Travail de Diplôme ES 2020

Constantin Herrmann

Avril – Juin 2020

M. Francisco Garcia

CFPT-I Technicien ES

Version 0.4 (révision avant évaluation intermédiaire)

1 Table de matières

2 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 reprend tout le projet à travers une analyse fonctionnelle et organique qui décrit précisément le processus de création, de développement et d'analyse des éléments clés du projet.

3 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.

4 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.

5 Analyse de l'existant

5.1 La Fourchette

Figure 1 Logo "lafouchette"

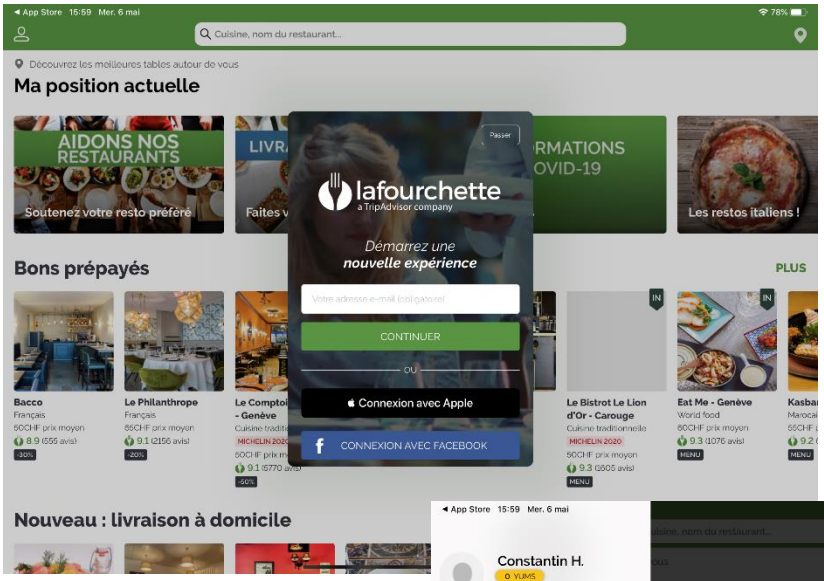
La fourchette verte est l'application qui se rapproche le plus de RESA. En effet, cette dernière regroupe tous les principaux domaines de la restauration. Elle dispose d'un site internet et d'une application mobile et tablette.

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

La fourchette verte possède deux interfaces très distinctes. Celles destinées aux clients et celles destinées aux restaurateurs.

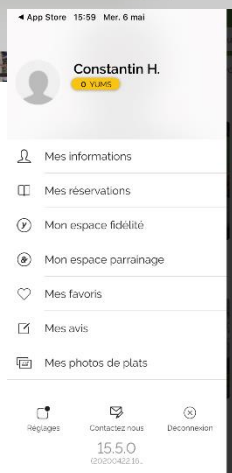
5.1.1 Clients

5.1.1.1 Interfaces

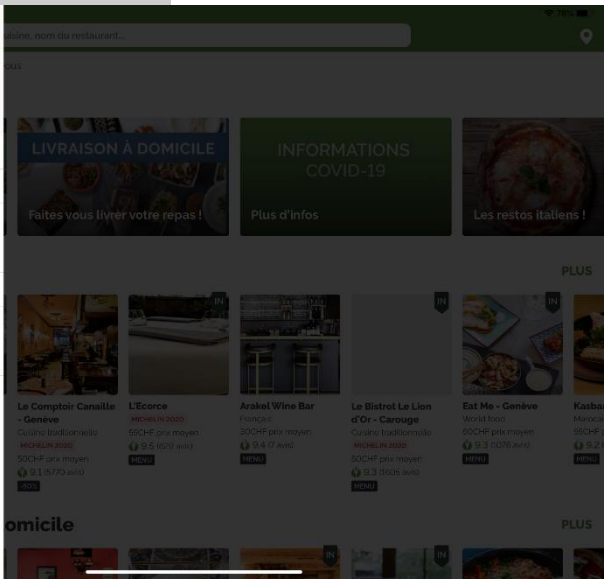


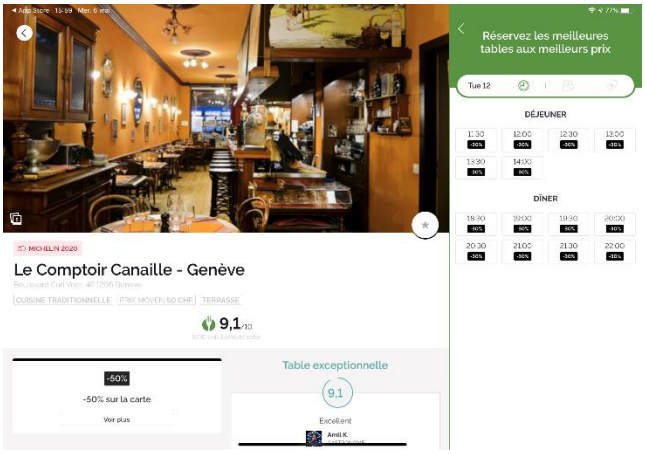
La page login est assez simple, elle permet à un utilisateur de facilement se connecter en entrant uniquement une adresse mail ou par google et Apple.

Comme on peut le voir à l'arrière, les restaurant sont affichés sous forme de cartes que l'on peut sélectionner afin d'avoir plus d'informations sur le restaurant.



Une fois connecté, l'utilisateur accède à ces options qui sont, ses informations, ses réservations, son espace de fidélité, son espace parrainage, ses avis, ses favoris et enfin ses photos de plats qu'il à publier sur le site afin de partager avec les autres utilisateurs.





Lorsque que l'utilisateur sélectionne le restaurant de son choix, on lui montre en grand la photo du restaurant, sa note, les réductions disponibles et quelques photos.

Il y a également une liste sur la droite avec les horaires disponibles pour le jour sélectionné. Un fois choisi, on lui demande le nombre de personnes et les réductions ou bons qu'il souhaite appliqués ou non.

5.1.1.2 Fonctionnalités clés

[A compléter]

5.1.2 Restaurateur

5.1.2.1 Interface

[A compléter]

5.1.2.2 Fonctionnalités clés

[A compléter]

6 Analyse fonctionnelle

6.1 Interfaces

L'analyse fonctionnelle reprend tous les éléments qui ont servi à créer l'application tel qu'elle est. Dans cette partie, nous allons analyser les interfaces et le processus de création de celles-ci.

6.1.1 Template

Afin de ne pas passer trop de temps sur la création de mes vues ainsi que sur le design des composants, j'ai décider de prendre un Template qui possède plusieurs styles de widgets, menus, interfaces et formulaires. Le Template que j'ai donc choisi ce nomme : Costic HTML[1] et est disponible sur le site themeforest[2].

Ce Template a été pensé pour les restaurants, ce qui était parfait pour mon projet. Une fois téléchargé et installer, j'ai changé les thèmes principaux afin que ceux-ci se rapprochent le plus possible du thème de RESA (vert et gris) et j'ai commencer à créer des nouvelles fenêtres en y intégrant les différents modules qui m'intéressaient.

Par exemple, pour la page d'acceuil de l'application.

Template affichage en « cartes »

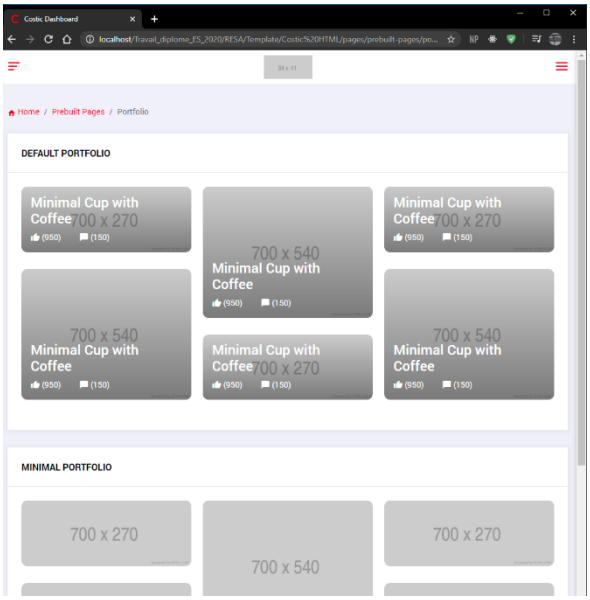


Figure 2 Capture d'écran du template

Page d'accueil RESA

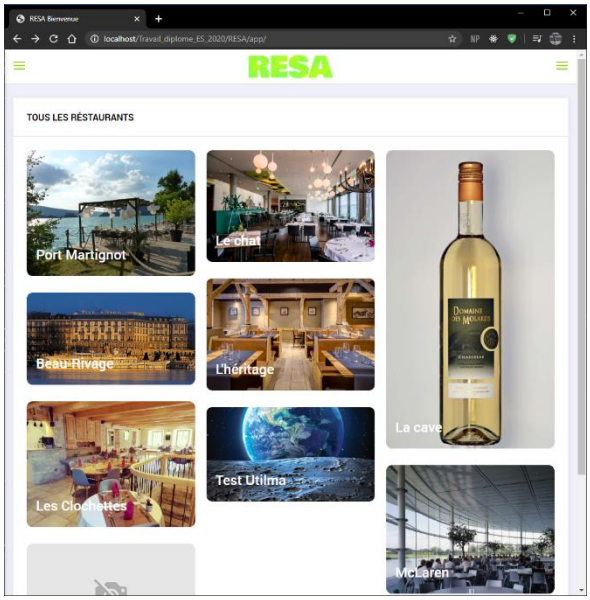


Figure 3 Capture d'écran de la page d'acceuil RESA

Comme on peut le constater, je n'ai vraiment que réutiliser les modules créer par themeforest. L'aménagement des fenêtres à été entièrement repenser par moi-même tout au long tu projet. Le template était facilement modulable et très bien structuré ce qui m'as laissé pas mal de possibilités.

