RAPPORT DE STAGE

Projet de Développement Web

Présenté par :
AIDORTSANG Tashi

Classe : BTS SIO SLAM
Année scolaire :2024 – 2025

REMERCIEMENTS

Ce projet a été réalisé de manière entièrement autonome.
Tout au long de sa création, j'ai dû apprendre, chercher,
tester, corriger et recommencer plusieurs fois.

Je n'ai reçu aucune aide extérieure directe durant cette période,
ce qui m'a obligé à me surpasser, à développer mes capacités
de recherche, de logique et de persévérance.

C'est grâce à cette démarche que j'ai pu acquérir des bases solides
en développement web, et mieux comprendre comment construire
un site de A à Z.

Je remercie néanmoins les plateformes en ligne, les documentations
et les outils comme Visual Studio Code, XAMPP, phpMyAdmin,
ainsi que les forums et tutoriels qui m'ont guidé, même indirectement.

Ce projet m'a appris à ne pas abandonner quand les choses
ne fonctionnaient pas, et m'a permis de mieux comprendre
la réalité du développement informatique.

Je suis fier d'avoir mené ce travail seul, et d'avoir construit
un site web fonctionnel pour un restaurant fictif, avec des
fonctions complètes comme la réservation, le formulaire de
contact, et une base de données reliée.

Présentation de l'organisme d'accueil : MOMO HUT

L'organisme d'accueil dans lequel j'ai réalisé mon projet est MOMO HUT, un restaurant situé à Central Parc – Place de Suze, 05100 Briançon.

Cette entreprise est spécialisée dans le domaine de la restauration, et plus précisément dans la cuisine tibétaine et asiatique. Le restaurant propose une carte variée composée de plats traditionnels comme les momos (raviolis tibétains faits maison), des woks, des riz sautés, des nouilles, ainsi que des desserts typiques de la région de l'Himalaya.

MOMO HUT offre une ambiance chaleureuse, moderne et familiale, avec une décoration soignée et une équipe accueillante. L'objectif est de faire découvrir une expérience culinaire authentique, tout en adaptant les recettes aux goûts locaux, avec des produits frais et de saison.

Mon tuteur de stage a été Monsieur Yeshi Gyamtso Aidortsang, le chef cuisinier et fondateur du restaurant. Bien qu'il ne travaille pas dans l'informatique, il m'a confié la mission de créer un site internet complet afin de moderniser l'image du restaurant et de faciliter la communication avec les clients.

Avant ce projet, le restaurant ne possédait aucun site web officiel. Le but de ma mission était donc de :

- créer une présentation claire de l'établissement,
- afficher le menu sous forme numérique,
- intégrer un système de réservation en ligne,
- et permettre aux clients de consulter les horaires d'ouverture, l'adresse ou encore les moyens de contact.

Ce projet m'a permis de découvrir comment répondre à des besoins réels d'une entreprise locale, avec des outils simples mais efficaces comme HTML, CSS, PHP, XAMPP et phpMyAdmin.

Objectifs du projet et Missions confiées

Objectifs du projet

Le projet avait pour but de concevoir un site web complet pour le restaurant MOMO HUT, situé à Briançon. L'établissement ne possédait pas de site officiel, ce qui limitait sa visibilité ainsi que la possibilité pour les clients d'accéder aux informations essentielles.
L'objectif principal était donc de développer un site permettant :

- de présenter l'établissement et ses spécialités culinaires,
- de consulter la carte sous forme visuelle et structurée,
- de permettre aux clients de réserver une table en ligne via un formulaire de réservation,
- d'intégrer un formulaire de contact,
- de rendre l'ensemble accessible sur tous les types d'appareils (ordinateur, tablette, téléphone).

Le site devait également offrir une navigation fluide, une interface simple et moderne, tout en restant facile à mettre à jour par la suite.

Missions réalisées

Dans le cadre de ce projet, plusieurs missions m'ont été confiées :

1. Création de l'architecture du site (HTML)

J'ai d'abord structuré le site à l'aide du langage HTML (HyperText Markup Language), qui permet de construire les pages web. Chaque page (accueil, menu, contact, réservation, etc.) a été codée à la main.ù

2. Mise en forme avec le langage CSS

Le langage CSS (Cascading Style Sheets) a été utilisé pour mettre en page les éléments HTML : couleurs, typographie, alignement, disposition, boutons, arrière-plans, etc.

3. Intégration d'un formulaire de réservation

J'ai développé un formulaire permettant aux visiteurs de choisir une date, une heure, un nombre de personnes, et de renseigner leurs coordonnées. Ces données sont ensuite enregistrées dans une base de données.

4. Développement d'un formulaire de contact

Un formulaire permet également aux visiteurs d'envoyer un message. Comme pour la réservation, les données sont stockées dans la base MySQL.

5. Utilisation de PHP pour la logique serveur

Le langage PHP (Hypertext Preprocessor) a été utilisé pour connecter les formulaires à la base de données. Il permet aussi d'afficher un message de confirmation lors de l'envoi réussi d'un formulaire.

