

**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 2](#_Toc39585685)

[2 Résumé (intermédiaire) 5](#_Toc39585686)

[3 Abstract (intermediary) 5](#_Toc39585687)

[4 Introduction 6](#_Toc39585688)

[5 Analyse de l’existant 6](#_Toc39585689)

[5.1 La fourchette Verte 6](#_Toc39585690)

[5.1.1 Clients 6](#_Toc39585691)

[5.1.1.1 Interface 6](#_Toc39585692)

[5.1.1.2 Fonctionnalités clés 6](#_Toc39585693)

[5.1.2 Restaurateur 6](#_Toc39585694)

[5.1.2.1 Interface 6](#_Toc39585695)

[5.1.2.2 Fonctionnalités clés 6](#_Toc39585696)

[6 Analyse fonctionnelle 7](#_Toc39585697)

[6.1 Interfaces 7](#_Toc39585698)

[6.2 Fonctionnalités 7](#_Toc39585699)

[6.2.1 Analyse globale 7](#_Toc39585700)

[6.2.2 Analyse détaillée 7](#_Toc39585701)

[7 Analyse organique 8](#_Toc39585702)

[7.1 Mise en place 8](#_Toc39585703)

[7.1.1 GitHub 8](#_Toc39585704)

[7.1.2 Trello 8](#_Toc39585705)

[7.2 Post-travail 8](#_Toc39585706)

[7.2.1 Programmation 8](#_Toc39585707)

[7.2.2 Installation de React 9](#_Toc39585708)

[7.2.3 Conventions 9](#_Toc39585709)

[7.2.3.1 En-têtes 9](#_Toc39585710)

[7.2.3.2 Diagrammes d’activités 9](#_Toc39585711)

[7.2.4 Organisationnel 9](#_Toc39585712)

[7.3 Environnement 9](#_Toc39585713)

[7.3.1 Laragon 9](#_Toc39585714)

[7.3.2 Visual Studio Code 10](#_Toc39585715)

[7.3.3 EDUGE 10](#_Toc39585716)

[7.3.4 Github Desktop 10](#_Toc39585717)

[7.4 Diagrammes d’activités 11](#_Toc39585718)

[7.4.1 Diagrammes d’activités de RESA 11](#_Toc39585719)

[7.4.1.1 Création de compte 12](#_Toc39585720)

[7.5 Base de données 12](#_Toc39585721)

[7.5.1 UML 12](#_Toc39585722)

[7.5.2 Privilèges 13](#_Toc39585723)

[7.5.3 Structure 13](#_Toc39585724)

[7.5.4 Données de tests 14](#_Toc39585725)

[7.5.4.1 Utilisateurs 14](#_Toc39585726)

[7.5.4.2 Etablissements 14](#_Toc39585727)

[7.6 API 15](#_Toc39585728)

[7.6.1 Structure 15](#_Toc39585729)

[7.6.2 Variables globales 14](#_Toc39585730)

[7.6.3 Gestion des images 14](#_Toc39585731)

[7.6.3.1 Mise en ligne d’une image 14](#_Toc39585732)

[7.6.3.1.1 Exemple 14](#_Toc39585733)

[7.6.3.2 Récupérer les images 15](#_Toc39585734)

[7.6.3.2.1 Exemples 15](#_Toc39585735)

[7.6.4 Cheat Sheet de l’API 16](#_Toc39585736)

[7.6.5 User 16](#_Toc39585737)

[7.6.5.1 Lecture 16](#_Toc39585738)

[7.6.5.2 Divers 18](#_Toc39585739)

[7.6.6 Floor 18](#_Toc39585740)

[7.6.6.1 Lecture 18](#_Toc39585741)

[7.6.6.2 Création 18](#_Toc39585742)

[7.6.7 Schedule 19](#_Toc39585743)

[7.6.7.1 Lecture 19](#_Toc39585744)

[7.6.8 Images 19](#_Toc39585745)

[7.7 Gestion du temps 22](#_Toc39585746)

[7.7.1 Lister les tâches 22](#_Toc39585747)

[7.7.2 Classer les tâches 22](#_Toc39585748)

[7.7.3 Séparation des tâches client et serveur 23](#_Toc39585749)

[7.7.4 Conclusion 23](#_Toc39585750)

[7.8 Raisonnements 23](#_Toc39585751)

[7.8.1 Réflexions personnelles 23](#_Toc39585752)

[7.8.1.1 Le journal de bord 24](#_Toc39585753)

[7.8.1.1.1 Structure 24](#_Toc39585754)

[7.8.1.1.2 Extrait du journal de bord lors de réflexions 24](#_Toc39585755)

[7.8.1.2 Analyse 24](#_Toc39585756)

[7.8.2 Communications avec M. Garcia 24](#_Toc39585757)

[8 Glossaire 25](#_Toc39585758)

# Résumé (intermédiaire)

De nos jours, les gérants de restaurants n’ont pas réellement de solutions de gestion pour leur restaurant. La réservation, la gestion de la salle, des horaires et du personnel n’est souvent pas unie dans une seule application. C’est là qu’entre en jeu RESA.

Avec son application intuitive pour les gérants, les serveurs et les clients, RESA offre la possibilité aux gérants de facilement créer leur établissement, leurs salles, leurs horaires et les réservations, aux clients de facilement voir les restaurants et faire des réservations.

