

Documentation technique

Site web de gestion immobilière

Réalisé par
ABECASSIS Thomas

Sommaire

Table des figures	1
Description du document	3
Fonctionnalités du site	4
Fonctionnalités globales	4
Rôles et droits des utilisateurs	7
Fonctionnalités de l'utilisateur connecté	7
Fonctionnalités du commercial	10
Fonctionnalités des administrateurs	10
Fonctionnalités du super administrateur	13
Résumé des rôles et de leurs droits	14
Organisation des fichiers	15
Rapport Technique	17
Back-end	17
Installation	17
Fonctionnement global MVC	18
Technologies utilisées	19
Diagrammes de classes	20
Fonctionnement	22
Utilisation du webservice	26
Front-end	28
Technologies utilisées	28
Javascript	28

Table des figures

Figure 1 - Barre de recherche	4
Figure 2 - Recherche approfondie	5
Figure 3 - Résultat de la recherche	5
Figure 4 - Fiche d'un lot	6
Figure 5 - Bouton sauvegarde de recherche	7
Figure 6 - Recherches enregistrées	8
Figure 7 - Menu paramètres d'un compte	8
Figure 8 - Modifications d'un compte	9
Figure 9 - Suppression d'un compte	9
Figure 10 - Informations commerciales	10
Figure 11 - Menu de personnalisation	11
Figure 12 - Modification des couleurs	11
Figure 13 - Modifications des informations	12
Figure 14 - Recherche d'un utilisateur	12
Figure 15 - Modification de rôle d'un utilisateur	13
Figure 16 - Hiérarchie des dossiers	15
Figure 17 - Balise base	17
Figure 18 - Configuration connexion à la base de données	17
Figure 19 - Schéma MVC	18
Figure 20 - Diagramme de séquence MVC	19
Figure 21 - Diagramme de classes des Model	20
Figure 22 - Diagramme de classe des Controller	21
Figure 23 - Diagramme des classes lib	21
Figure 24 - Diagramme de classe de la base de données	22
Figure 25 - Table alerte	23
Figure 26 - Encodage et décodage ModelAlerte	23
Figure 27 - Session dans la fonction searched	24
Figure 28 - Session dans la fonction created	24
Figure 29 - ControllerErreur	24
Figure 30 - Diagramme ModelLot	25
Figure 31 - Exemple de mise en forme de tableau de critères	25
Figure 32 - Exemple de build_path	26
Figure 33 - Exemple de génération de vue	26
Figure 34 - Vérification authentification	27
Figure 35 - Fichiers javascript	28
Figure 36 - Appel de la requeteAJAX	29
Figure 37 - Callback requête ajax	29

Description du document

Ce document est la documentation technique du site web de gestion immobilière Priam.

Ce document est divisé en trois majeure partie :

- La présentation des fonctionnalités
- L'organisation des fichiers
- Le rapport technique

1. Fonctionnalités du site

Dans cette partie nous allons décrire les fonctionnalités présentes sur le site. Tout d'abord, nous verrons les fonctionnalités globales puis nous nous pencherons sur les différents rôles et droits des utilisateurs.

1.1. Fonctionnalités globales

La fonction principale du site est la recherche de lots immobiliers suivant différents critères. Lorsqu'un visiteur accède au site il est redirigé vers l'accueil où est présent une simple barre de recherche avec quatre champs, le premier "où" qui correspond à la ville sur laquelle l'utilisateur veut faire sa recherche. En rentrant le nom ou le code postale d'une ville une liste déroulante apparaît sur laquelle l'utilisateur peut choisir la ville qu'il cherche. Pour les autres champs (min m², min €, max €) l'utilisateur peut renseigner les nombres qui conviennent à ses critères.

