PLATEFORME WEB B2B

PROJET :

DEVELOPPEMENT D'UNE PLATE-FORME D'ECHANGE LOGISTIQUE PROFESSIONNELLE (B2B)

But du projet : Gestion des interactions entre les différents professionnels et clients, Gestion des enchères, Gestion des réseaux, Gestion des stocks, entreposages et saisies de commandes

Auteur : Olivier EDWIGE

Table des matières

1) LES OBJECTIFS 3

2) PRESENTATION DU PROJET 5

3) OBJECTIFS A ATTEINDRE 6

4) CHOIX ET PRESENTATION DES OUTILS DE DEVELOPPEMENT 7

4.1) PHP/MYSQL, CSS, XML, WEB SERVICE 7

4.2) PRESENTATION D'UN WEB SERVICE 8

5) STRUCTURE DE LA BASE DE DONNEES (MODELE UML) 9

6) ARCHITECTURE DU SITE, MODELE MVC 26

6.1) PRESENTATION MODELES DE CLASSE (CMEMBRE…..) 28

7) PRESENTATION DE L'INTERFACE UTILISATEUR 31

7.1) STRUCTURE DE L'INTERFACE WEB 38

7.2) PRINCIPE DE L'INTERFACE DE MISE A JOUR (ADMIN) 38

8) PRESENTATION DE DRUPAL 40

9) EVOLUTION DU SITE 42

10) ANNEXE 43

1) Les Objectifs

L'objectif principal du projet est de proposer un site internet d'intermédiation entre les fournisseurs (déposants) et les différents acteurs du site : client (point de vente), entrepôt, transitaire et dépositaire. La régulation des échanges et des inscriptions étant gérée par l'administrateur et les validateurs.

Agréger des catalogues produits

Aujourd'hui, les agrégateurs en ligne fournissent les catalogues de leurs partenaires à l'aide de solutions propriétaires : certains d'entre eux assemblent et ajoutent leur marque au contenu agrégé en y insérant des liens pointant sur les sites web de leurs partenaires : d'autres téléchargent chaque nuit les informations nécessaires depuis les sites partenaires à l'aide de protocoles de transfert de fichiers comme FTP.

Ces formes d'agrégation ad hoc posent plusieurs problèmes techniques. Habituellement un agrégateur assemble le contenu issu de plusieurs sites partenaires. Chaque fois qu'il incorpore le contenu d'un nouveau partenaire, il doit fournir un effort de conception et de mise en œuvre important. Le partenaire doit, quant à lui, former chaque nouvel agrégateur à l'utilisation de son système d'information ; le déploiement devient de plus en plus laborieux et les clients obtiennent in fine des services moins bons.

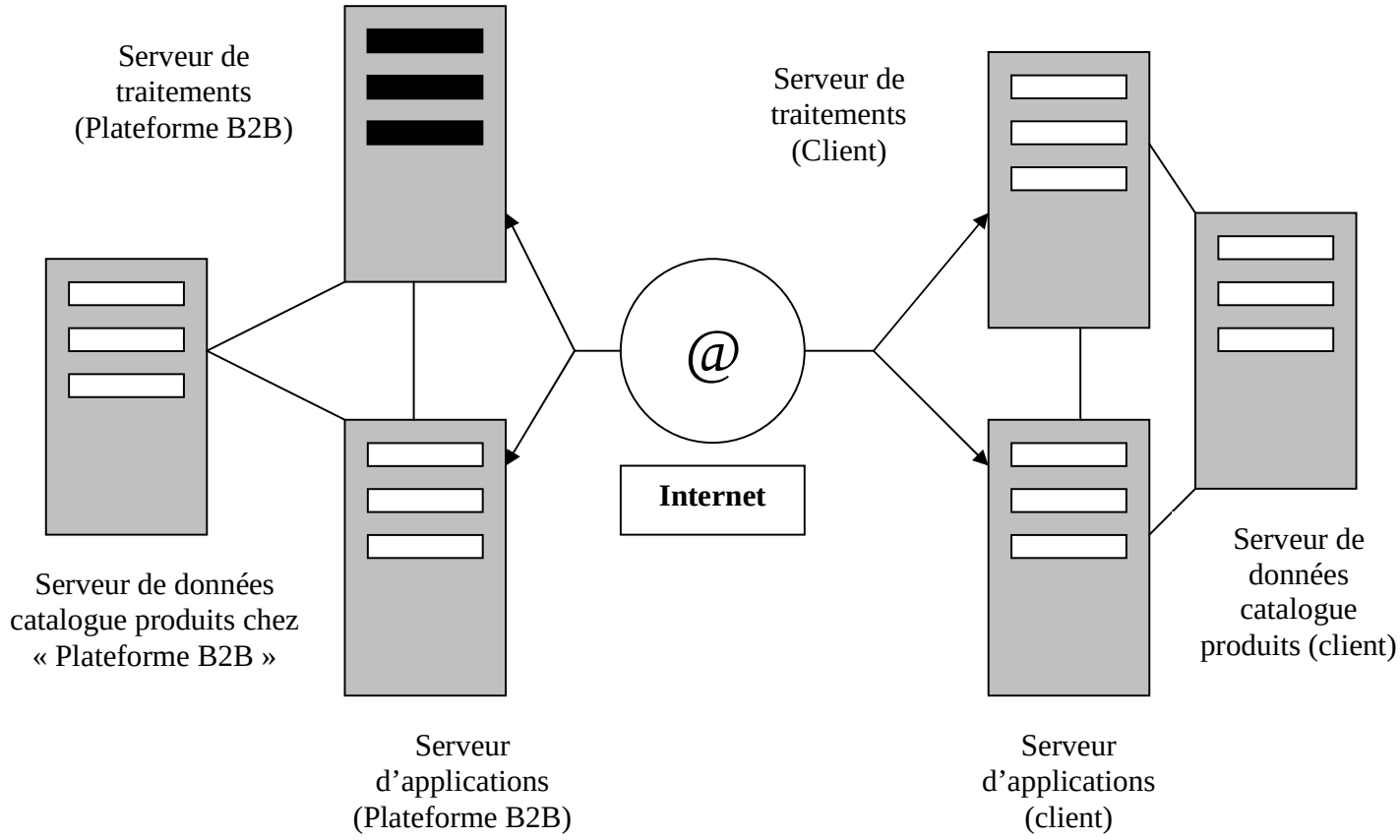
Ces approches exigent typiquement une intervention humaine pour régler les questions opérationnelles les plus simples :

- Comment les partenaires peuvent-ils transmettre à l'agrégateur les informations commerciales dont-il a besoin : disponibilité de certains produits, promotions,… ?
- A quelle fréquence la base de données du site doit-elle être synchronisée (ou mise à jour via fichiers d'export) avec la catalogue de produits de ses partenaires ?

- Comment informer les partenaires des éléments logistiques nécessaires à la livraison ?
- Comment transmettre aux partenaires les informations relatives à leur profil lors de leur connexion ?
- Comment gérer et suivre les commandes des partenaires ?

Proposer une approche innovante aux partenaires

Leur donner accès à de nouveaux marchés, tout en limitant au strict minimum les adaptations à faire subir à leur système d'information. Leur fournir un service à valeur ajoutée tels que la mise à disposition des statistiques d'utilisation du site (ciblées sur le compte) ou des informations sur les profils des clients.
La plateforme d'agrégation doit s'appuyer sur des protocoles standards pour limiter les difficultés de mise en œuvre et pour permettre aux partenaires, une fois cette plateforme établie, de dialoguer et d'échanger du contenu en toute sérénité. Schéma 1 :



Serveur d'application : il permettra aux utilisateurs de consulter le catalogue du déposant et de passer des commandes.

Serveur de données : il rassemblera l'ensemble des données importées du serveur de traitements, il contiendra l'intégralité des catalogues partenaires.

Serveur de traitement : Il assurera la récupération des catalogues des partenaires et l'intégration des données dans la base de données. C'est sur ce serveur que devra être réalisée l'agrégation de contenu. Il devra être capable de dialoguer avec les serveurs d'agrégation des partenaires (ex. échange EDI).

Cette architecture répartit pourra faire face à de fortes montées en charge. Le serveur ne supporte ainsi ni la charge liée aux requêtes adressées à la base de données, ni celle liée à la récupération des catalogues et aux différents traitements qui en découlent.

2) Présentation du projet

Etablir un cahier des charges précis et fonctionnel (entretien, collecte d'informations, validation auprès des profesionnels…)

Public visé/cible

- Professionnels

- Business to business (B2B)

Langues

- Multi langages

Spécifications techniques

- Création des modèles (templates) de pages « catalogue », « accueil » et formulaire divers

- Moteur de recherche
- PHP/Mysql
- hébergement WAMP
- Echange EDI (base de données client sous SQLServer).

Interactivité

- Inscription sur le site
- Création de commande
- Récupération de données
- Echange d'informations
- Appartenance à un réseau

Maintenance

- Interface admin ou répertoire admin ; mise à jour du site par l'intermédiaire de fichier (XML,…) ou via interface web.

3) Objectifs à atteindre

Le rôle de la Plateforme B2B sera de permettre :

L'inscription d'un nouveau membre (formulaire),
La gestion de sa connexion au site, déconnexion,
L'affichage des produits (infos, prix…),
La gestion des commandes (sélection des produits affichés sur le site, visualisation du caddy),
La recherche d'un produit (recherche syntaxique).

Consultation fiche produit
Création d'un attendu
La validation des commandes,
Le contrôle des stocks disponibles.

L'affichage du compte client et de son interface adaptée
La gestion du mot de passe oublié
La mise à jour du site
La consultation des fiches de présentation des acteurs (accessible par héritage)
L'échange d'informations entre partenaires

4) Choix et présentation des outils de développement

4.1) PHP, MySQL, CSS, WEB SERVICE

Le site sera développé en PHP avec MySQL comme base de données (j'aurai aussi pu opter pour une technologie JAVA/XML avec base de données Oracle et composant EJB, mais plus couteux pour le client final). Pour cela différents outils peuvent être implémentés, notamment la possibilité d'utiliser un CMS afin de construire rapidement les principales interfaces du site. Le CMS Drupal semble être un bon compromis pour lier des composants spécifiques (développés dans le cadre du projet) et des composants prédéfinis (communauté open-source) afin d'optimiser certaines phases du développement (l'implémentation d'un module E-commerce par exemple).

Pour l'utilisation des WEB SERVICE, le site **http://www.xmethods.net** propose une multitude de solutions dont certaines pourront faire l'objet d'une adaptation spécifique pour le projet (par exemple pour les statistiques).

4.2) Présentation d'un Web Service (pour informations d'implémentation)

IP2Pais est un web service permettant d'obtenir le pays d'origine d'un visiteur à partir de son adresse IP. Ce web service a été trouvé sur le site http://www.xmethods.net. La spécification se trouve dans le répertoire du site dans le fichier IP2Pais.txt. Ci-dessous le code source du client :

```
/* web service permettant d'obtenir le pays d'origine du visiteur */

include('../ressources/lib/nusoap.php');
$wsdl = "http://webservices.tekever.eu/ip2pais/?wsdl";
$client = new soapclient($wsdl, 'wsdl');
$parametres = $_SESSION["ip"];
$result = $client->call('IP2Pais', $parametres);

/* écriture dans un fichier */

$fichier = "Iporigine.txt";
$PtFic = fopen($fichier,"w");
fwrite($PtFic,$result."\n");
fclose($PtFic);
```

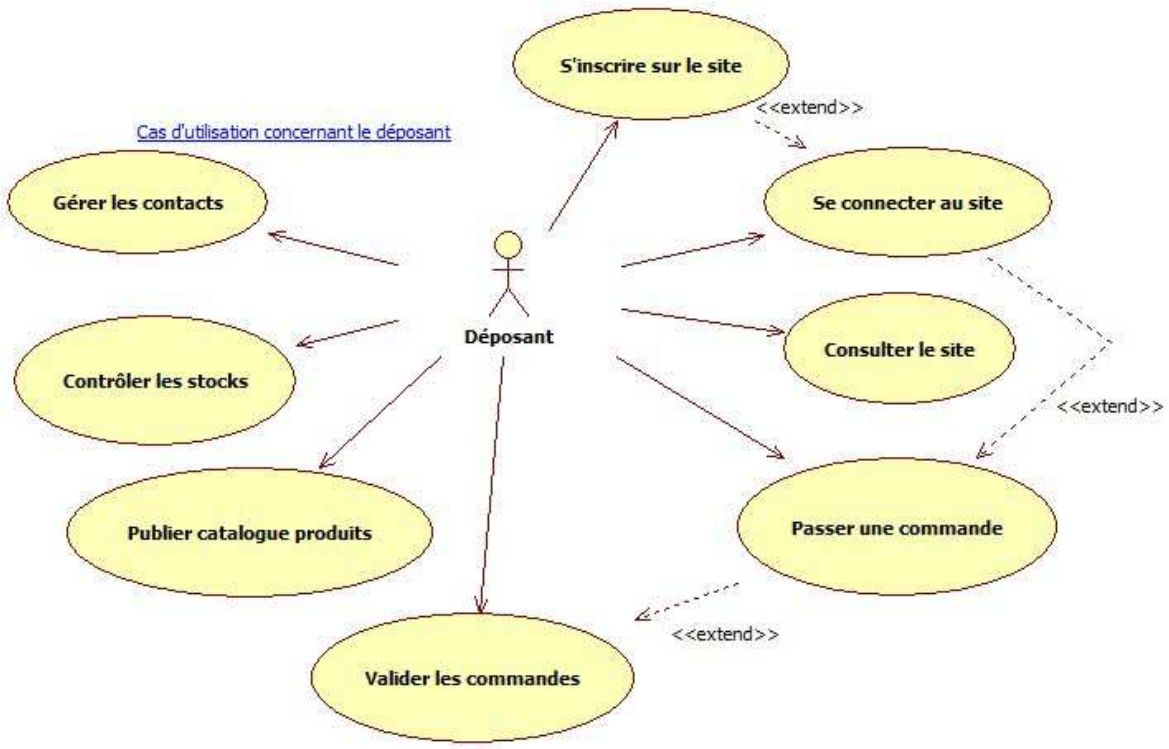
Cet exemple permettra de dresser des statistiques sur les pays d'origines des internautes.
Les web services sont des outils simples à mettre en œuvre et efficaces d'un point de vue fonctionnels

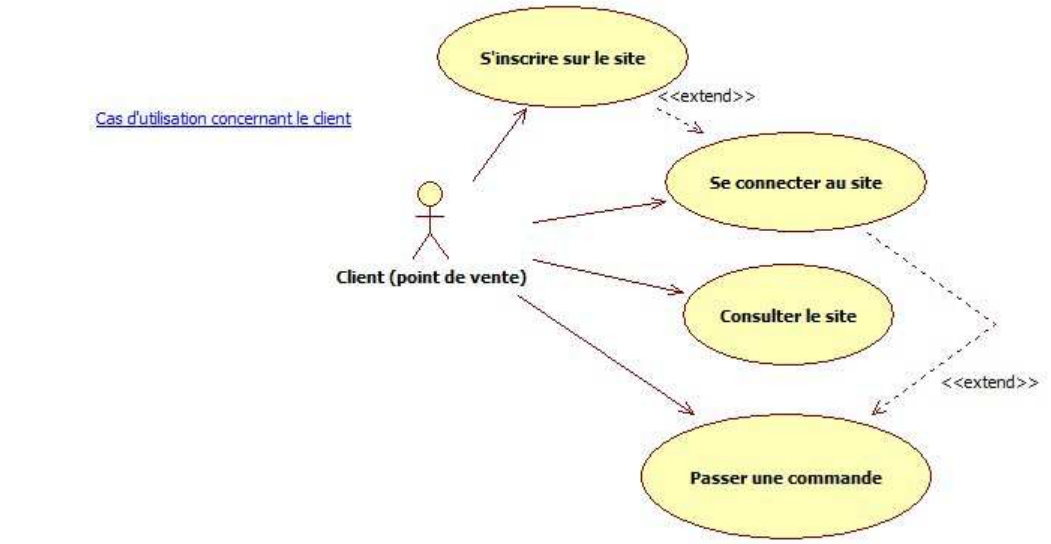
5) structure de la base de données (modele uml)

