

# Projet

## Gestion de l'hôtel California



Rédigé par

**David ROUMANET**  
Professeur BTS SIO



Changement

| Date | Révision |
|------|----------|
|      |          |
|      |          |
|      |          |
|      |          |
|      |          |
|      |          |
|      |          |

## Sommaire

|       |  |    |
|-------|--|----|
| A     | Introduction.....  | 1  |
| A.1   | Présentation.....  | 1  |
| A.2   | Compétences.....   | 1  |
| B     | Cahier des charges.....                                  | 2  |
| B.1   | Fonctionnalités Principales.....                         | 2  |
| C     | Modélisation de la base de données.....                  | 3  |
| C.1   | Rédiger le modèle UML du diagramme de classe.....        | 3  |
| C.2   | Recommandation de votre tuteur.....                      | 3  |
| C.3   | Rédiger un diagramme d'utilisation.....                  | 3  |
| D     | Création de la base de données.....                      | 4  |
| D.1   | Table client.....  | 4  |
| D.2   | Table Chambre.....                                       | 4  |
| D.3   | Table Réservation.....                                   | 5  |
| D.4   | Remplissage des tables.....                              | 5  |
| E     | Création du projet.....                                  | 6  |
| E.1   | Création d'un dépôt git.....                             | 6  |
| E.2   | Création de la structure du projet.....                  | 6  |
| E.3   | Création de code pour la gestion des chambres.....       | 7  |
| E.3.1 | Code de connexion à la base de données.....              | 7  |
| E.3.2 | Programmation de la page de démarrage.....               | 8  |
| E.3.3 | Programmation de l'affichage des chambres.....           | 9  |
| E.3.4 | Programmation de la création d'une chambre.....          | 11 |
| E.3.5 | Programmation de la page d'édition d'une chambre.....    | 13 |
| E.4   | Programmation du CRUD pour les clients.....              | 16 |
| E.5   | Programmation du CRUD des réservations.....              | 17 |
| E.5.1 | Explication de la requête.....                           | 17 |
| E.5.2 | Programmation de l'affichage des réservations.....       | 18 |
| F     | Mise en forme CSS : frameworks.....                      | 20 |
| F.1   | Exemple de mise en forme.....                            | 20 |
| F.2   | Frameworks.....  | 20 |
| F.2.1 | Bootstrap.....   | 21 |
| F.2.2 | Font-Awesome.....  | 22 |
| F.3   | Création d'un menu type navbar.....                      | 23 |
| F.4   | Mettre en place les frameworks dans chaque page.....     | 24 |
| G     | Gestion des erreurs.....                                 | 28 |
| G.1   | Gestion affichage.....                                   | 28 |
| G.2   | Gestion de transmission de message.....                  | 30 |
| G.3   | Adaptation à toutes les pages.....                       | 31 |
| G.4   | [Facultatif] : branche GIT et utilisation de cookie..... | 31 |
| H     | Gestion des accès.....                                   | 33 |
| I     | Annexes.....   | 35 |
| I.1   | Définition de la base de données.....                    | 35 |
| I.1.1 | Table chambres :.....                                    | 35 |
| I.1.2 | Table clients :.....                                     | 35 |
| I.1.3 | Table reservations :.....                                | 35 |

---

|                 |    |
|-----------------|----|
| I.2 Autres..... | 35 |
|-----------------|----|

Nomenclature :

- **Assimiler** : cours pur. Explication théorique et détaillée (globalement supérieur à 4 pages).
- **Décoder** : fiche de cours, généralement inférieure à 5 pages.
- **Découvrir** : Travaux dirigés. Faisable sans matériel.
- **Explorer** : Travaux pratiques. Nécessite du matériel ou des logiciels.
- **Mission** : Projet encadré ou partie d'un projet.
- **Voyager** : Projet en autonomie totale. Environnement ouvert : Vous êtes le capitaine !

## A Introduction

---

Vous travaillez comme stagiaire pour l'entreprise Eagle, une ESN<sup>1</sup> qui propose des applications pour ses clients.

### A.1 Présentation

Eagle a une demande pour la création d'une application de gestion de réservation de chambres pour l'hôtel California. Elle pense que vous pourriez participer au développement de ce projet en PHP et MariaDB.

### A.2 Compétences

Durant votre travail sur ce projet, vous allez :

- Concevoir un code structuré
- Concevoir une base de données avec des contraintes
- Générer une application simple
- Utiliser le concept de CRUD

---

1 Entreprise de Services Numériques

## B Cahier des charges

---

### B.1 Fonctionnalités Principales

L'**hôtel California** souhaite un logiciel permettant de gérer les chambres en prenant en compte leur capacité (nombre de personnes), les clients et les réservations.

Gestion des Chambres :

- Ajouter, modifier, supprimer des chambres.
- Afficher la liste des chambres avec leur capacité et disponibilité.

Gestion des Clients :

- Enregistrer les informations des clients : nom, téléphone, email, nombre de personnes.
- Associer un client à une réservation.

Réservation :

- Réserver une chambre pour une période donnée.
- Vérifier la disponibilité des chambres pour une période donnée.
- Afficher la liste des réservations.

Interface Utilisateur :

- Formulaire de réservation.
- Interface d'administration pour gérer les clients, les chambres et les réservations.

Technologies Utilisées

- Langage : PHP
- Base de données : MariaDB
- Front-end : HTML, CSS (via framework). optionnel : JavaScript pour améliorer l'interactivité

## C Modélisation de la base de données

---

### C.1 Rédiger le modèle UML du diagramme de classe

En utilisant les informations vues en cours, votre tâche est de modéliser la base de données en UML.

- Créer les classes nécessaires
- Placer les attributs avec leur type
- Ajouter les relations entre les classes
- Annoter avec les multiplicités

### C.2 Recommandation de votre tuteur

Damien vous explique :

*Pour créer des diagrammes UML, le plus simple, c'est d'utiliser une extension à ton IDE : pour ma part, je préfère utiliser UMLet, mais quelques collègues utilisent DrawIO... ça n'a pas d'importance. Pour UMLet, une fois installé, il suffit de créer un fichier avec l'extension .uxf dans un répertoire du projet (par exemple docs) et ensuite, de double-cliquer dessus.*

*C'est similaire pour drawIO.*

*Tu dois commencer par les classes, en remplissant les attributs : pour cela, on utilise le verbe avoir pour décider de quoi mettre. Client a un nom, client a un téléphone... on ne peut pas dire client a lire, écrire, etc.*

*Ensuite, il faut trouver le lien entre les classes : un simple trait, surtout pas de flèches !*

*Enfin, le calcul des multiplicités : classe Client : un client peut avoir X réservations, puis l'inverse, une réservation peut avoir Y client.*

*Note bien qu'il peut y avoir d'autres solutions et il sera nécessaire d'en discuter quand tu auras fini.*

### C.3 Rédiger un diagramme d'utilisation

En utilisant les informations vues en cours, créez le diagramme des cas d'utilisations pour l'hôtel California.

## D Création de la base de données

Après avoir modélisé la base, il faut mettre en place les trois tables, Client, Chambre et Reservation.

Dans les moteurs de base de données, il faut utiliser `utf8mb4` et `utf8_general_ci` pour l'encodage des caractères. Pour l'analyse, ne pas choisir `myISAM`, mais bien `InnoDB`.

Si vous utilisez le système de requête de Dbeaver ou PHPMyAdmin, il suffit de copier les requêtes ci-après.

Si vous déclarez vous-même tables et les colonnes, assurez-vous de respecter l'encodage et le moteur de base de données ci-dessus.



**Rappel : `myISAM` est plus performant mais ne prend pas en compte les contraintes d'intégrité. On peut alors effacer des enregistrements sans vérification ni contrôle qu'ils ont utilisés dans d'autres tables. Si rien n'est spécifié, il est plus prudent de prendre par défaut `InnoDB`.**

L'ordre de création des tables est important : la table `reservations` doit être déclarée en dernier. C'est logique, car elle utilise les clés primaires des tables `clients` et `chambres`.

### D.1 Table client

Damien vous propose de vous donner un exemple et vous ferez la table chambre.

Ici, c'est du SQL tout simple

```
-- Création de la table des clients
CREATE TABLE clients (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nom VARCHAR(100) NOT NULL,
  email VARCHAR(100) NOT NULL,
  telephone VARCHAR(20) NOT NULL,
  nombre_personnes INT NOT NULL
) ENGINE=InnoDB;
```

Tu auras vite compris que `CREATE TABLE` permet de créer une table nommée `clients`, puis, entre parenthèses, on transmet les attributs (colonnes) avec leur type.

