Projet 5 POO – PHP - Réservation de gîtes

Projet par groupes - Temps estimé : **15 jours**
Technologies, outils et concepts travaillés : PHP, programmation orientée objet (POO), HTML, CSS, JavaScript, MCD, UML, GIT, GITHUB

Ressources :

https://github.com/michelonlineformapro/Projet-5-Poo-Gites-Update-Router

Description du projet :

Vous allez réaliser une plateforme de réservation de gîtes se situant dans une zone géographique au choix).

- La plateforme est gérée par un seul administrateur qui pourra s'occuper d'ajouter, modifier ou supprimer des gîtes. CRUD
- Côté client on demande une connexion pour déposer une annonce d'un gite.
- On pourra visualiser la liste des différents gîtes ainsi que leurs détails.
- On pourra aussi effectuer une recherche pour trouver un gîte selon ses critères et ses disponibilités. (Date d'arrivée, date de départ, nombre de chambre, nombre salle de bain, zone géographique, Type de logement, etc…).

Il vous faudra créer une classe représentant un hébergement.

Il existera plusieurs catégories d'hébergements (exemple : chambre, appartement, maison, villa…) = clé étrangère.

- Intitulé de l'hébergement
- Description
- Photo(s)
- Nombre de chambres
- Nombre de salles de bain
- Emplacement géographique
- Prix
- Disponibilité
- Clé étrangère = Type de logement = Catégories

Chaque catégorie d'hébergement peut avoir des spécificités, par exemple pour un appartement ou un camping, on peut indiquer le nombre de pièces, pour une maison la présence d'un jardin particulier, douche, etc.

Côté administrateur on aura :

Un back-office (connecté à une base de données) permettant : CRUD

- De visualiser l'ensemble des gîtes et leur statut (occupé ou libre)
- D'ajouter des gîtes
- De supprimer des gîtes
- De les modifier

Côté utilisateur on aura :

- Une page accueil avec :
  - La liste des gîtes disponibles
  - Un formulaire de recherche (date de départ, date de fin, nombre de couchages, etc…)
- Une fiche par gîte avec un formulaire de réservation + une connexion
- La réception d'un mail lorsque la réservation est effectuée (mailTrap en local et phpmailer)

Gestion des disponibilités :

- Niveau 1 :
  - Lorsqu'un gîte est réservé il devient indisponible et ne plus apparaître dans les recherche. Une fois disponible on peut le réserver à nouveau
- Niveau 2 :
  - Lorsqu'un gîte est réservé pour une période il devient indisponible pour celle-ci. Il reste accessible par une recherche mais il faudra indiquer les jours d'indisponibilité. Il est possible de faire une réservation pour les jours disponibles.

Les plus : à réaliser si vous avez le temps

- Afficher un calendrier de disponibilité de chaque gîte
- Afficher la localisation des gîtes sur une carte
- Créer un espace membres pour les clients

Rappel des tâches :

1. Benchmark + maquette du site (temps estimé : 3 jours)
2. Concevoir un MCD (Model Conceptuel de Données)
3. Concevoir un MLD (Model Logique de Données)
4. Concevoir le modèle physique
5. Concevoir un diagramme de séquence

6. Concevoir le diagramme de la classe "hébergement" ensemble
7. Se répartir pour la réalisation du back-end & front-end
8. Créer un dépôt GitHub, effectué des commit régulier et fusionner (merge) les commit

Objectifs :

- Découvrir et manipuler la programmation orientée objet en PHP
- Consolider les bases du CRUD
- Consolider/approfondir les bases du langage SQL avec des appels de clé étrangère

Thèmes :

