RAPPORT DE STAGE CDI

"La plus grande attention doit être portée à la compréhension du problème, faute de quoi l'algorithme n'a aucune chance d'être correct". Denis Lapoire

"C'est toujours l'impatience de gagner qui fait perdre", Louis XIV cité dans "L'immortel" de FOG.

J'écoute et j'oublie.
Je lis et je retiens.
Je fais et j'apprends.
(Proverbe chinois)

Version du document : 0.1.1

Date de création : 10 mars 2017

Date de dernière mise à jour : 10 mars 2017

SOMMAIRE

CHAPITRE 1 - PRÉSENTATION GÉNÉRALE

1.1 - Avant-propos

Initialement venu de l'hotellerie et débutant en informatique au début de cette formation j'ai pu apprécier la puissance du développement ainsi que de la conception.L'UML ainsi que les languages de programmations ont été une confirmation sur ce que la nouvelle direction que je veux donner à ma carrière professionnelle.

La société Robin est une société immobilière spécialisée dans la gestion de locaux et buraux commerciant,le gérant m'a sollicité pour la création d'un site mettant en valeur ses biens et pouvant aussi permettre un suivi des visiteurs par des inscriptions sur le site

Le projet est donc un site internet permettant aux locataires ou prospects de s'inscrire sur le site et donc de consulter son compte ou de rechercher des biens .Le gérant à ainsi une visibilité sur les locaux à louer déjà occupés et ainsi que des informations complètes sur les inscrits

1.2 - Abstract

I create a website for the company Robin .This website.is the solution for this compagny to

show and help to rent the building.By travelling on the website the visitor can see the

different kinds of rentings the company propose and can also create an account to be

advised of the news.By this this time every one is using internet ,this website is a very good

solution to make the people discover this company and having new customers

.

1.3 - Remerciements

Je tiens à remercier M2i Formation ainsi que mes principaux formateurs Pascal Buguet et Sébastien Maloron pour leurs enseignements ainsi que leur pédagogie et leur aide comme tuteurs lorsque j'étais stagiaire « en entreprise ».

Je tiens aussi à remercier mes parents qui m'ont soutenu et aidé durant toute la formation et le stage « entreprise » et qui se sont beaucoup occupé de ma fille de 2 ans dont j'ai la garde.

Je remercie aussi le gérant de la société « Immo Robin » qui m'a accepté comme stagiaire et m'a fait confiance et continuera à me faire confiance puisqu'il décide de me laisser terminer les logiciels de son entreprise.

1.4 - La démarche générale

La démarche présentée en cours, inspirée de Pascal Roques, est celle que j'ai utilisée.
On part des besoins du client pour arriver au codes des logiciels.
Pour cela il faut passer par plusieurs étapes qui permettent de décomposer les problèmes en plus petits problèmes.

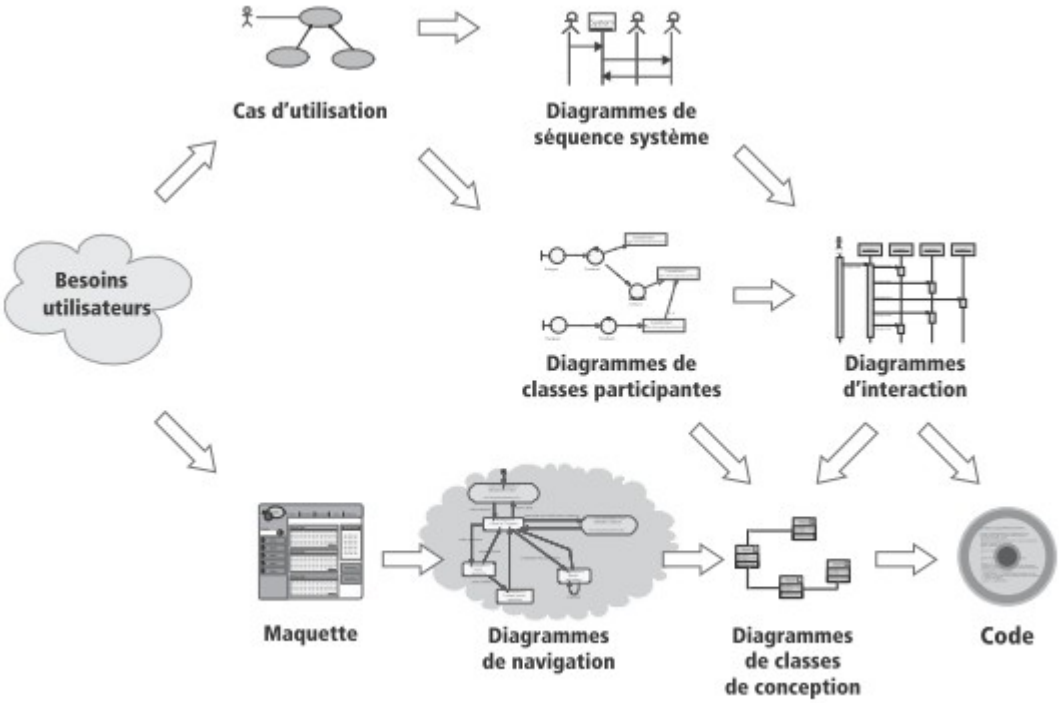


Figure 1–20 Schéma complet du processus de modélisation d'une application web

CHAPITRE 2 - CAHIER DES CHARGES

- Cahier des charges

Le cahier des charges à été défini avec le gérant de la société en partant de ses besoins.

Il définit les besoins du client et les contraintes imposées.

Besoins :

Un site web pour présenter l'activité de l'agence.
Le site web doit permettre de toucher de nouveaux clients potentiels.
Le site web doit permettre de présenter le catalogue des biens disponible à la location.
Le site web doit permettre aux clients et aux potentiels clients de créer et de gérer un compte personnel.

Un logiciel pour gérer l'activité de l'agence.
Il doit permettre de gérer les biens de l'agence.
Il doit permettre de gérer les clients (contrats, quittances, relances, ...) et les prospects de l'agence.

Contraintes :

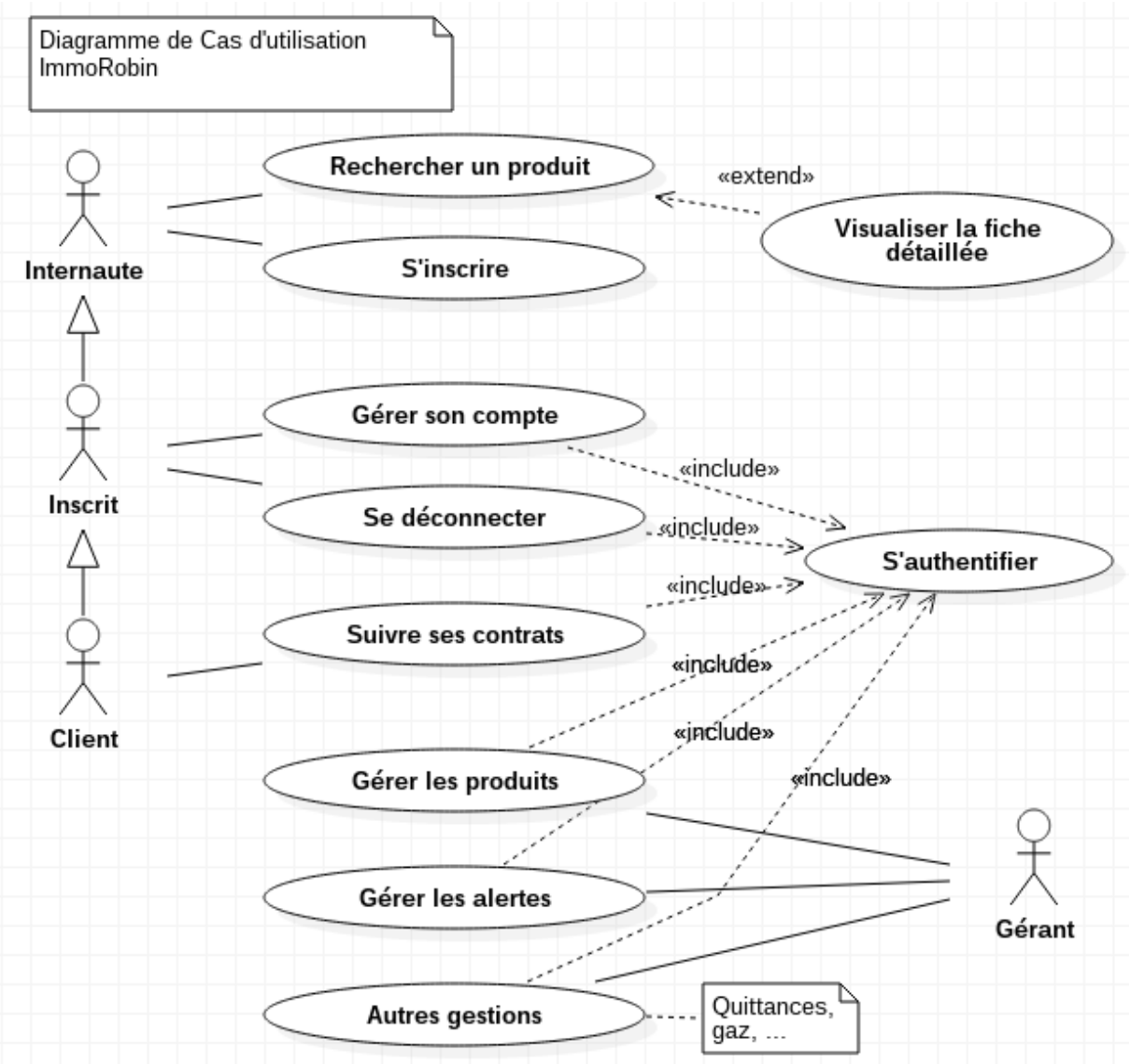
Site web développé en 2 mois.
Coût du développement de 0 euros en première phase.
Hébergeur d'un coût raisonnable.
Interface facile pour le gérant pour gérer le contenu (biens, contrats, quittances).

CHAPITRE 3 - ANALYSE

3.1 - Le Diagramme de Cas d'Utilisation

3.1.1 - Représentation graphique

Un diagramme de cas d'utilisation (DCU) est un ensemble de cas d'utilisation (CU). Il formalise graphiquement les besoins des utilisateurs.



Mots clés :

Diagramme de CU, CU, acteur, association, inclusion, extension, héritage

description :
Ce diagramme nous montre les différentes fonctionnalités du logiciel:ici on peut voir les différents acteurs. On voit que certains acteurs héritent des fonctionnalités d'autres acteurs.
Les acteurs sont associés à des cas d'utilisation,certains cas d'utilisation dépendent obligatoirement d'autres cas d'utilisation(include),c'est une factorisation(mise en commun) de fonctionnalités.Il y a d'autres CU qui eux sont optionnels(extend)
par exemple visualiser la fiche détaillée.

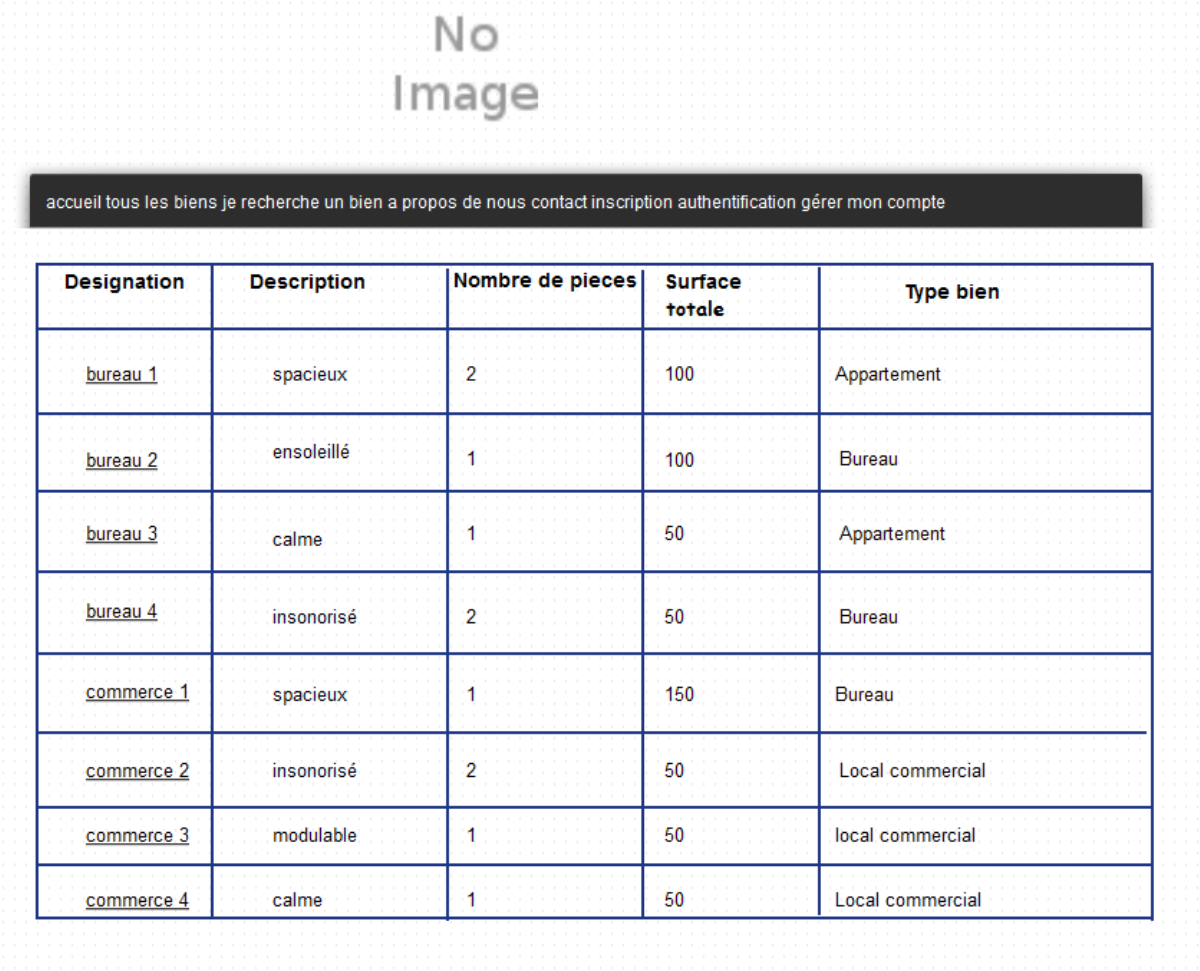
3.1.2 - Fiche de description textuelle d'un cas d'utilisation

Etapes	Description
Identification du CU	Titre : gérer un compte pour un client Résumé : visualiser, modifier, supprimer un compte Acteur : client Date de création : 10 avril 2017 Date de dernière modification : Version : 1.0.0 Auteur : Thomas Escouba
Pré-conditions	Authentification IHM disponible Connexion à la BD effective
Scenario nominal	Le client visualise les informations de son compte
Scenarii alternatifs	Le client modifie les informations de son compte Le client supprime les informations de son compte
Scenarii d'erreurs	Les données saisies sont non conformes
Post-conditions	Compte mis à jour
Exigences non fonctionnelles	Temps de réponse de 2 secondes maxi
Besoins d'IHM	Page web

3.2 - Les maquettes

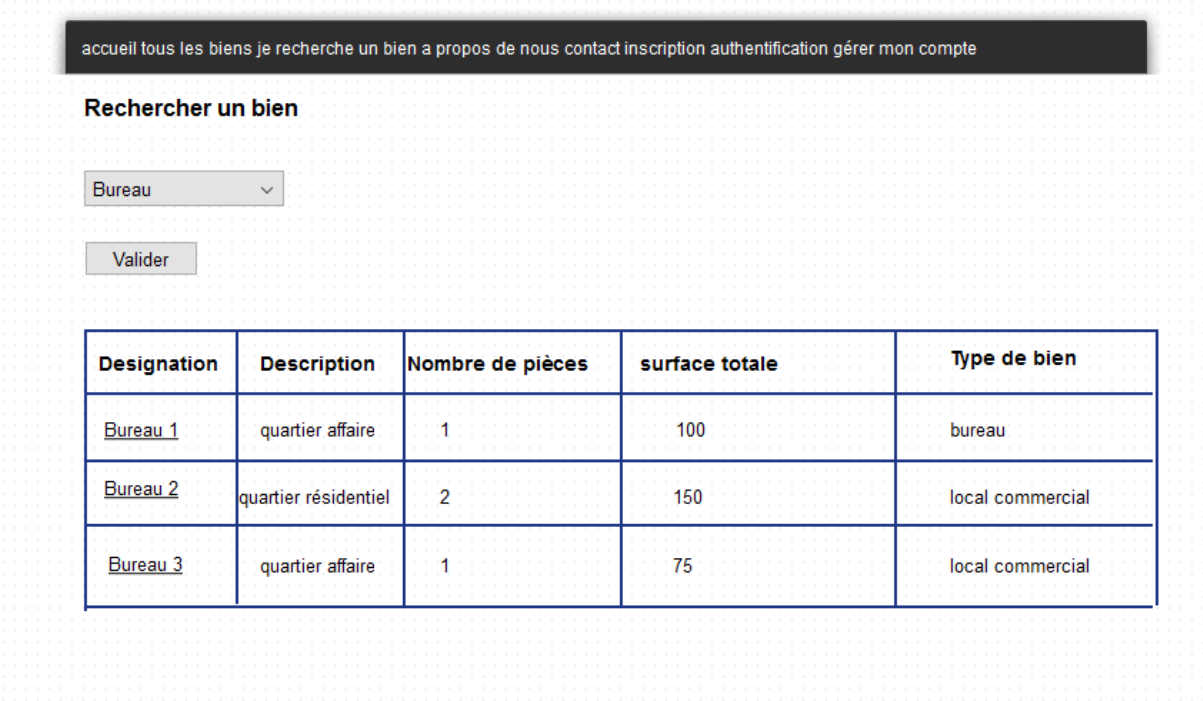
Les maquettes ont été réalisées avec Pencil Evolus et nous permettent de valider les différents écrans du logiciel avec le client .C'est à partir des cas d'utilisation que nous avons réalisé ces maquettes,par la suite nous allons établir le diagramme de navigation qui nous permettra avec le client de valider la cinématique du site.
Les maquettes nous permettrons par la suite de coder en HTML et CSS et eventuellement JavaScript.

