FSBM
FACULTE DES SCIENCES BEN M'SICK
UNIVERSITÉ HASSAN II DE CASABLANCA

PROJET
Annonces immobilières en JEE

MASTER DSBD

Réalisé par :

Lamhour Mohamed Akram

Encadré par :

Prof. Belangour Abdessamad

Année Universitaire :

2020-2021

Chapitre 2 : Analyse et conception.

3- Conception

3.1- Script de base de données

```
/*==============================================================*/
/* Nom de SGBD :  MySQL 5.0                                     */
/* Date de création :  10/01/2021 11:30:22 AM                   */
/*==============================================================*/
drop table if exists Admin;

drop table if exists Adresse;

drop table if exists Annonce;

drop table if exists Appartement;

drop table if exists Favoris;

drop table if exists Image;

drop table if exists Immobilier;

drop table if exists LocalCommercial;

drop table if exists Location;

drop table if exists LocationVacances;

drop table if exists MaisonEtVilla;

drop table if exists Message;

drop table if exists TerrainEtFerme;

drop table if exists User;

drop table if exists Vente;

/*==============================================================*/
/* Table : Admin                                                */
/*==============================================================*/
create table Admin
(
   idUser               int not null,
   idAdmin              int not null,
   primary key (idUser, idAdmin)
);

/*==============================================================*/
/* Table : Adresse                                              */
/*==============================================================*/
create table Adresse
(
   idAdresse            int not null,
   rue                  varchar(254),
   numero               int,
   ville                varchar(254),
   arrondissement       varchar(254),
   quartier             varchar(254),
   primary key (idAdresse),
```

```
   key AK_Identifiant_1 (idAdresse)
);

/*==============================================================*/
/* Table : Annonce                                              */
/*==============================================================*/
create table Annonce
(
   idAnnonce          int not null,
   idImmo             int not null,
   titre              varchar(254),
   description        varchar(254),
   duree              int,
   nbrVue             int,
   etatAnnonce        int,
   dateAjout          datetime,
   primary key (idAnnonce)
);

/*==============================================================*/
/* Table : Appartement                                          */
/*==============================================================*/
create table Appartement
(
   idImmo             int not null,
   idAppartement      int not null,
   salleBain          int,
   salon              int,
   chambre            int,
   cuisine            int,
   numeroEtage        int,
   balcon             bool,
   parking            bool,
   etat               char(9),
   primary key (idImmo, idAppartement)
);

/*==============================================================*/
/* Table : Favoris                                              */
/*==============================================================*/
create table Favoris
(
   idUser             int not null,
   idAnnonce          int not null,
   idFavoris          int not null,
   dateFavoris        datetime,
   primary key (idFavoris),
   key AK_Identifiant_1 (idFavoris),
   key AK_Identifiant_2 (idFavoris)
);

/*==============================================================*/
/* Table : Image                                                */
/*==============================================================*/
create table Image
(
   idImmo             int not null,
   idImage            int not null,
   urlImage           varchar(254),
   primary key (idImage)
);

/*==============================================================*/
/* Table : Immobilier                                           */
/*==============================================================*/
create table Immobilier
(
   idImmo             int not null,
   idAdresse          int not null,
   surface            float,
   prix               int,
```

```
   primary key (idImmo),
   key AK_Identifiant_1 (idImmo)
);

/*==============================================================*/
/* Table : LocalCommercial                                      */
/*==============================================================*/
create table LocalCommercial
(
   idImmo              int not null,
   idLocCom            int not null,
   salleBain           int,
   cuisine             bool,
   numEtage            int,
   etat                char(9),
   primary key (idImmo, idLocCom),
   key AK_Identifiant_1 (idLocCom)
);

/*==============================================================*/
/* Table : Location                                             */
/*==============================================================*/
create table Location
(
   idAnnonce           int not null,
   idLocation          int not null,
   primary key (idAnnonce, idLocation),
   key AK_Identifiant_1 (idLocation)
);

/*==============================================================*/