- Programmation orientée objet
- Bases de données

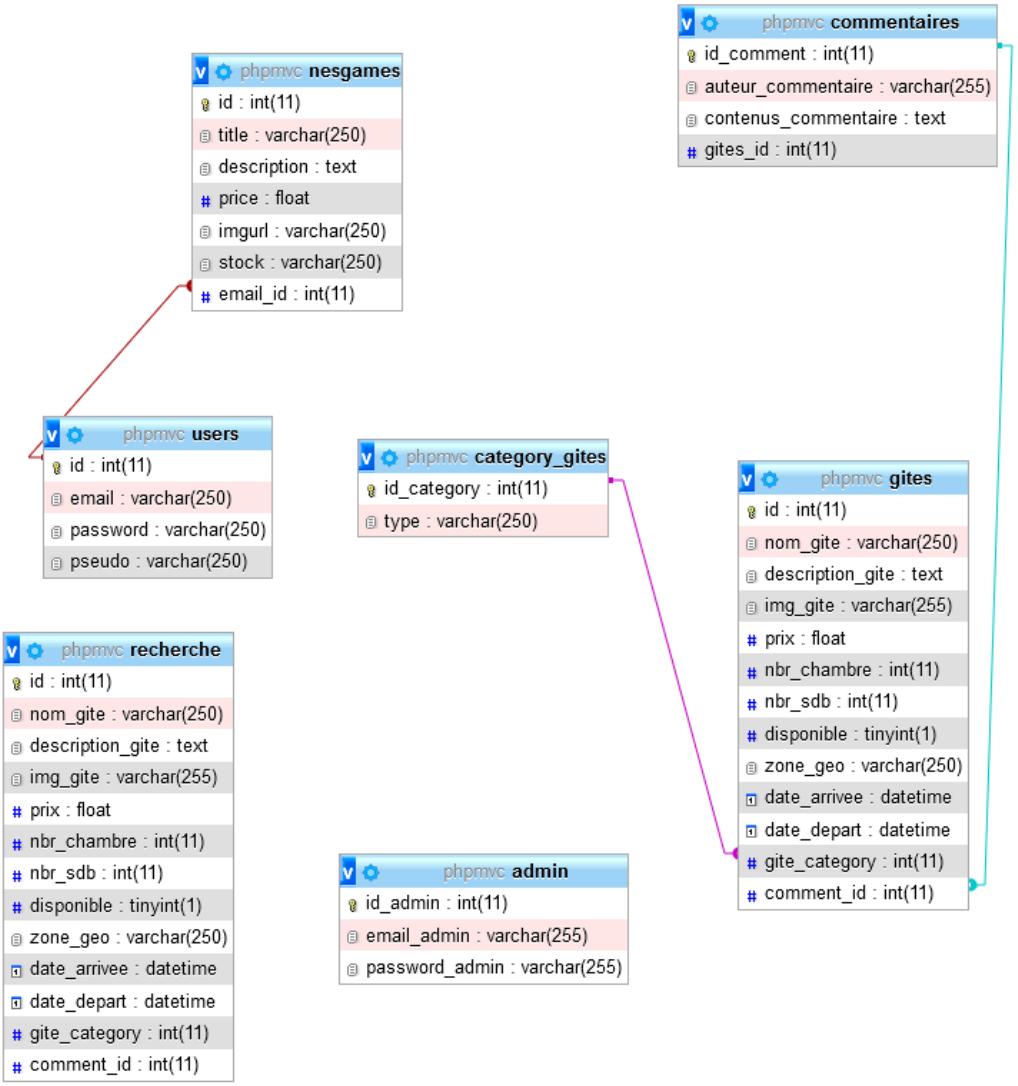
Ressources :

Parcours "PHP > Programmation orientée objet" sur vos-formations.com

- https://www.pierre-giraud.com/php-mysql-apprendre-coder-cours/introduction-programmation-orientee-objet/
- https://www.grafikart.fr/formations/programmation-objet-php
- https://sql.sh/

Etapes :

Votre MCD (vous pouvez faire le vôtre différemment)



BACKEND :

1- Créer un Template (navbar + conteneur principale + footer)
2- Créer une classe de connexion PDO (Classe + Méthode)

3- Votre espace d'admistration
   a. Dans la navbar -> connexion -> formulaire de connexion sécurisé
   b. Une fois connecter le bouton connexion disparait et devient déconnexion et laisse apparaitre l'onglet administration
   c. Utilisation de $_SESSION [''] et if {} else {} pour vérifier la connexion
4- Une fois connecter créer votre crud Gites avec des redirection ou modal
   a- Soit un formulaire de connexion pour l'administrateur

b- Connexion par email (et / ou nom) et mot de passe
c- Créer une classe Administration et une méthode login
d- Instance de la classe administration dans la page connexion et appel de la méthode login ()
e- Redirection vers votre page d'accueil du crud
f- Réalisé les 5 opération crud des gites.