3.2.1 - Maquette « Tous les biens »

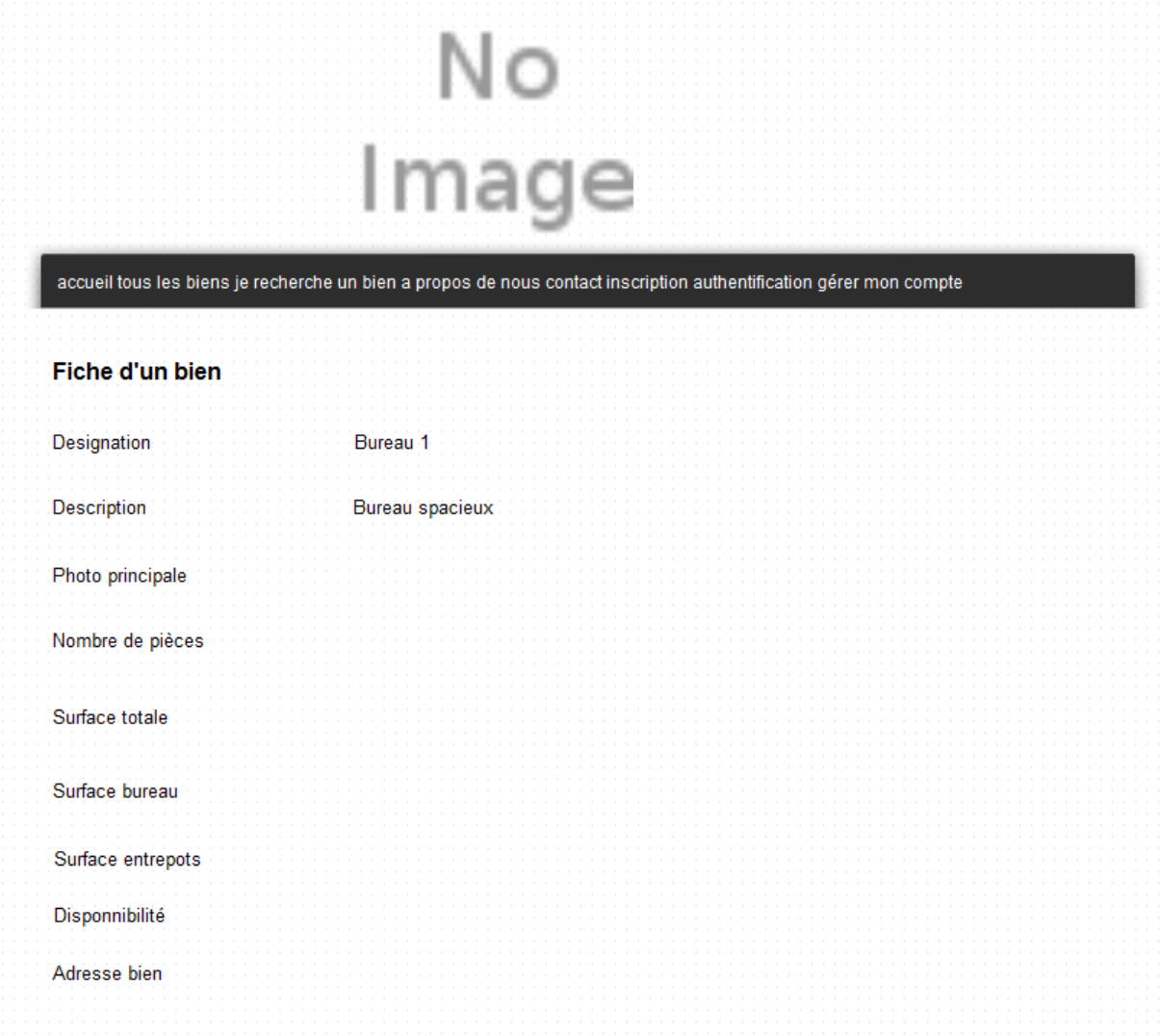


Designation	Description	Nombre de pieces	Surface totale	Type bien
bureau 1	spacieux	2	100	Appartement
bureau 2	ensoleillé	1	100	Bureau
bureau 3	calme	1	50	Appartement
bureau 4	insonorisé	2	50	Bureau
commerce 1	spacieux	1	150	Bureau
commerce 2	insonorisé	2	50	Local commercial
commerce 3	modulable	1	50	local commercial
commerce 4	calme	1	50	Local commercial

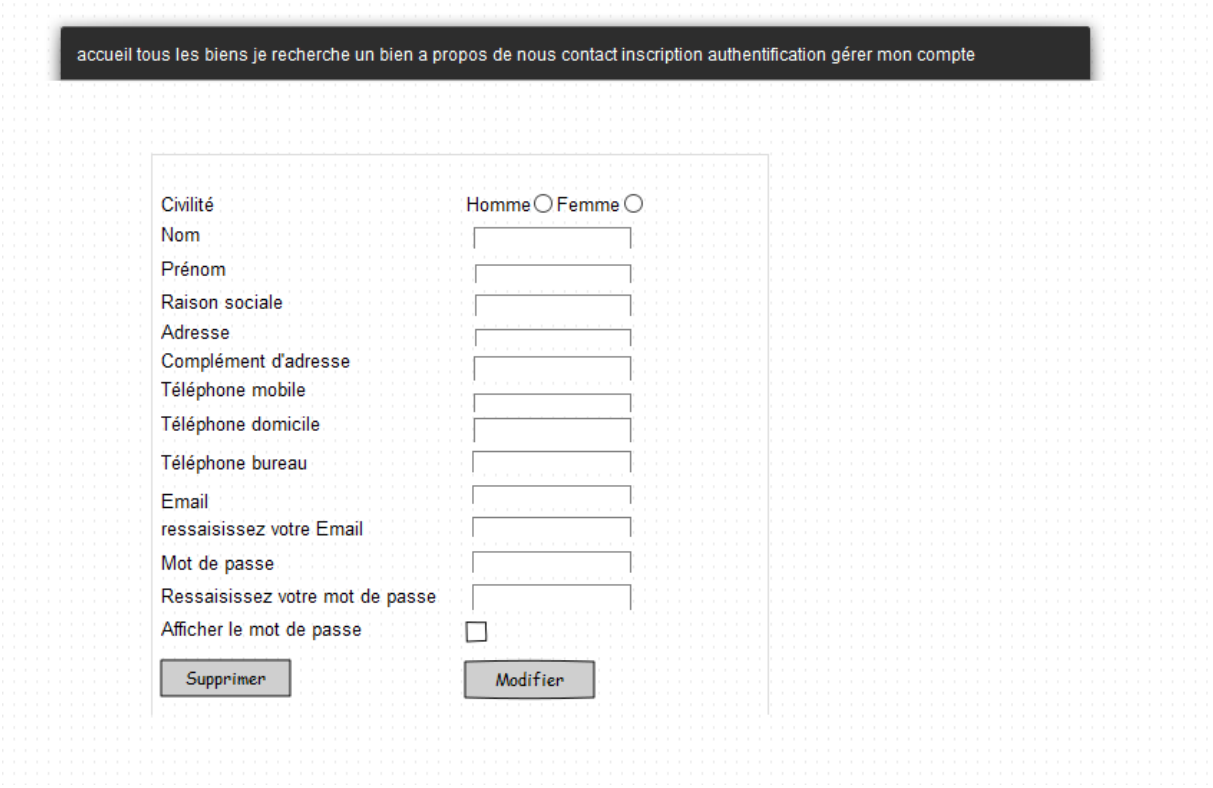
3.2.2 - Maquette « Rechercher un bien »



3.2.3 - Maquette « Un bien »

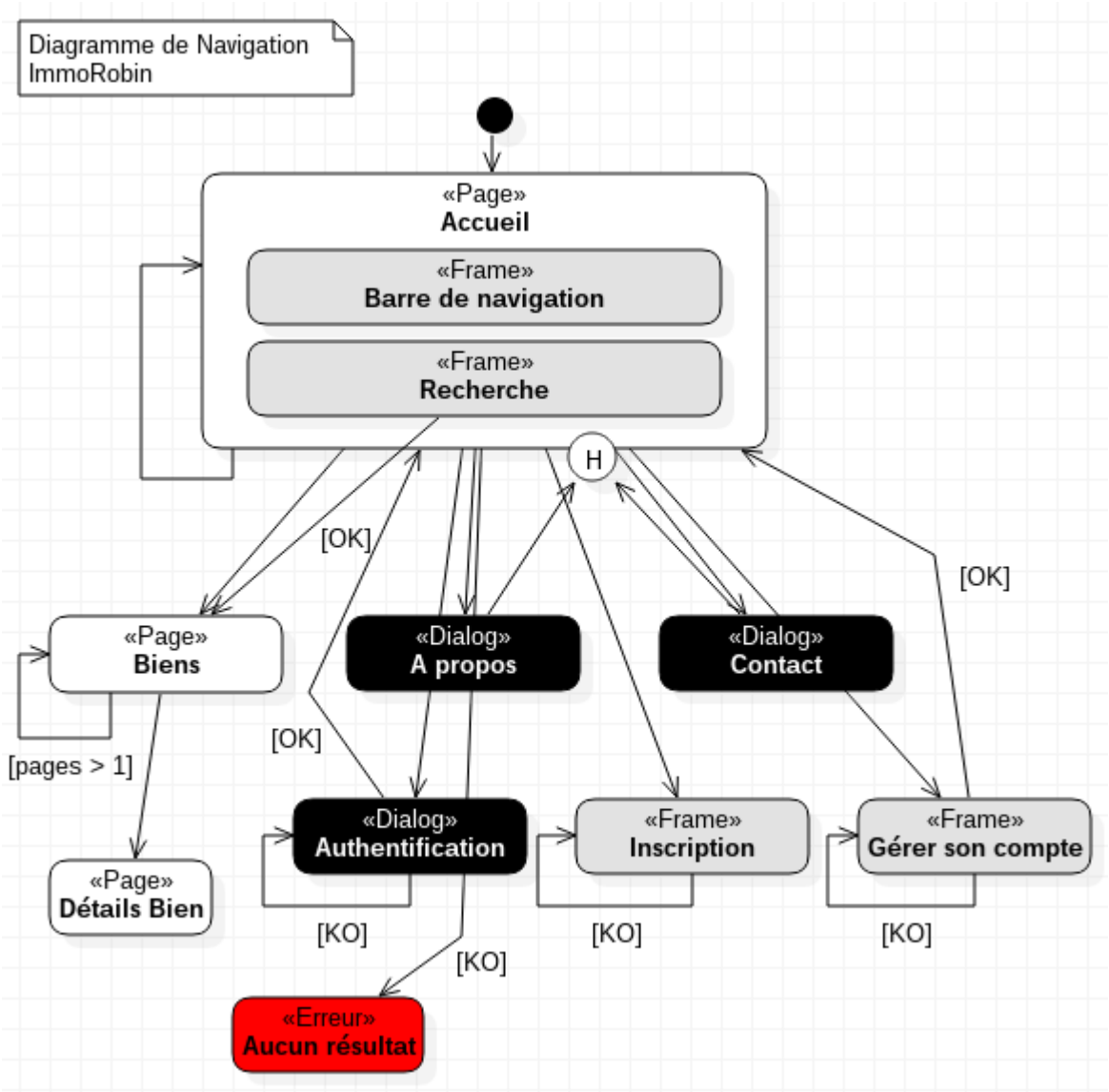


3.2.4 - Maquette « Gérer mon compte »



3.3 - Le Diagramme de Navigation

Le diagramme de navigation représente la cinématique des écran,comme écris précédemment il dérive en partie des différentes maquettes du paragraphe précédent.Il nous permettra ensuite de coder les différents liens et formulaires HTML .



Mots clés : page, frame, stéréotype, transition, garde, historique récent (shallow history), ...

Description :
ce diagramme nous montre comment l'internaute peut surfer entre les pages du site.Les pages sont soit monoblock ou multiblock.Les pages multiblock incluent des frames que l'on retrouve dans plusieures pages(ex menu,entete).Les transitions représentent, le passage d'une page à une autre,dans un site web c'est réalisé

3.4 - Les Diagrammes de Séquence Système

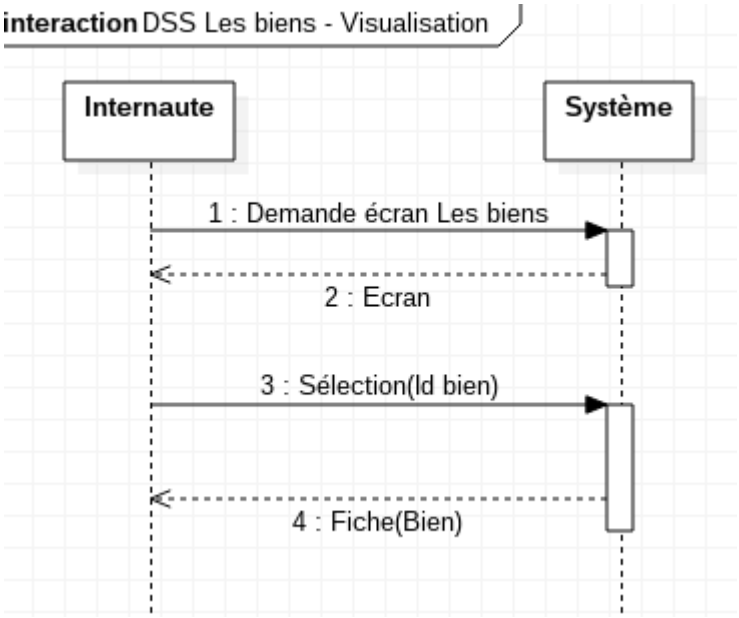
3.4.1 - Définition

Un Diagramme de Séquence Système (DSS) est une représentation graphique d'un CU ou même d'un scénario d'un CU. Un DSS ne comporte que 2 objets et 2 lignes de vie. C'est un diagramme interactif qui possède 2 axes : l'espace et le temps.

Le choix des présentations :

Les biens : un selectAll() et un selectOne().
Gérer mon compte : un selectOne(), un update() et un delete().

3.4.2 - Diagramme de la séquence « Consulter les biens »



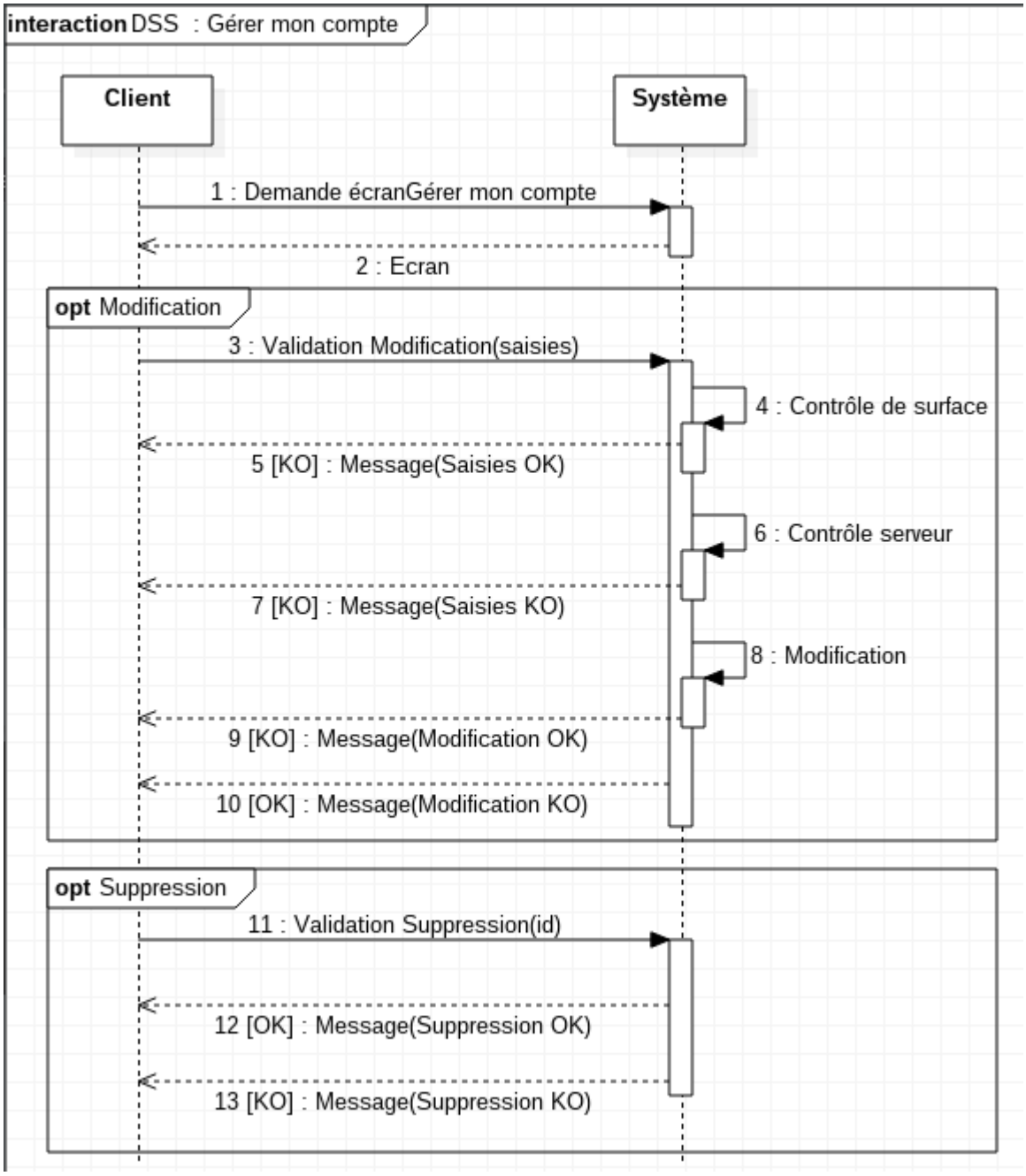
Mots clés :
objet, ligne de vie, message (synchrone, asynchrone, retour, création, destruction), paramètre de message, activation.

Description :

la première requête de l'utilisateur est de demander l'écran de visualisation de tous les biens. C'est une requête synchrone. Le système exécute le code d'extraction des données de la BD et la production de la page. Il est actif pendant ce temps-là. Ensuite le système est « au repos », il est inactif.

L'utilisateur, s'il le veut, peut obtenir une description détaillée.
Il effectue une 2e requête. La requête est accompagné d'un paramètre (l'ID du bien).
Le système extrait les données correspondantes puis prépare la page les informations du bien demandé.

3.4.3 - Diagramme de la séquence « Gérer mon compte »



Mots clés : les mêmes.

Description :

A la différence du DSS il présente plus de complexité.

Plusieurs cas optionnels sont présents en fonction des choix de l'utilisateur (il veut modifier son compte, il veut supprimer son compte, ...).

Plusieurs cas d'erreurs sont possibles.
L'utilisateur peut ne pas saisir toutes les informations nécessaires.
Il peut saisir des informations qui ne sont conformes au format attendu (email, mot de passe, ...).
Il peut saisir des informations qui sont incompatibles avec certaines règles (pseudo déjà existant).

