

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 them a textured appearance.

Constantin Herrmann

avril - juin 2020

M. Francisco Garcia

CFPT-I Technicien ES

Version 1.0 (Rendu final)

1 Table de matières

1	Table de matières	2
2	Résumé (intermédiaire)	11
3	Abstract (intermediary).....	11
4	Introduction	12
5	Analyse de l'existant	12
5.1	La Fourchette.....	12
5.1.1	Clients	13
5.1.2	Partie manager de « the Fork ».....	14
6	Cahier des charges.....	17
7	Analyse fonctionnelle	22
7.1	La structure de RESA.....	22
7.2	Interfaces.....	23
7.2.1	Template	23
7.2.2	RESA Client	24
7.2.2.1	Page d'accueil	24
7.2.2.2	Page de l'établissement	26
7.2.2.3	Se connecter	27
7.2.2.4	Profil utilisateur.....	27
7.2.3	Accès aux applications des managers RESA.....	29
7.2.3.1	Création d'un établissement	30
7.2.4	RESA Blog	31
7.2.4.1	Accueil.....	32
7.2.4.2	Paramètres.....	32
7.2.4.3	Les images.....	33
7.2.5	RESA Pro	34
7.2.5.1	Accueil.....	34
7.2.5.2	Réservations	34
7.2.5.3	Paramètres.....	35
7.2.6	RESA Full.....	36
7.2.6.1	Accueil.....	36
7.2.6.2	Paramètres.....	37
7.2.6.3	Réservations	37
7.2.6.4	Employés.....	37
7.2.6.5	Plan de table	39
7.2.6.5.1	Code couleur des tables	39
7.2.6.5.2	Modifier le plan de table	40

7.3	Les droits des utilisateurs	41
7.3.1	Utilisateurs.....	41
7.3.1.1	Administrateur	42
7.3.1.2	Manager (et chef)	42
7.3.1.3	Serveur.....	42
7.3.1.4	Stagiaire et externe	43
7.3.1.5	Client	43
8	Analyse organique	44
8.1	Mise en place.....	44
8.1.1	GitHub	44
8.1.2	Trello	44
8.2	Prétravail	44
8.2.1	Programmation.....	44
8.2.2	Installation de React.....	45
8.2.3	Conventions	45
8.2.3.1	En-tête de fichier	45
8.2.3.2	En-tête de fonction	46
8.2.3.3	Diagrammes d'activités	46
8.2.4	Organisationnel	47
8.3	Environnement.....	48
8.3.1	Laragon	48
8.3.2	Visual Studio Code.....	48
8.3.3	EDUGE	48
8.3.4	Github Desktop.....	48
8.4	Schémas de fonctionnements	49
8.4.1	RESA	49
8.5	Diagrammes de fonctionnement de RESA	49
8.6	Diagrammes d'activités.....	49
8.6.1	Diagrammes d'activités de RESA.....	50
8.7	Base de données.....	50
8.7.1	UML.....	50
8.7.2	Privilèges.....	50
8.7.3	Structure.....	51
8.7.4	Données de tests	51
8.7.4.1	Utilisateurs.....	51
8.7.4.2	Établissements	51
8.7.4.3	Les niveaux d'abonnement des restaurants	51

8.7.5	Tables	52
8.8	RESA Apps (Client/Blog/Pro/Full).....	52
8.8.1	Resa-project.ch	52
8.8.2	SESSION	52
8.9	API.....	53
8.9.1	Structure	53
8.9.2	Variables globales	53
8.9.3	Communications avec l'API.....	53
8.9.3.1	Envoi	53
8.9.3.1.1	Exemple.....	53
8.9.3.2	Réceptions	54
8.9.4	Gestion des images.....	55
8.9.4.1	Mise en ligne d'une image.....	55
8.9.4.1.1	Exemple.....	55
8.9.4.2	Récupérer les images.....	56
8.9.4.2.1	Exemples	56
8.9.5	Réservations et horaires	56
8.9.5.1	Le nombre de places disponibles grâce aux routines.....	56
8.9.5.1.1	Les routines MySQL	56
8.9.5.1.2	Création de ma routine	57
8.9.5.2	Système de réservations	58
8.9.6	Les établissements.....	61
8.9.6.1	Les Zones et les fournitures	61
8.9.6.2	Le fonctionnement des zones.....	62
8.10	L'application	62
8.10.1	Affichage du statut ouvert / fermé des restaurants	62
8.10.2	Login vs Fast-login vs Manager-login.....	63
8.10.2.1	Login.....	63
8.10.2.2	Fast-login.....	63
8.10.2.3	Manager-login.....	64
8.10.3	Le calendrier de réservation	64
8.11	Gestion du temps	64
8.11.1	Lister les tâches	65
8.11.2	Classer les tâches	65
8.11.3	Le planning.....	66
8.11.4	Séparation des tâches client et serveur.....	67
8.11.5	Conclusion.....	67

8.12	Raisonnements	67
8.12.1	Réflexions personnelles	67
8.12.1.1	Le journal de bord.....	68
8.12.1.1.1	Structure	68
8.12.1.1.2	Extrait du journal de bord lors de réflexions.....	68
8.12.1.2	Création de croquis	69
8.12.1.3	Analyse.....	69
8.12.2	Communications avec M. Garcia.....	69
8.12.3	Communication avec Mme Perdrizat.....	69
8.12.3.1	Réorganisation	70
8.13	Améliorations futures.....	71
8.13.1	Finaliser les vues.....	71
8.13.2	API	71
8.13.3	Paiement	71
8.13.4	Connexion par Google/Facebook/Apple.....	71
8.14	Bilans	71
8.14.1	Bilan Technique.....	71
8.14.2	Bilan personnel.....	71
8.14.2.1	Situation particulière	72
8.14.2.2	Apport personnel	72
8.15	Conclusion.....	72
9	Tables des figures	73
10	Tables des extraits de code	76
11	ANNEXES	77
11.1	Diagrammes D'activités.....	77
11.1.1	Création de comptes	77
11.1.2	Réservation	78
11.1.3	Laisser un commentaire	79
11.2	Diagrammes de fonctionnement.....	80
11.2.1	Login.....	80
11.2.2	Redirection sur bonne application	80
11.3	Cheat Sheet de l'API	81
11.3.1	Etablissement.....	81
11.3.1.1	Create.....	81
11.3.1.1.1	Création d'un établissement sans manager.....	81
11.3.1.1.2	Création d'un établissement avec un manager.....	81
11.3.1.1.3	Création en passant pas une form.....	81

11.3.1.2	Get.....	82
11.3.1.2.1	D'après un id.....	82
11.3.1.2.2	L'id du dernier insérer	82
11.3.1.2.3	Tous les établissements d'un manager (utilisateur).....	82
11.3.1.2.4	Niveau d'abonnement.....	83
11.3.1.2.5	L'horaire des zones.....	83
11.3.1.2.6	Horaire du restaurant de la journée (actuelle).....	83
11.3.1.2.7	Horaire et places disponible pour une date.....	84
11.3.1.2.8	Le jour de la semaine.....	84
11.3.1.2.9	Récupère les horaires de la semaine pour le restaurant.....	84
11.3.2	Etages	85
11.3.2.1	Get.....	85
11.3.2.1.1	Tous les étages de l'établissement.....	85
11.3.2.2	Create.....	85
11.3.2.2.1	Création par l'API.....	85
11.3.2.2.2	Création par un formulaire	85
11.3.3	Zones	86
11.3.3.1	GET	86
11.3.3.1.1	L'id de la dernière zone créée.....	86
11.3.3.1.2	Les places dans la zone	86
11.3.3.1.3	Tous les horaires de la zone.....	86
11.3.3.2	CREATE	87
11.3.3.2.1	Création par l'API.....	87
11.3.3.2.2	Lien entre étage et zone	87
11.3.3.2.3	Création par un formulaire	87
11.3.4	Fournitures	87
11.3.4.1	GET	87
11.3.4.1.1	Les fournitures d'une zone.....	87
11.3.4.1.2	Les fournitures d'un établissement	88
11.3.5	Images.....	88
11.3.5.1	GET	88
11.3.5.1.1	Les informations d'après l'id	88
11.3.5.1.2	Les images pour un établissement	89
11.3.5.1.3	Redirection sur l'image	89
11.3.6	Menu	89
11.3.6.1	GET	89
11.3.6.1.1	Récupérer les plats d'un restaurant	89

11.3.6.1.2	Récupérer tous les plats de la base.....	89
11.3.6.1.3	Récupérer les menus d'un restaurant.....	90
11.3.6.1.4	Récupérer l'id du menu du restaurant.....	90
11.3.6.1.5	Récupérer la carte complète du restaurant.....	90
11.3.6.1.6	Récupérer les réservations pour une intervalle d'heure.....	91
11.3.6.1.7	Vérifier la possibilité de réserver	91
11.3.7	Subscriptions.....	92
11.3.7.1	GET	92
11.3.7.1.1	Get all subscriptions.....	92
11.3.7.1.2	Get by Id	92
11.4	Journal de bord	93
11.4.1	06.04.20	93
11.4.2	07.04.20	93
11.4.3	08.04.20	93
11.4.3.1	API V1	94
11.4.3.2	Login avec l'API (MAJ le 23.04.2020).....	94
11.4.4	09.04.20	94
11.4.4.1	API : READ Client.....	94
11.4.5	14.04.20	94
11.4.5.1	Comptes pour les test BDD	95
11.4.5.1.1	Employés	95
11.4.5.2	Diverses informations pour le restaurant	95
11.4.5.2.1	Floors (étages).....	95
11.4.6	15.04.20	96
11.4.7	19.04.20	96
11.4.7.1	Les niveaux de permissions	96
11.4.7.2	API V2	97
11.4.7.3	Les changements par rapport à la v1	97
11.4.7.4	La table "user"	97
11.4.7.5	Récuperer tous les employés qui sont dans la table "user".....	98
11.4.8	20.04.20	98
11.4.8.1	/!\ Problème 1	98
11.4.8.1.1	Problématique.....	98
11.4.8.1.2	Solution possible.....	99
11.4.8.1.3	Solution.....	99
11.4.8.2	Etablissement.....	99
11.4.9	21.04.20	100

11.4.10 22.04.20	101
11.4.10.1 Récupérer le menus d'un restaurant.....	102
11.4.10.2 Unire les 2 résultats de recherche	104
11.4.10.3 Appel avec M. Garcia	104
11.4.10.4 Test de "distinc" et de "group by"	104
11.4.10.4.1 Group by	104
11.4.10.4.2 Distinc.....	104
11.4.10.4.3 Conclusion.....	104
11.4.11 23.04.20	104
11.4.11.1 APPEL M. Garcia	105
11.4.11.2 Bonnes pratiques SQL	105
11.4.12 24.04.20	106
11.4.12.1 Evalutation intermédiaire (appel avec m. Garcia)	108
11.4.12.2 Documentation	108
11.4.12.3 Github.....	108
11.4.12.4 Google Drive.....	108
11.4.12.5 Serveur.....	108
11.4.12.6 Login.....	109
11.4.12.7 Images.....	109
11.4.12.8 Poster	109
11.4.13 26.04.20	109
11.4.13.1 Page Login	109
11.4.13.2 Idées pour plus tard afin de encore plus me faciliter la vie	109
11.4.14 27.04.20	109
11.4.14.1 Appel avec m. Garcia	109
11.4.15 28.04.20	109
11.4.15.1 Appel avec m. Garcia	112
11.4.16 29.04.20	112
11.4.16.1 Requête de login employé	112
11.4.16.2 Création de la page de profil.....	113
11.4.16.3 Appel avec m. Garcia	113
11.4.17 30.04.20	113
11.4.17.1 Création d'un nouvel établissement.....	113
11.4.17.2 MAJ API.....	114
11.4.17.3 Appel avec m. Garcia	115
11.4.18 Programme pour demain (avant que j'oublie).....	115
11.4.19 03.05.20	115

11.4.20 04.05.20	117
11.4.20.1 Appel avec m. Garcia	118
11.4.21 05.05.20	118
11.4.21.1 Appel avec m. Garcia	118
11.4.22 06.05.20	119
11.4.23 Création poster.....	119
11.4.24 Appel avec m. Garcia	119
11.4.25 07.05.20	119
11.4.25.1 Appel avec m. Garcia	119
11.4.26 08.05.20	119
11.4.26.1 Appel avec m. Garcia	119
11.4.26.2 Application.....	120
11.4.27 12.05.20	120
11.4.27.1 Les horaires.....	121
11.4.27.2 Page admin	121
11.4.27.3 Zones et fournitures.....	121
11.4.28 Appel avec m. Garcia	122
11.4.29 13.05.20	122
11.4.29.1 Appel avec m. Garcia	123
11.4.29.2 MEMO	123
11.4.30 14.05.20	124
11.4.30.1 Technique.....	124
3. Problème.....	125
11.4.30.2 Appel avec M. Garcia	127
11.4.30.3 Application RESA Client	127
11.4.30.4 Application RESA Restaurant.....	127
11.4.30.5 Reunion avec Nadège	128
11.4.31 17.05.20	128
11.4.31.1 RESA.....	128
11.4.31.2 RESA BLOG.....	128
11.4.31.3 RESA PRO	128
11.4.31.4 RESA FULL	128
11.4.31.5 Nouveau planning des choses importantes à faire	130
11.4.31.5.1 RESA Client.....	130
11.4.31.5.2 RESA Blog.....	130
11.4.31.5.3 RESA Pro	130
11.4.31.5.4 RESA Full	130

11.4.32	18.05.20	130
11.4.33	Appel avec M. Garcia	130
11.4.34	19.05.20	131
11.4.34.1	Appel avec m. Garcia	132
11.4.35	20.05.20	132
11.4.35.1	Appel avec m. Garcia	134
11.4.36	21.05.20	134
11.4.37	22.05.20	134
11.4.37.1	Appel avec m. Garcia (avant évaluation intermédiaire)	134
11.4.38	23.05.20	135
11.4.38.1	Evaluation intermediaire	135
11.4.39	27.05.20	135
11.4.40	28.05.20	135
11.4.40.1	Appel avec m. Garcia	135
11.4.41	29.05.20	136
11.4.42	02.06.20	136
11.4.43	03.06.20	136
11.4.44	04.06.20	137
11.4.45	Jour 326493 (je pers la tête).....	137
11.4.45.1	Appel avec m. Garcia	137
11.5	Images (pleins format).....	138

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

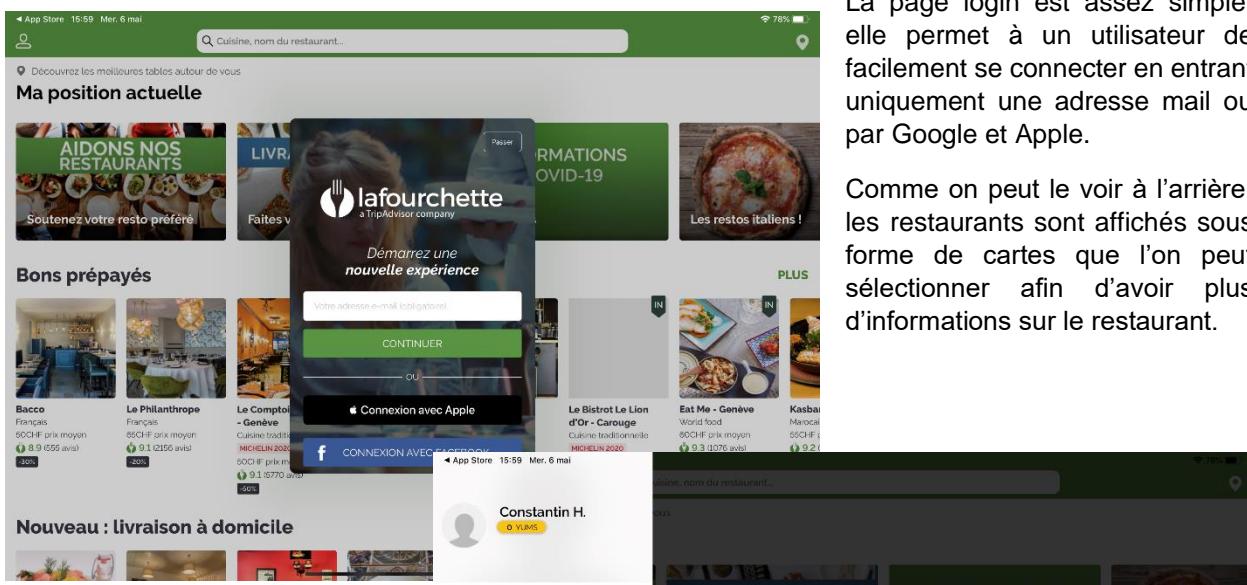


Figure 2 Page d'accueil

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

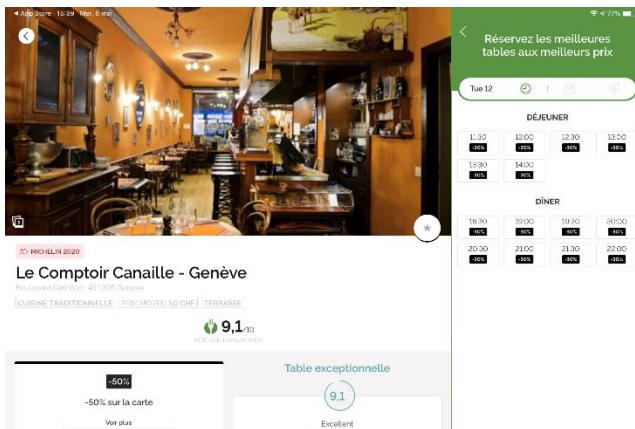
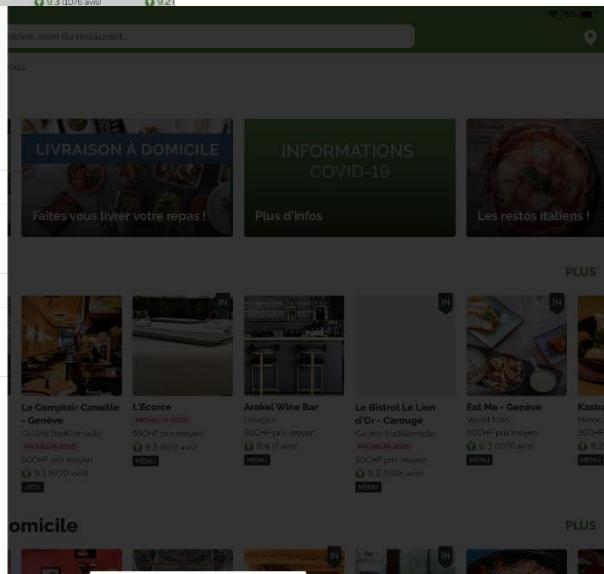


Figure 6 Page du restaurant

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.

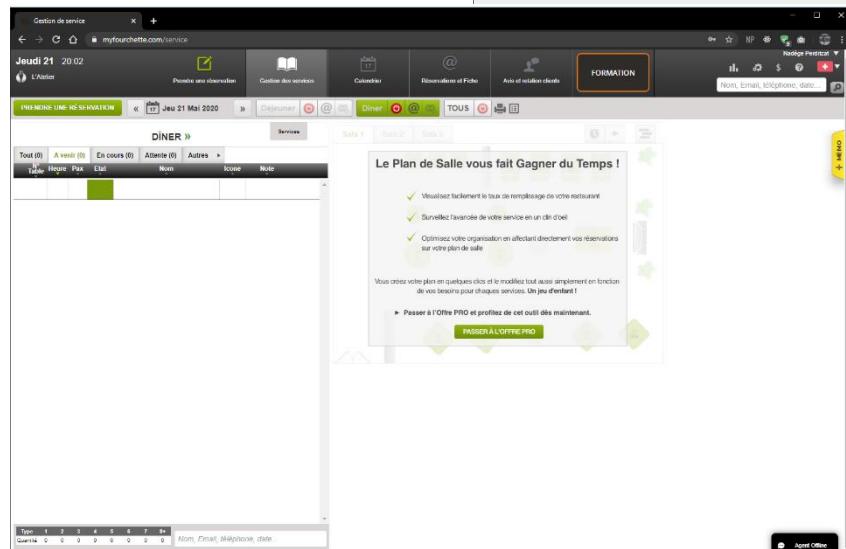
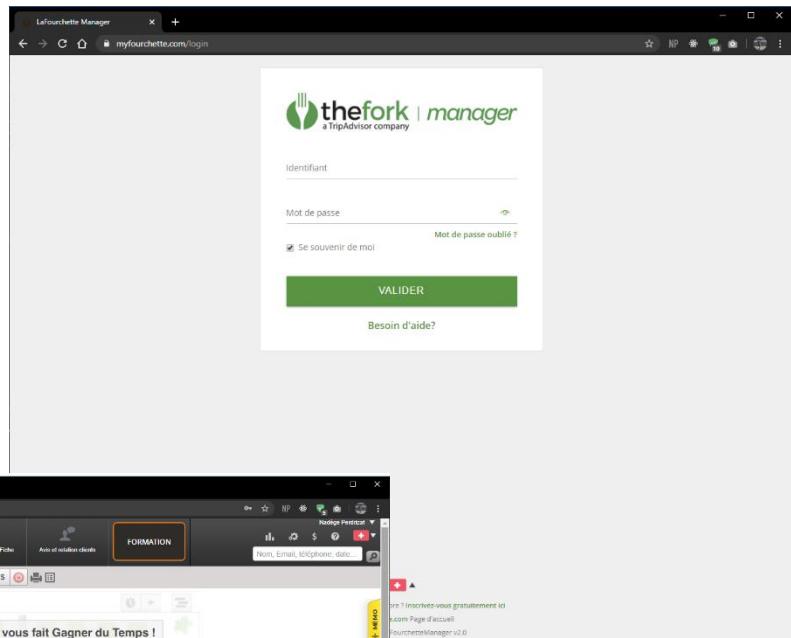


Figure 8 Page d'accueil

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 7 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/>

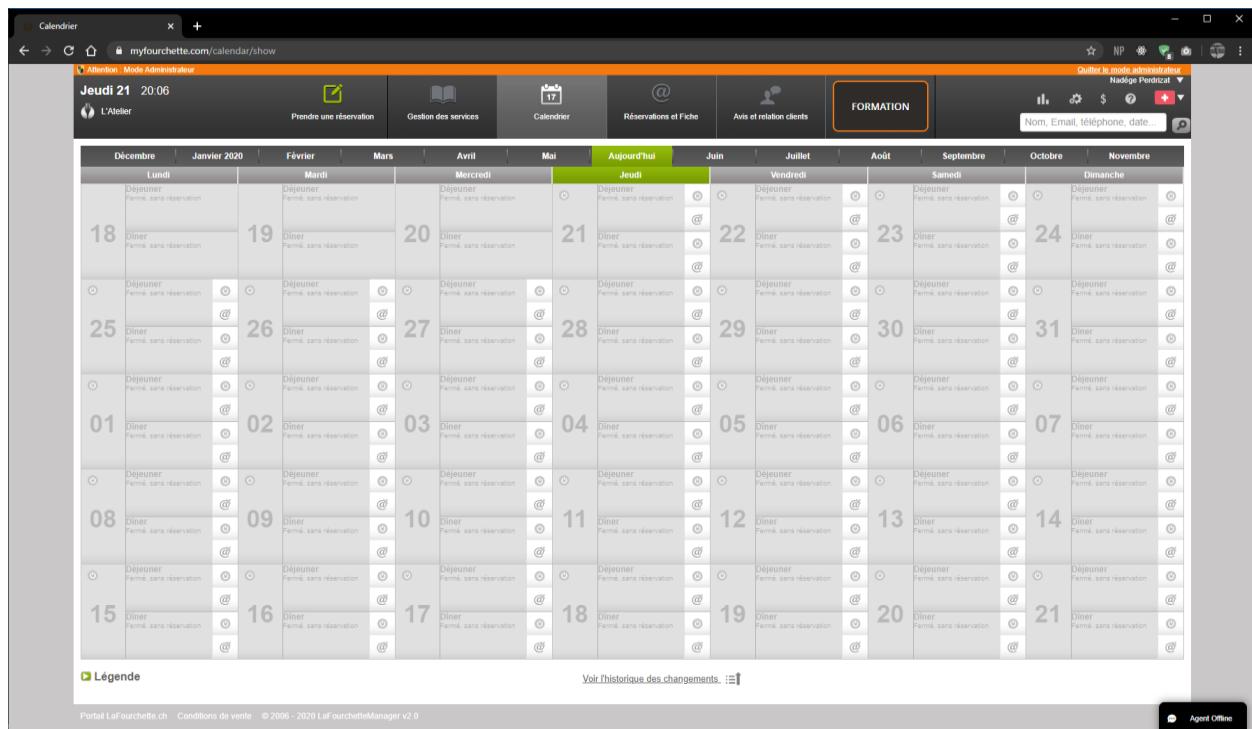


Figure 9 calendrier du manager

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.

Figure 11 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 10 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 14 Formulaire de création de réservation

Figure 12 Sélection heure réservation manager

Figure 13 Sélection date réservation manager

6 Cahier des charges



REPUBLIQUE ET CANTON DE GENEVE
 DEPARTEMENT DE L'INSTRUCTION PUBLIQUE, DE LA CULTURE ET DU SPORT
 ENSEIGNEMENT SECONDAIRE II POSTOBLIGATOIRE
Centre de formation professionnelle technique
Ecole d'informatique

Cahier des charges ES 2020

Données candidat	Données enseignant TPI
Nom : Herrmann Prénom : Constantin Téléphone (portable) : 079 882 89 25 E- Mail : constantin.hrrmn@eduge.ch	Nom : Prénom : E- Mail :
Titre du projet	
RESA	
Objectifs du projet	
<p>« Resa » est une application qui sera utilisée par les restaurateurs. Il permet de gérer le plan des tables et la gestion des réservations. La mandante de ce projet est Mlle Perdrizat Nadège. Mlle Perdrizat est la Manager de l'établissement l'Atelier Lounge Geneva aux Accacias.</p>	
Description détaillée	
<p>Description de l'application</p> <p>Actuellement certains restaurateurs utilisent des outils pour les réservations dans leurs établissements. Les outils ne sont pas appropriés à leurs besoins. Le but de cette application est de remplacer les outils existants de réservations.</p> <p>Mme. Perdrizat souhaite pouvoir visuellement créer un établissement en y ajoutant des zones, des tables, un bar, etc. Un serveur peut prendre une réservation et une table sera automatiquement attribuer à cette réservation.</p> <p>Établissement</p> <p>C'est l'ensemble des locaux accueillant les services de restauration. Dans le local, il y a des zones. À ces zones, on attribue le nombre de clients et le type de service. Ces zones peuvent avoir une disposition géographique dans l'établissement. L'application peut être utilisées avec ou sans le plan dessiné par le restaurateur.</p>	

L'utilisateur de l'application, peut dessiner : l'établissement, les salles, placer des tables, des meubles, dispositions différentes selon horaires. Il y aura donc une page de paramétrage du restaurant ou l'utilisateur authentifié pourra à l'aide de « drag and drop » créer/dessiner son restaurant.

L'utilisateur de l'application, peut configurer l'établissement : Nombre clients par table, Nombre de clients par zone

Mme. Perdrizat à quelques besoins supplémentaires :

- L'établissement peut avoir un ou plusieurs étages
- Chaque étage peut avoir une ou plusieurs salles
- Chaque salle a une ou des zones différentes
- Chaque zone est définie selon des dimensions géométriques
- A chaque zone, on peut attribuer un horaire d'ouverture
- A chaque zone, on peut attribuer un type de service : Total : Repas et Boissons, Boissons, Bar

Chaque établissement est identifiable par son nom, son adresse et son téléphone. Chaque établissement possède un manager. Des heures d'ouvertures (par zone dans le restaurant aussi), des médias, des plats du jour, une carte, des événements et des liens.

Les établissements sont créables avec des meubles qui contiennent toutes les infos comme la forme, est-ce que c'est un bar ou une table, haute ou basse.

Liste des fonctionnalités clés

- Créer un restaurant (Sous forme visuelle ou non)
- Ajouter des menus
- Ajouter la carte
- Ajouter le plat du jour
- Création d'une réservation
- Modification d'une réservation
- Mettre à jour une réservation quand le client est dans le restaurant en changeant l'état de la réservation (cf filtre de réservation)
- Annuler une réservation
- Modifier les horaires, les menus et la carte du restaurant

Réservations

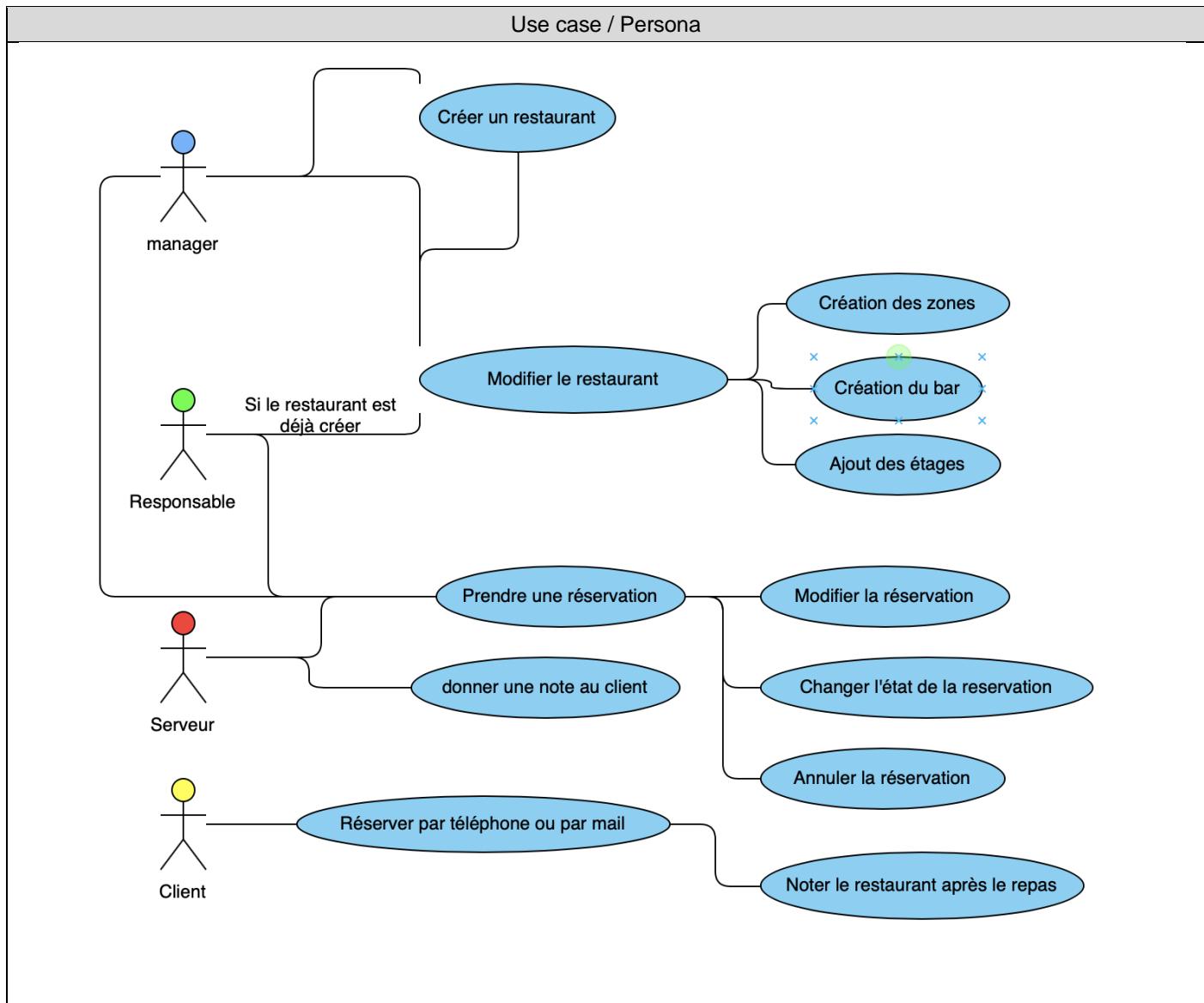
Une fois que les zones sont définies (dessinées ou non), on peut assigner une réservation.

On peut rechercher une réservation ou des réservations par des filtres :

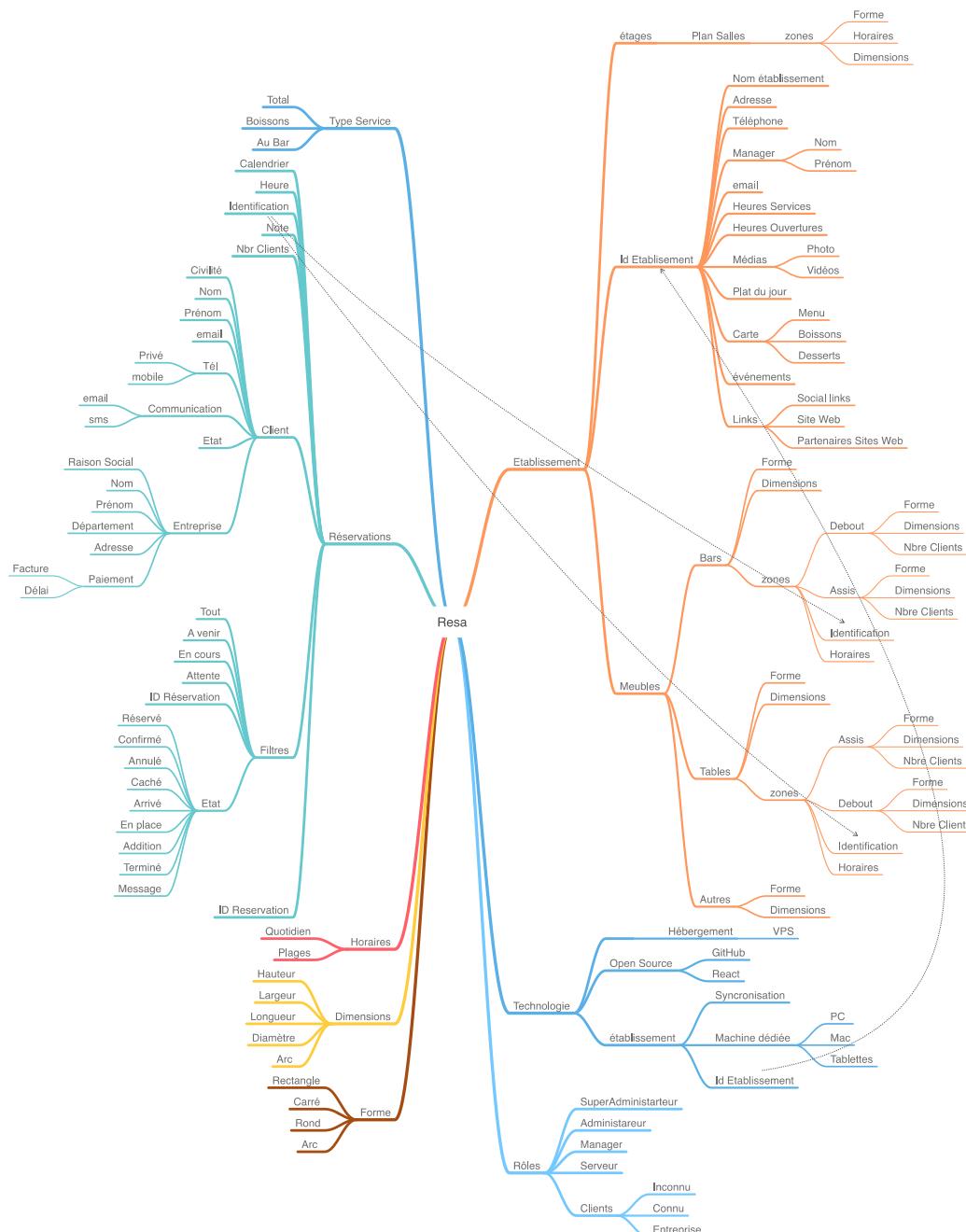
Toutes, À venir, En cours, Attente

Suppléments

Des rendez-vous avec la mandante Mme. Perdrizat sont prévu dans les semaines à venir. Le cahier des charges est susceptible de changer.



Maquettes / MinMap



SWOT	
FORCES	FAIBLESSES
<ul style="list-style-type: none"> - Donner une alternative gratuite pour les restaurants - Restaurant déjà intéressé par une alternative à « la fourchette verte » 	<ul style="list-style-type: none"> - Peu de temps pour prendre en main correctement les technologies
OPPORTUNITES	MENACES
<ul style="list-style-type: none"> - Avoir une application sérieuse qui peut intéresser beaucoup de restaurants - Découvrir de nouvelles technologies pour le dessin sur une page web 	<ul style="list-style-type: none"> - Les restaurants préfèrent garder « la fourchette verte » - Ne pas terminer le projet dans le temps imparti du travail de diplôme

Planning

Inventaire du matériel

Inventaire des logiciels

Afin de pouvoir dessiner des objets sur le site, je pense utiliser ou m'inspirer de « react-designer » :
<https://github.com/react-designer/react-designer>

Délivrables (documents à restituer)

- 1 carnet de bord
- 1 exemplaire papier de la documentation technique
- 1 exemplaire papier du mode d'emploi
- Mis à part le carnet de bord, tous les documents seront également restitués sur le serveur Moodle.

Signatures

Date :

Date :

Candidat :

Enseignant :

7 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.

7.1 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.