Évidemment, on a créé un `id`, parce qu'il peut y avoir des homonymes (plusieurs clients peuvent avoir le même nom). Cet `id` s'auto-incrémente et devient donc une clé primaire, mais sais-tu ce qui se passera si on efface un enregistrement dans la table ? Le numéro continuera à s'incrémenter : on ne revient jamais en arrière...

### D.2 Table Chambre

Réalisez vous-même la table chambre.

## D.3 Table Réservation

Damien : Voici un peu d'aide, parce qu'ici, on va utiliser des clés étrangères. Ce sont les clés primaires d'autres tables, qui sont inscrites dans la table reservation.

```
-- Création de la table des réservations
CREATE TABLE reservations (
  id INT AUTO_INCREMENT PRIMARY KEY,
  client_id INT NOT NULL,
  chambre_id INT NOT NULL,
  date_arrivee DATE NOT NULL,
  date_depart DATE NOT NULL,
  FOREIGN KEY (client_id) REFERENCES clients(id) ON DELETE CASCADE,
  FOREIGN KEY (chambre_id) REFERENCES chambres(id) ON DELETE CASCADE
) ENGINE=InnoDB;
```

Identifiant local de la colonne représentant la clé étrangère

Nom de la table

Voir la colonne dans la table clients

On déclare les colonnes qui contiennent les clés étrangères, puis on spécifie le lien entre notre table et la table distante en précisant la table distante et entre parenthèses, la colonne contenant la clé primaire.

Au passage, la clé primaire de notre table **reservation** est constituée ici de deux champs : `client_id`, `chambre_id`.

Petite astuce : je te recommande ici de créer un index, pour accélérer la recherche... je ne vais pas t'expliquer le fonctionnement d'un index dans une base de données, mais regarde la vidéo de Khadija Letrache: <https://www.youtube.com/watch?v=7FGH4fHrfd8> (5 minutes 30)

```
CREATE INDEX idx_reservation_dates ON reservations(chambre_id, date_arrivee, date_depart);
```

## D.4 Remplissage des tables

L'ordre de remplissage des tables devient important et voici ce que votre tuteur explique :

On va demander à une IA de générer un jeu de données, car c'est une opération un peu rébarbative et l'automatisation par une IA est suffisamment simple pour éviter des biais ou hallucinations.

Pour cela, on demande un jeu d'essai pour chaque table, en commençant par client et chambre et en finissant par réservation (qui ne peut être remplie qu'après avoir les clients et les chambres, puisqu'elle fait appel à eux dans la clé étrangère).

Comment lui demander ? Tu colles la requête de création de table et tu demandes à avoir une douzaine d'enregistrements.

Je te recommande quand même de jeter un œil pour vérifier que les index sont bons. Par exemple, il n'y a pas de réservations avec un client qui n'existe pas ou une chambre qui n'existe pas.



## E Création du projet

Afin de vous aider dans le développement de cette application, votre tuteur, Damien, vous donne des pistes.

### E.1 Création d'un dépôt git

Damien :

*Si tu veux coder en équipe avec ton binôme, tu dois commencer par créer un dépôt GIT. Dans Visual Studio Code, si tu as un compte Github, c'est très facile :*

*Fichier → Ouvrir dossier...*

*Tu crées ton dossier dans le workspace d'Apache (www), par exemple `resaHotelCalifornia` et tu rentres dedans.*

*Tu cliques ensuite sur l'icône de forge GIT dans le bandeau à gauche et tu cliques sur le bouton [Publier sur Github] et choisis de le faire en **projet public**, ça te servira pour tes prochaines recherches de stages.*

### E.2 Création de la structure du projet

Là-encore, Damien a quelques astuces pour vous...

*Un véritable projet ne peut pas se faire avec un seul fichier de code et la plupart de mes collègues préfèrent de loin, avoir plusieurs fichiers avec quelques centaines de lignes de code maximum. Le plus simple est de créer un répertoire par table ou affichage. La gestion des chambres dans un répertoire chambre, un répertoire également pour les clients et aussi pour les réservations.*

*Un répertoire particulier pour les éléments de configuration et un dernier répertoire pour les assets. Les assets sont des ressources pour le fonctionnement de l'appli : les images, les fichiers CSS ou les frameworks.*

```
/resaHotelCalifornia/  
  /config/  
    db_connect.php  
  /assets/  
  /chambres/  
    listChambres.php  
    createChambre.php  
    editChambre.php  
    deleteChambre.php  
  /clients/  
    listClients.php  
    createClient.php  
    editClient.php  
    deleteClient.php  
  /reservations/  
    listReservations.php  
    createReservation.php  
    editReservation.php  
    deleteReservation.php  
  index.php
```

## E.3 Création de code pour la gestion des chambres

Damien vous propose de travailler la partie chambre avec vous et vous devrez faire la partie client.

*On va d'abord faire un ensemble de fonction pour gérer les connexions à la base de données. Il y a deux méthodes possibles :*

*Ouvrir et fermer la connexion à chaque accès*

*Garder la connexion ouverte pendant toute la durée de l'exécution du programme, lorsqu'il y a beaucoup d'accès.*

*Ici, il s'agit d'un hôtel : l'enregistrement d'une réservation prend forcément du temps, on peut donc utiliser la première méthode.*

### E.3.1 Code de connexion à la base de données

Voici le code de connexion :

```
config/db_connect.php
<?php
// Fonction de connexion à la base de données qui retourne le handle
function openDatabaseConnection() {
    $host = 'localhost';
    $db = 'resaHotelCalifornia';
    $user = 'root';
    $pass = '';

    try {
        // Utilisation de PDO plutôt que MySQLi
        $conn = new PDO("mysql:host=$host;dbname=$db", $user, $pass);
        $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
        return $conn;
    } catch(PDOException $e) {
        echo "Connection failed: " . $e->getMessage();
        exit;
    }
}

function closeDatabaseConnection($conn) {
    $conn = null; // Destructeur se charge de clore la connexion
}
?>
```

*On utilise une connexion **PDO**, qui est plus sécurisée que **mysql** et **mysqli**. Pour le moment, on utilise le compte root de l'administration de la base de données, mais lorsque l'application sera à peu près opérationnelle, il faudra créer un compte dédié.*

### E.3.2 Programmation de la page de démarrage

Pour accéder à votre site par défaut, il faut une page `index.php`.

Apache recherche toujours un fichier `index.php` ou `index.html` ou `index.htm` avant d'abandonner.

Voici la page de démarrage, très simplifiée :

```
index.php
<!DOCTYPE html>
<html lang="fr">
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Système de Gestion d'Hôtel</title>
</head>
<body>
  <h1>Système de Gestion d'Hôtel</h1>
  <ul>
    <li><a href="./chambres/listChambres.php">Gestion des Chambres</a></li>
    <li><a href="./clients/listClients.php">Gestion des Clients</a></li>
    <li><a href="./reservations/listReservations.php">Gestion des Réservations</a></li>
  </ul>
</body>
</html>
```

*Ce sont juste des liens, pour tester les prochaines pages. On utilise ici un lien relatif :*



Les **liens relatifs** ne commencent **JAMAIS** par un `/` et indiquent le chemin de manière relative au fichier en cours. `index.php` étant à la racine du répertoire `resaHotelCalifornia`, il pointe vers les sous-répertoires. `./` signifie "ici" tandis que `../` signifie "remonte d'un répertoire".



Les **liens absolus** commencent **TOUJOURS** par un `/` et indiquent le chemin depuis le répertoire racine d'Apache, dans notre cas, `www`. Ainsi, pour atteindre `listClients.php` il faut indiquer `resaHotelCalifornia/clients/listClients.php`. Ce chemin fonctionne, quelque soit l'enplacement du fichier en cours.

