BERTRAND Julien - SIAME

DOUBRE Maxime - SIAME

DOUVRIN Sébastien - IGAI

Compte rendu MCO

AllBoardsShop

25 novembre 2020

R.OLIVEIRA

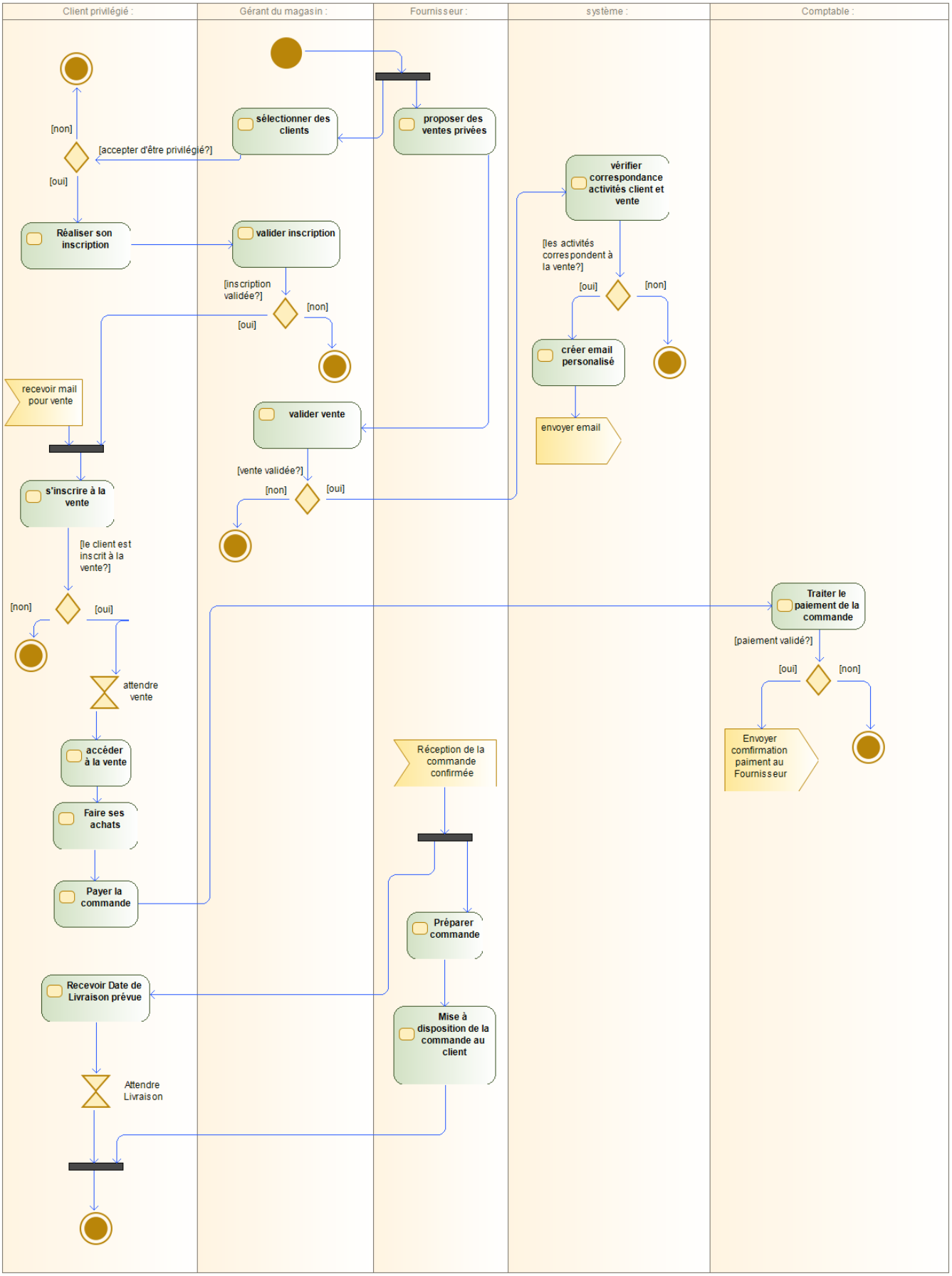
Au cours du TP et surtout dans le diagramme d'activité que nous avons fait en premier nous avons fait différents choix de modélisations.