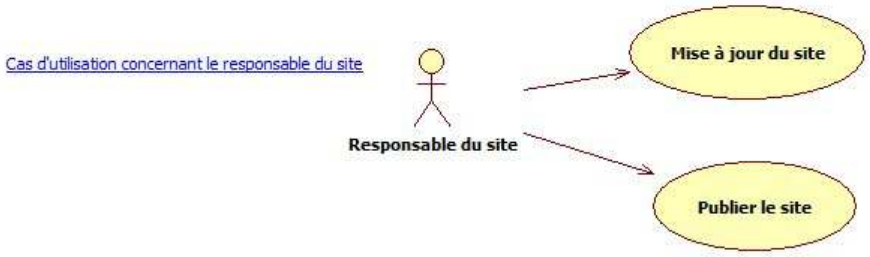
Afin d'établir les cas d'utilisation, j'ai d'abord identifié les acteurs :

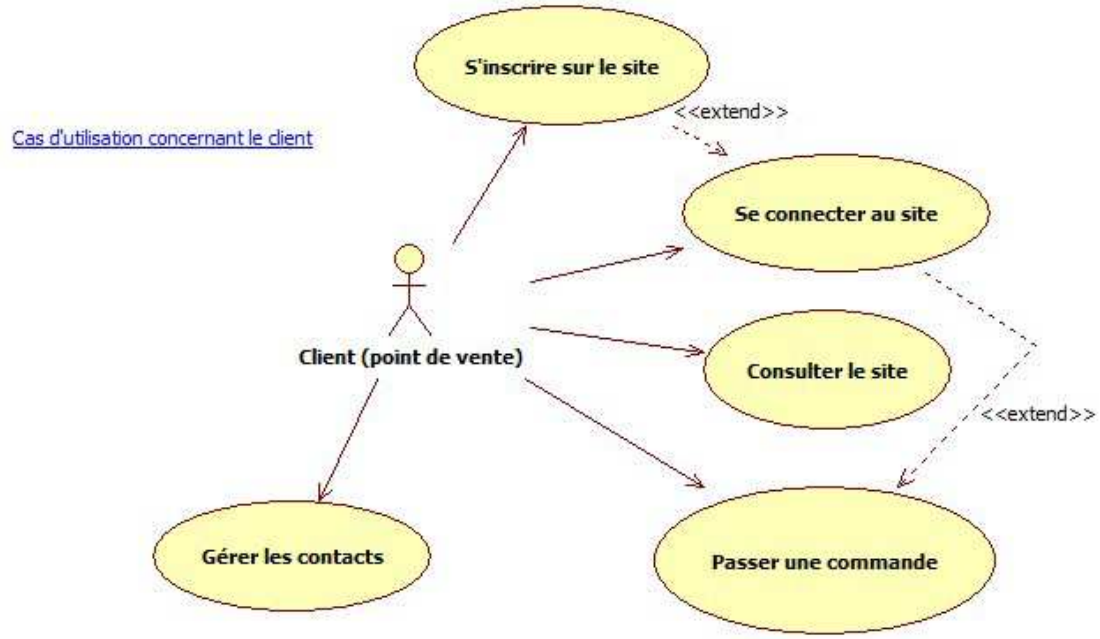
- L'internaute / Client (Point de vente)
- Le responsable du site (Administrateur)
- Le fournisseur (Déposant)
- Le transitaire
- Le validateur
- L'entrepôt
- Le dépositaire

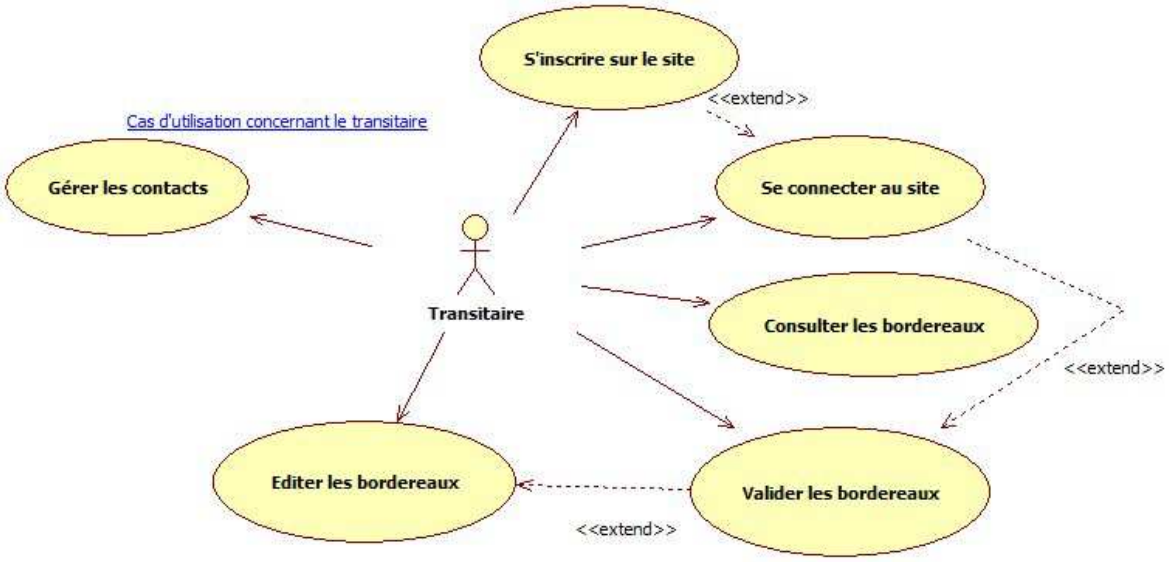
Ensuite on identifie les cas d'utilisation celui du Client et celui du Responsable :

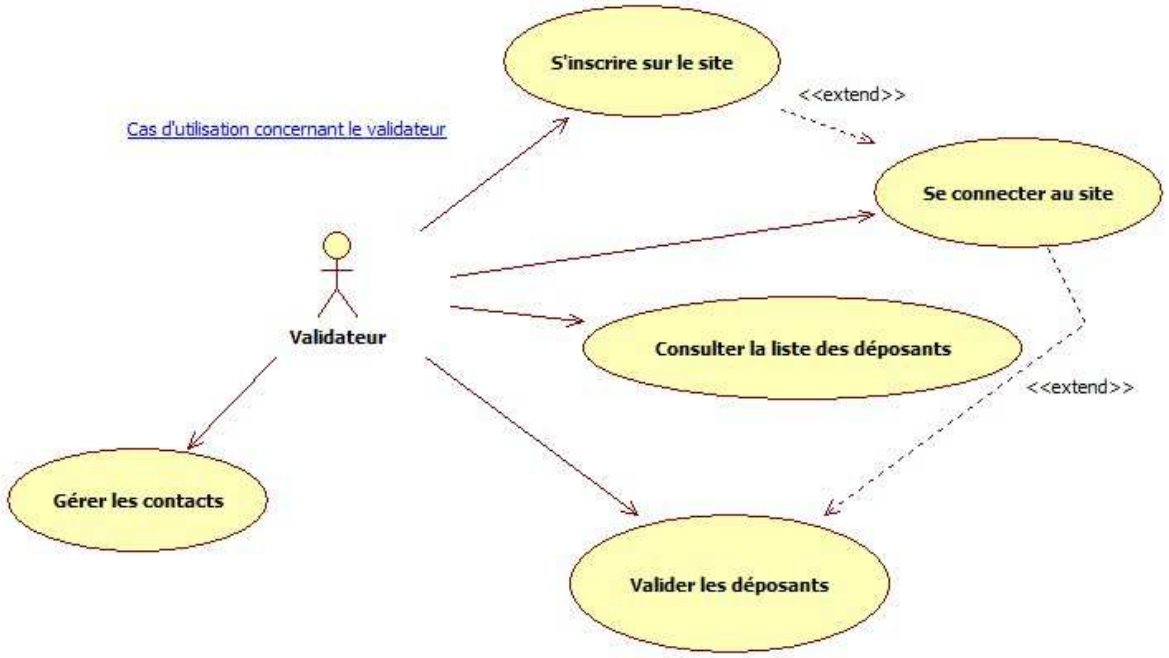


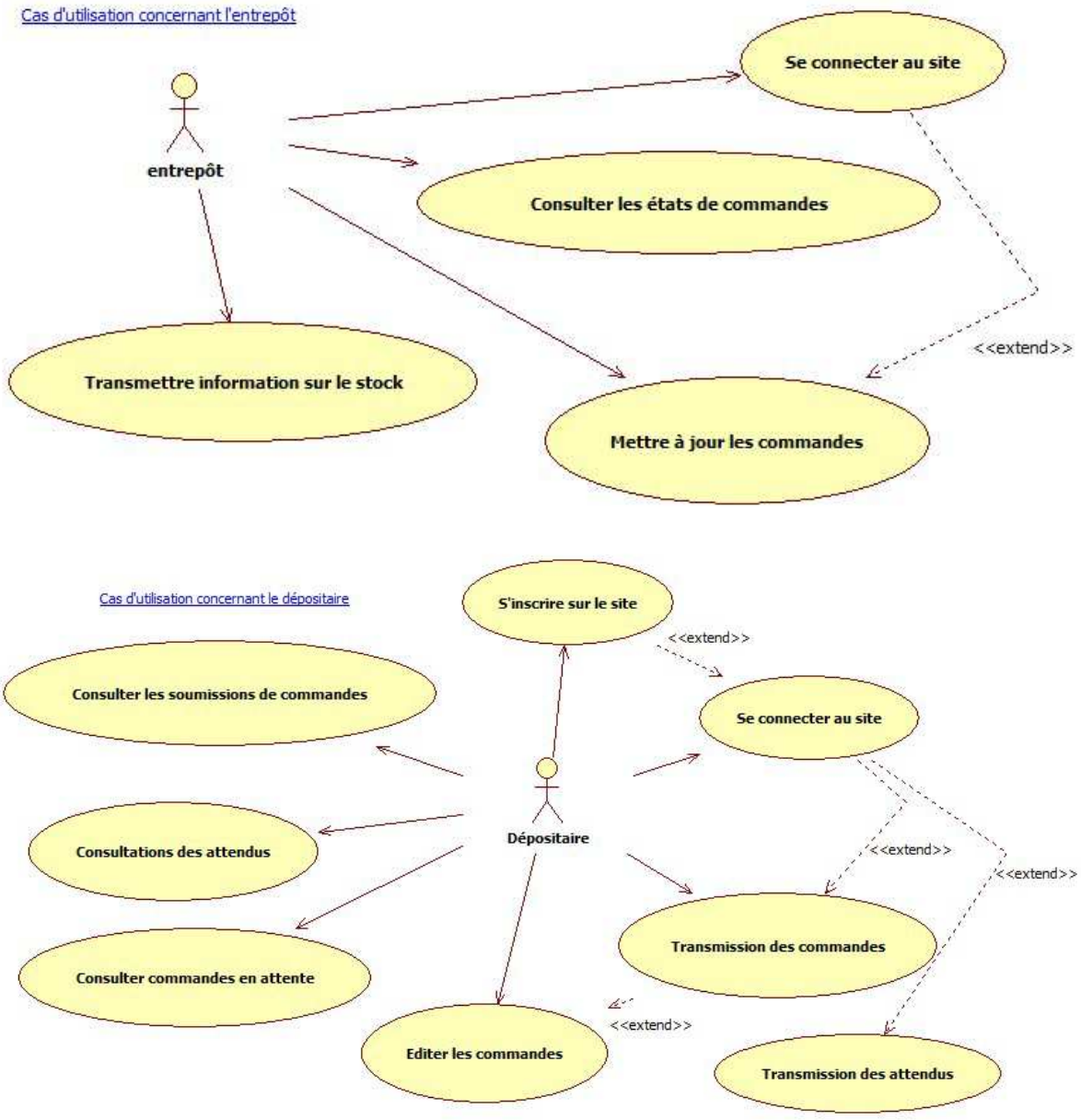

Cas d'utilisation concernant le client
S'inscrire sur le site
<<extend>>
Se connecter au site
Client (point de vente)
Consulter le site
<<extend>>
Passer une commande


Cas d'utilisation concernant le responsable du site
Mise à jour du site
Responsable du site
Publier le site

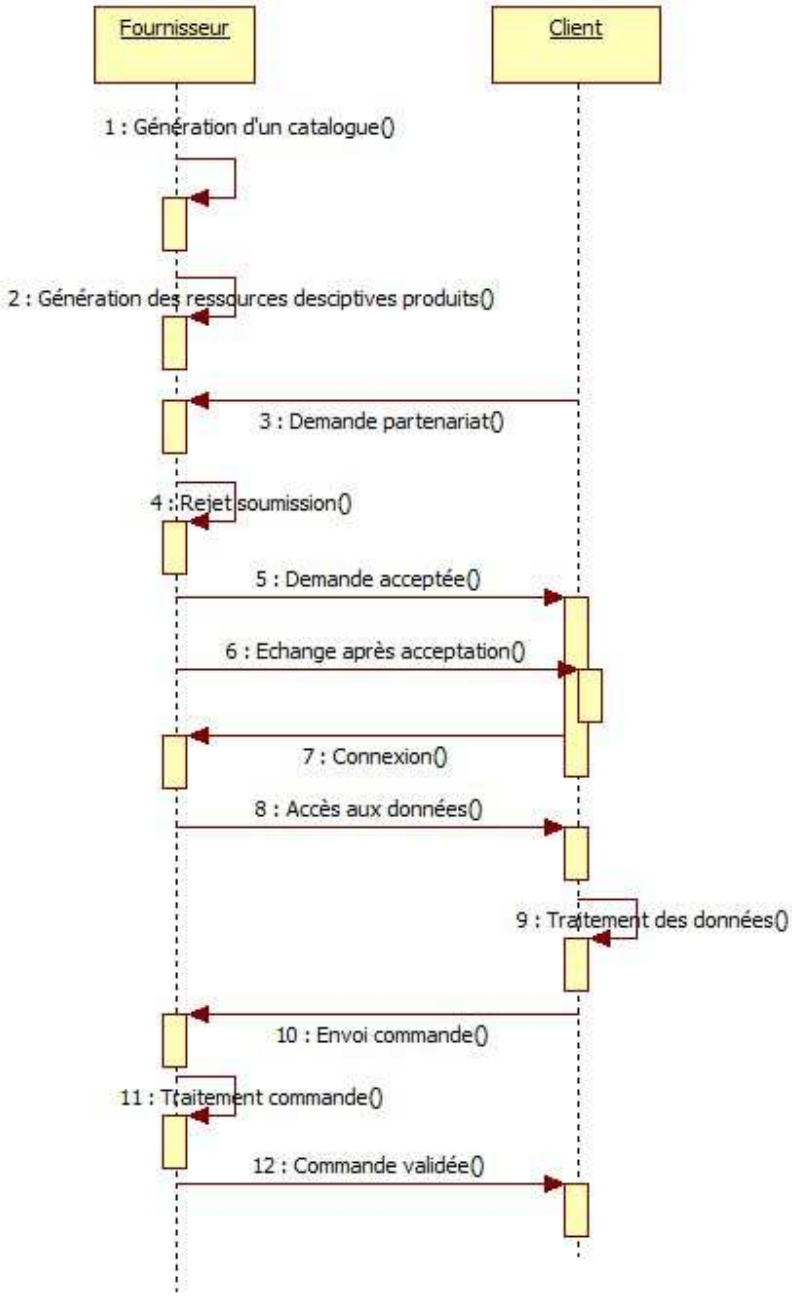

S'inscrire sur le site
<<extend>>
Cas d'utilisation concernant le client
Se connecter au site
Client (point de vente)
Consulter le site
<<extend>>
Gérer les contacts
Passer une commande