FRONTEND:

1- Un moteur de recherche de gite par date arrivée, date de départ et nombre de chambre
2- Une barre de recherche par nom des gites
3- Une page d'accueil avec tous les gites disponibles (grisé les gites non dispos ou les caché)
4- Réalisé une requête SQL avec une variable date départ < date du jour
5- Afficher seulement le bouton détails
6- Sur la page détail ajouter un bouton réserver
7- Créer un formulaire de réservation (Email au minimum + PAS OBLIGATOIRE = nom, prénom, adresse etc…)
8- Créer une classe PHP Mailer qui envoie un email à mailTrap
9- Renvoyer votre formulaire de réservation vers l'instance de la classe PHP Mailer et sa méthode d'envoi
10- Dans le gabarit email, ajouter toutes les infos du gite et un bouton confirmer la réservation
11- Quand le gite est réservé il devient grisé ou disparait jusqu'à l'échéance de la date de départ
12- Réalisé une redirection vers la page d'accueil

LES GROUPES :

1- Melissa et Maxime
2- Amine et Guillaume
3- Nina et Yves
4- Victor et Charlotte
5- Stéphane et Vincent
6- Ayoub et Chau

RAPPELS :

La programmation orientée objet :

- Le code que vous développez répondra à certaines contraintes et certains besoins.
- Structurez l'ensemble de votre code pour le rendre plus solide et facile à entretenir ou à faire évoluer.

- Être capable de faire les bons choix de structuration, logique et organisation de votre code vous différencie en tant que développeur

Les objets et les classes :

- En programmation orienté objet, une **classe** est un modèle ou un mode d'emploi tandis que les **objets** sont les objets créés à partir de ce modèle

- Une classe possède un ensemble de variables, appelées **propriétés** ainsi qu'un ensemble de fonctions appelées **méthodes**.
- Il est possible de créer une classe à partir d'une classe grâce au mot-clé new. On appelle cela une **instance**.
- Pour accéder aux propriétés ou aux méthodes d'une classe, on utilise le symbole ->.

Créer des classes :

- Les objets possèdent des **propriétés** et des **méthodes**
- Syntaxe de base < ?php class MaClasse {} ?>
- Pour désigner une instance, nous utilisons le mot clé **$this**, permettant de faire appel aux propriétés ou méthodes de notre objet au sein d'une autre méthode.

Une API publique de vos objets

- Vous réduisez la complexité d'utilisation des classes en privatisant les méthodes et les propriétés avec private.
- Nous pouvons appliquer **le principe d'encapsulation**, en utilisant des getters et des setters, permettant d'accéder aux propriétés et de les modifier en s'assurant de garder des valeurs cohérentes.
- En réduisant la complexité, vous réduisez le risque d'erreur.

Utilisez les propriétés et les méthodes statiques :

- En utilisant la staticité vous pouvez partager des valeurs, ou appeler des méthodes "utilitaires", n'ayant pas d'impact sur l'état de l'objet.
- Il est possible et recommandé de créer des constantes pour les informations qui ne changent pas, grâce au mot-clé const.
- Pour appeler une propriété ou méthode statique au sein d'une classe, il faut utiliser le mot clé **self**.

Exploitez les méthodes communes aux objets :

- Il existe des méthodes dites magiques, qui sont appelées automatiquement par PHP. Elles commencent par **__.**
- __construct est la méthode appelée lors de la création d'un objet ; __destruct lors de sa suppression de la mémoire.
- Il existe de nombreuses autres méthodes magiques disponibles sur la documentation PHP.

Les classes et l'héritages :

- L'héritage nous permet de passer les propriétés et méthodes d'une classe "parent" à des classes "enfants".
- Vous pouvez structurer votre code et éviter la duplication en héritant d'une classe.
- L'héritage s'effectue avec l'usage du mot clé extends juste après le nom de la classe à étendre, suivi de la classe dont il faut hériter.

