

# Projet De Conception des Systèmes d'Information

2014 - 2015

Ersagun Yalcintepe,  
Arthur Mourey,  
Matthieu Sahuguet,  
Tom Verhoof



L3 Miage  
Université de Lorraine  
13 rue Michel Ney  
54037 Nancy Cedex

Département Miage



# Projet De Conception des Systèmes d'Informations

Ersagun Yalcintepe,  
Arthur Mourey,  
Matthieu Sahuguet,  
Tom Verhoof

2014 - 2015

# Table des matières :

## Contenu

Introduction.....	4
Liste des Fonctionnalités du Site.....	5
Modèles Merise .....	7
Modèle Conceptuel de Données .....	8
Dictionnaire de données.....	9
Modèle Conceptuel de Traitement .....	9
Modèle Organisationnel de Traitement .....	14
Modèle Physique de Données.....	18
Modèle Physique de Traitement.....	19
Réalisation de la base de données.....	23
Création de la base de données MYSQL à partir du MPD .....	23
Modèle de la base de données .....	28
Créations des Triggers & Procédures à partir des MCT et MOT .....	28
Insertion de contenu dans la base de données .....	30
Réalisation du site web.....	31
Création des modèles et vérifications des fonctions d'insertion, modification et suppression sur les tables.....	31
Screenshot du site.....	33
La répartition des tâches et Organisation .....	34
Diagramme de Gantt .....	34
Organisation du groupe.....	34
Conclusion.....	35

# Introduction

Dans ce projet nous allons réaliser la conception d'un système d'informations d'une application pour un marchand spécialiste du Drive. Afin de créer l'application, nous allons utiliser les modèles Merise pour concevoir le SI.

Nous sommes un groupe de 4 étudiants de 3<sup>ème</sup> année de Licence Informatique MIAGE :

- YALCINTEPE Ersagun,
- MOUREY Arthur
- SAHUGUET Matthieu
- VERHOOF Tom

Comme il s'agit d'une application web, nous avons décidé d'utiliser les outils web suivants :

HTML 5, CSS, JavaScript (client) et PHP (serveur).

## Liste des Fonctionnalités du Site

### Les fonctionnalités attendues comprennent :

- La possibilité pour tout utilisateur de parcourir les différents produits
- La possibilité pour l'utilisateur de filtrer/rechercher les produits par nom, par catégorie et par prix.
- La possibilité pour tout utilisateur de s'inscrire, de se connecter et de se déconnecter.
- Chaque client peut voir son historique des produits achetés et des commandes effectuées
- L'acheteur peut remplir son panier en sélectionnant la quantité de produit désirée, ou vider un produit ou tout le contenu de son panier.
- Si le panier n'est pas vide, le client verra à tout moment sur la page le contenu de son panier, le nombre de produit à l'intérieur, et le montant total.
- Si le panier n'est pas vide, le client peut valider sa commande, en choisissant une date et un horaire pour retirer sa commande.
- Un rappel mail sera envoyé au client une heure avant la date où il devra récupérer sa commande.
- Le client peut voir la liste des offres promotionnelles et des réductions valables actuellement pour lui.

- L'administrateur pourra voir un bilan journalier, hebdomadaire ou mensuel des ventes.

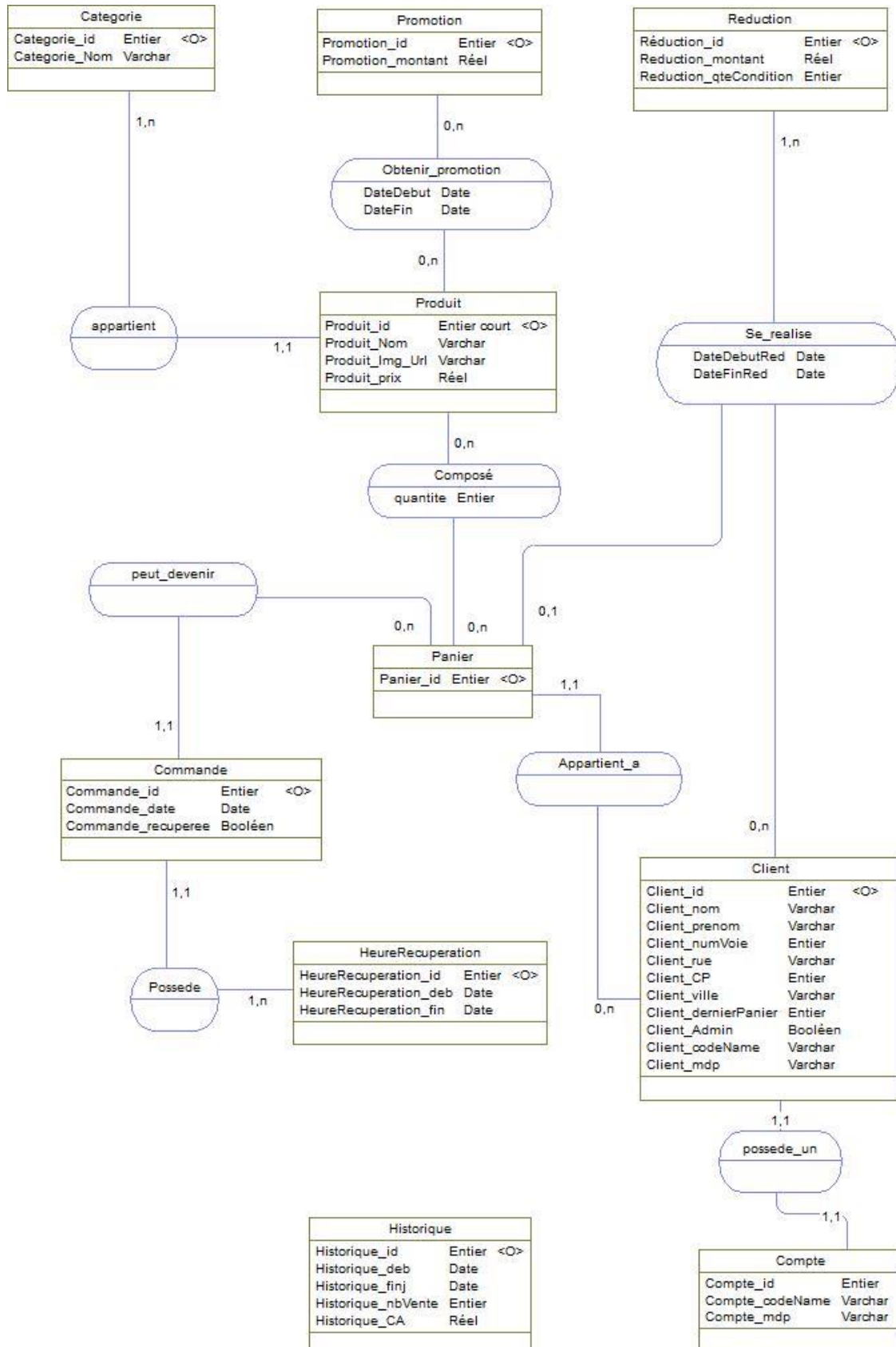
### **Les fonctionnalités attendues ne comprennent pas :**

- La gestion des stocks des produits, comme indiqué dans l'énoncé.
- Le paiement réel, et tout ce qui est gestion des paiements, factures et retards de paiement.
- La possibilité pour le client de modifier ou d'annuler sa commande une fois qu'elle a été enregistrée, ou de redemander une date et un horaire s'il rate la livraison.

# Modèles Merise



# Modèle Conceptuel de Données



# Dictionnaire de données

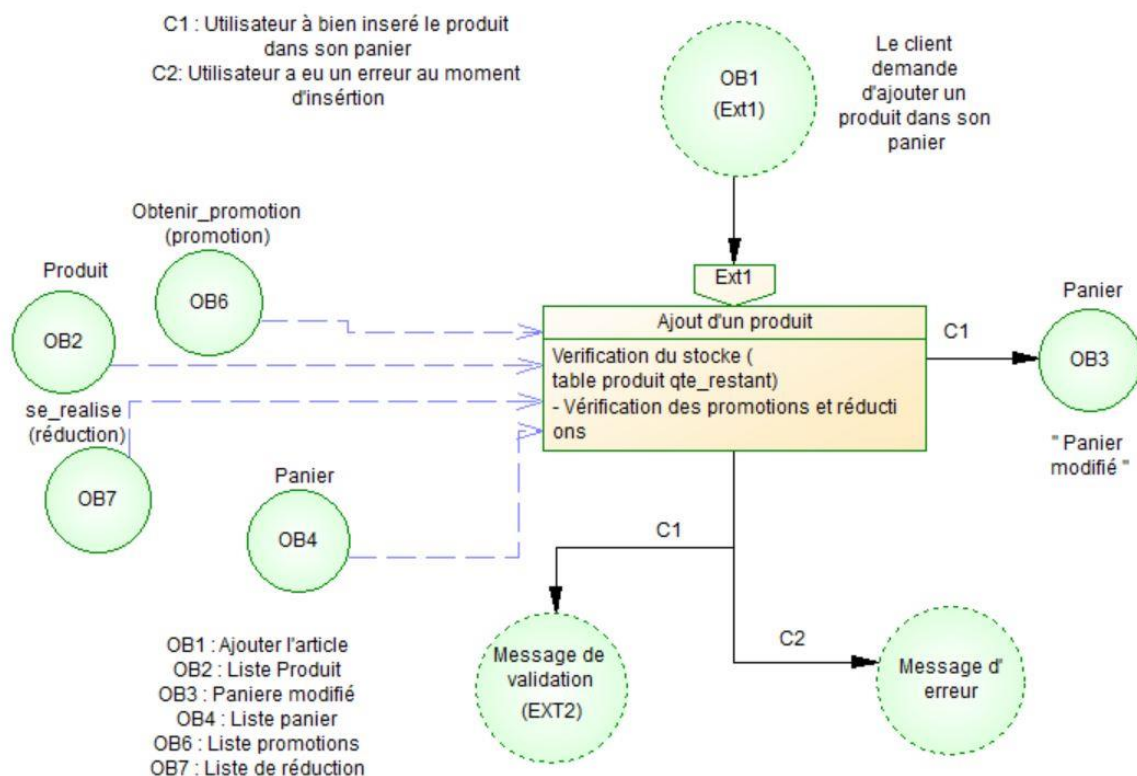
Veillez trouver notre dictionnaire de données dans le dossier CSI en format oxpfs.