Nous avons donc 6 diagrammes :

- Diagramme d'activité
- Diagramme de classes métier
- Diagramme de cas d'utilisation
- Diagramme de séquence système
- Diagramme de séquence détaillé
- Diagramme de classes participantes

Spécifications métier

Diagramme d'activité



Pour ce diagramme nous avons déroulé le sujet afin de créer les activités liées à chaque acteur.
De ce fait nous avons 5 acteurs :

- Le client privilégié
- Le gérant du magasin
- Le fournisseur
- Le système
- Le comptable

Nous avons jugé que nous devions créer un comptable car il est dit dans le sujet que les paiements se font par Paypal et il fallait don quelqu'un pour gérer les paiements effectués, ce que ne peut pas forcément faire le gérant du magasin.

Au début nous avions un acteur ''Client'' mais il n'avait qu'une seule activité qui était de s'inscrire en tant que client privilégié, nous l'avons donc supprimé et nous l'avons réduit à un seul acteur ''Client privilégié''.

Si on déroule le digramme, au début nous avons des activités en parallèle.

Le gérant du magasin sélectionne des clients et ces clients vont accepter ou pas d'être privilégié. S'ils acceptent ils vont alors s'inscrire puis leur inscription va être validée ou non par le gérant du magasin.

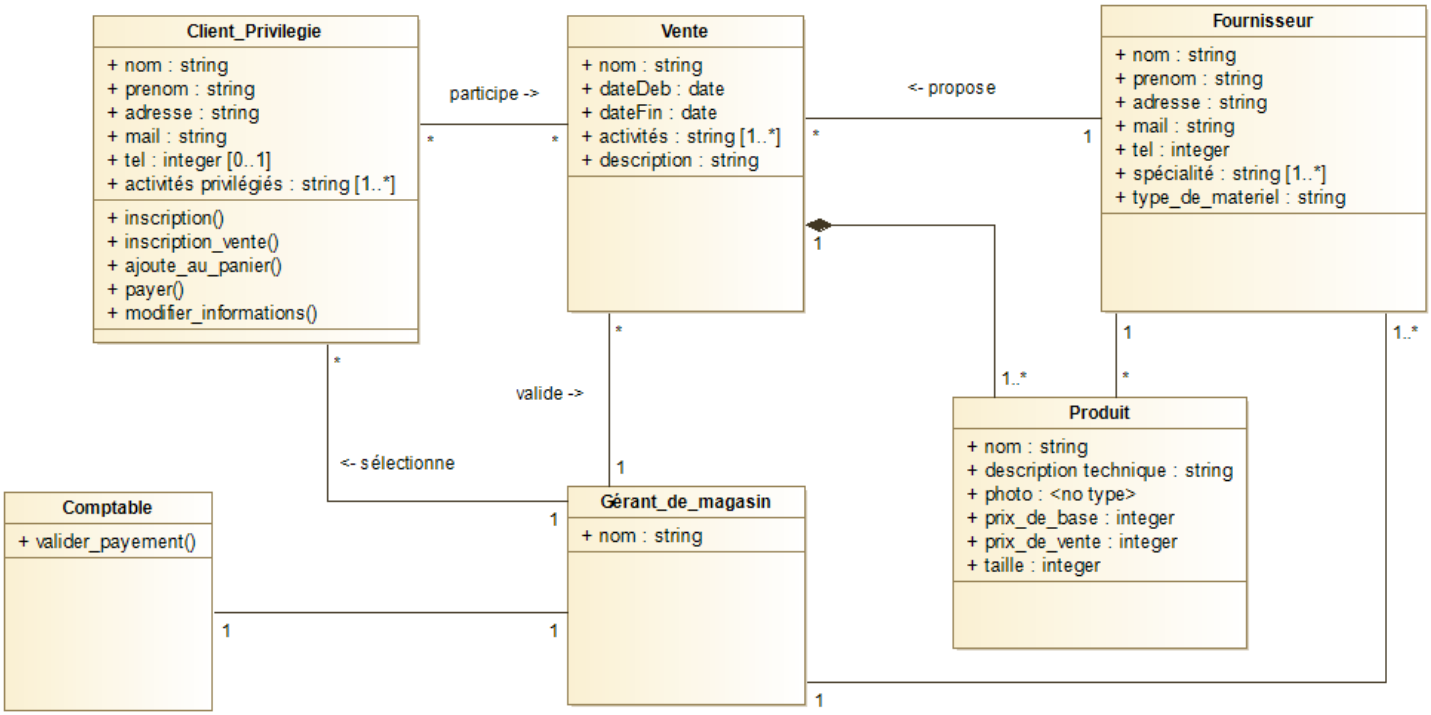
Dans el même temps, le fournisseur va proposer des ventes que le gérant du magasin va valider ou pas. Si elles sont validées le système va alors vérifier les correspondances des clients avec la vente en question et créer des emails personnalisés pour les clients en lien avec la vente. Il va ensuite envoyer les emails.