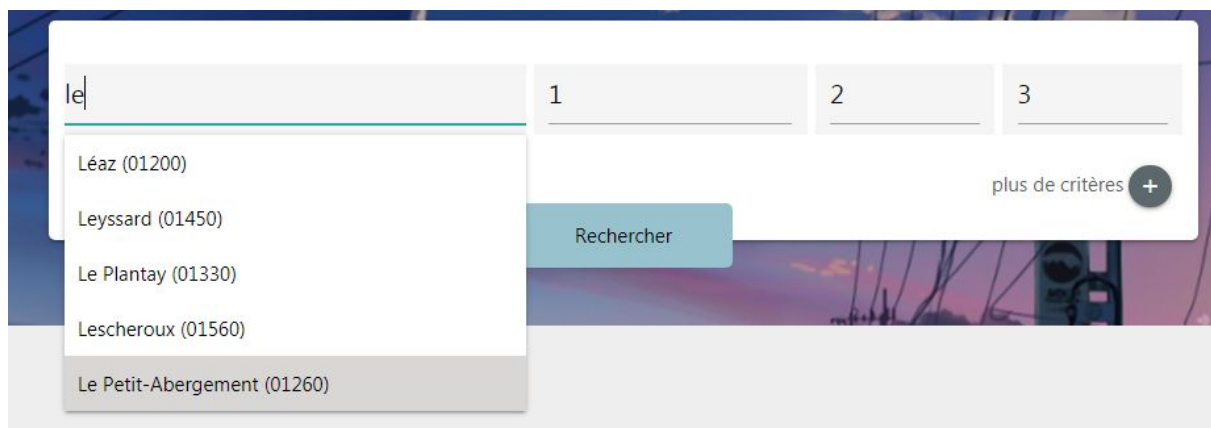
The image shows a search interface on a website. On the left, there is a text input field containing the letters 'le'. Below this field is a dropdown menu with five location suggestions: 'Léaz (01200)', 'Leyssard (01450)', 'Le Plantay (01330)', 'Lescheroux (01560)', and 'Le Petit-Abergement (01260)'. To the right of the text field are three separate input fields for numbers, labeled '1', '2', and '3'. Below these fields is a blue button labeled 'Rechercher'. To the right of the 'Rechercher' button is a link that says 'plus de critères' followed by a plus sign icon in a circle.

Figure 1 : Barre de recherche

S'il le souhaite, l'utilisateur peut accéder à une recherche approfondi en cliquant sur "plus de critères" où il aura accès à des critères supplémentaires.

Localisation

où ?

Budget

min
max

Surface

min
max

Type(s) de pièces


☐ salle de bain
☐ toilettes s

Rechercher

Figure 2 : Recherche approfondie

Une fois que l'utilisateur a renseigné les critères qui l'intéressent il peut faire sa recherche en cliquant sur "Rechercher". Une fois sa recherche faite il est redirigé vers une page contenant la liste des lots lui correspondant. Si plus de 15 lots correspondent à sa recherche la liste est divisée en plusieurs pages de 15 lots, dans ce cas l'utilisateur peut naviguer entre les pages en utilisant les boutons en bas de page.

← Retour à la recherche



location Maison 1 pièce

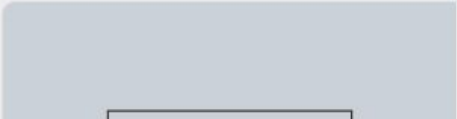
1 €/mois

🏠

1m²

📍

Ozan





location Appartement 3 pièces

Figure 3 : Résultat de la recherche

En cliquant sur un lot le visiteur a accès à différentes informations comme la ville où se trouve le lot, des images, une description. Si le lot intéresse le visiteur il peut contacter l'agence avec le mail et le numéro de téléphone fournis sur la page du lot.

[← Retour aux résultats](#)





location Maison 1 pièce
1€/mois CC

Spécificités

Type de bien : Maison	Adresse : Ozan
Surface : 1m²	Nombre de pièces : 1

Critères

Commodité(s)


- alarme

Description

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Ce bien vous intéresse ?

Contactez nous !