6. Connexion à une base de données MySQL via phpMyAdmin

J'ai créé une base de données locale avec XAMPP et phpMyAdmin, comportant des tables telles que reservations et messages. Celles-ci reçoivent les données saisies par les utilisateurs.

7. Création d'une galerie d'images pour la carte

Les pages du menu ont été affichées sous forme d'images (issues d'un PDF), organisées en grille avec un effet de survol.

8. Responsive design

Le site a été conçu pour s'adapter à toutes les tailles d'écran grâce à des règles CSS spécifiques, afin d'être lisible et fonctionnel sur mobile, tablette ou ordinateur.

Outils et technologies utilisés

Durant mon projet de création de site web, j'ai utilisé plusieurs outils et langages de programmation. Voici une explication complète de chacun d'eux :

1. Visual Studio Code (VS Code)

VS Code est un **éditeur de texte** spécialement conçu pour les développeurs. Il permet d'écrire, organiser, corriger et tester du code dans plusieurs langages. Il est gratuit, léger, rapide, et propose des fonctionnalités utiles comme :

- la coloration du code (plus facile à lire),
- l'autocomplétion (suggestions pendant qu'on tape),
- un aperçu direct des fichiers (HTML, CSS, PHP, etc.).

2. HTML – HyperText Markup Language

HTML est le **langage de base** utilisé pour **structurer le contenu d'un site web**.
Il permet de créer les différentes parties d'une page :
titres, paragraphes, liens, images, formulaires, boutons, etc.

Exemple :
<h1>MOMO HUT</h1>
* Bienvenue sur notre site *
Ici, <h1> représente un titre, et un paragraphe.

3. CSS – Cascading Style Sheets

CSS est le langage qui permet de **mettre en forme** les pages créées avec HTML.
Avec CSS, on peut :

- changer la **couleur**, la **taille du texte**,
- organiser les éléments (les centrer, les espacer),
- rendre un site **agréable visuellement**.
-

Exemple :
body {
background-color: white;
}

4. PHP – Hypertext Preprocessor

PHP est un **langage de programmation côté serveur**, utilisé pour **interagir avec une base de données** ou **gérer les formulaires**.
C'est grâce à PHP que :

- les réservations sont envoyées dans la base de données,
- un message de confirmation s'affiche après l'envoi du formulaire.

Exemple :

```
if ($_SERVER["REQUEST_METHOD"] == "POST") {
// traitement du formulaire
}
```

5. XAMPP

XAMPP est un **logiciel gratuit** qui permet de simuler un serveur web sur un ordinateur personnel (appelé aussi "serveur local").
Il contient :

- **Apache** (serveur web),
- **MySQL** (base de données),
- **PHP** (langage utilisé pour le traitement des données).

Grâce à XAMPP, j'ai pu tester le site en local, comme s'il était déjà en ligne.

6. phpMyAdmin

PhpMyAdmin est une **interface graphique** (accessible depuis un navigateur) pour **gérer une base de données MySQL**.
Il permet :

- de créer des bases et des tables,
- de consulter ou modifier les données enregistrées,
- de vérifier si les formulaires fonctionnent correctement.

6. Google Fonts

J'ai utilisé **Google Fonts** pour importer une police d'écriture élégante :
Playfair Display, qui donne une touche luxueuse et professionnelle au site.

Présentation de la base de données

Qu'est-ce qu'une base de données ?

Une **base de données** est un outil informatique qui permet **de stocker et organiser des informations** de manière structurée, pour pouvoir les retrouver facilement.

Dans mon projet, la base de données est utilisée pour **enregistrer les réservations** des clients et **les messages envoyés via le formulaire de contact**.

Outil utilisé : MySQL via phpMyAdmin

J'ai utilisé **phpMyAdmin**, un outil graphique très simple d'utilisation, fourni par **XAMPP**, qui permet de :

- créer des bases et des tables,
- voir ce que les visiteurs ont envoyé via les formulaires,
- modifier ou supprimer des enregistrements si besoin.

La base de données s'appelle par exemple :
momohut
Et elle contient plusieurs **tables**.

La table reservations

Elle contient les informations que les clients remplissent quand ils réservent une table via le formulaire.
Voici sa structure (les **champs** ou **colonnes** qu'elle contient) :

Nom du champ	type	Description
Id	Int(auto incr.)	Identifiant unique de chaque réservation
Name	VARCHAR(100)	Le nom du client
Email	VARCHAR(100)	L'adresse e-mail du client
Phone	VARCHAR(20)	Le numéro de téléphone
Date	Date	Le jour de la réservation
Time	Time	L'heure de la réservation
Guest	Int	Le nombre de personnes
Message	Text	Message optionnel laissé par le client
Created_at	Timestamp	Date et heure automatique d'enregistrement