[1] https://themeforest.net/item/costic-restaurant-admin-dashboard-html5-template/25634499
[2] https://themeforest.net/

6.1.2 La page d'accueil

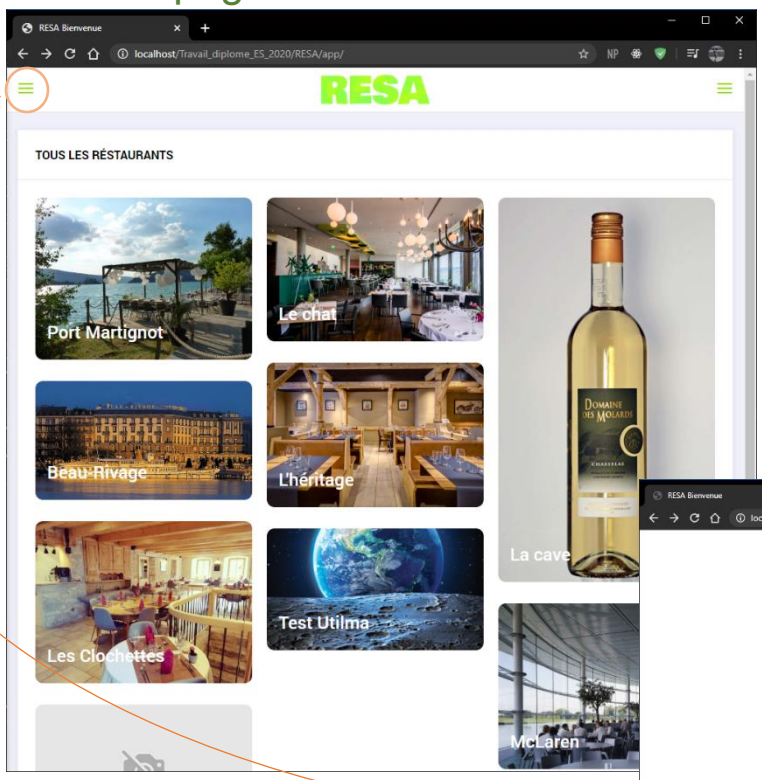


Figure 4 Page d'accueil

L'objectif principal de la page d'accueil à été de rendre l'affichage des restaurants dynamiques et simples à voir. La liste permet de parcourir tous les restaurant de la base.

Il y a également un filtre afin de trier tous les restaurants. Que ce soit par nom, capacité, horaire ou position (par rapport à une adresse), le filtre permet à l'utilisateur de facilement retrouver son restaurant afin de réserver une table.

Le menu de gauche affiche tous les raccourcis dont l'utilisateur connecté à besoin. Il retrouvera principalement l'accès facile à ces réservations, ses favoris et ses établissements si celui-ci est manager d'un ou plusieurs établissement(s).

L'administrateur quant à lui aura plus de liens que les autres. Par exemple il aura le lien d'administrations des établissements enregistrés, le lien de tous les utilisateurs afin de pouvoir effectuer des modifications en cas de besoin.

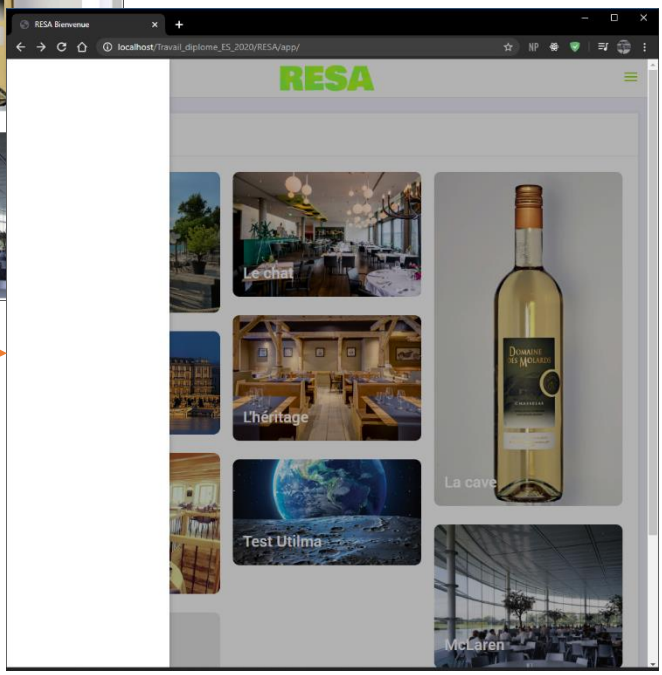


Figure 5 Barre de navigation latérale

Le menu de droite quant à lui est destiné aux actions concernant directement l'utilisateur. En cliquant sur l'icône, il aura le choix entre se connecter ou de créer un compte. (A noter qu'il est possible de voir les restaurants sans avoir de compte. Le compte est uniquement nécessaire en cas de réservation.

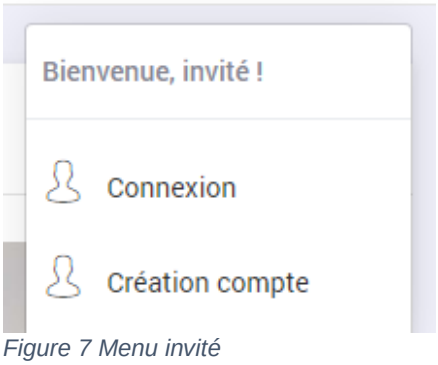


Figure 7 Menu invité

Si l'utilisateur est connecté, d'autres choix lui sont alors proposé.

- Accéder à son compte
- Se déconnecter

La photo de profil de l'utilisateur vient également remplacer la photo par défaut affichée dans le coin supérieur

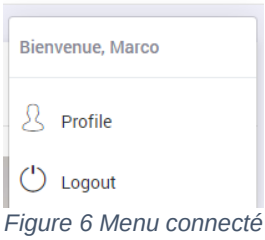


Figure 6 Menu connecté

6.1.3 Profil utilisateur

La page profil guide facilement l'utilisateur connecter afin d'accéder aux données dont il à besoin étant connecter.

La page de profil comporte une barre de navigation centrale qui reprend les mêmes éléments que dans la barre latérale sur la page d'accueil (La barre de navigation latérale est accessible depuis n'importe quelle page sur le site)

Il y a également un onglet de liens rapides. Ces liens rapides permettent d'effectuer des actions « flash » dans le widget prévu à cet effet. L'utilisateur se voit offert les trois onglets suivants :

1. Réservations
2. Changer la photo de profil
3. Création restaurant

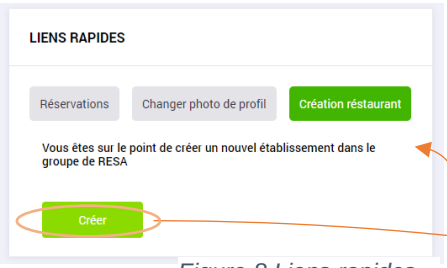


Figure 8 Liens rapides

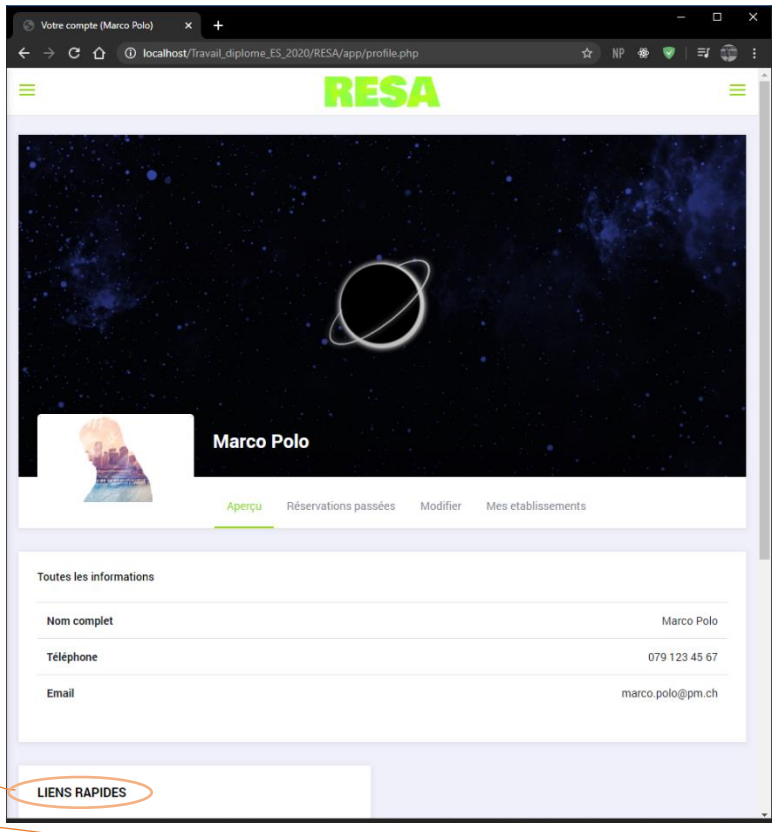


Figure 9 Page de profil

6.1.4 Création d'un établissement

En cliquant sur le bouton « Créer » ci-dessus ou en sélectionnant « créer restaurant » dans le menu de droite de la barre de navigation latérale, l'utilisateur va voir apparaître un popup avec un formulaire afin de pouvoir créer le restaurant.

