|  |
| --- |
|  |
| Vivehotel – Document technique |
| Document de conception technique |

|  |
| --- |
| EL AMRANI Bilel, OUANYOU Dje Arsène, VONIN Cédric  11/04/2023 |

Table des matières

[Description de l’application 3](#_Toc132666973)

[Environnement technique 4](#_Toc132666974)

[Langages de programmation 4](#_Toc132666975)

[Système d’exploitation 4](#_Toc132666976)

[Framework 4](#_Toc132666977)

[Serveur d’application 4](#_Toc132666978)

[Base de données 4](#_Toc132666979)

[Description de la base de données 5](#_Toc132666980)

[Le modèle conceptuel de données (MCD) 7](#_Toc132666981)

[Textuel 7](#_Toc132666982)

[MCD graphique 9](#_Toc132666983)

[Le modèle logique de données (modèle relationnel) 10](#_Toc132666984)

[Textuel 10](#_Toc132666985)

[Graphique 12](#_Toc132666986)

[Description des fichiers 13](#_Toc132666987)

[Dossiers « annexes » et « document » 13](#_Toc132666988)

[Dossier « \_dataset » 13](#_Toc132666989)

[Sous-dossier config du dossier application 13](#_Toc132666990)

[Sous-dossier framework du dossier application 13](#_Toc132666991)

[Ctr\_controleur.class.php 13](#_Toc132666992)

[fonction.php 14](#_Toc132666993)

[I\_crud.class.php 14](#_Toc132666994)

[Table.class.php 14](#_Toc132666995)

[Fichiers publics pour l’utilisateur dans le dossier « www » 15](#_Toc132666996)

[Fichier « index.php » 15](#_Toc132666997)

[Description des pages 16](#_Toc132666998)

[Description des pages du CRUD 16](#_Toc132666999)

[Gestion des chambres 16](#_Toc132667000)

[Gestion des clients 19](#_Toc132667001)

[Pages des hôtels 21](#_Toc132667002)

[Pages du personnel 24](#_Toc132667003)

[Pages des réservations 25](#_Toc132667004)

[Pages des services 27](#_Toc132667005)

[Structure physique de l'application 28](#_Toc132667006)

[Architecture MVC 28](#_Toc132667007)

[Fonctions et procédures 29](#_Toc132667008)

# Description 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’affaires annuel.
* Production de diverses statistiques :
  + Chiffre d’affaires par hôtel, par département.
  + Nombre de réservations par durées.
  + …etc.

« 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

## Langages de programmation

* JavaScript ES6
* PHP 8.2
* SQL

## Système d’exploitation

* Windows 10

## Framework

* Bootstrap (v5.0.0-beta2)
* Framework PDO MVC BOOTSTRAP (3.0)

## Serveur d’application

* XAMPP Control Panel Version 3.3.0 et Apache 2.4.54

### Base de données

* PhpMyAdmin 5.2.0

# Description de la base de données

La base de données complète du site web « Vivehotel » possède 10 tables et 12 contraintes d’intégrité. La liste de tous les champs et contraintes d’intégrités sont disponibles dans le fichier « vivehotel.sql » du dossier « \_dataset ».

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 exécutée.

Supposons que le lecteur a 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.

Le programme principal qui va connecter le site à la base de données est dans le fichier « config.php » du sous-dossier « config ».

Pour établir une connexion à une base de données, il est nécessaire d’indiquer des paramètres multiples à une instruction de connexion. Un ensemble possible de paramètres est le suivant (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 »)

Le fichier « config » contient toutes les données, sous forme de constantes. Les valeurs par défaut de cette constante permettent à un utilisateur « root » de se connecter à la base de données « vivehotel ».

Nous allons maintenant nous intéresser à la façon dont nous nous connectons à la base de données.

La classe PDO est une classe de connexion à des bases 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. Ces paramètres seront les constantes que nous avons définies précédemment dans le programme de « config ».

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

La connexion sera, tout comme les constantes, définies dans le fichier de configuration. Cette connexion pourra être utilisé par tous les autres programmes du site qui feront appel à ce fichier « config.php ». La réécriture de ce fichier n’est cependant pas suffisant car nous n’avons pas créé de base de données dans notre SGBD.

Une fois que la connexion à la base de données a été établie, il sera important de donner du sens à ce que nous faisons. Nous allons voir comment nous pouvons générer (ou réinitialiser) de la donnée de test, avec un ensemble de programmes PHP.