Cette table est créée grâce à une commande SQL (qu'on a insérée dans phpMyAdmin) comme ceci :

```
CREATE TABLE reservations (
id INT AUTO_INCREMENT PRIMARY KEY,
name VARCHAR(100) NOT NULL,
email VARCHAR(100) NOT NULL,
phone VARCHAR(20) NOT NULL,
date DATE NOT NULL,
time TIME NOT NULL,
guests INT NOT NULL,
message TEXT,
created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

La table messages (pour le formulaire de contact)

Cette table stocke les messages envoyés par les visiteurs via le formulaire “Contact”.

Nom du champ	Type	Description
id	INT	Identifiant du message
Name	VARCHAR (100)	Nom de la personne
Email	VARCHAR (100)	Adresse mail
Message	TEXT	Contenu du message
Created_at	TIMESTAMP	Date/heure de l’envoie

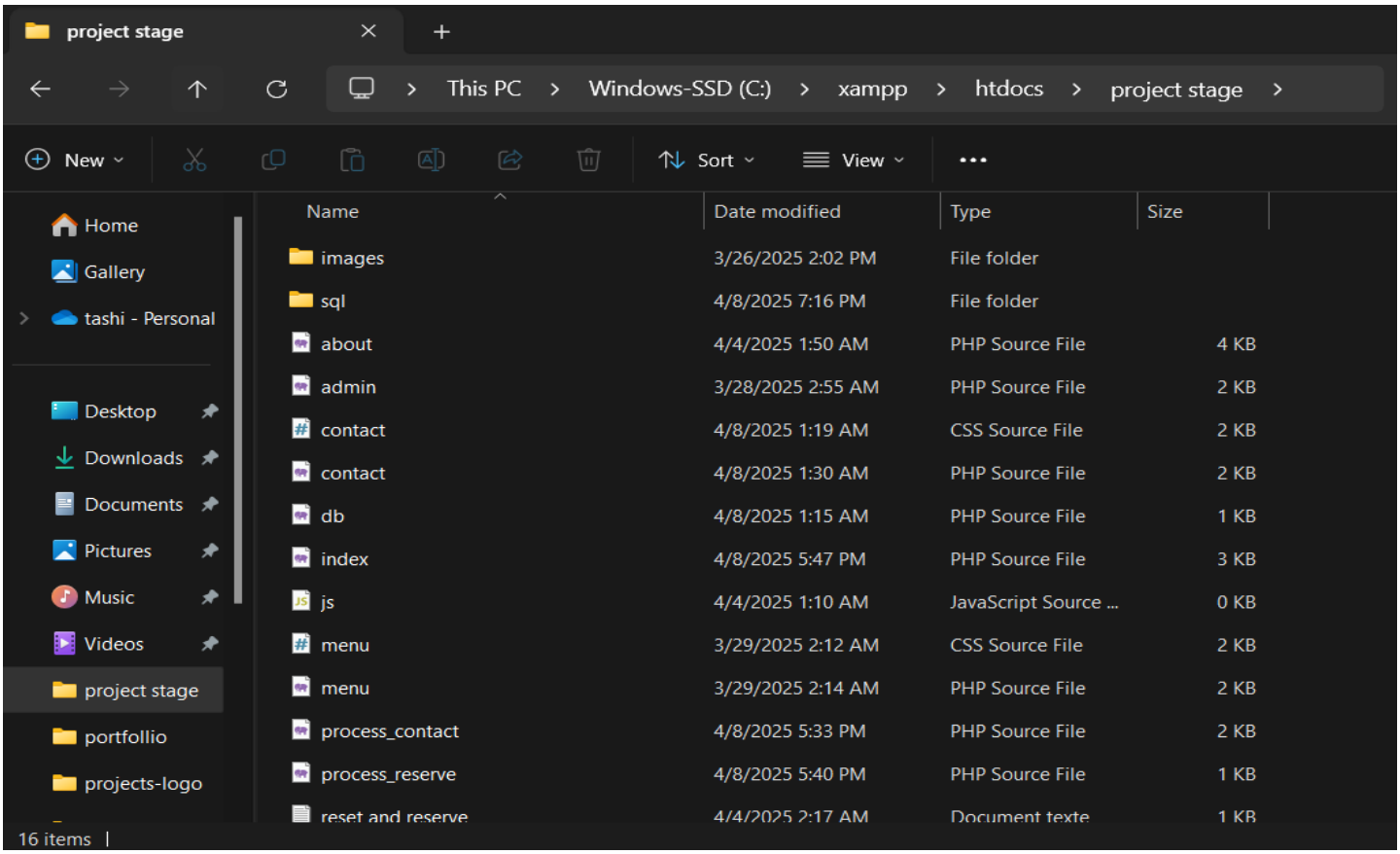
Résultat

Grâce à cette base de données, toutes les **réservations** et **messages** sont :

- **enregistrés automatiquement**,
- **consultables** depuis phpMyAdmin,
- **modifiables ou supprimables** si besoin.

Cela rend le site **fonctionnel** et **utile en situation réelle** pour un restaurant.

Organisation du projet (structure des fichiers)



Fichiers principaux (pages visibles)

Fichier	Rôle
Index.html	Page d'accueil du site (présente le restaurant et le lien vers le menu)
About.php	Présente l'histoire du restaurant, son ambiance, sa localisation
Contact.php	Contient le formulaire de contact (nom, e-mail, message)
Reserve.php	Formulaire de réservation d'une table (nom, date, heure, invités, etc.)
Menu.php	Affiche les photos de la carte du restaurant en galerie

Fichiers de traitement (backend)

Fichier	Rôle
Process_contact.php	Enregistre les messages du formulaire de contact dans la base de données
Process_reserve.php	Enregistre les réservations faites par les clients
Db.php	Fichier de connexion à la base de données MySQL via PHP

Ces fichiers ne sont pas visibles directement dans le navigateur, mais sont appelés lorsqu'un formulaire est envoyé.

Dossier Images

Le dossier images/ contient :

- les images du menu en format JPG (issues du PDF),
- l'image d'arrière-plan utilisée sur la page d'accueil (momos.jpg),
- les éventuelles icônes ou illustrations utilisées pour la décoration du site.

Cette organisation permet de garder un **projet clair, lisible et facile à maintenir**.
Chaque élément est séparé :

- le contenu (PHP),
- la présentation (CSS),
- les images,
- la logique (traitements des formulaires).

Fonctionnement global du site web

Le site que j'ai réalisé pour le restaurant MOMO HUT est un **site vitrine dynamique**.
Cela veut dire qu'il a des **pages statiques** (avec du texte, des images, etc.)
et des **pages dynamiques** (qui interagissent avec une base de données).
Voici le fonctionnement complet, étape par étape :

1. Navigation entre les pages

Le haut du site contient un **menu de navigation** visible sur toutes les pages (<nav> dans HTML).
Ce menu contient des liens vers :

- Accueil (index.php)
- À propos (about.php)
- Contact (contact.php)
- Réserver (resrerve.php)

Quand on clique sur un lien, on est redirigé vers la bonne page.
Ce menu est **fixe**, donc il reste visible même lorsqu'on fait défiler la page.

2. Accueil (index.php)

Cette page contient :

- une **grande image en arrière-plan** avec un **texte central** (nom du resto + phrase d'accroche),
- une **description simple du restaurant** (ambiance, style de cuisine),
- un **lien vers le menu** pour explorer les plats proposés,
- et un **pied de page (footer)** avec les liens légaux et les réseaux sociaux.

3. Page À propos (about.php)

Cette page explique :

- où se trouve le restaurant (Briançon),
- le type de cuisine servie (tibétaine et asiatique),
- l'ambiance souhaitée (moderne, chaleureuse),
- et invite les visiteurs à venir découvrir la carte.

Le contenu est **centré**, clair, et écrit dans un style simple et professionnel.