 **Téléphone**


 **e-mail**

Figure 4 : Fiche d'un lot

1.2. Rôles et droits des utilisateurs

Dans cette partie nous commencerons par nous pencher sur les fonctionnalités présentes pour les utilisateurs connectés puis nous nous pencherons sur celles des commerciaux ensuite nous verrons celles des administrateur puis nous énumérerons celles du super administrateur et pour finir nous aurons un tableaux récapitulatif de tous les droits des utilisateurs.

1.2.1. Fonctionnalités de l'utilisateur connecté

Le site propose une fonctionnalité de sauvegarde de recherche. Si l'utilisateur est connecté il peut sauvegarder une recherche effectuée en cliquant sur le bouton "sauvegarder la recherche" présent en bas de page pour les résultats d'une recherche.

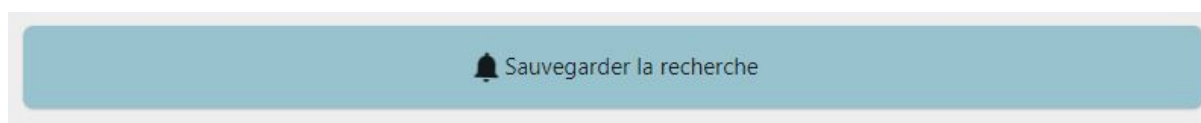


Figure 5 : Bouton sauvegarde de recherche

L'utilisateur retrouvera ensuite ses recherches enregistrées dans la page "Mes recherches". Sur cette page l'utilisateur peut modifier leurs noms pour mieux s'y retrouver, en supprimer certaines si elles ne lui sont plus utiles et voir les lots correspondant à ses recherches en utilisant le bouton "Voir les lots". De plus, le site propose une fonction d'alerte via mail si un nouveau lot respectant une des recherches de l'utilisateur est ajouté, le switch "m'alerter par mail" permet d'activer/désactiver ces alertes.

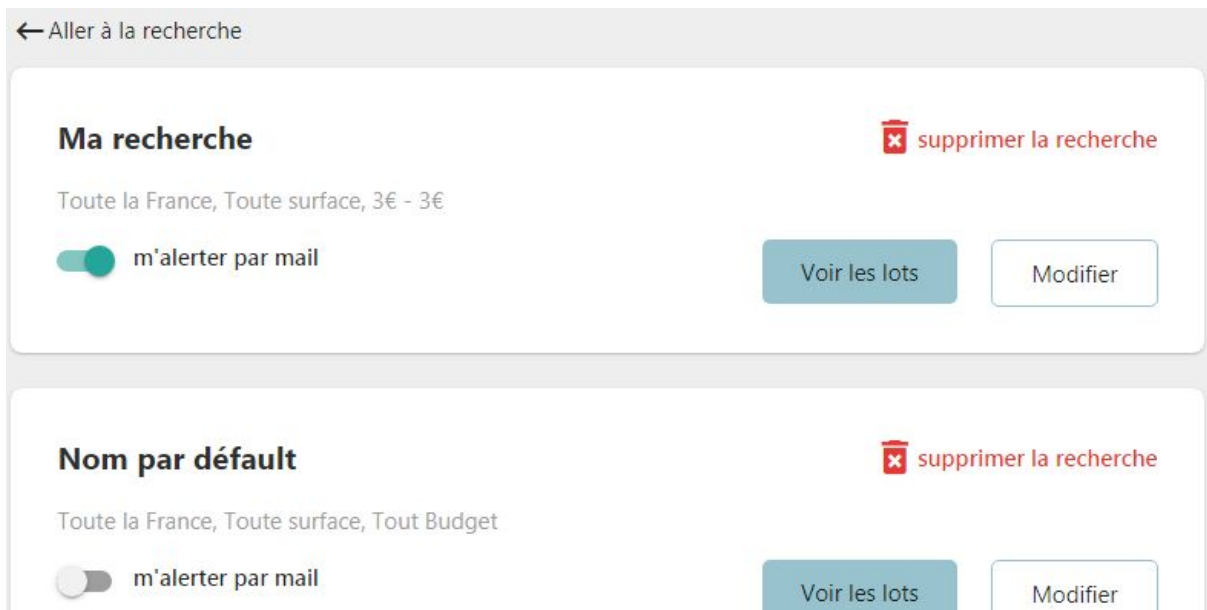


Figure 6 : Recherches enregistrées

L'utilisateur a la possibilité de modifier son compte en se rendant sur la page "Paramètres" (dans le menu "Mon compte").