### E.3.3 Programmation de l'affichage des chambres

On verra plus tard, pour la partie réservation, qui est un peu plus complexe, car elle s'appuie sur les clients et sur les chambres. Là, on va afficher uniquement les chambres :

```

chambres/listChambres.php
<?php
    require_once '../config/db_connect.php';

    $conn = openDatabaseConnection();
    $stmt = $conn->query("SELECT * FROM chambres ORDER BY numero");
    $chambres = $stmt->fetchAll(PDO::FETCH_ASSOC);
    closeDatabaseConnection($conn);
?>

<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Liste des Chambres</title>
</head>
<body>
    <h1>Liste des Chambres</h1>
    <a href="create.php">Ajouter une chambre</a>
    <table border="1" style="width: 60%; min-width: 400px; margin: 0 auto;">
        <tr>
            <th>ID</th>
            <th>Numéro</th>
            <th>Capacité</th>
            <th>Actions</th>
        </tr>
        <?php foreach($chambres as $chambre): ?>
        <tr>
            <td><?php echo $chambre['id']; ?></td>
            <td><?= $chambre['numero'] ?></td>
            <td><?= $chambre['capacite'] ?></td>
            <td>
                <a href="editChambre.php?id=<?= $chambre['id'] ?>">Modifier</a>
                <a href="deleteChambre.php?id=<?= $chambre['id'] ?>" onclick="return confirm('Êtes-vous sûr?')">Supprimer</a>
            </td>
        </tr>
        <?php endforeach; ?>
    </table>
</body>
</html>

```

Boucle automatique : dans toutes les chambres, récupère une ligne chambre

Id correspond à la colonne id de la table chambre.

Lien de suppression de la chambre, grâce à l'id de celle-ci.

Je vais t'expliquer ce code :

Au début du fichier, on va chercher un autre fichier PHP : cela allège l'affichage de notre fichier et permet de gérer du code commun à toutes les pages. Ici, **db\_connect.php** sera le fichier de gestion des connexions à la base de données.

On ouvre la base et on exécute la requête SQL, pour stocker le résultat dans une variable \$stmt, parce qu'en anglais, c'est l'abréviation de 'statement'.

Au passage, j'ai ajouté un peu de CSS (pour que le tableau n'occupe que 60 % de la page, mais reste à 400 pixels au minimum).

Et tu noteras deux syntaxes pour afficher une variable PHP : `<?php echo $v ?>` ou `<?= $v ?>`, c'est pareil.

Maintenant, observe bien cette ligne :

```
<a href="editChambre.php?id=<?= $chambre['id'] ?>">Modifier</a>
```

On crée un lien vers une page **editChambre.php** et on transmet par simple **méthode GET**, un ID... et tu remarques que le numéro d'ID est dynamique : c'est PHP qui reprend le numéro d'ID de l'enregistrement.

On fait la même chose pour la suppression :

```
<a href="deleteChambre.php?id=<?= $chambre['id'] ?>" onclick="return confirm('sûr?')">Supprimer</a>
```

Ici, on ajoute seulement une confirmation en JavaScript, grâce à la fonction `confirm()` qui affichera une boîte de dialogue. Le lien vers **deleteChambre.php** ne sera suivi que si la fonction `confirm()` renvoie `true`.

Théoriquement, avec votre jeu d'essai et le code ci-dessus, vous devriez déjà pouvoir afficher le tableau des clients dans la page web. Assurez-vous que votre code est bien dans le répertoire web d'Apache !

Tu devrais regarder l'URL qui s'affiche quand tu survoles les liens : pour les chambres, l'URL est presque toujours identique, seul le numéro change, grâce à PHP qui l'a créé automatiquement dans la boucle `foreach`.

### E.3.4 Programmation de la création d'une chambre

La création d'une chambre consiste à ajouter un enregistrement dans la table chambre.

*Si c'est clair pour toi, on va passer à la suite : la création d'une chambre. C'est relativement facile et court, car il s'agit d'insérer un nouvel enregistrement.*

*Au passage, n'oublie pas de faire une version git de ce code, en indiquant, par exemple, que c'est la partie CRUD chambre terminée.*

chambres/createChambre.php

```
<?php
require_once '../config/db_connect.php';

if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    $numero = $_POST['numero'];
    $capacite = $_POST['capacite'];

    $conn = openDatabaseConnection();
    $stmt = $conn->prepare("INSERT INTO chambres (numero, capacite) VALUES (?, ?)");
    $stmt->execute([$numero, $capacite]);
    closeDatabaseConnection($conn);

    header("Location: listChambres.php");
    exit;
}
?>

<!DOCTYPE html>
<html lang="fr">
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Ajouter une Chambre</title>
</head>
<body>
    <h1>Ajouter une Chambre</h1>
    <form method="post">
        <div>
            <label>Numéro:</label>
            <input type="text" name="numero" required>
        </div>
        <div>
            <label>Capacité:</label>
            <input type="number" name="capacite" required>
        </div>
        <button type="submit">Enregistrer</button>
    </form>
    <a href="listChambres.php">Retour à la liste</a>
</body>
</html>
```

*Ici, nous avons un simple formulaire HTML qui ne pointe sur aucune page. En effet, le code PHP vérifie si la page a été appelée avec la méthode POST, dans ce cas, les informations transmises sont envoyées vers la base de données.*

*Le fait d'avoir un bouton de type SUBMIT mais que la ligne `<form method="post">` ne contienne pas de propriété `action="page.php"` va utiliser l'URL courante dans le navigateur : la page s'appelle elle-même.*

*Je voudrais attirer ton attention sur la manière dont le programme transmet les variables PHP vers la base de données :*

```
$stmt = $conn->prepare("INSERT INTO chambres (numero, capacite) VALUES (?, ?)");  
$stmt->execute([$numero, $capacite]);
```

*Cette formulation permet un nettoyage des champs PHP pour éviter une injection SQL. La méthode `execute()` filtre et nettoie les variables en enlevant les signes comme " ou ' (ou plus précisément, les transformer en chaîne inactive).*

### E.3.5 Programmation de la page d'édition d'une chambre

Désormais, Damien vous propose de créer le fichier d'édition

chambres/editChambre.php

```
<?php
// Inclusion du fichier de connexion à la base de données
require_once '../config/db_connect.php';

// Méthode GET : on recherche la chambre demandée
$id = isset($_GET['id']) ? (int)$_GET['id'] : 0; // equiv. if isset() {$id = (int)$...} else { $id=0 }

// Vérifier si l'ID est valide
if ($id <= 0) {
    header("Location: listChambres.php");
    exit;
}

$conn = openDatabaseConnection();

// Méthode POST : Traitement du formulaire si soumis
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    $numero = $_POST['numero'];
    $capacite = (int)$_POST['capacite'];

    // Validation des données
    $errors = [];

    if (empty($numero)) {
        $errors[] = "Le numéro de chambre est obligatoire.";
    }

    if ($capacite <= 0) {
        $errors[] = "La capacité doit être un nombre positif.";
    }

    // Si pas d'erreurs, mettre à jour les données
    if (empty($errors)) {
        $stmt = $conn->prepare("UPDATE chambres SET numero = ?, capacite = ? WHERE id = ?");
        $stmt->execute([$numero, $capacite, $id]);

        // Rediriger vers la liste des chambres
        header("Location: listChambres.php?success=1");
        exit;
    }
} else {
    // Méthode GET : Récupérer les données de la chambre
    $stmt = $conn->prepare("SELECT * FROM chambres WHERE id = ?");
    $stmt->execute([$id]);
    $chambre = $stmt->fetch(PDO::FETCH_ASSOC);

    // Si la chambre n'existe pas, rediriger
    if (!$chambre) {
        header("Location: listChambres.php");
        exit;
    }
}

closeDatabaseConnection($conn);
?>

<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Modifier une Chambre</title>
    <link rel="stylesheet" href="../assets/style.css">

```



```
</head>
<body>
  <h1>Modifier une Chambre</h1>

  <?php if (isset($errors) && !empty($errors)): ?>
    <div>
      <?php foreach($errors as $error): ?>
        <p><?= $error ?></p>
      <?php endforeach; ?>
    </div>
  <?php endif; ?>

  <form method="post">
    <div class="form-group">
      <label for="numero">Numéro de Chambre:</label>
      <input type="text" id="numero" name="numero"
        value="<?= htmlspecialchars($chambre['numero']) ?>" required>
    </div>

    <div class="form-group">
      <label for="capacite">Capacité (nombre de personnes):</label>
      <input type="number" id="capacite" name="capacite"
        value="<?= $chambre['capacite'] ?>" min="1" required>
    </div>

    <div class="actions">
      <button type="submit">Enregistrer les modifications</button>
      <a href="listChambres.php">Annuler</a>
    </div>
  </form>
</body>
</html>
```

*Pour l'édition d'une chambre, il faut savoir quelle chambre existante, nous voulons éditer : l'appel provient normalement de la liste de chambre, avec une méthode GET qui envoie l'id de la chambre.*

*Le code va donc vérifier s'il y a un id valide, puis afficher la page avec les valeurs des champs récupérées avec un `SELECT FROM WHERE` classique.*

*Si c'est la méthode POST qui est reçue, cela signifie que le formulaire a été rempli et qu'il faut mettre à jour les valeurs dans la base de données (avec `UPDATE SET WHERE` sur l'id.*

*Là encore, tu remarques la structure similaire : le code PHP qui attend une méthode POST ou une méthode GET ! En général, POST sert pour les modifications et GET pour la transmission simple d'informations.*

*Ici, je veux te montrer comment on gère l'erreur d'un mauvais ID :*

```
if ($id <= 0) {  
    header("Location: listChambres.php");  
    exit;  
}
```

*Dans ce cas, la fonction header() indique simplement de rediriger vers une page web. On fera mieux un peu plus tard, pour la gestion des erreurs.*

```
$stmt = $conn->prepare("UPDATE chambres SET numero = ?, capacite = ? WHERE id = ?");  
$stmt->execute([$numero, $capacite, $id]);
```

*Encore une fois, tu peux constater que les données envoyées à PHP ne sont pas directement transmises dans la base de données, mais passées à une requête paramétrée. Chaque "?" représente une case vide qui sera remplie par la fonction execute().*

## E.4 Programmation du CRUD pour les clients

Damien attend désormais de vous, de vous inspirer des codes précédents pour gérer les clients.

