

Travail de Diplôme

The logo consists of the word "RESA" in a bold, sans-serif font. The letters are filled with a green, geometric pattern of triangles, giving it a modern and technical appearance.

Constantin Herrmann

avril – juin 2020

M. Francisco Garcia

CFPT-I Technicien ES

Version 0.6 (révision avant évaluation intermédiaire n°3)

1 Table de matières

1	Table de matières	2
2	Résumé (intermédiaire)	7
3	Abstract (intermediary).....	7
4	Introduction	8
5	Analyse de l'existant	8
5.1	La Fourchette.....	8
5.1.1	Clients	9
5.1.2	Partie manager de « the Fork »	10
6	Analyse fonctionnelle	13
6.1	La structure de RESA.....	13
6.2	Interfaces	14
6.2.1	Template	14
6.2.2	RESA Client	15
6.2.2.1	Page d'accueil	15
6.2.2.2	Page de l'établissement.....	17
6.2.2.3	Réservation par RESA	18
6.2.2.4	Se connecter	18
6.2.2.5	Profil utilisateur.....	19
6.2.3	Accès aux applications des managers RESA	20
6.2.4	RESA Blog	21
6.2.4.1	Création d'un établissement.....	21
6.2.4.2	Accueil.....	21
6.2.4.3	Paramètres	22
6.2.4.4	Les images	22
6.2.5	RESA Pro	23
6.2.5.1	Accueil.....	23
6.2.5.2	Réservations	23
6.2.5.3	Paramètres	24
6.2.6	RESA Full	24
6.2.6.1	Paramètres	24
6.2.6.2	Réservations	25
6.2.6.3	Employés.....	25
6.2.6.4	Plan de table	26
6.3	Les droits des utilisateurs	26
6.3.1	Utilisateurs.....	26
6.3.1.1	Administrateur	27

6.3.1.2	Manager (et chef)	27
6.3.1.3	Serveur.....	27
6.3.1.4	Stagiaire et externe	28
6.3.1.5	Client	28
7	Analyse organique	29
7.1	Mise en place.....	29
7.1.1	Github	29
7.1.2	Trello	29
7.2	Pré-travail	29
7.2.1	Programmation.....	29
7.2.2	Installation de React.....	30
7.2.3	Conventions	30
7.2.3.1	En-tête de fichier	30
7.2.3.2	En-tête de fonction	31
7.2.3.3	Diagrammes d'activités	31
7.2.4	Organisationnel	32
7.3	Environnement.....	33
7.3.1	Laragon	33
7.3.2	Visual Studio Code.....	33
7.3.3	EDUGE.....	33
7.3.4	Github Desktop.....	33
7.4	Schémas de fonctionnements	34
7.4.1	RESA.....	34
7.5	Diagrammes de fonctionnement de RESA.....	34
7.6	Diagrammes d'activités.....	34
7.6.1	Diagrammes d'activités de RESA	35
7.7	Base de données.....	35
7.7.1	UML.....	35
7.7.2	Privilèges.....	35
7.7.3	Structure	36
7.7.4	Données de tests	36
7.7.4.1	Utilisateurs.....	36
7.7.4.2	Etablissements	36
7.7.4.3	Les niveaux d'abonnement des restaurants	36
7.8	API	37
7.8.1	Structure	37
7.8.2	Variables globales	37

7.8.3	Communications avec l'API	37
7.8.3.1	Envoi	37
7.8.3.1.1	Exemple.....	37
7.8.3.2	Réceptions	37
7.8.4	Gestion des images	38
7.8.4.1	Mise en ligne d'une image	38
7.8.4.1.1	Exemple.....	39
7.8.4.2	Récupérer les images	40
7.8.4.2.1	Exemples.....	40
7.8.5.1	Le nombre de places disponibles grâce aux routines.....	40
7.8.5.1.1	Les routines MySQL.....	40
7.8.5.1.2	Création de ma routine	41
7.8.5.2	Système de réservations.....	42
7.8.6	Les établissements.....	44
7.8.6.1	Les Zones et les fournitures.....	44
7.8.6.2	Le fonctionnement des zones	45
7.9	L'application	45
7.9.1	Affichage du statut ouvert / fermé des restaurants	45
7.9.2	Login vs Fast-login vs Manager-login	46
7.9.2.1	Login.....	46
7.9.2.2	Fast-login.....	46
7.9.2.3	Manager-login	47
7.9.3	Le calendrier de réservation.....	47
7.10	Gestion du temps	47
7.10.1	Lister les tâches	48
7.10.2	Classer les tâches	48
7.10.3	Séparation des tâches client et serveur.....	49
7.10.4	Conclusion.....	49
7.11	Raisonnements	49
7.11.1	Réflexions personnelles	49
7.11.1.1	Le journal de bord	50
7.11.1.1.1	Structure	50
7.11.1.1.2	Extrait du journal de bord lors de réflexions	50
7.11.1.2	Création de croquis	51
7.11.1.3	Analyse.....	51
7.11.2	Communications avec M. Garcia	51
7.11.3	Communication avec Mme. Perdrizat	51

7.11.3.1	Réorganisation	52
8	Tables des figures.....	53
9	Tables des extraits de code	55
10	Glossaire	56
11	ANNEXES	57
11.1	Diagrammes D'activités	57
11.1.1	Création de compte	57
11.1.2	Réservation	58
11.1.3	Laisser un commentaire	59
11.2	Diagrammes de fonctionnement	60
11.2.1	Login.....	60
11.2.2	Redirection sur bonne application.....	60
11.3	Structure de l'AP	61
11.4	Cheat Sheet de l'API	62
11.4.1.1	Create.....	62
11.4.1.1.1	Création d'un établissement sans manager	62
11.4.1.1.2	Création d'un établissement avec un manager	62
11.4.1.1.3	Création en passant pas une form	62
11.4.1.2	Get.....	63
11.4.1.2.1	D'après un id	63
11.4.1.2.2	L'id du dernier insérer	63
11.4.1.2.3	Tous les établissements d'un manager (utilisateur).....	63
11.4.1.2.4	Niveau d'abonnement.....	64
11.4.1.2.5	L'horaire des zones	64
11.4.1.2.6	Horaire du restaurant de la journée (actuelle).....	64
11.4.1.2.7	Horaire et places disponible pour une date.....	64
11.4.1.2.8	Le jour de la semaine	65
11.4.1.2.9	Récupère les horaires de la semaine pour le restaurant	65
11.4.2	Etages	65
11.4.2.1	Get.....	65
11.4.2.1.1	Tous les étages de l'établissement	65
11.4.2.2	Create.....	66
11.4.2.2.1	Création par l'API	66
11.4.2.2.2	Création par un formulaire.....	66
11.4.3	Zones	66
11.4.3.1	GET	66
11.4.3.1.1	L'id de la dernière zone créée	66

11.4.3.1.2	Les places dans la zone	67
11.4.3.1.3	Tous les horaires de la zone	67
11.4.3.2	CREATE	67
11.4.3.2.1	Création par l'API	67
11.4.3.2.2	Lien entre étage et zone	68
11.4.3.2.3	Création par un formulaire	68
11.4.4	Fournitures	68
11.4.4.1	GET	68
11.4.4.1.1	Les fournitures d'une zone	68
11.4.4.1.2	Les fournitures d'un établissement	69
11.4.5	Images	69
11.4.5.1	GET	69
11.4.5.1.1	Les informations d'après l'id	69
11.4.5.1.2	Les images pour un établissement	69
11.4.5.1.3	Redirection sur l'image	70
11.4.6.1	GET	70
11.4.6.1.1	Récupérer les plats d'un restaurant	70
11.4.6.1.2	Récupérer tous les plats de la base	70
11.4.6.1.3	Récupérer les menus d'un restaurant	71
11.4.6.1.4	Récupérer l'id du menu du restaurant	71
11.4.6.1.5	Récupérer la carte complète du restaurant	71
11.4.6.1.6	Récupérer les réservations pour une intervalle d'heure	72
11.4.6.1.7	Vérifier la possibilité de réserver	72
11.5	Images (pleins format)	74

2 Résumé (intermédiaire)

RESA est une application intuitive permettant, d'une part, au restaurateur de centraliser ses réservations (téléphone, e-mail, en ligne), gérer son plan de table facilement, obtenir des statistiques clients et promouvoir son établissement en ligne. D'autre part, elle permet au client de réserver une table dans l'établissement de son choix rapidement et simplement avec disponibilité en temps réel.

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)

RESA is an intuitive application that allows restaurateurs to centralize their reservations (telephone, e-mail, online), manage their seating plan easily, obtain customer statistics and promote their establishment online. On the other hand, it allows the customer to book a table in the establishment of his choice quickly and simply with real-time availability.

RESA is a web application based on the php language and relies on the services of its API.

This document covers the entire project through a functional and organic analysis that precisely describes 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 "lafourchette"

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

5.1.1 Clients

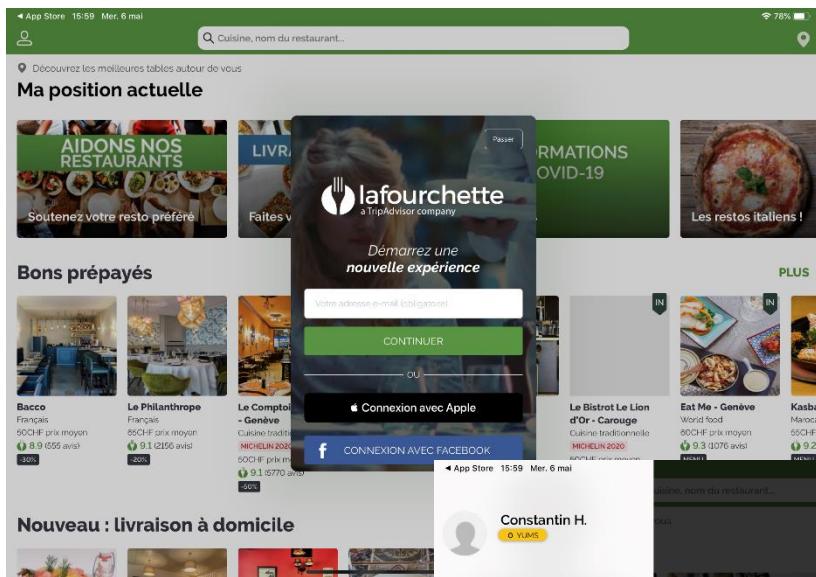


Figure 2 Page d'accueil

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 a publiés sur le site afin de partager avec les autres utilisateurs.

Figure 3 Menu de l'utilisateur

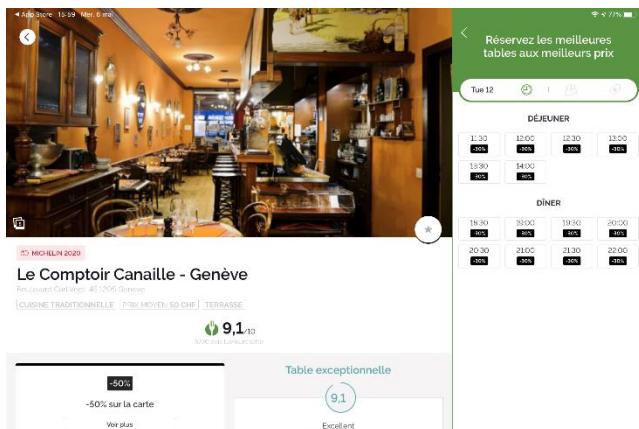
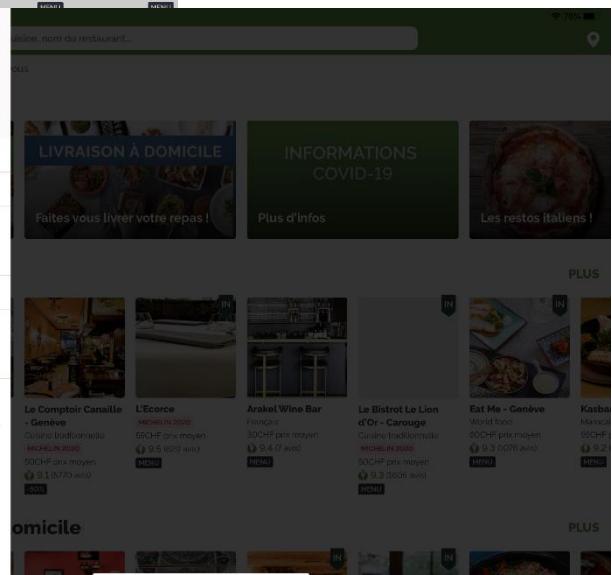


Figure 4 Page du restaurant

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 restaurants sont affichés sous forme de cartes que l'on peut sélectionner afin d'avoir plus d'informations sur le restaurant.



Lorsque 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. Une fois choisi, on lui demande le nombre de personnes et les réductions ou bons qu'il souhaite appliquer ou non.

5.1.2 Partie manager de « the Fork »

La première page sur laquelle tombe le manager en se rendant sur le site de myfourchette¹ est la page de login manager.

Cette page permet au manager de se connecter à l'aide de son email et son mot de passe.

Si le manager n'est pas encore enregistré, il peut le faire en cliquant sur le lien en bas de l'écran. Une analyse sur la création d'un restaurant se trouve plus bas.

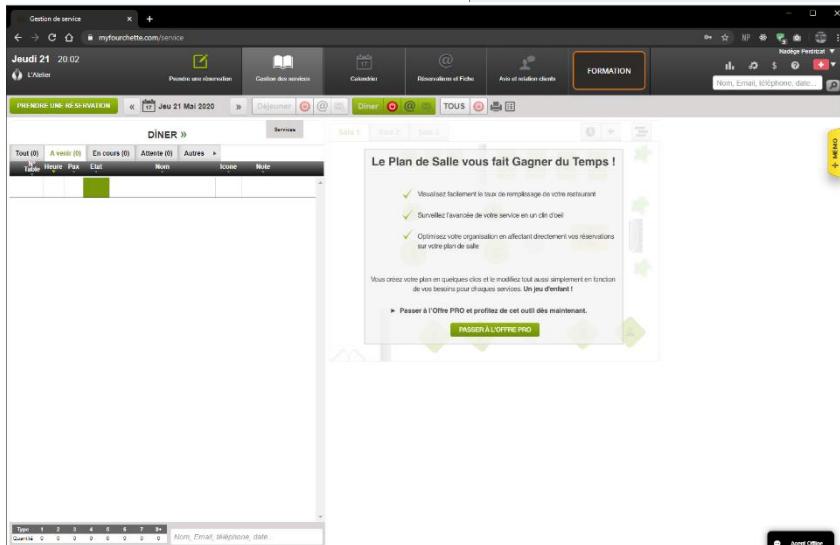


Figure 5 Page d'accueil

Si le manager a l'abonnement payant, alors il accède aux pages de gestion des réservations (également bloqué avec un compte gratuit), à la page du calendrier, à la page des avis clients et à la page des clients.

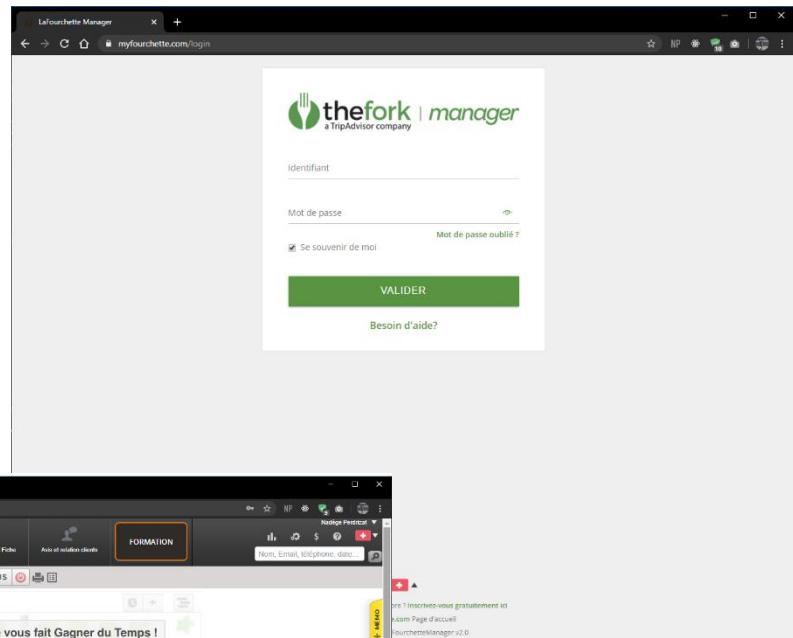


Figure 6 Login manager

Une fois connecté, le manager est redirigé sur la page d'accueil. Il peut à partir de ce point, facilement accéder aux différents menus.

L'option du plan de tables est bloquée, car il faut prendre l'abonnement payant.

¹ <https://www.myfourchette.com/>

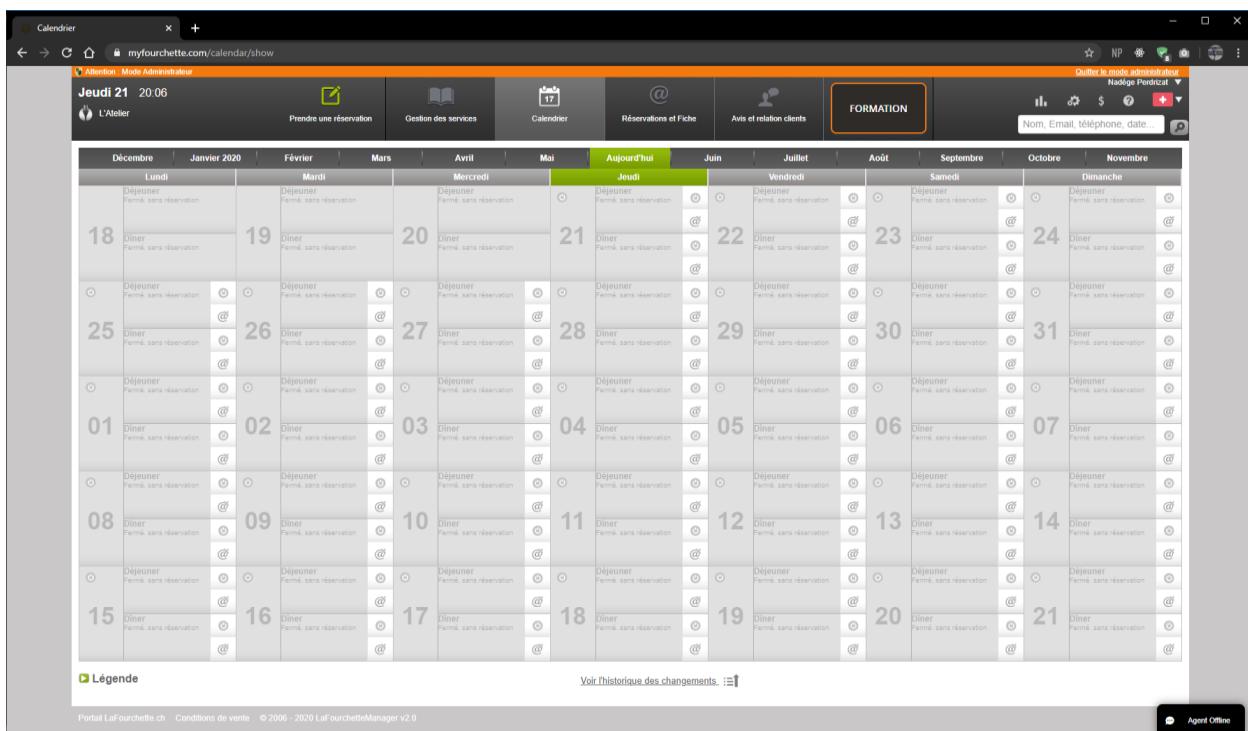


Figure 7 calendrier du manager

Sur la page du calendrier, le manager a un aperçu simple des réservations qu'il y a chaque jour. Il a aussi, la possibilité d'ouvrir ou de fermer son restaurant aux réservations pour un jour donné.

FICHE CLIENT

Civilité	<input type="text"/>
Nom *	<input type="text"/>
Prénom	<input type="text"/>
* Obligatoire	
<input checked="" type="button"/> Profil Personnel <input type="button"/> Profil Professionnel <input type="button"/> Préférences	
INFORMATIONS GÉNÉRALES	
Date de naissance	<input type="text"/>
Langue	<input type="text"/> Français
Status	<input type="button"/> Normal
VIP	<input type="checkbox"/>
CONTACTS PERSONNELS	
Email	<input type="text"/>
Téléphone Portable	<input type="text"/> Suiss...
Téléphone Fixe	<input type="text"/> Suiss...
Fax	<input type="text"/> Suiss...
Adresse	<input type="text"/>
Complément d'adresse	<input type="text"/>

Figure 8 Formulaire de création d'un client

Il peut afficher tous les clients ayant réservé une table dans le restaurant ainsi que de créer un client propre au restaurant.

Il faut alors entrer les données générales du client que l'on souhaite ajouter.

La Fourchette propose l'option « VIP » qui met en avant les réservations venant de ces clients. Ils sont alors considérés avec plus de soins.

Le manager peut sélectionner un « statut » pour le client. Ce statut affectera les prochaines réservations.

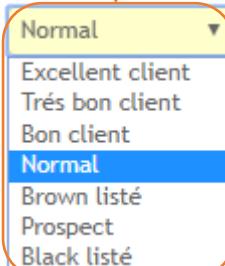


Figure 9 Dropdown menu du statut

Le formulaire pour la création d'une réservation par un serveur est composé des champs importants et qui peuvent être rapidement transmis par téléphone.

Le champ « N° Table » indique au serveur qu'il doit manuellement entrer le numéro de la table pour la réservation. Malheureusement, je n'ai pas les droits pour faire ces tests. Dans la partie client, il est possible de choisir la civilité, le bouton « compléter client » permet de rechercher dans la base de données du restaurant, les informations manquantes si existantes dans les données déjà entrées. Les tests m'ont montré qu'en entrant le nom du client, la fourchette propose automatiquement la fiche du client correspondant.

