|  |
| --- |
| [Nom de la société] |
| Document de conception technique |
| ViveHôtel 1.1 |

|  |
| --- |
| Projet fil rouge  21/04/2023 |

Table des matières

[Rappel des fonctionnalités de l’application 2](#_Toc132808501)

[Environnement technique 3](#_Toc132808502)

[Description de la base de données 4](#_Toc132808503)

[Le modèle logique de données (modèle relationnel) 4](#_Toc132808504)

[Description des pages 8](#_Toc132808505)

[Particularités communes des pages 8](#_Toc132808506)

# Rappel des fonctionnalités de l’application

Le groupe ViveHotel possède un réseau comprenant une cinquantaine d’hôtels répartis sur le territoire français.

ViveHotel souhaite disposer d’un système d’information centralisé permettant de gérer l’ensemble des réservations pour tout hôtel de son réseau.

Le système doit fournir les services suivants :

* Gestion des hôtels et des chambres.
* Gestion des réservations.
* Consultation des disponibilités des chambres.
* Calcul du chiffre d’affaire annuel.
* Production de diverses statistiques :
  + Chiffre d’affaire par hôtel, par département.
  + Nombre de réservations par durées.

ViveHotel dispose d’un Centre d’appels au SRC (Service de Réservation Centrale). Un numéro vert unique (le même pour tous les hôtels) est mis en place pour toutes demandes de réservation par téléphone, mais les clients peuvent également réserver directement auprès des hôtels ou en ligne sur le site internet de ViveHotel.

Que la réservation soit réalisée par un opérateur ou directement sur internet par le client lui-même, un compte client est créé (s’il n’existe pas) avec la création d’une réservation.

## Environnement technique

Pour la réalisation de la partie front-end de l’application Vivehotel nous avons utilisé les technologies suivantes :

* HTML 5
* CSS 3
* Framework bootstrap 4.6
* JavaS+cript

Pour la partie back-end de l’application Vivehotel Nous avons utilisé :

* PHP 8 comme langage de script
* SQL comme langage de base de données

Pour le système d’exploitation nous avons utilisé Windows10

Pour le serveur web nous avons utilisé la suite logiciel XAMPP comprenant :

* Le serveur web PHP Apache
* Le gestionnaire de bases de données MYSQL

# Description de la base de données

## Création et connexion à la base de donnée :

La base de données complète du site web « Vivehotel » possède 12 tables et 13 contraintes d’intégrité. La liste de tous les champs et contraintes d’intégrités sont disponibles dans le fichier « script\_vivehotel\_vincent.sql » du dossier « document ».

Lorsque nous téléchargeons le projet sur notre machine, aucune base de données n’est automatiquement configurée. Il est donc nécessaire de suivre certaines étapes afin que le site puisse être exécuté.

Voici les différentes étapes en supposant que le lecteur ait déjà démarré un serveur avec la bonne version de XAMPP et la bonne version de « phpMyAdmin ». Nous allons seulement détailler les étapes nécessaires pour créer la base de données sur notre serveur SQL.

Première étape, Il faut se rendre dans le programme principal qui va connecter le site à la base de données, ce fichier est le fichier « config.php » qui se trouve dans « application/config ».

Pour établir une connexion à une base de données, il est nécessaire d’indiquer des paramètres multiples à une instruction de connexion. Voici un ensemble des paramètres possible dont les informations entre parenthèses sont les valeurs par défaut :

* Nom de domaine du serveur (« localhost »)
* Le port de connexion (3306)
* Le nom d’utilisateur (« root »)
* Le mot de passe (vide)
* La base de données choisie (« vivehotel »)

Ce fichier **« config.php »** contient toutes les données de connexion, sous forme de constantes. Ces constantes seront ensuite placées en début de programme de la génération des pages.

Notons également qu’il y a une gestion d’erreurs de connexion au sein du fichier « config.php ». Elle permet de donner une « seconde chance » à la connexion d’aboutir.

Aucune base de données n’est créée par défaut lorsque le projet est utilisé ; LLa réécriture du fichier « config.php » n’est pas suffisant pour que l’accès à la base de données soit fonctionnel.

Nous allons maintenant nous intéresser à la connexion concrète dans la base de données en utilisant la Bibliothèque PDO.

La classe PDO est une classe de connexion à une base de données. Instancier un objet de type PDO, c’est créer une connexion stockée en attribut d’objet. Le constructeur d’un objet PDO prend des paramètres de connexion, créés précédemment sous forme de constantes.

Deuxième étape en supposant que, « PhpMyAdmin 5.2.0 » est démarré. Il faut lancer le script de création de la basse de donnée dans PhpMyAdimin.

Troisième étape, une fois que la connexion à la base de données a été établie, il sera important de créer de la donnée adéquate pour le fonctionnement du site. Nous allons voir comment nous pouvons générer (ou réinitialiser) de la donnée de test, avec un ensemble de programmes PHP.

Le dossier « database » contient un fichier nommé « vue\_database\_dataset.php ». Ce script fait appel à une fonction dataset qui va :

* Créer et ajouter un ensemble d’enregistrements cohérents pour chaque table de cette base

Lancer le script «vue\_database\_dataset.php » va créer la base de données complète de ce projet, avec des données aléatoires mais cohérentes. Paramétrer la connexion à la base de données et lancer ce script est suffisant pour terminer l’installation du site.

Mais il est aussi possible de créer la base de donnée en lançant le programme « vue\_database\_creer.php » qui va directement créer la base de donnée vide dans votre PHPMyadmin.

Il suffira ensuite de paramétrer un alias sur XAMPP pour accéder plus facilement au site.

Notons qu’**il est nécessaire de supprimer le dossier « \_dataset » une fois que la base de données est créée** par mesure de sécurité.

* La liste des utilisateurs de la base de données avec PhpMyAdmin est donnée par le tableau suivant :

|  |  |
| --- | --- |
| Nom d’utilisateur | Profil |
| Root | SU |

Les sections suivantes de ce chapitre vont indiquer l’ensemble des schémas de données créés pour la base de données du projet.