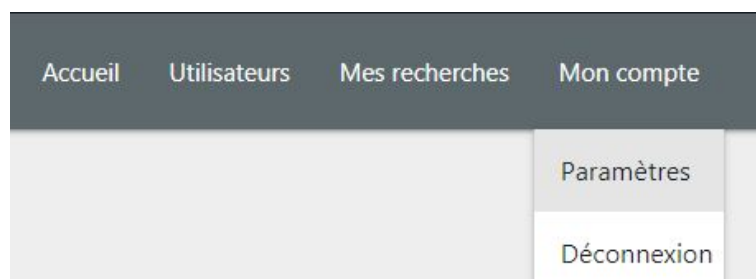


Figure 7 : Menu paramètres d'un compte

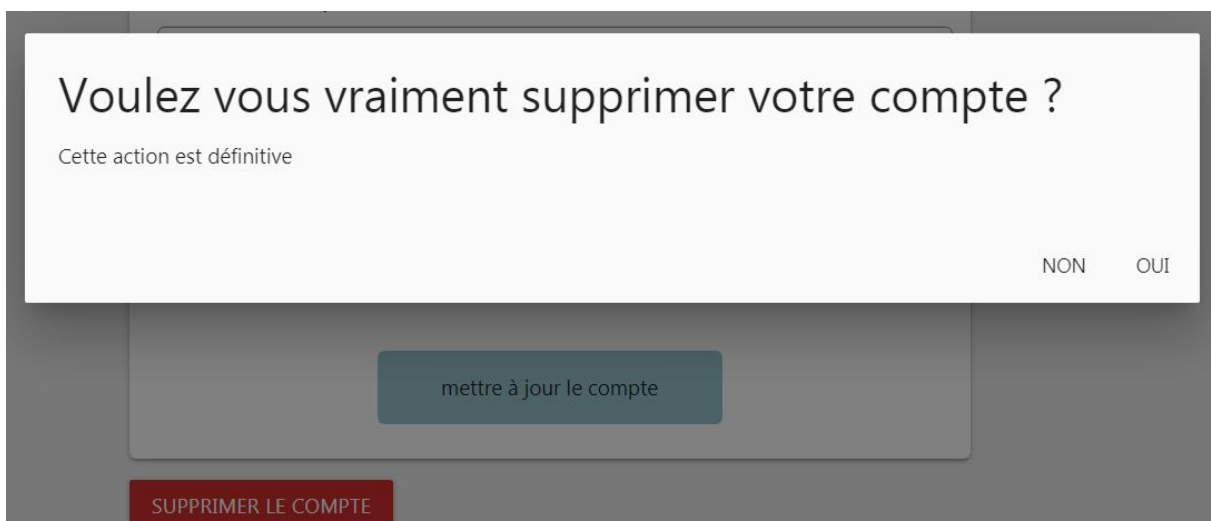
Sur cette page l'utilisateur peut modifier son nom, son prénom, son adresse e-mail et son mot de passe, ces deux derniers nécessitant de renseigner son mot de passe actuel.



The form is titled "Modifier vos informations" in a bold, dark font. It contains two input fields: one for "Nom" (Last Name) with the placeholder text "eaeae" and one for "Prenom" (First Name) with the placeholder text "edaasd". Below the input fields is a large, light blue button with the text "Mettre à jour vos informations".

Figure 8 : Modifications d'un compte

Pour les comptes sans rôles et les commerciaux la possibilité de supprimer leurs comptes est présente via le bouton "Supprimer le compte" en bas de la page.