/* Table : LocationVacances                                     */
/*==============================================================*/
create table LocationVacances
(
   idAnnonce           int not null,
   idLocVac            int not null,
   primary key (idAnnonce, idLocVac),
   key AK_Identifiant_1 (idLocVac)
);

/*==============================================================*/
/* Table : MaisonEtVilla                                        */
/*==============================================================*/
create table MaisonEtVilla
(
   idImmo              int not null,
   idMaison            int not null,
   salleBain           int,
   salon               int,
   chambre             int,
   cuisine             int,
   nombreEtage         int,
   etat                char(9),
   primary key (idImmo, idMaison)
);

/*==============================================================*/
/* Table : Message                                              */
/*==============================================================*/
create table Message
(
   idMessage           int not null,
   idUser              int not null,
   Use_idUser          int not null,
   content             varchar(254),
   etat                bool,
   date                datetime,
   primary key (idMessage),
   key AK_Identifiant_1 (idMessage),
   key AK_Identifiant_2 (idMessage)
```

```
);

/*==============================================================*/
/* Table : TerrainEtFerme                                       */
/*==============================================================*/
create table TerrainEtFerme
(
   idImmo               int not null,
   idTf                 int not null,
   type                 varchar(254),
   primary key (idImmo, idTf),
   key AK_Identifiant_1 (idTf)
);

/*==============================================================*/
/* Table : User                                                 */
/*==============================================================*/
create table User
(
   idUser               int not null,
   idAdresse            int not null,
   nom                  varchar(254),
   prenom               varchar(254),
   age                  int,
   login                varchar(254),
   password             varchar(254),
   cin                  varchar(254),
   tel                  bigint,
   dateInscription      datetime,
   primary key (idUser)
);

/*==============================================================*/
/* Table : Vente                                                */
/*==============================================================*/
create table Vente
(
   idAnnonce            int not null,
   idVente              int not null,
   primary key (idAnnonce, idVente),
   key AK_Identifiant_1 (idVente)
);

alter table Admin add constraint FK_Generalisation_8 foreign key (idUser)
      references User (idUser) on delete restrict on update restrict;

alter table Annonce add constraint FK_concerver foreign key (idImmo)
      references Immobilier (idImmo) on delete restrict on update restrict;

alter table Appartement add constraint FK_Generalisation_3 foreign key (idImmo)
      references Immobilier (idImmo) on delete restrict on update restrict;

alter table Favoris add constraint FK_consulter foreign key (idAnnonce)
      references Annonce (idAnnonce) on delete restrict on update restrict;

alter table Favoris add constraint FK_consulter foreign key (idUser)
      references User (idUser) on delete restrict on update restrict;

alter table Image add constraint FK_contient foreign key (idImmo)
      references Immobilier (idImmo) on delete restrict on update restrict;

alter table Immobilier add constraint FK_possede foreign key (idAdresse)
      references Adresse (idAdresse) on delete restrict on update restrict;

alter table LocalCommercial add constraint FK_Generalisation_1 foreign key (idImmo)
      references Immobilier (idImmo) on delete restrict on update restrict;

alter table Location add constraint FK_Generalisation_6 foreign key (idAnnonce)
      references Annonce (idAnnonce) on delete restrict on update restrict;

alter table LocationVacances add constraint FK_Generalisation_7 foreign key (idAnnonce)
```

```
    references Annonce (idAnnonce) on delete restrict on update restrict;

alter table MaisonEtVilla add constraint FK_Generalisation_2 foreign key (idImmo)
    references Immobilier (idImmo) on delete restrict on update restrict;

alter table Message add constraint FK_envoyer foreign key (idUser)
    references User (idUser) on delete restrict on update restrict;

alter table Message add constraint FK_recevoir foreign key (Use_idUser)
    references User (idUser) on delete restrict on update restrict;

alter table TerrainEtFerme add constraint FK_Generalisation_4 foreign key (idImmo)
    references Immobilier (idImmo) on delete restrict on update restrict;

alter table User add constraint FK_possede foreign key (idAdresse)
    references Adresse (idAdresse) on delete restrict on update restrict;

alter table Vente add constraint FK_Generalisation_5 foreign key (idAnnonce)
    references Annonce (idAnnonce) on delete restrict on update restrict;
```