Une fois ces deux chemins faits en parallèle le client va pouvoir s'inscrire à la vente ou non.

S'il s'inscrit alors il va avoir un temps d'attente jusqu'à ce que la vente soit ouverte. Quand celle-ci sera ouverte il pourra y accéder, faire ses achats puis payer la commande.

A ce moment-là le comptable est chargé de valider le paiement. Si c'est le cas alors il envoie un signal au fournisseur qui, à réception de la commande confirmée, va dans le même temps envoyer un email au client de date de livraison et préparer la commande puis la mettre à disposition du client à date prévue.

Diagramme de classes métier



Dans ce diagramme de classes étier nous retrouvons nos 4 acteurs : Client privilégié, Fournisseur, gérant magasin et comptable.

On a en plus : vente et produit.

Dans la classe ''Client privilégié'' on a 6 attributs : le nom du client, son prénom, son adresse, son mail, son numéro de téléphone et ses activités privilégiées.

On trouve aussi des méthodes associées au client :

- Inscription() qui est la méthode servant à s'inscrire en tant que client privilégié
- Inscription_vente() qui sert à s'inscrire à une vente privée qui nous concerne et nous intéresse
- Ajoute_au_panier() qui sert à mettre un produit dans notre panier lors d'une vente
- Payer() qui sert à payer lors d'une vente une fois que l'on a finit de faire notre choix parmi tous les produits proposés
- Modifier_insformations() qui sert à modifier ses informations personnelles et donc les valeurs de ses attributs
- Participe() qui sert à participer à une vente

Dans la classe ''Fournisseur'' on a 7 attributs : le nom du fournisseur, son prénom, son adresse, son mail, son numéro de téléphone, sa spécialité et le type de matériel qu'il fournit.

On a une méthode : propose() qui sert à proposer une vente

Dans la classe '' Gérant du magasin'' on a 1 attribut qui est le nom du gérant.

On a deux méthodes :

- Valide() qui sert à valider une vente proposée par un fournisseur
- Sélectionne() qui sert à sélectionner les clients qui peuvent devenir des clients privilégiés

Dans la classe ''Comptable'' on a une méthode qui est valider_payement() qui sert à valider ou non le paiement du client privilégié à la fin d'une vente.

Dans la classe ''Vente'' on a 5 attributs : le nom de la vente, les dates de début et de fin, les activités concernées par cette vente et une description de la vente.

Dans la classe ''Produit'' on a 6 attributs : le nom du produit, une description technique du produit, une photo, un prix de base, un prix de vente (après réduction) et une taille.

Concernant la cardinalité on a un fournisseur qui ''propose()'' jusqu'à n vente, le gérant du magasin qui ''valide()'' jusqu'à n ventes et le client privilégié qui participe à jusqu'à n ventes.

De son côté une vente peut avoir n participants (clients), 1 fournisseur et 1 gérant.

Une vente est composée de 1 à n produits mais chaque produit à une seule vente.

