Projet de Web Avancé : Hotel

Sommaire

[1. Introduction 3](#_Toc413407642)

[2. Présentation (Analyse des besoins) 3](#_Toc413407643)

[1) Le contexte du projet 3](#_Toc413407644)

[2) L'équipe 3](#_Toc413407645)

[3. Backlog du produit 3](#_Toc413407646)

[4. Les Spécifications fonctionnelles et techniques 3](#_Toc413407647)

[1) Le diagramme de cas d'utilisations 4](#_Toc413407648)

[2) Le diagramme de classe 4](#_Toc413407649)

[3) Merise 5](#_Toc413407650)

[5. L'environnement technique (Spécification technique) 5](#_Toc413407651)

[6. Le développement de l'application 5](#_Toc413407652)

[1. Le fichier "app.js" 6](#_Toc413407653)

[2. Le dossier "routes" 7](#_Toc413407654)

[3. Le dossier "views" 9](#_Toc413407655)

[4) PhpMyAdmin 10](#_Toc413407656)

[7. Versions 11](#_Toc413407657)

[8. Tests 11](#_Toc413407658)

[9. Conclusion 12](#_Toc413407659)

[10. Annexes 12](#_Toc413407660)

[1) Installation de Nodejs Express Project dans Eclipse 12](#_Toc413407661)

[2) Configuration des couleurs d'affichage des syntaxe EJS en syntaxe HTML (sur Eclipse) 12](#_Toc413407662)

[3) Documentations et ressources 13](#_Toc413407663)

# Introduction

Au cours du cursus d'Ingésup en Bachelor 3, nous sommes amenés de réaliser un projet d'application web sous thème d'un Hôtel en utilisant les derniers technologies tel que clojure ou Node.js (ou même d'autres technologies comme PHP5, Django, Python, etc.).

# Présentation (Analyse des besoins)

## Le contexte du projet

Le but du projet est de réaliser une application ou un site web sur le thème d'un hôtel avec les gestions des chambres, des réservations et des clients. Chaque gestion va nous permettre de créer, de modifier, de supprimer ainsi que lister les données. la méthodologie pour la gestion de ce projet est l'utilisation de la méthode agile SCRUM.

## L'équipe

Le projet est réalisé par un groupe de 4 étudiants d'Ingésup en Bachelor 3 ainsi que leur rôle en sein du groupe:

* Tran Thierry , le SCRUM Master
* Yannick Noël
* Jean-François Cadule
* Franck Procureur, le rôle du Client

2 sur 4 étudiant prennent les rôles du MOA (le chef d'équipe, la relation avec le client , le client). Cependant, les quatres restent quand même les MOE, c'est-à-dire les développeurs, les concepteurs.

# Backlog du produit

|  |  |  |  |
| --- | --- | --- | --- |
| Besoins | Priorisation | Charges | % avancement |
| Manipulation des réservation des clients | 2 | 1 | 80% |
| Gestion des clients | 1 | 1 | 100% |
| Gestion des chambres | 3 | 1 | 100% |
| Interface graphique simple , intuitive | 4 | 1 | 80% |

# Les Spécifications fonctionnelles et techniques

Nous allons détailler les différents spécifications du projet sur le système Hôtel avec ses acteurs , ses classes et ses tables à travers des diagrammes UML(cas d'utilisations, classes, de base de donnée).