L'utilisateur dois alors entrer le nom de l'établissement, son adresse complète, le numéro de téléphone de contact du restaurant ainsi que l'émail de contact.

Il doit également sélectionner des images qui servirons à mettre en valeur le restaurant sur le mur des restaurants de la page d'accueil. Si aucune photo n'est ajoutée, l'établissement aura la photo par défaut.

(Astuce développeur : Plus la photo ajoutée est grande, plus elle prendra de place sur le mur)

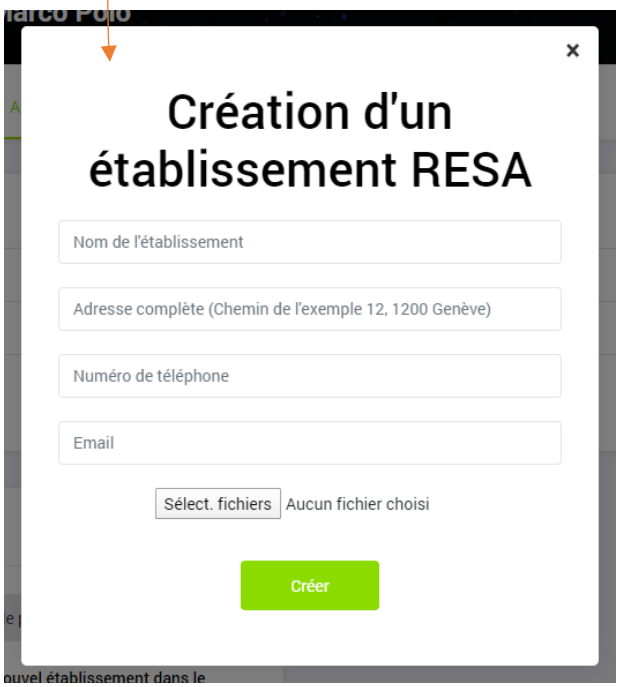


Figure 10 Popup création d'un établissement

6.1.5 Administration d'un établissement

Il est possible de voir la liste de ses établissements sur la page de profil de l'utilisateur ou dans la barre de navigation latérale de droite.

Sur la page de profil, les établissements s'affichent de la manière suivante :

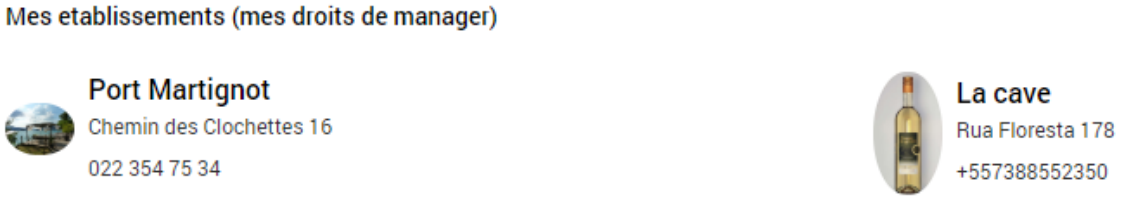


Figure 11 Affichage des établissement - page profil

Dans la barre latérale, les établissements s'affichent sous forme de liste. L'utilisateur peut alors sélectionner l'établissement qu'il souhaite manager.

En sélectionnant le restaurant de son choix, que ce soit sur sa page de profil ou dans la barre de navigation, l'utilisateur (ou dans ce cas le manager) sera automatiquement redirigé sur la page d'administration de son établissement.

Figure 12 Liste des établissement - barre de navigation

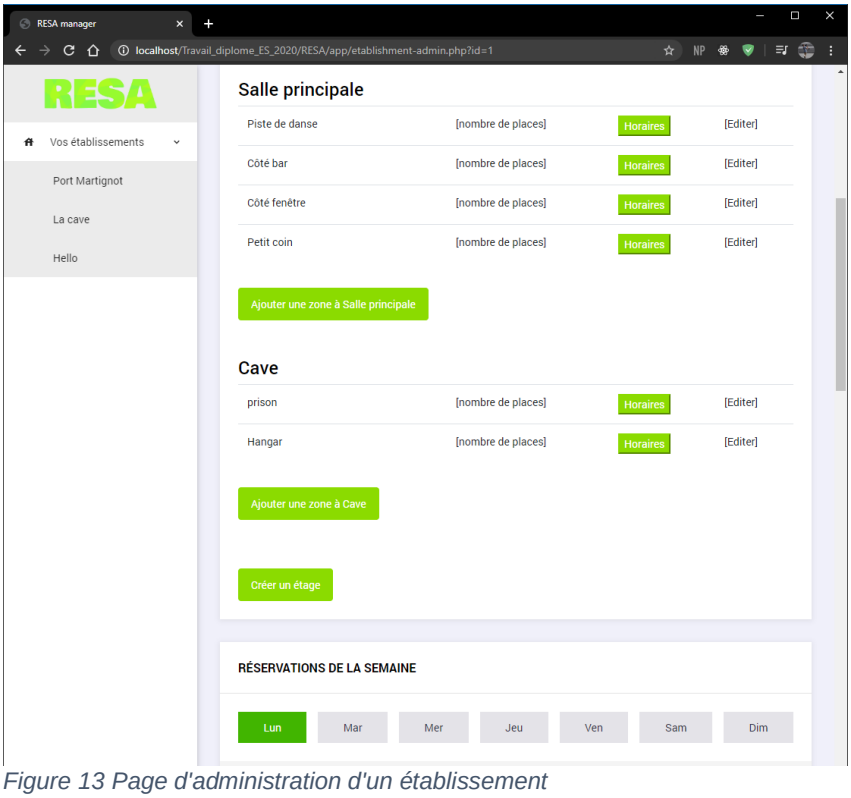


Figure 13 Page d'administration d'un établissement

Sur la page d'administration, le manager peut visualiser tous les étages et toutes les zones de son restaurant. Il peut également à l'aide des boutons dédiés, créer des nouveaux étages, puis des zones dans ceux-ci. Il peut également voir le nombre de place possibles dans la zone ainsi que les horaires.

Le manager à également accès à toutes les réservations de la semaine afin de facilement pouvoir les consulter, les modifier en cas de besoin ou les supprimer.

Il possède également un widget avec l'état actuel du restaurant et des clients qui se trouvent dans l'établissement.

6.2 Les droits des utilisateurs

Il existe un grand nombre de fonctionnalités dans RESA, c'est pourquoi elles ne seront pas toutes listées.

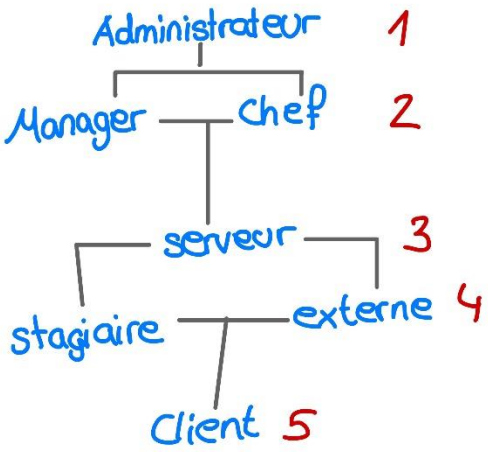


Figure 14 Hiérarchie de RESA

6.2.1 Utilisateur

Il existe différents types d'utilisateurs dans RESA, mais les principaux sont :

1. Administrateur
2. Manager
3. Serveur
4. Client

Chacun possède ses propres fonctionnalités qui ne sont pas accessibles via les autres types.

6.2.1.1 Administrateur

L'administrateur est celui possède le plus de pouvoir par rapport aux autres types. Il s'agit de celui qui peut accéder à toutes les données, en ajouter, modifier ou même supprimer.

Il à un contrôle total de RESA et à accès à tous les établissements en tant que manager.

Sont rôle est très puissant. Il faut donc le garder sécurisé au maximum et pour ce faire, le mot de passe est demandé avant chaque action critique.

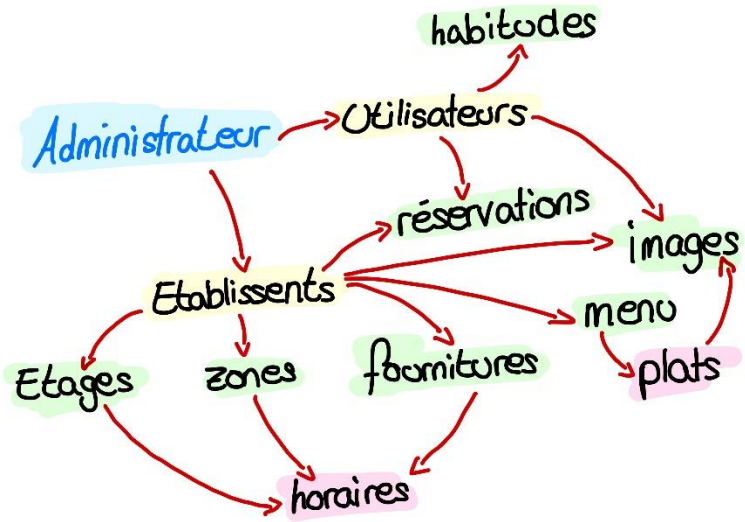
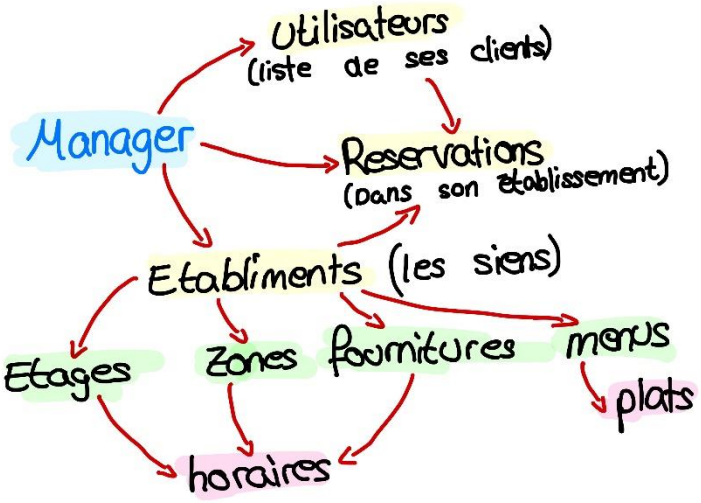


Figure 15 Schéma des droits d'un administrateur

6.2.1.2 Manager



Le manager est un utilisateur qui à créer un établissement, c'est-à-dire que cet utilisateur est un simple client pour tous les établissements, sauf les siens, où il a le contrôle total. Il peut ainsi facilement gérer son personnel et leurs accès, ses étages, zones et fournitures ainsi que les horaires de ces derniers. Il a également la possibilité de mettre à jour les données de son restaurant comme son menu, ses plats ou ses photos.