Profiter de l'héritages :

- La façon d'accéder aux propriétés des classes parentes s'effectue avec la flèche **->**.
- La façon d'accéder aux méthodes des classes parentes s'effectue également à l'aide de la flèche **->**.
- Lorsqu'il s'agit d'une méthode ou d'une propriété statique parente, l'accès s'effectue avec le mot clé **parent**.
- Vous pouvez réécrire une méthode existante dans un enfant, à condition de respecter sa signature.

Contrôler l'accès aux propriétés et aux méthodes des objets :

- En utilisant le mot clé private, vous empêchez quiconque autre que la classe courante de manipuler les propriétés et méthodes.
- En utilisant le mot clé protected, vous donnez l'autorisation à la classe courante ainsi qu'à ses enfants de manipuler les propriétés et méthodes ; mais pas aux éléments extérieurs à la classe, ou extérieurs aux objets de cette classe.
- Et comme nous l'avons vu précédemment, avec le mot-clé public, il est possible de manipuler la méthode ou la propriété depuis la classe, ses classes enfants et depuis l'extérieur.
- Autrement dit, pour modifier une propriété depuis un enfant, il faut qu'elle possède la visibilité public ou protected, ou qu'elle possède un mutateur.

Contraindre l'usage des classes :

- En utilisant le mot clé **abstract**, vous pouvez imposer à une classe d'être héritée.
- Une classe abstraite ne peut plus être instanciée seule, et peut contenir des méthodes abstraites.
- Une méthode abstraite doit être implémentée dans les classes enfants, ou alors celle-ci doit aussi être abstraite.
- Vous pouvez également interdire l'héritage à l'aide du mot clé final **sur** une classe ou sur une méthode.

Gérer le comportement d'une classe parente :

- Pour manipuler une constante et conserver la valeur de la classe courante, il faut y faire référence avec le mot clé **self**.
- Pour manipuler une constante et utiliser la valeur de la classe courante ou héritée si redéfinition, il faut y faire référence avec le mot clé **static**.

Les namespaces :

- Vous pouvez avoir plusieurs classes du même nom en les cloisonnant dans un espace de noms.
- Déclarer un espace de noms s'effectue à l'aide du mot clé **namespace**.
- Un espace de noms peut être de plusieurs niveaux séparés par un ****.
- Par défaut, PHP utilise un espace de noms global représenté par un ****.
- Lorsque vous exploitez un espace de noms spécifique pour une classe, vous pouvez le déclarer pour tout l'espace de noms courant avec le mot clé **use**.

La structure des projets :

- Les espaces de noms peuvent être utilisés pour correspondre à l'arborescence des fichiers, dans le but d'importer ces fichiers dynamiquement (PSR-4).
- Utiliser SPL combiné aux espaces de noms permet de charger les classes à la volée.
- Séparer les classes par fichiers réduit le nombre de fichiers à charger et à interpréter pour le langage.

Les traits (héritage multiple) :

- Les traits vous offrent la possibilité d'étendre "horizontalement" votre code.
- Vous pouvez utiliser plusieurs traits dans une classe.
- Un trait peut être composé d'autres traits.

Un contrat imposé avec les interfaces :

- Les interfaces vous offrent plus de souplesse et d'anticipation en permettant de :
    - Typer des arguments et retours de méthodes qui n'existent pas encore ;
    - Ou encore garantir qu'un code fonctionnera toujours sans s'imposer d'avoir à étendre des classes entières lors d'évolutions futures.

