Projet Analyse et conception d'applications

Analyse d'une application « gestion d'une ludothèque »

Scravatte Aurélie

Année 2013-2014

Ce rapport est réalisé dans le cadre du cours d'analyse et de conception d'applications du Bac Informatique de Gestion (2ème année)

Table des matières

I. Enoncé du projet

Ce projet aura pour but d'informatiser la gestion d'une ludothèque. Le personnel pourra gérer le stock de jeux disponibles/entrants/sortants, et les clients visualiser les jeux disponibles et attribuer des notes.

Les objectifs :

- gérer le stock de jeux disponibles, entrants et sortants ;
- gérer la liste des clients ;
- Permettre de recueillir des notes (1 -5 étoiles) ainsi que des commentaires des clients à propos de certains jeux ;
- gérer les évènements organisés par la ludothèque (présentations/tests de jeux ouverts au public) ;
- gérer le login dans l'application (permettre au personnel une gestion complète et aux clients, un aperçu des jeux disponibles/ empruntés).

II. Les besoins du client

L'application doit permettre au client de :

- S'identifier ;
- Modifier son compte (informations personnelles et mot de passe) ;
- Rejoindre un ou des évènements ;
- Modifier son pseudo ;
- Envoyer des messages/commentaire sur un/ou plusieurs jeu(x) ;
- lister ses jeux ;
- rechercher un/ou plusieurs jeux ;
- Consulter les commentaires ;
- évaluer les jeux ;

L'application doit permettre à un employé de :

- Réaliser les opérations d'un employé ;
- Gérer un ou des clients ;
    - Supprimer, ajouter, modifier ;
    - Consulter les jeux du client ;
- Gérer l'ensemble des jeux
    - Ajouter un ou des nouveaux jeux
    - Supprimer un ou des nouveaux jeux
    - Modifier un ou des nouveaux jeux

    - rechercher/lister un ou des nouveaux jeux
- Gérer les comptes des clients
    - Bannir un client
    - Réinitialiser le compte d'un client

L'application doit permettre à un administrateur de :

- Réaliser les opérations d'un client et employer ;

III. Les contraintes :

Une ludothèque est avant tout un lieu de loisir et détente. Cette application a pour but de faciliter l'accès à l'ensemble des jeux, évènements et son dossier personnel.

Gestion des familles membres.

- Formulaire rapide à compléter, ne nécessitant qu'un minimum de données à saisir.
- Détails sur chaque membre de la famille ou d'une classe au complet dans le même dossier.
- Recherche facile et presque instantanée sur plusieurs critères au choix.
- Commentaires illimités pour chaque adhérent permettant de faire des recherches.
- Gestion intuitive des cotisations et des pénalités pour retard, perte ou bris.
- Accès immédiat à tous les dossiers de prêts, actifs ou historiques.
- Renouvellement par un simple clic d'un seul ou de l'ensemble des prêts d'une famille

Gestion des jeux ou autres articles (jeux, cassettes, ...)

- Trouvez le jeu idéal dans votre inventaire en un clin d'œil grâce à des critères de recherche très flexibles.
- Naviguez facilement parmi tous les articles correspondants au(x) critère(s).
- Vous savez instantanément à qui est prêté un article ou tous ceux qui l'ont emprunté auparavant.
- Un clic et vous avez le dossier détaillé du membre lié qui apparaît.
- Vous avez plusieurs jeux semblables, ajoutez-les.

Gestion des prêts, locations, services, retours, bris, pertes et vols

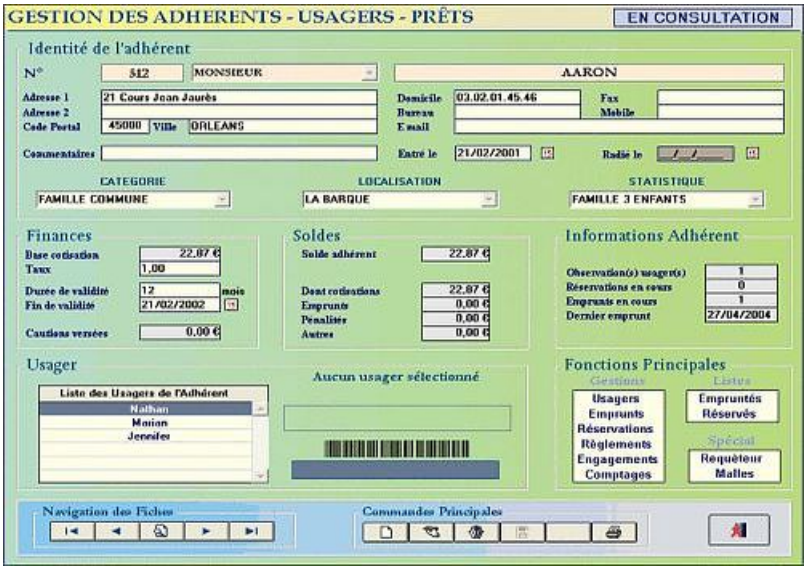
- vous irez plus vite à enregistrer le prêt ou le retour de plusieurs jeux.

- Les dates de retour sont déterminées automatiquement par le logiciel, mais vous pouvez les changer facilement, si besoin est, grâce à un calendrier intégré.
- Si vous avez des frais de location pour chaque article ou pour certains d'entre eux, ils seront affichés, pourront être modifiés et comptabilisés automatiquement.
- Les amendes sont aussi calculées automatiquement, en tenant compte de vos jours ouvrables, mais vous pouvez également les ajuster sur le champ.
- Vous obtenez une confirmation visuelle instantanée que vous avez prêtée ou récupérée le bon article.
- La recherche instantanée ou le rapport des membres ayant des prêts en retard vous permettent de rappeler rapidement les retardataires.