3.5 - Les Diagrammes d'Activité

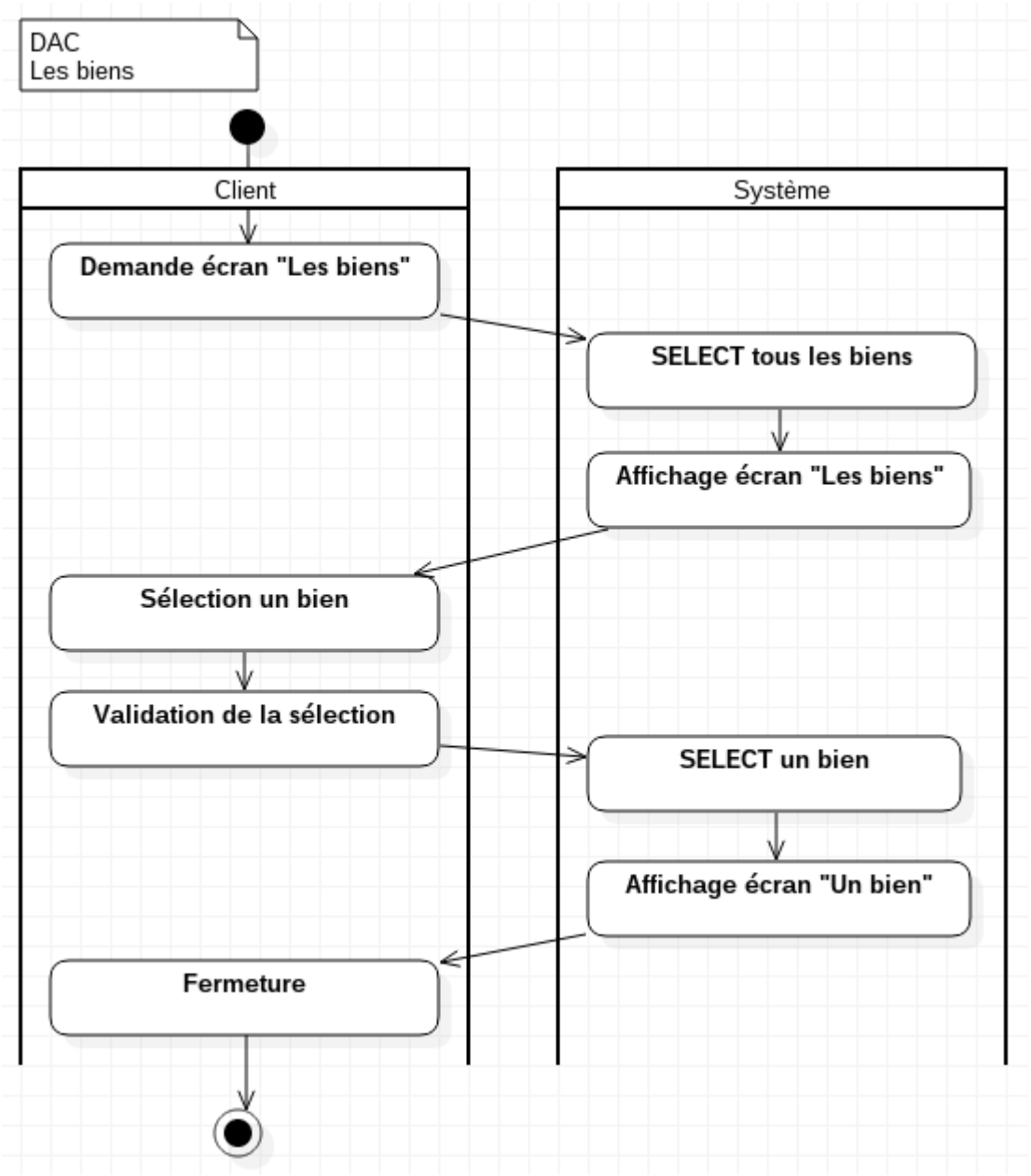
Définition :

Le diagramme d'activité représente graphiquement le déroulé d'un cas d'utilisation du point de vue de l'utilisateur.

Les composants essentiels du DAC sont les actions et les transitions.

C'est l'observation de l'utilisateur, si c'est possible, effectuant une tâche qui permet de le construire.

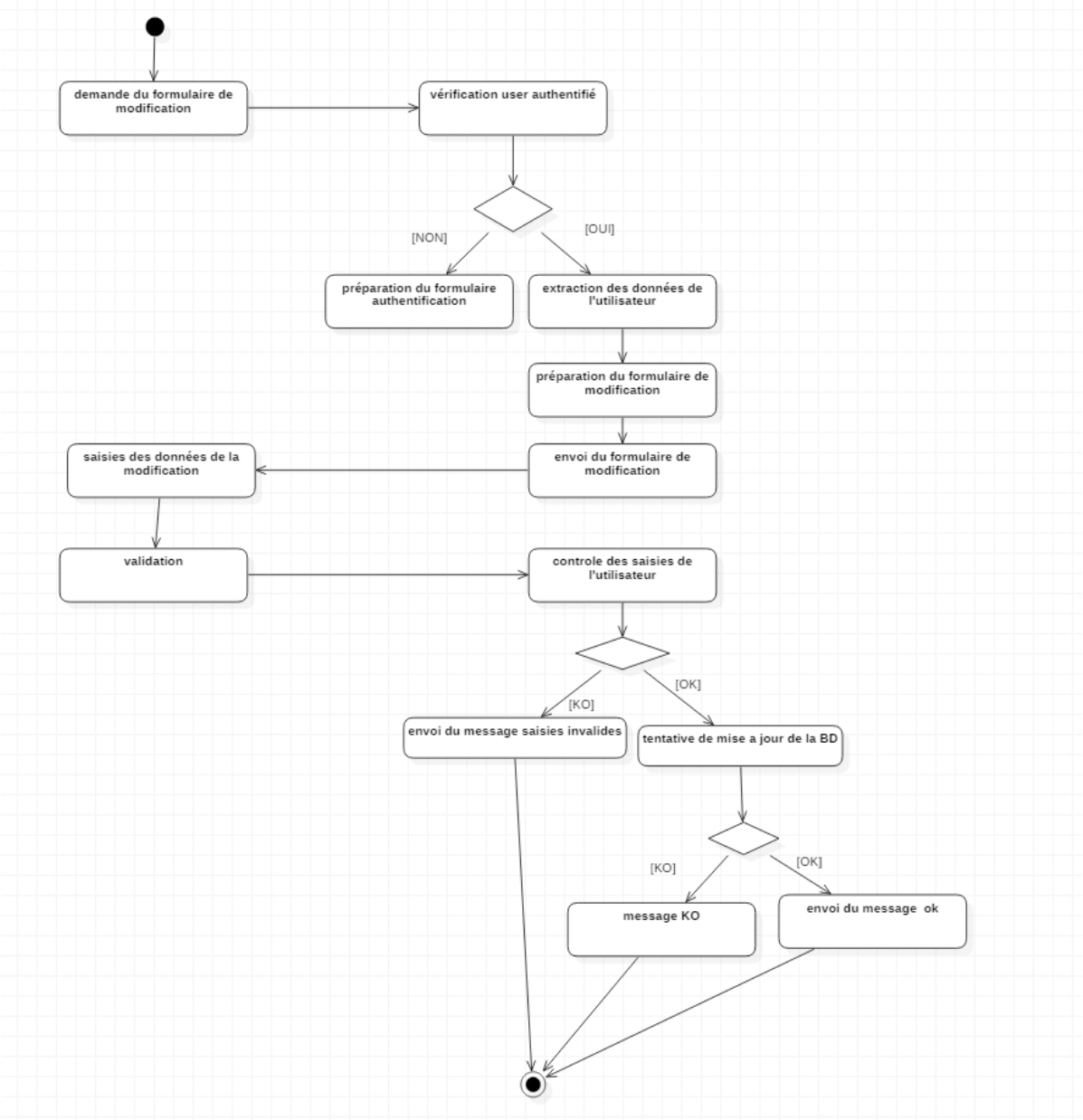
3.5.1 - Diagramme de l'activité « Consulter les biens »



Mots clés : ...
activité,final,gardes,init,synchronisations,flow d'objets

Description :
Ce diagramme nous montre le déroulé du cas d'utilisation « consulter les biens ».Il commence par un début (init) et se termine par le final.
Ce diagramme est organisé en couloirs qui mettent en évidence les activités du client ainsi que celle du systeme.Ces activités s'exécutent par des transitions.

3.5.2 - Diagramme de l'activité « Gérer mon compte »



Mots clés :les mêmes.

Description :

Ce diagramme nous montre le déroulé du cas d'utilisation « gérer mon compte ». A la différence du précédent, il nous montre des cas alternatifs (un losange représente une condition) et donc des transitions avec des gardes (oui/non, ok/ko) en fonction des résultats des tests et des chemins parallèles. Il permet de faire la synthèse des différents scénarios du cas d'utilisation « gérer mon compte »

Il présente un bon intérêt par rapport au DAC précédent de ce point de vue.

CHAPITRE 4 - CONCEPTION DE LA BASE DE DONNÉES

4.1 - La démarche utilisée

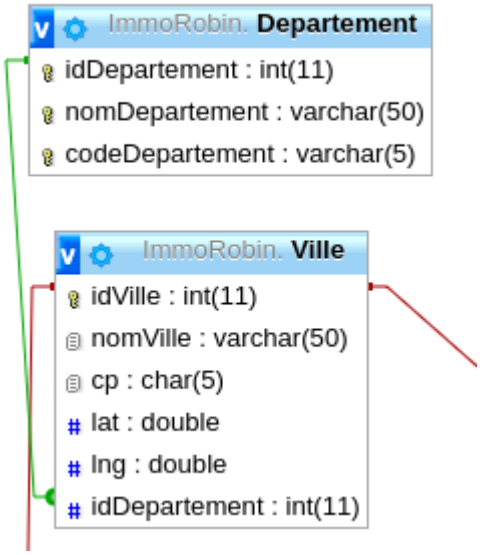
Pour concevoir et construire la base de donnée,on est parti d'échanges avec le client dans un premier temps et des maquettes validées par le client dans un 2eme temps.
Le cahier des charges nous a permis de déterminer une liste d' « objets »et de relations entre ces « objets ».Par exemple des « objets »biens,catégories de biens,lieux,personnes,catégories de personnes.Ces « objets » vont correspondre à des tables de la base de donnée.
Les maquettes nous ont permis de lister presque toutes les informations élémentaires qui doivent être stockées dans la base de données. Elles ont été ajoutées aux tables repérées précedemment

4.2 - Le MPD (Schéma de la BD,modèle physique de données)

Il représente la base de données stockée et gérée avec MySql.
La BD est divisée en 4 sous domaines:lieux,biens,personnes,locations.

4.2.1 - La BD de base

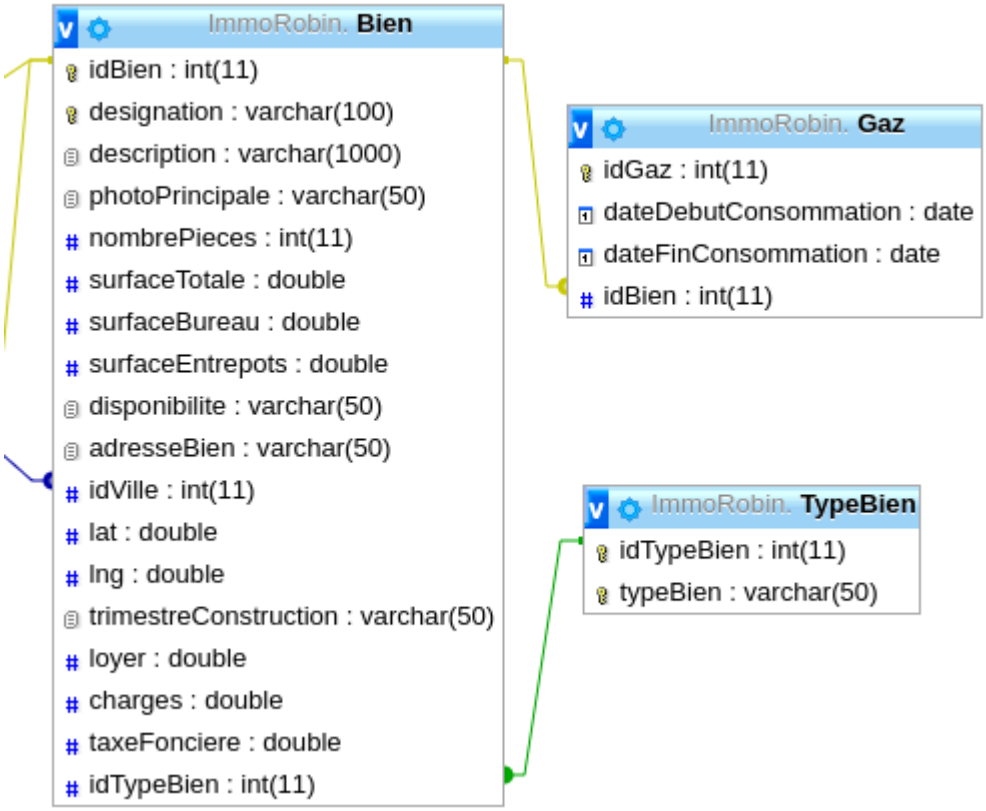
4.2.1.1 - Sous-domaine lieux



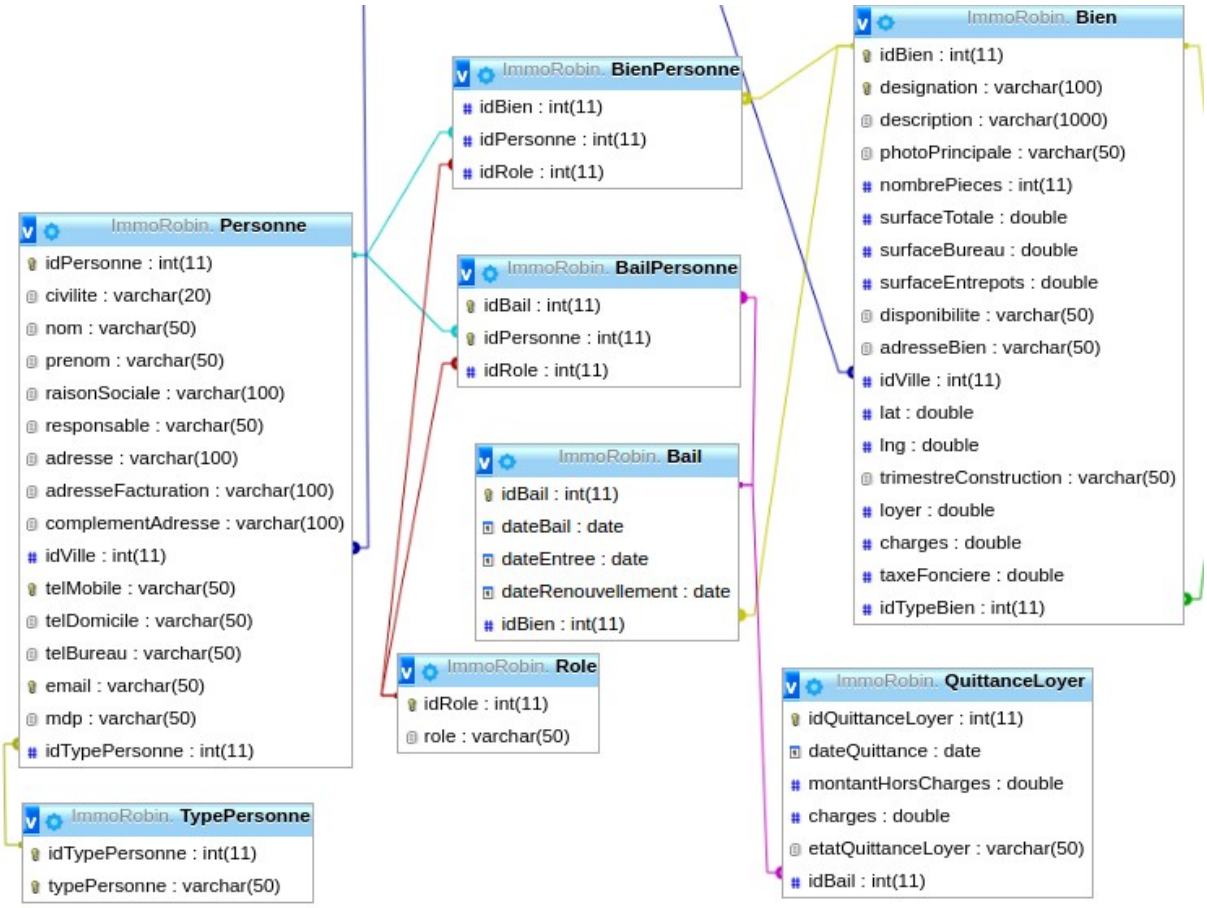
Ce schéma représente 2 tables contenant diverses colonnes.Chaque colonne est caractérisée par un type(numérique ou texte).
Les 2 tables présentées disposent chacune d'une clé primaire(idDepartement,idVille). La table département possède.2 colonnes avec des index uniques (nom et code), la table ville possède une clé étrangère indexée.Les 2 tables sont reliées entre elles. La liaison est réalisée entre la clé primaire de « département »(idDepartement) et la clé étrangère de « ville » (idDepartement).
On retrouve les mêmes types de caractéristiques dans les schémas qui suivent.

4.2.1.2 - Sous-domaine personnes

4.2.1.3 - Sous-domaine biens



4.2.1.4 - Sous-domaine locations



Tables à ajouter ?

4.2.1.5 Extension de la BD

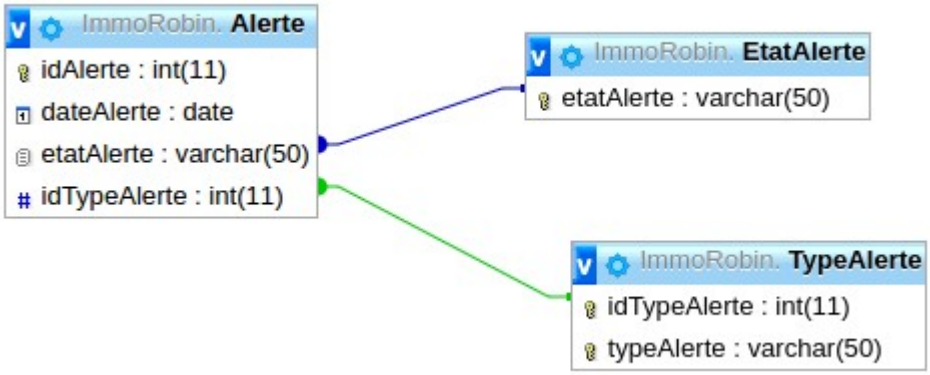
ContratLocation(idContratLocation,dateContratLocation,dureeLocation,idBien,idPersonne , ..)
QuittanceLoyer(idQuittance,dateQuitance,idBien,idPersonne,montantLoyer,...)
Relance(idRelance,dateRelance,idPersonne,idBien,...)

Colonnes à ajouter ?

Table	Colonne	Type	Longueur

4.2.2 - Le cas des alertes

Pour la gestion des alertes de paiement.
Le « Product owner » à demander à ...



4.2.3 - Les administrateurs

Table « paramètre » des personnes habilitées à modifier les contenus.

4.3 - SQL : LE LDD

Le code complet de création de la BD est dans les annexes.

Quelques exemples significatifs :

Quelques exemples significatifs

Création de la BD :

```
CREATE DATABASE IF NOT EXISTS immorobin
DEFAULT CHARACTER SET utf8
COLLATE utf8_general_ci;
```

Création des tables :

```
CREATE TABLE departement (
  idDepartement int(11) NOT NULL,
  nomDepartement varchar(50) NOT NULL,
  codeDepartement varchar(5) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

```
ALTER TABLE departement
  ADD PRIMARY KEY (idDepartement),
  ADD UNIQUE KEY nomDepartement (nomDepartement),
  ADD UNIQUE KEY codeDepartement (codeDepartement);
```

```
CREATE TABLE ville (
  idVille int(11) NOT NULL,
  nomVille varchar(50) NOT NULL,
  cp char(5) NOT NULL,
  lat double DEFAULT NULL,
  lng double DEFAULT NULL,
  idDepartement int(11) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

```
ALTER TABLE ville
  ADD PRIMARY KEY (idVille),
  ADD KEY nomVille (nomVille),
  ADD KEY idDepartement (idDepartement);
```

```
ALTER TABLE ville
  ADD CONSTRAINT fk_ville_departement FOREIGN KEY (idDepartement)
REFERENCES departement (idDepartement);
```

On voit qu'avec le typage des colonnes, les contraintes de type NOT NULL, les Primary Key (Clés Primaires), les Foreign Key (Clés étrangères), les index uniques, ..., éventuellement les procédures stockées (voir paragraphe suivant) SQL permet

de rendre les données plus sûres à la différence des données de type CSV (Comma Séparateur Value), JSON (JavaScript Object Notation) ou encore, dans une moindre mesure, XML (eXtensible Markup Langage).