Figure 12 Formulaire de création de réservation

Figure 10 Sélection heure réservation manager



Figure 11 Sélection date réservation manager

6 Analyse fonctionnelle

L'analyse fonctionnelle reprend tous les éléments qui ont servi à créer l'application. Dans cette partie, nous analyserons les interfaces et le processus de création.

6.1 La structure de RESA

Mon travail de diplôme nommé RESA, est un ensemble d'applications pour clients et restaurateurs afin de faciliter la gestion des réservations.

Il existe 4 applications RESA :

1. RESA client : L'application permet aux clients (visiteurs) de visualiser tous les restaurants de RESA et d'y réserver une table. L'utilisateur peut également laisser un avis, des photos et une note au restaurant après son repas.
2. RESA Blog : Cette application permet au restaurateur d'enregistrer une première fois son établissement gratuitement dans l'application. Depuis là, il pourra gérer les informations de base qui s'afficheront dans RESA comme le nom, l'adresse, les photos, les menus, les horaires, etc.
3. RESA Pro : Permet au restaurateur de bénéficier de tout ce qui est inclus dans RESA Blog, mais également la gestion des réservations automatisées. Pour ce faire, le restaurateur peut renseigner à RESA les tables qu'il possède afin que les réservations et leur gestion puissent se faire automatiquement.
4. RESA Full : L'expérience complète de RESA. Cette application en plus de reprendre tous les points des catégories Blog et Pro, permet d'avoir un plan de table visuel ainsi que la gestion totale de son établissement, ce qui veut dire, les zones, les horaires indépendants, la gestion des employés et bien plus.

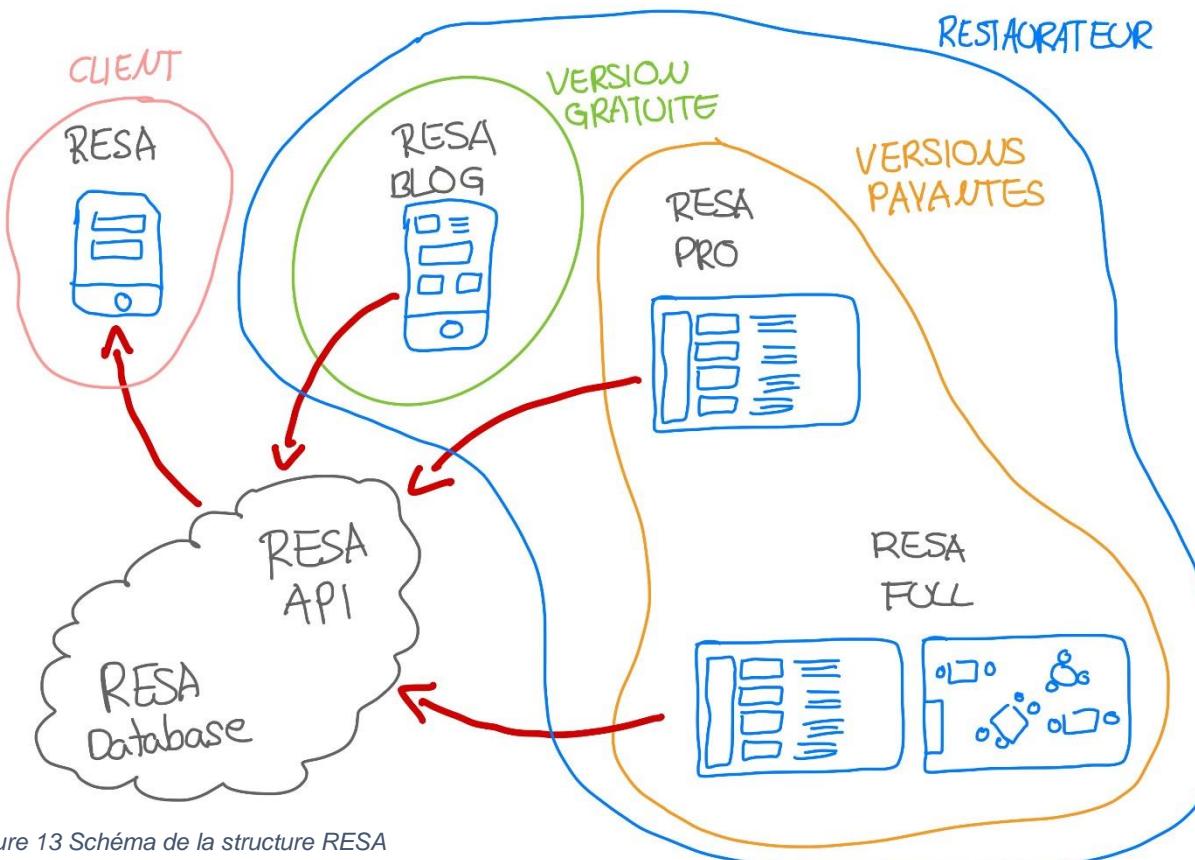


Figure 13 Schéma de la structure RESA

6.2 Interfaces

6.2.1 Template

Afin de ne pas passer trop de temps sur la création de mes vues ainsi que sur le design des composants, j'ai décidé de prendre un Template qui possède plusieurs styles de widgets, menus, interfaces et formulaires. Le Template que j'ai choisi ce nomme : Costic HTML². Il est disponible sur le site themeforest³.

Ce Template a été pensé pour les restaurants, ce qui était parfait pour mon projet. Une fois téléchargé et installé, j'ai changé les thèmes principaux afin que ceux-ci se rapprochent du thème de RESA (vert et gris). J'ai commencé à créer de nouvelles pages en y intégrant des modules.

Par exemple, pour la page d'accueil de l'application RESA Client :

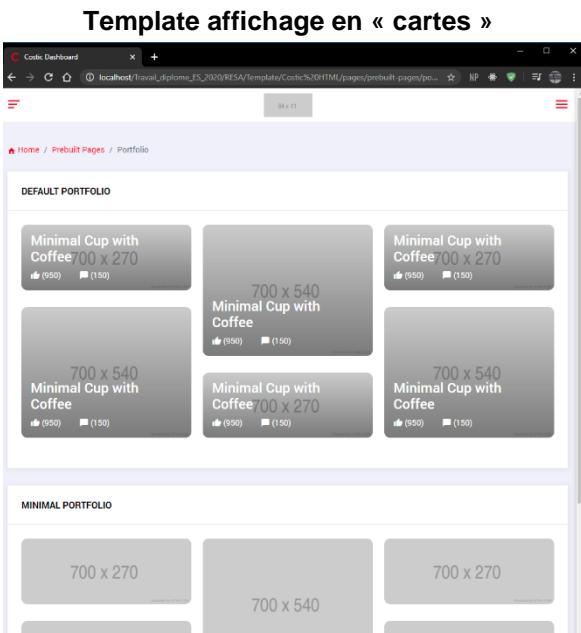


Figure 14 Capture d'écran du template

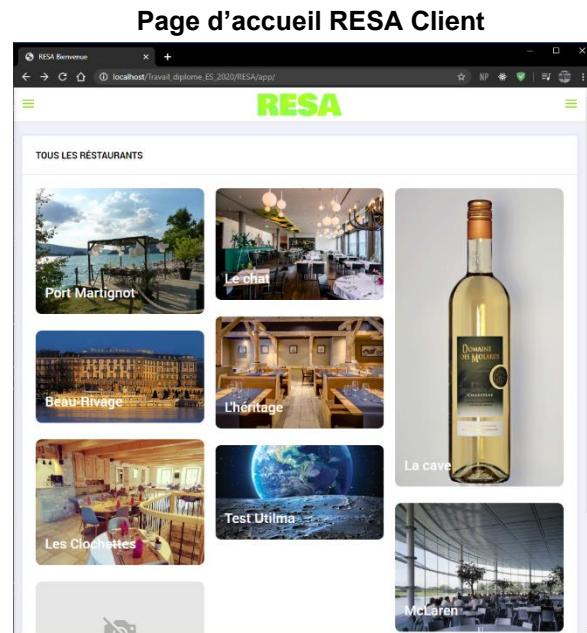


Figure 15 Capture d'écran de la page d'accueil RESA

On constate que j'ai uniquement utilisé les modules créés par themeforest. L'aménagement des pages a été repensé par moi-même tout au long du projet. Le template est facilement modulable et très bien structuré, ce qui m'a laissé beaucoup de possibilités.

² <https://themeforest.net/item/costic-restaurant-admin-dashboard-html5-template/25634499>

³ <https://themeforest.net/>

6.2.2 RESA Client

RESA client est l'application utilisée par tous les clients souhaitant réserver une table dans les restaurants inscrits. Elle possède une interface simple d'utilisation en proposant uniquement l'essentiel.

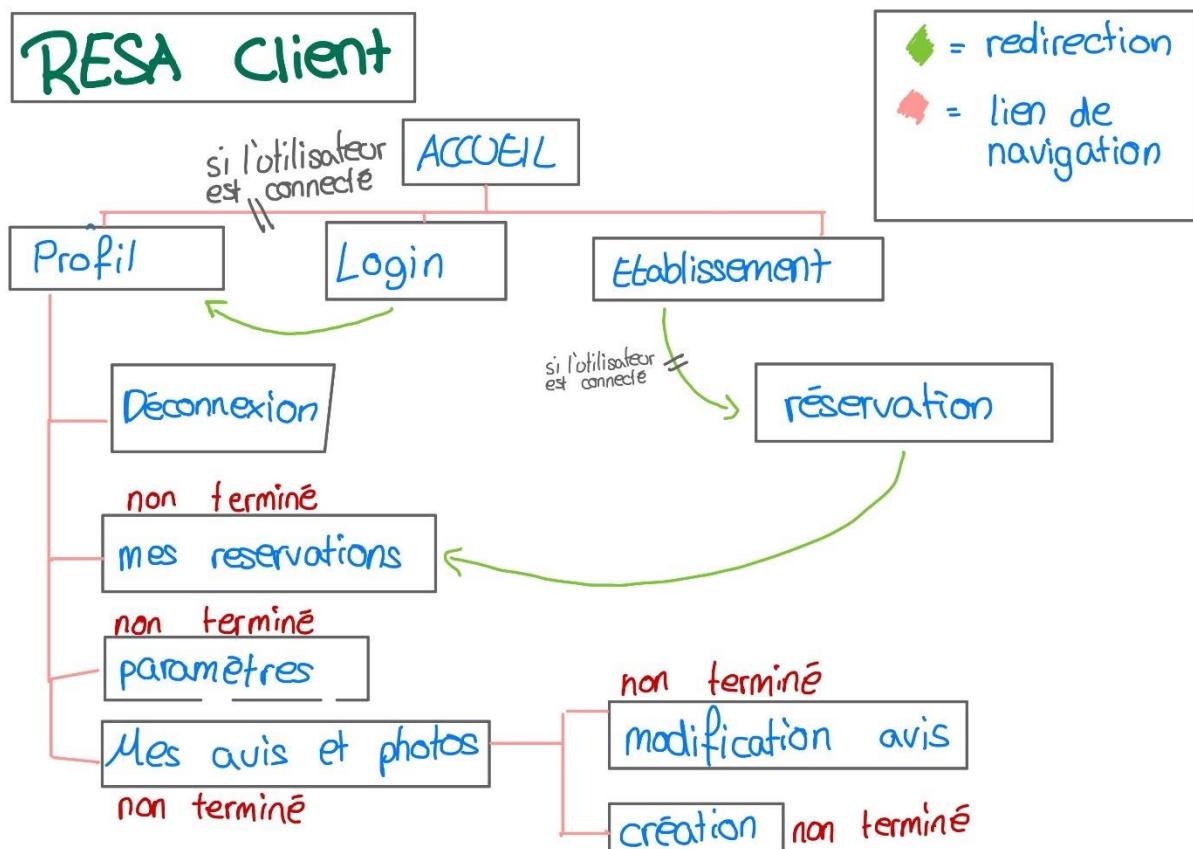
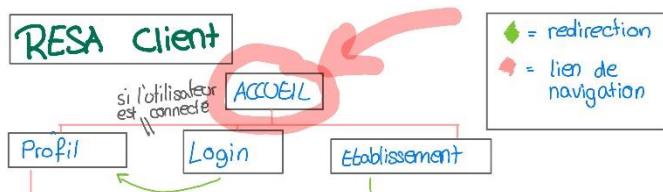


Figure 16 Schéma de navigation RESA Client

6.2.2.1 Page d'accueil



La page d'accueil se trouve à la racine de l'application client.

Figure 17 On est là - Accueil RESA Client

L'objectif principal de la page d'accueil a été de rendre l'affichage des restaurants dynamique et simple à voir. La liste permet de parcourir tous les établissements de la base. Il y a également un filtre afin de trier tous les restaurants par nom. Ce filtre permet à l'utilisateur de retrouver son restaurant et d'y réserver une table.

L'utilisateur voit les restaurants actuellement ouverts et ceux qui ne le sont pas. Même si le restaurant affiche fermé, l'utilisateur peut se rendre sur sa page afin de voir ses informations. Il peut réserver une table si le manager de l'établissement a souscrit aux applications payantes.

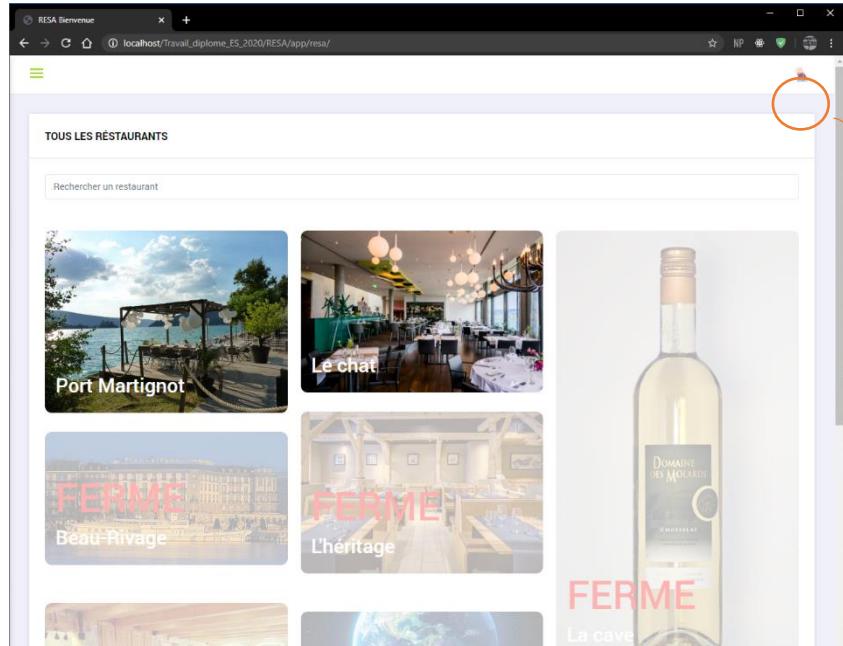
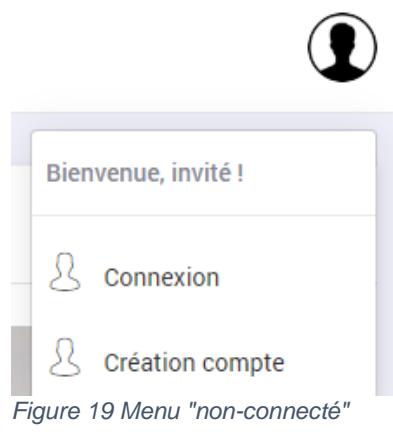


Figure 18 Page d'accueil de RESA client



En haut à droite, l'utilisateur peut cliquer sur le menu afin de soit se connecter, soit créer un compte.

Si l'utilisateur est connecté, d'autres choix lui sont proposés :

- 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 19 Menu "non-connecté"

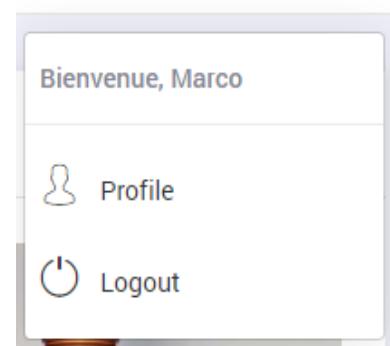


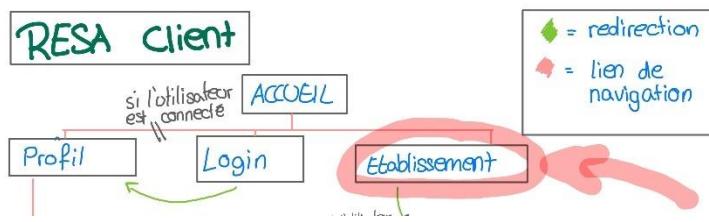
Figure 20 Menu "connecté"

6.2.2.2 Page de l'établissement

La page de l'établissement est accessible depuis la page d'accueil en sélectionnant un restaurant de la liste de mosaïques.

Figure 21 On est là - page de l'établissement

Cette page a pour but d'afficher les informations de base du restaurant, c'est-à-dire, le nom, l'adresse, le numéro de téléphone, l'email et le menu si le restaurant l'a mis à disposition. La page du restaurant peut avoir deux types d'affichages :



Le premier affiche le bouton pour téléphoner ou envoyer un email pour réserver une table

Cette méthode s'applique aux restaurants possédant un compte blog.



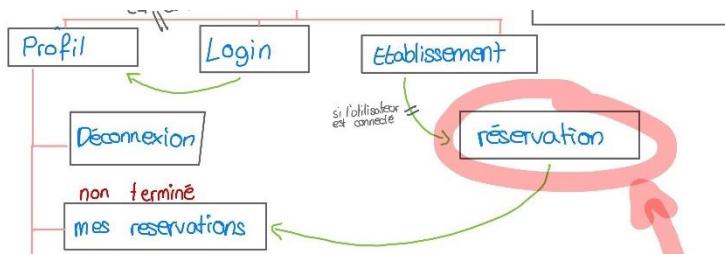
Port Martignot

Phone : 022 354 75 34
Email : info@portmartignot.ch

Lun	Mar	Mer	Jeu	Ven	Sa	Dm
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31

Le deuxième affichage remplace les boutons par un calendrier interactif afin de permettre à l'utilisateur de sélectionner une date et de passer à la page de réservation.

6.2.2.3 Réservation par RESA

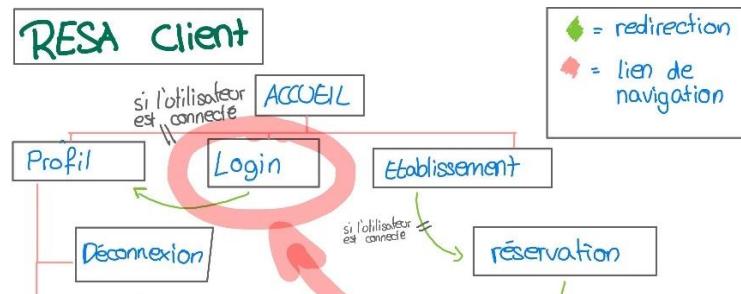


Cette page est réservée pour les réservations dans les établissements possédant l'un des abonnements payants.

Figure 22 On est là - page de réservation par RESA

Après avoir sélectionné la date sur le calendrier, l'utilisateur est redirigé sur la page de réservation. Depuis cette page, l'utilisateur se voit proposer tous les horaires disponibles dans le restaurant pour la journée. Lorsqu'il clique dessus, l'application affiche le nombre de réservations possibles.

6.2.2.4 Se connecter



L'utilisateur accède à cette page en sélectionnant le lien dans la barre de navigation latérale ou en sélectionnant « se connecter » sur l'icône en haut à droite.

Figure 23 On est là - Page de login

Pour accéder à son compte, ses informations et ses réservations, un utilisateur doit passer par la page de login. Cette dernière permet à un utilisateur de se connecter grâce à son email et son mot de passe.

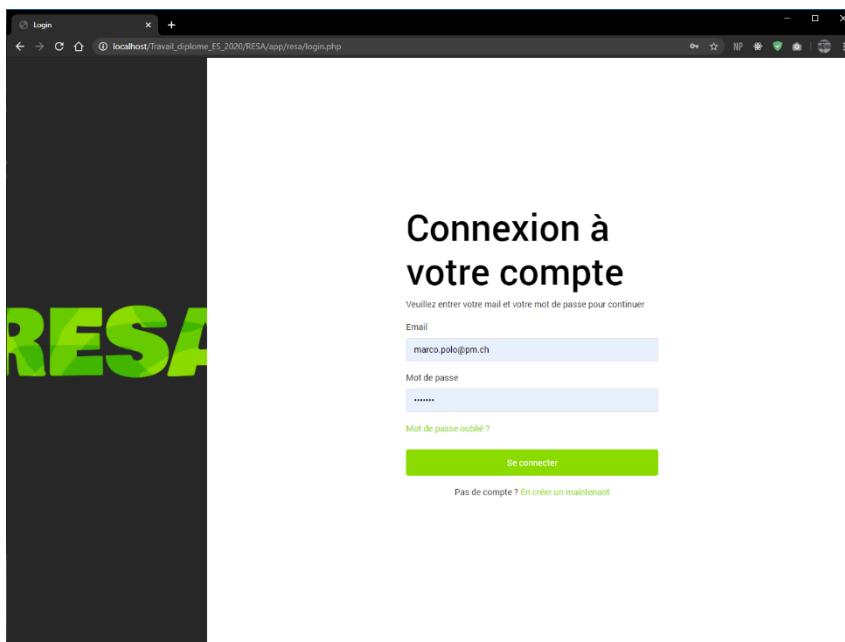
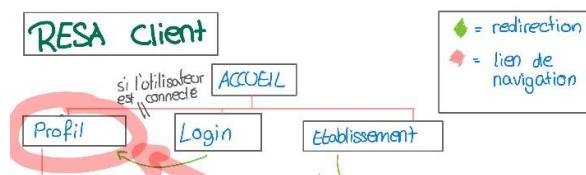


Figure 24 Login d'un utilisateur

L'utilisateur peut créer un nouveau profil en passant par le lien du bas de la page « Création d'un nouveau compte »

Il a la possibilité réinitialiser son mot de passe en passant par le lien « mot de passe oublié ». Il confirme son email dans la boîte affichée.

