Rapport de projet

Lancement du site « GiselleMagicArts.com » !

Podevin Jean-Clément

Boogearts Paul

Rigaut Arnaud

Kruger Maxence

Promo A1 2016-2017

Sommaire :

Sommaire des images :

Introduction

A- Rappel du contexte

Giselle souhaite vendre ses potions. Pour cela, elle fait appel à Miguel pour faire un site internet où cette dernière pourra vendre ses potions, ses ingrédients et ses onguents.
Les onguents possèdent les mêmes ingrédients que les potions sans diluant et sans la nécessité de devoir chauffer.
Concernant les ingrédients, Giselle doit faire appel à différents fournisseurs que cela soit pour des ingrédients différents ou des mêmes ingrédients.
Les ingrédients doivent respecter une certaine fraicheur qui s'évaluera en jours et il ne faut pas dépasser un certain nombre de jours, et chaque ingrédient doit être regroupé en fonction d'une même fraicheur.
Pour la fraicheur, l'ingrédient doit avoir un seuil de fraicheur permettant de déterminer s'il est possible d'être inférieur ou supérieur à un certain degré pour la potion ou non.
Contrairement aux situations vues précédemment, Giselle ne souhaite plus gérer le type d'ingrédient et le type de magie.
Cependant, les clients peuvent, en plus commander le récipient de leur choix (Fioles, Tubes, Pots) où chaque récipient a un prix et une quantité différente.
Ils peuvent également commander un même ingrédient à des degrés de fraicheur différents.
Pour les commandes, les onguents n'ont pas d'option, il n'est possible que de choisir la quantité voulue.

B- Le Besoin

Giselle souhaite qu'on l'aide à construire la base de données permettant de favoriser sa vente de produits.

Pour effectuer sa base de données, il faut d'abord avoir fait les prérequis comme identifier les acteurs et les flux afin d'effectuer le diagramme de flux du projet.

Ensuite il faut effectuer le dictionnaire de données qui va mener à la réalisation de la matrice de dépendance fonctionnelle, du MCD, du MLD et du MPD.

Une fois le MPD réalisé et la justification du système de gestion de base de données relationnel fait, il faudra préparer le script de création de la base de données avant de la remplir.

Pour remplir la base de données certaines contraintes réglementées à l'aide de requêtes :

1. **Les procédures stockées :**

- Affichage du stock des ingrédients et mise à jour à la réception de nouveaux ingrédients
- Consultation des commandes d'un client
- Suppression d'un ingrédient arrivé à expiration (pourra être éventuellement automatisé) - Mise à jour du stock après envoi d'une commande (la commande devra changer de statut)
- Ajouter une ou plusieurs nouvelles recettes
- Valider une recette qui a été acceptée
- Supprimer une recette qui n'a pas été retenue

2. **Les requêtes de recherche**

- Les potions ou les onguents réalisables avec un ingrédient précis
- Les couples (potions, onguents) utilisant la même liste d'ingrédients
- Les potions réalisables avec un diluant précis
- La liste des potions classée par température de préparation
- La comparaison entre le prix de vente d'une potion et le total du prix des ingrédients (et diluant) nécessaires à sa réalisation
- Le nombre moyen d'ingrédients utilisés pour fabriquer une potion (ou onguent)

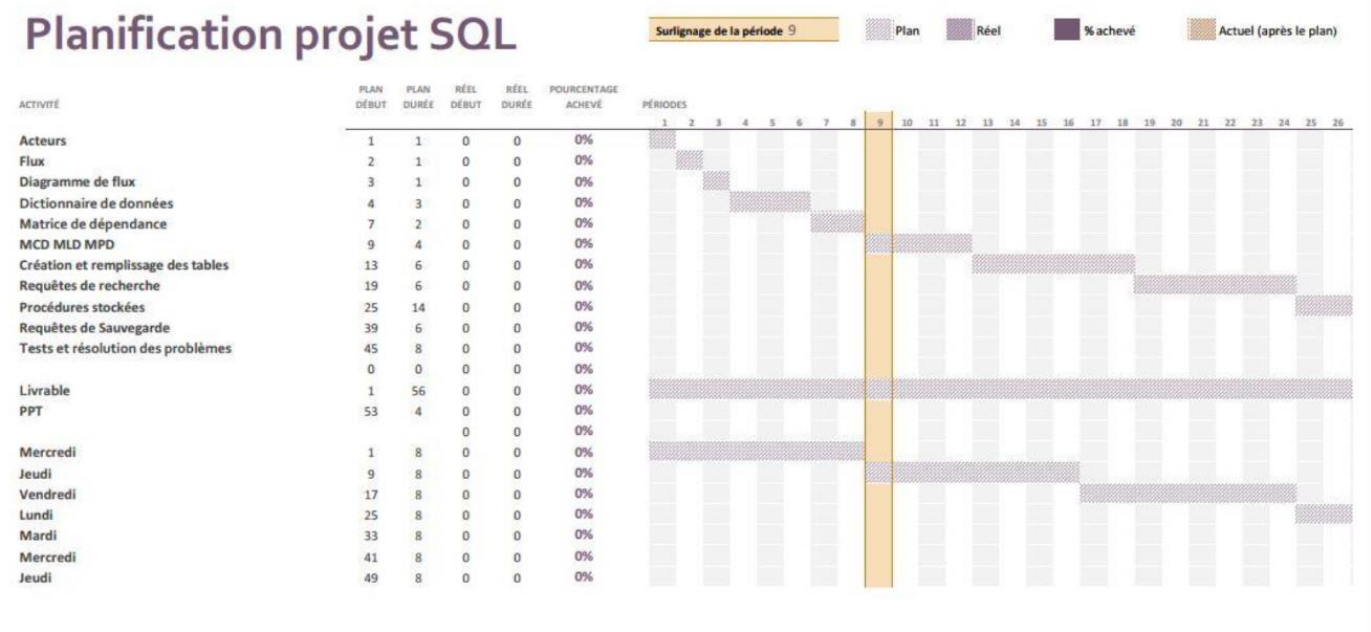
3. **Requêtes de sauvegarde**

- Recettes enregistrées dans la base de données
- Ingrédients associés à chaque fournisseur
- Commandes dont la réception n'a pas encore été confirmée

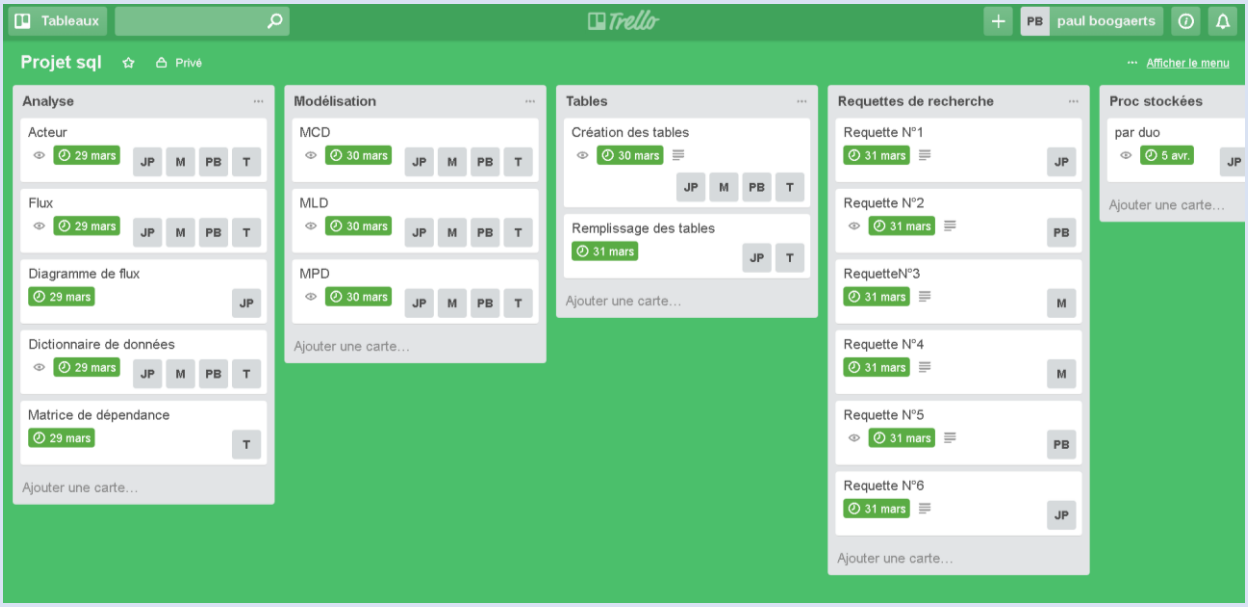
Il faut pouvoir restaurer les informations ci-dessus dans une base de données vide, de ce fait, les éléments doivent être sauvegardés.