RESA est une application web sur le langage de PHP qui se repose sur les services de son API.

Ce document ressens tout le projet a travers une analyse fonctionnelle et organique qui décrit précisément le processus de création, de développement et d’analyse des éléments clés du projet.

# Abstract (intermediary)

These days, restaurant managers do not really have management solutions for their restaurants. Booking, room, schedule and staff management is often not united in a single application. This is where RESA comes in.

With its intuitive application for managers, waiters and customers, RESA makes it easy for managers to create their establishment, rooms, schedules, and reservations, and for customers to easily view restaurants and make reservations.

RESA is a PHP-based web application that relies on the services of its API.

This document feels the whole project through a functional and organic analysis that describes precisely the process of creation, development and analysis of the key elements of the project.

# 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 de l’existant

## La fourchette Verte

La fourchette verte est l’application qui se rapproche le plus de RESA. En effet, cette dernière regroupe tous les principaux domaines de la restauration. Elle dispose d’un site internet et d’une application mobile et tablette.

Le lien vers le site internet : <https://www.lafourchette.ch/>

### Clients

#### Interface

#### Fonctionnalités clés

### Restaurateur

#### Interface

#### Fonctionnalités clés

# Analyse fonctionnelle

## Interfaces

## Fonctionnalités

### Analyse globale

### Analyse détaillée

# 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.

### Conventions

#### En-têtes

#### Diagrammes d’activités

### 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.

C’est également sur cette plateforme que j’effectue quotidiennement des backups de mon travail.

### 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.

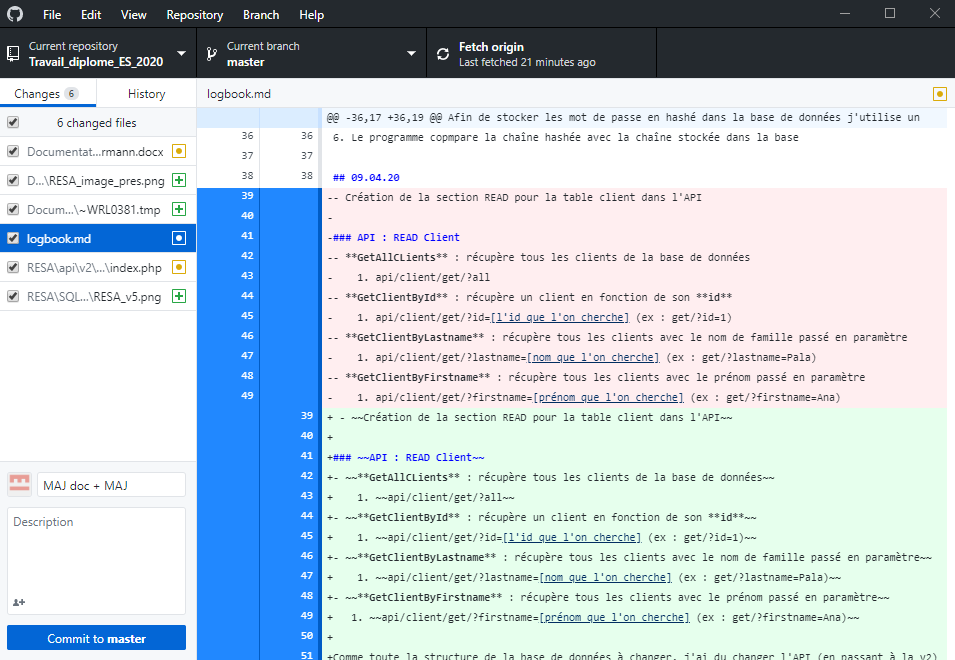


Figure 1 : Interface de Github Desktop

## Diagrammes d’activités

Pour la création des diagrammes d’activités, j’ai d’abord utilisé le site internet « dbdiagram.io », mais lorsque monsieur Garcia as vérifié que je partais bien dans le bon sens, nous nous sommes aperçus que je n’avais pas respecté les normes pour la création de diagrammes. C’est à ce moment-là que nous avons décidé de tous les refaire à partir du site draw.io[[1]](#footnote-1).

Afin d’être sûr de respecter les normes de diagrammes d’activités, je me suis fier au site nommé Sourcemaking[[2]](#footnote-2). Ce site explique clairement et facilement tous les objets, lignes et intersections d’un diagramme.

### Diagrammes d’activités de RESA

En gardant en tête l’objectif principale de la réservation, j’ai créé des diagrammes d’activités afin de mieux me représenter les tâches, fonctionnalités et vues que je devais développer.

#### Création de compte

Figure 2 Diagramme d'activité, création de compte

## Base de données

Afin de pouvoir stocker les données, j’ai créé une base de données nommée « resa ». Cette base de données me permet d’enregistrer toutes les données qui sont nécessaires au bon fonctionnement de l’application web.

### UML

Afin de pouvoir facilement créer le model UML de la base de données, je suis passé par le site dbdiagram.io[[3]](#footnote-3). Ce site permet de facilement et dynamiquement créer des modèles UML qui sont par la suite facilement exportable en fichier SQL afin de les ajouter dans notre base de données.