## Le diagramme de cas d'utilisations

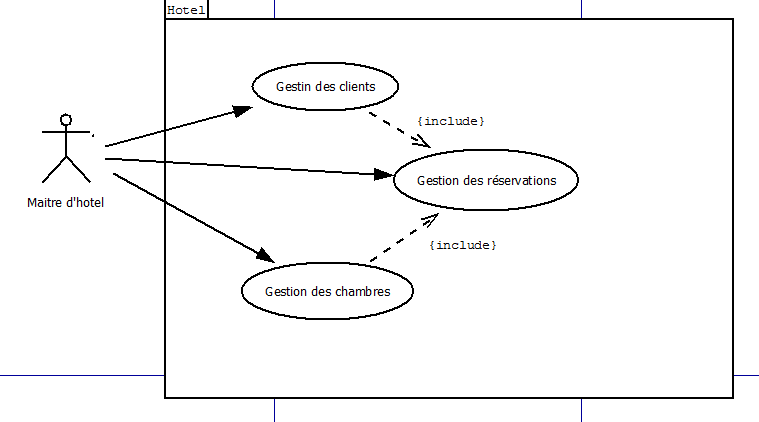


Figure - Diagramme de cas d'utilisations

## Le diagramme de classe

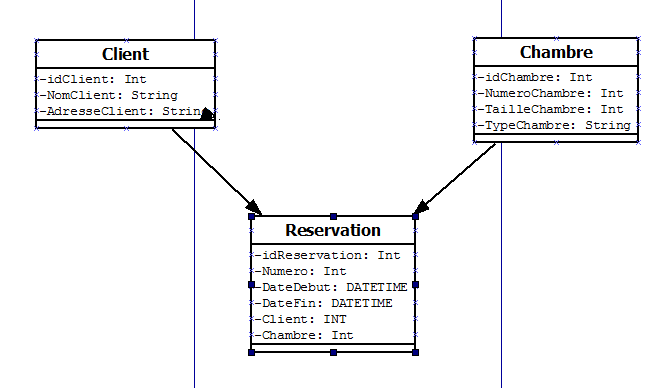


Figure - Le diagramme de classe

## Merise

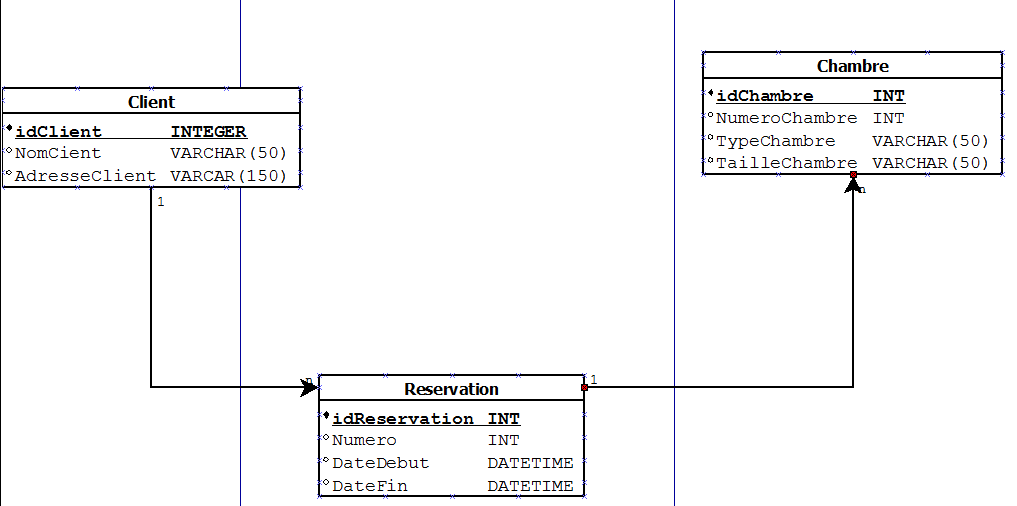


Figure - Merise

# L'environnement technique (Spécification technique)

Nous avions développé l'application avec nos propres ordinateurs en environnement Windows et Mac OS. Le choix du logiciel est l'IDE Eclipse avec sa facilité d'organisation et ses extensions pour le projet. Avec les choix données par l'enseignant , nous avions choisi Node.js , le langage JavaScript, le framework Express.js avec le template EJS , et les langages web (HTML, CSS) . Pour réaliser les configurations pour la réalisation du projet , voir la partie **annexe**. Pour la gestion de la base de données , on utilise MySQL avec l'interface graphique phpmyadmin . Pour le serveur APACHE, on avait le choix avec Easy-Php ou Wamp. Pour la présentation, on a utilisé Boostrap.

En résumé, les technologies qu'on va utiliser sont

* Nodejs
* Express.js
* IDE Eclipse version J2EE (ou SublimeText)
* les templates EJS
* MySQL
* Serveur Apache : EasyPhp ou Wamp
* Bootstrap
* HTML, CSS, JavaScripts

# Le développement de l'application

L'application est structuré par rapport au modèle MVC (Modèle, Vue, Contrôleur) comme montre l'arborescence des dossiers suivante:

# arborescence.png

Figure - L'arborescence des fichiers

## Le fichier "app.js"

Le fichier "app.js" permet de gérer l'URL du navigateur web. Elle permet de gérer les chemins vers quelle partie du site qu'on veut aller. Si vous avez suivi la procédure d'installation de Express.js dans Eclipse, il crée automatique ce fichier lorsque vous créer un **nouveau projet Nodejs Express Project**. Des modules étaient installés par défaut comme http, path, express et app.