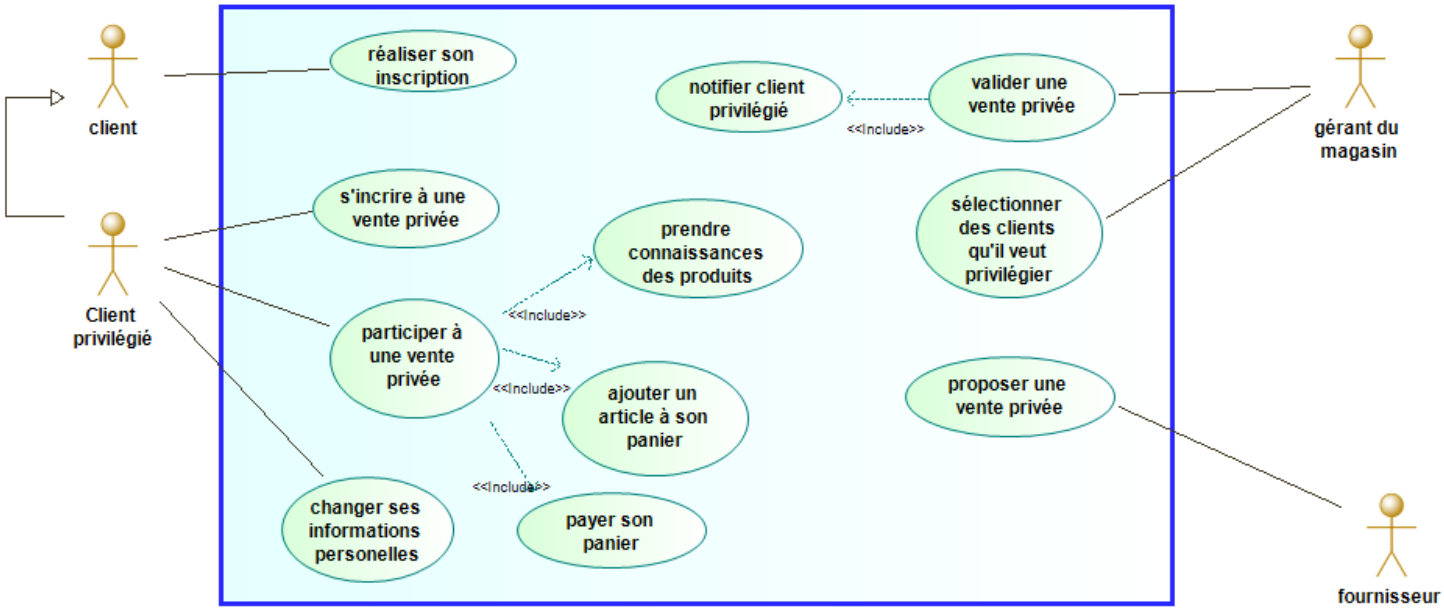
De la même sorte, un fournisseur a n produits qu'il propose dans une vente mais un produit vient d'un seul fournisseur.

Le gérant du magasin ''sélectionne()'' n client qui pourront devenir privilégiés mais un client est associé à un seul gérant de magasin.

Ce gérant peut avoir de 1 à n fournisseurs et mais chaque fournisseur est associé à ce seul gérant de magasin et le gérant a un seul comptable qui lui est associé et n'a que lui comme gérant de magasin aussi.

Spécifications Logiques

Diagramme de cas d'utilisation



Dans ce diagramme on retrouve 4 acteurs : client, client privilégié, gérant du magasin et fournisseur.

Un client peut réaliser son inscription.

On a par la suite un client privilégié qui est un client ayant réalisé son inscription.