4. Page Menu (menu.php)

Cette page affiche **les images du menu** (importées depuis un fichier PDF).
Chaque image est affichée comme une **vignette**, avec un petit effet visuel quand on passe la souris dessus (zoom léger).
Cela permet aux visiteurs **de voir tous les plats, les prix, et les boissons** en un seul coup d'œil.

5. Page Contact (contact.php)

Cette page contient un **formulaire** avec 3 champs :

- Nom
- Email
- Message

Quand l'utilisateur remplit le formulaire et clique sur "Envoyer" :

- les données sont **envoyées vers un fichier PHP** (process_contact.php),
- puis elles sont **stockées dans la base de données** (table messages),
- et un **message de confirmation** apparaît : "Votre message a bien été envoyé."

Cette page utilise PHP pour **gérer les données** du formulaire.

6. Page Réserver (resrerve.php)

Il s'agit d'un **formulaire de réservation**.
Le client peut y remplir :

- Nom
- Email
- Téléphone
- Date et heure souhaitées
- Nombre de personnes
- Message optionnel

Une fois le formulaire envoyé :

- les données sont transmises au fichier process_reserve.php,
- elles sont insérées dans la table reservations de la base de données,
- un message s'affiche en haut de la page pour confirmer l'enregistrement.

7. Base de données

Toutes les données envoyées via les formulaires sont **stockées dans une base de données MySQL**, créée via **phpMyAdmin**, avec deux tables :

- reservations (pour les réservations)
- messages (pour les messages du formulaire contact)

Les formulaires utilisent PHP pour **insérer les données** dans les bonnes colonnes.

8. Responsive Design

Le site a été conçu pour être **responsive**, c'est-à-dire **adapté à tous les écrans** :

- ordinateur,
- tablette,
- téléphone.

Les images s'adaptent à la taille de l'écran, et le menu se réorganise automatiquement.

Conception de la base de données du site MOMO HUT

Qu'est-ce qu'une base de données ?

Une base de données (souvent abrégée **B.D.**) est un **ensemble structuré d'informations**.
Elle permet de **stocker, organiser, rechercher et modifier des données facilement**.
Pour MOMO HUT, j'ai utilisé **MySQL**, un système de gestion de base de données relationnelle (SGBDR), via **phpMyAdmin**, un outil visuel fourni avec XAMPP.

But de ma base de données

Le site MOMO HUT devait permettre :

- aux clients de **réserver une table** en ligne,
- de **nous contacter** avec un message.

Pour cela, j'ai dû créer **deux tables distinctes** :

1. reservations pour les réservations
2. messages pour les messages du formulaire de contact

Structure de la base momohut

La base s'appelle momohut, et elle contient 2 tables :

Table reservations

Nom du champ	Type sql	Role
Id	Int(auto incr.)	Identifiant unique de chaque réservation
Name	VARCHAR(100)	Le nom du client
Email	VARCHAR(100)	L'adresse e-mail du client
Phone	VARCHAR(20)	Le numéro de téléphone
Date	Date	Le jour de la réservation
Time	Time	L'heure de la réservation
Guest	Int	Le nombre de personnes
Message	Text	Message optionnel laissé par le client
Created_at	Timestamp	Date et heure automatique d'enregistrement

Clé primaire : id

Table messages

Nom du champ	Type	Description
id	INT	Identifiant du message
Name	VARCHAR (100)	Nom de la personne
Email	VARCHAR (100)	Adresse mail
Message	TEXT	Contenu du message
Created_at	TIMESTAMP	Date/heure de l'envoie

Pourquoi ces choix de types ?

- VARCHAR = texte court (nom, email)
- TEXT = texte long (messages)
- INT = nombre entier (id, nombre de personnes)
- DATE = pour enregistrer des jours
- TIME = pour enregistrer des heures
- TIMESTAMP = enregistre automatiquement le moment exact de la création

Comment j'ai créé la base

1. Lancer **XAMPP** → Démarrer **Apache** et **MySQL**
2. Aller sur http://localhost/phpmyadmin
3. Créer une nouvelle base momohut
4. Onglet "SQL" → coller le code suivant :