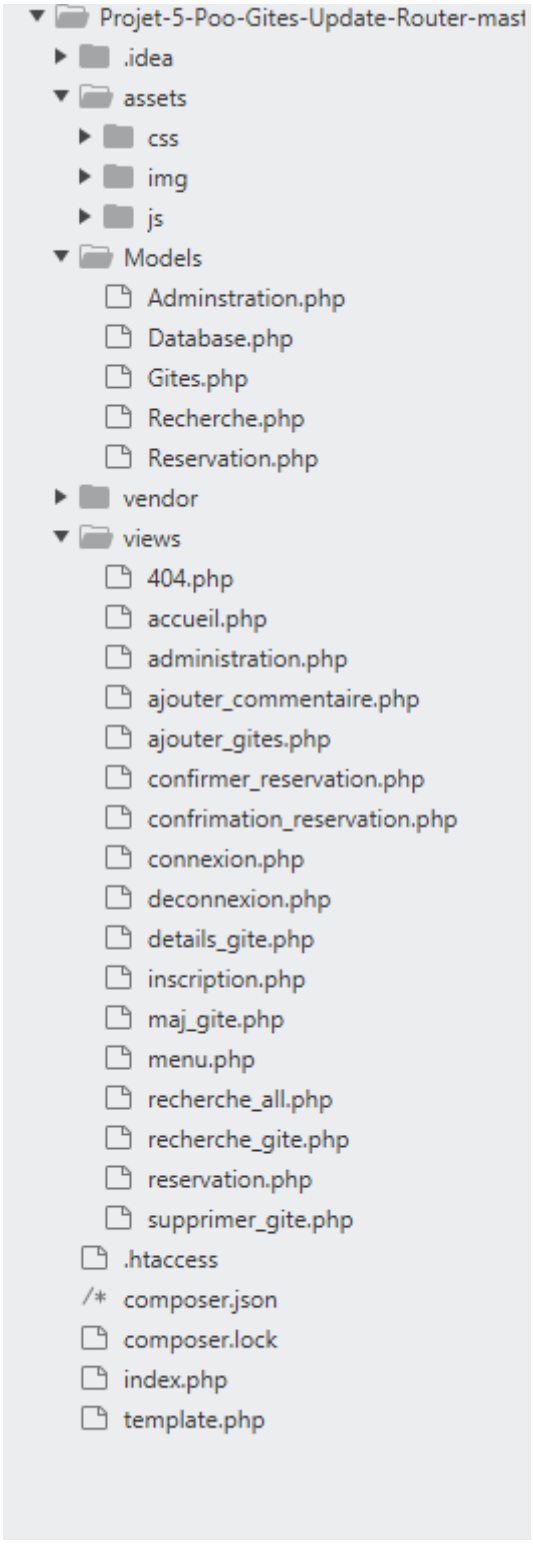
Composition VS héritage :

- Préférez la composition plutôt que l'héritage quand c'est possible.
- Pour savoir quand choisir l'un ou l'autre, posez-vous la question "est un"/"possède un".
- N'utilisez jamais les dépendances d'une dépendance directement, au risque d'avoir à reconstruire trop de choses après un simple changement.

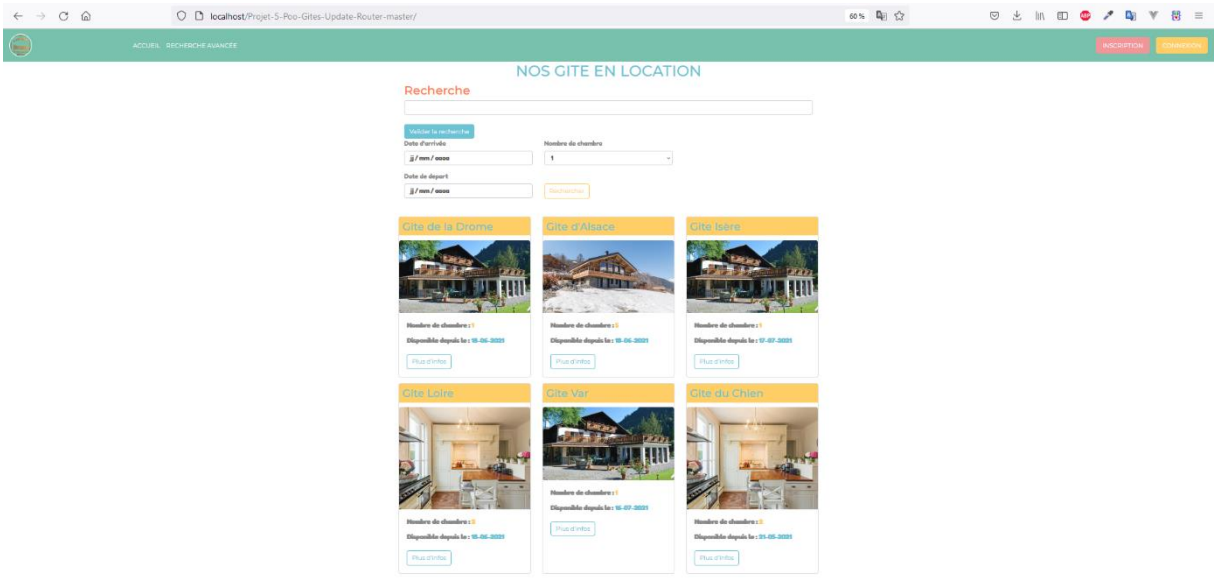
Gérer les erreurs avec try catch et exception :