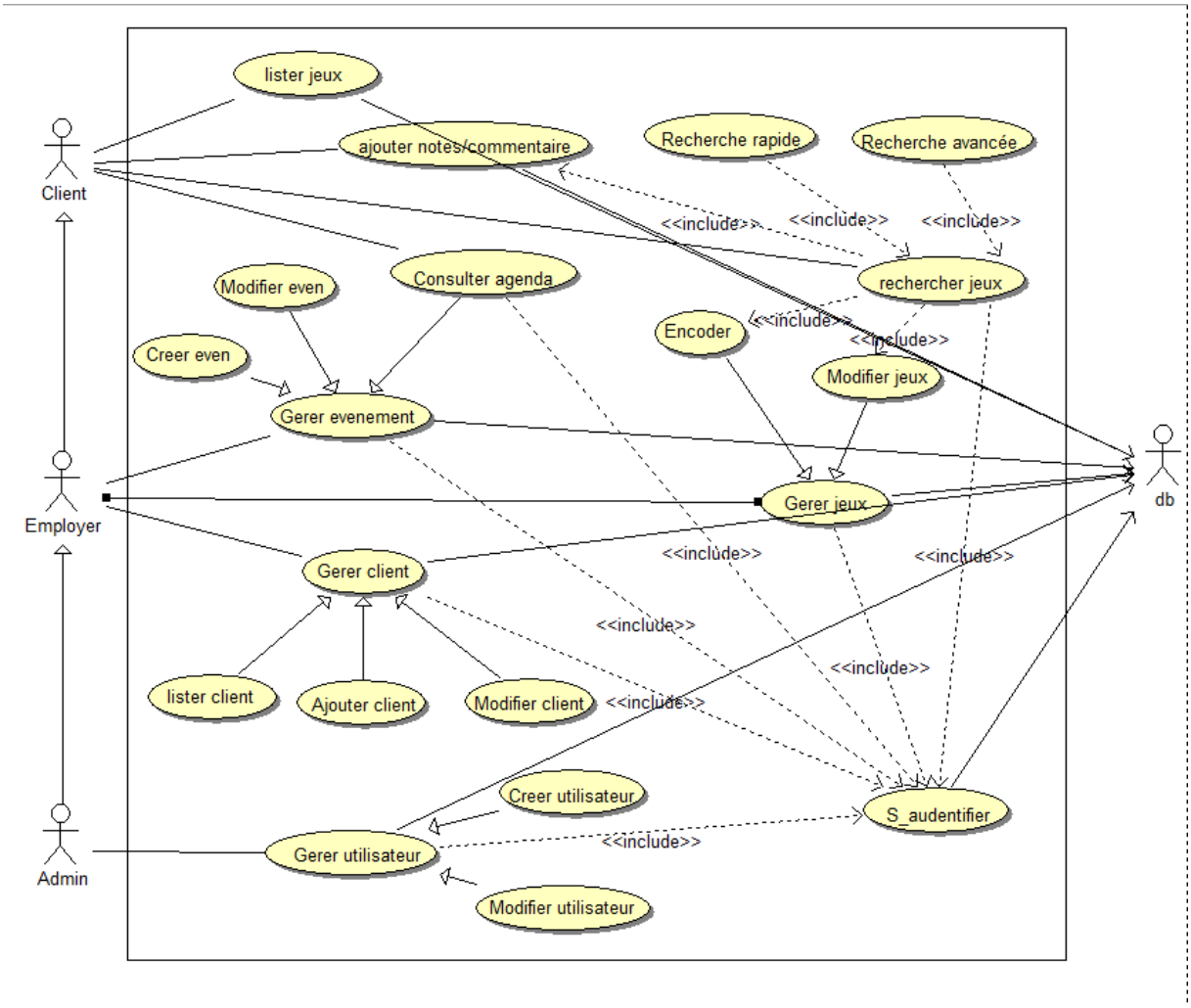
IV. L'existence de projet similaire :

Après diverses recherches sur le net, j'ai trouvé différente forme d'une gestion de ludothèque.

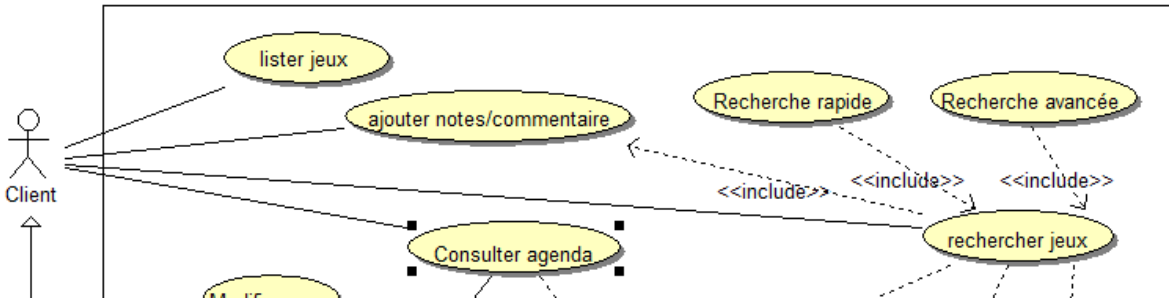
- Kawa Ludothèque : Un logiciel complètement adaptable et dédié à la gestion de vos ludothèques. Un produit fiable, puissant et complet qui est en évolution permanente depuis 1995 grâce à la participation de près de 160 ludothèques municipales et associatives. Les principales fonctions : Gestion de Stock, Adhérents, Prêts, Réservations, Jeu sur Place, Caisses, Cotisations et renouvellements, Relances Courriers et Mails, Bilans et Statistiques, et bien d'autres encore…



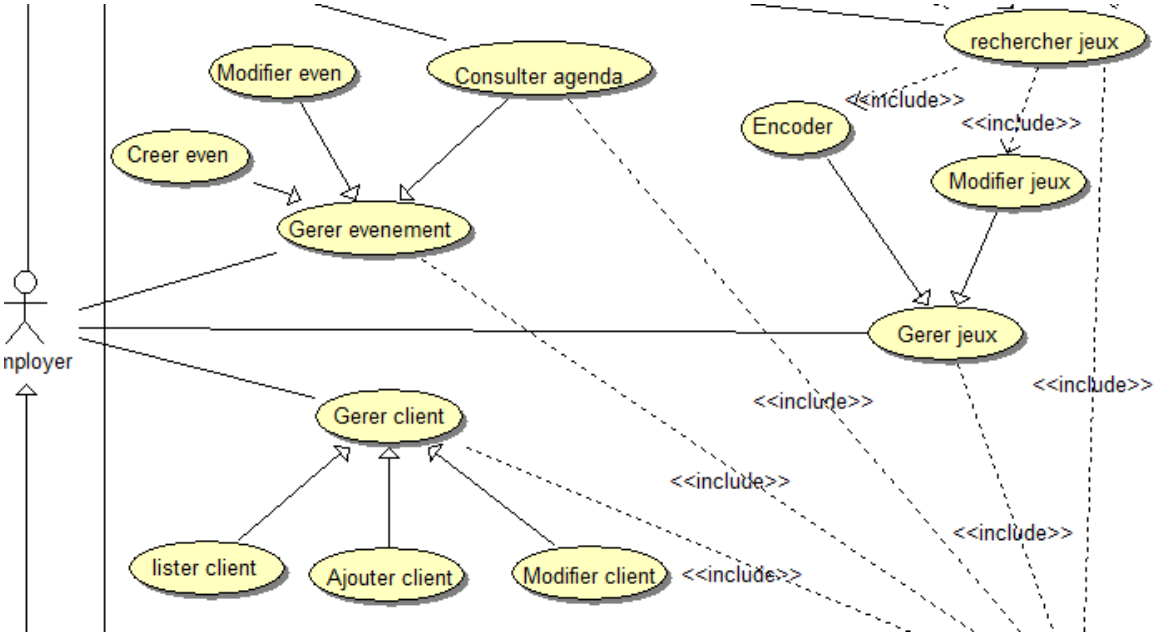
V. Diagrammes de use Case



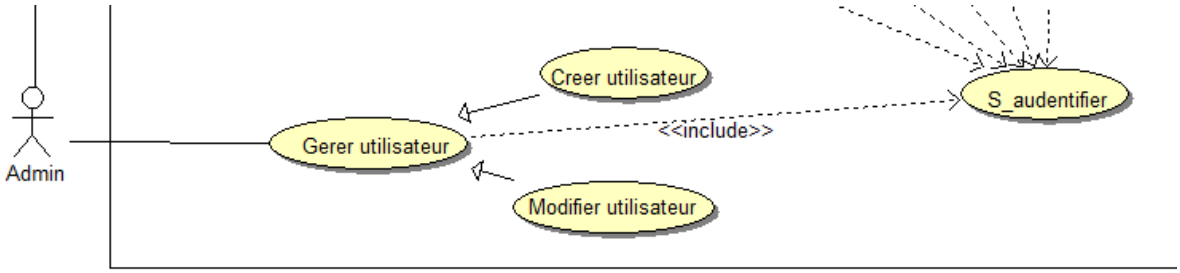
Le client :



L'employer :



L'administrateur :



Analyse du cas « S'inscrire comme utilisateur »

Identification

Nom du cas : encoder un nouveau client.

But : détaille les étapes permettant à un employé d'encoder un nouveau client.

Acteur principal : Employé.