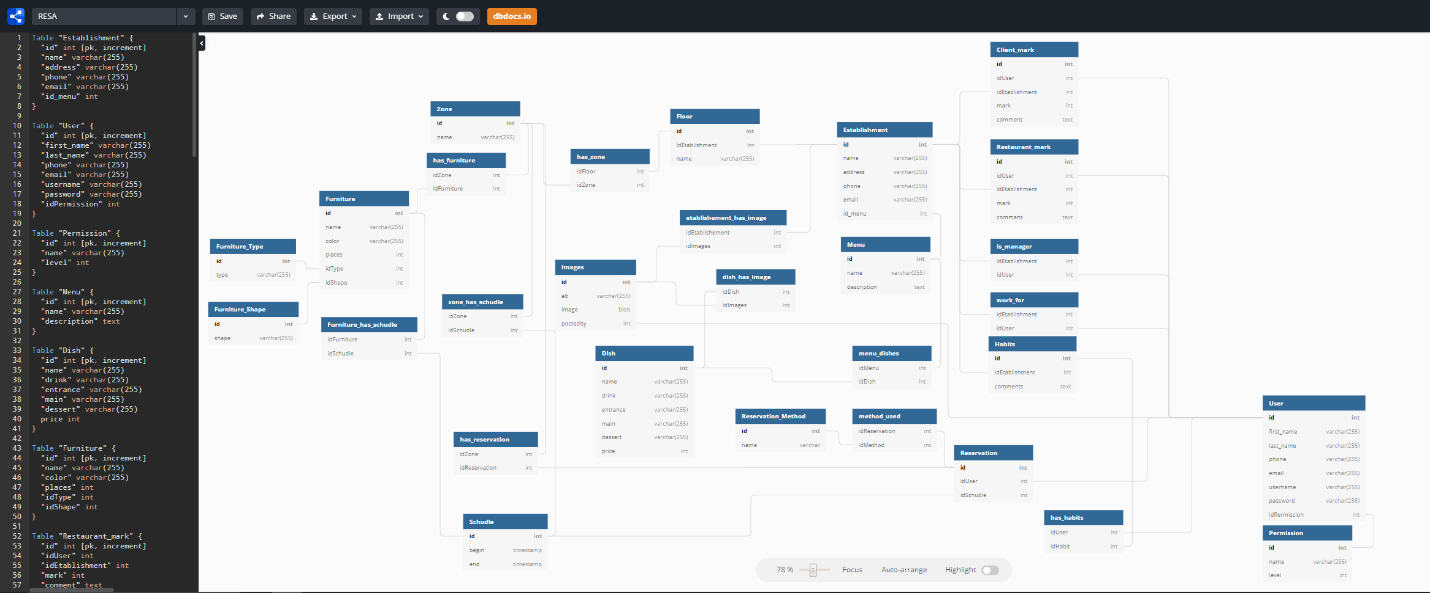


Figure 3 Interface de dbdiagram

### 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 :

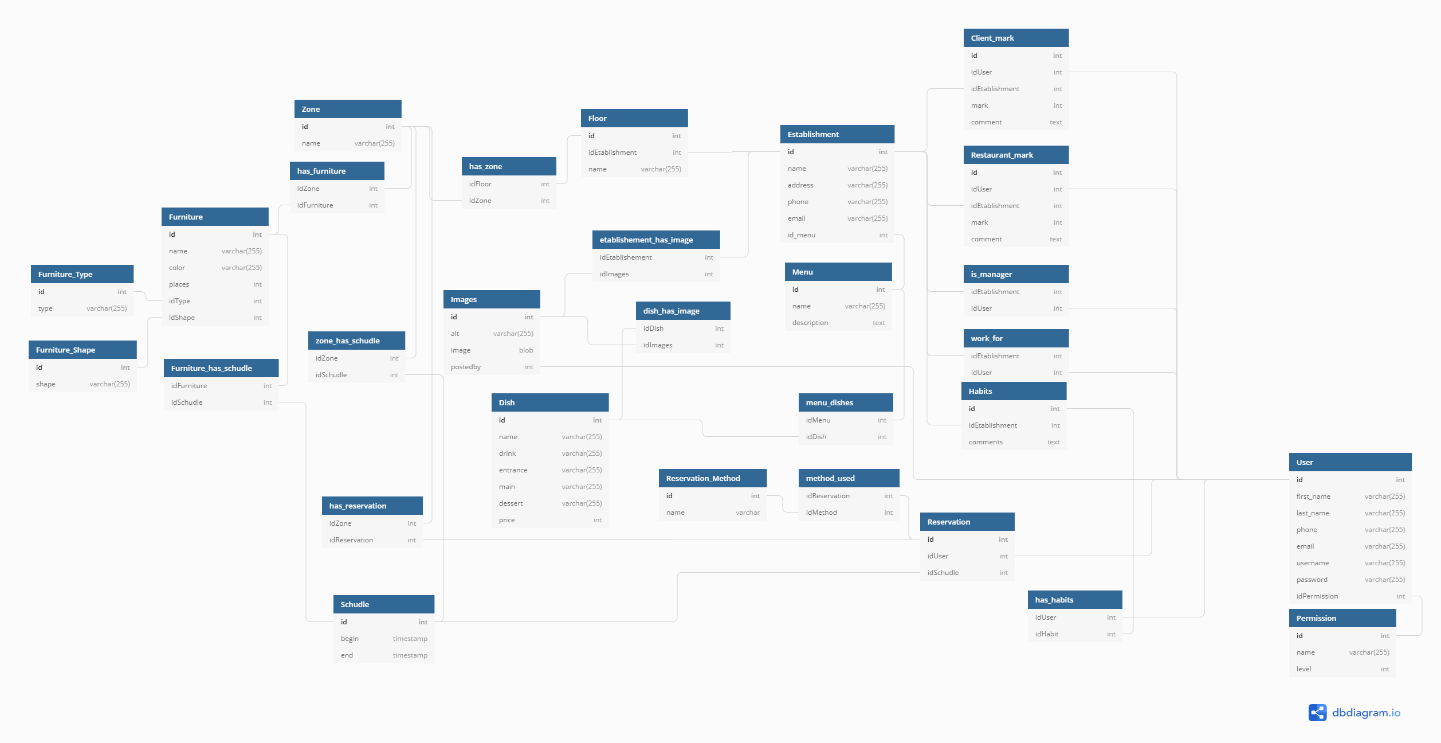


Figure 4 : la structure de la base de données en V5

### 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 de mon API dans sa version 2 :

Api/v2/

│ index.php

│ pdo.php

│ README.md

│ vars.php

│

├───etablishment

│ ├───create

│ │ │ index.php

│ │ │

│ │ └───form

│ │ index.php

│ ├───floor

│ │ ├───create

│ │ │ │ index.php

│ │ │ │

│ │ │ └───form

│ │ │ index.php

│ │ ├───get

│ │ │ index.php

│ │ └───zone

│ │ ├───create

│ │ │ │ index.php

│ │ │ │

│ │ │ └───form

│ │ │ index.php

│ │ ├───get

│ │ │ index.php

│ │ └───schedule

│ │ └───get

│ │ index.php

│ └───get

│ index.php

├───images

│ ├───background

│ │ │ index.php

│ │ └───images

│ │ no\_image.png

│ │ profile\_background.png

│ │ RESA\_700x1000.png

│ ├───dish

│ │ [images]

│ ├───get

│ │ index.php

│ ││ ├───restaurant

│ │ [images]

│ ├───upload

│ │ │ index.php

│ │ │

│ │ └───debug

│ │ index.php

│ └───user

| [images]

├───menu

│ └───get

│ index.php

├───reservation

└───user

├───employes

│ index.php

├───get

│ index.php

├───password

│ index.php

├───set

└───username

└───generate

index.php

### Variables globales

### Gestion des images

Pour rendre une application plus attirante visuellement, il ne faut pas négliger les images. Afin de me faciliter la gestion des images que ce soit leur mise en ligne ou tout simplement les récupérer, j’ai décidé de créer une gestion des images par mon API.

#### Mise en ligne d’une image

La mise en ligne est un peu spéciale, afin de facilement mettre en ligne une image, j’ai créé un fichier PHP qui suffit d’include dans le fichier qui souhaite enregistrer l’image. Je suis passé par cette option, car il s’agissait de la plus facile à mes yeux et que je ne souhaitais pas perdre de temps alors que cette option fonctionne bien.

##### Exemple

L’exemple va prendre en compte le formulaire de création d’un établissement par un utilisateur. Dans l’HTML, il y a un formulaire qui possède comme action l’url de création de l’API.