## Modèle Conceptuel de Traitement

### Ajout d'un produit :

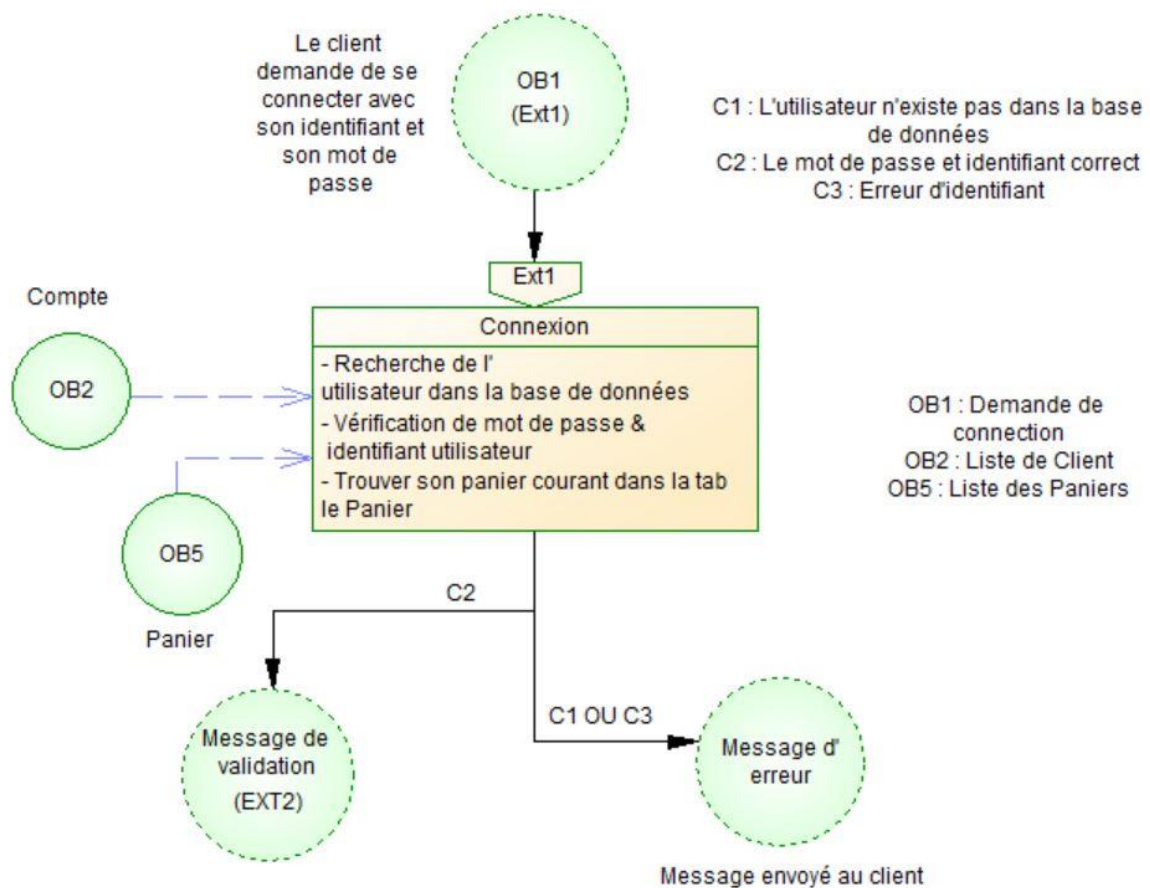
Ce MCT permet de modéliser l'ajout d'un produit, par l'utilisateur, dans son panier. Pour ce faire, il est nécessaire d'accéder au produit, aux potentiels réductions ou promotions, à la commande et au panier.

L'opération qui s'effectue par la suite vérifie plusieurs facteurs, tels que la présence du produit dans les stocks, les promotions ou réductions qui lui sont affectées, et modifie la commande si l'ajout peut être réalisé. D'autre part, un message ou signal avertie le client dans ce cas.



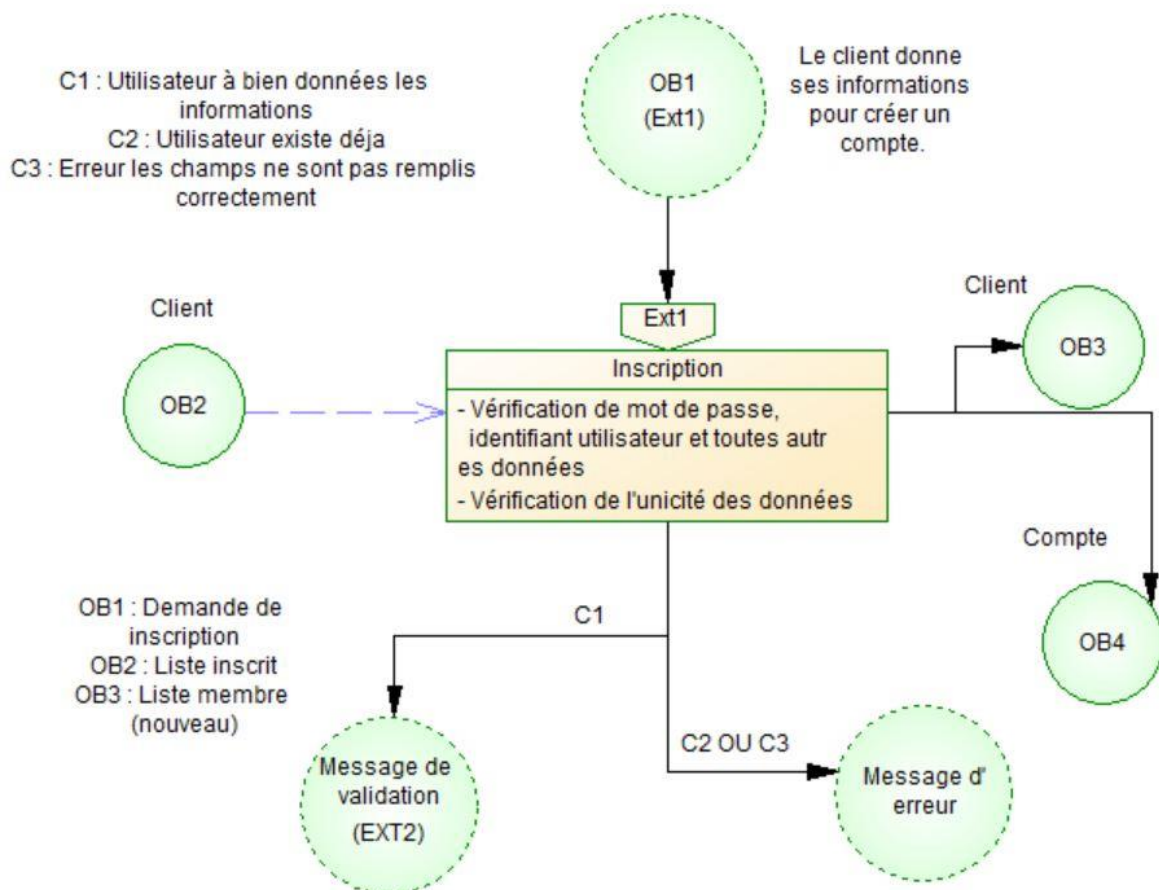
## Connexion :

Au moment où l'utilisateur désire se connecter, on va vérifier, en cherchant dans les tables, si cet utilisateur existe dans la base de données. On accède donc à la table compte et panier, afin de charger ses paramètres personnels et son panier si il parvient à se connecter. Si c'est le cas, on lui envoie un message de bienvenue, sinon une erreur lui informant que son identifiant ou mot de passe n'est pas valide.



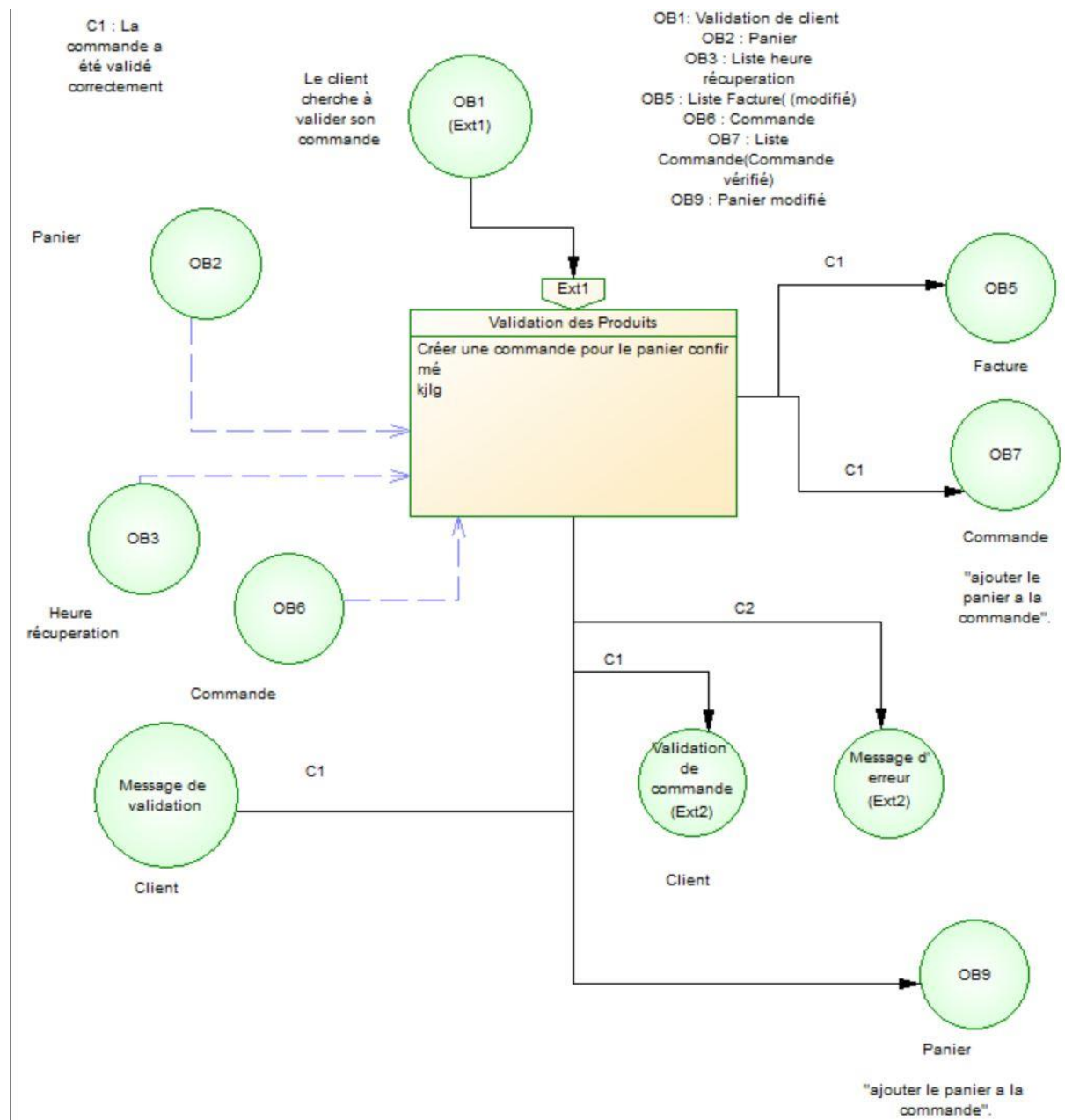
## Inscription :

Lorsque l'utilisateur fait une demande de création de compte. On accède à la base de données pour savoir si cet utilisateur existe déjà dans la table client. On vérifie également, dans l'opération, si les informations entrées sont valides ou non. Si l'opération s'est effectuée avec succès, alors on crée un nouveau client avec son compte et l'on informe l'utilisateur que son inscription s'est bien déroulée. Dans le cas contraire, on lui envoie un message d'erreur.