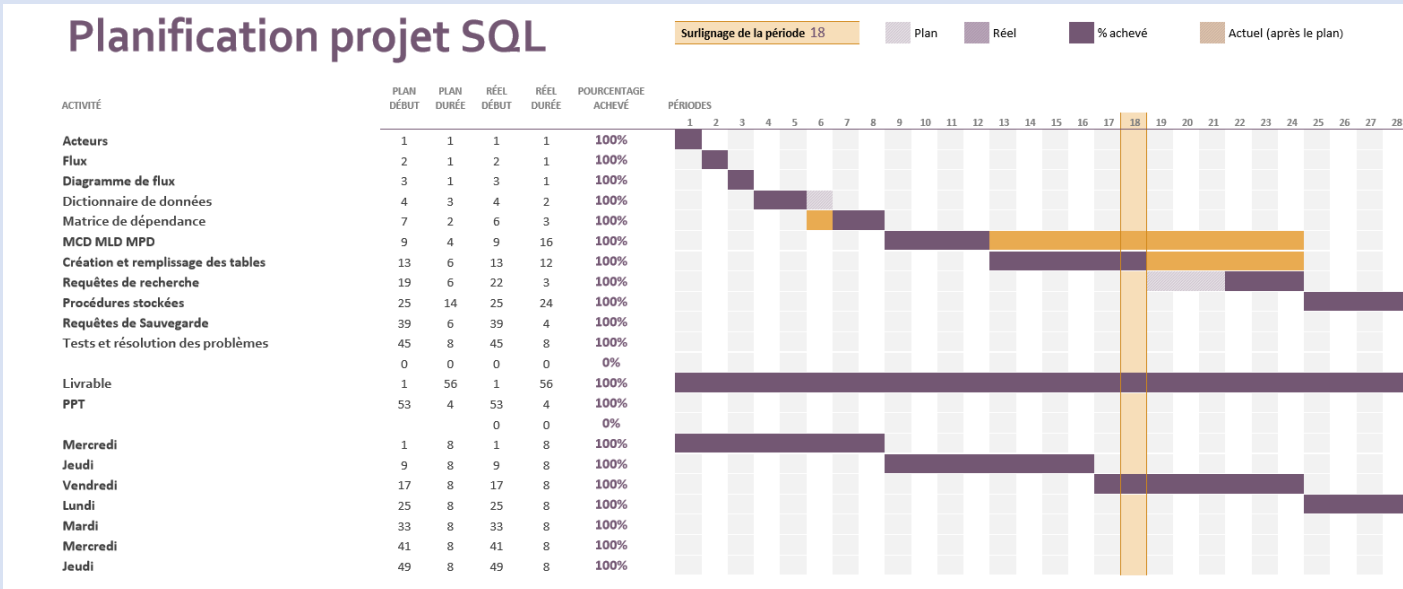
C- Organisation

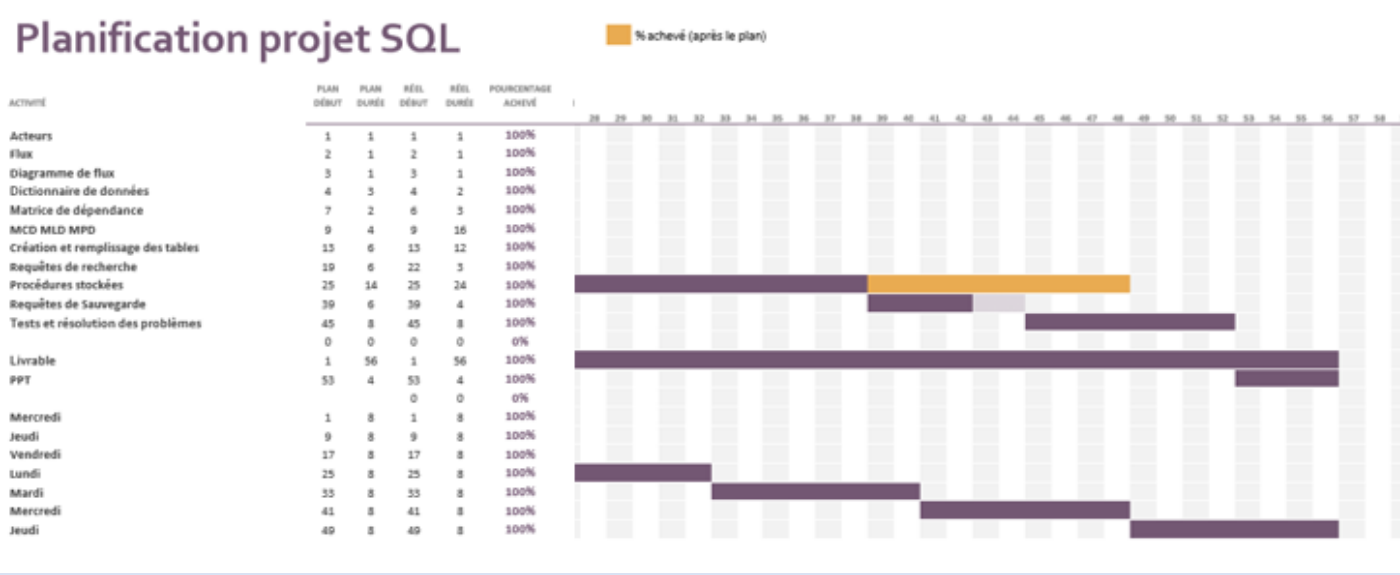
Concernant l'organisation de notre équipe tout au long du projet, nous avons dans un premier temps créé un planning prévisionnel permettant de nous répartir les tâches à effectuer durant le temps qui nous a été imparti :



Finalement nous avons opté pour un planning final ressemblant fortement au prévisionnel. Cependant les durées indiquées sont différentes en fonction des tâches effectuées et un Trello nous a semblé plus représentatif de ce qu'on devait établir durant







En comparant le planning prévisionnel au planning effectif, on remarque que ce dernier est bien respecté dans l'ensemble, seul un point a été différent, la réalisation du MCD car il a fallu le modifier plusieurs fois et cela n'a pas été pris en compte avant de faire le prévisionnel.

D- Contraintes

Durant ce projet nous avons été confronté à deux contraintes :

- Contrainte de durée. En effet nous avons dû réaliser le projet sur dix jours. Cela nous imposer une répartition des taches précise de la part du chef de projet
- Contrainte de modélisation. Dans le sujet il nous était imposer d'utiliser la méthode MERISE.

Analyse

A- Les Acteurs

Dans un premier temps, il nous a fallu identifier les différents acteurs pour pouvoir ensuite effectuer un diagramme de flux de la situation que l'on doit traiter.

De ce fait les différents acteurs sont :

- Le(s) Client(s)
- Miguel
- Le(s) Fournisseur(s)
- Giselle

B- Les Flux

Après avoir identifié les différents acteurs du projet, il nous a fallu identifier les flux qu'il y a entre eux :

- Entre Le Client et Giselle, il y a une interaction indirecte puisque le client passe une commande à Giselle par le biais d'un site internet. Celle-ci, va donc accepter ou nous la commande du client.