<form action="<?php echo $path."etablishment/create/form/"; ?>" method="post" enctype="multipart/form-data" id="creationEtablissement">

Dans le parameter enctype, il faut bien mettre "multipart/form-data" car c’est ce paramètre qui permet d’envoyer les images dans la variable $\_FILES vers un autre fichier.

Du côté de l’API, voici comment les informations et les photos sont récupérées :

if(isset($\_POST) && isset($\_FILES)){

    if(count($\_POST) > 0 && count($\_FILES) > 0){

        if(CheckData($\_POST)){

            SendData($\_POST, $\_FILES, $FullPathToAPI);

            header("Location: {$\_SERVER['HTTP\_REFERER']}");

            exit();

        }

    }

}

La fonction CheckData va vérifier que toutes les donnes envoyées dans la variable POST soient bien conformes aux attentes, c’est-à-dire que le nom soit bien une chaine de caractère, que l’email soit bien un email, etc.

Ensuite, la fonction SendData va envoyer les fichiers dans la base de données à l’aide de requêtes GET de l’API. Pour ce faire, j’utilise le querybuilder de PHP pour préparer correctement ma requête.

$queryData = array(

        'name' => $data['name'],

        'address' => $data['adress'],

        'phone' => $data['phone'],

        'email' => $data['email'],

        'creatorID' => $\_SESSION['user']->id

    );

$link1 = $path."etablishment/create/?".http\_build\_query($queryData);

file\_get\_contents($link1);

Afin d’enregistrer les images dans l’API, j’envoie uniquement le fichier tmp et le nom de l’image à l’API. Le fichier tmp représente le fichier mis en cache par le navigateur temporairement avant d’être enregistré à quelque part par le code.

SaveImageEtablishment($lastid->last, $\_SESSION['user']->id, $images['photos']['name'][$i], $images['photos']['tmp\_name'][$i]);

#### Récupérer les images

Afin de pouvoir récupérer le lien des images de mon API, il faut passer par des requêtes GET. Le lien principal pour récupérer les images est le suivant : /api/v2/images/get/

A partir de là, il faut ajouter les paramètres de ce que l’on cherche.