## Le modèle conceptuel de données (MCD)

<MCD>

<entité>

hotel

o id (AI)

o nom varchar(50)

o téléphone int

o adresse varchar(200)

o longitude int

o latitude int

o photo varchar(500)

o descriptif varchar(500)

o statut varchar(50)

standing

o id (AI)

o libelle varchar(100)

chambre

o id (AI)

o nom varchar(50)

o photo varchar(500)

o descriptif varchar(5000)

o statue varchar(50)

o surface int

o type de lit varchar(100)

o jacuzzi bool

o balcon bool

o wifi bool

o mini bar bool

o coffre bool

o avec vue bool

catégorie

o id (AI)

o libelle varchar(100)

individu

o id (AI)

o nom varchar(100)

o prenom varchar(100)

o login varchar(100)

o pwd warchar(500)

profil

o id (AI)

o libelle varchar(50)

client

o id (AI)

o nom varchar(100)

o prenom varchar(100)

o login varchar(100)

o pwd varchar(500)

réservation

o id (AI)

o date\_création date

o date\_maj date

o date\_début\_séjour datetime

o date\_fin\_séjour datetime

o commande varchar(500)

o prix\_total int

o état varchar(50)

service

o id (AI)

o libelle varchar(100)

</entité>

<association>

réaliser

o client 1,n

o réservation 0,1

lier

o hotel 0,n

o réservation 1,1

prendre

o chambre 1,n

o réservation 1,1

donner\_accès (quantité)

o réservation 1,n

o service 0,n

être

o client 1,1

o profil 1,N

gérer:

o individu: 1,1

o hotel: 1,n

proposer (tarif à l'unité)

o service 1,n

o hotel 0,n

avoir

o individu 1,1

o profil 1,n

appartenir

o chambre 1,1

o hotel 10,n

standing

ç stending 1,n

o hotel 1,1

catégoriser

o catégorie 1,n

o chambre 1,1

tarifer (tarif)

o standing 1,n

o catégorie 1,n

</association>

<MCD>

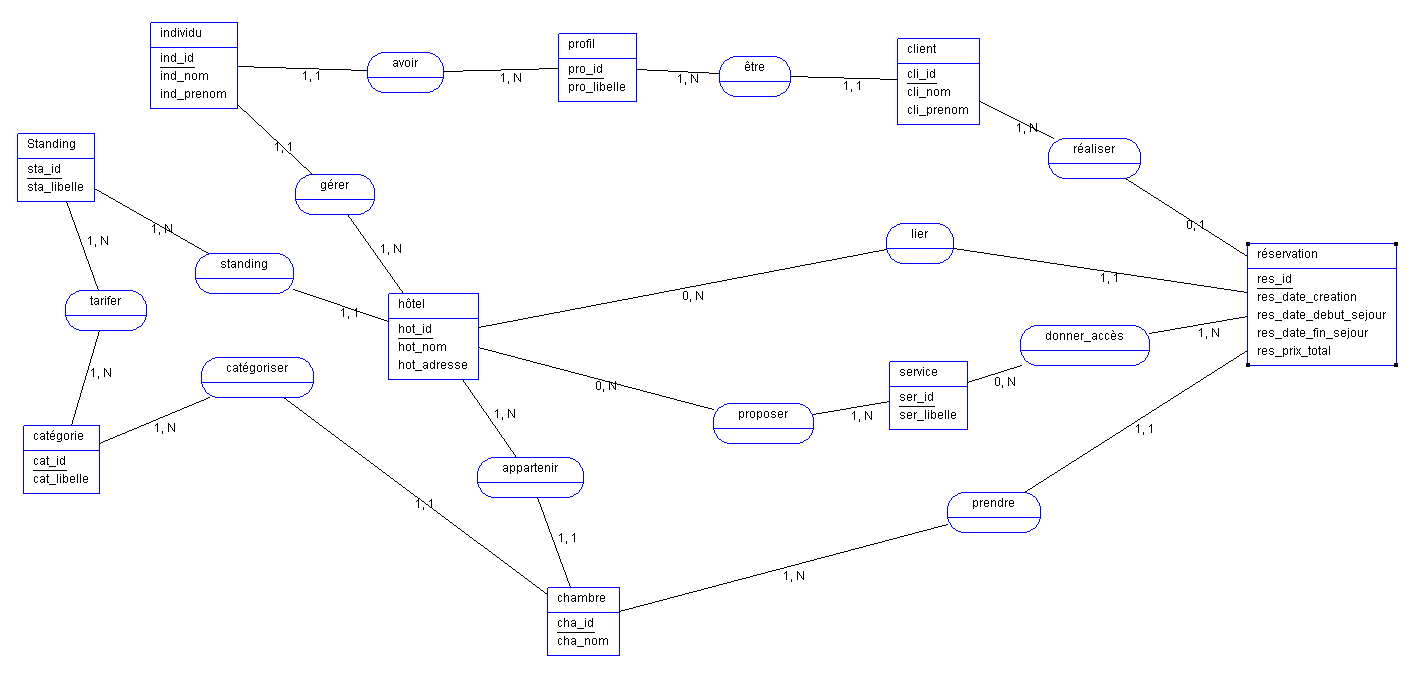


Figure MCD graphique

## Le modèle logique de données (modèle relationnel)

<MLD>

hotel

o id (PK)

o hot\_nom varchar(50)

o hot\_tel int

o hot\_adresse varchar(200)

o hot\_longitude int

o hot\_latitude int

o hot\_photo varchar(500)

o hot\_descriptif varchar(500)

o hot\_statut varchar(50)

o hot\_standing (FK)

standing

o sta\_id (PK)

o sta\_libelle varchar(100)

chambre

o cha\_id (PK)

o cha\_nom varchar(50)

o cha\_photo varchar(500)

o cha\_descriptif varchar(5000)

o cha\_statue varchar(50)

o cha\_surface int

o cha\_type\_lit varchar(100)

o cha\_jacuzzi bool

o cha\_balcon bool

o cha\_wifi bool

o cha\_mini bar bool

o cha\_coffre bool

o cha\_avec\_vue bool

o cha\_categorie (FK)

o cha\_hotel (FK)

catégorie

o cat\_id (PK)

o cat\_libelle varchar(100)

individu

o ind\_id (PK)

o ind\_nom varchar(100)

o ind\_prenom varchar(100)

o ind\_login varchar(200)