## Validation des produits :

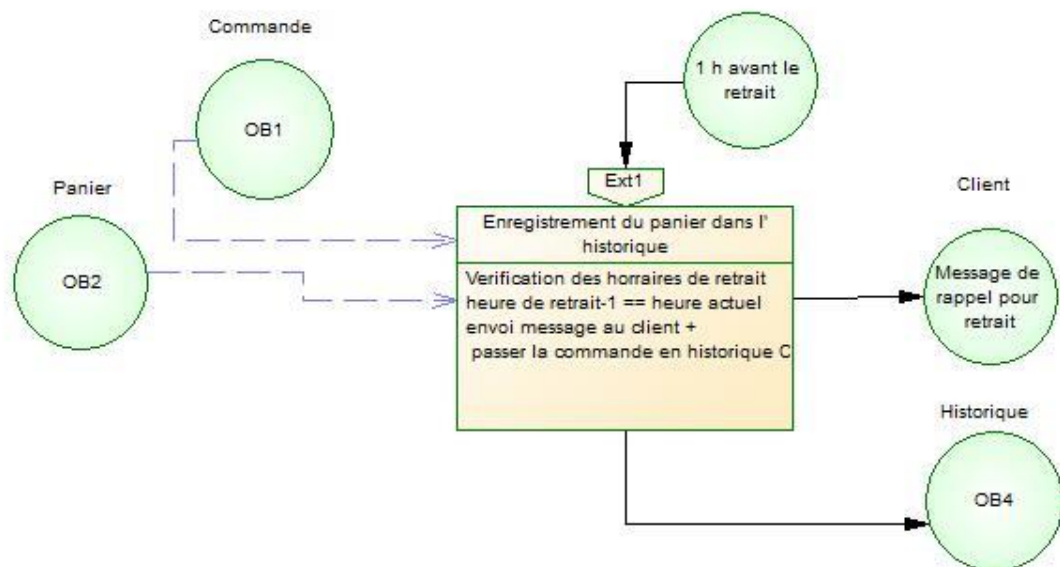
Au moment où l'utilisateur valide sa commande, on vérifie la validité de la commande et de l'horaire de récupération en magasin. S'ils sont cohérents, alors on modifie l'état du panier, qui devient confirmé, et on génère la facture correspondante. Ainsi la commande est validée, et un message de confirmation est envoyé à l'utilisateur.



## Envois message :

Une heure avant la récupération de la commande en magasin, on récupère les coordonnées du client et sa commande pour lui envoyer un message lui indiquant qu'il doit récupérer ses produits en magasin.

Procédure qui repete toutes les heures ou 2 heures.



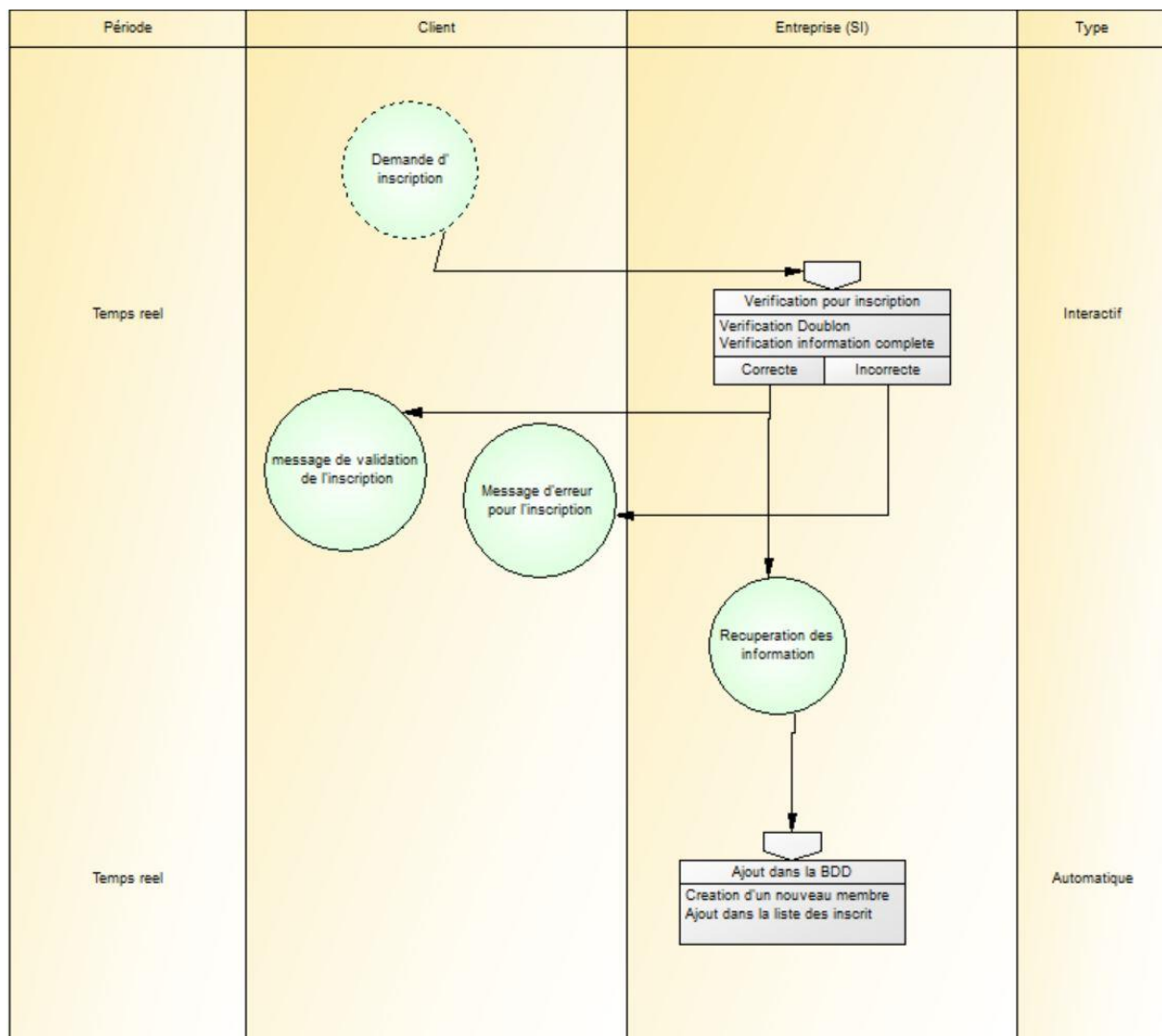
# Modèle Organisationnel de Traitement

Le modèle organisationnel des traitements s'attache à décrire les propriétés des traitements non traitées par le modèle conceptuel des données, c'est-à-dire :

- le temps
- les ressources
- le lieu

Le modèle organisationnel des traitements consiste donc à représenter le modèle conceptuel des traitements dans un tableau dont les colonnes sont la durée, le lieu, les responsables et ressources nécessaires à une action.

## Inscription :



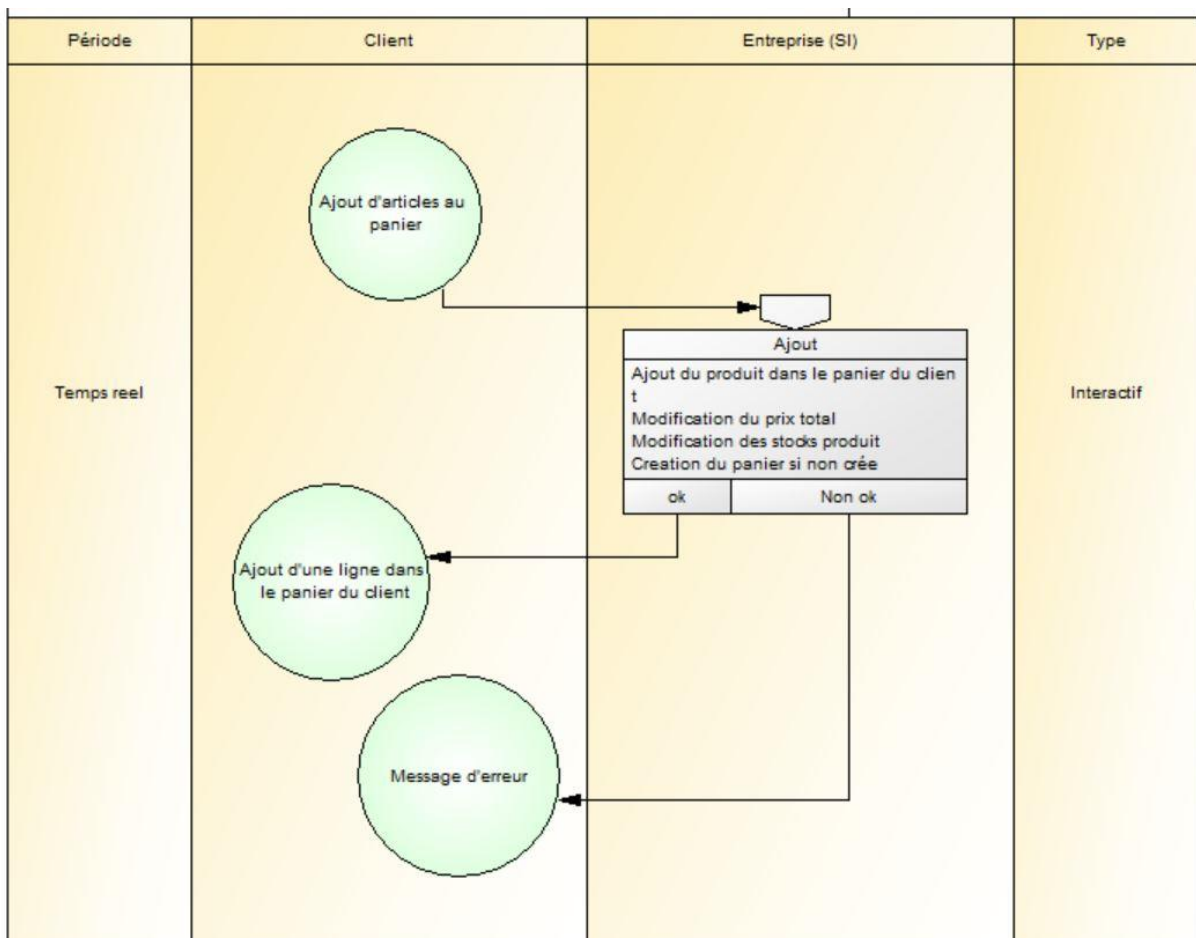
On a ajouté la période pour chaque événement, ici on a du temps réel pour la vérification des données d'inscription et pour l'ajout dans la base de données.