##### Exemples

Il existe tous les paramètres suivants afin de récupérer les informations ou les liens des images de l’API. Toutes les requêtes doivent posséder l’identifiant de l’utilisateur, l’établissement ou repas recherché.

|  |  |  |
| --- | --- | --- |
| **Nom du paramètre** | **Lien complet** | **Retour** |
| **data** | /api/v2/images/get/?data&id=XX | Un tableau avec les informations de la table |
| **etablishment** | /api/v2/images/get/? etablishment&id=XX | Un tableau avec l’id des images ainsi que leur lien complet |
| **dish** | /api/v2/images/get/? dish&id=XX | Un tableau avec l’id des images ainsi que leur lien complet |
| **user** | /api/v2/images/get/?user&id=XX | L’id de l’image ainsi que son chemin complet |
| **id** | /api/v2/images/get/?id=XX | Redirige directement sur l’image |

Le dernier paramètre (id) dois être utiliser lorsque l’on souhaite afficher directement l’image dans le path d’un img.

<img src="<?php echo $path."images/get/?id=73hd93"?>" alt="">

### Cheat Sheet de l’API

Afin de retrouver facilement les requêtes pour l’API, j’ai créer ce cheat sheet (disponible aussi en md sur le github).

### User

Lien : /api/v2/user/

#### Lecture

Récupérer les informations d’un user avec ses permissions

* + Lien : /api/v2/user/get/
  + Paramètres :
    - id : l’id de l’utilisateur recherché
  + Lien avec paramètres : /api/v2/user/get/?id=[id de l'utilisateur]
  + Retour :
    - Un tableau avec les champs :
      * id : l’id de l’utilisateur
      * first\_name : le prénom de l’utilisateur
      * last\_name : le nom de famille de l’utilisateur
      * phone : le numéro de téléphone de l’utilisateur
      * email : l’email de l’utilisateur
      * username : le code à 4 chiffre d’identification de l’utilisateur
      * permissions : un tableau avec toutes ses permissions. Chaque permission contient :
        + etablishment\_name : le nom de l’établissement (si il y en à un, sinon “-”)
        + permission\_name : le nom de la permission (ex: manager)

Récupérer les informations de base d’un user

* + Lien : /api/v2/user/get/
  + Paramètres :
    - user : (il n’y à pas besoin de valeur)
    - id : l’id de l’utilisateur recherché
  + Lien avec paramètres : /api/v2/user/get/?user&id=[id de l'utilisateur]
  + Retour :
    - Un tableau avec les champs :
      * id : l’id de l’utilisateur
      * first\_name : le prénom de l’utilisateur
      * last\_name : le nom de famille de l’utilisateur
      * phone : le numéro de téléphone de l’utilisateur
      * email : l’email de l’utilisateur
      * username : le code à 4 chiffre d’identification de l’utilisateur

Récupérer tous les utilisateurs avec une certaine permission

* + Lien : /api/v2/user/get/
  + Paramètres :
    - byPermission : l’id de la permission recherchée
  + Lien avec paramètres : /api/v2/user/get/?byPermission=[id de la permission]
  + Retour :
    - Un tableau de users, ou chaque user possède les champs :
      * first\_name : le prénom de l’utilisateur
      * last\_name : le nom de famille de l’utilisateur
      * phone : le numéro de téléphone de l’utilisateur
      * email : l’email de l’utilisateur
      * username : le code à 4 chiffres d’identification de l’utilisateur

Récupérer toutes les permissions d’un utilisateur

* + Lien : /api/v2/user/get/
  + Paramètres :
    - permissions : (il n’y à pas besoin de valeur)
    - id : l’id de l’utilisateur recherché
  + Lien avec paramètres : /api/v2/user/get/?permissions&id=[id de l'utilisateur]
  + Retour :
    - Un tableau de permissions, ou chaque permission possède les champs :
      * etablishment\_name : le nom de l’établissement (s’il y en a un, sinon “-”)
      * permission\_name : le nom de la permission (ex: manager)

Récupérer tous les utilisateurs

* + Lien : /api/v2/user/get/
  + Paramètres :
    - Aucun
  + Lien avec paramètres : /api/v2/user/get/
  + Retour :
    - Un tableau de users, ou chaque user possède les champs :
      * id : l’id de l’utilisateur
      * first\_name : le prénom de l’utilisateur
      * last\_name : le nom de famille de l’utilisateur
      * phone : le numéro de téléphone de l’utilisateur
      * email : l’email de l’utilisateur
      * username : le code à 4 chiffres d’identification de l’utilisateur

Récupérer tous les employés d’un etablissement

* + Lien : /api/v2/user/employes/
  + Paramètres :
    - workingFor : l’id de l’etablissement
  + Lien avec paramètres : /api/v2/user/employes/?workingFor=[id de l'etablissement]
  + Retour :
    - Un tableau de users, ou chaque user possède les champs :
      * id : l’id de l’utilisateur
      * user\_firstname : le prénom de l’utilisateur
      * user\_lastname : le nom de famille de l’utilisateur
      * permission\_name : le nom de sa permission (ex: Manager)
      * permission\_level : le niveau de la permission (ex : 2)

#### Divers

Login d’un employé dans son restaurant

* + Lien : /api/v2/user/get/
  + Paramètres :
    - login\_e : (il n’y à pas besoin de valeur)
    - username : l’identifiant à 4 chiffres de l’utilisateur
    - password : le mot de passe hashé en sha256
  + Lien avec paramètres : /api/v2/user/get/?login&username=[identifiant 4 chiffres]&password=[mot de passe hashé (sha256)]
  + Retour :
    - Un tableau avec les champs :
    - id : l’id de l’utilisateur
    - first\_name : le prénom de l’utilisateur
    - last\_name : le nom de famille de l’utilisateur
    - phone : le numéro de téléphone de l’utilisateur
    - email : l’email de l’utilisateur