Acteur secondaire : Client.

Date : 08/04/2014

Responsable : A. Scravatte

Version : 1.0

Séquencement

Le cas d'utilisation commence lorsqu'un employé souhaite encoder un nouveau client dans l'application.

Préconditions

Le client n'existe pas encore dans l'application.

Enchainement nominal

1) L'employé saisit les informations d'identification fournies par le client (nom, prénom, date de naissance, adresse, email, téléphone,…)

2) L'application vérifie qu'il n'existe pas déjà un client avec les mêmes informations.

3) L'application génère un nouveau numéro client.

4) L'application crée le nouveau client d'après les informations reçues et le nouveau numéro client généré.

5) L'application fourni une confirmation de l'ajout à l'employé ainsi que le numéro client généré.

Enchainement d'exception

E1 : Le client existe déjà dans l'application

L'enchainement démarre après le point 2 de la séquence nominale :

3) L'application quitte le mode ajout de nouveau client.

4) L'application ouvre l'écran de modification du client que l'employé souhaitait ajouter.

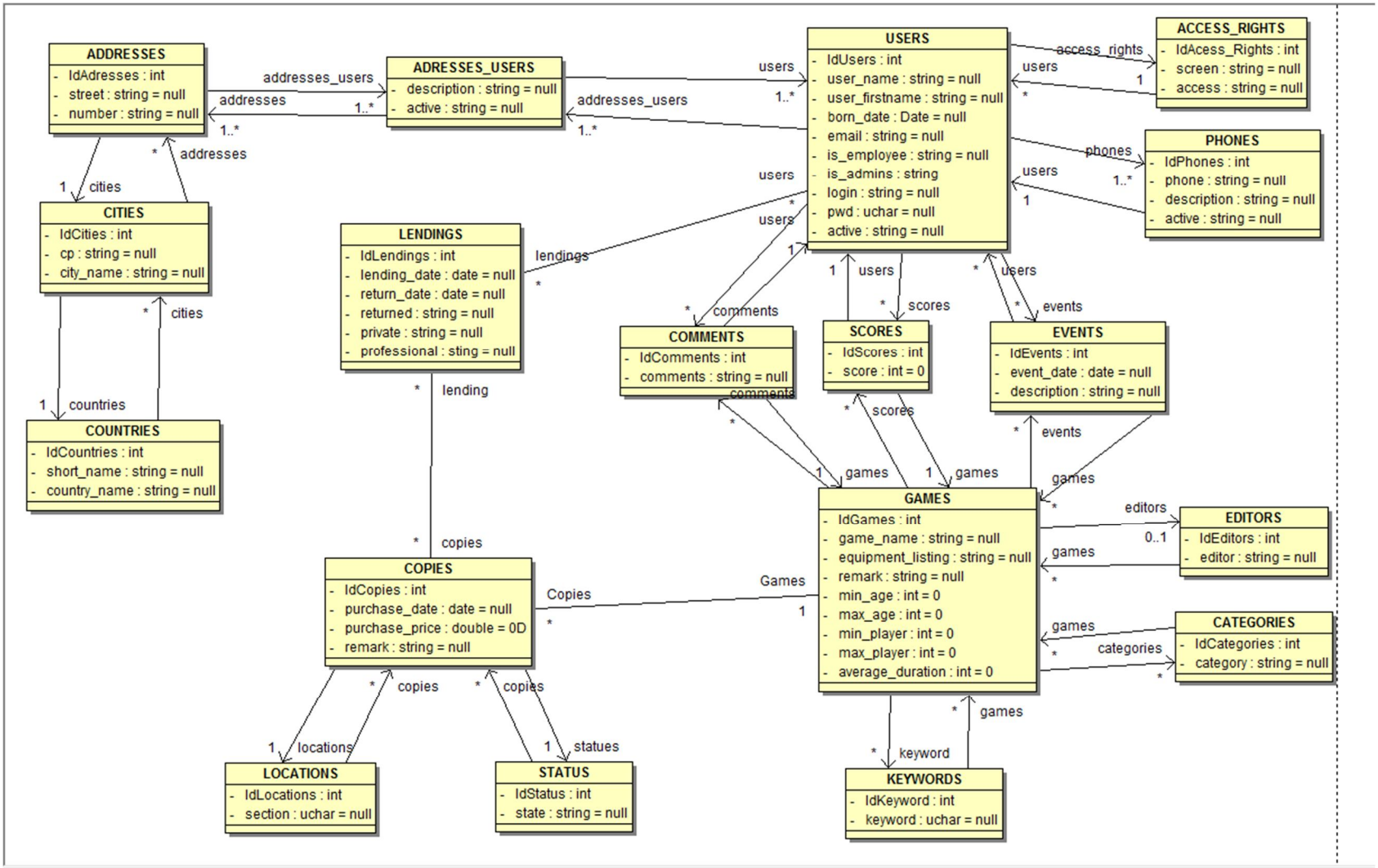
Post-conditions

Le système contient bien le client que l'employé souhaitait encoder.