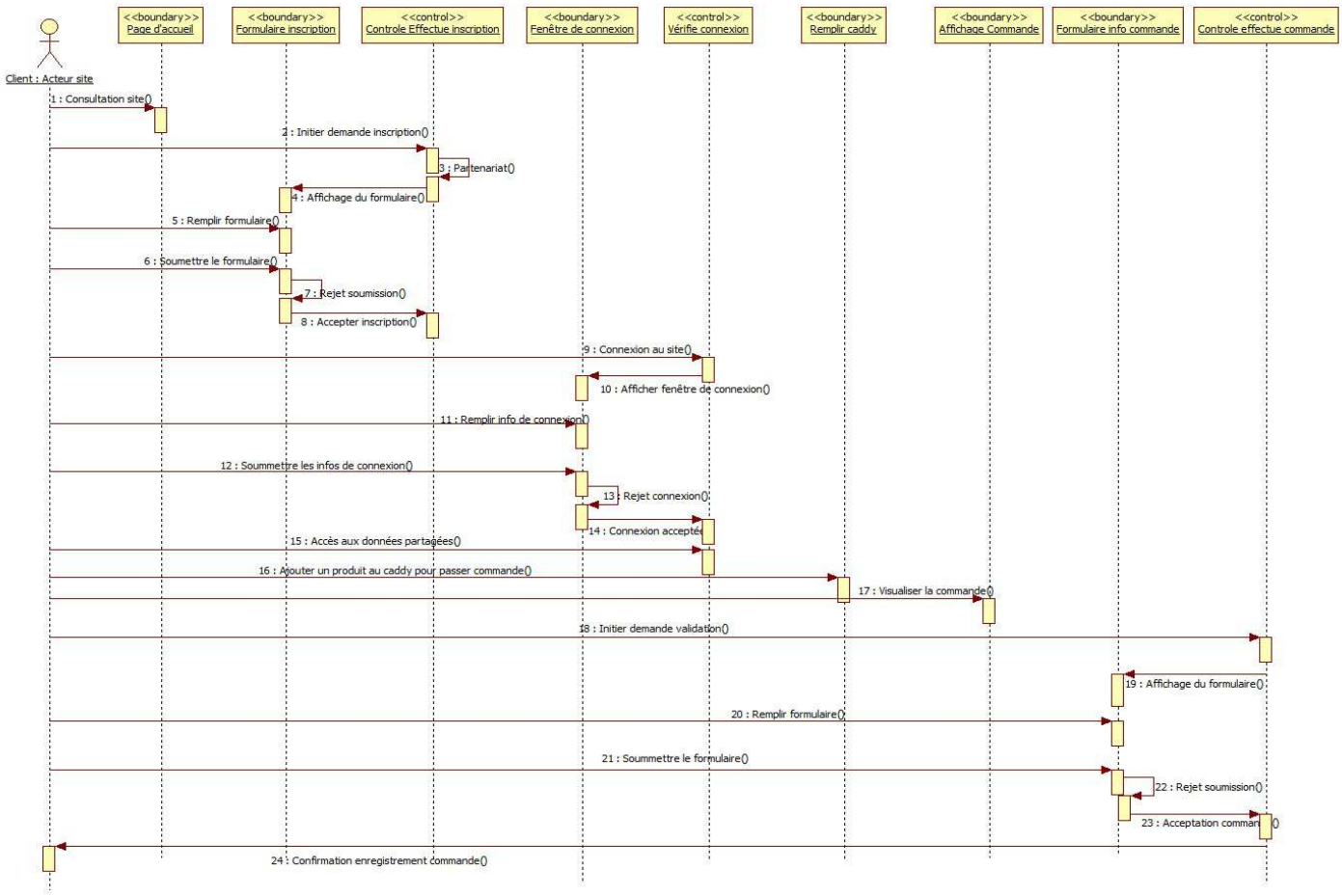

Cas d'utilisation concernant le transitaire
S'inscrire sur le site
<<extend>>
Gérer les contacts
Se connecter au site
Transitaire
Consulter les bordereaux
<<extend>>
Editer les bordereaux
Valider les bordereaux
<<extend>>


Cas d'utilisation concernant le validateur
S'inscrire sur le site
<<extend>>
Se connecter au site
Validateur
Consulter la liste des déposants
<<extend>>
Gérer les contacts
Valider les déposants



La cinématique du site peut être visualisée à partir des diagrammes de séquence ci-dessous :

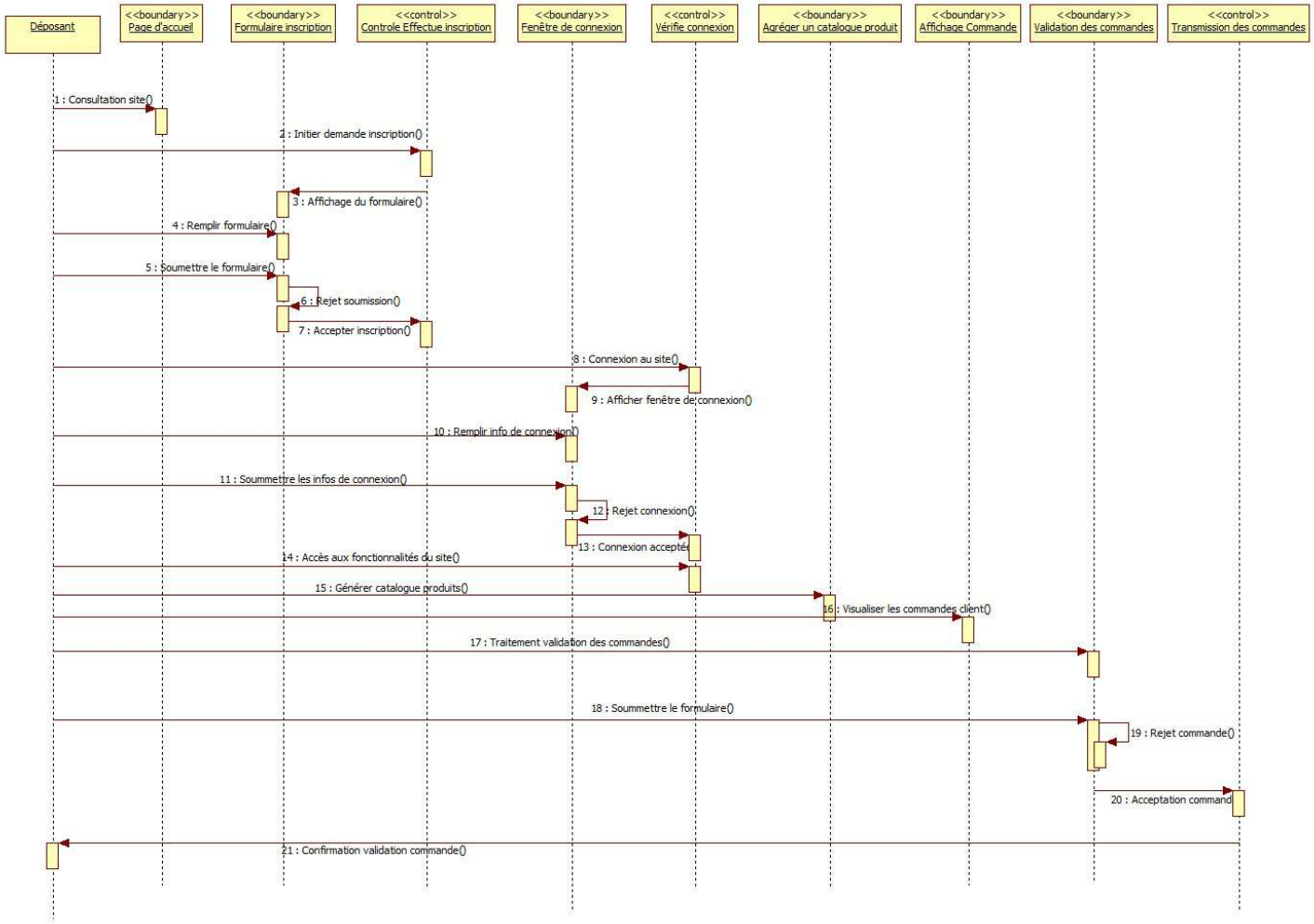

Fournisseur
Client
1 : Génération d'un catalogue()
2 : Génération des ressources desciptives produits()
3 : Demande partenariat()
4 : Rejet soumission()
5 : Demande acceptée()
6 : Echange après acceptation()
7 : Connexion()
8 : Accès aux données()
9 : Traitement des données()
10 : Envoi commande()
11 : Traitement commande()
12 : Commande validée()



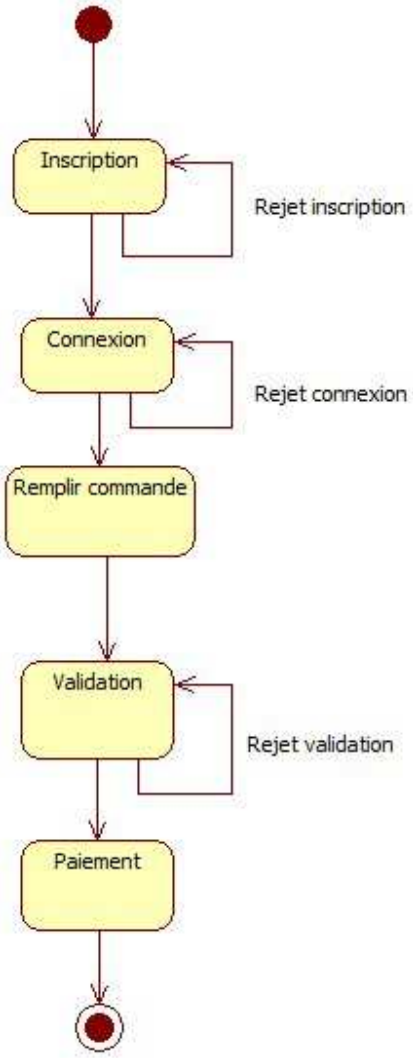
Explications :

1) L'internaute arrive sur le site
2) Il fait une demande d'inscription pour devenir membre
3) Il remplit puis soumet le formulaire
    a. Une vérification est effectuée en cas de problème on revient à l'étape 3 sinon on passe à l'étape 4
4) le client se connecte au site
    a. Une vérification est effectuée, en cas de problème on revient à l'étape 4 sinon on passe à l'étape 5
5) le client établit sa commande
6) le client valide sa commande en saisissant les informations demandée.
    a. Une vérification est effectuée, en cas de problème on revient à l'étape 6 sinon on passe à l'étape 7
7) la commande est enregistrée

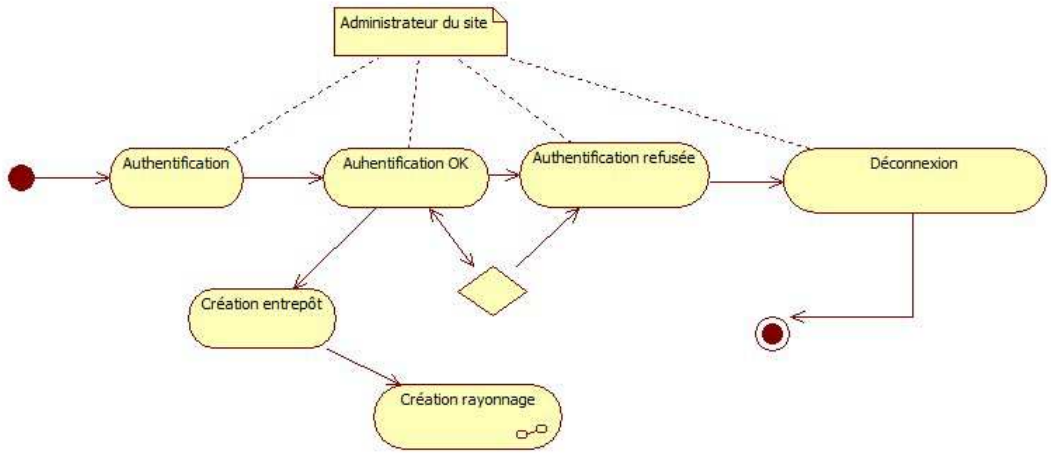
8) une confirmation est envoyée au client.

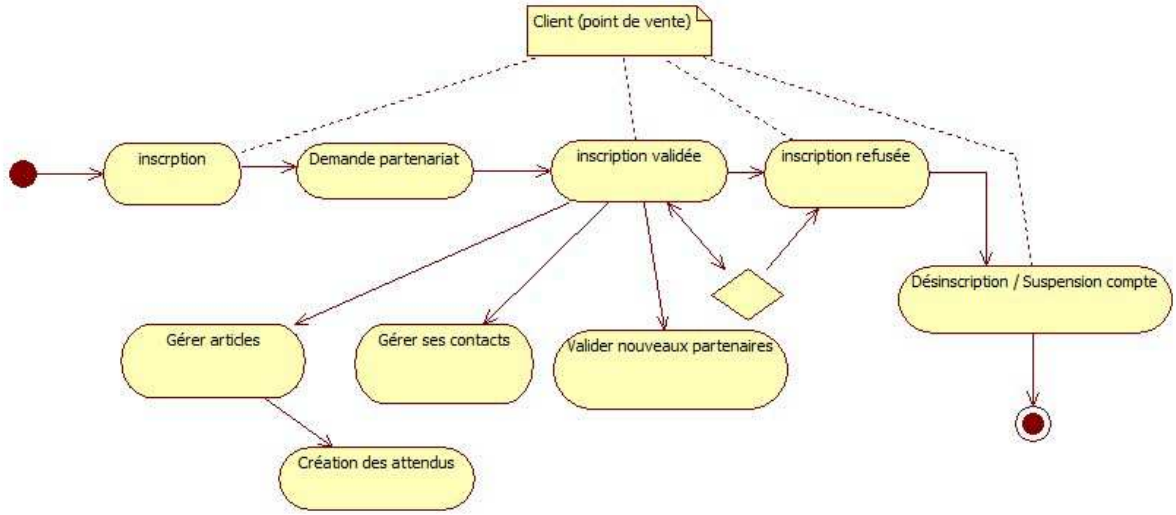


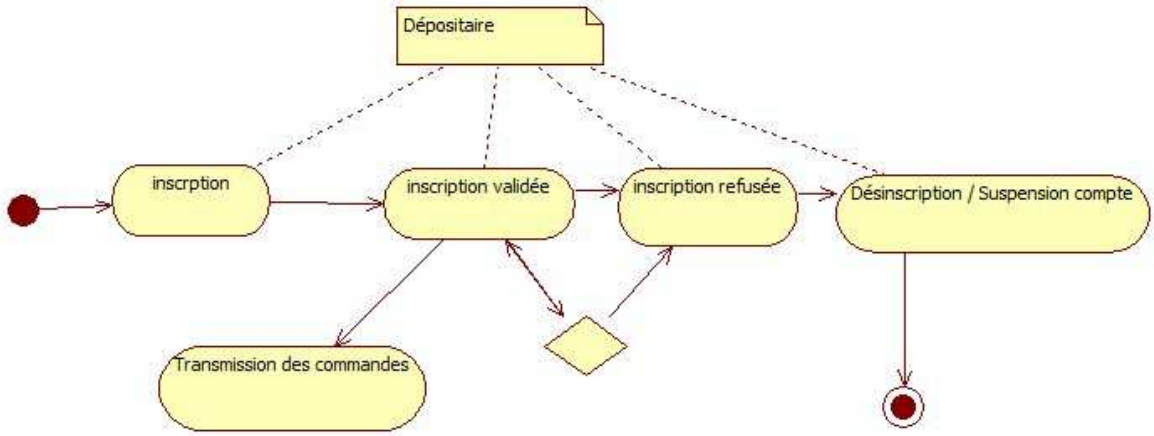
Le principe de la séquence pour le déposant est assez proche de celui décrit plus haut pour le client.
A partir de ces cinématiques nous pouvons établir le diagramme d'état/transition pour visualiser le cycle de vie des processus :