```
CREATE TABLE reservations (
id INT AUTO_INCREMENT PRIMARY KEY,
name VARCHAR(100) NOT NULL,
email VARCHAR(100) NOT NULL,
phone VARCHAR(20) NOT NULL,
date DATE NOT NULL,
time TIME NOT NULL,
guests INT NOT NULL,
message TEXT,
created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE messages (
id INT AUTO_INCREMENT PRIMARY KEY,
name VARCHAR(100) NOT NULL,
email VARCHAR(100) NOT NULL,
message TEXT NOT NULL,
created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

Sécurité et bonnes pratiques

- Tous les champs NOT NULL (Ce champ ne peut pas rester vide) sont obligatoires
- L'**id** est auto-incrémenté → évite les doublons
- Le **timestamp** est utile pour classer les données dans l'ordre chronologique
- Les données entrées par l'utilisateur sont **protégées** côté PHP avec mysqli_real_escape_string() pour éviter les injections SQL

Résultat final

Grâce à cette base de données :

- Chaque réservation est **stockée automatiquement**
- Je peux retrouver tous les messages reçus depuis le site
- Le site devient **dynamique et interactif**, pas juste une vitrine statique

Comprendre le fichier process_reserve.php (formulaire de réservation)

Ce fichier est utilisé pour **enregistrer les réservations** dans la base de données quand un client remplit le formulaire.

Objectif du fichier

Quand quelqu'un remplit le formulaire de réservation sur le site, ce fichier :

1. **Vérifie que le formulaire a été envoyé**
2. **Récupère les données du client**
3. **Les protège pour éviter les attaques**
4. **Les enregistre dans la base de données**
5. **Affiche un message de confirmation ou une erreur**

explication + code

Cette ligne importe le fichier db.php où on a écrit la connexion MySQL. Sans ça, on ne pourrait pas parler à la base de données

```
<?php
// Connexion à la base de données
include 'db.php';
```

Cette condition vérifie que le formulaire a bien été envoyé avec la méthode POST (et pas juste en rechargeant la page).
C'est une sécurité.

```
// Vérifie si le formulaire a été soumis
if ($_SERVER["REQUEST_METHOD"] == "POST") {
```

$_POST['name'] = **on récupère ce que l'utilisateur a écrit dans le champ name**
mysqli_real_escape_string(...) = **protège contre les injections SQL (très important pour la sécurité)**
Pour les champs date, time, guests, on n'a pas mis de protection ici car ce sont des valeurs simples, mais on pourrait.

```
// On récupère les données du formulaire
  $name = mysqli_real_escape_string($conn, $_POST['name']);
  $email = mysqli_real_escape_string($conn, $_POST['email']);
  $phone = mysqli_real_escape_string($conn, $_POST['phone']);
  $date = $_POST['date'];
  $time = $_POST['time'];
  $guests = $_POST['guests'];
  $message = mysqli_real_escape_string($conn, $_POST['message']);
```

C'est la requête SQL. Elle dit :
"Ajoute une nouvelle ligne dans la table reservations avec toutes les infos du formulaire."

```
  // On prépare la requête SQL pour insérer les données dans la table "reservations"
  $sql = "INSERT INTO reservations (name, email, phone, date, time, guests,
message)
        VALUES ('$name', '$email', '$phone', '$date', '$time', '$guests', '$message')";
```

mysqli_query(...) = **envoie la requête à la base**
Si tout marche bien, on redirige vers la page resrerve.php avec un message de succès
exit(); = **stoppe le script après la redirection (très important)**

```
  // Exécute la requête SQL
  if (mysqli_query($conn, $sql)) {
    header("Location: resrerve.php?success=1");
    exit();
  }
```

Résumé

Le fichier process_reserve.php est le cœur du système de réservation.
Il récupère les données, les protège, puis les insère dans la base.
Ensuite, il redirige vers la page pour afficher un message de succès ou d'erreur.