o ind\_pwd warchar(500)

o ind\_profil (FK)o

o ind\_hotel (FK)

profil

o pro\_id (PK)

o pro\_libelle varchar(50)

client

o cli\_id (PK)

o cli\_nom varchar(100)

o cli\_prenom varchar(100)

o cli\_login varchar(200)

o cli\_pwd varchar(500)

o cli\_profil (FK)

réservation

o res\_id (PK)

o res\_date\_creation date

o res\_date\_maj date

o res\_date\_début\_séjour datetime

o res\_date\_fin\_séjour datetime

o res\_commande varchar(500)

o res\_prix\_total int

o res\_état varchar(50)

o res\_client (FK)

o res\_hotel (FK)

o res\_chambre (FK)

service

o ser\_id (PK)

o ser\_libelle varchar(100)

donner\_acces

o don\_id (PK)

o don\_réservation (FK)

o don\_service (FK)

o don\_quantité int

Proposer\_service (prestation)

o pre\_id (PK)

o pre\_service (FK)

o pre\_hotel (FK)

o pre\_tarif int

tarifer

o tar\_id (PK)

o tar\_standing (FK)

o tar\_catégorie (FK)

o tar\_prix int

</MLD>

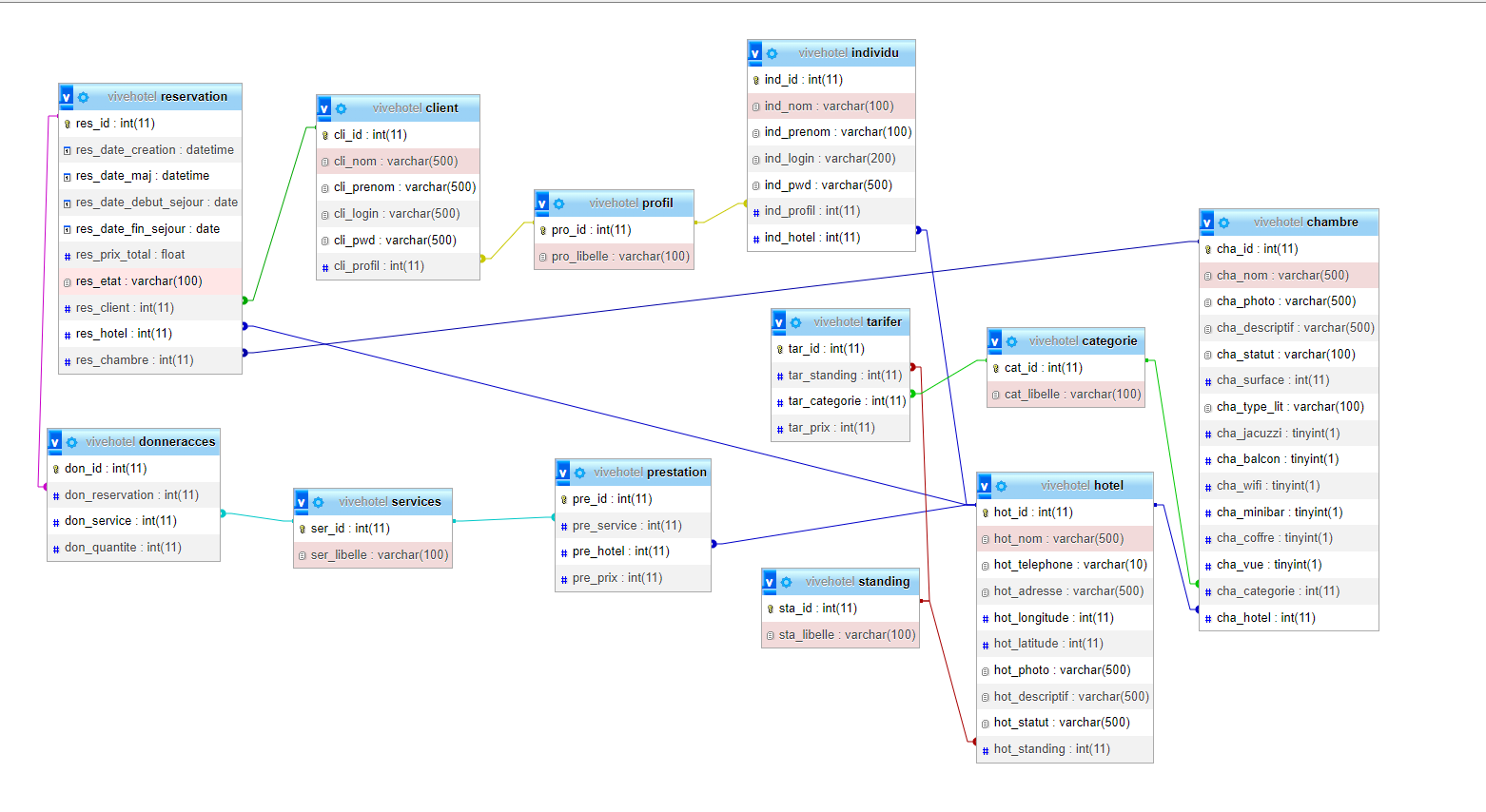


Figure MLD graphique

Pour le script voir “script\_vivehotel\_vincent.sql” dans « document ».

Description des dossiers et fichiers du projet :

## Dossier « document »

Les dossiers « document » contient l’ensemble des fichiers de documentation du projet. Il s’agit principalement de fichiers Word et PDF qui ne sont pas nécessaires au fonctionnement du projet mais nécessaires à la documentation de celui-ci, mais également le fichier du script de création de la base de donnée « script\_vivehotel\_vincent.sql ».

## Dossier « \_dataset »

Le fichier « config.php » du dossier « config » est la colonne vertébrale du projet. Il fait appel à l’ensemble des fichiers nécessaires au fonctionnement du site, incluant classes mères et fonctions. Ce fichier contient également **de nombreuses constantes, dont celles de connexion** **à la base de données.**

Avant de se pencher sur le dataset de la base de données, il convient de modifier les constantes de connexion afin que le projet soit configuré au niveau du SQL.

Le dossier « \_dataset » contient le fichier de génération de données (« index.php ») et ses dépendances qui sont des générateurs de chaque table.