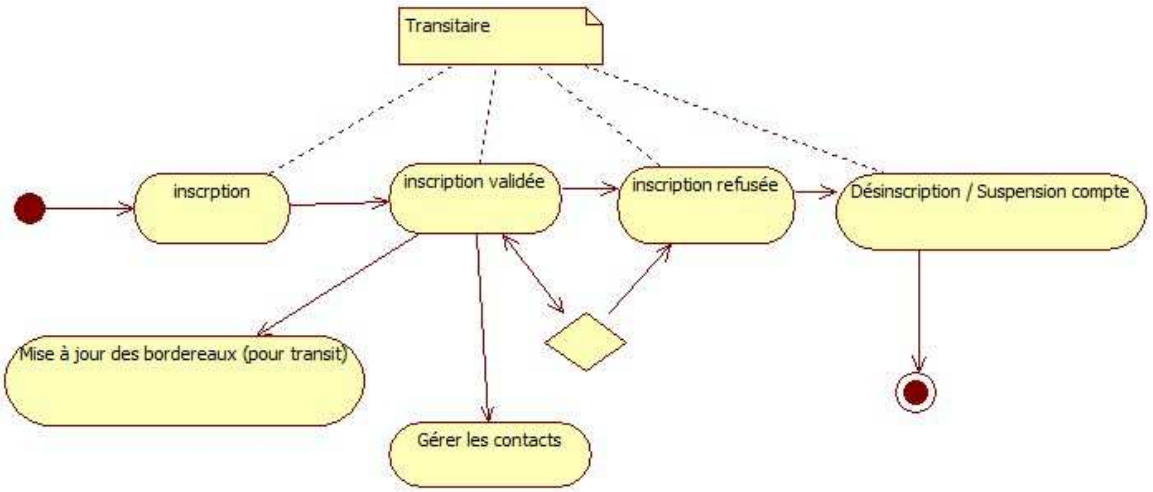


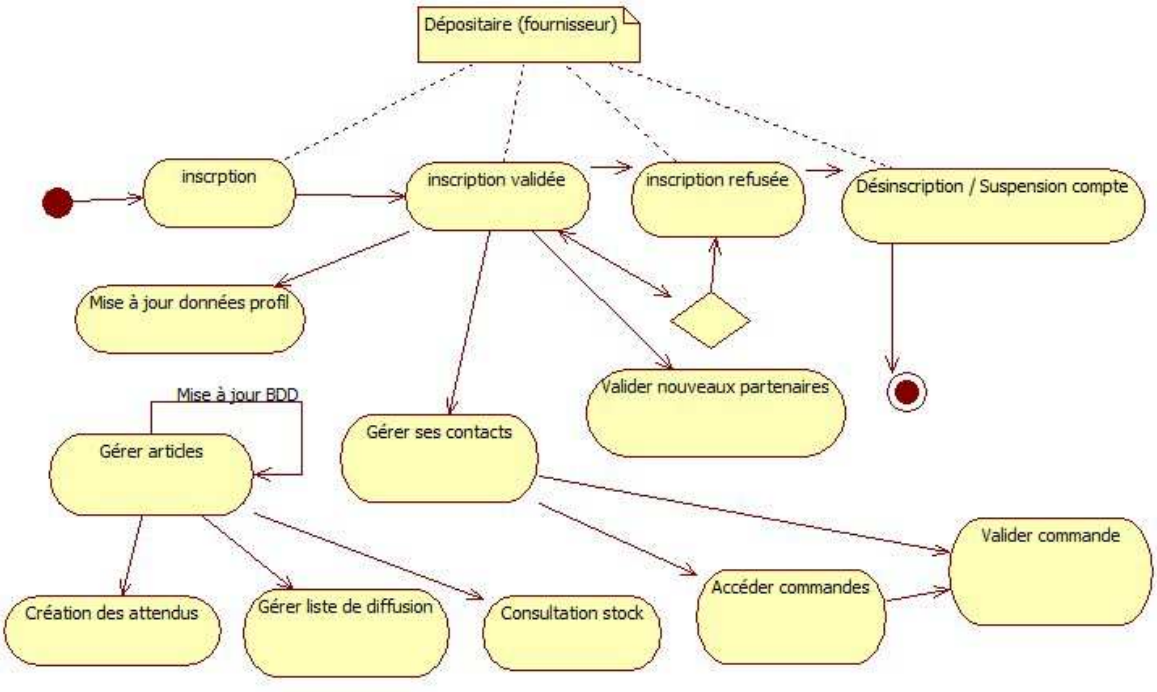
De même à partir des cas d'utilisation, nous pouvons établir les diagrammes d'activité pour les accès et mises à jour du site par les différents acteurs :


Administrateur du site
Authentification
Auhentification OK
Authentification refusée
Déconnexion
Création entrepôt
Création rayonnage

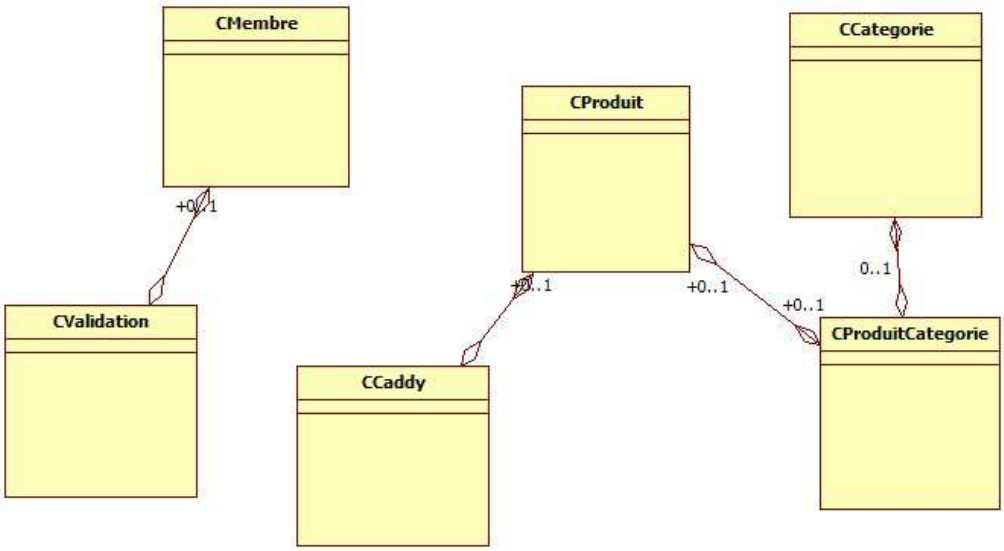

Client (point de vente)
inscrption
Demande partenariat
inscription validée
inscription refusée
Désinscription / Suspension compte
Gérer articles
Gérer ses contacts
Valider nouveaux partenaires
Création des attendus


Dépositaire
inscrption
inscription validée
inscription refusée
Désinscription / Suspension compte
Transmission des commandes



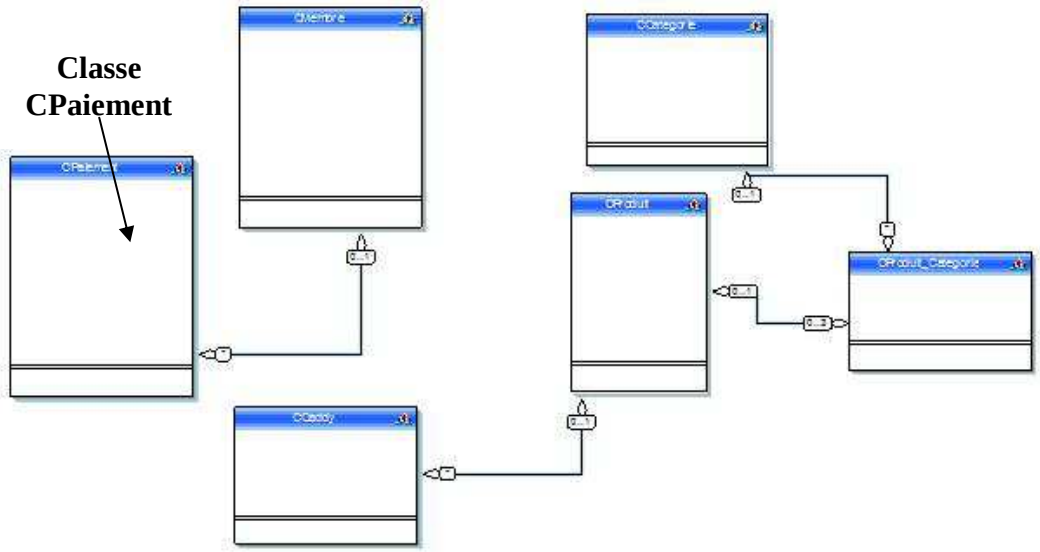


Avant de construire le diagramme de déploiement, établissons le diagramme de classe afin de visualiser la structure de la base de données pour le stockage des informations et les interactions entre classes :

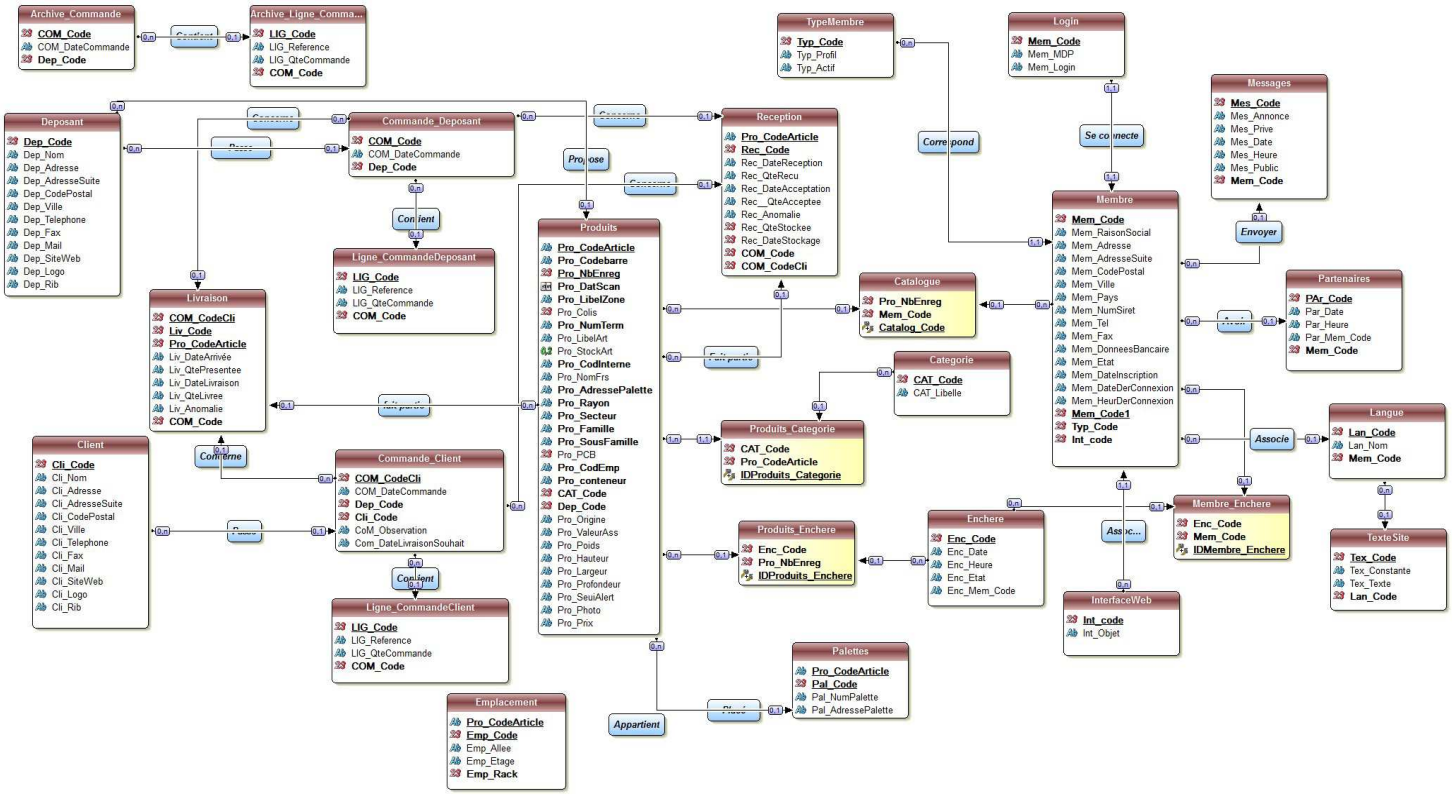


Récapitulatif de la structure :
- CMembre cette classe servira à la gestion des données des membres.
- CProduit cette classe contiendra les méthodes d'implémentations des objets produits (un partenaire accepté aura accès au catalogue produit du fournisseur l'ayant accepté) CValidation contiendra les méthodes permettant la validation d'une commande client CCaddy contiendra les méthodes liées à la constitution de la commande

Dans le cas de la mise en place d'un module de paiement en ligne en prendra soin d'ajouter une classe supplémentaire qui pourra s'appeler CPaiement :



Visualisons le modele MERISE (MLD) :



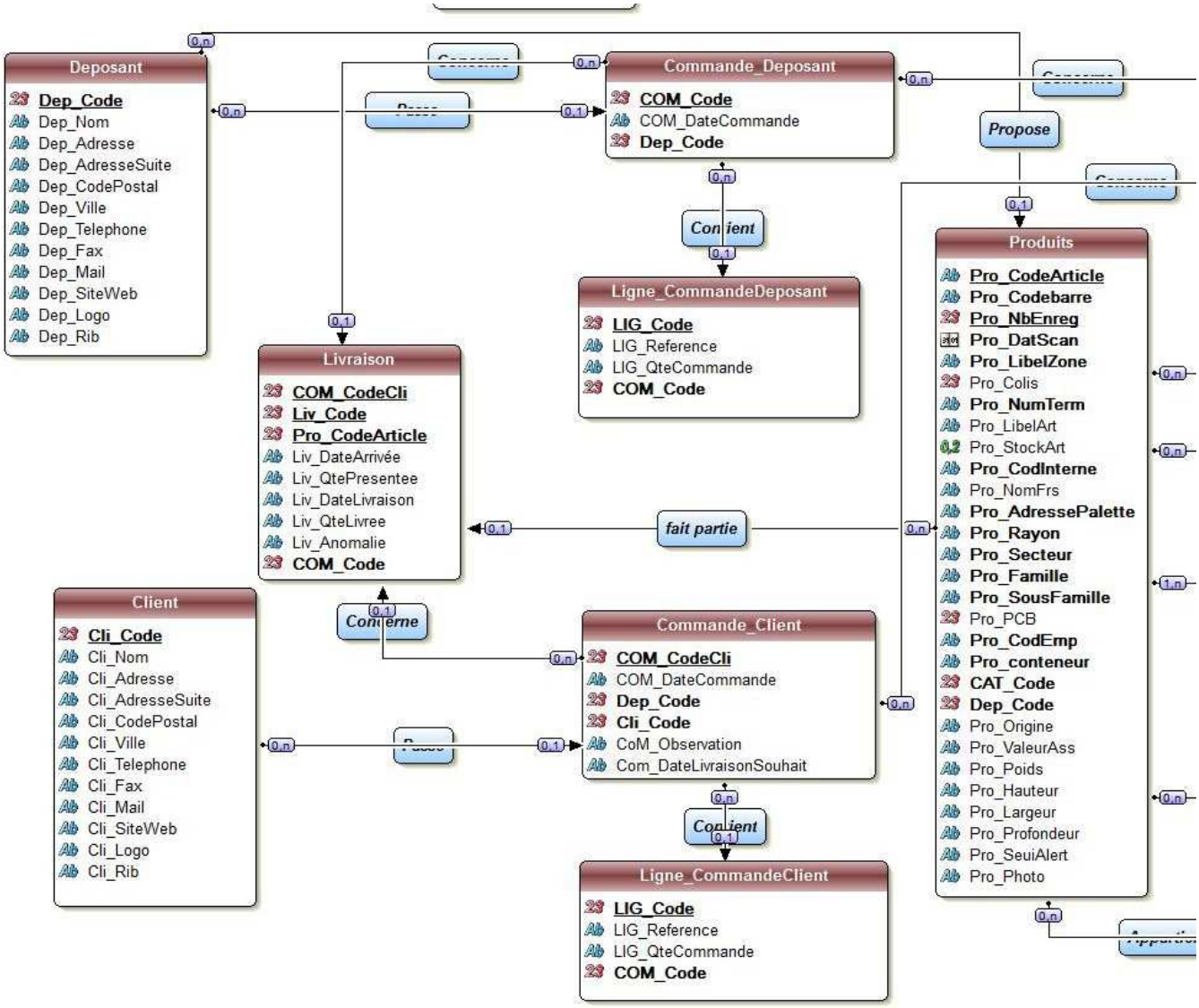
Quelques règles des contraintes d'intégrités :

