

CFPT

VEGAN *Yums*

Documentation Technique

Roxanne Grant

19/06/2017

Table des matières

1. Introduction	3
2. Cahier des charges	3
2.1 Organisation.....	3
2.2 Objectifs	3
2.3 Description détaillée du projet.....	4
2.4 Modèle conceptuel de base.....	5
2.5 Planning initial	5
3. Analyse fonctionnelle	6
3.1 Technologies	7
3.2 Maquette	8
3.3 Fonctionnalités du site	11
3.4 Définition des pages et des fenêtres modales.....	12
3.4.1 La page index.php	12
3.4.2 La fenêtre modale de connexion	12
3.4.3 La fenêtre modale d'inscription.....	12
3.4.4 La fenêtre modale d'ajout de recette.....	12
4. Analyse organique.....	13
4.1 Plan du site	13
4.2 Arborescence de fichier	14
4.3 Requêtes principales	15
4.4 Fonctions php de gestion	15
4.4.1 FormatIngredients(\$ingredients)	15
4.4.2 VerificationAdd(\$param)	16
4.4.3 VerifyImg(\$files).....	16
4.4.4 MoveImg(\$files).....	16
4.4.5 IsEmpty(\$param)	17
4.4.6 SignedIn(\$isAdmin).....	17
4.4.7 KeepModalOpen(\$alert,\$modalname)	17
4.4.8 ListIngredients(\$listIngredients)	17
4.4.9 RestrictLengthDescrip(\$descrip)	17

4.4.10 DeleteImg(\$dossier,\$value)	17
4.4.11 IsFav(\$value, \$favorite)	17
4.4.12 ShowComment(\$value, \$comment)	18
4.5 Architecture du projet	18
4.6 Diagramme de classe	19
4.7 MLD	19
4.8 Dictionnaire de données	20
4.9 Gestion des favoris	21
4.10 Gestion de la validation par l'administrateur	22
4.11 Gestion des commentaires	23
4.12 Ajout de recette par l'utilisateur	23
4.13 Auto complétion et recherche	23
4.14 Gestion des filtres	25
5. Rapport de tests	27
6. Conclusion	31
6.1 Différence entre le planning prévu et réel	31
6.1.1 Planning final	31
6.2 Améliorations possibles	32
6.3 Bilan personnel	32
7. Bibliographie	33
7.1 Sites utilisés pour la programmation	33
7.2 Recettes utilisées	33
7.3 Aide reçue	33
8. Table des figures	33
9. Annexes	34

1. Introduction

A travers cette documentation, nous allons voir les étapes nécessaires à la réalisation du projet VeganRecipes. Ce projet est fait dans le cadre de mon travail de TPI (Travail Pratique Individuel). Le but de ce site web est de pouvoir créer et retrouver des recettes véganes¹. Cette plateforme ne peut fonctionner que s'il y a des utilisateurs qui y ajoutent des recettes et utilisent le site comme un grand livre de recettes. Les utilisateurs ajoutent des recettes qui sont ensuite validées par l'administrateur. Les recettes validées peuvent être vues par tous les utilisateurs qu'ils soient connectés ou non. Les utilisateurs ayant un compte peuvent aussi mettre des recettes en favoris pour les retrouver plus facilement.

J'ai choisi de faire ce projet car je suis moi-même végane et j'aime beaucoup cuisiner et surtout manger. Il y a énormément de sites qui proposent aux utilisateurs d'ajouter des recettes mais il y a peu de plateformes de la sorte pour des végans.

Cette réalisation me permet d'approfondir mes connaissances en web et de les développer dans le cadre d'un projet concret. Ce projet constitue un portfolio de mes acquis.

2. Cahier des charges

2.1 Organisation

Elève :

Roxanne Grant
roxanne.grnt@eduge.ch

Maître d'apprentissage :

Jasmina Travnjak
edu-travnjakj@eduge.ch

Experts :

Serge Murisier
SMURISIER@cross-systems.com

Borys Folomietow
borys@folomietow.ch

2.1 Objectifs

L'objectif est de créer un site web qui permet à des utilisateurs inscrits de gérer des recettes véganes. Les autres utilisateurs pourront ensuite poster des commentaires sur ces recettes et les sauver dans leurs favoris. L'admin pourra gérer toutes les recettes, et les supprimer. Il devra accepter les recettes avant qu'elles soient publiées.

¹ végan : une personne excluant tout produit d'origine animale de son alimentation et de sa vie quotidienne

2.2 Description détaillée du projet

Seul un utilisateur connecté peut créer une recette. Le site doit donc permettre à un utilisateur de se créer un compte, de se connecter et de se déconnecter. Une fois le compte créé, l'utilisateur pourra se connecter et administrer ses recettes.

Il pourra consulter toutes les recettes créées, et en créer de nouvelles. Il pourra également sauver ses recettes préférées dans ses favoris et commenter les recettes des autres.

Pour créer une recette, l'utilisateur authentifié, pourra choisir dans un premier temps un titre de recette, une image du résultat et un type de repas (ex: entrée). Ensuite il entrera les différents ingrédients et les instructions pour réaliser la recette. Une fois qu'il est prêt, il pourra publier sa recette. Un administrateur devra ensuite accepter ou non que la recette soit publiée à tous les utilisateurs du site et ajoutée à la liste des recettes.

Depuis la liste des recettes, les utilisateurs pourront consulter les recettes déjà postées sur le site et les filtrer/trier par type de repas etc...

Les utilisateurs du site pourront rechercher des recettes grâce à une fonctionnalité de recherche. Celle-ci leur proposera des résultats en connexion avec titre de leur recherche. Les utilisateurs non connectés pourront également voir les recettes disponibles sur le site mais ne pourront pas les commenter.

L'administrateur du site à tous les droits. Il peut créer des recettes mais il peut aussi supprimer toutes les recettes sur le site. Il a aussi le droit de supprimer des commentaires s'il ne les juge pas adéquats. Il peut aussi gérer les recettes et accepter qu'elles soient publiées ou non.

3. Analyse fonctionnelle

Il y a beaucoup de sites qui permettent d'avoir une communauté d'utilisateurs qui ajoute des recettes et les laissent visionnées par tout le monde, mais il y en a peu qui le font avec seulement des recettes véganes. Un des exemples est le site <http://www.marmiton.org/> qui propose plus de 63'000 recettes. Marmiton laisse ses utilisateurs ajouter des commentaires, proposer des recettes ou encore sauver des recettes préférées. Malheureusement, Marmiton ne suffit pas aux nombreux végans de la planète.

Beaucoup de sites qui encouragent le véganisme comme <https://www.vegansociety.com/> ou encore <https://www.peta.org/> offrent aussi une rubrique recette où ils affichent régulièrement des nouvelles recettes et idées culinaires. Mais ces différents sites ne sont pas supportés par une communauté d'internautes qui ajoutent constamment des recettes. Ces sites sont plutôt informatifs et les recettes ne sont qu'une petite rubrique dans tout le site.

C'est donc un marché manquant sur la toile surtout quand on sait que le véganisme est en pleine expansion et qu'il y a tous les jours des nouveaux adhérents à la cause.

J'ai néanmoins trouvé un site français qui propose de laisser les utilisateurs soumettre des recettes. Ce site s'appelle <http://vegémiam.fr/>.

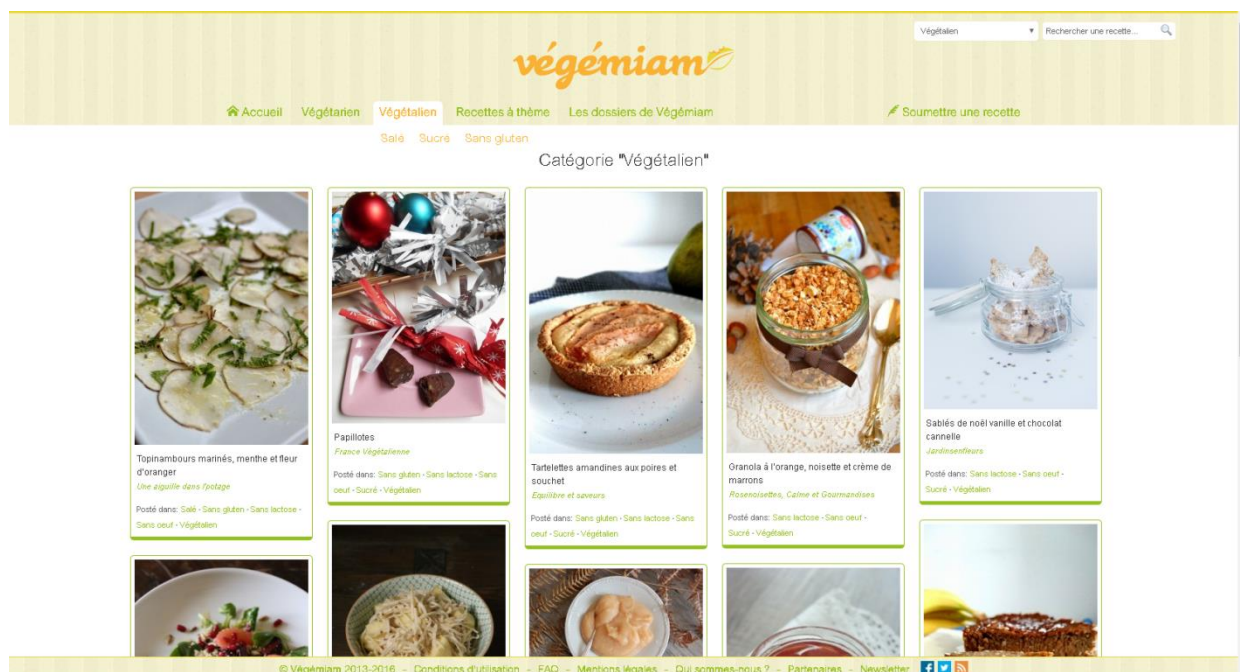


Figure 1 - Site végémiam

Le défaut de ce site c'est qu'il n'y a pas de filtres à part Végétalien et Végétarien. Le site n'est même pas mis à jour car comme le dit la page soumettre une recette, les administrateurs

prennent une pause de végémiam. Il n'y a pas d'utilisateurs sur végémiam et donc la partie gestion de favoris et gestion de ces propres recettes n'existent pas.

En conclusion, nous voyons qu'il n'y a donc aucun site sur internet aujourd'hui qui propose ce que je veux faire. Aucun site qui propose aux différents inspireurs de la cause végane de se retrouver autour de la même table sur le même pied d'égalité et de partager ensemble leur amour de la bonne nourriture tout en protégeant notre écosystème.

3.1 Technologies

Pour ce projet j'ai décidé de faire un site web. Le projet que j'ai choisi a beaucoup plus de sens en tant que site web car il peut être accédé par n'importe qui, n'importe où pour autant qu'ils aient un accès à internet.

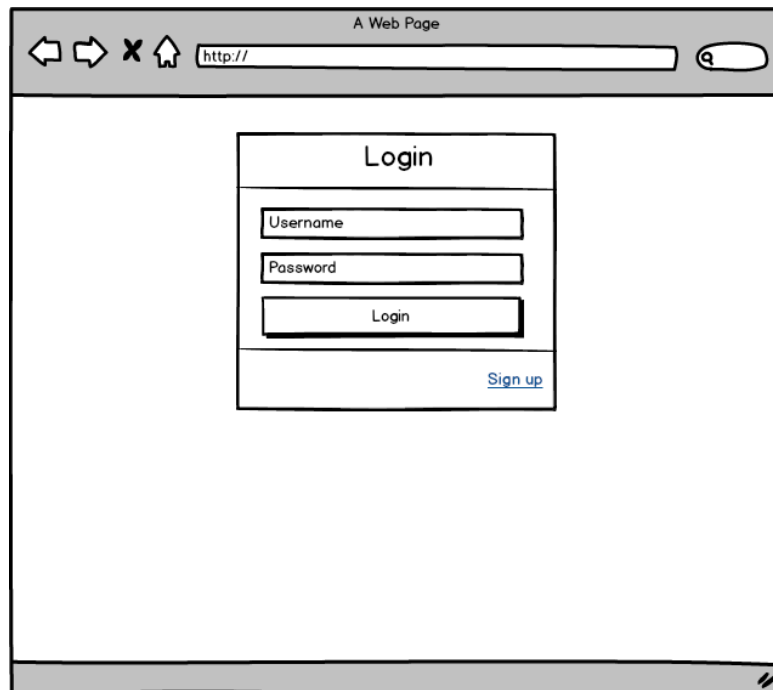
J'ai décidé de coder mon site en majorité en PHP car c'est le langage que nous avons vraiment approfondi au CFPT et que nous connaissons très bien. C'est un langage qui permet de créer des sites dynamiques, très utile pour avoir plusieurs utilisateurs qui doivent constamment envoyer des informations au serveur.

J'ai aussi utilisé du JavaScript et du JQuery pour des requêtes Ajax (Asynchronous Javascript and Xml). Ajax permet de faire des requêtes au serveur sans rafraîchir la page.

Pour le côté design du site, j'utilise un peu de css et surtout du Bootstrap qui me permet de facilement et rapidement implémenter des composants JavaScript et css dans mon site web. Bootstrap est un Framework css très utilisé par les développeurs pour éviter de passer trop de temps sur le design de leur site.

J'ai finalement utilisé phpmyAdmin pour la base de données. Il permet de créer et de gérer ses bases et il est directement implémenté dans Easyphp que j'utilise comme serveur local.

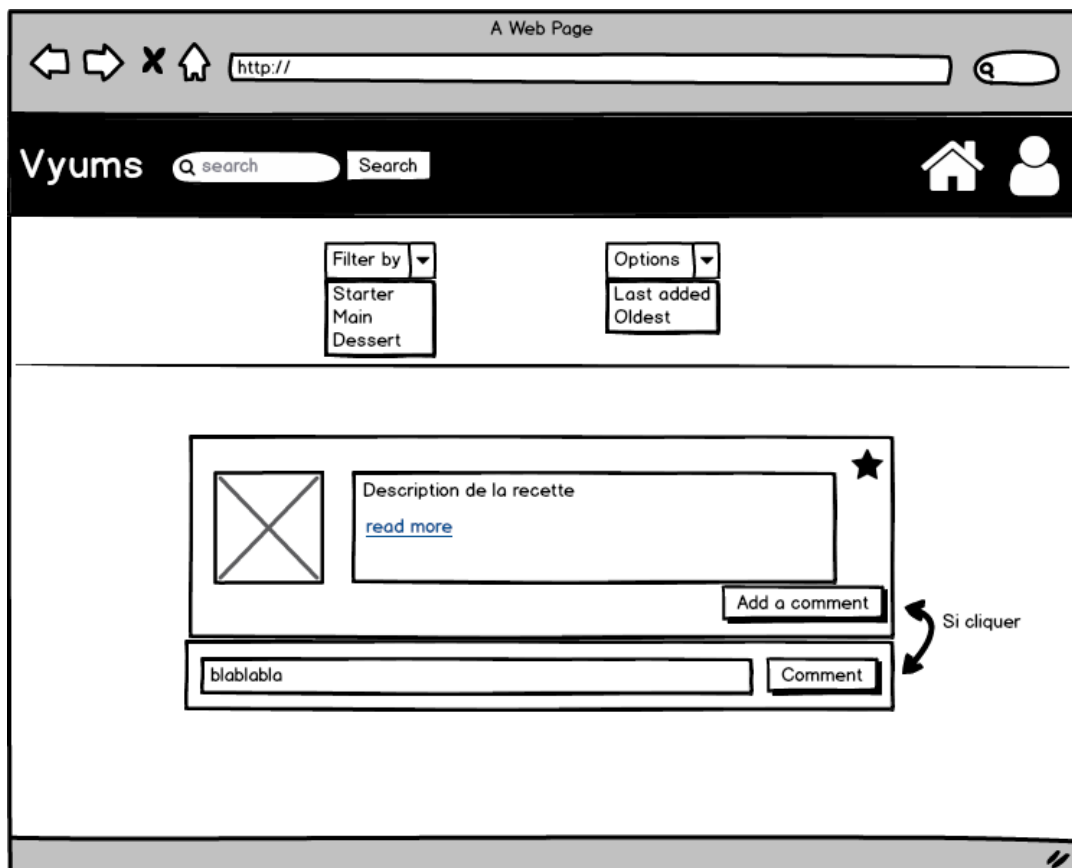
3.2 Maquette



A wireframe of a login modal. It features a title 'Login' at the top. Below the title are three input fields: 'Username', 'Password', and a 'Login' button. At the bottom right of the modal is a 'Sign up' link. The modal is centered on a page with a browser-like header and footer.

Created with Balsamiq - www.balsamiq.com

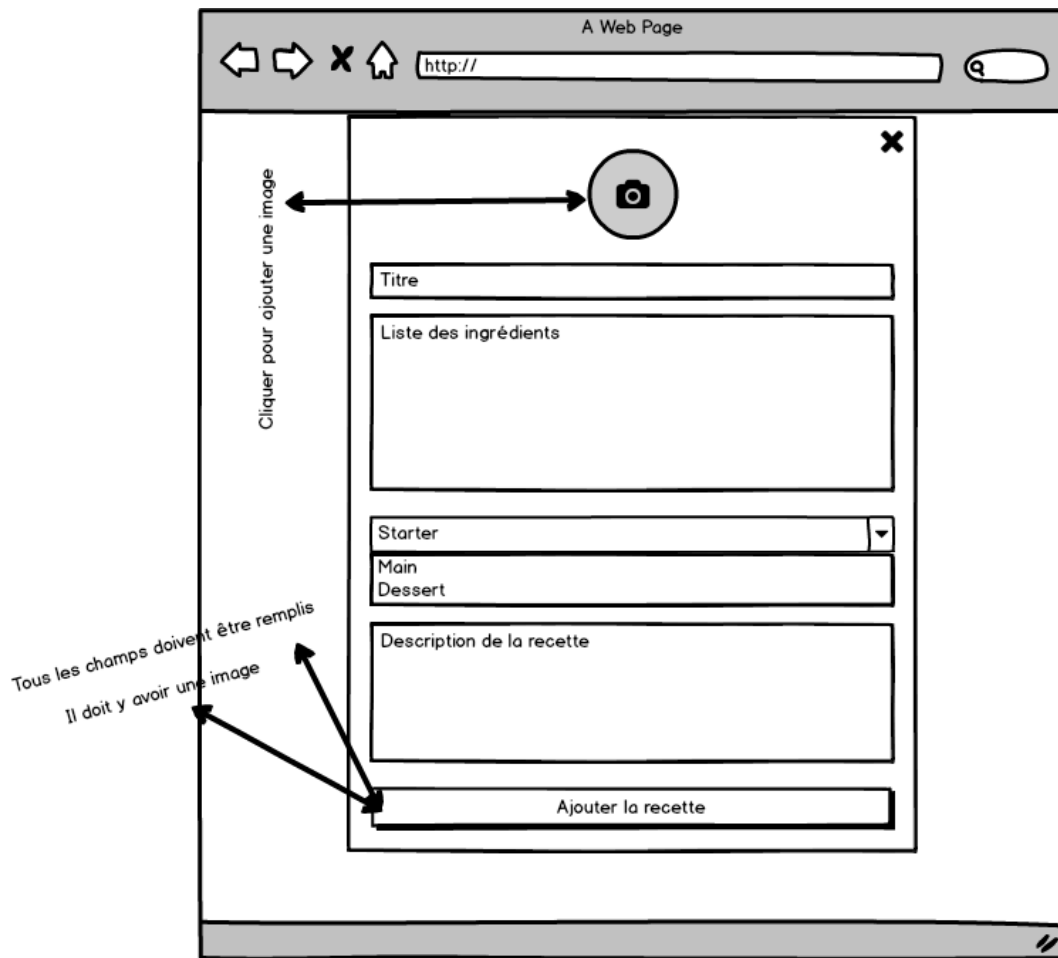
Figure 2 - Modal de login au site



A wireframe of the index page. The header includes the site name 'Vyums', a search bar, and navigation icons. Below the header are two dropdown menus: 'Filter by' (with options Starter, Main, Dessert) and 'Options' (with options Last added, Oldest). The main content area features a recipe card with a placeholder image, a description 'Description de la recette', a 'read more' link, and a star icon. Below the recipe card is a comment section with a text input field containing 'blabla' and a 'Comment' button. An arrow points to the 'Add a comment' button with the text 'Si cliquer'.