- Réutiliser le code de listeChambres.php pour l'affichage des clients.
- Réutiliser le code de création de chambre, mais avec le client.
- Réutiliser le code de suppression.
- Réutiliser le code d'édition.

*Maintenant que tu as une structure et une partie du code opérationnel, tu peux t'en inspirer et l'adapter pour gérer les clients. Mais avant, pense à créer une version de ton application dans git, avec un commentaire parlant.*

## E.5 Programmation du CRUD des réservations

Cette partie s'avère plus difficile que les étapes précédentes, car nous allons devoir gérer les données existantes (clients et chambres) et les associer pour créer une réservation.

*Dans un premier temps, il faut afficher toutes les réservations existantes, mais le personnel de l'hôtel ne veut pas voir ceci :*

| 123 id | 123 client_id | 123 chambre_id | date_arrivee | date_depart |
|--------|---------------|----------------|--------------|-------------|
| 1      | 1             | 1              | 2025-05-10   | 2025-05-15  |
| 2      | 2             | 1              | 2025-06-20   | 2025-06-25  |
| 3      | 1             | 1              | 2024-01-15   | 2024-01-20  |
| 4      | 3             | 3              | 2024-02-05   | 2024-02-07  |
| 5      | 2             | 4              | 2025-03-18   | 2025-03-25  |
| 6      | 5             | 2              | 2025-03-15   | 2025-03-22  |
| 7      | 4             | 6              | 2025-04-10   | 2025-04-17  |
| 8      | 6             | 8              | 2025-05-01   | 2025-05-05  |

*Si les dates sont relativement lisibles, les index des chambres et des clients ne sont pas parlants : tu vas devoir écrire une requête SQL qui récupère les noms des clients et les numéros des chambres.*

### E.5.1 Explication de la requête

Pour cela, il faut utiliser des **jointures**, c'est-à-dire, joindre deux ou plusieurs tables ensemble pour créer une table contenant toutes les informations. Votre tuteur vous donne la requête et vous explique :

```
SELECT
  r.id, r.date_arrivee, r.date_depart,
  c.nom AS client_nom, c.telephone AS client_telephone, c.email AS client_email, c.nombre_personnes,
  ch.numero AS chambre_numero, ch.capacite AS chambre_capacite
FROM reservations r
JOIN clients c ON r.client_id = c.id
JOIN chambres ch ON r.chambre_id = ch.id
```

*On trouve toujours la structure SELECT quelque chose FROM une table, mais ici on va préciser qu'on joint la table client sur le critère d'une égalité entre id du client dans la table réservation et l'id du client dans la table client.*

*Ensuite, on fait la même jointure entre notre table de réservation augmentée (réservation+client) et la table chambre.*

*JOIN table ON filtre est la manière la plus élégante de rédiger ce genre de requête.*

*Pour gagner de la place, on précise que la table client aura un alias **c**, réservation aura un alias **r** et chambre, pour alias **ch**. Ainsi, il est plus facile d'écrire les champs dont on a besoin : ch.numero signifie "dans la table chambre, prendre le champ numéro".*

*Cette notation est très utile pour les champs ayant le même nom, comme... id : id dans la table client n'est pas id dans la table chambre.*

## E.5.2 Programmation de l'affichage des réservations

Voici le code pour lister les réservations (avec les boutons pour éditer/supprimer)

listReservations.php

```
<?php
// Inclusion du fichier de connexion à la base de données
require_once '../config/db_connect.php';

// Fonction pour formater les dates
function formatDate($date) {
    $timestamp = strtotime($date);
    return date('d/m/Y', $timestamp);
}

// Récupération des réservations avec les informations des clients et des chambres
$conn = openDatabaseConnection();

$query = "SELECT r.id, r.date_arrivee, r.date_depart,
               c.nom AS client_nom, c.telephone AS client_telephone, c.email AS client_email,
               c.nombre_personnes,
               ch.numero AS chambre_numero, ch.capacite AS chambre_capacite
           FROM reservations r
           JOIN clients c ON r.client_id = c.id
           JOIN chambres ch ON r.chambre_id = ch.id
           ORDER BY r.date_arrivee DESC";

$stmt = $conn->query($query);
$reservations = $stmt->fetchAll(PDO::FETCH_ASSOC);
closeDatabaseConnection($conn);
?>

<!DOCTYPE html>
<html lang="fr">
<head>
    <title>Liste des Réservations</title>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <!-- Lien vers la feuille de style externe -->
    <link rel="stylesheet" href="../assets/style.css">
</head>
<body>
    <div class="container">
        <h1>Liste des Réservations</h1>

        <div class="actions">
            <a href="createReservation.php" class="btn btn-success">Nouvelle Réservation</a>
        </div>

        <table>
            <thead>
                <tr>
                    <th>ID</th>
                    <th>Client</th>
                    <th>Contact</th>
                    <th>Chambre</th>
                    <th>Personnes</th>
                    <th>Arrivée</th>
                    <th>Départ</th>
                    <th>Statut</th>
                    <th>Actions</th>
                </tr>
            </thead>
            <tbody>
                <?php if (count($reservations) > 0): ?>
                    <?php foreach($reservations as $reservation): ?>
                        <?php
                            $aujour_d_hui = date('Y-m-d');
                            $statut = '';

```

```

        if ($reservation['date_depart'] < $aujourd_hui) {
            $statut_class = 'status-past';
            $statut = 'Terminée';
        } elseif ($reservation['date_arrivee'] <= $aujourd_hui &&
$reservation['date_depart'] >= $aujourd_hui) {
            $statut_class = 'status-active';
            $statut = 'En cours';
        } else {
            $statut_class = '';
            $statut = 'À venir';
        }
    }
    ?>
    <tr>
        <td><?=$reservation['id'] ?></td>
        <td><?=$htmlspecialchars($reservation['client_nom']) ?></td>
        <td>
            <strong>Tél:</strong>
            <?=$htmlspecialchars($reservation['client_telephone']) ?><br>
            <strong>Courriel:</strong>
            <?=$htmlspecialchars($reservation['client_email']) ?>
        </td>
        <td>N° <?=$htmlspecialchars($reservation['chambre_numero']) ?>
            (<?=$reservation['chambre_capacite'] ?> pers.)</td>
        <td><?=$reservation['nombre_personnes'] ?></td>
        <td><?=$formatDate($reservation['date_arrivee']) ?></td>
        <td><?=$formatDate($reservation['date_depart']) ?></td>
        <td class="<?=$statut_class ?>"><?=$statut ?></td>
        <td>
            <a href="viewReservation.php?id=<?=$reservation['id'] ?>">
                Voir</a>
            <a href="editReservation.php?id=<?=$reservation['id'] ?>">
                Modifier</a>
            <a href="deleteReservation.php?id=<?=$reservation['id'] ?>"
                onclick="return confirm('Supprimer cette réservation?');">
                Supprimer</a>
        </td>
    </tr>
    <?php endforeach; ?>
    <?php else: ?>
    <tr>
        <td colspan="9">Aucune réservation trouvée.</td>
    </tr>
    <?php endif; ?>
</tbody>
</table>
</div>
</body>
</html>

```

*Ce code est plus compliqué car il y a beaucoup de codes PHP et HTML intriqués (mêlés).*

Rappel, vous pouvez ajouter des icônes dans les boutons avec `[Win][Shift][;]`, sinon, il existe des bibliothèques de symboles proposées avec Bootstrap, que nous utiliserons plus loin.