- 1 produit est proposé par 0 ou n déposant
- 1 déposant propose 0 ou n produit
- 1 produit fait partie du catalogue de 0 ou n membre
- 1 membre agrège son catalogue avec 0 ou n produit
- 1 produit appartient à 1 ou n catégorie
- 1 catégorie concerne 0 ou n produits
- 1 produit est placé sur 0 ou 1 palette
- 1 palette contient 0 ou n produits
- 1 produit appartient à 0 ou 1 emplacement
- 1 emplacement est occupé par 0 ou n produits
- 1 produit fait partie de 0 ou 1 réception
- 1 réception contient 0 ou n produits
- 1 produit fait partie de 0 ou n livraison
- 1 livraison concerne 0 ou n produits
- 1 déposant peut passer 0 ou n commandes
- 1 commande est associée à 0 ou 1 livraison
- 1 livraison concerne 0 ou n commandes
- 1 client peut passer 0 ou n commandes
- 1 commande est associée à 1 et 1 seul client
-

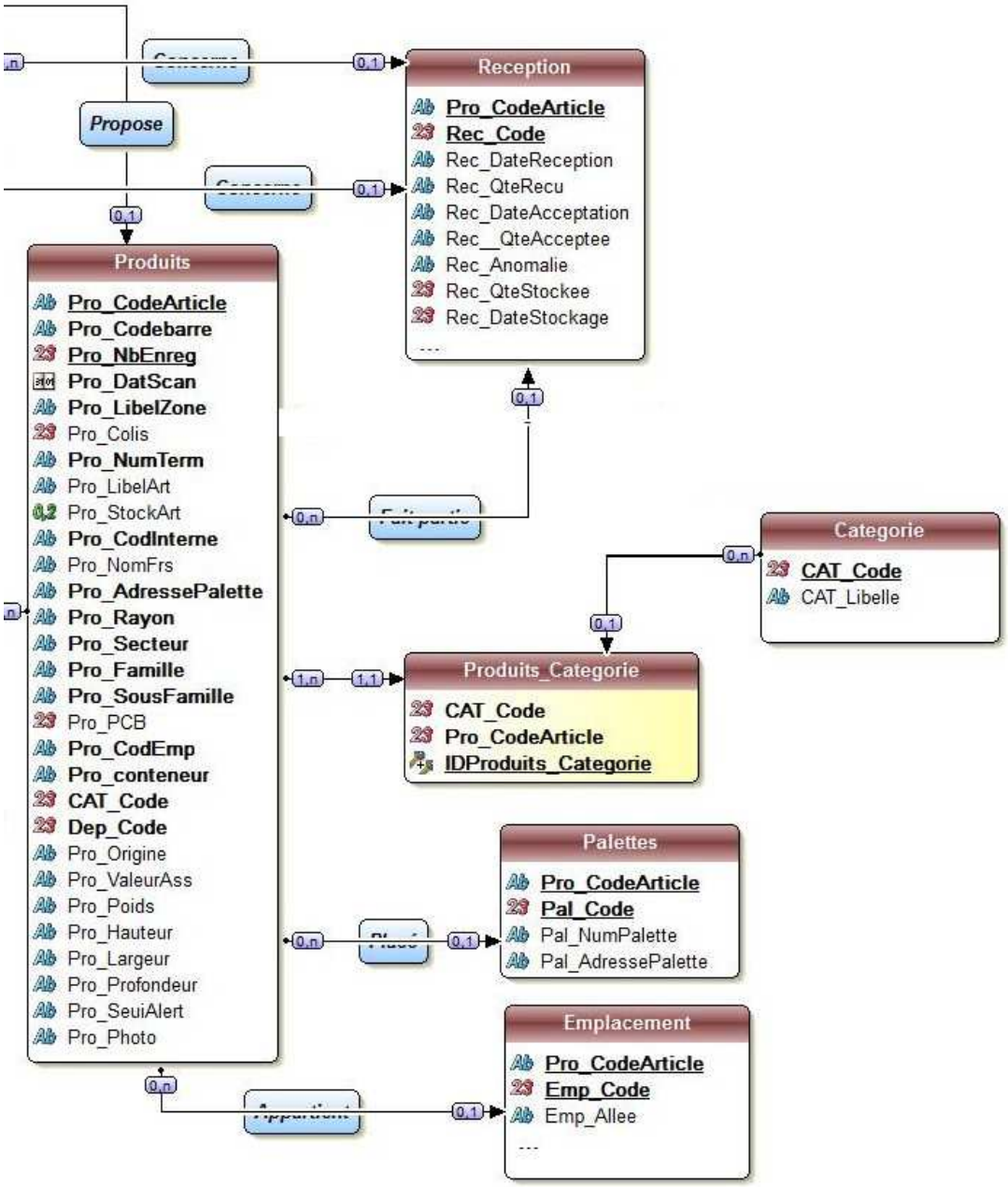
En Jaune la table « produit_categorie » servira à gérer la liaison entre les produits et leurs catégories. Idem pour catalogue avec la table « Membre »

- Validation_Commande (ValCom_Code, ValCom_date, ValCom_etat, ValCom_details, ValCom_infoMembre, ValCom_commentaire, ValCom_montant). Cette table enregistrera les informations de validation des commandes.
- Caddy (IdCad, prix, qte), contient la liste des produits sélectionnés par le client (membre).

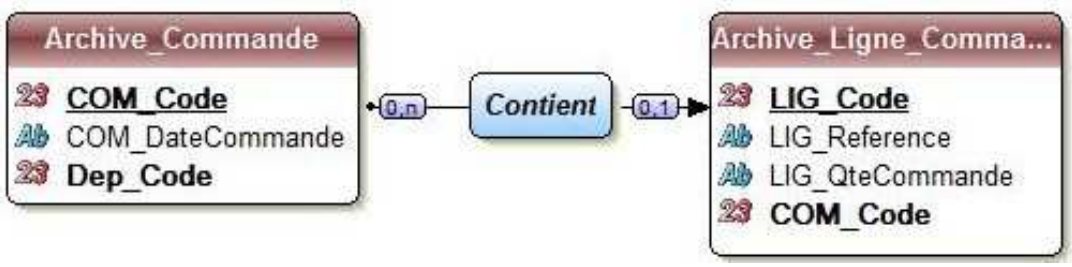
Gestion des commandes :



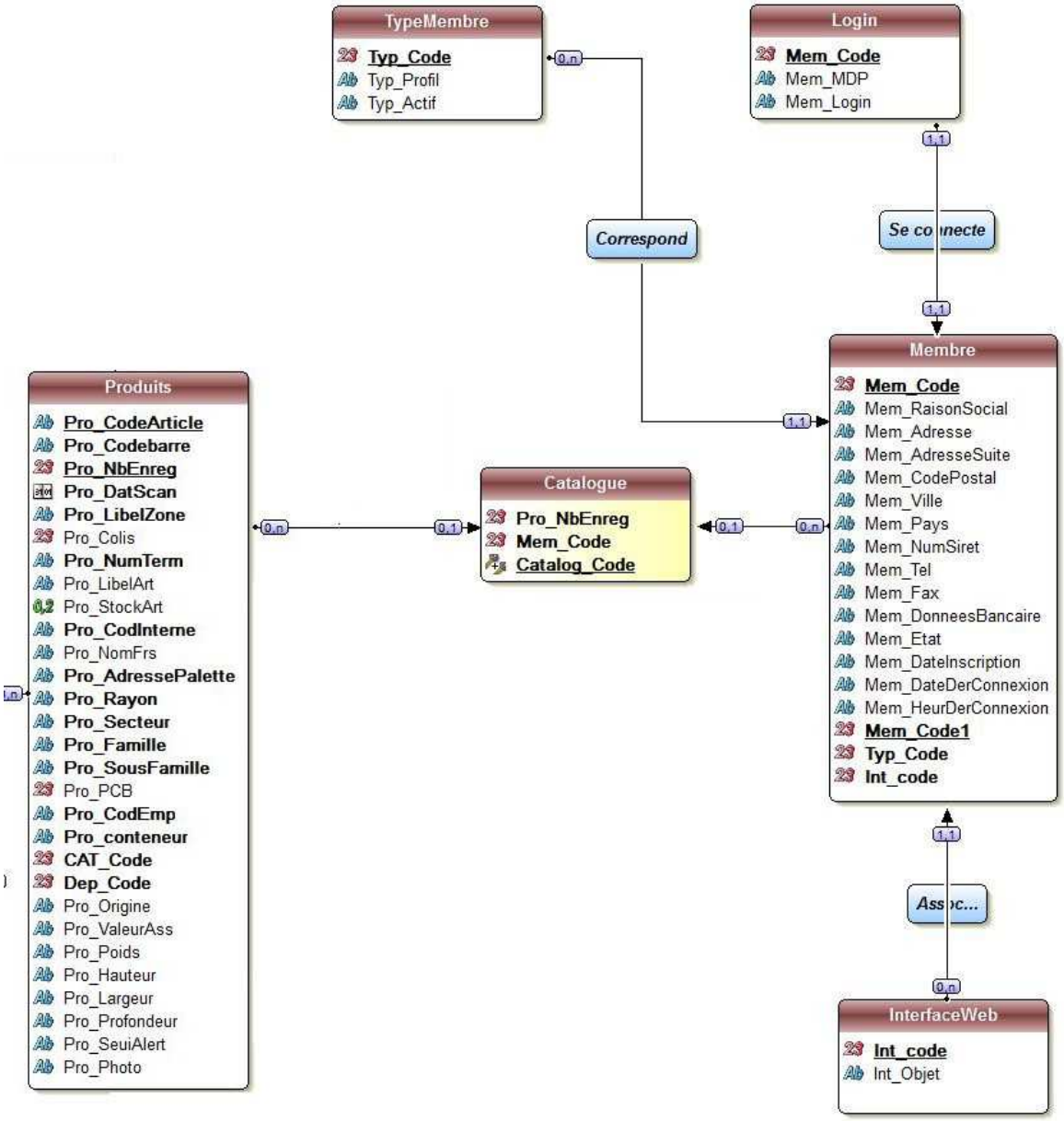
Gestion des réceptions :

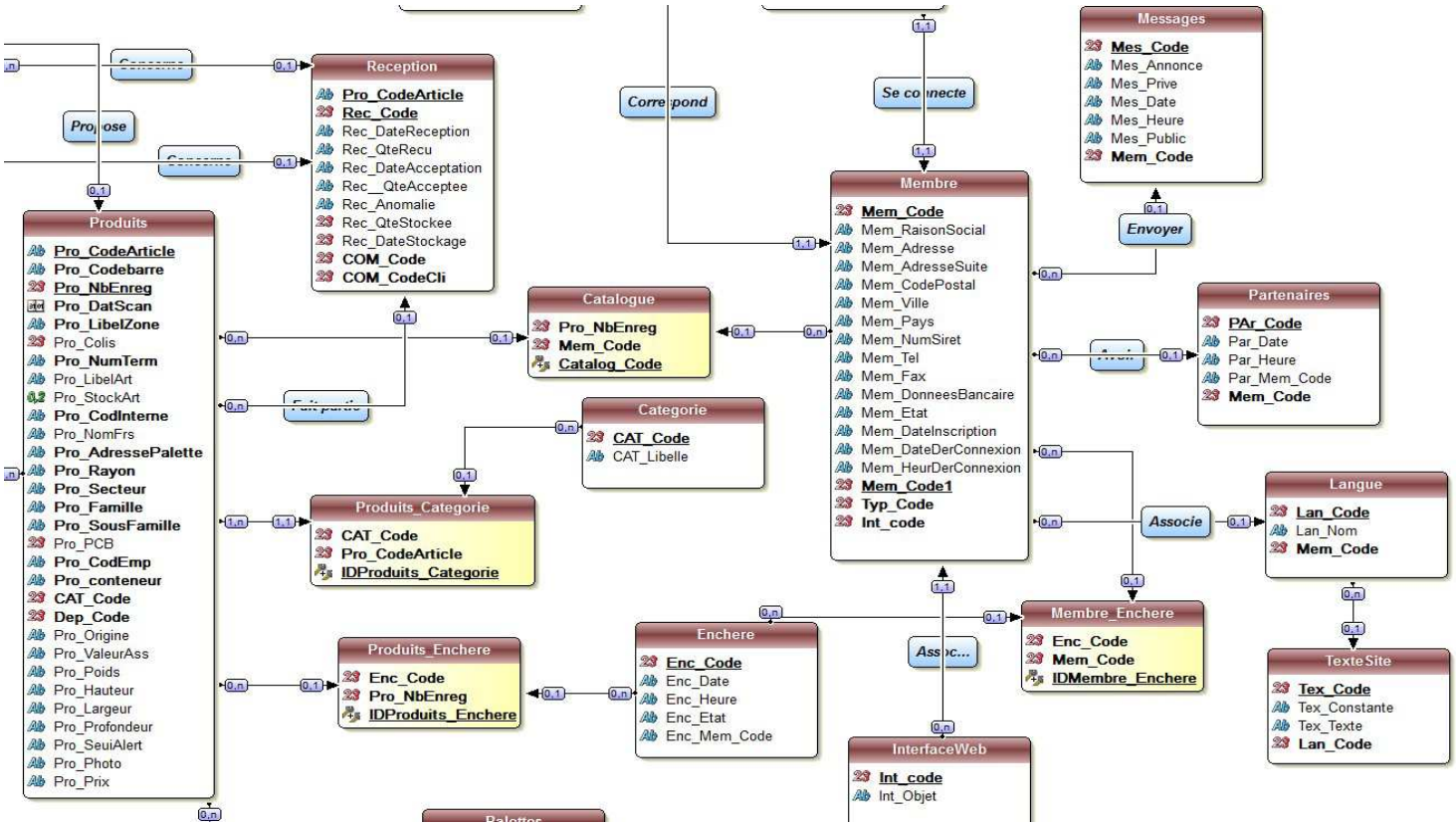


Gestion de l'archivage des commandes :



Gestion des catalogues de produits et des connexions au site par les différents types de membres :





Gestion des partenaires et des enchères

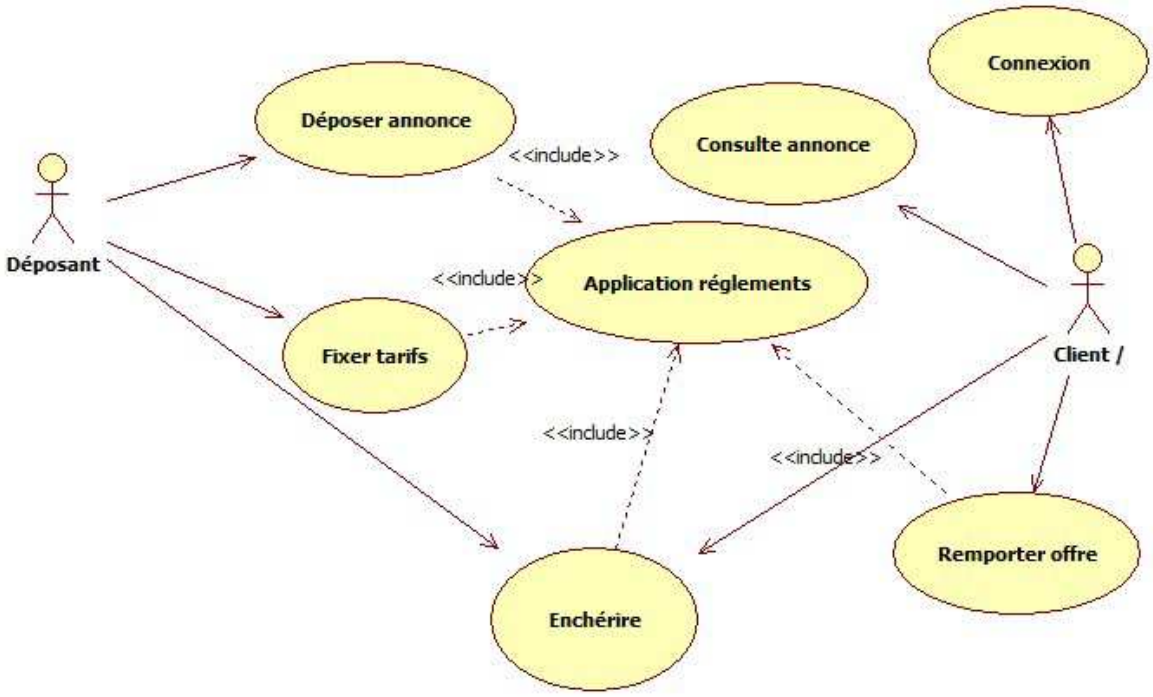
Pour les enchères on mémorisera le code du membre ayant remporté la mise ainsi que l'heure et la date. Une occurrence Enc_Etat indiquant si l'enchère est terminée, en cours ou en attente.

