www.isl.be

INSTITUT SAINT-LAURENT
enseignement de promotion sociale

rue Saint-Laurent 33 - 4000 LIEGE
04 223 11 31

Réalisation du site 'La Clémentine'

Liège, le 7 octobre 2018

Travail de fin d'études présenté par

Novitz Jean-Philippe

En vue de l'obtention du brevet de l'enseignement supérieur de

WebDeveloper

Enseignement supérieur économique de promotion sociale et de type court

Novitz Jean-Philippe

1. INTRODUCTION.

Voici un peu plus de trois ans, je me suis lancé dans une aventure : reprendre des études afin de concrétiser quelque chose d'inachevé.

Les enseignants que j'ai rencontré à L'institut Saint-Laurent m'ont appris énormément de choses, ils m'ont motivé non seulement en me montrant ce qu'il ya au bout du chemin mais en me montrant comment atteindre mon but.

Chacun m'a aidé, à sa manière, à progresser depuis l'initiation à la programmation, les bases de php vers une utilisation autonome d'un framework. J'ai tant appris, au niveau analyse, programmation, aussi bien du point de vue front-end que back-end.

Le moment est venu de réaliser mont travail final.

Cet exercice de fin d'études me permets de montrer à mes enseignants que je sais appliquer d'une manière autonome ce qu'ils m'ont apporté pendant ces soirées. Je sais que choisir d'enseigner dans une école de promotion sociale est guidé par une envie de transmettre.

Durant la réalisation du projet j'ai eu à coeur de démontrer ma capacité à apprendre par moi-même. En effet, la première leçon est qu'un bon developpeur n'est pas juste un codeur mais une personne qui sait réfléchir, evoluer et s'adapter .

Novitz Jean-Philippe

J'ai choisis de réaliser un site pour 'La Clémentine', une sandwicherie naissante de la banlieue liégeoise. Plus qu'une mise en situation c'est une réelle opportunité de mettre ce que j'ai appris au service d'une entrepreneuse de la vie réelle.

La Clémentine a besoin de mettre en forme le catalogue de ses produits, d'informer sa clientèle à propos des allergènes, faciliter la commande d'un repas.

Dans les pages qui suivent je vais vous montrer ce que j'ai mis en place pour analyser les besoins de ma cliente, quelles sont les technologies que j'ai mises en œuvre et pourquoi.

Si j'atteinds mon objectif vous allez voir, petit à petit, le site prendre forme et exister.

2. DESCRIPTION DU SITE

2.1. Thème et brève description

'La Clémentine' est une sandwicherie située à Saint-Nicolas (Liège). Le projet consiste en la création d'un site internet qui mettra les produits en valeurs.

2.2. Objectif

Mon travail devra s'orienter vers deux objectifs principaux : créer une vitrine et faciliter l'acte d'achat.

2.2.1. Une vitrine pour présenter les produits :

Vendre des sandwiches, c'est la raison d'être de l'entreprise. Une présentation claire et attractive des produits apportera une valeur ajoutée car le client potentiel aura le temps de réfléchir avant d'entrer en boutique. Sa vision sur l'offre de La Clémentine sera toujours plus claire dans l'écran de son smartphone que sur une carte ou un tableau.

Il serait bon d'instaurer une sensation de proximité, de mettre en place une interaction avec les clients potentiels . Je pourrais par exemple mettre une de ces solutions en place.

- Pour la page contact, proposer une carte avec comme marqueurs la localisation du magasin, d'un ou deux points de repères et celle du client afin de montrer clairement la proximité.

- La page photos serait un appercu des dernières photos publiées sur instagram. Cette page serait à la un support visuel montrant la vie de la sandwicherie mais aussi un point d'accès direct vers le compte Instagram de l'entreprise.

2.2.2. Proposer un service de commande en ligne :

La clémentine est une sandwicherie qui offre une amplitude d'ouverture très large c'est à dire qu'elle ouvre dès 8:30 le matin et ferme à 15h, c'est plus que la plus part des sandwicheries. L'horaire classique est généralement 10h-14h.

Mais tous le monde mange vers midi et, après six mois d'activité, la gérante a constaté qu'elle est confronté deux pics d'affluence. Un premier vers 11h30, l'autre entre 12h30 et 13h.

Cela représente un gros travail a exécuter par une seule personne. Proposer aux clients un système de commande permettrais :

- Aux clients de commander bien à l'avance en ayant le confort d'être devant un écran au lieu d'hésiter longement devant le comptoir. Le temps d'attente et le temps d'encombrement est nettement réduit, la charge est répartit plus tôt dans la journée.

- Si on parvenais à faire prendre l'habitue à la clientèle de commander en ligne et ensuite de passer chercher leur commande (ou à déguster sur place), il sera ensuite plus facile si nécessaire de passer à un système de livraison.

2.2.3. 3. Information sur les allergènes :

Le commercant est obligé légalement de fourrnir une information au consommateur concernat les allergènes alimentaire. Le règlement de l'AFSCA précise que le détaillant doit détenir un classeur avec une fiche pour chaque préparation avec indication des allergies qu'il peut provoquer.

En plus un paneau doit afficher une phrase signifiant à la clientèle qu'il est possible de connaître ou de consulter ces documents.

Je propose d'enregistrer chaque allergie dans la base de données, de les relier à la liste des ingrédient et d'automatiser l'affichage de la version numérique e cette information et de la proposer au consommateur une autre façon de la consulter.

2.3. Public cible

Le public cible est d'une part la gérante de la sandwicherie, d'autre part les clients (ceux qui n'ont jamais acheté et les habitués).

2.4. Technologies utilisées

Ce travail a nécéssité la mise en œuvre et l'apprentissage de nombreuses technologies ont les principales sont les suivantes :

Le langage php est le principal langage utilisé durant ma formation. Mon initiation à la programmation, mon apprentissage de la programmation orientée objet et de l'architecture mvc ont été faites avec le ce langage. Les *frameworks* font partie de la programmation moderne, ils aident à structurer notre code et à accélerer le développement. **Symfony** est le framework choisi, il s'agit d'un des deux frameworks de pointe. Le 'backend' de mon site est une production Symfony 4.

J'ai choisis d'utiliser, en plus de jQuery, le Framework Vue.js et ce sous plusieures formes.

- Le front-end, la 'vitrine' de mon site est composée à l'aide de vue-cli. Il s'agit d'une page html controllé par du javascript. La page n'est donc jamais rechargé mais évolue en temps réelle en fonction des élement dynamique (les composant) que l'on affiche, que l'on enlève ou modifie. Vue.js fonctionne en étroite collaboration avec son store vuex. Il s'agit s'une sorte de base de données en mémoir de tous les élements chargé par Vue. De cette manière les datas sont directement disponible n'importe ou dans l'application.

- Vue.js peut aussi s'utiliser à la manière de jQuery. C'est à dire que l'on va créer une instance de Vue qui va controller un élement précis. Exemple, dans une page du backend on n'a pas du tout besoin d'éviter les rechargements de page, seul les administrateurs voient les pages. Mais on peut avoir envie, quand on est sur une page, de cliquer sur un bouton et que le status d'un élément change instantanément. Sur la page d'une commande un bouton indique que la commande est en attente, un clique modifie l'aspect de la commane derrière cela une requête est faite en base de donnée.

Un autre exemple est le fait que Symfony a besoin de javascript dans ses formulaires pour aider à la l'utiliser des champs de type Collection. La documentation indique des exemples d'utilisation avec jQuery. Après avoir fait le premier formulaire avec jQuery je me suis employé à convertir l'exemple à Vue. Dans ce cas il s'agit de se greffer à un élement ou <ul> qui n'affiche rien et d'utiliser js pour l'afficher en utilisant le prototype présent dans le forulaire Symfony.

Je n'ai pas utilisé de thème ou template acheté à un site spécialisé. J'ai utilisé Semantic UI qui est un framework css disponible en version 'standard' qui contient des éléments jQuey et une version Semantic-vue qui est la variate vue.js de Semantic.

Le premier a été utilisé pour mettre en forme le back office, la seconde s'intègre parfaitement à mon interface publique déjà entièrement en vue. Dans ce cas semantic-vue me fournis des composants à intégrer à mon code html.

Semantic se compose de toute une libraire de fichers scss. Tel quel je peux facilement placer mes éléments, de les mettres en forme. Je peux configuer ou surcharger n'importe quel item, des feuilles de styles vides sont disponibles à cet effet. Semantic ui utilise Gulp regrouper le tout avant la mise en production.

Mon site contenant un tableau de bord en temps réel j'avais besoin d'intégrer Socket.io et Elephant.io pour permettre l'envoi de message, de Node JS en tant que serveur et Redis garder les commandes en mémoire et les fournir aux utilisateurs connecté.

Encore me permet de déveopper du javascript et du php au sein de la même application, webpack permet quant à lui de builder mes assets pour avoir un site visible dans n'importe quel navigateur.

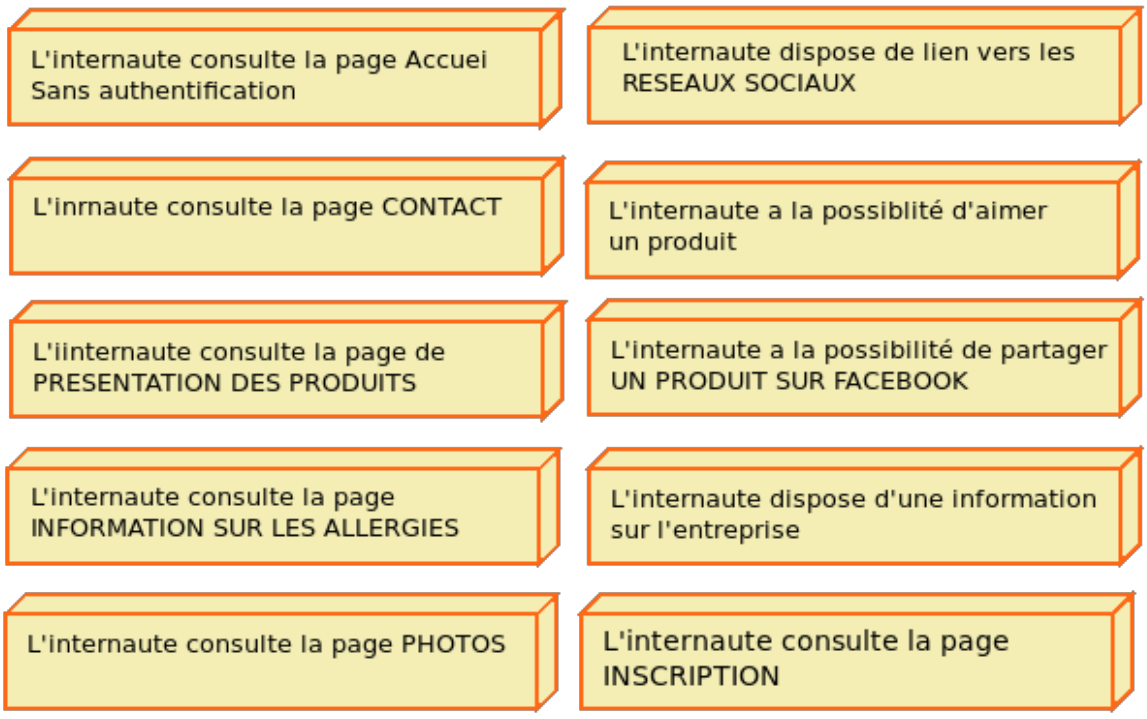
Novitz Jean-Philippe

3. EXIGENCES FONCTIONNELLES

3.1. Fonctionnalités

Les fonctionnalités sont données par ordre croissantes, il s'entend que chaque niveau dispose des même fonctionnalités des niveaux précédent avec quelque chose ne plus, quelque chose de différent.