Fonctionnement du fichier process_contact.php

Objectif

Ce fichier a pour but de **récupérer les messages envoyés par les visiteurs** via le formulaire de contact, puis de les **enregistrer dans la base de données** dans la table messages.

Où ce fichier est-il utilisé ?

Il est appelé depuis le fichier contact.php, dans la balise <form> :
<form action="process_contact.php" method="POST">
Cela signifie que **lorsque l'utilisateur clique sur "Envoyer"**, toutes les données tapées sont **envoyées à process_contact.php**.

Explication du code

Voici le code typique d'un fichier process_contact.php :

```
<?php
include 'db.php'; // Connexion à la base de données
```

Cette ligne **importe le fichier db.php**, qui contient les infos de connexion à la base de données. Sans ça, aucune opération SQL ne peut être faite.

```
if ($_SERVER["REQUEST_METHOD"] == "POST") {
```

ette condition vérifie si le **formulaire a bien été envoyé** (par la méthode POST). C'est une **sécurité** pour ne pas exécuter le script quand ce n'est pas nécessaire.

```
  // Récupération des données du formulaire
  $name = mysqli_real_escape_string($conn, $_POST['name']);
  $email = mysqli_real_escape_string($conn, $_POST['email']);
  $message = mysqli_real_escape_string($conn, $_POST['message']);
```

La fonction *mysqli_real_escape_string(...)* est utilisée pour **protéger la base de données** contre les attaques (par injection SQL).

```
// Requête SQL d'insertion
$sql = "INSERT INTO messages (name, email, message)
        VALUES ('$name', '$email', '$message')";
```

Ici, on **prépare la requête SQL** qui va insérer les données dans la table messages. Les noms des champs dans la requête doivent **correspondre exactement** aux champs créés dans la base.

```
if (mysqli_query($conn, $sql)) {
  header("Location: contact.php?success=1");
  exit();
}
```

"Si la requête SQL a réussi..."

- mysqli_query(...) sert à **exécuter la requête SQL** qu'on a écrite juste avant ($sql)
- Si tout se passe bien (pas d'erreur), alors cette fonction **retourne true**
- Donc if (...) = si la requête fonctionne → on entre dans le bloc { ... }

Si elle échoue (par exemple une faute dans ta requête ou un champ manquant), le bloc ne s'exécute pas et le else s'active à la place.

header("Location: contact.php?success=1");

Renvoie l'utilisateur vers la page contact.php en ajoutant ?success=1 dans l'adresse.

exit(); Pourquoi c'est important ?

Imagine que tu oublies le exit(); :

- Le code **continue à s'exécuter après la redirection**
- Cela peut **reprovoquer des erreurs** ou des comportements bizarres (comme renvoyer le formulaire encore une fois)

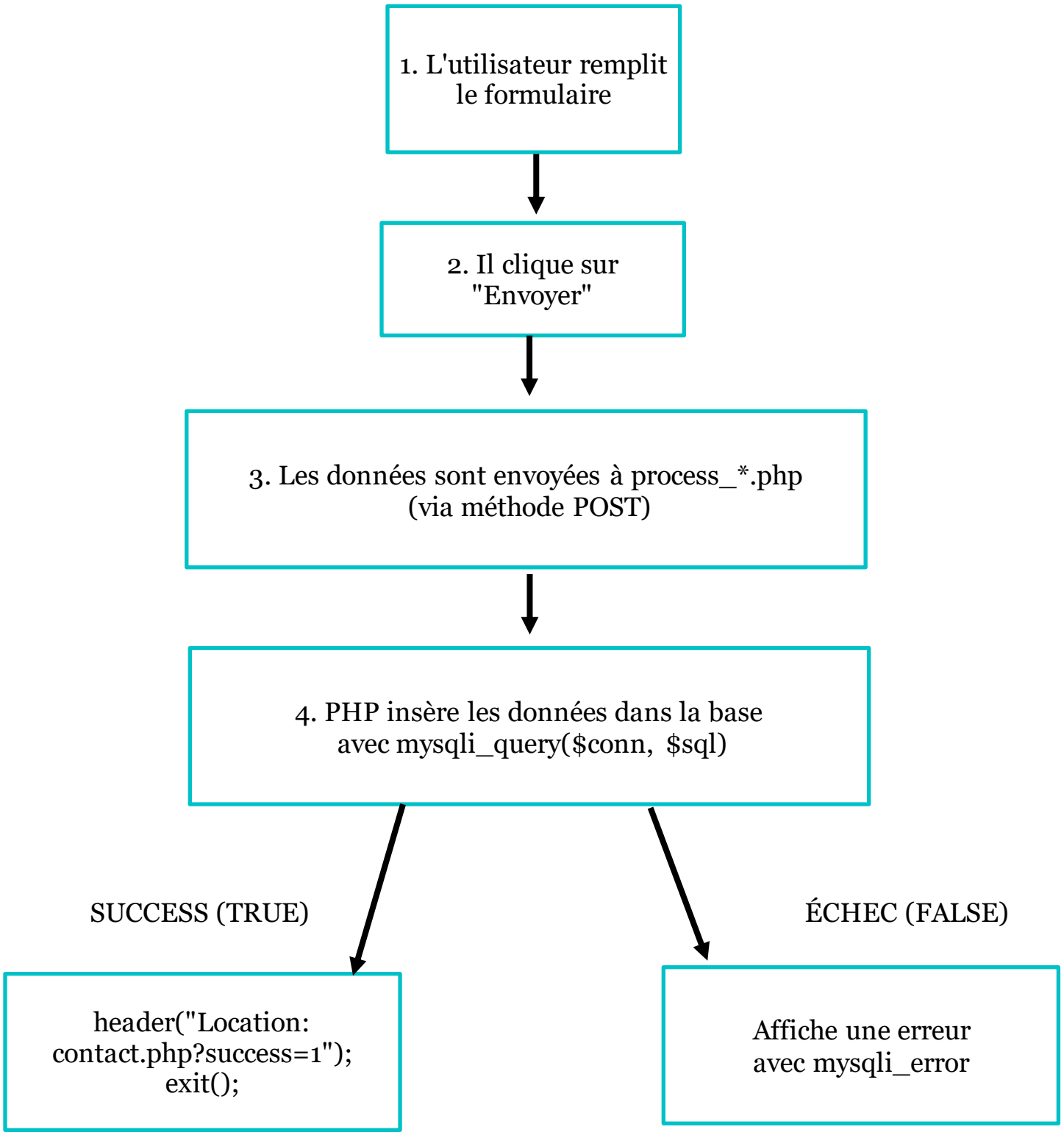
En gros, on dit :