4.4 - Les procédures stockées

Les procédures stockées contiennent du code SQL et du code procédural stockés dans la BD sur le serveur de BD.
Elles permettent de séparer le code applicatif du serveur Web (PHP) du code applicatif SQL.
Dans le cadre d'un système où plusieurs « clients » (client lourd de type Desktop, client léger de type Web) utilisent le même code SQL elles permettent de centraliser le code.

Je présente 4 procédures stockées qui réalisent la suppression, l'ajout, la récupération de tous les enregistrements et la récupération d'un enregistrement de la table « departement ».
Pour les autres tables c'est la même chose.
Ces procédures basiques présentées ne contiennent que du code SQL.

```
DELIMITER $$

CREATE PROCEDURE `departement_delete`(paramCODEDEPARTEMENT
CHAR(4) )
BEGIN
     DELETE
     FROM immorobin.departement
     WHERE codeDepartement = paramCODEDEPARTEMENT ;
END$$


CREATE PROCEDURE `departement_insert`(paramIDDEPARTEMENT INT
UNSIGNED,paramCODEDEPARTEMENT CHAR(4) ,paramNOMDEPARTEMENT
VARCHAR(45) )
BEGIN
     INSERT INTO
immorobin.departement(idDepartement,codeDepartement,nomDepartement)
VALUES(paramIDDEPARTEMENT,paramCODEDEPARTEMENT,paramNOMDEPARTEMENT)
;
END$$


CREATE PROCEDURE `departement_select_all`()
BEGIN
     SELECT *
     FROM immorobin.departement ;
END$$


CREATE PROCEDURE `departement_select_one`(paramCODEDEPARTEMENT
CHAR(4) )
BEGIN
     SELECT *
```

```
    FROM immorobin.departement
    WHERE codeDepartement = paramCODEDEPARTEMENT ;
END$$
DELIMITER ;
```

CHAPITRE 5 - CONCEPTION DE L'APPLICATION

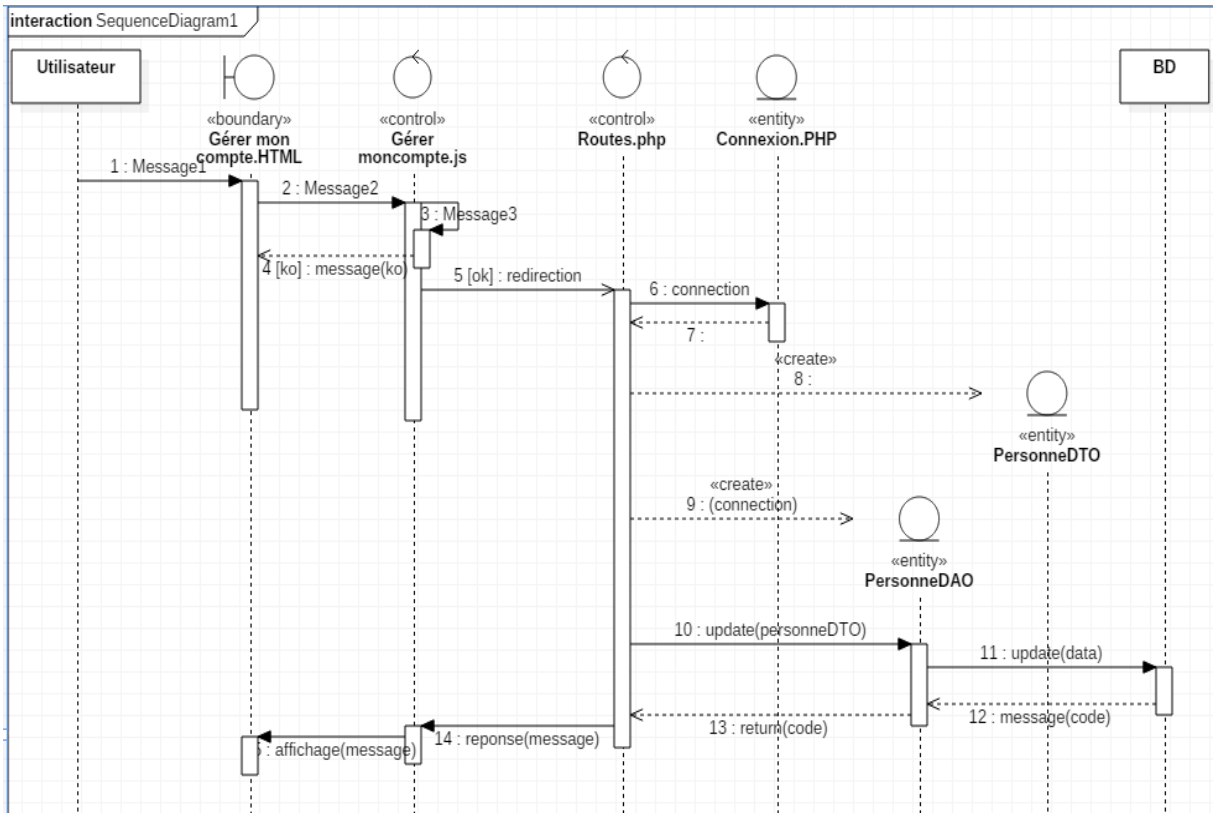
5.1 - Les Diagrammes de Séquence Détaillés

Ils sont « l'explosion » des DSS.

Le Système est divisé en 3 parties : les interfaces (views ou vues), les modèles (models ou modèle : DTO et DAO) et la couche intermédiaire (controllers ou contrôleurs).

Ils permettent de préparer le développement modulaire en couches.

5.1.1 - Diagramme de la séquence « Gérer son compte »



Mots clés :

boundary ou view (IHM), control ou controller (liaison entre les Données et les IHM), entity ou model (Accès aux données de la BD), utilisateur, bd, objet, ligne de vie, activation, message synchrone, message asynchrone, message de retour, self message, message create, garde.

Description :

L utilisateur clique sur le menu gérer son compte.

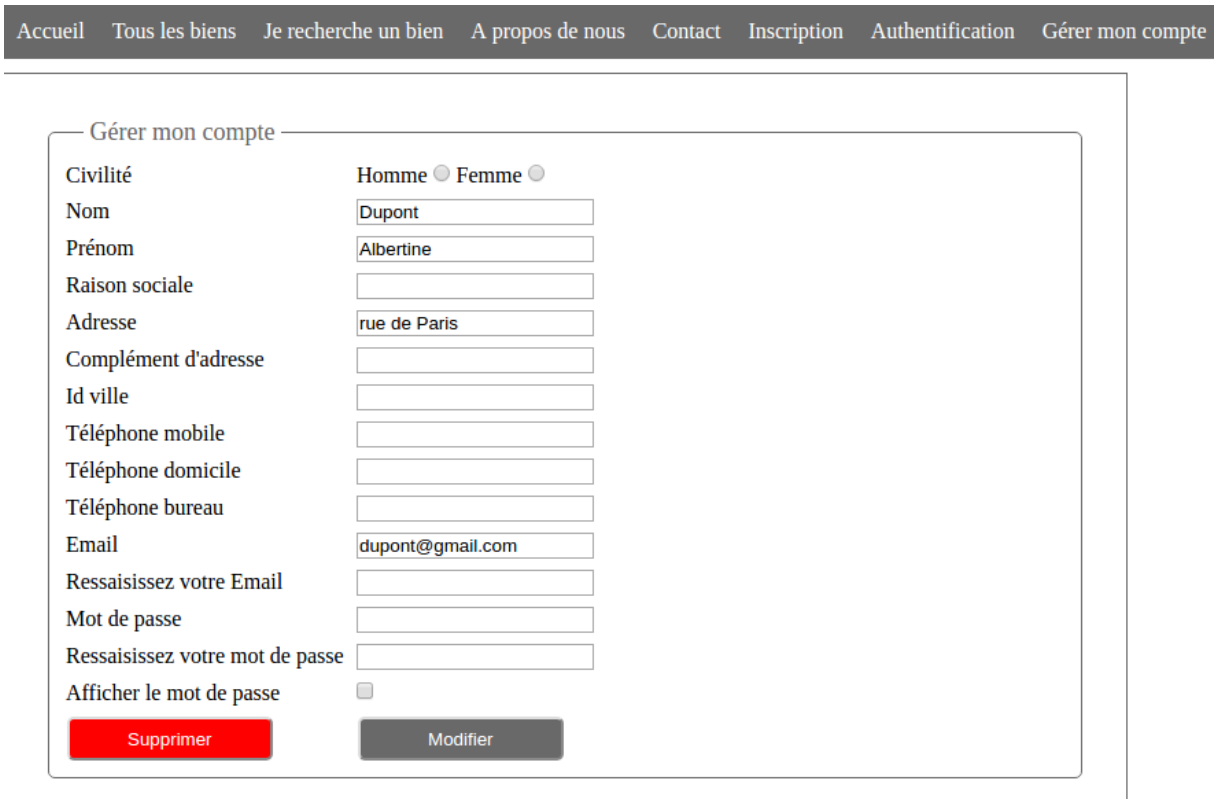
L'écran est créé.
L'utilisateur sélectionne une option (la modification).
L'utilisateur saisie les information nécessaires.
Si les informations sont correctes elles sont transmises au serveur qui teste à nouveau.
Si tout est correct le DAO tente d'insérer les informations dans le BD.
Si tout est correct le serveur renvoie un message de validation.
Dans le cas contraire il renvoie un message d'erreur.
Le contrôleur traite la réponse du serveur et prépare les données et/ou la page de résultat pour l'utilisateur.

CHAPITRE 6 - DÉVELOPPEMENT

6.1 - Technologies utilisées

Éléments	Technologies
Interfaces web statiques	HTML
Mise en forme	CSS,Bootstrap
Contrôles des saisies des interfaces statiques	Javascript,jQuery
Création de la base de données	SQL(LDD - langage de définition de données)
Gestion de la base de données	SQL(LMD – langage de manipulation de données)
Script serveur	PHP

6.1.1 - Copies d'écrans



6.1.2 - Codes statiques des écrans

HTML

```
<!DOCTYPE html>
<!--
-->
<html>
    <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width">


<!-- lien vers le CSS -->

        <link rel="stylesheet" type="text/css"
href="../css/main.css">
        <title>Public - GererMonCompte.html</title>
    </head>

    <body>
        <header id="header">
        </header>

        <nav id="nav">
            <ul>
                <li></li>
            </ul>
        </nav>

        <article id="article">
            <div id="container">

                <fieldset id="cadre">

                    <legend> Gérer mon compte </legend>

                    <table id='tableau'>

                        <tr>
                            <td>
                                <label
class='etiquette'>Civilité</label>
                            </td>
                            <td>
                                <label for='rbHomme'
class='etiquette'>Homme</label>
```

```
                                <input type="radio"
name="rbCivilite" id="rbHomme" value="1" />
                                <label for='rbFemme'
class='etiquette'>Femme</label>
                                <input type="radio"
name="rbCivilite" id="rbFemme" value="2" />
                            </td>
                        </tr>

                        <tr>
                            <td>
                                <label for='nom'
class='etiquette'>Nom</label>
                            </td>
                            <td>
                                <input type='text' value=''
name='nom' id='nom' />
                            </td>
                        </tr>
                        <tr>
                            <td>
                                <label for='prenom'
class='etiquette'>Prénom</label>
                            </td>
                            <td>
                                <input type='text' value=''
name='prenom' id='prenom' />
                            </td>
                        </tr>
                        <tr>
                            <td>
                                <label for='raisonSociale'
class='etiquette'>Raison sociale</label>
                            </td>
                            <td>
                                <input type='text' value=''
name='raisonSociale' id='raisonSociale' />
                            </td>
                        </tr>
                        <tr>
                            <td>
                                <label for='adresse'
class='etiquette'>Adresse</label>
                            </td>
                            <td>
                                <input type='text' value=''
name='adresse' id='adresse' />
                            </td>
                        </tr>
                        <tr>
                            <td>
                                <label for='complementAdresse'
class='etiquette'>Complément d'adresse</label>
```

```
                            </td>
                            <td>
                                <input type='text' value=''
name='complementAdresse' id='complementAdresse' />
                            </td>
                        </tr>
                        <tr>
                            <td>
                                <label for='idVille'
class='etiquette'>Id ville</label>
                            </td>
                            <td>
                                <input type='text' value=''
name='idVille' id='idVille' />
                            </td>
                        </tr>
                        <tr>
                            <td>
                                <label for='telMobile'
class='etiquette'>Téléphone mobile</label>
                            </td>
                            <td>
                                <input type='text' value=''
name='telMobile' id='telMobile' />
                            </td>
                        </tr>
                        <tr>
                            <td>
                                <label for='telDomicile'
class='etiquette'>Téléphone domicile</label>
                            </td>
                            <td>
                                <input type='text' value=''
name='telDomicile' id='telDomicile' />
                            </td>
                        </tr>
                        <tr>
                            <td>
                                <label for='telBureau'
class='etiquette'>Téléphone bureau</label>
                            </td>
                            <td>
                                <input type='text' value=''
name='telBureau' id='telBureau' />
                            </td>
                        </tr>
                        <tr>
                            <td>
                                <label for='email1'
class='etiquette'>Email</label>
                            </td>
                            <td>
```

```
                                <input type='text' value=''
name='email1' id='email1' />
                            </td>
                        </tr>
                        <tr>
                            <td>
                                <label for='email2'
class='etiquette'>Ressaisissez votre Email</label>
                            </td>
                            <td>
                                <input type='text' value=''
name='email2' id='email2' />
                            </td>
                        </tr>
                        <tr>
                            <td>
                                <label for='mdp1'
class='etiquette'>Mot de passe</label>
                            </td>
                            <td>
                                <input type='password' value=''
name='mdp1' id='mdp1' />
                            </td>
                        </tr>
                        <tr>
                            <td>
                                <label for='mdp2'
class='etiquette'>Ressaisissez votre mot de passe</label>
                            </td>
                            <td>
                                <input type='password' value=''
name='mdp2' id='mdp2' />
                            </td>
                        </tr>
                        <tr>
                            <td>
                                <label for="chkAfficherMasquerMDPs"
id="lblAfficherMDP">Afficher le mot de passe</label>
                            </td>
                            <td>
                                <input type="checkbox"
id="chkAfficherMasquerMDPs" />
                            </td>
                        </tr>
                        <tr>
                            <td>
                                <button type='button'
id='btSupprimer'>Supprimer</button>
                            </td>
                            <td>
                                <button type='button'
id='btModifier'>Modifier</button>
                            </td>
```

```
                        </tr>
                    </table>

                </fieldset>
                <p>
                </p>
                <p>
                    <label id="lblMessage"></label>
                </p>
            </div>
        </article>

        <footer id="footer">
        </footer>

        <script src="../js/lib/jquery/jquery.js"></script>

        <script src="../js/partials/head.js"></script>
        <script src="../js/partials/header.js"></script>
        <script src="../js/partials/nav.js"></script>
        <script src="../js/partials/footer.js"></script>
        <script src="../js/Controles.js"></script>
        <script src="../js/GererMonCompte.js"></script>
    </body>
</html>
```

JavaScript

```
/*
 * header.js
 */
var lsContenu = "<h1>IMMO ROBIN - Le Site</h1>";

$("#header").html(lsContenu);
```

CSS

6.1.3 - Codes dynamiques des écrans

Voici le script de contrôle des saisies en JavaScript :

Ce script contrôle les saisies de l'utilisateur en utilisant des expressions régulières sur l'écran de la gestion du compte.

Contrôles JavaScript

```
/*
 * GererMonCompte.js
 */

/**
 *
 * @returns {undefined}
 */
function init() {

    requeteAJAXSelectOneByEmail();

    $("#btSupprimer").on("click", supprimer);
    $("#btModifier").on("click", modifier);
    $("#chkAfficherMasquerMDPs").on("click", afficherMasquerMDPs);

} /// init


/**
 *
 * @returns {undefined}
 */
function afficherMasquerMDPs() {
    var coche = $("#chkAfficherMasquerMDPs").prop("checked");

    if (coche) {
        $("#mdp1").attr("type", "text");
        $("#mdp2").attr("type", "text");
    }
    else {
        $("#mdp1").attr("type", "password");
        $("#mdp2").attr("type", "password");
    }
}

/**
 *
 * @returns {undefined}
 */
function supprimer() {
```

```
    //var lbOK = validerSaisies();
    if ($("#email").val() !== "") {
        requeteAJAXSupprimer();
    } else {
        $("#lblMessage").css("color", "red");
    }
}

/**
 *
 */
function modifier() {
    var lbOK = validerSaisies();
    if (lbOK) {
//
        requeteAJAXModifier();
    } else {
        $("#lblMessage").css("color", "red");
    }
}


/**
 *
 * @returns {undefined}
 */
function validerSaisies() {

    $("#lblMessage").html("");
    $("#lblMessage").css("color", "black");

    var lsMessage = "";
    var lbOK = true;
    var lbControle;

    if ($("#rbHomme").prop("checked") === false && $
("#rbFemme").prop("checked") === false) {
        lbOK = false;
        lsMessage += "Civilité incorrecte<br>";
    }


    lbControle = isNomAvecAccent($("#prenom").val());
    if (!lbControle) {
        lbOK = false;
        lsMessage += "prenom incorrect<br>";

    }

    lbControle = isNomAvecAccent($("#nom").val());
    if (!lbControle) {
        lbOK = false;
        lsMessage += "nom incorrect<br>";
```

```
        console.log("nom incorrect");
    }

    lbControle = isAdresse($("#adresse").val());
    if (!lbControle) {
        lbOK = false;
        lsMessage += "Adresse incorrecte<br>";
    }

    if ($("#telDomicile").val() !== "") {
        lbControle = isTelephoneFR($("#telDomicile").val());
        if (!lbControle) {
            lbOK = false;
            lsMessage += "Téléphone domicile incorrect<br>";
        }
    }
    if ($("#telMobile").val() !== "") {
        lbControle = isTelephoneFR($("#telMobile").val());
        if (!lbControle) {
            lbOK = false;
            lsMessage += "Téléphone mobile incorrect<br>";
        }
    }
    if ($("#telBureau").val() !== "") {
        lbControle = isTelephoneFR($("#telBureau").val());
        if (!lbControle) {
            lbOK = false;
            lsMessage += "Téléphone bureau incorrect<br>";
        }
    }

    lbControle = isEmail($("#email1").val());
    if (!lbControle) {
        lbOK = false;
        lsMessage += "E-mail incorrect<br>";
    }

    lbControle = isEmail($("#email2").val());
    if (!lbControle) {
        lbOK = false;
        lsMessage += "E-mail 2 incorrect<br>";
    }

    lbControle = isMDPFort($("#mdp1").val());
    if (!lbControle) {
        lbOK = false;
        lsMessage += "Mot de passe 1 incorrect<br>";
        lsMessage += "Le mot de passe doit avoir au moins une
majuscule, une minuscule, un chiffre et comporté au moins 6
carcactères<br>";
    }

    lbControle = isMDPFort($("#mdp2").val());
```

```
    if (!lbControle) {
        lbOK = false;
        lsMessage += "Mot de passe 2 incorrect<br>";
        lsMessage += "Le mot de passe doit avoir au moins une
majuscule, une minuscule, un chiffre et comporté au moins 6
caractères<br>";
    }

    lbControle = $("#email1").val() === $("#email2").val();
    if (!lbControle) {
        lbOK = false;
        lsMessage += "Les 2 e-mails ne sont pas identiques<br>";
    }

    lbControle = $("#mdp1").val() === $("#mdp2").val();
    if (!lbControle) {
        lbOK = false;
        lsMessage += "Les 2 mots de passe ne sont pas
identiques<br>";
    }

    /*
     * Affichage :
     */
    $("#lblMessage").html(lsMessage);
    return lbOK;
} /// validerSaisies
```