Figure 16 Schéma des droits d'un manager

6.2.1.3 Serveur

Le serveur est l'utilisateur avec le moins de droits dans un établissement. Contrairement au manager, le Serveur à uniquement accès aux réservations pour en créer, les modifier ou les supprimer. Le serveur peut mettre à jour les données en salle comme le statut du client ou un commentaire. Il a également accès aux données du restaurant et à sa liste de client. Il n'a pas le droit de modification sur ces dernières.

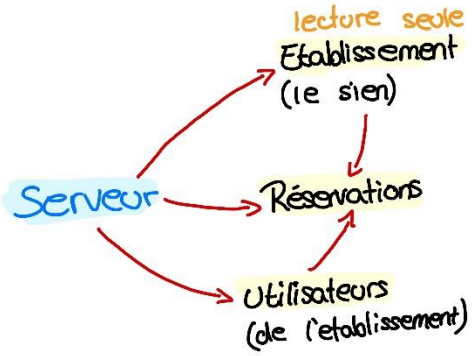


Figure 17 Schéma des droits d'un serveur

6.2.1.4 Client

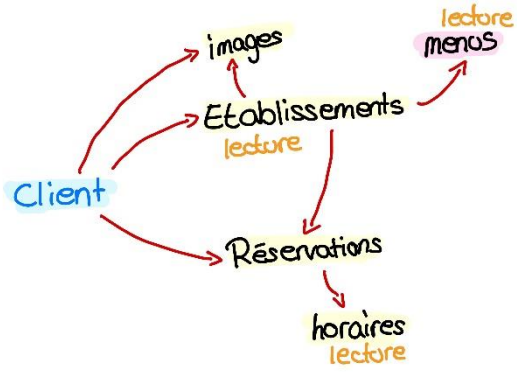


Figure 18 Schéma des droits d'un client

Le client ne possède aucun droit particulier dans l'application. Il peut cependant accéder à ses réservations pour les consulter, les modifier ou les supprimer et à la liste de tous les restaurants de RESA.

Il à également accès aux données des restaurants pour consulter les photos, les menus et les avis laissés par les autres clients du magasin.

Il à également le droit (comme tous les autres utilisateurs) de modifier ses données ainsi que sa photo de profil qui s'affiche dans les commentaires ainsi que dans les réservations.

7 Analyse organique

7.1 Mise en place

7.1.1GitHub

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
```

Extrait de code n°1. Arborescence de Github

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

7.1.2 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 demandent 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).

7.2 Pré-travail

7.2.1Programmation

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** »

7.2.2Installation 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
```

Extrait de code n°2. Commande cmd pour créer le projet react

"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
```

Extrait de code n°3. Arborescence d'un projet react basique

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

7.2.3 Conventions

7.2.3.1 En-tête de fichier

Afin de faciliter le développement et la gestion des fichiers de mon API ou de l'application, j'ai décidé de mettre le même en-tête sur les fichiers que je créais :

```
/*************************************************************************
AUTEUR      : Constantin Herrmann
LIEU        : CFPT Informatique Genève
DATE        : Avril 2020
TITRE PROJET: RESA
VERSION     : 1.0
*************************************************************************/
```

Extrait de code n°4. Header des fichier PHP de RESA

C'est en-tête me permet donc de facilement repérer les fichiers que j'ai créé et développer, ainsi que voir leur version dans un futur ou il y aura des mises à jour de l'application.

7.2.3.2 En-tête de fonction

Toutes les fonctions de l'API ont cet en-tête qui permet facilement d'identifier son but ainsi que les paramètres à envoyer à celle-ci.