Le sous-dossier « dataset » de « \_dataset » contient un ensemble de fichiers PHP. Chaque fichier PHP de ce dossier contient le script qui génère les enregistrements d’une table spécifique de la base de données « Vivehotel ». Les fichiers ne sont pas indépendants les uns des autres et devront être appelés dans un ordre logique, dans le cadre de la génération des données de la base « Vivehotel ».

* Le fichier « constants.php » du sous-dossier « datasets » est un fichier de constantes. Il sert à paramétrer la génération des données.
* Le fichier « index.php » à la racine va appeler et faire interpréter tous les fichiers nécessaires à la génération du dataset, dans un ordre logique. Le lancement du fichier PHP « index.php » équivaut à faire un « reset » de la base de données « Vivehotel ». Si la base de données n’a pas été créée au préalable, ce fichier « index » va créer la base de données nécessaire avant de lancer les datasets.

Il est important à noter que **le dossier « \_dataset » doit être supprimé en production** une fois utilisé, pour des raisons de sécurité.

## 

## Dossier « framework »

Le dossier Framework contient les fichiers de base du projet. Ces fichiers sont soit des classes mères, soit des fonctions qui permettent de faciliter l’écriture de certaines instructions redondantes.

### Ctr\_controleur.class.php

|  |  |
| --- | --- |
| FICHIER | Ctr\_controleur.class.php |
| DESCRIPTION | Ce fichier contient la classe mère contrôleur de l’application. |
| TYPE | Fichier séquentiel formaté |
| STRUCTURE | Ctr\_controleur : classe |

### fonction.php

|  |  |
| --- | --- |
| FICHIER | fonction.php |
| DESCRIPTION | Ce fichier contient l’ensemble des fonctions de l’application |
| TYPE | Fichier séquentiel formaté |
| STRUCTURE | Fonctions :   * AfficheTableHTML * CheckAuth * CheckAllow * Debug * FormRecherche * Hlien * monAutoLoad * monExceptionHandler * Mhe * Mres * Tableau2D * matriceSqlCD |

### I\_crud.class.php

Ce fichier n’est pas un fichier de classe PHP. Il contient une interface qui permet de forcer une programmation par contrat de toutes les classes héritées de la classe « Table ».

|  |  |
| --- | --- |
| FICHIER | I\_crud.class.php |
| DESCRIPTION | Ce fichier contient une interface qui impose la programmation par contrat de méthodes qui permettent de faire des opérations du CRUD. |
| TYPE | fichier séquentiel formaté |
| STRUCTURE | I\_CRUD : interface |

### Table.class.php

Souvenons-nous que « DRY » signifie « don’t repeat yourself ».

Lorsqu’un utilisateur se rend sur le site, l’application va procéder à la logique MVC. Grossièrement, l’application va aller chercher la donnée adéquate avant de la traiter et de l’afficher à l’utilisateur.

Le nombre de requêtes différentes peut être important. La solution naturelle serait de créer une fonction par requête mais cela n’est valable que dans le cadre procédural.

En programmation orientée objet, nous pouvons encapsuler les méthodes de modèles dans des classes. Pour chaque table de la base de données, la classe éponyme désignera l’ensemble des modèles de celle-ci.

La classe mère « Table » est une classe d’instruction de requêtes. Il s’agit d’une classe mère qui contient des méthodes de base du CRUD. « Table » implémente aussi l’interface « I\_CRUD », l’engageant « juridiquement » à spécifier des méthodes exécutant des modèles du CRUD.

Chaque classe qui interroge une base de données doit avoir une connexion pour fonctionner. La relation entre une connexion PDO et la classe « Table » est le verbe « avoir ». La classe « Table » contient donc une connexion de type PDO qui sera fournie par le fichier de configuration où nous avons bien spécifié les constantes de connexion.

Chaque classe contenant des modèles de requêtes d’une table spécifique est une classe d’instruction de requêtes sur la base de données. Le lien entre chacune de ces classes et la classe « Table » contient le verbe être. Ainsi, chaque classe de requêtes d’une table hérite de la classe « Table ».

Par exemple, « Réservation » et « Connexion » héritent de « Table ».

Nous décidons alors de factoriser des modèles classiques (create, read, update, delete entre eutres) dans une classe « mère » que nous avons nommé « Table » dans ce Framework.

Par exemple, la classe « Réservation » héritera de la classe « Table ». Instancions un objet de type réservation. Une méthode de l’objet créé va retourner tous les enregistrements de la table « Réservation », un autre va permettre de supprimer un enregistrement de la table « Réservation » ayant un identifiant précis.

Une requête de type SELECT renvoie des enregistrements alors qu’une requête DELETE ne renvoie rien (sauf en cas d’erreur). Les méthodes d’un objet Réservation va donc retourner un tableau de données ou alors un booléen en fonction de la requête demandée.

## 

## Fichiers publics pour l’utilisateur dans le dossier « www »

Par mesure de sécurité, l’utilisateur n’a pas le droit d’accéder aux fichiers PHP et SQL du site web. Ce dossier contient l’ensemble des fichiers au format Javascript, HTML et CSS qui sont les seules pages que l’utilisateur pourra consulter. Le seul fichier PHP accessible par l’utilisateur sera le fichier « index.php » en racine de ce dossier.

### Fichier « index.php »

|  |  |
| --- | --- |
| FICHIER | Index.php |
| DESCRIPTION | **Contrôleur principal de l’application**. Ce programme PHP est l’unique page web sur laquelle l’utilisateur navigue. Si l’utilisateur va sur la page « index.php » sans avoir indiqué de paramètre GET, la page principale du site lui sera retournée.  Lorsque l’utilisateur ajoute des paramètres dans l’appel de la page « index » (en cliquant sur des liens du site), le serveur lui retournera des pages différentes de la page d’accueil.  La page renvoyée à l’utilisateur dépendra des paramètres « a » (comme action) et « m » (comme module) envoyés avec le protocole « GET ». |
| TYPE | fichier séquentiel formaté |
| STRUCTURE | * (str) $\_GET[‘a’] : action de la page passé en paramètre * (str) $\_GET[‘m’] : module de la page passé en paramètre * (str) $module : nom de la classe contrôleur à charger * Instanciation d’un objet d’une classe enfant de Ctr\_Controleur. |