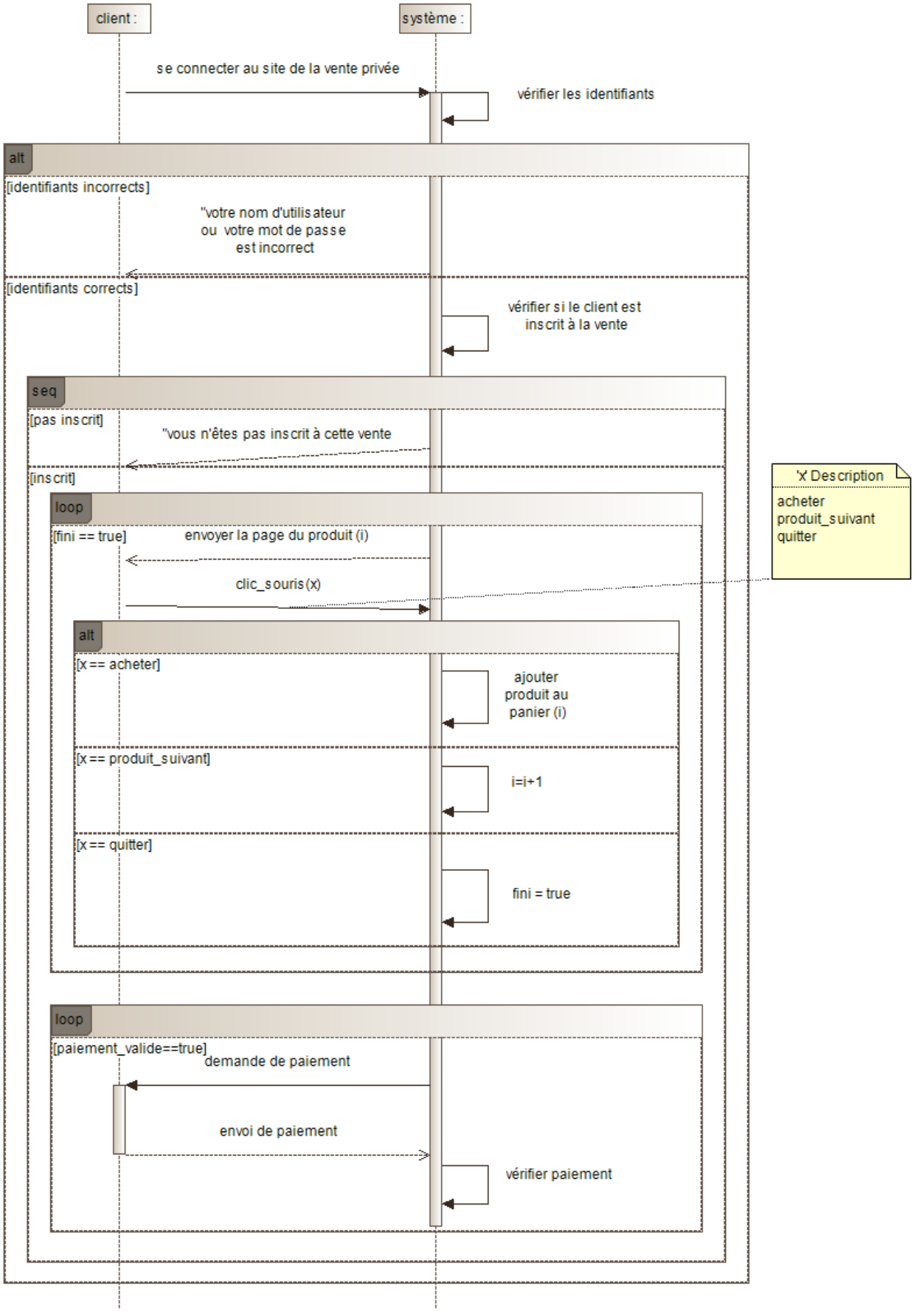
Il a trois choix : s'inscrire à une vente privée, changer ses informations personnelles ou participer à une vente privée.

Ce dernier choix inclue : prendre connaissance des produits, ajouter un article à son panier et payer son panier une fois les achats terminés.

De l'autre côté on a le fournisseur qui peut proposer une vente privée.

Enfin on le gérant du magasin qui a deux choix : sélectionner des clients qu'il veut privilégier et valider une vente privée qui inclue de notifier client privilégier qui sont concernés par cette vente d'après leurs informations personnelles.

Diagramme de séquence système



Dans ce diagramme de séquence système nous avons décidé de modéliser le cas ‘’participer à une vente’’.

On a donc notre client et notre système.

Comme vu dans le diagramme précédent on a inclus dans ‘’participer à une vente privée’’ le fait de ‘’prendre connaissances des produits’’, ‘’ajouter un article à son panier’’ et ‘’payer’’.

Au début de notre diagramme, notre client se connecte et le système vérifie qu’il est bien client privilégié ou non.

Si ce n’est pas le cas alors on lui affiche que son nom d’utilisateur ou mdp est incorrect.

Une fois la connexion vérifiée on vérifie si le client s’était bien inscrit à la vente et de la même façon, si ce n’est pas le cas alors on lui affiche qu’il n’est pas inscrit à cette vente.

S’il est inscrit alors on rentre dans une boucle qui affiche pour chaque produit sa fiche. On boucle tant que le client ne dit pas qu’il a fini.

A partir de là le système attend un clic souris de la part du client.