La table partenaires servira à mémoriser les partenaires validés par les déposants (avec également heure et date de confirmation).

Multi-Langage

A partir de la table TexteSite, on pourra stocker les différentes traductions qui seront disponibles au niveau du site. La table Langue contiendra les langues disponibles. L'occurrence Tex_Constante servira à faire le parallèle avec la constante présente dans les fichiers associés au langage (français.php, anglais.php…)

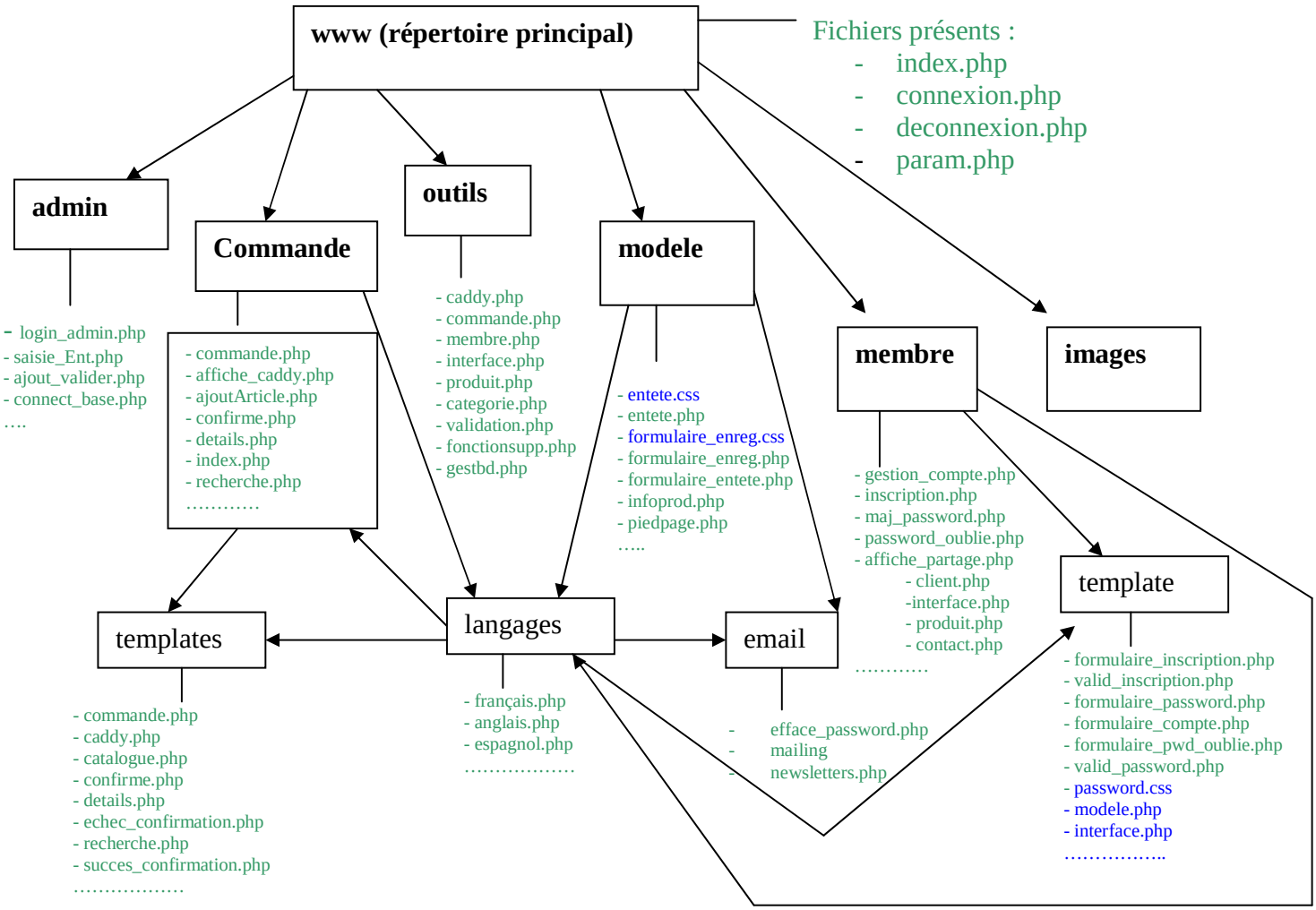
Pour les enchères nous pouvons établir le diagramme des cas d'utilisation ci-dessous pour représenter l'interaction des éléments :



NB : Le modèle logique des données a été élaboré dans l'AGL Windev. Tandis que les modèles UML ont été élaborés sous StarUML.

6) Architecture du site, modele MVC

Présentation de l'arborescence du site :



Dossier Admin : contient les scripts php permettant à l'administrateur de gérer le contenu du site. Il s'agira d'un accès sécurisé et non accessible depuis la page d'authentification principale du site.

- *login_admin.php gérera la connexion*
- *saisie_Ent.php permettra la création des entrepôts dans la base de données*
- *ajout_valider.php quand à lui validera les demandes d'inscriptions au site.*

Dossier Outils : dans ce dossier on pourra placer l'ensemble des scritps contenant les descriptions de classes à partir desquelles on génèrera les objets à manipuler au niveau du site.

- *commande.php contient les spécifications de la classe CCommande*

- caddy.php contient les spécifications de la classe CCaddy
- membre.php contient les spécifications de la classe CMembre
- interface.php contient les spcéfications de la classe CInterface
- produit.php contient les spcéfications de la classe CProduit
- categorie.php contient les spcéfications de la classe CCategorie
- validation.php contient les spcéfications de la classe CValidation
- index.php fichier contenant les appels de classes
- recherche.php moteur de recherche sur la base de données

Dossier Commande : contient les scripts permettant aux clients ainsi qu'aux déposants de créer, consulter et valider les commandes.

- commande.php pour la gestion des commandes
- affiche_caddy.php affichage du caddy lors de la phase de création de commandes
- ajoutArticle.php séléction des articles et ajout au caddy
- confirme.php confirmation de la commande
- details.php visualisation du detail de la commande
- index.php contient les appels aux différentes fonctions
- recherche.php moteur de recherche articles, fournisseur…

Dossier Membre : contient les scripts permettant de gérer les interfaces de connexion et de visualisation du site.

- gestion_compte.php affiche les informations du compte de l'utilisateur
- inscription.php gère les inscriptions du site en fonction du type choisi
- maj_password.php le membre peut modifier les informations de connexion au site
- password_oublie.php en cas d'oublie le mot de passe peut être renvoyé par email
- affiche_partage.php permet au membre de visualiser les données de ces partenaires (héritage lors de l'association à un membre).

Dossier Modèle : on rangera dans ce dossier les fichiers css du site. Ainsi que les fichiers gérant les newsletters ainsi que la gestion des emails aux membres du site.

Les dossiers Commande et Membre pourront contenir des sous-dossiers Templates contenant les habillages des différents types d'écrans pouvant être générés.

On placera dans le dossier Image l'ensemble des images liées au site.

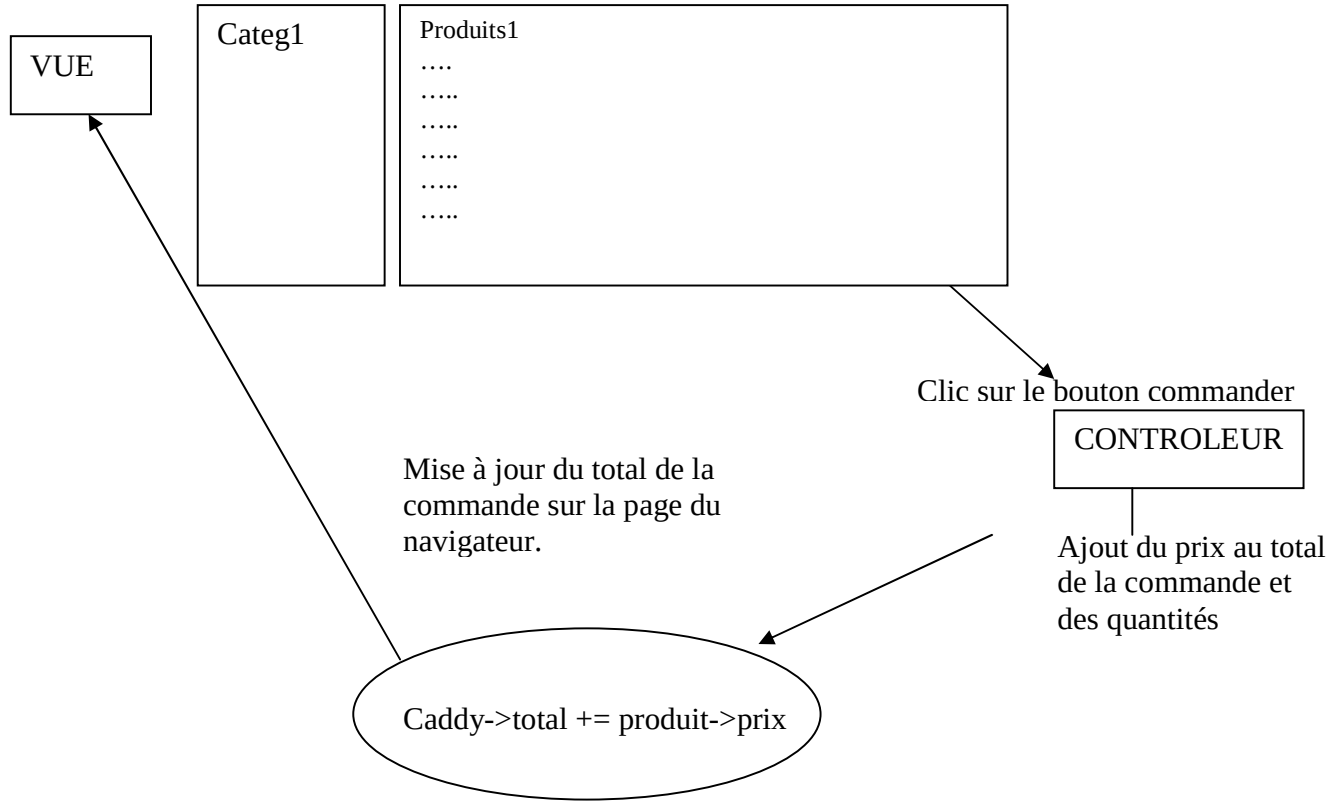
Le fichier fonctionsupp.php présent dans le répertoire outils contient des fonctions open source facilitant certaines vérifications de syntaxe notamment. L'arborescence générale du site s'inspire des solutions de site e-commerce open source ou autres CMS, de plus l'utilisation des classes permettrait d'implémenter la programmation objet et ainsi faciliter le transfert de données.

Dossier Langage :

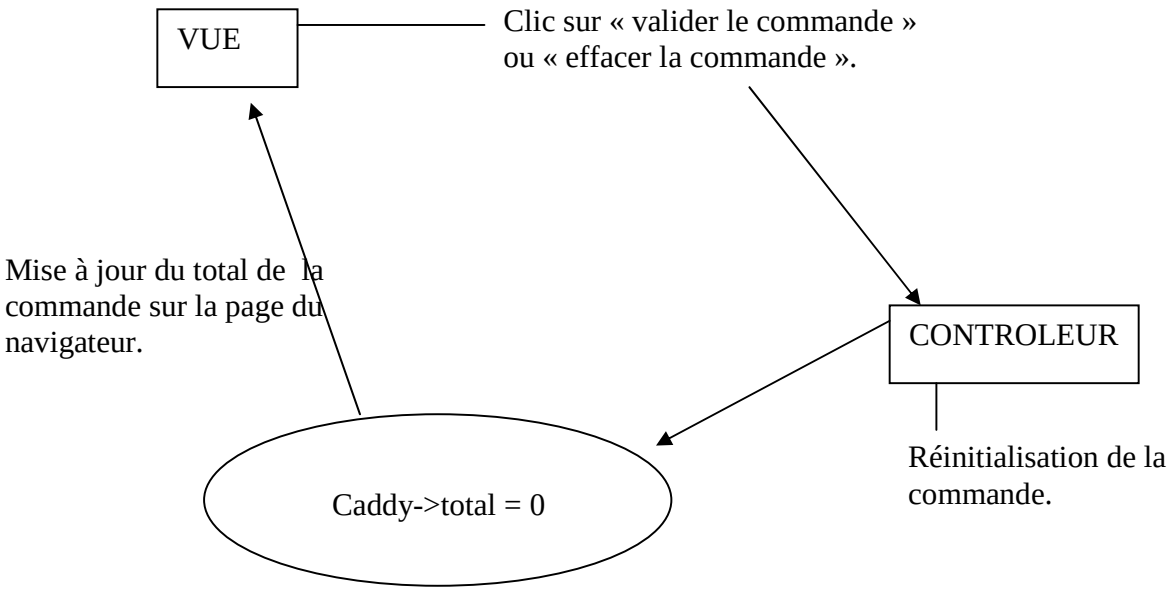
Le site étant multi-langage, il faudra définir au niveau de l'index des différents modules qui composeront le site (Modèle, Commande…) un appel vers le fichier contenant le langage associé au membre. Les fichiers langages (français.php, anglais.php…) contiendront deux types de déclarations :

- les constantes fixes ; ce seront les valeurs fixes en rapport avec le site (ex. : « #CLIC = « Cliquez ici » ») qui n'évolueront pas dans le temps.
- les constantes dynamiques ; par constante dynamique il faut comprendre une déclaration fixe (ex. : « Message_Membre » qui sera variable dans le temps par rapport à son affectation provenant du contenu d'une table (voir modèle Merise). Il sera alors possible de faire une mise à jour du contenu dynamique du site à partir des données stockées dans une table.
-

Modèle MVC pour l'ajout de produits dans la commande :



Modèle MVC pour traitement commande (en cas d'annulation ou de validation) :



6.1) Présentation de la classe CCaddy et CMembre pour implémentation.