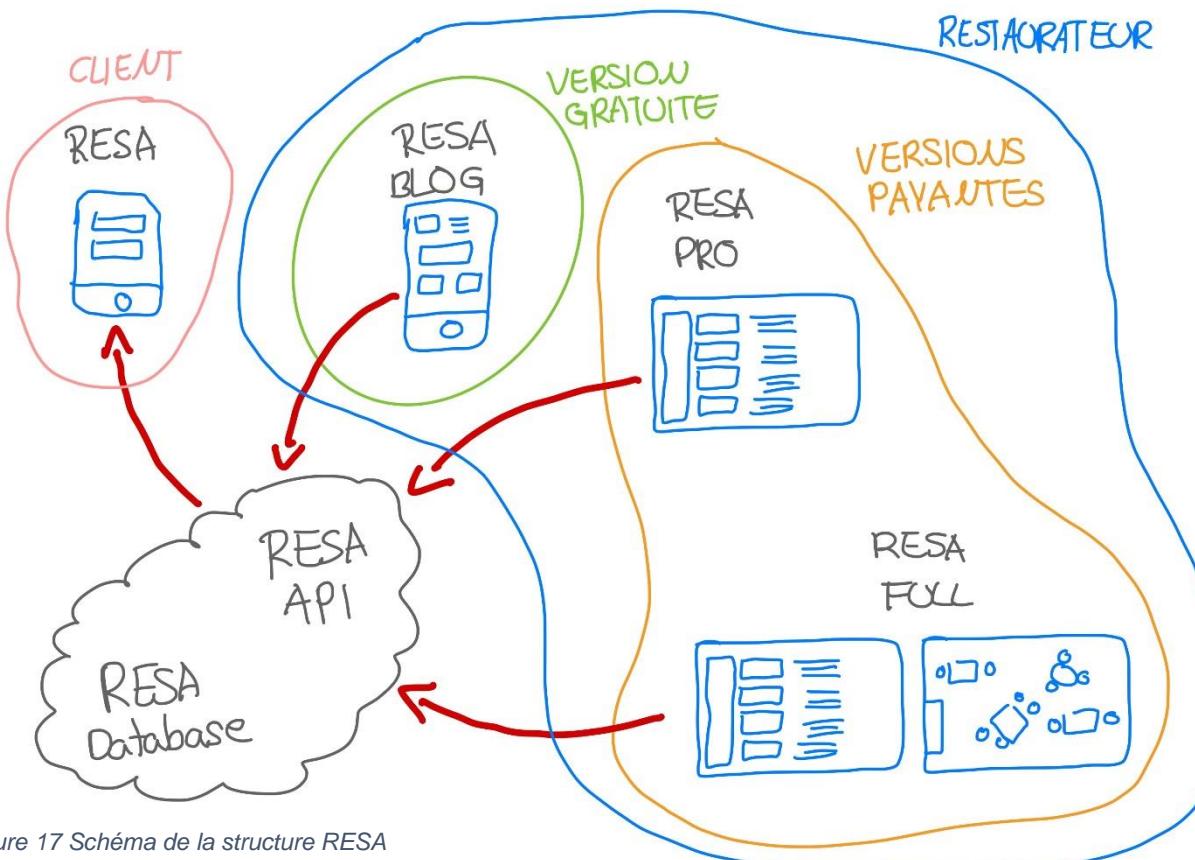


Figure 17 Schéma de la structure RESA

7.2 Interfaces

7.2.1 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 :

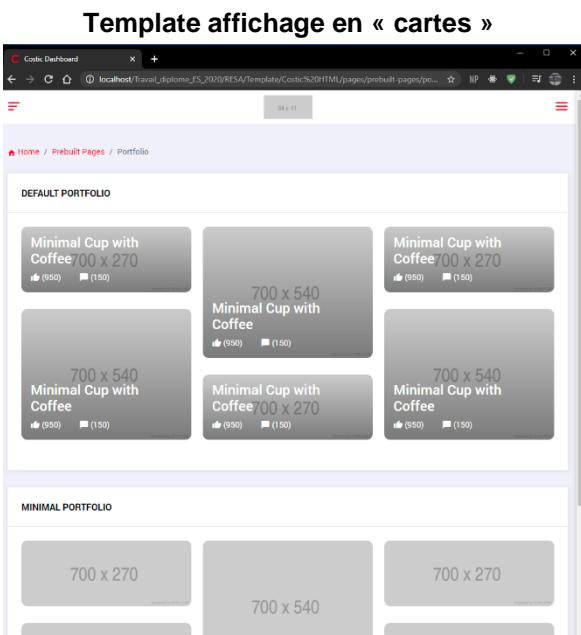


Figure 18 Capture d'écran du template

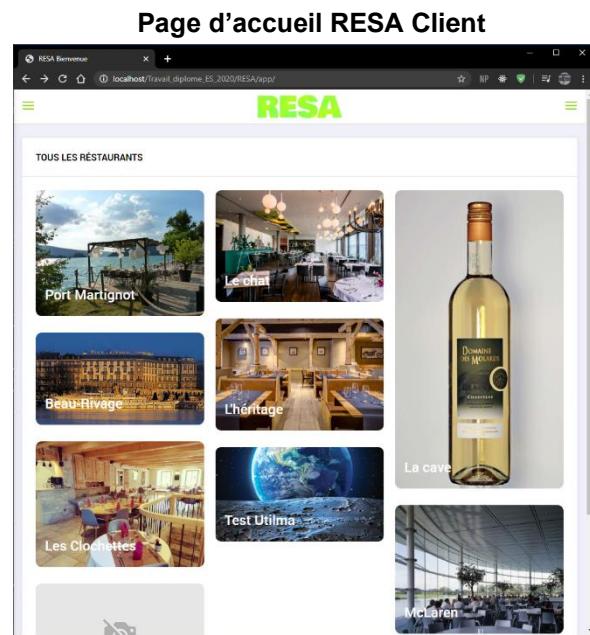


Figure 19 Capture d'écran de la page d'accueil RESA

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/>

7.2.2 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.

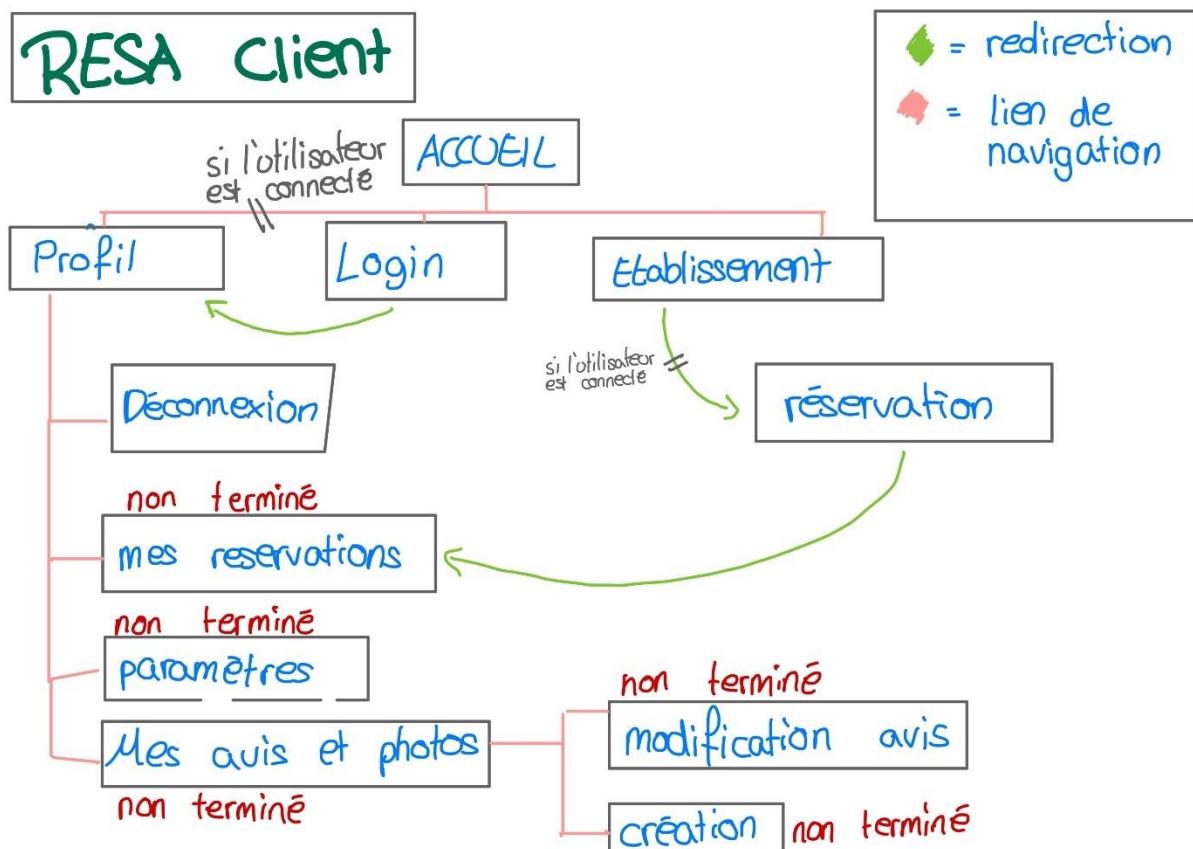
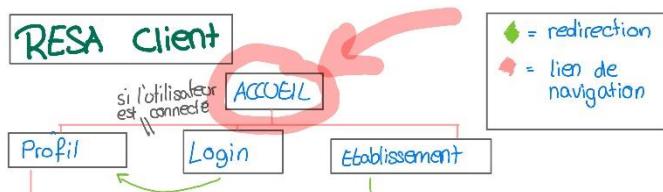


Figure 20 Schéma de navigation RESA Client

7.2.2.1 Page d'accueil



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

Figure 21 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.

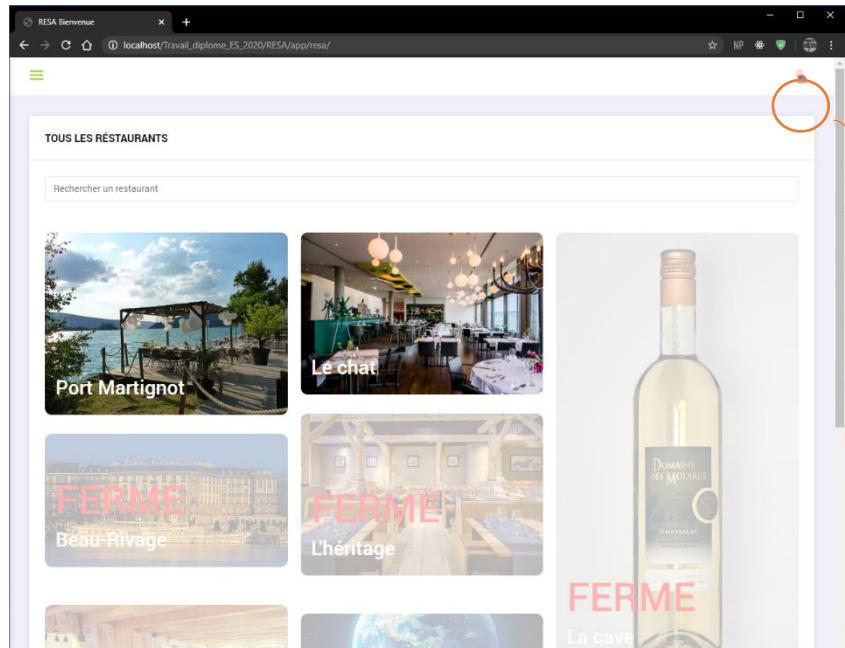


Figure 22 Page d'accueil de RESA client

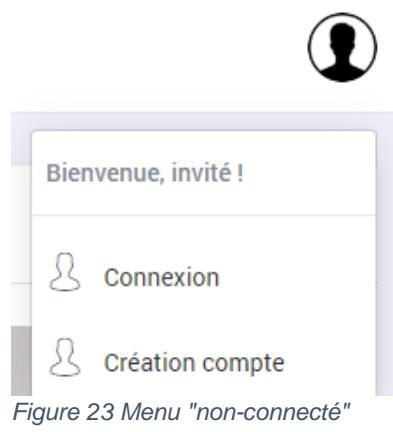


Figure 23 Menu "non-connecté"

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.

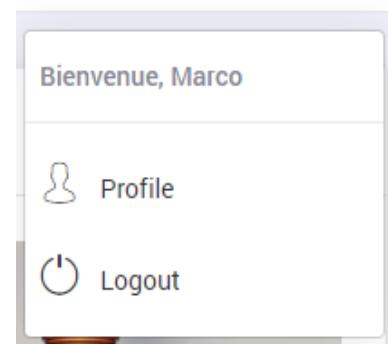


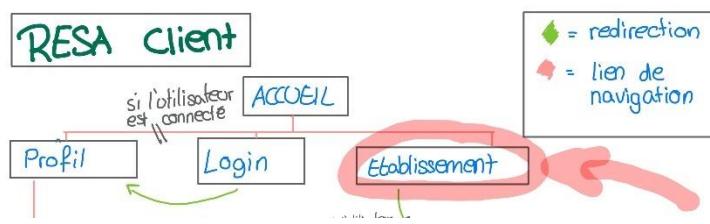
Figure 24 Menu "connecté"

7.2.2.2 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 25 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.

Sur la gauche, une capture d'écran de la page de l'établissement "Port Martignot". Elle affiche une photo extérieure avec des ballons blancs, le nom "Port Martignot", le numéro de téléphone "Phone : 022 354 75 34" et l'email "Email : info@portmartignot.ch". En bas, il y a deux photos de l'intérieur et un formulaire de réservation avec champs pour date, heure et nombre de personnes, suivis d'un bouton "Réserver".

Sur la droite, deux captures d'écran montrant les deux types d'affichage :

- Le chat :** Affiche une photo intérieure d'un restaurant élégant, le nom "Le chat", le numéro de téléphone "Phone : 022 463 73 82" et l'email "Email : info@chat.ch". Il y a aussi un bouton "Appeler" et "Mail".
- Le deuxième affichage :** Remplace les boutons par un formulaire de réservation. L'utilisateur choisit la date (06/09/2020), l'heure (12:15) et le nombre de personnes (4). Le bouton est "Réserver".

Une fois la demande envoyée, RESA retourna un message afin de valider la réservation ou le problème s'il y en a.

Validé

Votre réservation pour le 2020-09-06 à 12:15 pour 4 pers. à bien été prise en compte

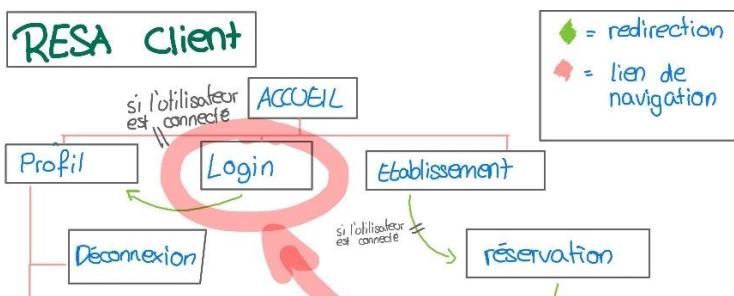
SUPER !

Pas de place !

Malheureusement le restaurant n'a pas de place disponible pour l'horaire choisi...

Je comprend

7.2.2.3 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 26 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.

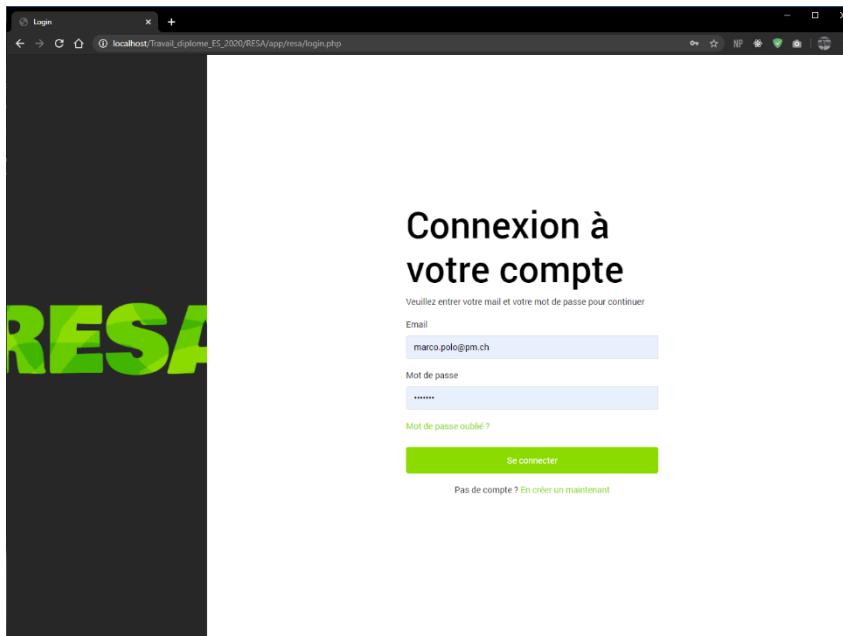
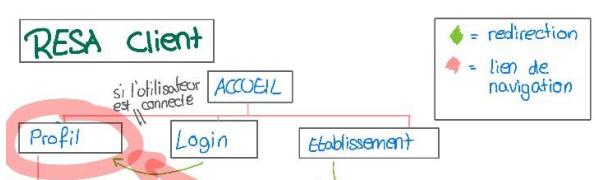


Figure 27 Login d'un utilisateur

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.

7.2.2.4 Profil utilisateur



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

Figure 28 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.

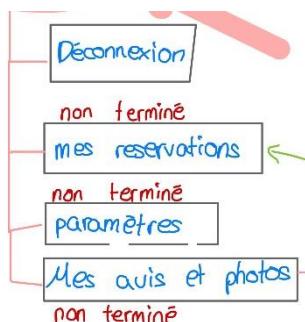


Figure 29 Navigation possibles depuis page de profil

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

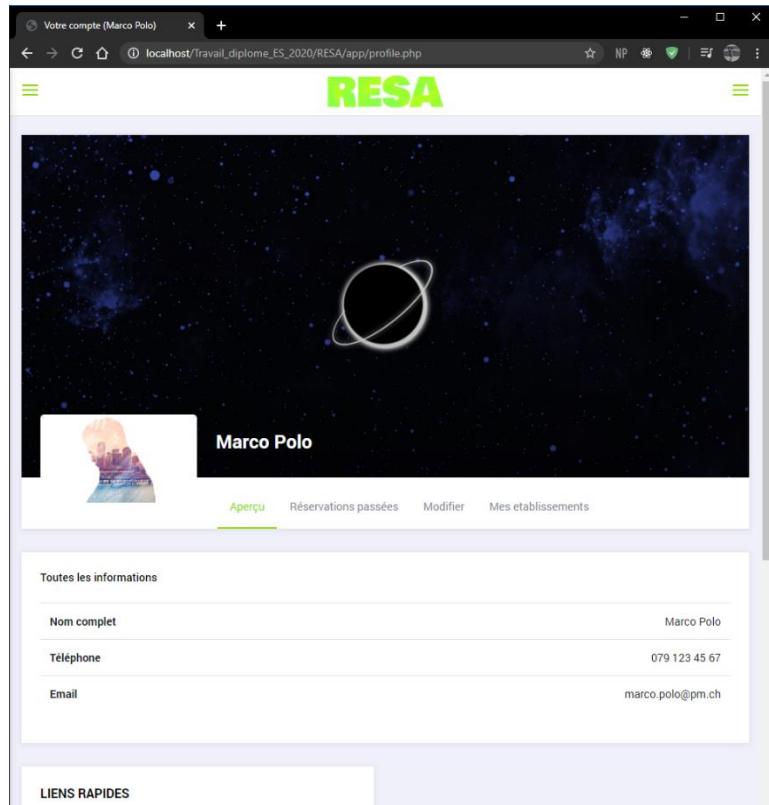


Figure 30 Page de profil

Le modèle de réservations est divisé en deux sections : "A venir" et "Passées".

- A venir:** Affiche deux réservations : "L'Atelier" le 20/08/20 à 19h30 et "Beau-rivage" le 23/08/20 à 12h30. Chaque réservation a un bouton "SUPPRIMER" et un bouton "modifier".
- Passées:** Affiche quatre réservations avec leurs notes et commentaires. Les réservations sont : "L'atelier" (20/06/20), "Port Marignot" (19/06/20), "Tesla" (18/06/20) et "Beau-rivage" (16/06/20). Chaque réservation a un bouton "COMMENTAIRE", un bouton "NOTE" et un bouton "SUPPRIMER".

La page de réservations affiche les réservations à venir ainsi que celles déjà passées.

L'utilisateur peut mettre une note ainsi qu'un commentaire. Il peut également modifier ces derniers.

L'utilisateur peut supprimer ou modifier les réservations à venir.

7.2.3 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 with the URL `localhost/Travail_diplome_ES_2020/RESA/app/manager/`. The page has a dark background with the word "RESA" in large green letters on the left. The main content area is titled "Connexion au restaurant". It contains three input fields: "Nom du Restaurant" with the value "Port Martignot", "Email" with the value "marco.polo@pm.ch", and "Mot de passe" with a masked value. Below these fields is a green "Se connecter" button. At the bottom of the form, there is a link "Pas de compte ? En créer un maintenant".

Figure 31 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.

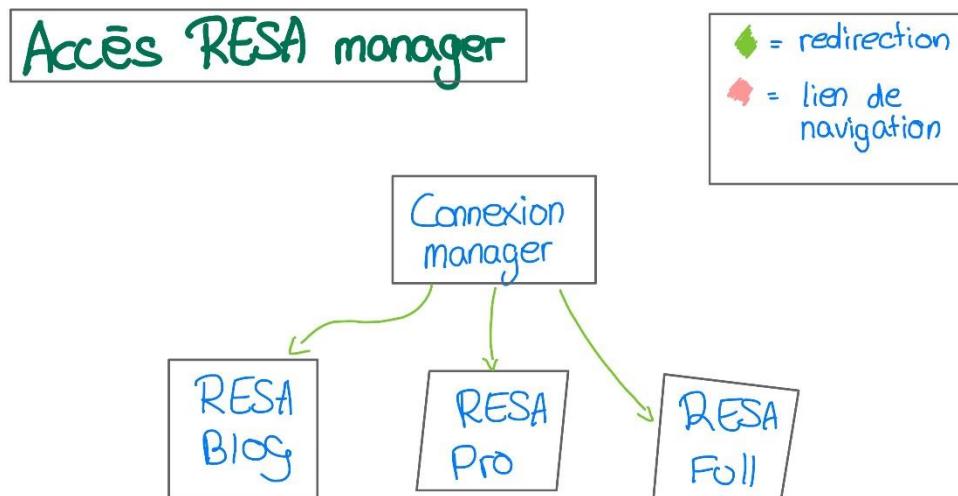


Figure 32 Schéma de navigation RESA manager

7.2.3.1 Création d'un établissement

L'utilisateur peut créer un nouvel établissement en sélectionnant le lien «En créer un maintenant».

Pas de restaurant ?[En créer un maintenant](#)

Figure 33 Lien création établissement

L'utilisateur devra alors se connecter (afin de nommer manager de l'établissement) ou alors créer un compte s'il n'en possède pas.

Ensuite, il devra choisir parmi les trois abonnements proposés par RESA.



Figure 34 Choix abonnement

L'utilisateur sélectionne l'abonnement. En choisissant un abonnement payant, l'utilisateur est redirigé sur la page de paiement.

Cette page résume l'abonnement sélectionné et affiche un formulaire pour entrer les données de la carte de crédit.

Dans le cadre de mon projet, je n'ai pas fait de vérification bancaire, c'est une fonctionnalité que j'aurais ajoutée en cas de prolongation du projet.

Un message affiche le bon ou mauvais fonctionnement de la transaction.

Paiement

Veuillez entrer vos coordonnées afin de continuer

Abonnement
PRO
9 CHF / mois
Titulaire de la carte
Numéro de carte
CVC
1
2020

[Valider](#)

SUPER !

Vous pouvez maintenant créer votre établissement

[C'est parti !](#)

Mince...

Il doit y avoir une erreur dans les données saisies

[je recommence](#)

Figure 35 Page de paiement d'abonnement

Après avoir payé, ou en ayant choisi l'abonnement gratuit, l'utilisateur doit renseigner les données de son établissement. Il peut au même moment ajouter des photos.

CRÉATION DE L'ÉTABLISSEMENT

Nom de l'établissement			
Nom de l'établissement			
Email de contact	Téléphone		
info@example.ch	0221234567		
Rue	NPA	City	Pays
Chemin de l'exemple 12	1206	Genève	Afghanistan
Images			
Sélection. fichiers	Aucun fichier choisi		
Créer			

Figure 36 Formulaire de création

Une fois les données entrées et validées, RESA redirige l'utilisateur sur l'application correspondante à l'abonnement sélectionné.

7.2.4RESA 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.

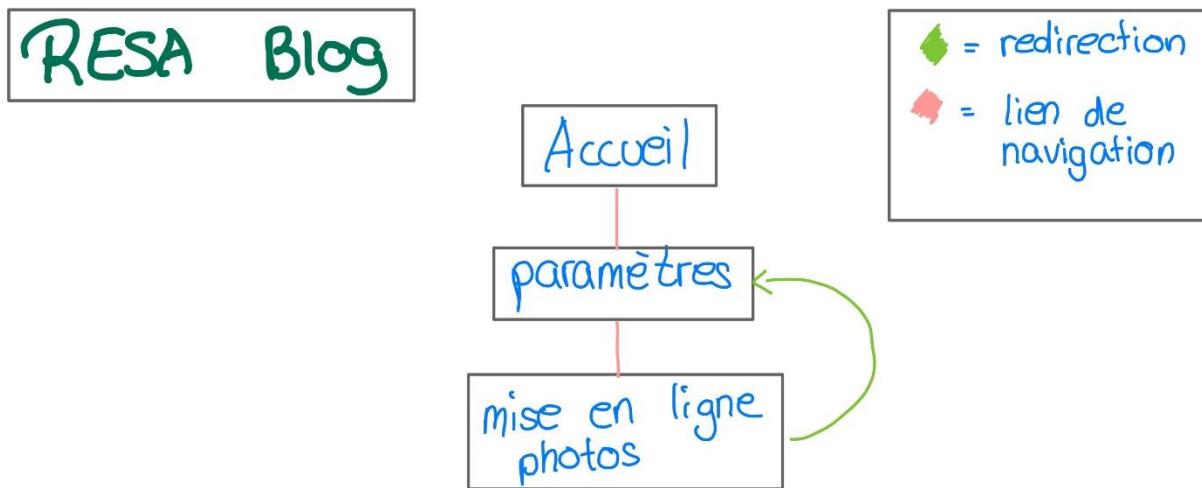


Figure 37 Schéma de navigation RESA Blog

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

7.2.4.1 Accueil

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

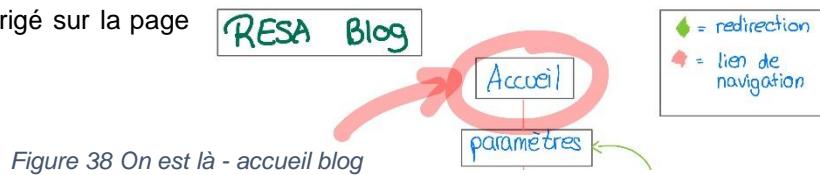


Figure 38 On est là - accueil blog

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

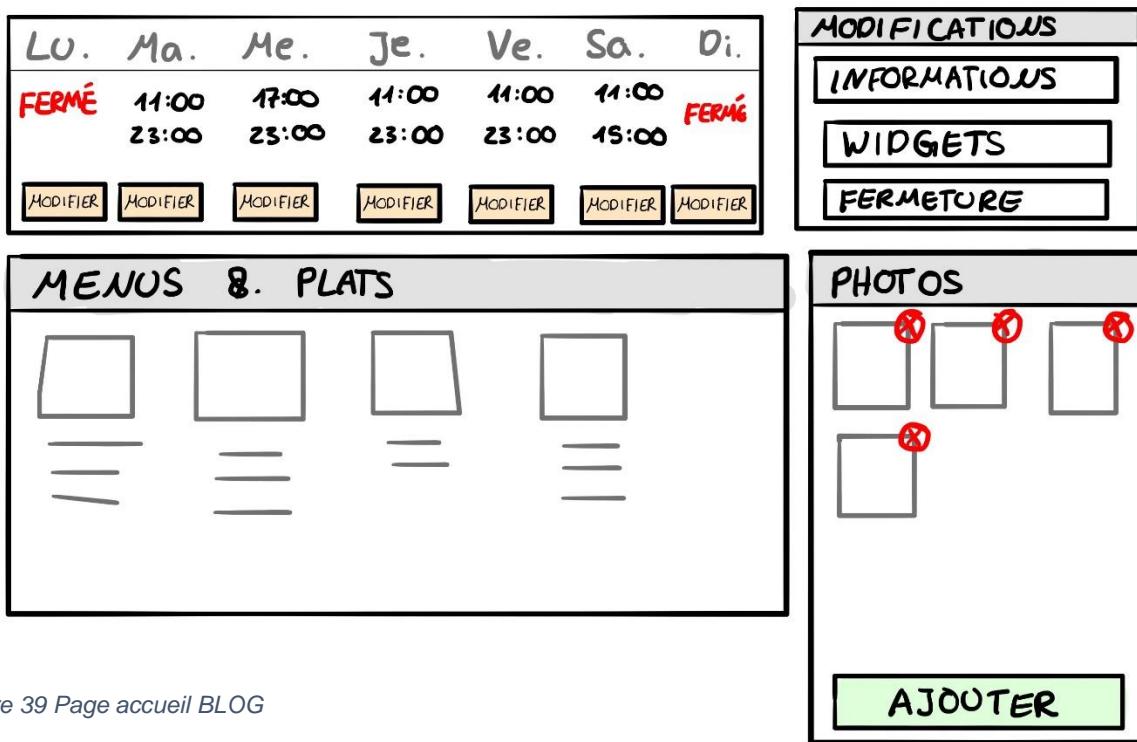


Figure 39 Page accueil BLOG

7.2.4.2 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
- Le menus et les plats
- Le nombre de places du restaurant

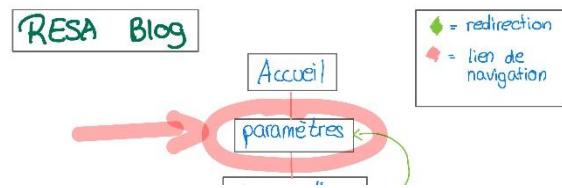


Figure 40 On est là - paramètres blog

7.2.4.3 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.

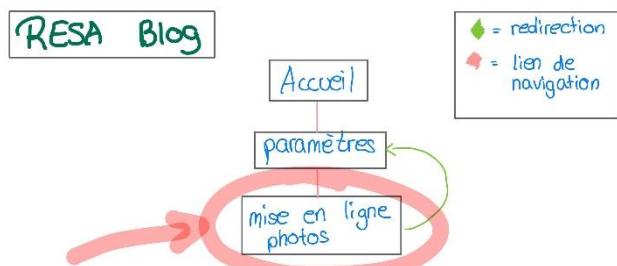


Figure 41 On est là - mise en ligne image blog

7.2.5 RESA Pro

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

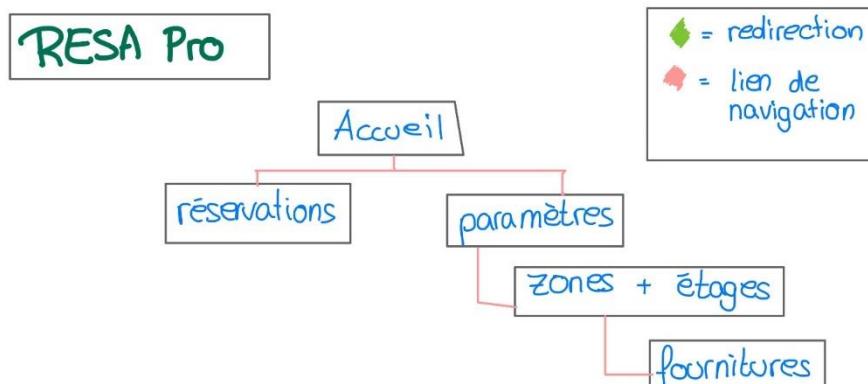


Figure 42 Schéma de navigation RESA Pro

En plus des pages accessibles via RESA Blog, RESA Pro permet la gestion des réservations, des clients ainsi que des commentaires laissés par ces derniers.

7.2.5.1 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 accéder aux réservations et aux fiches clients ainsi qu'aux commentaires laissés par ces derniers.

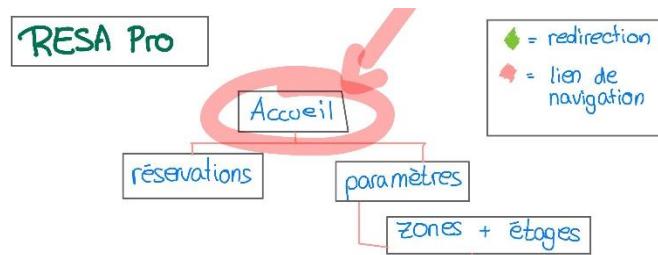
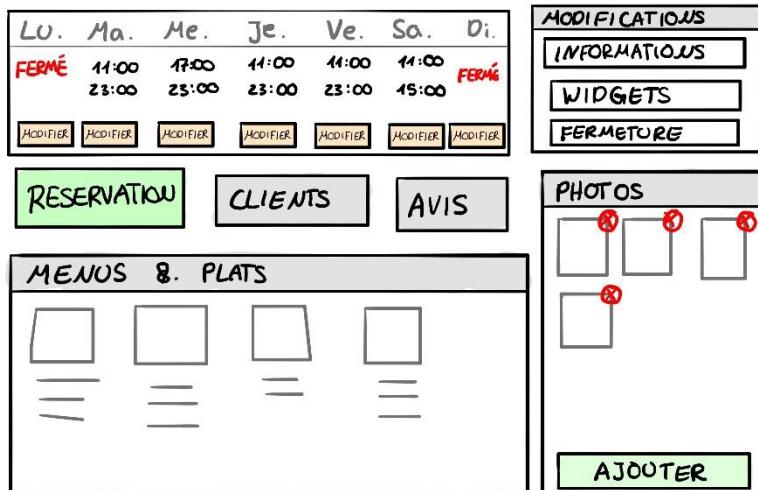


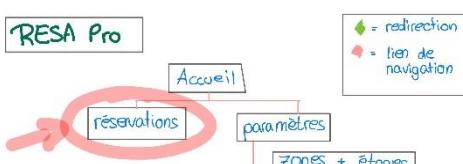
Figure 43 On est là - accueil RESA Pro



La page d'accueil reprend la même interface que RESA Blog en y ajoutant les liens vers la page des réservations, des clients et des avis.

7.2.5.2 Réservations

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



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

Figure 44 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.

LO	MA	ME	JE	VE	SA	DI
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					

RESERVATION SELECTIONNÉE

MARTIN DUBOIS [INFOS]

4 P | 17 ARRIVÉE

REMARQUE
Anniversaire de ma fille.
Mettre une bougie sur dessert.

SUPPRIMER MODIFIER

7.2.5.3 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.

RESA Pro

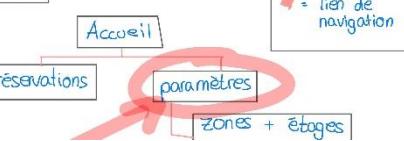


Figure 45 On est là - paramètres RESA Pro

7.2.6RESA 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.

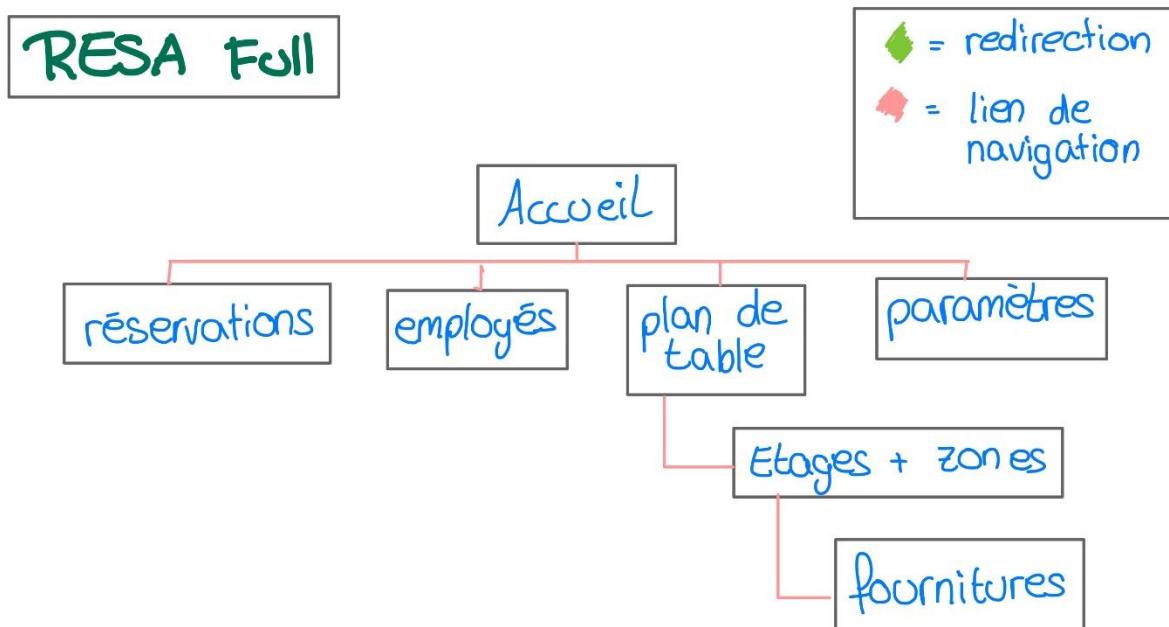


Figure 46 Schéma de navigation RESA Full

7.2.6.1 Accueil

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

Cette page affiche les mêmes informations que RESA Blog et RESA Pro en y ajoutant les liens pour le plan de table et la gestion des employés.

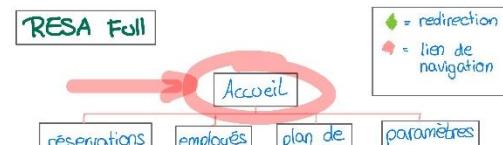
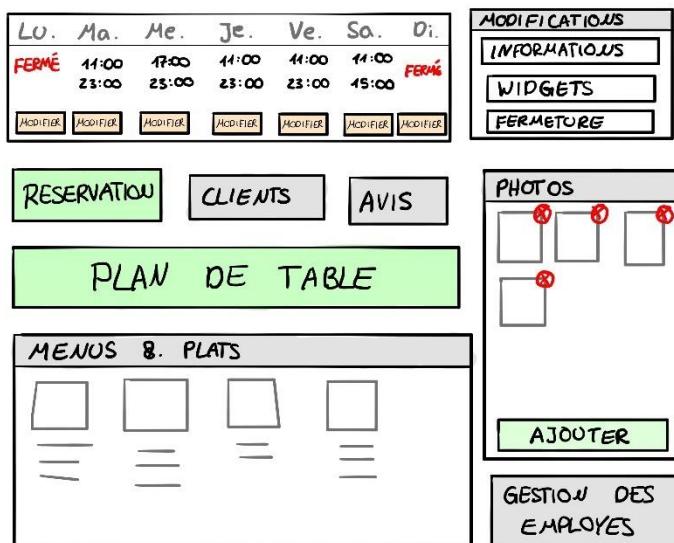


Figure 47 On est là - Accueil RESA Full

7.2.6.2 Paramètres

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

Sur la page d'administration, le manager peut visualiser tous les étages et toutes les zones de son établissement. Il créera de nouveaux étages, puis des zones dans ceux-ci. Il peut voir le nombre de places disponibles dans la zone ainsi que les horaires.