|  |  |
| --- | --- |
| FICHIER | \_js/index.js |
| DESCRIPTION | Il y a deux utilités à ce programme   * Supposons que nous avons désactivé le Javascript de notre navigateur. Si un administrateur du CRUD clique sur le bouton « supprimer » d’un enregistrement, l’enregistrement s’efface.   Pour éviter que cet accident ne se produise, nous allons créer une confirmation au moment du clic sur un bouton de suppression de données.  Cette protection se fera par un écouteur d’événement qui va lancer une fonction. Cette fonction va bloquer la redirection par défaut du lien puis demandera d’elle-même à l’utilisateur s’il confirme la suppression.   * L’action « index » du contrôleur « tarif » contient le tableau du prix de chaque chambre en fonction de la catégorie de chambre et d’hôtel de la chambre.   Lorsque l’utilisateur modifie un champ « td » (éditable) du tableau des tarifs, l’écouteur d’événements va constater cette modification.  L’ « event listener » lancera une méthode qui va envoyer les données utilisateur dans l’action « ajax » de « tarifer ». Cette page va faire la modification du tarif dans la base de données, sans que l’utilisateur n’ait eu à appuyer sur un bouton « Submit » |
| TYPE | Fichier javascript |
| STRUCTURE |  |

# Accès aux pages du site par un utilisateur

Supposons que le site soit hébergé en local. Si un utilisateur du site tape le lien « localhost/index.php?m=reservation&a=index », une page contenant la liste des réservations lui sera affiché. Notons que seul les administrateurs peuvent accéder à ce lien précis.

Pour pouvoir charger une page particulière, un système d’architecture MVC a été mis en place.

Lorsque l’on demande la page index.php avec les paramètres « GET » a=monaction et m=momodule, le site va instancier le contrôleur Ctr\_monmodule tout en chargeant sa méthode « monaction » qui représente la logique de la page chargée.

Ce contrôleur va ensuite charger le fichier PHP correspondant au gabarit de cette action par un appel de « require ». Le gabarit de la page peut donc varier en fonction du type de page (utilisateur, administrateur)

Ce gabarit va afficher la page avec le contenu principal et la vue. La vue chargée aura pour chemin « application/module/{nomodule}/vue\_{monmodule}\_{monaction}.php ».

Dans la suite de cette section, nous allons lister toutes les pages du site, triés par contrôleurs. Nous allons d’abord faire une section sur la liste des pages des administrateurs.

## Les pages du CRUD Administrateur :

### Authentification :

#### Connexion du personnel

Les membres du personnel peuvent être soit des « téléconseillers » (SRC), soit des « personnels hôtelier » soit des « administrateurs gérant ».

Le script « dataset » a généré une table « individu » avec une génération automatique du profil de chaque membre du personnel. Pour un membre du personnel : son nom, son rôle et son hôtel sont connus. Chaque membre a également un identifiant et un mot de passe déjà attribués.

Le super administrateur (les administrateurs gérant de chaque hôtel), ont accès à l’ensemble des données générées avec le mot de passe de chaque membre du personnel. C’est à eux d’envoyer le mot de passe d’un membre du personnel par écrit ou à l’oral. Ils peuvent également créer de nouveaux personnel hôtelier depuis la page « individu ».

L’administrateur peut se connecter à son profil par la connexion du personnel. Il existe un compte administrateur par défaut que l’on a généré avec les identifiant suivant.

|  |  |
| --- | --- |
| Login | Mariokart@vivehotel.fr |
| Mot de passe | Mariokart |

Nous avons généré l’ensemble des pages et fonctionnalité à l’ensemble des rôles. Pour les SRC et les personnels hôtelier, ils ont le profil modérateur et peuvent également se connecter par ce lien.

(photo Connexion personnel)

Dans les parties suivantes, nous allons décrire les pages du CRUD « administrateur » dûment programmé. Pour que le lecteur puisse accéder à ces pages, il devra se connecter.

### Gestion des chambres

#### Liste des chambres

Cette page contient la liste de toutes les chambres de la compagnie « ViveHôtel ».

(photo chambre admin)

La page contient un tableau. Chaque ligne du tableau contient plusieurs types d’éléments :

* Les données de la chambre (numéro, statut, surface, etc.)
* Bouton « modifier » (menant la page de modification de la chambre)
* Bouton « supprimer » (supprime une chambre de **manière logique**)

Un formulaire de recherche a été programmé en haut de page pour que l’administrateur puisse rechercher une chambre précise. Pour l’utiliser, il faut suivre les étapes suivantes

* Compléter le formulaire avec vos critères de recherche
* Cliquer sur rechercher

Par exemple, si l’administrateur tape « hôtel » dans le champ hôtel et sélectionne une catégorie et clique sur rechercher, il enverra une requête à cette page. En retour, cette page affichera l’ensemble des chambres de l’hôtel sélectionné ayant cette catégorie.

#### Modification des chambres

Si l’administrateur décide de modifier une, il accèdera alors à la page des modifications des chambres.

Cette page permet de modifier toute les informations relatives à la chambre à l’exception de son hôtel d’affectation.

(photo modifier chambre)

#### Création d’une chambre

S’il clique sur le bouton nouvelle chambre, on se retrouve sur un formulaire d’édition de chambre vide permettant d’un créer une.

(photo créer chambre)

### Gestion des clients

#### Liste des clients de « Vivehotel »

(photo client)

Cette page contient la liste de tous les clients du site « vivehotel ». Le contenu principal est un tableau dont chaque ligne contient de multiples données et liens concernant le client dont :

* Son nom
* Son login
* Un bouton « Supprimer » qui permet de supprimer un enregistrement d’un client

#### Suppression d’un client

SI l’administrateur clique sur le bouton supprimer, une suppression logique du client se fera dans la base de donnée et l’administrateur sera redirigé vers la page d’administration.

### Pages des hôtels

#### Liste de tous les hôtels

(photo hotel admin)

L’administrateur peut gérer la liste des hôtels du site. Lorsqu’il clique sur le lien « Hôtel » du menu de navigation, il aura à l’écran un tableau de la liste de tous les hôtels. Chaque ligne correspond à une entrée d’hôtel dans la table « Hôtel » de la base de données.