Générer un identifiant numérique aléatoire

* + Lien : /api/v2/user/username/generate/
  + Paramètres :
    - Aucun
  + Lien avec paramètres : /api/v2/user/username/generate/
  + Retour :
    - un numéro aléatoire

### Floor

Lien : /api/v2/etablishment/floor/

#### Lecture

Récupérer tous les étages, zones et horaires d’un etablissement

* + Lien : /api/v2/etablishment/floor/get/
  + Paramètres :
    - id : l’id de l’établissement
  + Lien avec paramètres : /api/v2/etablishment/floor/get/?id=[id de l'établissement]
  + Retour :
    - Un tableau avec comme clés les id des étages:
      * [id de l'étage] : l’id de l’étage comme index du tableau afin de facilement le retrouver
        + name : le nom de l’étage
        + zones : tableau des zones

[nom de la zone] : le nom de la zone

[xx:xx:xx] : l’heure de début (begin)

[xx:xx:xx] : l’heure de fin (end)

#### Création

Créeation d’un étage

* + Lien : /api/v2/etablishment/floor/create/
  + Paramètres :
    - create : (il n’y à pas besoin de valeur)
    - name : nom de l’étage
    - etablishment : l’id de l’établissement
  + Lien avec paramètres : /api/v2/etablishment/floor/create/?create&name=[nom de l'étage]&etablishment=[id de l'établissement]
  + Retour :
    - Nothing

Création d’un étage à partir d’un formulaire

* + Lien : /api/v2/etablishment/floor/create/form/
  + Paramètres :
    - $\_POST
      * name : nom de l’étage
      * etablishment : l’id de l’établissement
  + Lien avec paramètres : /api/v2/etablishment/floor/create/form/
  + Retour :
    - nothing

### Schedule

Lien : /api/v2/etablishment/floor/zone/schedule/

#### Lecture

Récupère l’id de la dernière zone ajoutée

* + Lien : /api/v2/etablishment/floor/zone/get/
  + Paramètres :
    - last : (il n’y à pas besoin de valeur)
  + Lien avec paramètres : /api/v2/etablishment/floor/zone/get/?last
  + Retour :
    - schedules :
      * schedule\_id : l’id de l’horaire
      * begin : l’heure de début
      * end : l’heure de fin

### Images

Lien : /api/v2/images/

Récupérer les informations completes d’une image

* + Lien : /api/v2/images/get/
  + Paramètres :
    - data : (il n’y à pas besoin de valeur)
    - id : l’id de l’image
  + Lien avec paramètres : /api/v2/images/get/?data&id=[l'id de l'image]
  + Retour :
    - [le path complet]

Etre rediriger sur l’image

* + Lien : /api/v2/images/get/
  + Paramètres :
    - id : l’id de l’image
  + Lien avec paramètres : /api/v2/images/get/?id=[l'id de l'image]
  + Retour :
    - Redirection sur la page de l’image

Récupérer toutes les images d’un etablissement

* + Lien : /api/v2/images/get/
  + Paramètres :
    - etablishment : (il n’y à pas besoin de valeur)
    - id : l’id de l’établissement
  + Lien avec paramètres : /api/v2/images/get/?etablishment&id=[l'id de l'établissement]
  + Retour :
    - Un tableau avec tous les chemins:
      * full\_path : le lien complet dans l’API pour rejoindre l’image

Récupérer toutes les images d’un repas

* + Lien : /api/v2/images/get/
  + Paramètres :
    - dish : (il n’y à pas besoin de valeur)
    - id : l’id du repas
  + Lien avec paramètres : /api/v2/images/get/?dish&id=[l'id du repas]
  + Retour :
    - Un tableau avec tous les chemins:
      * full\_path : le lien complet dans l’API pour rejoindre l’image

Récupérer la photo de profil d’un utilisateur

* + Lien : /api/v2/images/get/
  + Paramètres :
    - user : (il n’y à pas besoin de valeur)
    - id : l’id de l’établissement
  + Lien avec paramètres : /api/v2/images/get/?user&id=[l'id de l'établissement]
  + Retour :
    - Un tableau avec tous les chemins:
      * full\_path : le lien complet dans l’API pour rejoindre l’image

Mise en ligne d’une photo

* + Afin de mettre en ligne une photo il faut :
    1. Inclure le fichier /api/v2/images/upload/index.php
    2. Dans ce fichier ce trouvent les 2 fonctions suivantes :
       1. SaveImageEtablishement($idEtbalishement, $idUploader, $file, $target\_dir, $toMove)
       2. ~~SaveImageDish($idDish, $idUploader, $file, $target\_dir, $toMove)~~ **N’est plus valide**
       3. ~~SaveImageUser($idDish, $idUploader, $file, $target\_dir, $toMove)~~ **N’est plus valide**
    - Les deux fonctions prennent les paramètres suivants :
      * idEtablissement ou idDish ou idUser en fonction de quelle fonction on appelle
      * idUploader : l’id du user qui met en ligne la photo
      * file\_name : le nom de la photo à créer
      * $tmp : le fichier tempon de l’image

## Gestion du temps

Lors de mon travaille, je me suis souvent retrouvé face à des situations ou je devais faire la part des choses afin de me consacrer uniquement aux tâches réellement importantes.

