DOCUMENT D'ANALYSE

Comparateur interne

Table des matières

I - Introduction

A - Principe général de l'application

1 - Point de vue client

Dans le cadre du projet de Programmation Orienté Objet, nous allons développer une application dont l'objectif sera de permettre à un client d'effectuer une recherche au travers d'un certain nombre de critères. Ces critères seront croisés avec une base de données afin de fournir au client une liste d'objets disponibles dans notre « catalogue », qui sera au plus proche de son souhait.

2 - Point de vue administrateur

D'un point de vue administration, cette application offrira un accès à une interface d'administration permettant de gérer l'ensemble des offres pour les modifier, supprimer ou encore en ajouter.

B - L'application

Il s'agira donc d'une application développée en Java. Elle s'appuiera sur plusieurs interfaces graphiques (saisie de la requête, affichage des résultats, administration …), une API contenant l'ensemble des algorithmes de comparaison des champs de spécifications de l'objet de notre offre, ainsi qu'une base de données permettant de stocker l'ensemble de nos offres.

C - Thème de l'application

Afin de personnaliser le projet, nous avons choisi de baser nos offres et donc notre application autour des jeux vidéo, qui offrent un très large choix de critères : en terme de quantités, de type de champs (numérique, chaîne de caractères, choix multiples …), de complexité d'algorithmes. Notre cas d'utilisation sera donc le client d'un site de vente de jeux vidéo, qui est à la recherche d'un jeu précis ou d'un type de jeu avec des critères : de types, de prix, de note etc…

II - Expression des besoins

A - Cas d'utilisation

Un utilisateur souhaite consulter une liste de jeux qui correspondent à leurs attentes.

Un administrateur veut pouvoir ajouter, supprimer, modifier des jeux vidéo après s'être authentifié.

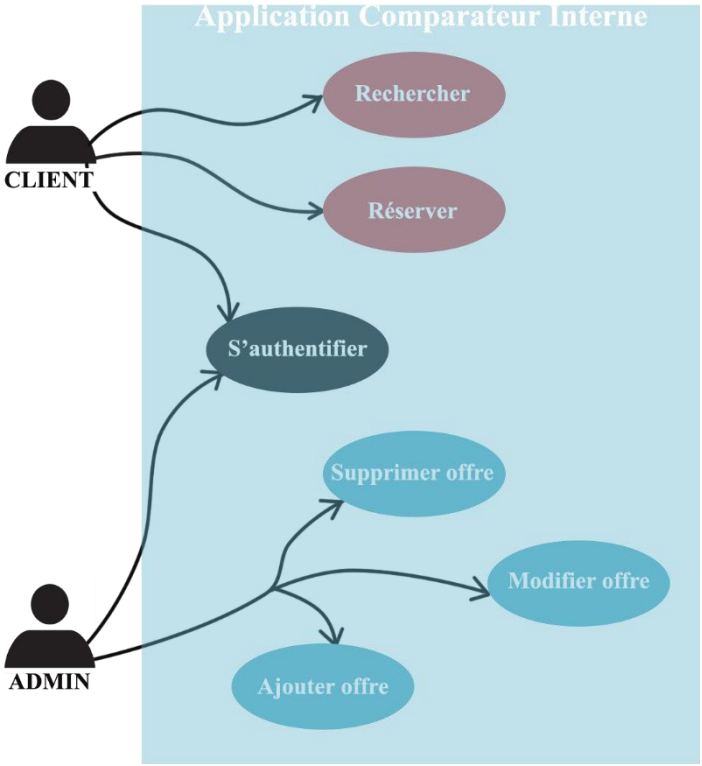


Schéma 1 : Diagramme UML cas d'utilisation

B - Séquence d'utilisation

1 - Côté client

1. Le client s'authentifie avec son mot de passe et son login.
2. L'application récupère l'ensemble des utilisateurs de la base de données.
3. L'application lui affiche l'interface de recherche.
4. Il saisit les critères sur l'interface afin de trouver les jeux qui correspondent à ses attentes.
5. L'application va contacter une base de données contenant l'ensemble des jeux répertoriés.
6. La base de données lui retourne l'ensemble des données (privées de celles ne correspondant pas aux critères bloquants).
7. L'application effectue toutes les comparaisons Offre <-> Demande, en leur affectant un score.
8. On retourne au client une liste contenant un certain nombre défini de jeux classés par ordre décroissant de pertinence vis-à-vis de la demande.
9. Le client réserve le jeu.