Comme dit précédemment, le fichier gère les routes. Alors, pour chaque gestion (client,c hambre, réservation), on ajoute ces fonctions:

**var** client = require('./routes/client');

**var** chambre = require('./routes/chambre');

**var** reservation = require('./routes/reservation');

//les routes en faisant pareil pour chambre et réservation

app.get('/client', client.list);

app.get('/client/add', client.add);

app.post('/client/add', client.save);

app.get('/client/edit/:id', client.edit);

app.post('/client/edit/:id', client.save\_edit);

app.get('/client/delete/:id', client.delete\_save);

Les fonctions telles que list,add,edit sont des fonctions exportés à partir du fichier .js du module comme client.js. Ce fichier se situe dans le dossier qu'on verra après en détails avec les fonctions.

"app.js" gère aussi la connectivité avec la base de données en utilisant ces modules:

**var** mysql = require('mysql');

**var** connexion = require('express-myconnection');

// Connexion avec la base de données

app.use(connexion(mysql, {

host : '127.0.0.1',

user : 'root',

password : '',

port : 3306,

database : 'hotel'

}, 'pool'));

Indication: la fonction app.use(app.router) doit être après les app.get()et app.post().

## Le dossier "routes"

Dans le dossier "routes", il est composé des fichiers permettant de gérer les requêtes HTTP et les envoyé des réponses (données) vers des fichiers dans le dossier "views".

Chaque gestion (Client, Chambre, Réservation) possède des fonctions qui sont similaires mais qui se diffèrent par des paramètres par rapport aux attributs de la base de données et/ou le formulaire.

Les fonctions doivent être précédés par le mot-clé **exports** . Voici la liste des fonctions nécessaires pour notre projet

* list : afficher les données

exports.list = **function**(req, res) {

req.getConnection(**function**(err, connexion) {

**var** query = connexion.query('SELECT \* FROM client',

**function**(err, rows) {

**if** (err)

console.log("Error Selecting : %s ", err);

res.render('client', {

page\_title : "Page du Client",

data : rows

});

});

});

};

* add : afficher le formulaire de l'enregistrement

exports.add = **function**(req, res) {

res.render('add\_client',{page\_title:'Hotel - Ajout d\'un Client'});

};

* save : enregistrement des données d'un formulaire

exports.save = **function**(req,res){

**var** input = JSON.parse(JSON.stringify(req.body));

req.getConnection(**function**(err,connexion){

**var** data = {

NomClient:input.nomClient,

AdresseClient:input.adresseClient

};

**var** query = connexion.query("INSERT INTO Client SET ?",data, **function**(err,rows){

**if**(err)

**throw** err;

res.redirect('/client');

});

});

};

* edit : afficher le formulaire pour modification d'une donnée

exports.edit = **function**(req,res){

**var** id = req.params.id;

req.getConnection(**function**(err,connexion){

**var** query = connexion.query('SELECT \* FROM client WHERE idClient=?',[id],**function**(err,rows){

**if** (err) **throw** err;

res.render('edit\_client',{page\_title:'Modification des données ',data:rows});

});

});

};

* save\_edit : enregistrement des modifications des données

exports.save\_edit = **function**(req,res){

**var** input = JSON.parse(JSON.stringify(req.body));

**var** id = req.params.id;

req.getConnection(**function**(err,connexion){

**var** data = {

NomClient:input.nomClient,

AdresseClient:input.adresseClient

};

**var** query = connexion.query("UPDATE Client SET ? WHERE idClient=?",[data,id], **function**(err,rows){

**if**(err)

**throw** err;

res.redirect('/client');

});

});

};

* delete\_save : suppression d'une donnée

exports.delete\_save = **function**(req,res){

**var** id = req.params.id;

req.getConnection(**function**(err,connexion){

**var** query = connexion.query("DELETE FROM Client WHERE idClient=?",[id],**function**(err,rows){

**if**(err) **throw** err;

res.redirect('/client');

});

});

};

Dans les requêtes SQL, vous pouvez les modifier le nom de table par celui que vous voulez interagir.

## Le dossier "views"

Le dossier "views" est principalement constitué des fichiers .ejs. La syntaxe de ces fichiers est proche du HTML avec le PHP et le symphony2 . Cependant, Symphony et Express ne sont pas la même chose en terme de fonctionnalités et d'outils.