### Lister les tâches

Avant de pouvoir faire un classement des tâches importantes à réaliser, je devais d’abord lister les tâches essentielles. Prenons l’exemple de la page de profil.

Lors de la création de cette page, je pensais à pleins de fonctionnalités qui permettraient à l’utilisateur de faire facilement des changements et d’accéder facilement aux établissements dont il est le manager, mais lorsque j’ai vu le temps que ça me prenait, il a fallu faire des choix. Voici la liste des tâches que j’ai rédiger pour cette page :

1. Changer la photo de profil
2. Accéder aux réservations en un coup d’œil (sous forme de widget)
3. Afficher la liste de ses établissements si l’utilisateur connecté en as
4. Afficher les informations de l’utilisateur sur sa page
5. Modifier ses informations

Après avoir fait cette liste, j’ai dû faire un classement d’importance afin de savoir lesquelles étaient indispensable avec l’objectif principal du projet : « Réserver une table »

### Classer les tâches

Une fois la liste écrite, j’ai mis tous les points dans un tableau pour pouvoir les classer par importance.

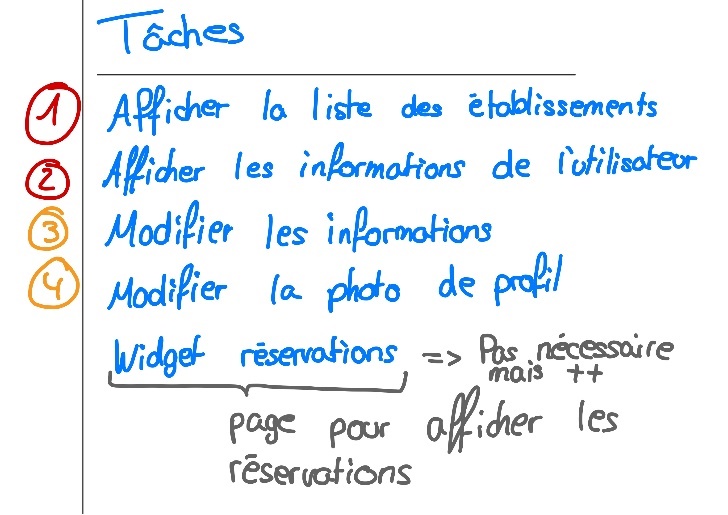
Le tableau affiche en rouge les numéros des tâches réellement critique pour continuer avec l’objectif principal et en gris les différents commentaires que je me faisais afin de ne pas oublier certaines choses qui pouvaient s’avérer utiles dans le futur.

Figure 5 Liste des tâches pour la page profil

### Séparation des tâches client et serveur

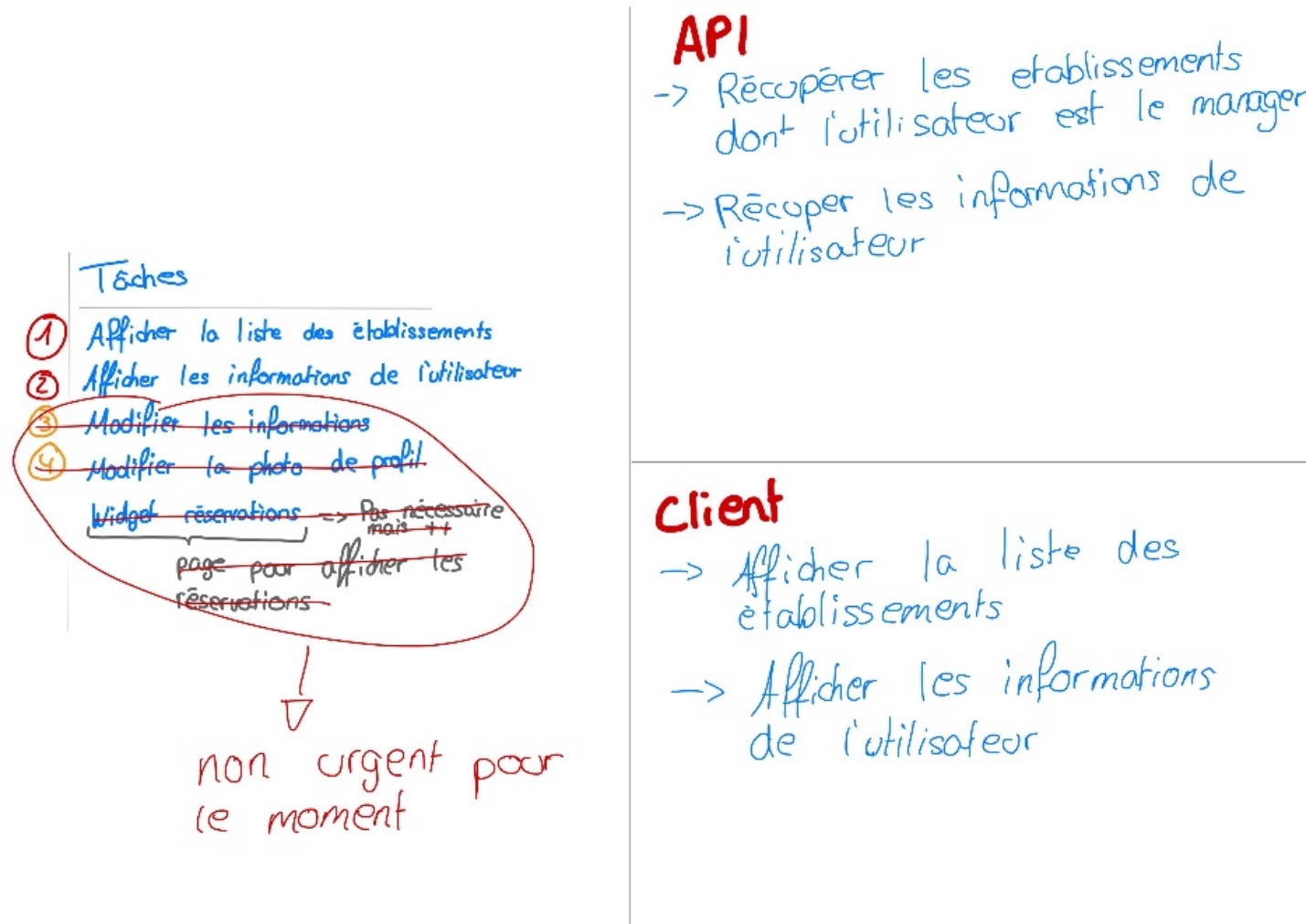
Une fois que les tâches étaient toutes classées par ordre d’importances, j’ai regardé comment séparer les tâches générales en sous-tâches afin de séparer le travail du côté client et du côté serveur.