```
/*
* Lie une image à un établissement
* Params:
*   - idEtbalishement : l'id de l'établissement à lier
*   - idUploader : l'id de l'utilisateur qui met en ligne la photo
*   - file : la photo à mettre en ligne
*/
```

Extrait de code n°5. Header d'une fonction PHP

7.2.3.3 Diagrammes d'activités

Les diagrammes d'activités demandent des normes bien spécifiques. Afin de respecter une norme, j'ai décidé de me fier au site de Sourcemaking[3]. Ce site reprend chaque évènement, action ou lien en expliquant clairement comment faire.

7.2.4Organisationnel

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.

7.3 Environnement

7.3.1 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.

J'ai 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.

7.3.2 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.

7.3.3 EDUGE

Je fais un backup de mon projet tous à chaque changement majeur 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.

[3] https://sourcemaking.com/uml/modeling-business-systems/external-view/activity-diagrams

7.3.4Github 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.

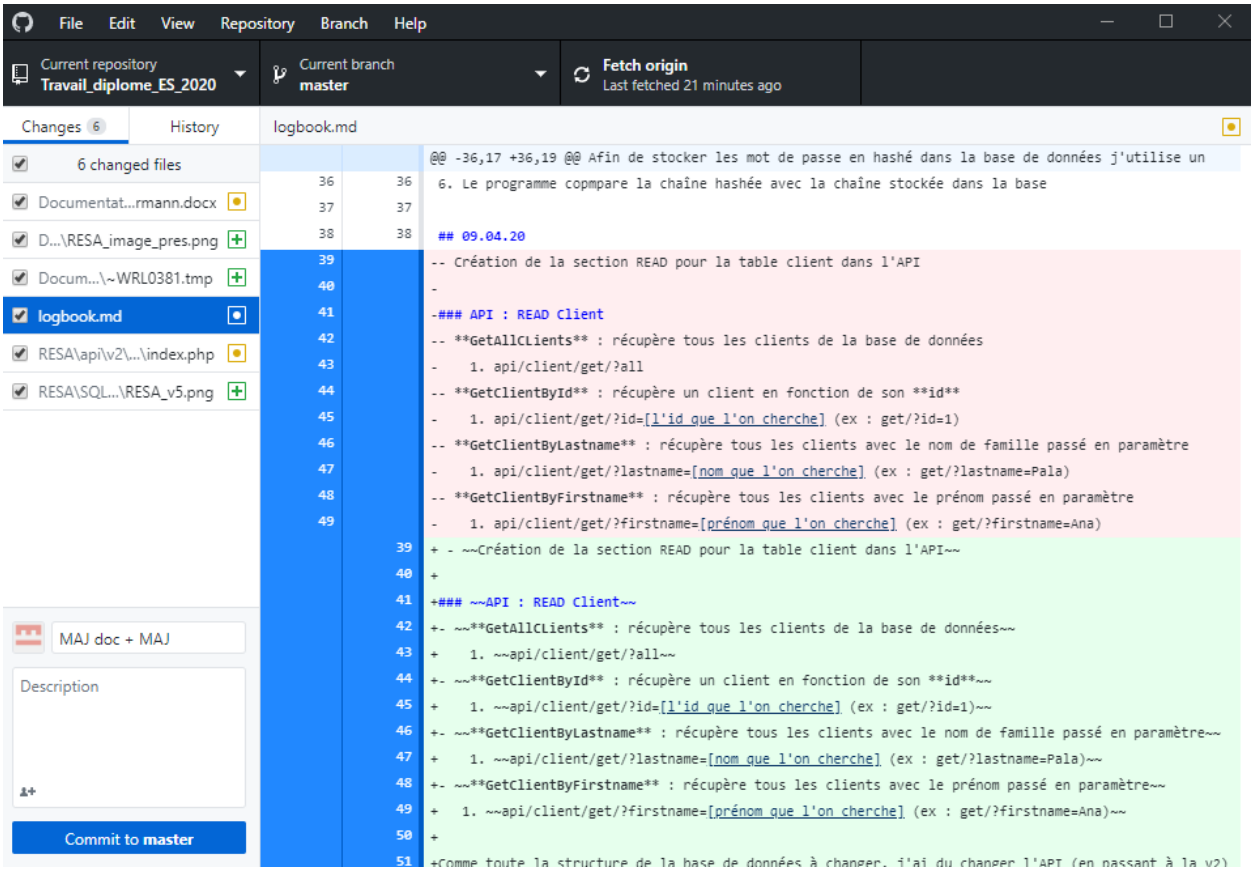


Figure 19 : Interface de Github Desktop

7.4 Schémas de fonctionnements

Afin de mieux visualiser l'analyse sur le fonctionnement de RESA et de son API, les différentes étapes clés seront illustrées par des schémas de fonctionnement.

7.4.1 RESA

RESA est un application web assez simple qui communique avec un service REST afin d'accéder à une base de données.

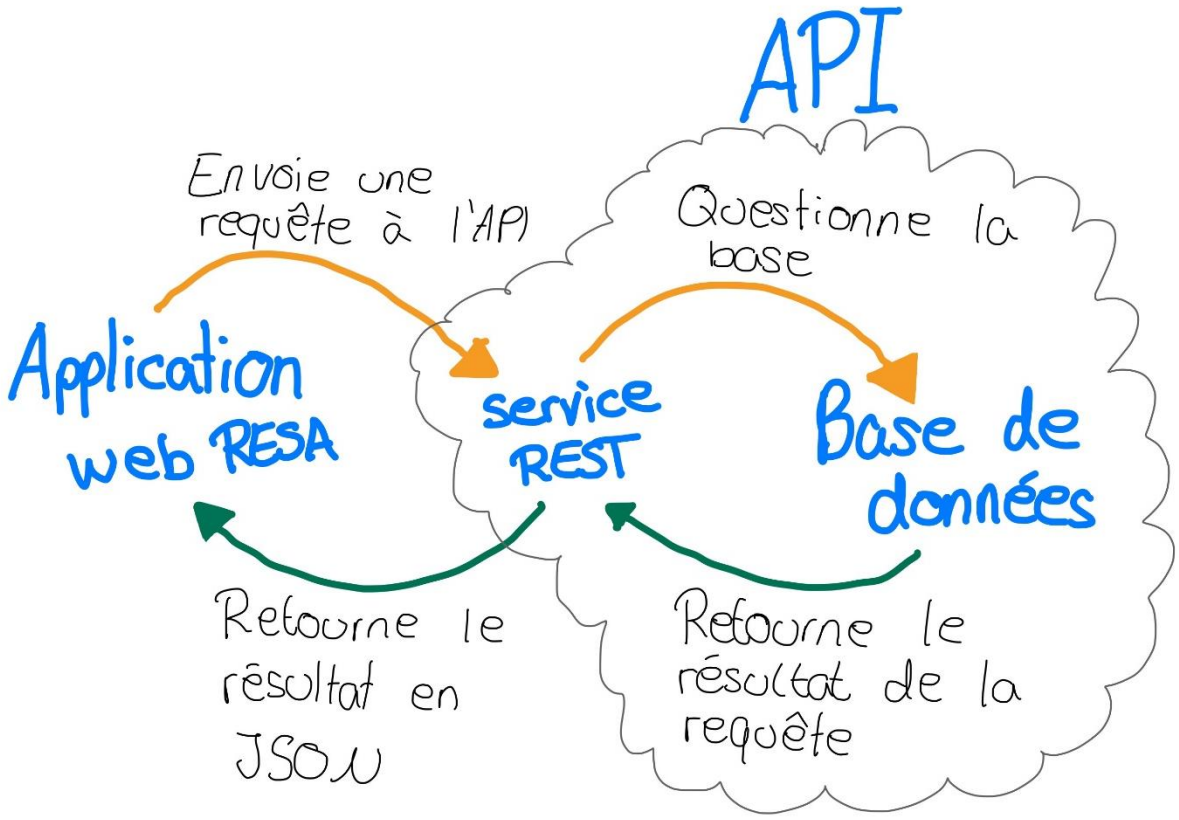


Figure 20 Diagramme de fonctionnement RESA

7.4.2 Login

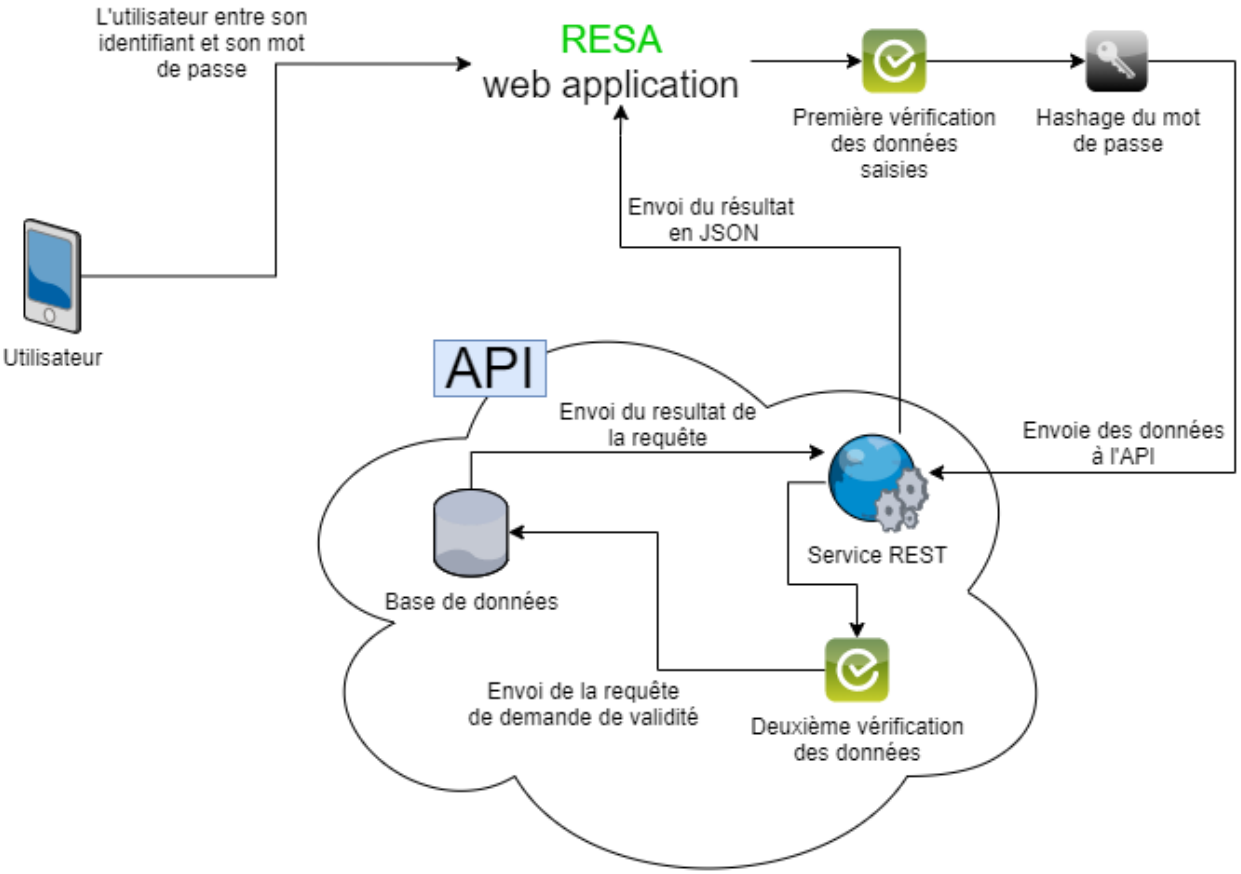


Figure 21 Diagramme de fonctionnement de login

7.4.3 Création Etablissement

[Schéma pour la création d'un établissement]

7.4.4 Mise en ligne d'une image

[Schéma pour la mise en ligne d'une image]

7.5 Diagrammes d'activités

Pour la création des diagrammes d'activités, j'ai d'abord utilisé le site internet « dbdiagram.io », mais lorsque monsieur Garcia a 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[4].

[4] Lien complet de l'application : https://app.diagrams.net

Afin d'être sûr de respecter les normes de diagrammes d'activités, je me suis fier au site nommé Sourcemaking[5]. Ce site explique clairement et facilement tous les objets, lignes et intersections d'un diagramme.

7.5.1Diagrammes 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.

Tous les diagrammes d'activités pour RESA sont visibles dans les Annexes [Diagrammes D'activités]

7.6 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.

7.6.1 UML

Afin de pouvoir facilement créer le model UML de la base de données, je suis passé par le site dbdiagram.io[6]. 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.

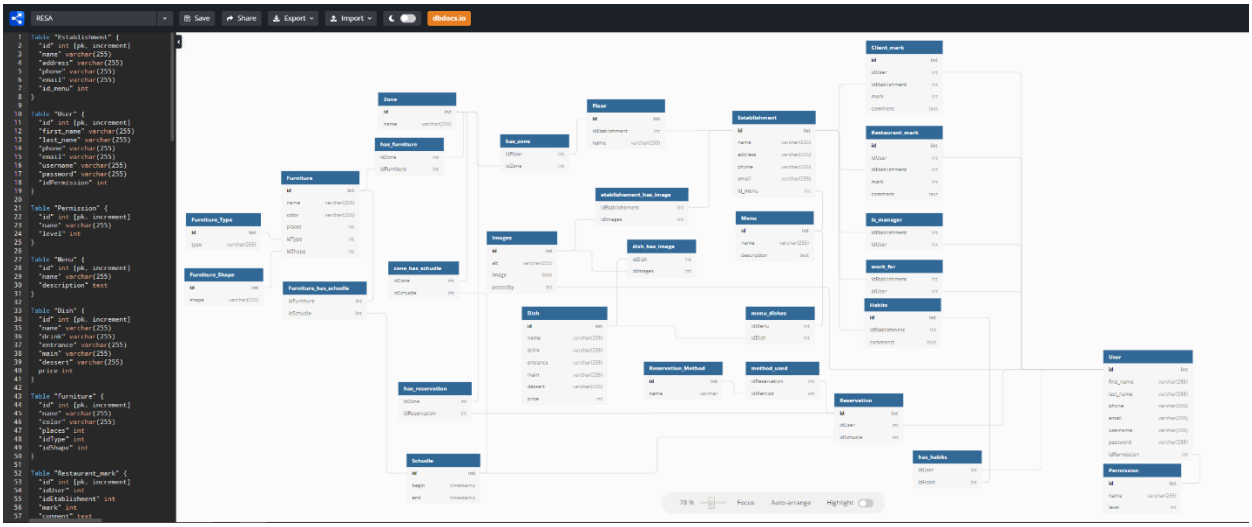


Figure 22 Aperçu interface dbdiagram.io

7.6.2Privilèges

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

- Username : resa_tech_es
- Password : WhutMerYmZeR6EHb

[5] Https://sourcemaking.com/uml/modeling-business-systems/external-view/activity-diagrams
[6] Https://dbdiagram.io

7.6.3 Structure

Ma base de données est structurée de la manière suivante :

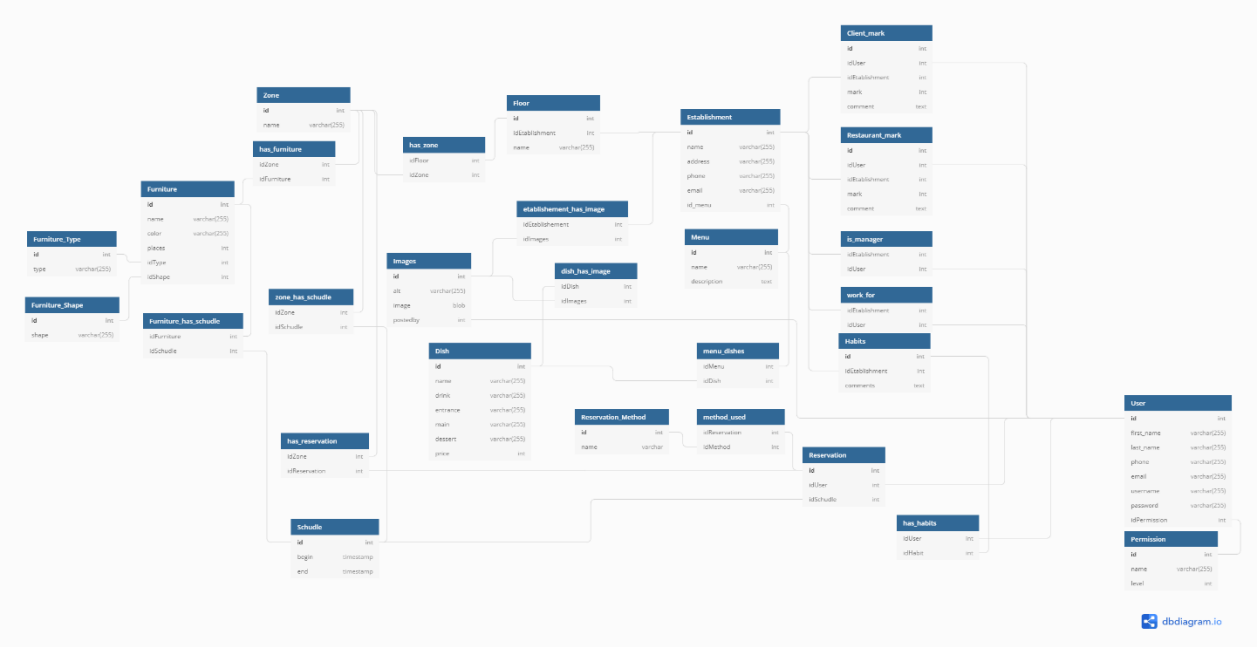


Figure 23 Aperçu MCD

7.6.4Données de tests

7.6.4.1 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

7.6.4.2 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

7.7 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.

7.7.1Structure

L'API v2 est structurée de tel manière à ce que les informations soient faciles à trouvées.

La structure et le Cheat Sheet complet de l'API sont dans les annexes.

7.7.2 Variables globales

L'API possède des variables qui sont nécessaires dans la plupart des fichiers de l'API. Ces variables sont les suivantes :

- `FullPathToAPI` qui représente le chemin complet jusqu'à l'API sur le web.
- `Key` qui est la clé avec laquelle je hash les mots de passes

Le fichier contenant les variables globales n'est pas accessible par les utilisateurs dans le navigateur.

7.7.3 Communications avec l'API

Le point le plus crucial pour le bon fonctionnement de l'API a été la communication. Comment envoyé les requêtes et comment récupérer les données résultantes.

7.7.3.1 Envois

Afin d'envoyer simplement des requêtes à l'API, je suis passé par le principe du service REST. C'est-à-dire qu'on envoie sous forme de requête http la demande à notre API.

7.7.3.1.1 Exemple

Je souhaite recevoir l'utilisateur correspondant à l'email et le mot de passe. Je commence donc par créer la requête correspondante avec les données saisies par l'utilisateur :

($username et $password sont vérifié et haché avant)

