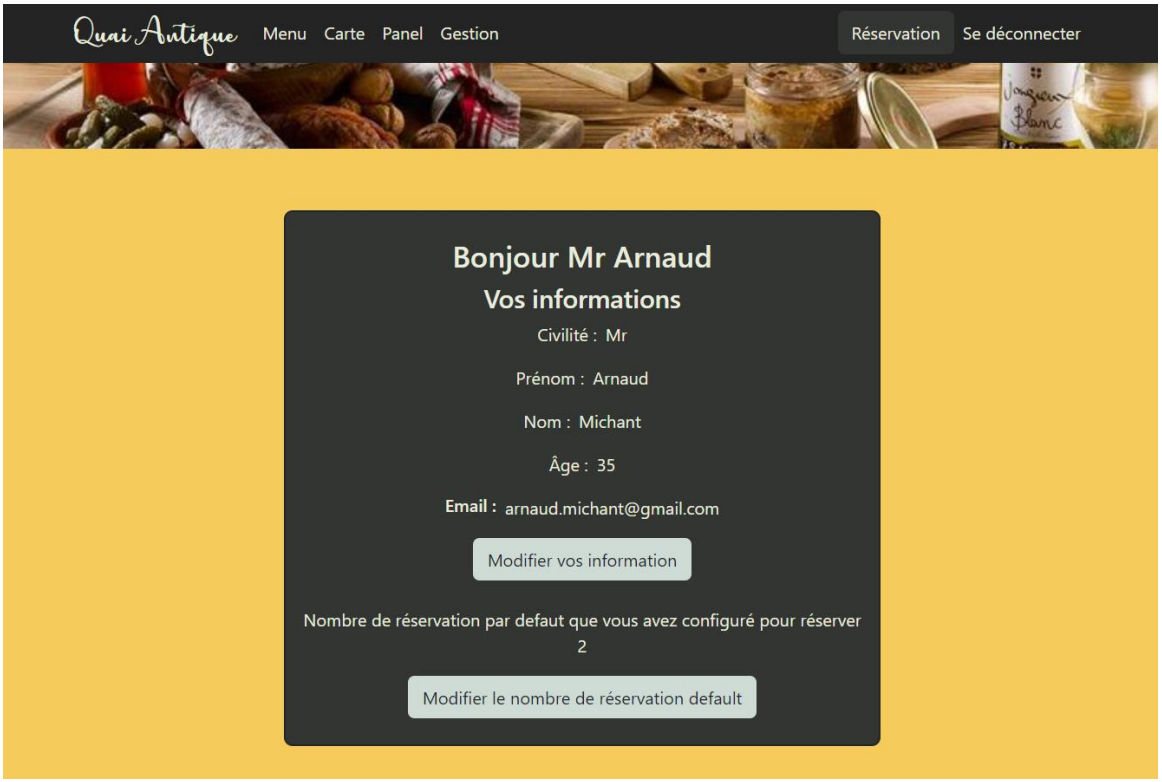
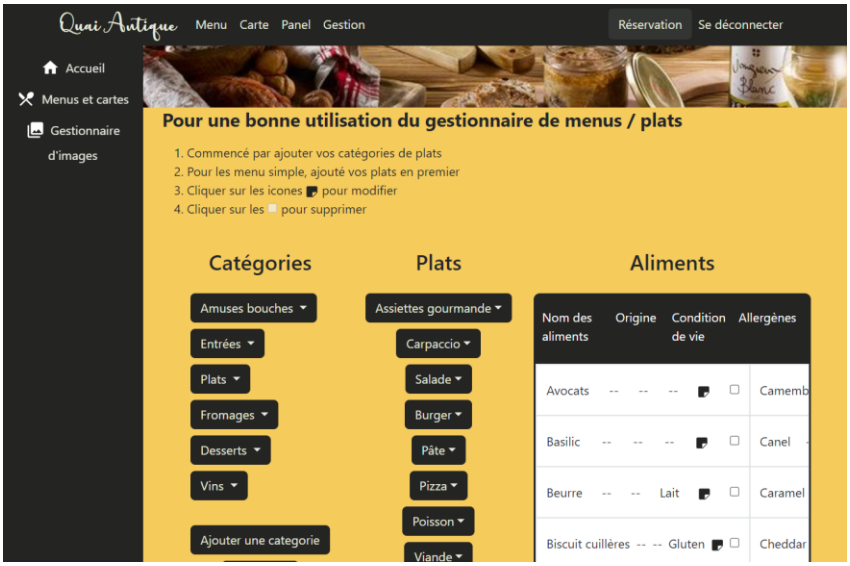


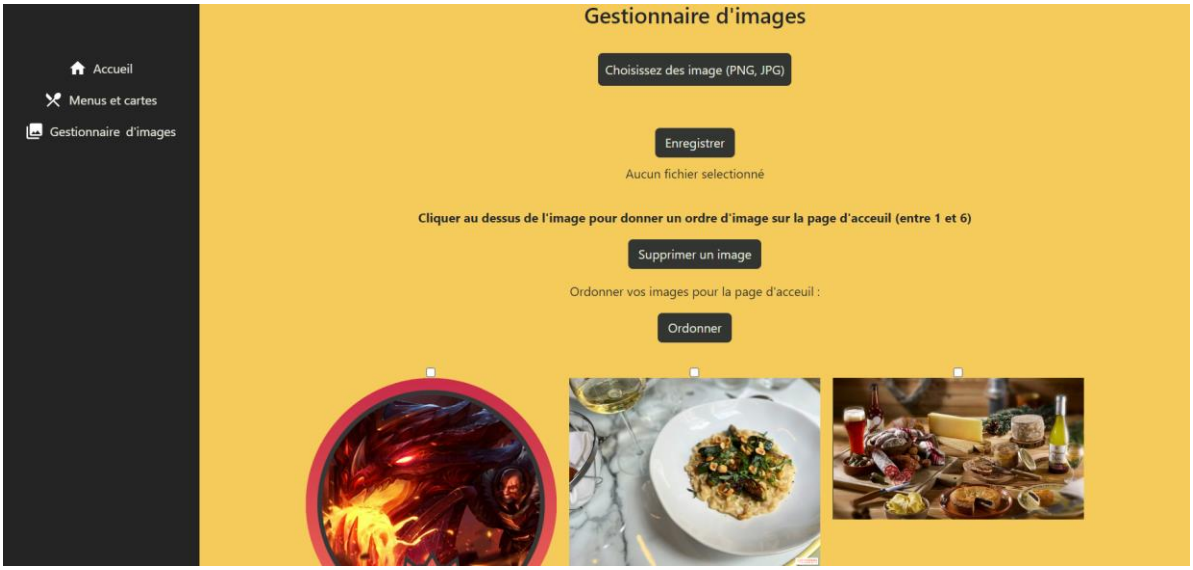
Tableau de bord administrateur :

Gestionnaire d'ouverture :

Gestionnaire alimentaire :



Gestionnaire d'images :



Page des menus :

Page des cartes :

Formulaire de réservation :