Le dossier « \_dataset » contient un fichier nommé « index.php ». Ce script fait appel à un ensemble de fichiers qui vont :

* Créer une connexion (config.php)
* Supprimer la base de données de « vivehotel » si elle existait déjà
* Créer la base de données « vivehotel » si elle n’existe pas
* Créer et ajouter un ensemble d’enregistrements cohérents pour chaque table de cette base

Lancer le script « index.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.

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)

### Textuel

<MCD>  
 <entités>

\*Client  
- id (AI)  
- nom (varchar(500))  
- identifiant (varchar(500))  
- mot\_de\_passe (varchar(500))  
- email (varchar(500))

\* Personnel  
- id (AI)  
- nom (varchar(500))  
- idenifiant (varchar(500))  
- mot\_de\_passe (varchar(500))  
- email (varchar(500))

\* Hotel  
- id (AI)  
- statut (int)  
- nom (varchar(500))  
- adresse (varchar(500))  
- departement (int)  
- description (text)  
- longitude (float)  
- latitude (float)

\*Chambre  
- id (int)  
- numero (int)  
- statut (varchar(500))  
- surface (int)  
- type\_lit\_un (varchar(500))  
- type\_lit\_deux (varchar(500) or NULL)  
- description (text)  
- jacuzzi (bool)  
- balcon (bool)  
- wifi (bool)  
- minibar (bool)  
- coffre (bool)  
- vue (bool)  
  
\*Service  
- id (AI)  
- nom (varchar(500))

\*ChCategorie  
- id(AI)  
- categorie\_chambre (varchar(500))

\* HoCategorie

- id (AI)  
- categorie\_hotel (varchar(500))

\* Reservation  
- Id(AI)  
- date\_creation (datetime)  
- date\_debut (datetime)  
- date\_maj (datetime)  
- date\_fin (datetime)  
- etat (varchar(500))  
</entités>

<association>  
Prendre  
- Client 1,n  
- Réservation 1,1

Conclure  
- Reservation 1,1  
- Hotel 1,n

Preciser  
- Chambre 1,n  
- Reservation 1,1

Commander (quantité)  
- Service 1,n   
- Reservation 1,n

Travailler (role)  
- Personnel 0,1  
- Hotel 1,n

Proposer (prix)  
- Hotel 0,n  
- service 0,n

Tarifer (prix)  
- HoCategorie 1,n  
- ChCategorie 1,n

HoAppartenir  
- hotel 1,1  
- HoCategorie 1,n

ChAppartenir  
- Chambre 1,1  
- ChCategorie 1,n  
</association>

</MCD>

### MCD graphique



*MCD graphique* de la base de données « vivehotel »

## Le modèle logique de données (modèle relationnel)

### Textuel

<MLD>  
client  
 - cli\_id (AI)  
 - cli\_nom (varchar(500))  
 - cli\_identifiant (varchar(500))  
 - cli\_mdp (varchar(500))  
 - cli\_email (varchar(500))

personnel  
 - per\_id (AI)  
 - per\_nom (varchar(500))  
 - per\_idenifiant (varchar(500))  
 - per\_mdp (varchar(500))  
 - per\_email (varchar(500))  
 - per\_role (varchar(500))  
 - per\_hotel (FK)

hotel  
 - hot\_id (AI)  
 - hot\_statut (varchar(500))  
 - hot\_nom (varchar(500))  
 - hot\_adresse (varchar(500))  
 - hot\_departement (int)  
 - hot\_description (text)  
 - hot\_longitude (float)  
 - hot\_latitude (float)  
 - hot\_hocategorie (FK)

chambre  
 - cha\_id (AI)  
 - cha\_numero (int)  
 - cha\_statut (varchar(500))  
 - cha\_surface (int)  
 - cha\_typelit1 (varchar(500))  
 - cha\_typelit2 (varchar(500) ou « NULL »)  
 - cha\_description (text)  
 - cha\_jacuzzi (bool)  
 - cha\_balcon(bool)  
 - cha\_wifi (bool)  
 - cha\_minibar (bool)  
 - cha\_coffre (bool)  
 - cha\_vue (bool)  
 - cha\_chcategorie (FK)

service  
 - ser\_id (AI)  
 - ser\_nom (varchar(500))

chcategorie  
 - chc\_id (AI)  
 - chc\_categorie (varchar(500))

hocategorie  
 - hoc\_id (AI)  
 - hoc\_categorie (varchar(500))

reservation  
 - res\_id (AI)  
 - res\_datecreation (datetime)  
 - res\_datedebut (datetime)  
 - res\_datemaj (datetime)  
 - res\_datefin (datetime)  
 - res\_etat (varchar(500))  
 - res\_client (FK)  
 - res\_hotel (FK)  
 - res\_chambre (FK)

commander   
 - com\_id (AI)  
 - com\_quantite (int)  
 - com\_service (FK)  
 - com\_reservation (FK)

proposer  
 - pro\_id (AI)  
 - pro\_prix (float)  
 - pro\_hotel (FK)  
 - pro\_service (FK)

tarifer  
 - tar\_id (AI)  
 - tar\_prix (float)  
 - tar\_hocategorie (FK)  
 - tar\_chcategorie (FK)  
</MLD>