- Ensuite, Giselle interagit avec le fournisseur puisque ce dernier va fournir Giselle en ingrédients suite à la commande qu'elle aura effectuée avant.
- Miguel, lui, agit sur Giselle puisqu'il va créer le site en question où elle va pouvoir vendre ses potions, ses onguents, ses ingrédients et ses récipients.

C- Le Diagramme de Flux

Après avoir effectué ces deux étapes, il suffit de les lier entre-elles par le biais d'un schéma : Le Diagramme de Flux ci-dessous.

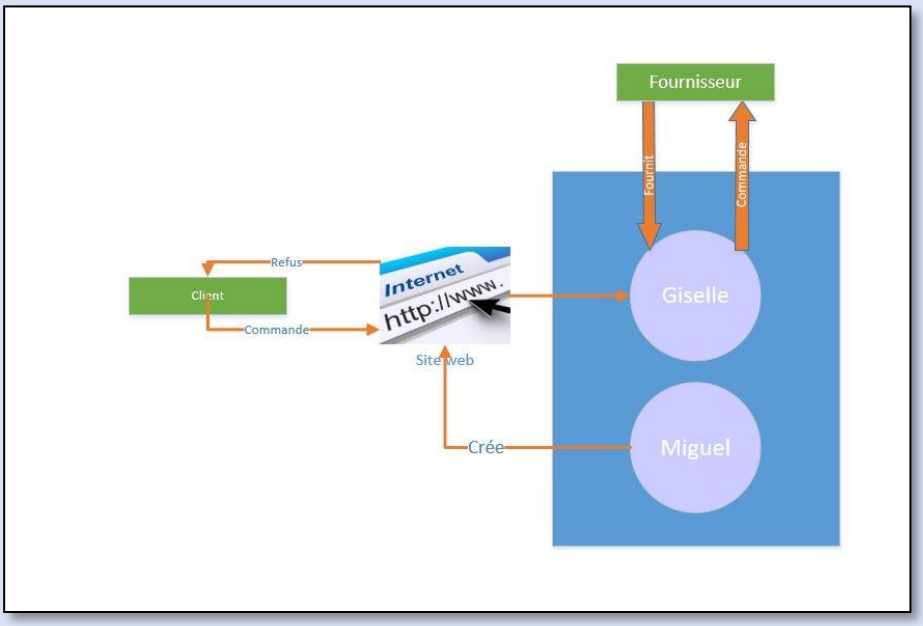


Figure 1: diagramme de flux

Modélisation

A- Dictionnaire de données

Le diagramme de flux établi, on a pu donc poursuivre dans la création de notre base de données.

La première chose à établir est donc le dictionnaire de données qui nous permet de lister tout ce dont on a besoin pour créer notre base de données.

	Nom	Type	Contrainte
1	**ID_Client**	Numerique	Obligatoire
2	Nom_client	Texte	Obligatoire
3	Prénom_client	Texte	Obligatoire
4	Tel_Client	Numerique	Obligatoire
5	Civilite	Texte	Obligatoire
6	Code_Postal	Numerique	Obligatoire
7	Ville_Client	Texte	Obligatoire
8	Adresse_Client	Alphanumérique	Obligatoire
9	Mail	Alphanumérique	
10	**ID_Ingredient**	Numerique	Obligatoire
11	Nom_Ingredient	Texte	Obligatoire
12	Prix_Ingredient	Monetaire	Obligatoire
13	Temps_conservation	Numerique	Obligatoire
14	Seuil_Fraicheur	Numerique	
15	**ID_Inventeur**	Numerique	Obligatoire
16	Nom_Inventeur	Texte	Obligatoire
17	Prénom_Inventeur	Texte	Obligatoire
18	**ID_Commande**	Numerique	Obligatoire
19	No_Commande	Numerique	Obligatoire
20	Prix_ Commande_total	Monetaire	Obligatoire
21	Date_de_commande	Date	Obligatoire
22	**ID_Progression**	Numerique	Obligatoire
23	Statut_Progression	Alphanumérique	Obligatoire
24	**ID_Fournisseur**	Numerique	Obligatoire
25	Nom_Fournisseur	Alphanumérique	Obligatoire
26	Tel_Fournisseur	Numerique	Obligatoire
27	Mail_Fournisseur	Alphanumérique	
28	**ID_Recipient**	Numerique	Obligatoire
29	Nom_Recipient	Texte	Obligatoire
30	Contenance_Recipient	Texte	Obligatoire
31	Prix_Recipient	Monetaire	Obligatoire
32	**ID_Onguent**	Numerique	Obligatoire
33	Nom_Onguent	Texte	Obligatoire

Figure 2: Dictionnaires de données

34	Prix_Onguent	Monetaire	Obligatoire
35	**ID_Dilluant**	Numerique	Obligatoire
36	Nom_Diluant	Texte	Obligatoire
37	Prix_Diluant	Numerique	Obligatoire
38	**ID_Potion**	Alphanumérique	Obligatoire
39	Nom_Potion	Monetaire	Obligatoire
40	Prix_Potion	Numerique	Obligatoire
41	Temperature	Numerique	Obligatoire
42	Valider_Potion	Date	Obligatoire
43	**ID_Ingredient en stock**	Numerique	Obligatoire
44	Date_Entrer	Date	Obligatoire
45	Arrivé_Depuis	Date	Obligatoire
46	**ID_Onguent_en_Stock**	Numerique	Obligatoire
47	Quantité_Stock_Onguent	Numerique	Obligatoire
48	**ID_Diluant_en_stock**	Numerique	Obligatoire
49	**ID_Personne**	Numerique	Obligatoire
50	Nom_Personne	Numerique	Obligatoire
51	**ID_Recipient_en_stock**	Numerique	Obligatoire
52	**ID_Stock_Potion**	Numerique	Obligatoire
53	Quantite_stock_potion	Numerique	Obligatoire

B- Matrice de dépendance fonctionnel

		1	2	3	4	5	6	7	8	9	10	11	12	13	15	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51
1	**ID_Client**																																												
2	Nom_client	x																																											
3	Prénom_client	x																																											
4	Tel_Client	x																																											
5	Civilite	x																																											
6	Code_Postal	x																																											
7	Ville_Client	x																																											
8	Adresse_Client	x																																											
9	Mail	x																																											
10	**ID_Ingredient**																																												
11	Nom_Ingredient										x																																	
12	Prix_Ingredient										x																																	
13	Temps_conservation										x																																	
14	Seuil_Fraicheur										x																																	
15	**ID_Inventeur**																																												
16	Nom_Inventeur															x																													
17	Prénom_Inventeur															x																													
18	**ID_Commande**																																												
19	No_Commande																		x																										
20	Prix_ Commande_total																		x																										
21	Date_de_commande																		x																										
22	**ID_Progression**																																												
23	Statut_Progression																		x																										
24	**ID_Fournisseur**																																												
25	Nom_Fournisseur																								x																				
26	Tel_Fournisseur																								x																				
27	Mail_Fournisseur																								x																				
28	**ID_Recipient**																																												
29	Nom_Recipient																												x																
30	Contenance_Recipient																												x																
31	Prix_Recipient																												x																
32	**ID_Onguent**																																												
33	Nom_Onguent																																x												
34	Prix_Onguent																																x												
35	**ID_Dilluant**																																												
36	Nom_Dilluant																																			x									
37	**ID_Potion**																																												
38	Nom_Potion																																						x						
39	Prix_Potion																																						x						
40	Temperature																																						x						
41	**ID_Frigo**																																												
42	Date_Entrer																																												
43	Arrivé_Depuis																																												
44	Quantité_par_date																																												

Figure 3: Matrice de dépendance fonctionnel

C- MCD MLD et MPD

Nous avons utilisé le logiciel JMerise pour créer notre MCD ainsi que le MLD, MPD et le script de création de la base.