```php
$queryData = array(
      'email' => $username,
      'password' => $password
);

$link = [Lien vers l'API]."?login&email=".http_build_query($queryData);
// On récupère les données brutes
$json = file_get_contents($link);
// On converti le JSON reçu en objet PHP
$data = json_decode($json);
```

Extrait de code n°6. Envoi d'une requête http à l'API

7.7.3.2 Réceptions

L'API envoie toutes les données sous format JSON. Ce format est facilement lisible dans la majorité des langages de programmations connus ce qui rend mon API facilement compatible avec d'autre système.

Voici ce que retourne l'API lors de l'envoie de la requête de l'exemple ci-dessus en JSON :

```
{"id":"2","first_name":"Marco","last_name":"Polo","phone":"079 123 45
67","email":"marco.polo@pm.ch"}
```

Extrait de code n°7. Valeur JSON retournée par une fonction de l'API

Cette chaine de caractères est ensuite décodée par l'application grâce à la méthode `json_decode` :

```
object(stdClass)#2 (5) { ["id"]=> string(1) "2" ["first_name"]=> string(5)
"Marco" ["last_name"]=> string(4) "Polo" ["phone"]=> string(13) "079 123 45 67"
["email"]=> string(16) "marco.polo@pm.ch" }
```

Extrait de code n°8. Objet PHP créer à partir de la valeur JSON

Toutes les valeurs retournées par l'API sont donc en JSON, mais ce n'est pas tout le temps uniquement les retours des requêtes SQL sur la base de données.

Il arrive, que l'API doit mettre les valeurs en forme avant de les envoyées en JSON :

```
// Création d'un tableau provisoire
$floors = array();

// On parcours toutes les données envoyées par la base de données
foreach ($res as $value){
// On vérifie si le tableau est à 0 ou si la clé (l'id de l'étage) n'est pas déjà
utilisés comme clé dans le tableau provisoire
      if(count($floors)<0 || !IsFloorInArray($floors, $value['floor_id'])){
      // On créer un enregistrement dans le tableau avec comme clé l'id de l'éta
ge et comme valeurs le nom de l'étages et les zones

      $floors[$value['floor_id']] = array("id" => $value['floor_id'], "name" =>
$value['floor_name'], "zones" => array());
      }

      // On ajoute la zone et ses horaires dans le tableau
      array_push($floors[$value['floor_id']]["zones"], array($value['zone_name']
, $vale['zone_id'], $value['begin'], $value['end']));
}

return $floors;
```

Extrait de code n°9. Mise en forme de données avant l'envoi en json

7.7.4 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.

7.7.4.1 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.

7.7.4.1.1 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">
```

Extrait de code n°10. Formulaire à créer pour correctement envoyer les images à l'API

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();
        }
    }
}
```

Extrait de code n°11. Vérification des données reçues par un formulaire

La fonction CheckData va vérifier que toutes les données 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 données de l'établissement 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);
```

Extrait de code n°12. Création de la requête http pour créer un établissement

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]);
```

Extrait de code n°13. Fonction de sauvegarde de l'image pour un établissement

7.7.4.2 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.

7.7.4.2.1 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 l'image
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="">
```

Extrait de code n°14. Balise HTML d'une image avec comme chemin l'image dans l'API

7.7.5Les établissements

La pièce centrale de mon application est donc la liste des établissements inscrits. Chaque établissement, possède des étages. Les étages possèdent des zones et les zones possèdent des fournitures.

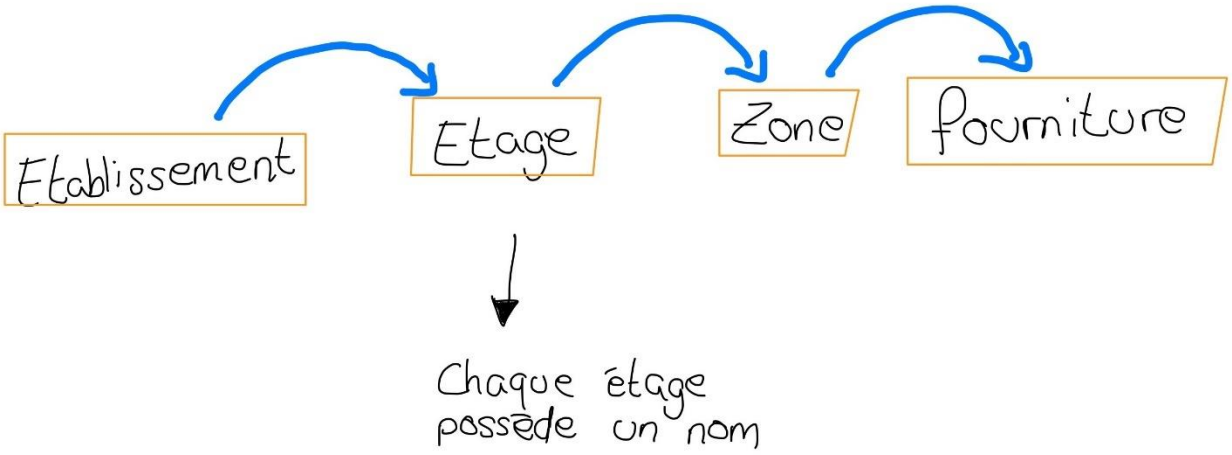


Figure 24 Schéma de lien entre les tables principales

7.7.5.1 Les Zones et les fournitures

Les zones et les fournitures composent ensemble les éléments réservables par les clients et qui possèdent les horaires de disponibilités. Le schéma ci-dessous montre comment une zone est construite ainsi que les liens entre les tables.