Ce tableau permet de voir les informations principales sur l’hôtel (statut, nom, photo, description….) Les boutons supplémentaires permettent d’accéder à des pages différentes :

* Statistiques : permet d’accéder à l’ensemble des statistiques de l’hôtel
* Nouveau hôtel : permet d’ajouter un hôtel au groupe
* Une liste des services associés à l’hôtel
* Modifier : permet de modifier les informations de l’hôtel
* Supprimer : permet de supprimer un hôtel de la base de données

#### Statistique des hôtels

Lorsque l’administrateur clique sur le bouton « statistique » d’un hôtel, il accède à cette page.

(photo statistique)

Cette page permet d’avoir les statistiques de son hôtel d’affectation mais aussi aux statistiques du groupe tout entier comme le chiffre d’affaire annuel.

Lorsque l’administrateur a fini de lire les données sur l’hôtel, il clique sur le bouton « retour aux hôtels» pour revenir à la liste des hôtels de la réservation.

#### Modification d’un hôtel

(photo modification hotel)

#### Création d’un hôtel

(photo création hotel)

En cliquant sur ce bouton l’utilisateur à accès à un formulaire d’édition d’un nouvel hôtel

Une fois complété l’hôtel sera ajouté à la base de donnée.

Lorsque l’utilisateur clique sur « hôtel » « modifier » dans la ligne d’un hôtel listé dans l’index des hôtels, il accède à une page de modification. Il pourra alors modifier l’ensemble des données de l’hôtel.

En bas de page, un bouton « Envoyer » permet de confirmer l’envoi du formulaire. L’administrateur qui a rempli le formulaire sera redirigé vers la page de la liste des hôtels.

#### Suppression d’un hôtel

Sur chaque ligne d’un enregistrement de la liste des hôtels, un bouton de suppression permet de supprimer un hôtel du projet « Vivehotel » de façon logique.

L’utilisateur qui a supprimé hôtel se verra redirigé vers la page de la liste des hôtels.

### Pages des individus

#### Page d’index du personnel (individu)

(photo individu)

La page d’index du personnel contient la liste des membres du personnel de tous les hôtels. Pour chaque ligne, des informations diverses sont indiqués sur un membre du personnel.

Chaque membre du personnel est identifié par son nom, son login, son hôtel et son profil. Les boutons supplémentaires permettent d’accéder respectivement aux pages de modification d’un membre du personnel et le supprimer.

#### Page de modification du personnel

Lorsque l’on clique sur le bouton « Modifier » de la ligne d’un membre du personnel, nous avons accès à un formulaire de modification dudit membre.

Ce formulaire permet de modifier le profil de l’individu, à le faire passer de modérateur à administrateur.

(photo édit individu)

### Pages des réservations

#### Page d’index des réservations

La page d’index des réservations contient les l’ensemble des réservation de l’année courante.

(photo réservation)

Chaque ligne représente un enregistrement de la table « réservation ». Il indique queqlques informations relatifs aux réservations.

En plus des données qui décrivent l’enregistrement de la réservation, il y a 4 boutons :

* Nouvelle réservation : qui permet l’édition d’une nouvelle réservation
* Gestion qui permet l’édition des services lié à la réservation
* Modifier qui liste les modifications de la réservation
* Supprimer qui supprime l’enregistrement affiché sur la même ligne

#### Page de Gestion des services d’une réservation

Lorsque l’utilisateur clique sur le bouton « Gestion » d’une réservation, il accède à une page qui est divisé en deux :

* En haut la liste des services lié à la réservation avec leur quantité
* En bas la liste des services que le client n’a pas pris et qu’il peut prendre

Dans le premier formulaire si l’on modifie la quantité et que l’on clique sur modifier la quantité est directement modifier dans la réservation. En revanche si l’on clique sur « supprimer » le service sera dissocier de la réservation.

Dans le second formulaire il est possible d’associer un service directement à une réservation en spécifiant la quantité et en cliquant sur « Ajouter ». Cela se fait instantanément.

En cliquant sur « retour aux réservation » on est rediriger ver la page des réservations.

#### Page d’édition des réservations

(photo res édit)

Lorsque l’utilisateur clique sur le bouton « éditer » d’une réservation, il accède à une page qui lui permet de modifier une réservation.

Le formulaire permet de modifier l’ensemble des données de la réservation. L’administrateur va remplir à sa guise le formulaire pour ensuite l’envoyer.

Lorsque le formulaire est envoyé, les données modifiées seront vérifiées par l’action de sauvegarde des réservations.

L’action de sauvegarde de réservations va vérifier que le contenu envoyé par l’utilisateur est cohérent. De multiples critères vont devoir être validés sur le formulaire pour que le site accepte les modifications de la réservation :

* Il ne faut pas que la date de début soit après la date de fin
* Il est impossible de prendre une réservation pour une chambre qui est déjà occupée pendant la période de réservation

Lorsque l’administrateur clique sur « Envoyer » en bas de page, le formulaire est vérifié.

Il y a alors deux scénarios qui peuvent se produire

* Le formulaire est incorrect et il n’est pas envoyé sur le serveur. Le formulaire est alors réinitialisé avec un message d’erreur
* Le formulaire est correct. Le script de sauvegarde de réservations va alors modifier la réservation. La page sera rechargée en affichant les nouvelles modifications

Page de suppression des réservations

Lorsque l’utilisateur clique sur le bouton « supprimer » d’une réservation, la réservation sera supprimée de façon « logique » (?). L’utilisateur sera renvoyé sur la page de la liste des réservations.

### Pages des services

#### Page d’index des services

(photo service)

La page d’index des services contient la liste de tous les services que peuvent proposer vivehotel. Si nous cliquons sur le bouton modifier, nous accèderons au formulaire d’édition. Si nous cliquons sur le bouton supprimer, nous supprimons le service qui est à la même ligne.

#### Page d’édition des services

Lorsque l’utilisateur (administrateur) de la page d’index de la liste des services, clique sur le bouton « éditer », il pourra éditer un service.