## F Mise en forme CSS : frameworks

Jusqu'à maintenant, nous avons créé notre squelette PHP et les différentes pages permettant de réaliser les fonctions demandées. Mais les pages sont laides, différentes et l'interface est donc désagréable.

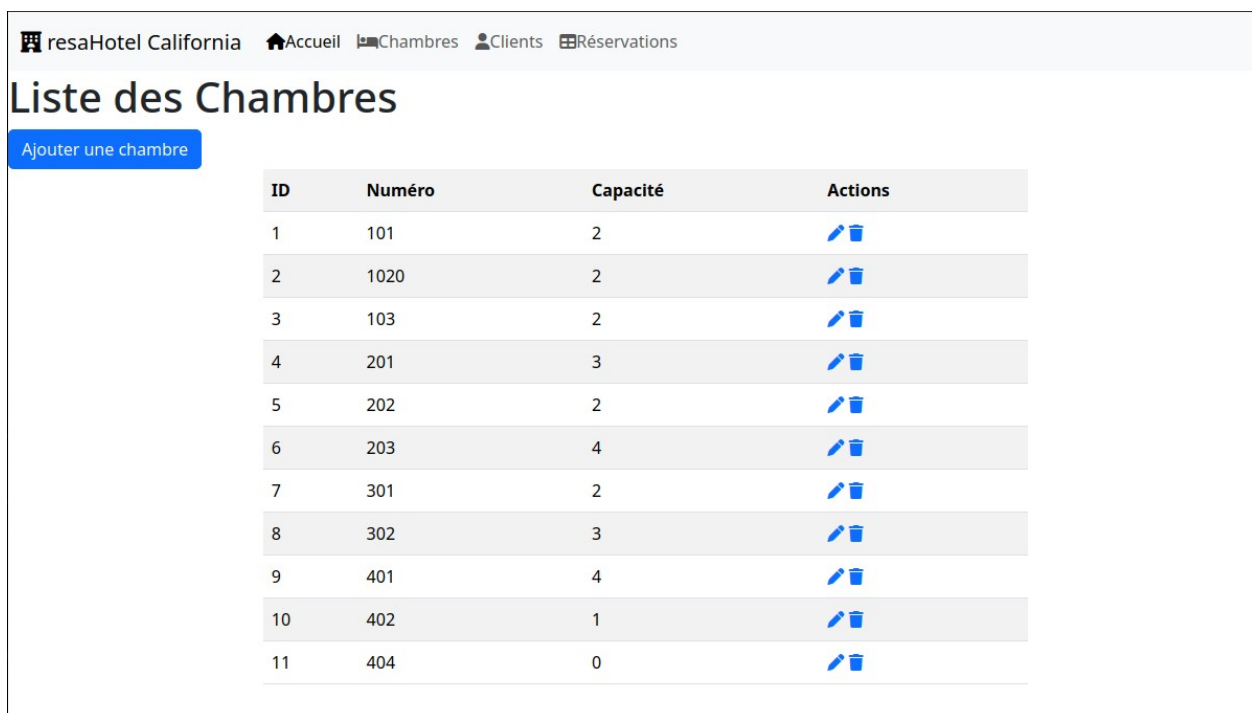
La plupart des employés auront le sentiment que l'application a été bâclée.

Pourtant, créer vous-même le code CSS pour ce site serait très chronophage, pour un résultat, souvent douteux.























Nous allons employer deux frameworks CSS qui vont fluidifier notre développement : l'application sera agréable, tout en ne nécessitant pas un temps de conception long. La contrepartie est que le site sera moins originale (aspect classique).

### F.1 Exemple de mise en forme

Voici un exemple du résultat attendu...



The screenshot shows a web application interface for 'resaHotel California'. The navigation bar includes 'Accueil', 'Chambres', 'Clients', and 'Réservations'. The main heading is 'Liste des Chambres', with a button 'Ajouter une chambre' on the left. Below is a table with 11 rows of room data.

| ID | Numéro | Capacité | Actions  |
|----|--------|----------|--|
| 1  | 101    | 2        |   |
| 2  | 1020   | 2        |   |
| 3  | 103    | 2        |   |
| 4  | 201    | 3        |   |
| 5  | 202    | 2        |   |
| 6  | 203    | 4        |   |
| 7  | 301    | 2        |   |
| 8  | 302    | 3        |   |
| 9  | 401    | 4        |   |
| 10 | 402    | 1        |   |
| 11 | 404    | 0        |   |

### F.2 Frameworks

Pour obtenir ce résultat, nous allons utiliser deux frameworks :

- **Bootstrap** qui propose des boutons, des couleurs et une interface proche de Twitter (et c'est normal, car c'est l'équipe de Twitter qui l'a conçu).
- **FontAwesome** qui propose des symboles à intégrer.

L'intégration d'un framework peut se faire de deux manières :

- Téléchargement et utilisation locale : le développeur télécharge la bibliothèque complète et la place dans un répertoire local du site.
- Lien de téléchargement distant CDN (Content Delivery Network). Le développeur indique au navigateur où trouver la bibliothèque.

Les deux méthodes ont leurs avantages et inconvénients. Ici, nous utiliserons la deuxième méthode, CDN.

## F.2.1 Bootstrap

Le site <https://getbootstrap.com/> est le site officiel du framework le plus connu. Vous devrez fournir deux liens dans vos pages pour qu'il fonctionne correctement :



### Include via CDN

When you only need to include Bootstrap's compiled CSS or JS, you can use [jsDelivr](#). See it in action with our simple [quick start](#), or [browse the examples](#) to jumpstart your next project. You can also choose to include Popper and our JS [separately](#).

```
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist." >
```

```
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dis" >
```

Le premier lien sera à mettre dans la section <head> de vos pages, car c'est la partie CSS de Bootstrap :

```
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-QWTKZyjpPEjISv5WaRU90FeRpok6YctnYmDr5pNlyT2brjXh0JMhJY6hW+ALEwIH" crossorigin="anonymous">
```



Les propriétés **integrity** et **crossorigin** sont utiles pour sécuriser la version et empêcher un accès vers un lien frauduleux.

- **Integrity** est une propriété qui contient le calcul de checksum du fichier, ici en utilisant la méthode d'empreinte SHA-384.
- **Crossorigin** permet, quant à elle, d'utiliser la ressource, sans envoyer de données sensibles (en restant donc anonyme) ;

Le deuxième lien est à placer à la fin du fichier, avant la balise </body>, car il s'agit des fonctions JavaScript associées.

```
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js" integrity="sha384-YvpcrYf0tY3lHB60NNkmXc5s9fDVZLESaAA55NDz0xhy9GkcIdslK1eN7N6jIeHz" crossorigin="anonymous" referrerpolicy="no-referrer"></script>
```



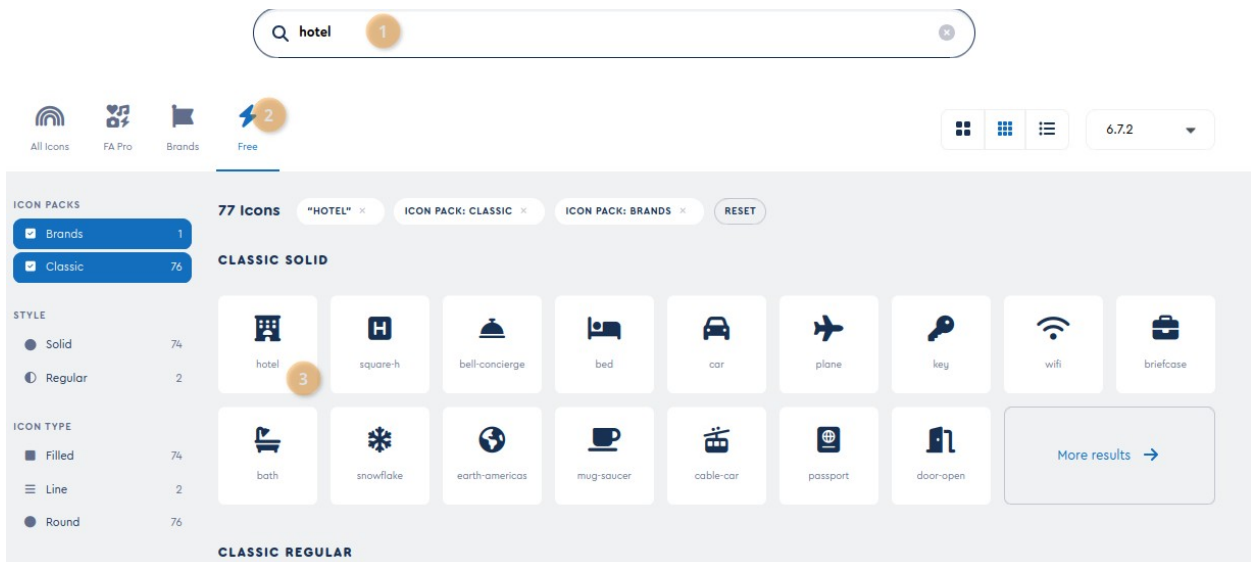


La propriété **referrerpolicy** est également une propriété pour protéger la confidentialité : elle indique au navigateur d'envoyer ou pas l'URL de la page contenant le script. La valeur no-referrer demande donc au navigateur de ne pas indiquer d'où il utilise ce CDN. Autrefois, le referrer était utilisé pour autoriser des accès à des ressources pour des partenaires : si on venait d'un site partenaire (transmission de l'URL de référence), alors on n'était pas obligé de s'authentifier.

