

**Travail de Diplôme ES 2020**

**Constantin Herrmann**

Avril – Juin 2020

**M. Francisco Garcia**

CFPT-I Technicien ES

Version 1.0

# Table de matières

[1 Table de matières 1](#_Toc38714684)

[2 Résumé 2](#_Toc38714685)

[3 Abstract 3](#_Toc38714686)

[4 Introduction 4](#_Toc38714687)

[5 Analyse fonctionnelle 4](#_Toc38714688)

[6 Analyse organique 5](#_Toc38714689)

[6.1 Mise en place 5](#_Toc38714690)

[6.1.1 GitHub 5](#_Toc38714691)

[6.1.2 Trello 5](#_Toc38714692)

[6.2 Post-travail 5](#_Toc38714693)

[6.2.1 Programmation 5](#_Toc38714694)

[6.2.2 Installation de react 6](#_Toc38714695)

[6.2.3 Organisationnel 6](#_Toc38714696)

[6.2.4 Environnement 6](#_Toc38714697)

[6.2.4.1 Laragon 6](#_Toc38714698)

[6.2.4.2 Visual Studio Code 6](#_Toc38714699)

[6.2.4.3 EDUGE 6](#_Toc38714700)

[6.2.4.4 Github Desktop 7](#_Toc38714701)

[6.2.5 Base de données 7](#_Toc38714702)

[6.2.5.1 UML 7](#_Toc38714703)

[6.2.5.2 Privilèges 7](#_Toc38714704)

[6.2.5.3 Structure 7](#_Toc38714705)

[6.2.6 Données de tests 8](#_Toc38714706)

[6.2.6.1 Utilisateurs 8](#_Toc38714707)

[6.2.6.1.1 Administrateur système : 8](#_Toc38714708)

[6.2.6.1.2 Manager du restaurant : 8](#_Toc38714709)

[6.2.6.1.3 Employé 1 : 8](#_Toc38714710)

[6.2.6.1.4 Employé 2 : 8](#_Toc38714711)

[6.3 API 9](#_Toc38714712)

[7 Glossaire 10](#_Toc38714713)

# Résumé

# Abstract

# Introduction

De nos jours, il devient de plus en plus facile pour une personne de réserver une table dans un restaurant, mais toutes ses applications que nous utilisons ne sont pas optimisées entièrement pour les restaurateurs. C’est pourquoi, avec l’aide de M. Garcia et de Mme. Perdrizat (gérante du restaurant « l’Atelier » à Genève), nous avons décidé de revoir entièrement le fonctionnement d’une application de gestion de réservation mais en concentrant nos efforts sur le restaurateur.

Cette application permettra donc facilement au restaurateur de gérer ses réservations, mais surtout son établissement.

# Analyse fonctionnelle

# Analyse organique

## Mise en place

### GitHub

Afin d’avoir un suivi constant de mon projet, j’ai décidé de créer un GitHub. Dans ce github j’ai donc régulièrement mis à jour le code et la documentation.

Le github est structuré de la manière suivante :

Travail\_Diplome\_ES\_2020

├── Documentation

├── Tests

├── RESA

├── README.md

├── logbook.md

Le dossier RESA contient tout le code source de l’application.

### Trello

Trello est un système de gestion du temps qui permet de facilement créer, déplacer et terminer des tâches.

J’ai créé 5 colonnes :

1. A faire
2. En cours
3. En validation
4. Terminés
5. En continu

La colonne 3 « En validation » ce sont les tâches terminées qui demande une validation de la part de M. Garcia afin de pouvoir classer la tâche dans la colonne terminée. La colonne 5 « En continu », représente la colonne des tâches que je dois suivre en continu (ex. le journal de bord).

## Post-travail

### Programmation

Afin de pouvoir réaliser au mieux mon travail, j’ai dû rechercher les langages de programmation et des librairies qui vont m’aider le mieux que possible à réaliser le travail qui m’est demandé dans le cahier des charges.

La bibliothèque JavaScript qui m’a semblée le plus adaptée à mes besoins est celle de « React ». En effet, React est une bibliothèque Javascript pensée pour la création d’interfaces utilisateurs.

« React est une bibliothèque JavaScript déclarative, efficace et flexible pour construire des interfaces utilisateurs (UI). Elle vous permet de composer des UI complexes à partir de petits morceaux de code isolés appelés « composants ». » - React

React fonctionne à base de « **composants** » qui peuvent prendre de propriétés nommées « **props** ». Ce composant renvoie une arborescence de vues à afficher via la méthode « **render** »

### Installation de react

Tout d’abord je dois disposer d’une mise à jour récente de Node.js. Afin de créer une application de test, je dois entrer la commande suivante dans le dossier ou je souhaite créer l’application :

npx create-react-app my-app

“my-app” représente le nom de l’application

Une fois l’application crée, on obtient un dossier contenant l’architecture suivante

my-app

├── README.md

├── node\_modules

├── package.json

├── .gitignore

├── public

│ ├── favicon.ico

│ ├── index.html

│ └── manifest.json

└── src

├── App.css

├── App.js

├── App.test.js

├── index.css

├── index.js

├── logo.svg

└── serviceWorker.js

Le dossier « src » contient tout le code de l’application en tant que tel, c’est-à-dire les pages html, js, etc.