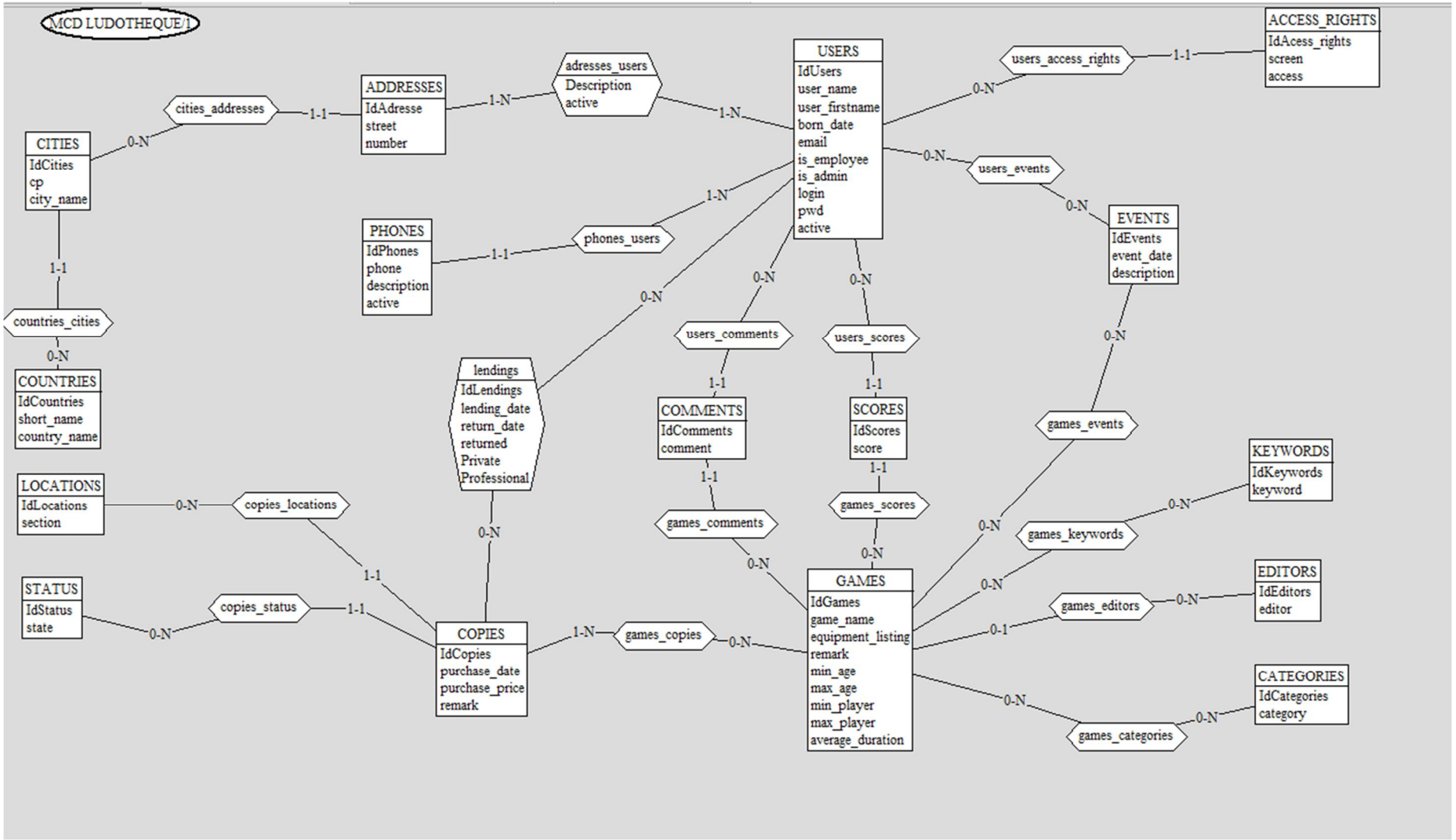
Rubriques optionnelles

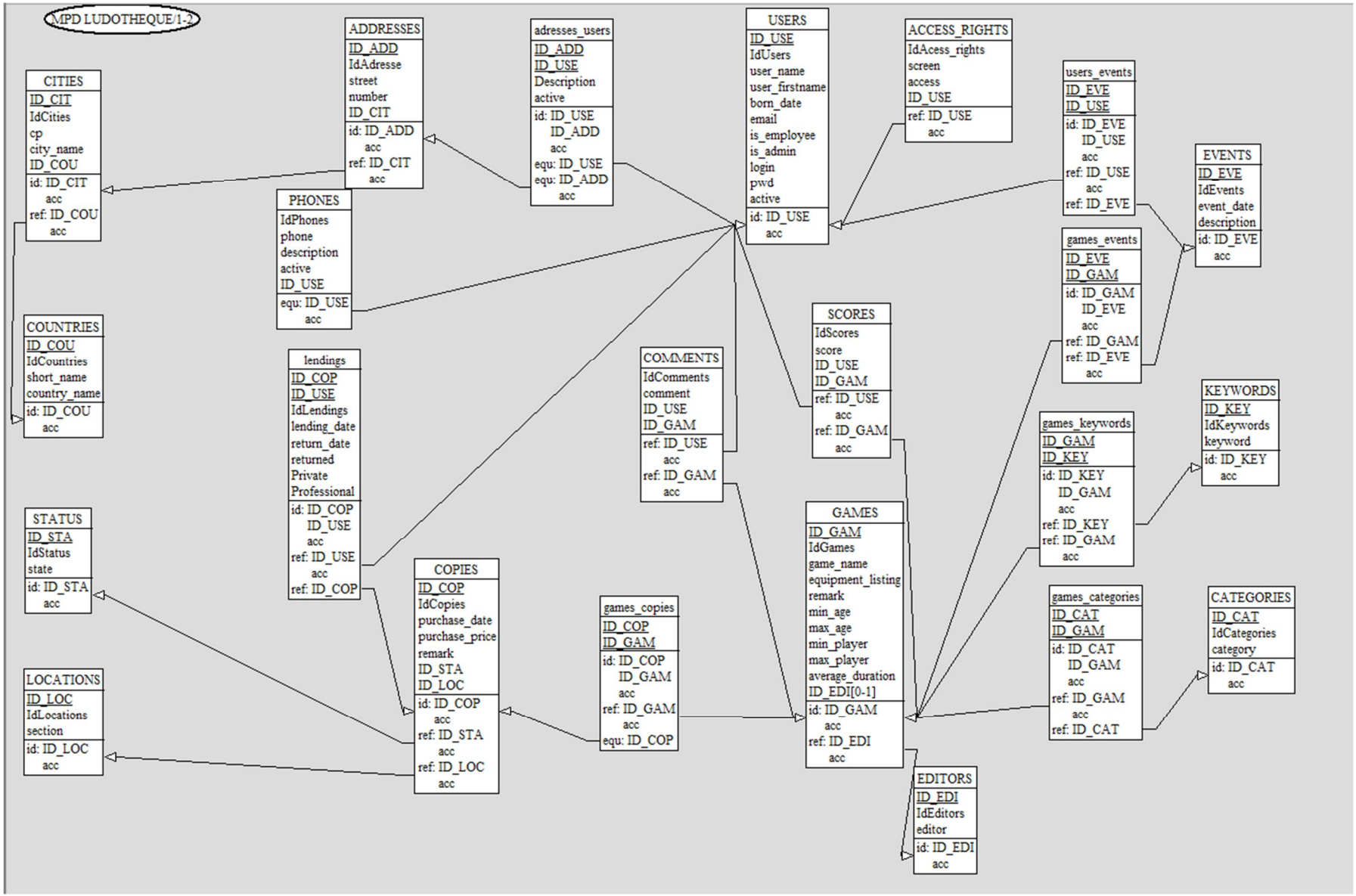
Néant

VI. Diagrammes de classe :



VII. MCD et MPD :




MPD LUDOTHEQUE/1-2
CITIES
ID_CIT
IdCities
cp
city_name
ID_COU
id: ID_CIT
acc
ref: ID_COU
acc
ADDRESSES
ID_ADD
IdAdresse
street
number
ID_CIT
id: ID_ADD
acc
ref: ID_CIT
acc
adresses_users
ID_ADD
ID_USE
Description
active
id: ID_USE
ID_ADD
acc
equ: ID_USE
equ: ID_ADD
acc
USERS
ID_USE
IdUsers
user_name
user_firstname
born_date
email
is_employee
is_admin
login
pwd
active
id: ID_USE
acc
ACCESS_RIGHTS
IdAcess_rights
screen
access
ID_USE
ref: ID_USE
acc
users_events
ID_EVE
ID_USE
id: ID_EVE
ID_USE
acc
ref: ID_USE
acc
ref: ID_EVE
EVENTS
ID_EVE
IdEvents
event_date
description
id: ID_EVE
acc
PHONES
IdPhones
phone
description
active
ID_USE
equ: ID_USE
acc
games_events
ID_EVE
ID_GAM
id: ID_GAM
ID_EVE
acc
ref: ID_GAM
ref: ID_EVE
acc
COUNTRIES
ID_COU
IdCountries
short_name
country_name
id: ID_COU
acc
lendings
ID_COP
ID_USE
IdLendings
lending_date
return_date
returned
Private
Professional
id: ID_COP
ID_USE
acc
ref: ID_USE
acc
ref: ID_COP
COMMENTS
IdComments
comment
ID_USE
ID_GAM
ref: ID_USE
acc
ref: ID_GAM
acc
SCORES
IdScores
score
ID_USE
ID_GAM
ref: ID_USE
acc
ref: ID_GAM
acc
KEYWORDS
ID_KEY
IdKeywords
keyword
id: ID_KEY
acc
games_keywords
ID_GAM
ID_KEY
id: ID_KEY
ID_GAM
acc
ref: ID_KEY
ref: ID_GAM
acc
STATUS
ID_STA
IdStatus
state
id: ID_STA
acc
COPIES
ID_COP
IdCopies
purchase_date
purchase_price
remark
ID_STA
ID_LOC
id: ID_COP
acc
ref: ID_STA
acc
ref: ID_LOC
acc
games_copies
ID_COP
ID_GAM
id: ID_COP
ID_GAM
acc
ref: ID_GAM
acc
equ: ID_COP
GAMES
ID_GAM
IdGames
game_name
equipment_listing
remark
min_age
max_age
min_player
max_player
average_duration
ID_EDI[0-1]
id: ID_GAM
acc
ref: ID_EDI
acc
games_categories
ID_CAT
ID_GAM
id: ID_CAT
ID_GAM
acc
ref: ID_GAM
acc
ref: ID_CAT
CATEGORIES
ID_CAT
IdCategories
category
id: ID_CAT
acc
LOCATIONS
ID_LOC
IdLocations
section
id: ID_LOC
acc
EDITORS
ID_EDI
IdEditors
editor
id: ID_EDI
acc