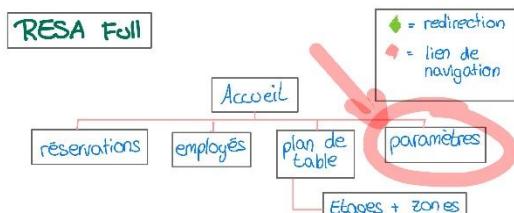


Figure 48 On est là - paramètres RESA Full

LES ZONES DE MON RESTAURANT						
Terrasse						
Vue mer	32	[Horaires]	[Fournitures]	[Modifier]		
Vue village	0	[Horaires]	[Fournitures]	[Modifier]		
Escaliers	0	[Horaires]	[Fournitures]	[Modifier]		
Vue port	0	[Horaires]	[Fournitures]	[Modifier]		
Poto	0	[Horaires]	[Fournitures]	[Modifier]		
Salut	0	[Horaires]	[Fournitures]	[Modifier]		
salon	0	[Horaires]	[Fournitures]	[Modifier]		

Ajouter une zone à Terrasse

FOURNITURES						
Nom	Couleur	Forme	Type	Places	Zone attribuée	Modifier
Curran	Bleu	Carrière	Comptoir	5	Vue mer	[modifier]
Angelica	Bleu	Ronde	Assise	6	Vue mer	[modifier]
Celeste	Bleu	Carrière	Basse	8	Vue mer	[modifier]
May	Bleu	Ovale	Basse	1	Vue mer	[modifier]
Sybil	Bleu	Carrière	Basse	2	Vue mer	[modifier]
Whitney	Bleu	Diférente	Assise	9	Piste de danse	[modifier]
Eaton	Bleu	Diférente	Haute	7	Piste de danse	[modifier]
Odysseus	Bleu	Ovale	Assise	9	Kernozet	[modifier]
Vemon	Bleu	Diférente	Assise	8	Kernozet	[modifier]

Figure 49 Page de paramètres

Le manager a accès à toutes les réservations de la semaine afin de pouvoir les consulter, les modifier ou les supprimer.

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

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.

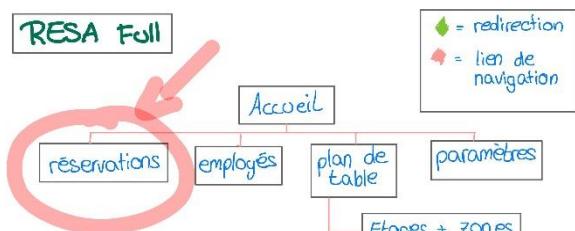
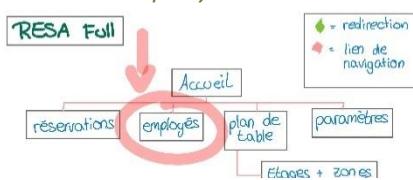


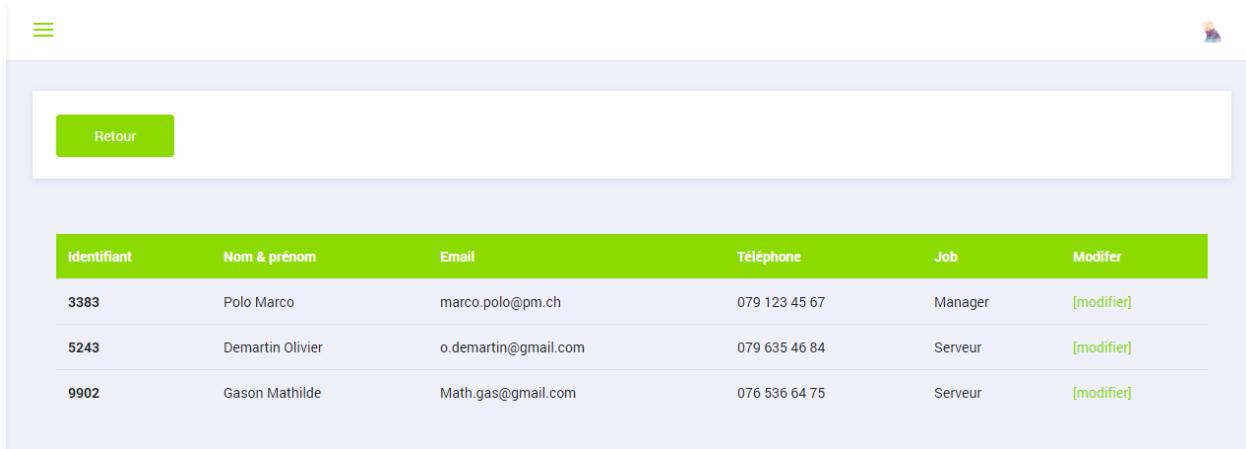
Figure 51 On est là - réservations RESA Full

7.2.6.4 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 52 On est là - Employés RESA Full



The screenshot shows a mobile application interface for managing employees. At the top left is a menu icon (three horizontal lines) and at the top right is a user profile icon. A green button labeled "Retour" is located in the top-left corner of the main content area. The main content is a table listing three employees:

Identifiant	Nom & prénom	Email	Téléphone	Job	Modifier
3383	Polo Marco	marco.polo@pm.ch	079 123 45 67	Manager	[modifier]
5243	Demartin Olivier	o.demartin@gmail.com	079 635 46 84	Serveur	[modifier]
9902	Gason Mathilde	Math.gas@gmail.com	076 536 64 75	Serveur	[modifier]

Figure 53 Page des employés

Le manager peut modifier les informations de ses employés dans son établissement. L'identifiant permet aux employés de facilement se connecter dans leur établissement afin d'accéder à l'application.

7.2.6.5 Plan de table

La page du plan de table est la page la plus complète de RESA. Cette page permet à un utilisateur (manager) de modifier son plan de table de son établissement, mais elle permet également aux serveurs d'avoir un aperçu sur l'activité du restaurant en temps réel. Grâce au code couleur, les employés voient quels sont les tables disponibles et l'avancement du client dans son repas. Il peut également voir le nombre de places totales disponibles dans les différentes zones de son établissement.

En sélectionnant une table, l'utilisateur peut voir les réservations qui lui sont attribuées ainsi que celle en cours. Si la table possède une réservation, l'utilisateur peut modifier le statut de la réservation (ce qui changera le code couleur de la table).

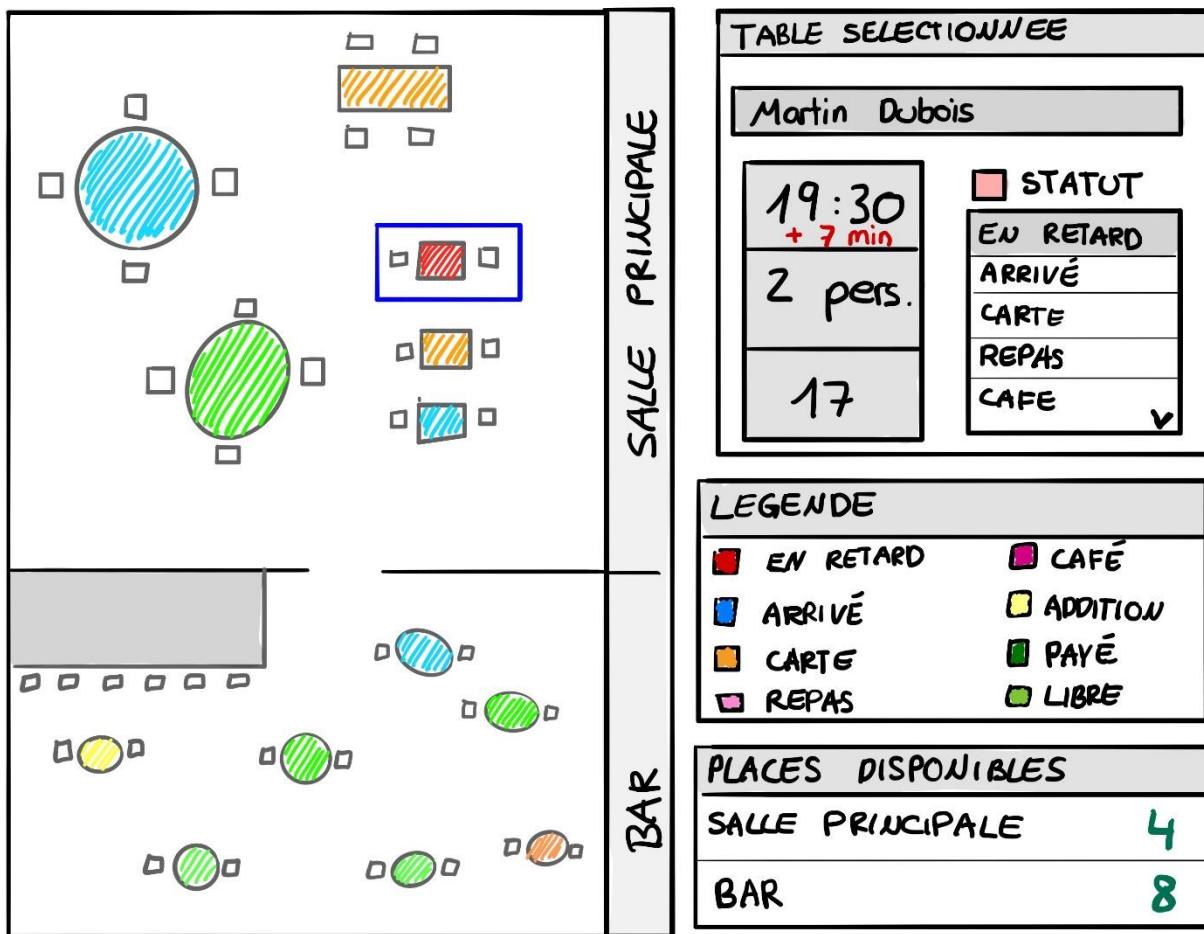


Figure 54 Page du plan de table RESA FULL

7.2.6.5.1 Code couleur des tables

LEGENDE	
■ EN RETARD	■ CAFÉ
■ ARRIVÉ	■ ADDITION
■ CARTE	■ PAYÉ
■ REPAS	■ LIBRE

Le code couleur de RESA pour les tables est le suivant.

Chaque couleur représente un état dans lequel ce trouve le client. L'orange par exemple, indique que le client est actuellement en train de lire la carte et qu'il faudra prendre sa commande bientôt.

7.2.6.5.2 Modifier le plan de table

Un manager peut modifier le plan de table directement depuis la page.

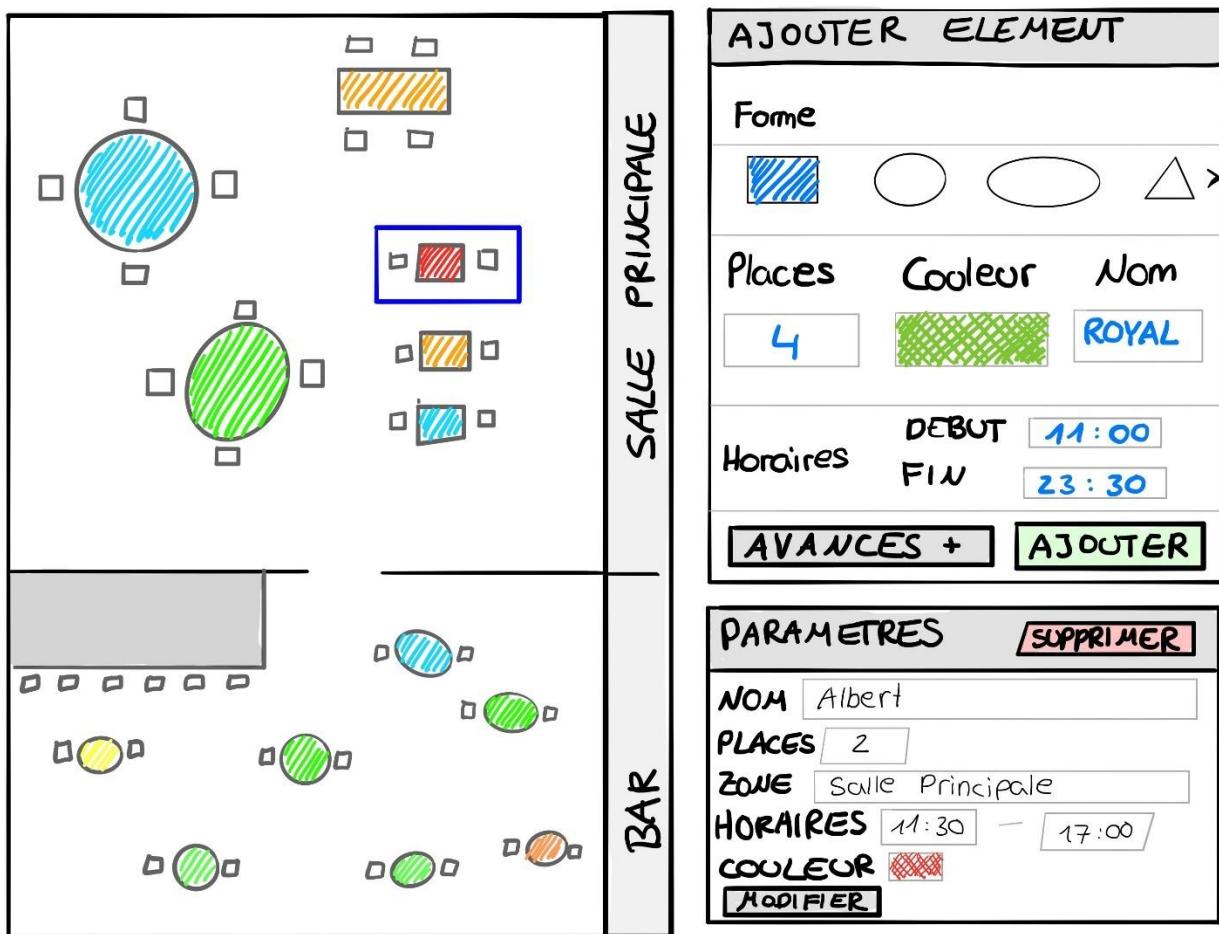


Figure 55 Page du plan de table en modifications

Le manager peut sélectionner une table ou une fourniture pour en modifier les paramètres comme son nom, son nombre de places ou ses horaires. Afin de changer la zone, le manager peut déplacer à la façon « drag and drop » sa table vers une autre zone.

Le manager peut aussi ajouter une fourniture en sélectionnant les paramètres dans le widget prévu à cet effet. Une fois créé, la fourniture apparaîtra au centre de l'établissement, le manager pourra ainsi la déplacer à l'emplacement souhaité.

7.3 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.

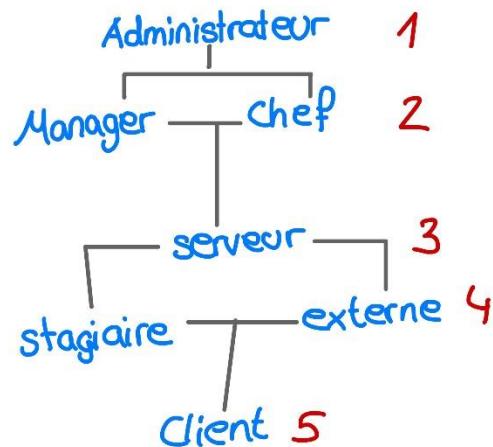


Figure 56 Hiérarchie de RESA

7.3.1 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

7.3.1.1 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.

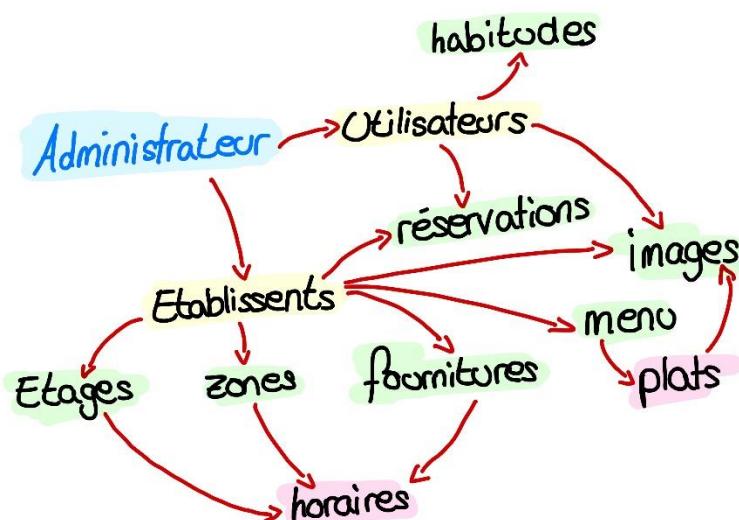


Figure 57 Schéma des droits d'un administrateur

7.3.1.2 Manager (et chef)

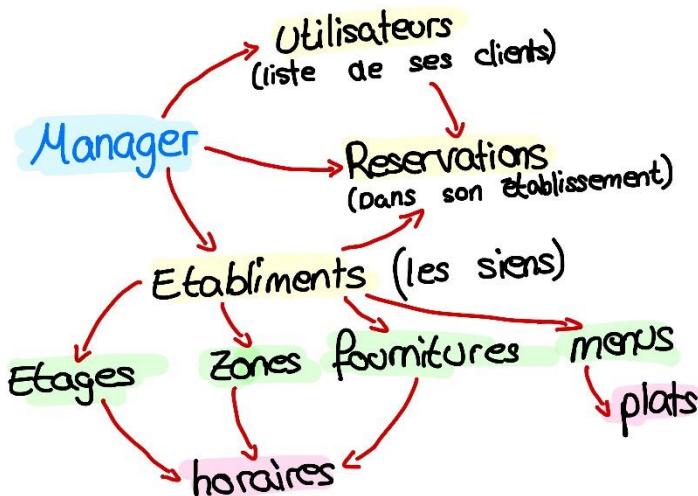


Figure 58 Schéma des droits d'un manager

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.

7.3.1.3 Serveur

Le serveur est l'utilisateur avec le plus petit rang (4) dans un établissement. Contrairement au manager, le serveur a accès aux réservations pour en créer, en modifier ou en supprimer. Le serveur peut mettre à jour les données en salle comme le statut du client ou un commentaire. Il a accès aux données du restaurant et à sa liste de clients. Il n'a pas le droit de modification sur ces dernières.

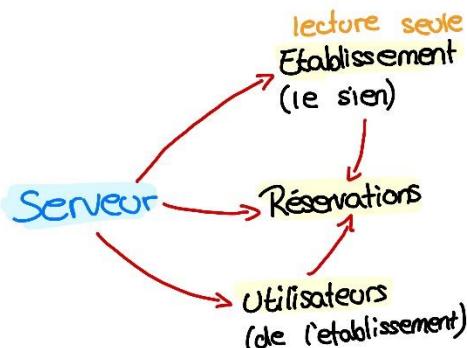


Figure 59 Schéma des droits d'un serveur

7.3.1.4 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.

7.3.1.5 Client

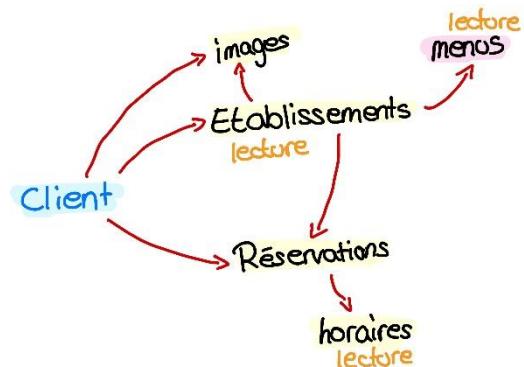


Figure 60 Schéma des droits d'un client

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.

8 Analyse organique

8.1 Mise en place

8.1.1 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.

8.1.2 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).

8.2 Prétravail

8.2.1 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** »

8.2.2 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.

8.2.3 Conventions

8.2.3.1 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.

8.2.3.2 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

8.2.3.3 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>

8.2.4 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.

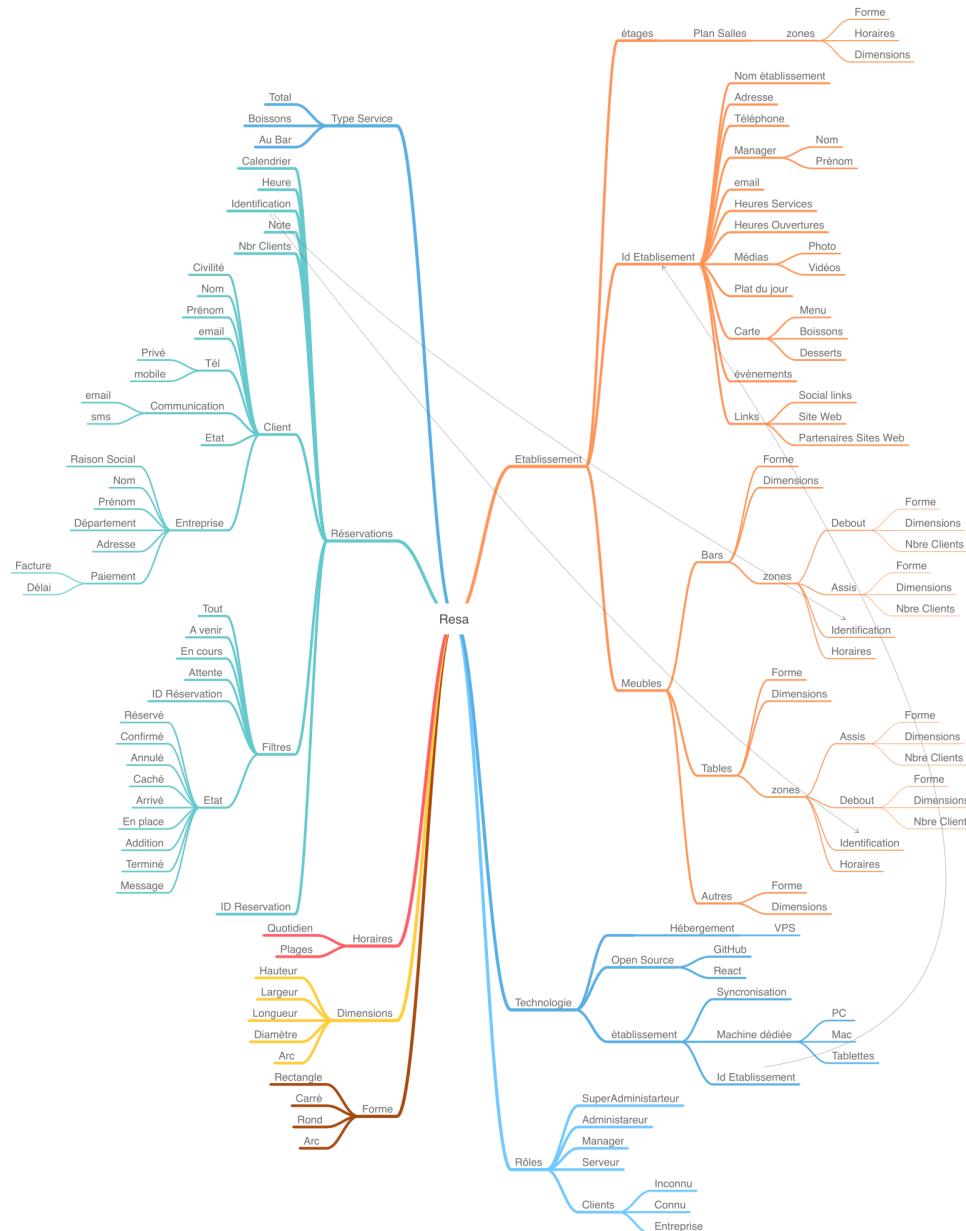


Figure 61 Mindmap de préparation

8.3 Environnement

8.3.1 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.

8.3.2 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.

8.3.3 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.

8.3.4 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.

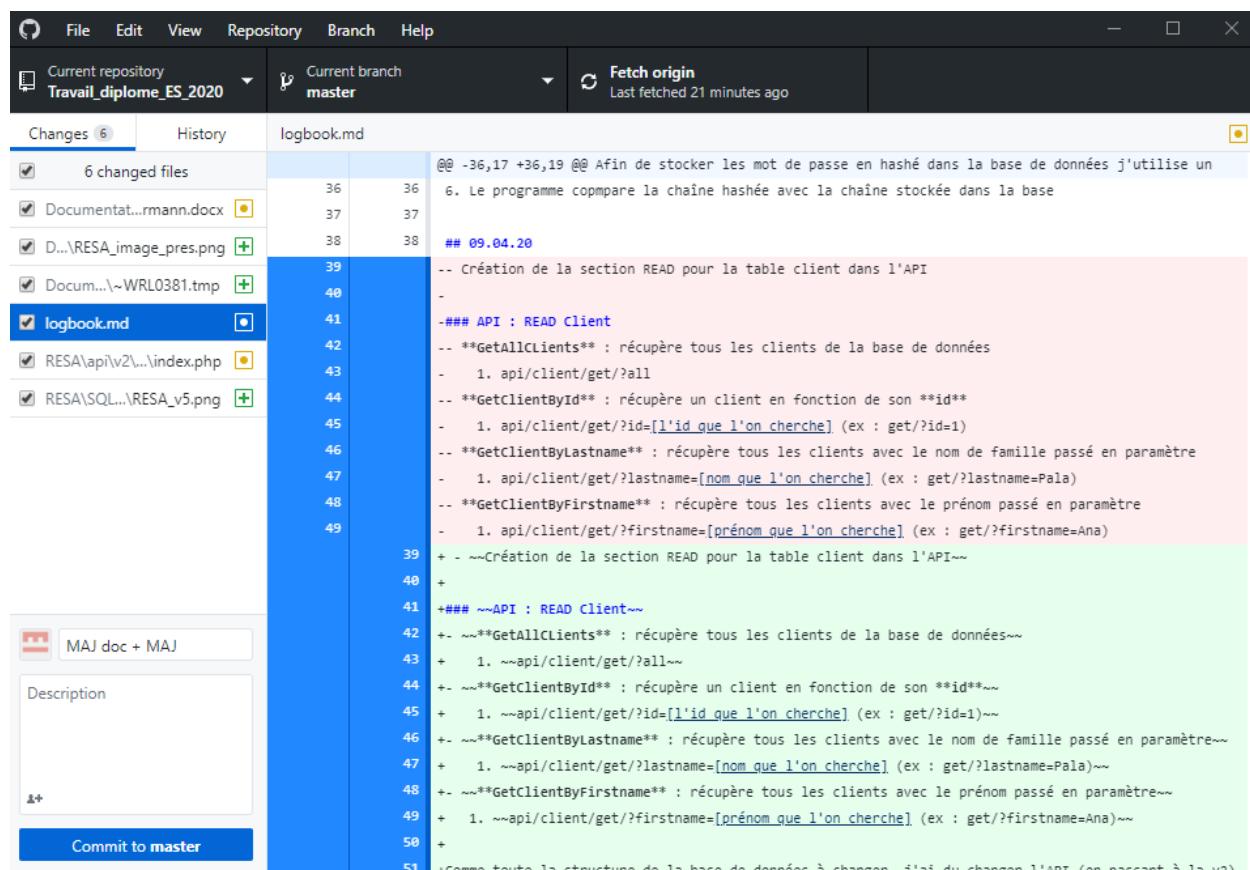


Figure 62 : Interface de Github Desktop

8.4 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.

8.4.1 RESA

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

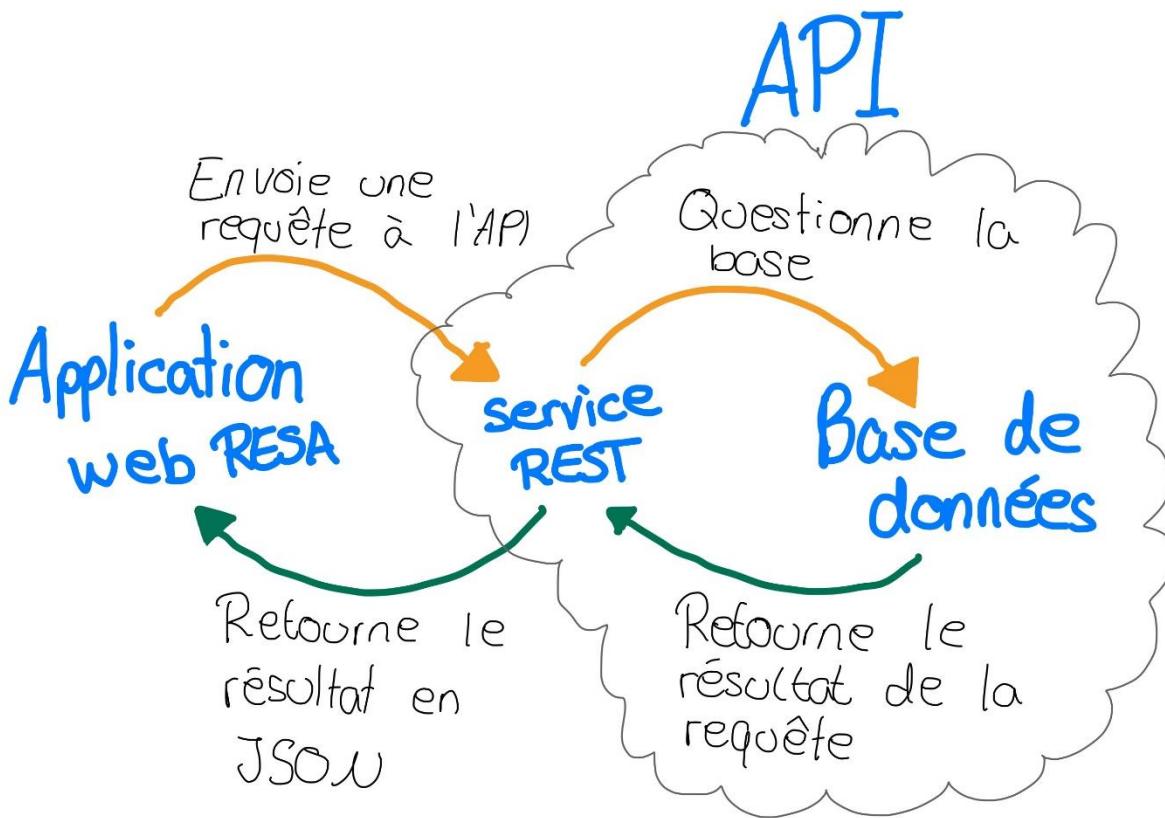


Figure 63 diagramme de fonctionnement RESA

8.5 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](#)]

8.6 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.

8.6.1 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](#)]

8.7 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.

8.7.1 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.

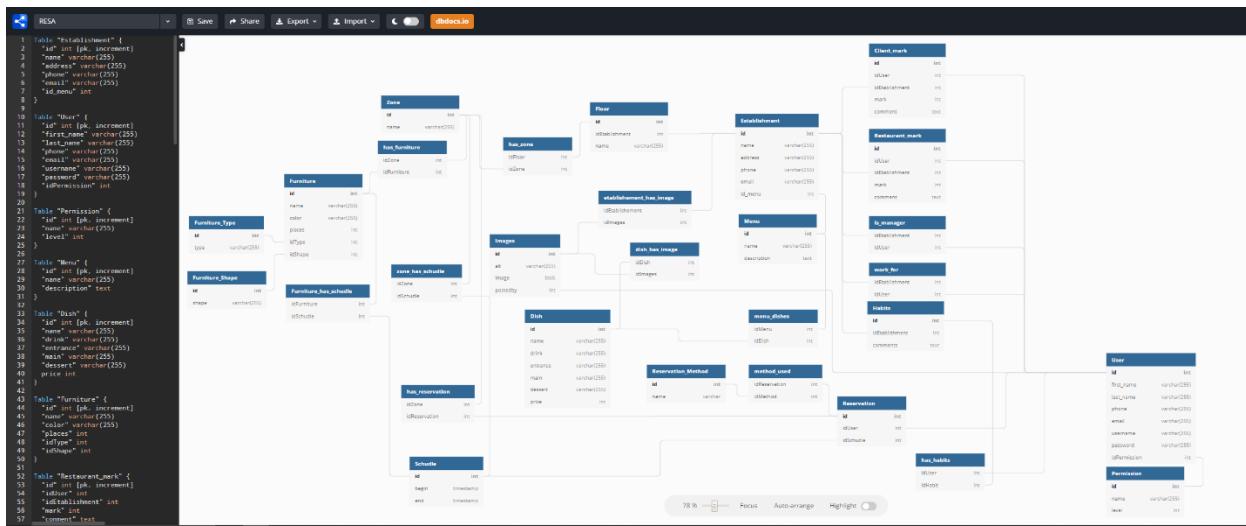


Figure 64 Aperçu interface dbdiagram.io

8.7.2 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://sourcemarking.com/uml/modeling-business-systems/external-view/activity-diagrams>

⁷ Base de données

⁸ <Https://dbdiagram.io>

8.7.3 Structure

Ma BDD est structurée de la manière suivante :

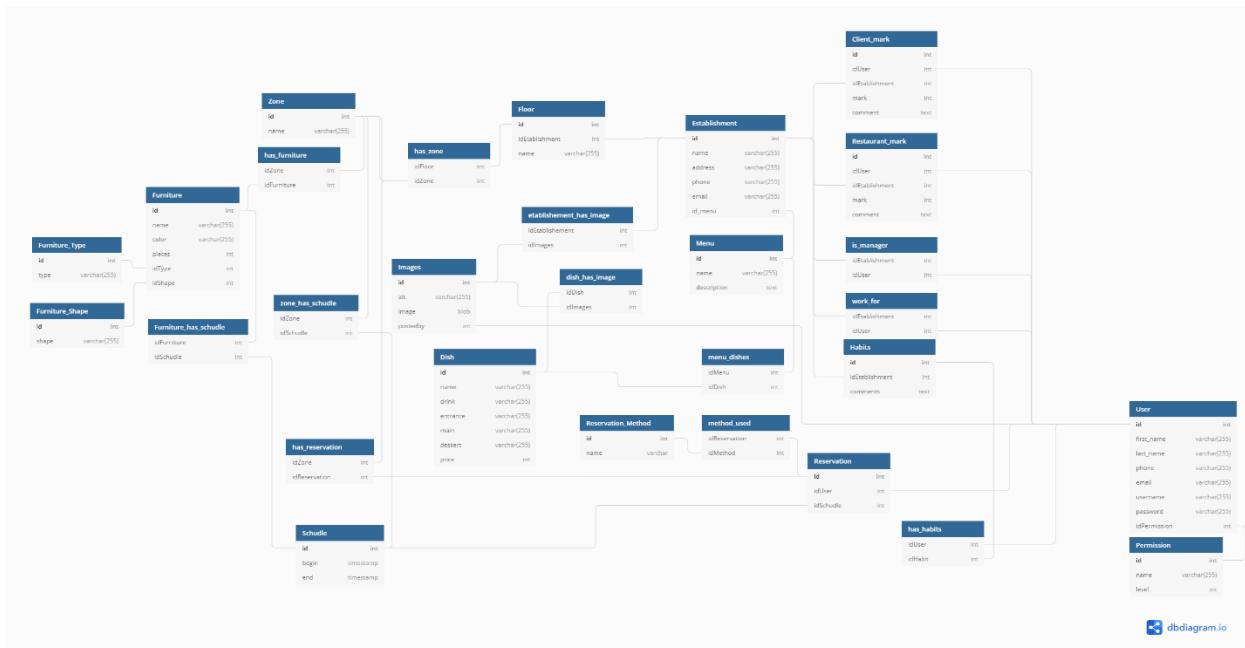


Figure 65 Aperçu MCD

8.7.4 Données de tests

8.7.4.1 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

8.7.4.2 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

8.7.4.3 Les niveaux d'abonnement des restaurants

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

8.7.5 Tables

RESA regroupe 37 tables, toutes les tables n'ont pas pu être utilisées dans le développement de l'application avec le temps imparti. Comme elles sont existantes, il suffit de créer les fonctions dans l'API afin de pouvoir les utiliser.

8.8 RESA Apps (Client/Blog/Pro/Full)

8.8.1 Resa-project.ch

Les applications RESA sont accessibles depuis le lien resa-project.ch. Pour accéder à la page de manager, il faut passer par manager.resaproject.ch.

8.8.2 SESSION

Pour efficacement passer des informations entre les pages (principalement au moment de la création d'un établissement), j'ai utilisé la variable globale de PHP nommée `$_SESSION`.

L'utilisation de cette variable se fait en utilisant la commande `session_start()` ; au début de la page qui l'utilise.

Nous allons voir comment la session a été utilisée lors de la création d'un établissement.

UTILISATION DE LA SESSION

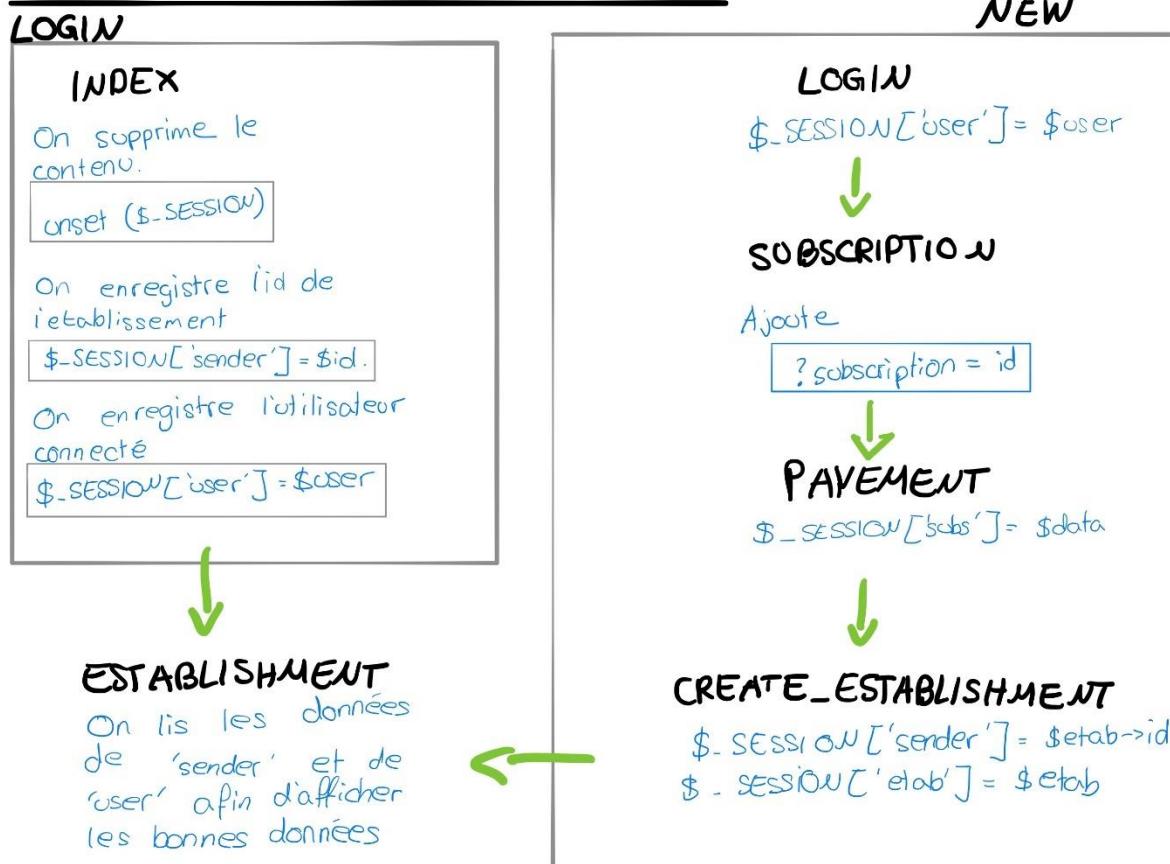


Figure 66 Schéma utilisation session

Dans mon exemple avec la création d'un établissement, on voit bien que la variable user est utilisée à plusieurs reprises. Il est bien de noté que s'il manque une variable dans la session, RESA renverra automatiquement l'utilisateur à l'étape précédente afin que celui-ci remplisse les demandes pour continuer.

Principalement, c'est la variable `$_SESSION['user']` qui a été utilisée. Cette dernière permet à toutes les pages de savoir si l'utilisateur est connecté ou non. Dans le cas où il ne l'est pas, la variable est égale à `NULL`.

Lors du logout de l'utilisateur (ou de son retour sur la page de login du manager) la session est remise à 0.

8.9 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.

8.9.1 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](#)].

8.9.2 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.

8.9.3 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.

8.9.3.1 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.

8.9.3.1.1 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
```

8.9.3.2 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 retornée 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 créer à partir de la valeur JSON

Toutes les valeurs renvoyées 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

8.9.4 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.

8.9.4.1 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.

8.9.4.1.1 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['address'],  
    '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. Crédit 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

8.9.4.2 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).

8.9.4.2.1 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	/api/v2/images/get/?data&id=XX	Un tableau avec les informations de l'image
establishment	/api/v2/images/get/?establishment&id=XX	Un tableau avec l'id des images ainsi que leur lien complet
dish	/api/v2/images/get/?dish&id=XX	Un tableau avec l'id des images ainsi que leur lien complet
user	/api/v2/images/get/?user&id=XX	L'id de l'image ainsi que son chemin complet
id	/api/v2/images/get/?id=XX	Redirige directement sur l'image

Le dernier paramètre (id) 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

8.9.5 Réservations et horaires

8.9.5.1 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.

8.9.5.1.1 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

8.9.5.1.2 Crédit 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 :

```
Avaible // 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.idEtablissement = 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.idEtablissement = 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 :

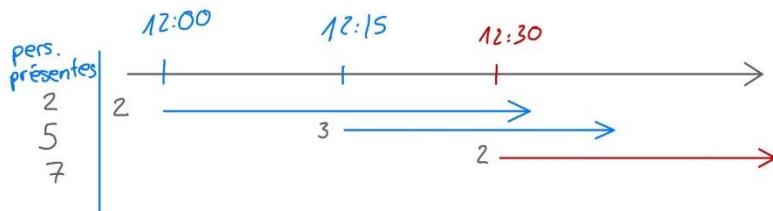
```
CALL IsPlaceForReservation(1, '2020-05-14', '12:30', 3600, @avaible);
```

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

8.9.5.2 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



$7 < 10$, donc la réservation

Figure 67 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.

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.

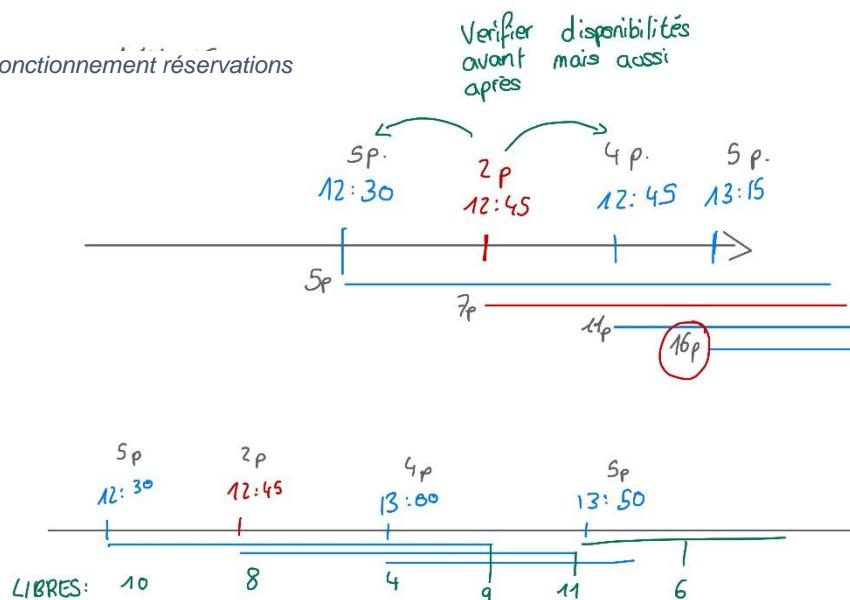


Figure 68 Schéma vérification de places

Après avoir raisonné, je suis parti sur cette solution :

Ma solution consiste à vérifier le nombre de places totales disponibles à l'instant [t], puis de soustraire le

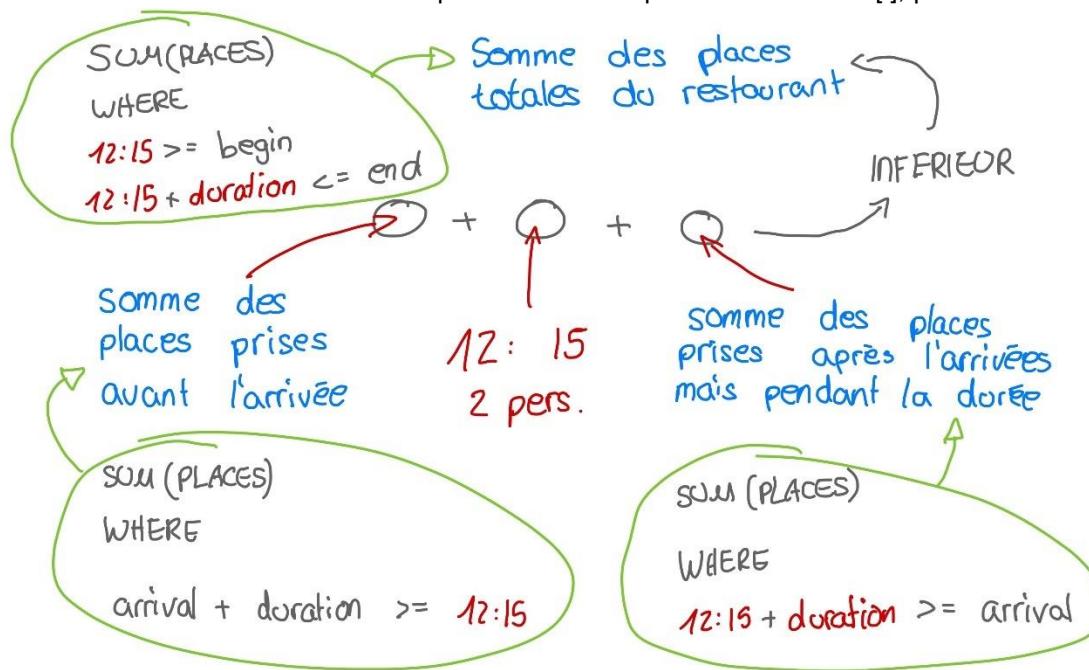


Figure 69 Solution fonctionnement de réservations

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].

8.9.6 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.

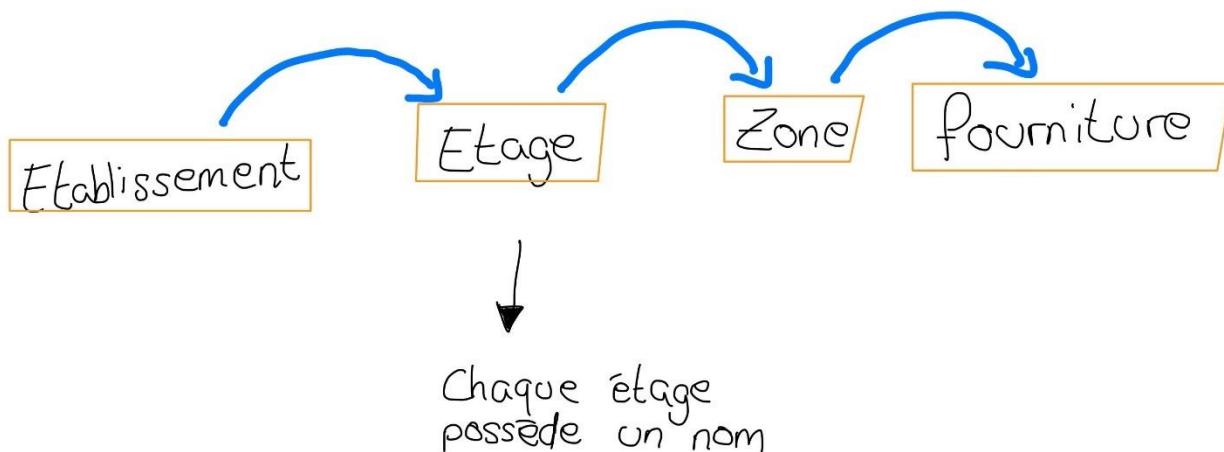


Figure 70 Schéma de lien entre les tables principales

8.9.6.1 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

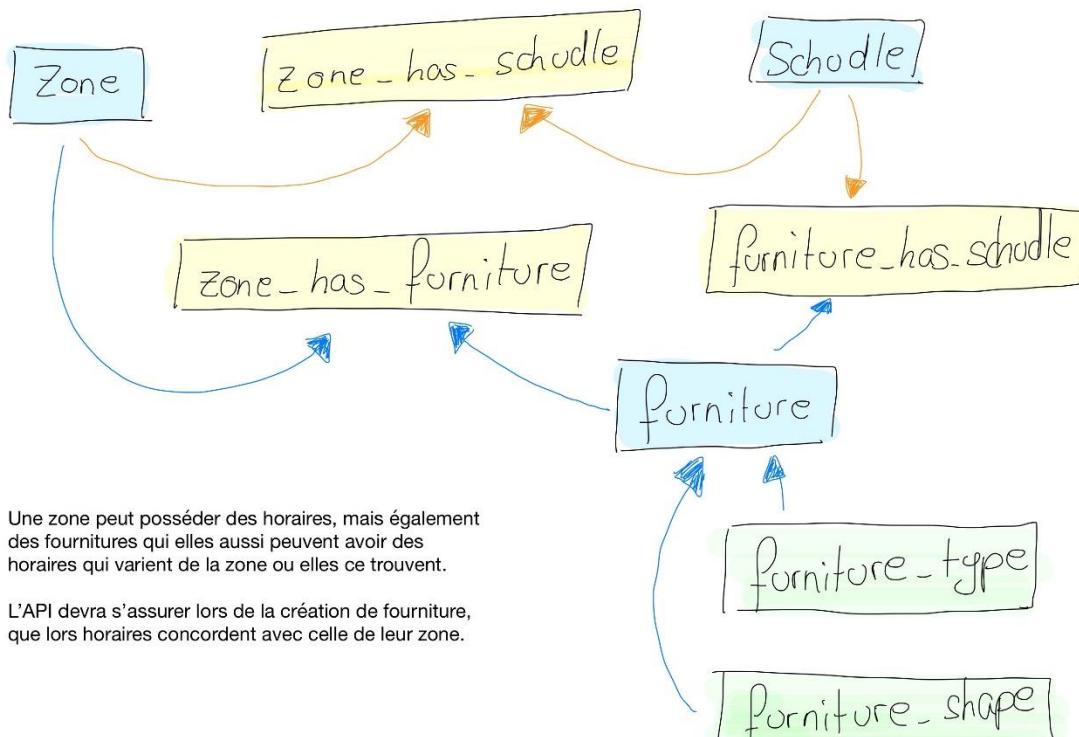


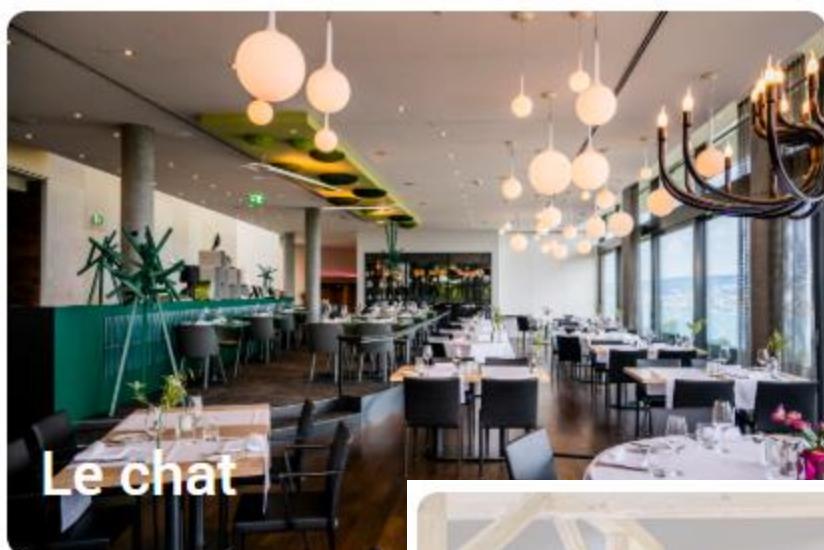
Figure 71 Schéma des liens entre les tables pour une zone

8.9.6.2 *Le fonctionnement des zones*
[parler du fonctionnement particulier des zones]

8.10 L'application

8.10.1 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_descripiton, (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).

8.10.2 Login vs Fast-login vs Manager-login

8.10.2.1 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.

8.10.2.2 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

8.10.2.3 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.

8.10.3 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

8.11 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>

8.11.1 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 »

8.11.2 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 72 Liste des tâches pour la page profil

8.11.3 Le planning

Pour visualiser le travail à faire et pour les comparer au résultat final, j'ai décidé de faire un planning prévisionnel ainsi qu'un planning réel.

En regardant les graphiques¹⁰, on voit tout de suite que les deux n'ont aucun point en commun.

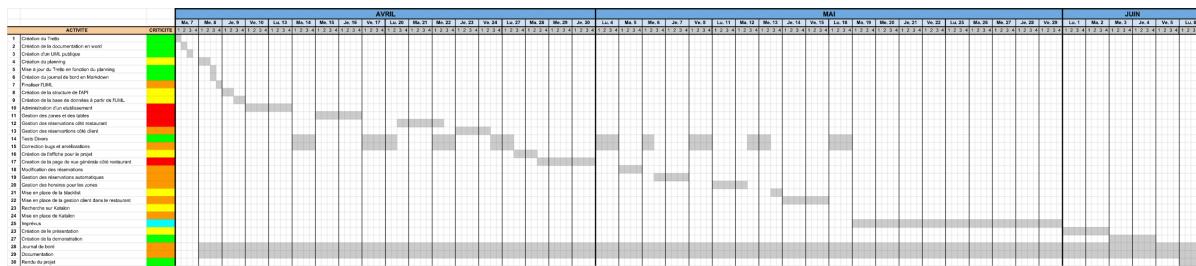


Figure 73 planning initial

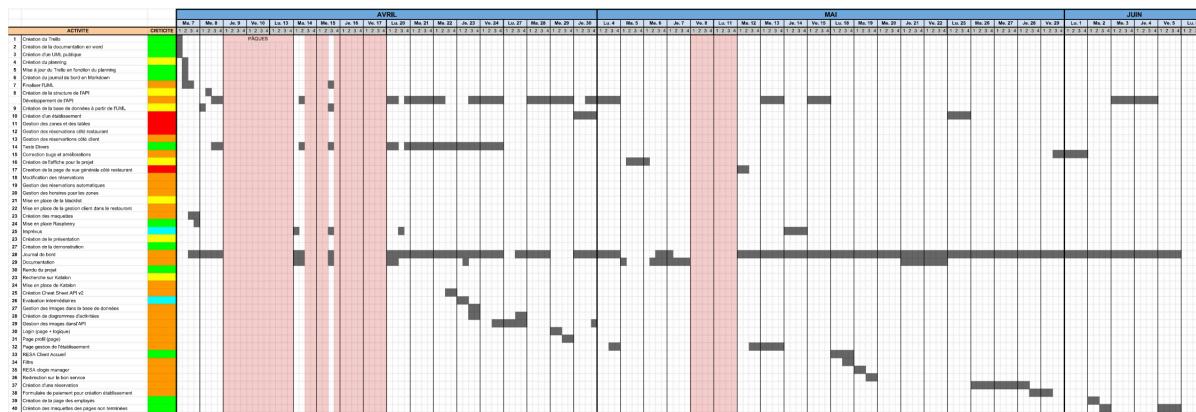


Figure 74 planning final

Lors de la réalisation du projet, beaucoup de tâches sont venues s'ajoutées au planning, il y a également eu le développement de l'API qui me prenait particulièrement de temps tout au long du projet. On peut également voir que certaines tâches ont de l'être reprises, car celles-ci changeaient au fur à mesure que le projet avançait (les demandes du client ou de l'enseignant).

Le planning initial n'a pas du tout pu être respecter dû à la méthode choisie pour le suivi du projet. La méthode utilisée dite « agile » me permettait de tous les jours faire un point de mon avancement et de faire des ajustements sur mes objectifs en cas de besoins.

¹⁰ Les deux plannings sont consultables dans les annexes.

8.11.4 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.

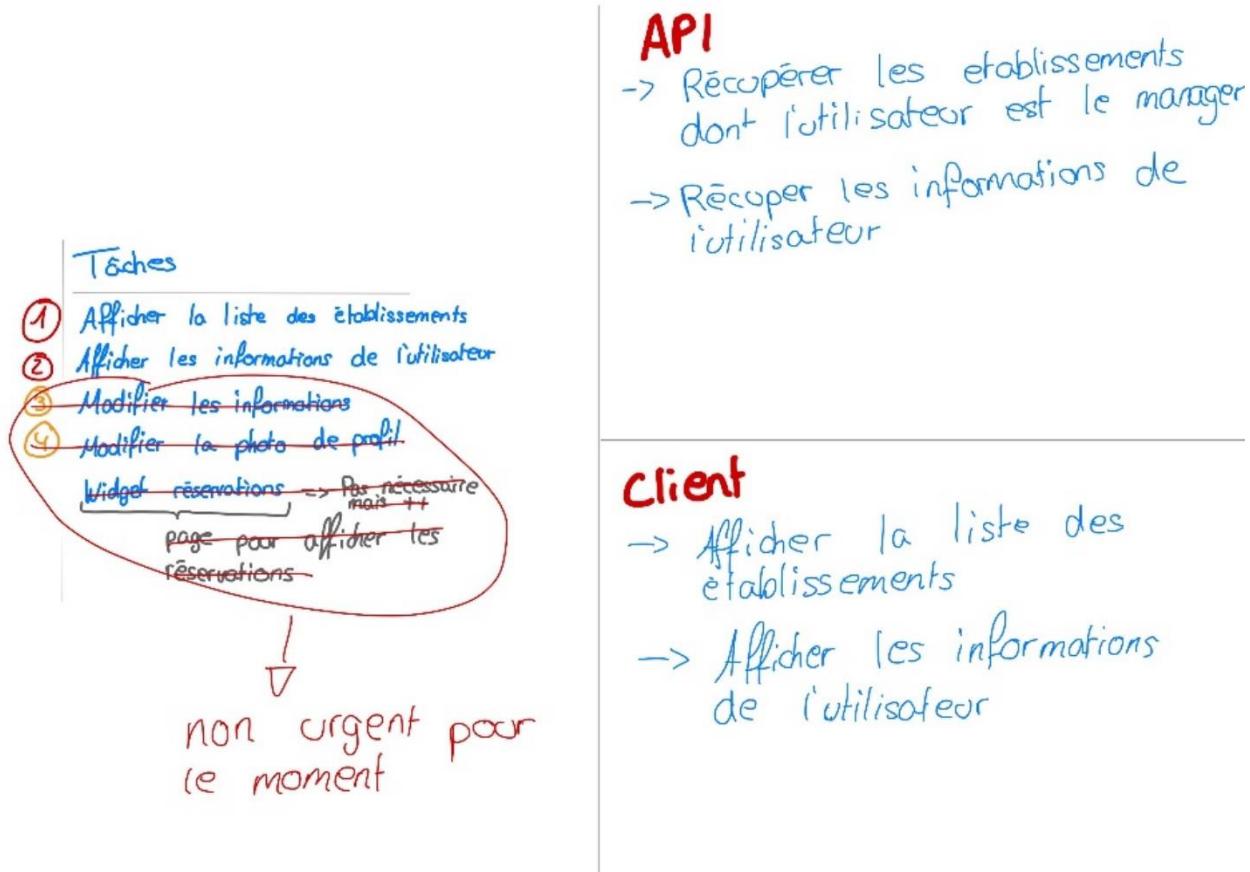


Figure 75 Séparation des tâches client et serveur (API)

8.11.5 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.

8.12 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.

8.12.1 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.

8.12.1.1 *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. Il est également consultable dans les annexes.

8.12.1.1.1 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.

8.12.1.1.2 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.

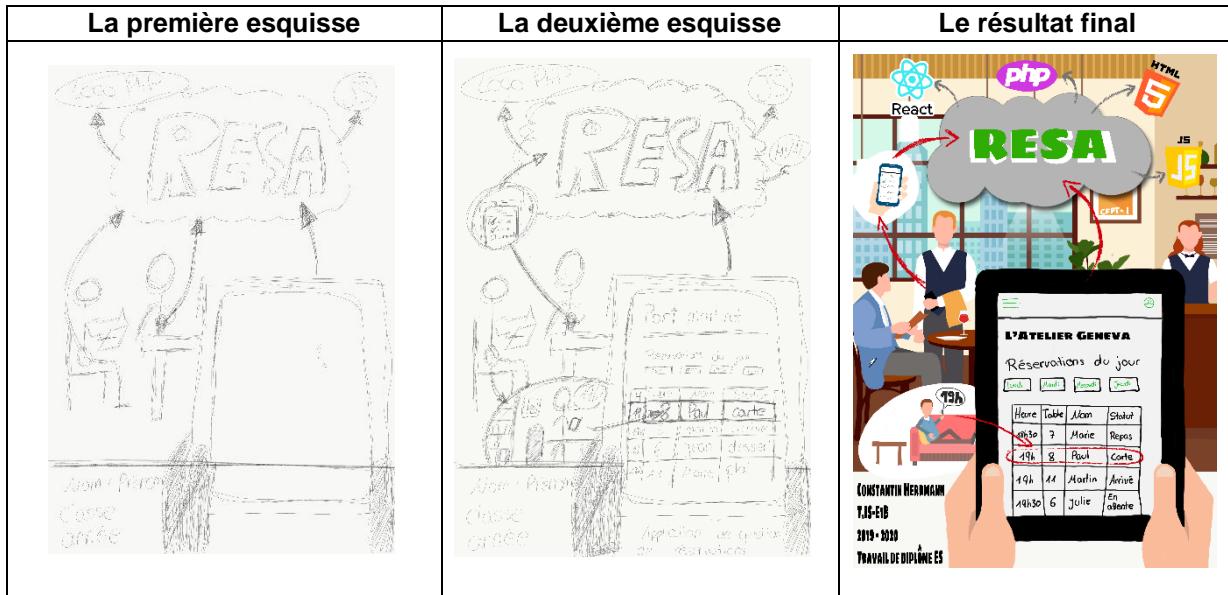
- Il faut que j'ajoute à l'API une fonctionnalité qui gère la différence
 - Je vais donc créer cette fonctionnalité
 - Afin de pouvoir ajouter cette fonctionnalité, j'ai un petit souci... Un utilisateur admin peut se connecter dans tous les restaurants, il faut donc que ma requête gère s'il s'agit d'un 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 Github¹¹

¹¹ https://github.com/ConstantinHrrmn/Travail_diplome_ES_2020/blob/master/logbook.md

8.12.1.2 *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 \(plein format\)](#).

8.12.1.3 *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.

8.12.2 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.

8.12.3 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.

8.12.3.1 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

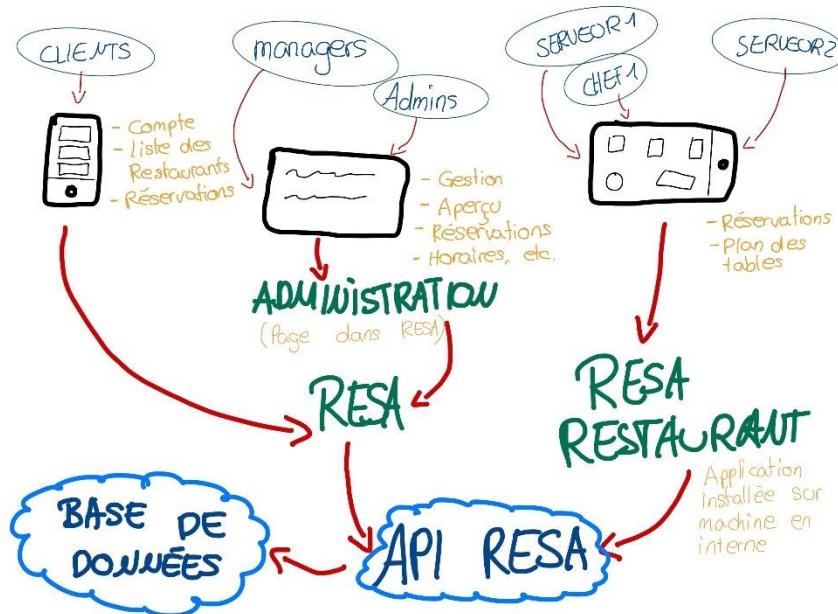


Figure 76 Plan de l'application avant les changements

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).

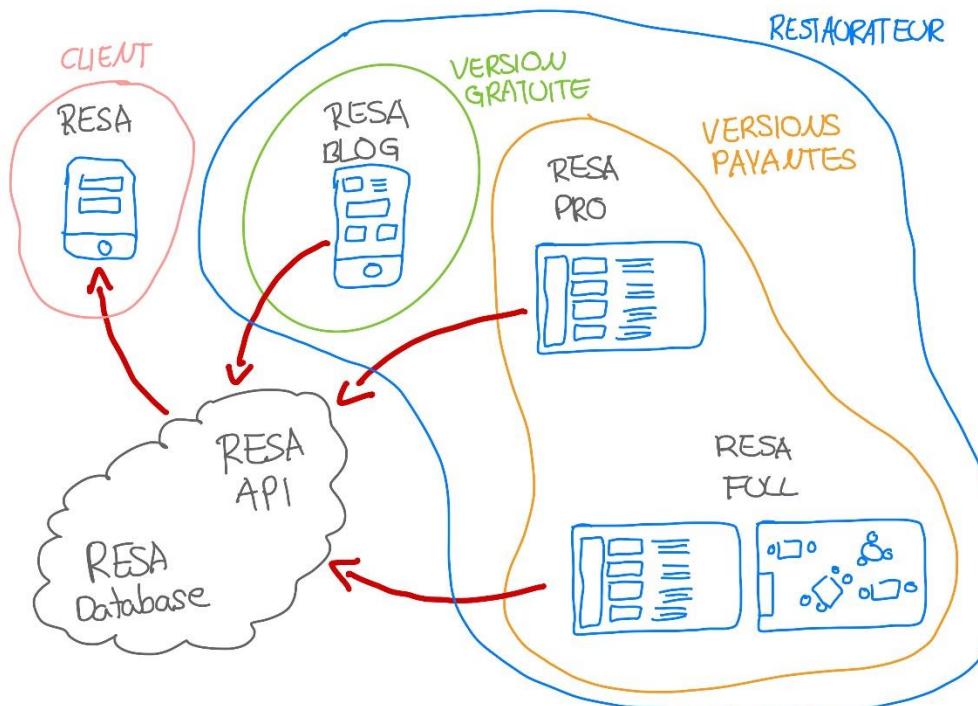


Figure 77 Plan de l'application après discussion sur les niveaux d'abonnements

8.13 Améliorations futures

8.13.1 Finaliser les vues

Avant toute amélioration, il faudrait finaliser toutes les vues afin que celles-ci soient navigables et utilisables. Pour le moment, les pages non terminées ou non faites sont remplacées par des images qui donnent un exemple des pages qu'il doit y avoir.

8.13.2 API

La première amélioration qu'il faudrait apporter à l'API serait celle de la sécurité pour les connexions externes. Je pense par exemple à l'utilisation de token pour les comptes.

Il faudrait ensuite améliorer la répartition des fonctions au sein de l'API. Comme le projet évoluait en permanence, les fonctions déjà faites ne changeaient pas de place.

8.13.3 Paiement

Il faudrait que je mette les fonctionnalités de paiement avec les banques et éventuellement ajouter les fonctionnalités de paiement par PayPal.

8.13.4 Connexion par Google/Facebook/Apple

Il serait bien de pouvoir aussi se connecter à l'aide des services connus tels que Google ou Facebook.

8.14 Bilans

Dans la partie des bilans, les textes seront écrits en « je », car il s'agit de mon ressenti sur le projet.

8.14.1 Bilan Technique

Le projet RESA représente une grande quantité de travail qui n'a pas pu entièrement réaliser dans le temps imparti. Le nombre de fonctionnalités demandées était gigantesque.

De plus, mon application a été entièrement faite par moi-même. J'ai uniquement utilisé Bootstrap pour le style de mes pages. Ce qui fait que j'ai perdu pas mal de temps sur la création de mon API par exemple.

Malheureusement je n'ai pas réussi à remplir entièrement mon cahier des charges, mais l'application est devenue navigable et il est possible de réaliser certaines fonctionnalités du cahier des charges.

Du côté de l'API, celle-ci est avancée et permet son utilisation au sein d'autres applications. Je n'ai pas réussi (ou eu assez de temps) afin de la protéger. C'est-à-dire que n'importe qui peut accéder aux données de cette dernière en entrant l'URL. Malgré ce manque de sécurité dans l'accès, j'ai mis en place la sécurité sur les requêtes créées avec les données de l'URL, ce qui fait qu'on ne peut pas supprimer ma table ou accéder à des données non souhaitées.

8.14.2 Bilan personnel

Avant de commencer ce travail, je savais que ça n'allait pas être facile et que j'allais avoir beaucoup de travail afin de réaliser toutes les tâches qui m'étaient demandées. Dès le début, j'ai dû faire des choix sur mes objectifs principaux.

Le fait de pouvoir parler tous les jours avec M. Garcia m'a permis de beaucoup me remettre en question sur les choses que je faisais ainsi que leur utilité. Vers la moitié du projet, quand je me suis rendu compte avec les retours de la cliente que je devais revoir la structure de mon projet, je me suis dit que ce que j'avais fait n'avait servi à rien, mais je me suis vite rendu que la manière dont j'avais développé mon API

me permettait de faire très facilement des changements dans mon application. Par moment, j'avais l'impression de faire du bricolage pour coller les morceaux que j'avais déjà faits sans devoir faire trop de changements dans les pages et scripts déjà créés.

Sans réellement le vouloir, le fait de ne pas avoir utilisé de Framework pour le développement de mon API m'a permis d'en apprendre beaucoup sur le PHP et sur son fonctionnement, principalement dans la gestion d'image et les variables globales telles que les sessions. J'ai également énormément appris sur SQL et les requêtes possibles à faire ainsi que les procédures.

Je reste déçu de ne pas avoir réussi à finir mon projet comme je l'aurais souhaité, mais je me rends compte que dans tous les cas, je n'aurais jamais pu finir.

Ce projet m'a beaucoup appris sur l'organisation et la gestion de travail 100% autonome comme je ne pouvais pas demander d'aide pour des choses que je ne savais à ma classe. Certes, je pouvais contacter M. Garcia, mais je devais d'abord chercher une solution par moi-même.

8.14.2.1 *Situation particulière*

Il est bien de mentionner que le travail de diplôme a été réalisé dans une situation dite extraordinaire causée par le virus COVID-19.

Le fait de devoir travailler en permanence depuis la maison n'aide pas toujours. Le contact avec les collègues de classe ainsi que les enseignants permettent de mieux stimuler dans le travail. Le fait de rester à la maison pouvait facilement m'entraîner à faire autre chose ou à me déconcentrer.

Finalement je suis heureux d'avoir réussi à gérer cette situation efficacement sans impacter mon travail.

8.14.2.2 *Apport personnel*

À l'exception de l'utilisation de Bootstrap pour le style du site et du template de COSTIC HTML, mon apport a été de 100%. En effet, je n'ai utilisé aucun Framework, quel qu'il soit pour réaliser mon application ou mon API.

8.15 Conclusion

RESA est un ensemble d'applications qui permet aux clients et aux managers d'établissement de facilement créer un lien de communication. Principalement orienté sur les réservations, RESA permet également la gestion du personnel, des clients, des avis et des menus.

Je suis content du travail que j'ai effectué et du projet qui est réalisable grâce à ma recherche préalable. Peut-être qu'un jour, resa-project.ch deviendra resa.ch !