Pour le formulaire d'enregistrement , vous devez taper ces lignes de codes dans le fichier **add\_client.js** par exemple

<%- include layouts/header.ejs %>

<div class=*"page\_data"*>

<form method=*"post"* action=*"/client/add"*>

<fieldset>

<legend>Ajout d'un client</legend>

<table>

<tr>

<td>Nom d'un client</td>

<td><input type=*"text"* name=*"nomClient"* /></td>

</tr>

<tr>

<td>Adresse d'un client</td>

<td><input type=*"text"* name=*"adresseClient"* /></td>

</tr>

<tr>

<td><input type=*"submit"* name=*"save"* value=*"Enregistrer"*/></td>

<td><input type=*"button"* name=*"cancel"* value=*"Annuler"* class=*"cancel"*

onClick="cancelClient()"></td>

</tr>

</table>

</fieldset>

</form>

</div>

<%- include layouts/footer.ejs %>

Ensuite, pour le formulaire de modification, dans le fichier **edit\_client.ejs**

<%- include layouts/header.ejs %>

<div class=*"page-data"*>

<form method=*"post"* action=*"/client/edit/<%=data[0].idClient%>"*>

<table cellpadding=*"11"*>

<tr>

<td>Nom du Client :</td>

<td><input type=*"text"* name=*"name"*

value=*"<%=data[0].NomClient%>"* required></td>

</tr>

<tr>

<td valign=*"top"*>Adresse du Client :</td>

<td valign=*"top"*><textarea name=*"address"* required cols=*"30"*

rows=*"5"*><%=data[0].AdresseClient%></textarea></td>

</tr>

<tr>

<td>&nbsp;</td>

<td>&nbsp;<input type=*"submit"* name=*"submit"* value=*"Save"*>

<input type=*"button"* name=*"cancel"* value=*"Cancel"* class=*"cancel"*

onClick="cancelClient()">

</td>

</tr>

</table>

</form>

</div>

<%- include layouts/footer.ejs %>

Le header contient les balises <!DOCTYPE html>,html, head, et body. Quant à le footer, il contient les fin des div , body et html.