Il a trois options :

- ‘’acheter’’ auquel cas le produit est ajouté à son panier
- ‘’produit-suivant’’ auquel cas on affiche la prochaine fiche produit
- ‘’quitter’’ auquel cas le client spécifie qu’il a fini et on arrête donc de boucler

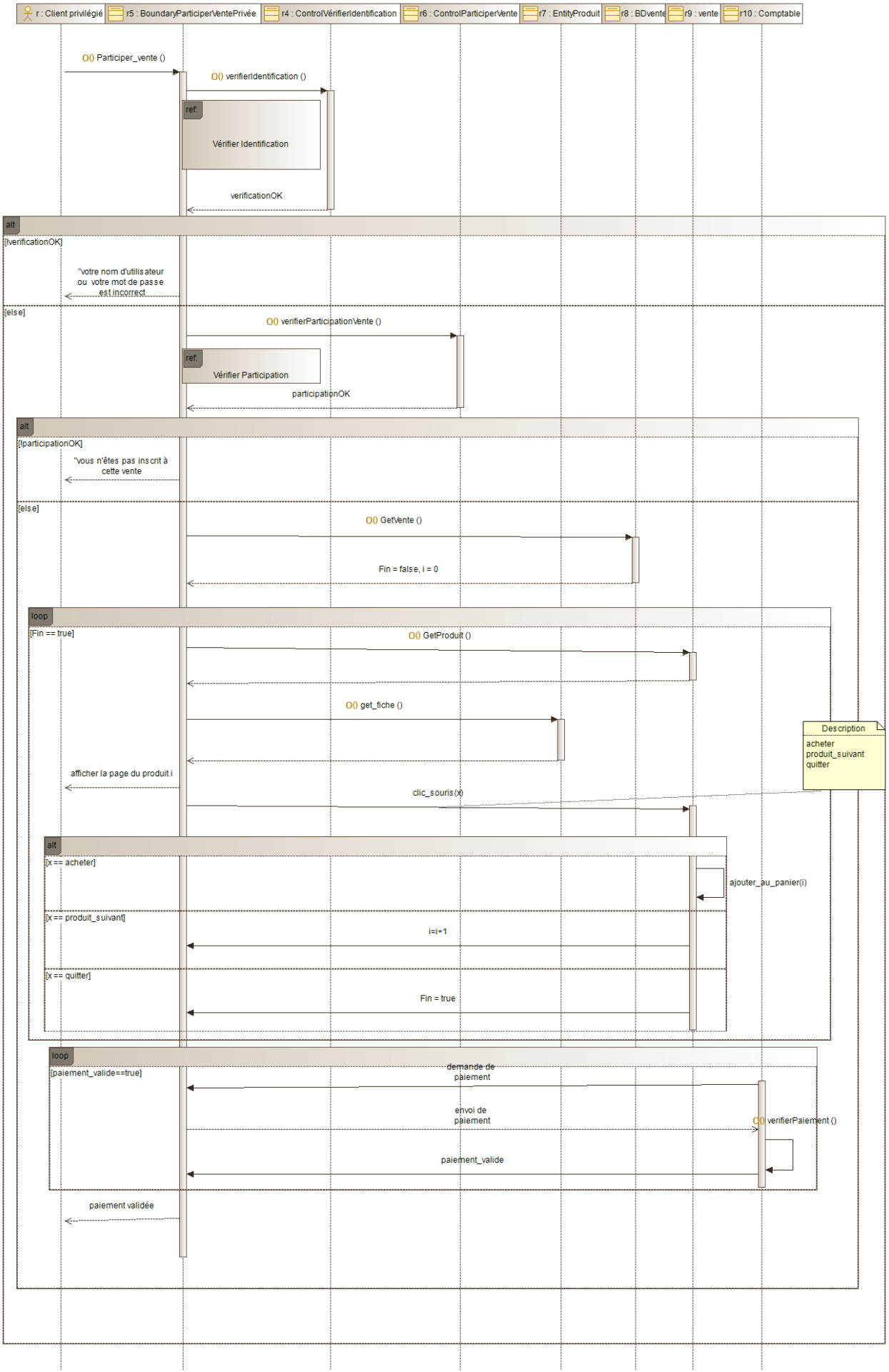
Une fois ses achats fait on passe alors dans une boucle qui sert à payer. On boucle tant que le paiement n’est pas validé par le système.