Classe CCaddy :

```
<?
/* caddy.php  */

class CCaddy {
      var $element;     /* tableau des éléments produits*/
      var $total;       /* total du caddy */

      function Cadd() {
      /* Constructeur */
            $this->init();
      }

      function init() {
      /* permet d'initialiser le caddy */
            $this->element = array();
            $this->total = 0;
      }

      function ajout(&$idprod, $qte) {
      /* ajout d'un produit au caddy et recalcul du prix total*/
            if (isset($idprod)) {
                  setdefault($this->element[$idprod], 0);
                  $this->element[$idprod] += $qte;
            }
```

```
	}

	function memo_qte(&$idprod, $qte) {
	/* mémorise la quantité d'un produit dans le caddy */
		if (isset($idprod)) {
			$this->element[$idprod] = (int) $qte;
		}
	}

	function supprime(&$idprod) {
	/* supprime un produit du caddy */
		if (isset($idprod)) {
			unset($this->element[$idprod]);
		}
	}
	function nettoyer() {
	/* permet de supprimer les produits du caddy dont la quantité est
inférieur à 1 */
		foreach ($this->element as $idprod => $qte) {
			if ($qte < 1) {
				unset($this->element[$idprod]);
			}
		}
	}
	function NbElement() {
	/* retourne le nombre de produits uniques ajoutés au caddy */
		$nbProd = 0;
		foreach ($this->element as $idprod => $qte) {
			$nbProd += $qte;
		}
		return $nbProd;
	}

	function list_produit() {
	/* retourne la liste des produits du caddy */
		$list_idprod = "";
		foreach ($this->element as $numprod => $qte) {
			$list_idprod .= ",'" . $numprod . "'";
		}
		/* on supprime la virgule */
		return substr($list_idprod, 1);
	}
	function calcul_total() {
	/* Recalcul le total du caddy après chaque opération (ajout,
suppression...) */
		$this->total = 0;
		$Liste_id = $this->list_produit();
		if (empty($Liste_id)) {
			return;
		}
		$requete = bd_requete("SELECT Pro_CodeArticle, Pro_prix FROM
produit WHERE Pro_CodeArticle IN ($Liste_id)");
		while ($produit = bd_parcour_objet($requete)) {
			$this->total += $this->element[$produit-> Pro_CodeArticle] *
$produit->Pro_prix;
		}
	}
```

```
}
?>
```

Nous pouvons définir la classe CMembre en nous basant sur les mêmes spécifications de base :

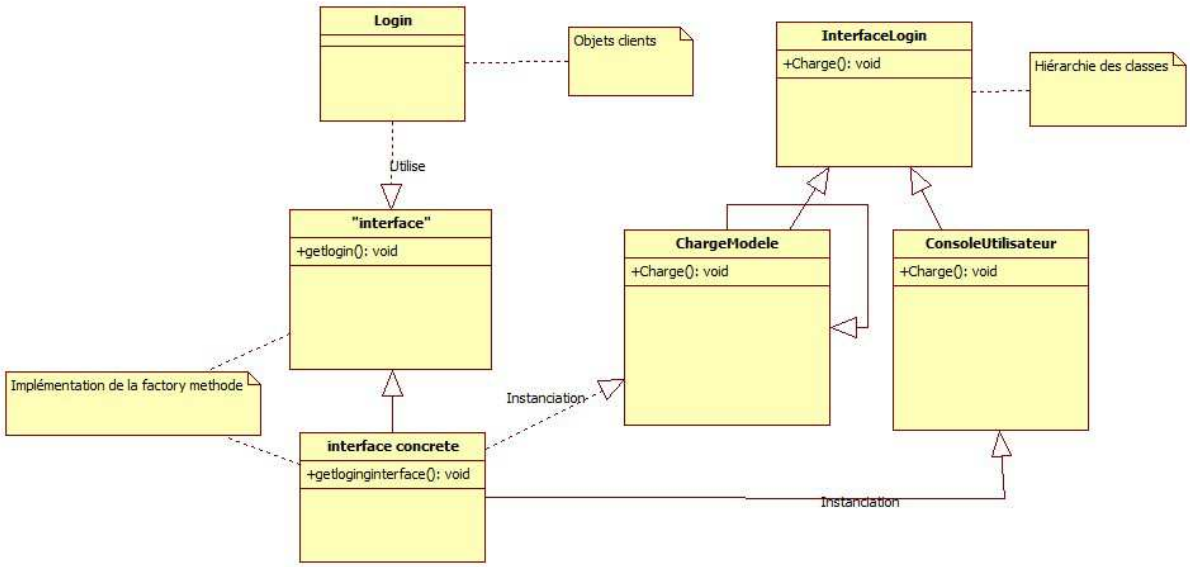
```
<?
/* membre.php  */

class CMembre {
      var $element;     /* tableau des éléments type utilisateur*/
      var $interface;   /* interface sélectionnée */

      function Membre() {
      /* Constructeur */
            $this->init();
      }

      function init() {
      /* permet d'initialiser le membre */
            $this->element = array();
            $requete = bd_requete("SELECT Int_Objet FROM interface WHERE
            InterfaceWeb.Int_Code=Membre.Int_Code");
            $this->interface = $requete;
      }

…….
```

7) Présentation de l'interface utilisateur

7.1) Structure de l'interface Web (utilisation de design pattern)

Compte tenu de l'aspect modulaire pour l'affichage de la page d'accueil en fonction du type du membre connecté (déposant, client,…), le choix de l'utilisation d'un design pattern pourrait s'avérer judicieux en termes de programmation objet, et limiterait les traitements à ce niveau.

L'un des pattern les plus connus pour ce type d'accès est la « factory method ». Ci-dessous le diagramme de classe présentant son implémentation :



Description

Dans une hiérarchie de classes, Toutes les sous-classes héritent des méthodes implémentées dans la classe mère. Ces sous-classes pourraient redéfinir ces méthodes pour les adapter et offrir des fonctionnalités différentes.

Quand on connaît exactement la fonctionnalité que l'on a besoin, on pourrait instancier directement la bonne classe dans la hiérarchie des classes qui offre cette fonctionnalité.

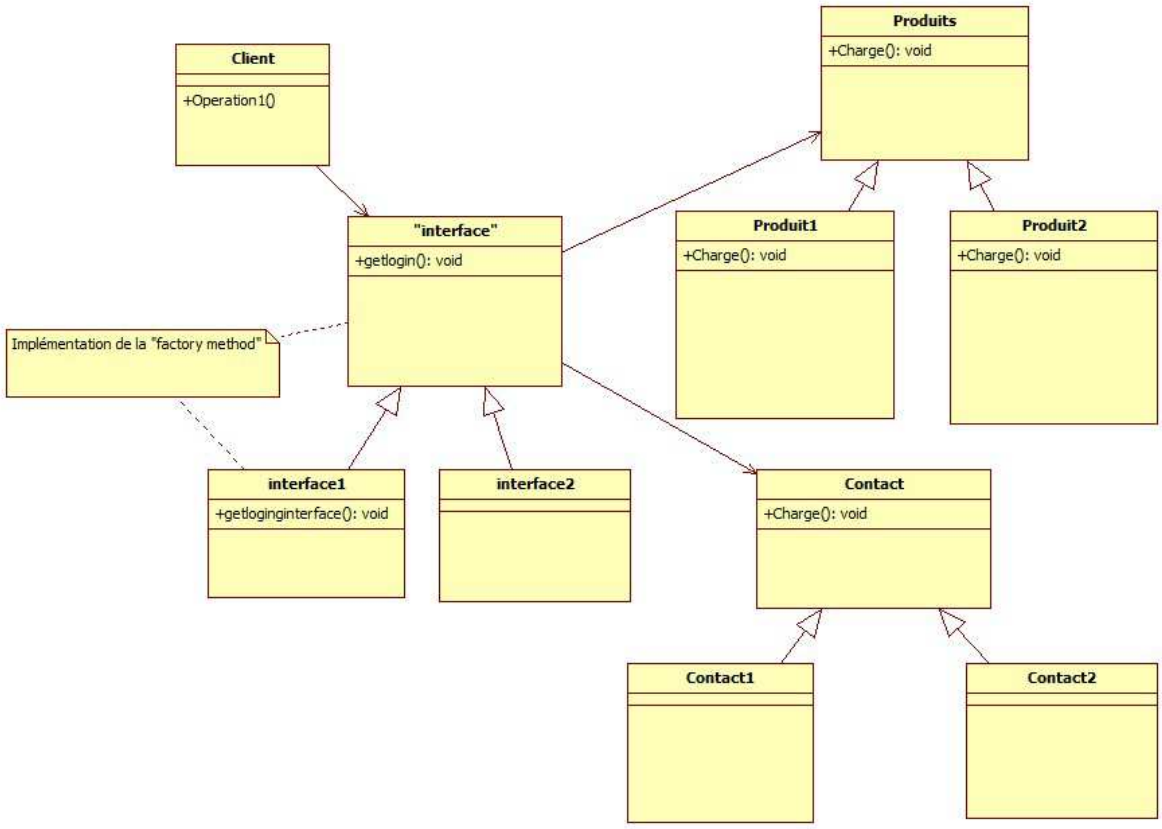
Or, parfois on ne peut pas connaître à l'avance la classe dont on aura besoin dans la hiérarchie.
Le choix pourrait dépendre de plusieurs facteurs comme:
-L'état d'exécution de l'application.
-Paramètres de configuration de l'application.
-prévision d'éventuel amélioration et évolution de l'application.

Dans ce cas, La conception normale consiste à implémenter un traitement de sélection de la bonne classe à instancier de la hiérarchie de classe.

En implémentant le Factory Method Pattern on bénéficie de plusieurs avantages:

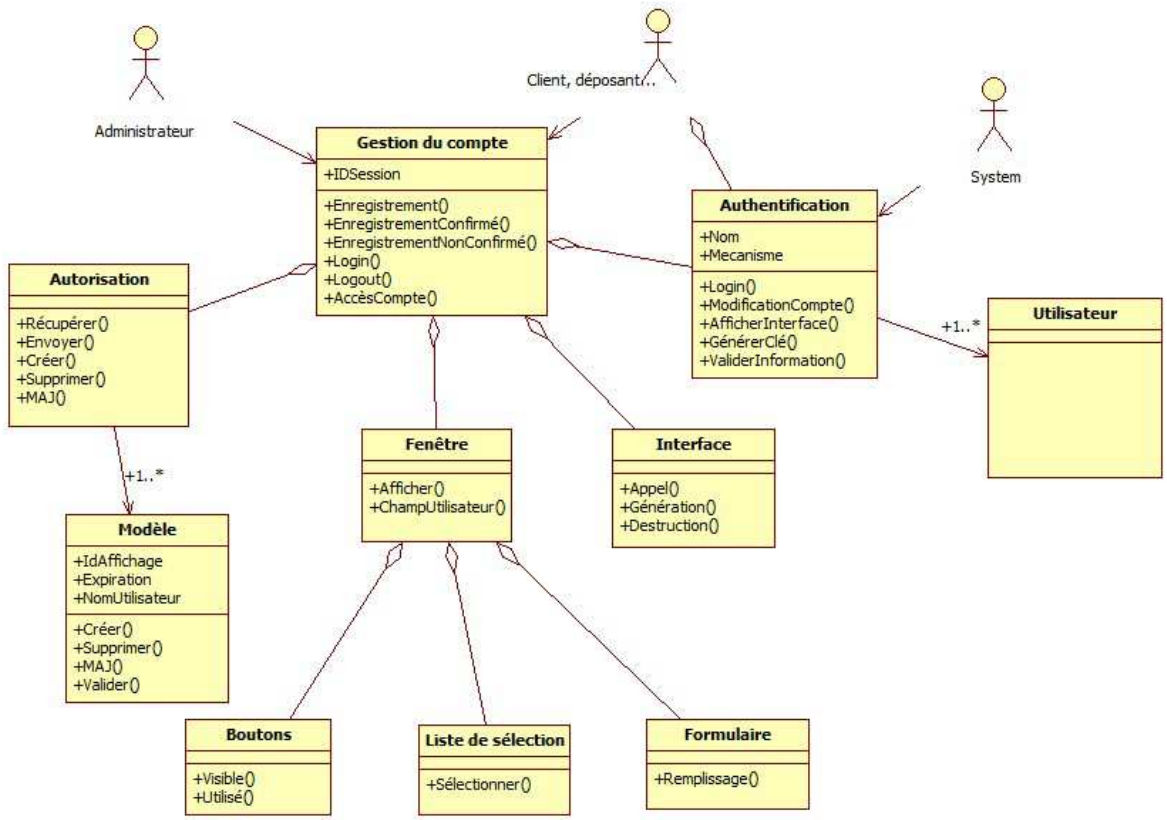
-L'application pourra utiliser la "Factory method" pour accéder à la bonne instance de classe et cela élimine le faite que les objets d'application implémentent le traitement de sélection de la classe.
-"Factory method" cache toute la complexité du choix de la bonne classe à instancier.
-Puisque la "Factory method" retourne l'instance de la classe mère alors les objets d'application n'ont pas besoin de connaître l'existence de la hiérarchie de classe.

Appliquons ce diagramme à l'héritage lié aux liaisons des comptes utilisateurs :



Un des avantages de la « factory method » est qu'un objet qui existe peut être réutilisé et ceci évite d'avoir à repasser par un constructeur pour en créer un nouveau. Tous les objets générés par la « factory method » peuvent

être nommé différemment, ce qui n'est pas le cas en utilisant la méthode des constructeurs.



Exemple d'implémentation de méthodes objets afin de générer une interface utilisateur spécifique.
Description de l'objet « Gestion du compte » :

```
<?
/* gestion_compte.php */
Class GestCompte{
      function est_connecte() {
            global $MEMBRE;
            if(session_is_registered('idmembre')){
                  return true;
            }
            else {
                  return false;
            }
      }
      /*    return isset($MEMBRE["membre"])
                  && !empty($MEMBRE["membre"]["pseudo"])
                  && nvl($MEMBRE["ip"]) == $_SERVER["REMOTE_ADDR"];
      } */
```

```
function authentification_membre() {
/* permet de controler l'authentification d'un membre */
      global $ConfigSite, $MEMBRE;
      if (! est_connecte()) {
            redirect("$ConfigSite->wwwroot/connexion.php");
      }
}
function construction_categorie_arbo(&$sortie, &$selection, $parent=1,
$ind="") {
/* fonction récursive permettant la construction de l'arborescence des
catégories */
      $idreq = bd_requete("SELECT Cat_Code, Cat_Libel FROM categorie
WHERE idparent = $parent");
      while ($cat =  bd_parcour_objet($idreq)) {
            $select = in_array($cat->idcat, $selection) ? "selection" :
"";
            $sortie .= "<option value=\"" . ov($cat->idcat) . "\"
$selection>$ind" . ov($cat->nom);
            if ($cat->idcat != $parent) {
                  construction_categorie_arbo($sortie, $selection, $cat-
>idcat, $ind."  ");
            }
      }
}
function affiche_produit($id) {
/* affiche les détails d'un produiit */
      $idreq = bd_requete("
      SELECT
             Pro_CodeArticle
            ,…
            ,…
      FROM
             Produits p
            ,produits_categorie pc
      WHERE p.Pro_CodeArticle = pc.Pro_CodeArticle
            AND p.CodeArticle = '$id'
      ");

      if ($idreq) {
            return bd_parcour_objet($idreq);
      } else {
            return false;
      }
}
function generateur_password($max=10) {
/* génère un mot de passe de façon aléatoire */
      $C = "bcdfghjkmnpqrstvwxz";
      $V = "aeiouy";
      $totalC = strlen($C)-1;
      $totalV = strlen($V)-1;
      $pw = "";
      while (strlen($pw) < $max) {
            $pw .= substr($C, rand(0, $totalC), 1)
                  . substr($V, rand(0, $totalV), 1)
                  . substr($V, rand(0, $totalV), 1);
      }
      $pw = substr($pw, 0, $max);
```

```
	$pw[rand(0, strlen($pw) - 1)] = rand(2, 9);
	return $pw;
}
function erreur(&$erreur) {

	if (isset($erreur)) {
		echo "<font color=#ff0000>&lt;&lt;</font>";
	}
}
function erreur2(&$erreur) {
	if (isset($erreur)) {
		echo "<font color=#ff0000>&gt;&gt;</font>";
	}
}
function Identifiant_Existe($Identifiant) {
/* test l'existence de l'identifiant dans le cas d'une création de
compte */
	$idreq = bd_requete("SELECT 1 FROM Membre WHERE Mem_RaisonSocial =
'$Identifiant");
	return bd_nbenreg($idreq);
}
function existe_email($email) {
/* test l'existence de l'adresse email */

	$idreq = bd_requete("SELECT 1 FROM Membre WHERE Mem_email =
'$email'");
	return bd_nbenreg($idreq);
}
function efface_password($Identifiant) {
/* function permettant d'effacer le mot de passe et d'envoyer le nouveau
par email */
	global $ConfigSite; // $ConfigSite est une variable contenant les
paramétre d'affichage de l'interface web pour le membre connecté.
	/* récupère les informations du membre */
	$idreq = bd_requete("SELECT * FROM Membre WHERE Mem_RaisonSocial =
'$Identifiant");
	$user = bd_parcour_objet($idreq);
	$result = mysql_fetch_array($idreq);
	/* effacement du mot de passe */
	$nouvpassword = generateur_password();
	$idreq = bd_requete("UPDATE Login SET Log_MDP = '" .
md5($nouvpassword) ."' WHERE Mem_Login = '$Identifiant'");
	/* envoi du mail au membre avec le nouveau mot de passe */
	$data = new Object;
	$data->Identifiant = $Identifiant;
	$data->nomComplet = $result["Mem_RaisonSocial"];
	$data->nouvpassword = $nouvpassword;
	$data->support = $ConfigSite->support;
	$emailbody = read_template("$ConfigSite-
>templatedir/email/efface_password.php", $data);
	$Adremail = $result["Mem_email"];
	mail(
		"$data->nomComplet <$Adremail>",
		"",
		$emailbody,
		"From: $data->support");
```

```
	}
```

// Fonction Qui sera utile en cas de mise en place d'un système de paiement en ligne sur la plateforme B2B

```
	function cache_num($cbnum) {
	/* fonction pour cacher une partie du numéro de carte bancaire, ceci
	pour des raisons de confidentialité */
		return substr($cbnum, 0, 4) . "..." . substr($cbnum, -4);
	}
	function enreg_info(&$frm) {
		global $MEMBRE;
		$paye = new Object();
		$paye->acheteur = $frm["acheteur"];
		$paye->contact = $frm["contact"];
		$paye->adresse = $frm["adresse"];
		$paye->cartebancaire = $frm["cartebancaire"];
		$paye->dateexpire = $frm["dateexpire"];
		$paye->commentaire = $frm["commentaire"];
		$_SESSION["infoMembre"] = &$paye;
	}
	function charge_info() {
		global $MEMBRE;
		if (empty($_SESSION["infoMembre"])) {
			return false;
		} else {
			return $_SESSION["infoMembre"];
		}
	}
	function efface_info() {
		global $MEMBRE;
		unset($_SESSION["infoMembre"]);
	}
	function affiche_details() {
	/* affiche les détails d'une liste de produits */
		global $ConfigSite;
		$idreq = bd_requete("SELECT Pro_CodeArticle, Pro_LibelArt, …, FROM
	Produits");
	}
	……….
}
```

Choix d'action sans identification :

- Consultation actualités
- Inscription newsletter
- Présentation
- S'inscrire sur le site

Ecran d'identification : login + mot de passe

Choix d'action après identification :

- Consultation du compte (modification informations)
- Moteur de recherche de produits
- Modification du mot de passe
- Consultation du catalogue des produits
- Consultation des commandes
- Validation ou annulation des commandes
- Déconnexion

…..(voir description menu dans cahier des charges fonctionnel pixell)

7.2) Principe de l'interface de mise à jour (admin)

L'interface Web développée utilise une base de données MySQL. Pour l'alimenter, le choix de PHP c'est imposé.