VIII. code SQL :

```
-- ********************************************
-- * SQL MySQL generation
-- *------------------------------------------
-- * DB-MAIN version: 9.1.6
-- * Generator date: Feb 25 2013
-- * Generation date: Wed Jun 04 10:17:09 2014
-- * LUN file: C:\Users\Zarias\Desktop\java\EXAMEN\MCDMPD\A_SCRAVATTE.lun
-- * Schema: MPD LUDOTHEQUE/1-2
-- ********************************************
-- Database Section
-- _________________

create database MPD LUDOTHEQUE;
use MPD LUDOTHEQUE;

-- Tables Section
-- _____________

create table ACCESS_RIGHTS (
     IdAcess_rights char(1) not null,
     screen varchar(255) not null,
     access char not null,
     ID_USE int not null);

create table ADDRESSES (
     ID_ADD int not null auto_increment,
     IdAdresse char(1) not null,
     street -- Compound attribute -- not null,
     number varchar(10) not null,
     ID_CIT int not null,
     constraint ID_ID primary key (ID_ADD));

create table adresses_users (
     ID_ADD int not null,
     ID_USE int not null,
     Description varchar(1) not null,
     active char not null,
     constraint ID_adresses_users_ID primary key (ID_USE, ID_ADD));

create table CATEGORIES (
     ID_CAT int not null auto_increment,
     IdCategories char(1) not null,
     category varchar(255) not null,
     constraint ID_ID primary key (ID_CAT));

create table CITIES (
     ID_CIT int not null auto_increment,
     IdCities char(1) not null,
```

```
      cp varchar(10) not null,
      city_name varchar(255) not null,
      ID_COU int not null,
      constraint ID_ID primary key (ID_CIT));

create table COMMENTS (
      IdComments char(1) not null,
      comment varchar(255) not null,
      ID_USE int not null,
      ID_GAM int not null);

create table COPIES (
      ID_COP int not null auto_increment,
      IdCopies char(1) not null,
      purchase_date date not null,
      purchase_price int not null,
      remark varchar(255) not null,
      ID_STA int not null,
      ID_LOC int not null,
      constraint ID_ID primary key (ID_COP));

create table COUNTRIES (
      ID_COU int not null auto_increment,
      IdCountries char(1) not null,
      short_name varchar(10) not null,
      country_name varchar(255) not null,
      constraint ID_ID primary key (ID_COU));

create table EDITORS (
      ID_EDI int not null auto_increment,
      IdEditors char(1) not null,
      editor varchar(255) not null,
      constraint ID_ID primary key (ID_EDI));

create table EVENTS (
      ID_EVE int not null auto_increment,
      IdEvents char(1) not null,
      event_date date not null,
      description varchar(255) not null,
      constraint ID_ID primary key (ID_EVE));

create table GAMES (
      ID_GAM int not null auto_increment,
      IdGames char(1) not null,
      game_name varchar(250) not null,
      equipment_listing varchar(255) not null,
      remark varchar(255) not null,
      min_age int not null,
      max_age int not null,
      min_player int not null,
      max_player int not null,
      average_duration int not null,
```

```
      ID_EDI int,
      constraint ID_ID primary key (ID_GAM));

create table games_categories (
      ID_CAT int not null,
      ID_GAM int not null,
      constraint ID_games_categories_ID primary key (ID_CAT,
ID_GAM));

create table games_copies (
      ID_COP int not null,
      ID_GAM int not null,
      constraint ID_games_copies_ID primary key (ID_COP,
ID_GAM));

create table games_events (
      ID_EVE int not null,
      ID_GAM int not null,
      constraint ID_games_events_ID primary key (ID_GAM,
ID_EVE));

create table games_keywords (
      ID_GAM int not null,
      ID_KEY int not null,
      constraint ID_games_keywords_ID primary key (ID_KEY,
ID_GAM));

create table KEYWORDS (
      ID_KEY int not null auto_increment,
      IdKeywords char(1) not null,
      keyword varchar(255) not null,
      constraint ID_ID primary key (ID_KEY));

create table lendings (
      ID_COP int not null,
      ID_USE int not null,
      IdLendings char(1) not null,
      lending_date date not null,
      return_date date not null,
      returned char not null,
      Private char not null,
      Professional char not null,
      constraint ID_lendings_ID primary key (ID_COP, ID_USE));

create table LOCATIONS (
      ID_LOC int not null auto_increment,
      IdLocations char(1) not null,
      section varchar(255) not null,
      constraint ID_ID primary key (ID_LOC));

create table PHONES (
      IdPhones char(1) not null,
```

```
      phone varchar(50) not null,
      description varchar(255) not null,
      active char not null,
      ID_USE int not null);

create table SCORES (
      IdScores char(1) not null,
      score int not null,
      ID_USE int not null,
      ID_GAM int not null);

create table STATUS (
      ID_STA int not null auto_increment,
      IdStatus char(1) not null,
      state varchar(255) not null,
      constraint ID_ID primary key (ID_STA));

create table USERS (
      ID_USE int not null auto_increment,
      IdUsers char(1) not null,
      user_name varchar(255) not null,
      user_firstname varchar(255) not null,
      born_date date not null,
      email varchar(255) not null,
      is_employee char not null,
      is_admin char not null,
      login varchar(255) not null,
      pwd varchar(255) not null,
      active char not null,
      constraint ID_ID primary key (ID_USE));

create table users_events (
      ID_EVE int not null,
      ID_USE int not null,
      constraint ID_users_events_ID primary key (ID_EVE,
ID_USE));

-- Constraints Section
-- ____________________

alter table ACCESS_RIGHTS add constraint
FKusers_access_rights_FK
      foreign key (ID_USE)
      references USERS (ID_USE);

-- Not implemented
-- alter table ADDRESSES add constraint ID_CHK
--     check(exists(select * from adresses_users
--                  where adresses_users.ID_ADD = ID_ADD));

alter table ADDRESSES add constraint FKcities_addresses_FK
      foreign key (ID_CIT)
```

```
      references CITIES (ID_CIT);

alter table adresses_users add constraint FKadr_USE
      foreign key (ID_USE)
      references USERS (ID_USE);

alter table adresses_users add constraint FKadr_ADD_FK
      foreign key (ID_ADD)
      references ADDRESSES (ID_ADD);

alter table CITIES add constraint FKcountries_cities_FK
      foreign key (ID_COU)
      references COUNTRIES (ID_COU);

alter table COMMENTS add constraint FKusers_comments_FK
      foreign key (ID_USE)
      references USERS (ID_USE);

alter table COMMENTS add constraint FKgames_comments_FK
      foreign key (ID_GAM)
      references GAMES (ID_GAM);

-- Not implemented
-- alter table COPIES add constraint ID_CHK
--     check(exists(select * from games_copies
--                  where games_copies.ID_COP = ID_COP));

alter table COPIES add constraint FKcopies_status_FK
      foreign key (ID_STA)
      references STATUS (ID_STA);

alter table COPIES add constraint FKcopies_locations_FK
      foreign key (ID_LOC)
      references LOCATIONS (ID_LOC);

alter table GAMES add constraint FKgames_editors_FK
      foreign key (ID_EDI)
      references EDITORS (ID_EDI);

alter table games_categories add constraint FKgam_GAM_3_FK
      foreign key (ID_GAM)
      references GAMES (ID_GAM);

alter table games_categories add constraint FKgam_CAT
      foreign key (ID_CAT)
      references CATEGORIES (ID_CAT);

alter table games_copies add constraint FKgam_GAM_2_FK
      foreign key (ID_GAM)
      references GAMES (ID_GAM);

alter table games_copies add constraint FKgam_COP
```

```
      foreign key (ID_COP)
      references COPIES (ID_COP);

alter table games_events add constraint FKgam_GAM_1
      foreign key (ID_GAM)
      references GAMES (ID_GAM);

alter table games_events add constraint FKgam_EVE_FK
      foreign key (ID_EVE)
      references EVENTS (ID_EVE);

alter table games_keywords add constraint FKgam_KEY
      foreign key (ID_KEY)
      references KEYWORDS (ID_KEY);

alter table games_keywords add constraint FKgam_GAM_FK
      foreign key (ID_GAM)
      references GAMES (ID_GAM);

alter table lendings add constraint FKlen_USE_FK
      foreign key (ID_USE)
      references USERS (ID_USE);

alter table lendings add constraint FKlen_COP
      foreign key (ID_COP)
      references COPIES (ID_COP);

alter table PHONES add constraint FKphones_users_FK
      foreign key (ID_USE)
      references USERS (ID_USE);

alter table SCORES add constraint FKusers_scores_FK
      foreign key (ID_USE)
      references USERS (ID_USE);

alter table SCORES add constraint FKgames_scores_FK
      foreign key (ID_GAM)
      references GAMES (ID_GAM);

-- Not implemented
-- alter table USERS add constraint ID_CHK
--     check(exists(select * from adresses_users
--                  where adresses_users.ID_USE = ID_USE));
-- Not implemented
-- alter table USERS add constraint ID_CHK
--     check(exists(select * from PHONES
--                  where PHONES.ID_USE = ID_USE));

alter table users_events add constraint FKuse_USE_FK
      foreign key (ID_USE)
      references USERS (ID_USE);