On ajoute aussi les acteurs impactés, ici le client qui est externe et l'entreprise (le SI), il s'agit des deux acteurs qui sont visé par ces événements.

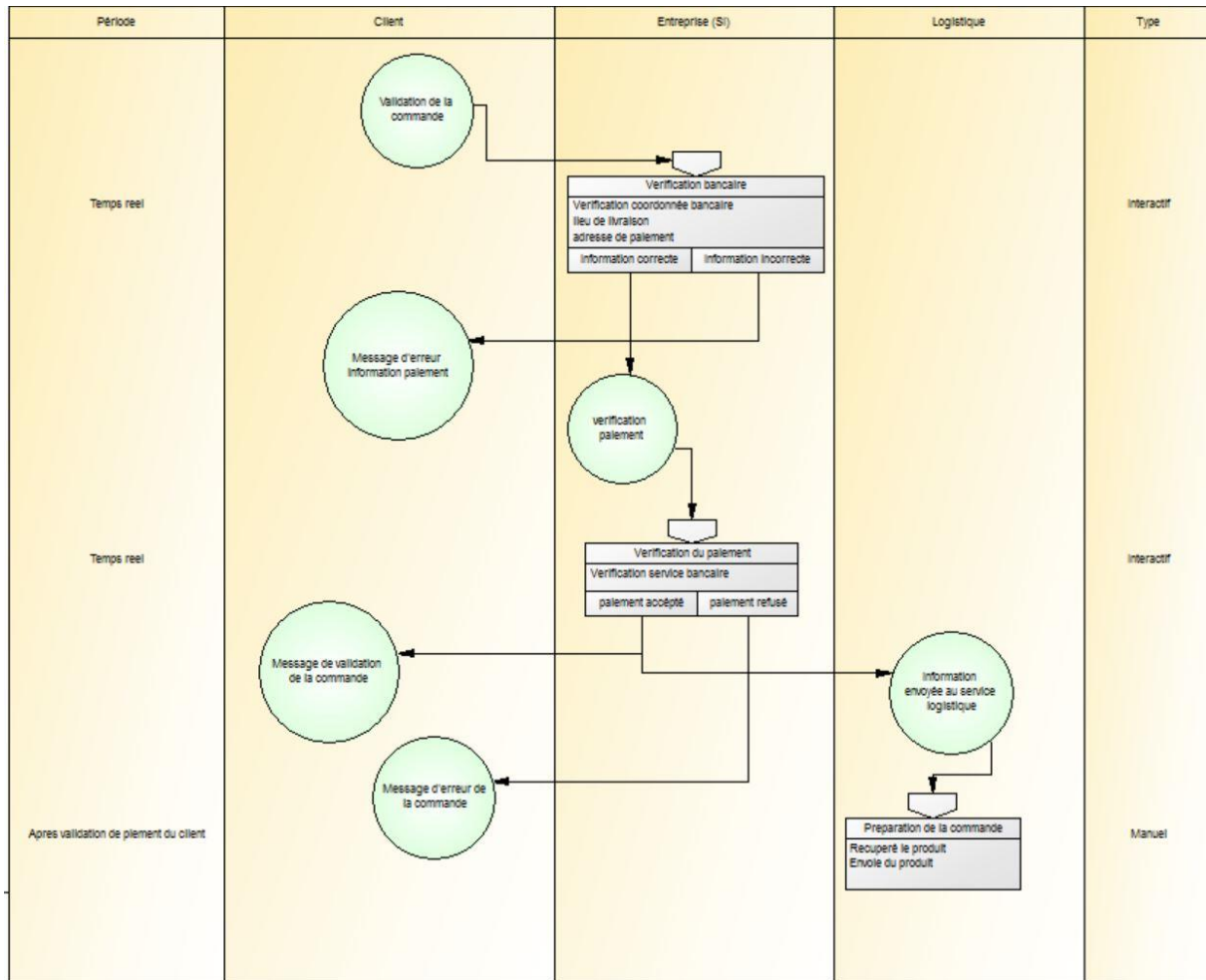
On a ajouté le type, il s'agit de savoir si il y a une interaction, une action automatique ou manuel, différé ou autre entre les acteurs pour l'événement : Dans ce cas il y a une interaction pour la vérification des données car le client entre des informations que l'on vérifie et on lui renvoie un message d'erreur ou de validation et une action automatique pour l'ajout dans la BDD quand la vérification est validé