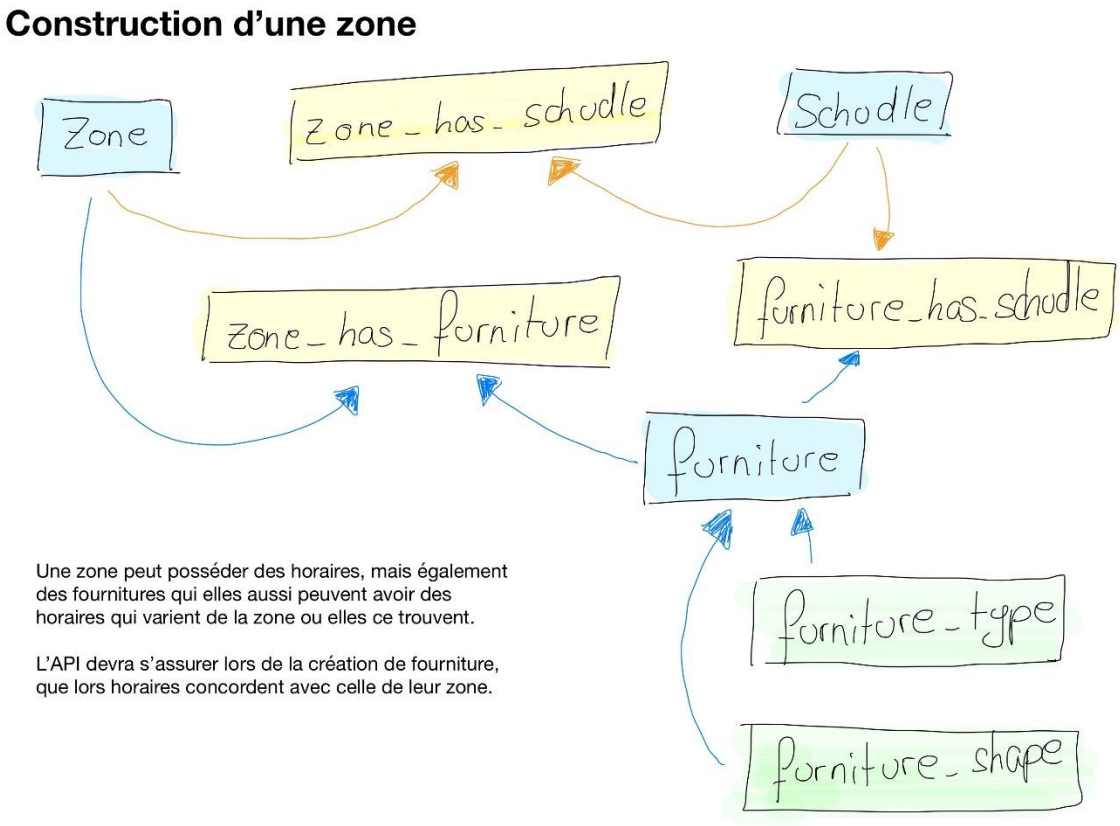


Figure 25 Schéma des liens entres les tables pour une zone

7.7.5.2 Le fonctionnement des zones

[parler du fonctionnement particulier des zones]

7.8 Gestion du temps

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

7.8.1 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 »

7.8.2 Classer les tâches

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

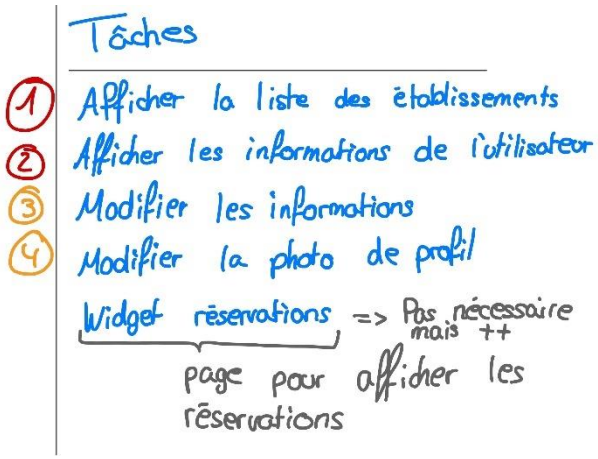


Figure 26 Liste des tâches pour la page profil

Le tableau affiche en rouge les numéros des tâches réellement critiques 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.

7.8.3 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.

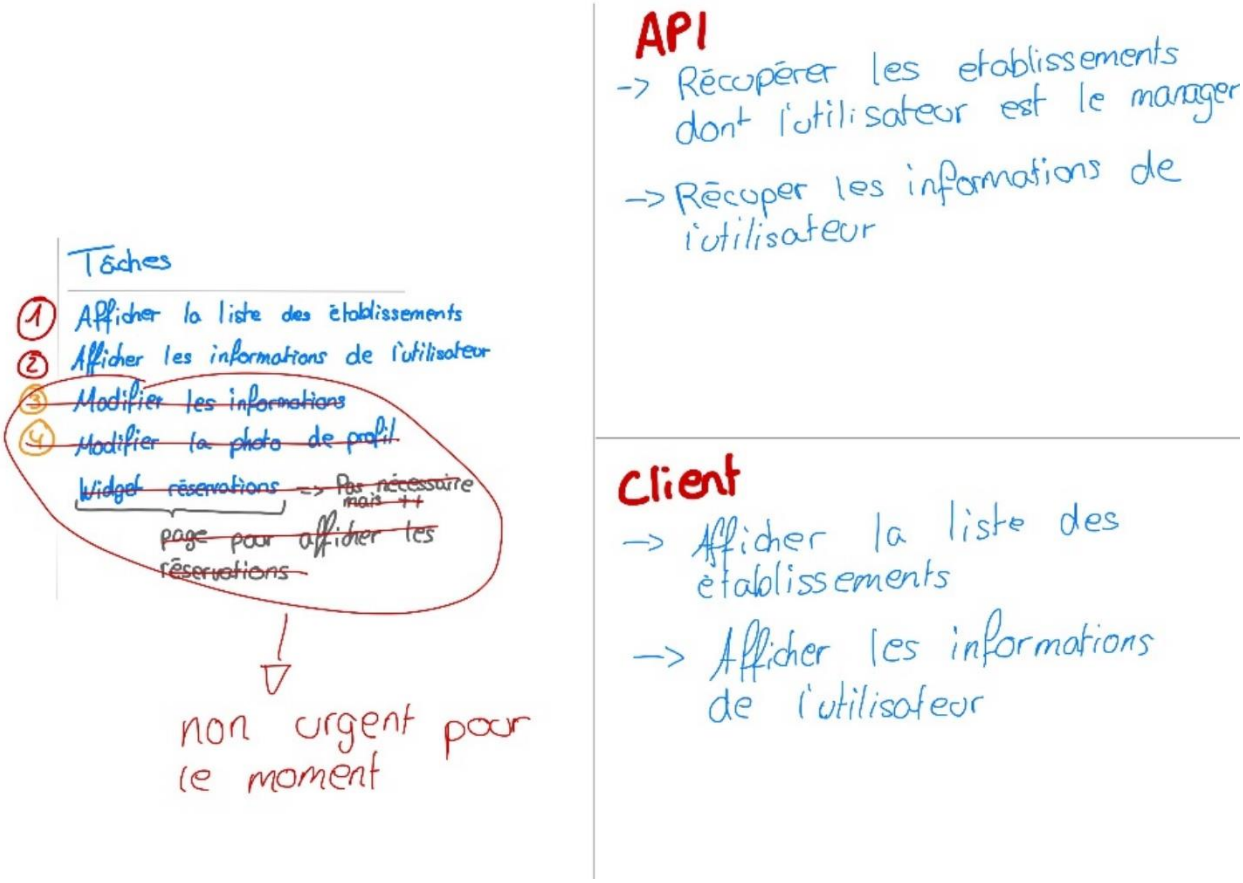


Figure 27 Séparation des tâches client et serveur (API)

7.8.4 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.

7.9 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.

7.9.1 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.

7.9.1.1 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 partie 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 décidé de faire mon journal de bord en markdown.