```

```
alter table users_events add constraint FKuse_EVE
      foreign key (ID_EVE)
      references EVENTS (ID_EVE);

-- Index Section
-- _____________

create index FKusers_access_rights_IND
      on ACCESS_RIGHTS (ID_USE);

create unique index ID_IND
      on ADDRESSES (ID_ADD);

create index FKcities_addresses_IND
      on ADDRESSES (ID_CIT);

create unique index ID_adresses_users_IND
      on adresses_users (ID_USE, ID_ADD);

create index FKadr_ADD_IND
      on adresses_users (ID_ADD);

create unique index ID_IND
      on CATEGORIES (ID_CAT);

create unique index ID_IND
      on CITIES (ID_CIT);

create index FKcountries_cities_IND
      on CITIES (ID_COU);

create index FKusers_comments_IND
      on COMMENTS (ID_USE);

create index FKgames_comments_IND
      on COMMENTS (ID_GAM);

create unique index ID_IND
      on COPIES (ID_COP);

create index FKcopies_status_IND
      on COPIES (ID_STA);

create index FKcopies_locations_IND
      on COPIES (ID_LOC);

create unique index ID_IND
      on COUNTRIES (ID_COU);

create unique index ID_IND
      on EDITORS (ID_EDI);

```

```
create unique index ID_IND
     on EVENTS (ID_EVE);

create unique index ID_IND
     on GAMES (ID_GAM);

create index FKgames_editors_IND
     on GAMES (ID_EDI);

create unique index ID_games_categories_IND
     on games_categories (ID_CAT, ID_GAM);

create index FKgam_GAM_3_IND
     on games_categories (ID_GAM);

create unique index ID_games_copies_IND
     on games_copies (ID_COP, ID_GAM);

create index FKgam_GAM_2_IND
     on games_copies (ID_GAM);

create unique index ID_games_events_IND
     on games_events (ID_GAM, ID_EVE);

create index FKgam_EVE_IND
     on games_events (ID_EVE);

create unique index ID_games_keywords_IND
     on games_keywords (ID_KEY, ID_GAM);

create index FKgam_GAM_IND
     on games_keywords (ID_GAM);

create unique index ID_IND
     on KEYWORDS (ID_KEY);

create unique index ID_lendings_IND
     on lendings (ID_COP, ID_USE);

create index FKlen_USE_IND
     on lendings (ID_USE);

create unique index ID_IND
     on LOCATIONS (ID_LOC);

create index FKphones_users_IND
     on PHONES (ID_USE);

create index FKusers_scores_IND
     on SCORES (ID_USE);

create index FKgames_scores_IND
```

```
      on SCORES (ID_GAM);

create unique index ID_IND
      on STATUS (ID_STA);

create unique index ID_IND
      on USERS (ID_USE);

create unique index ID_users_events_IND
      on users_events (ID_EVE, ID_USE);

create index FKuse_USE_IND
      on users_events (ID_USE);
```

IX. Classes java :

classe USERS :

```
1 package Classes;
2 import java.sql.Date;

public class USERS {
6 private int IdUsers;
7 private String user_name = null;
8 private String user_firstname = null;
9 private Date born_date = null;
10 private String email = null;
11 private String is_employee = null;
12 private String is_admins;
13 private String login = null;
14 private char pwd;
15 private String active = null;
16 private LENDINGS lendings;
17 private ADDRESSES addresses;
18 private PHONES phones;
19 private ACCESS_RIGHTS access_rights;
20 private EVENTS events;
21 private SCORES scores;
22 private COMMENTS comments;
23 private ADRESSES_USERS addresses_users;
24 public String getUser_name() {
25 return user_name;
26 }
27 public void setUser_name(String user_name) {
28 this.user_name = user_name;
29 }
30 public String getUser_firstname() {
31 return user_firstname;
32 }
33 public void setUser_firstname(String user_firstname) {
34 this.user_firstname = user_firstname;
35 }
36 public Date getBorn_date() {
37 return born_date;
38 }
39 public void setBorn_date(Date born_date) {
40 this.born_date = born_date;
41 }
42 public String getEmail() {
43 return email;
44 }
45 public void setEmail(String email) {
46 this.email = email;
47 }
48 public String getLogin() {
49 return login;
50 }
51 public void setLogin(String login) {
52 this.login = login;
53 }
54 public char getPwd() {
```