## Ajout panier :

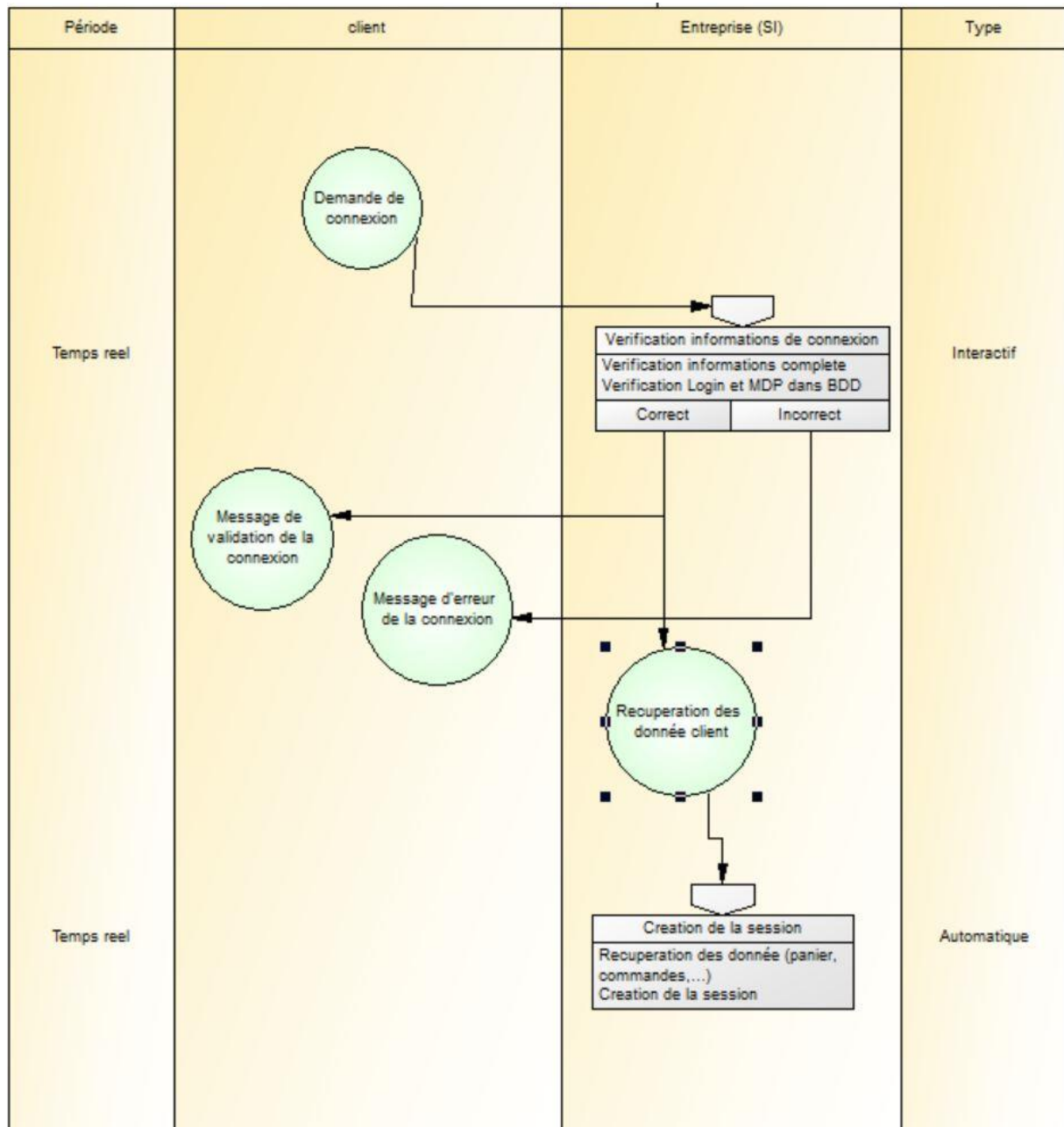




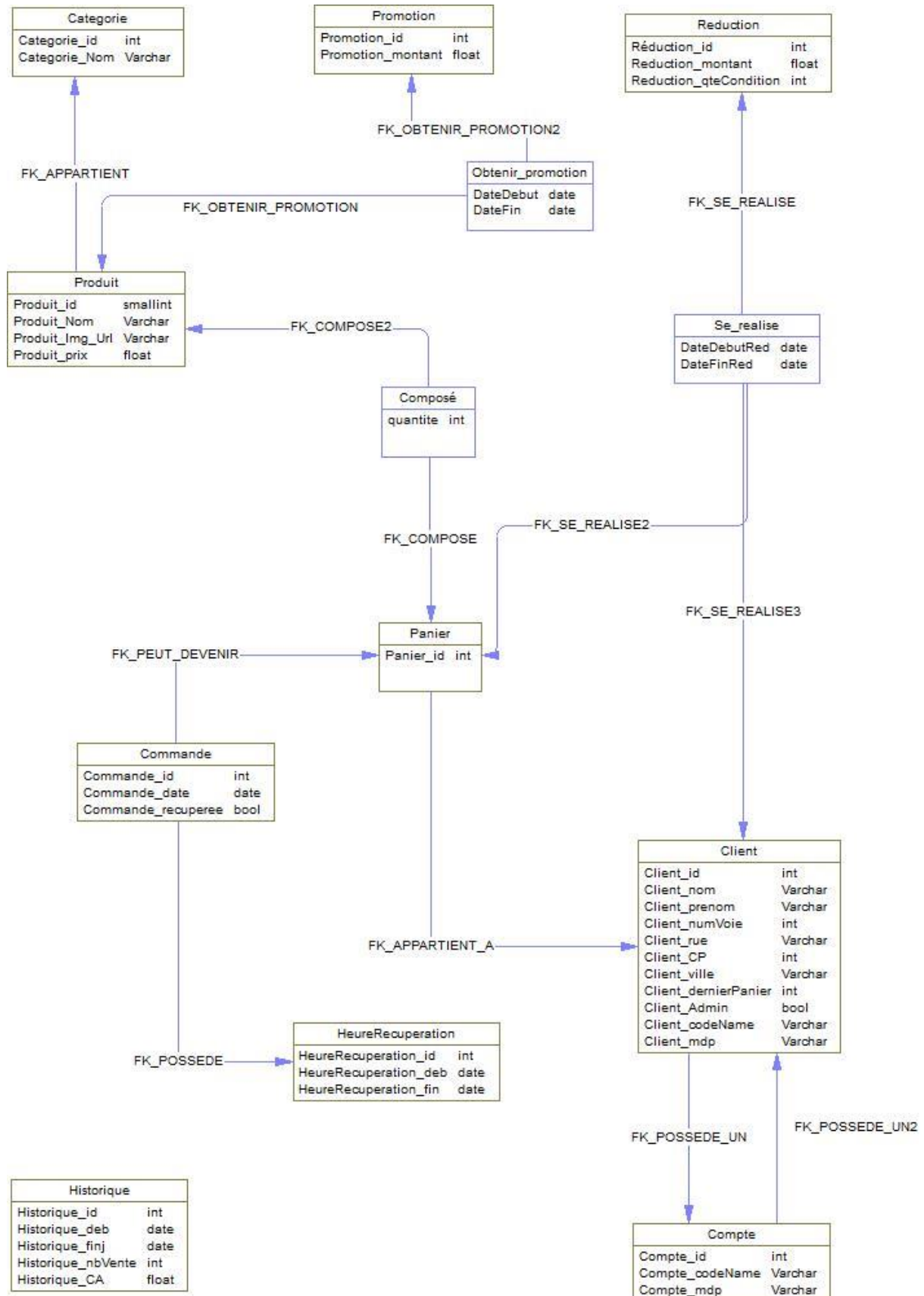
## Validation de commande



## Connexion



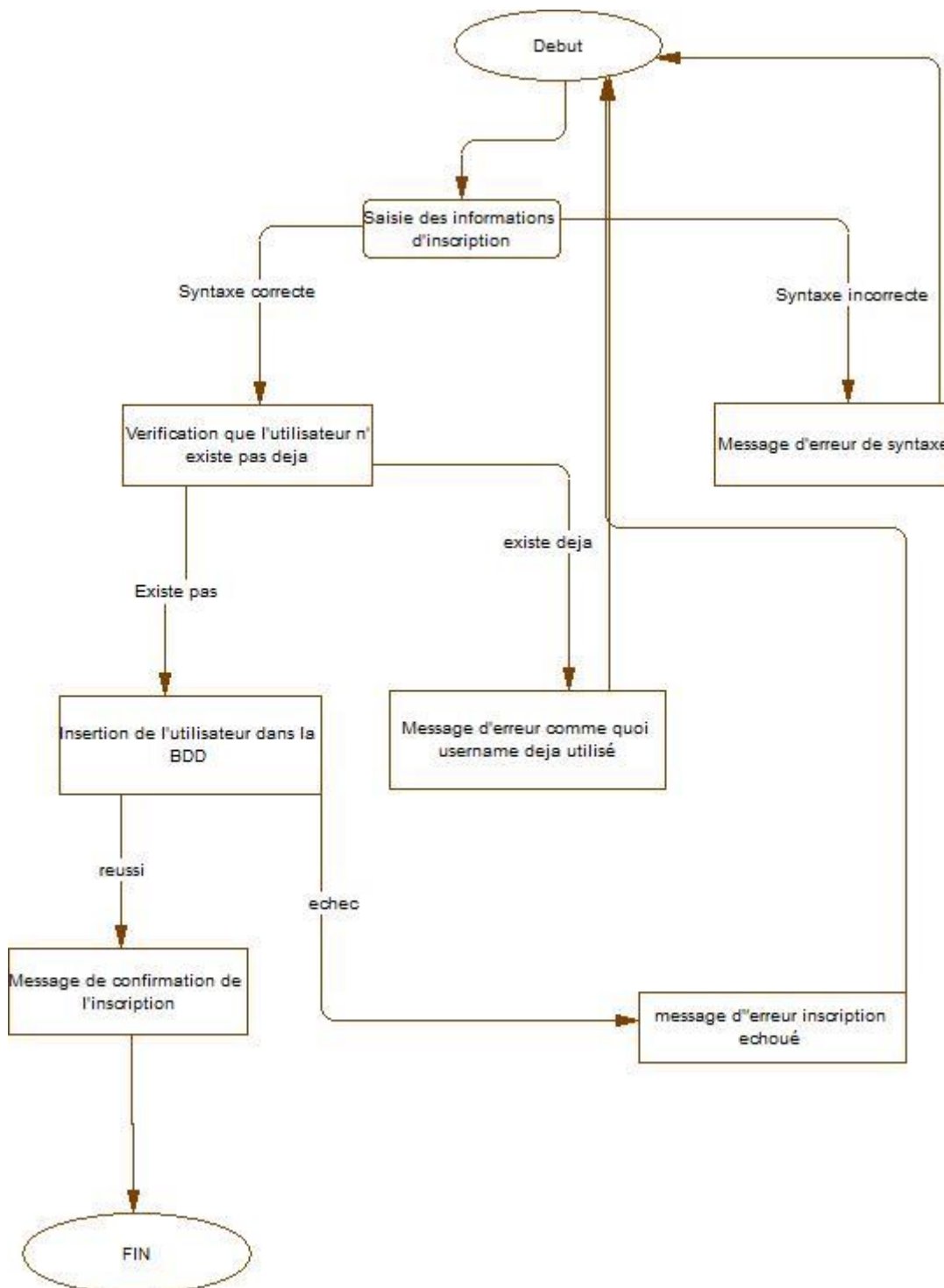
# Modèle Physique de Données



## Modèle Physique de Traitement

Pour les traitements, il s'agit du dernier stade avant la programmation. Le modèle opérationnel des traitements consiste en la définition détaillée des traitements qui sont exprimés en langage algorithmique ou en utilisant le formalisme graphique correspondant.

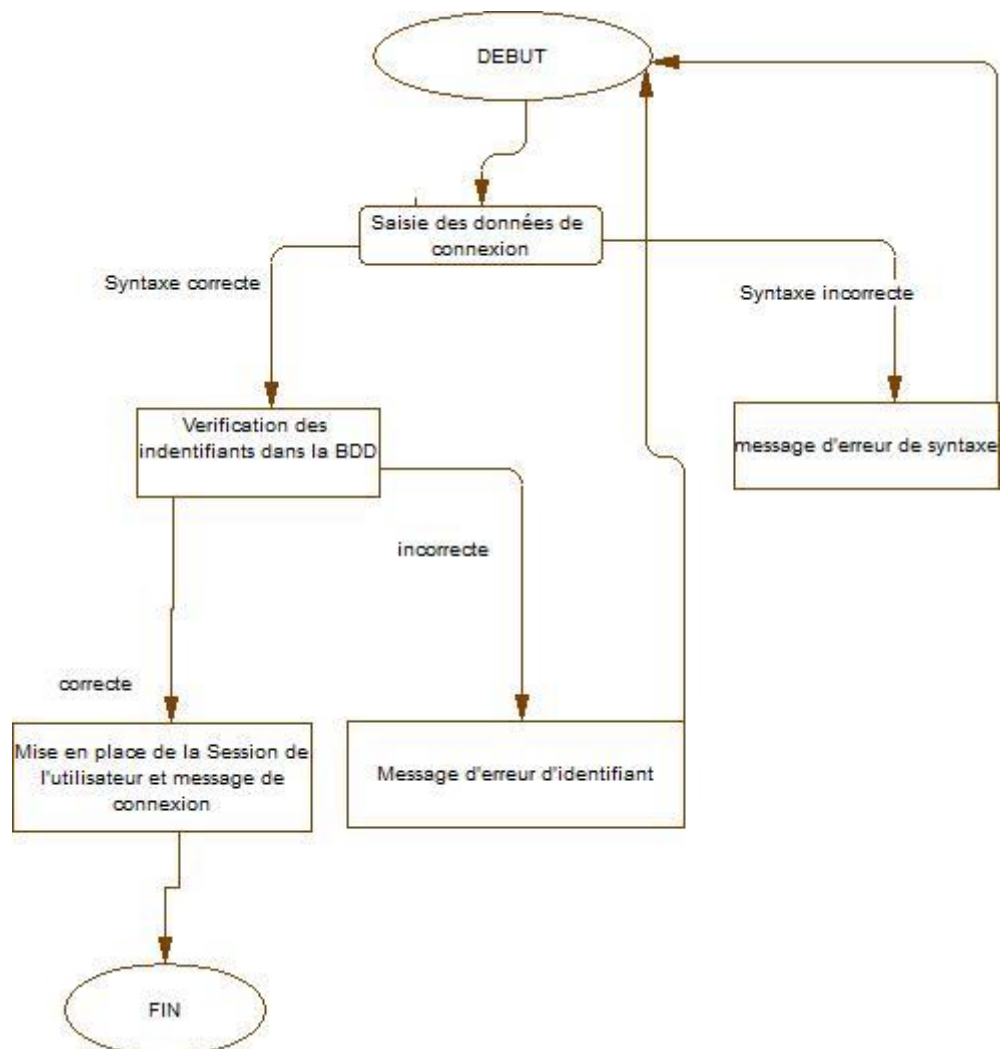
### Inscription :