7.9.1.1.1 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
```

Extrait de code n°15. Séparation des jours dans le logbook

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

7.9.1.1.2 Extrait du journal de bord lors de réflexions

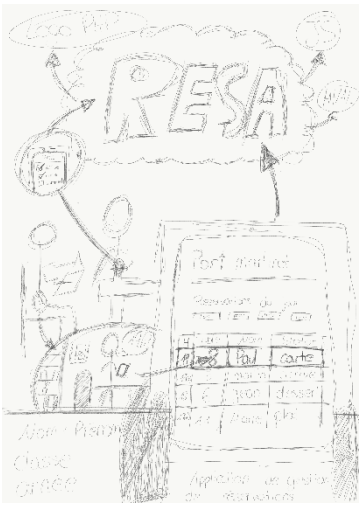
```
- Je suis en train de faire le login d'un utilisateur, je me suis rendu compte qu
e je devais faire la différencation de si il s'agissait d'un login client ou d'un
 login utilisateur local.
  - Il faut que j'ajoute à l'API une fonctionnalité qui gère la différence
    - Je vais donc créer cette fonctionnalité
    - Afin de pouvoir ajouter cette fonctionnalité, j'ai un petit souci... Un uti
listeur admin peut se connecter dans tous les restaurants, il faut donc que ma re
quête 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 ... Pou
r 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
```

Extrait de code n°16. Extrait du logbook disponible sur Github[7]

[7] https://github.com/ConstantinHrrmn/Travail_diplome_ES_2020/blob/master/logbook.md

7.9.1.2 Création de croquis

En plus du journal de bord, je faisais beaucoup de schémas afin de toujours pouvoir visualiser le résultat avant de créer le rendu final. Pour illustrer ma manière de faire, je vais utiliser l'exemple du poster.

La première esquisse	La deuxième esquisse	Le résultat final
		

Comme on peut l'analyser, je passe par plusieurs étapes afin de pouvoir visualiser au mieux le résultat final. En effet pour le poster, je souhaitais faire passer le message de la simplicité et de la connectivité de RESA. J'ai donc d'abord dessiné une tablette (outil principalement utilisé avec RESA), puis un décor et ensuite les liens entre les protagonistes et la tablette (le nuage gris central). Puis j'ai ajouté la couleur et les lignes nettes pour arriver au résultat final.

Les deux schémas et le résultat final sont en plein formats dans les annexes : Images (pleins format).

7.9.1.3 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.

7.9.2 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.

8 Tables des figures

9 Tables des extraits de code

10 Glossaire

11 ANNEXES

11.1 Diagrammes D'activités

11.1.1 Création de compte

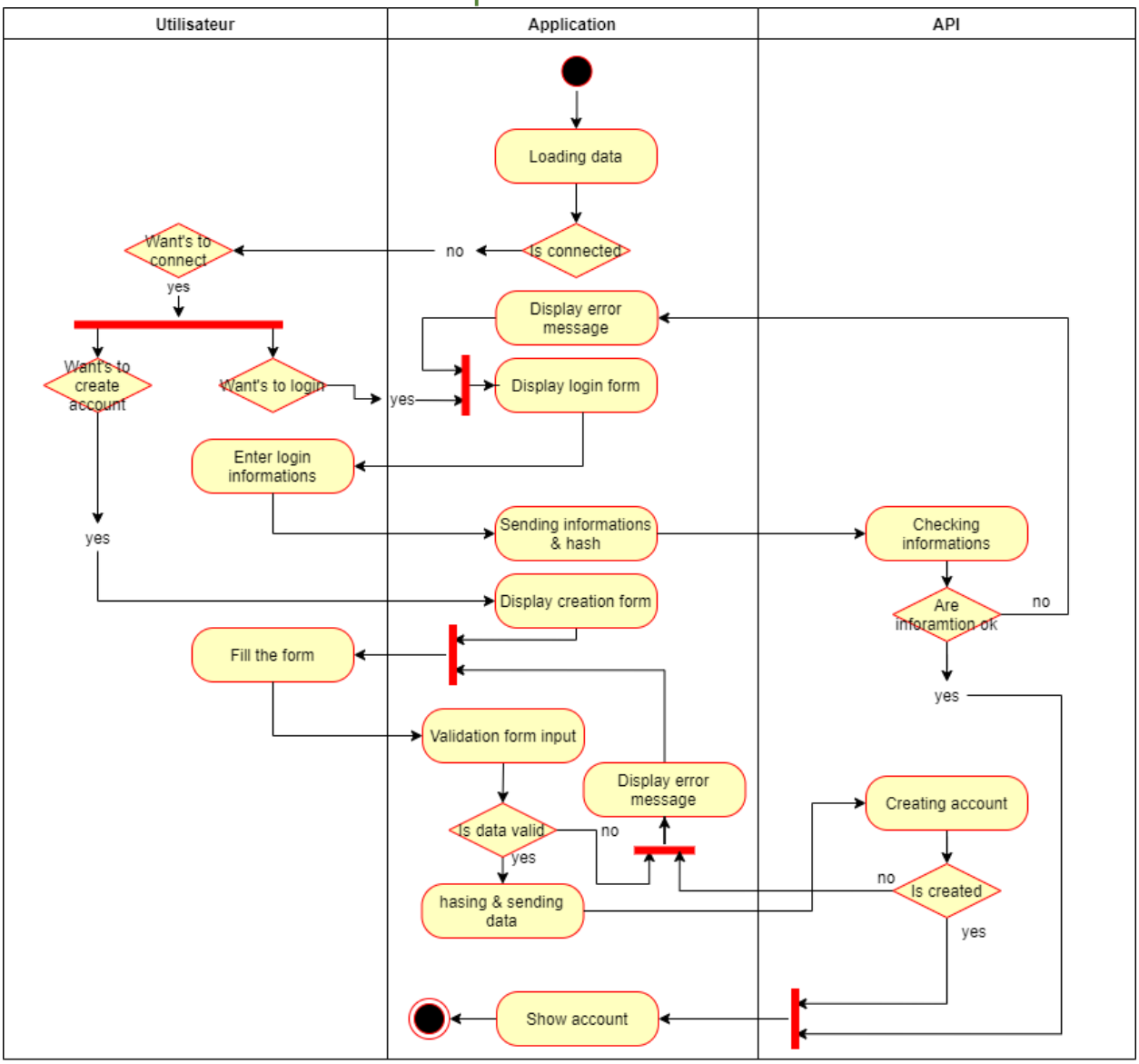


Figure 28 Diagramme d'activité - création d'un compte et login

11.1.2 Réservation

11.2 Structure de l'AP

```
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
```

11.3 Cheat Sheet de l'API

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

11.3.1 User

Lien : `/api/v2/user/`

11.3.1.1 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)

11.3.1.2 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

11.3.2 Floor

Lien : `/api/v2/etablishment/floor/`

11.3.2.1 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`)

11.3.2.2 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`

11.3.3 Schedule

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

11.3.3.1 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

11.3.4 Images

Lien : `/api/v2/images/`

11.3.4.1 Lecture

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

11.3.4.2 Création

`Mise en ligne d'une photo`

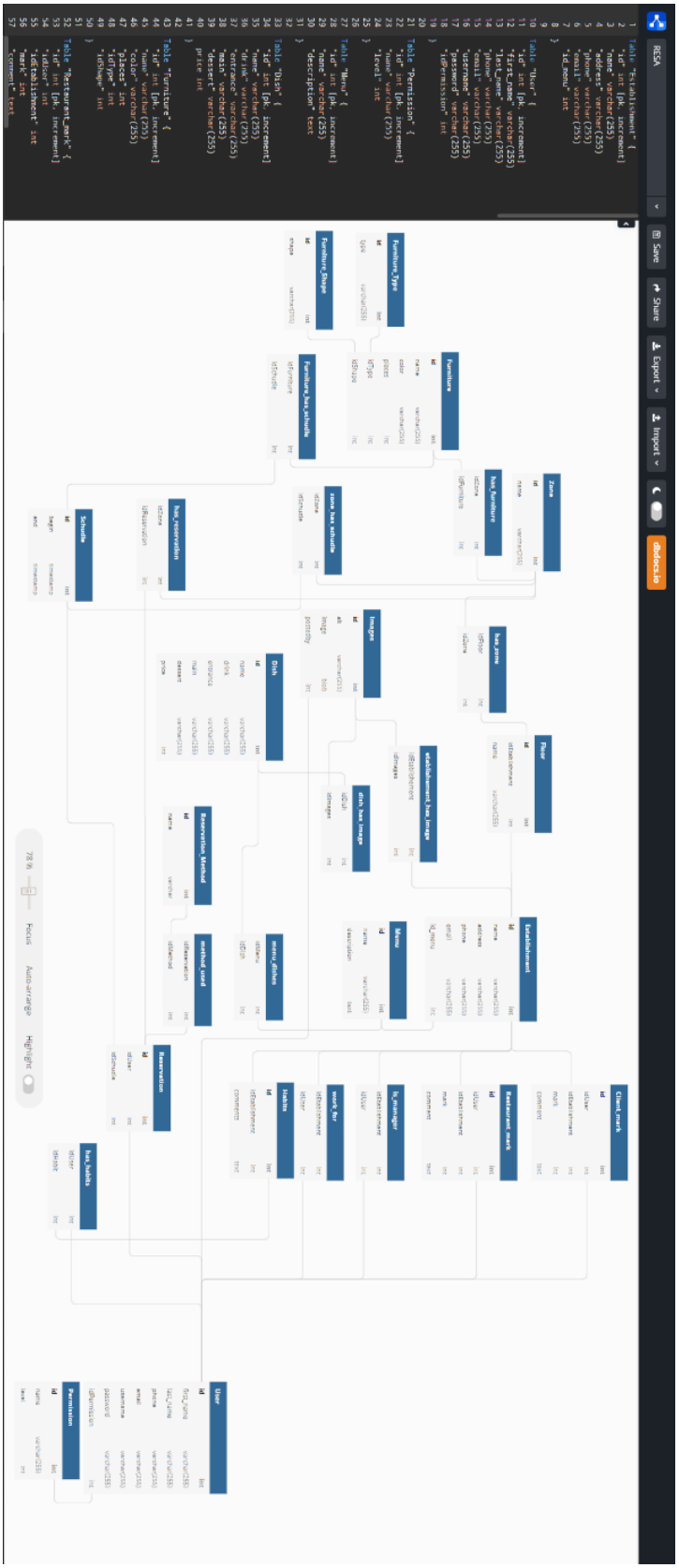
  - Afin de mettre en ligne une photo il faut :

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

11.4 Images (pleins format)

11.4.1 Interface dbdiagram.io



11.4.2 MCD de la base de données

11.4.3 Esquisse n°1 Poster

11.4.4 Esquisse n°2 Poster

11.4.5 Poster final RESA