Titre professionnel

Développeur Logiciel

Niveau III



Réalisation d’un site web d'hôtellerie

Mendoza Miguel



Tout d’abord, je tiens à remercier l’équipe pédagogique de l'École de Simplon labellisée par la Grande École du Numérique, de m’avoir donné la possibilité de faire partie d’une deuxième promotion qui m’a permis de voir une nouvelle méthode d’apprentissage. Je remercie en premier l’équipe encadrante, Jean-Philippe Denet, Ishane Rouvier et ma chargée de promotion Emmanuelle Mouzay qui, tout au long de ces sept mois nous ont accompagnés. C’est grâce à leur pédagogie et à leurs savoir-faire que j’ai pu acquérir de nouvelles connaissances théoriques mais surtout pratiques et que j’ai appris à utiliser les outils nécessaires à l’élaboration de mon projet de fin de formation. Je voudrais également remercier tous les professionnels que nous avons pu rencontrer pour leurs disponibilités et leurs conseils.

Je voudrais aussi remercier tous mes camarades de promotion avec qui j’ai passé d’agréables moments. Ils m’ont permis de passer cette formation dans des conditions adéquates pour réussir, avec une volonté d’apprendre, de la cohésion et de la bonne humeur.



**Remerciements 2**

**Résumé en anglais 4**

**Introduction 5**

**1 - Présentation de la formation à simplon.co 6**

**2 - Compétences couvertes par le projet 7**

[**2.1 Maquetter une application :**](#_j6aa09iaaibb) **8**

[**2.2 Concevoir une base de données :**](#_qa7xzpr86vo) **9**

[**2.3 Mettre en place une base de données**](#_iyoqqhj9k309) **12**

[**2.4 Développer une interface utilisateur**](#_pqjljg4oj2rf) **13**

[**2.5 Développer des composants d’accès aux données :**](#_rb860uwxvq9l) **14**

[**2.6 Développer des pages en liens avec une base de données :**](#_3e5kse70ye5s) **15**

[**2.7 Utiliser l’anglais dans son activité professionnelle numérique :**](#_7vfhp0ldlrd) **15**

**3 - Cahier de charges 16**

**4 - Spécifications techniques 17**

[**4.1 Logiciels utilisés :**](#_3nm7m5e8axiw) **17**

[**4.2 Langages utilisés :**](#_iklgb3be4nbq) **18**

**5 - Réalisations 19**

[**5.1 Ensemble du projet**](#_72sd9o300ljz) **19**

[**5.2 Partie base de données**](#_9p7vuyz1osqa) **19**

[**5.3 Introduction à Node JS**](#_mk5pguiopnkz) **21**

[**5.4 Conception et réalisation de la gestion des administrateurs**](#_jjoi2frlqemc) **22**

[**5.5 Conception et réalisation de la gestion de chambres**](#_7f6wp6eu28on) **29**

[**5.6 Conception et réalisation de la gestion des réservation**](#_d79ck9u5rku6) **34**

[**5.7 Conception et réalisation de la gestion de récapitulatif**](#_mcdgacrn4ugs) **35**

[**Bonus partie logo**](#_m6a4slwvbr74) **36**

**6 - Conclusion 38**

**LEXIQUE 39**



First, i would like to talk about my project. It is a website and which concerns hotel rooms order. The point of my website is to allow customers to order some hotel rooms, they can do that with a form without having to subscribe or sign in. Secondly customers can also only browse or see which room is available. As matter of course if they have any problem, users can contact the website administrator with a form in the page “contact”. Consequently customers write their message, email, name… When they post their email-contact, we can help them and reply.

At last, I want to talk about the technical realisation of my project. On the one hand I had to install phpmyadmin, Nodejs and its modules. Then, I decided to create many diagrams to help me understand what the users will need in this website and which tables will be necessary for my database.

I began my idea with a paper, a pencil, I also used the software balsamiq and I make some sketches. It helped me to understand what I needed, how the website works and what I want for the design. So, after that I decided to realise many diagrams, they helped me to understand what users need and which table will be necessary for my database. Finally I have made a use case diagram, class diagram and sequence diagram.

In short, my database was realised with mySql and connected with NodeJs.

Furthermore I used HTML/CSS , JavaScript, frameworks like Bootstrap and jQuery for the front-end and NodeJs and mysql for the back-end.



Pour bien débuter mon projet, j’ai commencé par choisir les technologies que j’allais employer. D’abord, j’ai choisi de faire une maquette sur balsamiq pour avoir l’idée de la structure de mon site et ensuite faire quelques diagrammes comme le diagramme de cas d’utilisation, le diagramme de classe et le diagramme de séquence. Après avoir fait les diagrammes dont le but était d’avoir une idée des tables et des champs dont j’allais avoir besoin, j’ai ensuite commencé à coder avec les langages Node Js, SQL, HTML/CSS et JavaScript avec les Frameworks Bootstrap, JQuery.

Pour la conception du site web de mon projet, j’ai eu besoin d'installer phpmyadmin, Node js et ses modules afin de pouvoir coder avec les langages qui allaient être nécessaires pour la création du site web.

Le site permet à l’utilisateur de réserver une chambre d’hôtel via un formulaire sans avoir besoin de se connecter ou s’inscrire. Il peut aussi simplement naviguer et visualiser les chambres disponibles et en cas de problème, il peut contacter un administrateur du site via le formulaire dans la page contact en entrant ses coordonnées et son message. Pour mieux répondre à sa demande ou/et qu’on puisse le recontacter, il est obligé de renseigner son adresse email.

La formation Dev à Simplon.co c’est quoi ? 

* Apprendre à créer des sites Internet et des applications web, et en faire son métier
* Une pédagogie innovante :
  + Un apprentissage par la pratique, par projets (learning by doing)
  + Évaluation par les pairs (peer assessment)
  + Mise en situation de transmission (learning by teaching)
  + Apprendre à apprendre
  + Programmation en binôme (pair programming)
* Une formation entièrement prise en charge et gratuite pour les apprenant·e·s.
* Une formation ouverte à tous, et en priorité à des profils encore peu représentés dans le domaine : non-diplômés, demandeurs d’emploi, bénéficiaires de minima sociaux, personnes originaires de quartiers prioritaires, zones rurales, objectif de parité femmes-hommes. Et ce car il n’y a pas besoin d’être ingénieur ou très diplômé pour programmer.
* Une formation dont la sélection se fait sur la motivation et l’envie de travailler en équipe !
* Un écosystème riche comprenant une communauté de mentors, de professionnels, des partenaires publics, privés et non-marchands…
* Une équipe pédagogique renforcée, avec un·e lead formateur·trice, des formateurs·trices expert·e·s, un·e chargé·e de médiation emploi, une équipe placement…

## Une formation donnant accès au passage du titre professionnel « développeur logiciel »

Ce titre professionnel de niveau III est délivré par le Ministère du Travail (RNCP 5927), et représente l’équivalent d’un bac +2. En fin de formation (comme c’est le cas pour moi-même) ou après le contrat de professionnalisation, les apprenants sont accompagnés dans la réalisation de leur chef-d’œuvre, en vue d’une soutenance devant un jury de professionnel·le·s.