"Le travail est fini, stop ici, pas besoin d'en faire plus."

```
else {
   echo "Erreur : " . $sql . "<br>" . mysqli_error($conn);
  }
}
?>
```

Si jamais la requête échoue, on affiche une **erreur technique** avec le message fourni par MySQL (mysqli_error), ce qui est très utile pour corriger un bug.

Schéma du processus



Message de succès avec ?success=1 dans contact.php et reserve.php

Objectif

Quand une personne remplit le formulaire, on :

1. **insère les données dans la base**
2. on **redirige vers la page contact.php?success=1**
3. et ensuite, si success=1 est dans l'URL, on **affiche un message vert** avec "Votre message a bien été envoyé"

Code PHP pour afficher un message conditionnel

Voici le petit bout de code PHP qu'on place **juste au-dessus du formulaire** :

```
<?php if (isset($_GET['success'])): ?>

  <p class="success">Votre message a bien été envoyé !</p>

<?php endif; ?>
```

`<?php if (isset($_GET['success'])): ?>`
Cela veut dire :
"Si dans l'URL, il y a quelque chose qui s'appelle **success**, alors ..."
Exemple d'URL :
http://localhost/project/contact.php?success=1
Ici $_GET['success'] existe → donc on peut afficher un message.

`<p class="success">Votre message a bien été envoyé !</p>`
Ce texte s'affiche **uniquement** si la condition est vraie (donc si success=1 est présent dans l'URL)

`<?php endif; ?>`
Fin de la condition (comme } en JavaScript ou C++).
C'est la **syntaxe spéciale de PHP** quand on utilise if dans du HTML.

Adaptation mobile et Responsive Design

Qu'est-ce que le responsive design ?

Le **responsive design** (ou design adaptatif) est une technique qui permet à un site web de s'adapter automatiquement à **toutes les tailles d'écran**, qu'il soit ouvert sur :

- Un **ordinateur**
- Une **tablette**
- Un **telephone**

Cela permet d'éviter les textes trop petits, les images trop larges, les barres de défilement horizontales, ou les boutons inaccessibles.
C'est aujourd'hui **obligatoire** : plus de 70% des internautes visitent les sites via leur smartphone.

Mise en place dans mon projet MOMO HUT

Pour rendre mon site responsive, j'ai utilisé **des techniques simples en CSS**, sans framework comme Bootstrap.
L'idée était de **comprendre et apprendre** à le faire moi-même.
Voici les **étapes et explications techniques** :

1. Ajout de la balise <meta viewport>

Dans le fichier HTML (index.php, about.php, etc.), j'ai ajouté cette ligne dans la balise <head> :
<meta name="viewport" content="width=device-width, initial-scale=1.0">

Explication :

- *width=device-width* : le site utilise la largeur de l'écran réel (pas un zoom automatique).
- *initial-scale=1.0* : le zoom de base est de 100%.

Cela dit au téléphone : "n'essaie pas de zoomer, affiche le site tel quel".

2. Utilisation de max-width + margin: auto

```
.container {
  max-width: 1000px;
  margin: auto;
  padding: 20px;
}
```

Explication :

- max-width: 1000px : le contenu ne s'étire pas trop sur grand écran
- margin: auto : le contenu est **centré automatiquement**
- padding : on ajoute un peu d'espace autour du contenu

Cela permet d'avoir un site qui **s'adapte aux petits écrans sans se casser**.

3.Largeur fluide des images et sections

```
img {
    width: 100%;
    height: auto;
}
```

Cela permet aux images de **se redimensionner automatiquement** en fonction de l'écran.
Elles ne débordent plus sur mobile.