### Graphique



*Modèle logique de données graphique*

**Script SQL :** voir “vivehotel.sql” dans le dossier « \_dataset »

# Description des fichiers

## Dossiers « annexes » et « document »

Les dossiers « document » et « annexes » contiennent 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.

## 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 ou informatives**.

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

## Sous-dossier config du dossier application

## Sous-dossier framework du dossier application

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 ces requêtes 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 requêtes 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 » ou « 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. Cet objet instancié peut être utilisé pour lancer des méthodes de sélection, la création, la modification ou la suppression des données.

Chaque méthode de la classe « Réservation » va lancer une requête et retourner la donnée, généralement sous forme d’un array d’enregistrements (array).

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

Ce dossier contient l’ensemble des fichiers Javascript, HTML et CSS qui permettent de gérer la partie front-end. Un

### 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/script.php |
| DESCRIPTION | Ce fichier est au format PHP mais a un header « Javascript ».  Il y a deux utilités à ce programme   * Supposons que nous avons désactivé Javascript de notre navigateur. Si un administrateur du CRUD clique sur le bouton « supprimer » d’un enregistrement, l’enregistrement s’efface. Nous avons alors créé un écouteur d’événement qui va demander la confirmation de l’utilisateur avant d’effacer de la donnée * L’action « index » du contrôleur « tarif » contient le tableau du prix des chambres en fonction de leur catégorie et de la catégorie d’hôtel dans lequel elles font partie.   Lorsque l’utilisateur modifie un champ « td » (éditable) du tableau des tarifs, l’écouteur d’événement va constater cette modification. L’écouteur lancera une méthode qui va envoyer les données utilisateur dans l’action « ajax » de « tarifer » pour qu’elle soit instantanément enregistrée sur le serveur. |
| TYPE | Fichier javascript/PHP |
| STRUCTURE | nom : chaîne de caractères  prénom: chaîne de caractères  age : entier |

# Description des pages

Par exemple, si l’utilisateur tape le lien « adressesite/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. Ce gabarit va afficher la page avec le contenu principal et la vue. La vue est disponible dans le fichier « 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.

# Description des pages du CRUD

### Gestion des chambres

#### Liste des chambres

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

|  |
| --- |
|  |
| Capture d’écran de la liste des chambres |

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

* Les données de la chambre (numéro, statut, surface, etc.)
* Bouton « réservation » (menant à la liste des réservations de la chambre)
* 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

* Cliquer sur le bouton correspond au critère de recherche
* Taper le texte correspondant au critère de recherche

Par exemple, si l’administrateur tape « ma ville » dans l’input de la liste des chambres et sélectionne le bouton « Description » et clique sur envoyer, il enverra une requête à cette page. En retour, cette page affichera l’ensemble des chambres dont la description contient le terme « ma ville ».

#### Réservation des chambres

Si l’administrateur connecté clique sur le bouton « Réservation » dans la liste des chambres, il va pouvoir accéder à l’ensemble des réservations de la chambre

Cette page doit avoir le paramètre supplémentaire qui est le numéro de la chambre. En fonction du numéro de chambre, la liste des réservations prises pour cette chambre seront listés dans un tableau.

|  |
| --- |
|  |
| Liste des réservations de la chambre 1 |

Le tableau contient pour chaque ligne, une réservation de la chambre. Chaque colonne va indiquer les informations suivantes :

* Le numéro de la réservation à la ligne correspondante
* Les dates, l’état de la réservation
* Le client et l’hôtel de la réservation
* Un bouton de modification de la réservation qui nous amènera au formulaire de modification de la page
* Le bouton de suppression qui fera une suppression logique de la réservation sur le site internet (?)

#### Modification des chambres

Si l’administrateur décide de modifier une chambre plutôt que d’en afficher la liste des réservations, il accèdera alors à la page des modifications des chambres.

Cette page contient un formulaire qui permet de modifier toutes les informations de la réservation à l’exception des données qui sont exclusives à la structure de la base de données SQL.