9 Tables des figures

Figure 1 Logo "lafouchette"	12
Figure 2 Page d'accueil	
Figure 1 Logo "lafouchette"	Erreur ! Signet non défini.
Figure 2 Page d'accueil.....	13
Figure 3 Menu de l'utilisateur	
Figure 2 Page d'accueil	Erreur ! Signet non défini.
Figure 3 Menu de l'utilisateur	13
Figure 4 Page du restaurant	
Figure 3 Menu de l'utilisateur.....	13
Figure 4 Page du restaurant	13
Figure 5 Page d'accueil	
Figure 4 Page du restaurant	Erreur ! Signet non défini.
Figure 5 Page d'accueil.....	14
Figure 6 Login manager	
Figure 5 Page d'accueil.....	Erreur ! Signet non défini.
Figure 6 Login manager.....	14
Figure 7 calendrier du manager	
Figure 6 Login manager	Erreur ! Signet non défini.
Figure 7 calendrier du manager	15
Figure 8 Formulaire de création d'un client	
Figure 7 calendrier du manager	Erreur ! Signet non défini.
Figure 8 Formulaire de création d'un client.....	15
Figure 9 Dropdown menu du statut	
Figure 8 Formulaire de création d'un client Erreur ! Signet non défini.	
Figure 9 Dropdown menu du statut.....	15
Figure 10 Sélection heure réservation manager	
Figure 9 Dropdown menu du statut Erreur ! Signet non défini.	
Figure 10 Sélection heure réservation manager	16
Figure 11 Sélection date réservation manager	16
Figure 12 Formulaire de création de réservation	16
Figure 13 Schéma de la structure RESA	
Figure 12 Formulaire de création de réservation	16
Figure 13 Schéma de la structure RESA	22
Figure 14 Capture d'écran du template	23
Figure 15 Capture d'écran de la page d'accueil RESA	23
Figure 16 Schéma de navigation RESA Client.....	24
Figure 17 On est là - Accueil RESA Client.....	24
Figure 18 Page d'accueil de RESA client.....	25
Figure 19 Menu "non-connecté"	25
Figure 20 Menu "connecté"	25
Figure 21 On est là - page de l'établissement.....	26
Figure 22 On est là - Page de login	27
Figure 23 Login d'un utilisateur	27
Figure 24 On est là - Page de profil	27
Figure 25 Navigation possibles depuis page de profil	28

Figure 26 Page de profil.....	28
Figure 27 Login pour un manager de restaurant.....	29
Figure 28 Schéma de navigation RESA manager.....	29
Figure 29 Lien création établissement	30
Figure 30 Choix abonnement.....	30
Figure 31 Page de paiement d'abonnement	30
Figure 32 Formulaire de création	31
Figure 33 Schéma de navigation RESA Blog.....	31
Figure 34 On est là - accueil blog	32
Figure 35 Page accueil BLOG	32
Figure 36 On est là - paramètres blog	32
Figure 37 On est là - mise en ligne image blog.....	33
Figure 38 Schéma de navigation RESA Pro	34
Figure 39 On est là - accueil RESA Pro.....	34
Figure 40 On est là - Réservations RESA Pro	34
Figure 41 On est là - paramètres RESA Pro	35
Figure 42 Schéma de navigation RESA Full	36
Figure 43 On est là - Accueil RESA Full	36
Figure 44 Page de paramètres d'un établissement.....	37
Figure 45 On est là - paramètres RESA Full.....	37
Figure 46 On est là - réservations RESA Full	37
Figure 47 On est là - Employés RESA Full	37
Figure 48 Page des employés	38
Figure 49 Page du plan de table RESA FULL.....	39
Figure 50 Page du plan de table en modifications	40
Figure 51 Hiérarchie de RESA.....	41
Figure 52 Schéma des droits d'un administrateur	42
Figure 53 Schéma des droits d'un manager	42
Figure 54 Schéma des droits d'un serveur.....	42
Figure 55 Schéma des droits d'un client	43
Figure 56 Mindmap de préparation	47
Figure 57 : Interface de Github Desktop	48
Figure 58 diagramme de fonctionnement RESA.....	49
Figure 59 Aperçu interface dbdiagram.io	50
Figure 60 Aperçu MCD	51
Figure 61 Schéma utilisation session.....	52
Figure 62 Schéma questionnement fonctionnement réservations	59

Figure 63 Schéma vérification de places	59
Figure 64 Solution fonctionnement de réservations	60
Figure 65 Schéma de lien entre les tables principales	61
Figure 66 Schéma des liens entre les tables pour une zone	61
Figure 67 Liste des tâches pour la page profil	65
Figure 68 Séparation des tâches client et serveur (API)	67
Figure 69 Plan de l'application avant les changements	70
Figure 70 Plan de l'application après discussion sur les niveaux d'abonnements	70
Figure 71 Diagramme d'activité - création d'un compte et login	77
Figure 72 Diagramme d'activité - faire une réservationFigure 71 Diagramme d'activité - création d'un compte et login	77
Figure 72 Diagramme d'activité - faire une réservation	78
Figure 73 Création d'un commentaireFigure 72 Diagramme d'activité - faire une réservation	78
Figure 73 Création d'un commentaire	79
Figure 74 Diagramme de fonctionnement de loginFigure 73 Création d'un commentaire	79
Figure 74 Diagramme de fonctionnement de login	80
Figure 75 Redirection application correspondante au login	80

10 Tables des extraits de code

Extrait de code n°1.	Arborescence de Github	44
Extrait de code n°2.	Commande cmd pour créer le projet react.....	45
Extrait de code n°3.	Arborescence d'un projet react basique.....	45
Extrait de code n°4.	Header des fichiers PHP de RESA.....	45
Extrait de code n°5.	Header d'une fonction PHP	46
Extrait de code n°6.	Envoi d'une requête http à l'API.....	54
Extrait de code n°7.	Valeur JSON retornée par une fonction de l'API.....	54
Extrait de code n°8.	Objet PHP créer à partir de la valeur JSON.....	54
Extrait de code n°9.	Mise en forme de données avant l'envoi en json	54
Extrait de code n°10.	Formulaire à créer pour correctement envoyer les images à l'API	55
Extrait de code n°11.	Vérification des données reçues par un formulaire	55
Extrait de code n°12.	Création de la requête http pour créer un établissement	56
Extrait de code n°13.	Fonction de sauvegarde de l'image pour un établissement	56
Extrait de code n°14.	Balise HTML d'une image avec comme chemin l'image dans l'API.....	56
Extrait de code n°15.	Création d'une routine dans MySQL.....	57
Extrait de code n°16.	Comment appeler une routine en MySQL.....	57
Extrait de code n°17.	Les variables utilisées dans ma routine SQL	57
Extrait de code n°18.	La variable de sortie de ma routine SQL.....	57
Extrait de code n°19.	Ma procédure pour vérifier s'il y a de la place.....	58
Extrait de code n°20.	Appel de ma routine SQL avec paramètres	58
Extrait de code n°21.	Requête SQL horaire magasin	62
Extrait de code n°22.	Requête SQL qui récupère tous les restaurants	63
Extrait de code n°23.	Login d'un utilisateur	63
Extrait de code n°24.	Redirection sur la page « fast_login.php ».....	64
Extrait de code n°25.	Unset d'une variable de la session	64
Extrait de code n°26.	Redirection header sur avec une variable.....	64
Extrait de code n°27.	Affichage du calendrier	64
Extrait de code n°28.	Séparation des jours dans le logbook	68
Extrait de code n°29.	Extrait du logbook disponible sur Github.....	68

11 ANNEXES

11.1 Diagrammes D'activités

11.1.1 Création de comptes

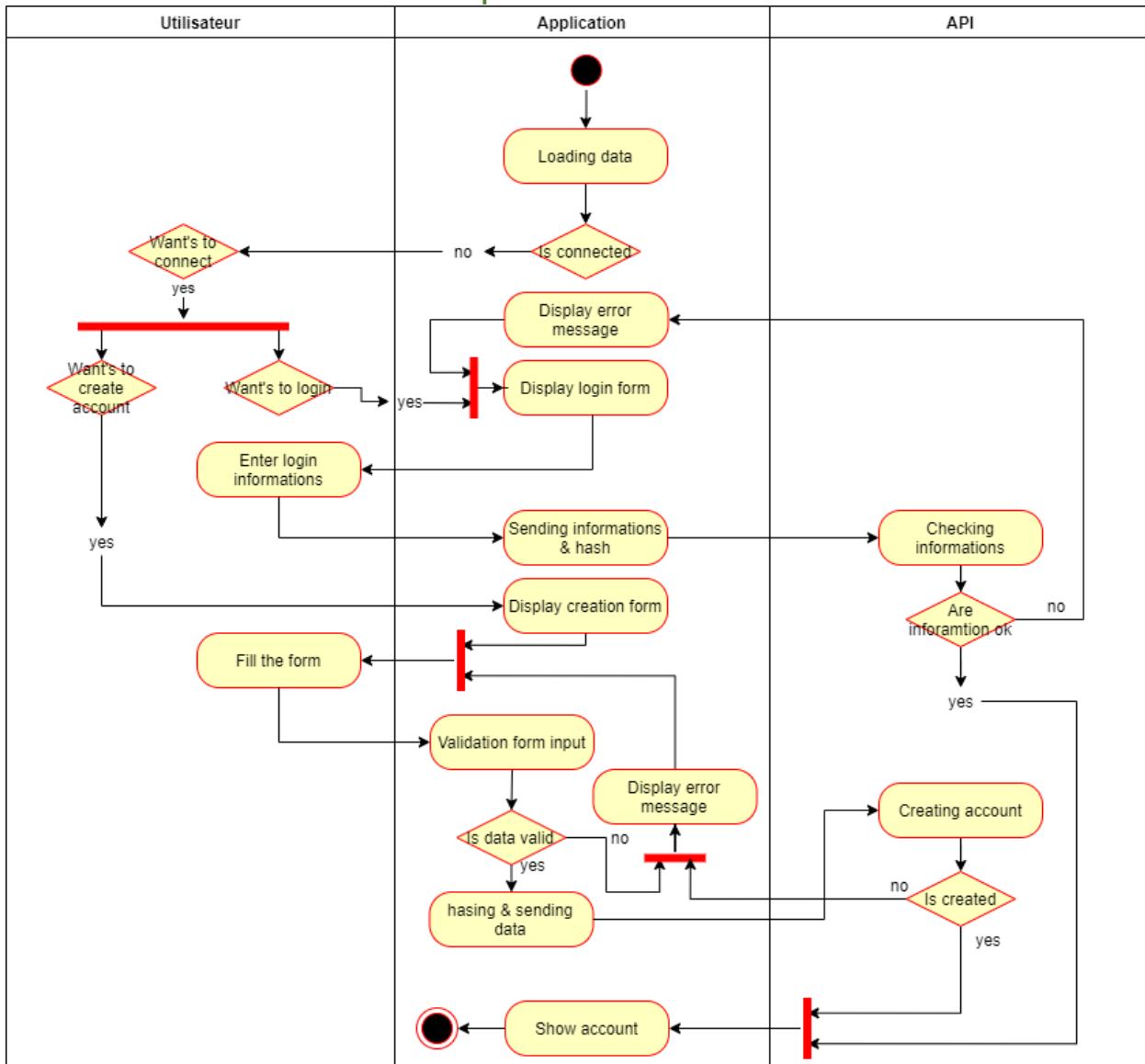


Figure 78 Diagramme d'activité - création d'un compte et login

Figure 79 Diagramme d'activité - faire une réservationFigure 80 Diagramme d'activité - création d'un compte et login

11.1.2 Réservation

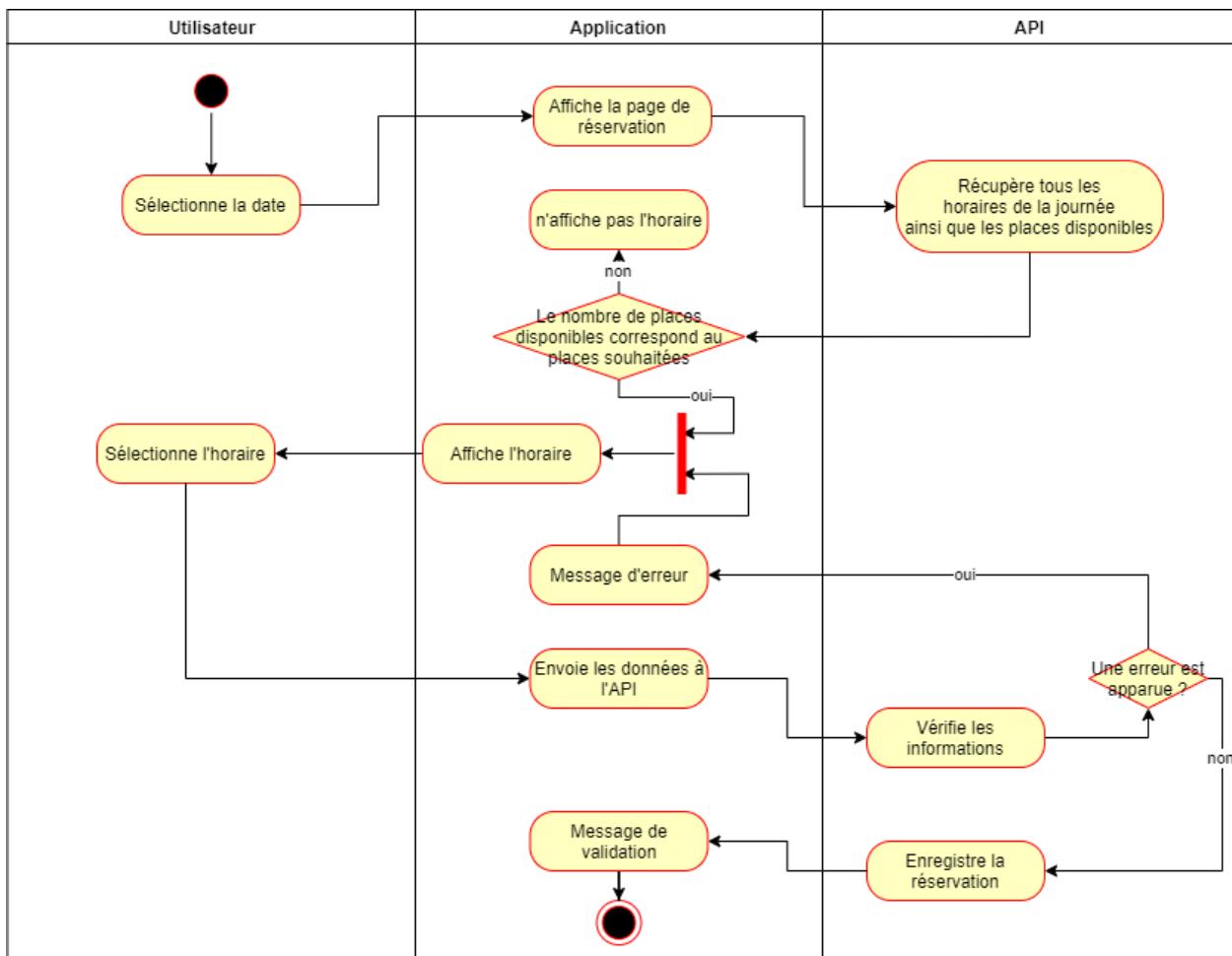


Figure 81 Diagramme d'activité - faire une réservation

Figure 82 Crédit d'un commentaireFigure 83 Diagramme d'activité - faire une réservation

11.1.3 Laisser un commentaire

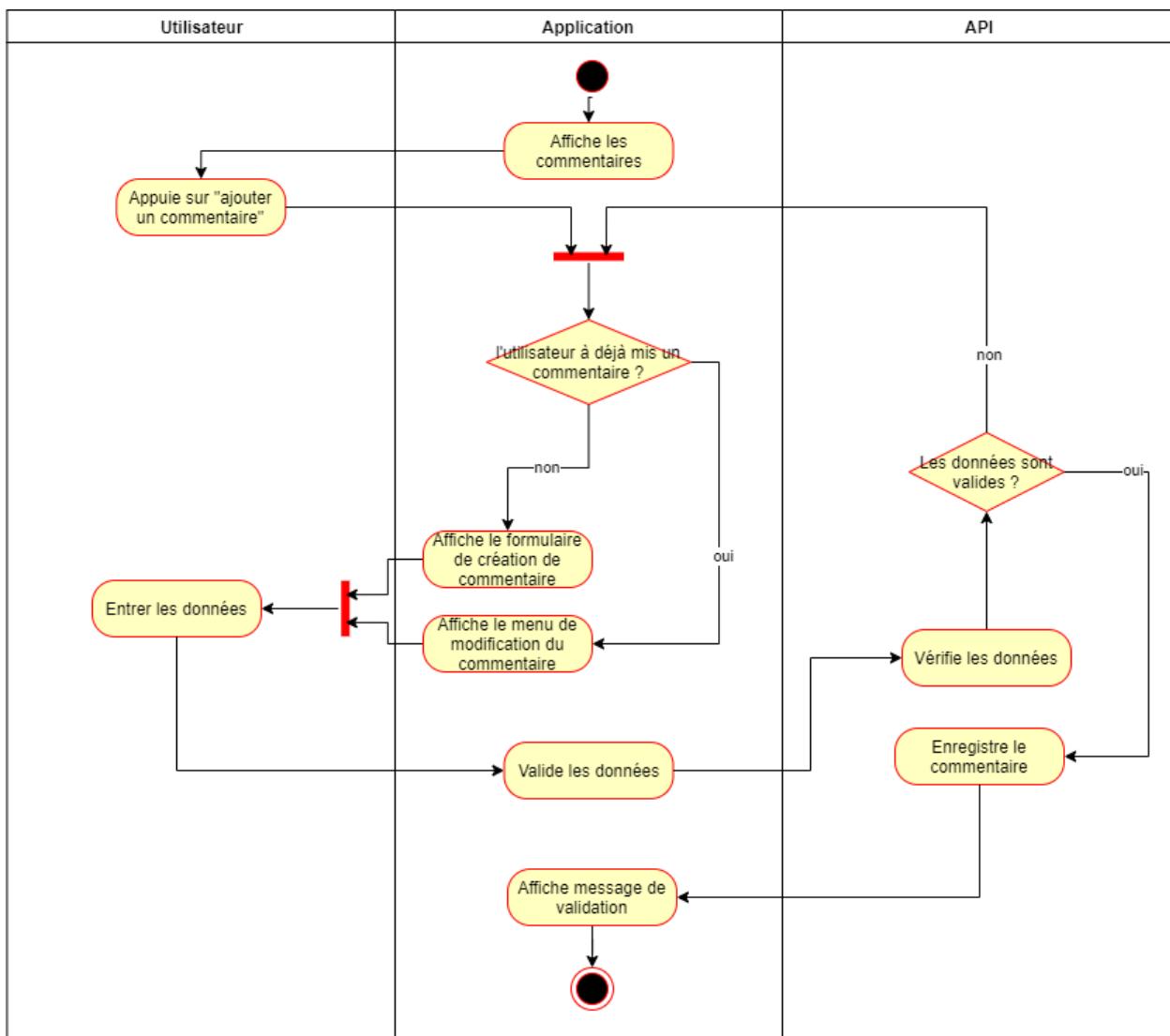


Figure 84 Crédit d'un commentaire

Figure 85 Diagramme de fonctionnement de loginFigure 86 Crédit d'un commentaire

11.2 Diagrammes de fonctionnement

11.2.1 Login

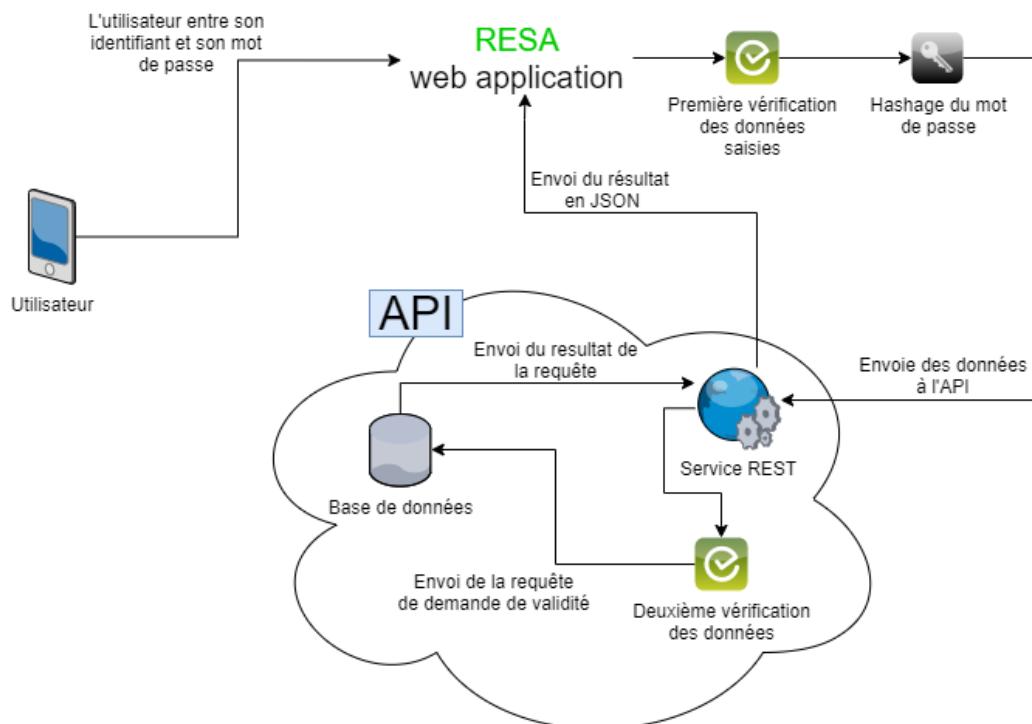


Figure 87 Diagramme de fonctionnement de login

11.2.2 Redirection sur bonne application

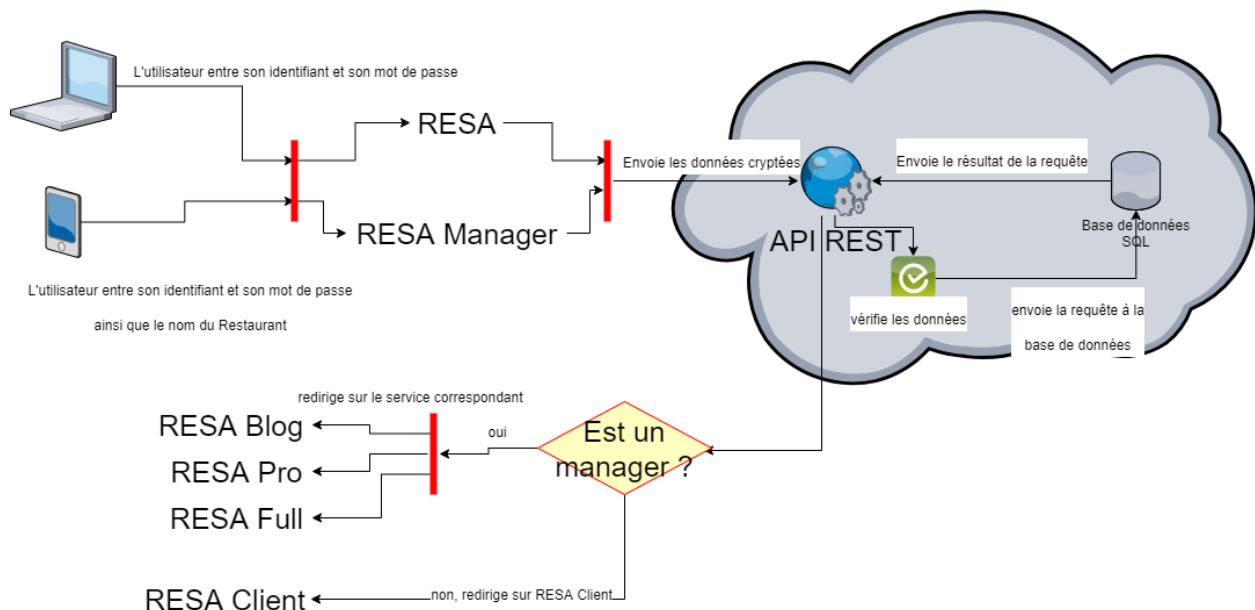


Figure 88 Redirection application correspondante au login

11.3 Cheat Sheet de l'API

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

11.3.1 Etablissement

11.3.1.1 Create

11.3.1.1.1 Création d'un établissement sans manager

Lien :

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

Paramètres :

```
Name : le nom du l'établissement  
Street : la rue et le numéro de l'établissement  
Town : la ville de l'adresse  
Npa : le numéro NPA de l'adresse  
Country : l'id du pays de la table pays  
Subs : l'id de l'abonnement  
Phone : le téléphone du l'établissement  
Email : l'email du l'établissement
```

Lien avec paramètres :

```
/api/v2/establishment/create/?name=[nom]&phone=[phone]&email=[email]&street=[street]&town=[town]&n  
pa=[npa]&country=[id country]&subs=[id subs]
```

11.3.1.1.2 Création d'un établissement avec un manager

Lien :

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

Paramètres :

```
Name : le nom du l'établissement  
Street : la rue et le numéro de l'établissement  
Town : la ville de l'adresse  
Npa : le numéro NPA de l'adresse  
Country : l'id du pays de la table pays  
Subs : l'id de l'abonnement  
Phone : le téléphone du l'établissement  
Email : l'email du l'établissement  
creatorID : l'id de l'utilisateur qui créer le restaurant
```

Lien avec paramètres :

```
/api/v2/establishment/create/?name=[nom]&phone=[phone]&email=[email]&street=[street]&town=[town]&n  
pa=[npa]&country=[id country]&subs=[id subs]&creatorID=[id utilisateur]
```

11.3.1.1.3 Création en passant pas une form

Lien :

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

Paramètres :

```
$_POST :  
Name : le nom du restaurant
```

Street : la rue et le numéro de l'établissement

Town : la ville de l'adresse

Npa : le numéro NPA de l'adresse

Country : l'id du pays de la table pays

Phone : le téléphone du restaurant

Email : l'email du restaurant

creatorID : l'id de l'utilisateur qui créer le restaurant

\$_FILES :

Les images ajoutées

11.3.1.2 Get

11.3.1.2.1 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)

11.3.1.2.2 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

11.3.1.2.3 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

11.3.1.2.4 Niveau d'abonnement

Lien :

/api/v2/establishment/get/

Paramètres :

Level : (aucune valeur)
I : l'id du restaurant

Lien avec paramètres :

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

Retourne :

Level : le niveau d'abonnement

11.3.1.2.5 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 :

-

11.3.1.2.6 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

11.3.1.2.7 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 :

-

11.3.1.2.8 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

11.3.1.2.9 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 »

11.3.2 Etages

11.3.2.1 Get

11.3.2.1.1 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

11.3.2.2 Create

11.3.2.2.1 Crédation 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 établissement]

Retourne :

-

11.3.2.2.2 Crédation 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 :

-

11.3.3 Zones

11.3.3.1 GET

11.3.3.1.1 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

11.3.3.1.2 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

11.3.3.1.3 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

11.3.3.2 CREATE

11.3.3.2.1 Création 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 :

-

11.3.3.2.2 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 :

-

11.3.3.2.3 Création 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 :

-

11.3.4 Fournitures

11.3.4.1 GET

11.3.4.1.1 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

11.3.4.1.2 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 :

-

11.3.5 Images

11.3.5.1 GET

11.3.5.1.1 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

11.3.5.1.2 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

11.3.5.1.3 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

11.3.6 Menu

11.3.6.1 GET

11.3.6.1.1 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

11.3.6.1.2 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

11.3.6.1.3 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

11.3.6.1.4 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

11.3.6.1.5 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

11.3.6.1.6 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

11.3.6.1.7 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 :

Avalaible : le nombre de places disponibles

11.3.7 Subscriptions

11.3.7.1 GET

11.3.7.1.1 Get all subscriptions

Lien :

`/api/v2/establishment/subscriptions/`

Paramètres :

All : (aucune valeur)

Lien avec paramètres :

`/api/v2/establishment/subscriptions/ ?all`

Retourne :

Tableau

Id : l'id de l'abonnement
Name : nom de l'abonnement
Price : le prix de l'abonnement
Level : le niveau de l'abonnement

11.3.7.1.2 Get by Id

Lien :

`/api/v2/establishment/subscriptions/`

Paramètres :

Id : l'id de l'abonnement recherché

Lien avec paramètres :

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

Retourne :

Id : l'id de l'abonnement
Name : nom de l'abonnement
Price : le prix de l'abonnement
Level : le niveau de l'abonnement

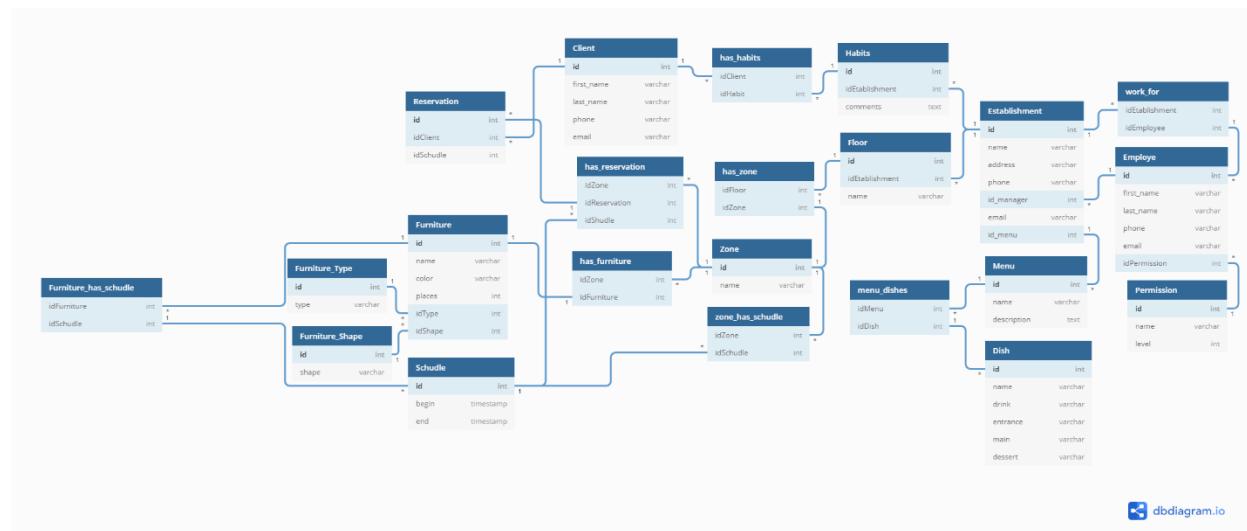
11.4 Journal de bord

11.4.1 06.04.20

- Création du Github
- Création du Trello
- Appel avec m. Garcia pour mise en route du travail
- Création du document word pour la documentation

11.4.2 07.04.20

- Création du planning
- Mise à jour du trello en fonction du planning
- UML : Je pense qu'il va manquer quelques tables, mais je n'arrive pas à savoir lesquelles. C'est pourquoi je vais commencer le projet avec les tables actuelles et j'ajouterais les tables au fur et à mesure.



- Mise en place et branchement du raspberry PI sur mon réseau local
- Mon raspberry aura une adresse statique sur mon réseau privé afin de pouvoir tester l'application depuis différents appareils

11.4.3 08.04.20

- Mise en place des liens entre les tables de la base de données. La base de données n'est pas encore complète. Il manque principalement la structure pour le stockage des positions et formes pour l'UI style "drag & drop".
- Création de l'api

- /!\ L'API va me prendre plus de temps que ce que je pensais, je vais donc changer le Trello afin de créer des sous-tâches pour le développement de l'API.

11.4.3.1 API V1

Pour le moment mon API est encore très basique. Je n'ai que accès au employés et aux login. Afin de pouvoir accéder à la base de données, voici les logins :

- username : resa_tech_es
- password : WhutMerYmZeR6EHb

11.4.3.2 Login avec l'API (MAJ le 23.04.2020)

Afin de stocker les mot de passe en hashé dans la base de données j'utilise un système de hashage.

1. l'utilisateur entre son mot de passe
2. Le mot de passe est ensuite hashé en sha256
3. la clé (u7csu5qH6Cp9xWkrIgtGvTsOosnKvH9RhQOXteJtNhknqrEHcj8dCGYuv02SBoHGsbRoN0zGeGeToULmWUDTb2HAgNSGntNJHmg) est aussi hashée en sha256
4. On concatène les deux chaînes
5. La chaîne finale est encore une fois hashée en sha256
6. Le programme compare la chaîne hashée avec la chaîne stockée dans la base

11.4.4 09.04.20

- ~~Création de la section READ pour la table client dans l'API~~

11.4.4.1 API : READ Client

- **GetAllClients** : récupère tous les clients de la base de données
 1. api/client/get/?all
- **GetClientById** : récupère un client en fonction de son id
 1. api/client/get/?id=[l'id que l'on cherche] (ex : get/?id=1)
- **GetClientByLastname** : récupère tous les clients avec le nom de famille passé en paramètre
 1. api/client/get/?lastname=[nom que l'on cherche] (ex : get/?lastname=Pala)
- **GetClientByFirstname** : récupère tous les clients avec le prénom passé en paramètre
 1. api/client/get/?firstname=[prénom que l'on cherche] (ex : get/?firstname=Ana)

Comme toute la structure de la base de données à changer, j'ai du changer l'API (en passant à la v2) et donc les fonctionnalités ci-dessus ne sont plus d'actualités.

11.4.5 14.04.20

- Mise à jour du Trello et réponses aux remarques de M. Garcia

- Mise à jour du planning prévisionnel en incluant la mise en place de tests unitaires
- Modification de la base de données
 - Ajout de la table "is_manager". Cette table permet à un gérant d'avoir plusieurs établissements
 - Ajout des tables de scores afin que les clients et les restaurateurs puissent ce mettre des notes anonymement
- Ajout de données dans la base de données afin de pouvoir faire les tests de l'API
- Les 'username' pour les employés sont générés automatiquement et sont unique à la base de données. Le login est composé d'uniquement 4 chiffres (je ne pense pas qu'un restaurant ai plus de 9999 employés...). Ce choix est fait pour permettre un login rapide en tapant uniquement les 4 chiffres du login et le mot de passe. A voir si je ne vais pas enlever le mot de passe lorsque l'on se trouve sur le réseau du restaurant afin de pouvoir encore plus rapidement ce connecter lors du service.
- La table "is_manager" permet de stocker l'identifiant de l'employé qui est le manager. Ce manager peut avoir plusieurs restaurant et aussi être employé de ceux-ci ou d'autres (dont il n'est pas le manager).

11.4.5.1 *Comptes pour les test BDD*

11.4.5.1.1 Employés

11.4.5.1.1.1 Administrateur

- username : 2008
- password : admin (b3ab939cbbaa34ecf36b7e07bdcefce4d2c913517bd345daecab7f91c69fbe269)

11.4.5.1.1.2 Manager du Restaurant "Port Martignot"

- username : 3383
- password : manager (30963bb3ca13371a5b776434c2959c87b53113d7817a23a1f523c15863350ee7)

11.4.5.1.1.3 Employés du Restaurant

11.4.5.1.1.3.1 Olivier

- username : 5243
- password : e1 (3710f1a472febbc82026d64452ee8f1b38e801149b1c72310dc8e576b1d3b972)

11.4.5.1.1.3.2 Mathilde

- username : 9902
- password : e2 (4c4b520592f51d5456edd751d3f8d771a5c0895d65de5f8c9537804cffc32ad0)

11.4.5.2 *Diverses informations pour le restaurant*

11.4.5.2.1 Floors (étages)

11.4.5.2.1.1 La terrasse

- zone :
 1. Vue mer
 2. Vue village
- horaires :
 1. Vue mer : 11:00 - 15:00

2. Vue village : 11:00 - 15:00

11.4.5.2.1.2 La salle principale

- zone :
 1. Piste de danse
 2. Coté bar
 3. Coté fenêtre
 - horaires :
 1. Piste de danse : 11:15 - 19:30
 2. Coté bar : 11:15 - 23:30
 3. Coté fenêtre : 11:15 - 23:30
-

11.4.6 15.04.20

- Discussion avec M. Garcia
 1. Ce focaliser sur la réservation interne
 2. On est ouvert sur l'évolution de la visibilité extérieur
 3. Fusionner les tables client et employé
 4. (Tâche future) Simuler de la charge
 - Modification de la base de données
 1. Ajout de la table images
 2. Ajout de la table de liaison entre menu et images + table de liaison entre établissement et images
 3. Fusion de la table client et employé
 - Après avoir mis à jour la base de données avec l'uml. Je dois refaire l'importation dans mon serveur et mettre à jour les fichiers de l'API que j'avais déjà développé
 - Je dois également réintroduire des données dans la base. Pour le moment je ne vais que créer des utilisateurs (les mêmes que cité ci-dessus), un restaurant, quelques zones, horaires et tables. Je m'occuperais de toute la partie réservation plus tard.
 - Pour demain : mettre à jour la base de donnée du serveur
-

11.4.7 19.04.20

- La v2 de la base de données a été ajoutée au serveur
- Création des liens entre les tables

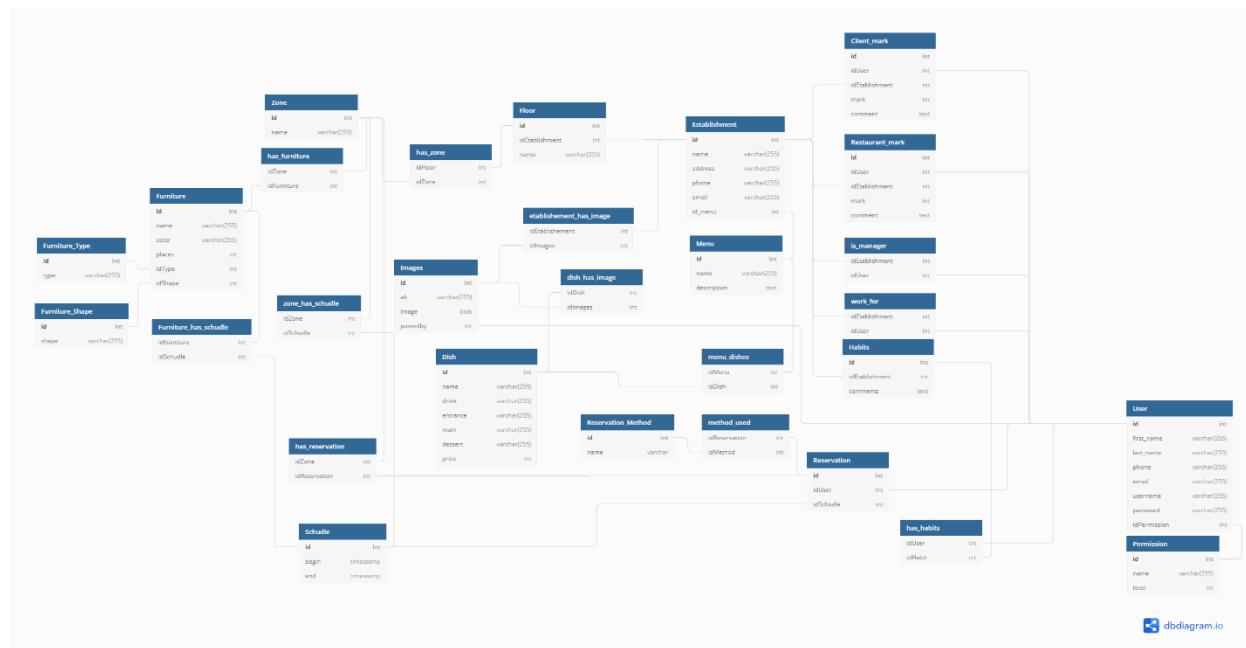
11.4.7.1 *Les niveaux de permissions*

1. Administrateur
2. Manager
3. Serveur
4. Stagiaire (Je ne sais pas si ce niveau sera réellement utile, mais on ne sait jamais)
5. Client
6. Invité

11.4.7.2 API V2

Afin de maintenir la base de données à jour avec l'api, j'ai décidé de reprendre l'api v1 et d'en faire une v2 afin de garantir le fonctionnement de la v1.

Voici le diagramme de la base de données dans la V2 de l'API :



11.4.7.3 Les changements par rapport à la v1

- Il n'y a plus de dossier "user" et "employe", car les deux ont été fonctionné (comme les tables) en user.
- Dans le dossier user, on retrouve toujours le dossier get et set (qui ne sont pas terminés à ce stade) et un dossier employe. Ce dossier va me permettre de mieux gerer la partie uniquement employés. Je pense également mettre un dossier "clients" afin de mieux gerer les clients de la base de données.

11.4.7.4 La table "user"

La table user change donc. Il y a toujours le champ (permission qui risque de changer de table car un user peut être manager d'un restaurant mais client d'un autre...)

! Du coup je viens de me rendre compte que je dois changer ma base de données... Je ne vais pas créer un v3, je vais juste supprimer les tables non nécessaires et ajoutés celles dont je vais avoir besoin.

- Je supprime donc les tables "work_for" et "is_manager" ainsi que le champ "permission_id" de la table "user"
- je vais créer une table de liaison nommée "is_in_as" comme référence à : is in établissement as ... ". Cette table va prendre 3 champs
 - L'id de l'user
 - L'id du restaurant (établissement)
 - L'id de la permission

- L'id du restaurant reste NULL si c'est un administrateur qui a tous les droits sur tous les restaurants et sur l'application en entier

11.4.7.5 Récuperer tous les employés qui sont dans la table "user"

Pour récupérer tous les user qui sont employés d'un ou de plusieurs restaurants, je dois passer par la table de liaison "is_in_for". Du coup j'ai créé une commande sql qui permet de retrouver tous les user en fonction de leur(s) permission(s).

