Documentation Technique

L'application est écrite en PHP, JavaScript et HTML.

Les détails de l'installation de l'application sont décrit dans le fichier README.md présent à la racine de l'application.

Arborescence :

```
hotel/
├── conf/
├── Controllers/
├── Core/
├── datas/
├── Models/
├── Views/
├── webroot/
├── .htaccess
├── README.md
└── index.php
```

Les données sont sauvegardée sur un serveur MySQL (version 5.5 requise). La configuration de la base de donnée se fait dans le fichier *conf/database.php*.

A la racine de l'application, le fichier *.htaccess* réécrit l'URL afin que le fichier *index.php* puisse lire les informations contenu dans la requête http et lancer l'action demandée. L'application respecte une architecture modèle-vue-contrôleur.

Les modèles (*Models/*) traitent les données et interagissent avec la base de données.

Les vues (*View/*) sont répertoriées par dossier et contiennent les différentes interfaces de l'application.

Les contrôleurs (*Controllers/*) prennent en charge les événements, contrôle les données et joue le rôle d'intermédiaire entre les modèles et les vues.

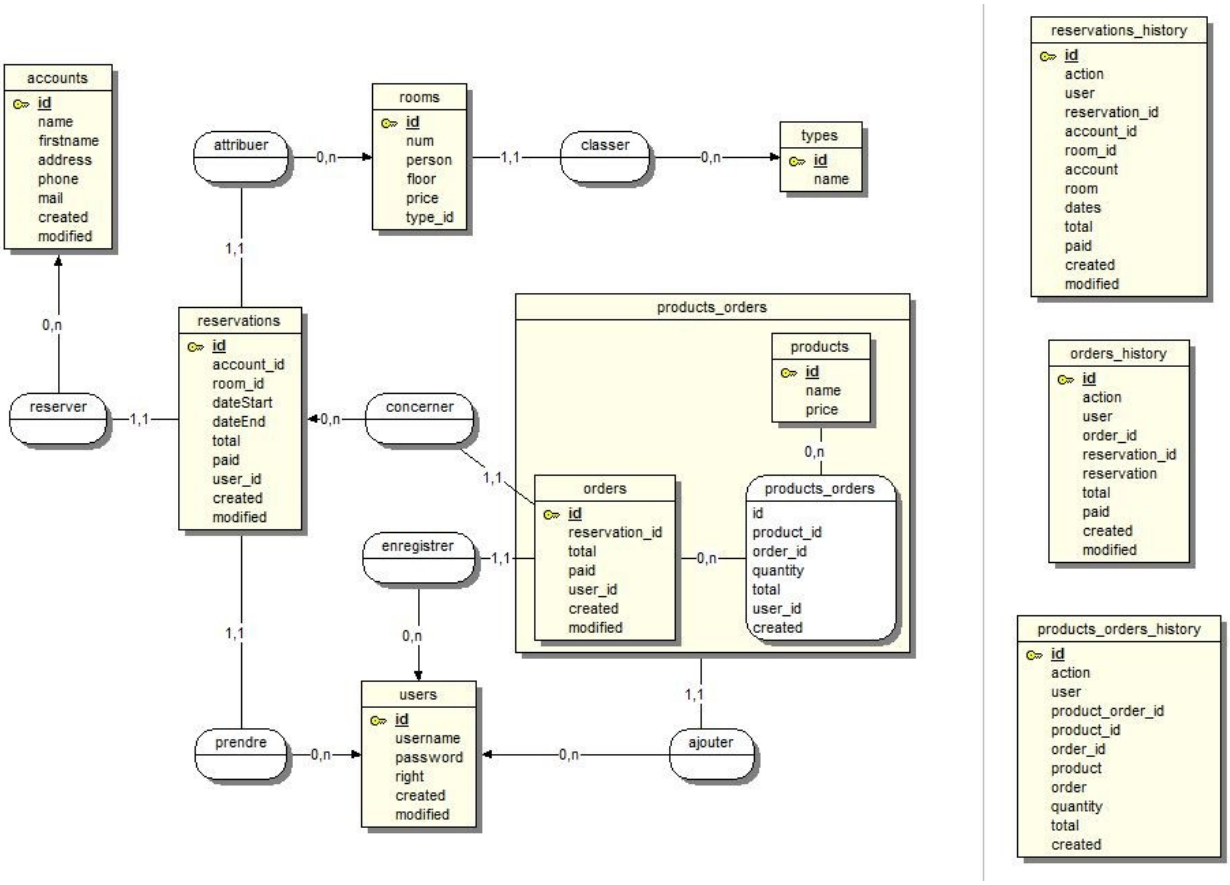
Le répertoire *Core/* contient les principaux fichiers, le contrôleur principal (*Controller.php*) et le modèle principal (*Model.php*). Les routes de l'application sont définis dans le fichier *Router.php*. Un dossier (*Addons/*) contient 3 modules de l'appliactions :