Les données pouvant être modifiés sont :

* Statut
* Surface
* Type lits
* Statut
* Description
* Catégorie de la chambre

Un bouton « Envoyer » permet de confirmer la modification de la chambre une fois que l’administrateur a modifié la chambre. Il est renvoyé sur la page de la liste des chambres.

### Gestion des clients

#### Liste des clients de « Vivehotel »

|  |
| --- |
|  |
| Extrait de la liste de tous les clients de « vivehotel » |

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 identifiant
* Son email
* Un bouton « Réservation » qui permet d’accéder à la liste des réservations du client
* Un bouton « Modifier » qui permet d’accéder à la page de modification du client
* Un bouton « Supprimer » qui permet de supprimer un enregistrement d’un client

#### Réservations d’un client de Vivehotel

Cette page contient la liste des réservations qu’un client a prises. Chaque ligne du tableau de la liste de ses réservations contient le numéro de réservation, la date de début et de fin de celle-ci ainsi que d’autres informations. En supplément, un bouton de suppression d’une réservation et un bouton de modification sont référencés.

Si l’administrateur clique sur « supprimer », il fera une suppression logique de la réservation dans la base de données et sera redirigé sur la même page. Si l’utilisateur clique sur le bouton « Modifier », il accèdera à la page de modification de la réservation (que nous décrirons plus tard).

|  |
| --- |
|  |
| Extrait des réservations du client « client 1 » |

#### Modification d’un client

|  |
| --- |
|  |
| Exemple de formulaire de modification |

Lorsque l’administrateur clique sur le bouton « modifier » du tableau des « clietns », il accède à un formulaire de modification des « clients ». Il peut alors modifier certaines informations du client à l’exception du mot de passe.

Une fois qu’il clique sur le bouton envoyer, les modifications sont envoyées sur le serveur puis l’administrateur est redirigé dans la page de la liste des clients.

#### 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

|  |
| --- |
|  |
| Extrait de la liste des hôtels |

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, département). Les boutons supplémentaires permettent d’accéder à des pages différentes :

* Statistiques : permet d’accéder à l’ensemble des données de l’hôtel
* Services : permet d’accéder à l’ensemble des services de 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.

|  |
| --- |
|  |
| Données statistiques de l’hôtel « hôtel 1 » de la compagnie Vivehoel |

Cette page permet d’avoir de multiples informations sur l’hôtel. Lorsque l’administrateur a fini de lires les données sur l’hôtel, il clique sur le bouton « liste des hôtels » pour revenir à la liste des hôtels de la réservation.

#### Services d’un hôtel

|  |
| --- |
|  |
| Liste des services de « hôtel 1 » |

Supposons que nous sommes sur la page « hôtel » « index » au niveau du tableau de la liste des hôtels. Si l’utilisateur clique sur le bouton « service » d’un hôtel alors il pourra accéder à la liste des services de l’hôtel.

Il pourra alors cliquer sur le bouton « modifier » pour modifier le prix d’un service de son hôtel. S’il clique sur le bouton « supprimer », il supprimera l’hôtel qui est sur la même ligne que ledit bouton.

Il pourra ajouter un service sur son hôtel grâce au formulaire d’ajout d’un service à l’hôtel courant. Il indiquera quelle est la quantité de service que prend le client de l’hôtel.

#### Modification d’un hôtel

|  |
| --- |
|  |
| Modification des données de l’hôtel « hôtel 1 » |

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

Dans la page index 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 du personnel

#### Page d’index du personnel

|  |
| --- |
|  |
| Extrait du personnel de l’hôtel |

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

Chaque membre du personnel est identifié par son nom, son email et son identifiant. Les boutons supplémentaires permettent d’accéder respectivement aux pages d’édition d’un membre du personnel et le supprimer.

#### Page d’édition du personnel

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

Ce formulaire permet de modifier les informations suivantes de l’individu :

* Son nom
* Son identifiant
* Son mail
* Son mot de passe
* Son rôle
* Son hôtel

Lorsque le formulaire est envoyé avec le bouton « Enregistrer », les données de celui-ci vont modifier le membre du personnel. Le mot de passe sera modifié même si il n’est pas indiqué. Il convient alors de faire attention à bien retaper un mot de passe complet pour qu’il soit correctement modifié.