- La commande : `SELECT `user`.`first_name`, `user`.`last_name`, `user`.`phone`, `user`.`email`, `user`.`username` FROM `user` WHERE `user`.`id` IN (SELECT `is_in_as`.`idUser` FROM `is_in_as` WHERE `is_in_as`.`idPermission` = [id de la permission que l'on cherche])`

Cette commande je vais la mettre dans le dossier "get" du dossier "user" et sera accessible comme ceci : Travail_diplome_ES_2020/RESA/api/v2/user/get/?as=[id de la permission que l'on cherche]

Je pense aussi faire la version où on peut chercher en mettant le nom. par exemple : "manager" au lieu de "2", mais je ne sais pas si c'est vraiment utile en sachant que c'est l'application qui va faire les appels à l'API et non l'utilisateur

11.4.8 20.04.20

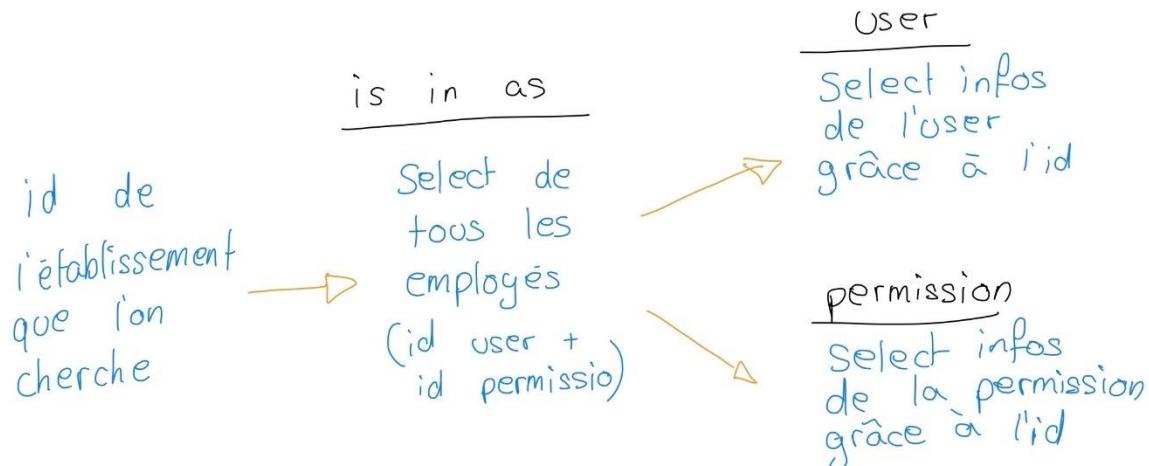
- Afin de garantir la sécurité de mes requêtes, j'ai quand même mis en place quelques mesures de sécurité. Par exemple, dans la requête ci-dessus, avant de l'exécuter, je vérifie bien que le paramètre envoyé est bien un entier afin de ne pas laisser entrer des requêtes externes.
- Avancement de l'API V2
- Avancement de la documentation en suivant les demandes faites par M. Garcia.
- Création d'un répertoire de sauvegarde dans EDUGE + ajout du lien dans la documentation et le journal de bord
- Pour trouver tous les employés d'un restaurant, qu'il soit stagiaire, serveur ou manager, je vais créer une fonction qui regroupe tous les employés uniquement en tapant l'id du restaurant.
 - La requête SQL : `SELECT `user`.`first_name`, `user`.`last_name`, `user`.`phone`, `user`.`email`, `user`.`username` FROM `user` WHERE `user`.`id` IN (SELECT `is_in_as`.`idUser` FROM `is_in_as` WHERE `is_in_as`.`idEstablishment` = [id de l'établissement])`

11.4.8.1 /!\ Problème 1

11.4.8.1.1 Problématique

Je me retrouve face à un problème : la table "is_in_as" possède l'id du user, l'id de l'établissement et l'id de la permission.

Le but est que uniquement en mettant l'id de l'établissement, on puisse trouver les informations de l'utilisateur avec ses permissions correspondantes pour l'établissement en question.



11.4.8.1.2 Solution possible

1. J'ai d'abord regarder la solution de "UNION", mais ça ne correspond pas à ce que je cherche.
2. Faire un join et ensuite un select (pas sûr)

11.4.8.1.3 Solution

La solution à donc bien été de faire des inner join (plus facile que ce que je pensais... !). La requête SQL ressemble donc à ceci :

- SELECT u.first_name as user_firstname, u.last_name as user_lastname, p.name as permission_name, p.level as permission_level FROM is_in_as as iis INNER JOIN permission as p ON p.id = iis.idPermission INNER JOIN user as u ON u.id = iis.idUser WHERE iis.idEstablishement = [id de l'établissement rechercher]

A présent, pour trouver tous les employés travaillant dans un restaurant, il suffit de rechercher l'id du restaurant désiré.

ex : /RESA/api/v2/user/employes/?workingFor=1 (1 = Port Martignot)

11.4.8.2 Etablissement

Il est possible qu'une de mes tables porte a confusion. La table "menu". En réalité cette table devrait s'appeler "carte", car la carte regroupe les différents menus que propose le restaurant.

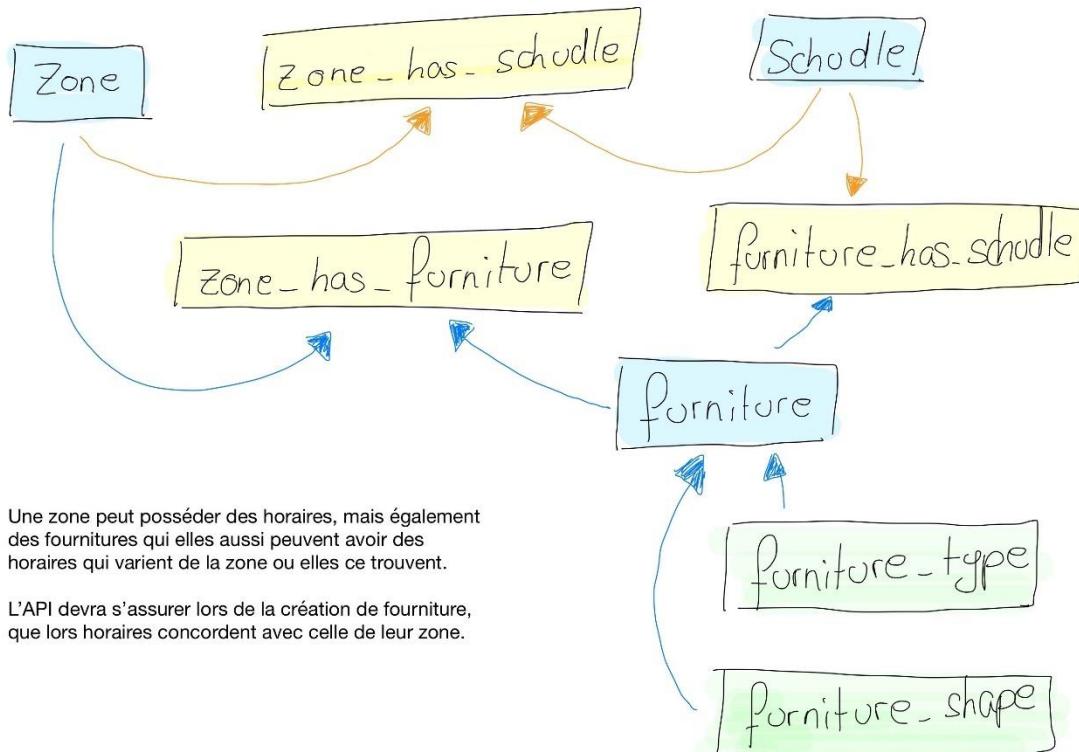
Il faudrais aussi que j'ajoute une table pour des plats à l'unité mais ce n'est pas l'option clé de cette application.

- Ajout de la fonction pour pouvoir rechercher tous les restaurants ou juste un d'après son id.

11.4.9 21.04.20

- Voici le schéma pour construire une zone dans un restaurant :

Construction d'une zone



Afin de réaliser des tests, je vais utiliser les zones (appartenant au restaurant port m.) et les horaires suivante :

1. Vue mer : 11:00 - 15:00
2. Vue village : 11:00 - 15:00
3. Piste de danse : 11:15 - 19:00
4. Côté bar : 17:00 - 23:00
5. Côté fenêtre : 11:15 - 23:30

"vue mer" et "vue village" se trouvent sur "la terrasse", alors que "piste de danse", "côté bar" et côté fenêtre" se trouvent dans la salle principale.

1. Terrasse
 1. Vue mer
 2. Vue village
2. Salle principale
 1. Piste de danse
 2. Côté bar
 3. Côté fenêtre

La table horaire contiens les données suivantes:

1. 11:00 - 15:00
2. 11:15 - 23:30
3. 11:15 - 19:00
4. 17:00 - 23:00

Afin de vérifier que les liasons soient correctes, je vais ajouter dans l'api une fonction qui permet de récupérer tous les étages avec toutes les informations les concernant d'un restaurant.

- Requête SQL pour récupérer tous les étages (floor) d'un restaurant : SELECT f.name FROM `floor` as f WHERE f.idEstablishment = 1
- Requête SQL pour récupérer tous les étages avec leurs zones : SELECT f.id, f.name, z.name FROM `floor` as f JOIN `has_zone` as hz ON hz.idFloor = f.id JOIN `zone` as z ON z.id = hz.idZone WHERE f.idEstablishment = 1
- Requête SQL pour récupérer toutes les infos sur les étages : SELECT f.id as floor_id, f.name as floor_name, z.name as zone_name, s.begin, s.end FROM `floor` as f JOIN `has_zone` as hz ON hz.idFloor = f.id JOIN `zone` as z ON z.id = hz.idZone JOIN `zone_has_schedule` as zhs ON zhs.idZone = z.id JOIN `schedule` as s ON s.id = zhs.idSchedule WHERE f.idEstablishment = 1

Je dois maintenant faire le tri dans les données que je recois, car quand j'appelle la méthode, l'API me retourne pour le moment un tableau avec plusieurs fois le même étage (parce que cet étage a plusieurs zones). Hors, moi je veux 1 étages avec un tableau de ses zones.

Voici le résultat que j'obtiens avant le tri

floor_id	floor_name	zone_name	begin	end
1	Terasse	Vue mer	11:00:00	15:00:00
1	Terasse	Vue village	11:00:00	15:00:00
2	Salle principale	Piste de danse	11:15:00	19:00:00
2	Salle principale	Côté bar	17:00:00	23:00:00
2	Salle principale	Côté fenêtre	11:15:00	23:30:00

Afin de trier les données, j'ai créé une fonction qui va automatiquement créer un tableau d'étages qui contiendra le nom de l'étage et un tableau des zones

```
// Crédit à l'auteur de ce code pour la partie de tri
// On parcourt toutes les données envoyées par la base de données
foreach ($res as $value){
    // On vérifie si le tableau est à 0 ou si la clé (l'id de l'étage) n'est pas déjà utilisée comme clé dans le tableau provisoire
    if(count($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'étages et les zones
        $floors[$value['floor_id']] = array("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['begin'], $value['end']));
}
```

11.4.10 22.04.20

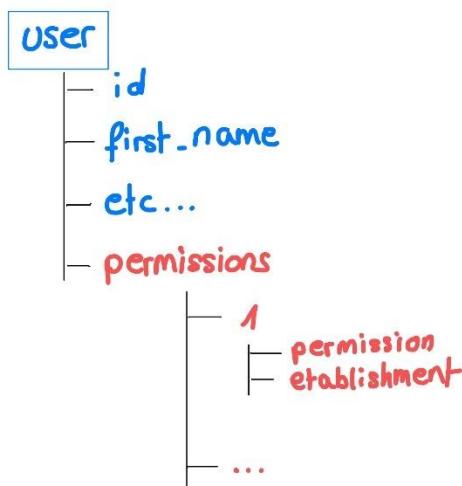
- J'ai ajouté 2 salles dans un nouvel établissement afin de tester le fait que la suppression d'un étage supprime bien les zones qui lui étaient associées

- Avant de faire le point ci-dessus, je vais d'abord faire en sorte de pouvoir créer un établissement, des étages et des zones
- Je viens de penser au fait que je devrais ajouter une table qui permet de gérer les exceptions, par exemple un jour une zone doit fermer plus tôt
- Création du "CREATE" pour les établissements
- Je souhaite récupérer toutes les permissions pour un utilisateurs. Pour ce faire je procède en 3 étapes :
 1. Je récupère les infos de l'utilisateur
 - `SELECT `id`, `first_name`, `last_name`, `phone`, `email`, `username` FROM `user` WHERE `id` = [id de l'utilisateur]`
 2. Je récupère toutes les permissions qui sont accordées à l'utilisateur
 - `SELECT IFNULL(e.name, "-") establishment_name, IFNULL(p.name, "-") as permission_name FROM user as u LEFT JOIN is_in_as as iia ON iia.idUser = u.id LEFT JOIN establishment as e ON e.id = iia.idEstablishment LEFT JOIN permission as p ON p.id = iia.idPermission WHERE u.id = [id de l'utilisateur]`

11.4.10.1 Récupérer le menus d'un restaurant

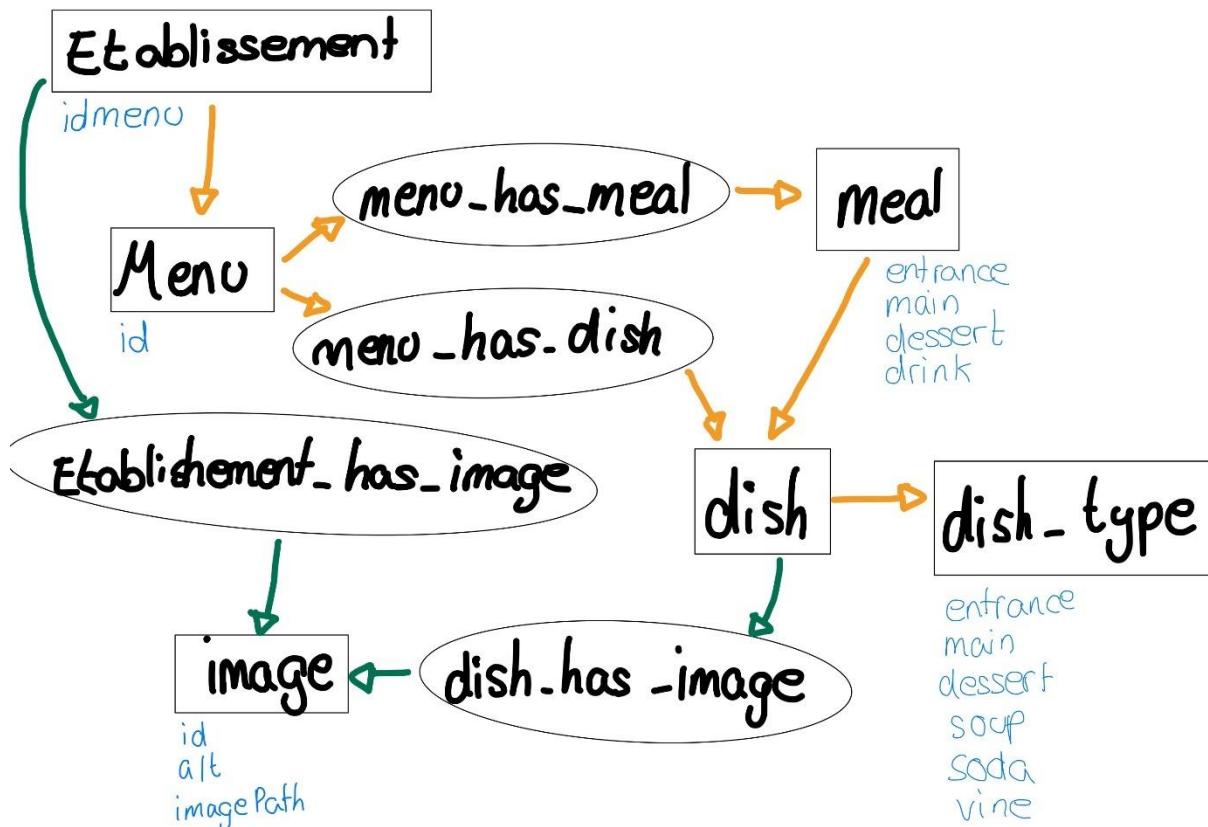
Afin de pouvoir récupérer tous les plats et compositions dans le menu d'un restauruant je dois procéder étape par étape.

1. Récupérer le menu du restaurant :
 - `SELECT m.id, m.name, m.description FROM `menu` as m WHERE m.id IN (SELECT e.id_menu FROM `establishment` as e WHERE e.id = [id du restaurant])`
2. Récuperer tous les plats qui ce trouvent dans ce menu
 - `SELECT d.name as dish_name, d.price as dish_price, dt.name as type_name FROM `menu` as m INNER JOIN `menu_has_dishes` as mhd ON mhd.idMenu = m.id INNER JOIN `dish` as d ON d.id = mhd.idDish INNER JOIN `dish_type` as dt ON dt.id = d.idType WHERE m.id IN (SELECT e.id_menu FROM `establishment` as e WHERE e.id = [id du restaurant])`
3. La prmière requête me donne un tableau avec les données de l'utilisateur et la deuxième me rend un tableau de tableau avec les permissions pour chaque établissement. J'ajoute donc un index nommé "permissions" dans le tableau de l'utilisateur ce qui donne ceci



:

- Ajout de la table meal et de la table menu_has_meal
- La base de données gère comme ceci les menus des restaurants :



- J'ai ajouté des données dans la base afin de pouvoir faire des tests. (Pour nouveau avoir le même résultat qu'avec les étages, je vais devoir reprendre la fonction ci-dessus et l'ajuster pour les menus)

Pour récupérer les menus composés, il faut exécuter une autre requête SQL

- SELECT ml.id as meal_id, ml.name as meal_name, entrance.name as entrance_name, main.name as main_name, dessert.name as dessert_name, IFNULL(drink.name, "-") as drink_name, ml.price as price FROM `menu_has_meal` as mhm INNER JOIN `meal` as ml ON ml.id = mhm.idMeal INNER JOIN `dish` as entrance ON ml.entrance = entrance.id INNER JOIN `dish` as main ON ml.main = main.id INNER JOIN `dish` as dessert ON ml.dessert = dessert.id LEFT JOIN `dish` as drink ON ml.drink = drink.id INNER JOIN `menu` as menu ON menu.id = mhm.idMenu WHERE mhm.idMenu IN (SELECT e.id_menu FROM establishment as e WHERE e.id = 2)
- Résultat de la requête SQL :

	meal_name	entrance_name	main_name	dessert_name	drink_name	price
1	La découverte	Salade verte	Tartare de Saumon (Servi avec Frites et Toast)	Tarte au citron	-	59
2	La joie	Carpaccio de Saumon	Steack de boeuf	Mousse au chocolat	Vin blanc de Genève	0

11.4.10.2 Unire les 2 résultats de recherche

Le but à présent est d'unir les 2 résultats. J'ai créer une fonction qui fait appel des deux fonctions ci-dessus et qui créer un tableau suivant :

- Menu
 - infos
 - id
 - name
 - description
 - dishes
 - Tableau des plats
 - Meals
 - Tableau des menus composés

11.4.10.3 Appel avec M. Garcia

- Ajouter un état "non réservable" sur les fournitures
- Utiliser "distinct" ou "group by" pour la fonction de recherche de étages

11.4.10.4 Test de "distinc" et de "group by"

11.4.10.4.1 Group by

- J'ai ajouté le paramètre suivant à ma requête SQL : GROUP BY f.id ou f.id = floor id. Mais SQL me donnais une erreur comme quoi je devais trier avec tous les champs.
- J'ai donc transformer le group by comme ceci : GROUP BY f.id, z.name, s.begin, s.end. Le résultat était exactement le même que sans rien...

11.4.10.4.2 Distinc

Le paramètre "distinc" permet de retourner uniquement des valeurs différentes. Le problème est que tous mes champs sont déjà différents, car chaque zone n'est attribuée que à un étage.

11.4.10.4.3 Conclusion

Les deux outils sont puissants et vont certainement êtres utiles dans un futur proche, mais pour cette fonction, ils ne sont malheureusement pas adaptés à ce que je veux faire.

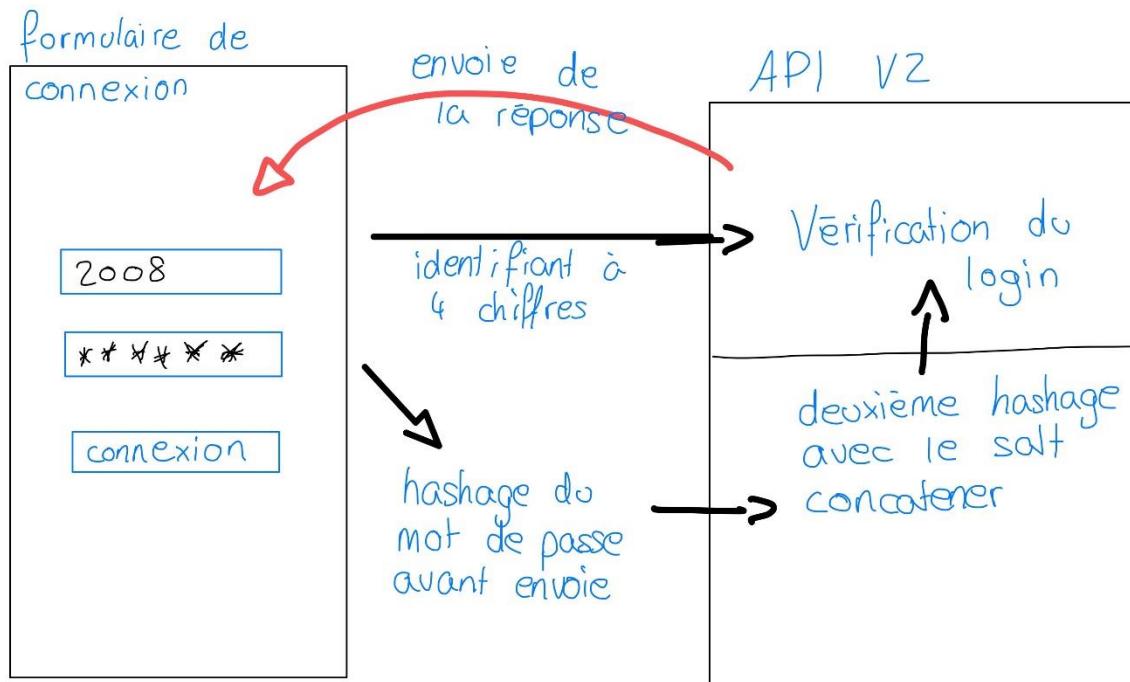
Le développement de l'API va me prendre beaucoup plus de temps que prévu... je vais devoir faire des choix sur les fonctionnalités à ajouter... A suivre

11.4.11 23.04.20

L'objectif du jour est de continuer un maximum l'API. Et de commencer 2 - 3 vues qui récupèrent les restaurants et leurs menus.

La complexité de mon travail de diplôme est la gestion de réservation en fonction des horaires des zones. Pour le moment je ne me suis pas encore penché sur le problème, car l'API de base, me prend du temps. Je vais revoir mon planning.

- Géré le Login d'un utilisateur
 - J'utilise la même méthode que dans la V1 de l'API



- Création d'un cheat sheet pour l'API afin de rendre son utilisation plus facile par un externe
 - Le but de ce cheat sheet est de faciliter l'utilisation de l'API aux personnes qui aurons besoin de l'utiliser afin de continuer l'application ou pour une autre utilisation.
 - Ca me permet également de retrouver où et comment récupérer un certain type de données que je cherche.
- Au début je pensais commencer des vues, mais le cheat sheet me prend pas mal de temps, je commencerais donc les premières vues demain.

11.4.11.1 APPEL M. Garcia

- Regarder pour convention / norme requête SQL

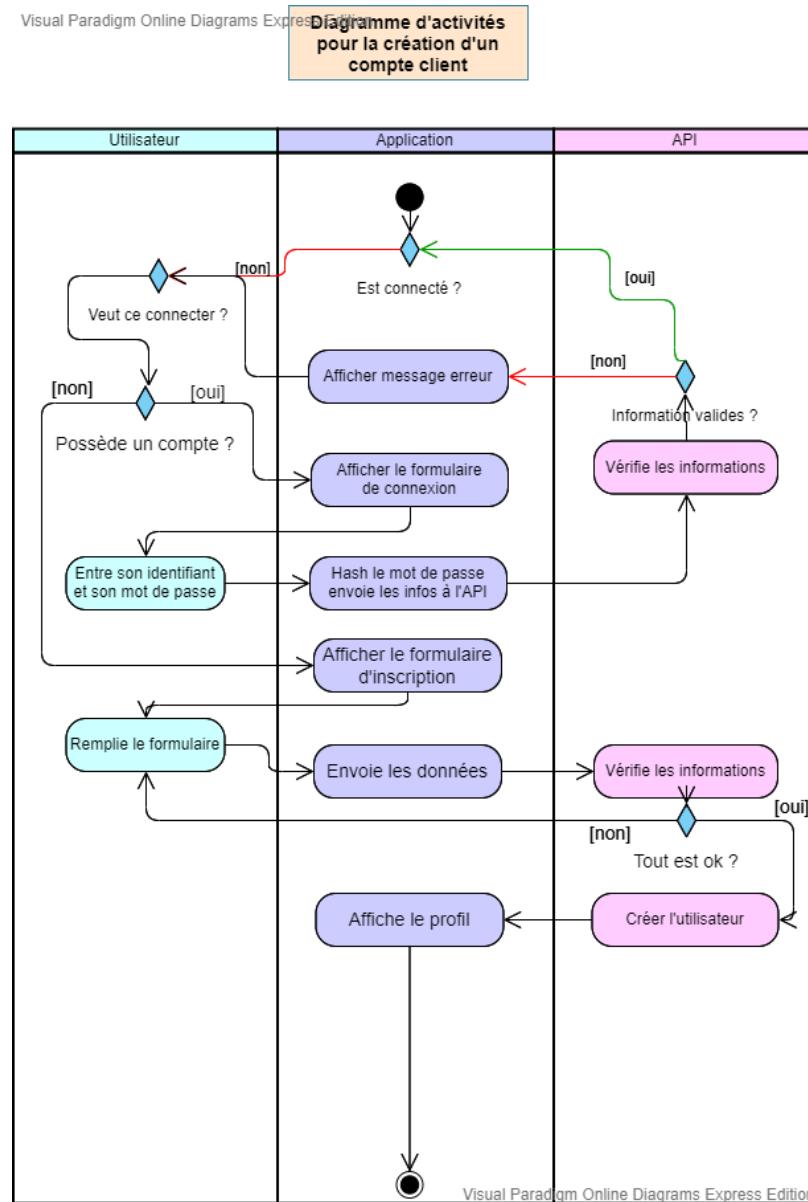
11.4.11.2 Bonnes pratiques SQL

Afin d'unifier mon code avec les bonnes pratiques, je me suis référé au [site](#) que m. Garcia m'as fait parvenir.

11.4.12 24.04.20

CES DIAGRAMMES d'ACTIVITES NE SONT PLUS A JOUR ! VOIR PLUS BAS ...

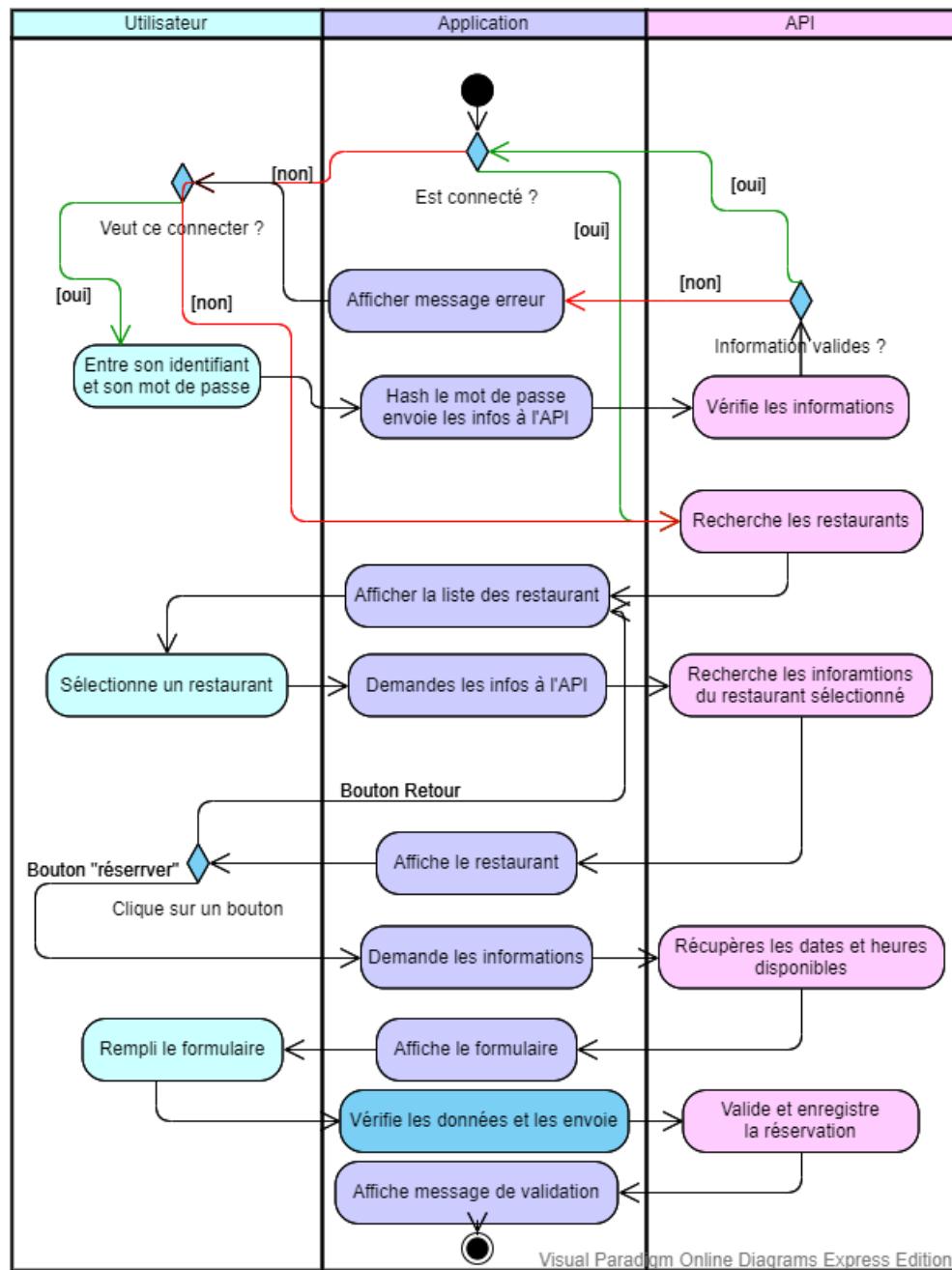
- Création du diagramme d'activité pour la création d'un compte client dans l'application :



- Création du diagramme d'activité pour faire une réservation dans un restaurant :

Visual Paradigm Online Diagrams Express Edition

Diagramme d'activités pour la réservation faite par un client



- Objectif actuellement :
 - Faire les vues et le script qui tourne derrière afin de déjà faire fonctionner les diagrammes ci-dessus
 - Mettre à jour la documentation avec les demandes de m. Garcia : **OK**

- Mettre en place le système des images afin d'avoir de belles vues : **OK [terminé le 24.04 à 23:53]**
- Afin de pouvoir mettre les images dans la base de données avec un uniqID, je dois créer une page provisoire pour uploader les photos sur le serveur
 - Je vais donc créer une page php avec une form simple qui me permet de selectionner un des plats de la base et la photo à mettre en ligne
 - La page est accessible via : /api/v2/images/upload/debug/
 - J'ai vite du ajouter une fonction dans l'API, une fonction qui me permet de récupérer tous les plats de la base
 - J'ai donc créer un formulaire qui permet de sélectionner le restaurant ou le repas auquel on souhaite ajouter une photo.
 - Il faut maintenant que je fasse la partie de l'API qui permet d'ajouter le lien entre la photo et le plat ou l'établissement
 - J'ai créer la partie de l'api qui me permet de mettre en ligne un photo avec un uniqid. (Ca m'as pris un temps fou)
 - L'upload est totalement terminé ! /api/v2/images/upload/
 - Pour utiliser ce fichier, il faut l'include dans celui qui va envoyer l'image (Plus d'explication dans le Cheat sheet de l'API v2)
 - Il est possible de récupérer les informations d'une image d'après son id et le paramètre "data". /api/v2/images/get/?path&id=[id de l'image]
 - Il est possible d'être directement rediriger sur la photo en indiquant uniquement son id : /api/v2/images/get/?id=[id de l'image]
 - Il est possible de récupérer toutes les images d'un établissement !
 - Je suis en train d'ajouter la requête pour récupérer toutes les photos d'un plat mais je me dis que je devrais aussi faire en sorte que l'on puisse récupérer toutes les images d'un menu entier d'un restaurant. A retenir

11.4.12.1 *Evaluation intermédiaire (appel avec m. Garcia)*

11.4.12.2 Documentation

- Ajouter le numéro de version sur la page de garde : **OK**
- Titre de la table des matières : **OK**
- Mettre des numéros devant les titres : **OK**
- Ajouter le mot "page" devant le numéro de page + "/" et le numéro de pages totaux du document : **OK**
- Créer un diagramme d'activités (par qui ça passe) -> J'en ai fait 2, ce sont les deux avec lesquels je vais commencer les vues

11.4.12.3 Github

- Penser à supprimer les mot de passes

11.4.12.4 Google Drive

- (Ajout la synchronisation avec google Drive)
- Faire des backups QUOTIDIEN et non tous les 2 jours
- Règle de nommage : année_mois_jour(_heure_minute_secondes)_[nom fichier].extension

11.4.12.5 Serveur

- J'aurais du utiliser WSL au lieu de laragon

11.4.12.6 Login

- Au lieu d'utiliser des cartes RFID

11.4.12.7 Images

- Utiliser la fonction unique ID de PHP pour renommer les images mises en ligne
- <https://www.php.net/manual/fr/function.uniqid.php>

11.4.12.8 Poster

- Expert ont de la peine à ce retrouver -> faire un diagramme physique

11.4.13 26.04.20

- Suite de la documentation, commencement de la partie de la gestion des images.
- J'ai commencer la structure de l'API dans la documentation, mais je ne vais pas la terminer car celle-ci risque de changer.

11.4.13.1 Page Login

- Création de la page de login à partir du template
 - Nettoyage de la page et des composants non-nécessaires

11.4.13.2 Idées pour plus tard afin de encore plus me faciliter la vie

- Ajout d'un nouveau fichier dans le dossier images de l'API. Ce fichier va permettre de directement envoyer le résultat d'un forms directement à ce fichier afin qu'il puisse ajouter la photo dans la base de données.

11.4.14 27.04.20

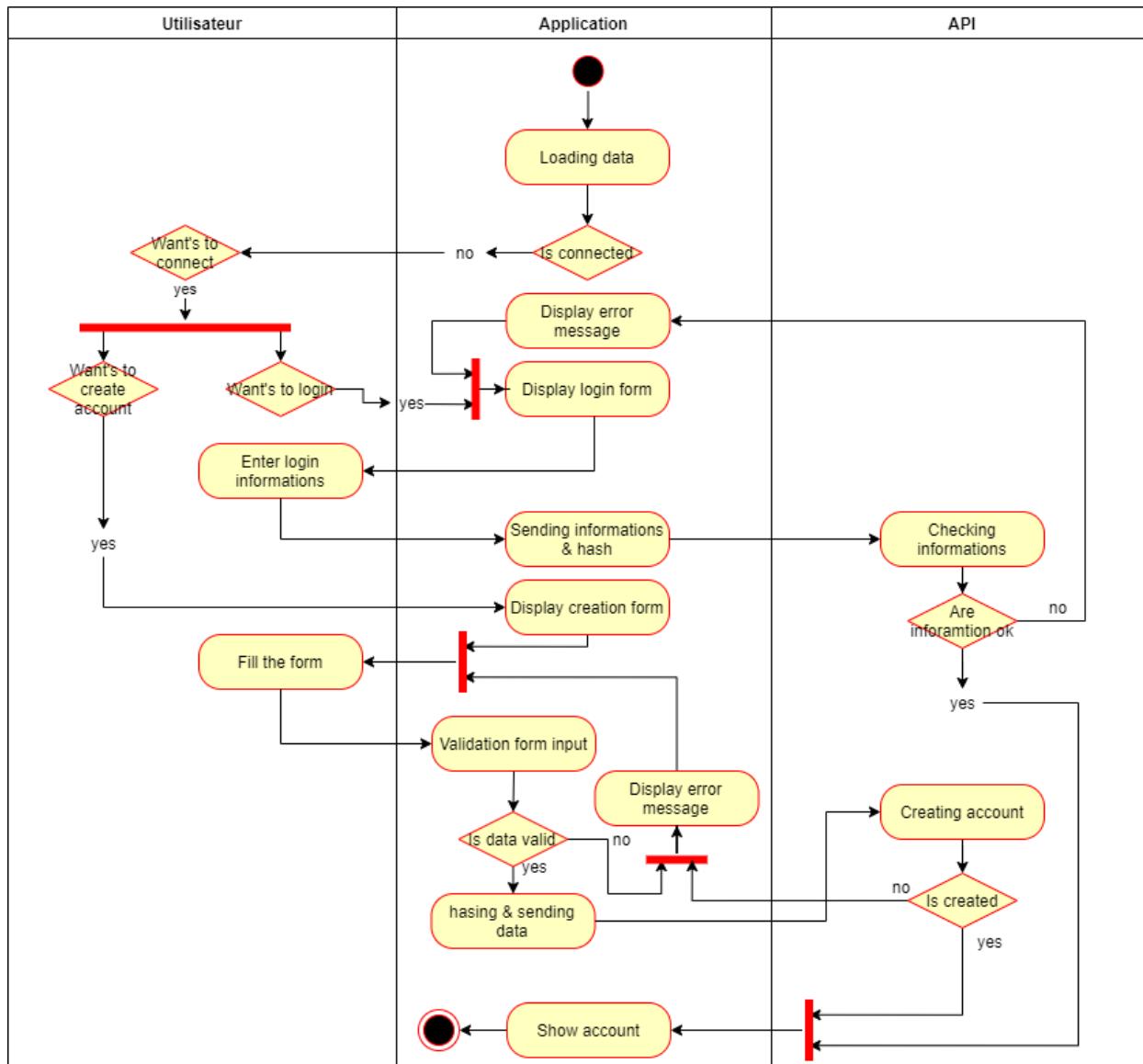
11.4.14.1 Appel avec m. Garcia

- Voir si Laraval et iyy peut être plus efficace
- J'ai effectuer des recherches pour voir si Laravel ou iyy allaient pouvoir me faciliter la tâche pour mon travail. Malheureusement rien ne correspond à ce dont j'ai réellement besoin, car mes requêtes sont tellement spécifiques que un générateur de code ne me sert à rien.