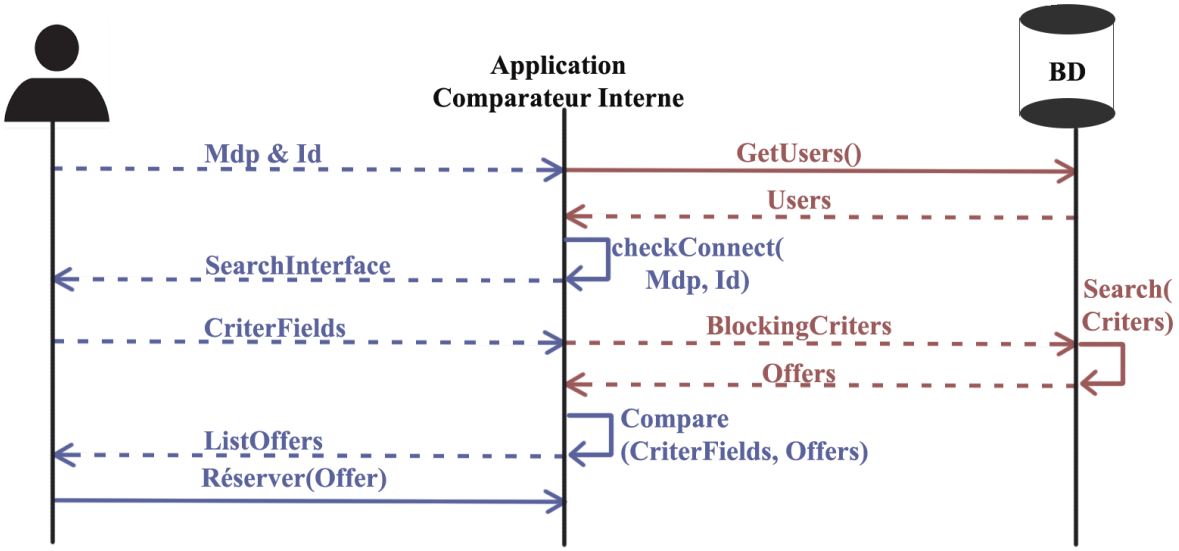


Schéma 2 : Diagramme UML de séquence Client

2 - Côté administrateur

1. Le client s'authentifie avec son mot de passe et son login.
2. L'application récupère l'ensemble des utilisateurs de la base de données.
3. L'application contrôle l'authentification.
4. Elle récupère auprès de la base de données l'ensemble des offres.
5. L'application lui affiche l'interface de gestion des offres.
6. Il modifie | ajoute | supprime des offres.
7. Une fois l'action réalisée sur l'interface celle-ci sera répercutée dans la base de données.
8. Puis l'administrateur se retrouvera sur la même interface, mise à jour des modifications effectuées.