6.2.2.5 Profil utilisateur



L'utilisateur accède à cette page en sélectionnant « profil » dans la barre de navigation ou après avoir complété son login.

Figure 25 On est là - Page de profil

La page profil guide l'utilisateur connecté pour accéder à ses données personnelles et réservations.

À partir de cette page, l'utilisateur peut accéder à ses réservations, ses paramètres et ses avis laissés sur des réservations passées. Il peut également se déconnecter.

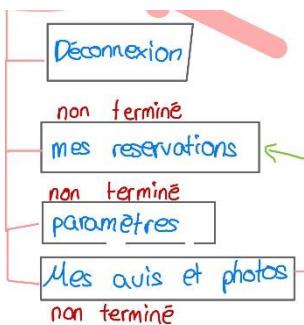


Figure 26 Navigation possibles depuis page de profil

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

1. Réservations
2. Changer la photo de profil

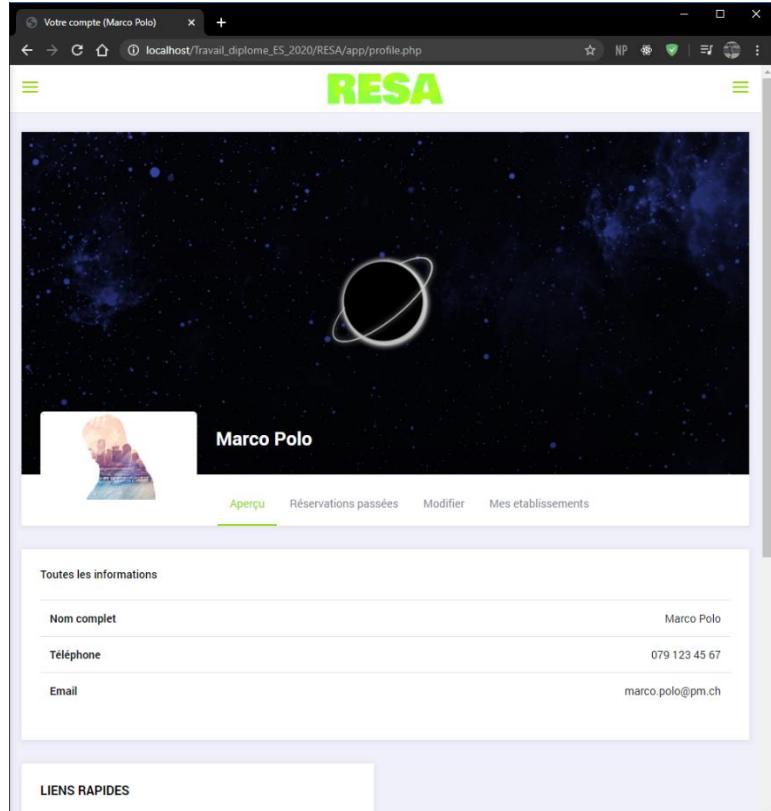


Figure 27 Page de profil

6.2.3 Accès aux applications des managers RESA

L'accès à la page RESA manager est destiné aux manager d'établissement souhaitant accéder aux paramètres.

Le manager peut accéder à tout moment à son espace de management de l'établissement. Pour ce faire, le restaurateur passe par la page de login dédiée :

The screenshot shows a browser window titled "Login" with the URL "localhost/Travail_diplome_ES_2020/RESA/app/manager/". The main content is a "Connexion au restaurant" form. It includes fields for "Nom du Restaurant" (set to "Port Martignot"), "Email" (set to "marco.polo@pm.ch"), "Mot de passe" (redacted), and a "Se connecter" button. Below the form is a link "Pas de compte ? En créer un maintenant".

Figure 28 Login pour un manager de restaurant

Le manager entre son email et son mot de passe (identique que pour RESA Client), mais il doit mentionner le nom de son établissement dans le champ dédié.

Une fois loggé, RESA redirigera le manager sur le menu correspondant à l'abonnement de l'établissement.

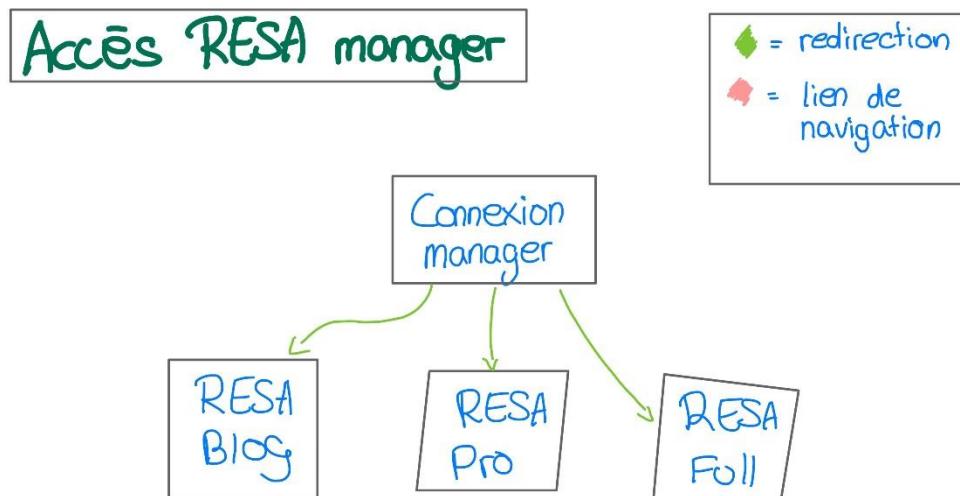


Figure 29 Schéma de navigation RESA manager

6.2.4 RESA Blog

RESA Blog est l'application de management gratuit. Elle permet à un manager de créer son établissement, puis d'y renseigner les données correspondantes. Cette application permet aussi de mettre à jour les photos, les horaires et les menus.

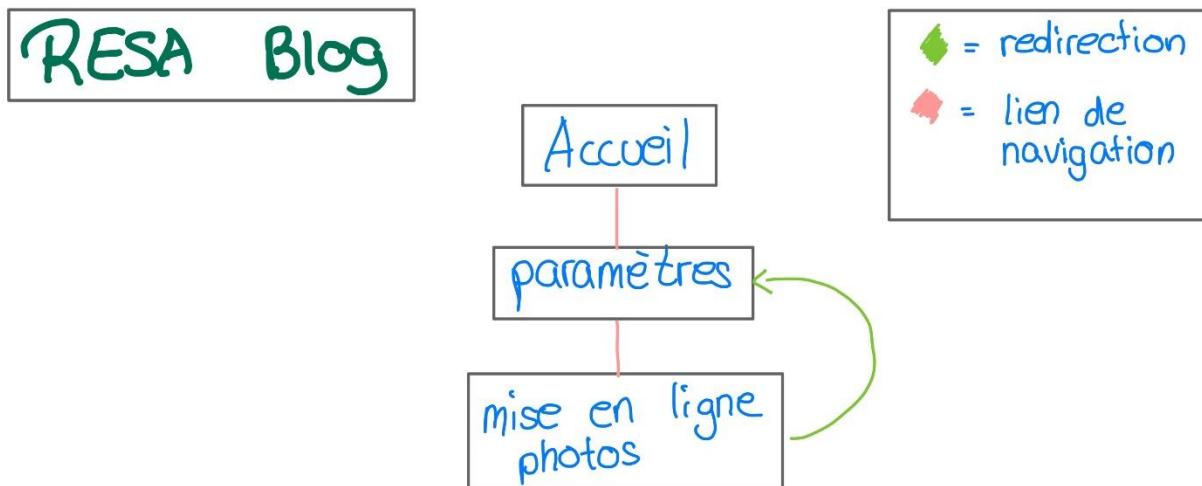


Figure 30 Schéma de navigation RESA Blog

Le schéma de navigation montre les accès limités de l'application blog. Changer les paramètres et mettre en ligne des photos.

6.2.4.1 Cration d'un tablissement

En cliquant sur le bouton « Créer » ci-dessus ou en sélectionnant « Créer établissement » dans le menu de droite de la barre de navigation latérale, l'utilisateur verra un formulaire apparaître.

L'utilisateur doit alors entrer le nom de l'établissement, son adresse complète, le numéro de téléphone de contact ainsi que l'email de contact.

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

[Compléter quand les vues seront prêtes]

6.2.4.2 Accueil

Une fois loggé, le manager est dirigé sur la page d'accueil de son établissement.



Figure 31 On est là - accueil blog

Depuis cette page, le manager peut voir d'un coup d'œil les différentes informations relatives à son établissement.

[Compléter quand les vues seront prêtes]

6.2.4.3 Paramètres

Le manager accède à cette page par la barre de navigation.

Depuis cette page, le manager peut mettre à jour les données de son établissement.

Les champs possibles d'être mis à jour :

- L'adresse de l'établissement
- Le numéro de téléphone
- L'adresse email
- Les menus et les plats
- Le nombre de places du restaurant

[complété quand les vues seront prêtes]

6.2.4.4 Les images

Le manager peut aussi modifier les photos de son restaurant. Il peut les ajouter en passant par la page accessible via le lien dans les paramètres.

La page possède un formulaire qui permet au manager de sélectionner un nombre de photos (maximum 5), puis de les ajouter à son établissement.

[Compléter quand les vues seront prêtes]



Figure 32 On est là - paramètres blog

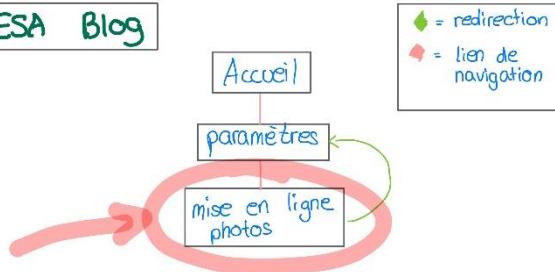


Figure 33 On est là - mise en ligne image blog

6.2.5 RESA Pro

RESA Pro possède les options de RESA Blog et ajoute la gestion des réservations automatisées.

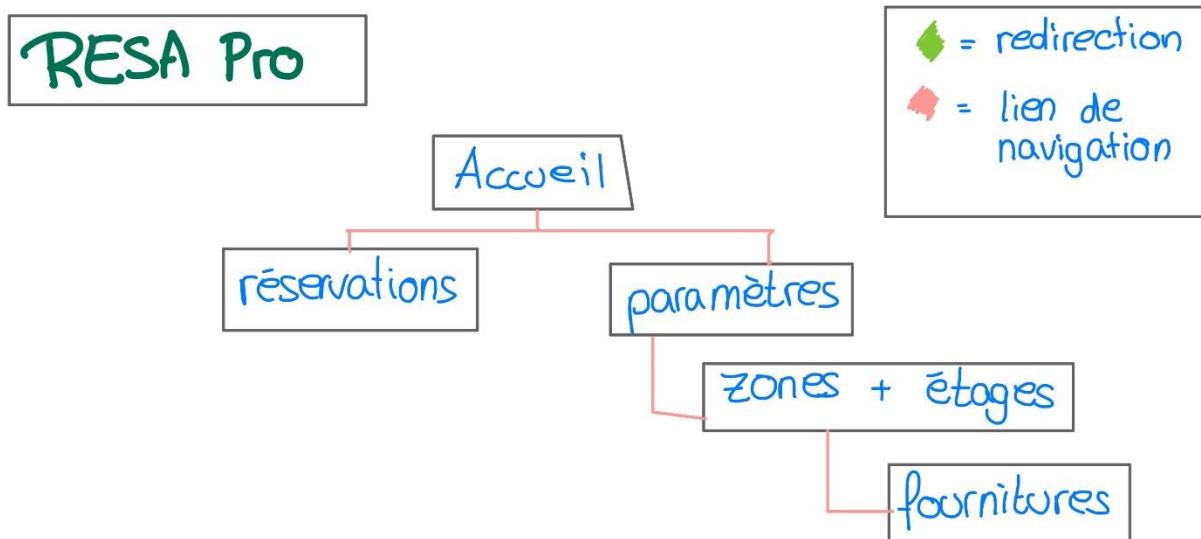


Figure 34 Schéma de navigation RESA Pro

En plus des pages accessibles via RESA Blog, RESA Pro ajoute les pages de gestion des zones, des étages et des fournitures de ces dernières.

6.2.5.1 Accueil

Le manager accède directement à cette page après avoir finalisé son login.

La page d'accueil reprend le même affichage que celui de RESA Blog, mais en ajoutant des widgets pour afficher un aperçu des étages et des zones.

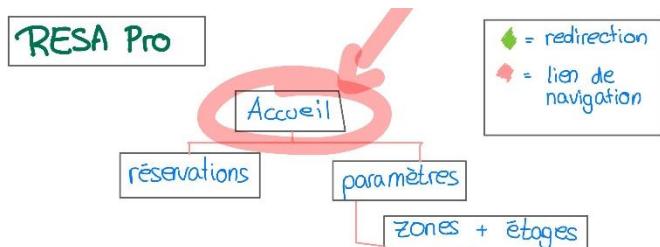


Figure 35 On est là - accueil RESA Pro

[Parler de la page d'accueil quand la vue sera prête]

6.2.5.2 Réservations

La fonctionnalité ajoutée de RESA Pro est celle de la gestion de réservation.



Figure 36 On est là - Réservations RESA Pro

Sur cette page, le manager de l'établissement peut consulter les réservations de la journée, mais aussi de la semaine ou du mois. Il peut également les modifier ou les supprimer. Les réservations sont consultables par tous les serveurs et autres employés de l'établissement. Les serveurs peuvent également ajouter, modifier ou supprimer des réservations suivant les besoins.

[Parler de la page des réservations quand la vue sera prête]

6.2.5.3 Paramètres

Le manager peut accéder à cette page par la barre de navigation.

Cette page permet au manager de modifier les données de son établissement (comme RESA Blog).

Le manager peut gérer les zones, les étages et les fournitures à partir de cette page.

[Compléter les paramètres quand les vues seront prêtes]

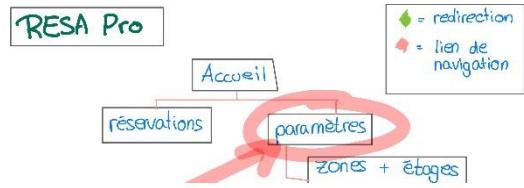


Figure 37 On est là - paramètres RESA Pro

6.2.6 RESA Full

Cette application est la plus complète de RESA. Elle permet, en plus des fonctionnalités de RESA Blog et RESA Pro, de créer un plan de table de son établissement, la gestion des employés, des utilisateurs et des avis laissés.

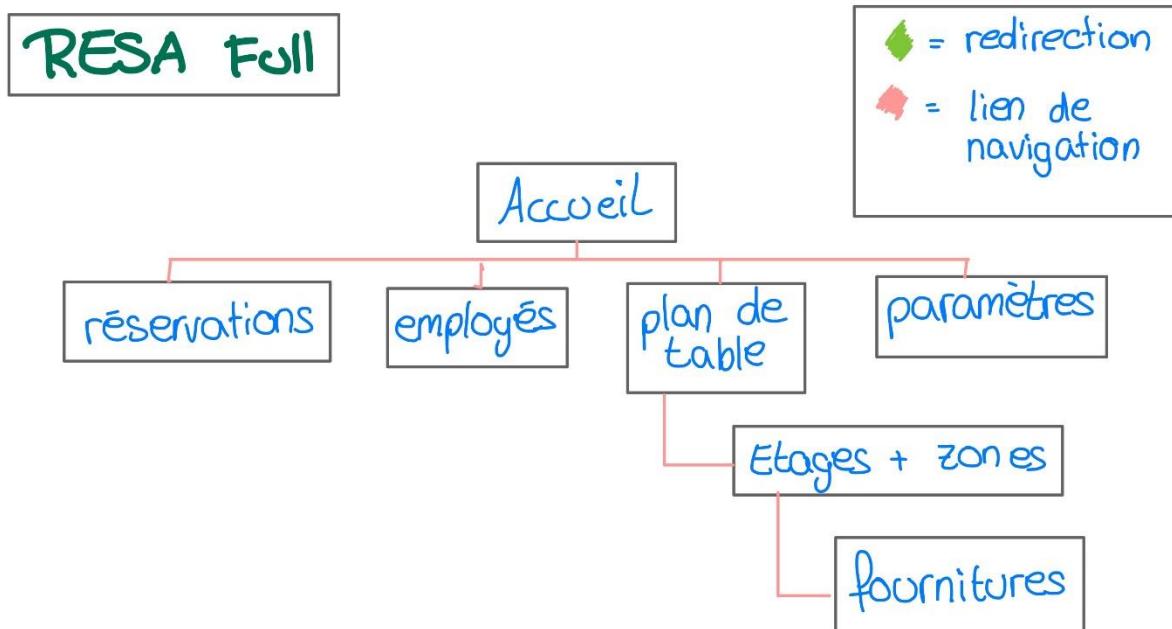


Figure 38 Schéma de navigation RESA Full

6.2.6.1 Paramètres

Le manager accède à cette page par la barre de navigation.

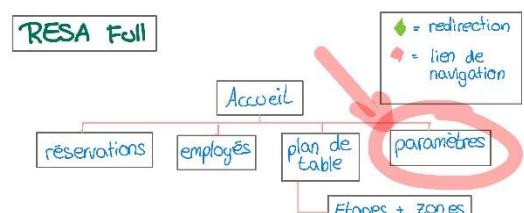


Figure 39 Page de paramètres d'un établissement

6.2.6.2 Réservations

La page des réservations est strictement identique à celle de RESA Pro. [Réservations]

La page est accessible par le manager par la barre de navigation.

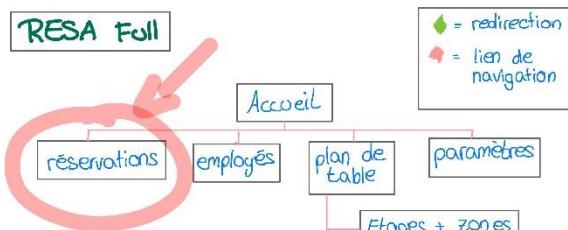


Figure 40 On est là - réservations RESA Full

[parler des réservations quand les vues seront prêtes]

6.2.6.3 Employés



RESA Full propose également au manager de gérer ses employés, leur rang ainsi que leur accès à l'application au sein de l'établissement.

Figure 41 On est là - Employés RESA Full

[Compléter quand les vues seront prêtes]

6.2.6.4 Plan de table

[compléter quand les vues seront prêtes]

6.3 Les droits des utilisateurs

L'application de RESA possède des fonctionnalités. Elles sont accessibles ou non selon les droits des utilisateurs. Les niveaux des droits vont du rang 1 jusqu'au 5.

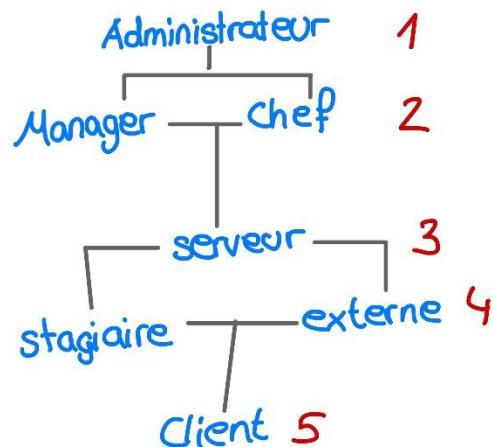


Figure 42 Hiérarchie de RESA

6.3.1 Utilisateurs

Les types d'utilisateurs dans RESA sont nombreux, mais les principaux sont :

1. Administrateur
2. Manager
3. Serveur
4. Client
5. Stagiaire
6. Chef
7. Externe

6.3.1.1 Administrateur

L'administrateur est celui qui a le plus haut rang. Il peut accéder à toutes les données, en ajouter, en modifier ou en supprimer.

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

Son rang (1) étant le plus important, il faut le sécuriser au maximum. Le mot de passe est demandé avant chaque action critique.

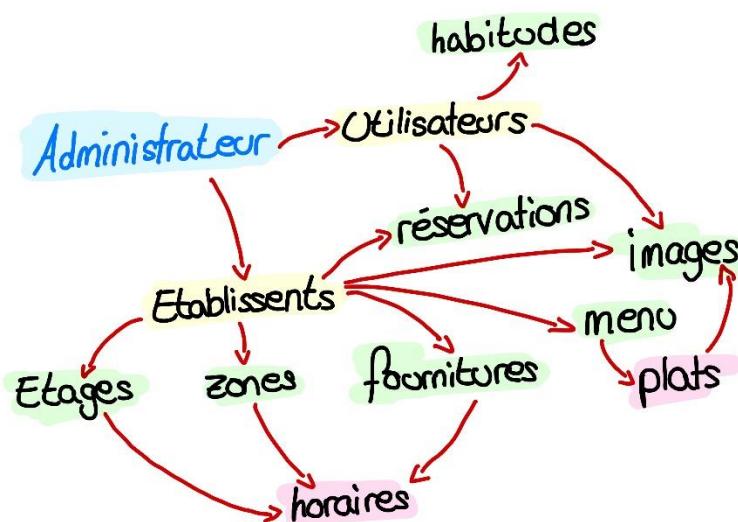


Figure 43 Schéma des droits d'un administrateur

6.3.1.2 Manager (et chef)

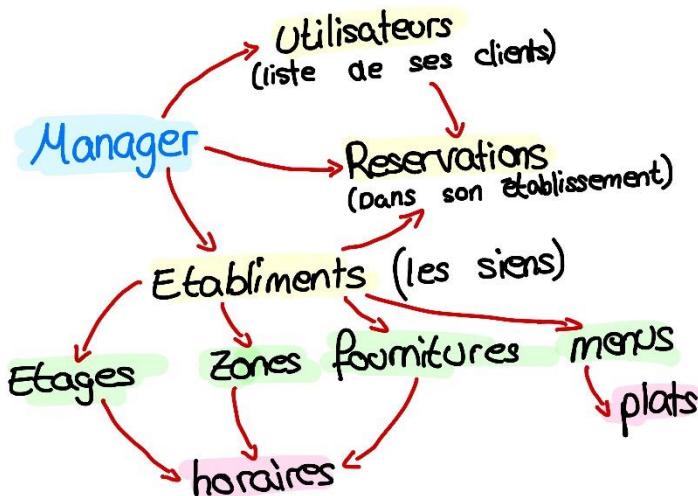


Figure 44 Schéma des droits d'un manager

Le manager (2) est un utilisateur qui a créé un établissement, c'est-à-dire que cet utilisateur est un simple client pour tous les établissements, sauf pour les siens, où il a le contrôle total. Il peut ainsi gérer son personnel et leurs accès, ses étages, zones et fournitures ainsi que les horaires de ces derniers. Il peut mettre à jour les données de son établissement comme son menu, ses plats ou ses photos.