11.4.15 28.04.20

- Création du fichier des tâches à faire pour l'API

- Je vais reprendre mes diagrammes d'activités
 - Il faut que je comprenne d'abord, comment réaliser correctement un diagramme d'activité.
 - Je vais utiliser le lien ci-dessus comme modèle pour les normes que j'utilise.
- J'ai un peu continué la documentation
- Création du diagramme d'activité pour la connexion et création de compte :



Après avoir pris 2 jours de recul sur mon projet, j'ai pu me rendre compte de ce que j'avais réellement à faire. Je vais donc faire un nouveau tableau des tâches à faire (dans l'ordre) et des priorités. Comme c'est un projet qui demande beaucoup de temps et de tables, je dois faire la part des choses.

- Croquis des choses à faire dans un premier temps :

- Choses à faire maintenant
- Gestion d'un login / création de compte
la page est créée, mais pas la logique
- Affichage des restaurants
→ Affichage de leurs horaires
- Réservation
→ création réservation par client
→ Affichage réservation restaurant + client

- Mise au propre des choses à faire :



M. Garcia valide ma nouvelle approche.

11.4.15.1 Appel avec m. Garcia

- Utilisation de DrawIO
 - Permet de sauver directement dans le Google Drive
 - Produit directement les UML
- Trouver des normes pour les diagrammes d'activités : <https://sourcemaking.com/uml/modeling-business-systems/external-view/activity-diagrams>
- Inconsistances dans les diagrammes
 - S'inspirer des normes
 - Revoir mes diagrammes

11.4.16 29.04.20

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

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

- Il faut que j'ajoute une fonction de login avec l'email
- (je ne sais pas si je dois aussi hasher le username et l'email ou non ... Pour le moment je ne vais pas le faire car les emails sont public...)
- Ajout du fond d'écran de tous les profils utilisateurs
 - Accessible à partir de l'API
- Il faut que je fasse un table de liaison entre les utilisateurs et une photo de profil
 - La table de liaison est terminée
 - Ajout d'une image test grâce au debug
- Il faut mettre à jour l'API pour récupérer l'image l'utilisateur
 - C'est fait ! Pour accéder à la photo, il faut récupérer le chemin de l'image sur ce lien : [http://localhost/Travail_diplome_ES_2020/RESA/api/v2/images/get/?user&id=\[id de l'utilisateur\]](http://localhost/Travail_diplome_ES_2020/RESA/api/v2/images/get/?user&id=[id de l'utilisateur])

Malheureusement aujourd'hui je n'ai pas fait tout ce que je voulais... Je n'ai fait que le login avec l'affichage de l'utilisateur connecté. Demain j'affiche tous les restaurants !

11.4.16.1 Requête de login employé

Vérifie uniquement si l'utilisateur travail bien pour l'établissement : SELECT u.id as idUser, u.first_name as firstnameUser, u.last_name as lastnameUser, u.phone as phoneUser, u.email as emailUser, p.name as namePermission, p.level as levelPermission, IFNULL(e.id, "-") as idEstablishment, IFNULL(e.name, "-") as nameEstablishment FROM `is_in_as` as iia INNER JOIN `permission` as p ON p.id = iia.idPermission LEFT JOIN `establishment` as e ON e.id = iia.idEstablishment INNER JOIN `user` as u ON u.id = iia.idUser WHERE iia.idUser IN (SELECT `id` FROM `user` WHERE `username` = '[numéro d'identification]' AND `password` = '[mot de passe hashé]') AND iia.idEstablishment = [id de l'établissement]

Retourne le nom de l'utilisateur si l'identifiant d'utilisateur et le mot de passe existent bien dans l'établissement ou est membre de l'administration :
 SELECT u.id as idUser, u.first_name as firstnameUser, u.last_name as lastnameUser, u.phone as phoneUser, u.email as emailUser, p.name as namePermission, p.level as levelPermission, IFNULL(e.id, "-") as idEstablishment, IFNULL(e.name, "-") as nameEstablishment FROM `is_in_as` as iia INNER JOIN `permission` as p ON p.id = iia.idPermission LEFT JOIN `establishment` as e ON e.id = iia.idEstablishment INNER JOIN `user` as u ON u.id = iia.idUser WHERE iia.idUser IN (SELECT `id` FROM `user` WHERE `username` = '[numéro d'identification]' AND `password` = '[mot de passe hashé]') AND (iia.idEstablishment = [id de l'établissement] OR iia.idPermission = 1)

La requête ci-dessus ne me sert pas dans l'imédiat, mais au moins elle est prête !

11.4.16.2 *Création de la page de profil*

- Reprise du template pour créer la page de profil de l'utilisateur
- Il faut maintenant que je récupère les données de la session afin de vérifier que l'utilisateur soit bien connecté et qu'il puisse bien avoir ces informations

11.4.16.3 *Appel avec m. Garcia*

- Chercher un outil qui intègre des données dans la base
 - Il va contacter un ancien élève qui avait utilisé ça. En attendant de ses nouvelles, je vais continuer mon programme.
- Liens potentiels pour le remplissage automatique de données :
 - <http://www.generatedata.com/?lang=fr#t1>
 - <https://github.com/benkeen/generatedata>

11.4.17 30.04.20

- Finitions de la page des établissements
- Ajout d'établissements dans la base afin de tester la mise en page
 - J'ai fait exprès de ne pas mettre de photo pour un restaurant afin que celui-ci affiche l'image par défaut
- Je vais faire la page qui affiche le restaurant afin de pouvoir commencer à faire la logique des restaurants
- J'ai créé dans la page de l'utilisateur un panel qui contient 3 tabs
 - Le premier permettra de voir les dernières ou futures réservations
 - Le deuxième permettre de changer la photo de profil (C'est peut-être la que je vais utiliser le Dropzone.js [REF : ci-dessous])
 - Le dernier permet de créer facilement un établissement
 - J'ai du mettre en place le formulaire avec les tests pour vérifier que les champs étaient bien valides
 - J'ai du ajouter une partie à l'API

11.4.17.1 *Création d'un nouvel établissement*

Voici comment je crée la query que j'envoie ensuite pour la création d'un nouvel établissement

\$queryData = array(

```
'name' => $data['name'],
'address' => $data['adress'],
'phone' => $data['phone'],
'email' => $data['email'],
'creatorID' => $creatorID
);

$link1 = $path."establishment/create/?".http_build_query($queryData);

file_get_contents($link1);
```

- J'ai du faire beaucoup de changements du côté de l'API dans les images
 - SavelImageDish ainsi que `SavelImageUser` ne fonctionnent plus actuellement, mais ce ne sont pas des priorités pour le moment

La deuxième étape, est de faire un upload des images sur le serveur... un peu plus compliqué.

- ~~Je viens de voir un problème que j'avais dans ma logique d'enregistrer mes images... malheureusement je les enregistrait dans mon API, hors je dois les enregistrer sur le site web et uniquement envoyer le lien à l'API~~

11.4.17.2 MAJ API

- Ajout de la fonctionnalité "is in as" qui permet de directement mettre le créateur de l'établissement en temps que manager
- Cette fonction peut être utilisée pour tout type d'ajout (pas seulement manager)
- Afin que je puisse directement assigné le créateur en temps que manager, je dois récupérer le dernier ID introduit dans la base de données.
- Il faut que je finisse la méthode is in as car je l'ai oubliée ...
 - C'est fait ! J'ai perdu énormément de temps car j'avais inversé 2 champs (tristesse)
- Il faut que j'ajoute aussi un fonction qui me permet de récupérer tous les restaurants d'un manager
 - D'abord je récupère tous les établissements ou travaille le manager (à garder, elle pourra me servir plus tard)
 - SQL : SELECT e.id, e.name FROM `is_in_as` as iia INNER JOIN `establishment` as e ON e.id = iia.idEstablishement WHERE iia.idUser = [id du manager]
 - J'ajoute juste le AND à la fin avec l'id des manager (2)
 - SQL : SELECT e.id, e.name FROM `is_in_as` as iia INNER JOIN `establishment` as e ON e.id = iia.idEstablishement WHERE iia.idUser = [id du manager] AND iia.idPermission = 2

Dropzone.js

- ~~Afin de pouvoir autoriser le drag & drop de photos dans mes formulaires, j'ai utilisé un librairie standalone. Cette librairie s'appelle dropzone.js~~
- Malheureusement ça ne fonctionnais pas comme je le pensais et ça n'est pas compatible avec mon utilisation... Je ne vais pas m'attarder plus sur ça aujourd'hui. Si jamais j'ai à nouveau besoin, voici les 2 includes :
 - <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/dropzone/5.5.1/dropzone.css" />
 - <script src="https://cdnjs.cloudflare.com/ajax/libs/dropzone/5.5.1/dropzone.js"></script>

11.4.17.3 Appel avec m. Garcia

- Les utilisateurs doivent être archivés
 - Afin de garder l'historique
- Trouver un outil qui fait toute l'arboration de l'API
 - La commande tree dans le cmd

11.4.18 Programme pour demain (avant que j'oublie)

- Afficher la liste de tous les établissements appartenant à un utilisateur
 - Afficher la liste de tous les établissements où travaille l'utilisateur
 - Mettre à jour le Cheat Sheet pour l'API
 - Mettre à jour les commentaires de l'API
 - Continuer A FOND la documentation
-

11.4.19 03.05.20

Finalement ce n'était pas le lendemain mais dimanche...

- Afin de pouvoir afficher tous les établissements dont un utilisateur est le manager, il faut d'abord que j'ajoute des fonctionnalités à l'API :
 1. Récupérer tous les établissements dont il est le manager
 - Afin de pouvoir faire ça, je dois faire une requête qui récupère tous les établissements où l'utilisateur est membre du staff et qu'il a l'id du manager
 - SQL : SELECT e.id, e.name FROM `is_in_as` AS iia INNER JOIN `establishment` AS e ON e.id = iia.idEstablishment WHERE iia.idUser = [id de l'utilisateur] AND iia.idPermission = [id de la permission]
 - OK, je récupère tous les établissements dont l'utilisateur est le manager
 2. Créer la page d'administration de l'établissement
 - Il faut que je créer à partir des composants du template une page qui affiche tous les menus importants pour la gestion du restaurant

Voici les objectifs de la semaine du 04.05 au 08.05

Nouveaux objectifs

- Créer la page de management de l'établissement
- Continuer à fond la doc
- Réaliser le poster

En détails les objectifs de la documentation, elle doit être prête pour l'évaluation intermédiaire pour vendredi

Documentation

- Faire les résumés (français + Anglais)
- Parler des tables et de leurs liens
- Parler des ajouts possibles
- Compléter le schéma de l'api
- Ajouter du contenu
 - ↳ Analyse de la concurrence
 - ↳ Analyse des priorités
 - ↳ Analyse du planning

Les objectifs pour le poster

Poster

→ Trouver une mise en page vendue

→ ! Comme le projet ressemble énormément à la fourchette verte, il faut que j'arrive à vendre mon projet

→ Faire des croquis

→ Demander à des personnes leurs points de vues.

11.4.20 04.05.20

- J'ai continuer la page de gestion de l'établissement
 - J'ai un problème avec les liens dans la navbar
 - Quand je met un lien générer en php, ça change le CSS du lien
 - Je laisse ça de côté
- Affichage des boutons pour ajouter des zones à un étage
 - Création du formulaire
 - il faut que je fasse la partie de l'API maintenant
 - SQL (assez basique): INSERT INTO `zone`(`name`) VALUES ('[nom de la zone]')
 - Il faut maintenant faire la logique dans l'API
 - Il faut faire le lien entre l'étage et la zone que je veux lier
 - Je créer donc une fonction qui va ajouter la nouvelle zone à l'étage
 - Il est possible de créer une zone pour un étage !
 - Il faut maintenant faire le formulaire de création de l'étage
 - Comme pour la zone, je vais faire un bouton qui permet de faire apparaître le formulaire de la création de l'étage
- Il est possible de créer des étages et de créer des zones qui vont dans ces étages. Maintenant il faut que je créer les formulaires qui permettent de mettre à jour ces zones en y insérant des horaires et des fournitures
- Pour les horaires et les fournitures je vais devoir créer les vues et les API

- Il faut que je récupère tous les horaires pour une zone
 - SQL : SELECT s.id as schedule_id, s.begin as begin, s.end as end FROM `zone_has_schedule` as zhs INNER JOIN `schedule` as s ON s.id = zhs.idSchedule WHERE zhs.idZone = [id de la zone]
 - Il est possible d'afficher les horaires d'une zone en cliquant sur le bouton "horaires"
- Il faut ajouter une table qui affiche les fournitures de la zone, je vais donc utiliser l'application que m. Garcia m'avais envoyé pour générer des fournitures
 - J'ai générer 100 meubles
 - Ils ont tous un nom aléatoire (d'un être humain haha)
 - ils sont tous bleus
 - Ils ont tous une forme et un type différent
- Pour récupérer tous les fournitures d'une zone :
 - SQL : SELECT f.id as furniture_id, f.name as furniture_name, f.color as furniture_color, f.places as furniture_places, ft.id as type_id, ft.type as type_name, fs.shape as shape_name FROM `has_furniture` as hf INNER JOIN `furniture` as f ON f.id = hf.idFurniture INNER JOIN `furniture_type` as ft ON ft.id = f.idType INNER JOIN `furniture_shape` as fs ON fs.id = f.idShape WHERE hf.idZone = [id de la zone]
- Compter le nombre de places totales dans une zone :
 - SQL : SELECT SUM(f.places) as places_totales FROM `has_furniture` as hf INNER JOIN `furniture` as f ON f.id = hf.idFurniture WHERE hf.idZone = [id de la zone]

11.4.20.1 Appel avec m. Garcia

- Il faut absolument continuer la documentation pour jeudi 17h
-

11.4.21 05.05.20

- Téléchargement de Photoshop pour le poster
 - Création des maquettes du poster
 - Esquisse 1 :
 - Esquisse 2 :
- Avancement de la documentation
 - Ajout de pleins de chapitres et de sections

11.4.21.1 Appel avec m. Garcia

- Ajouter un point d'ancrage X et Y pour les zones et fournitures
 - limiter la taille du restaurant
 - taille maximale par zone
 - Révision de la documentation demain matin
 - Ajouter un paragraphe pour expliquer la zone
-

11.4.22 06.05.20

- Finalisation du poster avec les commentaires de m. Garcia
- Créations de divers schémas de fonctionnement afin de mieux comprendre l'application.

11.4.23 Crédit poster

11.4.24 Appel avec m. Garcia

- Mettre le texte en justifier
- Chose essentiel qui manque :
 - Ajouter des schémas de fonctionnement
 - Ou est l'application ?
 - Qu'est-ce qu'elle fait ?
 - Logiciel utilisés
- Comment détecter que c'est une image qui est envoyée
- Pouvoir envoyer un rollback pour confirmer que tout a bien été fait

11.4.25 07.05.20

- Finalisation de la documentation pour l'évaluation intermédiaire

11.4.25.1 Appel avec m. Garcia

- Faire aussi une analyse sur le lien que m. Garcia
- Ajouter des légendes aux codes
- Documentation à la programmation manquante
 - Parler de ce que je reçois en JSON
 - Zoomer sur le fond du code
- Pour le MCD, entourer les parties user, reservations, etc.

11.4.26 08.05.20

11.4.26.1 Appel avec m. Garcia

- Créer des renvois pour le schéma du login
- Mettre un schéma pour le json
- Renvoie vers les paragraphes pour les utilisateurs
- Ajouter le hashage dans la documentation
- Parler des versions de l'API (faire évoluer)
- Ce renseigner sur l'unqid (13 digits)

- Archiver les utilisateurs et les établissements au lieu de les supprimer

11.4.26.2 Application

- Ajouter les dates sur le widget des réservations
- Ajouter un calendrier des réservations

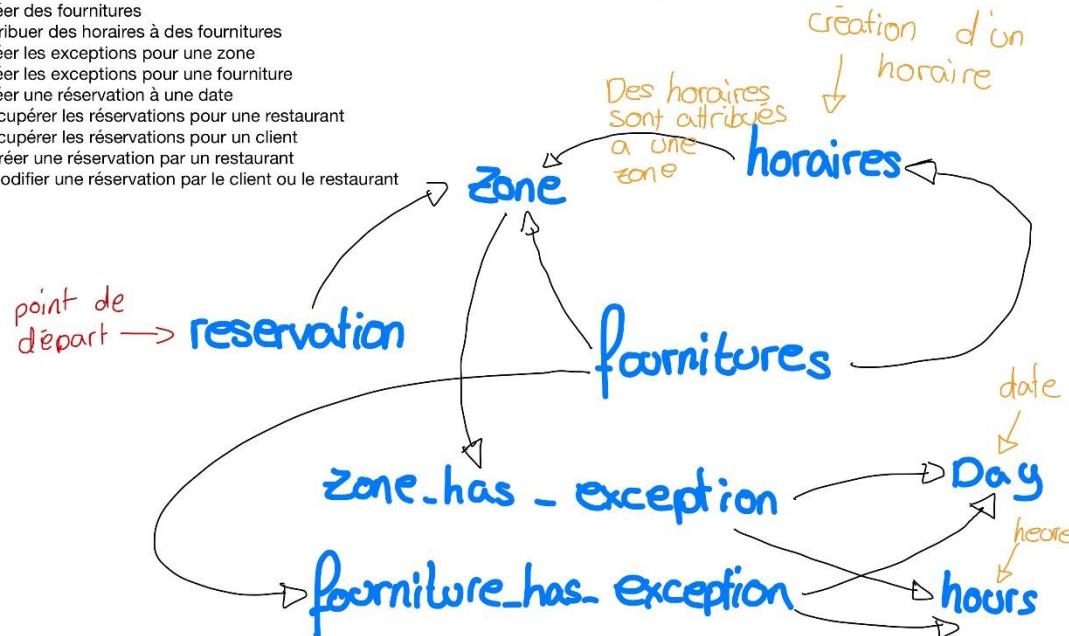
11.4.27 12.05.20

Objectifs de la semaine:

- Ajouter des horaires dans les zones
- Créer des fournitures
- Ajouter des horaires au fournitures
- Récupérer les réservations pour le restaurant
- Récupérer les réservations pour le client
- Créer une réservation dans un établissement
 - Il faut prendre en compte les horaires des zones et de l'établissement
 - Il faut prendre en compte les crénaux qui ne sont pas encore réservés afin de proposer une réservation au client
- Créer d'autres schéma de fonctionnement
- Créer d'autres diagrammes d'activités

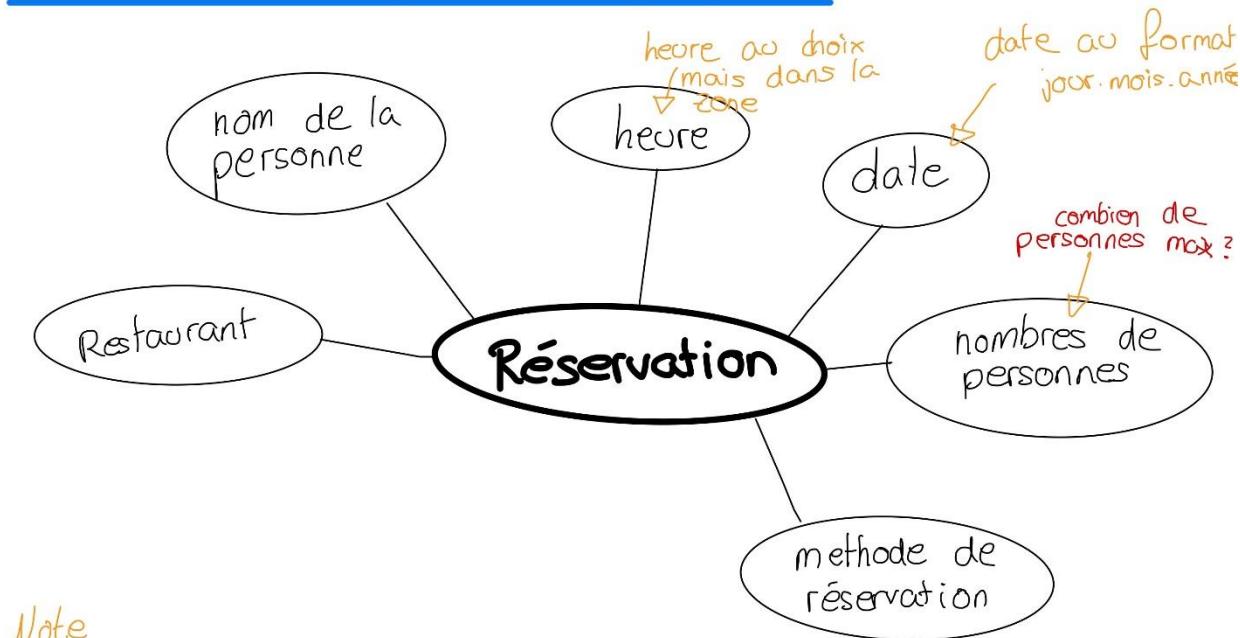
Liens entre les tâches et Liste des tâches à faire

1. Créer des horaires
2. Attribuer des horaires aux zones
3. Créer des fournitures
4. Attribuer des horaires à des fournitures
5. Créer les exceptions pour une zone
6. Créer les exceptions pour une fourniture
7. Créer une réservation à une date
8. Récupérer les réservations pour une restaurant
9. Récupérer les réservations pour un client
10. Créer une réservation par un restaurant
11. Modifier une réservation par le client ou le restaurant



Il faut d'abord que je définisse ce qu'est réellement une réservation :

une réservation, c'est quoi ?



11.4.27.1 Les horaires

Comme on peut le voir sur la liste de tâches, la première chose à faire est de créer une page d'administration où l'on peut créer des horaires ou récupérer ces proposés par les autres établissements.

- Une fois un horaire créé, il est impossible pour un utilisateur standard de le supprimer, car l'horaire peut être utilisé par un autre restaurant
- Création d'un horaire :
- SQL : `INSERT INTO `schudle`(`begin`, `end`) VALUES ('[heure de début]', '[heure de fin]')`

11.4.27.2 Page admin

Création de la page d'administration générale :

- Ensuite, je vais mettre des widgets sur la page de l'établissement du manager, ça sera mieux pour l'UI

11.4.27.3 Zones et fournitures

- Il faut que je récupère toutes les places dans une zone

- SQL SELECT IFNULL(SUM(f.places), 0) as "places_total" FROM `has_furniture` as hf INNER JOIN furniture as f ON f.id = hf.idFurniture WHERE hf.idZone = [id de la zone]
- La page d'administration affiche le nombre de places totales dans une zone
- Il faut ajouter une page de gestion des fournitures pour les zones, c'est à dire une page où l'on peut voir, modifier ou supprimer les fournitures de la zone. On doit également pouvoir changer les horaires de ceux-ci
 - Je vais ajouter sur la page d'administration un bouton pour chaque zone qui affiche les fournitures de cette dernière
 - Le bouton va créer et j'arrive à ouvrir un modal
 - Il faut maintenant que je récupère les fournitures de la zone
 - SQL : SELECT f.id as fid, f.name as fname, f.color as fcolor, f.places as fplaces, fs.id as fsid, fs.shape as fsname, ft.id as ftid, ft.type as ftname FROM `has_furniture` as hf LEFT JOIN `furniture` as f ON f.id = hf.idFurniture LEFT JOIN `furniture_type` as ft ON ft.id = f.idType LEFT JOIN `furniture_shape` as fs ON fs.id = f.idShape WHERE hf.idZone = [id de la zone]
 - J'arrive à afficher toutes les fournitures de la zone dans un modal
- Il faut aussi ajouter la possibilité de modifier ou de créer une exception sur une zone (plus tard)

11.4.28 Appel avec m. Garcia

- Discussion de retour de Nadège sur la documentation
 - Plus faire la différence entre le client et le restaurateur

11.4.29 13.05.20

Il faut créer un petit formulaire pour créer un nouveau meuble

- Je me suis dis qu'il devait être possible de créer un meuble, mais que celle-ci soit ensuite déplacée dans le restaurant.
 - C'est à dire, ajouter un champ dans la table des fournitures afin d'ajouter l'id de l'établissement
 - Donc, je vais commencer par créer un widget, afin de voir toutes les fournitures du restaurant
 - Depuis ce widget, il sera possible de voir dans quelle zone ce trouve la fourniture et il sera aussi possible de modifier cette zone ainsi que directement le meuble
 - Il faut ajouter à l'API le fait de récupérer toutes les fournitures d'un établissement
 - SQL : SELECT zone.id as zoneid, zone.name as zonename, f.id as fid, f.name as fname, f.color as fcolor, f.places as fplaces, ft.id as ftid, ft.type as ftname, fs.id as fsid, fs.shape as fsname FROM `floor` as floor LEFT JOIN `has_zone` as hz ON hz.idFloor = floor.id LEFT JOIN `zone` as zone ON zone.id = hz.idZone LEFT JOIN has_furniture as hf ON hf.idZone = zone.id LEFT JOIN furniture as f ON f.id = hf.idFurniture LEFT JOIN furniture_type as ft ON ft.id = f.idType LEFT JOIN furniture_shape as fs ON fs.id = f.idShape WHERE floor.idEstablishment = [id de l'établissement recherché] ORDER BY zone.id
 - Je me suis rendu compte que ça ne rentrait pas exactement ce que je voulais... en effet j'ai besoin de récupérer TOUTES les fournitures d'un restaurant et pas seulement celles qui sont dans des zones.
 - Nouveau SQL : SELECT f.id as fid, f.name as fname, f.color as fcolor, f.places as fplaces, ft.id as ftid, ft.type as ftname, fs.id as fsid, fs.shape as fsname, z.id as zid, z.name as zname FROM furniture as f LEFT JOIN furniture_type as ft ON ft.id = f.idType LEFT JOIN furniture_shape as fs ON fs.id = f.idShape LEFT JOIN

- has_furniture as hf ON hf.idFurniture = f.id LEFT JOIN zone as z ON z.id = hf.idZone WHERE f.idEtablissement = [id de l'établissement recherché] ORDER BY fid ASC
- l'api fonctionne et on peut récupérer les données avec ce lien : establishment/floor/zone/furniture/get?establishment&id=XX

Choses qu'il me reste à faire avant de pouvoir gérer les réservations

- Ajouter une table de liaison entre établissement et horaire
 - Par exemple une table qui gère les horaires (10:00-23:00) de l'établissement et une qui gère le jour (lundi, mardi, mercredi, etc.)
 - Comme ça, si une zone n'a pas d'horaires particuliers, elle prendra par défaut les horaires du restaurant
- Ajouter la modification d'une fourniture et d'une zone ainsi que la suppression
- Créer et attribuer des horaires pour un restaurant
 - Il faut que j'ajoute un widget qui affiche les horaires du restaurant ainsi qu'un petit bouton pour modifier les horaires
 - Il faut d'abord récupérer tous les horaires de la base de données
 - SQL : SELECT d.id as did, d.name as dname, s.id as sid, s.begin as sbegin, s.end as send FROM opening as o LEFT JOIN schudle as s ON s.id = o.idSchudle LEFT JOIN days as d ON d.id = o.idDay WHERE o.idEtablissement = [id de l'établissement]
 - La requête ci-dessus me retourne tous les horaires du restaurant, mais moi je veux tous les jours de la semaine.
 - Je vais faire ça dans l'API, car il y a des choses à prendre en compte
 - Requête pour récupérer tous les jours : SELECT * FROM days(plutôt simple)
 - J'arrive à récupérer les horaires d'un restaurant
 - Pour chaque jour :
 - Si le restaurant est ouvert : did => id du jour, dname => nom du jour, sid => id de l'heure, sbegin => heure de début, send => heure de fin
 - Si le restaurant est fermé : ````did => id du jour, dname => nom du jour, closed => closed````
 - L'application va charger de tester si la variable ````closed```` existe dans le tableau ou non
 - Je vais donc créer le widget pour la lecture des horaires de l'établissement
 - Les horaires s'affichent correctement
 - Il faut maintenant pouvoir les modifier
 - Pour ce faire, je dois créer un formulaire qui regroupe tous les horaires du restaurant ainsi qu'un champ de création si l'on souhaite ajouter un autre horaire que ceux déjà enregistrés

11.4.29.1 Appel avec m. Garcia

- Bonne (et mauvaise) nouvelle : Rendez-vous avec Nadège demain soir 17h.
- Appel demain à 16:45

11.4.29.2 MEMO

A faire demain pour la présentation :

- Créer un schéma logique sur l'API et les deux applications
 - Expliquer la différence entre l'application web et l'application du restaurant
 - Expliquer les avantages de l'API

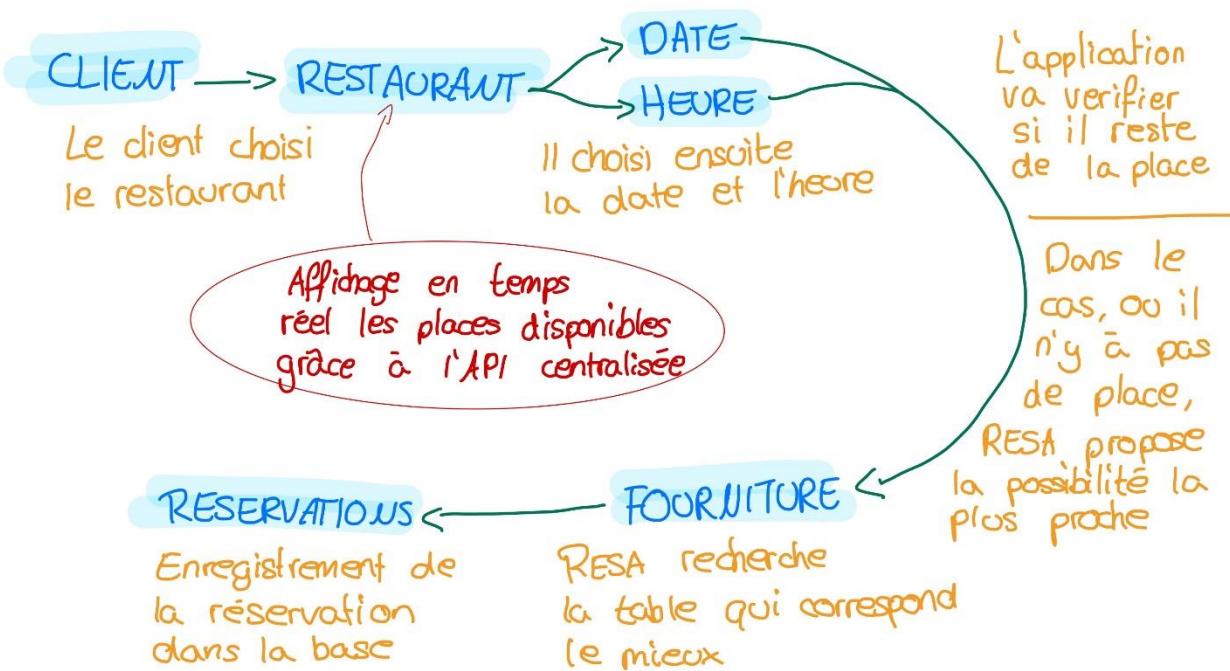
11.4.30 14.05.20

Pour créer une nouvelle réservation, je dois pouvoir récupérer les disponibilités d'un restaurant pour un horaire

- Récupérer tous les horaires du restaurant
- Récupérer les réservations
 - Attribuer des réservations avec des tables

Afin de m'aider dans ma réflexion:

Fonctionnement des réservations



11.4.30.1 Technique

1. Pour commencer, je récupère l'heure d'arrivée, la durée et l'heure estimée de départ de chaque réservation pour un établissement :

- ~~SQL : SELECT r.time as arrival, CAST(r.duration as time) as estimated_duration, CAST(r.time + r.duration as time) as estimated_end FROM reservation as r WHERE r.idEtablissement = 1~~
2. Je dois récupérer tous les horaires de l'établissement pour une journée
 3. Récupérer les places disponibles par heure de recherchée
 1. Récipérer toutes les places du restaurant existantes à l'heure recherchée
 1. Ce n'est pas la version finale, il faut que je regarde si celle-ci fonctionne correctement : `SELECT SUM(src.places) as places 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) <= '11:00:00' AND CAST(s.end as time) >= '12:00:00' AND hf.idZone IS NOT NULL AND f.idEtablissement = 1) src`
 2. Donc avec cette requête, j'arrive à récupérer toutes les places qu'il y a à une heure donnée de disponible dans un restaurant
 2. Il faut maintenant récupérer le nombres de places prises une certaine date avec une certaine heure

3. Problème

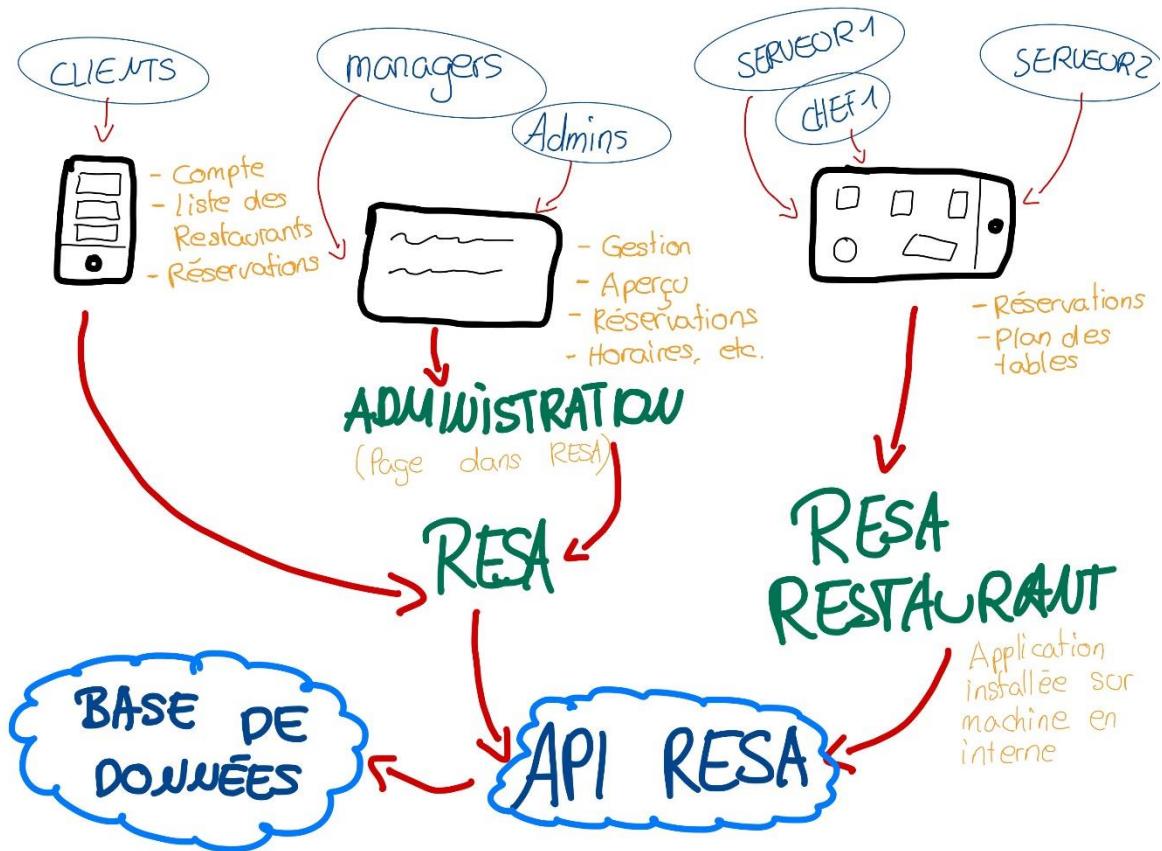
1. J'ai 2 enregistrements
 1. 12:30
 2. 12:45
2. Quand j'ajoute 1h15 aux deux à l'aide de la requête SQL, le premier me renvoi 13:45 mais le second me renvoi null. Je pense que c'est du au fait que au lieu d'ajouter 1h je dois d'abord arrondir puis ajouter une heure... je ne trouve pas comment faire sur internet. je vais essayer de trouver des solutions, sinon je vais contacter m. Garcia
 1. Il faut passer par le UNIX_TIMESTAMP pour avoir l'heure final
 2. SQL : `SELECT arrival, FROM_UNIXTIME(UNIX_TIMESTAMP(arrival)+duration) as estimated_end FROM reservation WHERE idEtablissement = [id de l'établissement]`
4. Je récupère toutes les places disponibles dans l'établissement pour une date et une heure donnée
 1. `SET @arrival = '12:30:00'; // Heure d'arrivée de la personne`
 2. `SET @duration = 3600; // Durée estimée du repas en secondes`
 3. `SET @date = '2020-05-14'; // Date du repas`
 4. `SET @etab = 1; // id de l'établissement`
 - 5.
 6. `SELECT SUM(src.places) -`
 7. `(`
 8. `SELECT SUM(f.places) not_available`
 9. `FROM reservation as r`
 10. `INNER JOIN furniture as f ON r.idFurniture = f.id`
 11. `WHERE r.idEtablissement = @etab`
 12. `AND CAST(r.arrival as DATE) = @date`
 13. `AND CAST(r.arrival as TIME) <= @arrival`
 14. `AND CAST(FROM_UNIXTIME(UNIX_TIMESTAMP(r.arrival)+r.duration) as TIME) >= @arrival`
 15. `) total`
 16. `FROM (`
 17. `SELECT f.places`
 18. `FROM `zone_has_schudle` as zhs`
 19. `INNER JOIN schudle as s ON s.id = zhs.idSchudle`
 20. `INNER JOIN zone as z ON z.id = zhs.idZone`
 21. `LEFT JOIN has_furniture as hf ON hf.idZone = z.id`

22. LEFT JOIN furniture as f ON f.id = hf.idFurniture
23. WHERE CAST(s.begin as time) <= @arrival
24. AND CAST(s.end as time) >=