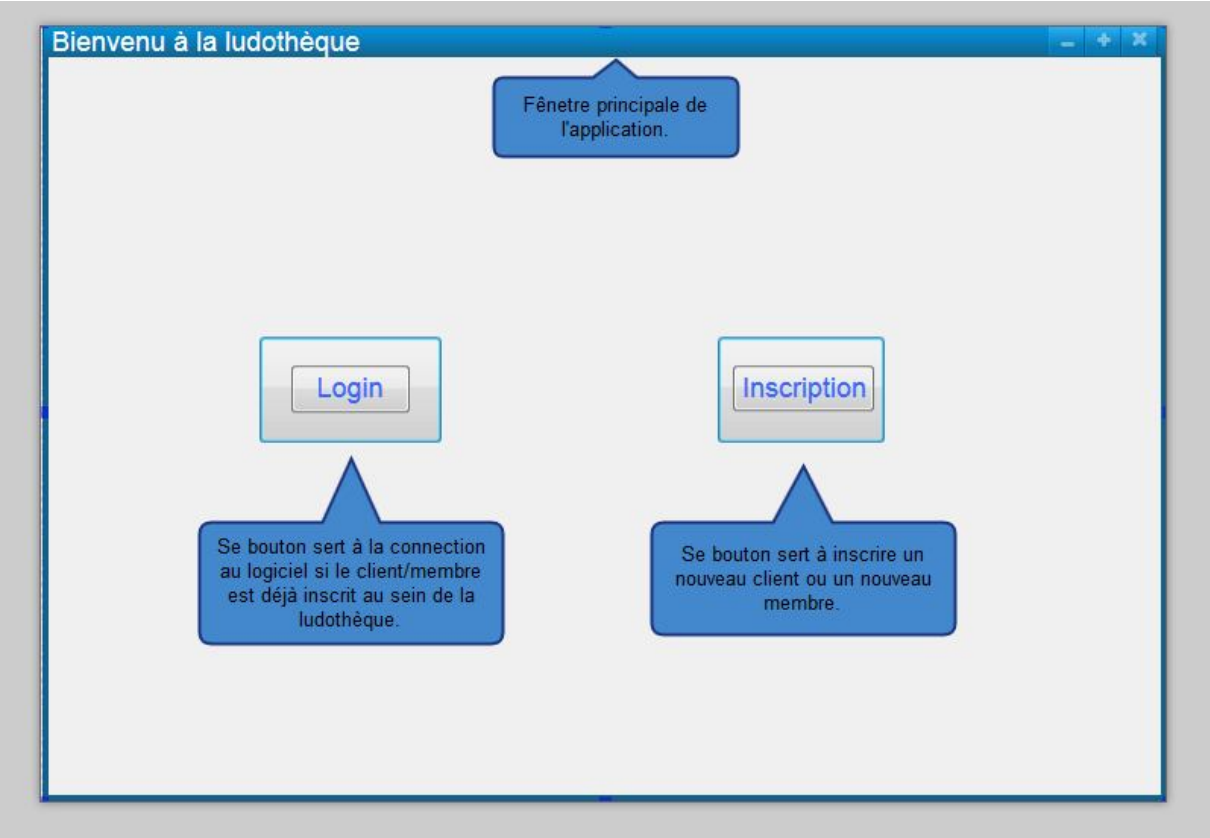
```
55 return pwd;
56 }
57 public void setPwd(char pwd) {
58 this.pwd = pwd;
59 }
60 public String getActive() {
Page 1
USERS.java
61 return active;
62 }
63 public void setActive(String active) {
64 this.active = active;
65 }
66 public ADDRESSES getAddresses() {
67 return addresses;
68 }
69 public void setAddresses(ADDRESSES addresses) {
70 this.addresses = addresses;
71 }
72 public PHONES getPhones() {
73 return phones;
74 }
75 public void setPhones(PHONES phones) {
76 this.phones = phones;
77 }
78 public ACCESS_RIGHTS getAccess_rights() {
79 return access_rights;
80 }
81 public void setAccess_rights(ACCESS_RIGHTS access_rights) {
82 this.access_rights = access_rights;
83 }
84 public COMMENTS getComments() {
85 return comments;
86 }
87 public void setComments(COMMENTS comments) {
88 this.comments = comments;
89 }
90 @Override
91 public String toString() {
92 return "USERS [user_name=" + user_name + ", user_firstname="
93 + user_firstname + ", born_date=" + born_date + ", email=" + email
94 + ", login=" + login + ", pwd=" + pwd + ", active=" + active
95 + ", addresses=" + addresses + ", phones=" + phones + "]";
96 }
97
98
99
100 }
101
```

Classes Adresses

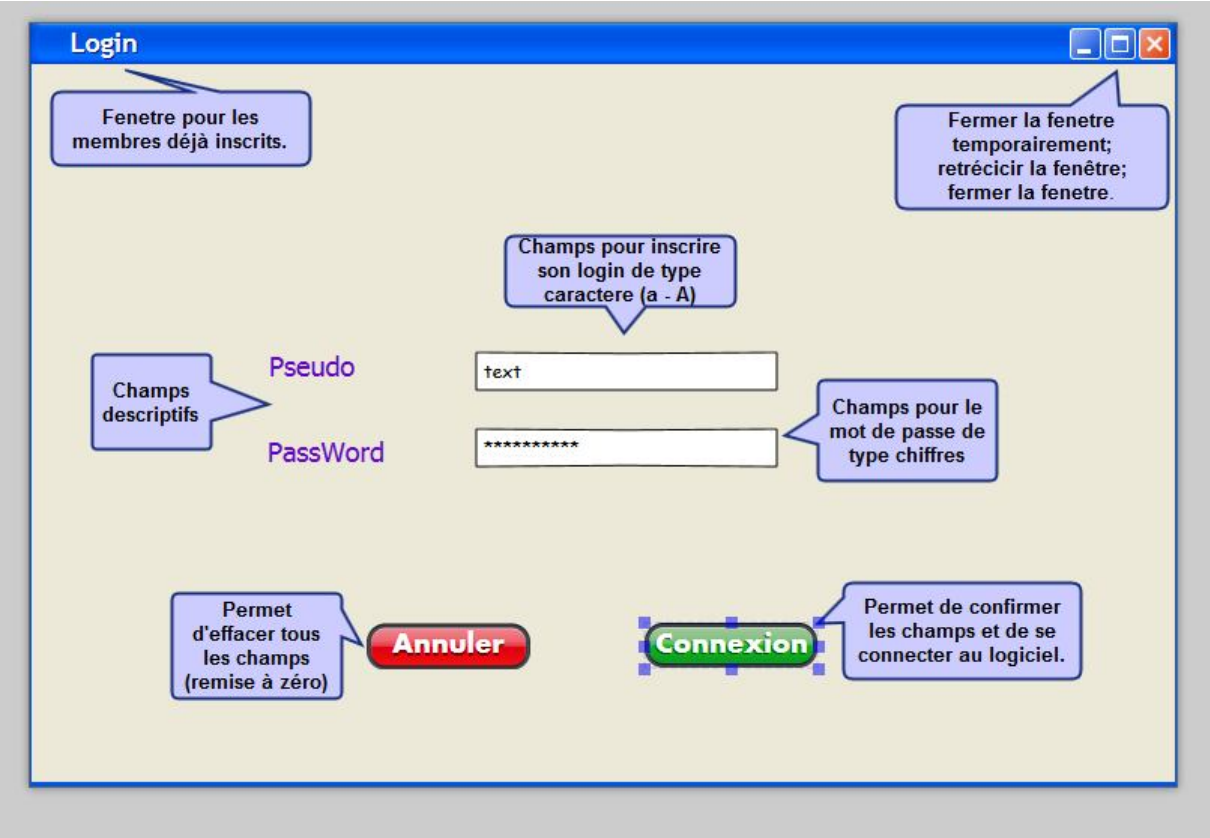
```
1 package Classes;
2 class ADDRESSES {
3 private int IdAdresses;
45
private String street = null;
6 private String number = null;
7 private CITIES cities;
8 private USERS users;
9 private ADRESSES_USERS addresses_users;
10
11 public int getIdAdresses() {
12 return IdAdresses;
13 }
14 public void setIdAdresses(int idAdresses) {
15 IdAdresses = idAdresses;
16 }
17 public String getStreet() {
18 return street;
19 }
20 public void setStreet(String street) {
21 this.street = street;
22 }
23 public String getNumber() {
24 return number;
25 }
26 public void setNumber(String number) {
27 this.number = number;
28 }
29 public CITIES getCities() {
30 return cities;
31 }
32 public void setCities(CITIES cities) {
33 this.cities = cities;
34 }
35 public USERS getUsers() {
36 return users;
37 }
38 public void setUsers(USERS users) {
39 this.users = users;
40 }
41 public ADRESSES_USERS getAddresses_users() {
42 return addresses_users;
43 }
44 public void setAddresses_users(ADRESSES_USERS addresses_users) {
45 this.addresses_users = addresses_users;
46 }
47 @Override
48 public String toString() {
49 return "ADDRESSES [IdAdresses=" + IdAdresses + ", street=" + street
50 + ", number=" + number + ", cities=" + cities + ", users=" + users
51 + ", addresses_users=" + addresses_users + "]";
52 }
53
54 }
```