Le chef de cuisine peut réaliser les mêmes actions que le manager à l'exception de la gestion des employés et de la suppression de l'établissement.

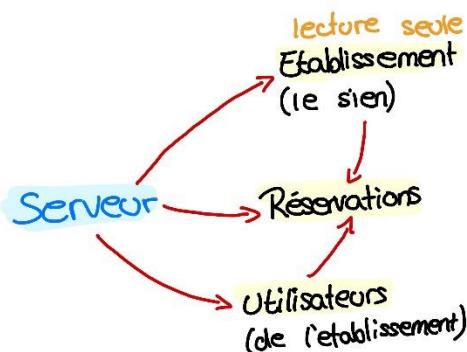


Figure 45 Schéma des droits d'un serveur

6.3.1.4 Stagiaire et externe

Les stagiaires et les externes (rang 4), sont limités dans leurs actions au sein des différentes applications de RESA. Les stagiaires et les externes peuvent consulter les réservations et le plan de table, mais ne peuvent apporter des changements qu'avec le code d'authentification d'un utilisateur avec le rang 4 ou supérieur.

6.3.1.5 Client

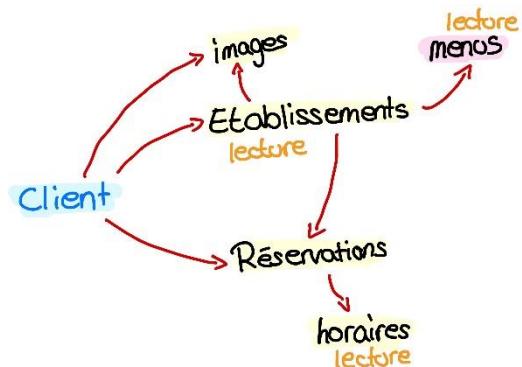


Figure 46 Schéma des droits d'un client

Le client (rang 5) ne possède aucun droit particulier dans l'application. Il peut cependant accéder à ses réservations pour les consulter, les modifier ou les supprimer. Il voit la liste de tous les établissements de RESA.

Il a accès aux données des établissements pour consulter les photos, les menus et les avis laissés par les autres clients.

Il a 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

7.1.1 GitHub

Afin d'avoir un suivi constant de mon projet, j'ai décidé de créer un GitHub. Dans ce github j'ai 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 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 », contient 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 suis en permanence (ex. le journal de bord).

7.2 Prétravail

7.2.1 Programmation

Afin de pouvoir réaliser au mieux mon travail, j'ai dû rechercher les langages de programmation et des librairies qui m'aideront le mieux possible à réaliser le travail qui m'est demandé pour réaliser mon travail de diplôme.

La bibliothèque JavaScript qui m'a semblé la 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** »

7.2.2 Installation de React

Tout d'abord je dois disposer d'une mise à jour récente de Node.js. Pour créer une application de test, je dois entrer la commande suivante dans le dossier de destination :

```
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 créée, 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

Pour faciliter le développement et la gestion des fichiers de mon API et de l'application, j'ai décidé de mettre le même en-tête sur les fichiers créés:

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

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

Cet en-tête me permet de repérer les fichiers que j'ai créés et développés, ainsi que voir leurs versions dans un futur où 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 d'identifier son but ainsi que les paramètres à envoyer à celle-ci.

```
/*
 * Lie une image à un établissement
 * Params:
 *   - idEtablissement : 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 spécifiques. Afin de respecter une norme, j'ai décidé de me fier au site de Sourcemaking⁴. Ce site reprend chaque évènement, action ou lien en expliquant clairement leurs fonctionnements.

⁴ <https://sourcemarking.com/uml/modeling-business-systems/external-view/activity-diagrams>

7.2.4 Organisationnel

Pour mieux comprendre les besoins du client, nous avons décidé avec M. Garcia d'aller sur les lieux afin de discuter avec la manager. Lors de cette discussion nous avons donc mis au clair les points qui jusqu'à là, étaient encore flous.

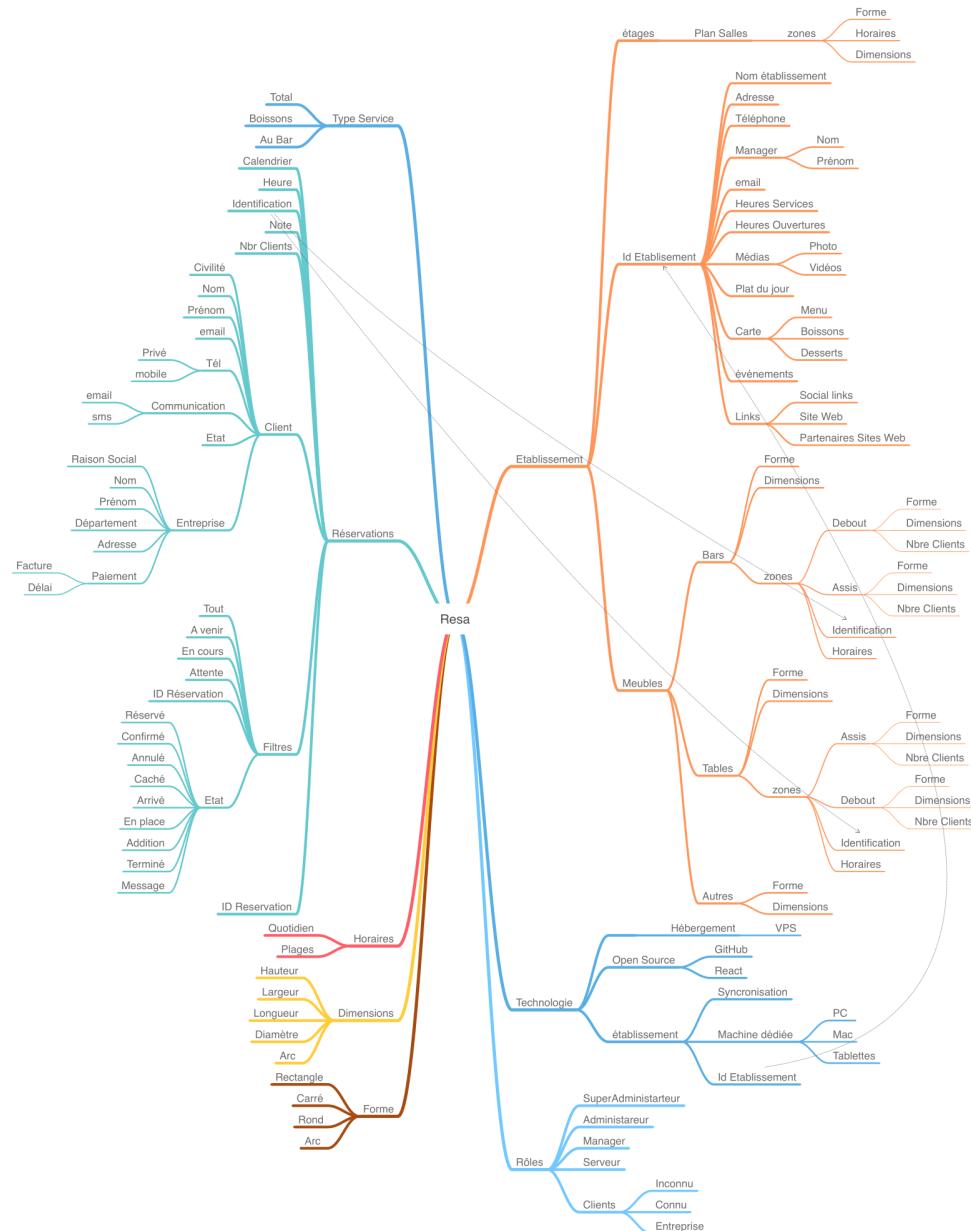


Figure 47 Mindmap de préparation

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 d'avoir une base de données MySQL.

J'ai décidé d'utiliser Laragon, car c'est lui que j'ai utilisé le plus fréquemment.

7.3.2 Visual Studio Code

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

7.3.3 EDUGE

Je fais un backup de mon projet tout à chaque changement majeur sur mon drive EDUGE. J'ai choisi EDUGE, car cette plateforme est stable et fonctionnelle.

7.3.4 Github Desktop

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

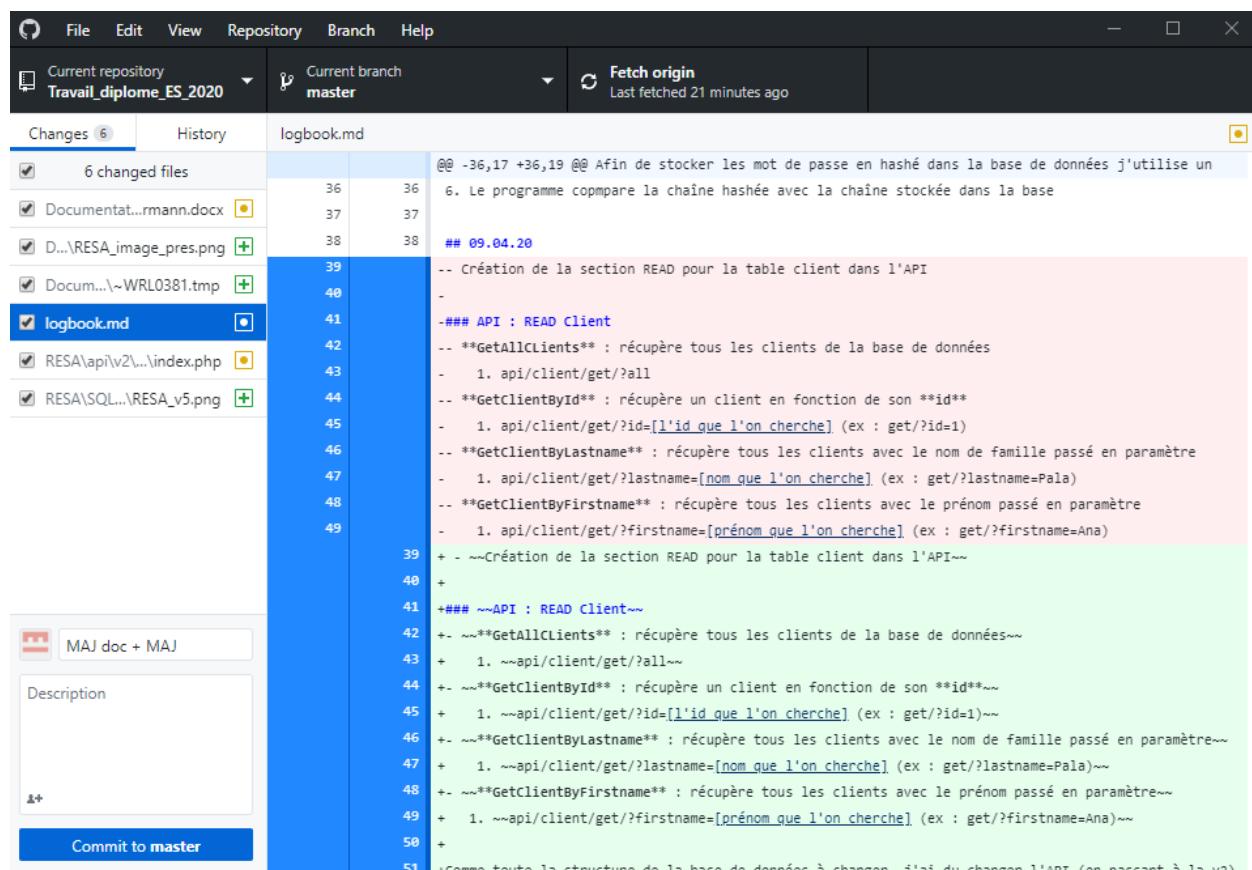


Figure 48 : Interface de Github Desktop

7.4 Schémas de fonctionnements

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

7.4.1 RESA

RESA est un groupe d'applications web qui communiquent avec un service REST afin d'accéder à une base de données.

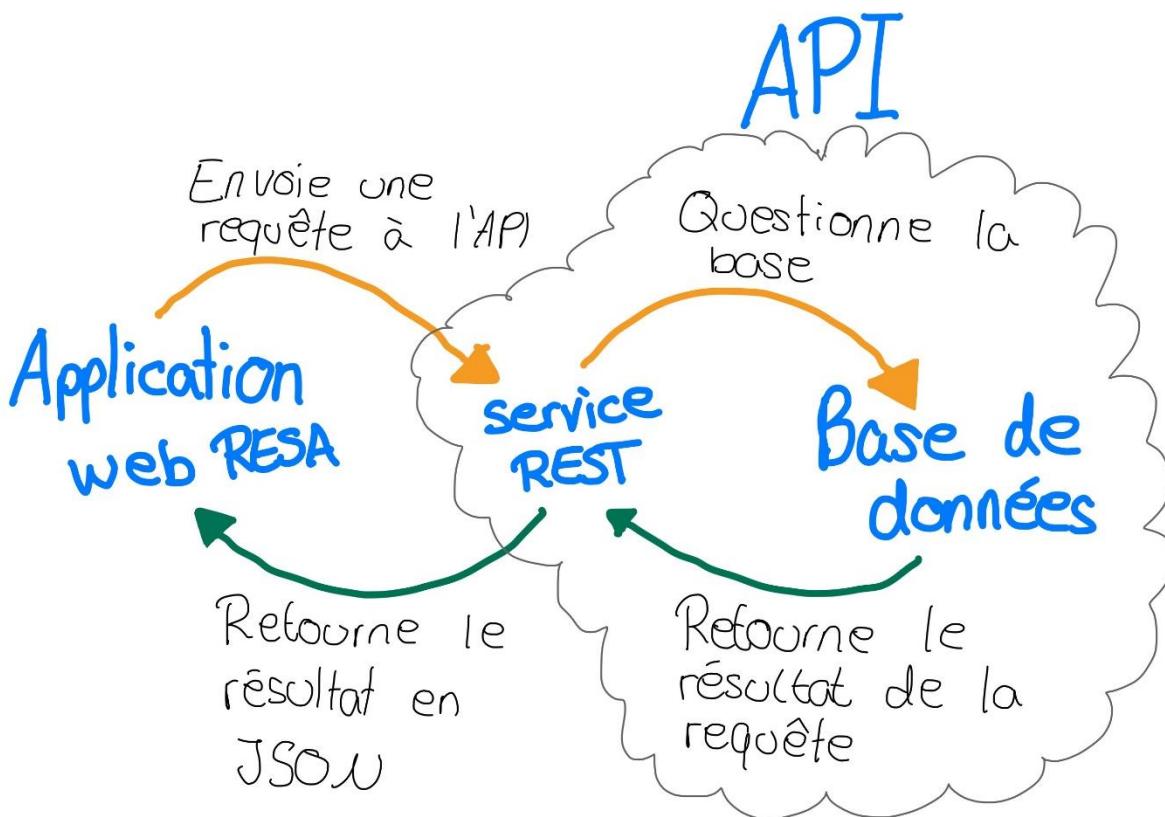


Figure 49 diagramme de fonctionnement RESA

7.5 Diagrammes de fonctionnement de RESA

Les diagrammes de fonctionnement permettent de représenter les actions et le fonctionnement de RESA dans le fond.

Il est possible de visualiser la totalité des diagrammes de fonctionnement dans les annexes [[Diagrammes de fonctionnement](#)]

7.6 Diagrammes d'activités

Pour la création des diagrammes d'activités, j'ai d'abord utilisé le site internet « dbdiagram.io », mais lorsque M. 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.io⁵.

Afin d'être sûr de respecter les normes de diagrammes d'activités, je me suis fié au site nommé Sourcemarking⁶. Ce site explique tous les objets, lignes et intersections d'un diagramme.

7.6.1 Diagrammes d'activités de RESA

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

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

7.7 Base de données

Afin de pouvoir stocker les données, j'ai créé une BDD⁷ nommée « resa ». Cette BDD me permet d'enregistrer toutes les données qui sont nécessaires au bon fonctionnement des applications web.

7.7.1 UML

Pour créer le model UML de la BDD, je suis passé par le site dbdiagram.io⁸. Ce site permet de créer des modèles UML qui sont par la suite exportable en fichier SQL afin de les ajouter dans notre BDD.

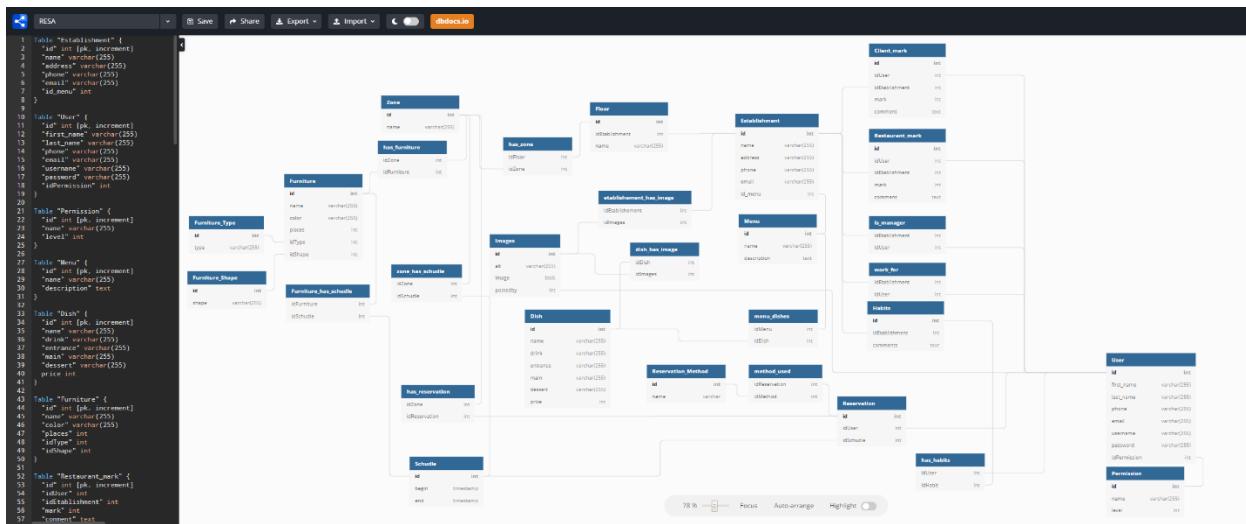


Figure 50 Aperçu interface dbdiagram.io

7.7.2 Privilèges

Pour accéder à la BDD, il faut utiliser les privilèges suivants :

- Username : resa_tech_es
- Password : WhutMerYmZeR6EHb

⁵ Lien complet de l'application : <https://app.diagrams.net>

⁶ [Https://sourcemaking.com/uml/modeling-business-systems/external-view/activity-diagrams](https://sourcemaking.com/uml/modeling-business-systems/external-view/activity-diagrams)

⁷ Base de données

⁸ [Https://dbdiagram.io](https://dbdiagram.io)

7.7.3 Structure

Ma BDD est structurée de la manière suivante :

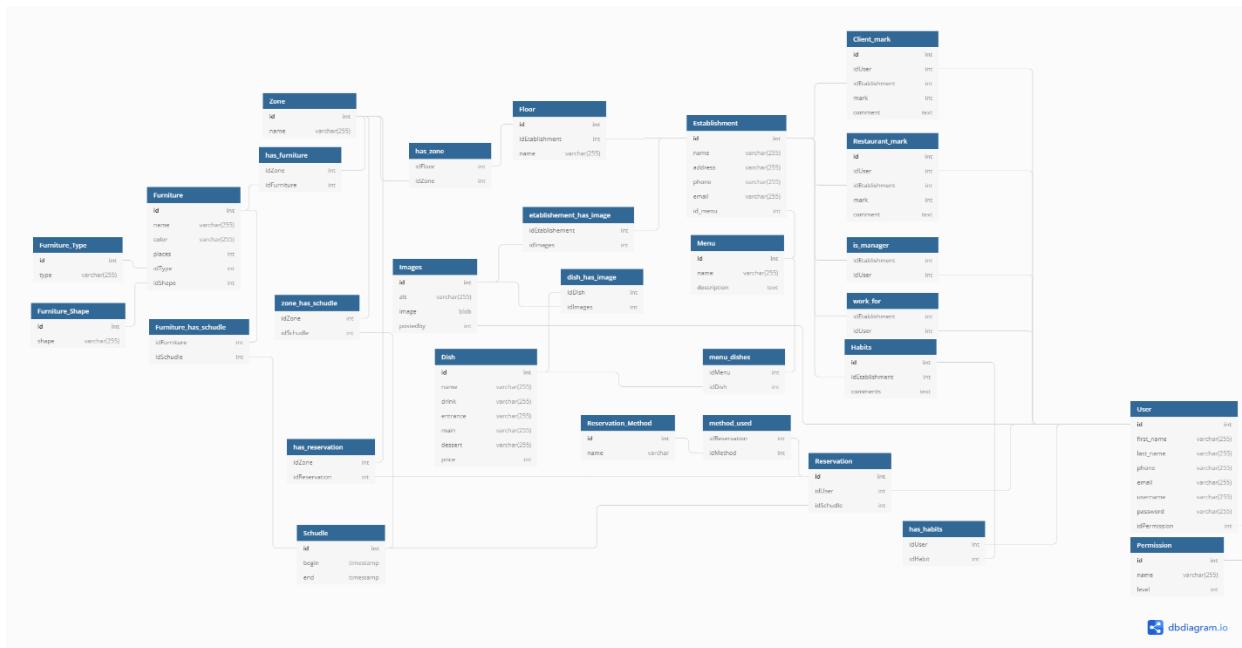


Figure 51 Aperçu MCD

7.7.4 Données de tests

7.7.4.1 Utilisateurs

Pour tester mon api, j'ai créé des utilisateurs, des établissements 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 de l'établissement:

- Username : 3383
- Password : manager

Employé 1 :

- Username : 5243
- Password : e1

Employé 2 :

- Username : 9902
- Password : e2

7.7.4.2 Établissements

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

7.7.4.3 Les niveaux d'abonnement des restaurants

1. BLOG (gratuit)
2. PRO
3. FULL

7.8 API

Avec l'objectif d'accéder à ma BDD à distance ou depuis différents supports, j'ai dû mettre en place une API afin de communiquer avec elle.