CAST(FROM_UNIXTIME(UNIX_TIMESTAMP(@arrival)+@duration) as TIME)
25. AND hf.idZone IS NOT NULL
26. AND f.idEtablissement = @etab
27.) src
- 28.
29. En version copiable : SET @arrival = '12:30:00'; SET @duration = 3600; SET
 @date = '2020-05-14'; SET @etab = 1; SELECT SUM(src.places) - (SELECT
 SUM(f.places) not_available FROM reservation as r INNER JOIN furniture as f ON
 r.idFurniture = f.id WHERE r.idEtablissement = @etab AND CAST(r.arrival as
 DATE) = @date AND CAST(r.arrival as TIME) <= @arrival AND
 CAST(FROM_UNIXTIME(UNIX_TIMESTAMP(r.arrival)+r.duration) as TIME) >=
 @arrival) total FROM (SELECT f.places FROM `zone_has_schedule` as zhs
 INNER JOIN schedule as s ON s.id = zhs.idSchedule 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 = @etab) src
5. Je suis en train de me dire que je devrais peut-être ajouter la table dans la réservation, comme ça je pourrais juste récupérer les tables disponibles pour une certaine tranche horaire
 1. J'ai donc ajouter une colonne avec l'id de la fourniture réservée.
 2. J'ai donc modifier la requête SQL ci-dessus afin qu'elle reprenne le nombre de place des tableaux réservées et non pas le nombre de personnes
4. Afin d'afficher les bonnes réservations sur les bons jours, il faut que je récupère toutes les dates de la semaine
 0. Trouver toutes les réservations d'une date A à une date B
 1. SELECT
 2. r.id AS rid,
 3. r.arrival AS arrival,
 4. r.amount AS amount,
 5. f.id as fid,
 6. f.name as fname,
 7. f.places as fplaces,
 8. u.id AS uid,
 9. u.first_name AS ufirstname,
 10. u.last_name AS ulastname,
 11. u.phone AS uphone,
 12. u.email AS umail
 13. FROM
 14. reservation AS r
 15. INNER JOIN user AS u ON u.id = r.idUser
 16. LEFT JOIN furniture AS f ON f.id = r.idFurniture
 17. WHERE
 18. CAST(r.arrival AS DATE) >= '2020-05-11' // Date de début
 19. AND CAST(r.arrival AS DATE) <= '2020-05-17' // Date de fin
 20. AND r.idEtablissement = 1 // L'id de l'établissement
 21. ORDER BY r.arrival
 - 22.
 23. En version copiable : SELECT r.id AS rid, r.arrival AS arrival, r.amount AS amount,f.id as fid, f.name as fname, f.places as fplaces, u.id AS uid, u.first_name AS ufirstname, u.last_name AS ulastname, u.phone AS uphone, u.email AS umail FROM reservation AS r INNER JOIN user AS u ON u.id = r.idUser LEFT

```
JOIN furniture AS f ON f.id = r.idFurniture WHERE CAST(r.arrival AS DATE) >=
'[date de début]' AND CAST(r.arrival AS DATE) <= '[date de fin]' AND
r.idEtablissement = [id de l'établissement] ORDER BY r.arrival
```

Afin de pouvoir mieux expliquer à Nadège le fonctionnement de mon application, j'ai réalisé le schéma suivant :



11.4.30.2 Appel avec M. Garcia

- Déplacer la page d'administration dans RESA Restaurant

11.4.30.3 Application RESA Client

- Voir tous les restaurants inscrits
- Voir le bouton réserver si restaurant pro

11.4.30.4 Application RESA Restaurant

- Accéder aux données pro de l'application
 - Création d'étages, de zones et de fournitures

11.4.30.5 Reunion avec Nadège

- Niveau de droits
 - 1. Avoir accès aux réservations
 - 2. Avoir accès à tous le reste
 - Option bloquer que les réservations de resa ou toutes les réservations
 - Par périodes
 - Temps moyens midi : 1h / Temps moyens le soir : 2h
-

11.4.31 17.05.20

Création des différentes applications de RESA:

11.4.31.1 RESA

L'application que les utilisateurs téléchargent

- Afficher la liste des restaurants
- Afficher les menus, les images, les horaires et les informations de contact
- Reserver une table si le restaurant à le compte pro ou full

11.4.31.2 RESA BLOG

L'application gratuite pour les restaurateurs

- Création d'un restaurant
- Création / modifications des photos, menus, horaires et informations de contact
- Possibilité de passer à la version PRO ou full

11.4.31.3 RESA PRO

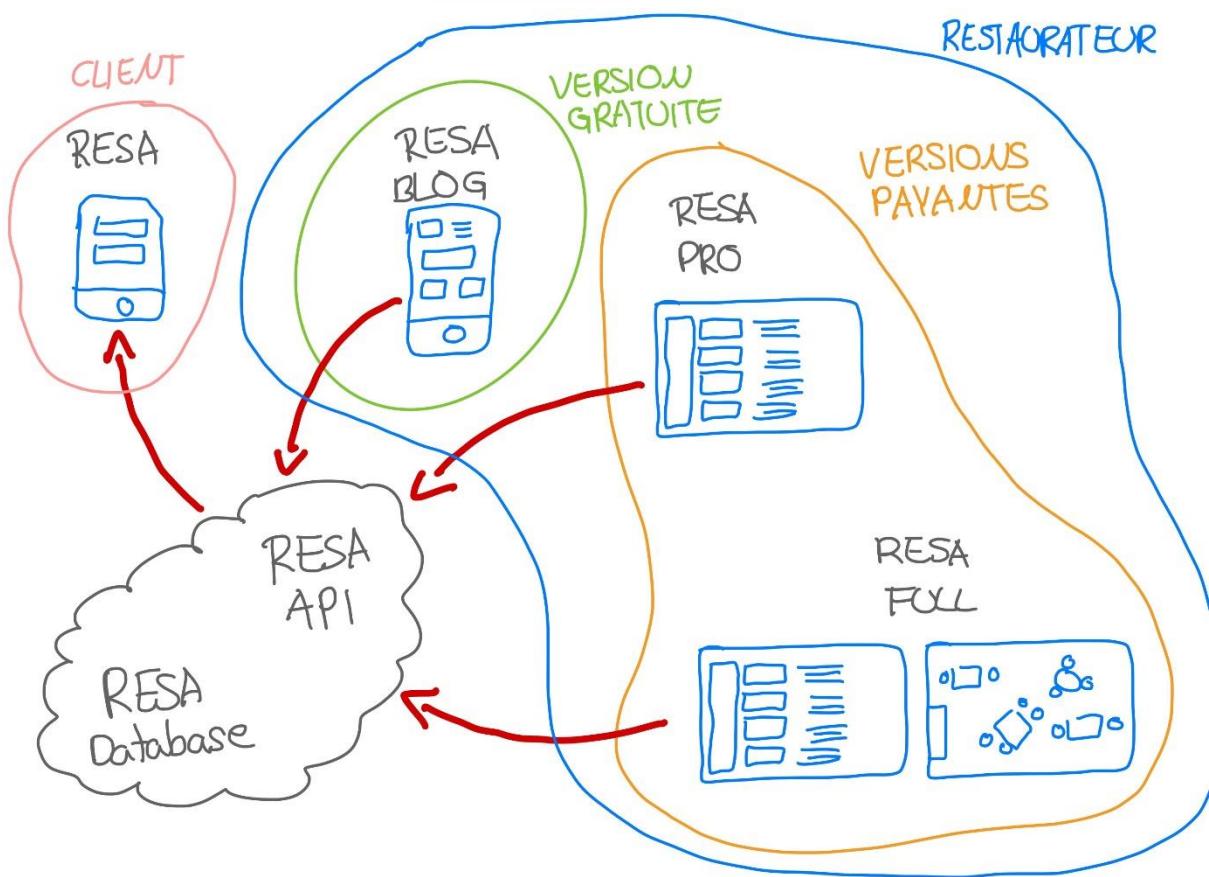
L'application qui permet de gérer les réservations de l'établissement

- Tout ce qui est dans RESA BLOG
- Ajout des fournitures du restaurant
- Gestion des réservations

11.4.31.4 RESA FULL

L'application qui permet le contrôle total sur le restaurant

- Tout ce qui est dans RESA PRO
- Gestion du plan de la salle en direct
- Gestion des zones et des horaires des zones
- Gestion du personnel et de leurs accès



Je vais commencer par faire la version gratuite. Celle qui concerne à créer un établissement et d'y ajouter les photos, les horaires et les menus

- Suppression de la gestion "admin" de l'application RESA de base.
 - Il s'agit bien d'une application à part entière. C'est pourquoi, même un manager peut ce connecter pour réserver une table dans un restaurant.

Afin de pouvoir faire correctement la redirection vers le bon service du restaurant, je vais donc créer un script qui va automatiquement rediriger le manager sur la bonne page de gestion de l'établissement

- Création du champ dans la base de données
 - 1 = BLOG
 - 2 = PRO
 - 3 = FULL
- Il faut ajouter dans l'API la gestion du niveau d'abonnement
 - Je peut récupérer le niveau de l'établissement grâce à son id
 - Lien API : establishment/get/?level&i=XX
- Il faut aussi que je fasse le login utilisateur pour un manager. C'est à dire que le manager doit entrer les informations suivantes :
 - Le nom du restaurant
 - Son identifiant (adresse mail)
 - Son mot de passe
 - SQL : SELECT e.id FROM establishment as e INNER JOIN is_in_as as iia ON iia.idEstablishment = e.id WHERE e.name = "[nom du restaurant]" AND iia.idUser IN

(SELECT u.id FROM user AS u WHERE email = "[email]" AND password = "^[mot de passe hashé]"") AND iia.idPermission = 2

- Création du script de redirection sur la page correspondante au niveau de sécurité du bâtiment

11.4.31.5 Nouveau planning des choses importantes à faire

11.4.31.5.1 RESA Client

- Finaliser la page d'accueil de RESA client
 - Mettre un filtre pour les restaurants
 - Afficher la page du restaurant avec les données de ce dernier

11.4.31.5.2 RESA Blog

- Afficher un menu simple pour Ajouter/modifier/supprimer les photos, les menus, les horaires, etc.
- Afficher les boutons pour découvrir les options payantes

11.4.31.5.3 RESA Pro

- Mettre en place la page admin que j'avais déjà créer afin de controller les fournitures ainsi que les réservations
- Gestion des login des employés

11.4.31.5.4 RESA Full

- Utilisation du plan de la salle
- Gestion des zones, des fournitures et des horaires
- Gestion des employés du restaurant et de leurs accès

11.4.32 18.05.20

Il faut que je mette à jour l'API avec mes recherches. C'est-à-dire compléter en ajoutant la recherches de places disponibles et les réservations de la semaine en cours (date de début et date de fin) Ce qui permet de rechercher par semaine les

- La longue requête que j'avais faite ci-dessus pour récupérer le nombre de place disponible à une certaine heure ne fonctionne pas avec PHP PDO car il ne gère pas le multi-requête
- je vais donc commencer par la page de RESA Client
 - Afficher le nombres de places disponibles pour l'heure à venir
 - Afficher la page du restaurant quand on clique sur la vignette.

11.4.33 Appel avec M. Garcia

- Parler du Coronavirus dans le bilan
- Faire un tableau de ce que peut faire chaque utilisateur dans l'application

11.4.34 19.05.20

Il faut que je modifie la requête

- SELECT SUM(src.places) - (SELECT SUM(f.places) not_avaiable FROM reservation as r INNER JOIN furniture as f ON r.idFurniture = f.id WHERE r.idEtablissement = :etab AND CAST(r.arrival as DATE) = :d AND CAST(r.arrival as TIME) <= :arrival AND CAST(FROM_UNIXTIME(UNIX_TIMESTAMP(r.arrival)+r.duration) as TIME) >= :arrival) total 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 = :etab) src
- Mais du coup elle ne fonctionne pas...

Voilà comment je vais faire à partir de maintenant sur la page principale de RESA:

1. Finir la page principale de RESA avec un filtre. Il permet de filtrer les restaurants ouverts et fermés ainsi que ceux qui ont encore de la place
 2. Mettre un place une recherche
 3. Mettre en place la page qui affiche les informations du restaurant (photos, menus, informations de réservation)
- Afin de faire le filtre pour les restaurants ouverts ou non, je dois vérifier leur état au chargement
 - Je récupère donc d'abord le jour actuel dans la base de données d'après le jour actuel getdate de PHP
 - establishment/schudle/get?today
 - Avec cette fonction, je devrais pouvoir créer une autre fonction qui me retourne le statut actuel d'un restaurant au moment donnée
 - Voici la requête qui me permet de récupérer le statut actuel du restaurant
 - SELECT s.id FROM opening as o INNER JOIN schudle as s ON s.id = o.idSchudle WHERE o.idDay = 2 AND o.idEtablissement = 1 AND UNIX_TIMESTAMP(s.begin) < UNIX_TIMESTAMP(NOW()) AND UNIX_TIMESTAMP(s.end) > UNIX_TIMESTAMP(NOW())
 - Elle retourne l'id de l'horaire si c'est ouvert et NULL si le restaurant n'est pas ouvert
 - J'ai fusionné cette requête avec la requête qui récupère tous les restaurants afin de pouvoir directement avoir le status quand je les récupères tous
 - SQL final : SELECT e.id, e.name, e.address, e.phone, e.email, m.name as menu_name, m.description as menu_description, (SELECT s.id FROM opening as o INNER JOIN schudle as s ON s.id = o.idSchudle WHERE o.idDay = [id du jour (Générer par le php directement)] AND o.idEtablissement = 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
 - Pour créer le filtrage j'ai eu des petits soucis à cause des divs que j'avais créer avec bootstrap
 - Je peut effectuer des recherches par nom maintenant
 - Création de la page du restaurant
 - Affichage des photos du restaurant sous forme de slider ou de gallery
 - Il faut que je regarde sur bootstrap comment faire un slider
 - Il faut aussi que je mregarde si c'est plus beau de faire un slider ou un gallery des photos du restaurant

- Afin de valider l'accès à la page, je dois juste mettre à jour la requête qui récupère les infos du restaurant
 - il faut que la requête retour aussi si le restaurant est ouvert ou non
- Les restaurants ont tous des images mais certains non, il faut donc que je fasse une requête qui me retourne toutes les images si elles existent, ou alors le lien par défaut si il y en a pas
 - Je perd trop de temps sur ça, on va faire plus simple.
 - C'est l'api qui va vérifier et si elle trouve null, elle enverra le lien par défaut
- Affichage des photos du restaurant
- Affichage d'un calendrier trouver sur ce site : <https://www.startutorial.com/articles/view/how-to-build-a-web-calendar-in-php>
 - Je vais devoir le modifier pour qu'il corresponde à mes besoins

11.4.34.1 Appel avec m. Garcia

- Ne pas mettre de style dans l'html
 - Favoriser le nom de classe

ATTENTION :

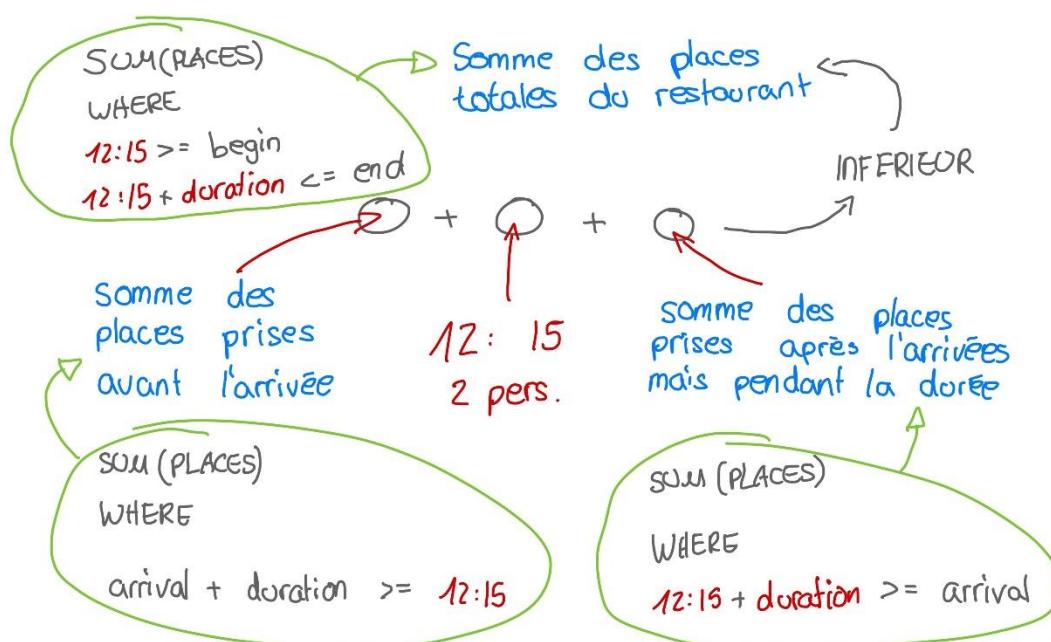
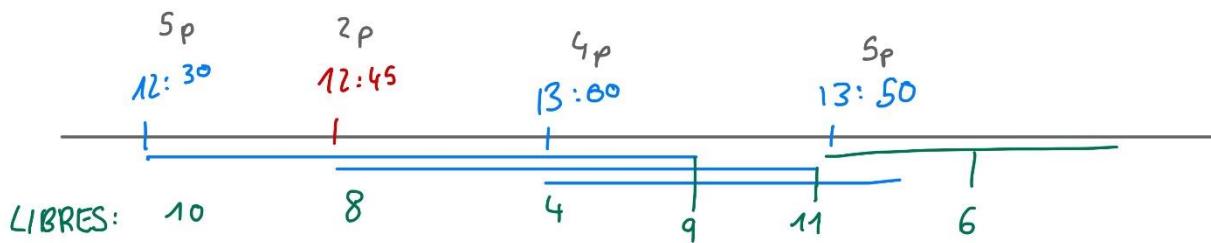
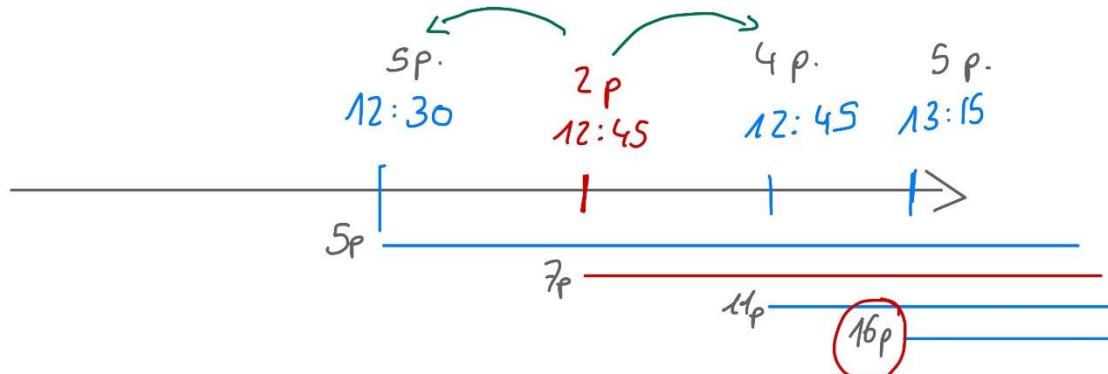
- ~~Il faut pas que j'oublie que le calendrier doit seulement s'afficher pour les restaurants qui ont l'option ! [OK]~~
- Il faut également que je fasse des diagrammes de fonctionnement en plus que celui que j'avais fait.
- -> Jeudi + Vendredi = documentation à fond pour mettre à jour

11.4.35 20.05.20

- Ajout du numéro de téléphone et de l'adresse mail sur la page du restaurant
- Il faut maintenant que je créer la page de réservation
 - Cette page va récupérer tous les horaires disponibles pour le restaurant durant la journée
 - Elle va également permettre à l'utilisateur de changer la date si il le souhaite à nouveau
 - Une fois validé, la réservation pour le restaurant sera faite automatiquement
- Si un utilisateur n'est pas connecté, il est automatiquement redirigé sur la page du fast login
 - Cette page permet de logger une personne rapidement et de revenir sur la page actuelle
 - Cette page nécessite dans la session une variable returnlink qui permet la redirection après le login
- J'ai créé une routine MySQL afin de pouvoir facilement appeler la fonction quand j'en ai besoin
 - la fonction s'appelle : GetAvailablePlaces
 - Paramètres :
 - idETab : id de l'établissement
 - dateday : la date
 - arrival : heure d'arrivée
 - duration : durée en secondes estimées du repas
 - sortie :
 - available : le nombre de places disponibles
 - Appeler : Call GetAvailablePlaces(1,'2020-05-14','12:30', 3600, @available)
 - Changement : Call IsPlaceForReservation(1,'2020-05-14','12:00', 3600, @available) (Il fallait que je fasse attention aussi au rendez-vous qui étaient déjà pris après, car si il n'y avait plus de places à 19h, je ne pouvais pas accepter une réservation à 18h50 si il y avait des places de libres)

MAX: 15 p.

Verifier disponibilités
avant mais aussi
après



- Pour la page de réservation je ne sais pas si je dois d'abord choisir l'heure et ensuite afficher le nombre de places disponibles ou demander le nombres de personnes et ensuite afficher les horaires disponibles.
 - Je pense que je vais d'abord demander les horaires et que en fonction des horaires je vais chercher le nombres de places disponibles
- ~~Il faut que j'améliore la procédure et pour ce faire je dois mettre un boucle dans celle-ci ~~
 - Lien : <https://www.ibm.com/support/producthub/iias/docs/content/SSHRBY/com.ibm.swg.im.dashdb.apdv.sqlpl.doc/doc/c0024352.html>
- Cette fonction me donne donc le nombres de places disponibles pour une heure avec une durée
 - Il faut maintenant ajouter la requête correspondante dans l'API

11.4.35.1 *Appel avec m. Garcia*

- Ajouter des pages sur la page d'accueil
 - Afin d'afficher maximum 25-30 restaurants par page
- Filtre par région
- Parler dans la documentation du fait que nous avons eu des retours de nadège, c'est pour ça qu'il y atout le temps des changements

11.4.36 21.05.20

- Pour récupérer tous les cérenaux horaires des restaurants du jour
 - Le ou les crenaux horaires du jour : SELECT s.id, s.begin,s.end FROM opening as o INNER JOIN schudle as s ON s.id = o.idSchudle WHERE o.idEtablissement = 1 AND o.idDay = 4
- ENORME JOURNÉE DOCUMENTATION

11.4.37 22.05.20

[A FAIRE]

- Mettre à jour tout le cheat sheet de l'API
- Ajouter des diagrammes d'activités [OK]

11.4.37.1 *Appel avec m. Garcia (avant évaluation intermédiaire)*

- Enlever le (intermédiaire) dans les résumés

11.4.38 23.05.20

11.4.38.1 *Evaluation intermediaire*

- Reprendre l'export de la BDD et d'y mettre des commentaires (en anglais) et l'ajouter dans la doc
 - Mettre à jour le .htaccess
 - Ajouter les planning dans la doc
 - Passer la doc dans antidote
-

11.4.39 27.05.20

Aujourd'hui, l'objectif est de finir les réservations du côté du client. Il faut donc que je regarde ce que j'ai déjà fait:

- un client voit le calendrier du mois en cours, il faudrait ajouter en orange le jour actuel et en rouge les jours fermés. [plus tard]
 - Pour réserver, je dois demander à l'utilisateur, quel nombre de personnes il(s) seront
 - Pour ce faire, je vais afficher des petites cartes avec le nombres de personnes
 - je perd beaucoup de temps à trouver les bonnes requêtes et à créer les vues (c'est pour cette raison que mon journal de bord est un peu vide)
-

11.4.40 28.05.20

J'étais parti sur le fait que l'utilisateur choisissait une date sur le calendrier, puis il choisissait le nombre de personnes et le créneau horaire.

- Après une discussion avec m. Garcia, j'ai décidé de tout mettre dans une sorte de widget.
 - LE gros calendrier a été remplacer par un petit widget avec un formulaire qui demande la date, l'heure et le nombre de personnes. Ce formulaire redirige après sur une page qui va vérifier si la réservation est bien valide et si c'est le cas, la créer dans la base de données.
- Le formulaire existe et envoie 3 données :
 - la date
 - l'heure
 - le nombre de personnes (l'utilisateur est déjà connecté et enregistré dans la session, pas besoin de l'envoyer)
- Il me manque que à faire les messages d'avertissements pour la validation ou le refus de la réservation.
- (Je dois encore faire l'algorithme qui va proposer au client un autre horaire le plus proche quand il n'y a pas de place.) à voir comment je vais faire

11.4.40.1 *Appel avec m. Garcia*

- J'ai privilégié le fait de vouloir aller dans le restaurant au lieu de rechercher les restaurants disponibles à une date
- Changer les boutons par un dropdown

11.4.41 29.05.20

- Il faut maintenant que je trouve les fournitures qui ont de la place au moment où le client souhaite venir.
 - Il faut également que je gère la fusion de tables si il n'y a plus de places

// On vérifie la durée estimée pour les réservations // Entre 6h et 11h : 1800 sec = 30 min // entre 11h et 18h : 3600 sec = 1 h // entre 18h et 23h : 7200 sec = 2 h // entre 23h et 6h : 5000 sec = 1h25

- Il y a donc des messages d'erreurs qui s'affichent quand un utilisateur ne peut pas réserver et un message de validation quand tout est ok.
 - Maintenant il faut passer à la partie où l'utilisateur réserve effectivement et celle où le gérant du restaurant reçoit bien les réservations
- Avant ça, il faut que je fasse ça (-> - Il faut également que je gère la fusion de tables si il n'y a plus de places)
 - Une fois que ça sera fait, il faudra que je fasse la page de gestion de l'établissement
 - (je crois d'ailleurs que j'ai pas mal de modifications à faire ... à voir)
- Donc, il faut que je fasse une requête SQL qui vérifie que les tables soient bien disponibles au moment de la réservation, ensuite, il faut que je vérifie si une table avec le minimum de nombre de place est disponible
- J'y pense maintenant, mais il faut que je fasse le formulaire de création de compte

11.4.42 02.06.20

- choses faites
 - Création d'un utilisateur à partir de toutes les pages de login
 - L'utilisateur est automatiquement redirigé sur la page correspondante en fonction d'où il vient
 - Comme j'ai changé la table des établissements, je dois m'assurer que toutes les requêtes que j'avais déjà faites soient compatibles
 - Si elles ne le sont pas, l'application va afficher des erreurs !
 - Création de la page de nouvel établissement
 - L'utilisateur va pouvoir choisir l'abonnement qu'il souhaite prendre
 - Je vais faire un vérificateur de cartes factis, afin de justifier le concept
- Je vais devoir aussi m'occuper de la partie administration des établissements
- Je dois aussi correctement finir la partie des réservations afin que ces dernières s'affichent dans les menus

11.4.43 03.06.20

- Ajout d'une table appelée "subscription" qui va enregistrer les différents niveaux d'abonnement
 - Modification de la redirection et de la requête dans l'API

11.4.44 04.06.20

- Afin de pouvoir ajouter le bouton d'ouverture et de fermeture spontannée de l'établissement, il faudrait que j'ajoute une table nommée "instant closing" qui viendrais faire un overwrite de la table schudle.
 - A noter dans la doc.
- Création du formulaire dans le manager
 - Il faut maintenant que je fasse la liaison avec l'api
 - Changement de l'api pour pouvoir mettre aux nouvelles normes de ma BDD
 - Supression du champ "adresse"
 - Ajout des champs : route, npa, ville, pays

11.4.45 Jour 326493 (je pers la tête)

11.4.45.1 Appel avec m. Garcia

- Mettre l'heure de la réservation à l'heure actuelle
-
- Un manager peut donc créer un établissement facilement depuis l'application de manager
 - Il faut que j'ajoute dans l'API le niveau d'abonnement que l'utilisateur à pris [ok]
 - Il faut aussi que je change la manière dont j'envoie les données [ok]
- Une fois que j'aurais fini correctement la création de l'établissement, il faudra que je crée la page d'administration du blog, afin que l'utilisateur puisse gérer les données de base de son établissement.
- J'avais un gros problème de redirection dans mon application...
 - Quand je me connectais sur un établissement créer depuis l'application, je ne me faisait pas rediriger sur la page de gestion de l'établissement. J'ai donc du changer le script de la page de login pour m'accorder.

Il ne me reste plus que la soirée pour coder, je vais donc juste faire la modification des horaires d'un restaurant afin de l'afficher ouvert ou fermé

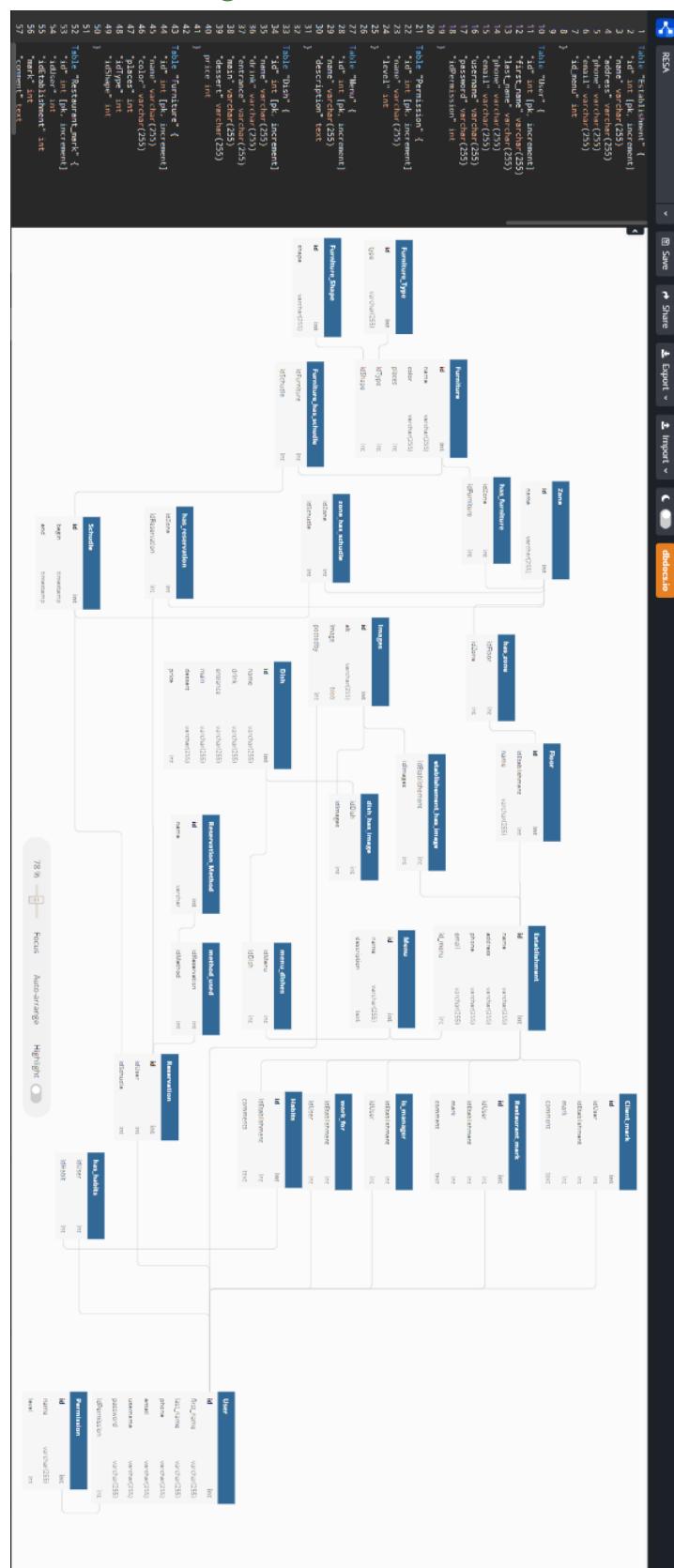
Pour mettre en ligne mon application je dois changer le chemin dans ces fichiers

- api/v2 : vars.php
- manager/style/assets/css/ : style.css
 - il y a 2 endroits
- resa/assets/css/ : style.css
 - il y a 2 endroits
- resa/assets/js : reservation.js
- J'a ajouter la page de gestion des zones pour l'abonnement de RESA Full
 - J'ai du changer les variables demandées + les conditions de redirections
- Je vais créer la page pour afficher les employés du restaurant
 - Pour ce faire, je dois d'abord créer la requête dans l'API
 - C'est tout bon
- J'ai créer et ajouter toutes les photos des pages manquantes dans l'application

11.5 Images (pleins format)

10.4.1	Interface dbdiagram.io	139
10.4.2	MCD de la base de données	140
10.4.3	Esquisse n°1 Poster	141
10.4.4	Esquisse n°2 Poster	142
10.4.5	Poster final RESA	143
10.4.6	Planning initial.....	144
10.4.7	Planning final	145

11.5.1 Interface dbdiagram.io



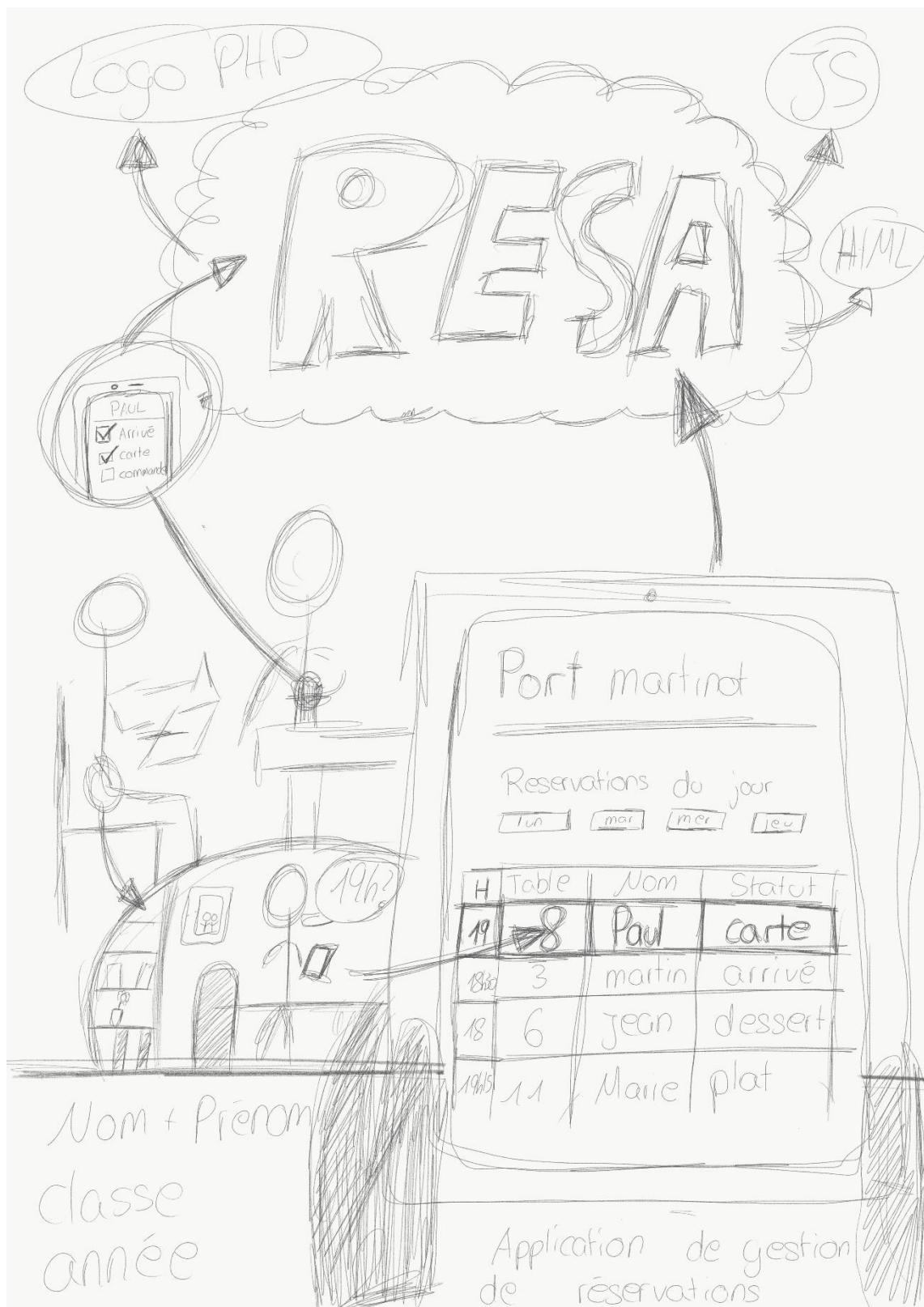
11.5.2 MCD de la base de données



11.5.3 Esquisse n°1 Poster



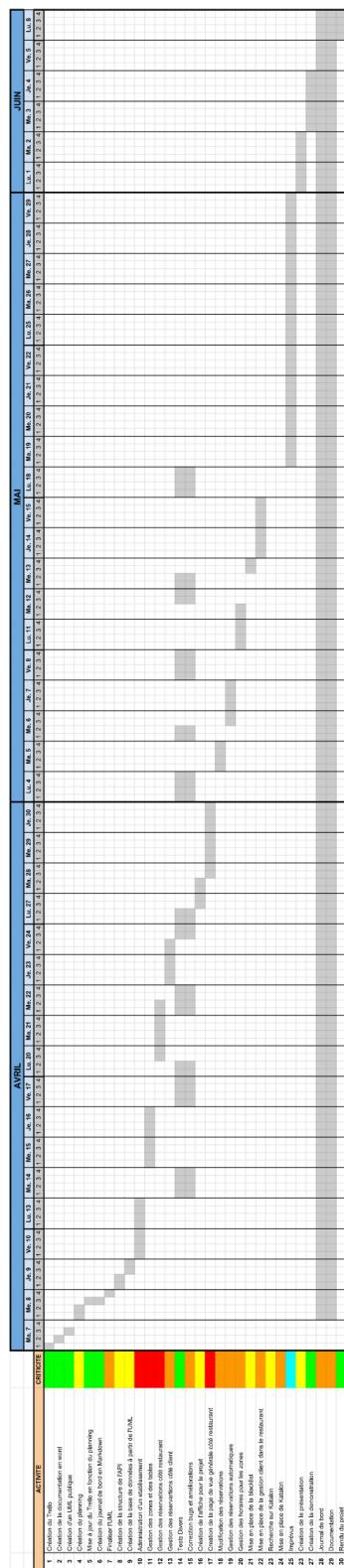
11.5.4 Esquisse n°2 Poster



11.5.5 Poster final RESA



11.5.6 Planning initial



11.5.7 Planning final