Bibliothèque

```
/*
 * Controles.js
 */
/**
 *
 * @param {type} psChaine
 * @returns {Boolean}
 */
function isEmail(psChaine) {
//
    var motif = "^[0-9a-zA-Z_-]+([.][0-9a-zA-Z_-]+)?@[0-9a-zA-Z._-]
{2,}[.][a-zA-Z]{2,5}$";
    var er = new RegExp(motif);
    return er.test(psChaine);
}

/**
 *
 * @param {type} psChaine
 * @returns {Boolean}
 */
function isMDPFort(psChaine) {
//
    // De 6 a 10 caractères, au moins 1 chiffre, au moins 1 majus,
au moins une minus
    var motif = "^(?=.*\\d)(?=.*[a-z])(?=.*[A-Z]).{6,10}$";
    var er = new RegExp(motif);
    return er.test(psChaine);
}

/**
 *
 * @param {type} psChaine
 * @returns {Boolean}
 */
function isCPFR(psChaine) {
    var motif = "^\\d{5}$";
    var er = new RegExp(motif);
    return er.test(psChaine);
}

/**
 *
 * @param {type} psChaine
 * @returns {Boolean}
 */
function isTelephoneFR(psChaine) {
    // Indicatif (1 : USA, 961 : Liban)
    // Telephone : 0000000000 ou 00-00-00-00-00
    // ou encore (+nnnn)000000000
```

```
    var motif = "^0[1-9]\\d{8}|0[1-9]([-]\\d{2}){4}|\\(\\+[0-9]
{1,4}\\)[1-9]\\d{8}$";
    var er = new RegExp(motif);
    return er.test(psChaine);
}


/**
 *
 * @param {type} psChaine
 * @returns {Boolean}
 */
function isNomSansAccent(psChaine) {
    var motif = "^[A-Z][A-Za-z '-]+$";
    var er = new RegExp(motif);
    return er.test(psChaine);
}

/**
 *
 * @param {type} psChaine
 * @returns {Boolean}
 */
function isNomAvecAccent(psChaine) {
    var motif = "^[A-Za-zàâäéèêëîïôöüûù '-]+$";
    var er = new RegExp(motif);
    return er.test(psChaine);
}

/**
 *
 * @param {type} psChaine
 * @returns {Boolean}
 */
function isAdresse(psChaine) {
    var motif = "^[0-9A-Za-zàâäéèêëîïôöüûù ,'-]+$";
    var er = new RegExp(motif);
    return er.test(psChaine);
}

/**
 *
 * @param {type} psChaine
 * @returns {Boolean}
 */
function isVille(psChaine) {
    var motif = "^[0-9A-Za-zàâäéèêëîïôöüûù '-]+$";
    var er = new RegExp(motif);
    return er.test(psChaine);
}

/**
 *
```

```
 * @param {type} psChaine
 * @returns {Boolean}
 */
function isNomUnderscorise(psChaine) {
    // ?, *, +
    // ? : 0,1
    // * : 0,n
    // + : 1,n
    // Accepte les chiffres
    // mots valides : ville, id_pays, nom_du_pays,
code007_du_produit
    var motif = "^[a-z0-9]+(_?[a-z0-9]+)*$";
    var er = new RegExp(motif);
    return er.test(psChaine);
}

/**
 *
 * @param {type} psChaine
 * @returns {Boolean}
 */
function isConstante(psChaine) {
    // ?, *, +
    // ? : 0,1
    // * : 0,n
    // + : 1,n
    // MMM et n fois _MMM
    // Accepte les chiffres
    // Mots valides : NOM_DU_SERVEUR,
    // Mots invalides : Nom_DU_SERVEUR,
    var motif = "^[A-Z]+(_[A-Z]*)*$";
    var er = new RegExp(motif);
    return er.test(psChaine);
}

/**
 *
 * @param {type} psChaine
 * @returns {Boolean}
 */
function isCamelize(psChaine) {
    // ?, *, +
    // ? : 0,1
    // * : 0,n
    // + : 1,n
    // N'accepte pas les chiffres
    // Mots valides : ville, idPays, nomDuPays, ControlesEr,
ControlesExpressionsRationnelles
    // Mots invalides : Controles_Expressions_Rationnelles,
ControlesExpressionsRationnelles007, URL_DU_SERVEUR
    var motif = "^[A-Za-z][a-z]+([A-Z][a-z]*)*$";
//    var motif = "^[A-Za-z]";
    var er = new RegExp(motif);
```

```
    return er.test(psChaine);
}

/**
 *
 * @param {type} psChaine
 * @returns {Boolean}
 */
function isDateFR(psChaine) {
    // Date jj/mm/aaaa ou j/m/aaaa ou j/mm/aaaa ou jj/m/aaaa
    //var motif = "^(\\d{2}/\\d{2}/\\d{4})|(\\d/\\d/\\d{4})|
(\\d/\\d{2}/\\d{4})|(\\d{2}/\\d/\\d{4})$";
    var motif = "^\\d{1,2}/\\d{1,2}/\\d{4}$";
    var er = new RegExp(motif);
    return er.test(psChaine);
}

/**
 *
 * @param {type} psChaine
 * @returns {Boolean}
 */
function isURL(psChaine) {
    // + : 1,n
    // http://... ou https://... ou file://... ou file:///...
    var motif = "^(http://|https://|file://|file:///).+$";
    var er = new RegExp(motif);
    return er.test(psChaine);
}
```

6.2 - Continuez avec la partie Entities/Models et DAO

Fichiers properties/Connexion

Testez ! Testez ! Testez !

Beans entities/Entities PHP/…

Testez ! Testez ! Testez !

Interface DAO/DAOs/Factory de DAOs/...

Testez ! Testez ! Testez !

Du générique ? C'est anti-SUN/Oracle. Mais pas interdit !

```
-- immorobin.properties
[section_connexion]
protocole=mysql
-- serveur=10.57.255.168
serveur=127.0.0.1
port=3306
bd=ImmoRobin
ut=root
mdp=
```

```php
<?php

/*
 *
 */

class PersonneDAO {

    /**
     *
     * @param PDO $pcnx
     * @param type $data
     * @return string
     */
    public static function selectOneByEmailMDP(PDO $pcnx, $data) {
        /*
         * Pour l'authentification
         * Pas très bon le resultat, il faudrait un POJO ou son
equivalent
         * cf selectOneByEmail()
         */
        $t = array();
        try {
// Le SELECT
            $lsSelect = "SELECT * FROM Personne WHERE email=? AND
mdp=?";

            $lrs = $pcnx->prepare($lsSelect);
            $lrs->execute(array($data["email"], $data["mdp"]));
            $enr = $lrs->fetch();
            if ($enr) {
                $t["message"] = $enr["email"];
            } else {
                $t["message"] = "-1";
            }
        } catch (Exception $e) {
            $lrs = null;
//echo "Erreur : " . $e->getMessage() . "<br>";
            //$t["message"] = "Erreur : " . $e->getMessage();
            $t["message"] = "-2";
        }
        return $t;
    }

    /**
     *
     * @param PDO $pcnx
     * @param type $data
     * @return string
     */
    public static function selectOneByEmail(PDO $pcnx, $data) {
        $t = array();
```

```
        try {
// Le SELECT
            $lsSelect = "SELECT * FROM Personne WHERE email=?";
//echo "<br>", $data["email"], "<br>";
            $lrs = $pcnx->prepare($lsSelect);
            $lrs->execute(array($data["email"]));
            $lrs->setFetchMode(PDO::FETCH_ASSOC);
            $enr = $lrs->fetch();
            if ($enr) {
                $t["message"] = $enr;
            } else {
                $t["message"] = "-1";
            }
        } catch (Exception $e) {
            $lrs = null;
//echo "Erreur : " . $e->getMessage() . "<br>";
            $t["message"] = "-2";
        }
        return $t;
    }

    /**
     *
     * @param PDO $pcnx
     * @param type $data
     * @return string
     */
    public static function insert(PDO $pcnx, $data) {
        $t = array();
        try {
// Le SQL
            $civilite = $data["civilite"];
            $nom = $data["nom"];
            $prenom = $data["prenom"];
            $raisonSociale = $data["raisonSociale"];
            $adresse = $data["adresse"];
            $complementAdresse = $data["complementAdresse"];
            $idVille = $data["idVille"];
            $telMobile = $data["telMobile"];
            $telDomicile = $data["telDomicile"];
            $telBureau = $data["telBureau"];
            $email = $data["email"];
            $mdp = $data["mdp"];
            $idTypePersonne = $data["idTypePersonne"];

            $lsSQL = 'INSERT INTO
Personne(civilite,nom,prenom,raisonSociale,adresse,complementAdress
e,idVille,telMobile,telDomicile,telBureau,email,mdp,idTypePersonne)
VALUES(?,?,?,?,?,?,?,?,?,?,?,?,?)';

            $pcnx->beginTransaction();
            $lstmt = $pcnx->prepare($lsSQL);
```

```
            $lstmt->execute(array($civilite, $nom, $prenom,
$raisonSociale, $adresse, $complementAdresse, $idVille, $telMobile,
$telDomicile, $telBureau, $email, $mdp, $idTypePersonne));

            $pcnx->commit();
            //echo $lstmt->rowCount();
            if ($lstmt->rowCount() == 1) {
                $t["message"] = "OK";
            } else {
                $t["message"] = "???";
            }
        } catch (Exception $e) {
            $pcnx->rollBack();
            $t["message"] = "Erreur : " . $e->getMessage();
        }
        return $t;
    }

    /**
     *
     * @param PDO $pcnx
     * @param type $data
     */
    public static function delete(PDO $pcnx, $data) {
        $t = array();
        try {
// Le SQL
            $email = $data["email"];
//            $mdp = $data["mdp"];

            $lsSQL = 'DELETE FROM Personne WHERE email = ?';
            $pcnx->beginTransaction();
            $lstmt = $pcnx->prepare($lsSQL);
            $lstmt->execute(array($email));
            $pcnx->commit();
            //echo $lstmt->rowCount();

            if ($lstmt->rowCount() == 1) {
                $t["message"] = "OK";
            } else {
                $t["message"] = "Introuvable";
            }
        } catch (Exception $e) {
            $pcnx->rollBack();
            $t["message"] = "Erreur : " . $e->getMessage();
        }
        return $t;
    }

    /**
     *
     * @param PDO $pcnx
     * @param type $data
```

```
     */
    public static function update(PDO $pcnx, $data) {
        $t = array();
        try {
// Le SQL
            $civilite = $data["civilite"];
            $nom = $data["nom"];
            $prenom = $data["prenom"];
            $raisonSociale = $data["raisonSociale"];
            $adresse = $data["adresse"];
            $complementAdresse = $data["complementAdresse"];
            $idVille = $data["idVille"];
            $telMobile = $data["telMobile"];
            $telDomicile = $data["telDomicile"];
            $telBureau = $data["telBureau"];
            $email = $data["email"];
            $mdp = $data["mdp"];

            $lsSQL = 'UPDATE Personne
SET
civilite=?,nom=?,prenom=?,raisonSociale=?,adresse=?,complementAdres
se=?,idVille=?,telMobile=?,telDomicile=?,telBureau=?,email=?,mdp=?
WHERE email = ?';

            $pcnx->beginTransaction();
            $lstmt = $pcnx->prepare($lsSQL);

            $lstmt->execute(array($civilite, $nom, $prenom,
$raisonSociale, $adresse, $complementAdresse, $idVille, $telMobile,
$telDomicile, $telBureau, $email, $mdp, $email));

            $pcnx->commit();
            //echo $lstmt->rowCount();

            $t["message"] = "OK";
        } catch (Exception $e) {
            $pcnx->rollBack();
            $t["message"] = "Erreur : " . $e->getMessage();
        }
        return $t;
    }

}
```

6.3 - Terminez avec la « glue » : les controls

AJAX.

Servlets/Codes PHP/...

Testez ! Testez ! Testez !

Requête AJAX

```
/**
 *
 * @returns {undefined}
 */
function requeteAJAXSelectOneByEmail() {
    var email = "";

    var jqXHR = $.get(
            "../routes/routes.php",
            {email: email,
                action: "personneSelectOneByEmail"},
    "json"
            );

    jqXHR.done(function(data) {
        var objetJSON = JSON.parse(data);

        $("#nom").val(objetJSON.message.nom);
        $("#prenom").val(objetJSON.message.prenom);
        $("#adresse").val(objetJSON.message.adresse);
        $("#email1").val(objetJSON.message.email);
    });

    jqXHR.fail(function(xhr, statut, erreur) {
        var lsTexte = xhr.status + ":" + xhr.statusText;
        $("#lblMessage").html(lsTexte);
    });
} /// requeteAJAXSelectOne


/**
 *
 * @returns {undefined}
 */
function requeteAJAXModifier() {
    var civilite = "";
    if ($('#rbHomme').prop("checked") == true) {
        civilite = "Monsieur";
    }
    if ($('#rbFemme').prop("checked") == true) {
        civilite = "Madame";
    }
    var nom = $('#nom').val();
    var prenom = $('#prenom').val();
    var raisonSociale = $('#raisonSociale').val();
    var adresse = $('#adresse').val();
    var complementAdresse = $('#complementAdresse').val();
    var idVille = $('#idVille').val();
    var telMobile = $('#telMobile').val();
    var telDomicile = $('#telDomicile').val();
```

```
    var telBureau = $('#telBureau').val();
    var email = $('#email1').val();
    var mdp = $('#mdp1').val();

    var jqXHR = $.post(
            "../routes/routes.php",
            {civilite: civilite,
                nom: nom,
                prenom: prenom,
                raisonSociale: raisonSociale,
                adresse: adresse,
                complementAdresse: complementAdresse,
                idVille: idVille,
                telMobile: telMobile,
                telDomicile: telDomicile,
                telBureau: telBureau,
                email: email,
                mdp: mdp,
                action: "personneModifier"},
    "json"
            );

    jqXHR.done(function(data) {
        var objetJSON = JSON.parse(data);
        $("#lblMessage").html(objetJSON.message);
    });

    jqXHR.fail(function(xhr, statut, erreur) {
        var lsTexte = xhr.status + ":" + xhr.statusText;
        $("#lblMessage").html(lsTexte);
    });


    /**
     * Affichage
     */
//$("#lblMessage").html(lsMessage);
} /// requeteAJAXModification
```

Routes.php

```
session_start();
ob_start();
require_once '../php/lib/simplejson/simplejson.php';
require_once '../daos/Connexion.php';

$lcnx =
Connexion::getConnexionViaProperties("../conf/immorobin.properties"
);

$tMessage = array();
$tMessage["message"] = "";
$stringJSON = toJSON($tMessage);
```

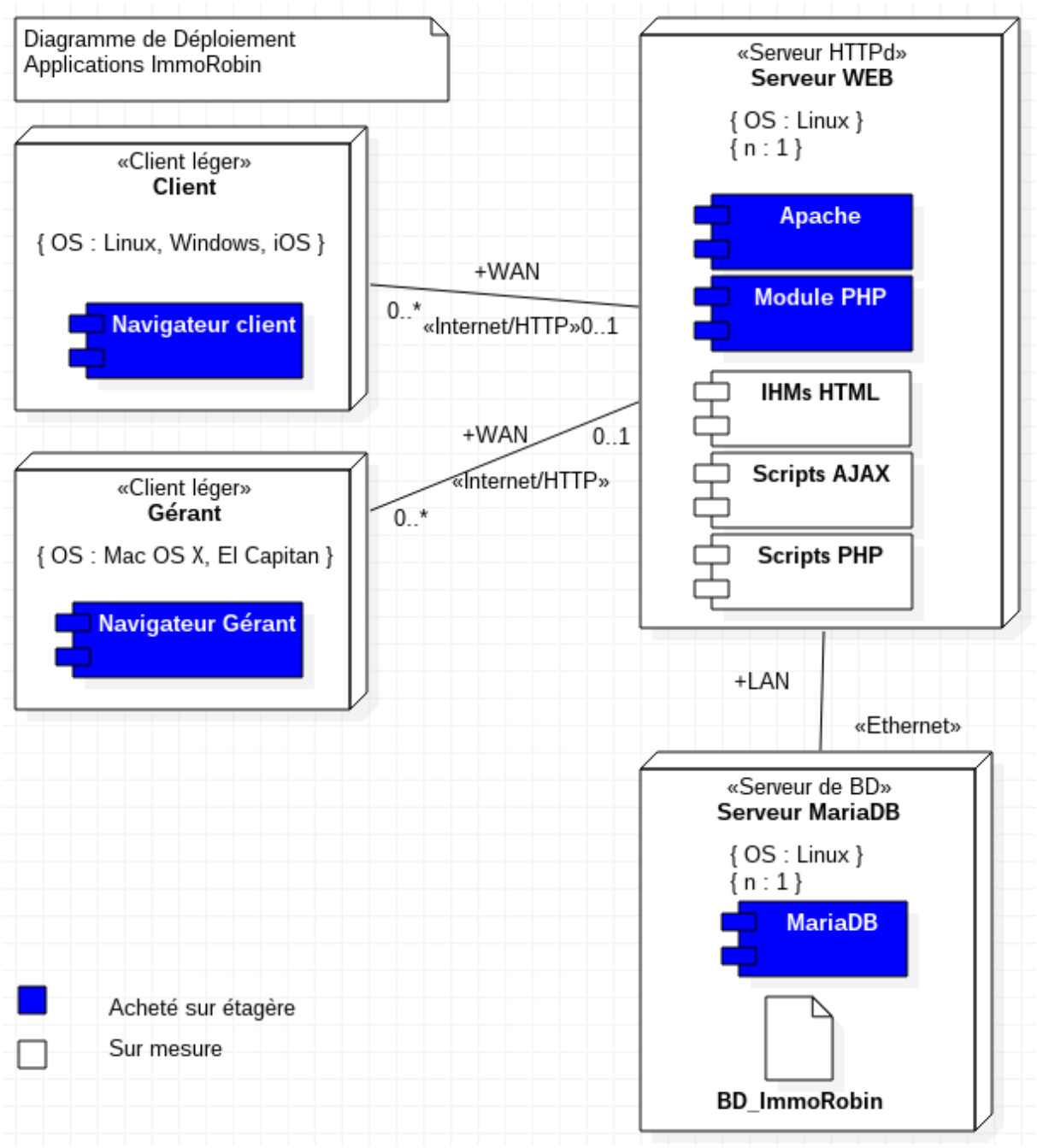
```
    /*
     * ca vient du NAV
     */
    if ($_GET["action"] == "gererMonCompte") {
        $lsCible = "";
        if (isSet($_SESSION["email"])) {
            $lsCible = "GererMonCompte.html";
        } else {
            $lsCible = "Authentification.html";
        }

        $tMessage["message"] = "";
        $stringJSON = toJSON($tMessage);

        /*
         * Redirection
         */
        header("location: ../boundaries/$lsCible");
    }

    /*
     * ca vient du init() GererMonCompte.js
     */
    if ($_GET["action"] == "personneSelectOneByEmail") {
        require_once '../daos/PersonneDAO.php';
        $data = array();
        $data["email"] = $_SESSION["email"];
        //$data["email"] = $_GET["email"];
        $tMessage = PersonneDAO::selectOneByEmail($lcnx, $data);
        $stringJSON = toJSON($tMessage);
    }
```

```
    /*
     * personneModifier
     */
    if ($_POST["action"] == "personneModifier") {
        require_once '../daos/PersonneDAO.php';
        $retour = PersonneDAO::update($lcnx, $_POST);
        $stringJSON = toJSON($retour);
    }
```

CHAPITRE 7 - DÉPLOIEMENT

7.1 - Le Diagramme de Déploiement

Ce diagramme doit être utile à l'administrateur System et à l'administrateur BD.