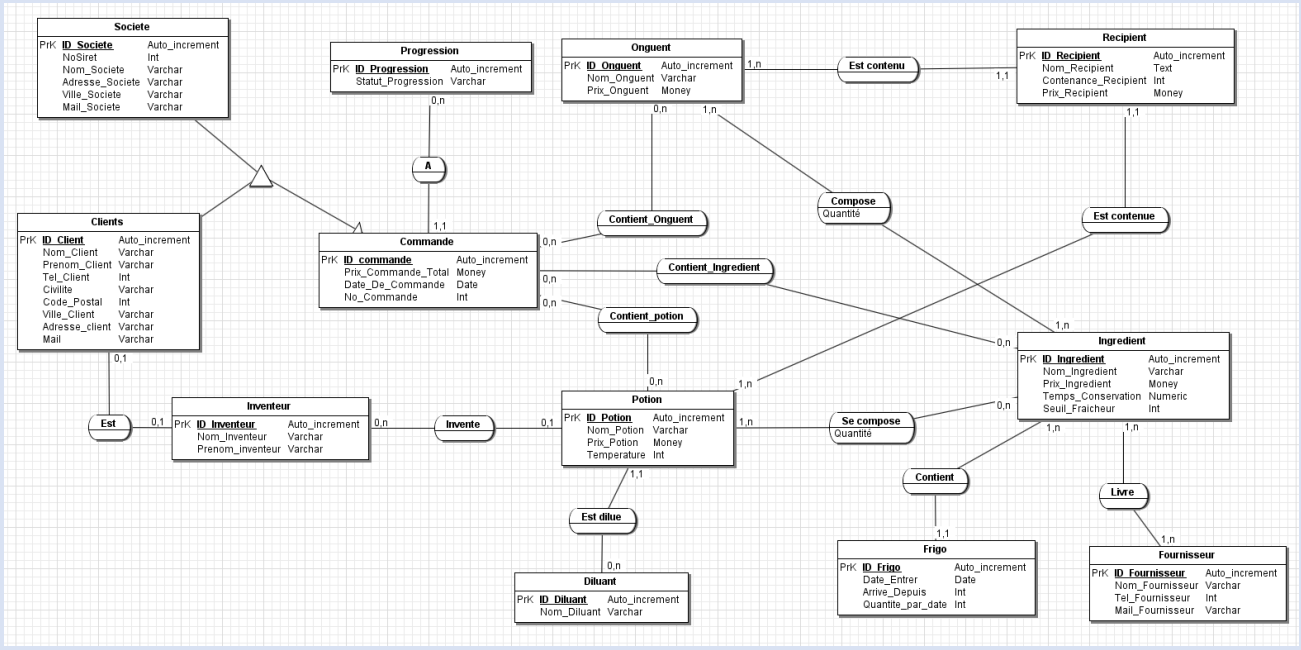


Figure 4: MCD

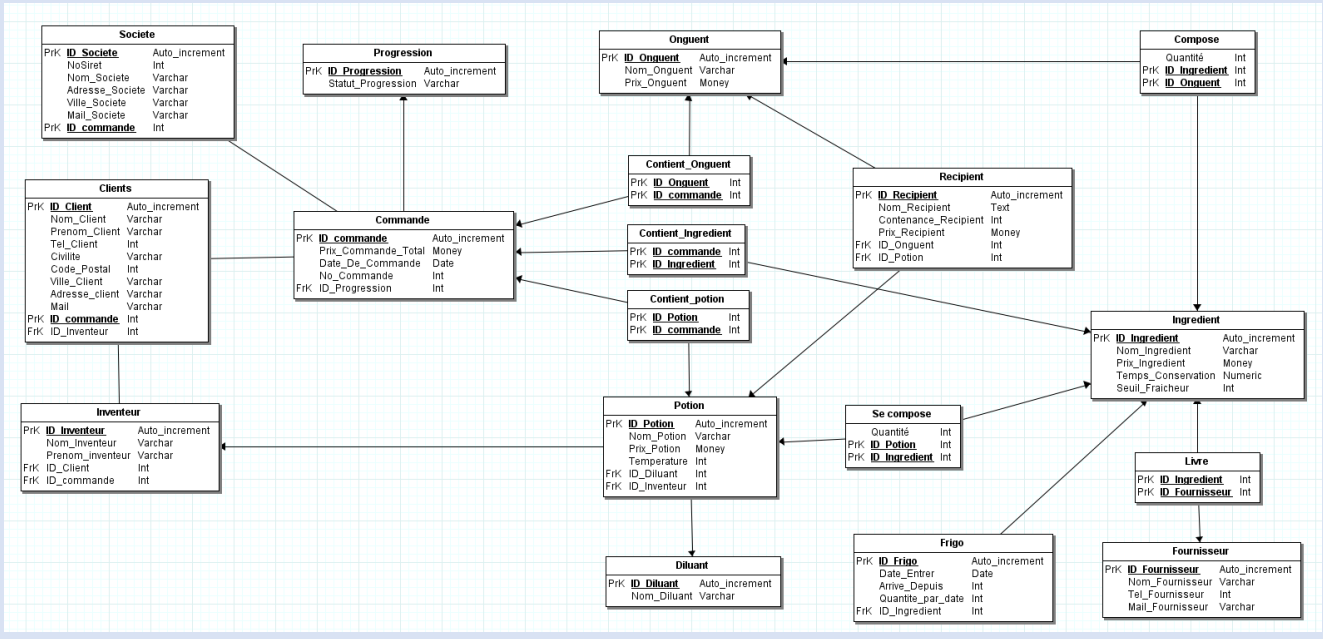


Figure 5: MLD

Les Requêtes

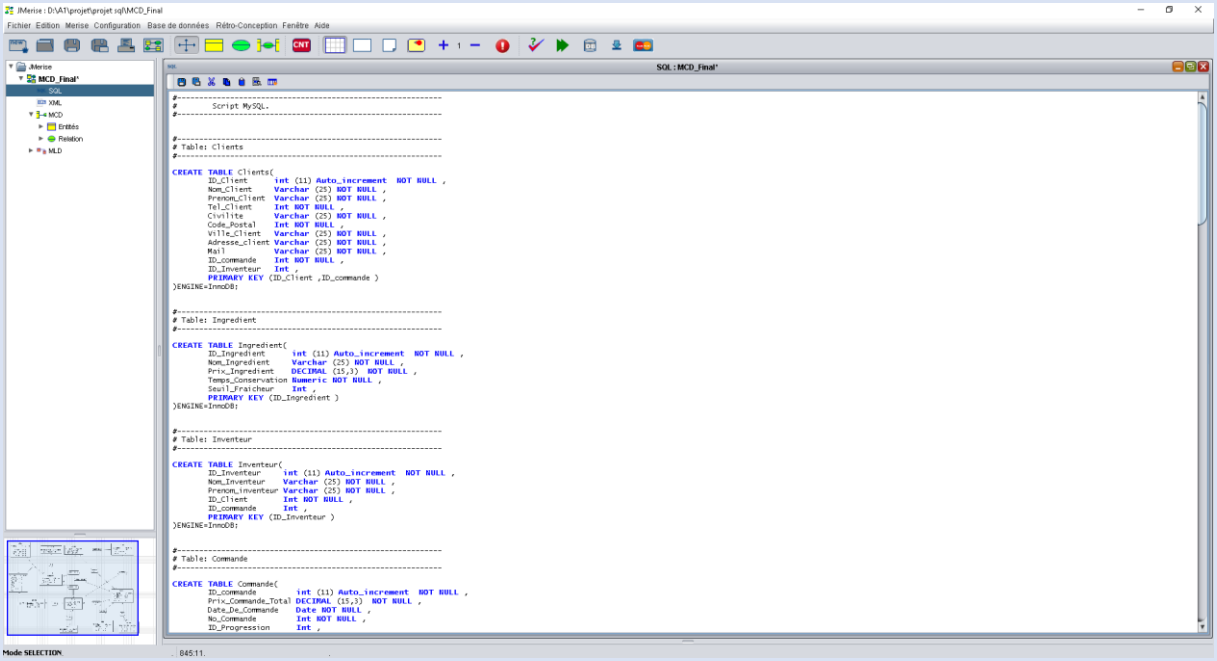


Figure 6: Script de création de la base

Pour avoir le script final et total se référer au GITHUB

Choix du SGBDR :

Nous avons choisi MySQL pour réaliser notre projet car nous avons été former à celui-ci pendant nos prosit et il est actuellement gratuit.