Figure 6 Séparation des tâches client et serveur (API)

### Conclusion

La répartition des tâches et le classement de celles-ci par ordre de criticité à été d’une grande importance tout au long de mon projet, car il y avait énormément à faire et il était impossible pour moi de tout réaliser dans le temps impartis, j’ai donc fait les choix de cette manière, ce qui s’est avérer très efficace.

## Raisonnements

Une grande partie de mon travail s’est orientée sur les raisonnements que j’avais. En effet, lors de mon travail, je me suis retrouvé à de nombreuses reprises dans des situations ou je devais faire preuve de raisonnement afin de trouver la solution.

### Réflexions personnelles

Tout au long du projet, j’écrivais en permanence les réflexions personnelles et la tâche que j’avais fini, que j’étais en train de faire ou que j’allais faire. Le fait d’écrire toutes ses choses m’as permis d’avoir un suivi constant de l’avancement de mes tâches ainsi que des choses à faire.

#### Le journal de bord

Afin d’avoir un suivi constant de mon travail, il m’a été demandé de rédiger tout au long de la durée, un journal de bord qui fait parti de l’évaluation de mon travail de diplôme.

Dans le but que le journal de bord soit lisible facilement et rapidement depuis le GitHub, j’ai décider de faire mon journal de bord en markdown.

##### Structure

J’ai opté pour une structure simple et efficace qui me permet de directement savoir quand je passe d’un jour à l’autre.

---

**## 29.04.20**

De cette manière, tous les jours sont séparer par un trait horizontal.

##### Extrait du journal de bord lors de réflexions

- Je suis en train de faire le login d'un utilisateur, je me suis rendu compte que je devais faire la différencation de si il s'agissait d'un login client ou d'un login utilisateur local.

  - Il faut que j'ajoute à l'API une fonctionnalité qui gère la différence

    - Je vais donc créer cette fonctionnalité

    - Afin de pouvoir ajouter cette fonctionnalité, j'ai un petit souci... Un utilisteur admin peut ce connecter dans tous les restaurants, il faut donc que ma requête gère si il s'agit de d'un admin

    - **### Il ne faut pas que j'oublie de mettre à jour le cheat sheet !**

  - Il faut que j'ajoute une fonction de login avec l'email

  - (je ne sais pas si je dois aussi hasher le username et l'email ou non ... Pour le moment je ne vais pas le faire car les emails sont public...)

- Ajout du fond d'écran de tous les profils utilisateurs

  - Accessible à partir de l'API

- Il faut que je fasse un table de liaison entre les utilisateurs et une photo de profil

#### Analyse

Après avoir fait la moitié de mon projet, je me suis demandé si les réflexions que je me faisais étaient pertinentes et si elles me menaient à mon objectif. En relisant mon journal de bord, je me suis rendu compte que l’écriture de chaque étape me permettait de facilement retrouver les informations manquantes à un instant ‘t’. Le journal de bord m’as permis de suivre, analyser et améliorer mon travail en cours sans devoir passer du temps inutilement à chercher des informations que j’avais notées.

### Communications avec M. Garcia

Afin de toujours avoir un suivi permanent de mon travail de diplôme et suite aux circonstances extraordinaires de 2020, nous avions tous les jours une conversation en visioconférence afin d’être à jour sur l’avancement du projet.

Lors de ces appels, m. Garcia m’as beaucoup aidé à raisonner sur les méthodes et les objectifs que je devais avoir afin de compléter correctement mon travail.

# Glossaire

1. Lien complet de l’application : https://app.diagrams.net [↑](#footnote-ref-1)
2. Https://sourcemaking.com/uml/modeling-business-systems/external-view/activity-diagrams [↑](#footnote-ref-2)
3. Https://dbdiagram.io [↑](#footnote-ref-3)