## F.2.2 Font-Awesome

Le site <https://fontawesome.com/> vous permettra d'intégrer de belles icônes dans votre site, sous la forme de balises ayant la forme `<i class="fas fa-objet"></i>`.

En utilisant le moteur de recherche sur le site, vous pourrez trouver de nombreux symboles utilisables gratuitement :



Par exemple, dans la page listChambres.php, voici comment remplacer le texte "modifier" par une icône de stylo.

Dans la partie `<head>` il faut ajouter la balise suivante :

```
<link href="https://cdn.jsdelivr.net/npm/font-awesome@6.7.2/css/all.min.css" rel="stylesheet">
```

Et dans le corps de la page, à l'endroit du lien de modification, remplacez le texte "modifier" par `<i class="fas fa-pen"></i>` :

```
<a href="editChambre.php?id=<?=$chambre['id'] ?>"><i class="fas fa-pen"></i></a>
```

### F.3 Création d'un menu type navbar

Maintenant que vous disposez des fonctionnalités de Bootstrap et FontAwesome dans vos pages, vous allez pouvoir créer un menu unique pour toutes vos pages. Damien intervient :

*Je vois que tu as bien avancé : je vais maintenant te montrer comment faire une barre de menu très simple en HTML et Bootstrap. La doc de Bootstrap est disponible sur <https://getbootstrap.com/docs/5.3/getting-started/introduction/> et il suffit de taper 'navbar' dans le champ de recherche pour trouver toute l'information utile.*

*Il y a même un exemple visible : <https://getbootstrap.com/docs/5.3/components/navbar/#how-it-works>*

*Voici le fichier navbar.php à mettre dans le répertoire des assets :*

```
navbar.php
<nav class="navbar navbar-expand-lg navbar-light bg-light">
  <div class="container-fluid">
    <a class="navbar-brand" href="/resaHotelCalifornia/index.php"><i class="fas fa-hotel"></i>
resaHotel California</a>
    <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-
target="#navbarNav" aria-controls="navbarNav" aria-expanded="false" aria-label="Toggle navigation">
      <span class="navbar-toggler-icon"></span>
    </button>
    <div class="collapse navbar-collapse" id="navbarNav">
      <ul class="navbar-nav">
        <li class="nav-item">
          <a class="nav-link" aria-current="page" href="/resaHotelCalifornia/index.php">
            <i class="fas fa-home"></i>Accueil</a>
          </li>
        <li class="nav-item">
          <a class="nav-link" href="/resaHotelCalifornia/chambres/listChambres.php">
            <i class="fas fa-bed"></i>Chambres</a>
          </li>
        <li class="nav-item">
          <a class="nav-link" href="/resaHotelCalifornia/clients/listClients.php">
            <i class="fas fa-user"></i>Clients</a>
          </li>
        <li class="nav-item">
          <a class="nav-link" href="/resaHotelCalifornia/reservations/listReservations.php">
            <i class="fas fa-table"></i>Réservations</a>
          </li>
        </ul>
      </div>
    </div>
  </nav>
```

*Attention à bien adapter tes liens avec ton dossier d'origine, car j'utilise ici des liens absolus.*







Il faut maintenant placer cette barre de navigation au bon endroit dans tous les fichiers, par exemple, juste après la balise <body> pour que la barre de navigation soit tout en haut de la partie supérieure de la page :

```
<body>
  <?php include '../assets/navbar.php'; ?>
```

## F.4 Mettre en place les frameworks dans chaque page

Intégrez les deux frameworks (Bootstrap et FontAwesome) dans toutes vos pages en copiant/collant les liens CDN aux bons endroits (comme indiqué au paragraphe F.2.1 Bootstrap).







Essayez d'obtenir un aspect proche de celui-ci :

| ID | Numéro | Capacité | Actions  |
|----|--------|----------|--|
| 1  | 101    | 2        |   |
| 2  | 102    | 2        |   |
| 3  | 103    | 2        |   |
| 4  | 201    | 3        |   |

Pour le tableau ci-dessus, dans la balise `<table>` on utilise `class="table striped-table"`.

Pour le bouton vert, on utilise le code suivant :

```
<div class="actions">
  <a href="createChambre.php" class="btn btn-success">Ajouter une chambre</a>
</div>
```

| ID | Client        | Contact   | Chambre          | Personnes | Arrivée    | Départ     | Statut  | Actions   |
|----|---------------|---|------------------|-----------|------------|------------|---------|---|
| 10 | Isabelle Roux | <b>Tél:</b> 0698765432<br><b>Email:</b> isabelle.roux@email.com | N° 401 (4 pers.) | 3         | 01/07/2025 | 10/07/2025 | À venir |   |
| 12 | Marie Martin  | <b>Tél:</b> 0687654321<br><b>Email:</b> marie.martin@email.com  | N° 101 (2 pers.) | 3         | 20/06/2025 | 25/06/2025 | À venir |   |
| 2  | Marie Martin  | <b>Tél:</b> 0687654321<br><b>Email:</b> marie.martin@email.com  | N° 101 (2 pers.) | 3         | 20/06/2025 | 25/06/2025 | À venir |   |

Pour vous aider, voici quelques éléments complémentaires :

*L'utilisation d'une div de classe 'container' avec Bootstrap permet d'éviter que les éléments HTML ne collent au bord de la page. Il faut y placer tout le code la page web **après** la barre de navigation (elle, elle doit rester au bord de la page)*

```
<div class="container">
  <h1>Liste des Réservations</h1>
  <div class="actions">
    <a href="./createReservation.php" class="btn btn-success">Nouvelle Réservation</a>
  </div>
  ... code HTML ici...
</div>
```

*Les boutons ont plusieurs couleurs possibles :*

*<https://getbootstrap.com/docs/5.3/components/buttons/>*

*Prends les 7 minutes de temps pour cette vidéo qui est très utile pour comprendre les grilles Bootstrap : <https://www.youtube.com/watch?v=X-OhpQJtTcQ>*

*Je vais aussi t'expliquer quelques classes utilisées ici, en Bootstrap :*

Pour cette partie, il faut comprendre le fonctionnement de Bootstrap : le CSS s'applique en fonction de classes sélectionnées et elles peuvent être cumulatives, par exemple, pour la table des chambres ou des clients, on a utilisé :

```
<table class="table table-striped" ...>
```

Pour Bootstrap, la classe **table** permet une mise en forme des bordures, espacements, marges... tandis que **table-striped** va agir sur une ligne sur deux.

Pour un container, il est possible d'appliquer une classe mt-[0 à 5] ou mb-[0 à 5] pour obtenir une **marge top** ou **bottom**, qui sera fonction de la taille de police utilisée :

- mt-0 n'applique pas de marge supérieure
- mt-5 applique une grande marge supérieure
- mb-5 applique une grande marge inférieure

Bootstrap propose également un système de grille qui – par défaut – découpe la largeur de la page en douze espaces proportionnels. Utiliser une classe col-[1 à 12] signifie donc prendre de 1 à 12 cases de large sur les 12 cases disponibles :

|       |   |   |       |   |       |       |   |   |    |    |    |
|-------|---|---|-------|---|-------|-------|---|---|----|----|----|
| 1     | 2 | 3 | 4     | 5 | 6     | 7     | 8 | 9 | 10 | 11 | 12 |
| col-3 |   |   | col-2 |   | col-1 | col-6 |   |   |    |    |    |

Ces classes peuvent s'appliquer sur une classe `container` (classe générale) ou sur une classe `row` (ligne) ou `col` (colonne), par exemple :

```
<!-- Champ Numéro -->
<div class="row mb-3">
  <label for="numero" class="col-2 col-form-label text-end">Numéro</label>
  <div class="col-4">
    <input type="text" id="numero" class="form-control" placeholder="Entrez le numéro">
  </div>
</div>
<!-- Champ Capacité -->
<div class="row mb-3">
  <label for="capacite" class="col-2 col-form-label text-end">Capacité</label>
  <div class="col-4">
    <input type="text" id="capacite" class="form-control" placeholder="Entrez la capacité">
  </div>
</div>
```

Cela permet d'aérer les formulaires par une marge basse, tout en étant responsive (souvent, les tailles s'appliquent en fonction de la hauteur de police).