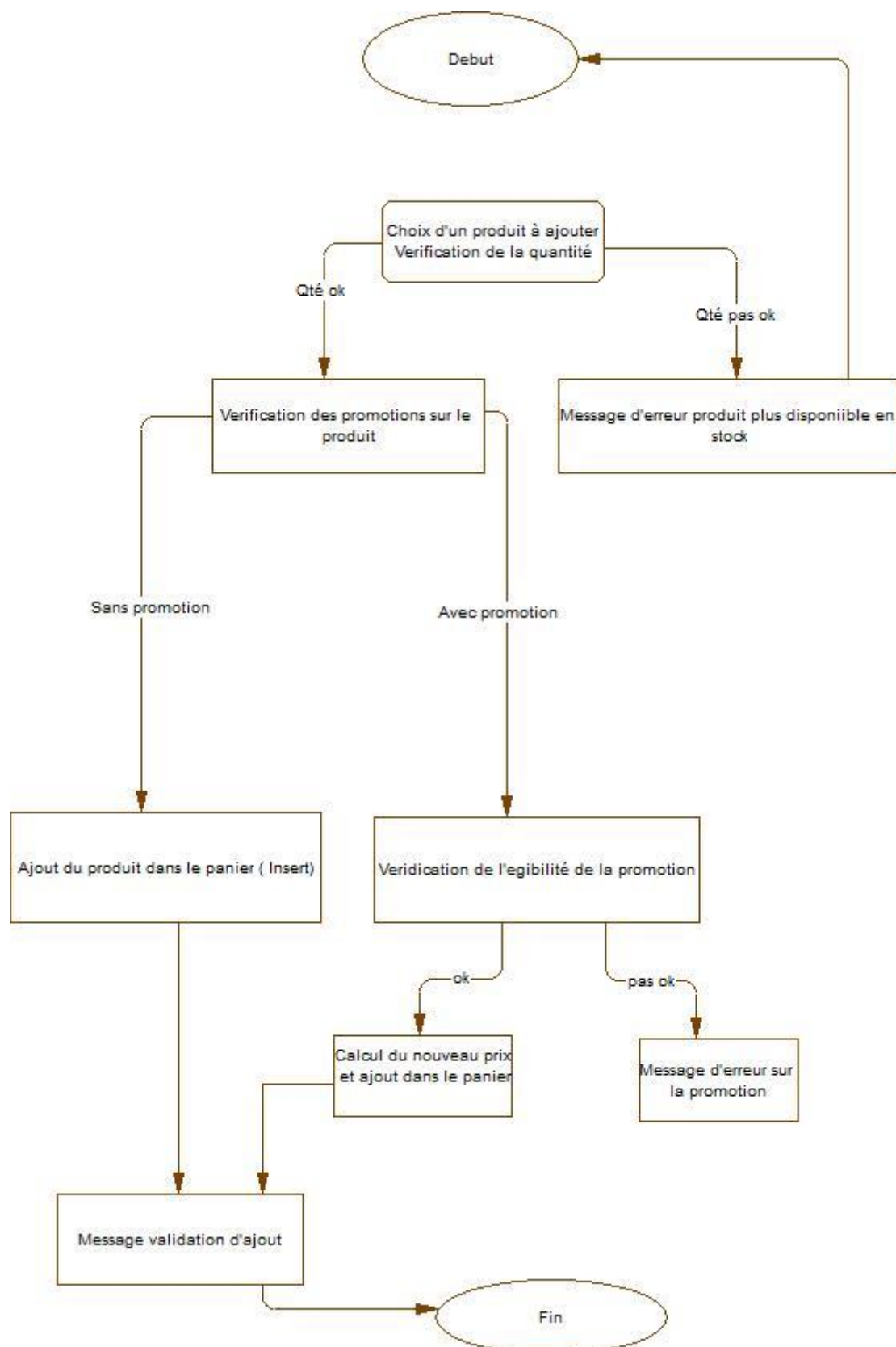
Le MPT permet de ce visualisé de façon plus claire à quoi va ressembler le code, l'algorithme pour effectuer l'événement ici de l'inscription sous forme de schéma.

On peut voir que au départ il y a la vérification de la syntaxe sur les informations reçu de l'utilisateur , si la syntaxe est incorrecte on envoie un message d'erreur de syntaxe et on retourne au début , si la syntaxe est correcte on passe à l'étape suivante qui est la vérification de doublon pour l'identifiant , si il existe déjà alors message d'erreur d'identifiant déjà utilisé et retour au début sinon on insère dans la base de donnée l'utilisateur si il y a un échec on prévient l'utilisateur et il retourne au début sinon on lui envoie un message de confirmation de l'inscription.

## Ajout panier :



## Connexion :



# Réalisation de la base de données

## Création de la base de données MYSQL à partir du MPD

### Script de création des tables :

```
--
-- Base de données : `csi_projet`
--

-----

--
-- Structure de la table `categorie`
--

CREATE TABLE IF NOT EXISTS `categorie` (
  `Categorie_Id` int(11) NOT NULL AUTO_INCREMENT,
  `Categorie_Nom` varchar(500) NOT NULL,
  PRIMARY KEY (`Categorie_Id`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1 AUTO_INCREMENT=1 ;

-----

--
-- Structure de la table `client`
--

CREATE TABLE IF NOT EXISTS `client` (
  `Client_Id` int(11) NOT NULL AUTO_INCREMENT,
  `Client_Nom` varchar(200) NOT NULL,
  `Client_Prenom` varchar(200) NOT NULL,
  `Client_Numvoie` int(11) NOT NULL,
  `Client_Rue` varchar(200) NOT NULL,
  `Client_Cp` int(11) NOT NULL,
  `Client_Ville` varchar(200) NOT NULL,
  `Client_Dernierpanier` int(11) NOT NULL,
  `Client_Admin` tinyint(1) NOT NULL,
  `Client_Codename` varchar(200) NOT NULL,
  `Client_mdp` varchar(50) NOT NULL,
  `Compte_Id` int(11) NOT NULL,
  PRIMARY KEY (`Client_Id`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1 AUTO_INCREMENT=1 ;
```



```

--
-- RELATIONS POUR LA TABLE `client`:
-- `Compte_Id`
--   `compte` -> `Compte_Id`
--
-- -----
--
-- Structure de la table `commande`
--
CREATE TABLE IF NOT EXISTS `commande` (
  `Commande_Id` int(11) NOT NULL AUTO_INCREMENT,
  `Commande_date` date NOT NULL,
  `Comande_recuperee` tinyint(1) NOT NULL,
  `HeureRecuperation_id` int(11) NOT NULL,
  `Panier_Id` int(11) NOT NULL,
  PRIMARY KEY (`Commande_Id`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1 AUTO_INCREMENT=1 ;

--
-- RELATIONS POUR LA TABLE `commande`:
-- `HeureRecuperation_id`
--   `heurerecuperation` -> `HeureRecuperation_id`
-- `Panier_Id`
--   `panier` -> `Panier_Id`
--
-- -----
--
-- Structure de la table `compose`
--
CREATE TABLE IF NOT EXISTS `compose` (
  `Produit_Id` int(11) NOT NULL,
  `Panier_Id` int(11) NOT NULL,
  `Quantite` int(11) NOT NULL,
  PRIMARY KEY (`Produit_Id`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1;

--
-- RELATIONS POUR LA TABLE `compose`:
-- `Panier_Id`
--   `panier` -> `Panier_Id`
-- `Produit_Id`
--   `produit` -> `Produit_Id`
--

```

-- -----

--  
-- Structure de la table `compte`  
--

```
CREATE TABLE IF NOT EXISTS `compte` (  
  `Compte_Id` int(11) NOT NULL AUTO_INCREMENT,  
  `Compte_Codename` varchar(200) NOT NULL,  
  `Compte_mdp` varchar(50) NOT NULL,  
  PRIMARY KEY (`Compte_Id`)  
) ENGINE=MyISAM DEFAULT CHARSET=latin1 AUTO_INCREMENT=1 ;
```

-- -----

--  
-- Structure de la table `heurerecuperation`  
--

```
CREATE TABLE IF NOT EXISTS `heurerecuperation` (  
  `HeureRecuperation_id` int(11) NOT NULL AUTO_INCREMENT,  
  `HeureRecuperation_Deb` date NOT NULL,  
  `HeureRecuperation_Fin` date NOT NULL,  
  PRIMARY KEY (`HeureRecuperation_id`)  
) ENGINE=MyISAM DEFAULT CHARSET=latin1 AUTO_INCREMENT=1 ;
```

-- -----

--  
-- Structure de la table `historique`  
--

```
CREATE TABLE IF NOT EXISTS `historique` (  
  `Historique_Id` int(11) NOT NULL AUTO_INCREMENT,  
  `Historique_Deb` date NOT NULL,  
  `Historique_Fin` date NOT NULL,  
  `Historique_Nbvente` int(11) NOT NULL,  
  `Historique_Ca` double NOT NULL,  
  PRIMARY KEY (`Historique_Id`)  
) ENGINE=MyISAM DEFAULT CHARSET=latin1 AUTO_INCREMENT=1 ;
```