Un parseur de données développé sur la base du standard EDI s'occupera de mettre la base de données Web à jour et remontera à une heure déterminé les commandes vers le système de gestion du fournisseur.

L'accès à cette interface de saisie est protégé par une clé de session ou un serveur HTTPS.

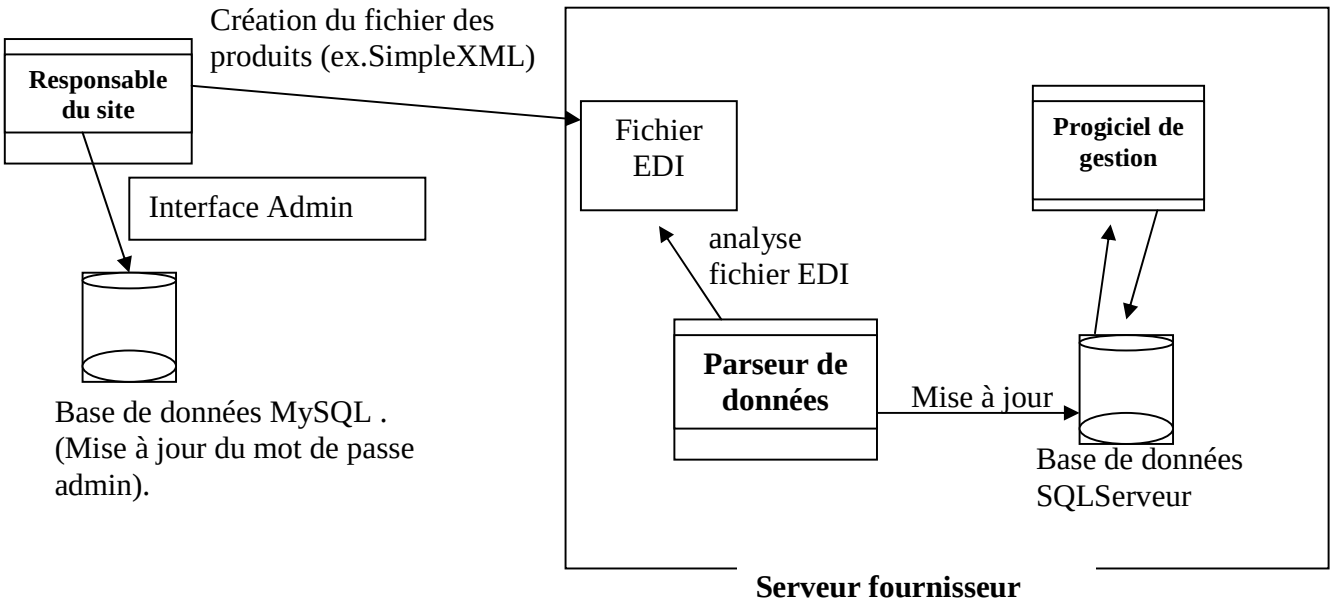
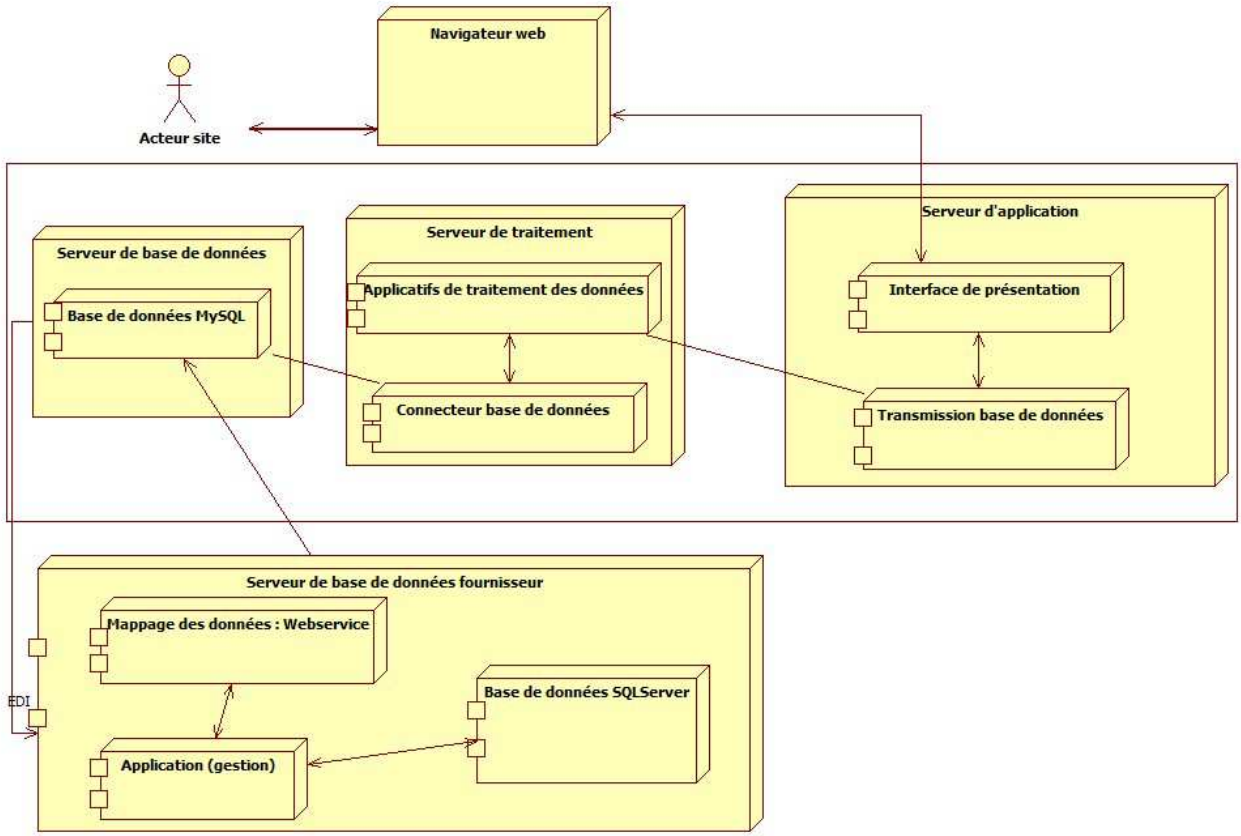


Diagramme de déploiement :



Sur le diagramme de déploiement ci-dessus, nous pouvons voir schématiquement les interactions entre les différents modules composant le site B2B

8) PRESENTATION DE DRUPAL

(Référence : encoreungeek.com)

La force de Drupal vient de son aspect modulaire, qui vous permets de n'ajouter que les fonctions que vous avez besoin, au fur et à mesure que vous en avez besoin. Si cela peut sembler déconcertant au début, vous ne pourrez pas y échapper car cela fait vraiment partie de la philosophie de Drupal, c'est à dire que chaque module apporte une fonction bien précise, et que la fonctionnalité d'un site se créer en ajoutant plusieurs modules qui travaillent les uns avec les autres. Il n'est pas rare pour un site complexe d'avoir des dizaines de modules installés sans que cela ne vienne compromettre la vitesse du site ou sa flexibilité, au contraire!

Quelques modules intéressants :

CCK: Permets de créer n'importe quel type de contenu et y ajouter le nombre de champs que l'on veut

Taxonomy: Module qui permet de catégoriser le contenu du site en différentes catégories et sous catégories... Notez que les types de contenu ne sont pas des catégories comme sous la plupart des autres CMS, mais que dans Drupal, tous les types de contenu peuvent partager les mêmes catégories, ou avoir un ensemble de catégories séparées pour chaque type de contenu...

Views: Le module views est une application qui permet de construire des listes dynamiques de contenu basés sur des critères et des filtres. Idéal pour présenter tout le contenu d'une telle catégorie, le contenu le plus populaire, ou n'importe quelle autre liste semblable. Ce qui est intéressant c'est que le module views accepte les arguments, ce qui permet de créer une seule page qui peut s'occuper de tout le site par exemple. Par exemple si on crée une vue pour voir le contenu crée par un membre du site, la vue attends l'argument (le nom de l'usager) et génère la liste sur demande. La même vue sert donc à voir le contenu de n'importe quel membre, au lieu de devoir générer une liste par membre sur le serveur...

Locations + Gmap: Ces deux modules permettent de facilement assigner des informations géographiques à n'importe quel contenu. Idéal pour un site ou les gens peuvent indiquer leur localisation... On peut intégrer un 'champ' localisation sur n'importe quel type de contenu, afin de recherche du contenu par une carte google par exemple... Idéal pour les sites de petites annonces, de rencontres, de pages jaunes, etc...

Image: Le module image, comme son nom l'indique permets d'ajouter des fonctions de gestion des images et de galeries photos sur un site en Drupal. Idéal pour les blogs, surtout en collaboration avec les modules Img_assist et TinyMce, qui permettent d'intégrer facilement et rapidement des images dans le texte. Notez qu'il existe aussi un champ 'image' pour le module CCK qui permets lui aussi d'intégrer des images dans un texte avec Drupal.

Book module: Ce module permets de créer des 'pages collaboratives' à la wiki, afin de permettre aux membres d'un site de créer des guides avec une hiérarchie, mais aussi de collaborer de manière commune à l'écriture de contenu. Notez que Drupal compte un ensemble d'une dizaine de modules qui permettent relativement aisément de copier exactement le fonctionnement d'un wiki.

Calendar et Date API : Ces deux modules permettent de transformer Drupal en un véritable portail communautaire axé sur la promotion et présentation d'événements. Vous pouvez afficher vos événements sur des calendriers, ainsi qu'ajouter un champ 'date' à vos contenus CCK. Idéal pour créer un système de gestion de tâches en Intranet... (On peut facilement ajouter une date limite pour un projet par exemple... en installant un champ date sur un contenu CCK)...

TinyMCE: Le module TinyMCE permet d'intégrer l'éditeur WYSIWYG TinyMCE à n'importe quelle installation Drupal, pour le bonheur de vos utilisateurs qui ne veulent pas savoir comment utiliser des tags html de base... ou même du BBCode.

9) EVOLUTION DU SITE

En mettant en avant la conception objet dés le départ pour la création du projet, on évitera des surprises dans le cas d'une montée en puissance de la plateforme avec les demandes d'évolutions qui ne manqueront pas d'être demandées.

C'est pourquoi le site devra être le plus souple possible dans son architecture afin de permettre des évolutions rapides, voir de les anticiper dès le commencement.
Il est également important de prévoir si le site sera multi langage afin de bien préparer les modules correspondant.

L'interface d'administration du site ainsi que l'aspect modulaire des écrans d'accueil des différents membres qui se connecterons, constituent la phase la plus importante du projet, car il sera pénible de revenir sur ces analyses par la suite, d'où les coûts et les retards de livraison qui pourront en découdre.

10) ANNEXE

- **Administrateur** : Utilisateur ayant accès aux différentes fonctionnalités des différents comptes.
- **Attendu** : Arrivages annoncés (par un déposant ou par un entrepôt) pour un entrepôt en nombre de colis de chaque référence.
- **Bordereau d'expédition** : C'est comme une commande, mais dédiée au transit. Un bordereau d'expédition permettra de suivre l'avancée d'une expédition de bout en bout. Un peu comme un numéro de suivi Chronopost, mais avec le détail de la commande.
- **Commande** : Marchandises demandées par un point de vente ou par un déposant.
- **Compte** : Ensemble des ressources informatiques attribuées à un utilisateur. Il existe 5 types de compte : administrateur, dépositaire, déposant, point de vente, transitaire et validateur.
- **Consolidation** : Regroupement de marchandise dans une même unité de transport. En mettre le maximum.
- **Destinataire** : Entité qui reçoit la commande ou l'attendu (Entrepôt, déposant, point de vente)
- **Déposant** (Fournisseur) : Propriétaire des marchandises dans un entrepôt. C'est le client de l'entrepôt.
- **Dépositaire** : C'est la personne (utilisateur) qui gère l'entrepôt, ou du moins qui est en charge de la validation de la commande globale à destination de l'entrepôt.
  Dans la plateforme, il sera aussi en charge de la création des bordereaux d'expédition.
- **Design pattern** : En informatique, un patron de conception, motif de conception ou modèle de conception est un concept de génie logiciel destiné à résoudre les problèmes récurrents suivant le paradigme objet. Les trois termes sont des traductions de l'expression anglophone design pattern, tentant d'exprimer la réutilisabilité d'un concept qu'on retrouve dans les patrons et les motifs.
- **Entrée :** C'est la suite physique d'un attendu. Un attendu donne lieu à une entrée en stock.
- **Entrepôt (Dépôt)** : Un entrepôt est un bâtiment logistique destiné au stockage de biens en vue de leur expédition vers un client (interne ou externe à l'entreprise).
- **Groupage** : Regroupement d'un maximum de déposant dans une même unité De transport.
- **Point de vente** : Partenaire d'un fournisseur (producteur, grossiste, importateur, commerçant…). C'est le client du déposant, il lui passe commande.
- **Portefeuille de commande** : C'est l'endroit ou sont centralisées les commandes, à chaque niveau. Dans le portefeuille de commandes du déposant, on trouvera l'ensemble des

commandes des points de vente. Dans le portefeuille de commandes du dépositaire, on trouvera l'ensemble des commandes des déposants du ou des entrepôts affectés au dépositaire.

- **Référencer** : Décider qu'un article donné fera l'objet de futures commandes. (Mettre en favoris). Ca peut être 4 ou 5 articles sur 200 références disponibles. Le référencement est une opération très répandue, et effectuée par tous les acheteurs de la grande distribution.
- **Stock** : Ensemble de produits d'un déposant à l'intérieur d'un entrepôt.
- **Transitaire** : C'est l'entreprise qui organise la liaison entre les différents transporteurs et assure la continuité du transport, ainsi que toutes les opérations administratives connexes s'y rapportant (formalités de douane…)
- **Unité de transport** : Container, semi-remorques, palettes aériennes
- **Utilisateur** : Personne physique utilisant la plateforme. Un utilisateur est associé à un type de compte**.**
- **Validateur :** utilisateur validant les comptes déposants

Olivier EDWIGE

Digitally signed by Olivier EDWIGE
DN: cn=Olivier EDWIGE gn=Olivier EDWIGE c=Martinique l=MQ o=EDEO Technologies e=o.edwige@wanadoo.fr
Reason: I am the author of this document
Location:
Date: 2015-06-13 22:50-04:00