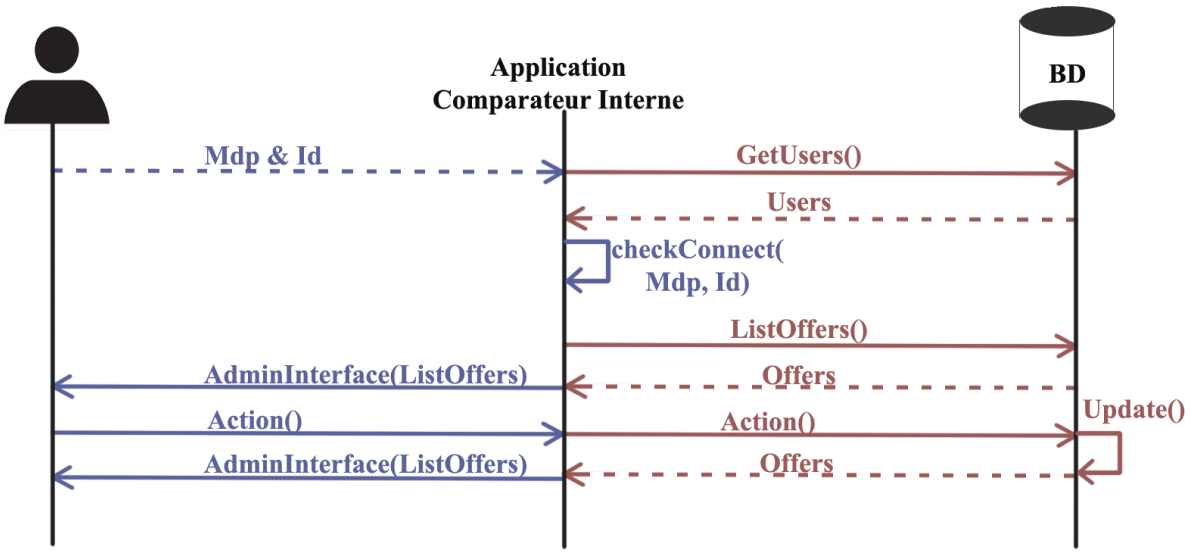


Schéma 3 : Diagramme UML de séquence Administrateur

C - Les exigences

Pour le client, il y a un certain nombre de critères qu'il faut que l'application respecte :

- Le comparateur ne doit pas passer plus de 10 secondes avant de répondre à une requête.
- Dans le classement, seules les informations principales seront affichées. Un clic sur le résultat permettra d'afficher toutes les informations
- Le nombre de résultats retourné devra être cohérent.
- Les résultats seront pertinents et pourront être justifiés.

III - Caractérisation de l'application

A - Choix du thème

Comme exposé plus haut dans ce document, nous avons décidé de développer notre application autour des jeux vidéo.

Afin de rendre ce projet réalisable dans le temps imparti, nous avons réduit notre offre à un nombre limité de domaines (thèmes) :

- **Aventure**
  - **Fiction interactive.**
  - **Visual novel.**
- **Action-Aventure**
  - **Infiltration.**
  - **Survival horror.**
- **Jeu de rôle**
  - **MMORPG.**
  - **A-RPG.**

B - Méthodes de comparaison et ensemble des choix

La comparaison se fera donc entre les éléments de notre base de données et la demande de l'utilisateur. Pour ces comparaisons, nous avons décidé que la proximité entre les critères de choix et les éléments de notre offre globale sera notée entre 1 et 1000.

Chaque caractéristique sera pondérée, plus le critère est proche (ou égal) à une caractéristique d'un élément, plus celui-ci marque des points. Le classement décroissant des éléments en fonction de ces points sera le résultat de la demande de l'utilisateur. Plusieurs méthodes seront utilisées pour marquer des points.

1 - Comparaison par Mots-clés

L'utilisateur rentre des mots-clés qui seront recherchés dans le champ des éléments de notre offre :

- **Nom du jeu**
- **Description**
- **Editeur**

2 - Comparaison par Intervalles

L'utilisateur donne une fourchette dans laquelle le champ doit s'inscrire. Si ce n'est pas le cas, l'écart à la fourchette réduira intelligemment le nombre de points obtenus, jusqu'à un minimum de 0

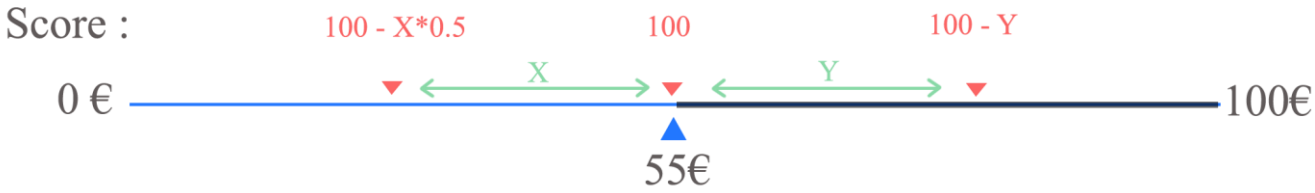


Schéma 4 : Illustration de la comparaison par intervalles

- **Prix**
- **Note du jeu**
- **Date de sortie**

3 – Comparaison par Choix simples

L'utilisateur choisit un attribut parmi plusieurs valeurs, si l'élément possède cet attribut alors il gagne des points.

- **Mode de jeu (Hors/Ligne)**
- **Mode de paiement (Abonnement, achat d'une licence)**

4 - Comparaison par Choix simples pondérés par un écart au critère

Même principe que pour le choix simple, à la différence que les attributs peuvent rapporter des points s'ils se rapprochent suffisamment du critère voulu.

- **Difficulté**
- **Durée de vie**

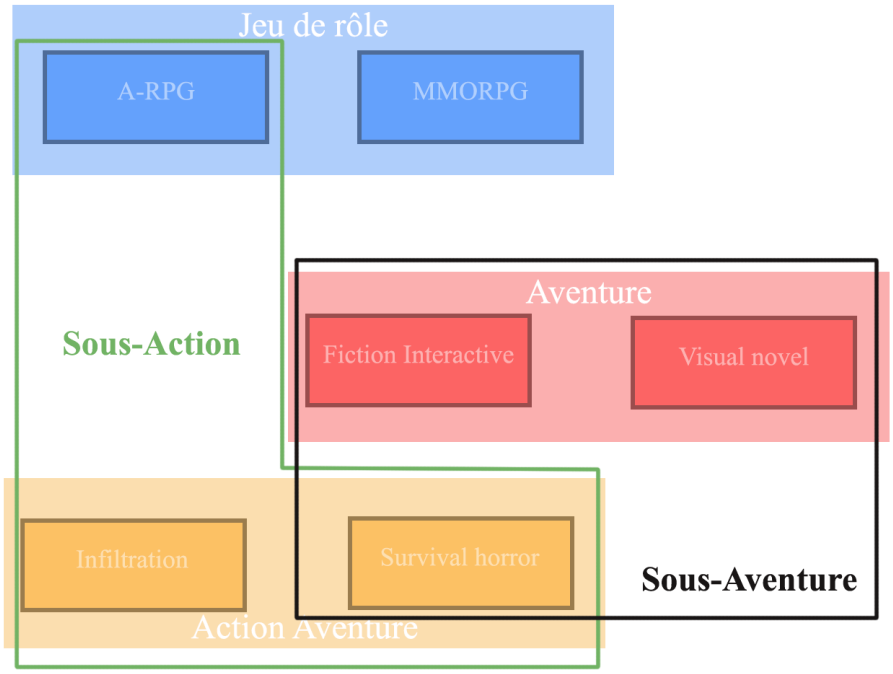
5 - Comparaison par Choix Multiple

L'utilisateur choisit quelques attributs parmi une collection de valeurs, pour chaque attribut retrouvé dans les caractéristiques du champ de l'élément, celui-ci gagne des points. Un attribut «bonus » (qui est en plus de TOUS les autres qui sont déjà vérifiés) ne rapporte pas de points, alors qu'un attribut « en trop » enlève des points.

- **Style de l'histoire**
- **Support de jeu**
- **Jouable avec accessoires**
- **Lieu d'achat (Internet | magasin)**

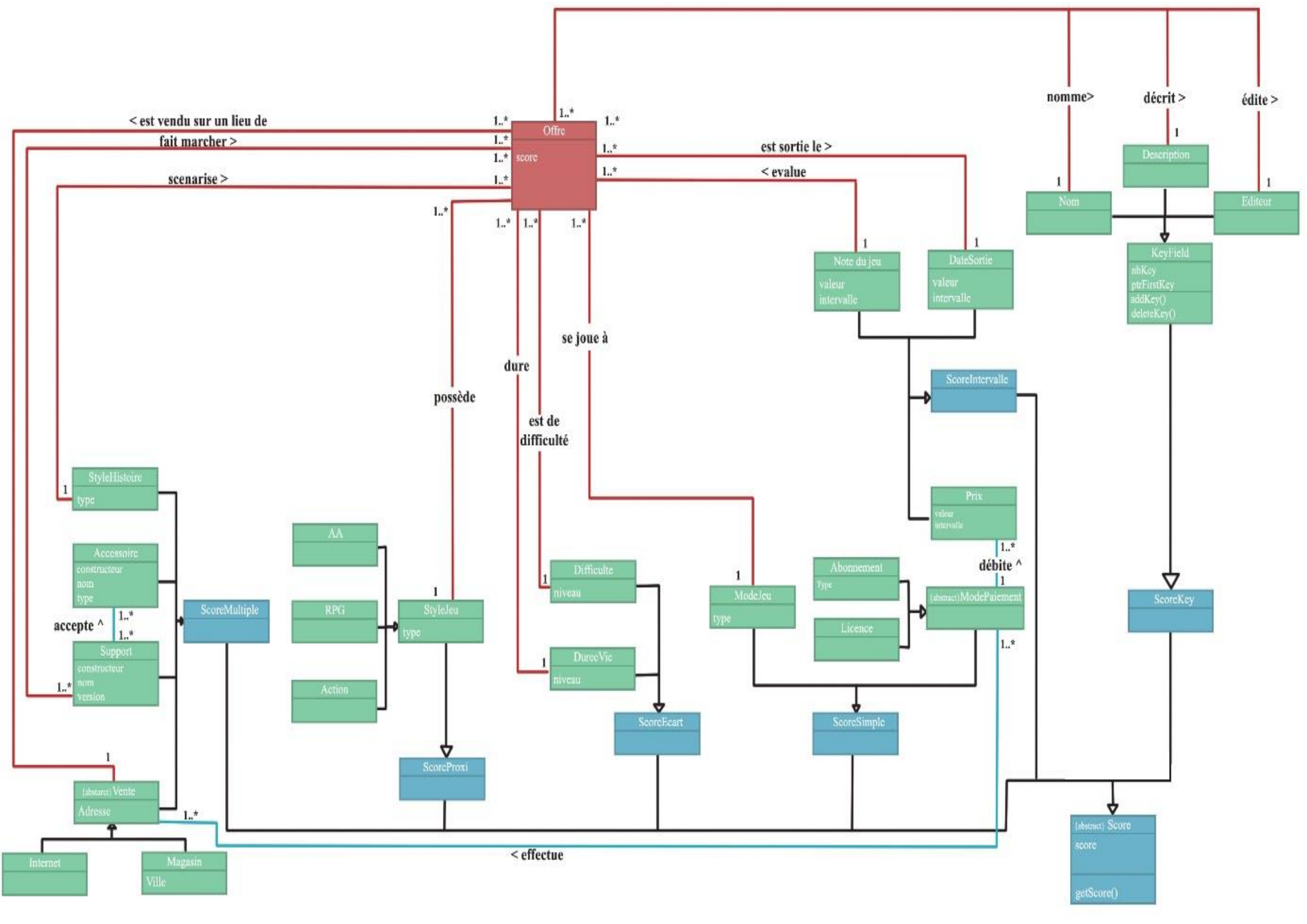
6 - Le cas du champ « Type de jeu »