7.8.1 Structure

L'API v2 est structurée de telle manière à ce que les informations soient simplement consultables.

La structure et le Cheat Sheet complet de l'API sont dans les annexes [[Cheat Sheet de l'API](#)].

7.8.2 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 passe

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

7.8.3 Communications avec l'API

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

7.8.3.1 Envoi

Afin d'envoyer 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.

7.8.3.1.1 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 vérifiés et hachés avant)

```
$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 convertit le JSON reçu en objet PHP
$data = json_decode($json);
```

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

7.8.3.2 Réceptions

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

Voici ce que retourne l'API lors de l'envoi 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 chaîne 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 créé à partir de la valeur JSON

Toutes les valeurs retournées par l'API sont en JSON, mais ce ne sont pas tout le temps des retours de requêtes SQL sur la BDD.

Il arrive, que l'API mette les valeurs en forme avant de les envoyer en JSON :

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

// On parcourt 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 crée un enregistrement dans le tableau avec comme clé l'id de l'étage
        // et comme valeurs le nom de l'étage 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']  
, $value['zone_id'], $value['begin'], $value['end']));
}

return $floors;
```

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

7.8.4 Gestion des images

Pour rendre une application plus attrayante visuellement, il ne faut pas négliger les images. Pour gérer les images que ce soit pour les mettre en ligne ou pour les récupérer, j'ai décidé de créer une gestion des images par mon API.

7.8.4.1 Mise en ligne d'une image

La mise en ligne est un peu spéciale, j'ai créé un fichier PHP qu'il faut inclure dans le fichier qui souhaite enregistrer l'image. Je suis passé par cette option, car il s'agissait de la plus optimale et que je ne souhaitais pas perdre de temps alors que cette option fonctionne bien.

7.8.4.1.1 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."establishment/create/form/"; ?&gt;"<br/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 paramètre `enctype`, il faut bien mettre "`multipart/form-data`", car ceci 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` vérifie que toutes les données envoyées dans la variable POST soient bien conformes aux attentes, c'est-à-dire que le nom soit une chaîne de caractère, que l'email soit un email, etc.

Ensuite, la fonction `SendData` envoie les données de l'établissement dans la BDD à l'aide de requêtes GET de l'API. Pour ce faire, j'utilise le querybuilder de PHP pour préparer ma requête.

```
$queryData = array(  
    'name' => $data['name'],  
    'address' => $data['adress'],  
    'phone' => $data['phone'],  
    'email' => $data['email'],  
    'creatorID' => $_SESSION['user']->id  
);  
  
$link1 = $path."establishment/create/?".http_build_query($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 l'image mise en cache par le navigateur temporairement avant d'être enregistré dans le répertoire des images.

```
SaveImageEstablishment($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

7.8.4.2 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/`

À partir de là, il faut ajouter les paramètres de/des (l')image(s) recherché(es).

7.8.4.2.1 Exemples

Voici la liste de des paramètres possibles pour 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	<code>/api/v2/images/get/?data&id=XX</code>	Un tableau avec les informations de l'image
establishment	<code>/api/v2/images/get/?establishment&id=XX</code>	Un tableau avec l'id des images ainsi que leur lien complet
dish	<code>/api/v2/images/get/?dish&id=XX</code>	Un tableau avec l'id des images ainsi que leur lien complet
user	<code>/api/v2/images/get/?user&id=XX</code>	L'id de l'image ainsi que son chemin complet
id	<code>/api/v2/images/get/?id=XX</code>	Redirige directement sur l'image

Le dernier paramètre (id) doit être utilisé lorsque l'on souhaite afficher l'image dans le path d'une balise img.

```
" alt="">
```

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

7.8.5 Réservations et horaires

7.8.5.1 Le nombre de places disponibles grâce aux routines

Avant de faire une réservation, le client doit d'abord être guidé sur les horaires qui sont affichés comme étant disponibles où ayant de la place. Pour ce faire, j'ai créé une routine dans mon serveur SQL.

7.8.5.1.1 Les routines MySQL

Il est possible, dans MySQL, de créer des routines (plus communément nommées « des fonctions »). Ces routines nous permettent d'appeler des requêtes complexes très simplement.

La création d'une routine est bien documentée. Il suffit de donner un nom à notre routine, puis d'y mettre nos requêtes, paramètres, etc.

```
DELIMITER //
CREATE PROCEDURE GetAllProducts()
BEGIN
    SELECT * FROM products;
END
// DELIMITER ;
```

Extrait de code n°15. Création d'une routine dans MySQL

Dans cet exemple, on aperçoit que la routine se nomme « GetAllProducts » et que celle-ci exécute un Select de tous les champs sur la table « products ». Pour appeler cette routine, il faut créer une requête SQL comme ceci :

```
CALL GetAllProducts();
```

Extrait de code n°16. Comment appeler une routine en MySQL

7.8.5.1.2 Création de ma routine

Pour mieux comprendre l'utilité d'une routine pour ma requête, il suffit de comprendre les limites du PDO de PHP. En effet ce dernier ne permet pas d'exécuter plusieurs requêtes pour recevoir un résultat. Ce qui fait que je ne pouvais pas déclarer mes variables pour ensuite les utiliser dans ma requête finale.

Les variables utilisées plusieurs fois dans la requête finale sont :

```
idEtab // L'id de l'établissement recherché [int]
dateday // La date du jour recherché [date]
arrival // L'heure recherchée [time]
duration // La durée estimée en secondes du temps du repas [int]
```

Extrait de code n°17. Les variables utilisées dans ma routine SQL

Il faut également signaler à la routine quelle sera la variable de sortie. Il faut déclarer cette variable comme paramètre :

```
Available // Le nombre de places disponibles [int]
```

Extrait de code n°18. La variable de sortie de ma routine SQL

J'ai donc créé une procédure (routine) nommée **IsPlaceForReservation** qui prend comme paramètres tous les champs cités ci-dessus dans l'extrait de code.

Ensuite, dans la définition de la procédure, j'ai mis ma requête en intégralité avec les variables dans les bons endroits.

```
BEGIN
    SELECT SUM(src.places) - IFNULL((
        SELECT SUM(f.places) not_avaible
        FROM reservation as r
        INNER JOIN furniture as f ON r.idFurniture = f.id
        WHERE r.idEstablishement = idEtab
        AND CAST(r.arrival as DATE) = dateday
        AND CAST(r.arrival as TIME) <= arrival
        AND CAST(FROM_UNIXTIME(UNIX_TIMESTAMP(r.arrival)+r.duration) as TIME) >=
arrival
    ),0) - IFNULL((
        SELECT SUM(f.places) not_avaible
        FROM reservation as r
        INNER JOIN furniture as f ON r.idFurniture = f.id
        WHERE r.idEstablishement = idEtab
        AND CAST(r.arrival as DATE) = dateday
        AND CAST(r.arrival as TIME) >= arrival
        AND CAST(FROM_UNIXTIME(UNIX_TIMESTAMP(arrival)+duration) as TIME) >=
r.arrival
    ),0) avaible
    FROM (
        SELECT f.places
        FROM `zone_has_schudle` as zhs
        INNER JOIN schudle as s ON s.id = zhs.idSchudle
        INNER JOIN zone as z ON z.id = zhs.idZone
```

```

    LEFT JOIN has_furniture as hf ON hf.idZone = z.id
    LEFT JOIN furniture as f ON f.id = hf.idFurniture
    WHERE CAST(s.begin as time) <= arrival
    AND CAST(s.end as time) >=
    CAST(FROM_UNIXTIME(UNIX_TIMESTAMP(arrival)+duration) as TIME)
    AND hf.idZone IS NOT NULL AND f.idEtablissement = idEtab
) src;

END

```

Extrait de code n°19. Ma procédure pour vérifier s'il y a de la place

À présent, si je souhaite savoir s'il y a une place de disponible (par exemple : le 14.05.2020 à 12 :30 dans l'établissement portant l'id 1), j'exécute la requête suivant dans SQL :

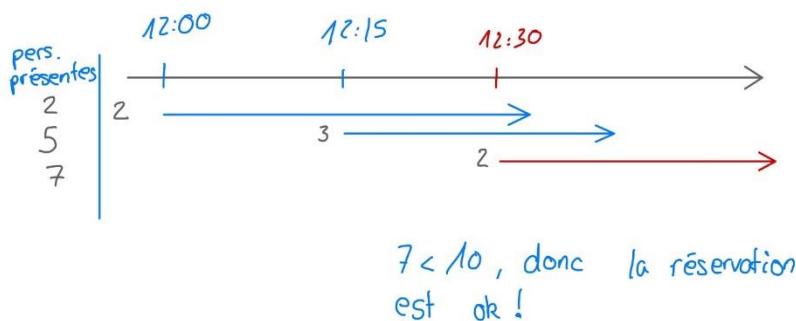
```
CALL IsPlaceForReservation(1, '2020-05-14', '12:30', 3600, @available);
```

Extrait de code n°20. Appel de ma routine SQL avec paramètres

7.8.5.2 Système de réservations

Les réservations sont les enregistrements les plus importants de RESA. Il faut prendre en compte une multitude de différents statuts avant de valider et d'enregistrer une réservation faite par un client ou un établissement.

pers. max : 10



Avant de réserver, je vérifiais que la place pour le nombre de personnes demandé était encore disponible au moment de l'arrivée du client. C'est-à-dire que je devais d'abord vérifier le nombre de places disponibles à l'heure donnée dans l'établissement, mais également le nombre de places prises par les réservations arrivées avant et où leur heure de départ estimée était après l'heure d'arrivée de la nouvelle réservation.

Figure 52 Schéma questionnement fonctionnement réservations

Ce système fonctionnait bien jusqu'à ce que je découvre un nouveau problème : Il ne faut pas seulement vérifier qu'il y a de la place dans le restaurant au moment de l'arrivée, mais également de vérifier qu'il reste bien de la place pour les réservations qui viennent après.

MAX: 15 p.

Verifier disponibilités
avant mais aussi
après

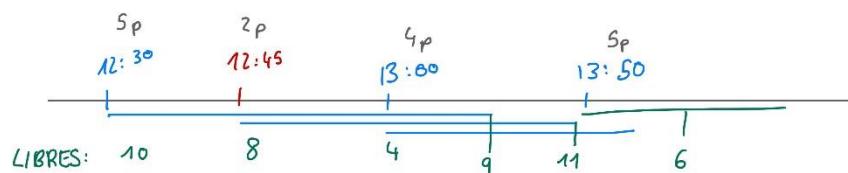
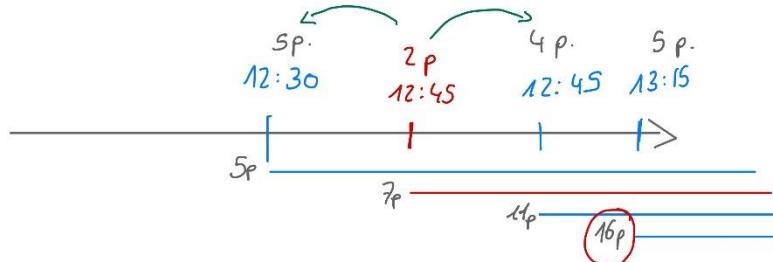


Figure 53 Schéma vérification de places

Après avoir raisonnable, je suis parti sur cette solution :

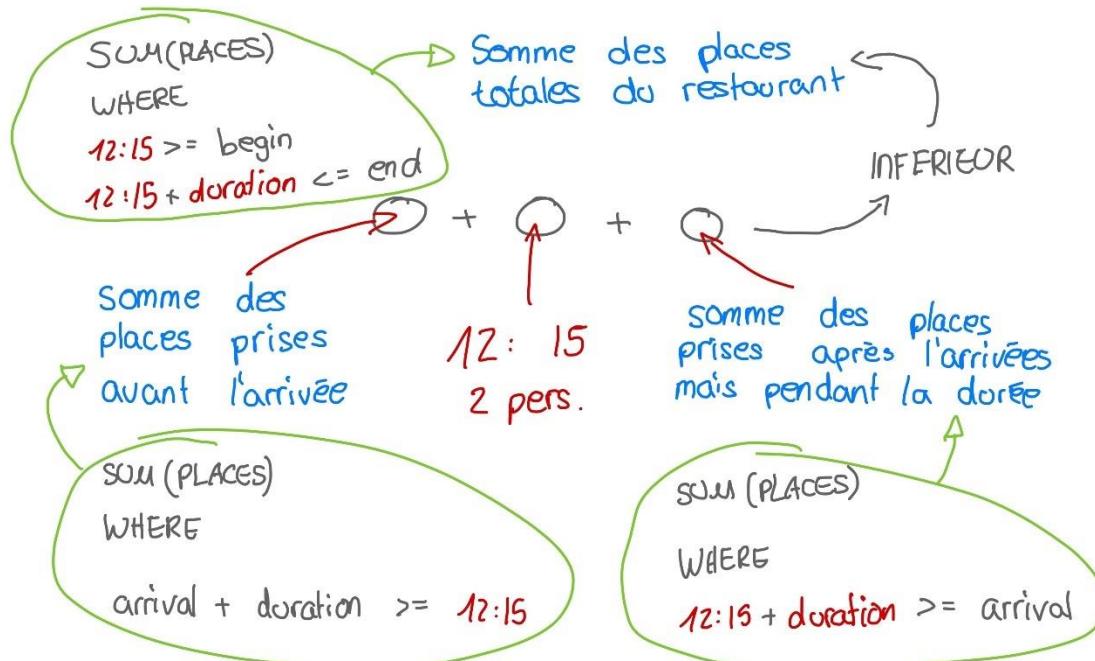


Figure 54 Solution fonctionnement de réservations

Ma solution consiste à vérifier le nombre de places totales disponibles à l'instant [t], puis de soustraire le nombre de places estimées comme occupées encore après l'heure d'arrivée du nouveau client. On y soustrait également le nombre de places occupées par les réservations qui arrivent avant l'heure estimée du départ de ce même nouveau client. Tout ce calcul nous donne comme résultat le nombre de places disponibles pour son créneau. Si le nombre de places disponibles est inférieur au nombre de personnes dans la réservation, il n'est pas possible de réserver pour l'instant [t].

7.8.6 Les 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.

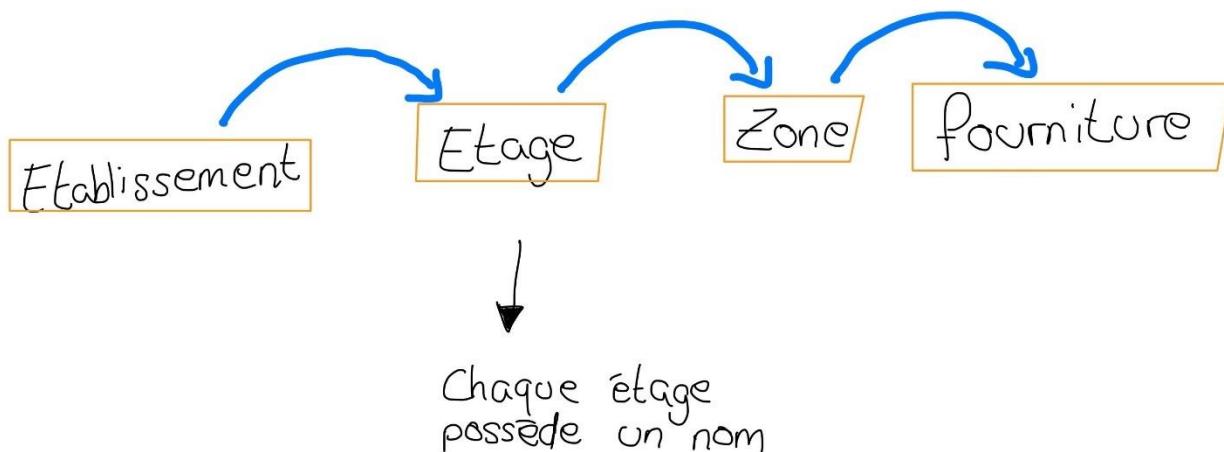


Figure 55 Schéma de lien entre les tables principales

7.8.6.1 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.

Construction d'une zone

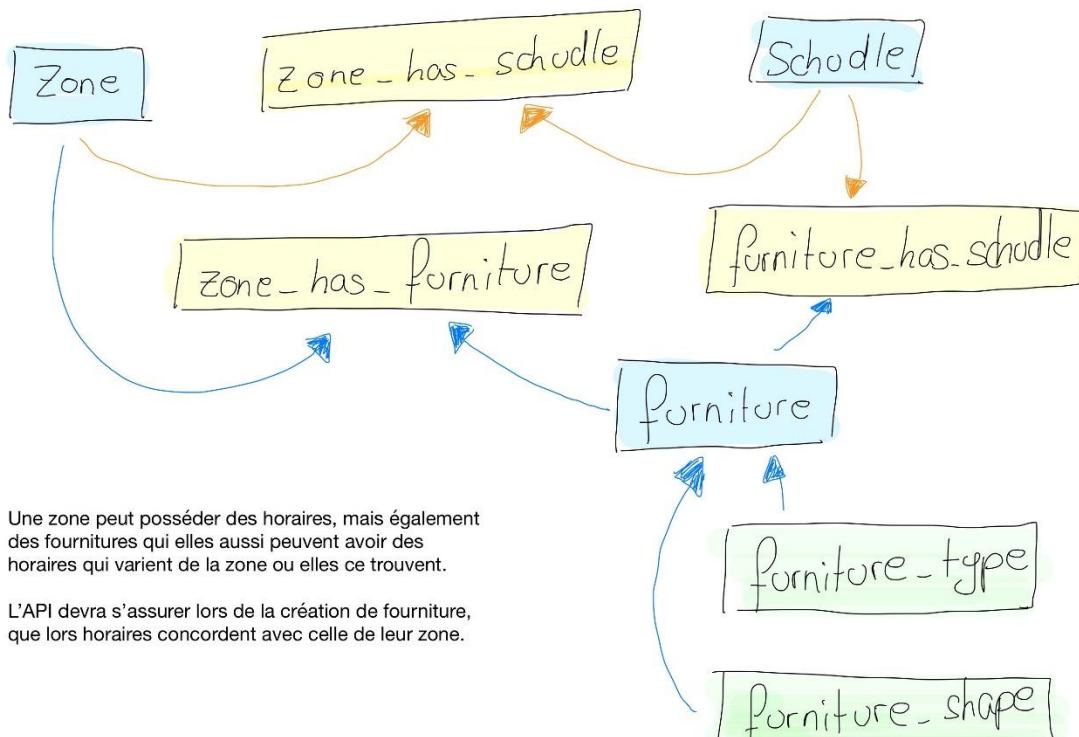


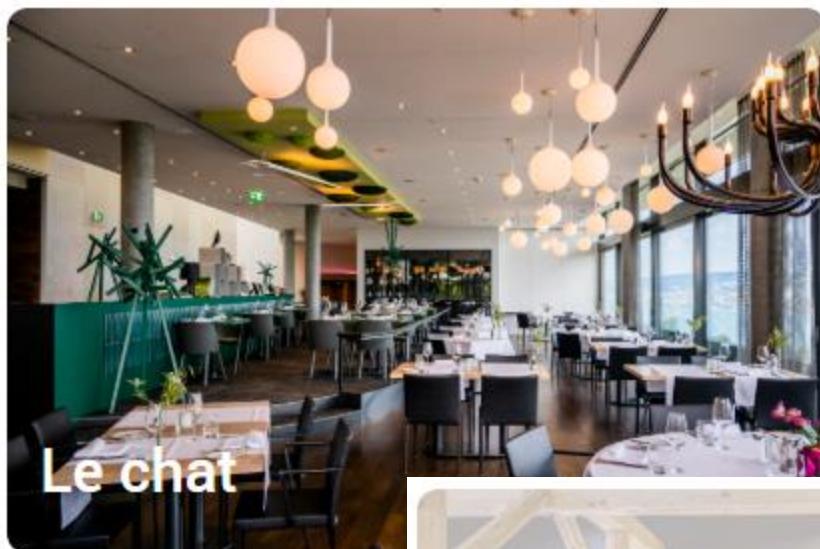
Figure 56 Schéma des liens entre les tables pour une zone

7.8.6.2 Le fonctionnement des zones [parler du fonctionnement particulier des zones]

7.9 L'application

7.9.1 Affichage du statut ouvert / fermé des restaurants

Sur la page principale de RESA (là où sont affichés tous les restaurants), il y a deux types d'affichages possibles pour les restaurants :



Lorsque le restaurant est ouvert, son image s'affiche clairement et son nom apparaît dans le coin inférieur gauche



Lorsque le restaurant est fermé, l'image est grisée et il y a un texte rouge qui affiche le message « fermé » en majuscule.

Afin de pouvoir afficher ses données, je stocke dans l'API les horaires du jour de la semaine du restaurant et je le compare à l'heure et au jour actuel (au moment où l'on charge la page).

```
(SELECT s.id FROM opening as o INNER JOIN schudle as s ON s.id = o.idSchudle WHERE
E o.idDay = '.$idDay.' AND o.idEstablishement = e.id AND UNIX_TIMESTAMP(s.begin) <
UNIX_TIMESTAMP(NOW()) AND UNIX_TIMESTAMP(s.end) > UNIX_TIMESTAMP(NOW())) as open
```

Extrait de code n°21. Requête SQL horaire magasin

Cette requête renvoie l'id de l'horaire s'il existe, sinon elle renvoie null. Je vérifie alors dans la page si le résultat est null ou pas et s'il ne l'est pas, je considère que l'établissement comme ouvert.

Cette requête, je l'ai intégrée dans la requête SELECT de tous les établissements de ma base.

```
SELECT e.id, e.name, e.address, e.phone, e.email, m.name as menu_name, m.description as menu_descripitpion, (SELECT s.id FROM opening as o INNER JOIN schudle as s ON s.id = o.idSchudle WHERE o.idDay = '.$idDay.' AND o.idEtablisshement = e.id AND UNIX_TIMESTAMP(s.begin) < UNIX_TIMESTAMP(NOW()) AND UNIX_TIMESTAMP(s.end) > UNIX_TIMESTAMP(NOW())) as open FROM `establishment` as e LEFT JOIN menu as m ON e.id = m.id
```

Extrait de code n°22. Requête SQL qui récupère tous les restaurants

Le résultat fait en sorte de retourner tous les établissements avec un champ nommé « open » qui est soit un int (quand il est ouvert) soit null (quand il est fermé).

7.9.2 Login vs Fast-login vs Manager-login

7.9.2.1 Login

RESA possède trois pages de login différentes. Le premier « login » est la page basique qui permet à un utilisateur de se connecter et d'être automatiquement redirigé sur sa page de profil.

Une fois que l'utilisateur à entrer son email et son mot de passe, le mot de passe est chiffré en sha256 puis est envoyé avec le username à l'API.

```
$password = hash('sha256', $_POST['password']);
$username = $_POST['username'];

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

$link = $link."?login&".http_build_query($queryData);
// Takes raw data from the request
$json = file_get_contents($link);
// Converts it into a php object
$data = json_decode($json);
```

Extrait de code n°23. Login d'un utilisateur

Si le login est validé, l'utilisateur est retourné et stocké dans la variable de SESSION. Dans le cas contraire, un message d'erreur s'affiche.

7.9.2.2 Fast-login

Fast-login permet également à un utilisateur de se connecter, mais cette fois-ci, si le la connexion est validée, l'utilisateur est redirigé vers une page mise en paramètre. Dans le cas de RESA, cette page est utilisée quand un utilisateur regarde les restaurants sans être connecté, puis décide de réserver une table. RESA va alors lui afficher la page de connexion et si c'est validé, l'utilisateur est alors redirigé vers la page de réservation à la date choisie avant de se connecter.

Afin de correctement pouvoir utiliser ma page de fast-login, je dois l'appeler de la manière suivante :

Il faut impérativement que la variable `$_SESSION['returnlink']` soit définie avec un chemin valable. Puis, on redirige l'utilisateur sur la page :

```
header("Location: fast_login.php");
exit();
```

Extrait de code n°24. Redirection sur la page « fast_login.php »

Une fois sur la page de fast-login, le chemin de redirection est stocké, puis on fait un unset de la session.

```
$location = $_SESSION['returnlink'];
unset($_SESSION['returnlink']);
```

Extrait de code n°25. Unset d'une variable de la session

Lorsque l'utilisateur se connecte avec succès, il est redirigé sur le lien reçu

```
header("Location: $location");
exit();
```

Extrait de code n°26. Redirection header sur avec une variable

7.9.2.3 Manager-login

Puis, manager-login est la page de login qui permet à un manager de s'identifier dans son restaurant afin de pouvoir accéder aux réglages de ce dernier.

En plus de gérer le login comme login et fast-login, le manager-login vérifie que la personne qui se connecte est bien le manager du restaurant dont il a mentionné le nom.

7.9.3 Le calendrier de réservation

Lorsqu'un restaurant possède un abonnement Pro ou Full, il est possible depuis sa page de réserver directement sur le site de RESA grâce au calendrier interactif proposer.

Ce calendrier provient du site startutorial.com⁹. Il suffit de copier la classe proposée et de l'include dans la page où l'on souhaite l'utiliser.

J'ai dû changer quelques petites choses de la classe, notamment le constructeur, j'avais besoin d'envoyer en paramètre l'id de l'établissement dans lequel le client souhaite réserver. J'ai également dû changer les cases des jours pour les transformer en liens. Le lien redirige sur la page de réservation avec comme paramètre GET : l'id de l'établissement, l'année, le mois, le jour.

```
<?php
include './assets/php/calendar.php';
$calendar = new Calendar($_GET['id']);
echo $calendar->show();
?>
```

Extrait de code n°27. Affichage du calendrier

7.10 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.

⁹ <https://www.startutorial.com/articles/view/how-to-build-a-web-calendar-in-php>

7.10.1 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 plein 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 rédigée 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 a
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 indispensables avec l'objectif principal du projet : « Réserver une table »

7.10.2 Classer les tâches

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

Tâches	
①	Afficher la liste des établissements
②	Afficher les informations de l'utilisateur
③	Modifier les informations
④	Modifier la photo de profil
	Widget réservations => Pas nécessaire mais ++ page pour afficher les réservations

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.

Figure 57 Liste des tâches pour la page profil

7.10.3 Séparation des tâches client et serveur

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

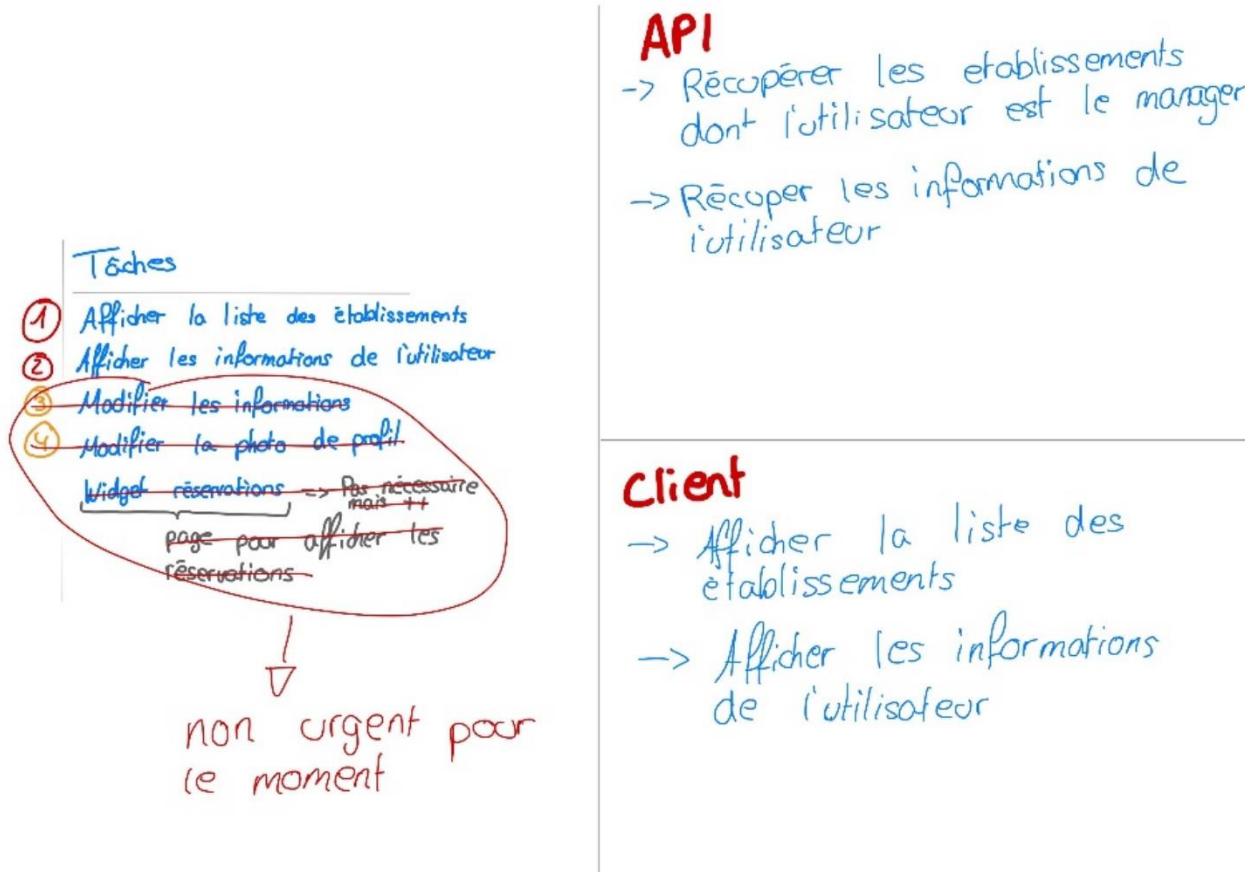


Figure 58 Séparation des tâches client et serveur (API)

7.10.4 Conclusion

La répartition des tâches et le classement de celles-ci par ordre de criticité a 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 avéré très efficace.

7.11 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 où je devais faire preuve de raisonnement afin de trouver la solution.

7.11.1 Réflexions personnelles

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

7.11.1.1 *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.

7.11.1.1.1 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°28. Séparation des jours dans le logbook

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

7.11.1.1.2 Extrait du journal de bord lors de réflexions

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

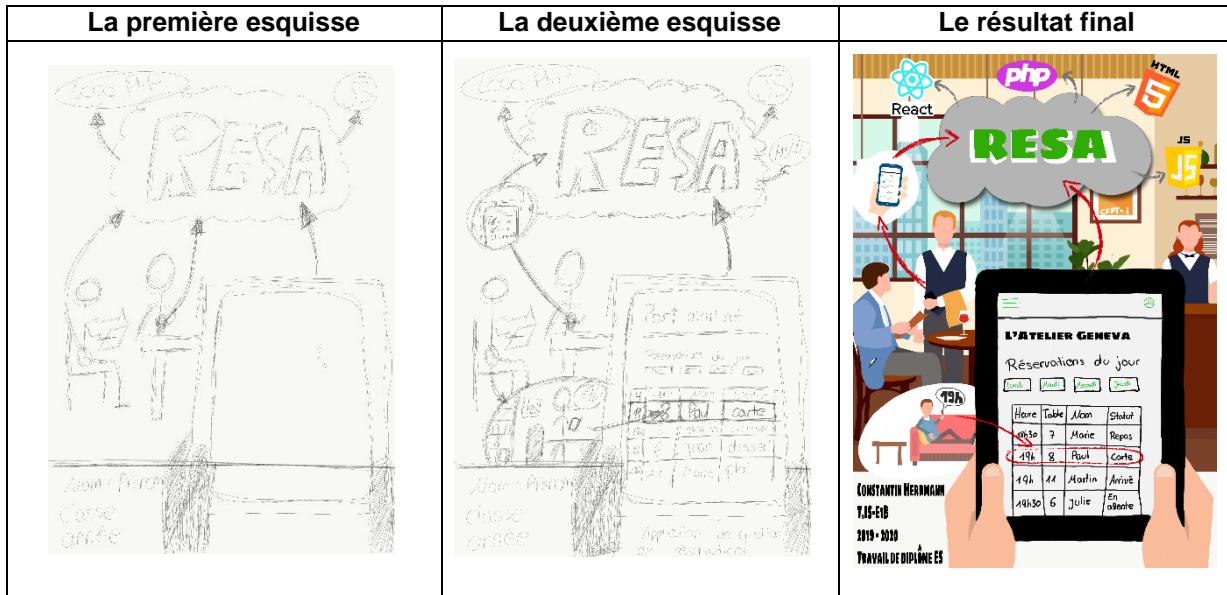
- Il faut que j'ajoute à l'API une fonctionnalité qui gère la différence
 - Je vais donc créer cette fonctionnalité
 - Afin de pouvoir ajouter cette fonctionnalité, j'ai un petit souci... Un utilisateur admin peut se connecter dans tous les restaurants, il faut donc que ma requête gère s'il s'agit d'un 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 publics...)
 - Ajout du fond d'écran de tous les profils utilisateurs
 - Accessible à partir de l'API
 - Il faut que je fasse une table de liaison entre les utilisateurs et une photo de profil

Extrait de code n°29. Extrait du logbook disponible sur Github¹⁰

¹⁰ https://github.com/ConstantinHrrmn/Travail_diplome_ES_2020/blob/master/logbook.md

7.11.1.2 *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.



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 \(plein format\)](#).

7.11.1.3 *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.

7.11.2 Communications avec M. Garcia

Afin de toujours avoir un suivi permanent de mon travail de diplôme et pour donner 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'a beaucoup aidé à raisonner sur les méthodes et les objectifs que je devais avoir afin de compléter correctement mon travail.

7.11.3 Communication avec Mme Perdrizat

Lors de mon travail, nous avons eu plusieurs rendez-vous (à distance, au vu de la situation particulière dans laquelle ce travail a été réalisé) avec Mme Perdrizat. Ces rendez-vous m'ont permis de m'assurer que je partais bien sûr le bon chemin et que ma manière de voir les choses évoluait bien dans son sens à elle aussi.

7.11.3.1 Réorganisation

Lors d'un rendez-vous avec Mme Perdrizat, je lui ai présenté mon plan de l'application suivant :

Après avoir discuté longuement, nous nous sommes rendu compte que je ne prenais pas en compte les

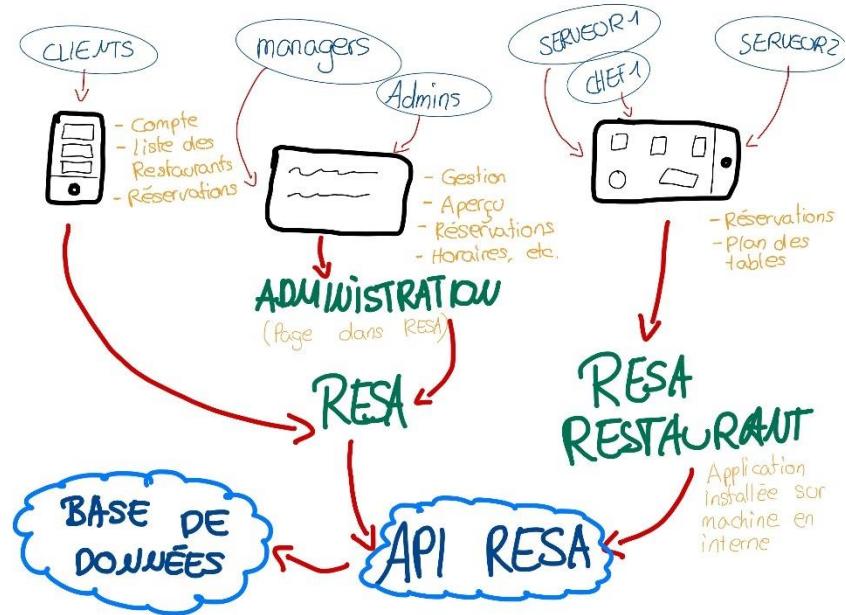


Figure 59 Plan de l'application avant les changements

différents abonnements des restaurants (important de préciser que la notion d'abonnement est apparue lors de cette conversation). Nous avons alors imaginé le nouveau plan de RESA (qui est celui présenté au début de l'analyse fonctionnelle).

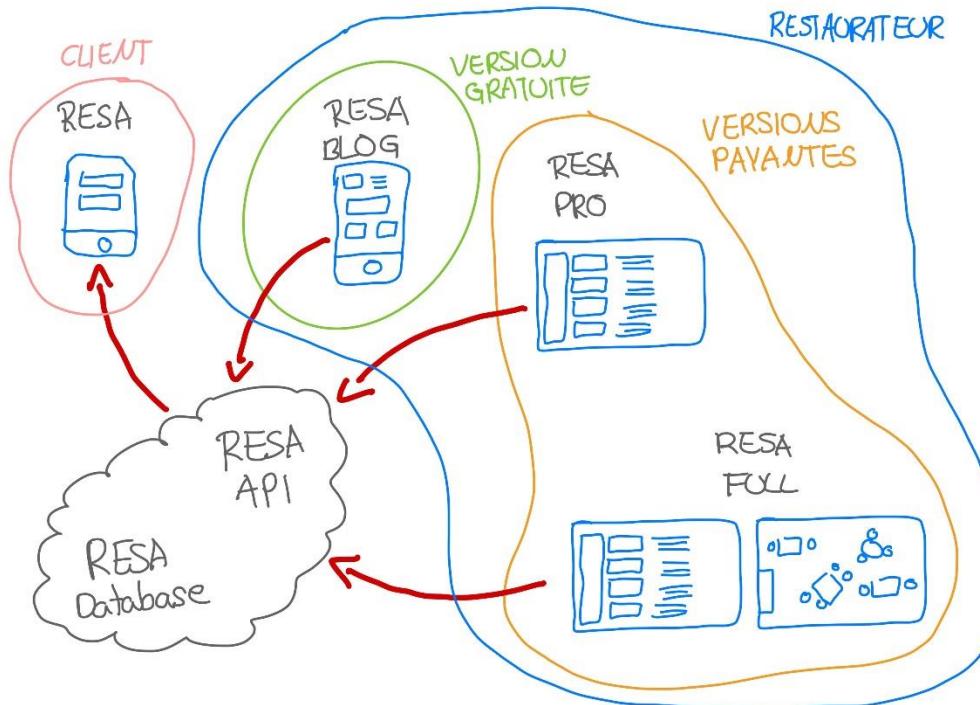


Figure 60 Plan de l'application après discussion sur les niveaux d'abonnements

8 Tables des figures

Figure 1 Logo "lafouchette"	8
Figure 2 Page d'accueil.....	9
Figure 3 Menu de l'utilisateur	9
Figure 4 Page du restaurant	9
Figure 5 Page d'accueil.....	10
Figure 6 Login manager	10
Figure 7 Calendrier du manager	11
Figure 8 Formulaire de création d'un client.....	11
Figure 9 Dropdown menu du statut.....	11
Figure 10 Sélection heure réservation manager.....	12
Figure 11 Sélection date réservation manager.....	12
Figure 12 Formulaire de création de réservation	12
Figure 13 Schéma de la structure RESA	13
Figure 14 Capture d'écran du template.....	14
Figure 15 Capture d'écran de la page d'accueil RESA.....	14
Figure 16 Schéma de navigation RESA Client	15
Figure 17 On est là - Accueil RESA Client.....	15
Figure 18 Page d'accueil de RESA client	16
Figure 19 Menu "non-connecté"	16
Figure 20 Menu "connecté"	16
Figure 21 On est là - page de l'établissement	17
Figure 22 On est là - page de réservation par RESA	18
Figure 23 On est là - Page de login	18
Figure 24 Login d'un utilisateur	18
Figure 25 On est là - Page de profil	19
Figure 26 Navigation possibles depuis page de profil	19
Figure 27 Page de profil.....	19
Figure 28 Login pour un manager de restaurant	20
Figure 29 Schéma de navigation RESA manager	20
Figure 30 Schéma de navigation RESA Blog	21
Figure 31 On est là - accueil blog	21
Figure 32 On est là - paramètres blog	22
Figure 33 On est là - mise en ligne image blog	22
Figure 34 Schéma de navigation RESA Pro.....	23
Figure 35 On est là - accueil RESA Pro.....	23
Figure 36 On est là - Réservations RESA Pro.....	23

Figure 37 On est là - paramètres RESA Pro.....	24
Figure 38 Schéma de navigation RESA Full.....	24
Figure 39 Page de paramètres d'un établissement	25
Figure 40 On est là - réservations RESA Full	25
Figure 41 On est là - Employés RESA Full.....	25
Figure 42 Hiérarchie de RESA.....	26
Figure 43 Schéma des droits d'un administrateur	27
Figure 44 Schéma des droits d'un manager	27
Figure 45 Schéma des droits d'un serveur	27
Figure 46 Schéma des droits d'un client.....	28
Figure 47 Mindmap de préparation	32
Figure 48 : Interface de Github Desktop	33
Figure 49 Diagramme de fonctionnement RESA.....	34
Figure 50 Aperçu interface dbdiagram.io.....	35
Figure 51 Aperçu MCD	36
Figure 52 Schéma questionnement fonctionnement réservations.....	42
Figure 53 Schéma vérification de places	43
Figure 54 Solution fonctionnement de réservations	43
Figure 55 Schéma de lien entre les tables principales	44
Figure 56 Schéma des liens entres les tables pour une zone	44
Figure 57 Liste des tâches pour la page profil	48
Figure 58 Séparation des tâches client et serveur (API)	49
Figure 59 Plan de l'application avant les changements	52
Figure 60 Plan de l'application après discussion sur les niveaux d'abonnements	52
Figure 61 Diagramme d'activité - création d'un compte et login	57
Figure 62 Diagramme d'activité - faire une réservation	58
Figure 63 Création d'un commentaire.....	59
Figure 64 Diagramme de fonctionnement de login.....	60
Figure 65 Redirection application correspondante au login.....	60

9 Tables des extraits de code

Extrait de code n°1.	Arborescence de Github.....	29
Extrait de code n°2.	Commande cmd pour créer le projet react.....	30
Extrait de code n°3.	Arborescence d'un projet react basique.....	30
Extrait de code n°4.	Header des fichier PHP de RESA	30
Extrait de code n°5.	Header d'une fonction PHP	31
Extrait de code n°6.	Envoi d'une requête http à l'API	37
Extrait de code n°7.	Valeur JSON retornée par une fonction de l'API.....	38
Extrait de code n°8.	Objet PHP créer à partir de la valeur JSON.....	38
Extrait de code n°9.	Mise en forme de données avant l'envoi en json	38
Extrait de code n°10.	Formulaire à créer pour correctement envoyer les images à l'API	39
Extrait de code n°11.	Vérification des données reçues par un formulaire	39
Extrait de code n°12.	Création de la requête http pour créer un établissement	39
Extrait de code n°13.	Fonction de sauvegarde de l'image pour un établissement.....	39
Extrait de code n°14.	Balise HTML d'une image avec comme chemin l'image dans l'API	40
Extrait de code n°15.	Création d'une routine dans MySQL	40
Extrait de code n°16.	Comment appeler une routine en MySQL.....	40
Extrait de code n°17.	Les variables utilisées dans ma routine SQL	41
Extrait de code n°18.	La variable de sortie de ma routine SQL.....	41
Extrait de code n°19.	Ma procédure pour vérifier si il y a de la place.....	42
Extrait de code n°20.	Appel de ma routine SQL avec paramètres	42
Extrait de code n°21.	Requête SQL horaire magasin.....	45
Extrait de code n°22.	Requête SQL qui récupère tous les restaurants	46
Extrait de code n°23.	Login d'un utilisateur.....	46
Extrait de code n°24.	Redirection sur la page « fast_login.php »	47
Extrait de code n°25.	Unset d'une variable de la session.....	47
Extrait de code n°26.	Redirection header sur avec une variable.....	47
Extrait de code n°27.	Affichage du calendrier.....	47
Extrait de code n°28.	Séparation des jours dans le logbook	50
Extrait de code n°29.	Extrait du logbook disponible sur Github.....	50

10 Glossaire

11 ANNEXES

11.1 Diagrammes D'activités

11.1.1 Création de comptes

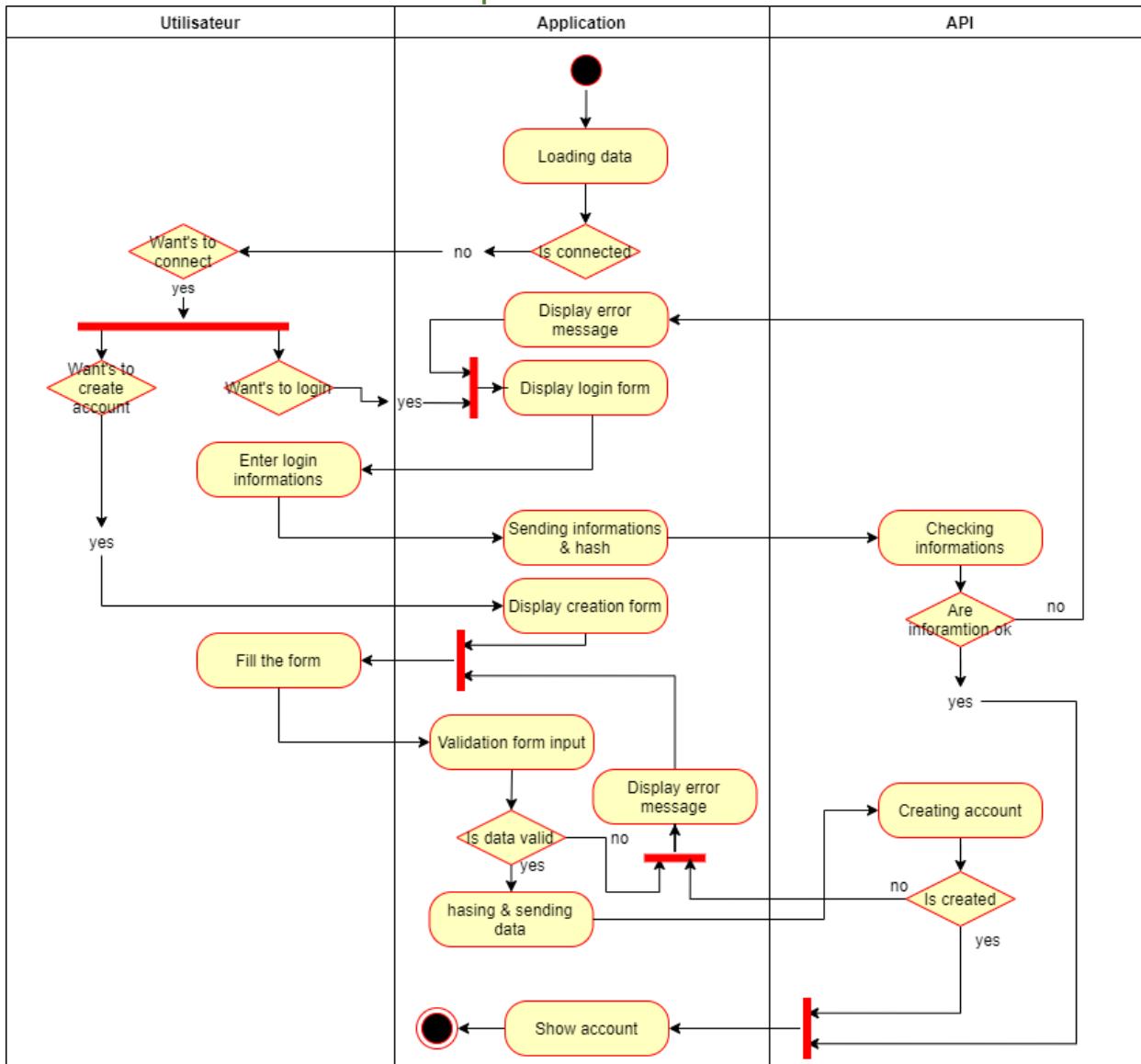


Figure 61 Diagramme d'activité - création d'un compte et login

11.1.2 Réservation

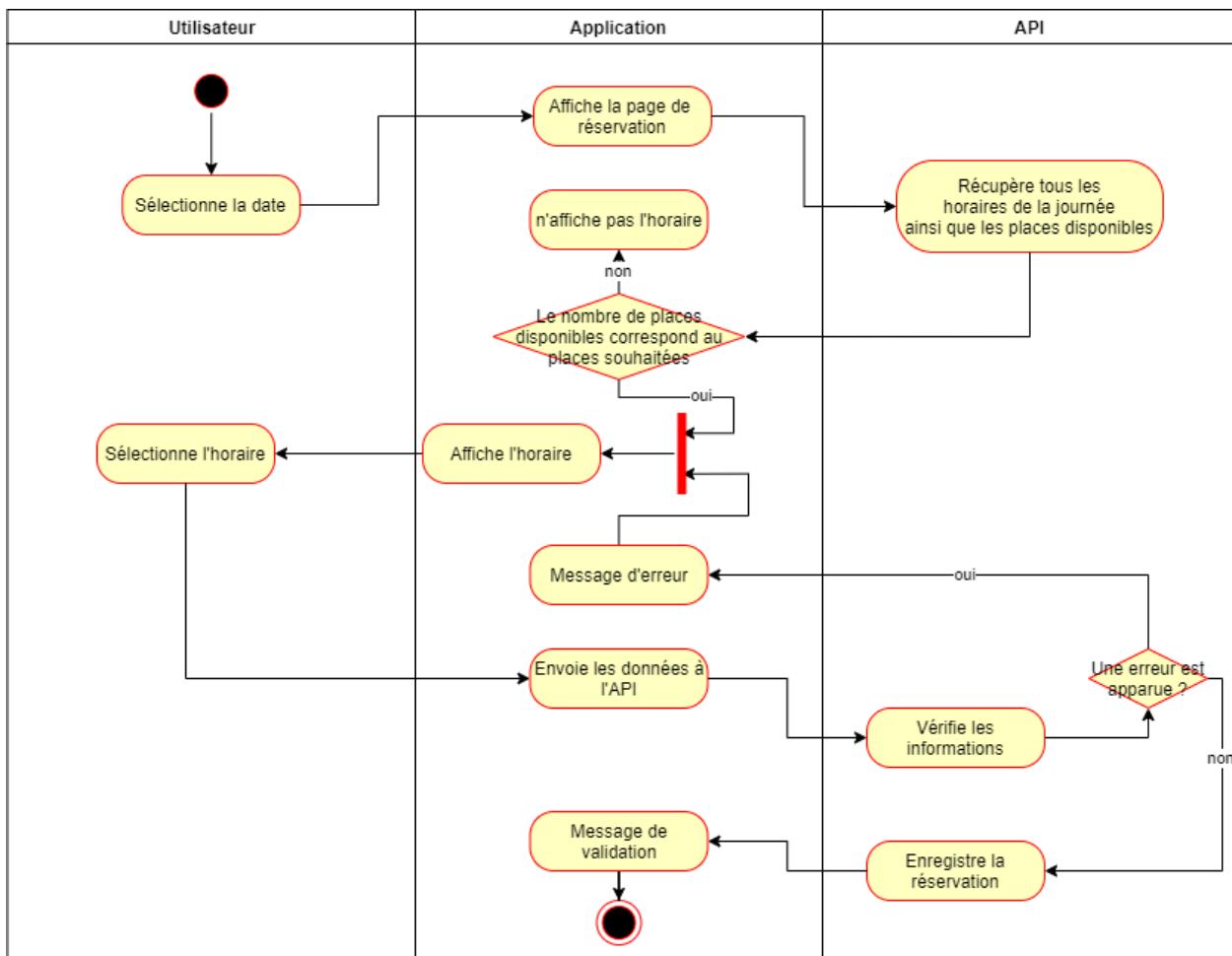


Figure 62 Diagramme d'activité - faire une réservation

11.1.3 Laisser un commentaire

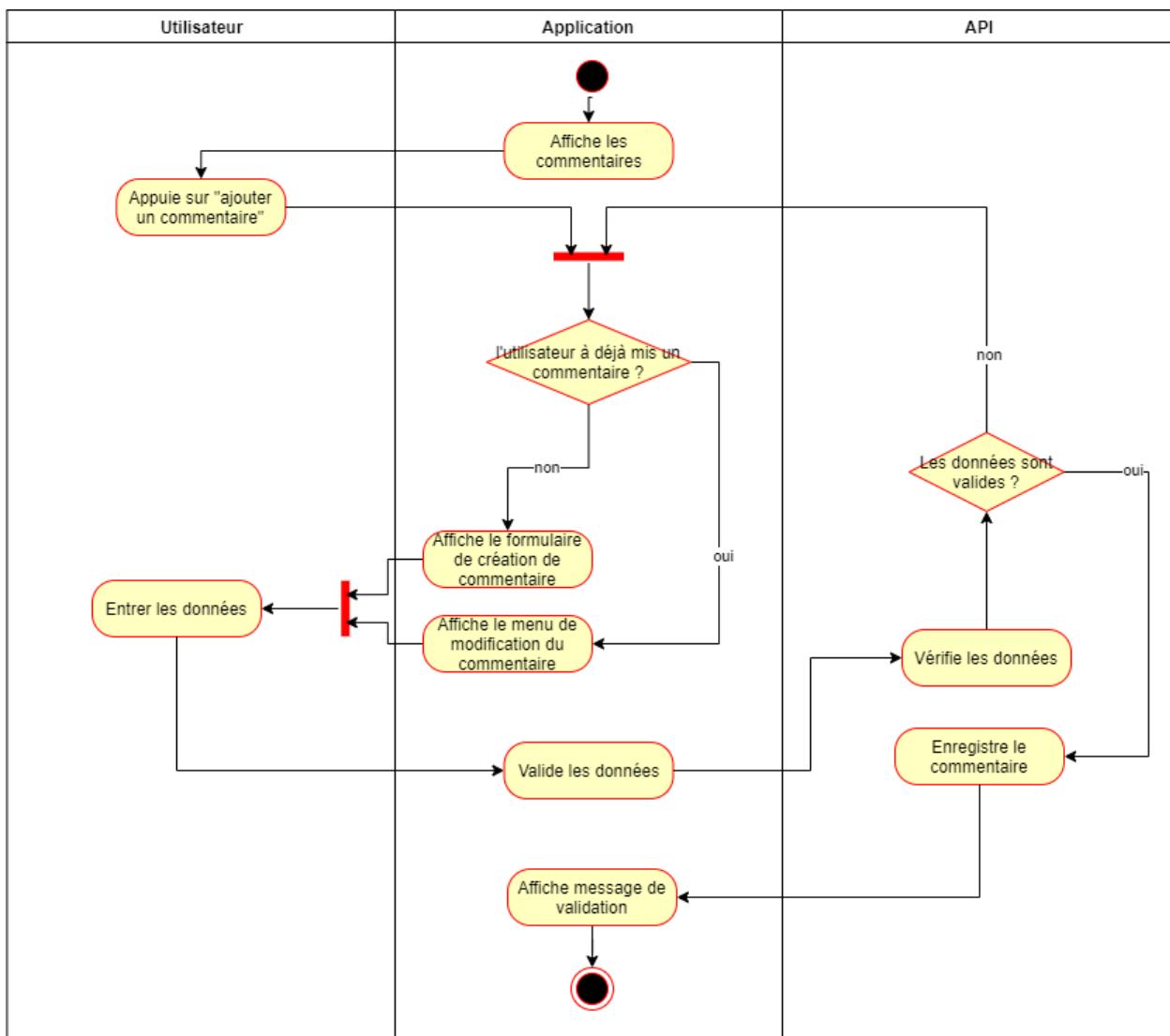


Figure 63 Création d'un commentaire

11.2 Diagrammes de fonctionnement

11.2.1 Login

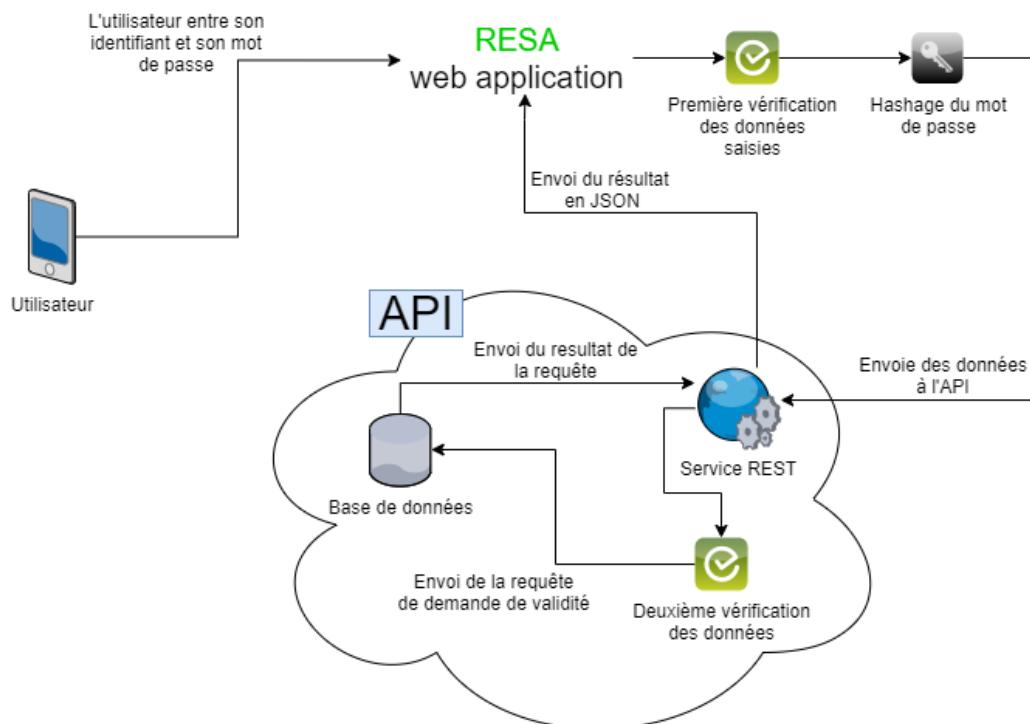


Figure 64 Diagramme de fonctionnement de login

11.2.2 Redirection sur bonne application

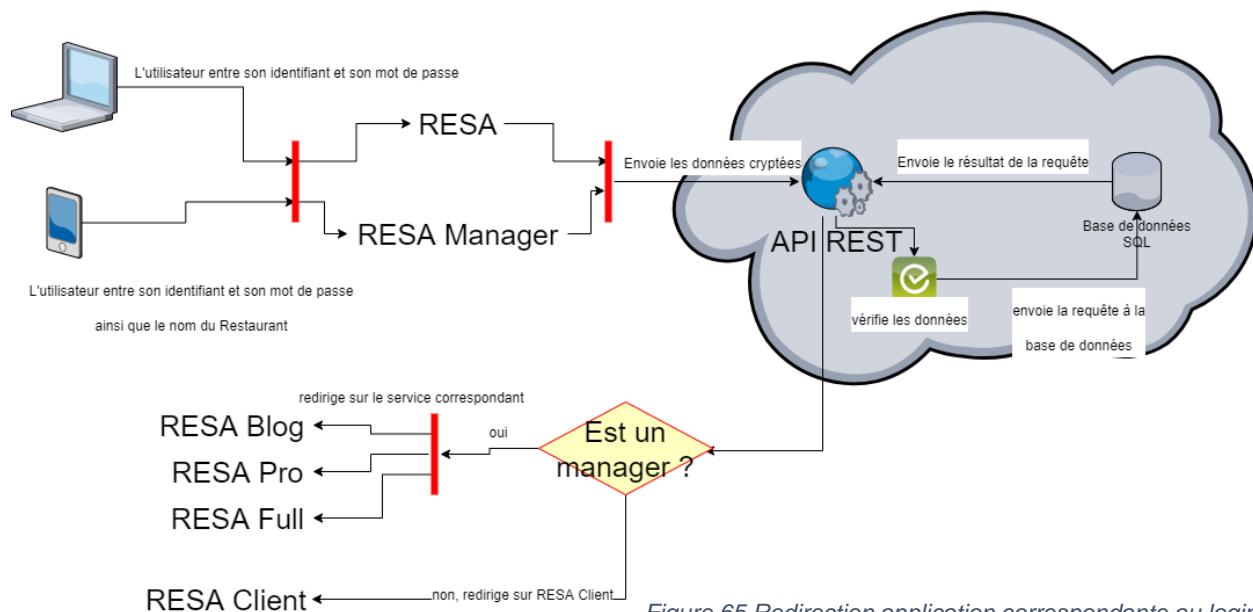
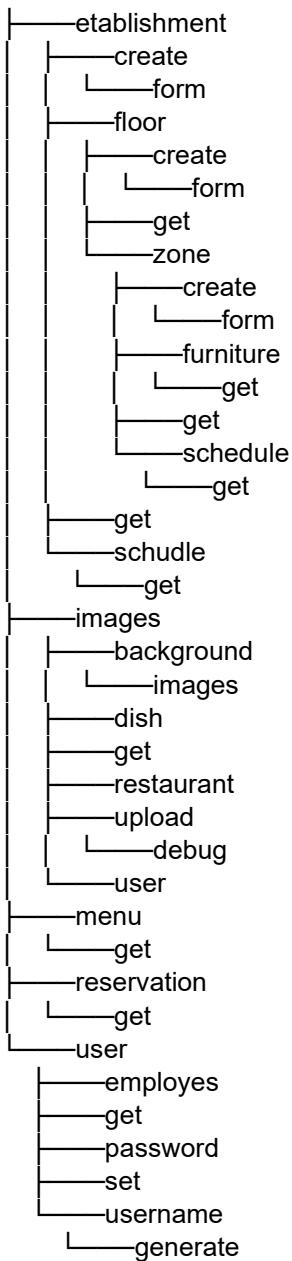


Figure 65 Redirection application correspondante au login

11.3 Structure de l'AP



11.4 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).

11.4.1 Etablissement

11.4.1.1 Create

11.4.1.1.1 Création d'un établissement sans manager

Lien :

```
/api/v2/establishment/create/
```

Paramètres :

```
Name : le nom du restaurant  
Address : l'adresse complète du restaurant  
Phone : le téléphone du restaurant  
Email : l'email du restaurant
```

Lien avec paramètres :

```
/api/v2/establishment/create/?name=[nom]&address=[adresse]&phone=[phone]&email=[email]
```

11.4.1.1.2 Création d'un établissement avec un manager

Lien :

```
/api/v2/establishment/create/
```

Paramètres :

```
Name : le nom du restaurant  
Address : l'adresse complète du restaurant  
Phone : le téléphone du restaurant  
Email : l'email du restaurant  
creatorID : l'id de l'utilisateur qui créer le restaurant
```

Lien avec paramètres :

```
/api/v2/establishment/create/?name=[nom]&address=[adresse]&phone=[phone]&email=[email]&creatorID=[id utilisateur]
```

11.4.1.1.3 Création en passant pas une form

Lien :

```
/api/v2/establishment/create/form/
```

Paramètres :

```
$_POST :  
    Name : le nom du restaurant  
    Address : l'adresse complète du restaurant  
    Phone : le téléphone du restaurant  
    Email : l'email du restaurant  
    creatorID : l'id de l'utilisateur qui créer le restaurant  
$_FILES :  
    Les images ajoutées
```

11.4.1.2 Get

11.4.1.2.1 D'après un id

Lien :

`/api/v2/establishment/get/`

Paramètres :

id : l'id du restaurant

Lien avec paramètres :

`/api/v2/establishment/get/ ?id=[id]`

Retourne :

Id : l'id du restaurant

Name : le nom du restaurant

Address : l'adresse du restaurant

Phone : le numéro de téléphone

Email : l'adresse email

Subscription : le niveau d'abonnement

Menu_name : le nom du menu (null si pas existant)

Menu_description : la description du menu (null si pas existant)

Open : l'id de l'horaire d'ouverture du jour (null si fermé)

11.4.1.2.2 L'id du dernier insérer

Lien :

`/api/v2/establishment/get/`

Paramètres :

Last : (aucune valeur)

Lien avec paramètres :

`/api/v2/establishment/get/ ?last`

Retourne :

Last : l'id du dernier restaurant ajouté

11.4.1.2.3 Tous les établissements d'un manager (utilisateur)

Lien :

`/api/v2/establishment/get/`

Paramètres :

manager : (aucune valeur)

Iduser : l'id de l'utilisateur

Lien avec paramètres :

`/api/v2/establishment/get/ ?manager&iduser=[id]`

Retourne :

Tableau

Id : l'id du restaurant

Name : le nom du restaurant

Address : l'adresse du restaurant

Phone : le numéro de téléphone

Email : l'adresse email

11.4.1.2.4 Niveau d'abonnement

Lien :

`/api/v2/establishment/get/`

Paramètres :

Level : (aucune valeur)

Id : l'id du restaurant

Lien avec paramètres :

`/api/v2/establishment/get/ ?level&id=[id]`

Retourne :

Level : le niveau d'abonnement

11.4.1.2.5 L'horaire des zones

(Non fonctionnelle pour le moment)

Lien :

`/api/v2/establishment/schudle/get/`

Paramètres :

zone : (aucune valeur)

Id : l'id de la zone

Lien avec paramètres :

`/api/v2/establishment/schudle/get?zones&id=[id]`

Retourne :

-

11.4.1.2.6 Horaire du restaurant de la journée (actuelle)

Lien :

`/api/v2/establishment/schudle/get/`

Paramètres :

todayschudles: (aucune valeur)

IdEtab : l'id du restaurant

Lien avec paramètres :

`/api/v2/establishment/schudle/get?todayschudles&idEtab=[id]`

Retourne :

Tableau

Id : l'id de l'horaire

Begin : l'heure de début

End : l'heure de fin

11.4.1.2.7 Horaire et places disponible pour une date

(Non fonctionnelle pour le moment)

Lien :

`/api/v2/establishment/schudle/get/`

Paramètres :

schudlesandplaces: (aucune valeur)

IdEtab : l'id du restaurant

Date : la date recherchée

Lien avec paramètres :

/api/v2/ establishment/schudle/get?schudlesandplaces&idEtab=[id]&date=[date format (YYYY-MM-DD)]

Retourne :

-

11.4.1.2.8 Le jour de la semaine

Lien :

/api/v2/establishment/schudle/get/

Paramètres :

today: (aucune valeur)

Lien avec paramètres :

/api/v2/ establishment/schudle/get?today

Retourne :

Id : l'id du jour

Name : nom du jour

11.4.1.2.9 Récupère les horaires de la semaine pour le restaurant

Lien :

/api/v2/establishment/schudle/get/

Paramètres :

id: l'id du restaurant

Lien avec paramètres :

/api/v2/ establishment/schudle/get?id=[id]

Retourne :

Tableau

Did : id du jour

Dname : le nom du jour

Si le restaurant est ouvert :

Sid : l'id de l'horaire

Sbegin : l'heure de début

Send : l'heure de fin

Si le restaurant est fermé :

Closed : « closed »

11.4.2 Etages

11.4.2.1 Get

11.4.2.1.1 Tous les étages de l'établissement

Lien :

/api/v2/ establishment/floor/get/

Paramètres :

id: l'id du restaurant

Lien avec paramètres :

/api/v2/ establishment/floor/get?id=[id]

Retourne :

Tableau [index du tableau = id de l'étage]

Id : id de l'étage

Name : nom de l'étage

Zones : tableau de zones

1 : Nom de l'étage : string

2 : Heure de début (si existe, sinon null)

3 : Heure de fin (si existe, sinon null)

4->places_total : le nombre de places total dans la zone

11.4.2.2 Create

11.4.2.2.1 Création par l'API

Lien :

/api/v2/ establishment/floor/create/

Paramètres :

Name : le nom de la nouvelle zone

id: l'id du restaurant

Lien avec paramètres :

/api/v2/ establishment/floor/create/?create&name=[nom]&establishment=[id etablissement]

Retourne :

-

11.4.2.2.2 Création par un formulaire

Lien :

/api/v2/ establishment/floor/create/form/

Paramètres :

\$_POST

Name : le nom de la zone

Establishment : l'id de l'établissement

Lien avec paramètres :

/api/v2/ establishment/floor/create/form/

Retourne :

-

11.4.3 Zones

11.4.3.1 GET

11.4.3.1.1 L'id de la dernière zone créée

Lien :

/api/v2/ establishment/floor/zone/get/

Paramètres :

Last : (aucune valeur)

Lien avec paramètres :

/api/v2/ establishment/floor/zone/get/?last

Retourne :

Last : l'id de la dernière zone ajoutée

11.4.3.1.2 Les places dans la zone

Lien :

/api/v2/ establishment/floor/zone/get/

Paramètres :

Places : (aucunes valeur)

Id : id de la zone

Lien avec paramètres :

/api/v2/ establishment/floor/zone/get/?places&id=[id]

Retourne :

Places_totales : le nombre de places totales dans la zone

11.4.3.1.3 Tous les horaires de la zone

Lien :

/api/v2/ establishment/floor/zone/schedule/get/

Paramètres :

all : (aucunes valeur)

Id : id de la zone

Lien avec paramètres :

/api/v2/ establishment/floor/zone/schudle/get/?all&id=[id]

Retourne :

Tableau

Schedule_id : l'id de l'horaire

Begin : l'heure de début

End : l'heure de fin

11.4.3.2 CREATE

11.4.3.2.1 Crédation par l'API

Lien :

/api/v2/ establishment/floor/zone/create/

Paramètres :

create : (aucunes valeur)

name : le nom de la nouvelle zone

Lien avec paramètres :

/api/v2/ establishment/floor/zone/create/?create&name=[nom]

Retourne :

-

11.4.3.2.2 Lien entre étage et zone

Lien :

```
/api/v2/ establishment/floor/zone/create/
```

Paramètres :

link : (aucunes valeur)

zone : l'id de la zone

floor : l'id de l'étage

Lien avec paramètres :

```
/api/v2/ establishment/floor/zone/create/?link&floor=[id etage]&zone=[id zone]
```

Retourne :

```
-
```

11.4.3.2.3 Crédation par un formulaire

Lien :

```
/api/v2/ establishment/floor/zone/create/form/
```

Paramètres :

\$_POST

Name : le nom de la zone

Floor : l'id de l'étage

Lien avec paramètres :

```
/api/v2/ establishment/floor/zone/create/form/
```

Retourne :

```
-
```

11.4.4 Fournitures

11.4.4.1 GET

11.4.4.1.1 Les fournitures d'une zone

Lien :

```
/api/v2/ establishment/floor/zone/furniture/get/
```

Paramètres :

Zone : (aucune valeur)

Id : id de la zone

Lien avec paramètres :

```
/api/v2/ establishment/floor/zone/furniture/get/?zone&id=[id]
```

Retourne :

Tableau

Fid : l'id de la fourniture

Fname : le nom de la fourniture

Fcolor : la couleur de la fourniture

Fplaces : le nombre de places

11.4.4.1.2 Les fournitures d'un établissement

Lien :

```
/api/v2/establishment/floor/zone/create/form/
```

Paramètres :

\$POST

Name : le nom de la zone

Floor : l'id de l'étage

Lien avec paramètres :

```
/api/v2/establishment/floor/zone/create/form/
```

Retourne :

```
-
```

11.4.5 Images

11.4.5.1 GET

11.4.5.1.1 Les informations d'après l'id

Lien :

```
/api/v2/images/get/
```

Paramètres :

Data : (aucune valeur)

Id : l'id de l'image

Lien avec paramètres :

```
/api/v2/images/get/?data&id=[id]
```

Retourne :

Si l'image existe :

Id : l'id de l'image

Alt : la description de l'image

Path : le chemin de l'image sur l'api

userID : l'id de l'utilisateur qui a mis en ligne l'image

first_name : le prénom de l'utilisateur qui a mis en ligne l'image

Si l'image n'existe pas :

false

11.4.5.1.2 Les images pour un établissement

Lien :

```
/api/v2/images/get/
```

Paramètres :

establishment : (aucune valeur)

Id : l'id de l'établissement

Lien avec paramètres :

```
/api/v2/images/get/? establishment &id=[id]
```

Retourne :

Tableau

Full_path : le chemin complet de l'image dans l'api

11.4.5.1.3 Redirection sur l'image

Lien :

/api/v2/images/get/

Paramètres :

Id : l'id de l'image

Lien avec paramètres :

/api/v2/images/get/?id=[id]

Retourne :

Redirection sur l'image dans le navigateur

11.4.6 Menu

11.4.6.1 GET

11.4.6.1.1 Récupérer les plats d'un restaurant

Lien :

/api/v2/menu/get/

Paramètres :

Dishes : (aucune valeur)

Id : l'id de l'établissement

Lien avec paramètres :

/api/v2/menu/get/?dishes&id=[id]

Retourne :

Tableau

Dish_name : le nom du plat

Dish_price : le prix

Dish_type : le type de plat

11.4.6.1.2 Récupérer tous les plats de la base

Lien :

/api/v2/menu/get/

Paramètres :

Dishes : (aucune valeur)

Lien avec paramètres :

/api/v2/menu/get/?dishes

Retourne :

Tableau

Id : l'id du plat

Dish_name : le nom du plat

Dish_price : le prix

Dish_type : le type de plat

11.4.6.1.3 Récupérer les menus d'un restaurant

Lien :

```
/api/v2/menu/get/
```

Paramètres :

meals : (aucune valeur)

Id : l'id de l'établissement

Lien avec paramètres :

```
/api/v2/menu/get/?meals&id=[id]
```

Retourne :

Tableau

Id : l'id du plat

Dish_name : le nom du plat

Dish_price : le prix

Dish_type : le type de plat

11.4.6.1.4 Récupérer l'id du menu du restaurant

Lien :

```
/api/v2/menu/get/
```

Paramètres :

menu : (aucune valeur)

Id : l'id de l'établissement

Lien avec paramètres :

```
/api/v2/menu/get/?menu&id=[id]
```

Retourne :

Tableau

Id : l'id du menu

name: le nom du menu

description : description du menu

11.4.6.1.5 Récupérer la carte complète du restaurant

Lien :

```
/api/v2/menu/get/
```

Paramètres :

all : (aucune valeur)

Id : l'id de l'établissement

Lien avec paramètres :

```
/api/v2/menu/get/?all&id=[id]
```

Retourne :

Infos

Id : l'id du menu

Name : le nom du menu

Description : la description du menu

Dishes

Dish_name : le nom du plat

Dish_type : le type de plat

Dish_price : le prix du plat

Meals

Meal_id : l'id du menu composé

Meal_name : le nom du menu composé

Entrance_name : le nom de l'entrée

Main_name : le nom du plat principal

Dessert_name : le nom du dessert

Drink_name : le nom de la boisson

Price : le prix du menu composé

11.4.6.1.6 Récupérer les réservations pour une intervalle d'heure

Lien :

/api/v2/reservation/get/

Paramètres :

range : (aucune valeur)

begin : l'heure de début

end : l'heure de fin

etab : l'id de l'établissement

Lien avec paramètres :

/api/v2/reservation/get/ ?range&begin=[heure début]&end=[heure fin]&etab=[id établissement]

Retourne :

Tableau

Rid : L'id de la réservation

Arrival : date et heure d'arrivée

Amount : le nombre de personnes dans la réservation

Fid : l'id de la fourniture

Fname : le nom de la fourniture

Fplaces : le nombre de places disponibles sur la fourniture

Uid : l'id de l'utilisateur

Ufirstname : le prénom de l'utilisateur

Ulastname : le nom de famille de l'utilisateur

Uphone : le numéro de téléphone de l'utilisateur

Umail : l'email de l'utilisateur

11.4.6.1.7 Vérifier la possibilité de réserver

Lien :

/api/v2/menu/get/

Paramètres :

Full : (aucune valeur)

arrival : heure d'arrivée

duration : La durée estimée du repas en secondes

date : la date de réservation

etab : l'id de l'établissement

Lien avec paramètres :

/reservation/get/?full&arrival=[heure]&duration=[durée en sec.]&date=[date au format YYYY-MM-DD]&etab=[id établissement]

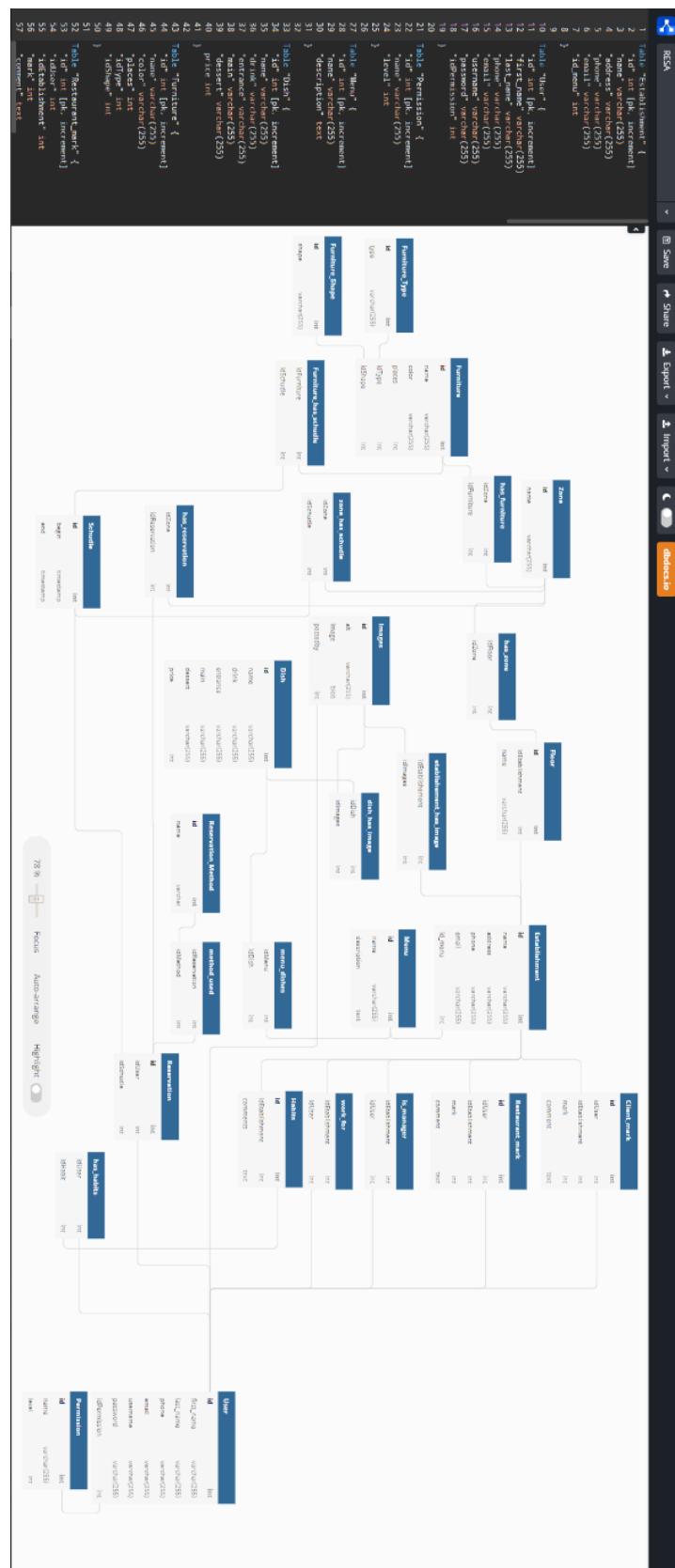
Retourne :

Avaible : le nombre de places disponibles

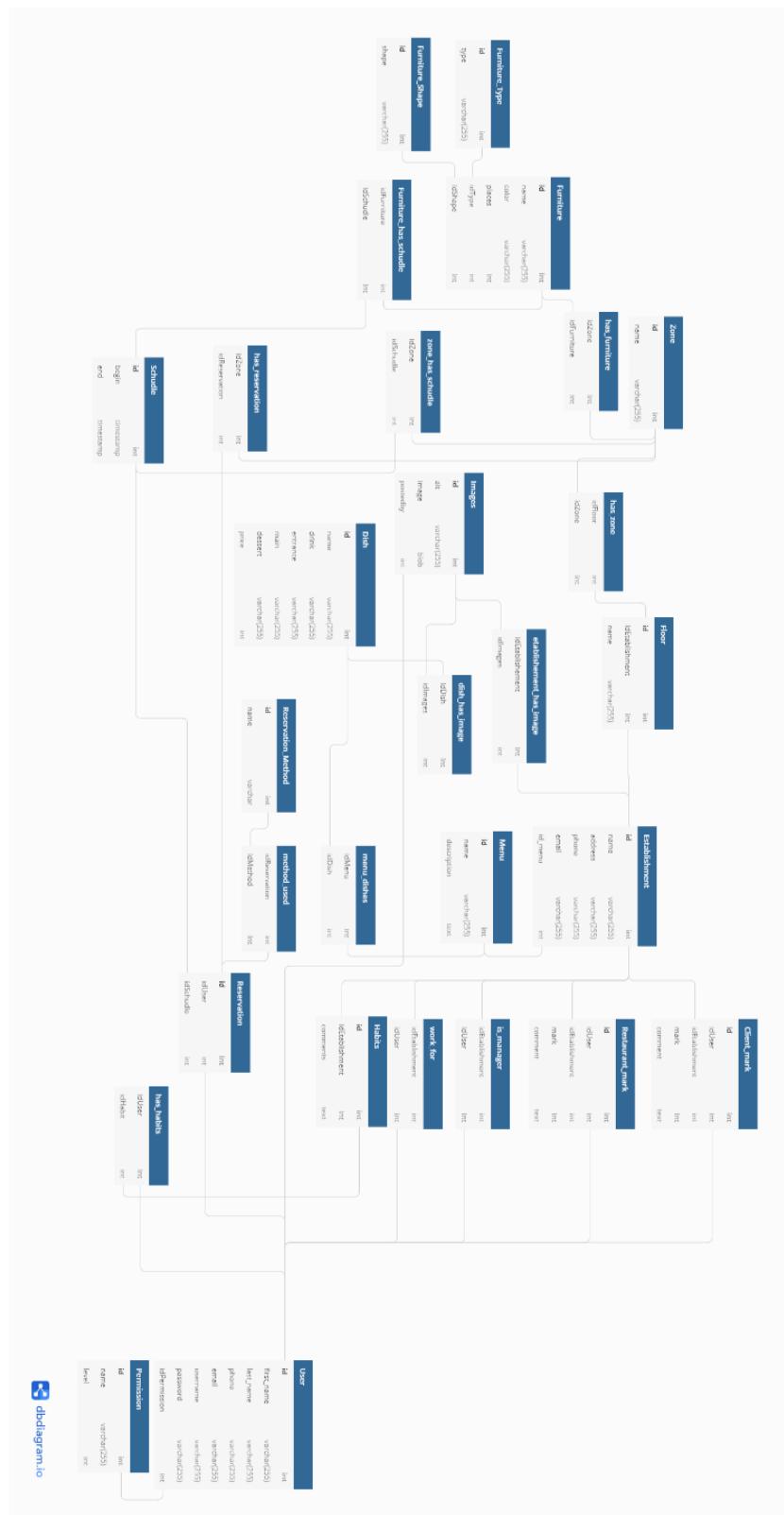
11.5 Images (pleins format)

7.8.5	Réservations et horaires.....	40
11.4.1	Etablissement	62
11.4.6	Menu	70
11.5.1	Interface dbdiagram.io	75
11.5.2	MCD de la base de données	76
11.5.3	Esquisse n°1 Poster	77
11.5.4	Esquisse n°2 Poster	78
11.5.5	Poster final RESA	79

11.5.1 Interface dbdiagram.io



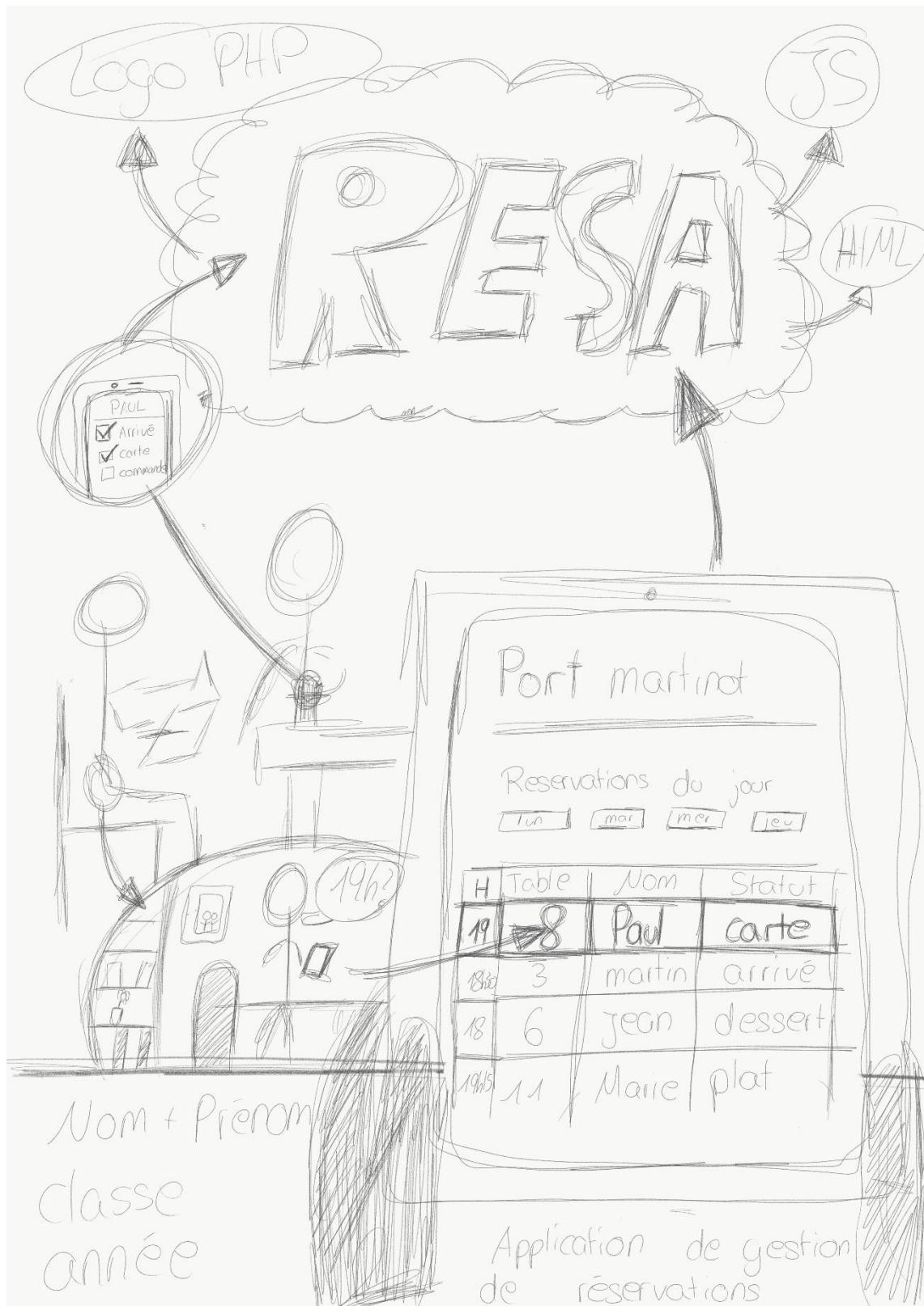
11.5.2 MCD de la base de données



11.5.3 Esquisse n°1 Poster



11.5.4 Esquisse n°2 Poster



11.5.5 Poster final RESA