-- -----

--  
-- Structure de la table `obtenir\_promotion`  
--

```
CREATE TABLE IF NOT EXISTS `obtenir_promotion` (  
  `Promotion_Id` int(11) NOT NULL,
```

```
`Produit_Id` int(11) NOT NULL,  
`Date_Debut` date NOT NULL,  
`Date_Fin` date NOT NULL,  
PRIMARY KEY (`Promotion_Id`,`Produit_Id`)  
) ENGINE=MyISAM DEFAULT CHARSET=latin1;
```

```
--  
-- RELATIONS POUR LA TABLE `obtenir_promotion`:  
-- `Produit_Id`  
--   `produit` -> `Produit_Id`  
-- `Promotion_Id`  
--   `promotion` -> `Promotion_Id`  
--
```

```
-- -----
```

```
--  
-- Structure de la table `panier`  
--
```

```
CREATE TABLE IF NOT EXISTS `panier` (  
  `Panier_Id` int(11) NOT NULL AUTO_INCREMENT,  
  `client_id` int(11) NOT NULL,  
  `DateDebutRed` date NOT NULL,  
  `DateFinRed` date NOT NULL,  
  `Reduction_Id` int(11) NOT NULL,  
  PRIMARY KEY (`Panier_Id`)  
) ENGINE=MyISAM DEFAULT CHARSET=latin1 AUTO_INCREMENT=1 ;
```

```
--  
-- RELATIONS POUR LA TABLE `panier`:  
-- `Reduction_Id`  
--   `reduction` -> `Reduction_id`  
-- `client_id`  
--   `client` -> `Client_Id`  
--
```

```
-- -----
```

```
--  
-- Structure de la table `produit`  
--
```

```
CREATE TABLE IF NOT EXISTS `produit` (  
  `Produit_Id` int(11) NOT NULL AUTO_INCREMENT,  
  `Produit_Nom` varchar(500) NOT NULL,  
  `Produit_Img_Url` varchar(500) NOT NULL,  
  `Produit_Prix` double NOT NULL,  
  `Categorie_Id` int(11) NOT NULL,  
  PRIMARY KEY (`Produit_Id`),
```

```
    UNIQUE KEY `Produit_Nom` (`Produit_Nom`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1 AUTO_INCREMENT=1 ;
```

```
--
-- RELATIONS POUR LA TABLE `produit`:
--   `Categorie_Id`
--   `categorie` -> `Categorie_Id`
--
```

```
-- -----
```

```
--
-- Structure de la table `promotion`
--
```

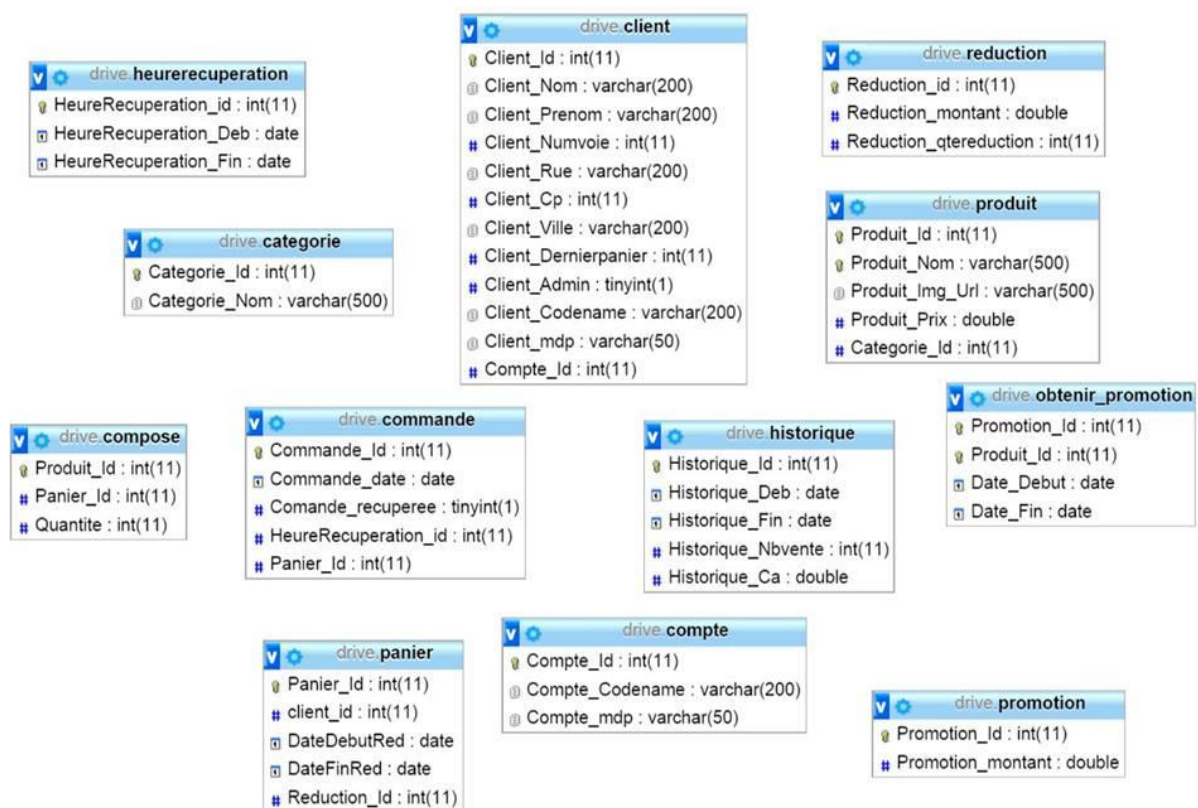
```
CREATE TABLE IF NOT EXISTS `promotion` (
  `Promotion_Id` int(11) NOT NULL AUTO_INCREMENT,
  `Promotion_montant` double NOT NULL,
  PRIMARY KEY (`Promotion_Id`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1 AUTO_INCREMENT=1 ;
```

```
-- -----
```

```
--
-- Structure de la table `reduction`
--
```

```
CREATE TABLE IF NOT EXISTS `reduction` (
  `Reduction_id` int(11) NOT NULL AUTO_INCREMENT,
  `Reduction_montant` double NOT NULL,
  `Reduction_qtereduction` int(11) NOT NULL,
  PRIMARY KEY (`Reduction_id`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1 AUTO_INCREMENT=1 ;
```

# Modèle de la base de données



## Créations des Triggers & Procédures à partir des MCT et MOT

### //Trigger de vérification de l'inscription :

```
CREATE TRIGGER inscription AFTER INSERT ON client FOR EACH ROW
```

```
BEGIN
```

```
IF NEW.Client_Nom IS NOT NULL OR NEW.Client_Prenom IS NOT NULL THEN
```

```
    SELECT 'Erreur : Vous devez renseigner votre nom et votre prénom.'
```

```
ELSEIF LENGTH(NEW.Client_mdp)>8 THEN
```

```
    SELECT 'Erreur : Votre mot de passe doit contenir au moins 8 caractères pour plus de sécurité.'
```

```
END IF
```

```
END
```

## **//Trigger de vérification de l'horaire de récupération de la commande :**

```
CREATE TRIGGER validation_horaire AFTER INSERT ON heurerecuperaction FOR  
EACH ROW
```

```
BEGIN
```

```
    IF (SELECT DATEDIFF(NEW.HeureRecuperation_Deb,  
CURRENT_DATE()))<=0 THEN
```

```
        SELECT 'Erreur : LA date de récupération de votre commande doit être  
postérieure à la date actuelle.'
```

```
    ELSE
```

```
        SELECT 'Un message vous sera envoyé une heure avant la récupération de  
votre commande.'
```

```
    END IF
```

```
END
```

## **//Archivage des commandes :**

```
CREATE PROCEDURE archivage()
```

```
BEGIN
```

```
    IF HeureRecuperation_Deb==CURRENT_DATE() THEN
```

```
        //Envois du message
```

```
        INSERT INTO `historique` (*commande courant*) VALUES ()
```

```
    END IF
```

```
END
```

# Insertion de contenu dans la base de données

## Quelques requêtes d'insertions dans la table produit :

```
INSERT INTO `produit`(`Produit_Id`, `Produit_Nom`, `Produit_Img_Url`,  
`Produit_Prix`, `Categorie_Id`) VALUES (1, 'baguette',  
'images/alimentaire/baguette.jpg', 0.87, 1);
```

```
INSERT INTO `produit`(`Produit_Id`, `Produit_Nom`, `Produit_Img_Url`,  
`Produit_Prix`, `Categorie_Id`) VALUES (2, 'banane',  
'images/alimentaire/banane.jpg', 0.95, 1);
```

```
INSERT INTO `produit`(`Produit_Id`, `Produit_Nom`, `Produit_Img_Url`,  
`Produit_Prix`, `Categorie_Id`) VALUES (3, 'bébé chat',  
'images/alimentaire/bebe_chat.jpg', 3.50, 1);
```

```
INSERT INTO `produit`(`Produit_Id`, `Produit_Nom`, `Produit_Img_Url`,  
`Produit_Prix`, `Categorie_Id`) VALUES (4, 'cereales',  
'images/alimentaire/cereales.jpg', 1.20, 1);
```

```
INSERT INTO `produit`(`Produit_Id`, `Produit_Nom`, `Produit_Img_Url`,  
`Produit_Prix`, `Categorie_Id`) VALUES (5, 'champignons',  
'images/alimentaire/champignons.jpg', 2.50 ,1);
```

```
INSERT INTO `produit`(`Produit_Id`, `Produit_Nom`, `Produit_Img_Url`,  
`Produit_Prix`, `Categorie_Id`) VALUES (6, 'cookies',  
'images/alimentaire/cookies.jpg', 2.15 ,1);
```

```
INSERT INTO `produit`(`Produit_Id`, `Produit_Nom`, `Produit_Img_Url`,  
`Produit_Prix`, `Categorie_Id`) VALUES (7, 'croissant',  
'images/alimentaire/croissant.jpg', 0.86 ,1);
```

```
INSERT INTO `produit`(`Produit_Id`, `Produit_Nom`, `Produit_Img_Url`,  
`Produit_Prix`, `Categorie_Id`) VALUES (8, 'eau', 'images/alimentaire/eau.jpg', 1.30  
,1);
```

## Quelques requêtes d'insertions dans la table categorie :

```
INSERT INTO `categorie`(`Categorie_Id`, `Categorie_Nom`) VALUES  
(1,'Alimentaire');
```

```
INSERT INTO `categorie`(`Categorie_Id`, `Categorie_Nom`) VALUES (2,'Beauté');  
INSERT INTO `categorie`(`Categorie_Id`, `Categorie_Nom`) VALUES (3,'Bricolage');  
INSERT INTO `categorie`(`Categorie_Id`, `Categorie_Nom`) VALUES  
(4,'Electroménager');  
INSERT INTO `categorie`(`Categorie_Id`, `Categorie_Nom`) VALUES (5,'Maison');  
INSERT INTO `categorie`(`Categorie_Id`, `Categorie_Nom`) VALUES (6,'Vêtement');
```

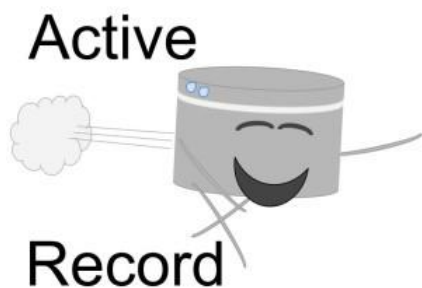
## Réalisation du site web

### Création des modèles et vérifications des fonctions d'insertion, modification et suppression sur les tables

Le M de MCV est pour Modèle. La partie Modèle du MCV consiste à gérer les données qui entrent et qui sortent de l'application, ainsi que leur structure dans le code. Le but est que celui qui s'occupe des autres parties du site puisse appeler des méthodes comme ajouter(), modifier() sans se soucier de récupérer les données, de leur format et des modalités de sauvegarde (base de données, fichier XML, objet sérialisé...). Pour ce faire, nous avons utilisé le pattern Active Record.