- PHP permet de gérer les erreurs avec les classes Exception en faisant remonter l'erreur à travers toute la pile d'exécution, grâce au mot clé throw.
- Les Exceptions offrent plus de finesse pour la gestion des erreurs avec le bloc try**...catch**.
- Créer vos exceptions vous donne plus de clarté dans votre code et dans le traitement des erreurs.
- Il est possible d'enchaîner les catchs pour gérer les différents types d'erreurs.
- Le bloc finally placé après le(s) bloc(s) catch permet d'exécuter du code, qu'il y ait eu des erreurs, ou non.

La structure du projet POO :

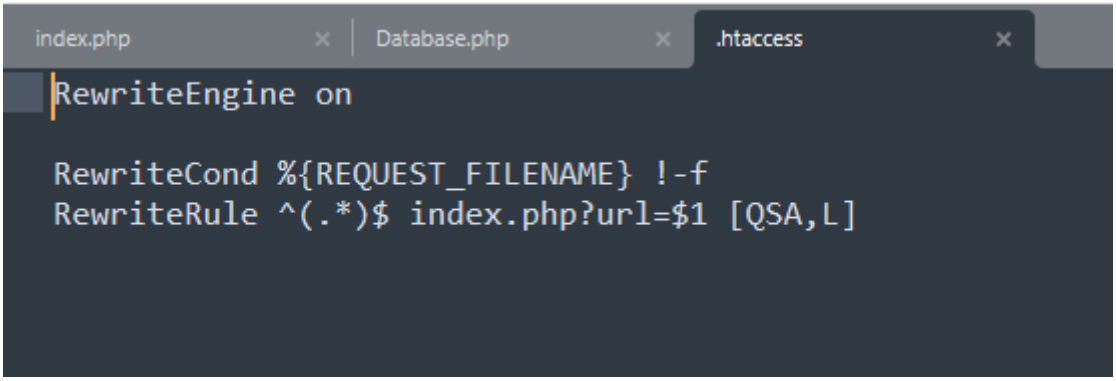


La page d’accueil :



Etapes :

1. Configuration apache (.htaccess)



2. Le router et des URL propres

```php
<?php
session_start();
ob_start();
//Utilisation de la variable superglobale $_GET['']

if(isset($_GET['url'])){
    $url = $_GET['url'];
}else{
    $url = "accueil";
}

if(!isset($_GET['url']) || empty($_GET['url'])){

}

if($_GET['url'] === ''){
    $url = 'accueil';
}


//Appel des vues

// url nom du dossier/index.php?url=accueil

if($url === 'accueil'){
    require 'views/accueil.php';
}elseif ($url === 'connexion'){
    require 'views/connexion.php';
}elseif ($url === 'deconnexion'){
    require 'views/deconnexion.php';
}elseif (isset($_SESSION['connecter']) && $_SESSION['connecter'] === true && $url === "administration"){
    require 'views/administration.php';
}elseif (isset($_SESSION['connecter']) && $_SESSION['connecter'] === true && $url === "ajouter_gites"){
    require "views/ajouter_gites.php";
}elseif ($url === "details_gite" && isset($_GET['id']) && $_GET['id'] > 0){
    require "views/details_gite.php";
}elseif (isset($_SESSION['connecter']) && $_SESSION['connecter'] === true && $url === "supprimer_gite" && isset($_GET['id']) && $_GET['id'] > 0){
    require "views/supprimer_gite.php";
}elseif (isset($_SESSION['connecter']) && $_SESSION['connecter'] === true && $url === "maj_gite" && isset($_GET['id']) && $_GET['id'] > 0){
    require "views/maj_gite.php";
}elseif ($url === "rechercher_gite"){
    require "views/recherche_gite.php";
}elseif ($url === "reservation" && isset($_GET['id']) && $_GET['id'] > 0){
    require 'views/reservation.php';
}elseif ($url === "confirmer_reservation"){
    require "views/confirmer_reservation.php";
}elseif ($url === "incription"){
    require "views/inscription.php";
}elseif (isset($_SESSION['connecter_user']) && $_SESSION['connecter_user'] === true && $url === "ajouter_commentaire" && isset($_GET['id']) && $_GET['id'] > 0){
    require "views/ajouter_commentaire.php";
}elseif ($url === "rechercher"){
    require "views/recherche_all.php";
}


elseif($url !=  '#:[\w]+)#'){
    require 'views/404.php';
}



$content = ob_get_clean();
require "template.php";
```