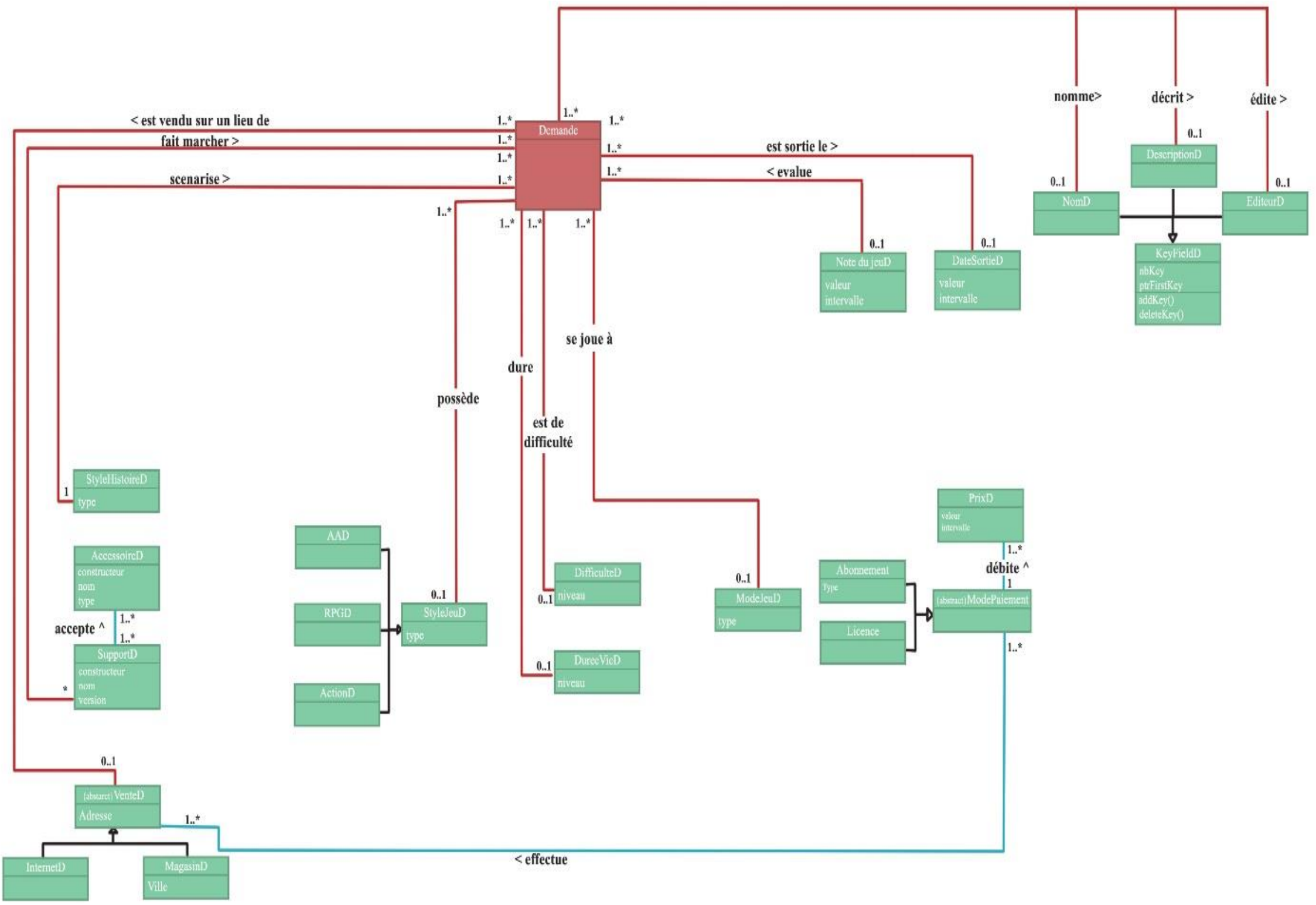
Pour cette dernière catégorie il s'agit de spécifier les relations entre les types de jeux. Pour ce faire, nous avons décidé de catégoriser chaque thème, un thème sera donc lié à une catégorie primaire [ex : A-RPG appartient aux Jeux de Rôle], puis pourra être lié à des sous-catégories (voir le schéma ci-dessous).



IV - Diagramme UML d'étude de domaine

Vous trouverez dans les deux pages suivantes le diagramme UML d'analyse, nous avons séparé la représentation de la demande et de l'offre en deux diagrammes distincts pour une meilleure lisibilité.


< est vendu sur un lieu de
fait marcher >
scenarise >
Offre
score
nomme>
décrit >
édite >
Description
Nom
Editeur
KeyField
nbKey
ptrFirstKey
addKey()
deleteKey()
est sortie le >
< evalue
Note du jeu
valeur
intervalle
DateSortie
valeur
intervalle
ScoreIntervalle
Prix
valeur
intervalle
débite ^
possède
dure
se joue à
est de
difficulté
StyleHistoire
type
Accessoire
constructeur
nom
type
accepte ^
Support
constructeur
nom
version
ScoreMultiple
AA
RPG
Action
StyleJeu
type
Difficulte
niveau
DureeVie
niveau
ModeJeu
type
Abonnement
type
Licence
(abstract)ModePaiement
ScoreKey
ScoreProxi
ScoreEcart
ScoreSimple
(abstract) Vente
Adresse
Internet
Magasin
Ville
< effectue
(abstract) Score
score
getScore()


Demande
< est vendu sur un lieu de
fait marcher >
scenarise >
est sortie le >
< evalue
nomme>
décrit >
édite >
NomD
DescriptionD
EditeurD
KeyFieldD
nbKey
ptrFirstKey
addKey()
deleteKey()
Note du jeuD
valeur
intervalle
DateSortieD
valeur
intervalle
possède
dure
se joue à
est de
difficulté
StyleHistoireD
type
AccessoireD
constructeur
nom
type
accepte ^
SupportD
constructeur
nom
version
AAD
RPGD
ActionD
StyleJeuD
type
DifficulteD
niveau
DureeVieD
niveau
ModeJeuD
type
Abonnement
type
Licence
PrixD
valeur
intervalle
débite ^
(abstract) ModePaiement
(abstract) VenteD
Adresse
InternetD
MagasinD
Ville
< effectue