- *Request.php* permet d'envoyer des requêtes SQL au serveur de base de données.
- *Session.php* gère l'ouverture et la fermeture des session des utilisateurs.
- *Access.php* permet de verrouiller l'accès à certains utilisateurs à certaines actions.

Le répertoire *webroot/* contient les bibliothèques lues par le navigateur (fichiers CSS, JavaScript, et les icônes).

Les scripts SQL sont situés dans le répertoire *datas/sql/* et les documents annexes dans *datas/docs/*.

Base de données



Tout les nommages sont en miniscule et en anglais (exepté les propriété de plus d'un mot qui sont en camelCase). Le nom des tables est au pluriel, les clés primaires se nomment « id » et les clè étrangère se nomment par leur table de référence suivie de « _id ».

- Une réservation est réservée pour un compte, lui est attribuer une chambre qui est classé selon son type et est prise par un utilisateur.

- Une commande concerne une réservation, est enregistré par un utilisateur et contient plusieurs lignes caractérisées par un produit et une quantité.

- Les actions de créations, modifications et de suppressions sont stockées dans trois tables et peuvent être consultées uniquement par le ou les administrateur(s).

Les fichiers présents dans le répertoire *datas/sql/* :

00_drop_db_hotel.sql	Supprime la base de données
01_init_db_hotel.sql	Initialise la base de donnée
09_clear_tables_hotel.sql	Vide toutes les tables de la base de données (sauf administrateur)
10_struct_hotel.sql	Créé la structure de la base (tables + administrateur)
15_triggers_hotel.sql	Créé les déclencheurs (pour les calculs de totaux des réservations et des commandes, pour l'historisation)
20_datas_test_hotel.sql	Jeu de test

Fonctionnement

L'URL demandée par l'utilisateur est divisée par des slash « / », entre chaque slash se trouvent les paramètres de la demande. Le premier élément détermine le contrôleur à instancier, le second la méthode, les autres dont les paramètres passés à cette dernière. Le lanceur à la racine la l'application instancie le contrôleur et lance la fonction *start()* qui effectue les premiers traitements.

Les contrôleurs

Chaque fonction d'un contrôleur effectue une action. Voici les fonctions utiles héritées du contrôleur principal :

- *render()* : lance la vue passé en paramètre.
- *set()* : ajoute le tableau passé en paramètre à la liste des variables qui seront passées à la vue à son lancement. A chaque case du tableau son index deviendra le nom de la variable et sa valeur le contenu de la variable (ex : ['nom' => 'Jean'] devient $nom = 'Jean').
- *loadModel()* : charge le modèle passé en paramètre, si un identifiant est précisé la donnée sera chargée depuis la base de données.
- *setLayout()* : change le layout par défaut (mise en page). Le fichier est chargé depuis le dossier *Views/Layout/* dans laquelle la future vue sera inclus.
- *setViews()* : change le dossier de vues par défaut. Le répertoire ciblé est situé dans le dossier */Views/*.

Les modèles

Chaque modèle est lié à une table de la base de données, ainsi qu'avec ces colonnes. Ils permettent de lire, créer, modifier, supprimer des éléments de la base de données. Voici les fonction utiles héritées du modèle principal :

- *listAll()* : retourne un tableau contenant toutes les lignes de la table.
- *get()* : charge la donnée correspondant à l'identifiant passé en paramètre dans le modèle.
- *set()* : charge le tableau passé en paramètre dans le modèle.
- add() : ajoute le modèle dans la base de donné.
- up() : met à jour la base de donnée avec le modèl.
- del() : supprime le modèle de la base de donné.
- toArray() : retourne le modèle sous forme de tableau.
- loadModel() : retourne un modèle instancié.

Les vues

Chaque dossier dans le répertoire *Views/* correspond à un contrôleur, et contient également un dossier *home/* pour la page d'authentification, *error/* pour les erreurs et *Layout/* pour les mise en page. C'est dans les vues que sont les fichiers HTML (enregistrés en *.php* pour le traitement des scripts PHP).

Les requêtes SQL

Le module *Request.php* gère la lecture et l'écriture de la base de données. Les fonctions *execute()* et *fetch()* exécutent la requête SQL passé en paramètre, *execute()* retourne le nombre de lignes traitées et *fetch()* retourne le résultat de la requête (Utilisé pour les « Select »).

Les sessions utilisateurs

Les sessions sont géré par le module Session.php qui ouvre, ferme et récupère les session des utilisateurs.

Les restrictions d'accès

Le module *Access.php* bloque certaines pages aux utilisateurs qui n'ont pas l'autorisation d'y accéder. Ces règles sont définis par la fonction *authorize()* présent dans les contrôleurs. Les utilisateurs ont un droit d'attribué (right), et à chaque demande de page le module vérifie si l'action demandé par l'utilisateur est bien référencée dans la liste des actions qui lui sont autorisées dans a*uthorize()*.

Les utilisateurs avec le droit « all » bénéficient des droits d'administrateurs et ont donc accès à toute l'application.

Librairies externes utilisées :
- Materialize http://materializecss.com/
- jQuery https://jquery.com/

Fichiers

Fichiers	Rôle	Contenu
index.php	Lanceur de l'application	
.htaccess	Réécrit L'URL	
conf/database.php	Configurer la connexion à la BDD	Variables globales MySQL
Core/Controller.php	Contrôleur principal	Classe Controller
Core/Model.php	Modèle principal	Classe Model
Core/Router.php	Routeur	Classe Router
Core/Addons/Access.php	Module de contrôle d'accès	Trait Access
Core/Addons/Request.php	Module d'envoi de requêtes SQL	Trait Request
Core/Addons/Session.php	Module d'ouverture de session	Trait Session
Controllers/AccountController.php	Contrôleur des comptes clients	Classe AccountConctroller
Controllers/LogController.php	Contrôleur des historiques	Classe LogController
Controllers/OrderController.php	Contrôleur des commandes	Classe OrderController
Controllers/ProductController.php	Contrôleur des produits	Classe ProductController
Controllers/ProductOrderController.php	Contrôleur des lignes de commande	Classe ProductOrderController
Controllers/ReservationController.php	Contrôleur des réservation	Classe ReservationController
Controllers/RoomController.php	Contrôleur des chambres	Classe RoomController
Controllers/TypeController.php	Contrôleur des types de chambre	Classe TypeController
Controllers/UserController.php	Contrôleur des utilisateurs	Classe UserController
Models/AccountModel.php	Modèle des comptes clients	Classe AccountModel
Models/OrderModel.php	Modèle des commandes	Classe OrderModel
Models/ProductModel.php	Modèle des produits	Classe ProductModel
Models/ProductOrderModel.php	Modèle des lignes de commande	Classe ProductOrderModel
Models/ReservationModel.php	Modèle des réservations	Classe ReservationModel
Models/RoomModel.php	Modèle des chambres	Classe RommModel
Models/TypeModel.php	Modèle des types de chambre	Classe TypeModel
Models/UserModel.php	Modèle des utilisateurs	Classe UserMode