Mots clés : les nœuds, les composants, les artéfacts +, les dépendances, les chemins de communication (Communication path), ...

7.2 - Le déploiement

Mots clés : les nœuds, les composants, les artéfacts +, les dépendances, les chemins de communication (Communication path), ...

Ce diagramme montre 4 nœuds (4 types de machines) une machine cliente pour le gérant un machine cliente pour le client (locataire,internaute). Sur ces machines sont installés des logiciels achetés chez les éditeurs (un navigateur ,un serveur web apache, un serveur de BD MariaDB,un module de script serveur PHP) et aussi les logiciels que j'ai codés (les pages web, les scripts JavaScript, les scripts PHP) et enfin la base de données (artefacts).
Les logiciels sont appelés composants logiciels en UML.
La différence entre composants et artefacts correspond à la différence entre éléments statiques et éléments dynamiques;ces derniers permettent de modifier les premiers.
Pour communiquer entre machines on besoin de réseau (réseau local LAN,réseau global WAN)

CHAPITRE 8 - CONCLUSION

Débutant en développement informatique au début de cette formation,

j'ai pu enrichir considérablement

mes connaissances dans la conception ainsi que le développement

le stage en entreprise m'a définitivement conforté dans ce choix de reconversion

professionelle. J'ai appris énormément dans ce contexte professionnel .

J'ai pu me rendre compte des difficultés et de la minutie de la conception d'un site web

J'ai grandement apprécié le principe qui m'impose de toujours me remettre en

question, afin de trouver et proposer la meilleure solution pour résoudre un problème, grâce

à une réflexion et une logique de conception.

La découverte de nouveaux languages ou nouveau outils animent la richesse de ce

metier et permettent d'avoir une permanente remise en question en enrichissant ses

competences en restant en état de veille sur des technologies en constante évolution.

J'ai largement profité des phases intenses de réflexion, dont, paradoxalement, je

trouve qu'elles me détendent.

Encore merci à toutes les personnes que j'ai côtoyé tout au long de ma formation et

de mon stage en entreprise.

CHAPITRE 9 - ANNEXES

9.1 - Outils utilisés

Outil	Objectif	Commentaire
phpMyAdmin	Gestion de la BD	Simple, rapide, parfois lourd
ToadForMySQL	Gestion de la BD	Lourd mais puissant pour son QBE
NetBeans	IDE	Idéale pour HTML, JavaScript et PHP.
StarUML	Modéliser en UML	Gratuit, design propre. Ne génère pas de code directement exploitable en PHP ou Java. Et pas de SQL dans tous les cas. La version 2 est disponible sur Linux, Mac et Windows.
PowerAMC	Modéliser en UML ou en Merise	Payant mais version d'essai assez puissante. Permet de générer du SQL, du Java, du PHP.
PencilEvolus	Outil de maquettage	Windows, Mac, Linux
GANTT Project	Diagramme de GANTT	Windows, Mac, Linux

9.2 - Bibliographie et Webographie

9.2.1 - Bibliographie

Christian Soutou, « Programmer avec MySQL », Eyrolles, 2015

Pascal Roques, « UML 2, Modéliser une application web », Eyrolles, 2008.

Christian Soutou avec la contribution de Frédéric Brouard, « UML 2 pour les bases de données », Eyrolles, 2012.

Rich Cannings, Himanshu Dwivedi, Zane Lackey , « Hacking sur le Web 2.0: Vulnérabilité du Web 2.0 et sécurisation » , Pearson, 2008.

Eric Daspet, Cyril Pierre De Geyer, « PHP 5 avancé », Eyrolles, 2012

Maxime Gréau, « Apache Maven », Eyrolles, 2011

9.2.2 - Webographie

9.2.2.1 - git

https://git-scm.com/

9.2.2.2 - maven

http://maven-guide-fr.erwan-alliaume.com/maven-guide-fr/site/reference/introduction.html

9.2.2.3 - HTML

http://www.w3.org/

9.2.2.4 - Java

http://docs.oracle.com/javase/7/docs/

http://www.oracle.com/technetwork/java/javaee/documentation/index.html

9.2.2.5 - PHP

http://www.php.net/

9.2.2.6 - SQL

http://sqlpro.developpez.com/

9.2.2.7 - MySQL

http://www.mysql.fr/

9.2.2.8 - JavaScript

https://developer.mozilla.org/fr/docs/Web/JavaScript

9.2.2.9 - jQuery

https://jquery.com/

https://jqueryui.com/

9.3 - Glossaire/Lexique ou/et Liste de mots-clés et sigles

Mot clé	Description
CRUD	Concerne le LMD (Langage de manipulation de données). C : create R : read U : update D : delete
HTTP	Protocole de communication entre 2 logiciels (un client et un serveur)

9.4 - Autres codes

9.4.1.1 - Code complet de création de la BD

```
SET SQL_MODE = "NO_AUTO_VALUE_ON_ZERO";
SET time_zone = "+00:00";

/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS
*/;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40101 SET NAMES utf8mb4 */;

CREATE DATABASE IF NOT EXISTS `immorobin` DEFAULT CHARACTER SET utf8
COLLATE utf8_general_ci;
USE `immorobin`;

DELIMITER $$
DROP PROCEDURE IF EXISTS `departement_delete`$$
CREATE DEFINER=`root`@`localhost` PROCEDURE `departement_delete`
(`paramCODEDEPARTEMENT` CHAR(4))  BEGIN
        DELETE
        FROM immorobin.departement
        WHERE codeDepartement = paramCODEDEPARTEMENT ;
END$$

DROP PROCEDURE IF EXISTS `departement_insert`$$
CREATE DEFINER=`root`@`localhost` PROCEDURE `departement_insert`
(`paramIDDEPARTEMENT` INT UNSIGNED, `paramCODEDEPARTEMENT` CHAR(4),
`paramNOMDEPARTEMENT` VARCHAR(45))  BEGIN
        INSERT INTO
immorobin.departement(idDepartement,codeDepartement,nomDepartement)

VALUES(paramIDDEPARTEMENT,paramCODEDEPARTEMENT,paramNOMDEPARTEM
ENT) ;
END$$

DROP PROCEDURE IF EXISTS `departement_select_all`$$
CREATE DEFINER=`root`@`localhost` PROCEDURE `departement_select_all` ()
BEGIN
        SELECT *
        FROM immorobin.departement ;
END$$

DROP PROCEDURE IF EXISTS `departement_select_one`$$
CREATE DEFINER=`root`@`localhost` PROCEDURE `departement_select_one`
(`paramCODEDEPARTEMENT` CHAR(4))  BEGIN
        SELECT *
        FROM immorobin.departement
```

```
	WHERE codeDepartement = paramCODEDEPARTEMENT ;
END$$

DROP PROCEDURE IF EXISTS `departement_update`$$
CREATE DEFINER=`root`@`localhost` PROCEDURE `departement_update`
(`paramIDDEPARTEMENT` INT UNSIGNED, `paramCODEDEPARTEMENT` CHAR(4),
`paramNOMDEPARTEMENT` VARCHAR(45))  BEGIN
	UPDATE immorobin.departement
	SET idDepartement = paramIDDEPARTEMENT,nomDepartement =
paramNOMDEPARTEMENT
	WHERE codeDepartement = paramCODEDEPARTEMENT ;
END$$

DROP PROCEDURE IF EXISTS `proprietaire_delete`$$
CREATE DEFINER=`root`@`localhost` PROCEDURE `proprietaire_delete`
(`paramIDPRORIETAIRE` INT UNSIGNED)  BEGIN
	DELETE
	FROM immorobin.proprietaire
	WHERE idProrietaire = paramIDPRORIETAIRE ;
END$$

DROP PROCEDURE IF EXISTS `proprietaire_insert`$$
CREATE DEFINER=`root`@`localhost` PROCEDURE `proprietaire_insert`
(`paramIDPRORIETAIRE` INT UNSIGNED, `paramNOMPROPRIETAIRE`
VARCHAR(45), `paramPRENOMPROPRIETAIRE` VARCHAR(45),
`paramRAISONSOCIALE` VARCHAR(45), `paramTELDOMICILE` VARCHAR(45),
`paramTELMOBILE` VARCHAR(45), `paramTELBUREAU` VARCHAR(45),
`paramADRESSE` VARCHAR(45), `paramEMAILPROPRIETAIRE` VARCHAR(45),
`paramIDVILLE` INT UNSIGNED)  BEGIN
	INSERT INTO
immorobin.proprietaire(idProrietaire,nomProprietaire,prenomProprietaire,raisonSo
ciale,telDomicile,telMobile,telBureau,adresse,emailProprietaire,idVille)

VALUES(paramIDPRORIETAIRE,paramNOMPROPRIETAIRE,paramPRENOMPROPRIET
AIRE,paramRAISONSOCIALE,paramTELDOMICILE,paramTELMOBILE,paramTELBUR
EAU,paramADRESSE,paramEMAILPROPRIETAIRE,paramIDVILLE) ;
END$$

DROP PROCEDURE IF EXISTS `proprietaire_select_all`$$
CREATE DEFINER=`root`@`localhost` PROCEDURE `proprietaire_select_all` ()
BEGIN
	SELECT *
	FROM immorobin.proprietaire ;
END$$

DROP PROCEDURE IF EXISTS `proprietaire_select_one`$$
CREATE DEFINER=`root`@`localhost` PROCEDURE `proprietaire_select_one`
(`paramIDPRORIETAIRE` INT UNSIGNED)  BEGIN
	SELECT *
	FROM immorobin.proprietaire
	WHERE idProrietaire = paramIDPRORIETAIRE ;
END$$

DROP PROCEDURE IF EXISTS `proprietaire_update`$$
```

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `proprietaire_update`
(`paramIDPRORIETAIRE` INT UNSIGNED, `paramNOMPROPRIETAIRE`
VARCHAR(45), `paramPRENOMPROPRIETAIRE` VARCHAR(45),
`paramRAISONSOCIALE` VARCHAR(45), `paramTELDOMICILE` VARCHAR(45),
`paramTELMOBILE` VARCHAR(45), `paramTELBUREAU` VARCHAR(45),
`paramADRESSE` VARCHAR(45), `paramEMAILPROPRIETAIRE` VARCHAR(45),
`paramIDVILLE` INT UNSIGNED)  BEGIN
	UPDATE immorobin.proprietaire
	SET nomProprietaire = paramNOMPROPRIETAIRE,prenomProprietaire =
paramPRENOMPROPRIETAIRE,raisonSociale = paramRAISONSOCIALE,telDomicile
= paramTELDOMICILE,telMobile = paramTELMOBILE,telBureau =
paramTELBUREAU,adresse = paramADRESSE,emailProprietaire =
paramEMAILPROPRIETAIRE,idVille = paramIDVILLE
	WHERE idProrietaire = paramIDPRORIETAIRE ;
END$$

DROP PROCEDURE IF EXISTS `ville_delete`$$
CREATE DEFINER=`root`@`localhost` PROCEDURE `ville_delete` (`paramIDVILLE`
INT UNSIGNED)  BEGIN
	DELETE
	FROM immorobin.ville
	WHERE idVille = paramIDVILLE ;
END$$

DROP PROCEDURE IF EXISTS `ville_insert`$$
CREATE DEFINER=`root`@`localhost` PROCEDURE `ville_insert` (`paramIDVILLE`
INT UNSIGNED, `paramCPVILLE` VARCHAR(45), `paramNOMVILLE` VARCHAR(45),
`paramIDDEPARTEMENT` INT UNSIGNED)  BEGIN
	INSERT INTO immorobin.ville(idVille,cpVille,nomVille,idDepartement)

VALUES(paramIDVILLE,paramCPVILLE,paramNOMVILLE,paramIDDEPARTEMENT) ;
END$$

DROP PROCEDURE IF EXISTS `ville_select_all`$$
CREATE DEFINER=`root`@`localhost` PROCEDURE `ville_select_all` ()  BEGIN
	SELECT *
	FROM immorobin.ville ;
END$$

DROP PROCEDURE IF EXISTS `ville_select_one`$$
CREATE DEFINER=`root`@`localhost` PROCEDURE `ville_select_one`
(`paramIDVILLE` INT UNSIGNED)  BEGIN
	SELECT *
	FROM immorobin.ville
	WHERE idVille = paramIDVILLE ;
END$$

DROP PROCEDURE IF EXISTS `ville_update`$$
CREATE DEFINER=`root`@`localhost` PROCEDURE `ville_update`
(`paramIDVILLE` INT UNSIGNED, `paramCPVILLE` VARCHAR(45),
`paramNOMVILLE` VARCHAR(45), `paramIDDEPARTEMENT` INT UNSIGNED)
BEGIN
	UPDATE immorobin.ville
```

```
        SET cpVille = paramCPVILLE,nomVille = paramNOMVILLE,idDepartement =
paramIDDEPARTEMENT
        WHERE idVille = paramIDVILLE ;
END$$

DELIMITER ;

DROP TABLE IF EXISTS `alerte`;
CREATE TABLE `alerte` (
  `idAlerte` int(11) NOT NULL,
  `dateAlerte` date NOT NULL,
  `etatAlerte` varchar(50) NOT NULL,
  `idTypeAlerte` int(11) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

INSERT INTO `alerte` (`idAlerte`, `dateAlerte`, `etatAlerte`, `idTypeAlerte`)
VALUES
(2, '2017-03-28', 'A régler', 1),
(3, '2017-03-27', 'A régler', 1),
(4, '2017-03-30', 'A régler', 1),
(5, '2017-03-30', 'A régler', 2),
(17, '2017-03-28', 'A régler', 2),
(19, '2017-03-28', 'A régler', 1),
(26, '2017-04-02', 'Réglée', 4),
(35, '2017-04-02', 'A régler', 4),
(36, '2017-04-03', 'A régler', 1);

DROP TABLE IF EXISTS `bail`;
CREATE TABLE `bail` (
  `idBail` int(11) NOT NULL,
  `dateBail` date NOT NULL,
  `dateEntree` date NOT NULL,
  `dateRenouvellement` date NOT NULL,
  `idBien` int(11) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

DROP TABLE IF EXISTS `bailpersonne`;
CREATE TABLE `bailpersonne` (
  `idBail` int(11) NOT NULL,
  `idPersonne` int(11) NOT NULL,
  `idRole` int(11) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

DROP TABLE IF EXISTS `bien`;
CREATE TABLE `bien` (
  `idBien` int(11) NOT NULL,
  `designation` varchar(100) NOT NULL,
  `description` varchar(1000) NOT NULL,
  `photoPrincipale` varchar(50) DEFAULT NULL,
  `nombrePieces` int(11) NOT NULL,
  `surfaceTotale` double NOT NULL,
  `surfaceBureau` double NOT NULL,
  `surfaceEntrepots` double NOT NULL,
  `disponibilite` varchar(50) NOT NULL,
```

```
  `adresseBien` varchar(50) NOT NULL,
  `idVille` int(11) NOT NULL,
  `lat` double DEFAULT NULL,
  `lng` double DEFAULT NULL,
  `trimestreConstruction` varchar(50) NOT NULL,
  `loyer` double NOT NULL,
  `charges` double NOT NULL,
  `taxeFonciere` double NOT NULL,
  `idTypeBien` int(11) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

INSERT INTO `bien` (`idBien`, `designation`, `description`, `photoPrincipale`,
`nombrePieces`, `surfaceTotale`, `surfaceBureau`, `surfaceEntrepots`,
`disponibilite`, `adresseBien`, `idVille`, `lat`, `lng`, `trimestreConstruction`,
`loyer`, `charges`, `taxeFonciere`, `idTypeBien`) VALUES
(1, 'Bureau 1', 'Bureau 1', NULL, 1, 100, 100, 0, 'Dispo', 'rue', 1, NULL, NULL, '3',
0, 0, 300, 1),
(2, 'Bureau 2', 'Bureau 2', NULL, 1, 50, 50, 0, 'Dispo', 'fg', 1, NULL, NULL, '3', 0, 0,
300, 1),
(3, 'Bureau 3', 'Bureau 3', NULL, 1, 50, 50, 0, 'Dispo', 'fg', 1, NULL, NULL, '3', 0, 0,
300, 4),
(4, 'Bureau 4', 'Bureau 4', NULL, 1, 50, 50, 0, 'Dispo', 'bd', 1, NULL, NULL, '3', 0, 0,
300, 4),
(5, 'Commerce 1', 'Commerce 1', NULL, 1, 50, 50, 0, 'Dispo', 'fg', 1, NULL, NULL,
'3', 0, 0, 300, 3),
(6, 'Commerce 2', 'Commerce 2', NULL, 1, 50, 50, 0, 'Dispo', 'bd', 1, NULL, NULL,
'3', 0, 0, 300, 3),
(7, 'Commerce 3', 'Commerce 3', NULL, 1, 50, 50, 0, 'Dispo', 'bd', 1, NULL, NULL,
'3', 0, 0, 300, 3),
(9, 'Bureau 10', 'Bureau 10', 'bureau10.jpg', 1, 100, 50, 50, 'Oui', 'rue de la Paix',
1, 0, 0, '1200', 0, 0, 1000, 4);

DROP TABLE IF EXISTS `bienpersonne`;
CREATE TABLE `bienpersonne` (
  `idBien` int(11) NOT NULL,
  `idPersonne` int(11) NOT NULL,
  `idRole` int(11) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

DROP TABLE IF EXISTS `contratlocation`;
CREATE TABLE `contratlocation` (
  `idContratLocation` int(11) NOT NULL,
  `dateContratLocation` date NOT NULL,
  `dureeContratLocation` int(11) NOT NULL,
  `idBien` int(11) NOT NULL,
  `idPersonne` int(11) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