A- Les procédures stockées

Affichage du stock des ingrédients et mise à jour à la réception de nouveaux ingrédients

```
DELIMITER |
CREATE PROCEDURE Ajout_Ingredient(ingre INT, nbr INT, fourni INT, voiringre CHAR)
BEGIN
INSERT INTO `Ingredient_en_stock` (`Date_Entrer`,`Arrive_Depuis`,`ID_Ingredient`, `Quantite_Ingredient`,`ID_Fournisseur`)
VALUES
(CURDATE(), 0, ingre, nbr, fourni);

SELECT ingredient_en_stock.*
FROM ingredient INNER JOIN ingredient_en_stock ON ingredient.ID_Ingredient = ingredient_en_stock.ID_ingredient
WHERE ingredient.Nom_Ingredient = voiringre;
END |
DELIMITER ;
```

Figure 7: procédure d'affichage du stock des ingrédients et mise à jour à la réception de nouveaux ingrédients

Première procédure stockée dont l'objectif est avant tout de mettre à jour le stock suite à la réception d'ingrédients. Pour cela on va insérer dans la bonne table les informations concernant l'ingrédient. Pour la date d'entrée on spécifie la date actuelle, et on précise l'ID de l'ingrédient ainsi que l'ID du fournisseur. Dans un second temps ou affiche la table où sont stockés les différents ingrédients. Cela permet de vérifier si l'ajout s'est bien effectué.

Consultation des commandes d'un client

```
DELIMITER |
CREATE PROCEDURE Consultation_Commande(IN p_nom VARCHAR(255))
BEGIN
SELECT DISTINCT Commande.ID_Commande, Client.Prenom_Client, Client.Nom_Client, Client.Adresse_Client, Client.Ville_Client,
                Client.Code_Postal, Commande.Prix_Commande_Total, Commande.Date_De_Commande,
                Contient_Recipient_Potion.Quantite_Recipient, recipient.Nom_Recipient
FROM ((((Client INNER JOIN Commande ON Commande.ID_Personne = Client.ID_Personne)
                INNER JOIN Contient_Recipient_Potion ON Commande.ID_Commande = Contient_Recipient_Potion.ID_Commande)
                INNER JOIN recipient_en_stock on Contient_Recipient_Potion.ID_Stock_Potion = recipient_en_stock.ID_Recipient_en_Stock)
                INNER JOIN recipient ON recipient_en_stock.ID_Recipient = recipient.ID_Recipient)
INNER JOIN Personne ON Client.ID_Personne = Personne.ID_Personne
WHERE Personne.Nom_Personne = 'p_nom' ;
END |
DELIMITER ;
```

Figure 8: Procédure de consultation des commandes

Deuxième procédure stockée dont le nom est « Consultation_Commande » et où l'objectif est de consulter la ou les commande(s) passée(s) par un client précis. Celui-ci sera préciser dans le paramètre 'p_nom' pour paramètre_nom. Le choix de stocker cette procédure permet à l'utilisateur de la base de données de chercher à n'importe quel moment les commandes d'un client et ce à n'importe quel moment. Pour réaliser l'objectif nous avons dû chercher dans les tables commande et clients des informations tel que L'ID de la commande et les informations relatifs au client. En plus de ces données élémentaires nous avons choisi d'afficher une partie de la commande. Ici il s'agira du récipient. Après appel de la commande on obtient :

ID_Commande	Prenom_Client	Nom_Client	Adresse_Client	Ville_Client	Code_Postal	Prix_Commande_Total	Date_De_Commande	Quantite_Recipient	Nom_Recipient
1	Ysolda	Fuche	47 avenue de la longue vue	Unilo	77333	1500.000	2017-10-12	1	Fiole
6	Ysolda	Fuche	47 avenue de la longue vue	Unilo	77333	1900.000	2017-10-14	1	Fiole

Figure 9: Résultat de la procédure de consultation des commandes

Dans l'exemple, le paramètre que nous avons mis est 'fuche'. Ainsi la personne dont le nom est Fuche a passé 2 commande, la commande 1 et 6.

Suppression d'un ingrédient arrivé à expiration

```
DELIMITER |
CREATE PROCEDURE Supression_Ingredient()
BEGIN
UPDATE Ingredient_en_stock
SET Arrive_Depuis = DATEDIFF(CURDATE(), Ingredient_en_stock.Date_Entrer);

DELETE s
FROM Ingredient i
INNER JOIN Ingredient_en_stock s ON i.ID_Ingredient = s.ID_Ingredient
WHERE i.Temps_Conservation <= s.Arrive_Depuis;
END |
DELIMITER ;
```

Figure 10: Procédure de suppression d'un ingrédient

Dans cette procédure nous avons voulu faciliter la suppression d'un ingrédient dont la date d'expiration avait atteint la limite. Le choix de stocker cette procédure vient du fait que celle-ci devra être lancée tous les jours. Cette procédure est découpée en deux étapes. La première est de mettre à jour le champ qui précise depuis quand est arrivé le produit. Pour cela on fait une différence entre la date actuelle et la date d'entrée du produit. Dans un second temps nous faisons la comparaison entre le temps de conservation du produit et la date depuis lequel est arrivé le produit. Dans le cas où le temps de conservation est inférieur ou égal au temps depuis lequel est stocké le produit, ce dernier est supprimé

Mise à jour du stock après envoi d'une commande

```
DELIMITER |
CREATE PROCEDURE MAJ_Stock(nbrcom INT, retrait_recipient INT, retrait_potion INT, retrait_onguent INT, retrait_ingredient INT)
BEGIN

UPDATE ((recipient_en_stock INNER JOIN Contient_Recipient_Potion ON recipient_en_stock.ID_Recipient_en_Stock = Contient_Recipient_Potion.ID_Recipient_en_Stock)
                            INNER JOIN Commande ON Contient_Recipient_Potion.ID_Commande = Commande.ID_Commande)
SET recipient_en_stock.Quantite_Recipient = recipient_en_stock.Quantite_Recipient - retrait_recipient
WHERE Commande.ID_Commande = nbrcom;


UPDATE ((Stock_Potion INNER JOIN Contient_Recipient_Potion ON Stock_Potion.ID_Stock_Potion = Contient_Recipient_Potion.ID_Stock_Potion)
                      INNER JOIN Commande ON Contient_Recipient_Potion.ID_Commande = Commande.ID_Commande)
SET Stock_Potion.Quantite_Potion = Stock_Potion.Quantite_Potion - retrait_potion
WHERE Commande.ID_Commande = nbrcom;


UPDATE ((Onguent_en_stock INNER JOIN Contient_Onguent ON Onguent_en_stock.ID_Onguent_en_stock = Contient_Onguent.ID_Onguent_en_stock)
                          INNER JOIN Commande ON Contient_Onguent.ID_Commande = Commande.ID_Commande)
SET Onguent_en_stock.Quantite_onguent = Onguent_en_stock.Quantite_onguent - retrait_onguent
WHERE Commande.ID_Commande = nbrcom;


UPDATE ((ingredient_en_stock INNER JOIN Contient_ingredient ON ingredient_en_stock.ID_ingredient_en_stock = Contient_ingredient.ID_ingredient_en_stock)
                             INNER JOIN Commande ON Contient_ingredient.ID_Commande = Commande.ID_Commande)
SET ingredient_en_stock.Quantite_Ingredient = ingredient_en_stock.Quantite_Ingredient - retrait_ingredient
WHERE Commande.ID_Commande = nbrcom;
END |
DELIMITER ;
```

Figure 11: Procédure de mise à jour du stock après envoi d'une commande

Après envoi d'une commande Giselle devra impérativement supprimer les ingrédients afin d'être à jour. C'est ce qu'elle sera amenée à faire le plus souvent il était donc impératif de stocker la procédure. En paramètre on précisera le numéro de la commande, ainsi que les différents retraits qu'il sera effectué. Ainsi pour faire la mise à jour il faudra dans un premier temps se référer à la bonne table. Ensuite grâce aux différents paramètres on enlève un certain nombre d'entités.