### Organisationnel

Afin de mieux comprendre les besoins du client, nous avons décidé avec M. Garcia d’aller sur les lieux afin de discuter avec la gérante. Lors de cette discussion nous avons donc pu mettre au clair les points qui jusqu’à la, étaient encore flous.

### Environnement

#### Laragon

Afin de pouvoir développer et tester mon application sur mon poste de travail, j’ai décidé d’utiliser l’application Laragon. Celle-ci me permet également d’avoir une base de données phpMyAdmin.

J0ai décidé d’utiliser Laragon, car au cours des cinq dernières années j’ai eu l’occasion de l’utiliser en plus de EasyPHP et Xamp. Laragon fut le seul à fonctionné « out of the box » et sans aucun problème.

#### Visual Studio Code

Visual Studio Code me permet de facilement accéder au code stocker sur mon github. Il me permet également de voir en temps réel mes fichiers markdown avant de les publier sur github.

#### EDUGE

Je fais un backup de mon projet tous les deux jours sur mon drive EDUGE afin de répondre aux demandes de mon enseignant sur mon évaluation. Si j’ai choisi EDUGE, c’est pour la raison que c’est une plateforme stable et fonctionnelle qui me permet de facilement partager des fichiers avec mon enseignant.

#### Github Desktop

Ce logiciel me permet de facilement pouvoir mettre à jour le github avec mes fichiers stockés en local. Lorsqu’une modification dans un fichier est faite, github le détectes automatiquement et me propose de faire un nouveau commit.

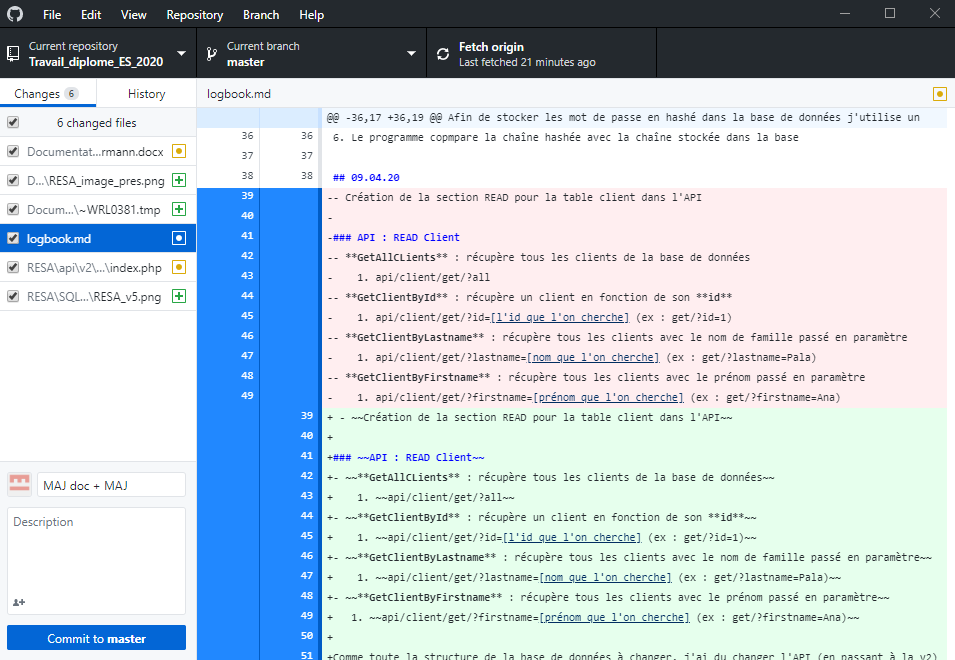


Figure 1 : Interface de Github Desktop

### Base de données

Afin de pouvoir stocker les données, j’ai créé une base de données nommée « resa ».

#### UML

Afin de pouvoir facilement créer le model UML de la base de données, je suis passé par le site dbdiagram.io. Ce site permet de facilement et dynamiquement créer des modèles UML qui sont par la suite facilement exportable en fichier SQL afin des les ajouter dans notre base de données.