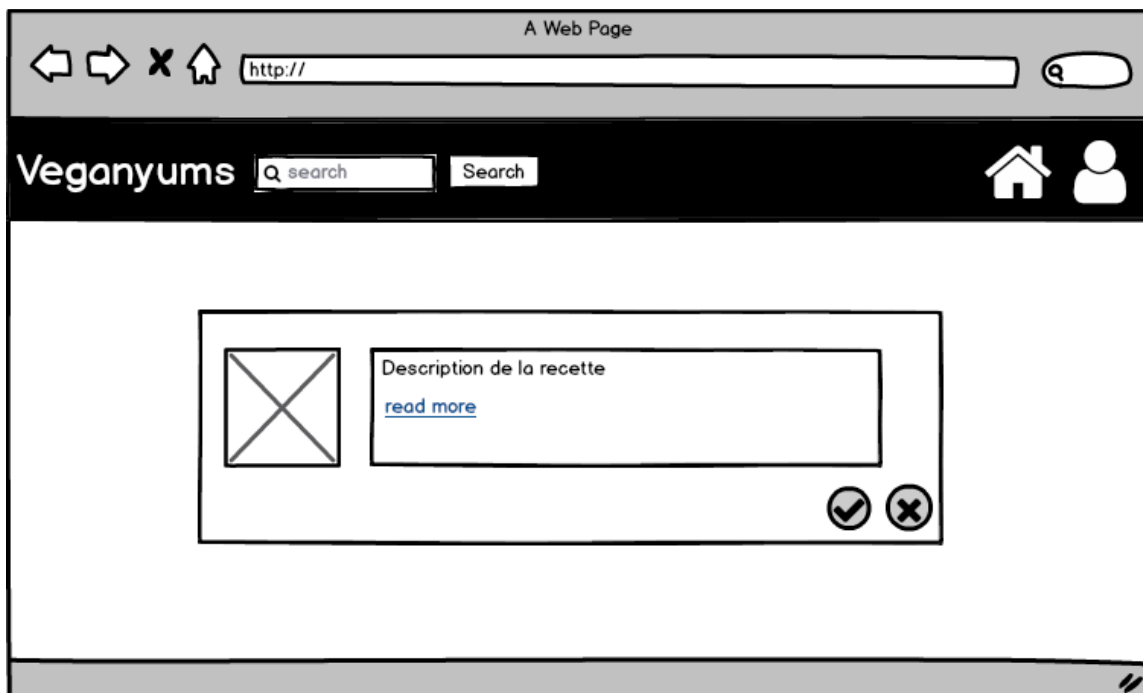
Created with Balsamiq - www.balsamiq.com

Figure 3 - Page index.php



Created with Balsamiq - www.balsamiq.com

Figure 4 - Modal d'ajout de recette



Created with Balsamiq - www.balsamiq.com

Figure 5 - Validation par l'administrateur

La fenêtre modale de « Sign up » pour s'inscrire au site est similaire à la fenêtre modale de login à part le texte du titre et le texte du bouton qui deviennent « Sign up ». Les autres pages sont toutes très similaires à la page index. Il y a juste les icônes du haut qui changent par rapport aux statuts de l'utilisateur.



L'utilisateur non connecté :

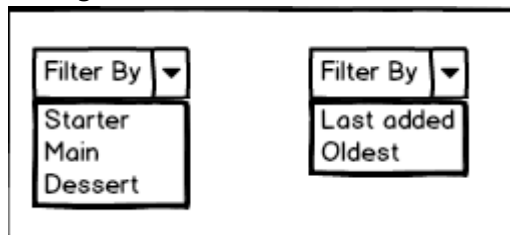


L'utilisateur connecté :

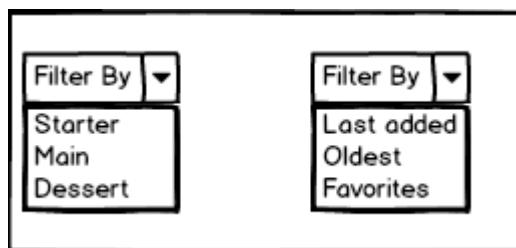


Admin connecté :

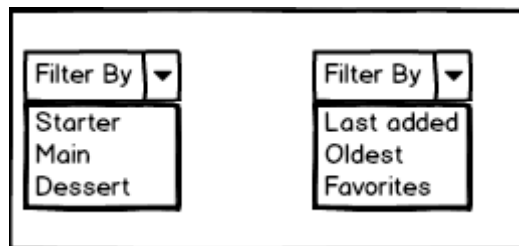
Les filtres sur la page home change aussi en fonction du statut de l'utilisateur.



L'utilisateur non connecté :



L'utilisateur connecté :



Admin connecté :

3.3 Fonctionnalités du site

Dans la barre de navigation :

- En cliquant sur le logo on revient sur la page d'accueil
- On peut écrire quelque chose dans la barre de recherche et avoir des suggestions qui apparaissent.
- Si on clique sur « search » après avoir écrit dans la barre de recherche, on obtient les résultats qui correspondent à notre recherche.
- L'utilisateur connecté peut ajouter une recette en cliquant sur +
- L'utilisateur peut en cliquant sur la petite icône de liste voir toutes les recettes qu'il a créé et les supprimer.
- L'utilisateur connecté peut se déconnecter
- L'admin connecté peut valider les recettes des autres utilisateurs en cliquant sur l'écrou

Dans la page principale :

- La page principale nous affiche toutes les recettes du site.
- Si l'utilisateur est connecté, il peut cliquer sur l'étoile à côté d'une recette pour la mettre dans ses favoris.
- En cliquant sur add a comment, l'utilisateur connecté peut saisir un commentaire à ajouter puis cliquer sur Comment pour l'envoyer.
- L'utilisateur peut filtrer les recherches de recettes par type.
 - Starter-> Les recettes affichées sont des entrées
 - Main -> Les recettes affichées sont des plats principaux
 - Dessert -> Les recettes affichées sont des desserts
- L'utilisateur peut à des options de filtrage en plus, sous le deuxième « Filter by »
 - Last added-> les recettes sont affichées dans l'ordre décroissant.
 - Oldest post -> Les recettes sont affichées dans l'ordre croissant.
- L'utilisateur a plus d'options s'il est connecté
 - Favorites -> Toutes les recettes qu'il a ajouté à ses favoris.
- En cliquant sur la petite icône du user, la modal de connexion s'ouvre.
- L'admin connecté peut supprimer des recettes ou des commentaires

Dans la modal de connexion :

- L'utilisateur peut se connecter
- L'utilisateur peut atteindre la modal d'inscription

Dans la modal d'inscription :

- L'utilisateur peut s'inscrire
- L'utilisateur peut atteindre la modal de connexion

3.4 Définition des pages et des fenêtres modales

3.4.1 La page *index.php*

La page d'index est la page d'accueil du site. Il n'y a pas besoin d'être connecté pour voir cette page. Ici, sont affichées toutes les recettes du site validées par l'administrateur. Elles peuvent être filtrées par date de publication ou par type de recettes.

Sur cette page, si on est un utilisateur, on peut aussi voir ces propres recettes en cliquant sur l'icône de liste. Si on est un admin, on peut voir les recettes à valider en cliquant sur l'icône de l'écrou.

3.4.2 La fenêtre modale de connexion

La fenêtre modale de connexion permet à l'utilisateur ou l'administrateur non connecté de se connecter. Une fois que la connexion est validée, il est automatiquement redirigé vers la page d'accueil. Il y a un lien vers la modal d'inscription sur la modal de connexion.

3.4.3 La fenêtre modale d'inscription

La fenêtre modale d'inscription permet à un utilisateur de créer un compte. Une fois que son inscription est validée, il est automatiquement redirigé vers la page d'accueil. Il y a un lien vers la modal de connexion sur la modal d'inscription

3.4.4 La fenêtre modale d'ajout de recette

La fenêtre modale d'ajout permet l'utilisateur connecté de créer une nouvelle recette. Il peut ajouter une image, le type de recette, la liste des ingrédients et la description de la recette.

4. Analyse organique

4.1 Plan du site

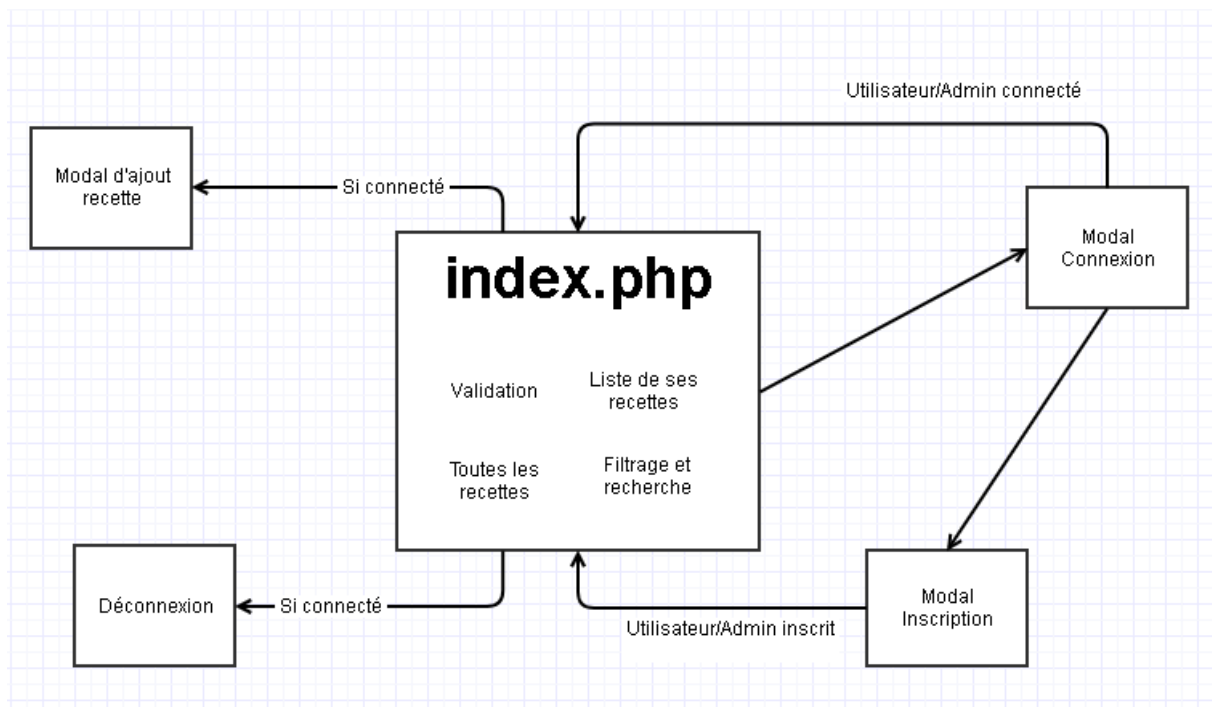


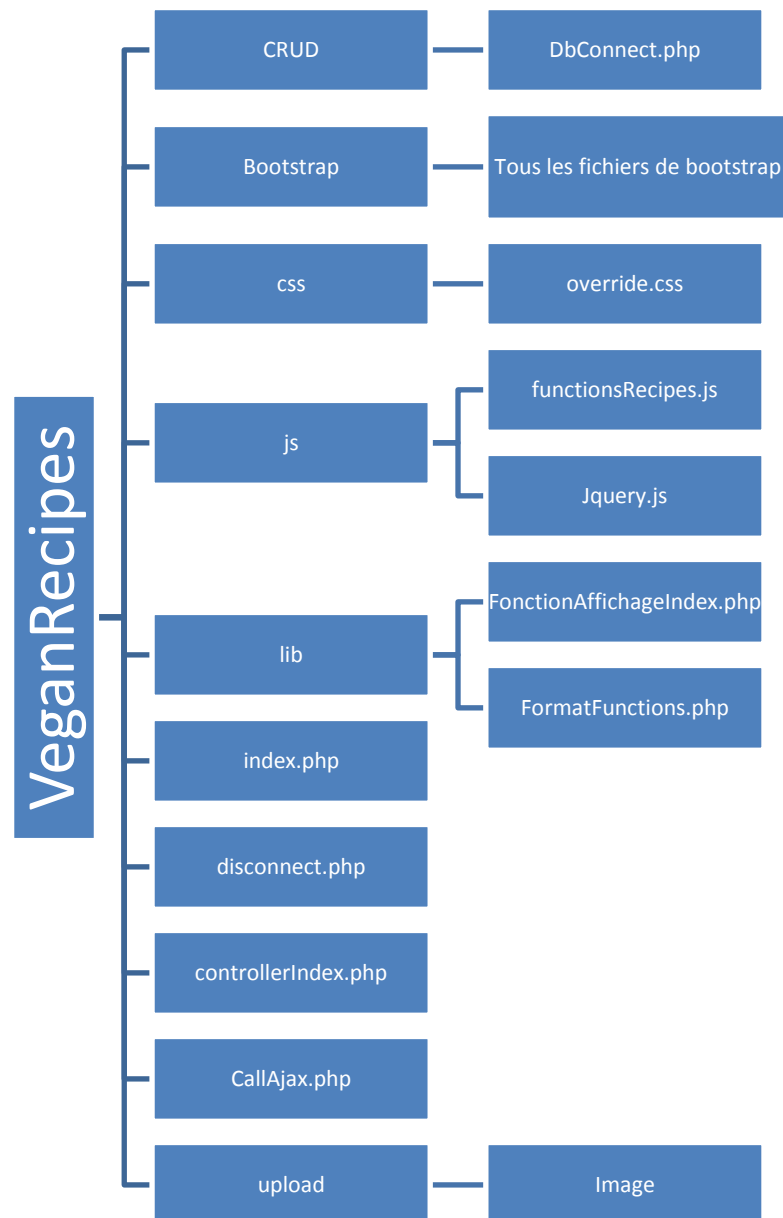
Figure 6 - Plan du site

Sur ce diagramme, on voit comment le site se compose. La majeure partie des actions se passe sur la page index. Depuis cette page, on peut filtrer la liste, rechercher quelque chose dans la barre de navigation et si on est connecté, on peut ajouter des recettes favorites ou des commentaires. Depuis index, l'admin peut aussi valider les recettes à ajouter au site.

Si un utilisateur n'est pas connecté, depuis index il peut atteindre la fenêtre modale de connexion et depuis celle-ci, celle d'inscription.

Une fois qu'il est inscrit ou connecté, il est redirigé vers la page index. Depuis là, il peut ajouter une recette avec la fenêtre modale d'ajout ou se déconnecter.

4.2 Arborescence de fichier



Ma structure est organisée autour de la page `index.php` qui est ma page principale et ma vue. La page index a un contrôleur qui s'appelle `controllerIndex.php`. Il y a deux fichiers de fonctions dans le dossier `lib`. La page PHP de fonction `formatFonctions` qui contient toutes les fonctions de gestion et la page `fonctionAffichageIndex` qui contient des fonctions pour l'affichage sur index.

Il y a un dossier `css` qui contient le fichier `css override.css` utilisé sur la page index. Le dossier `Bootstrap` contient tous les fichiers concernant Bootstrap. Les pages utilisées sur index sont `bootstrap.min.js` et `bootstrap.min.css`. Le dossier `js` contient les fichiers JavaScript. Le fichier `Jquery.js` contient JQuery, aussi nécessaire pour que le fichier `bootstrap.min.js` fonctionne, et le fichier `functionsRecipes` contient mes fonctions JavaScript.

La page `CallAjax.php` est un contrôleur pour les calls Ajax qui doivent afficher certaines informations à certains endroits sans mettre tout le data dans le body.

Finalement, le dossier CRUD contient la classe DBConnect qui contient toutes les fonctions de requêtes à la base de données. La base de données est seulement appelée par le contrôleur.

4.3 Requêtes principales

```
SELECT * FROM utilisateurs WHERE Username= :user AND Password= :pwd
```

Cette requête permet de chercher un utilisateur avec un certain nom et mot de passe pour voir s'il existe dans la base de données

```
INSERT INTO recettes  
(`IdRecette`,`Titre`,`Ingredient`,`Description`,`Valider`,`NomFichierImg`,`IdUtilisateur`,`IdType`)  
SELECT ",:title,ingredients,:descrip,0,:img,:id,IdType FROM types WHERE NomType= :type"
```

Cette requête permet d'insérer une recette d'un utilisateur dans la base de données. Il faut insérer l'id du type c'est pour ça que je vais chercher l'id par apport au nom que j'ai déjà.

```
SELECT * FROM `recettes` WHERE Valider= :Valid
```

Cette requête permet de chercher les recettes par apport à leurs statuts de validation

```
SELECT IdRecette FROM favoris WHERE IdUtilisateur=:uid
```

Cette requête permet d'obtenir les recettes mis en favoris par un utilisateur en fonction de son id.

```
INSERT INTO commentaires VALUES (",:comment,:uid,:idR)
```

Cette requête permet d'insérer un nouveau commentaire sur une recette spécifique

```
SELECT * FROM recettes WHERE Valider=1 AND Titre like :keyword ORDER BY titre LIMIT  
0,6
```

Cette requête permet de trouver les recettes dont le titre ressemble à ce que l'utilisateur a recherché dans la base. Le :keyword est enfaite la recherche de l'utilisateur suivi du joker « % ».

```
SELECT * FROM `favoris` JOIN recettes USING (IdRecette) WHERE favoris.IdUtilisateur = :uid
```

Cette requête permet de chercher les toutes les recettes misent en favoris par un certain utilisateur.

4.4 Fonctions php de gestion

4.4.1 *FormatIngredients(\$ingredients)*

Les ingrédients sauver dans la base de données sont au format « - ingrédient 1 – ingrédient 2 ». Cette fonction sert donc à formater le string des ingrédients pour le mettre dans le bon format pour l'insertion dans la base de données.

4.4.2 *VerificationAdd(\$param)*

Cette fonction permet de nettoyer tous les champs remplis par l'utilisateur pour qu'il ne fasse pas d'injection de SQL ni de JavaScript. La fonction retourne un nouveau tableau avec tous les champs nettoyés.

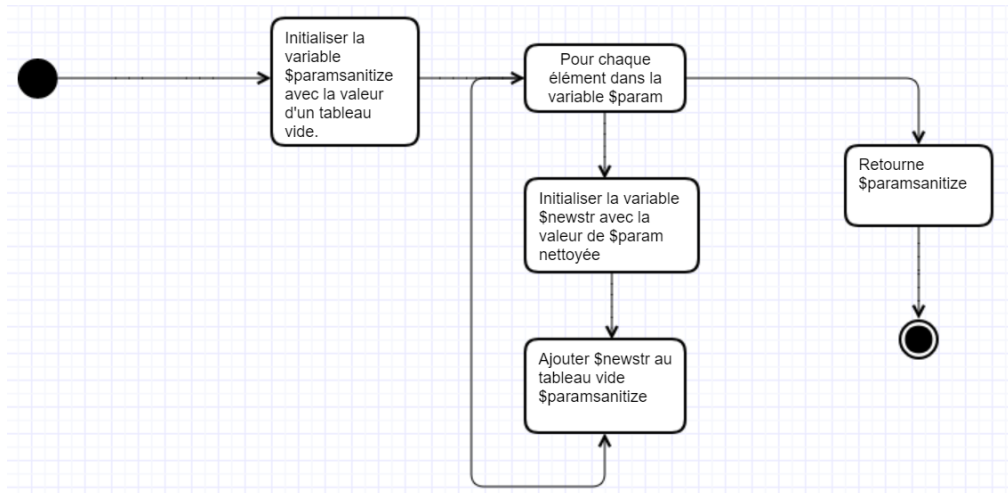


Figure 7 - *VerificationAdd(\$param)*

4.4.3 *VerifyImg(\$files)*

Cette fonction sert à vérifier qu'une image a été sélectionnée et qu'elle a la bonne extension.

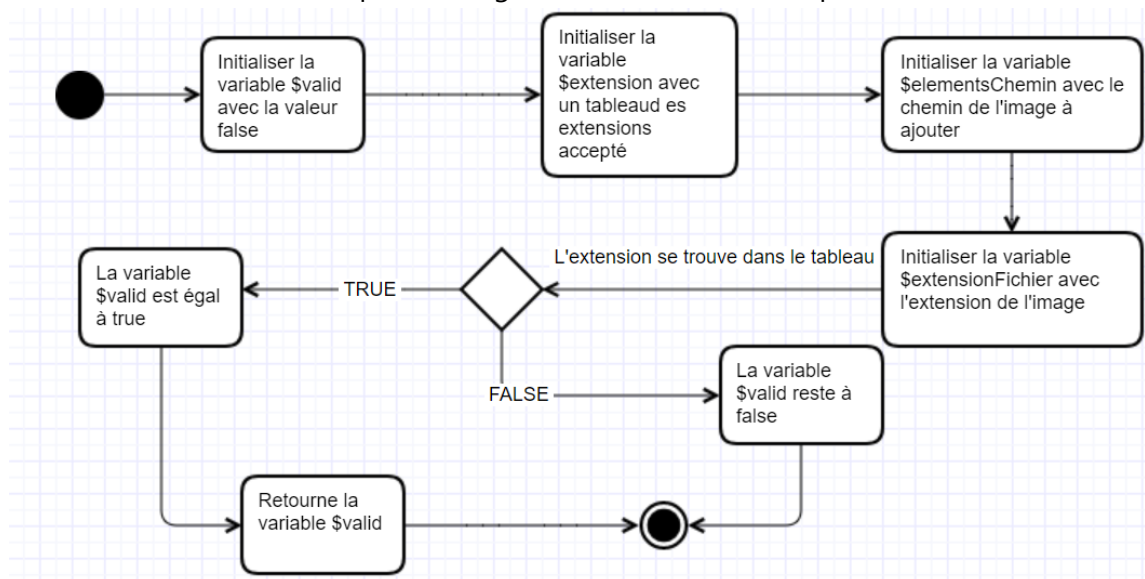


Figure 8 - *VerifyImg(\$files)*

4.4.4 *MoveImg(\$files)*

Regarde si \$_FILES est vide, appelle la fonction *VerifyImg(\$files)*. Si l'image est dans les normes, fonction essaye de la bouger dans le répertoire d'upload. Si la condition retourne vrai, on insère l'image avec la recette sinon on insère la recette sans image.

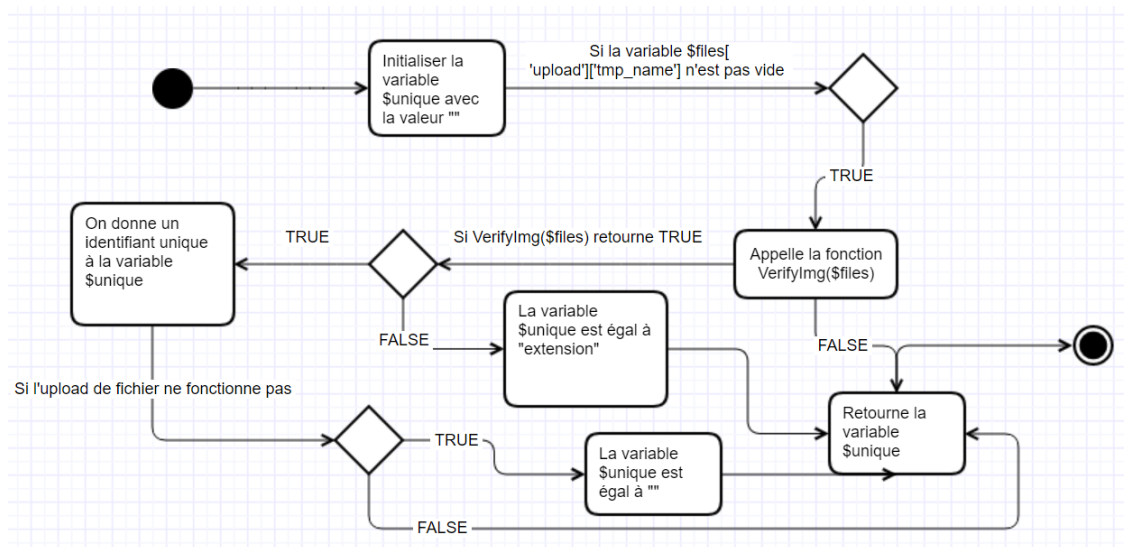


Figure 9 - MoveImg(\$files)

4.4.5 IsEmpty(\$param)

Cette fonction permet de vérifier si un des champs rempli par l'utilisateur est vide. Elle retourne le nombre de champs vides.

4.4.6 SignedIn(\$isAdmin)

Cette fonction est une fonction d'affichage. Elle affiche la bonne navigation en fonction du statut de l'utilisateur connecté (utilisateur simple ou admin).

4.4.7 KeepModalOpen(\$alert,\$modalname)

Cette fonction permet de garder la modal de connexion ou d'ajout ouverte s'il y a une erreur.

4.4.8 ListIngredients(\$listIngredients)

Cette fonction permet de formater le string des ingrédients de la base de données en liste à puce.

4.4.9 RestrictLengthDescrip(\$descrip)

Cette fonction permet de restreindre le nombre de caractères affiché dans la description et que malgré cela on coupe à un mot complet.

4.4.10 DeleteImg(\$dossier,\$value)

Cette fonction permet de supprimer l'image du répertoire des uploads si l'administrateur décide de supprimer une recette.

4.4.11 IsFav(\$value, \$favorite)

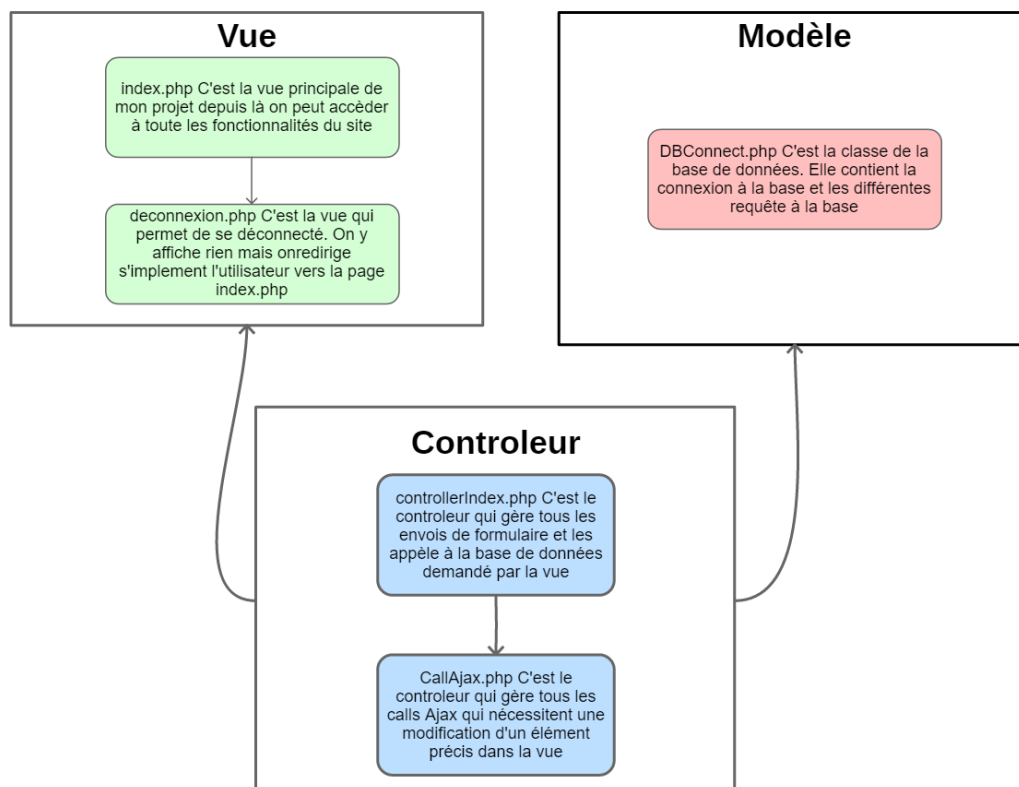
Parcours tous les favoris de l'utilisateur pour voir s'il y en a un sur la recette en question se trouvant dans \$value.

4.4.12 ShowComment(\$value, \$comment)

Parcours tous les commentaires de la base de données pour voir s'il y en a un ou plusieurs sur la recette en question se trouvant dans \$value.

4.5 Architecture du projet

Le projet est réalisé en MVC (Model – Vue - Controller). C'est une architecture qui comme le nom l'indique permet de séparer le projet en plusieurs parties. De cette manière le code est plus clair, compréhensible et peut être facilement repris par un autre développeur. Pour mieux comprendre la structure, voici une petite maquette :



Ce schéma démontre comment ma structure est organisée. Toute l'intelligence de ma page index est gérée dans le constructeur qui s'occupe d'interroger le modèle, qui dans ce cas-là, est ma base de données. Certaines fonctions JavaScript appellent le contrôleur `CallAjax.php` qui permet de recharger que certains éléments de la page.

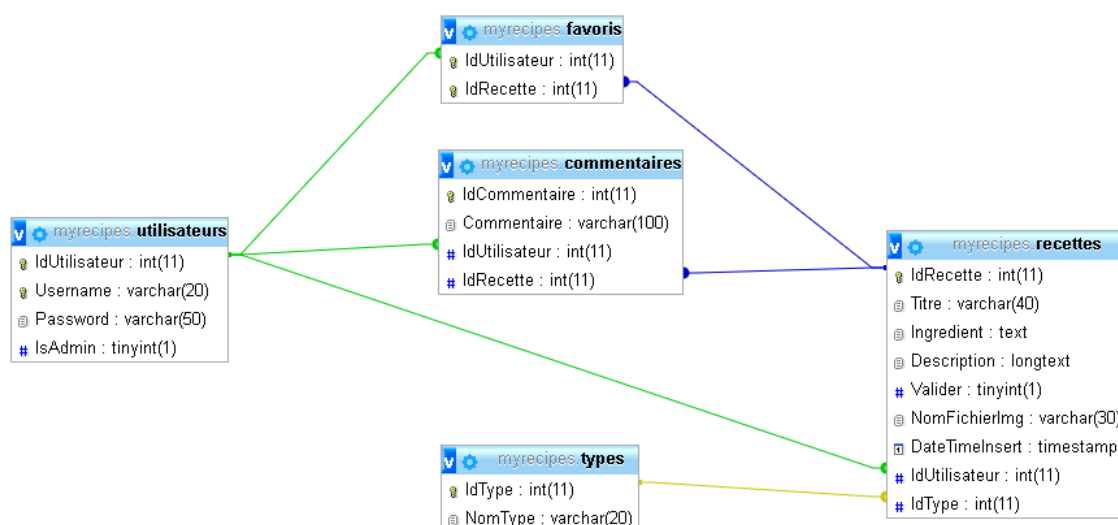
4.6 Diagramme de classe

Pour ce projet, j'ai une classe PHP qui contient ma connexion et mes requêtes à la base de données.

DbConnect
-ps_getRegistration: PDO statment -ps_register: PDO statment -ps_insertRecip : PDO statment -ps_validateRecipe: PDO statment -ps_removeRecipe: PDO statment -ps_addFav: PDO statment -ps_removeFav: PDO statment -ps_insertComment: PDO statment -ps_autocomplete: PDO statment -dbb: PDO statment
GetRegistration(\$user, \$pwd) : array Register(\$user, \$pwd) : void() InsertRecipe(\$param, \$id, \$img) : void() ValidateRecipe(\$idR) : void() RemoveRecipe(\$idR) : void() AddFav(\$uid, \$idR) : array removeFav(\$uid, \$idR) : void() insertComment(\$comment, \$uid, \$idR) : void() Autocomplete(\$keyword) : array

Je n'ai pas mis tous les éléments dans ce diagramme de classe, car la plupart se ressemblent. Vous trouverez le diagramme de classe complet en annexe.

4.7 MLD



Ce qui change entre le Modèle Conceptuel créé lors de l'élaboration du cahier des charges et le Modèle Logique, c'est la table type. Dans le Modèle Conceptuel, mon maître

d'apprentissage n'avait pas mis cette table. Je l'ai rajouté pour éviter des redondances au niveau des recettes. La relation entre la table utilisateurs et recettes étaient de type 0, N c'est pour ça que dans le MLD j'ai une table favorite qui a comme clé primaire les identifiants des deux tables.

Pour finir, j'ai ajouté un champ DateTimeInsert pour garder l'heure à laquelle la recette a été ajoutée par l'utilisateur. De cette manière, je peux filtrer les recettes par date d'ajout.

4.8 Dictionnaire de données

commentaires

Colonne	Type	Null	Défaut	Relié à
IdCommentaire	int(11)	Non		
Commentaire	varchar(100)	Non		
IdUtilisateur	int(11)	Non		utilisateurs -> IdUtilisateur
IdRecette	int(11)	Non		recettes -> IdRecette

favoris

Colonne	Type	Null	Défaut	Relié à
IdUtilisateur	int(11)	Non		utilisateurs -> IdUtilisateur
IdRecette	int(11)	Non		recettes -> IdRecette

recettes

Colonne	Type	Null	Défaut	Relié à
IdRecette	int(11)	Non		
Titre	varchar(40)	Non		
Ingredient	text	Non		
Description	longtext	Non		
Valider	tinyint(1)	Non		
NomFichierImg	varchar(30)	Non		
DateTimeInsert	timestamp	Non	CURRENT_TIMESTAMP	
IdUtilisateur	int(11)	Non		utilisateurs -> IdUtilisateur
IdType	int(11)	Non		types -> IdType

types

Colonne	Type	Null	Défaut
IdType	int(11)	Non	
NomType	varchar(20)	Non	

utilisateurs

Colonne	Type	Null	Défaut
IdUtilisateur	int(11)	Non	
Username	varchar(20)	Non	
Password	varchar(50)	Non	
IsAdmin	tinyint(1)	Non	

4.9 Gestion des favoris

Pour gérer les favoris, l'utilisateur connecté peut cliquer sur une petite étoile. Cette étoile se remplit quand il met une recette en favori. De la même manière, si l'utilisateur veut enlever le favori, il n'a qu'à recliquer sur la petite étoile qui est remplie et elle se videra.

Quand on passe la souris sur une étoile vide, elle se remplit et se vide quand on ressort notre souris pour montrer à l'utilisateur qu'elle est cliquable. Tout ceci est géré par 2 calls Ajax qui permettent de favoriser et défavoriser une recette.

```
function Favorite(tag) {
    var id = $(tag).attr("id");
    $.ajax({
        type: "POST",
        url: "index.php",
        data: 'favorite=' + id,
        success: function () {
            $(tag).removeClass();
            $(tag).addClass("glyphicon glyphicon-star");
            $(tag).on('click', function() { UnFavorite(tag); });
            $(tag).attr('onmouseover', '');
            $(tag).attr('onmouseout', '');
        },
        error: function (error) {
            $('#msg').append("<div class='alert alert-danger' role='alert'>" + error + "</div>").fadeIn('slow');
            $('#msg').delay(1000).fadeOut('slow');
        }
    });
}
```

Figure 10 - Call ajax pour ajouter un favori

Ici, on ajoute un favori en cliquant sur l'étoile. Le call Ajax va envoyer l'id de la recette pour savoir sur quoi on a cliqué. En succès du call, on lui ajoute la classe Bootstrap qui nous permet d'avoir l'icône de l'étoile remplie. On ajoute aussi un événement pour défavoriser sur l'icône. De cette manière, quand on favorise une recette en recliquant on peut la défavoriser et ainsi de suite. Le call pour défavoriser ajoute un événement pour favoriser.

Pour finir si le call ne marche pas, on affiche le message d'erreur. Du côté du PHP, avec l'id reçu du call Ajax, le PHP appelle la DB pour insérer un favori dedans.

4.10 Gestion de la validation par l'administrateur

En étant connecté en tant qu'administrateur sur la page index, l'admin peut valider les recettes ajoutées par les nombreux utilisateurs pour les afficher aux yeux de tous. En cliquant sur le petit écrou, la page index lui montre les recettes à valider. Il peut alors décider de les valider ou de les supprimer définitivement du site. Ici aussi pour recevoir, valider et supprimer les recettes, on utilise des calls Ajax pour que l'utilisation soit homogène.

```
function GetRecipesToValidate() {  
    $.ajax({  
        type: "POST",  
        url: "index.php",  
        data: 'tovalidate=true',  
        success: function (data) {  
            $("#body").html("");  
            $("#body").html(data);  
        },  
        error: function (error) {  
            $('#msg').append("<div class='alert alert-danger' role='alert'>" + error + "</div>").fadeIn('slow');  
            $('#msg').delay(1000).fadeOut('slow');  
        }  
    });  
}
```

Figure 11 - Afficher recette pour la validation

Cette fonction est appelée quand l'administrateur clique sur l'écrou. Elle permet de récupérer les recettes à valider. Ici le call Ajax envoie la variable « tovalidate » au PHP qui appelle la classe de la DB pour afficher les recettes. Si ce call est fait avec succès, le body affiche les informations reçues, sinon on affiche un message d'erreur.

Pour valider la recette ou la supprimer on utilise exactement le même principe. On a juste besoin de récupérer l'id de la recette. On fait cela en recherchant l'id du div le plus proche qui contient l'identifiant de la recette.

```
var id = $(tick).closest(".YesNo").attr("id");
```

Figure 12 - Trouver l'id de la recette

On envoie ensuite l'id dans le data pour que du côté PHP, on puisse faire la requête à la base de données et changer la recette à une recette valide ou la supprimer.

Comme on utilise la même fonction pour supprimer les recettes non validées et supprimer les recettes validées, on ajoute un énumérateur 0 ou 1 dans le data envoyé par le call Ajax. Si c'est 0 la recette n'avait pas encore été validée, cela veut dire qu'on est sur la vue validation pour l'administrateur. Si c'est 1 la recette avait déjà été validée, on doit donc afficher la vue de home.

4.11 Gestion des commentaires

La gestion des commentaires a un fonctionnement très similaire au fonctionnement des favoris. L'utilisateur peut ajouter des commentaires en cliquant sur « Add a comment », un input s'ouvre où il peut insérer son commentaire. Quand il clique sur « send comment » cela déclenche un call Ajax.

```
function AddComment(button) {  
    //gets id of recipe according to clicked button  
    var idR = $(button).closest(".collapse").attr("id");  
    //Get value of comment  
    var comment = $(button).closest(".collapse").find("input[name=comment]").val();  
    $.ajax({  
        type: "POST",  
        url: "CallAjax.php",  
        data: {commenttext: comment, idRecipe: idR},  
        success: function (data) {  
            $(button).closest(".collapse").parent().find(".Allcomments").html("");  
            $(button).closest(".collapse").parent().find(".Allcomments").html(data);  
            $(button).closest(".collapse").find("input[name=comment]").val("");  
        },  
        error: function (error) {  
            $('#msg').append("<div class='alert alert-danger' role='alert'>" + error + "</div>").fadeIn('slow');  
            $('#msg').delay(1000).fadeOut('slow');  
        }  
    });  
}
```

Dans le data, on envoie un tableau avec le texte du commentaire et l'id de la recette. Pour obtenir ces informations, on recherche l'input qui est l'enfant du div avec la classe « collapse ».

Du côté du PHP, on reçoit les informations et on nettoie le commentaire pour éviter les injections. On finit par insérer le commentaire et obtenir tous les commentaires. Pour que l'administrateur puisse supprimer un commentaire, on envoie que l'id du commentaire dans le data du call Ajax.

4.12 Ajout de recette par l'utilisateur

Pour ajouter une recette, l'utilisateur doit cliquer sur le +. Une modal s'ouvre qui lui permet d'insérer les informations. Il est obligé d'insérer un titre, une liste d'ingrédient, et une description. L'image est optionnelle. Le rond pour ajouter une image est un hit box qui appelle une fonction JavaScript. Cette fonction ajoute un input de type Files à la forme et déclenche un clic sur celui-ci. De cette manière l'utilisateur peut ensuite sélectionner son image.

Du côté PHP, on vérifie que les champs obligatoires soient bien remplis. J'utilise le filter_sanitize pour éviter les injections JavaScript et SQL. Ensuite, on regarde si l'utilisateur a ajouté une image, si oui, on vérifie que l'image a la bonne extension. Pour finir, on essaye d'uploader l'image dans notre dossier, si ça fonctionne, on insère les infos avec l'image dans la base de données. Pour le faire, on utilise les fonctions VerificationAdd(\$param), VerifyImg(\$files), MoveImg(\$files), IsEmpty(\$param).

4.13 Auto complétion et recherche

Tous les utilisateurs, peu importe leur statut, peuvent rechercher dans la barre de recherche. Si un utilisateur tape quelque chose dans la barre de recherche et que la base de données contient une recette qui commence par les lettres que l'utilisateur a saisies, une liste

déroulante lui affichera cette proposition. Si l'utilisateur clique dessus, la proposition va s'écrire dans sa boîte de recherche. Le submit fait un call Ajax pour retourner les résultats de la base de données qui commencent par ce que l'utilisateur a tapé dans la barre de recherche.

```
function CallTitles() {
    $("#search-box").keyup(function () {
        $.ajax({
            type: "POST",
            url: "CallAjax.php",
            data: 'keyword=' + $(this).val(),
            async: true,
            beforeSend: function () {
                $("#search-box").addClass(".loading");
            },
            success: function (data) {
                $("#suggestion-box").show();
                $("#suggestion-box").html(data);
            }
        });
    });
}
```

Figure 13 - Call AJAX pour l'auto complétion

On appelle cette fonction chaque fois qu'une touche est relevée. Dans le data, on envoie la valeur de l'input que l'utilisateur a saisi dans la barre de recherche. Si la fonction est réussie, on affiche le data dans la boîte de suggestion qui sont les propositions de la base de données.

On remarque que ce call appelle la page CallAjax.php et non index.php. C'est parce que si elle avait appelé index elle aurait affiché tout le contenu de la page index dans la liste de suggestion. En envoyant le call sur une autre page, on évite cette erreur et on affiche que les suggestions car lors d'un call ajax on a pas besoin d'afficher toute la vue en entier

Si l'utilisateur clique sur une suggestion, cela appelle une fonction JavaScript qui donne la valeur de l'élément cliqué à la barre de recherche. De cette manière, l'utilisateur peut utiliser les suggestions pour se faciliter la vie pour rechercher des recettes.

```
function selectTitle(val) {
    $("#search-box").val(val);
    $("#suggestion-box").hide();
}
```

Si l'utilisateur clique sur « submit », on envoie le contenu de la barre de recherche, à l'aide d'un call Ajax, à la base de données et on lui renvoie ce qui ressemble le plus à sa recherche.

4.14 Gestion des filtres

L'utilisateur a quelques options de filtrages :

- Par type de repas
 - Entrée
 - Plat
 - Dessert
- Par date
 - Recettes plus récentes
 - Recettes plus vieilles

L'utilisateur ou l'administrateur peut utiliser ces filtres à plusieurs endroits différents :

- Sur toutes les recettes
- Sur le résultat de sa recherche
- Sur ces recettes (seulement pour l'utilisateur normal)
- Sur les recettes à valider (seulement pour l'administrateur)

Ceci est réalisé avec un call Ajax et une requête SQL que nous allons détailler ici.

```
var FilterRecipes = $(e.target).closest("ul").attr("id");
var type = "";
var sort = "";
if (FilterRecipes === "MealTypes") {
    type = $(e.target).text();
    sort = $("input[name=filterSort]").val();
}
else {
    sort = $(e.target).text();
    type = $("input[name=filterType]").val();
}
//Gets criteria of current recipes
var searchKeyword = $("input[name=searchKeyword]").val();
var isMyRecipes = $("input[name=SearchByUserRecipes]").val();
var isValid = $("input[name=searchbyNotValidated]").val();
```

Figure 14 - Fonction de filtrage

Avant de faire le call Ajax, il faut définir quel filtre a été cliqué et si on a déjà choisi d'afficher certaines recettes. Pour faire ceci, j'attrape l'id de l'élément liste cliqué pour savoir sur quel filtre on se trouve. J'ai stocké dans des inputs de type caché des valeurs pour savoir si on a déjà utilisé un filtre ou si on affiche déjà les recettes d'un certain type.

Ensuite, dans le data de mon call ajax, j'envoie toutes ses variables pour les ajouter à ma requête SQL.

```
$( "input[name=filterType"] ).val( type );
$( "input[name=filterSort"] ).val( sort );
$( "input[name=searchKeyword"] ).val( searchKeyword );
$( "input[name=SearchByUserRecipes"] ).val( isMyRecipes );
$( "input[name=searchbyNotValidated"] ).val( isValid );
```

Figure 15 - Succès du call Ajax

Dans le succès du call Ajax, on met les nouvelles valeurs dans les inputs cachés pour pouvoir réutiliser les filtres.

```
SELECT * FROM `recettes` NATURAL JOIN types WHERE (NomType= :type OR :type = "") AND  
(Titre like :search OR :search = "") AND (IdUtilisateur = :uid OR :uid = "") AND (Valider= :valid  
OR :valid = "")
```

Du côté de SQL, on fait une requête avec une combinaison de AND et OR pour une recherche plus puissante. De cette manière, on peut ajouter des paramètres et s'ils sont vides, ils ne sont pas pris en compte. On arrive ainsi à faire un filtrage sur différents éléments avec une seule requête.

Pour ajouter le paramètre de la date d'insertion, je suis obligée d'ajouter un if dans la fonction d'insertion à la base de données. Dépendant ce qu'il y a dans la variable on ajoute un ORDER BY DESC ou un ORDER BY ASC. Il n'y pas d'autre manière de faire car on ne peut pas mettre un « bindparam » sur un ORDER BY.

5. Rapport de tests

	Numéro de test	Date	Résultat attendu	OK/KO
Utilisateur non connecté	T1	16/06/17	index.php, on voit toutes les recettes qui s'affichent	OK
	T2	16/06/17	En cliquant sur « Read more... » sur l'une des recettes, une fenêtre modale s'ouvre et donne la recette entière	OK
	T3	16/06/17	En cliquant sur une des options dans le premier « Filter By » on a que les recettes qui sont de ce type qui s'affichent	OK
	T4	16/06/17	En cliquant sur une des options dans le « Sort By » on a les recettes qui viennent dans l'ordre croissant qu'elles ont été postées ou l'ordre décroissant	OK
	T5	16/06/17	En cliquant sur un des filtres dans « Filter By » et un des filtres dans « Sort By » on a les recettes du type choisi triées par l'ordre choisi	OK
	T6	16/06/17	Si on tape f dans la barre de recherche, on obtient la suggestion « falafels »	OK
	T7	16/06/17	Après T6, Si on clique sur « falafels » il apparaît dans la barre de recherche	OK
	T8	16/06/17	Après T7, en cliquant sur submit, seule la recette des falafels apparaît	OK
	T10	16/06/17	En cliquant sur l'icône de la maison, on revient sur toutes les recettes.	OK
	T11	16/06/17	En cliquant sur l'icône du bonhomme on a une fenêtre modale qui s'ouvre pour se connecter	OK
Fenêtre Modale de Login	T12	16/06/17	Depuis la fenêtre modale Login, en cliquant sur le lien « Sign up » on a une fenêtre modale qui s'ouvre pour s'inscrire.	OK
	T13	16/06/17	Si on entre User et Super dans la fenêtre modale de Login et qu'on clique sur Login, on devient un utilisateur connecté	OK
	T14	16/06/17	Depuis la fenêtre modale de Login si on entre aucune information ou des informations fausses, on a un message d'erreur qui apparaît.	OK
	T15	16/06/17	Si on clique sur la petite croix et qu'on quitte la fenêtre modale de login quand il y a un message d'erreur, on revient sur la page principale. En re cliquant sur l'icône du	OK