B- Les requêtes de recherche

Avant d'exposer les différentes requêtes que nous avons réalisées il y a un point commun entre celles qui ont un lien avec les potions. En effet nous avons réalisé les recherches avec la condition où la potion est validée.

Les potions ou les onguents réalisables avec un ingrédient précis

```
-- Recherche 1 --

SELECT Potion.Nom_Potion, Onguent.Nom_Onguent, Ingredient.Nom_Ingredient
FROM ((((Potion INNER JOIN Se_compose ON Potion.ID_Potion = Se_compose.ID_Potion)
INNER JOIN Ingredient ON Se_compose.ID_Ingredient =Ingredient.ID_Ingredient)
INNER JOIN est_compose ON Ingredient.ID_Ingredient = est_compose.ID_Ingredient)
INNER JOIN Onguent ON est_compose.ID_Onguent = Onguent.ID_Onguent)
Where Ingredient.Nom_Ingredient = 'Nom-Ingredient'
AND Valider_Potion = 'Valider';
```

Figure 12: Recherche des potions et des onguents réalisables avec un ingrédient précis

Cette première requête va nous permettre de trouver les potions et les onguents réalisables avec un ingrédient précis. Cela servira à un client si ce dernier veut réaliser quelque chose à partir d'un ingrédient précis. Cet ingrédient devra être précisé à la place de 'Nom-Ingrédient'. Par exemple avec l'ingrédient chenilles on obtient :

Nom_Potion	Nom_Onguent	Nom_Ingredient
Amortentia	Creme morte	chenilles

Figure 13: Résultat recherche des potions et des onguent réalisables avec un ingrédient précis

On peut donc conclure qu'avec l'ingrédient 'chenilles' on peut réaliser la potion 'Amortentia' et l'onguent 'Crème morte'.

Les couples (potions, onguents) utilisant la même liste d'ingrédients

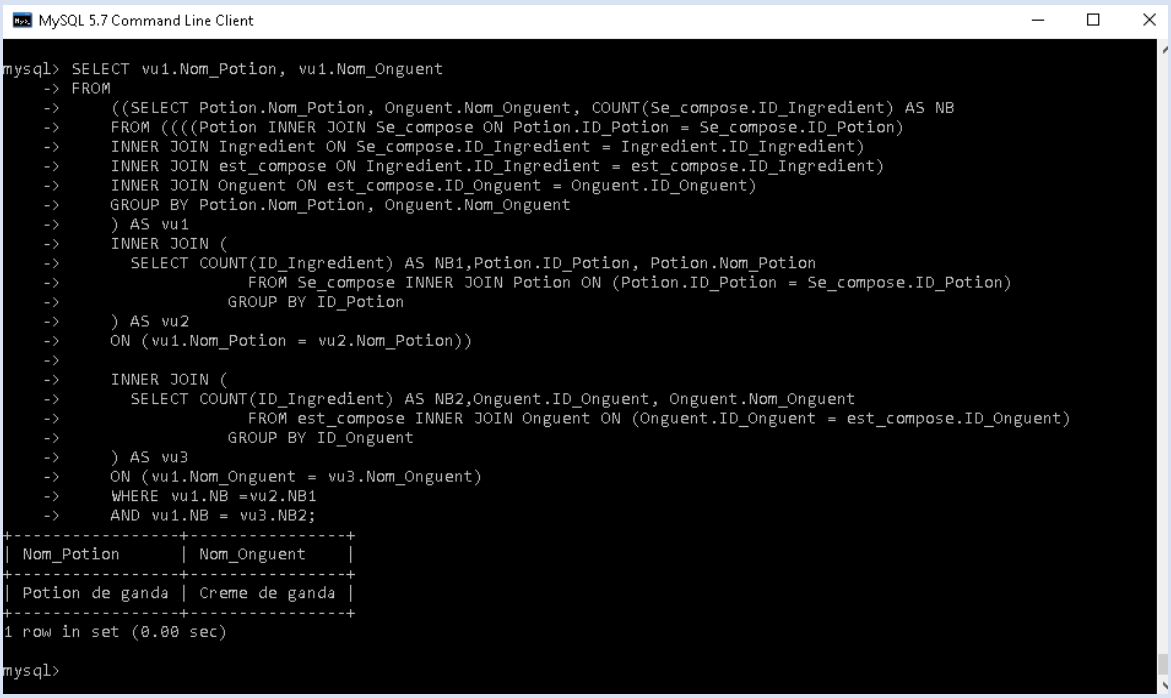
```
SELECT vu1.Nom_Potion, vu1.Nom_Onguent
FROM
    ((SELECT Potion.Nom_Potion, Onguent.Nom_Onguent, COUNT(Se_compose.ID_Ingredient) AS NB
      FROM ((((Potion INNER JOIN Se_compose ON Potion.ID_Potion = Se_compose.ID_Potion)
                      INNER JOIN Ingredient ON Se_compose.ID_Ingredient = Ingredient.ID_Ingredient)
                      INNER JOIN est_compose ON Ingredient.ID_Ingredient = est_compose.ID_Ingredient)
                      INNER JOIN Onguent ON est_compose.ID_Onguent = Onguent.ID_Onguent)
                      GROUP BY Potion.Nom_Potion, Onguent.Nom_Onguent
                      HAVING NB  IN  (SELECT COUNT(ID_Ingredient)
                          FROM Se_compose
                          GROUP BY ID_Potion)
        ) AS vu1
                INNER JOIN (
                SELECT COUNT(ID_Ingredient) AS NB1,Potion.ID_Potion, Potion.Nom_Potion
                    FROM Se_compose INNER JOIN Potion ON (Potion.ID_Potion = Se_compose.ID_Potion)
                                                GROUP BY ID_Potion
        ) AS vu2
                 ON (vu1.Nom_Potion = vu2.Nom_Potion))
                 INNER JOIN (SELECT COUNT(ID_Ingredient) AS NB2,Onguent.ID_Onguent, Onguent.Nom_Onguent
                              FROM est_compose INNER JOIN Onguent ON (Onguent.ID_Onguent = est_compose.ID_Onguent)
                              GROUP BY ID_Onguent
        ) AS vu3
ON (vu1.Nom_Onguent = vu3.Nom_Onguent)
WHERE vu1.NB =vu2.NB1
AND vu1.NB = vu3.NB2;
```

Dans cette recherche on va chercher à trouver les onguents qui ont les mêmes ingrédients de création qu'une potion.

La requête de couple Onguent – Potion permet de savoir quelles sont les Potions et les onguents ayant la même liste d'ingrédient.

La requête est composée de quatre grandes parties :

1. Nombre d'ingrédients en commun entre les potions et onguents
2. Nombre d'ingrédients par potion
3. Nombre d'ingrédients par onguent
4. Conditions : le nombre d'ingrédients en commun doit être égal au nombre d'ingrédients par potion et au Nombre d'ingrédients par onguent.