Le système demande au client de payer et le client envoie donc le paiement au système qui le vérifie. Si le paiement est validé alors on sort de la boucle, sinon on reboucle.

Nous avons choisi de boucler afin de permettre au client de réessayer de payer dans le cas où le paiement n’est pas validé.

Réalisation du cas d'utilisation choisi (participer à une vente privée)

Diagramme de séquence détaillé



Le diagramme de séquence détaillé reprend le diagramme de séquence mais en détaillant entre quels acteurs se font les actions.

On a donc 8 classes que l'on verra dans le diagramme de classes participantes après ce diagramme.

On a une première flèche qui va de Client privilégié en faisant appel à la méthode ''participer_vente()'' de la classe ''boundaryParticiperVentePrivée''.

On a ensuite la vérification de l'identification entre la classe ''boundaryParticiperVentePrivée'' et le classe ''ControlVerifierIdentification''. Elle pren une string qui sont les identifiants et renvoie si l'identification est ok ou non.

On rentre ensuite dans un ''alt''. On a donc le cas ou l'identification n'est pas ok dans lequel on affiche le même message que dans le diagramme précédent à l'utilisateur.

Si l'identification est ok alors on vérifie que le client s'est bien inscrit à la vente. Cette vérification se fait entre la classe ''boundaryParticiperVentePrivée'' et la classe ''controlParticiperVente''.

On rentre donc dans une deuxième 'alternative'. Si le client n'est pas inscrit alors on lui affiche qu'il n'est pas inscrit, sinon on va appeler la méthode ''getVente()'' de la classe ''BDVente''. Elle contient les ventes et on appelle donc la méthode avec un numéro de vente pour qu'elle nous renvoie la bonne vente.

On rentre après ça dans la loop qui gère les produits de chaque vente.

On appelle donc la méthode ''getProduit()'' qui retourne un produit de la vente. Cette méthode prend en paramètre un numéro de produit. On le fait entre la classe 'boundaryParticiperVentePrivée'' et ''vente''.

Après ca on appelle la méthode ''getFiche()'' entre ''boundaryParticiperVentePrivée'' et ''EntityProduit''. Elle affiche au client la fiche du produit en question.

Le client a alors le choix entre ajouter le produit à son panier, passer au produit suivant ou quitter.

Une fois le choix fait, s'il ne quitte pas alors on reboucle sur le prochain produit de la vente.

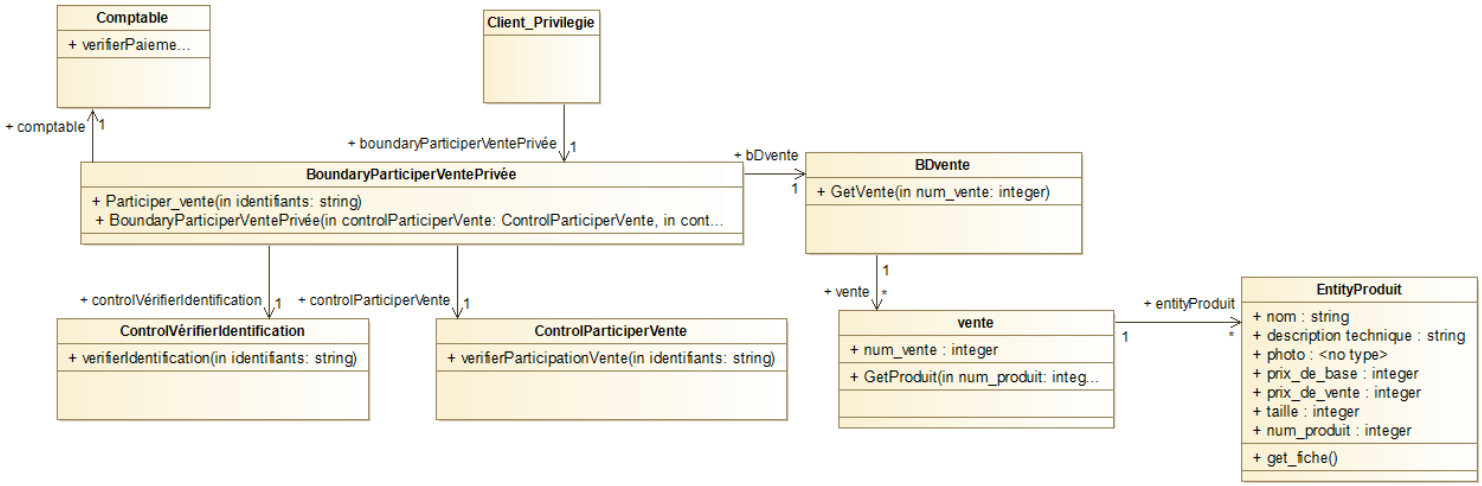
Une fois cette boucle terminée on passe à la boucle de paiement.