Un service de la table « service » est caractérisé par son nom et une clé primaire. Un service peut être proposé dans plusieurs hôtels de la compagnie « Vivehotel ». Un enregistrement de la table service ne porte pas des propriétés issues d’une association à une hôtel (prix, quantité).

Il n’est alors possible que de modifier le nom d’un service dans la page d’édition d’un service. Le formulaire est très simple, permettant à l’administrateur de corriger une faute d’orthographe sur le nom d’un service, pour tous les hôtels.

Lorsque l’administrateur finir d’éditer un service, il sera alors redirigé sur la liste des services que peuvent proposer chaque hôtel de la compagnie « Vivehôtel ».

#### Page de suppression des services

Si nous cliquons sur le bouton supprimer d’un service dans la page des « services » alors l’administrateur sera redirigé sur cette page. Cette page va supprimer le service de la base de données et rediriger l’utilisateur vers la liste des services de la chaîne « Vivehôtel ».

### Pages de tarifs

#### Page de modification des tarifs

Cette page affiche l’ensemble des tarifs des chambres de l’hôtel. C’est un tableau à double entrée. Sur l’axe des abscisses, il y a l’ensemble des catégories d’hôtel. Sur l’axe des ordonnées, il y a l’ensemble des chambres de l’hôtel.

(photo tarif)

L’administrateur peut cliquer sur un des éléments du tarif qui deviendra alors éditable. Lorsqu’il modifiera l’une des cases, un événement Javascript se déclenchera. Cet événement récupérera le nombre entré par l’utilisateur et fera une modification sur le serveur via une requête AJAX.

Si l’utilisateur tape un nombre entier, la modification va réussir. Cependant, si l’utilisateur tape une chaîne non-numérique alors le nouveau prix ne se répercutera pas sur l’enregistrement concerné en base de données.

#### Page de l’espace personnel :

(photo espace personnel)

Chaque utilisateur lors de sa connexion arrive sur sa page d’espace personnel, sur laquelle est relater l’ensemble des réservations en cour à la date du jour dans son hôtel.

C’est tout pour les actions réalisables pour un administrateur et un super administrateur.

## Les Pages du modérateur et personnel :

Pour un modérateur personnel du groupe hôtelier ils ont les mêmes privilèges et accès quasiment aux mêmes pages qu’un admin à l’exception près qu’il ne peut pas réaliser de suppression, ni réaliser de création d’entité pour le groupe, à l’exception des réservations sur lesquels il à tous contrôle sauf la suppression.

Les illustrations de ses pages sont en annexe.

## Les Pages des utilisateurs clients :

Pour les clients il on accès au pages suivantes :

* Hôtel : avec une présentation des différents hôtels du groupe
* Chambre : avec une présentation des chambres du groupe
* Réservation avec l’ensemble de ses réservation passées et actives.
* Son espace personnel.

# Structure physique de l'application

## Architecture MVC

Le projet est organisé selon l’architecture MVC, comme vu précédemment.

* Le fichier « Table.class.php » du dossier « framework » est un programme PHP contenant classe qui, à partir d’une composition avec une connexion PDO, programme l’ensemble des méthodes de base de traitement des données. Cette classe est une classe mère pour l’ensemble des modèles du projet. Elle permet en outre de : créer, lire, modifier et supprimer des enregistrements de la base de données.
* Les modèles de données sont disponibles dans le sous-dossier « application/table ». Chaque classe du fichier PHP de modèles d’une table hérite de la classe « Table » contenu dans le fichier « Table.class.php » du dossier de framework. Chaque classe enfant de modèles surcharge son constructeur afin de spécifier la table sur laquelle la classe travaille.
* Chaque dossier du sous-dossier « application/module » représente un contrôleur de l’application. Soit {contrôleur} le nom d’un sous-dossier du dossier « application/module ». Le dossier {contrôleur} contient alors un fichier qui se nomme Ctr\_{contrôoleur}.php. Ce fichier contient une classe « contrôleur » dont chacune de ses méthode est une action de ce dernier.
* Chaque sous-dossier de « application/module » contient aussi des fichiers PHP nommés « vue\_{moncontrôleur}\_{monaction}.php . Ce sont des fichiers qui représentent la vue de l’action « a\_{monaction} » (en tant que méthode) de la classe contrôleur « Ctr\_{moncontrôleur} ».

|  |
| --- |
|  |
| Arborescence du projet « Vivehotel » |

# Fonctions et procédures

Un Framework MVC va récupérer de la donnée en serveur, la traiter et l’afficher. Le dossier « Framework » correspond surtout à des comportements associés à la première lettre de l’abréviation « MVC » (modèle).

Les fichiers de ce dossier vont configurer la base de données, commander l’appel par « autoload » des classes de chaque table de la base de données.

Certaines fonctionnalités sont utilisées de très nombreuses fois sur le site (échapper du texte des failles Javascript par exemple). Il est alors surchargé de faire appel à des objets de classe pour les faire.

Nous définissons alors un fichier de fonctions dans le dossier « framework ». C’est le fichier « functions.php ». Nous allons brièvement décrire ces fonctions, en mettant l’accent sur celles qui sont les plus complexes au premier abord.

|  |  |
| --- | --- |
| FONCTION/METHODE | checkAuth |
| Description | Vérifie si l’utilisateur est connecté ou non. Le redirige vers la page d’accueil si il ne l’est pas. |
| Auteur | Auteur du framework PDO 3.0. |
| Date et version | 3.0 – 23/02/2023 |
| Paramètres d’entrée | Aucun |
| Paramètres de sortie | Aucun |
| Fichiers accédés | Aucun |
| Données manipulées | \* $\_SESSION["cli\_id"] : identifiant du client, si connecté \* $\_SESSION['per\_role'] : rôle d’un membre du personnel, si connecté \* $\_SESSION[‘message’][] : stocke temporaire des messages d’erreurs |
| Accès aux données | Lecture des variables de session |
| Retour(si fonction) | Pas de retour |