pour l'affichage des données sous forme de table, dans le fichier **client.ejs**

<div class=*"data-table"*>

<table border=*"1"* cellpadding=*"7"* cellspacing=*"7"*>

<tr>

<th width=*"50px"*>No</th>

<th>Name</th>

<th>Address</th>

<th width=*"120px"*>Action</th>

</tr>

<% if(data.length){ for(var i = 0;i < data.length;i++) { %>

<tr>

<td><%=(i+1)%></td>

<td><%=data[i].NomClient%></td>

<td><%=data[i].AdresseClient%></td>

<td><a class=*"a-inside edit"*

href=*"../client/edit/<%=data[i].idClient%>"*>Edit</a> <a

class=*"a-inside delete"*

href=*"../client/delete/<%=data[i].idClient%>"*>Delete</a></td>

</tr>

<% } }else{ %>

<tr>

<td colspan=*"3"*>Pas de clients</td>

</tr>

<% } %>

</table>

## PhpMyAdmin

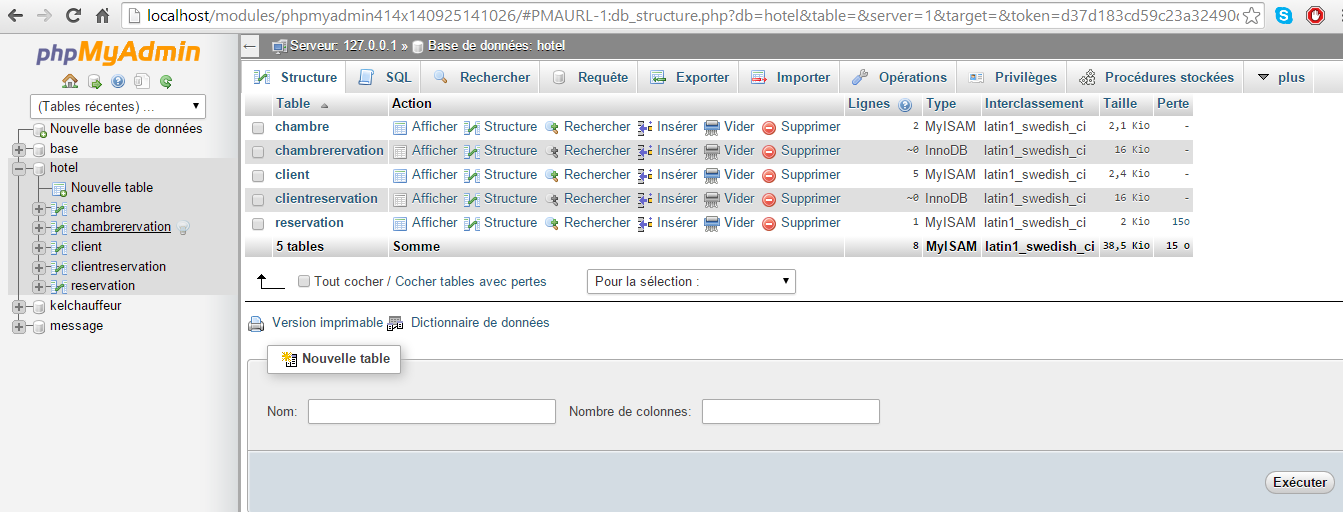


Figure - Phpmyadmin

Dans l'onglet SQL, on peut exécuter des requêtes SQL tels que INSERT, DELETE, DROP, ALTER TABLE,etc.

# Versions

Pour chaque version, on va détailler quelles sont les fonctionnalités qui sont actives:

* Version 1.0:
* Gestion des clients
* Gestion des chambres
* Gestion des réservations
* Interface graphique simple

# Tests

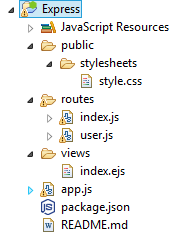
|  |  |  |  |
| --- | --- | --- | --- |
| **Versions** | **Tâches** | **Fonctionnalités** | **Résultats** |
| 1.0 | Gestion des clients | Création d'un client | Enregistrement effectué |
|  |  | Suppression d'un client | Suppression effectué |
|  |  | Modification d'un client | Opérationnel |
|  |  | Lister les Clients | Liste affichée |
|  | Gestion des chambres | Création d'une chambre | Enregistrement effectué |
|  |  | Suppression d'une chambre | Suppression effectué |
|  |  | Modification d'une chambre | Opérationnel |
|  |  | Lister les chambres | Liste affichée |
|  | Gestion des réservations | Enregistrement d'une réservation | Enregistrement effectué |
|  |  | Suppression d'une réservation | Suppression effectué |
|  |  | Modification d'une réservation | Opérationnel |
|  |  | Liste des réservation | Liste affichée |
|  | Interface graphique | Page des clients | Opérationnel |
|  |  | Page des chambres | Opérationnel |
|  |  | Page des réservations | Opérationnel |
|  |  | Page d'accueil | Opérationnel |

# Conclusion

Le projet a été l'opportunité d'employer, de tester et de maitriser les langages et les technologies web comme NodeJs avec les framework Express.js. De plus, il nous a permis d'utiliser la méthodologie SCRUM pour avoir des livrables au plus vite après chaque sprint malgrés le temps imparti du projet.

# Annexes

### Installation de Nodejs Express Project dans Eclipse

* Télécharger NodeEclipse Enide sur le site : http://www.nodeclipse.org/enide/
* Télécharger Nodejs et l'installer .
* Après télécharger NodeEclipse, allez sur Eclipse->Help->Install New SoftWare->Add->Archive->le path du dossier téléchargé après avoir décompressé->OK
* Après Installation , normalement , il y a dans File-->New-->Node.js Express Project
* Après choisi Express , vous nommez votre projet avec le nom de votre choix. Pour notre projet , on le nomme sobrement Hotel.
* Il faut choisir le type de template. Notre choix était donc EJS.
* Ensuite , appuyez sur la touche *Entrée*, Eclipse génère des dossiers et des fichiers ci-dessous
* Fin de l'installation . 

## Configuration des couleurs d'affichage des syntaxe EJS en syntaxe HTML (sur Eclipse)

* Allez ***Windows->Preferences***
* Puis, allez ***General->Editor->Content Types***
* Cliquez sur ***Text***->***HTML***
* Cliquez sur le bouton***Add***puis ajoutez l'extension **\*.ejs** puis OK
* Enfin , appuyez sur ***OK***

## Documentations et ressources

* NodeJS : http://nodejs.org/
* IDE Eclipse : https://eclipse.org/downloads/
* SublimeText : http://www.sublimetext.com/
* EasyPHP: http://www.easyphp.org/
* Wamp: http://www.wampserver.com/
* Bootstrap : http://getbootstrap.com/