DROP TABLE IF EXISTS `departement`;
CREATE TABLE `departement` (
  `idDepartement` int(11) NOT NULL,
  `nomDepartement` varchar(50) NOT NULL,
  `codeDepartement` varchar(5) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

```
INSERT INTO `departement` (`idDepartement`, `nomDepartement`,
`codeDepartement`) VALUES
(1, 'AIN', '01'),
(2, 'AISNE', '02'),
(3, 'ALLIER', '03'),
(4, 'HAUTES-ALPES', '05'),
(5, 'ALPES-DE-HAUTE-PROVENCE', '04'),
(6, 'ALPES-MARITIMES', '06'),
(7, 'ARDÈCHE', '07'),
(8, 'ARDENNES', '08'),
(9, 'ARIÈGE', '09'),
(10, 'AUBE', '10'),
(11, 'AUDE', '11'),
(12, 'AVEYRON', '12'),
(13, 'BOUCHES-DU-RHÔNE', '13'),
(14, 'CALVADOS', '14'),
(15, 'CANTAL', '15'),
(16, 'CHARENTE', '16'),
(17, 'CHARENTE-MARITIME', '17'),
(18, 'CHER', '18'),
(19, 'CORRÈZE', '19'),
(20, 'CORSE-DU-SUD', '2a'),
(21, 'HAUTE-CORSE', '2b'),
(22, 'CÔTE-D\'OR', '21'),
(23, 'CÔTES-D\'ARMOR', '22'),
(24, 'CREUSE', '23'),
(25, 'DORDOGNE', '24'),
(26, 'DOUBS', '25'),
(27, 'DRÔME', '26'),
(28, 'EURE', '27'),
(29, 'EURE-ET-LOIR', '28'),
(30, 'FINISTÈRE', '29'),
(31, 'GARD', '30'),
(32, 'HAUTE-GARONNE', '31'),
(33, 'GERS', '32'),
(34, 'GIRONDE', '33'),
(35, 'HÉRAULT', '34'),
(36, 'ILE-ET-VILAINE', '35'),
(37, 'INDRE', '36'),
(38, 'INDRE-ET-LOIRE', '37'),
(39, 'ISÈRE', '38'),
(40, 'JURA', '39'),
(41, 'LANDES', '40'),
(42, 'LOIR-ET-CHER', '41'),
(43, 'LOIRE', '42'),
(44, 'HAUTE-LOIRE', '43'),
(45, 'LOIRE-ATLANTIQUE', '44'),
(46, 'LOIRET', '45'),
(47, 'LOT', '46'),
(48, 'LOT-ET-GARONNE', '47'),
(49, 'LOZÈRE', '48'),
(50, 'MAINE-ET-LOIRE', '49'),
(51, 'MANCHE', '50'),
```

```
(52, 'MARNE', '51'),
(53, 'HAUTE-MARNE', '52'),
(54, 'MAYENNE', '53'),
(55, 'MEURTHE-ET-MOSELLE', '54'),
(56, 'MEUSE', '55'),
(57, 'MORBIHAN', '56'),
(58, 'MOSELLE', '57'),
(59, 'NIÈVRE', '58'),
(60, 'NORD', '59'),
(61, 'OISE', '60'),
(62, 'ORNE', '61'),
(63, 'PAS-DE-CALAIS', '62'),
(64, 'PUY-DE-DÔME', '63'),
(65, 'PYRÉNÉES-ATLANTIQUES', '64'),
(66, 'HAUTES-PYRÉNÉES', '65'),
(67, 'PYRÉNÉES-ORIENTALES', '66'),
(68, 'BAS-RHIN', '67'),
(69, 'HAUT-RHIN', '68'),
(70, 'RHÔNE', '69'),
(71, 'HAUTE-SAÔNE', '70'),
(72, 'SAÔNE-ET-LOIRE', '71'),
(73, 'SARTHE', '72'),
(74, 'SAVOIE', '73'),
(75, 'HAUTE-SAVOIE', '74'),
(76, 'PARIS', '75'),
(77, 'SEINE-MARITIME', '76'),
(78, 'SEINE-ET-MARNE', '77'),
(79, 'YVELINES', '78'),
(80, 'DEUX-SÈVRES', '79'),
(81, 'SOMME', '80'),
(82, 'TARN', '81'),
(83, 'TARN-ET-GARONNE', '82'),
(84, 'VAR', '83'),
(85, 'VAUCLUSE', '84'),
(86, 'VENDÉE', '85'),
(87, 'VIENNE', '86'),
(88, 'HAUTE-VIENNE', '87'),
(89, 'VOSGES', '88'),
(90, 'YONNE', '89'),
(91, 'TERRITOIRE DE BELFORT', '90'),
(92, 'ESSONNE', '91'),
(93, 'HAUTS-DE-SEINE', '92'),
(94, 'SEINE-SAINT-DENIS', '93'),
(95, 'VAL-DE-MARNE', '94'),
(96, 'VAL-D\'OISE', '95'),
(97, 'MAYOTTE', '976'),
(98, 'GUADELOUPE', '971'),
(99, 'GUYANE', '973'),
(100, 'MARTINIQUE', '972'),
(101, 'RÉUNION', '974');

DROP TABLE IF EXISTS `etatalerte`;
CREATE TABLE `etatalerte` (
  `etatAlerte` varchar(50) NOT NULL
```

```
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

INSERT INTO `etatalerte` (`etatAlerte`) VALUES
('A régler'),
('En attente'),
('Réglé'),
('Réglée');

DROP TABLE IF EXISTS `gaz`;
CREATE TABLE `gaz` (
  `idGaz` int(11) NOT NULL,
  `dateDebutConsommation` date NOT NULL,
  `dateFinConsommation` date NOT NULL,
  `idBien` int(11) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

DROP TABLE IF EXISTS `personne`;
CREATE TABLE `personne` (
  `idPersonne` int(11) NOT NULL,
  `civilite` varchar(20) NOT NULL,
  `nom` varchar(50) NOT NULL,
  `prenom` varchar(50) NOT NULL,
  `raisonSociale` varchar(100) NOT NULL,
  `responsable` varchar(50) NOT NULL,
  `adresse` varchar(100) NOT NULL,
  `adresseFacturation` varchar(100) NOT NULL,
  `complementAdresse` varchar(100) NOT NULL,
  `idVille` int(11) NOT NULL,
  `telMobile` varchar(50) NOT NULL,
  `telDomicile` varchar(50) NOT NULL,
  `telBureau` varchar(50) NOT NULL,
  `email` varchar(50) NOT NULL,
  `mdp` varchar(50) NOT NULL,
  `idTypePersonne` int(11) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

INSERT INTO `personne` (`idPersonne`, `civilite`, `nom`, `prenom`,
`raisonSociale`, `responsable`, `adresse`, `adresseFacturation`,
`complementAdresse`, `idVille`, `telMobile`, `telDomicile`, `telBureau`, `email`,
`mdp`, `idTypePersonne`) VALUES
(1, 'Monsieur', 'Escoubas', 'Thomas', '', '', 'Rue Alexandre Dumas', '', '', 1,
'0707070707', '0101010101', '0102030405', 'tom@gmail.com', 'Mdp123', 4),
(2, 'Monsieur', 'Client1', 'Client1', 'Client1', 'Responsable1', 'Rue de la Paix', 'Rue
de la Paix', '', 1, '060600606', '0101010101', '0111111111', 'client1@gmail.com',
'Mdp123', 2),
(3, 'Madame', 'Client2', 'Client2', 'Client2', 'Responsable2', 'Rue de la Guerre',
'Rue de la Guerre', '', 1, '0707070777', '0101010102', '0101010122',
'client2@free.fr', 'Mdp123', 2),
(6, 'Monsieur', 'dupont', 'dupont', '', '', 'rue de Paris', '', '', 1, '0707070701',
'0101010101', '0102030401', 'dupont@gmail.com', 'Mdp123', 5),
(8, 'Monsieur', 'b', 'p', '', '', 'rue de Paris', '', '', 1, '0707070756', '0101010155',
'0102030455', 'pb@gmail.com', 'Mdp123', 3);

DROP TABLE IF EXISTS `quittanceloyer`;
```

```
CREATE TABLE `quittanceloyer` (
  `idQuittanceLoyer` int(11) NOT NULL,
  `dateQuittance` date NOT NULL,
  `montantHorsCharges` double NOT NULL,
  `charges` double NOT NULL,
  `etatQuittanceLoyer` varchar(50) NOT NULL,
  `idBail` int(11) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

DROP TABLE IF EXISTS `role`;
CREATE TABLE `role` (
  `idRole` int(11) NOT NULL,
  `role` varchar(50) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

INSERT INTO `role` (`idRole`, `role`) VALUES
(1, 'Propriétaire'),
(2, 'Locataire');

DROP TABLE IF EXISTS `typealerte`;
CREATE TABLE `typealerte` (
  `idTypeAlerte` int(11) NOT NULL,
  `typeAlerte` varchar(50) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

INSERT INTO `typealerte` (`idTypeAlerte`, `typeAlerte`) VALUES
(4, 'Impôts sur les sociétés'),
(3, 'Taxe professionnelle'),
(1, 'TVA'),
(2, 'URSSAF');

DROP TABLE IF EXISTS `typebien`;
CREATE TABLE `typebien` (
  `idTypeBien` int(11) NOT NULL,
  `typeBien` varchar(50) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

INSERT INTO `typebien` (`idTypeBien`, `typeBien`) VALUES
(1, 'Appartement'),
(4, 'Bureau'),
(3, 'Local commercial'),
(2, 'Maison');

DROP TABLE IF EXISTS `typepersonne`;
CREATE TABLE `typepersonne` (
  `idTypePersonne` int(11) NOT NULL,
  `typePersonne` varchar(50) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

INSERT INTO `typepersonne` (`idTypePersonne`, `typePersonne`) VALUES
(3, 'Commercial'),
(4, 'Gérant'),
(2, 'Locataire'),
(1, 'Proprétaire'),
```

```
(5, 'Prospect');

DROP TABLE IF EXISTS `ville`;
CREATE TABLE `ville` (
  `idVille` int(11) NOT NULL,
  `nomVille` varchar(50) NOT NULL,
  `cp` char(5) NOT NULL,
  `lat` double DEFAULT NULL,
  `lng` double DEFAULT NULL,
  `idDepartement` int(11) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

INSERT INTO `ville` (`idVille`, `nomVille`, `cp`, `lat`, `lng`, `idDepartement`)
VALUES
(1, 'Paris', '', NULL, NULL, 76);


ALTER TABLE `alerte`
  ADD PRIMARY KEY (`idAlerte`),
  ADD KEY `idTypeAlerte` (`idTypeAlerte`),
  ADD KEY `etatAlerte` (`etatAlerte`);

ALTER TABLE `bail`
  ADD PRIMARY KEY (`idBail`),
  ADD KEY `idBien` (`idBien`);

ALTER TABLE `bailpersonne`
  ADD PRIMARY KEY (`idBail`,`idPersonne`),
  ADD KEY `idRole` (`idRole`),
  ADD KEY `FK_BailPersonne_personne` (`idPersonne`);

ALTER TABLE `bien`
  ADD PRIMARY KEY (`idBien`),
  ADD UNIQUE KEY `designation` (`designation`),
  ADD KEY `idVille` (`idVille`),
  ADD KEY `idTypeBien` (`idTypeBien`);

ALTER TABLE `bienpersonne`
  ADD KEY `idRole` (`idRole`),
  ADD KEY `FK_BienPersonne_role` (`idBien`),
  ADD KEY `FK_BienPersonne_personne` (`idPersonne`);

ALTER TABLE `contratlocation`
  ADD PRIMARY KEY (`idContratLocation`),
  ADD KEY `idx_contrat_bien` (`idBien`),
  ADD KEY `idx_contrat_personne` (`idPersonne`);

ALTER TABLE `departement`
  ADD PRIMARY KEY (`idDepartement`),
  ADD UNIQUE KEY `nomDepartement` (`nomDepartement`),
  ADD UNIQUE KEY `codeDepartement` (`codeDepartement`);

ALTER TABLE `etatalerte`
  ADD PRIMARY KEY (`etatAlerte`);
```

```
ALTER TABLE `gaz`
  ADD PRIMARY KEY (`idGaz`),
  ADD KEY `idBien` (`idBien`);

ALTER TABLE `personne`
  ADD PRIMARY KEY (`idPersonne`),
  ADD UNIQUE KEY `email` (`email`),
  ADD UNIQUE KEY `telMobile` (`telMobile`),
  ADD KEY `fk_personne_ville` (`idVille`),
  ADD KEY `idTypePersonne` (`idTypePersonne`);

ALTER TABLE `quittanceloyer`
  ADD PRIMARY KEY (`idQuittanceLoyer`),
  ADD KEY `idBail` (`idBail`);

ALTER TABLE `role`
  ADD PRIMARY KEY (`idRole`);

ALTER TABLE `typealerte`
  ADD PRIMARY KEY (`idTypeAlerte`),
  ADD UNIQUE KEY `typeAlerte` (`typeAlerte`);

ALTER TABLE `typebien`
  ADD PRIMARY KEY (`idTypeBien`),
  ADD UNIQUE KEY `typeBien` (`typeBien`);

ALTER TABLE `typepersonne`
  ADD PRIMARY KEY (`idTypePersonne`),
  ADD UNIQUE KEY `typePersonne` (`typePersonne`);

ALTER TABLE `ville`
  ADD PRIMARY KEY (`idVille`),
  ADD KEY `nomVille` (`nomVille`),
  ADD KEY `idDepartement` (`idDepartement`);


ALTER TABLE `alerte`
  MODIFY `idAlerte` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=37;
ALTER TABLE `bail`
  MODIFY `idBail` int(11) NOT NULL AUTO_INCREMENT;
ALTER TABLE `bien`
  MODIFY `idBien` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=10;
ALTER TABLE `contratlocation`
  MODIFY `idContratLocation` int(11) NOT NULL AUTO_INCREMENT;
ALTER TABLE `departement`
  MODIFY `idDepartement` int(11) NOT NULL AUTO_INCREMENT,
AUTO_INCREMENT=102;
ALTER TABLE `gaz`
  MODIFY `idGaz` int(11) NOT NULL AUTO_INCREMENT;
ALTER TABLE `personne`
  MODIFY `idPersonne` int(11) NOT NULL AUTO_INCREMENT,
AUTO_INCREMENT=9;
ALTER TABLE `quittanceloyer`
```

```
  MODIFY `idQuittanceLoyer` int(11) NOT NULL AUTO_INCREMENT;
ALTER TABLE `role`
  MODIFY `idRole` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=3;
ALTER TABLE `typealerte`
  MODIFY `idTypeAlerte` int(11) NOT NULL AUTO_INCREMENT,
AUTO_INCREMENT=5;
ALTER TABLE `typebien`
  MODIFY `idTypeBien` int(11) NOT NULL AUTO_INCREMENT,
AUTO_INCREMENT=5;
ALTER TABLE `typepersonne`
  MODIFY `idTypePersonne` int(11) NOT NULL AUTO_INCREMENT,
AUTO_INCREMENT=6;
ALTER TABLE `ville`
  MODIFY `idVille` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=2;

ALTER TABLE `alerte`
  ADD CONSTRAINT `Alerte_ibfk_1` FOREIGN KEY (`idTypeAlerte`) REFERENCES
`typealerte` (`idTypeAlerte`),
  ADD CONSTRAINT `Alerte_ibfk_2` FOREIGN KEY (`etatAlerte`) REFERENCES
`etatalerte` (`etatAlerte`);

ALTER TABLE `bail`
  ADD CONSTRAINT `fk_bail_bien` FOREIGN KEY (`idBien`) REFERENCES `bien`
(`idBien`);

ALTER TABLE `bailpersonne`
  ADD CONSTRAINT `FK_BailPersonne_bail` FOREIGN KEY (`idBail`) REFERENCES
`bail` (`idBail`),
  ADD CONSTRAINT `FK_BailPersonne_personne` FOREIGN KEY (`idPersonne`)
REFERENCES `personne` (`idPersonne`),
  ADD CONSTRAINT `FK_BailPersonne_role` FOREIGN KEY (`idRole`) REFERENCES
`role` (`idRole`);

ALTER TABLE `bien`
  ADD CONSTRAINT `Bien_ibfk_1` FOREIGN KEY (`idTypeBien`) REFERENCES
`typebien` (`idTypeBien`),
  ADD CONSTRAINT `fk_biens_ville` FOREIGN KEY (`idVille`) REFERENCES `ville`
(`idVille`);

ALTER TABLE `bienpersonne`
  ADD CONSTRAINT `FK_BienPersonne_bien` FOREIGN KEY (`idRole`)
REFERENCES `role` (`idRole`),
  ADD CONSTRAINT `FK_BienPersonne_personne` FOREIGN KEY (`idPersonne`)
REFERENCES `personne` (`idPersonne`),
  ADD CONSTRAINT `FK_BienPersonne_role` FOREIGN KEY (`idBien`) REFERENCES
`bien` (`idBien`);

ALTER TABLE `gaz`
  ADD CONSTRAINT `Gaz_ibfk_1` FOREIGN KEY (`idBien`) REFERENCES `bien`
(`idBien`);

ALTER TABLE `personne`
  ADD CONSTRAINT `Personne_ibfk_1` FOREIGN KEY (`idTypePersonne`)
REFERENCES `typepersonne` (`idTypePersonne`),
```

```
  ADD CONSTRAINT `fk_personne_ville` FOREIGN KEY (`idVille`) REFERENCES
`ville` (`idVille`);

ALTER TABLE `quittanceloyer`
  ADD CONSTRAINT `QuittanceLoyer_ibfk_1` FOREIGN KEY (`idBail`) REFERENCES
`bail` (`idBail`);

ALTER TABLE `ville`
  ADD CONSTRAINT `fk_ville_departement` FOREIGN KEY (`idDepartement`)
REFERENCES `departement` (`idDepartement`);

/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;
```