D'ailleurs, testez le fonctionnement du code, en réduisant la fenêtre, pour fois le comportement des polices et des éléments.

Astuce : certaines classes peuvent avoir des points de ruptures, avec les abréviations suivantes dans leur code :

| Abréviation (infix) | signification     | Définition d'écran    |
|---------------------|-------------------|-----------------------|
|                     | Extra small       | < 576 pixels de large |
| sm                  | small             | < 768 pixels          |
| md                  | medium            | < 992 pixels          |
| lg                  | large             | < 1200 pixels         |
| xl                  | Extra large       | < 1400 px             |
| xxl                 | Extra extra large | au-delà               |

Ainsi, vous pourrez trouver une classe `class="row mb-md-4"` qui signifie qu'en dessous de 992 pixels, l'écart ne s'applique pas et au-delà, Bootstrap applique une marge basse de 4 ¼ de rem (1 rem = root em, c'est un tantième de la taille de la police **racine** du document)

Écrire `class="col-12 col-md-6 col-xl-2"` permet alors d'avoir une seule colonne pour un petit écran (les objets passent les uns sous les autres), deux colonnes pour un affichage moyen et six colonnes sur un écran large.

Ainsi, pour placer un formulaire ou des champs, il peut être intéressant de choisir une grille comme celle-ci.

Exemple, pour ajouter une chambre :

```
<div class="container">
  <h1>Ajouter une Chambre</h1>
  <div class="container mt-3">

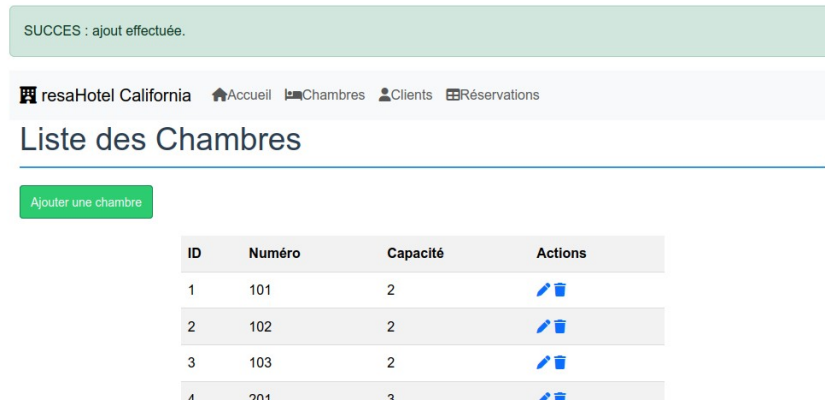
    <form method="POST">
      <!-- Champ Numéro -->
      <div class="row mb-3 align-items-center">
        <label for="numero" class="col-2 col-form-label text-end">Numéro</label>
        <div class="col-4">
          <input type="text" id="numero" class="form-control" placeholder="Entrez le numéro">
        </div>
      </div>
      <!-- Champ Capacité -->
      <div class="row mb-3 align-items-center">
        <label for="capacite" class="col-2 col-form-label text-end">Capacité</label>
        <div class="col-4">
          <input type="text" id="capacite" class="form-control" placeholder="Entrez la
capacité">
        </div>
      </div>
      <div class="row">
        <div class="col-2 text-end">
          <!-- Bouton de retour -->
          <a href="listChambres.php" class="btn btn-secondary">
            <i class="fas fa-arrow-left"></i> Retour
          </a>
        </div>
        <div class="col-4 text-end">
          <!-- Bouton validation -->
          <button type="submit" class="btn btn-primary">Valider</button>
        </div>
      </div>
    </form>
  </div>
</div>
```

*Je pense que tu peux adapter les commandes Bootstrap pour avoir un joli formulaire d'ajout de chambre ou d'ajout de client, je te laisse travailler dessus et lorsque c'est réussi, n'oublie pas d'effectuer un commit vers ton dépôt GIT.*



## G Gestion des erreurs

Nous allons étudier un mécanisme permettant d'afficher des messages sur une page différente de la page ayant généré l'action. Par exemple, si l'ajout d'une chambre fonctionne dans la base, l'affichage dans la page `listChambres.php` permettra de le savoir :



Le premier moyen de récupérer cette information est l'utilisation de la méthode GET. L'appel à la page `listChambres.php` peut fournir des paramètres que la page `listChambres.php` pourra afficher.

L'autre moyen serait l'utilisation d'un cookie serveur (qui sera proposé en annexe, si le temps le permet) ;

### G.1 Gestion affichage

La page `listChambres.php` doit être modifié et voici la proposition de Damien :

*Le principe est de s'appuyer au maximum sur le framework Bootstrap. Il propose des boites d'alertes esthétique et dynamique, dont l'exemple ci-dessous de la documentation :*

```
<div class="alert alert-warning alert-dismissible fade show" role="alert">
  <strong>Holy guacamole!</strong> You should check in on some of those fields below.
  <button type="button" class="btn-close" data-bs-dismiss="alert" aria-label="Close"></button>
</div>
```

*Cet exemple utilise la classe **alert-dismissible**, mais associe un bouton pour cette alerte (qui est géré par la partie JavaScript du framework). L'un ne va pas sans l'autre.*

*Dans notre code, il faut intégrer la récupération d'une variable par la méthode GET :*

```
<?php
if (isset($_GET['message'])) {
    $message = htmlspecialchars(urldecode($_GET['message']));
}
?>
```

*Ici, on vérifie si une variable message existe et si c'est le cas, on nettoie les caractères spéciaux avec `htmlspecialchars()` et comme on transmet via une URL, on décode les caractères encodés.*

Je t'explique : dans une URL, on ne peut pas transmettre un caractère espace ' ', donc le navigateur remplace ce caractère par %20, comme les liens du site de ton prof, monsieur Roumanet :

<http://david.roumanet.free.fr/BTS-SIO/Bloc2-Dev/270%20-%20PHP%20expert>

Il faut donc décoder ces caractères (les accents et autres caractères qui ne doivent pas apparaître dans une URL). C'est l'action de `urldecode()`.

À l'inverse, `htmlspecialchars()` va modifier les caractères pouvant introduire du code dans un texte à afficher, le rendant inoffensif. Par exemple, `<script>alert('XSS')</script>` deviendra `&lt;script&gt;alert('XSS')&lt;/script&gt;`; ce qui n'est pas exécutable par le navigateur.



Une injection XSS (Cross-Site Scripting ou script par site croisé) consiste à injecter un code HTML/CSS dans une saisie, avec l'espoir qu'il sera affiché tel quel par le site (donc interprété par le navigateur). L'objectif est généralement de récupérer le cookie-session de l'utilisateur, pour obtenir un accès non autorisé.

En adaptant cet exemple, il suffit de prévoir l'emplacement où tu veux écrire ton message d'erreur et :

listChambres.php

```
<?php
// Gestion des messages d'erreurs
if (isset($_GET['message'])) {
    $message = htmlspecialchars(urldecode($_GET['message'])); // limiter les injections XSS

    if (strpos($message, 'ERREUR') !== false) {
        echo "<div class='alert alert-warning alert-dismissible fade show' role='alert'>"
            . $message
            . "<button type='button' class='btn-close' data-bs-dismiss='alert' aria-label='Close'>"
            . "</button></div>";
    } else {
        echo "<div class='alert alert-warning alert-dismissible fade show' role='alert'>"
            . $message
            . "<button type='button' class='btn-close' data-bs-dismiss='alert' aria-label='Close'>"
            . "</button></div>";
    }
}
?>
```

Dans cette situation, j'ai ajouté le choix d'une couleur pour le message, en fonction du contenu.

Pour simplifier la compréhension, je recherche la présence de la chaîne "ERREUR" pour afficher avec la classe Bootstrap `alert-warning` ou bien avec la classe `alert-success` si tout va bien. La concaténation des chaînes a seulement pour but de rendre la lecture du code plus facile qu'une longue ligne de code.