#### Privilèges

Pour accéder à la base de données, il faut utiliser les privilèges suivants :

* Username : **resa\_tech\_es**
* Password : **WhutMerYmZeR6EHb**

#### Structure

Ma base de données est structurée de la manière suivante :

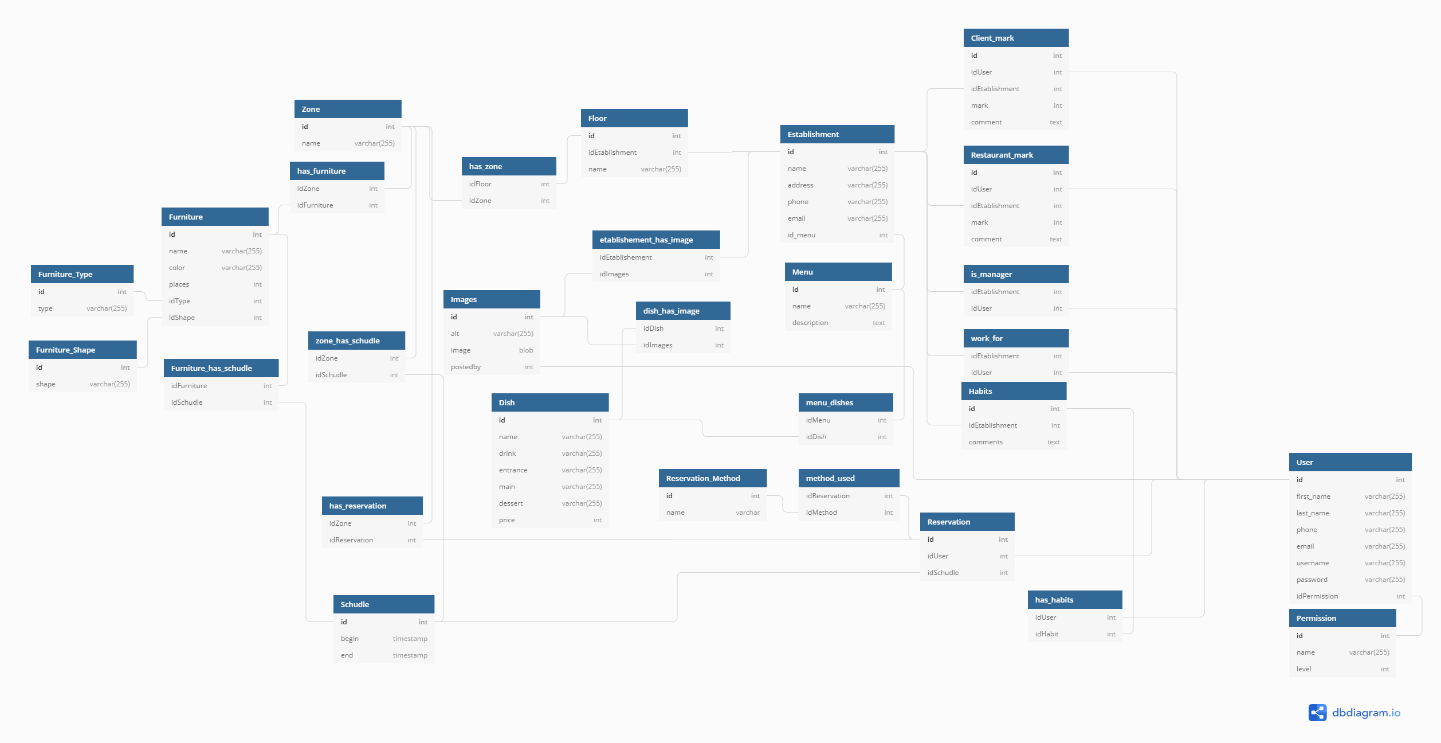


Figure 2 : la structure de la base de données en V5

### Données de tests

#### Utilisateurs

Pour tester mon api, j’ai créé des utilisateurs, des restaurants et toutes les données dont j’avais besoin afin d’effectuer des tests. Voici la liste des utilisateurs :

|  |  |
| --- | --- |
| Administrateur système :  * Username : 2008 * Password : admin | Manager du restaurant :  * Username : 3383 * Password : manager |
| Employé 1 :  * Username : 5243 * Password : e1 | Employé 2 :  * Username : 9902 * Password : e2 |

#### Etablissements

J’ai également créé des établissements, qui possèdent chacun un nombre différent d’images, d’employés, de menus et de plats.

* Port Martignot
* Beau-Rivage
* Les Clochettes

## API

Avec l’objectif d’accéder à mon API à distance ou depuis différents supports, j’ai dû mettre en place une API afin de communiquer avec ma base de données.

### Structure

L’API v2 est structurée de tel manière à ce que les informations soient faciles à trouvées (REF : 6.3.3 Cheat Sheet de l’API) pour plus de détails sur le fonctionnement de l’API.

Voici la structure :

### Gestion des images

### Cheat Sheet de l’API

# Glossaire