Les potions réalisables avec un diluant précis

```
-- Recherche 3 --

SELECT Diluant.Nom_Diluant, Potion.Nom_Potion
FROM (Diluant INNER JOIN Potion ON Potion.ID_Diluant = Diluant.ID_Diluant)
WHERE Diluant.Nom_Diluant = 'Nom-diluant'
AND Valider_Potion = 'Valider';
```

Figure 14: Recherche des potions réalisables avec un diluant précis

L'objectif ici est de ressortir les potions réalisables avec un diluant précis. De même que dans la recherche précédente on précisera le nom du diluant dans 'Nom-Diluant'. Par exemple avec le diluant eau on obtient

Nom_Diluant	Nom_Potion
Eau	Potion de ratatinage
Eau	Potion furoncles
Eau	Potion de ganda

Figure 15: Résultat des potions réalisables avec un diluant précis

On en conclue donc que le diluant eau est utilisé dans la recette des potions 'de ratinage', 'furoncles' et de 'de ganda'.

La liste des potions classée par température de préparation

```
-- Recherche 4 --

SELECT Potion.Nom_Potion, Potion.Temperature
FROM Potion
WHERE Valider_Potion = 'Valider'
ORDER BY Potion.Temperature;
```

Figure 16: Recherche de la liste des potions classée par température de préparation

On cherche avec cette requête à afficher les potions triées par ordre de températures nécessaire pour les faires. De plus on choisira d'ordonner les potions par ordre croissant de température. Le résultat obtenu est :

Nom_Potion	Temperature
Potion de ganda	15
Filtre de mort vivante	20
Amortentia	25
Filtre de paix	35
Potion de ratatinage	40
Potion furoncles	45
Potion de vol	50
Souffle du dragon	55
Potion de confusion	60

Figure 17: Résultat de la recherche de la liste des potions classée par température de préparation

La comparaison entre le prix de vente d'une potion et le total du prix des ingrédients (et diluant) nécessaires à sa réalisation

```
-- Recherche 5 --

SELECT Potion.Nom_Potion, Potion.Prix_Potion,
SUM((Ingredient.Prix_Ingredient * Se_compose.Quantite_Ingredient) + Diluant.Prix_Diluant) AS Prix_ingredient_et_diluant
FROM (((Diluant INNER JOIN Potion ON Diluant.ID_Diluant = Potion.ID_Diluant)
INNER JOIN Se_compose ON Potion.ID_Potion = Se_compose.ID_Potion)
INNER JOIN Ingredient ON Se_compose.ID_Ingredient =Ingredient.ID_Ingredient)
GROUP BY Potion.Nom_Potion;
```

Figure 18: Recherche pour la comparaison entre le prix de vente d'une potion et le total du prix des ingrédients

Dans cette cinquième requête on va comparer les prix de vente d'une potion et la somme des prix des ingrédients utiles à la conception de cette potion ainsi que le prix du diluant. La recherche s'effectuera sur l'ensemble des potions. On obtient après exécution :

Nom_Potion	Prix_Potion	Prix_ingredient_et_diluant
Amortentia	300.000	273.000
Filtre de mort vivante	400.000	231.000
Filtre de paix	450.000	357.000
Potion de confusion	500.000	248.000
Potion de ganda	8500.000	8029.000
Potion de ratatinage	350.000	267.000
Potion de vol	300.000	227.000
Potion furoncles	200.000	96.000
Souffle du dragon	700.000	666.000

Figure 19: Résultat de la comparaison entre le prix de vente d'une potion et le total du prix des ingrédients

Ainsi on remarque que les potions coûtent plus cher que les prix ingrédient.

Le nombre moyen d'ingrédients utilisés pour fabriquer une potion

```
-- Recherche 6 --

SELECT AVG (Quantite_ingredient) AS Moyenne_Ingredient,Potion.Nom_potion
FROM Potion INNER JOIN se_compose ON se_compose.ID_Potion= Potion.ID_Potion
GROUP BY Potion.Nom_potion;
```

Figure 20: Recherche du nombre moyen d'ingrédients utilisé pour fabriquer une potion

Dernière requête où l'objectif est de calculer la moyenne d'ingrédient nécessaire à la réalisation d'une potion. De même que la requête précédente celle-ci s'applique sur l'ensemble de potions. Sur la base de données cela permet d'avoir le résultat suivant :

Moyenne_Ingredient	Nom_potion
9.0000	Amortentia
3.0000	Filtre de mort vivante
2.0000	Filtre de paix
2.7500	Potion de confusion
128.2000	Potion de ganda
3.8000	Potion de ratatinage
6.3333	Potion de vol
2.6667	Potion furoncles
27.0000	Souffle du dragon

Figure 21: Résultat du nombre moyen d'ingrédients utilisé pour fabriquer une potion

Par exemple on remarque que pour la potion de ganda il faut 128 ingrédients pour la faire.

Ajouter une ou plusieurs nouvelles recettes

```
DELIMITER |
CREATE PROCEDURE AjoutRecette (IN IDPersonne INT, IN NomPotion VARCHAR(255),
IN Ingredient1 VARCHAR(255) ,IN QuantiteIngredient1 INT(11),
IN Ingredient2 VARCHAR(255),IN QuantiteIngredient2 INT(11),
IN Ingredient3 VARCHAR(255),IN QuantiteIngredient3 INT(11),
IN Ingredient4 VARCHAR(255),IN QuantiteIngredient4 INT(11),
IN Ingredient5 VARCHAR(255),IN QuantiteIngredient5 INT(11),
IN PrixPotion DECIMAL (15,3),IN NomDiluant VARCHAR(255),IN Temperatur INT(255))
BEGIN
INSERT INTO Potion (`Nom_Potion`,`Prix_Potion`, `Temperature`, `Valider_Potion`,`ID_Personne`,`ID_Diluant`)
VALUES  ( NomPotion, PrixPotion, Temperatur, 'Attente',IDPersonne,(SELECT ID_Diluant FROM Diluant WHERE Nom_Diluant = NomDiluant));
SET @idpotion = (SELECT ID_Potion FROM Potion ORDER BY ID_Potion DESC LIMIT 1);
SET @iding1 = (SELECT ID_Ingredient FROM Ingredient WHERE Nom_Ingredient = Ingredient1);
SET @iding2 = (SELECT ID_Ingredient FROM Ingredient WHERE Nom_Ingredient = Ingredient2);
SET @iding3 = (SELECT ID_Ingredient FROM Ingredient WHERE Nom_Ingredient = Ingredient3);
SET @iding4 = (SELECT ID_Ingredient FROM Ingredient WHERE Nom_Ingredient = Ingredient4);
SET @iding5 = (SELECT ID_Ingredient FROM Ingredient WHERE Nom_Ingredient = Ingredient5);
INSERT INTO Se_compose (`Quantite_Ingredient`, `ID_Potion`, `ID_Ingredient`)
VALUES ( QuantiteIngredient1, @idpotion, @iding1);
DELETE FROM `Se_compose`
WHERE ID_Ingredient = 22;
INSERT INTO Se_compose (`Quantite_Ingredient`, `ID_Potion`, `ID_Ingredient`)
VALUES ( QuantiteIngredient2, @idpotion, @iding2);
DELETE FROM `Se_compose`
WHERE ID_Ingredient = 22;
INSERT INTO Se_compose (`Quantite_Ingredient`, `ID_Potion`, `ID_Ingredient`)
VALUES ( QuantiteIngredient3, @idpotion, @iding3);
DELETE FROM `Se_compose`
WHERE ID_Ingredient = 22;
INSERT INTO Se_compose (`Quantite_Ingredient`, `ID_Potion`, `ID_Ingredient`)
VALUES ( QuantiteIngredient4, @idpotion, @iding4);
DELETE FROM `Se_compose`
WHERE ID_Ingredient = 22;
INSERT INTO Se_compose (`Quantite_Ingredient`, `ID_Potion`, `ID_Ingredient`)
VALUES ( QuantiteIngredient5, @idpotion, @iding5);
DELETE FROM `Se_compose`
WHERE ID_Ingredient = 22;
END |
DELIMITER ;
```

Figure 22: Procédure pour ajouter une ou plusieurs nouvelles recettes

Cette longue requête permet d'ajouter des recettes. Pour faire cela nous passons par différentes variables utilisateur symbolisées par @.