3. La connexion à la base de données (model)

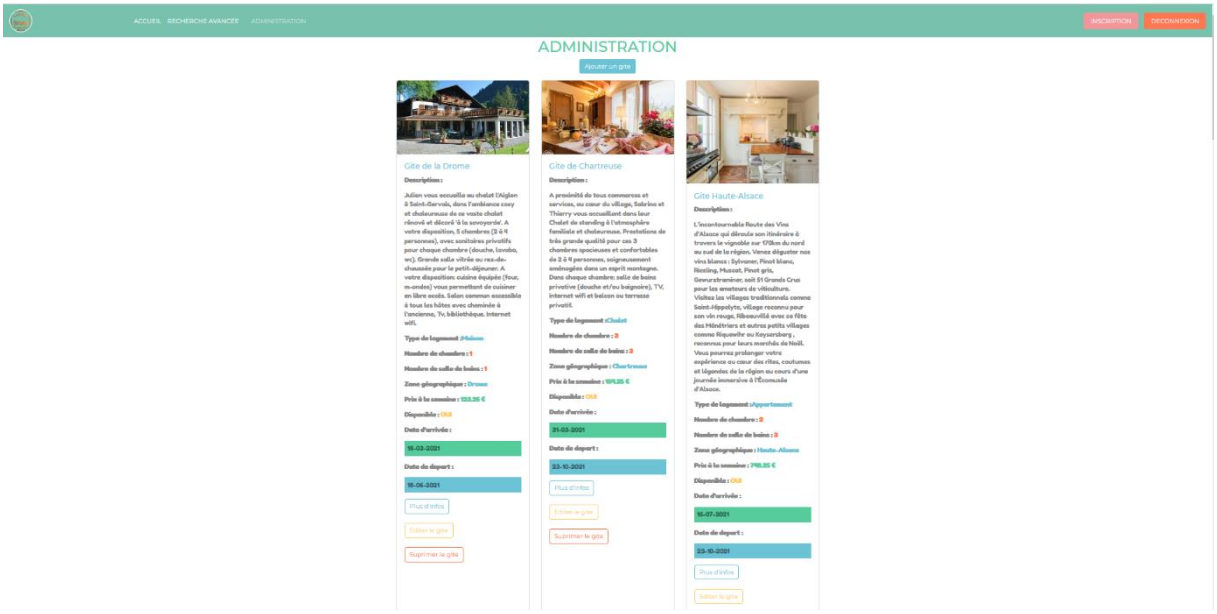
```php
<?php


class Database
{
    private $host = "localhost";
    private $dbname = "phpmvc";
    private $user = "root";
    private $pass = "";

    public $isConnected = null;


    public function getPDO(){
        try {
            $db = new PDO("mysql:host=".$this->host.";dbname=".$this->dbname.";charset=utf8", $this->user, $this->pass);
            $db->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
            //echo "Connexion a PDO";
            //Retourne la propriété $dbpour etre utlisé dans autres fichier
            return $db;

        }catch (PDOException $e){
            echo "erreur de connexion " .$e->getMessage();
        }
    }

}
```