|  |  |
| --- | --- |
| FONCTION/METHODE | checkAllow |
| Description | Après avoir vérifié si un utilisateur est connecté (checkAuth), vérifie si l’utilisateur a le profil demandé en paramètres.  Si le visiteur de la page n’a pas le profil demandé, il est alors redirigé sur une autre page. |
| Auteur | Auteur du framework 3.0 |
| Date et version | 3.0 – 23/02/2023 |
| Paramètres d’entrée | $profil, chaîne de caractères |
| Paramètres de sortie | Aucun |
| Fichiers accédés | Aucun |
| Données manipulées | Clés de la superglobale de session :  - « per\_role » : profil d’un individu du personnel, si connecté - « cli\_id » : identifiant d’un client, si connecté  - «message » : tableau d’erreurs temporaire, affichés puis supprimés de la variable de session |
| Accès aux données |  |
| Retour(si fonction) | Pas de retour |

|  |  |
| --- | --- |
| FONCTION/METHODE | Mres |
| Description | Lorsque nous envoyons du texte sur un serveur web, il est souvent nécessaire de protéger la chaîne de caractère sous peine de recevoir des données non-protégées.  La fonction procédurale « mysqli\_real\_escape\_string » existe par défaut dans PHP 8.2 et permet d’échapper du texte contre les injections SQL. Cette fonction nécessite une connexion de type « Mysqli ». Notre projet se base sur une connexion de type « Pdo ». La fonction mysqli est donc inutilisable dans le contexte de notre projet.  Les objets de classe PDO ont une méthode statique qui permet de protéger une chaîne de caractère. La fonction en question va prendre en paramètre un texte à protéger et sans surprise, retourner un texte protégé au format « string ».  La classe mère « Table » est une classe qui permet d’instruire des requêtes dans la base de données. Elle contient un objet statique qui est la connexion à la base de données. Le nom de la variable est « $link ».  Protéger le texte des injections SQL avec une méthode de l’objet PDO est donc une solution viable car cette solution existe dans l’objet de connexion d’une classe du projet.  La fonction « mres » va appeler l’objet statique PDO de la classe Table. A partir de cet objet, elle va lancer la méthode « quote » sur le texte passé en paramètre de « mres »  La méthode statique de l’attribut $link va retourner un texte qui sera immédiatement retournée par la fonction « mres ». |
| Auteur | Auteur du Framework PDO 3.0 |
| Date et version | 24/02/2023 – 3.0 |
| Paramètres d’entrée | $s : chaîne de caractères à protéger des injections SQL |
| Paramètres de sortie | Aucun |
| Fichiers accédés | Aucun |
| Données manipulées | Objet de connexion PDO de la classe Table |
| Accès aux données | Aucun |
| Retour(si fonction) | Chaîne de caractères protégé des injections SQL |

|  |  |
| --- | --- |
| FONCTION/METHODE | tableau2D |
| Description | Crée un tableau à double entrées (array d’arrays). Cette matrice a pour dimension $dimensionX\*$dimensionY.  Le paramètre $defaultValue donnera la valeur de chaque cette matrice. |
| Auteur | EL AMRANI Bilel, OUANYOU Arsène, VONIN Cédric |
| Date et version | 19/04/2023 – 1.0 |
| Paramètres d’entrée | $dimensionX (entier) $dimensionY (entier) $defaultValue = 'X' (string) |
| Paramètres de sortie | Aucun |
| Fichiers accédés | Aucun |
| Données manipulées | Table $mytable (lecture/écriture) |
| Accès aux données | passés en paramètres |
| Retour(si fonction) | $mytable (tableau) |

|  |  |
| --- | --- |
| FONCTION/METHODE | matriceSqlCD |
| Description | Cette fonction crée un tableau croisé dynamique à partir d’enregistrements de requêtes SQL.  Le premier paramètre de cette fonction est un tableau qui a pour clés X1 et X2. Chaque clé est associée à une valeur qui est un entier naturel strictement positif.  La fonction « matriceSqlCD » appelle tout d’abord la fonction tableau2D pour créer un tableau T de dimension D1\*D2 où :  \* D1 = $dimensionsTab['X1']  \* D2 = $dimensionsTab['X2']  A partir du tableau D1\*D2 créé par la méthode tableau2D, cette fonction crée un tableau croisé dynamique avec pour axes :  - X1 : la clé en axe des X  - X2 : la clé en axes des Y  - Y : la valeur dans la case de coordonnées (X1,X2)  Le nom des axes X1,X2 est donné dans le deuxième paramètre de la fonction ($nomAxe)  Le troisième paramètre est un array de résultats SQL à index numériques. Chaque clé est associée à un array contenant un tableau associatif d’un enregistrement d’une requête SQL.  Chaque enregistrement d'un résultat SQL doit être un array de la forme (l’ordre des clés n’importe pas) :  [cle1=>val1, cle2=>val2, ... X1=>valX1, X2=>valX2, Y=>valY]  Pour chaque enregistrement itéré, la fonction va remplir une case du tableau T de coordonnées (valX1,valX2) avec la valeur « valY ». |
| Auteur | EL AMRANI Bilel, OUANYOU Arsène, VONIN Cédric |
| Date et version | 19/04/2023 – 1.0 |
| Paramètres d’entrée | * $dimensionsTab (array) - $dimensionsTab[‘X1’] (int (> 0))  - $dimensionsTab[‘X2’] (int (> 0)) * $nomAxe (array) - $nomAxe[‘X1’] (string) - $nomAxe[‘X2’] (string) * $mysqlRecords (array) :  - $mysqlRecords[int] (array) |
| Paramètres de sortie | Aucun |
| Fichiers accédés | Aucun |
| Données manipulées | $tableauSqlCD (array) : tableau croisé dynamique des données demandées en paramètres. |
| Accès aux données | Lecture des paramètres. |
| Retour(si fonction) | $tableauSqlCD (array) |

|  |  |
| --- | --- |
| FONCTION/METHODE | <nom> |
| Description | <description> |
| Auteur | Nom des auteurs |
| Date et version | Version et date de dernière modification |
| Paramètres d’entrée | xxxx, entier,  xxxxx, chaîne de caractères  xxxx, objet X |
| Paramètres de sortie | xxxx, entier,  xxxxx, chaîne de caractères  xxxx, objet X |
| Fichiers accédés | <nom du fichier> (lecture/écriture) |
| Données manipulées | Table XXXX (lecture/écriture) |
| Accès aux données | Requête / Procédure stockée  (Nom et paramètres) |
| Retour(si fonction) | xxxx, entier |