#### Qu'est-ce que l'Active Record ?

L'active record est un patron de conception qui consiste à faire une classe pour chaque table d'une base de données, chaque champ de la table correspondant à un attribut de la classe, pour modéliser efficacement la base de données. Ainsi, à une instance d'une classe correspond une ligne de la table de même nom. Chacune de ces classes possède des méthodes de recherche, d'insertion, puis de modification et/ou de suppression suivant les besoins. La connexion à la base de données est gérée par une classe singleton Base qui stocke la même connexion durant toute la durée de l'application.

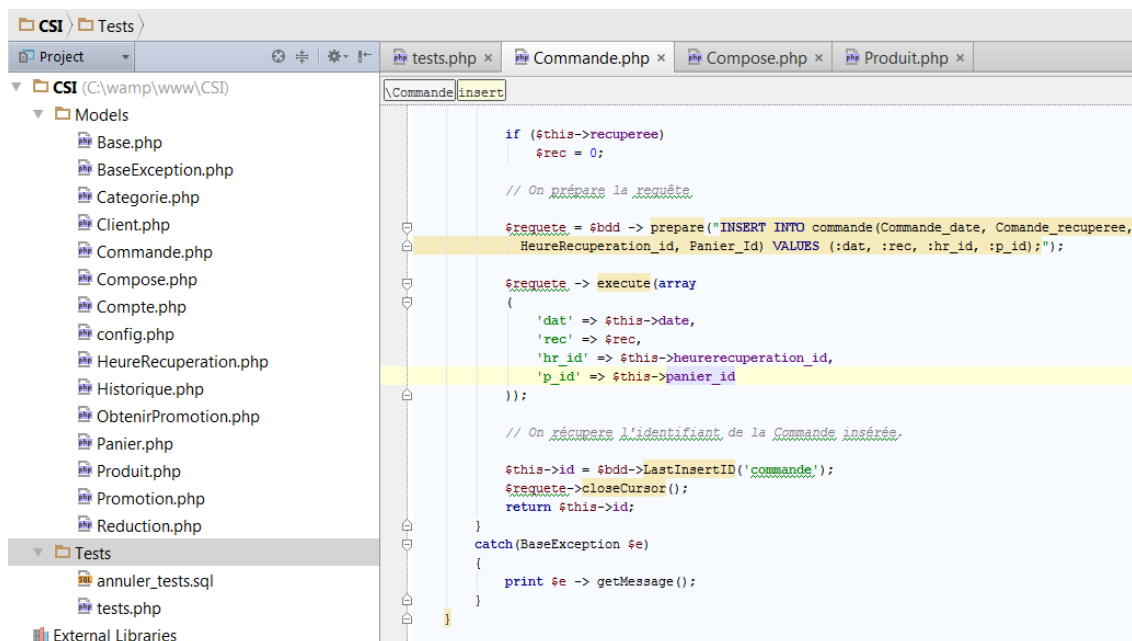




## Réalisation de l'Active Record ?

Pour commencer, il fallait que le script de création de la base de données soit fait. L'implantation de l'Active Record s'est déroulée sans grande difficulté. Il faut réfléchir aux besoins de chaque table spécifique, par exemple, il faut prévoir les différents critères de recherche pour la classe Produit, ou une recherche par Panier\_Id pour la Commande, prendre en compte les besoins exprimés par les différents MCT et MOT ainsi que des vérificateurs d'unicités pour les champs uniques.

Nous avons aussi dû réfléchir sur des choix de passages de certains objets (ou ligne de table) par référencement (sélectionner uniquement l'ID), ou par valeur (sélectionner tous les attributs). La sélection par référence complique le travail du programmeur ensuite, et oblige à faire plusieurs requêtes SQL, mais le passage par valeur entraîne le risque de demander à la base de données des champs qui seront très peu utilisés.



```
if ($this->recuperee)
    $rec = 0;

// On prépare la requête
$requete = $bdd -> prepare("INSERT INTO commande(Commande_date, Commande_recuperee,
HeureRecuperation_id, Panier_Id) VALUES (:dat, :rec, :hr_id, :p_id);");

$requete -> execute(array
(
    'dat' => $this->date,
    'rec' => $rec,
    'hr_id' => $this->heurerecuperation_id,
    'p_id' => $this->panier_id
));

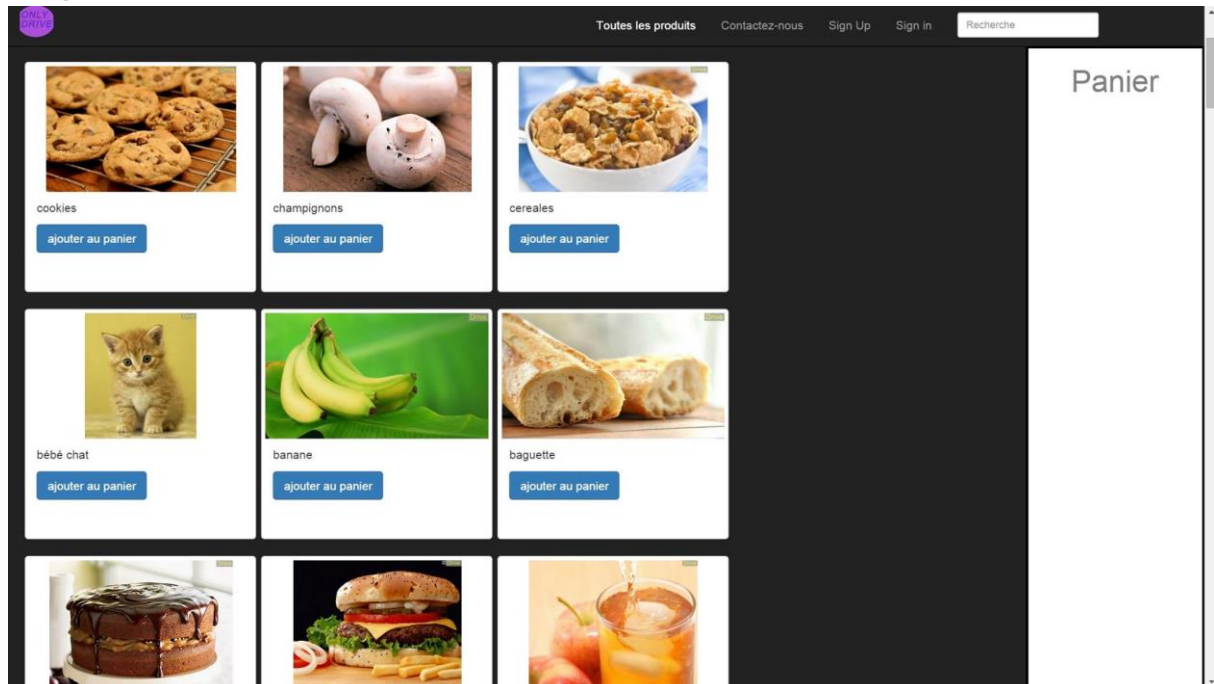
// On récupère l'identifiant de la Commande insérée.
$this->id = $bdd->LastInsertID('commande');
$requete->closeCursor();
return $this->id;
}
catch(BaseException $e)
{
    print $e -> getMessage();
}
```

*Le codage pour l'Active Record s'est fait à l'aide du logiciel PHPStorm*

Nous avons enfin dû réaliser les tests unitaires de chaque méthode de chaque classe afin de vérifier qu'elles se comportaient comme nous l'avions souhaité. Une fois que tous les tests marchaient, il ne restait plus qu'à créer les différentes pages du site Web sans se soucier de la base de données.

# Screenshot du site

## Page d'accueil :

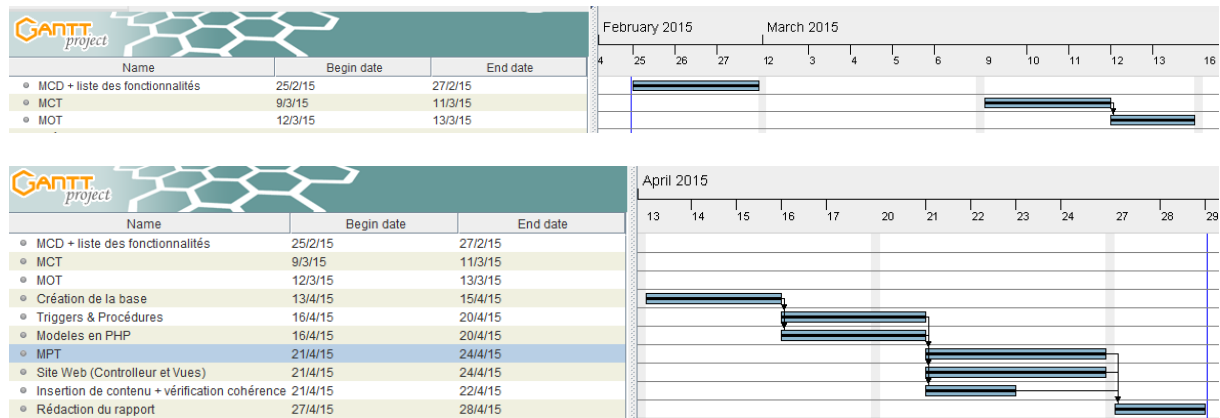


## Page d'inscription :

Nous sommes toujours en cours de développement du site pour satisfaire toutes les fonctionnalités attendues. Nous avons une page de test en php qui vérifie toutes les insertions suppressions sur la base de données, notre site sera prête pour avec plus de fonctionnalités possible pour une démonstration au moment de la présentation.

# La répartition des tâches et Organisation

## Diagramme de Gantt



## Organisation du groupe

Nous avons utilisées un dépôt Git privée. Le site web est développé sur l'IDE Netbeans. Pour les modèles MERISE nous avons utilisées Power AMC. Gestion de la base de données MYSQL est avec MYSQL Workbench. Nous avons utilisé un serveur local apache WAMP server. Les diagrammes de Gantt sont faites avec Gantt project.

# Conclusion

En conclusion ce projet nous a permis de comprendre comment résoudre un problème complexe en suivant la méthode de conception MERISE. De plus nous avons appris à travailler en équipe.