3.2- Conclusion

À travers ce chapitre, nous avons vu les différentes phases de pré-développement, qui sont l'analyse et la conception, ces deux phases qui définissent l'application, et permettent de parler de la spécification des besoins verbaux en une application, et nous avons vu aussi les différents diagrammes utilisés pour avoir une bonne idée sur le fonctionnement de l'application, avant même de démarrer la phase de développement.

3.3- Architecture physique :

Notre application se présente sous la forme d'une architecture trois tiers ou ce qu'on appelle également architecture à trois niveaux. L'architecture trois tiers est l'application du modèle le plus général qui est le multi-tiers et c'est également une extension du modèle Client/serveur.

Plus spécifiquement c'est une architecture partagée entre :

- ✓ Un client : L'ordinateur demandeur de ressources, équipé d'une interface utilisateur (généralement un navigateur web) chargé de la présentation.
- ✓ Un serveur d'application : Chargé de fournir la ressource mais faisant appel à un autre serveur.
- ✓ Un serveur de base de données : Fournissant au serveur d'application les données dont il a besoin. Etant donné l'emploi massif du terme de l'architecture à 3 niveaux, celui-ci peut parfois désigner aussi les architectures suivantes :

  - Partage d'application entre client, serveur intermédiaire, et serveur d'entreprise.
  - Partage d'application entre client, serveur d'application, et serveur de base de données de l'entreprise.

3.4- Architecture physique adoptée :

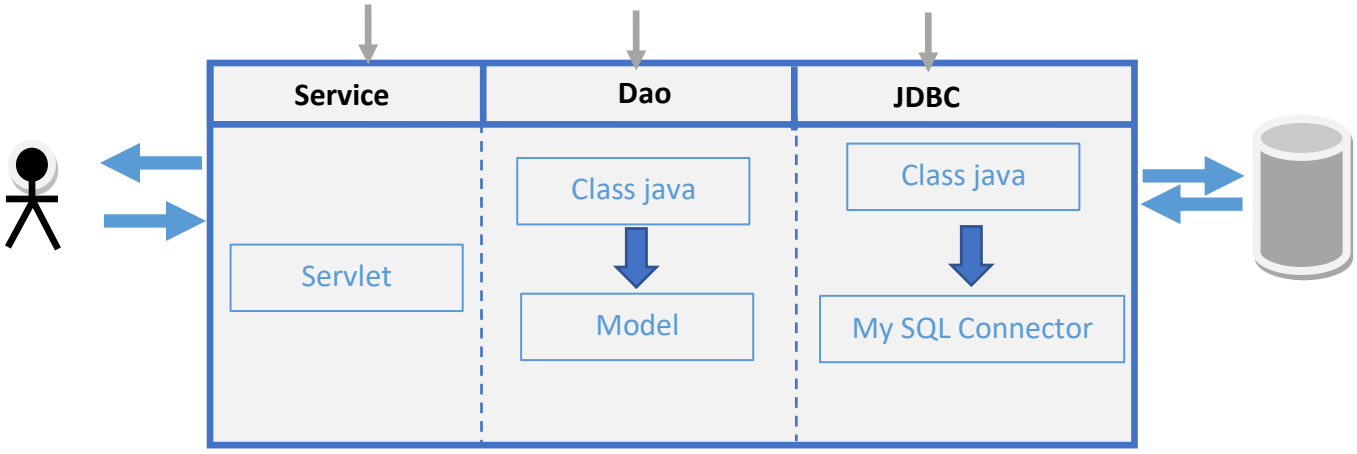
Pour mieux s'expliquer notre application se base sur deux serveurs :

- Serveur de base de données -MySQL : MySQL est un serveur de bases de données relationnelles

SQL, très rapide, multithread, robuste et multiutilisateurs. MySQL est un logiciel libre développé sous double licence GPL (General Public License) et licence commerciale. Il est le serveur de base de données le plus utilisé dans le monde. Il fonctionne sur beaucoup de plates-formes différentes et il est accessible en utilisant plusieurs langages de programmation.

- Serveur HTTP - Apache : Apache HTTP Server est un serveur HTTP créé et maintenu au sein de la fondation Apache. C'est le serveur HTTP le plus populaire du WorldWideWeb.

Présentant alors l'architecture matérielle de notre application par un diagramme de déploiement qui englobe les nœuds correspondant aux supports physiques.