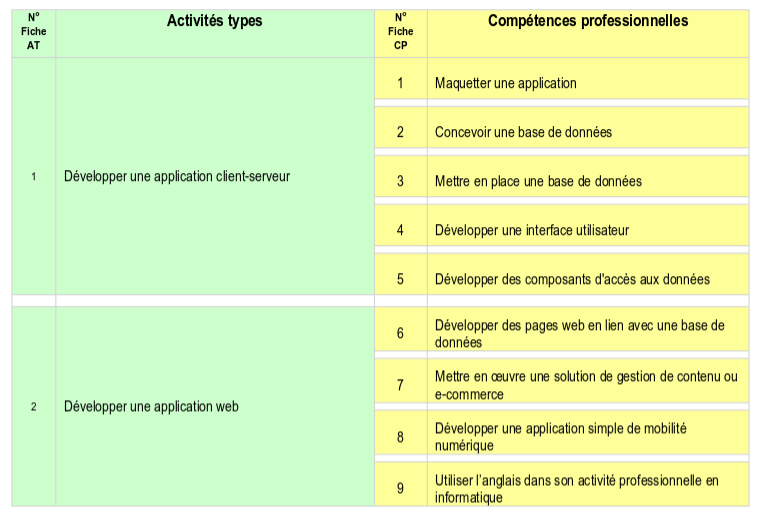


Pour valider le titre professionnel, ce projet à besoin de répondre au maximum de compétences réparties en deux activités type :

Développer une application client-serveur

Développer une application web

Pour rappel, voici l’ensemble des compétences à couvrir :



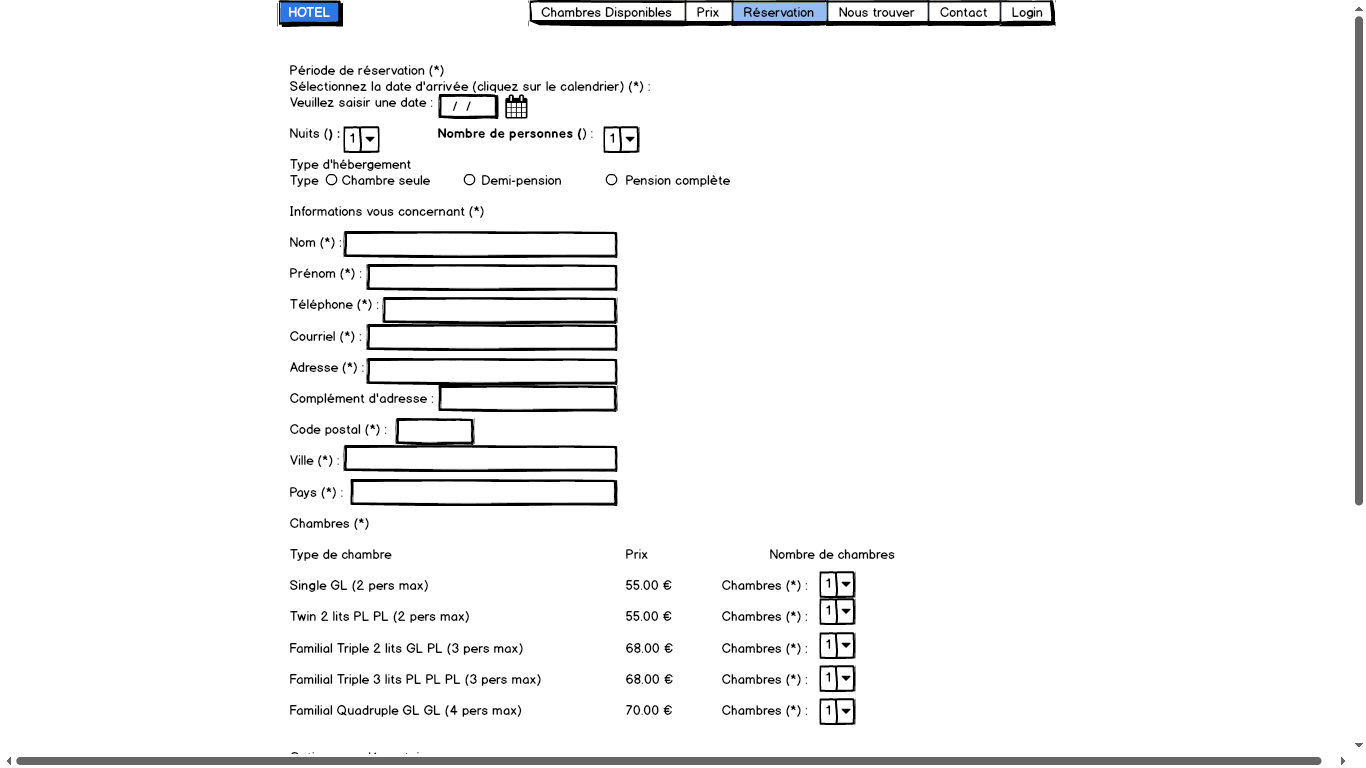


Ce projet a couvert plusieurs compétences pour l’obtention du Titre professionnel autant pour le front que pour le back-end. Suivant une formation de développeur Full-stack Javascript, nous avons eu l'occasion d’en apprendre sur les deux activités types demandées client-side et client-serveur. La partie Front-end pour réaliser notre partie interface web et la partie back-end afin de gérer nos pages web et la sécurité du site avec les requêtes envoyées au serveur.

## **2.1 Maquetter une application :**

J’ai réalisé cette maquette grâce au logiciel en ligne balsamiq pour avoir une idée de la partie graphique de mon site et de la façon d’organiser mes pages.

Voici à quoi ressemble mon site web.

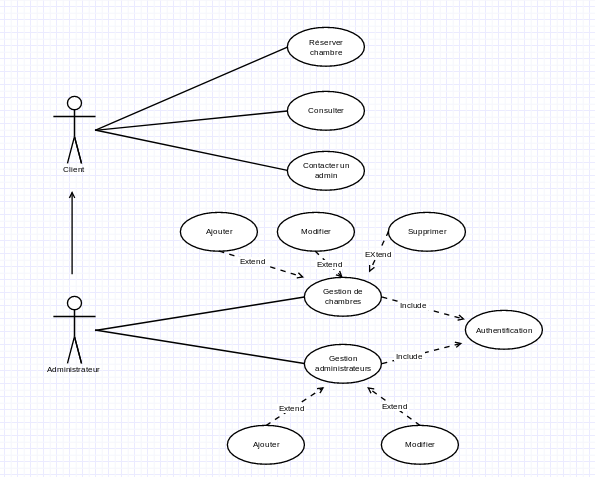


Page Réservation



## **2.2 Concevoir une base de données :**

Dans un premier temps pour la conception de ma base de données, j’ai eu besoin de faire mes diagrammes UML (use case, classe,séquence) pour déterminer la fonctionnalité de mon site et pour la conception de ma base de données.