4. Grilles adaptatives avec grid et auto-fit

```
.grid-container {
  display: grid;
  grid-template-columns: repeat(auto-fit, minmax(250px, 1fr));
  gap: 30px;
}
```

Explication :

Grid est un système qui te permet de créer facilement des mises en page en colonnes et en lignes.
Il sert à organiser des éléments (comme des images, des textes, etc.) dans un tableau invisible.
auto-fit : le site **calcule automatiquement** combien de colonnes il peut afficher

- minmax(250px, 1fr) : chaque élément fait minimum 250px, et prend le reste de l'espace si possible
- gap : espace entre les éléments

Résultat : sur ordinateur il y a 3 images côte à côte, mais sur téléphone, elles se mettent **l'une en dessous de l'autre automatiquement**.

Limites du projet et pistes d'amélioration

Limites du projet

Même si le site est fonctionnel et bien structuré, il présente certaines limites techniques dues :

- à mon **niveau débutant en programmation**,
- au **temps limité** pour le stage,
- et au fait que le site est pour l'instant **hébergé uniquement en local (XAMPP)**.

Quelques limites identifiées :

- ❌ Le site n'est pas encore **hébergé en ligne**, donc inaccessible en dehors de mon ordinateur.
- ❌ Il n'y a **pas de système d'administration sécurisé** pour gérer les réservations.
- ❌ La partie "menu" utilise des **images de PDF** au lieu d'un affichage dynamique des plats.
- ❌ Il n'y a pas de **système d'authentification** (connexion pour le personnel, par exemple).
- ❌ Pas de **vérification avancée** côté serveur (ex : pas de vérification du format de téléphone, ou des créneaux horaires disponibles).
- ❌ Pas encore de **gestion des horaires de fermeture automatique** dans le formulaire (ex : éviter de réserver le lundi).

Pistes d'amélioration

Voici ce que je pourrais améliorer si j'avais plus de temps ou si je continue ce projet :

Technique

- Créer une **interface d'administration** avec un **login sécurisé** pour voir, modifier ou supprimer les réservations.
- Utiliser un **hébergeur web (OVH, Hostinger…)** pour que le site soit visible en ligne sur téléphone ou ordinateur.
- Ajouter des **animations modernes** (sliders d'images, transitions douces).
- Faire un **vrai menu dynamique** (avec des données en base + filtres par plat, prix, allergènes…).

Fonctionnel

- Ajouter une **section "Avis des clients"** (avec étoiles et commentaires).
- Proposer un système de **réservation avec créneaux précis** (éviter les doubles réservations).
- Ajouter un bouton de **traduction (FR/EN)**.
- Mettre une **Google Map interactive** pour trouver facilement le restaurant.

Conclusion de la page

Ce projet m'a permis de comprendre les bases du développement web (HTML, CSS, PHP, SQL), mais aussi de voir qu'un bon site prend du temps, de la réflexion, et doit s'adapter aux besoins réels du client.
J'ai maintenant une meilleure vision de ce qu'il est possible d'améliorer dans un projet web réel.

Bilan personnel du stage – Rédigé en phrases

Pendant mon stage, j'ai eu l'opportunité de créer un site web complet pour le restaurant MOMO HUT, situé à Briançon. C'était la première fois que je réalisais un projet de cette ampleur, et j'ai tout appris depuis zéro.

Avant ce stage, j'avais déjà été initié aux langages HTML, CSS et PHP, mais je rencontrais beaucoup de difficultés à les utiliser concrètement dans un vrai projet. Ce stage m'a permis de mieux comprendre leur fonctionnement, leur utilité, et surtout comment les faire interagir ensemble dans un site dynamique.

J'ai appris à structurer correctement une page web avec **HTML** , à améliorer le design avec **CSS** , et à rendre le site interactif avec **PHP**, un langage de programmation côté serveur.

J'ai également renforcé mes compétences en **base de données**, notamment grâce à **phpMyAdmin**, un outil visuel qui m'a permis de créer des tables, de stocker des données (comme les messages ou les réservations), et de les lire ensuite depuis le site. Ce stage m'a surtout aidé à **appliquer mes connaissances**, à mieux comprendre les erreurs que je rencontrais, et à progresser dans la **logique du développement web**.Ce stage m'a appris à être plus autonome, à chercher des solutions quand j'étais bloqué, à corriger mes erreurs, et à progresser malgré les difficultés. Certaines parties du site étaient plus compliquées que prévu, notamment la mise en page sur mobile, les problèmes de compatibilité navigateur, ou encore l'enregistrement des données dans la base.

Cependant, j'ai réussi à surmonter ces obstacles, souvent seul, en me concentrant et en testant différentes approches.
Je suis fier du résultat final. Le site fonctionne, il est esthétique, responsive, et les fonctionnalités de contact et de réservation sont opérationnelles. Je ne pensais pas être capable de réaliser cela au début du stage, mais maintenant je sais que c'est possible avec de la persévérance.
D'après ce je suis maintenant motivé à continuer à apprendre, à améliorer ce site.