### Pages des réservations

#### Page d’index des réservations

La page d’index des réservations contient les 100 premiers enregistrements de la table « personnel ». Ces enregistrements sont

|  |
| --- |
|  |
| Extrait de la liste des réservations (capture à modifier) |

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 3 boutons :

* Service qui liste l’ensemble des services prises pour la réservation
* Modifier qui liste les modifications de la réservation
* Supprimer qui supprime l’enregistrement affiché sur la même ligne

#### Page de services des réservations

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

* D’un côté, il y a la liste de tous les services de la réservation sous forme d’un tableau.
* La deuxième partie de la page est un formulaire qui permet d’ajouter une réservation. Le formulaire comprend deux inputs :
  + Le premier qui permet d’indiquer quel est le champ à modifier
  + Le deuxième permet de créer un nouveau service pour une réservation
* Le premier input permet de sélectionner le nouveau service ajouté au client
* Le deuxième input permet d’indiquer la quantité de services pris pour le client

Lorsque l’utilisateur clique sur le bouton « Envoyer » du formulaire page, il revient sur cette même page. Les données qu’il avait entrées dans le formulaire sont envoyées en base de données.

La page rechargée affiche alors le nouvel enregistrement dans la liste des services pris de la réservation.

#### Page d’édition des réservations

|  |
| --- |
|  |
| Formulaire d’édition d’une réservation |

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

|  |
| --- |
|  |
| Liste des services proposées par la chaîne Vivehotel |

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

UNDER PROGRESS

#### 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 ».

#### Page de modification des tarifs

Cette page affiche l’ensemble des tarifs des chambres de l’hôtel. C’est un tableau à double entrée. Dans 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.

A l’intersection de la ligne de la catégorie « macategorie » et de la colonne de « machambre », il y a une cellule qui contient le prix de la chambre pour les chambres de type « machambre » dans les hôtels du type « macategorie ».

|  |
| --- |
|  |
| Grille tarifiaire des chambres en fonction du type de chambre prise et du type d’hôtel de la chambre |

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 prix modifié ne sera pas envoyé.

# 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 sont disponibles dans le sous-dossier « application/table ». Chaque fichier PHP de modèles d’une table hérite de la classe « Table » contenu dans le fichier « Table.class.php » du dossier de framework.
* Chaque dossier du sous-dossier « application/module » représente un contrôleur de l’application. Ce dossier contient un fichier qui se nomme Ctr\_{contrôoleur}.php. Ce fichier contient une classe « contrôleur » dont chaque méthode est une action de ce dernier.
* Chaque dossier de contrôleur contient aussi des fichiers PHP nommés « vue\_{moncontrôleur}\_monaction.php . Ce sont des fichiers qui représentent la vue de l’action « monaction » du contrôleur {moncontrôleur}. Le fichier PHP représentant le contrôleur et les fichiers PHP qui représentent chacun une vue d’une action du cotnrôleur.

|  |
| --- |
|  |
| 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 au premier mot 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.

Bien que le Framework soit MVC, il n’est pas toujours judicieux de définir des fonctions dans des classes, étant donné qu’elles sont utilisées partout sur le site.

Nous définissons alors un fichier de fonctions dans le dossier « framework ». C’est le fichier « functions.php ». Nous allons brièvement décrire les 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. Le redirige vers la page d’accueil si il ne l’est pas. |
| Auteur | Auteur du framework PDO 3.0. |
| Date et version | Version et date de dernière modification |
| Paramètres d’entrée | Aucun |
| Paramètres de sortie | Aucun |
| Fichiers accédés | // |
| 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 paramètrs. |
| Auteur | Auteur du framework 3.0, mis à jour par les développeurs du site |
| Date et version | 18/04/2023 |
| 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\_idl » : 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 etpermet d’échapper du texte contre les injections SQL. Cette fonction nécessite cependant une connexion de type « Mysqli », n’existant nullement dans ce projet. Son usage n’est donc pas pertinent.  Les objets de classes 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 ».  Cette solution est viable car nous possédons une connexion PDO pour pouvoir se faire.  La classe mère « Table » de connexion à la base de données du projet. Il s’agit d’une classe PDO. Son nom est « $link ».  La fonction « mres » va prendre une chaîne de caractère et la protéger des injections SQL à l’aide de la méthode statique de l’objet PDO de la classe Table. |
| Auteur | Auteur du Framework PDO 3.0 |
| Date et version | 24/02/2023 – 3.0 |
| Paramètres d’entrée | $s : chaîne de caractère |
| 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 |

|  |  |
| --- | --- |
| 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 |

|  |  |
| --- | --- |
| 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 |

|  |  |
| --- | --- |
| 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 |

|  |  |
| --- | --- |
| 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 |

|  |  |
| --- | --- |
| 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 |