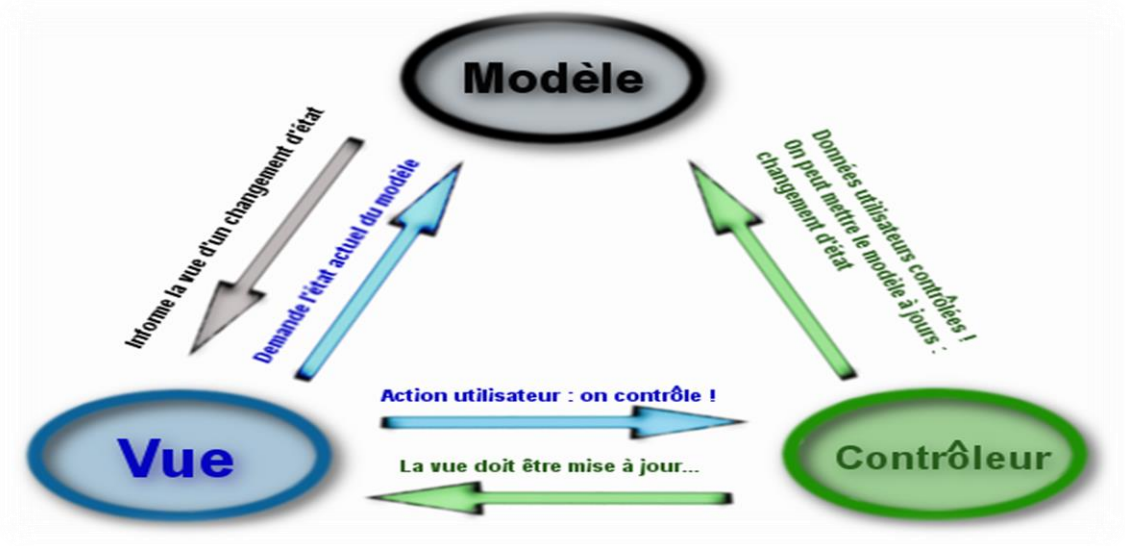


3.5- Architecture logique :

3.5.1- Design pattern MVC :

L'architecture vise à ce que l'application soit la plus maintenable possible. Dans ce cadre le Framework utilisé s'est orienté vers l'architecture MVC. Ce modèle d'architecture impose la séparation entre les données, la présentation et les traitements, ce qui donne trois parties fondamentales dans l'application finale : le modèle, la vue et le contrôleur.

- Le Modèle : Présente le comportement de l'application : traitements des données, interactions avec la base de données, etc.
- La Vue : Correspond à l'interface avec laquelle l'utilisateur interagit. Sa première tâche est de présenter les résultats renvoyés par le modèle. Sa seconde tâche est de recevoir toutes les actions de l'utilisateur (clic de souris, bouton, . . .).
- Le Contrôleur : Prend en charge la gestion des événements de synchronisation pour mettre à jour la vue ou le modèle et les synchroniser. Il reçoit tous les événements de l'utilisateur et enclenche les actions à effectuer.