Ces différentes variables nous aident pour récupérer l'ID de l'ingrédient, plus facile à intégrer ensuite dans la table.

Valider une recette qui a été acceptée

```
UPDATE Potion
SET Valider_Potion = "Valider"
WHERE Valider_Potion = "Attente";
```

Figure 23:Requête pour valider une recette qui a été acceptée

Cette requête change juste le champ à valider si avant le champ est à attente

Supprimer une recette qui n'a pas été retenue

```
DELETE SC
FROM  Potion P INNER JOIN Se_Compose SC ON SC.ID_Potion = P.ID_Potion
WHERE P.Valider_Potion =  "Refuser";

DELETE FROM Potion
WHERE Potion.Valider_Potion =  "Refuser";
```

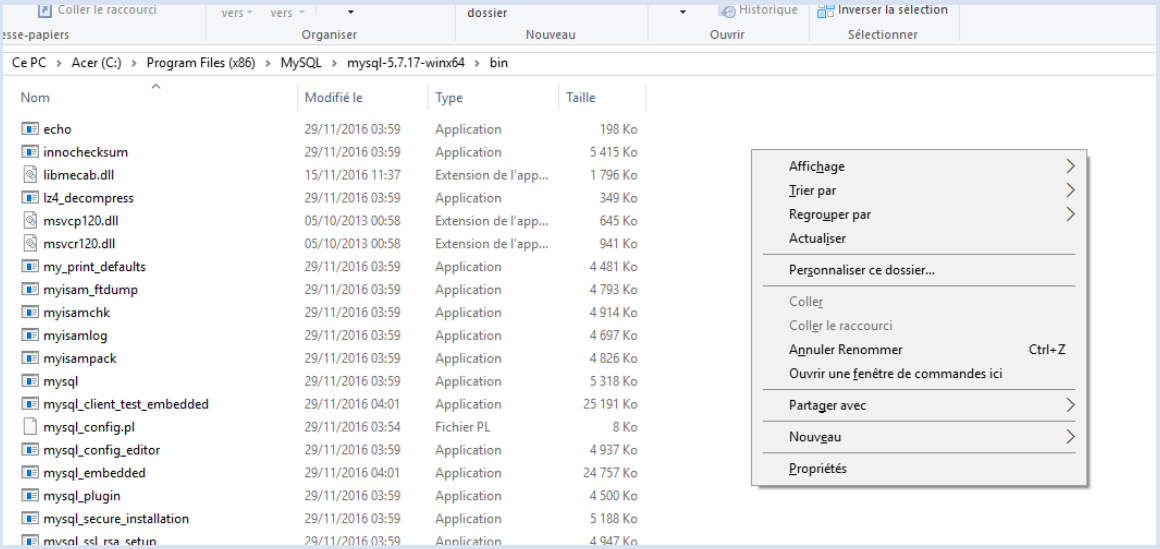
Figure 24: Requêtes pour supprimer une recette qui n'a pas été retenue

En deux temps. D'abord il est essentiel de supprimer dans la table Se_Compose les différents ingrédients qui compose la potion. Cela évite les erreurs de clé étrangères. Ensuite on peut passer à la suppression de la potion.

C- Les requêtes de sauvegarde

La sauvegarde apporte une sécurité pour la protection des données stockées dans la base de données

Pour ce faire, il faut télécharger MySQL server lancé une commande



Il faut ouvrir une commande dans le dossier C:\Program Files (x86)\MySQL\mysql-5.7.17-winx64\bin.

Une fois la commande ouverte il faut rentrer la commande suivante :

Mysqldump –u [Nom User] –h « Ip de l'user ou localhost » –p «Nom de la base » « Nom de la table » > cheminspécifié\nomdufichierasave.sql

```
CREATE VIEW CommandeNonrecu
AS
SELECT Date_de_commande , No_Commande , Statut_Progression , Nom_client
FROM Commande , Progression , Client
WHERE Progression.ID_Progression = Commande.ID_Progression
AND Commande.ID_client = Client.ID_client
AND Commande.ID_Progression = 2 ;
-- Commande non recu
Mysqldump -u root -h localhost -p  ProjetGiselle CommandeNonrecu > C:\Users\Adurtis\Desktop\sauvegarde\CommandeNonrecu.sql
-- Recette Potion
Mysqldump -u root -h localhost -p ProjetGiselle Potion Diluant se_compose Ingredient > C:\Users\Adurtis\Desktop\sauvegarde\RecettePotion.sql
--Ingrédient fournisseur
Mysqldump -u root -h localhost -p ProjetGiselle  Ingredient Donne_Ingredient Fournisseur > C:\Users\Adurtis\Desktop\sauvegarde\IngredientFournisseur.sql

CREATE USER 'Natsu'@'localhost' IDENTIFIED BY 'Projet_BDD';

GRANT SELECT ON CommandeNonrecu TO Natsu IDENTIFIED BY 'Projet_BDD';

Mysql -u root -h localhost -p ProjetGiselle < C:\Users\Adurtis\Desktop\sauvegarde\CommandeNonrecu.sql
```

Conclusion

- Problèmes rencontrés

Durant ce projet nous avons rencontré un certain nombre de problèmes. Sur les débuts projets nous avions oubliés beaucoup de données important. Ainsi nous avons dû recommencer un grand nombre de fois le dictionnaire de données, la matrice de dépendances ainsi que le MCD. Ce dernier nous a également posé problème. Alors que nous pensions avoir fini il s'est avéré que ce dernier était faux et qu'il manquait un certain nombre d'associations importantes à la réalisation des requêtes

En deuxième phase de projet, sur les différentes requêtes, certaines d'entre elles nous ont retardés. Ayant besoin d'une requête pour en faire une autre il était crucial de trouver une solution.

- Bilan général

Dans l'ensemble nous avons bien réalisé ce projet, en effet nous avons réussi à faire toute les demandes et dans les temps. Nous avons essayé de répondre au mieux aux besoins, et de créer une base de données fonctionnelle. Nous avons aussi pu consolider nos connaissances en SQL lors de création de requête et de procédures assez complexes.

- Bilan Individuel

Paul : Ce projet était intéressant, j'ai pu consolider mes connaissances en SQL lors de la création des requêtes et des procédures assez complexes. Je trouve que j'ai bien accompli mon travail de chef de projet puisque l'ensemble des objectifs ont été atteint par l'équipe.

Arnaud : J'ai trouvé que ce projet, bien que flou, a permis de revoir l'intégralité de l'UE et a permis de découvrir de nouvelle fonctionnalité. Le groupe était volontaire et à l'écoute des uns des autres.

Jean clément : Pour ce projet BDD, toutes les compétences acquises lors des prosits ont été regroupé dans ce dernier. De plus j'ai appris de nouvelles choses, tels que les sauvegardes et la restauration. En ce qui concerne le groupe, il y a eu une bonne coordination et une bonne écoute malgré les problèmes rencontres, dans l'ensemble le projet fut bien.

Maxence : Ce projet était un bon projet dans l'ensemble, certaines difficultés ont été présentes et ont été relevées. J'ai pu m'améliorer énormément en base de données car j'avais pas mal de difficultés dans ce domaine. L'ensemble du groupe était impliqué et productif et complétement différent des anciens groupes précédents ce qui a permis de travailler différemment de d'habitude.