			bonhomme une fenêtre modale login vide s'ouvre	
Fenêtre Modale de Sign up	T16	16/06/17	Depuis la modale Sign up, en cliquant sur le lien « Sign in » on a une fenêtre modale de Login qui s'ouvre.	OK
	T17	16/06/17	Si on entre aucune information dans la fenêtre modale et qu'on clique sur Sign up, on ne peut pas s'inscrire	OK
	T18	16/06/17	Si le Pseudo qu'on veut utiliser pour notre compte est déjà utilisé dans la base, un message d'erreur apparaît.	OK
	T19	16/06/17	Si on clique sur la petite croix et qu'on quitte la fenêtre modale de « Sign up » quand il y a un message d'erreur, on revient sur la page principale. En re cliquant sur le l'icône du bonhomme un fenêtre modale login vide s'ouvre	OK
	T20	16/06/17	Si le Pseudo que l'on veut utiliser ne contient pas que des lettres et des chiffres et qu'on clique sur « Sign up », un message d'erreur apparaît.	OK
	T21	16/06/17	Si le mot de passe et la confirmation ne sont pas pareils et qu'on clique sur « Sign up », un message d'erreur apparaît.	OK
	T22	16/06/17	Si on entre un pseudo et un mot de passe valide et qu'on clique sur « Sign up », on devient un utilisateur connecté	OK
Utilisateur connecté	T23	16/06/17	En cliquant sur une étoile vide sous une recette, elle se remplit. On a ajouté un favori	OK
	T24	16/06/17	En cliquant sur une étoile remplie, elle se vide. On a enlevé un favori	OK
	T25	16/06/17	En cliquant sur « Add a comment » un input pour ajouter un commentaire apparaît.	OK
	T26	16/06/17	En cliquant sur « Add a comment » quand l'input est déjà là, l'input disparaît	OK
	T27	16/06/17	Quand l'input est présent, si on écrit quelque chose dedans et qu'on clique sur « send comment », le commentaire apparaît sous la recette avec notre pseudo	OK
	T28	16/06/17	Faire T6, T7, T8 Faire T25, T27 La recherche apparaît avec le commentaire ajouté au bon endroit	OK
	T29	16/06/17	En cliquant sur l'icône de liste toutes les recettes que l'utilisateur a ajoutées	OK

			apparaissent	
T30	16/06/17		En cliquant sur l'icône étoile dans la navigation, toutes les recettes déjà mises en favori apparaissent.	OK
T31	16/06/17		En cliquant sur l'icône plus dans la navigation, la fenêtre modale d'ajout de recette apparaît	OK
T32	16/06/17		Faire le T31 Si on clique sur « Add Recipe » Un message d'erreur apparaît disant qu'il faut remplir tous les champs	OK
T33	16/06/17		Faire le T31 Si on ajoute un Titre, une liste d'ingrédient et qu'on clique sur « Add Recipe » Un message apparaît disant qu'il faut remplir tous les champs mais les champs remplis auparavant sont toujours remplis.	OK
T34	16/06/17		Faire T31 Faire T33 En cliquant sur le bouton pour ajouter une image, un pop-up apparaît pour sélectionner un fichier. Si on sélectionne un fichier image et on clique sur ajouter, le rond devient un aperçu de l'image.	OK
T35	16/06/17		Faire T31 Faire T33 En cliquant sur le bouton pour ajouter une image, un pop-up apparaît pour sélectionner un fichier. Si on sélectionne un fichier qui n'est pas une image et on clique sur ajouter, le rond devient est vide	OK
T36	16/06/17		Faire T35 Cliquer sur « Add Recipe » un message apparaît disant que l'image n'a pas la bonne extension.	OK
T37	16/06/17		Remplir tous les champs sans ajouter une image. Cliquer sur « Add image ». Aller sur la vue de ses propres recettes. On retrouve la recette qu'on vient d'ajouter sans image.	OK
T38	16/06/17		Remplir tous les champs en ajoutant une image. Cliquer sur « Add image ». Aller sur la vue de ces propres recettes. On retrouve la recette qu'on vient d'ajouter avec l'image.	OK
T39	16/06/17		Faire T38 Cliquer sur la petite croix à côté de la recette. La recette est supprimée	OK

	T40	16/06/17	Faire tous les tests de l'utilisateur non connecté	OK
	T41		Faire T29 S'il n'y a rien dans la page utilisateur, un message apparaît disant que l'utilisateur n'a pas encore ajouté de recettes.	OK
	T42	16/06/17	Faire T29 Tester les filtres sur les recettes utilisateurs. L'utilisateur doit pouvoir trier ses recettes par type et date.	OK
Administrateur	T43	16/06/17	Cliquer sur l'icône avec la flèche. L'utilisateur est déconnecté.	OK
	T44	16/06/17	Faire tous les tests de l'utilisateur connecté	OK
	T45	16/06/17	En cliquant sur l'icône de l'écrou dans la barre de navigation, l'administrateur peut voir les recettes à valider	OK
	T46	16/06/17	Faire T45 S'il y a une recette, cliqué sur le vu. La recette disparaît et un message de validation apparaît	OK
	T47	16/06/17	Faire T45 S'il y a une recette, cliqué sur la croix. La recette disparaît et un message de suppression apparaît.	OK
	T48	16/06/17	Faire T45 Tester les filtres sur les recettes à valider. L'administrateur doit pouvoir trier les recettes à valider par type et par date	OK

6. Conclusion

6.1 Différence entre le planning prévu et réel

6.1.1 Planning final

[illegible]

Contrairement à mon planning initial que l'on trouve sous le cahier des charges, le planning final était beaucoup plus décousu. Il y a des jours comme on le voit où je n'ai même pas fait de documentation car j'étais vraiment à fond dans mon développement. On remarque aussi que j'ai rajouté certaines choses au planning comme l'inscription ou encore la gestion de ses recettes pour l'utilisateur. C'était des tâches nécessaires pour accomplir mon cahier des charges mais elles ne se trouvaient pas dans le planning initial. Finalement, je pense avoir passé beaucoup de temps sur certaines fonctionnalités qui devaient être beaucoup plus courtes initialement comme la création des recettes. Cette tâche aurait dû me prendre ½ journée mais à la place cela m'a pris une journée et un quart. Il y a aussi des tâches qui auraient dû me prendre plus de temps mais qui ont été plus courtes comme la gestion des droits des administrateurs et utilisateurs. Initialement, cette tâche devait prendre 1 journée, elle m'a finalement pris 1 quart d'une journée. Malgré tous ces petits changements, j'ai quand même réussi à tout faire sur les 10 jours impartis.

6.2 Améliorations possibles

Il y a plusieurs choses qui pourraient être améliorées dans mon projet :

- Donner la possibilité à l'utilisateur de modifier ses recettes qui n'ont pas encore été validées.
- Donner cette même possibilité à l'administrateur pour qu'il puisse légèrement modifier des recettes qui pourraient être publiées.
- Permettre à l'utilisateur de supprimer ses propres commentaires
- Améliorer la saisie pour l'insertion de recette avec un sticky form sur l'image
- Refactoriser le code pour qu'il soit plus optimal surtout au niveau des requêtes SQL

6.3 Différences entre l'analyse et le produit fini

Durant l'analyse fonctionnelle, j'avais mis comme option de filtre « Favorites » pour que l'utilisateur puisse filtrer par ses favoris. Pour finir, comme les filtres sont cumulables, j'ai dû enlever cette option des filtres et la mettre dans la barre de navigation. Ainsi, l'utilisateur peut retrouver ses favoris en cliquant sur l'étoile dans la barre de navigation. C'est le seul gros changement qu'il y a entre l'analyse et le produit final.

6.4 Bilan personnel

Je suis assez satisfaite du résultat que j'ai obtenu. Ces deux semaines de TPI étaient très intenses et je suis contente d'être arrivée au bout avec un programme qui fonctionne bien. La difficulté avec un travail comme celui-ci pour moi est d'évaluer le temps que chaque tâche prend et de savoir à la fin de chaque journée quel pourcentage du travail a réellement été accompli. Malgré cela, j'ai beaucoup aimé être complètement autonome et devoir gérer mon propre temps de travail. C'est très satisfaisant de voir que je peux faire un projet de cette ampleur et démontrer les compétences que j'ai accumulées durant ces trois ans d'apprentissage.

7. Bibliographie

7.1 Sites utilisés pour la programmation

- <https://stackoverflow.com/>
- <http://php.net/>
- <http://jquery.com/>
- <http://getbootstrap.com/>
- <https://bootsnipp.com/>
- <http://phpspot.com/>
- <https://github.com/>
- <https://dev.mysql.com/>
- <https://developer.mozilla.org/fr/>
- <https://www.w3schools.com/>
- <https://git-scm.com/>

7.2 Recettes utilisées

- <https://minimalistbaker.com/>
- <http://ohsheglows.com/>
- <http://www.jamieoliver.com/recipes/category/special-diets/vegan/>

7.3 Aide reçue

- Mme Travnjak
- Mes camarades
- M Bonvin

8. Table des figures

Figure 1 - Site végémiam.....	6
Figure 2 - Modal de login au site	8
Figure 3 - Page index.php	8
Figure 4 - Modal d'ajout de recette	9
Figure 5 - Validation par l'administrateur	9
Figure 6 - Plan du site.....	13
Figure 7 - VerificationAdd(\$param).....	16
Figure 8 - VerifyImg(\$files).....	16
Figure 9 - MoveImg(\$files).....	17
Figure 10 - Call ajax pour ajouter un favori.....	21
Figure 11 - Afficher recette pour la validation	22
Figure 12 - Trouver l'id de la recette.....	22

Figure 13 - Call AJAX pour l'auto complétion	24
Figure 14 - Fonction de filtrage	25
Figure 15 - Succès du call Ajax.....	25

9. Annexes

- Classe DBConnect
- Planning initial
- Planning final
- Manuel utilisateur
- Code source
- Journal de bord manuscrit