1. Internaute anonyme.



Toute personne qui arrive doit pouvoir avoir accès, sans aucune identification à :

- **Page d'accueil**

  La page d'accueil sera visible de tout le monde, elle présentera un cliché de ce qu'est la Sandwicherie, où la trouver et un appercu de quelques produits. L'objectif est de tout de suite savoir ce dont il s'agit.

- **Page de présentation de l'entreprise**
  page de présentation plus complète des activités de l'entreprise, cet élément ne devrait pas coûter cher en temps mais pourrait rapporter en termes de moteurs de recherche ou d'entreprises qui souhaiterais faire appel à *La Clémentine* pour des évènements ou des anniversaires.

- **Page Contact**
  Indication de divers moyens de contacts :

  - Adresse ;
  - téléphone ;
  - email ;
  - liens vers les reseaux sociaux sur lesquels l'entreprise est présente :
    - Facebook ;
    - Instagram.

- **Page produits**
  Cette page présentera les differents produits proposé par *La Clémentine*. Une liste des plats et sandwiches que chacun pourra avoir dans sa poche, l'idéal pour réfléchir avant de venir acheter.

- La Clémentine est présente sur Instagram. Plusieurs éléments vont concourrir à donner un sentiment de proximité à la clientèle grâce à ce réseau.
  Des photos sont régulièrement publiées sur Instagram. Une page photos reprendra ces photos. De cette façon à chaque nouvelle publication sur Instagram cette galerie photo est mise à jour et chaque jour différente.

- Partager / aimer sur facebok:
  Sur la page de description d'un produit, deux boutons en rapport avec facebook permettrons de partager l'expérience avec les autres utilisateurs.

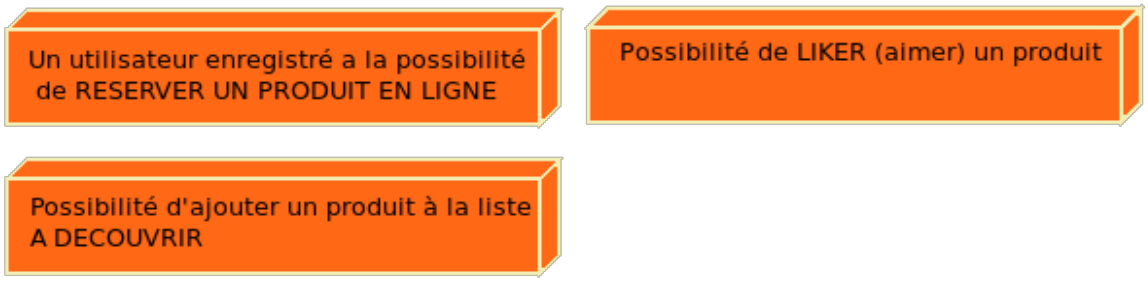
- Page allergies
  Informer la clientèle sur les allègènes alimentaire est une obligation de l'AFSCA. La gérante de la sandwicherie a décidé de prendre cette obligation à bras le corps, de saisir l'occasion de communiquer avec les clients et de ne pas subir.
  Une page d'information présentant les quatorze allergies alimentaires et une information dans la fiche produits indiquant quelles allergies sont susceptible d'êtres provoquées avec un message indiquant que la sandwicherie est attentive aux allergies que pourrait provoquer ses produits et invitant les clients à en parler au besoin.
  On passe d'une contrainte à un rapport de confiance.

- Page inscription :
  Il faut un lien pour indiquer au visiteur qu'il a la possibilité de s'inscrire et de devenir un utilisateur.

- Infos données :
  Un visiteur inscrit fourni des données à l'entreprise. Ces données ne sont ni données ni revendues à des tiers. Les informations récoltées servent simplement à permettre la connexion et à communiquer B to C entre l'entreprise et le consommateur.
  Cette information sera présente d'un manière ou d'une autre sur le site internet.

Novitz Jean-Philippe

2. Utilisateur enregistré



Un internaute peut décider de s'inscrire. Après verification/validation de son profil il aura, en plus des fonctionnalités ci-dessus, la possibilité de :

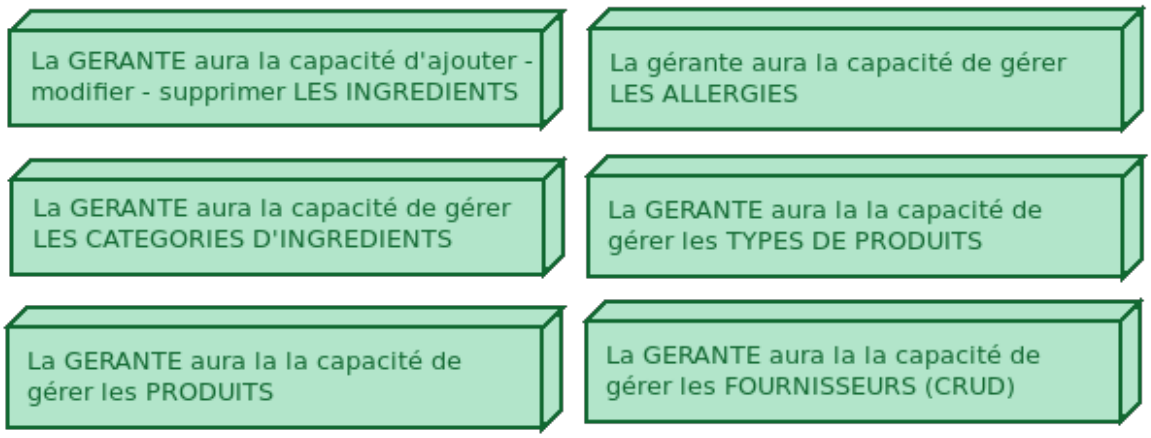
- **Aimer (liker) un produit** :
  Il serait intéressant pour la commercante de proposer aux clients enregistrés un bouton pour signaler qu'ils ont apprécié un produit et, de cette façon, savoir quels sont les produits les plus apréciés. C'est une autre façon d'avoir un retour venant du consommateur.
  Le consommateur aura la possibilté de visualiser une liste des produits qu'il a aimer.

- **Ajouter un produit à la liste 'a decouvrir'**
  Il est souvent difficile de faire un choix parmi le large menu. Parfois on se dis je prends ceci aujourd'hui, je prendrais cela demain. Encore faut-il s'en souvenir ou le noter quelque part.
  Je propose la fonctionnalité pour un utilisateur une liste des sandwichs qui lui donnent envie.
  De cette façon il pourrait à chaque visite prendre un produit dans sa liste. Un autre aspect de cette fonctionnalité est que plus cette liste sera longue et plus le client aura tendance à revenir s'il se prend au jeu d'enrichir sa liste et de tester.

- **Réservation en ligne :**

  Petit à petit les clients commencent à réserver dans la matinée, quelques élèves se regroupent et téléphones chacun à son tour et viennent chercher.
  La Clémentine aimerait généraliser cette pratique par exemple aux employé communaux et aux ouvrier qui n'ont qu'une demi-heure de pause. C'est une façon pour eux de gagner du temps.
  La réservation en ligne sera un outils supplémentaire pour rendre ce geste de réservation plus facile.

3. La gérante.



La Gérante du magasin pourra.

Gerer les Ingrédients :

- **ajouter – supprimer – modifier un ingrédient**
  La gérante de la sandwicherie aura la possibilité d'ajouter un ingrédient, de lui assigner à une ou plusieures catégories.
  Elle aura également la possibilité de modifier les informations relatives à cet ingrédient ou de le supprimer.

- **Ajouter – supprimer – modifier une catégorie**
  La cheffe ajoutera, modifiera ou supprimera une catégorie. Les catégories sont des entités simples qui permettent de faire des classements, des regroupement ou des sélections d'ingrédients.

- Ajouter – supprimer – modifier un fournisseur
  Clémentine encodera chaque nouveau fournisseur, répercutera chaque changement dans la fiche signalétique d'un fournisseur. Un fournisseur pourra être supprimé s'il n'a jamais été actif.

- Ajouter – supprimer – modifier une entrée de marchandise
  Lors d'une entrée de marchandie la responsable pourra marquer une entrée de marchandise en ajoutant un scan de la facture d'entrée et en encodant des mots clés représentant les principales marchandiese.
  De cette manière il sera facile de retrouver une facture en fonction de ces mots clés.

- Ajouter – supprimer – modifier une allergie
  La gérante aura la responsabilité d'ajouter, de modifier ou de supprimer une allergie. L'encodage permettra :

  - d'indiquer un nom, une description éventuelle, une image

  - relier l'allergie à des catégories et /ou des ingrédients
    relier une allergie pouvant être reliée à un ingrédient ou a une categorie permet plus de souplesse en permettant des cas particuliers.

4. Administrateur du site.

L'administrateur du site a sensiblement les même droits que la gérante du magasin avec quelque privilèges suppplémentaires :

- Gérer les données relatives aux informations d'établissement.
- Possibilité de supprimer un utilisateur
- Activer / désactiver un utilisateur

3.2. Règles métiers

Le nom d'utilisateur doit compter au moins cinq caractères
Le mot de passe doit compter au moins cinq caractères
L'utilisateur doit fournir une adresse email valide.
Un utilisateur recoit une demande de confirmation par email
Un utilisateur doit confirmer son inscription
Un compte est désactivé après trois essais infructueux.
L'utilisateur doit pouvoir demander la reinitialisation de son mot de passe

3.3. Interface

Les pages doivent montrer en permanence les informations de contact.

Les pages doivent proposer un lien vers l'interface de connexion / inscription.

Le site doit proposer une information quant à l'utilisation des données fournies lors de l'inscription.

4. EXIGENCES NON FONCTIONNELLES

4.1. Conception

Un nom de domaine ***laclementine.be*** sera déposé.