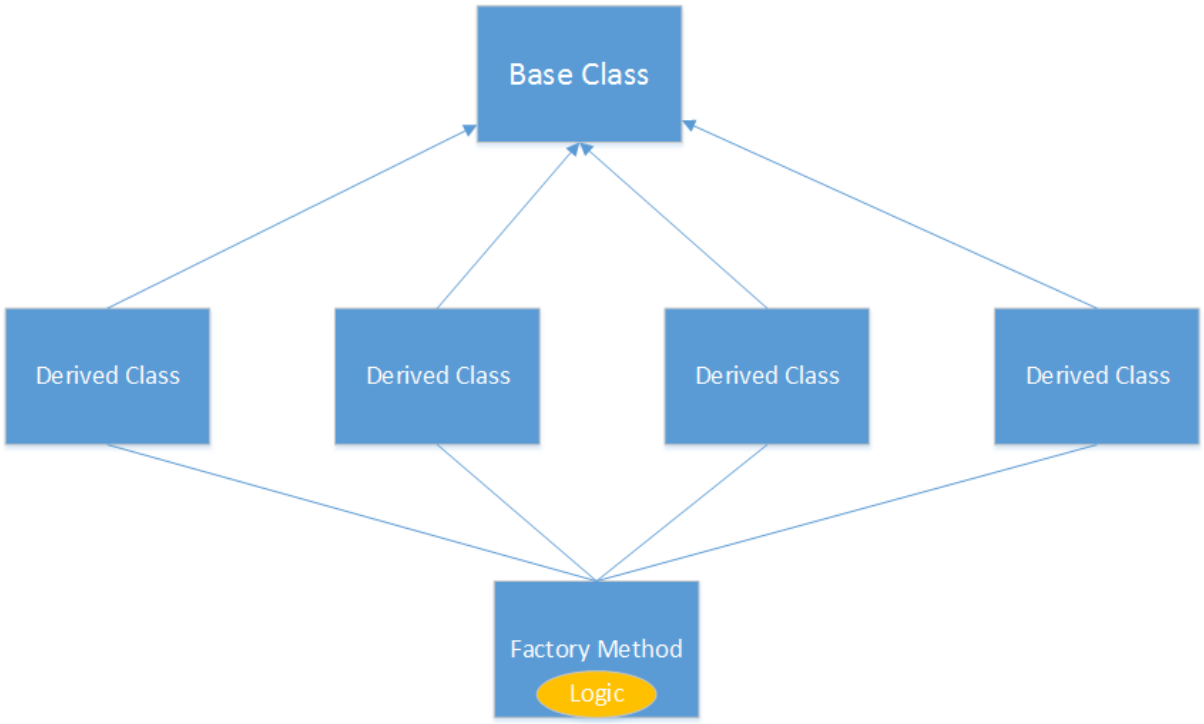


➢ Avantages du MVC :

- Vitesse de création de pages.
- Gain de temps de maintenance.
- Simplicité de mise à jour.
- Clarté de l'architecture qu'il impose grâce à la séparation des données de la vue et du contrôleur.

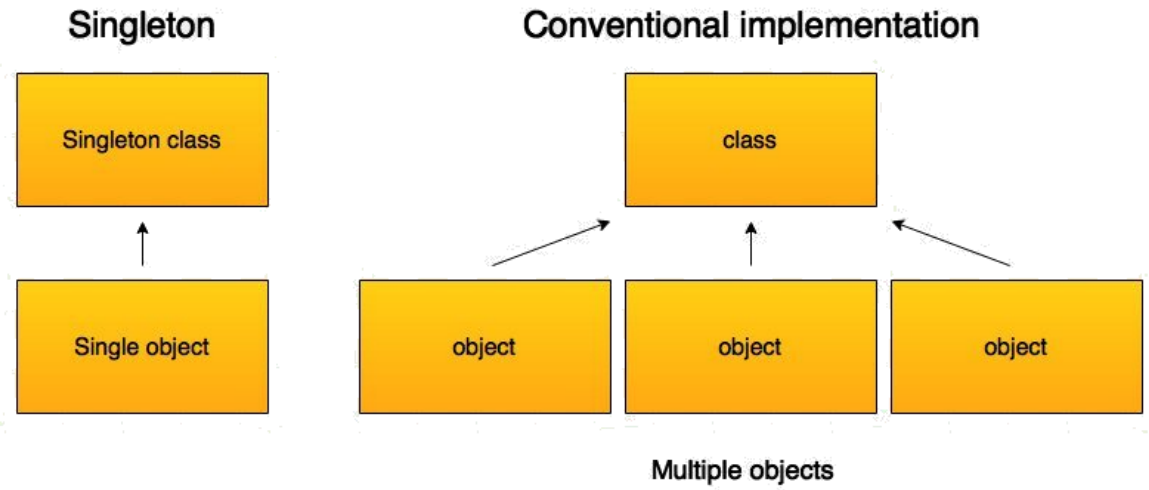
3.5.2- Design pattern factory :

Le design pattern Factory, ou Fabrique est un design pattern permettant de séparer la création d'objets dérivant d'une classe mère de leur utilisation. De ce fait, on a alors la possibilité de créer plusieurs objets issus d'une même classe mère. Les fabriques étant en général uniques dans un programme, on utilise souvent le patron de conception singleton pour les implémenter.



3.5.3- Design pattern singleton

En génie logiciel, le singleton est un patron de conception *(design pattern)* dont l'objectif est de restreindre l'instanciation d'une classe à un seul objet (ou bien à quelques objets seulement). Il est utilisé lorsqu'on a besoin exactement d'un objet pour coordonner des opérations dans un système. Le modèle est parfois utilisé pour son efficacité, lorsque le système est plus rapide ou occupe moins de mémoire avec peu d'objets qu'avec beaucoup d'objets similaires.



4- Conclusion

Ce chapitre a été consacré pour la spécification des besoins fonctionnels et non fonctionnels du système résultant, ce qui correspondait aux différentes activités de la première phase du cycle de développement du notre système. Dans le chapitre suivant, nous étudierons la phase de conception. Dans le chapitre suivant, nous allons présenter la conception qui a été mise en œuvre tout au long de la réalisation de ce projet.