X. Maquettes d'ihm :

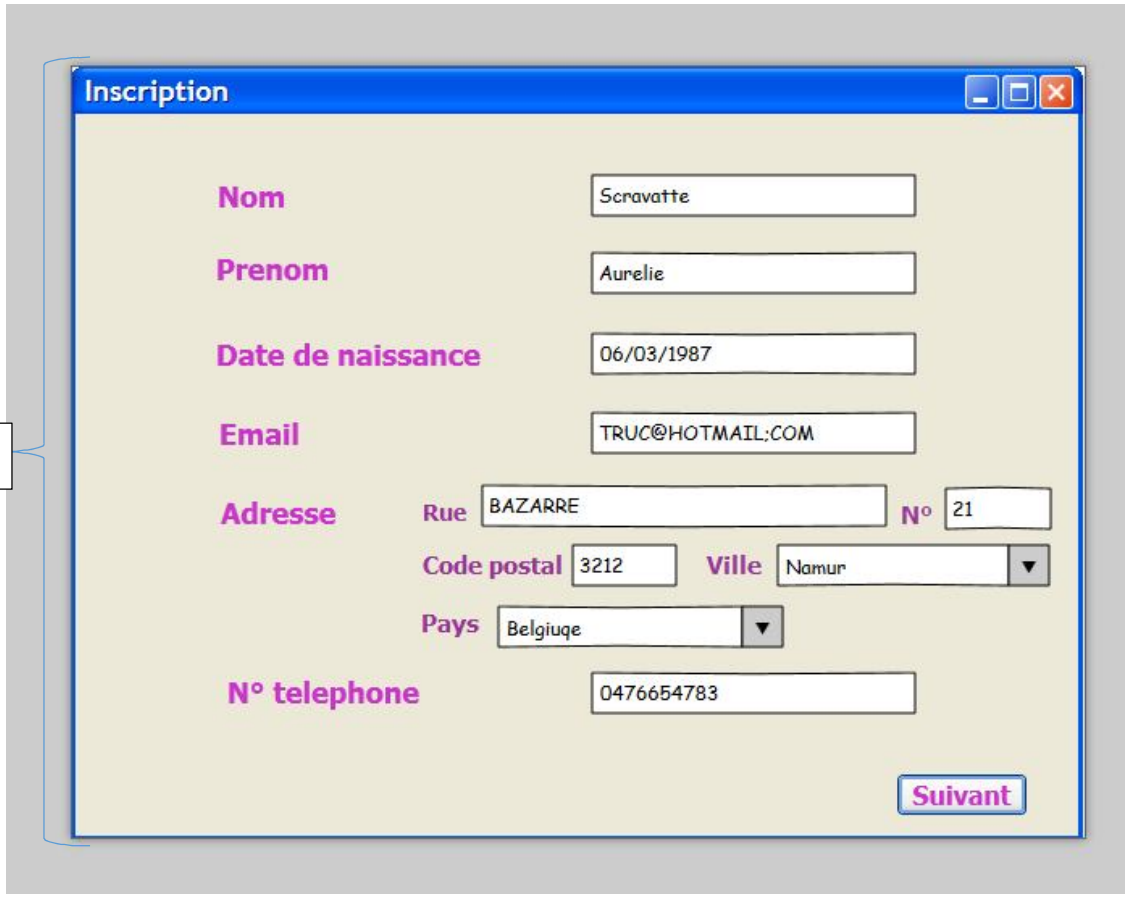
Ihm principales :

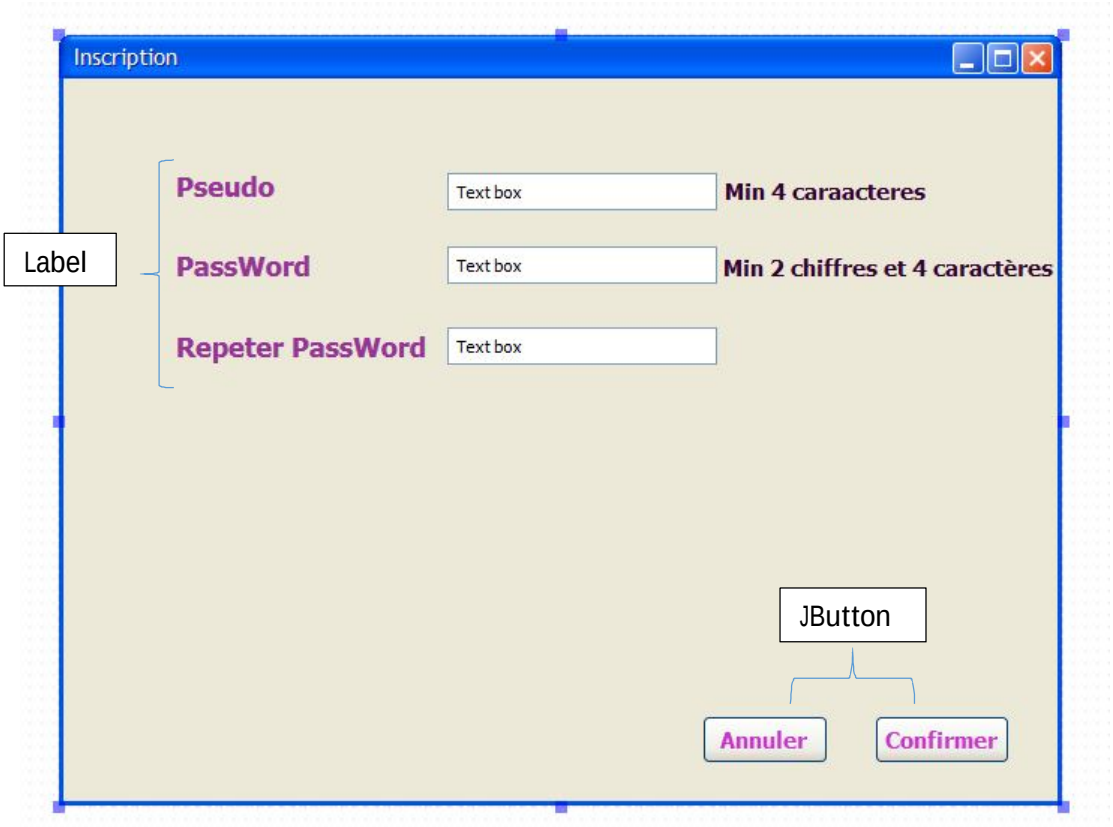


Ihm du bouton login :



Ihm du bouton inscription :






Confirmation de l'inscription
Vous allez recevoir un mail de confirmation dans quelque instant.
OK

Ihm de l'application pour client: