# RAPPORT PROJET WEB 2

-Olivier Mertz

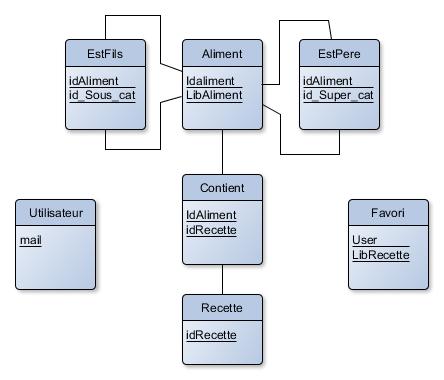
-André Erba

-Jérome Lartillot

**Introduction**

Pour ce projet à réaliser à trois nous avons utilisé **GIT** (un outil de versionnement).  
Nous nous somme répartie les taches selon trois parties principales, l’accès hiérarchiques, la connexion utilisateur et la gestion des favoris. Et enfin la partie supplémentaire de recherche à partir d’ensemble d’aliments.

1. **Organisations de la base données**



La base de données a été normalisé le plus possible pour correspondre aux besoins fonctionnelles. L’installation de la base est prise en charge par initialisation.php. Nous avons aussi choisi d’implémenter des id uniques pour chaque recette et chaque aliment pour donner plus de rapidité dans les requêtes.

La table estFils sert à trouver quelles sont les sous catégories de l’idAliment.  
la table estPere sert à trouver quelles sont les super catégories de l’idAliment.

La table contient permet de connaitre grâce à la double clé primaire, les aliments contenus dans chaque recette.

La table aliment nous permet d’avoir accès à l’id d’un aliment et son libellé.

La table Favori contient les recettes préférées de l’utilisateur en fonction de son nom.

La table utilisateur nous permet de stocker quant à elle toute les données relatives à un utilisateur.

**2.Gestion/vérification du formulaire**

Login.php contient formulaireEnregistrement et formulaireIdentification.   
Ses fonctions affichent respectivement leur formulaire ou indiquent quel'utilisateur est déjà connecté. Elles testent éventuellement le formulaire s’il est passé à l'aide de loginconfig.php.  
Voici comment sont organisés les fichiers :

loginconfig utilise loginsql.php.  
login.php contient les deux principales fonctions à appeller pour   
afficher le formulaire d'enregistrement ou de connexion ainsi qu'une   
petite fonction postToField très utilisée dans le formulaire qui se   
contente d'afficher un champ si il existe dans $\_POST.  
loginsql.php contient toutes les fonction utilisant du sql qui les   
formulaires vont utiliser, notamment la création de la table   
utilisateur, l'ajout d'un utilisateur et le la fonction qui permet à un   
utilisateur de se connecter via mail et mot de passe. C'est également   
dans ce fichier qu'est définie la fonction pour hacher le mot de passe   
(sha 256)  
session.php contient principalement les fonctions utiles à la session.   
startSession démarre la session si ce n'est pas fait, et s'assure que la   
variable connu vaut true ou false, pour savoir si l'utilisateur est   
connecté ou non. Si ce n'est pas le cas, un pseudo "Inconnu(e)" lui est   
attribué. printSession affiche Bienvenue <pseudo>Connécté/non connecté.   
Cette fonction est appelée lorsque l'utilisateur est connecté.   
fillSession($array) remplit la session avec un tableau associatif   
valide. C'est surtout prévu pour $\_POST.

**3.Gestion des favoris**

Les favoris ne sont accessibles que si l’utilisateur est connu, nous pouvons savoir cela grâce au tableau $\_SESSION qui contient une valeur booléenne. Une fois que l’utilisateur est connecté il faut recharger la page pour que l’information sois prise en compte

Il y’a deux composantes principales :

-La première est l’ajoute de favori, cela est géré par un champ d’auto complétion (grâce à JQUERY UI que nous détaillerons plus tard.), dans ce champ on peut choisir une recette et en cliquant sur le bouton « Ajouter un favori » cette recette sera ajouter dans la table Favori.

-la deuxième est la suppression d’une recette favorite, on a toujours un champ d’auto complétion, celui-ci prends ses données directement de la table Favori pour aider l’utilisateur dans la saisie.

Une composante supplémentaire : On affiche les favoris de l’utilisateur grâce à une requête sql qui va chercher chaque recette dans les favoris de l’utilisateur.

L’interaction de suppression et d’ajout de favori sont réalisés en AJAX grâce à la méthode $.Post, par contre l’affichage lui nécessite un rechargement de la page. Ici il y’a donc une amélioration à faire en effet. Àchaque action de type ajout/suppression on aurait pu modifier directement le contenu html de la div qui sert d’affichage toujours avec $.post et une fonction en troisième argument qui se charge de modifier le contenu de la div.

Exemple :$.post(url,data,function(data){ $(div).html(data) ;}) ;

4. **Gestion de la partie « Trouver ma recette »**

Dans cette partie il y’a une utilisation concrète de la fonction donnée exemple dans la partie précédente.

Dans un premier temps nous avons mis en place deux colonnes de boutons. Ainsi au début il n’y que deux boutons apparent mais on peut en afficher jusqu’à 6 au total. Ce choix a été fait car il n’y a que peu de recette qui contienne autant d’aliment.  
 De plus ce nombre total de 6, veux dire que l’on peut avoir jusqu’à 6 aliments que l’on veut ou 5 + 1 que l’on ne veut pas ou 4 +2 que l’on ne veut pas etcetc. Chaque bouton n’est pas obligé d’être remplie,s’il n’y a rien, une simple chaine vide sera envoyée et ignorer par le script averagesearch.php qui se charge de traiter les données.

Toutes les données sont encapsulés par le php de façon procédural(avec mysqli\_real\_escape\_string).  
 Dans un deuxième temps les données sont envoyés grâce à $.post à averagesearch.php qui se charge de rechercher tousles fils des aliments voulues et tous les fils des aliments non voulues, cela donne lieu à l’existence de deux tableaux différents. Puis d’un seul qui représente la différence du tableau d’aliments voulue et d’aliment non voulue. Enfin on va chercher dans la base de données les recettes qui contiennent ces aliments.  
  
 Maintenant il est temps d’attribuer une note à chaque recette. Cela a été fait de la façon la plus logique possible, c’est-à-dire que chaque recette contient des aliments voulus mais aussi des aliments qui sont neutres (pas spécialement désiré par l’utilisateur mais pas détesté non plus).

Donc la note consiste en un simple rapport des aliments désirés et contenu dans une recette, comparé aux nombres d’aliments total de la recette.

Soit pour chaque recette : (NB\_ALIMENT\_VOULUE) /(NB\_ALIMENT\_TOTAL\_RECETTE\_COURANTE)

Après avoir traité cela le script averageseach.php se charge d’afficher (cela rendue possible grâce l’utilisation de .html(data) dans $.post) chaque recette trouvé avec leur note .

**5. Recette (accès hiérarchique)**

Cette partie est divisée en trois zones.

1. Zone qui fait apparaitre par défaut la liste de tous les aliments depuis la racine.  
   Un bouton indique qu’on est sur « Aliments ».  
   A chaque fois qu’on clique « Voir les sous-catégories », la liste correspondante apparait à la place et un nouveau bouton est ajouté. Cette série de boutons permet de parcourir la hiérarchie. Il suffit de cliquer dessus. Si on clique sur le nom de l’aliment, on fait apparaitre la liste des recettes correspondant à cet aliment dans la partie droite de la fenêtre.
2. Dans la liste des recettes, il suffit de cliquer sur le nom de la recette pour que celle-ci apparaissent au bas de la page.
3. Dans la description de la recette, on affiche le titre, les ingrédients ainsi que la préparation. Si une photo existe, celle-ci est affichée.

Toute la technologie utilisée est décrite dans le fichier interface\_script.js

**6. Utilisation de JQUERY et JQUERY UI**

Pour faciliter des interactions, la compatibilité et parfois tout simplement le design du site,

nous avons utilisé ces deux libraires. Jquery permet de prendre en compte beaucoup d’évènement supplémentaires en particulier grâce à $.post et à l’évènement .click.

L’ensemble des boutons modulables dans la parties « Trouver ma recette » a été réalisé grâce à JQUERY qui déclenche un événement sur le bouton « Ajouter un Aliment » à chaque clique sur se bouton il y’a un nouveau bouton qui est ajouter(.append()) dans le wrapper contenant les boutons.  
Le nombre de boutons disponible peux être facilement modifié dans le javascript , la limite du nombre des boutons n’étant qu’une variable entière à définir.

* 1. Auto Completion

JqueryUi a été très utile pour une implémentation simple de l’auto complétion en effet il a fallu juste invoquer .autocomplete() sur un sélecteur Jquery , .autocomplete() appelant juste un script php qui se charge d’aller chercher les données qu’il faut et les renvoyer en JSON à la fonction. De plus cette méthode implémente bien sûr l’ajax dans l’auto complétion. Tout cela permet un meilleur guidage de l’utilisateur dans les saisies.

* 1. Affichage des données en AJAX dans différentes balises

Grâce à la méthode $.post implémenté dans JQUERY nous avons pu afficher dynamiquement de nombreuse données et ainsi crée une interface simple d’utilisation pour l’utilisateur.

* 1. Différents widget

Nous avons utilisé .menu() sur notre liste qui gère notre navigation, cela a permis une mise en forme rapide. De plus les liens entre les div ont été réalisés par des sélecteurs JQUERY qui se charge de faire des FadeIn ou FadeOut selon la composante de notre site qui est ciblé par l’évènement :target.

Il y a aussi une implémentation de .DatePicker() sur l’input date du formulaire. Ce widget permet de déployer rapidement un calendrier cliquable pour l’utilisateur, ceci toujours dans le souci d’aider l’utilisateur.

* 1. Theming de JQUERY UI

Pour stylisé les différents widgets, il y’a à chaque fois des classes déjà présentes sur les plusieurs niveaux que comporte un widget. Le thème peut aussi être définie directement sur le site du JQUERY UI puis télécharger et utiliser dans l’application, ce qui fait que à ce moment-là on se décharge de toute la partie design des widgets. De plus si les widgets sont utilisés de la bonne façon le site peux être produit rapidement et facilement.

7. CONCLUSION

Ce projet nous a permis de comprendre les interactions en jeu sur un site web du côté client, au côté serveur. Il a permis aussi d’avoir une bonne pratique dans le domaine de l’AJAX et du traitement de base de données en PHP.