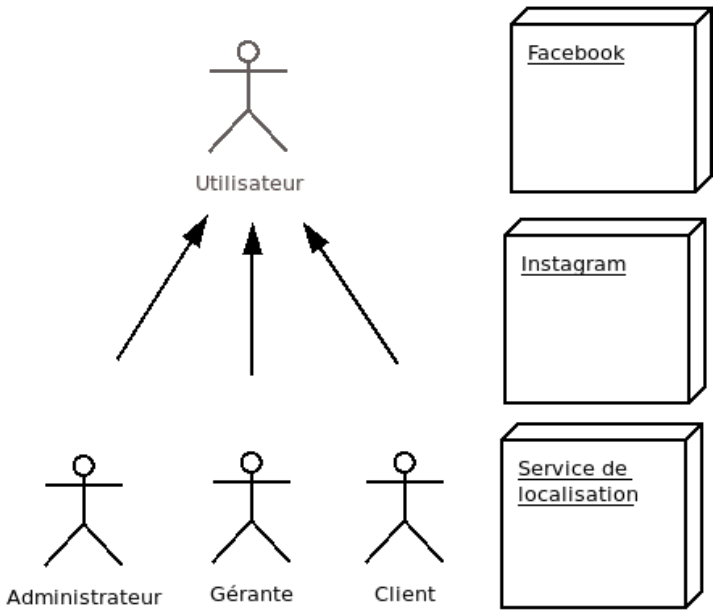
Le url affichées seront réécrites pour être compréhensible, pas de id=55 ou de parametres incompréhensibles mais des chemins vers *laclementine.be/menu* ou *laclementine.be/allergies.* L'objectif est ici d'être lisible par un humain mais aussi de gagner des points vis-à-vis des moteurs de recherche.

Le layout du site sera responsive first c'est à dire adaptés avant-tout aux smartphones. De cette façon le client aura le menu de La Clémentine dans sa poche. Le site s'adaptera ensuite aux écrans plus grands.

On essaiera de limiter autant que possible le nombre de cliques pour aller de l'arrivée de l'utilisateur sur le site jusqu'à l'action finale, l'objet de la visite. Que ce soit une information ou la réservation d'un produit.

Le site doit donner l'impression d'être rapide, un clique modifie instantanément l'affichage sans regarchement, sans passage senssible d'une page à l'autre.

4.2. Acteurs



Novitz Jean-Philippe

Les acteurs impliqués dans ce projet ainsi que les interactions entre eux sont assez simples. En voici une description.

Internaute: représente n'importe quel visiteur anonyme. Cela peut-être un nouveau visiteur, un administrateur ou un client enregistré qui ne s'est pas connecté. Peu importe, **le niveau d'accès est public.**

Client : représente un utilisateur anonyme qui a pris le temps de s'enregistré et de confirmer son inscription. **En plus d'accéder à tout ce qui est public, il a la possibilité de modifier ses données personnelles, commander un produit, signaler qu'il apprécie un produit ou l'ajouter à sa liste de produits à découvrir.**

Gérante : la gérante dispose d'un **niveau d'accès qui permet la gestion au quotidien du site internet.** La propriétaire de la boutique a, en effet, les accès suffisants pour gerer les ingrédients et leurs categories, les produits et leur classement en types, les allergies et à quelles catégories elles sont sensibles. Elle pourra acceder à la liste des réservations faites en ligne. Son rôle est important car tout changement encoé dans son interface administration aura des répercussions dans les parties publique et client.

Administrateur : **Niveau d'accès le plus haut.** L'admin a les mêmes droit d'accès que la gérante avec, en plus, la possibilité de gérer les utilisateurs. Il peut débloquer le compte d'un utilisateur distrait par trois fois, désactiver un compte ou le supprimer. Il a le pouvoir de modifier le 'role' ou droit d'accès d'un utilisateur. Par exemple pour qu'un utilisateur devienne 'gerante' c'est lui qui promeu un client en 'gerant'.

Deux acteurs non 'non humains' sont présents. Il s'agit de d'Instagram et du service de localisation Open Street Map.

Instagram : ce réseau consacré au partage de photos servira à nourir la page photo du site internet.

Novitz Jean-Philippe

Service de localisation : Pour pouvoir signaler la positionde la sandwicherie ou positioner un client il nous faut un service de localisation externe.

5. CAS D'UTILISATION

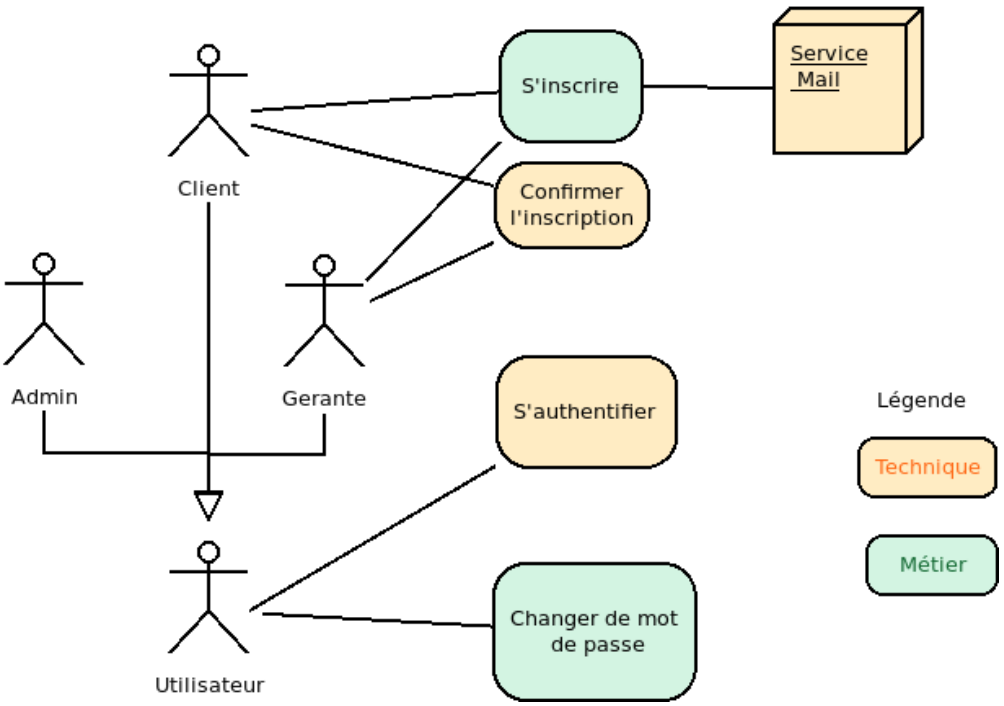
Définitions :

- Action métier : Une action faite pour l'utilisateur, cette action change quelque chose. Par exemple un internaute s'inscrit et le site l'ajoute à la liste des utilisateurs. Un utilisateur désire changer sont mot de passe il le demande.

- Action technique : actions qui ne change rien à la situation actuelle.

5.1 Inscription / connexion

5.1.1 Inscription / connexion : contexte



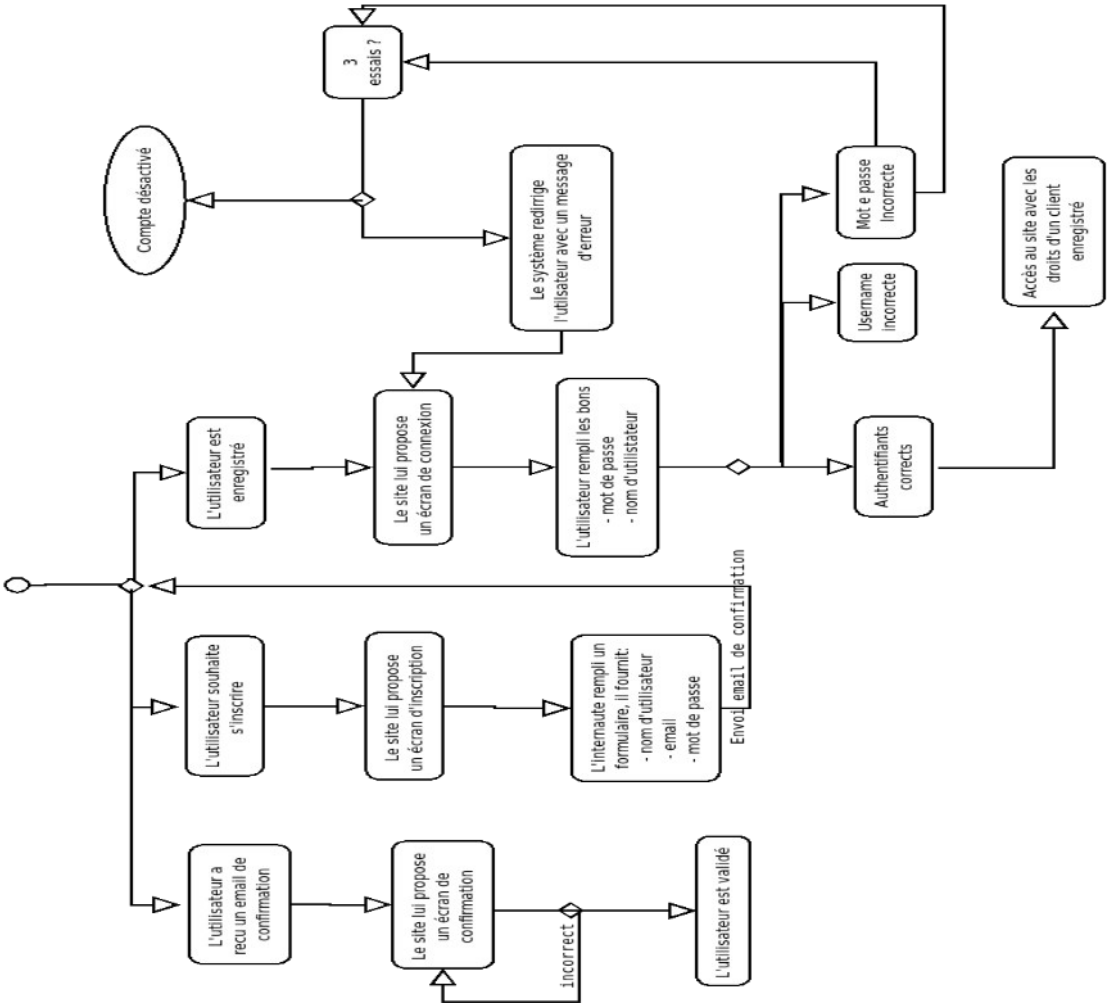
5.1.2 Inscription / connexion : scénario