9.4.1.2 - Code complet des procédures stockées

```
DELIMITER $$
DROP PROCEDURE IF EXISTS `departement_delete`$$
CREATE DEFINER=`root`@`localhost` PROCEDURE `departement_delete`
(`paramCODEDEPARTEMENT` CHAR(4))  BEGIN
     DELETE
     FROM immorobin.departement
     WHERE codeDepartement = paramCODEDEPARTEMENT ;
END$$

DROP PROCEDURE IF EXISTS `departement_insert`$$
CREATE DEFINER=`root`@`localhost` PROCEDURE `departement_insert`
(`paramIDDEPARTEMENT` INT UNSIGNED, `paramCODEDEPARTEMENT` CHAR(4),
`paramNOMDEPARTEMENT` VARCHAR(45))  BEGIN
     INSERT INTO
immorobin.departement(idDepartement,codeDepartement,nomDepartement)

VALUES(paramIDDEPARTEMENT,paramCODEDEPARTEMENT,paramNOMDEPARTEMENT)
;
END$$

DROP PROCEDURE IF EXISTS `departement_select_all`$$
CREATE DEFINER=`root`@`localhost` PROCEDURE
`departement_select_all` ()  BEGIN
     SELECT *
     FROM immorobin.departement ;
END$$

DROP PROCEDURE IF EXISTS `departement_select_one`$$
CREATE DEFINER=`root`@`localhost` PROCEDURE
`departement_select_one` (`paramCODEDEPARTEMENT` CHAR(4))  BEGIN
     SELECT *
     FROM immorobin.departement
     WHERE codeDepartement = paramCODEDEPARTEMENT ;
END$$

DROP PROCEDURE IF EXISTS `departement_update`$$
CREATE DEFINER=`root`@`localhost` PROCEDURE `departement_update`
(`paramIDDEPARTEMENT` INT UNSIGNED, `paramCODEDEPARTEMENT` CHAR(4),
`paramNOMDEPARTEMENT` VARCHAR(45))  BEGIN
     UPDATE immorobin.departement
     SET idDepartement = paramIDDEPARTEMENT,nomDepartement =
paramNOMDEPARTEMENT
     WHERE codeDepartement = paramCODEDEPARTEMENT ;
END$$

DROP PROCEDURE IF EXISTS `proprietaire_delete`$$
CREATE DEFINER=`root`@`localhost` PROCEDURE `proprietaire_delete`
(`paramIDPRORIETAIRE` INT UNSIGNED)  BEGIN
```

```
    DELETE
    FROM immorobin.proprietaire
    WHERE idProrietaire = paramIDPRORIETAIRE ;
END$$

DROP PROCEDURE IF EXISTS `proprietaire_insert`$$
CREATE DEFINER=`root`@`localhost` PROCEDURE `proprietaire_insert`
(`paramIDPRORIETAIRE` INT UNSIGNED, `paramNOMPROPRIETAIRE`
VARCHAR(45), `paramPRENOMPROPRIETAIRE` VARCHAR(45),
`paramRAISONSOCIALE` VARCHAR(45), `paramTELDOMICILE` VARCHAR(45),
`paramTELMOBILE` VARCHAR(45), `paramTELBUREAU` VARCHAR(45),
`paramADRESSE` VARCHAR(45), `paramEMAILPROPRIETAIRE` VARCHAR(45),
`paramIDVILLE` INT UNSIGNED)  BEGIN
    INSERT INTO
immorobin.proprietaire(idProrietaire,nomProprietaire,prenomPropriet
aire,raisonSociale,telDomicile,telMobile,telBureau,adresse,emailPro
prietaire,idVille)

VALUES(paramIDPRORIETAIRE,paramNOMPROPRIETAIRE,paramPRENOMPROPRIETA
IRE,paramRAISONSOCIALE,paramTELDOMICILE,paramTELMOBILE,paramTELBURE
AU,paramADRESSE,paramEMAILPROPRIETAIRE,paramIDVILLE) ;
END$$

DROP PROCEDURE IF EXISTS `proprietaire_select_all`$$
CREATE DEFINER=`root`@`localhost` PROCEDURE
`proprietaire_select_all` ()  BEGIN
    SELECT *
    FROM immorobin.proprietaire ;
END$$

DROP PROCEDURE IF EXISTS `proprietaire_select_one`$$
CREATE DEFINER=`root`@`localhost` PROCEDURE
`proprietaire_select_one` (`paramIDPRORIETAIRE` INT UNSIGNED)
BEGIN
    SELECT *
    FROM immorobin.proprietaire
    WHERE idProrietaire = paramIDPRORIETAIRE ;
END$$

DROP PROCEDURE IF EXISTS `proprietaire_update`$$
CREATE DEFINER=`root`@`localhost` PROCEDURE `proprietaire_update`
(`paramIDPRORIETAIRE` INT UNSIGNED, `paramNOMPROPRIETAIRE`
VARCHAR(45), `paramPRENOMPROPRIETAIRE` VARCHAR(45),
`paramRAISONSOCIALE` VARCHAR(45), `paramTELDOMICILE` VARCHAR(45),
`paramTELMOBILE` VARCHAR(45), `paramTELBUREAU` VARCHAR(45),
`paramADRESSE` VARCHAR(45), `paramEMAILPROPRIETAIRE` VARCHAR(45),
`paramIDVILLE` INT UNSIGNED)  BEGIN
    UPDATE immorobin.proprietaire
    SET nomProprietaire = paramNOMPROPRIETAIRE,prenomProprietaire =
paramPRENOMPROPRIETAIRE,raisonSociale =
paramRAISONSOCIALE,telDomicile = paramTELDOMICILE,telMobile =
paramTELMOBILE,telBureau = paramTELBUREAU,adresse =
```

```
paramADRESSE,emailProprietaire = paramEMAILPROPRIETAIRE,idVille =
paramIDVILLE
     WHERE idProrietaire = paramIDPRORIETAIRE ;
END$$

DROP PROCEDURE IF EXISTS `ville_delete`$$
CREATE DEFINER=`root`@`localhost` PROCEDURE `ville_delete`
(`paramIDVILLE` INT UNSIGNED)  BEGIN
     DELETE
     FROM immorobin.ville
     WHERE idVille = paramIDVILLE ;
END$$

DROP PROCEDURE IF EXISTS `ville_insert`$$
CREATE DEFINER=`root`@`localhost` PROCEDURE `ville_insert`
(`paramIDVILLE` INT UNSIGNED, `paramCPVILLE` VARCHAR(45),
`paramNOMVILLE` VARCHAR(45), `paramIDDEPARTEMENT` INT UNSIGNED)
BEGIN
     INSERT INTO
immorobin.ville(idVille,cpVille,nomVille,idDepartement)

VALUES(paramIDVILLE,paramCPVILLE,paramNOMVILLE,paramIDDEPARTEMENT)
;
END$$

DROP PROCEDURE IF EXISTS `ville_select_all`$$
CREATE DEFINER=`root`@`localhost` PROCEDURE `ville_select_all` ()
BEGIN
     SELECT *
     FROM immorobin.ville ;
END$$

DROP PROCEDURE IF EXISTS `ville_select_one`$$
CREATE DEFINER=`root`@`localhost` PROCEDURE `ville_select_one`
(`paramIDVILLE` INT UNSIGNED)  BEGIN
     SELECT *
     FROM immorobin.ville
     WHERE idVille = paramIDVILLE ;
END$$

DROP PROCEDURE IF EXISTS `ville_update`$$
CREATE DEFINER=`root`@`localhost` PROCEDURE `ville_update`
(`paramIDVILLE` INT UNSIGNED, `paramCPVILLE` VARCHAR(45),
`paramNOMVILLE` VARCHAR(45), `paramIDDEPARTEMENT` INT UNSIGNED)
BEGIN
     UPDATE immorobin.ville
     SET cpVille = paramCPVILLE,nomVille =
paramNOMVILLE,idDepartement = paramIDDEPARTEMENT
     WHERE idVille = paramIDVILLE ;
END$$
```

9.4.1.3 - Code pour l'insertion de données de test dans la BD

```
INSERT INTO alerte (idAlerte, dateAlerte, etatAlerte, idTypeAlerte)
VALUES
(2, '2017-03-28', 'A régler', 1),
(3, '2017-03-27', 'A régler', 1),
(4, '2017-03-30', 'A régler', 1),
(5, '2017-03-30', 'A régler', 2),
(17, '2017-03-28', 'A régler', 2),
(19, '2017-03-28', 'A régler', 1),
(26, '2017-04-02', 'Réglée', 4),
(35, '2017-04-02', 'A régler', 4),
(36, '2017-04-03', 'A régler', 1);
```

INSERT INTO bien (idBien, designation, description, photoPrincipale, nombrePieces, surfaceTotale, surfaceBureau, surfaceEntrepots, disponibilite, adresseBien, idVille, lat, lng, trimestreConstruction, loyer, charges, taxeFonciere, idTypeBien) VALUES
(1, 'Bureau 1', 'Bureau 1', NULL, 1, 100, 100, 0, 'Dispo', 'rue', 1, NULL, NULL, '3', 0, 0, 300, 1),
(2, 'Bureau 2', 'Bureau 2', NULL, 1, 50, 50, 0, 'Dispo', 'fg', 1, NULL, NULL, '3', 0, 0, 300, 1),
(3, 'Bureau 3', 'Bureau 3', NULL, 1, 50, 50, 0, 'Dispo', 'fg', 1, NULL, NULL, '3', 0, 0, 300, 4),
(4, 'Bureau 4', 'Bureau 4', NULL, 1, 50, 50, 0, 'Dispo', 'bd', 1, NULL, NULL, '3', 0, 0, 300, 4),
(5, 'Commerce 1', 'Commerce 1', NULL, 1, 50, 50, 0, 'Dispo', 'fg', 1, NULL, NULL, '3', 0, 0, 300, 3),
(6, 'Commerce 2', 'Commerce 2', NULL, 1, 50, 50, 0, 'Dispo', 'bd', 1, NULL, NULL, '3', 0, 0, 300, 3),
(7, 'Commerce 3', 'Commerce 3', NULL, 1, 50, 50, 0, 'Dispo', 'bd', 1, NULL, NULL, '3', 0, 0, 300, 3),
(9, 'Bureau 10', 'Bureau 10', 'bureau10.jpg', 1, 100, 50, 50, 'Oui', 'rue de la Paix', 1, 0, 0, '1200', 0, 0, 1000, 4);

INSERT INTO personne (idPersonne, civilite, nom, prenom, raisonSociale, responsable, adresse, adresseFacturation, complementAdresse, idVille, telMobile, telDomicile, telBureau, email, mdp, idTypePersonne) VALUES
(1, 'Monsieur', 'Escoubas', 'Thomas', '', '', 'Rue Alexandre Dumas', '', '', 1, '0707070707', '0101010101', '0102030405', 'tom@gmail.com', 'Mdp123', 4),
(2, 'Monsieur', 'Client1', 'Client1', 'Client1', 'Responsable1', 'Rue de la Paix', 'Rue de la Paix', '', 1, '060600606', '0101010101', '0111111111', 'client1@gmail.com', 'Mdp123', 2),
(3, 'Madame', 'Client2', 'Client2', 'Client2', 'Responsable2', 'Rue de la Guerre', 'Rue de la Guerre', '', 1, '0707070777', '0101010102', '0101010122', 'client2@free.fr', 'Mdp123', 2),
(6, 'Monsieur', 'dupont', 'dupont', '', '', 'rue de Paris', '', '', 1, '0707070701', '0101010101', '0102030401', 'dupont@gmail.com', 'Mdp123', 5),
(8, 'Monsieur', 'b', 'p', '', '', 'rue de Paris', '', '', 1, '0707070756', '0101010155', '0102030455', 'pb@gmail.com', 'Mdp123', 3);

9.4.1.4 - DAO complet

```
<?php

/*
 *
 */

/**
 * Description of BienDAO
 *
 * @author thomas
 */
class BienDAO {

    private $pdo;

    public function __construct(PDO $pdo) {
        $this->pdo = $pdo;
    }

    /**
     *
     * @return type
     */
    public function selectAll() {
        $sql = 'SELECT * ';
        $sql .= ' FROM ImmoRobin.Bien';
        $sql .= ' ORDER BY designation';
        // Renvoie un tableau ordinal de
        return $this->pdo->query($sql)->fetchAll(PDO::FETCH_NUM);
    }

    /**
     *
     * @return type
     */
    public function selectAllWithTypeBien() {
//        $sql = 'SELECT
b.idBien,b.designation,b.description,b.photoPrincipale,b.nombrePieces,b.surfaceT
otale,b.surfaceBureau,b.surfaceEntrepots,b.disponibilite,b.adresseBien,b.idVille,b.
lat,b.lng,b.trimestreConstruction,b.loyer,b.charges,b.taxeFonciere';
//        $sql .= ',t.typeBien';
//        //$sql = 'SELECT * ';
//        $sql .= ' FROM ImmoRobin.Bien b INNER JOIN ImmoRobin.TypeBien t';
//        $sql .= ' ON b.idTypeBien = t.idTypeBien ';
//        $sql .= ' ORDER BY b.designation';

        $sql = 'SELECT
b.idBien,b.designation,b.description,b.nombrePieces,b.surfaceTotale';
```

```
        $sql .= ',t.typeBien';
        //$sql = 'SELECT * ';
        $sql .= ' FROM ImmoRobin.Bien b INNER JOIN ImmoRobin.TypeBien t';
        $sql .= ' ON b.idTypeBien = t.idTypeBien ';
        $sql .= ' ORDER BY b.designation';
        // Renvoie un tableau ordinal de
        return $this->pdo->query($sql)->fetchAll(PDO::FETCH_NUM);
    }

    /**
     *
     * @param type $typeBien
     * @return type
     */
    public function selectByTypeBien($typeBien) {

        $sql = 'SELECT
b.idBien,b.designation,b.description,b.photoPrincipale,b.nombrePieces,b.surfaceT
otale,b.surfaceBureau,b.surfaceEntrepots,b.disponibilite,b.adresseBien,b.idVille,b.
lat,b.lng,b.trimestreConstruction,b.loyer,b.charges,b.taxeFonciere';
        $sql .= ',t.typeBien';
        $sql .= ' FROM ImmoRobin.Bien b INNER JOIN ImmoRobin.TypeBien t';
        $sql .= ' ON b.idTypeBien = t.idTypeBien ';
        $sql .= ' WHERE t.idTypeBien = ? ';
        $sql .= ' ORDER BY b.designation';
        // Renvoie un tableau ordinal de biens
        $tEnrs = array();
        try {
            $lrs = $this->pdo->prepare($sql);
            $lrs->bindParam(1, $typeBien, PDO::PARAM_STR);
            $lrs->execute();
            $lrs->setFetchMode(PDO::FETCH_NUM);
            while ($enr = $lrs->fetch()) {
                $tEnrs[] = $enr;
            }
        } catch (Exception $exc) {
            $tEnrs[] = $exc->getTraceAsString();
        }

        return $tEnrs;
    }

    /**
     *
     * @param type $id
     * @return type
     */
    public function selectOne($id) {
        $enr = null;
        try {
            $sql = 'SELECT * FROM ImmoRobin.Bien WHERE idBien = ?';
            $lrs = $this->pdo->prepare($sql);
            $lrs->bindParam(1, $id, PDO::PARAM_STR);
            $lrs->execute();
```

```
//          $lrs->setFetchMode(PDO::FETCH_NUM);
            $enr = $lrs->fetch(PDO::FETCH_NUM);
        } catch (PDOException $e) {

        } /// catch
        return $enr;
    }

    /**
     *
     * @param PDO $pcnx
     * @param type $data
     * @return string
     */
    public function insert($data) {
        $t = array();
        try {
// Le SQL
            $description = $data["description"];
            $photoPrincipale = $data["photoPrincipale"];
            $nombrePieces = $data["nombrePieces"];
            $surfaceTotale = $data["surfaceTotale"];
            $surfaceBureau = $data["surfaceBureau"];
            $surfaceEntrepots = $data["surfaceEntrepots"];
            $disponibilite = $data["disponibilite"];
            $adresse = $data["adresse"];
            $trimestreConstruction = $data["trimestreConstruction"];
            $taxeFonciere = $data["taxeFonciere"];
            $lat = $data["lat"];
            $lng = $data["lng"];
            $idVille = $data["idVille"];
            $idTypeBien = $data["idTypeBien"];

            $lsSQL = 'INSERT INTO
Bien(description,photoPrincipale,nombrePieces,surfaceTotale,surfaceBureau,surfa
ceEntrepots,disponibilite,adresse,trimestreConstruction,taxeFonciere,lat,lng,idVill
e,idTypeBien)
VALUES(?,?,?,?,?,?,?,?,?,?,?,?,?,?)';
            $lstmt = $this->pdo->prepare($lsSQL);
            $lstmt->execute(array($description, $photoPrincipale, $nombrePieces,
$surfaceTotale, $surfaceBureau, $surfaceEntrepots, $disponibilite, $adresse,
$trimestreConstruction, $taxeFonciere, $lat, $lng, $idVille, $idTypeBien));
//          echo $lstmt->rowCount();

            $t["message"] = "OK";
        } catch (Exception $e) {
            $pcnx->rollBack();
            $t["message"] = "Erreur : " . $e->getMessage();
        }
        return $t;
    }

    /**
     *
```

```
     * @param PDO $pcnx
     * @param type $data
     */
    public function delete($data) {
        $t = array();
        try {
// Le SQL
            $lsSQL = 'DELETE FROM Bien WHERE designation = ?';
            $lstmt = $this->pdo->prepare($lsSQL);
            $lstmt->execute(array($data["designation"]));
            //echo $lstmt->rowCount();
            $t["message"] = "OK";
        } catch (Exception $e) {
            $pcnx->rollBack();
            $t["message"] = "Erreur : " . $e->getMessage();
        }
        return $t;
    }

    /**
     *
     * @param PDO $pcnx
     * @param type $data
     */
    public function update($data) {
        $t = array();
        try {
// Le SQL
//          $lsSQL = 'UPDATE Personne
//SET
civilite=?,nom=?,prenom=?,raisonSociale=?,adresse=?,complementAdresse=?,id
Ville=?,telMobile=?,telDomicile=?,telBureau=?,email=?,mdp=?
//WHERE email = ?';
            $lstmt = $this->pdo->prepare($lsSQL);
//          $lstmt->execute(array($civilite, $nom, $prenom, $raisonSociale,
$adresse, $complementAdresse, $idVille, $telMobile, $telDomicile, $telBureau,
$email, $mdp, $email));
            //echo $lstmt->rowCount();

            $t["message"] = "OK";
        } catch (Exception $e) {
            $pcnx->rollBack();
            $t["message"] = "Erreur : " . $e->getMessage();
        }
        return $t;
    }

}
```

9.4.1.5 - Autres

```
/*
 * BienRecherche.js
 */

/**
 *
 * @returns {undefined}
 */
function init() {

    $("#btValider").on("click", function() {
        requeteAJAX("bienParType", $("#listeTypesDeBiens").val());
    });

} /// init

/**
 *
 * @param {type} psAction
 * @param {type} typeBien
 * @returns {undefined}
 */
function requeteAJAX(psAction, typeBien) {

$("#tbodyBiens").empty();

    console.log("requeteAJAX");
    console.log(typeBien);

    var url = '../routes/routes.php';

    var jqXHR = $.get(
            url,
            {
                action: psAction,
                critere: typeBien
            }
    );

    jqXHR.done(function(data) {
        console.log("data");
        console.log(data);
        // String 2 objet
        var objetJSON = JSON.parse(data);
        console.log("objetJSON");
        console.log(objetJSON);
        console.log("objetJSON.message");
        console.log(objetJSON.message);
```

```
        var liNbBiens = objetJSON.message.length;
        var bien;
        console.log("liNbBiens");
        console.log(liNbBiens);
        for (var i = 0; i < liNbBiens; i++) {
            bien = objetJSON.message[i];
            console.log("Bien");
            console.log(bien);
//            console.log(bien[1]);
//            console.log(bien[2]);

            var liNbChamps = bien.length;
            console.log("liNbChamps");

            console.log(liNbChamps);
            var tr = $("<tr>");
            for (var cle in bien) {
                var td = $("<td>");
                td.html(bien[cle]);
                $(tr).append(td);
            }
            $("#tbodyBiens").append(tr);
        }
    });

    jqXHR.fail(function(xhr, statut, erreur) {
        var lsTexte = xhr.status + ":" + xhr.statusText;
        $("#lblMessage").html(lsTexte);
    });
} /// requeteAJAX


$(document).ready(init);
```

9.5 - Quelques constantes

Constante	Valeur
Disponibilité (bien)	Oui| Non | Réservé
TypeBien	(1, 'Appartement'), (4, 'Bureau'), (3, 'Local commercial'), (2, 'Maison')

CHAPITRE 10 - TABLES ET INDEX

10.1 - Table des illustrations

Index personnalisé