Use case

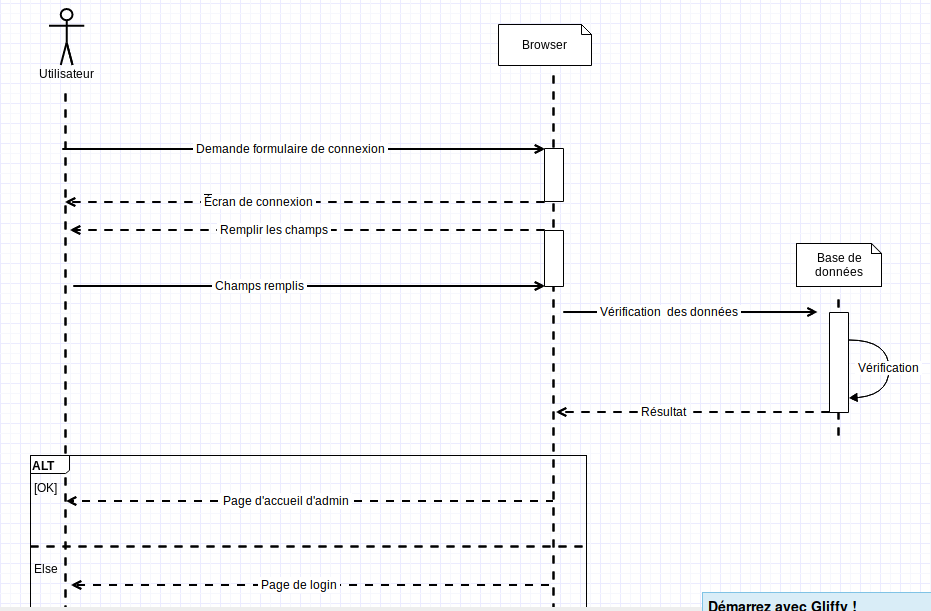


Diagramme de séquence : Connexion

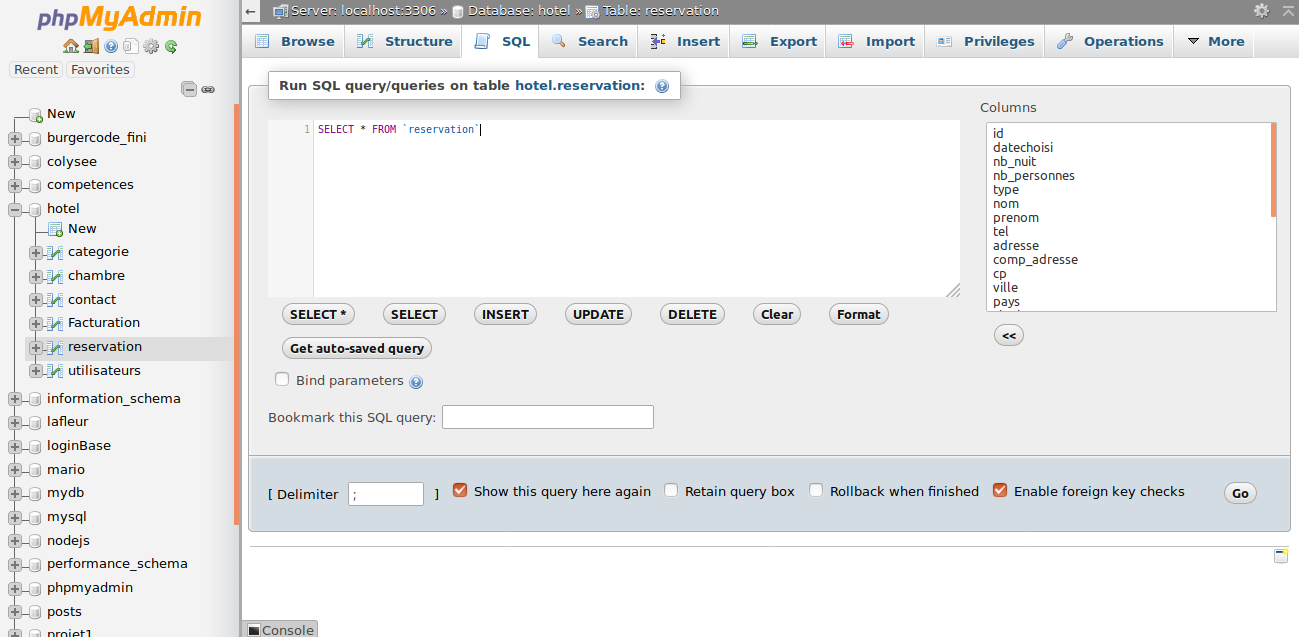
diagramme de classe(extraction de phpmyadmin)

Et grâce à ces diagrammes j’ai pu avoir une idée des tables et des champs dont j’aurai besoin pour créer ma base de données, afin de réaliser au mieux mon projet. Par la suite, après avoir créé les tables et les champs, j’ai réalisé mon diagramme de classe, avec les relations entre les tables.



## **2.3 Mettre en place une base de données**

Pour mettre en place ma base de données j’ai dû installer LAMP et PHPMYADMIN afin que je puisse gérer mes tables et champs en local. Ensuite, ayant eu les idées de mes tables et champs, j’ai commencé à remplir ma base de données au fur et à mesure de l'évolution du site.

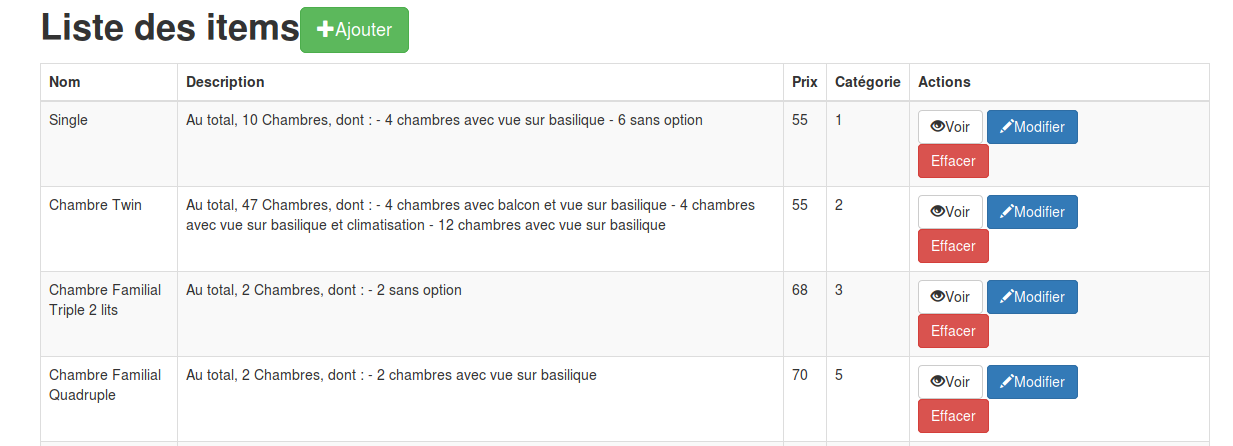


Base de données hotel et ses tables



## **2.4 Développer une interface utilisateur**

Pour la création de l’interface utilisateur j’ai du utiliser HTML/CSS, JavaScript et les Framework Bootstrap, Jquery pour la partie responsive et animation. La structure du site est basée sur le modèle MVC (Modèle, Vue, Contrôleur). Les langages utilisés dans le back-end sont Node Js et ses modules comme mysql, express, ejs pour le template de la page, bcrypt pour le cryptage de mot de passe, body-parser pour la récupération de données entrées par l'utilisateur via un formulaire etc …

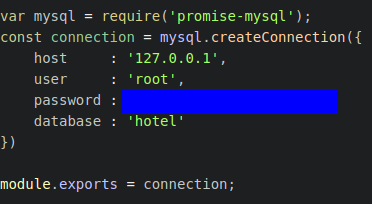


Page CRUD(Create:créer, read:lire, update:mettre à jour, delete:supprimer)de L’hôtel



## **2.5 Développer des composants d’accès aux données :**

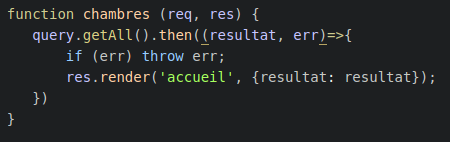
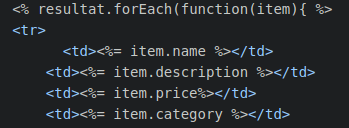
Pour avoir accès à ma base de données en utilisant comme langage principal Node Js et le module promise-mysql de l’application web.

Code connection à la base de données hotel

On fait appel au module promise-mysql avec le require, ensuite on utilise la fonction createConnection() pour effectuer la connexion entre ma base de données et NodeJs et on exporte la variable connection pour l’utiliser par la suite dans le modèle MVC .



## **2.6 Développer des pages en liens avec une base de données :**

Partie Node JS Partie EJS

Pour la partie Front et back end en liaison avec une base de données, j’ai utilisé EJS(Ember JavaScript), un template qui me permet d’afficher les données depuis ma base de données à ma Vue en passant par une requête sql. Comme on peut le constater dans les images ci-dessus, la fonction chambres nous renvoie le résultat de ma requête getAll() dans laquelle je demande à ma base de données de faire un SELECT \* de ma table. Par la suite, on envoie ce résultat à la page accueil. Finalement, pour parcourir la totalité de mon tableau, j’ai utilisé la boucle forEach qui m'a permis d’afficher tous les champs de ma table.

## **2.7 Utiliser l’anglais dans son activité professionnelle numérique :**

Pendant toute la formation, nous nous sommes obligés à utiliser l’anglais dans nos travaux, que ce soit seul ou en groupe.

J’utilise l’environnement de développement Intégré WebStorm en anglais.

Certains tutoriels utilisés sont anglais.

Je consulte le forum stackoverflow.com, w3schools.com, les documentations des langages sur les pages officielles qui sont plus complètes en anglais qu’en français quand j’ai besoin de réponses.



Le projet est un site web dans le domaine de l'hôtellerie sur un serveur LAMP qui doit être accessible sur tout type de navigateur.

Le développement sera effectué en utilisant des langages orientés web : HTML/CSS, JavaScript (NodeJs et ses modules, comme EJS pour la partie front-end, promise-mysql pour la partie back-end).

Il sera possible d’ajouter différents Framework propre à ces langages (Bootstrap, JQuery, Express ...)

L’application devra être responsive. L’ensemble des données seront stockées sur une base de données MySQL. Les données récupérées depuis nos formulaires, seront stockées dans notre base de données hotel, avec le middleware body-parser qui lui est aussi un module appartenant à Node JS.

**Partie Gestion des administrateurs:**

Les administrateurs du site peuvent se connecter via la page connection pour ensuite avoir dans la barre de navigation des liens qui leur seront attribués spécialement et seulement pour eux. Ils auront donc des liens vers la modification des chambres et l’ajout des autres administrateurs.

L’administrateur a le droit d’avoir un accès à la liste des administrateurs inscrits.

Il peut soit les modifiés soit les supprimer.

**Partie Gestion de chambres:**

Ajouter modifier et supprimer les chambres. Pour pouvoir effectuer ses actions, l’administrateur doit obligatoirement s’authentifier .

En ce qui concerne la partie codage toute cette partie est faite avec le CRUD (*“Create” qui permet la création des données dans la base de données,“Read” qui permet de lire les données, “Update” mettre à jour les données, “Delete” pour supprimer les données*).

**Partie Gestion des réservation:**

L’utilisateur sélectionne le type de chambre, le nombre de nuits souhaitées et pendant quelle période il veut réserver via un formulaire.

**Partie Gestion de récapitulatif:**

À la fin de la validation de la réservation de la chambre le client reçoit le récapitulatif de toutes les données entrées afin qu’il puisse passer au paiement.



## **4.1 Logiciels utilisés :**

Vous trouverez ci-joint une liste des logiciels utilisés lors de mon apprentissage en formation et ceux utilisés pour mon projet.

|  |  |  |  |
| --- | --- | --- | --- |
| ***Liste des logiciels*** | | ***Utilisé oui/non*** | ***Raison de son utilisation ou non utilisation*** |
| MySQL  Workbench | SGBD | non | Mon choix s’est porté sur PhpMyAdmin |
| PhpMyAdmin | Interface d’administration pour le SGBD MySql | oui | Prise en main facile sans avoir de grandes connaissances en base de données. Dans le cadre de mes études j’ai eu l’occasion de l’utiliser. |
| Atom | Éditeur de texte | oui | Pour commencer le projet Atom était le seul éditeur de texte que je pouvais utiliser. Il est plus léger qu’un IDE. |
| WebStorm | IDE( integrated de-  velopment envi-  ronment) | oui | C’est un IDE pour le full javascript et en tant qu'étudiant on a eu une licence gratuite. |
| Git(GitHub) | gestion de ver-  sions décentralisé | oui | Pour la sauvegarde de mon projet et avoir une version antérieure de mon projet lors qu’il y a des modifications majeures par exemple. |
| Balsamiq | Outil de Ma-  quettage | oui | Un wireframe utilisé auparavant et facile d’utilisation, pas besoin d’avoir de grandes connaissances sur le logiciel, on peut facilement créer des maquettes |
| pidoco | Outil de Ma-  quettage | non | Mon choix s’est porté sur balsamiq. |



## **4.2 Langages utilisés :**

Vous trouverez ci-joint une liste des langages et frameworks utilisés lors de mon apprentissage en formation et utilisés pour mon projet.

|  |  |  |
| --- | --- | --- |
| ***Liste des langages*** | ***Utilisées oui/non*** | ***Raison de son utilisation ou non utilisation*** |
| Javascript | oui | Le Javascript a été utilisé dans un plugin par google traduction pour permettre que le site soit sous différents langages |
| MySQL | oui | Pour faire des requêtes, appels aux données vers ma base de données |
| EJS | oui | C’est un langage client side qui me permet d’afficher les données dans les VUE html |
| HTML | oui | C’est pour la structure de mon site web tout en respectant les normes du W3C |
| CSS | oui | Pour la partie design du site et rendre plus attirante l’application |
| NodeJS | oui | Node permet d’interpréter le langage Javascript côté serveur, il m’a fallu l'utiliser pour développer la partie back-end de mon application avec ses modules |
| Express | oui | Ce Framework a été utilisé pour la gestion des routes à partir de NodeJS |
| Bootstrap | oui | Pour la partie responsive de mon site et les boutons déjà fait par les classes du framework |
| JQuery | oui | Pour les animations de mon site comme le petit calendrier lors de la réservation |



## **5.1 Ensemble du projet**

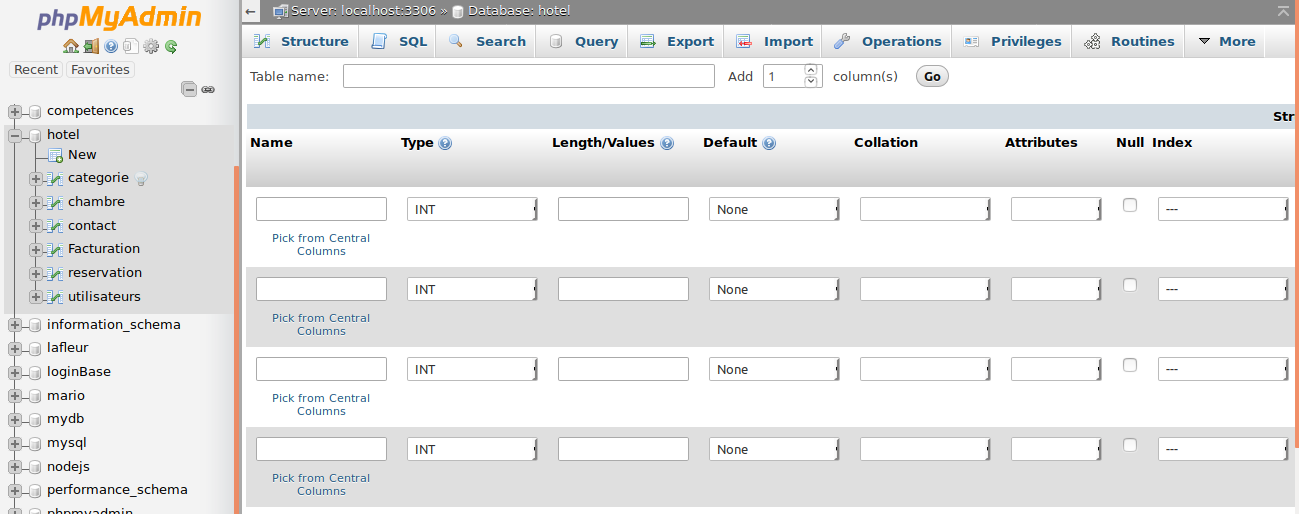
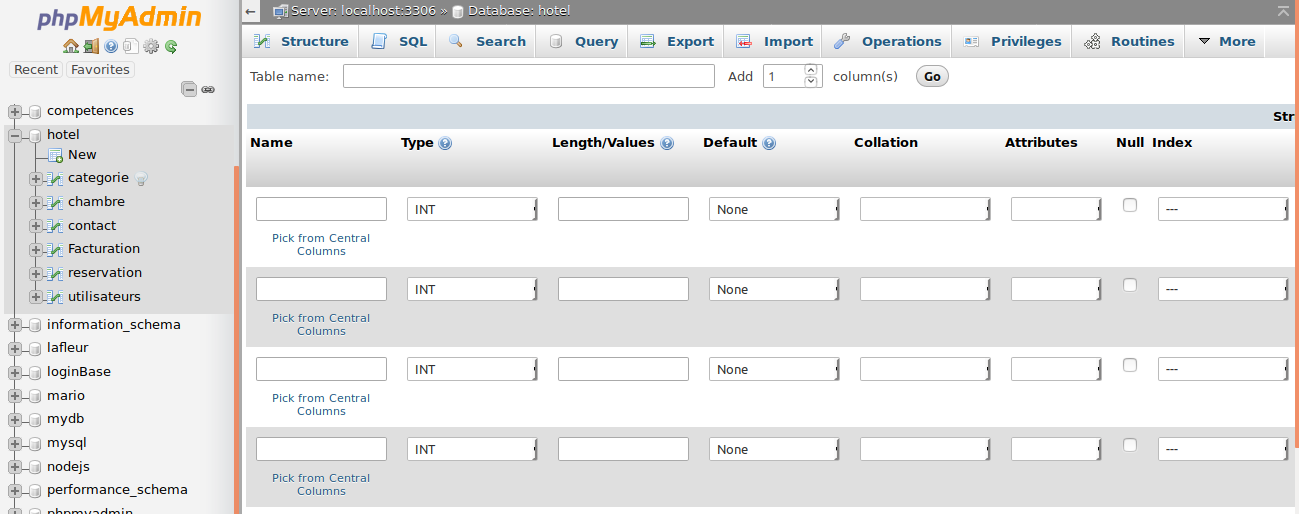
Pour commencer la présentation des objectifs du projet je vais commencer par expliquer la connexion et l’installation des modules utilisés pour la réalisation de l’application web et la façon dont elle a été codée.

Premièrement, pour la connexion à la base de données, j’ai installé le module promise\* mysql via le terminal en tapant npm install promise-mysql, cela va générer automatiquement le fichier node\_modules et l’écrire dans la partie dependencies. Les dépendances sont renseignées sous forme d’objet dans le fichier package.json, exemple { "promise-mysql" : "et la version"}. Mais avant de faire cela il faut taper la commande npm init qui est indispensable pour pouvoir utiliser npm\* avec notre projet, cela génère aussi automatiquement le fichier package.json.

Ensuite, dans la page de connexion à la bdd\* on fait appel au module mysql-promise avec un require. Pour créer la connexion à la bdd on fait appel à la méthode .createConnection() avec l’objet mysql c’est-à-dire mysql.createConnection() car quand on appelle le module, l’objet qui est créé est nommé mysql. La variable créée pour la connexion sera exportée afin de pouvoir la réutiliser dans la partie modèle,c’est-à-dire lorsqu’on fera des requêtes sql, et on refera appel au module avec le chemin relatif.

## **5.2 Partie base de données**

Pour travailler la partie bdd et ayant travaillé avec phpmyadmin, toute cette partie “création” de base de données ainsi que les tables ont été créées via l’interface de phpmyadmin. Elle nous facilite la tâche sans toucher au code pour la création des tables, c’est-à-dire, sans les requêtes sql et c’est de même pour l’insertion des données dans la base si on veut tester des données. Pour faire les jointures entre les tables et les autres commandes je suis parti voir sur le site sql.sh afin de comprendre leur fonctionnement. Ce site nous donne la totalité des commandes sql avec des tutoriels et cours.

Création d’une table et colonnes via l’interface graphique phpmyadmin

Dans la partie “table name” on met le nom de la table et à la suite on a le nombre de colonnes qu’on souhaite ajouter dans la table. Il est possible de mettre le nom des colonnes, le type de la colonne, la longueur des caractères et autres options dont on ne se servira pas vraiment dans le projet. Alors qu’avec une requete sql il faut tapé cette commande dans la partie sql sur l’interface :

CREATE TABLE nom\_de\_la\_table  
(  
 colonne1 type\_donnees,  
 colonne2 type\_donnees(longueur),  
 colonne3 type\_donnees(longueur),  
 colonne4 type\_donnees(longueur)  
)



## **5.3 Introduction à Node JS**

Qu’est-ce que c’est node JS?

Node.js offre un environnement côté serveur qui nous permet aussi d'utiliser le langage JavaScript pour générer des pages web. Globalement, il vient en remplacement de langages serveur comme PHP, Java EE, etc.



Schéma représentant de l'exécution de node js côté serveur et côté client



Modules utilisés de Node JS :

* promise-mysql : permet de faire ma connection à ma base de données.
* method-override : Permet de gérer les requêtes REST\*, c’est-à-dire les requêtes adressées au serveur sur le type de la requête. Dans la partie HTML des formulaires ils possèdent un attribut method dont la valeur peut-être GET(les valeurs transmises au serveur seront affichées dans l’URL de la requête), POST(contrairement au GET avec cette méthode les données saisies dans le formulaire sont cachées lors de la transmission au serveur, elles sont inscrite dans l'entête de la requête HTTP envoyé au serveur).
* express-session : stocke les données de session sur le serveur, il ne sauvegarde que l’ID session dans le cookie lui-même, mais pas les données de session.
* express-fileupload : permet de télécharger des fichiers via un formulaire et de les stocker en local.
* express : est un framework basé sur le concept de middlewares*.* Gestion des routes(URL) à l’application.
* ejs : Template pour afficher les données extraites depuis ma base de données via les requêtes sql et les afficher dans la page HTML.
* body-parser : récupérer les données des formulaires.
* bcrypt : hasher\* les mots de passe des utilisateurs.

## **5.4 Conception et réalisation de la gestion des administrateurs**

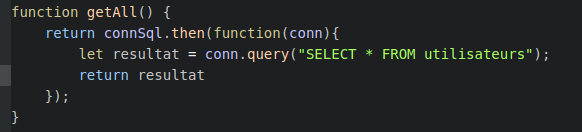
Dans cette partie, un administrateur peut gérer les membres ajoutés par lui-même soit en les supprimant ou en les modifiant.

Quand l’administrateur se connecte au site web il aperçoit une barre de navigation qu’un utilisateur internaute n’a pas le droit d’avoir sans s’y être connecté.BarreNavAdmin.png

Barre de navigation pour l’administrateur

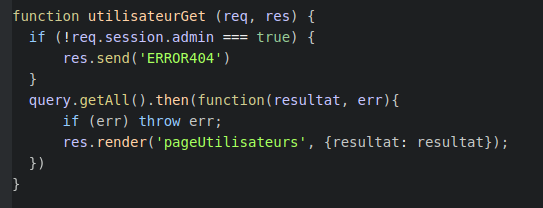


Après s’être connecté, il est dirigé vers une page dans laquelle on aperçoit la liste des administrateurs du site où il peut modifier et supprimer les administrateurs et peut gérer la gestion de chambres comme on peut l'apercevoir ci-dessus. “Add administrateur” est la page dans laquelle on peut ajouter des nouveaux administrateurs, “Modification” page dans laquelle on a la gestion des chambres.



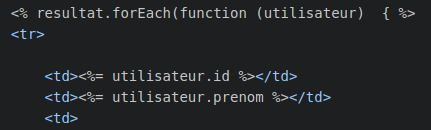
Requete sql pour afficher tous les utilisateur de la table utilisateurs : Partie Modèle

Dans cette partie du code on utilise la variable connSql qui est un appel au module de la connexion à la base de données ensuite on utilise la function .then() pour récupérer le résultat ou l'erreur d'une promesse. Pour exécuter la requête on utilise la function .query() qui nous retourne un résultat (si il y en a) en tant qu’objet et ce résultat on l’affecte à une variable qui va s’appeler resultat qu’on va retourner dans la function getAll(). Notre requête ici affichera tous les champs de la table utilisateurs grâce au “SELECT \*” astérisque c’est pour la sélection des tous nos champs de notre table au lieu d'écrire tous les noms de colonnes à la main, c’est-à-dire si dans notre table utilisateur on a un champ id, identifiant, password pour sélectionner les 3 champs on fera (SELECT \*).

Affichage de la requête effectué vers la vue HTML : Partie Contrôleur



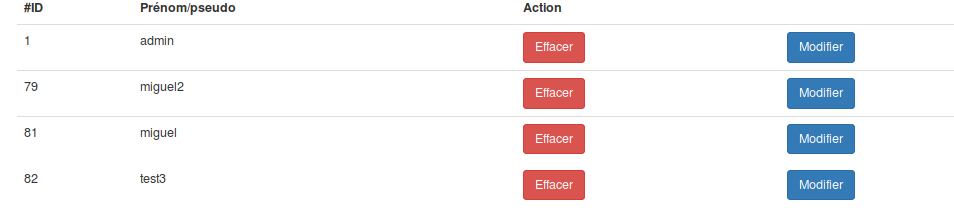
Dans cette function utilisateurGet(), on vérifie si l’utilisateur est bien connecté avec le module express-session avant d'accéder au lien qui lui affichera la liste des utilisateurs et lui donner les liens qui ne sont dédiés qu’aux administrateurs du site web. Ici, notre query est une variable qui fait appel au module utilisateur de notre partie modèle puis, on utilise la function qui lance notre requête à notre bdd. Enfin, ce resultat est envoyé vers la vue pageUtilisateurs mais notre URL sera “/Utilisateurs”. Ici la function res.render () renvoie notre template qui est en local, le html au client, et ce qu’il y a entre les accolades est le résultat de la requête qu’on transmet à la vue HTML qui sera affiché avec notre template EJS.



Affichage des utilisateurs avec EJS

Comme template pour afficher la liste des utilisateurs, j’ai utilisé EJS, celui-ci on le déclare dans notre app.js pour l’utiliser. Pour utiliser EJS dans notre page HTML, il faut toujours utiliser ses balises propres à EJS qui sont <% %> . On récupère resultat qui a été envoyé depuis notre contrôleur, on utilise la boucle forEach qui nous permettra de parcourir tout notre tableau resultat afin de pouvoir afficher ce qu’on veut de notre tableau à notre page html. À l'intérieur de notre forEach on crée une function avec un paramètre qui est utilisateur celui-ci nous permettra d’afficher toutes les données de chaque champs de notre table utilisateurs comme ici. On affiche l’id et le prénom des administrateurs. Ce qui donne dans notre page html cette vue ci dessous.

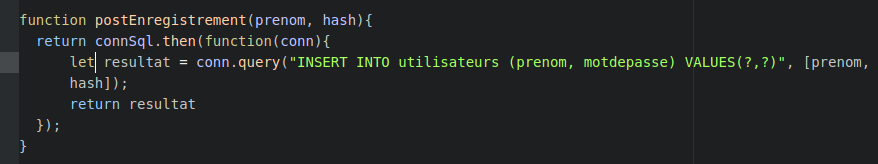




Liste des utilisateurs avec EJS

Partie enregistrement des administrateurs :

L’enregistrement des nouveaux administrateurs qui n’est possible que si l’admin est connecté au site web se déroule de cette manière. Dans la partie du code “toutes nos function seront exportée afin de pouvoir les réutiliser” :



Function pour ajouter un admin dans la bdd

Ici la function postEnregistrement() a besoin de deux paramètres qui sont prenom pour l’identifiant et hash qui sera le mot de passe de l’admin crypté avec le module bcrypt que l’on déclare dans le contrôleur et vient évidemment avant de créer la function “const bcrypt = require('bcrypt');”. On fait appel au module pour lui dire qu’on va l’utiliser et “const saltRounds = 10;” pour la force de cryptage. Dans notre insertion à la bdd on envoie le prénom et le mdp qu’il tape dans le formulaire.



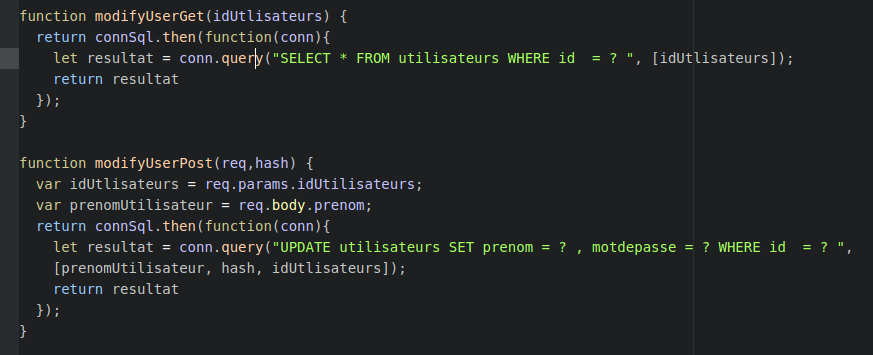


Affichage de la vue enregistrement Administrateur

La vue pageFormulaire est visible à l’admin que s’il est connecté au site, si un internaute essaie d'accéder via l’URL “/enregistrement” alors qu’il n’est pas connecté on lui renvoie ERROR404. Si l’admin est connecté lors de l’ajout d’un autre admin il saisit le prénom et le mot de passe. Avant d'insérer dans la bdd, on crypte le mdp avec bcrypt et sa méthode hashSync qui prend comme paramètre le mot de passe et le force du cryptage. Ensuite on l'envoie à la requête si tout est bien inséré, aucun champ vide et tout est validé on le redirige vers la liste des utilisateurs. Si le champ est vide on lui renvoie la page d’enregistrement pour les admins.

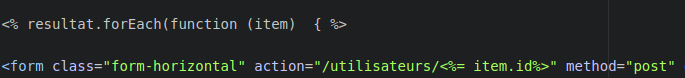


Partie modification des administrateurs :



Modification utilisateur partie Modèle

Modification utilisateur partie Contrôleur



ID dans l’URL dans la vue modification

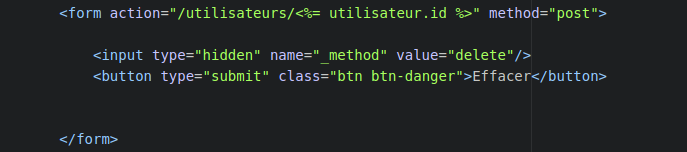


Dans la première function on récupère les informations de l’utilisateur sélectionné dans la vue avec le “SELECT \*”. Lorsque l’admin clique sur modification, on renvoie l’id de celui sélectionné par l’URL. Par la suite, dans notre sql, on n’extrait les informations que de celui sélectionné grâce au “WHERE id =?” et on retourne notre résultat de la requête qui seront affichés dans la vue modification qui sera envoyé par la function modifyUserget() de la partie Contrôleur.

Lorsque l’admin modifie les informations de l’utilisateur dans notre formulaire on récupère les informations rentrées par l’admin, si modifié, on fait un UPDATE dans notre requête ce qui permettra de faire une mise à jour des données dans la bdd tout en sélectionnant l’id de l’utilisateur. Si modification effectuée, on redirige vers la vue qui nous affiche la liste des utilisateurs.

Partie suppression des administrateurs :

Pour supprimer un admin de la liste des utilisateurs, on utilisera dans notre back-end le module methodOverride qui nous permettra de créer notre propre méthode supprimer. Cette méthode sera utilisé avec un input caché dans notre formulaire mais la méthode dans le formulaire sera toujours du POST.



Bouton supprimer partie Vue

Lorsque l’on déclare le module methodOverride on lui dit qu’il nous faut un paramètre(c’est-à-dire le nom de notre méthode) et ce paramètre sera récupéré grâce à une function dans laquelle retourne la méthode et qui sera récupéré dans le formulaire. Dans notre vue l’attribut name sera utilisé dans un champs caché et value qui nous indiquera le name de la méthode à utiliser. Pour la partie SQL ça sera un DELETE à utiliser tout en vérifiant l’id qu’on lui envoie à la requête, si utilisateur supprimé on reste sur la page de la liste des utilisateurs.

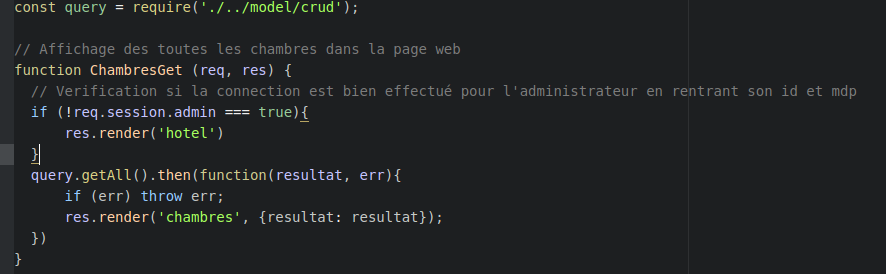
## 

## **5.5 Conception et réalisation de la gestion de chambres**

Comme vu précédemment, l’administrateur possède une barre de navigation propre à lui lorsqu’il se connecte(vous pouvez la retrouver à la page 22). Quand il clique sur modification il est dirigé vers une Vue qui nous affiche la liste des chambres. Dans cette liste, on aura le nom de la chambre, sa description, son prix et sa catégorie et les actions qu’il peut effectuer pour chaque chambre(Voir page 13).

Pour afficher la liste des chambres on fait de la même manière que pour la liste des utilisateurs dans notre partie modèle le fichier appeler crud.js on exportera le module de la connexion à la bdd pour envoyer des requêtes SQL.

On vérifie que l’admin est bien connecté sinon on lui renvoie la Vue accueil

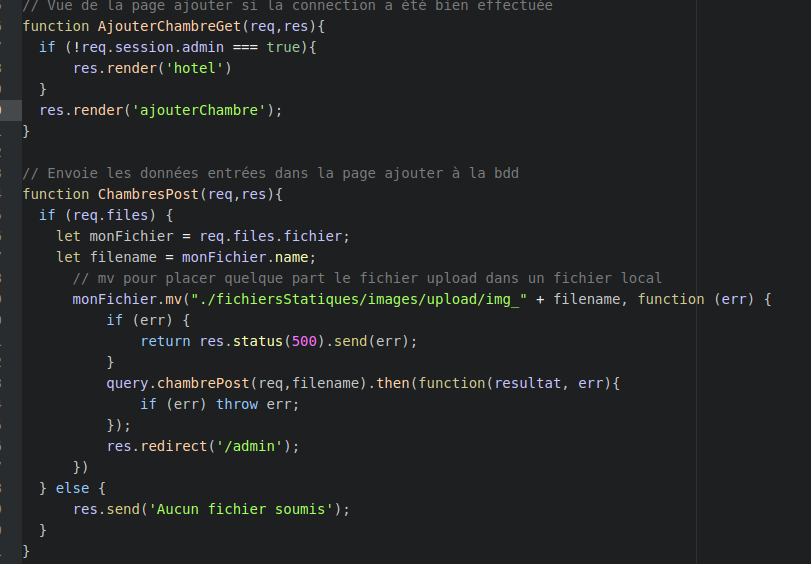


Visualisation de la liste des chambres : Partie contrôleur

Dans notre partie Modèle on fait la requête suivante “SELECT \* FROM chambre” dans une function qui est getAll(), elle nous retournera le résultat de la requête, la function sera exportée pour l’utiliser dans notre contrôleur. Ensuite, notre function ChambresGet() renvoie le résultat de notre requête vers la Vue pour la gestion des chambres(Voir page 15 : partie 2.6). Dans cette Vue on aura la liste des chambres plus les actions que l’administrateur pourra effectuer(page 13).



Partie Ajouter chambre :

Insertion de chambres : Partie Contrôleur

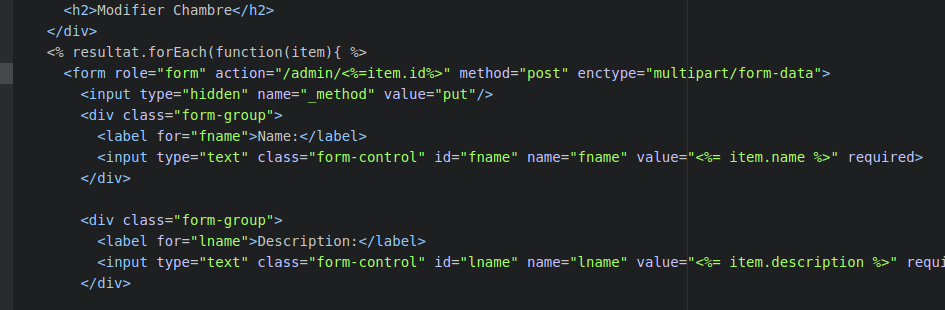
Quand l’admin clique sur l’action “AJOUTER” dans la Vue de gestion des chambres il est redirigé vers une page dans laquelle il y a un formulaire à remplir. Celui-ci fait entrer le nom, la description, le prix, la photo, et la catégorie de la chambre. Pour l’ajout d’une photo et le stockage en local, j’ai utilisé le module express-fileupload qui me permet de placer l’image dans un fichier de l’application. Dans notre cas, ça sera dans notre dossier fichierStatiques. Lorsque l’on stocke l’image dans notre application, on ajoute un préfixe avant le nom de l’image, question de sécurité. Le préfixe utilisé ici est “img\_ + nom de l’image” et dans notre base de données on stocke le nom de l’image.



Pour cela, on extrait le nom de l’image comme toutes les autres informations de la chambre avec le body-parser mais il ne faut pas oublier d'insérer encType="multipart/form-data" dans le formulaire lors du form sinon sans ça, l’upload des fichiers ne fonctionne pas. Enfin lorsqu’on récupère toutes les informations de la chambre, on les insère dans la bdd avec un INSERT INTO nom de la table (nom des champs) et les values (qui sont récupérées dans le formulaire).

Partie Modification chambre :

Pour la vue modification on refait la même chose que pour avoir les informations des administrateurs c’est-à-dire un “SELECT \* FROM chambre where id= (la chambre choisi)” dans la requête pour avoir toutes les informations de la chambre dans notre vue et dans chaque input. Pour cela on utilisera la boucle foreach dans le HTML avec ejs(template pour afficher les données de notre base de données) toujours avec laquelle on parcourra tout notre objet. Cet objet sera le résultat de notre requête.



Extrait de la vue Modifier Chambre

Dans cet extrait de code on peut apercevoir la boucle ForEach et avant cette boucle on a notre objet qui nous retourne le résultat de la requête. Cette requête sera affectée à une variable déclarée dans la function(se trouve dans la partie Modèle) et qui sera retournée par elle-même dans notre contrôleur lorsqu'on enverra la page à l’admin avec notre objet, le résultat de la requête.

Pour afficher les infos de la chambre en utilisant notre boucle on met entre les balise de EJS <%= item.nom\_de\_la\_colonne\_de\_notre\_table %>.



Et lorsque l’admin effectuera une modification on utilisera dans la partie Modèle dans une function .query(“Update chambre SET colonne = ?, colonne =? …”) les points d’interrogation seront les données récupérées avec le body-parser dans notre formulaire, c’est-à-dire les données entrées par l’admin dans le formulaire de la page modifier. Si modifications effectuées on lui renvoie la page de gestion de chambres.

Partie Voir chambre :

Pour l’action “voir” de la page gestion de chambre c’est la même action que dans la partie modification sans l’update car on refera un “SELECT \* from chambre where id=(chambre sélectionnée)” pour la requête dans notre partie modèle et dans notre partie contrôleur on exécutera la requête tout en renvoyant le résultat de cette requête dans la vue “Voir chambre”.



Vue Voir chambre



On affiche toutes les informations de la chambre sauf que dans cette Vue il ne pourra que voir les informations et pas les modifier. On les affiche avec notre boucle ForEach tout en utilisant les balises de EJS.

Partie Supprimer chambre :

Pour supprimer une chambre, on utilisera notre module method-override, celui qui permet d’utiliser la méthode delete avec un input caché dans le html. Pour utiliser la méthode DELETE, il suffit d’indiquer la valeur “delete” dans l’attribut value du champ caché. Ensuite, lors de la suppression dans notre Modèle on aura une function deletechambre(paramatre ID) dans laquelle la requête envoyée à notre bdd sera un DELETE FROM chambre where= ? (paramètre de la function). Enfin, dans notre partie Contrôleur, celle qui exécutera l’action demandé par l’admin, dans notre cas ici ça sera la suppression d’une chambre, si la chambre est supprimé l’admin ne sera pas redirigé vers une autre page, il reste sur la même page.

## 

## 

## 

## 

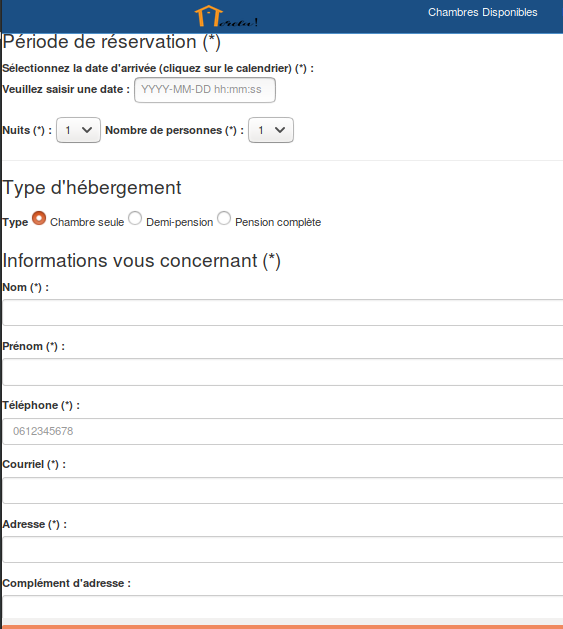
## 

## 

## 

## **5.6 Conception et réalisation de la gestion des réservation**

La réservation s’effectue avec un formulaire dans lequel on aura besoin de rentrer ses coordonnées, le nombre de nuits et les chambres qu’on voudra occuper et il faudra renseigner la date et l’heure d'arrivée afin que la chambre demandée soit disponible et en état de pour recevoir le client.



Extrait de la Vue reservation

Le client peut réserver sans avoir besoin de se connecter au site, toutes les données entrées par le client seront directement enregistré dans la bdd.

Dans la partie code, j’ai créé un fichier à part appelé Client.js dans le Modèle et utilisateurController.js partie contrôleur, dans notre modèle on fait appel à notre module de connexion a la bdd.



Ensuite, on fait notre function qui nous permettra d'insérer les données entrées par le client. On les récupère avec le body-parser via le req.body.(la valeur mise dans name des balises html) et ces informations seront envoyé dans la bdd avec la requête “INSERT INTO reservation(champs) VALUES(values récupérer depuis le formulaire)”. Si réservation effectuée, on dirige le client vers une page de récapitulatif des données entrées par le client lui-même.

## **5.7 Conception et réalisation de la gestion de récapitulatif**

Comme dit précédemment si réservation effectuée par le client, il aura ensuite une page dans laquelle il verra toutes les informations renseignées pour sa réservation.



Extrait vue récapitulatif

Côté code, dans cette page, j’ai fait un inner join pour lui attribuer un id facture à la réservation faite par le client, c’est-à-dire que chaque réservation lors de l’insertion à la bdd aura un id facture qui lui sera automatiquement relié avec la table Facturation.

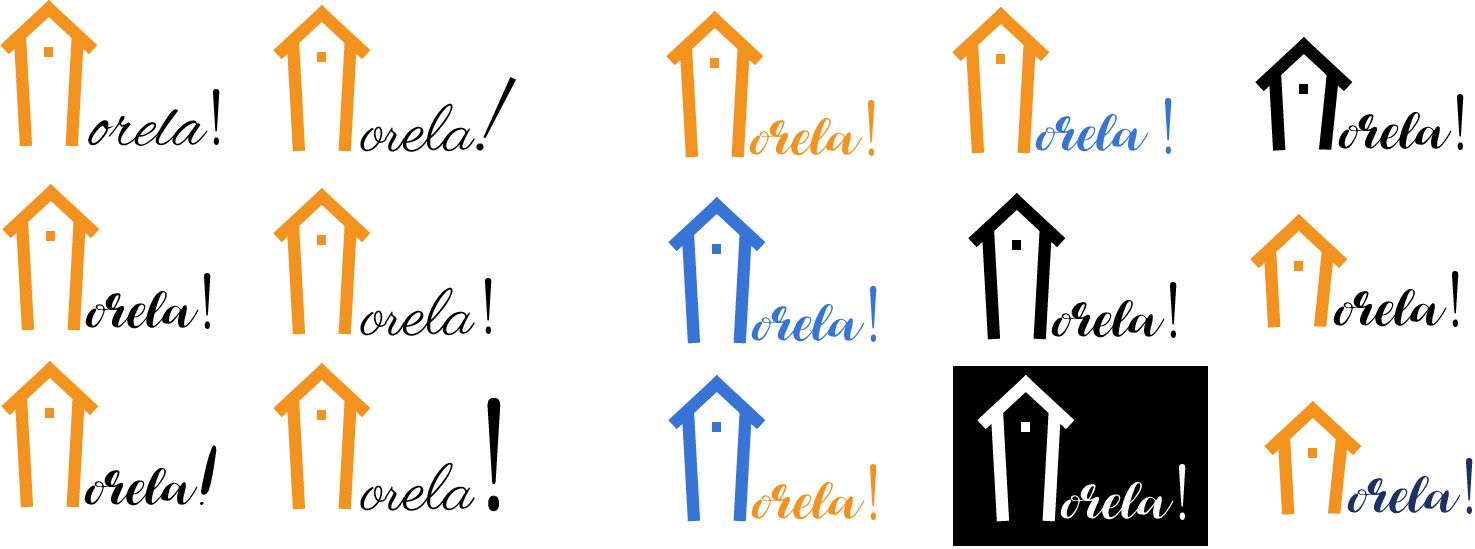


Pour cela, le champ id\_facturation sera une clé étrangère dans la table reservation qui fera référence à la table Facturation et plus explicitement au champs id de cette table.

Lors de la requête dans notre Modèle, on fera “SELECT \* FROM reservation INNER JOIN Facturation ON reservation.id\_facturation = Facturation.id” pour afficher tous les champs de deux tables reservation et Facturation, ensuite on retourne le resultat et dans notre Contrôleur on renvoie ce résultat dans la Vue HTML afin d’afficher les données au client.

## **Bonus partie logo**

Il a été réalisé sous illustrator avec deux typos parce que casse un typo calligraphie avec du sans serif c'est moderne, ça donne du peps et de la jeunesse. Le choix de la couleur orange s’est fait car elle se marie bien avec la couleur bleue du site, mais aussi parce que c'est la communication cette couleur. C'est aussi visible de loin, le but étant que ça reste en tête de la clientèle et pour qu'ils se souviennent du site de l’hôtel et aient envie de revenir.



Liste de logo effectué





Logo du site

Pourquoi Horela! ?

Premièrement car c’est un mot inconnu dans les recherches de Google et que du coup, il y aura plus de chances de le retrouver en premier de la liste en cas de mise en ligne du site. Le nom du site vient de Ho = Hogar en espagnol qui veut dire maison et rela= Relax pour définir le calme, le repos, et pour passer un moment dans toute tranquilité.



La création de mon site web n’a pas été aussi facil que je pensais que ça allait être car je commencai a apprendre un nouveau langage web qui était node JS, que je ne connaissais pas du tout, même si j’avais déjà des petites base en javascript l’apprentissage de nodeJS et ses modules n’ont pas été facil à mettre en pratique dans mon projet mais en regardant des tutoriels et en me renseignant sur le web j’ai réussi à me débrouiller, et sans l’aide des mes camarades mon projet n’aurait pas été réalisable, je les remercie pour leur apport à mon apprentissage durant la formation.

je suis aussi conscient que ce site peut évoluer dans l’avenir et donc il faut vraiment au niveau du développement faire un travail précis et clair afin de faciliter le travail de la personne qui le reprendra ou pour les améliorations avenir comme un système de paiement, offres promotionnels pour les clients etc ...

Ce projet m’auras permit d’apprendre davantage sur un langage qui est en pleine évolution et que l'intérêt pour ce langage ne cesse d’augmenter depuis 2011 jusqu'au jour d'aujourd'hui car node JS peut effectivement être utilisé comme serveur sans avoir besoin d’une solution externe tels que Apache ou lighttpd, il reste avant tout une plateforme de développement JavaScript permettant de répondre à de nombreux besoins clients/serveurs grâce à la mise à disposition d’une API.



Npm(Node Package Manager) : Est un gestionnaire de paquet. L’usage le plus courant que l’on en fait est l’installation de dépendances depuis *la plateforme npm* .

npm install <nom\_du\_paquet>.

Logique et simple, il en va de même pour la suppression :

npm uninstall <nom\_du\_paquet>.

promise : Et bien comme son nom l'indique vous pouvez voir ça comme la promesse que vous allez recevoir une valeur. Comme toute promesse, elle peut être tenue, la valeur est arrivée et on peut s'en servir, ou ne pas l'être, dans ce cas une erreur arrive et on peut réagir en conséquence. Ce mécanisme permet de remplacer les callbacks d'une manière plus élégante.

Une promesse peut avoir plusieurs états au cours de son existence :

* en cours : la valeur qu'elle contient n'est pas encore arrivée
* résolue : la valeur est arrivée, on peut l'utiliser
* rejetée : une erreur est survenue, on peut y réagir

Une promesse possède 2 fonctions : then et catch, then pour récupérer le résultat ou l'erreur d'une promesse et catch pour récupérer l'erreur d'une ou plusieurs promesses.

bdd : base de données.

rest(representational state transfer**)** : Nous utiliserons la seconde règle de l’architecture REST qui est d’utiliser les verbes HTTP existant plutôt que d’inclure l’opération dans l’URI de la ressource. Ainsi, généralement pour une ressource, il y a 4 opérations possibles (CRUD).

HTTP propose les verbes correspondant :

* Créer (create) => POST
* Afficher (read) => GET
* Mettre à jour (update) => PUT
* Supprimer (delete) => DELETE

Hasher **:** Prévenir les accès frauduleux aux comptes des utilisateurs en cas de faille de sécurité. Le but est donc simple : que cela soit plus coûteux pour un hacker de décoder les mots de passe.