Un internaute **s'inscrit** sur le site	Un internaute se rends sur la page connexion. Il indique • son nom d'utilisateur (username) • un mot de passe d'au moins cinq caractères • une adress email valide. Le systeme lèvera une alerte si le mot de passe est trop court ou le nom d'utilisateur absent. Il détectera également si l'adresse email semble ne pas être valide. Si les condition mentionnées ci-dessus ne sont pas remplies. Tout s'arrête et tout est à recommencer.
L'internaute **confirme**	L'internaute qui s'est inscrit comme décrit plus haut recoit un email lui indiquant un lien vers une url de confirmation. En cliquant sur le lien il arrive à une page de confirmation. Il s'agit d'un formulaire qui lui permet de completer son profil en indiquant • son username (confirmation) • son mot de passe (confirmation)

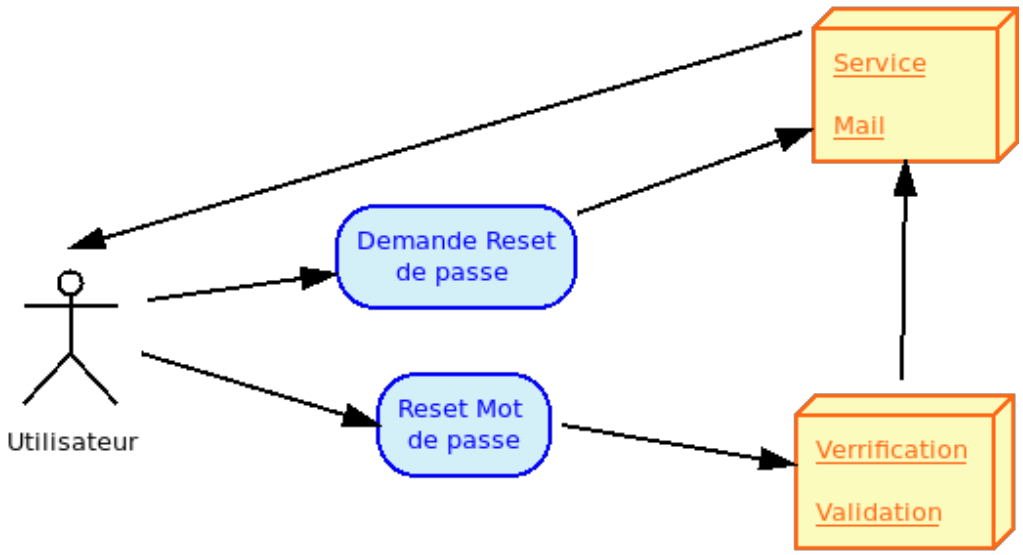
	• son adresse email (confirmation) • son nom • son prénom • son adresse • son numero de téléphone. Les username, mot de passe et email sont des informations de confirmation c'est à dire qu'elles servent à vérifier le compte. Ce sont les inormation les plus importantes. Les coordonnées servent à completer le profil et pourraient ne pas être nécesséssaire. Le numero de téléphone est important pour pouvoir contacter le client mais l'adresse email pourrait suffire. Si cette étape se déroule normalement l'utilisateur recoit un nouvel email qui lui indique un feu vert et qu'il peut se connecter.
Un utilisateur enregistré devient **administrateur**	Le site est fait de telle facon qu'un utilisateur qui s'inscrit a le status de utilisateur enregistre (client). Pour devenir 'gérante' Il suffit d'une action de l'administrateur. A priori il n'y a qu'une seule gérante mais on garde la possibilité de promouvoir un utilisateur en gérant pour le cas ou l'entreprise engagerait une personne, il faudrait lui donner la possibilité

	de modifier un produit ou d'accéder au tableau de bord. Au niveau de la sécurité, aucune action sensible n'est possible avec ce status.
Un utilisateur **se connecte** sur le site	L'utilisateur e rends du la page de connexion (login). Il entre sont username et son mot de passe . Les authentifiants sont les bons => ile est redirrigé vers la page d'acueil en ayant le status d'utilisateur enregistré. Les authentifiants ne sont pas les bons => le site affiche un message d'erreur.

Novitz Jean-Philippe

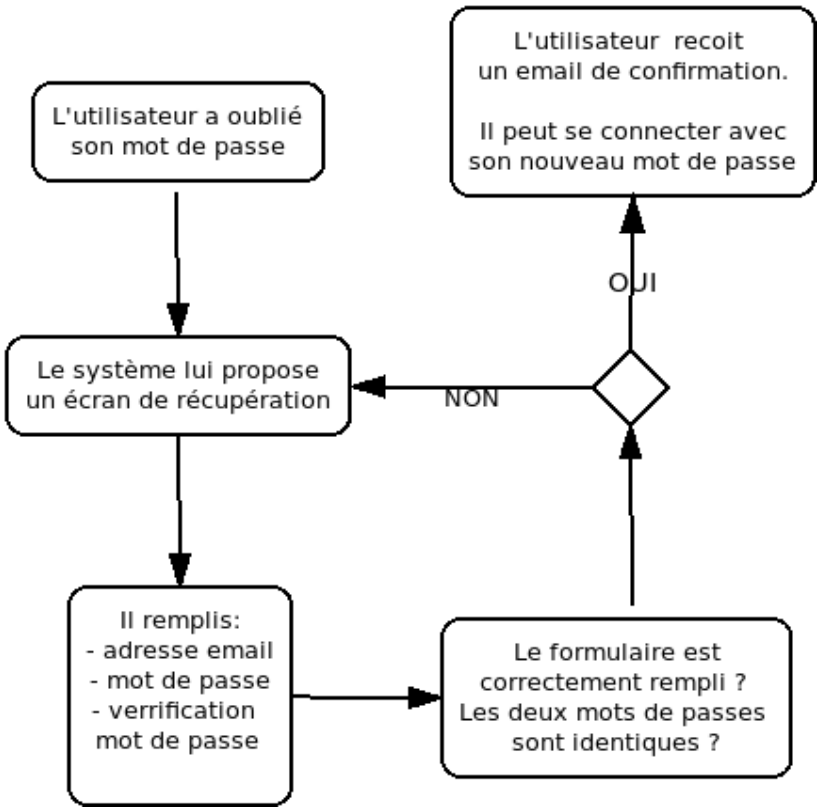


5.2.1 Récupération mot de passe : contexte



5.2.2 Récupération mot de passe : scénario

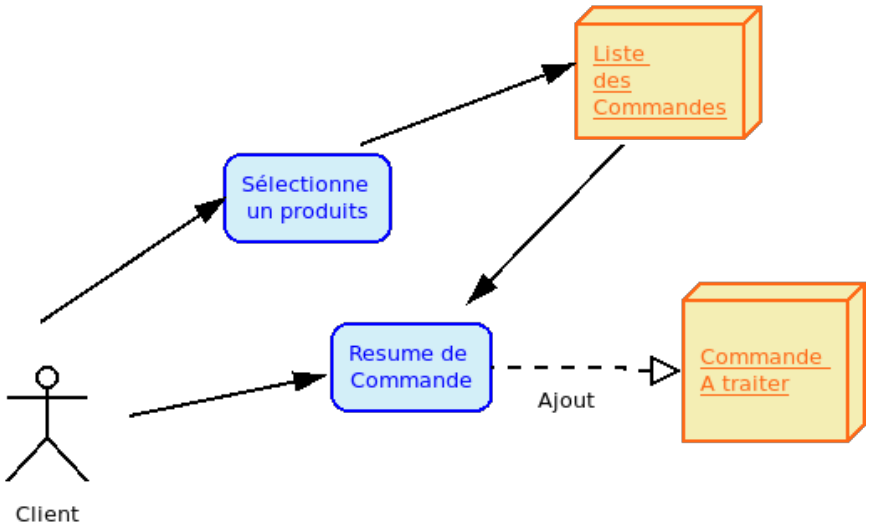
Novitz Jean-Philippe



L'utilisateur demande la récupération de son mot de passe	L'utilisateur a oublié son mot de passe et souhaite le réinitialisé. Il se rends sur une page de demande de récupération de mot de passe. Il indique son adresse email. Le système vérifie que l'adresse email correspond à un utilisateur. S'il y a correspondance un email avec un lien est envoyé à l'utilisateur
L'utilisateur remet à jour son mot de	L'utilisateur a recu un email lui proposant

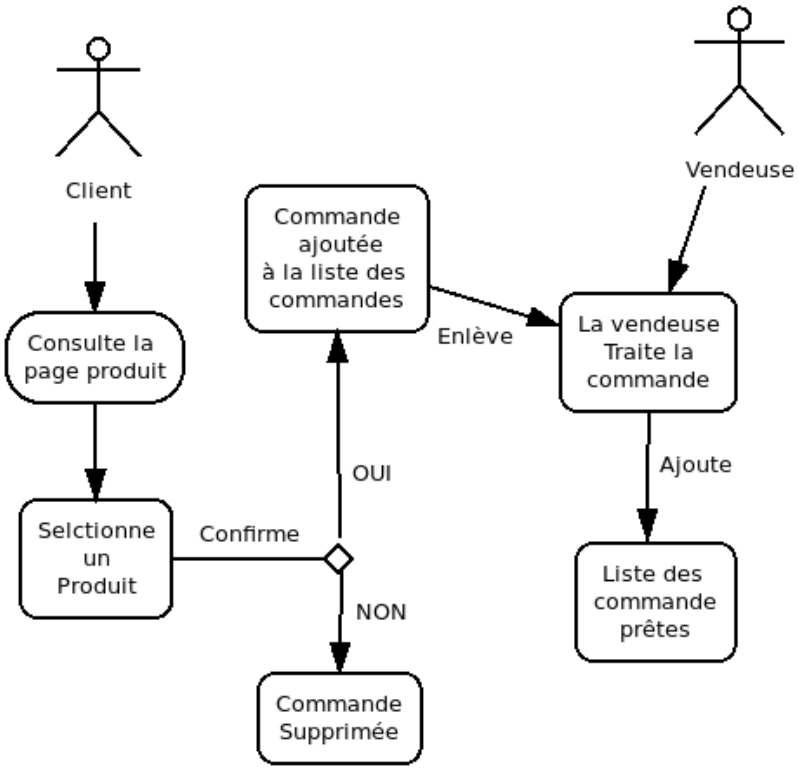
passe	de changer son mot de passe. Un lien le renvoi vers la page de reinitialisation. Il y indique son adresse, email et un mot de passe (et une deuxième fois le mot de passe pour verification) Le système verifie que les deux mots e passes sont valides et identiques. Si tout est correct, le mot de passe est changé et un email est envoyé pour confirmation.
L'utilisateur se connecte avec son nouveau mot de passe.	L'utilisateur recoit un email, son nouveau mot de passe. Sa connection est à nouveau fonctionnelle.

5.2.1 Commande de produit : contexte



5.3.2 Commande de produit : scénario

Novitz Jean-Philippe

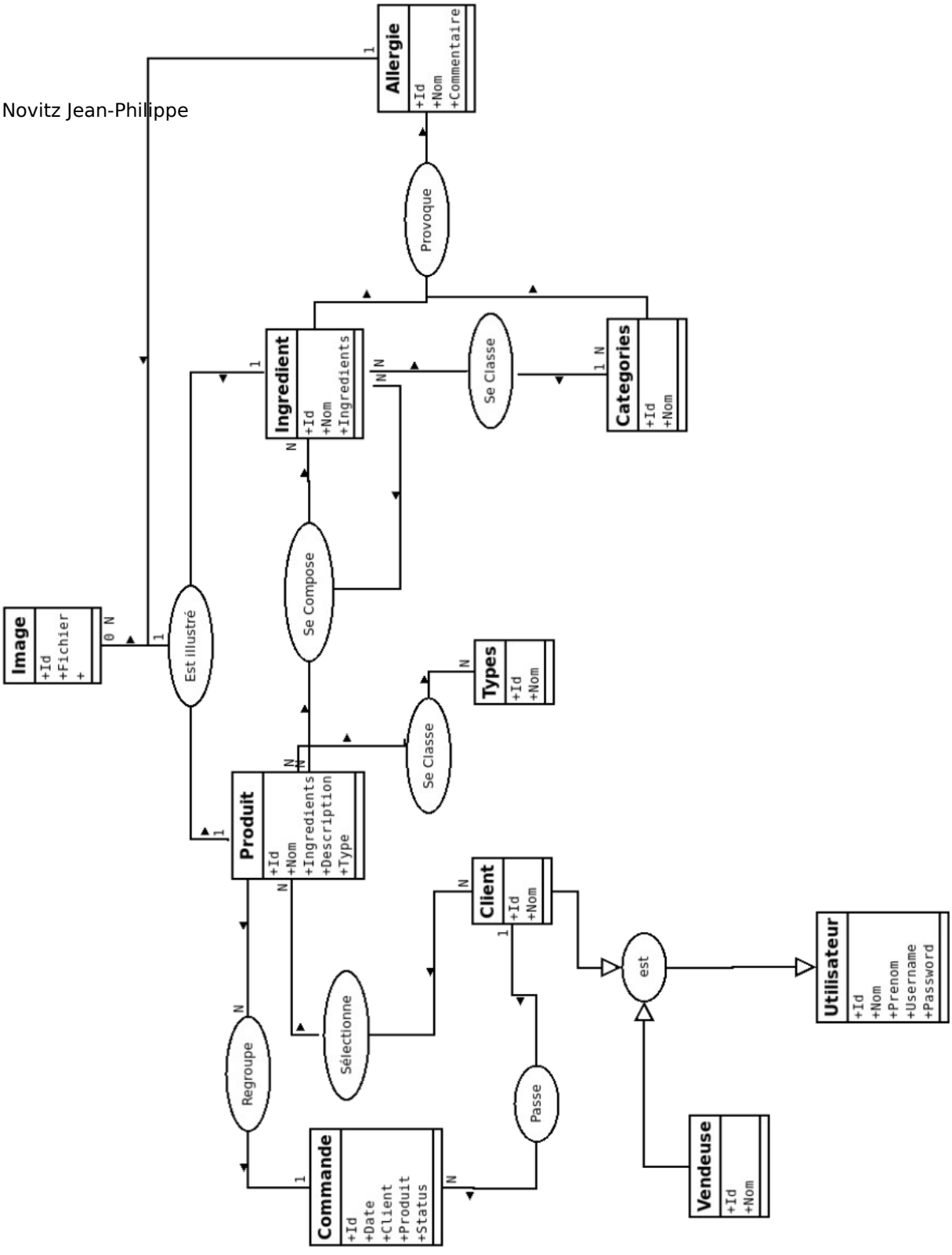


Novitz Jean-Philippe

Le client visualise les produits	Une page produit permet au client de visualiser l'ensemble des produits.
Le client ajoute un produit à sa commane	Chaque produit dispose d'un bouton 'ajouter' en cliquant le client ajoute le produit à sa liste des produits commandés.
Le client valide sa commande.	Le Client visualise l'ensemble de sa commande, il peut choisir son pain ou ses crudités et valider sa commande.
La vendeuse exécute la commande	Le système regroupe l'ensemble des commandes. Celles-ci sont visibles dans un tableau de bord. La vendeuse sélectionne la commade et la signale comme 'classée'. La commande est prête et le client n'a plus qu'à venir la chercher.

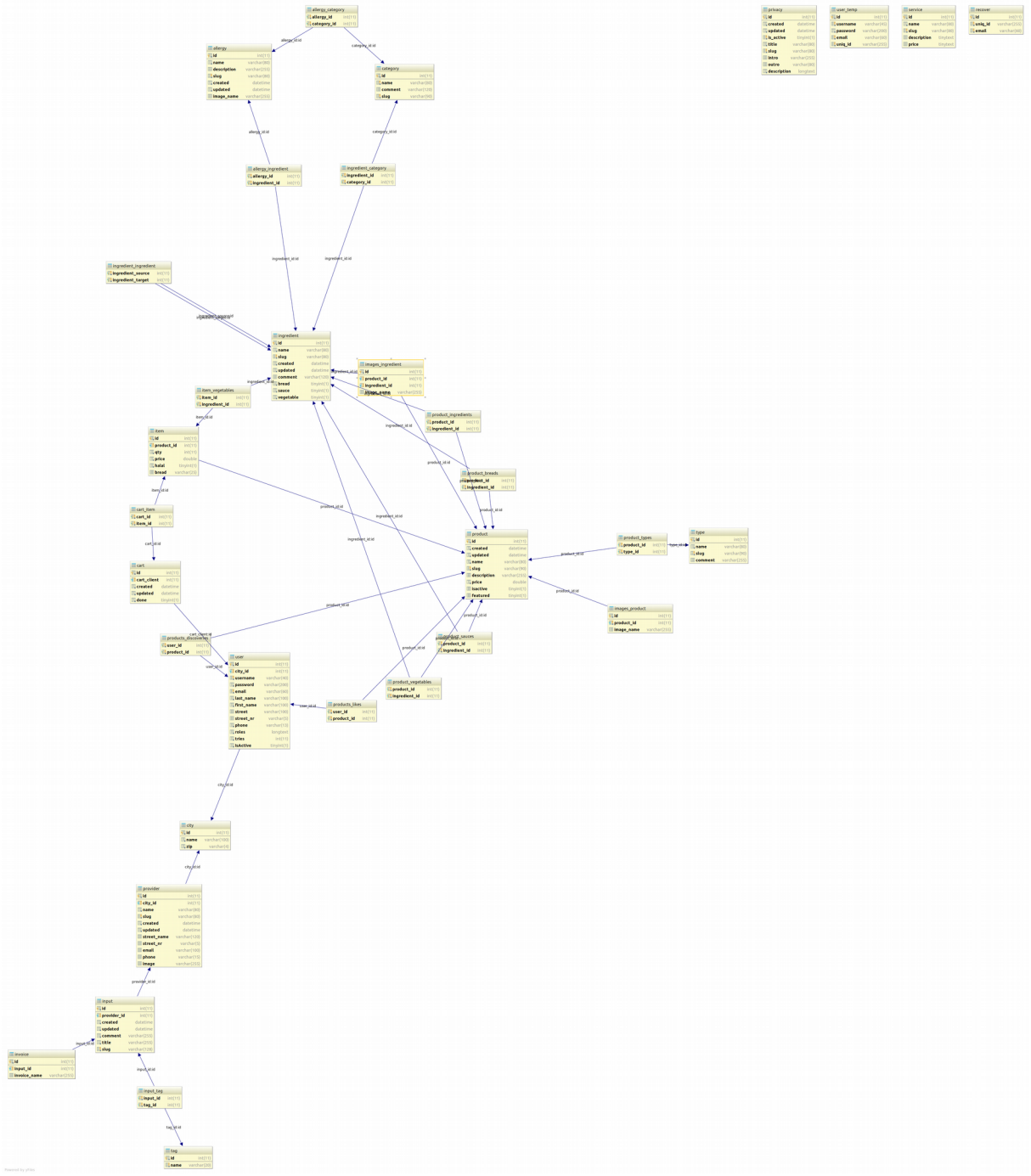
6. DESCRIPTION DE LA BASE DE DONNÉES

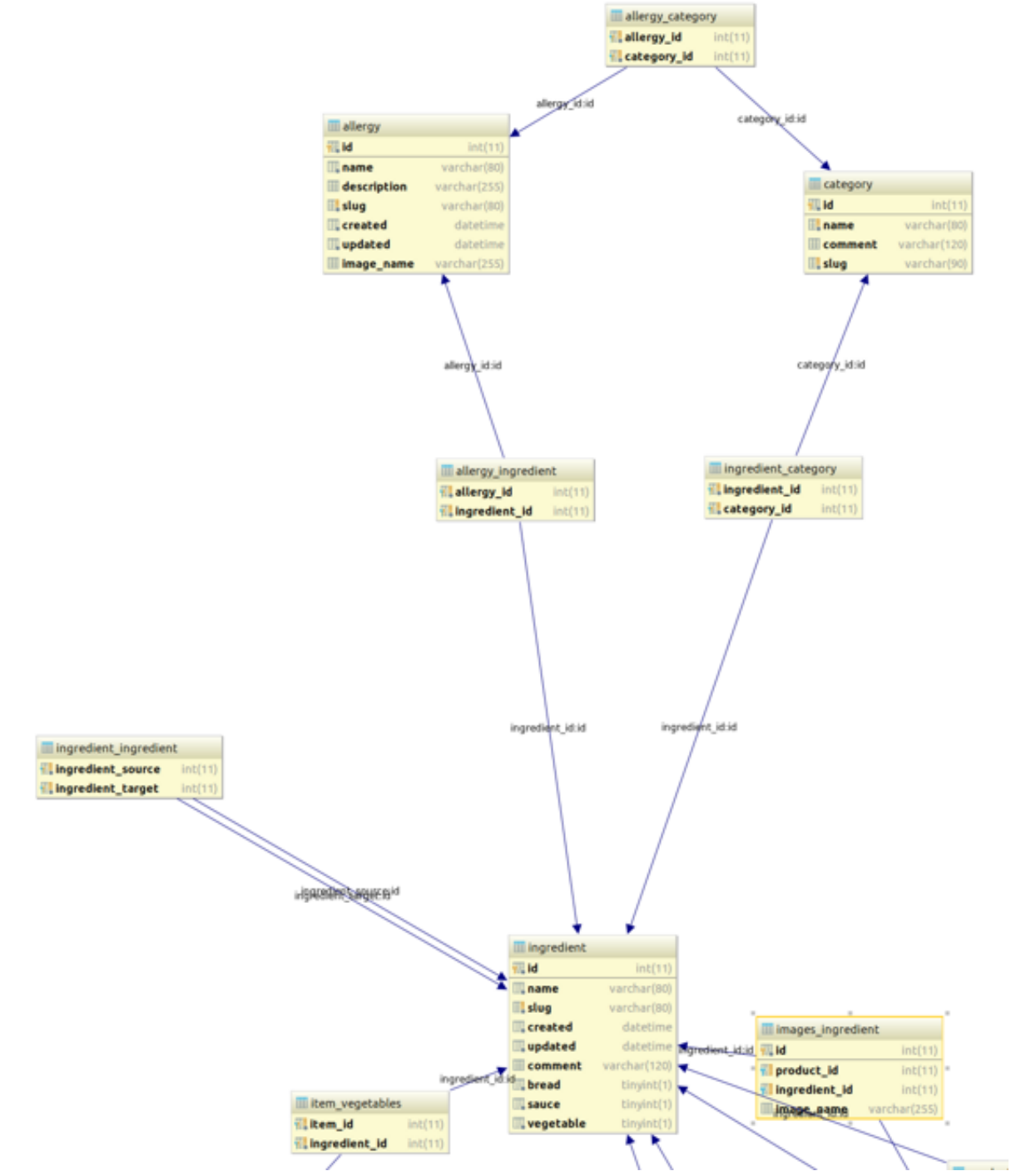
6.1. Description de la base de donnéesSchéma conceptuel


Allergie
+Id
+Nom
+Commentaire
Provoque
Ingredient
+Id
+Nom
+Ingredients
Se Classe
Categories
+Id
+Nom
Se Compose
Image
+Id
+Fichier
+
Est illustré
Types
+Id
+Nom
Se Classe
Produit
+Id
+Nom
+Ingredients
+Description
+Type
Client
+Id
+Nom
est
Utilisateur
+Id
+Nom
+Prenom
+Username
+Password
Regroupe
Sélectionne
Passe
Commande
+Id
+Date
+Client
+Produit
+Status
Vendeuse
+Id
+Nom

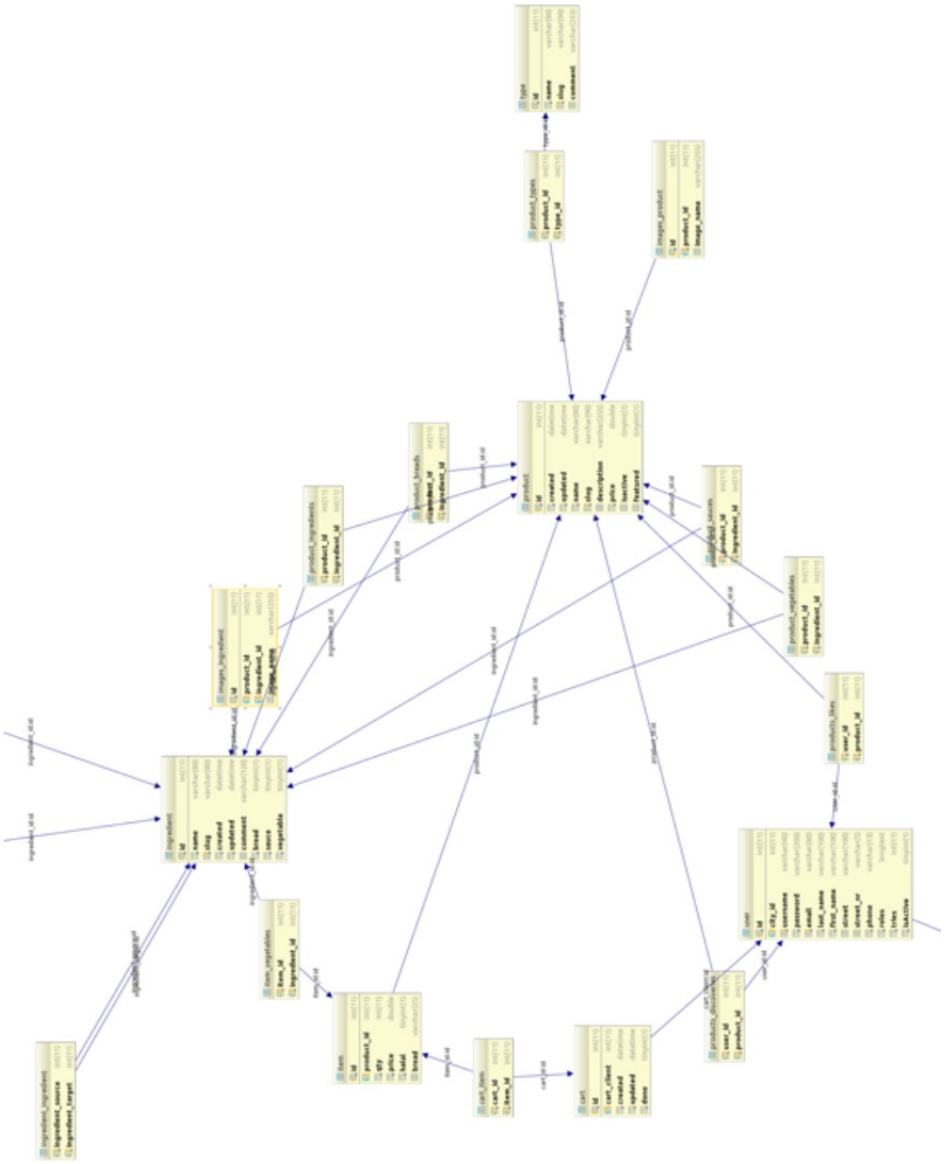
6.2. Schéma physique

Novitz Jean-Philippe

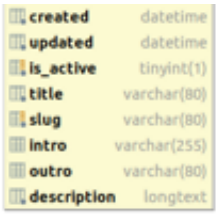


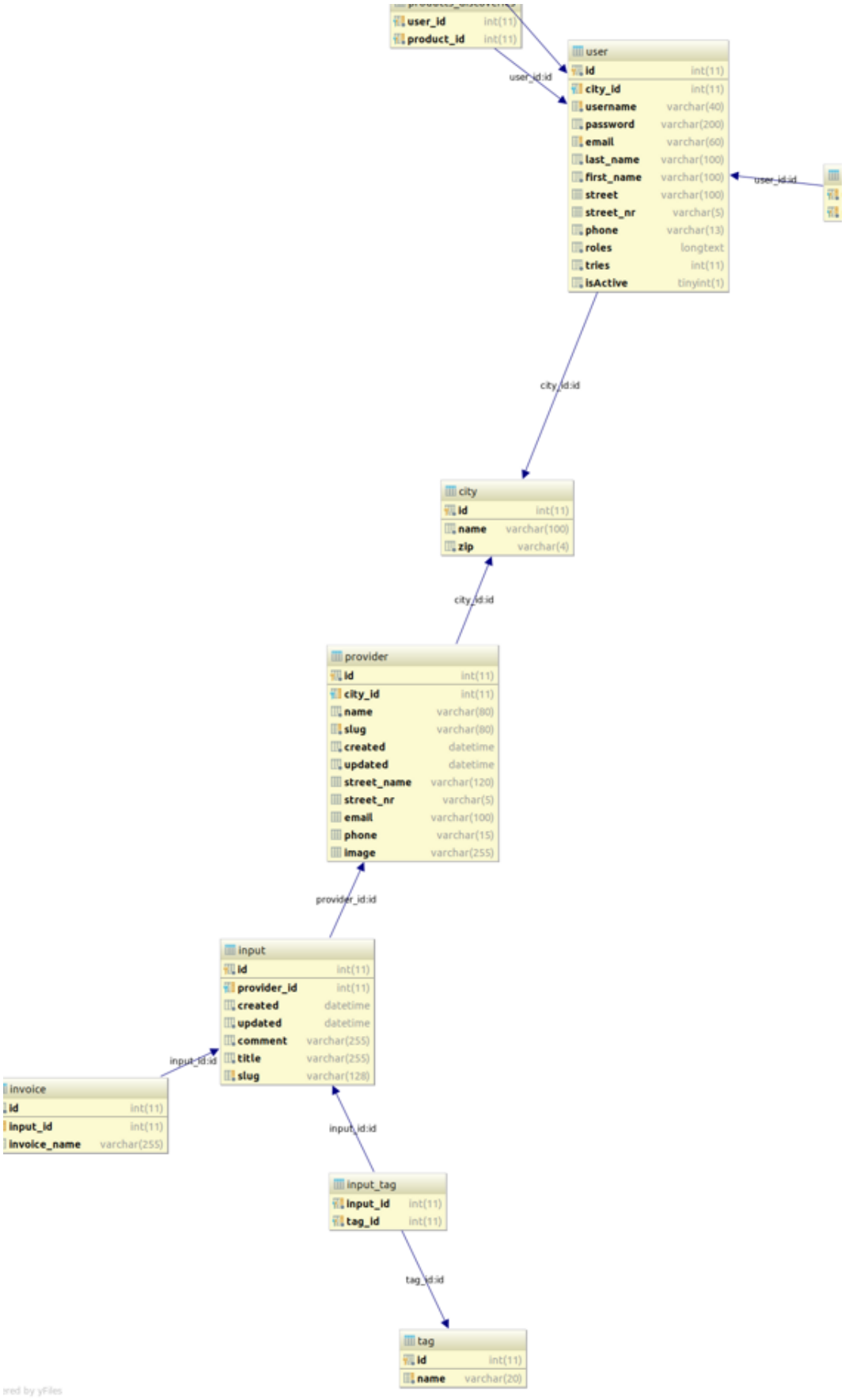

allergy_category
allergy_id int(11)
category_id int(11)
allergy_id:id
category_id:id
allergy
id int(11)
name varchar(80)
description varchar(255)
slug varchar(80)
created datetime
updated datetime
image_name varchar(255)
category
id int(11)
name varchar(80)
comment varchar(120)
slug varchar(90)
allergy_id:id
category_id:id
allergy_ingredient
allergy_id int(11)
ingredient_id int(11)
ingredient_category
ingredient_id int(11)
category_id int(11)
ingredient_id:id
ingredient_id:id
ingredient_ingredient
ingredient_source int(11)
ingredient_target int(11)
ingredient
id int(11)
name varchar(80)
slug varchar(80)
created datetime
updated datetime
comment varchar(120)
bread tinyint(1)
sauce tinyint(1)
vegetable tinyint(1)
images_ingredient
id int(11)
product_id int(11)
ingredient_id int(11)
image_name varchar(255)
item_vegetables
item_id int(11)
ingredient_id int(11)

Novitz Jean-Philippe



Novitz Jean-Philippe




user_id int(11)
product_id int(11)
user_id:id
user
id int(11)
city_id int(11)
username varchar(40)
password varchar(200)
email varchar(60)
last_name varchar(100)
first_name varchar(100)
street varchar(100)
street_nr varchar(5)
phone varchar(13)
roles longtext
tries int(11)
isActive tinyint(1)
user_id:id
city_id:id
city
id int(11)
name varchar(100)
zip varchar(4)
city_id:id
provider
id int(11)
city_id int(11)
name varchar(80)
slug varchar(80)
created datetime
updated datetime
street_name varchar(120)
street_nr varchar(5)
email varchar(100)
phone varchar(15)
image varchar(255)
provider_id:id
input
id int(11)
provider_id int(11)
created datetime
updated datetime
comment varchar(255)
title varchar(255)
slug varchar(128)
input_id:id
invoice
id int(11)
input_id int(11)
invoice_name varchar(255)
input_id:id
input_tag
input_id int(11)
tag_id int(11)
tag_id:id
tag
id int(11)
name varchar(20)

7.7. Présentation des problèmes et solutions envisagées.

7.1. Envoyer des emails en local.

Mon interface prévoit d'envoyer des emails lors de l'enregistrement d'un utilisateur. Cela m'a amené a me poser une question et m'a confronté à un problème.

La question : Quels sont les outils disponibles pour le faire en local car si je veux tester les envois d'email je voudrais ne pas le faire réellement, cela n'aurait pas de sens.

J'ai rapidement trouvé la réponse : **mailtrap.**

Mailtrapo.io est un site qui permet de tester les envois d'émails dans un environnement adéquat, je suis certain que l'envoi de mails fonctionne. Grâce à une boite 'inbox' virtuelle je vois si les envois sont arrivés, je peux les consulter.

Le problème : Comment envoyer des emails techniquement.

J'ai eu l'occasion de tester mailtrap.io sur un système d'exploitation sous Windows. Pour ce travail de fin d'étude j'ai décidé d'utiliser Linux et mon premier essai ne s'est pas averé concluant, j'ai dû effectuer un peu de configuration pour atteindre mon but. C'est ma solution.

L'idéal aurait été d'installer des applications telles que Postfix ou Sendmail . Ces solution auraient été parfaites mais (pour l'instant) en dehors du fait de tester mes email je n'en aurait pas d'autres utilisation.

SSMPT m'a parrut la solution idéale, c'est une petit programme qui se charge simplement de relayer les mails vers mon fournisseur internet. C'est assez léger et si je remplace les coordonnées, login et mot de passe de mon fournisseur d'accès à internet par celles fournies par mailtrap cela fait ce que je veux.

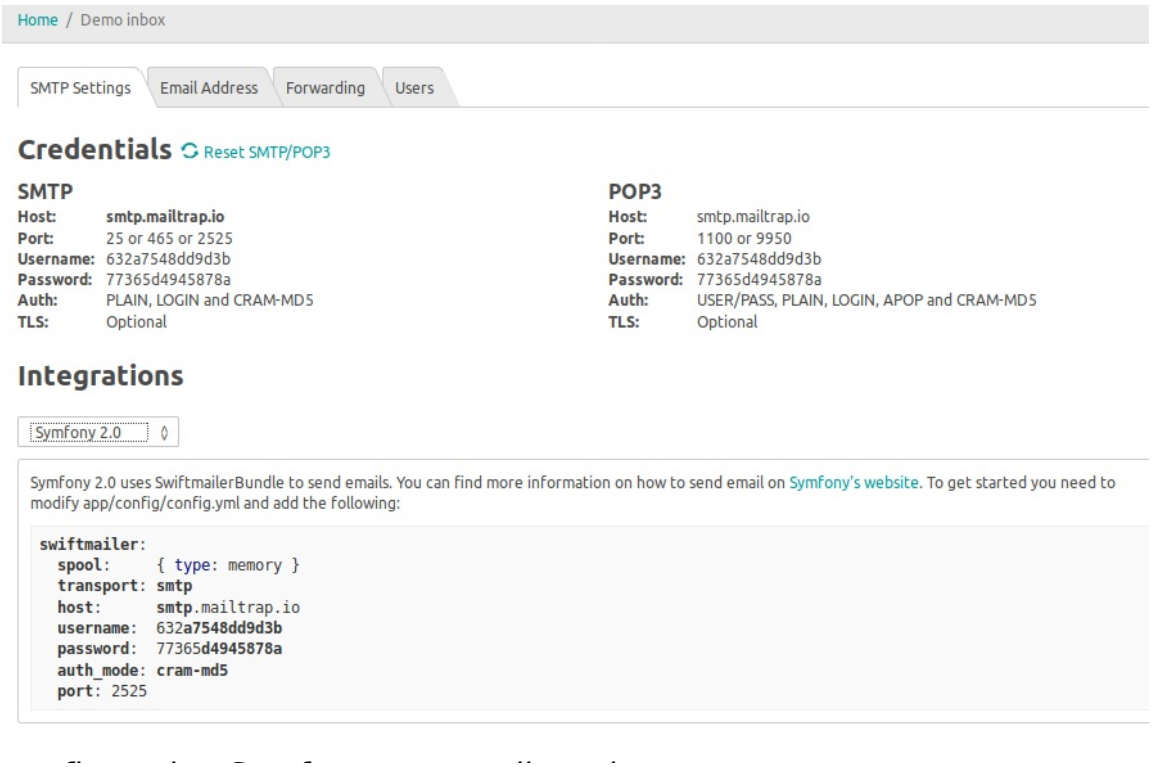
SMTP s'installe via la ligne de commance sudo apt-get install ssmtp, il suffit ensuite de configurer ssmtp.conf pour permettre une connection. Après un test via mailx (paquet qui permet d'envoyer des emails en ligne de commande.

Pour être complet j'ajoute que Symfony utilise SwiftMailer pour l'envoi de mail. La configuration de Swift mailer se fait via le fichier App/config/config.yml, les paramètres à modifier se trouvent dans le fichier parameters.yml.

Le principe est qu'il faut instancier un 'transport', une variable 'message'.
On embarque dans la variable message tout ce dont elle a besoin : un destinataire, un objet, un message (qui peut être du texte ou un template twig).

Le transport utilise une méthode telle que SwiftMailer utilise smtp, sendmail ou gmail pour envoyer le message. La méthode send a été dépréciée depuis la version 5.4.5 de SwiftMailer.

J'ai placé les envois de mails dans de petit services qui les envoient au bons moments.



configuration sSMTP (http://www.ced-info.com)

```
### CONFIGURATION GENERALE ###

MailHub=smtp.de.votre.fai       # Serveur SMTP vers lequel forwarder les mails
RewriteDomain=                  # Domaine depuis lequel est envoyé le mail
                                # (on peut le laisser vide)
Hostname=nom.de.votre.serveur   # Nom de la machine
FromLineOverride=yes            # Ré-écriture de l'expéditeur (champ from)
Root=srv-ced@ced-info.com       # redirige les mails à destination de "root"
                                # vers srv-ced@ced-info.com

### CONFIGURATION DE L'AUTHENTIFICATION ###

UseTLS=yes                       # Utilisation d'une connexion sécurisée SSL
                                # ou TLS (si votre FAI l'utilise sinon mettre no
AuthUser=votre.utilisateur      # Nom d'utlisateur pour l'authentification SMPT
AuthPass                        # Le mot de passe correspondant
```

7.2 Sélectionner les aliments ne provoquant pas certaines allergies.

Je souhaite proposer à mes visiteurs un menu qui s'adapte aux allergies. En sélectionnant une allergie l'interface propose un menu sans les produits suspects.

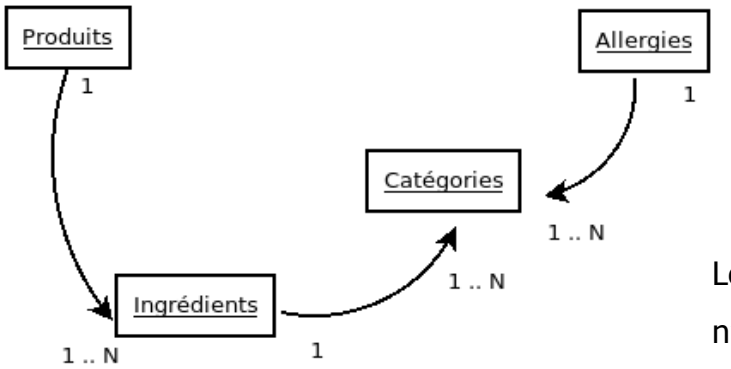
Je pose le problème :

- j'ai une entité qui représente mes produits. Les produits se composent d'ingrédients ;
- J'ai une entité qui représente mes allergies.
- Je veux mettre de côté des éléments d'une des entités (produits) qui correspondent à la deuxième (les allergies).

Ce dont j'ai besoins : un dénominateur commun.

L'élément qui relie un produit à une allergies est l'ingédient car un ingrédient provoque une allergie. Mais sélectionner chaque ingrédient peut être fastidieux car pour allergie allergie je devrais sélectionner chaque ingrédient.

J'ai décidé de regrouper les ingrédients par catégorie. Après réflexion, catégorie me convient bien : à l'encodage d'un nouvel ingrédient, je précise sa catégorie, en général une catégorie n'est pas provoquée par un aliment précis mais 'plutôt' par un groupe d'aliment. Si je suis allergique au lactose je ne supporterais pas les produits à base de lait.



Les ingrédients n'ont pas conscience de quels produits il font partie.

Les catégories n'ont pas conscience des allergies dont elles peuvent être la cause, non plus quels ingrédient elles regroupent.

La démarche est d'avoir deux point d'entrée dans ce schéma :

- du point de vue du produit
- du point de vue des allergies.

Comment ?

Au départ, j'ai choisi d'utiliser Symfony pour le backend ce qui sous entends que les entités Doctrine contiennent mes datas. La meilleure façon de 'transmettre' ces information est d'uliser une api Rest.

Rest permet d'être indépendant au niveau de l'affichage. Si mon client décide de changer la technologie par exemple passer de vuejs à react, angular ou toute autre framework cela ne posera pas de problème. La méthode restera la même : consommer l'api rest.

Ce qui se passe :

Le framework vue.js utilise un système de store (vuex) pour stocker les données utiles mon application. Vuex envoie (via des actions) des requêtes http pour interroger les api rest de symfony.

« Je voudrais la liste des allergies»

« Je voudrais la liste des produits »

« Je voudrais la liste des ingrédients »

Les controllers Symfony recoivent les demandes et interrogent la base de donnée mysql via Doctrine. Les controllers renvoient une réponse au format Json.

Si tout s'est bien déroulé Vuex recoit les datas, sinon il recoit un message d'erreur. Arpès cela le store contient simplement des données (state) des getters et des setters (commit).

Quand le client se rends sur la liste des produits :

→ l'app récupère dans le store

- la liste des produits avec photos et infos
- la liste des allergies
- la liste des produits et la liste des allergies.

Quand le client sélectionne une allergie

→ vue.js ne récupère aucune autre information

→ vue.js analyse les données :

- Quelles catégories provoquent cette allergie
- Quels sont les ingrédients incriminés
- Quels produits contiennent ces ingrédients

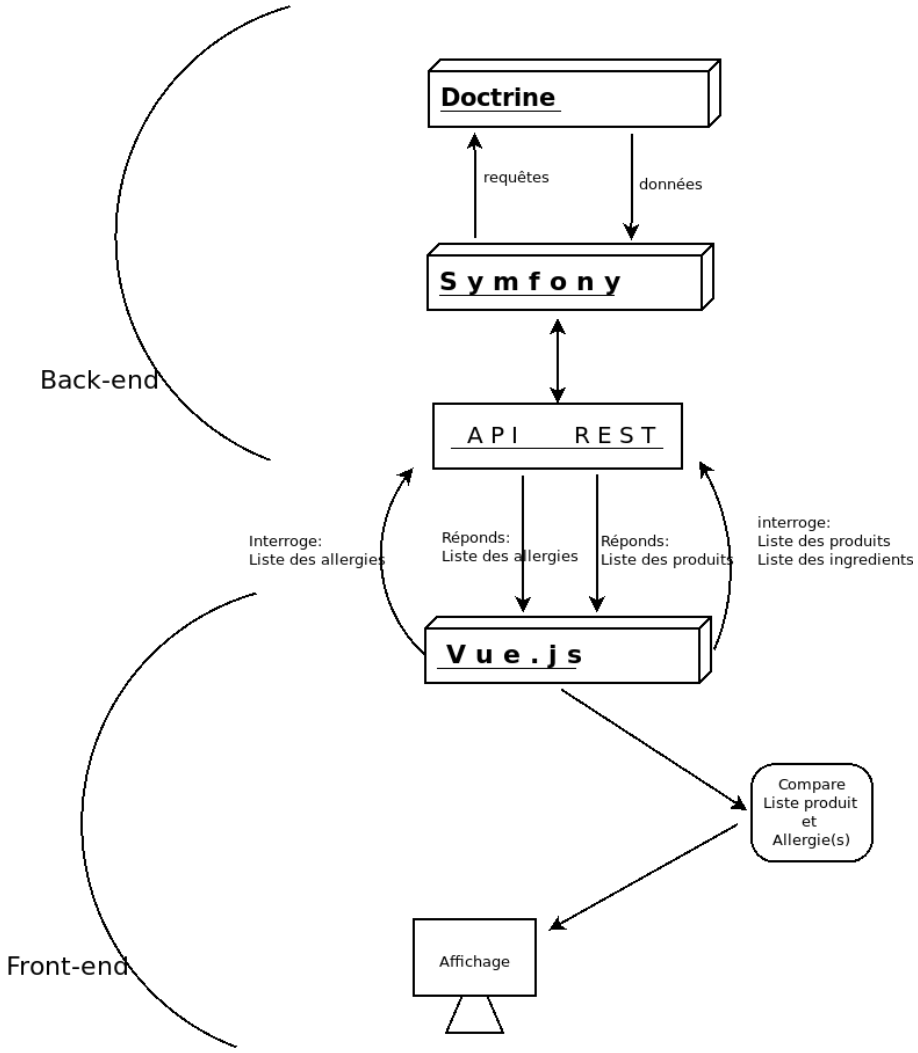
Vue.js utilise le data binding pour modifier l'affichage en temps réel.

En fait, en arrière plan le js analyse les donées en comparant simplement des objets et/ou des tableau entre eux pour obtenir les information nécessaires.

Les data fournies soit par des variables dites 'computed' ou par des valeur renvoyées par des fonction sont liées à leur équivalent dans la partie html.

J'affiche toujours la liste de produit que je trie ou pas et la liste se modifie en temps réel.

Novitz Jean-Philippe



7.3 Google Map devient payant

Pour ce projet j'ai passé du temps a implémenter google map pour proposer une geolocalisation. Quelque jours plus tard Google Map devenait payant.

J'ai recommencé le même travail en utilisant Open Street Map.

Le principe est d'afficher une carte et de programmer des marqueurs que l'on place sur la carte. Cela me permet d'aider le visiteur à se situer, a situer la sandwicherie par rapport à son quartier.

7.4 La mise en production

La mise en production n'est pas quelque chose de très compliqué à condition de ne pas le faire un vendredi soir. Mais comme toute chose nouvelle les premiers pas sont hésitants et les erreurs sont là, au bord du chemin pour nous décourager.

J'ai surmonté cette dernière épreuve en raisonnant (je crois) comme un développeur, en lisant de la doc et en faisant preuve de pragmatisme et de réflexion.

Je me suis dis quelles sont mes contraintes :

- J'utilise Symfony pour une partie, c'est un frameworks qui necéssite sont environnement, j'ai dû l'installer au départ. Je suis suceptible de devoir le réinstaller pour que le site fonctionne.

- La partie publique est en vue.js, c'est un framework javascript qui utilise lui aussi un environnement. Vue-cli utilise est 'traduit' par webpack, ce qui le force à passer par une phase de build avant de pouvoir être utilisé en production.

- Questions :

    1. si 'npm run dev' lance un serveur de developpement, que ce soit en local ou sur un serveur distant, comment se comporte une page html/vue sur un 'vrai' serveur.

    2. Comment faire avec ssh et ovh ?

Quels sont mes outils :

- Depuis le début de mon aventure j'ai appris à utiliser la ligne de commande, ma nouvelle amie. La première chose à faire était pour moi de regarder comment me connecter sur mon espace ovh. La connexion établie ce n'est pas très différent de mon espace en local.

  j'ai essayé d'y installer :

    1. composer

    2. git

- J'ai bien avancé, j'ai beaucoup d'élements qui me permettent d'utiliser symfony, reste à intégrer vue.js :

- Tout fonctionnerai à la perfection si je placais vue dans le répértoire web et totalement indépendant de symfony avec sa propre url avec une extension html.

- Je ne me contente pas de cela, je veux utiliser twig même si c'est une utilisation minimum : tout se passera entre les balises *<div id="> ,* entre ces frontières twig n'est pas très utile mais je veux pouvoir intégrer des choses via twig.
  Par exemple si je veux ajouter des choses avant ces balise, quelque chose dans le header ou une information dans le bas de page.

- La solution utilisée est de modifier les paramètres de webpack :

  - j'ai placé le dossier vue de développement en dehors de /web

  - les dossier statiques et publiques de vue, par exemple les fichiers js, css et les images sont dans /web

  - la compilation crée un fichier index.html qui se accompagne les fichiers statiques. J'ai modifier cela pour en faire un fichier index.twig qui se trouvera, lui, dans le dossier app/ressources. A partir de là je connais : je peux lier lier les fichiers js nécessaire avec asset ou assetics.

Il me reste à faire un test pour verifier que tout fonctionne, puisque j'ai régulièrement fais des *git push* pour sauvegarder mon travail … et si je faisais :

- un *git pull* à partir de mon dossier distant

- un composer install pour installer toutes les dépendances nécéssaires

- un php bin/cosole cache:clear –env=prod est indispensable et je m'en suis vite rendu compte : sans cela rien ne fonctionne, tout comme il faudra que je chasse tous les var_dump qui trainent avant de mettre en production.

J'ai fais le tour du fonctionnement de la mise en production, reste à m'organiser, j'ai travaillé sous la forme de petite 'sprints' :

- Je me défini une tâche à accomplir
- Je crée une branche si j'estime cela nécessaire pour cette tâche (en fonction de la complexité)
- Je travaille le backend symfony éventuelement nécéssaire
- Je travaille le front end vue.js nécéssaire avec le serveur de développement.
- Je compile et verifie que ce la fonctionne en local
- Je pousse mon travail sur git
- Je rappatrie mon travail sur serveur distant, cache-clear fait le ménage
- si j'ai des pages blanches, je regarde les messages d'erreur qui se trouve dans les logs, je refais le cycle pour corriger ces erreurs jusqu'à satisfaction.

Quid des fichiers lourd et de la base de donnée ?

A ce stade de mon évolution je profite d'avoir exactement l'interface phpMyAdmin en local et distant pour me contenter d'un export du local et d'un import en distant.

Pour les fichiers images par exemple, j'utilise le ftp classique car git n'aime pas trop les fichiers lourd, cela me convient car je n'en ai pas énormément à traiter.

Cette façon de travailler me permet d'à chaque fois travailler pour un objectif atteignable, d'avoir une série d'objectifs clairs. Il utile d'avancer régulièrement, il est motivant d'en avoir la sensation.

8. CONCLUSION

Ce projet a été l'occasion de travailler sur mon projet e grande ampleur.

Il m'a permis de me situer par rapport à mon évolution tout au long de la formation.

J'ai eu l'occasion de travailler le backend, le front end, l'intégration, plusieurs langages.

Ce projet n'est pas parfais mais je suis fiers de mes effort et du résultat.

Merci.

Novitz Jean-Philippe

Bibliographie.

Envoyer des mails :

- Site Swift Mailer :
  http://swiftmailer.org/docs/sending.html#using-the-sendmail-transport

- Symfony Book : configuration de Swift Mailer
  http://symfony.com/doc/current/reference/configuration/swiftmailer.html

- ced-info.com: Envoyer des mails simplement avec sSMTP
  http://www.ced-info.com/administration-reseaux/envoyer-des-mails-simplement-avec-ssmtp

- sSMTP : documentation ubuntu :
  https://doc.ubuntu-fr.org/ssmtp

- Symfony Book : How to send Mail
  http://symfony.com/doc/current/email.html

- Mailtrap.io : site officiel
  https://mailtrap.io/

Mettre son projet Symfony en production :

- Openclassroom : 'déployer son site Symfony en production' https://openclassrooms.com/courses/developpez-votre-site-web-avec-le-framework-symfony2/deployer-son-site-symfony2-en-production

- Symfony.com : How to deploy a Symfony Application : https://symfony.com/doc/current/deployment.html

Symfony Book : Utilisation de Assetic

http://symfony.com/doc/current/assetic/asset_management.html

Vich uploader:

https://github.com/dustin10/VichUploaderBundle/blob/master/Resources/doc/usage.md C

https://symfony.com/doc/master/bundles/EasyAdminBundle/integration/vichuploaderbundle.html

https://github.com/skornev/vichbundle-sandbox-not-cp-symfony3.3/blob/master/src/AppBundle/Entity/InvoiceAttachment.php

Mime types:

https://symfony.com/doc/current/reference/constraints/File.html#mimetypes

https://www.freeformatter.com/mime-types-list.html#mime-types-list

https://stackoverflow.com/questions/34019407/how-to-use-mimetype-assert-with-vichuploader/34281440

Validation:

Novitz Jean-Philippe

https://symfony.com/doc/current/reference/constraints.html

Novitz Jean-Philippe

Table des matières