## G.2 Gestion de transmission de message

La création d'un message à envoyer à la page `listeChambres.php` se fait chaque fois que l'on trouve le code suivant :

```
header("Location: listeChambres.php");
```

Dans la page de création de chambre ou dans la page d'édition de chambre ou encore, dans la page de suppression de chambre. Le code sera le même (à quelques adaptations près) sur chacune des pages.

Comme tu vois, la fonction `header()` appelle une URL, il est donc possible de transmettre des informations avec les signes `?` et `&` : `listeChambres.php?message=Hello&autrevariable=1...`

Je te propose de n'utiliser qu'une variable `message` pour le moment. Il faut modifier la ligne d'appel comme ceci :

```
header("Location: listeChambres.php?message=$encodedMessage");
```

La variable `$encodedMessage` doit contenir notre message, après encodage au format acceptable dans une URL, par la fonction `urlencode()`.

Voici le code que je te propose pour la page `createChambre.php`, tout en haut du fichier :

`createChambre.php`

```
<?php
require_once '../config/db_connect.php';

if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    $numero = $_POST['numero'];
    $capacite = $_POST['capacite'];
    if (empty($numero) || empty($capacite) || !is_numeric($numero) || !is_numeric($capacite)) {
        $encodedMessage = urlencode("ERREUR : une ou plusieurs valeurs erronée(s).");
        header("Location: listeChambres.php?message=$encodedMessage");
    } else {
        $conn = openDatabaseConnection();
        $stmt = $conn->prepare("INSERT INTO chambres (numero, capacite) VALUES (?, ?)");
        $stmt->execute([$numero, $capacite]);
        closeDatabaseConnection($conn);

        $encodedMessage = urlencode("SUCCES : ajout effectuée.");
        header("Location: listeChambres.php?message=$encodedMessage");
        exit;
    }
}
?>
```

Tu peux désormais tester tes modifications en appelant la page `createChambre.php` et en validant sans rien saisir (message d'erreur) ou en saisissant un numéro et une capacité valide (message de succès).

L'intérêt est que ton utilisateur puisse avoir un **feedback** : il est certain qu'il a réalisé une opération et que le serveur a réalisé un traitement.

### G.3 Adaptation à toutes les pages

Si le code fonctionne correctement, vous devez maintenant faire évoluer toutes les pages :

- editChambre.php
- deleteChambre.php
- createClient.php
- editClient.php
- deleteClient.php
- listClients.php
- listReservations.php

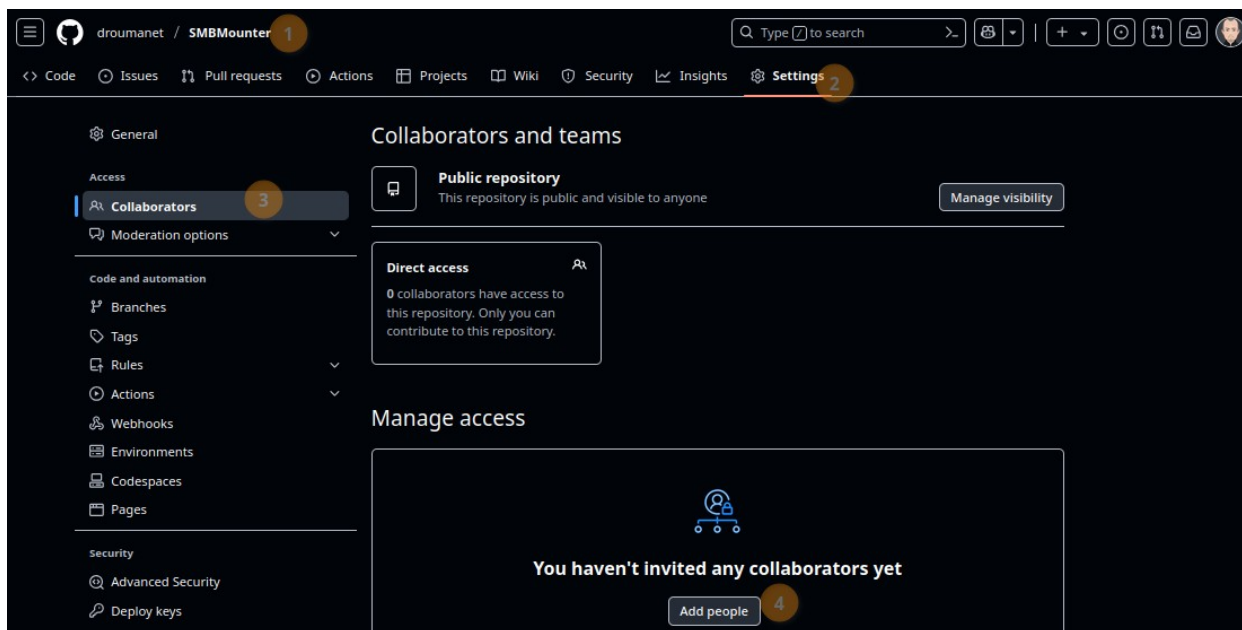
Pensez à effectuer des commits régulièrement.

### G.4 [Facultatif] : branche GIT et utilisation de cookie

L'utilisation de **cookie** est plus propre et souvent préféré pour les sites web. Nous allons simuler l'arrivée d'un autre développeur sur votre projet...

Trouvez-vous un binôme dans la classe, qui acceptera de travailler sur votre projet et vous sur le sien.

Dans Github, ajoutez son pseudo à votre projet :



Sélectionnez votre projet, dans **Settings**, choisir **collaborators** puis **[Add People]**

Recherchez le pseudo de votre binôme et ajoutez-le en tant que contributeur.

En tant que contributeur, vous devez cloner le projet :

- Pour cela, placez-vous dans le répertoire www et créer un répertoire 'clone' puis rentrez dedans
- Dans l'invite de commandes, faites un git clone du projet de votre binôme
- Vous devriez maintenant avoir une arborescence www/clone/resaHotelCalifornia (son projet) et www/resaHotelCalifornia (votre projet)

Pour ne pas casser le code, vous devez créer une branche 'cookie' et vous y placer avec les commandes

```
git checkout -b cookie
git branch
```

Effectuez les modifications suivantes :

createChambre.php

```
// Définir un cookie pour le message d'erreur
setcookie("message", "ERREUR : une ou plusieurs valeurs erronée(s).", time() + 10, "/"); // Expire
dans 10 secondes
header("Location: listChambres.php");
exit;
```

Et coté listChambres.php :

listChambres.php

```
<?php
// Gestion des messages d'erreurs
if (isset($_COOKIE['message'])) {
    $message = htmlspecialchars(urldecode($_COOKIE['message'])); // limiter les injections XSS

    if (strpos($message, 'ERREUR') !== false) {
        echo "<div class='alert alert-warning alert-dismissible fade show' role='alert'"
        . $message
        . "<button type='button' class='btn-close' data-bs-dismiss='alert' aria-label='Close'"
        . "</button></div>";
    } else {
        echo "<div class='alert alert-warning alert-dismissible fade show' role='alert'"
        . $message
        . "<button type='button' class='btn-close' data-bs-dismiss='alert' aria-label='Close'"
        . "</button></div>";
    }
}
?>
```

Envoyez la branche sur le projet distant (de votre binôme) :

```
git push origin cookie
```

Et enfin, effectuez une fusion avec son projet (remplacez master par main selon la branche initialement présente dans le projet) :

```
git checkout master
git merge cookie
git push origin master
```

## H Gestion des accès

---

La gestion des accès reprend le principe des cookies, mais il faut ajouter des choses dans la base de données :

Les utilisateurs du système (les clients et les employés) devront avoir un identifiant et un mot de passe.

## H.1

## I Annexes

---

### I.1 Définition de la base de données

Note : les champs date sont souvent complexes à gérer entre HTML et SQL. Le plus simple est de fournir une date au format "AAAA-MM-JJ", sous forme d'une chaîne de caractères.

#### I.1.1 Table chambres :

id (INT, clé primaire)

numero (VARCHAR)

capacite (INT)

#### I.1.2 Table clients :

id (INT, clé primaire)

nom (VARCHAR)

telephone (VARCHAR)

email (VARCHAR)

nombre\_personnes (INT)

#### I.1.3 Table reservations :

id (INT, clé primaire)

chambre\_id (INT, clé étrangère vers chambres.id)

client\_id (INT, clé étrangère vers clients.id)

date\_arrivee (DATE)

date\_depart (DATE)

### I.2 Autres