4. Inscrire des administrateur (héritage de la classe mère Database.php) model/Administration.php
5. 3 méthodes
    a. Connecter l'administrateur
    b. Connecter les utilisateurs
    c. Inscrire des utilisateurs (pour la réservation de gite)

6. Pour la connexion : 2 Rôles

AU CLIC (jQuery)

7. Coté Administration (après connexion)



8. Réalisé les opérations de CRUD :
    a. Un model (classe) model/Gites.php
    b. Les vues (HTML + import du fichier model + instance du model + appel de la méthode concernée
    c. Exemple : ajouter_gite.php

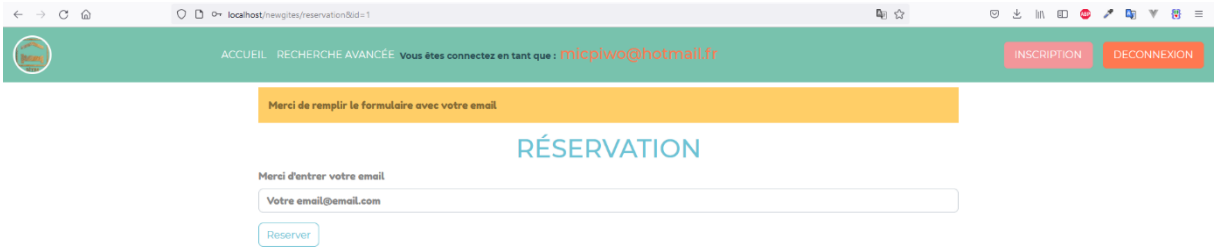
INSTANCE :

```php
<?php
$title = "Mon Gites.com -AJOUTER Gite-";
//Appel du fichier model
require "Models/Gites.php";
//Instance de la classe gite
$gites = new Gites();
```

UPLOAD IMAGE ET APPEL DE METHODE DE LA CLASSE gite.php

```php
<?php
//Gestion upload image
if(isset($_FILES['img_gite'])){
    $uploaddir = 'assets/img/';
    $img_gite = $uploaddir . basename($_FILES['img_gite']['name']);
    $_POST['img_gite'] = $img_gite;
    if(move_uploaded_file($_FILES['img_gite']['tmp_name'], $img_gite)){

        echo '<p class="alert-success">Le fichier est valide et à été téléchargé avec succès !</p>';
    }else{
        echo '<p class="alert-danger">Une erreur s\'est produite, le fichier n\'est pas valide !</p>';
    }
}else{
    echo "<p class='alert-warning mt-2 p-2'>Merci de respecter le format d'image valide : png, svg, jpg, jpeg, webp
        !</p>";
}

$gites->addGite();

/*
var_dump($_POST['nom_gite']);
var_dump($_POST['description_gite']);
var_dump($_POST['img_gite']);
var_dump($_POST['nbr_chambre']);
var_dump($_POST['nbr_sdb']);
var_dump($_POST['zone_geo']);
var_dump($_POST['prix']);
var_dump($_POST['disponible']);
var_dump($_POST['date_arrivee']);
var_dump($_POST['date_depart']);
var_dump($_POST['type_gite']);
var_dump($_FILES);
*/

?>
```

9. Créer un modèle (classe) Réservation (Modeles/Reservation.php)
10. Ajouter et configurer PHP Mailer() et créer un compte mailTrap
    a. https://github.com/PHPMailer/PHPMailer
    b. https://mailtrap.io/
11. Le visiteur doit être connecter pour réserver un gite



12. Un fois la réservation effectuée mailTrap (via un liens) valide la réservation et redirige vers votre site
13. Créer un modèle (classe) Rechercher (2 type de recherche pour filtrer les données via des requête SQL). Models/Recherche.php

Exemple de requête SQL tri entre 2 date :

```php
$today = date("Y-d-m");
$sql = "SELECT * FROM gites WHERE `date_depart` BETWEEN '$today' AND `date_arrivee` AND nbr_chambre = ? AND prix <= ?";
$req = $db->prepare($sql);
```

CONSEILS :

Faite de nombreux commits et des merges via git Kraken

https://www.gitkraken.com/