On a des échanges entre la classe ''comptable'' et ''boundaryParticiperVentePrivée''.

Pour commencer le comptable effectue une demande de paiement, après on a de ''boundaryParticiperVentePrivée'' à ''comptable'' un envoie de paiement, puis le ''comptable'' vérifie le paiement et quand le paiement est validé alors il envoie au ''boundaryParticiperVentePrivée'' un signal de paiement validé.

Pour finir on envoie de la classe ''BoundaryParticiperVentePrivée'' à la classe ''client privilégié'' un message pour lui signifier que le paiement a été validé.

Diagramme de classes participantes



On a donc 8 classes participantes :

- BoundaryParticiperventePrivée
- ControlVeirifieridentification
- ControlParticiperVente
- BDVente
- Vente
- EntityProduit
- Comptable
- Client_privilégié

La classe ‘’boundaryParticiperVentePrivée’’ est la classe principale. Elle contient deux méthodes : ‘’Participer_vente()’’ qui prend en arguments une string qui est les identifiants et ‘’BoundaryParticiperVentePrivée()’’ qui prend deux classses ‘’controlVerifierIdentification et controlParticiperVente.

La classe ‘’ControlVerifierIdentification’’ a une méthode ‘’verifieridentification()’’ qui prend en argument une string avec les identifiants du client privilégié. Elle vérifie si les identifiants correspondent à ceux d’un client. Elle renvoie un booléen qui dit si l’identification est validée ou non.

La classe ‘’ControlParticiperVente’’ a une méthode ‘’verifierParticipationVente()’’ qui prend en argument une string avec les identifiants du client. Elle vérifie que les identifiants du client correspondent à ceux s’étant inscrits à la vente au préalable. Elle renvoie un booléen qui valide ou non le fait que le client s’était bien inscrit à la vente.

La classe ‘’BDVente’’ a une méthode ‘’getVente()’’ qui prend en argument un numéro de vente. Elle renvoie la vente après être allé la chercher dans la base de données.

La classe ‘’Vente’’ a un attribut num_vente qui est un entier. C’est le numéro de la vente dont on se sert pour aller chercher la vente dans la base de données.

Elle possède une méthode ‘’getProduit()’’ qui prend un argument un entier qui est un numéro de produit. Elle renvoie le produit associé au numéro passé en paramètre.

La classe ‘’ EntityProduit’’ a 7 attributs :

- Nom
- Description_technique
- Photo
- Prix_de_base
- Prix_de_vente
- Taille
- Num_produit : numéro dont on se sert dans ‘’Vente’’ pour renvoyer le produit

Elle a une méthode get_fiche() qui renvoie la fiche produit afin qu’elle soit affichée au client quand il fait ses achats.

La classe ‘’comptable’’ a une méthode ‘’vérifierPaiement()’’ qui sert à vérifier que le paiement du client est valide. La méthode renvoie un booléen selon si le paiement est valide ou non.

Pour la cardinalité, a classe ‘’BoundaryParticiperVentePrivée’’ a une classe ‘’controlVerifieridentification’’, une classe ‘’ controlParticiperVente’’, une classe ‘’comptable’’ et une classe ‘’BDVente’’.

La classe ‘’BDVente’’ peu avoir jusqu’à n’’vente’’ mais une ‘’vente’’ n’a qu’une ‘’BDVente’’.

La classe ‘’Vente’’ peut avoir jusqu’à n ’’EntityProduit’’ et chaque ‘’EntityProduit’’ ne peut être associé qu’à une vente.

La classe ‘’Client privilégié’’ est liée à ‘’BoundaryParticiperVentePrivée’’ et un ‘’clientprivilégier’’ a 1 ‘’BoundaryParticiperVente’’.