The dialog box has a white background and a dark border. The main text is "Voulez vous vraiment supprimer votre compte ?" in a large, bold font. Below it, in a smaller font, is the text "Cette action est définitive". At the bottom right, there are two buttons: "NON" and "OUI". In the background, a blurred view of the account management page is visible, showing a "mettre à jour le compte" button and a "SUPPRIMER LE COMPTE" button.

Figure 9 : Suppression d'un compte

1.2.2. Fonctionnalités du commercial

Un compte commercial a accès à des informations supplémentaires sur les fiches des lots comme par exemple l'adresse exacte du lot et le numéro de téléphone du propriétaire.

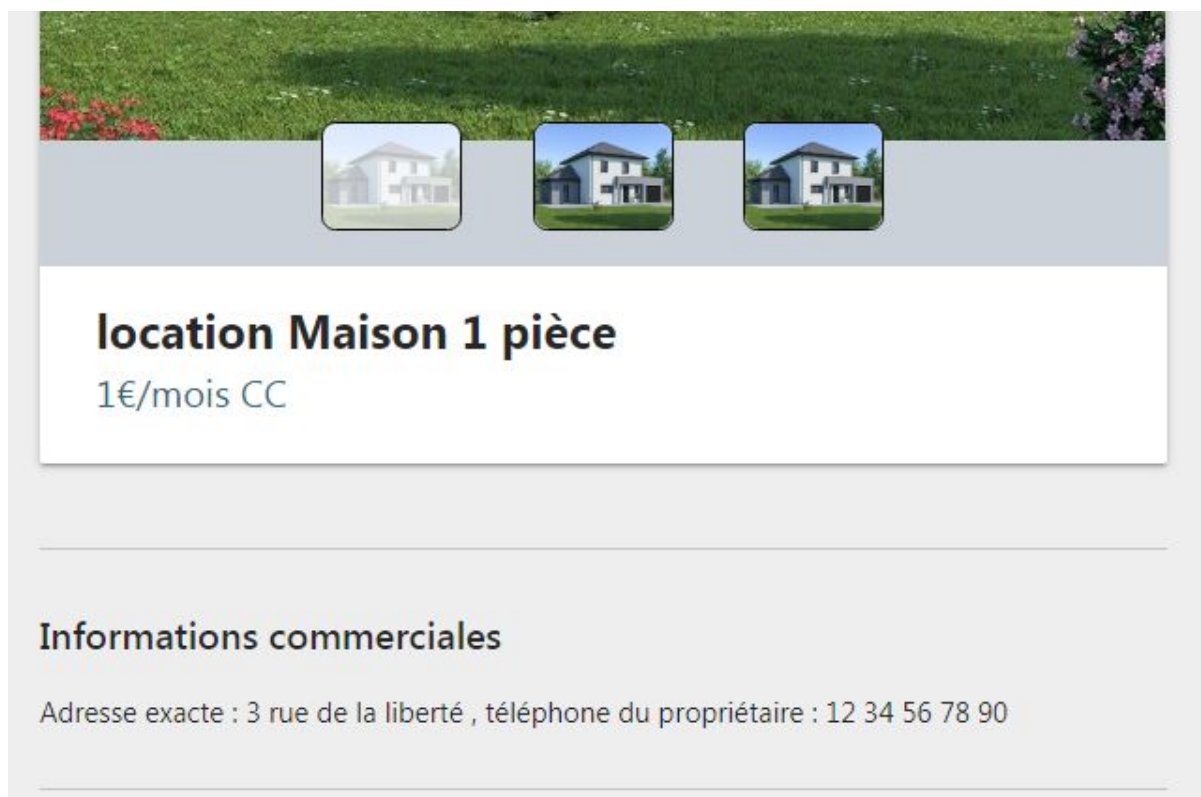


Figure 10 : Informations commerciales

1.2.3. Fonctionnalités des administrateurs

Les comptes administrateurs ont accès à des fonctionnalités supplémentaires comme la personnalisation du site web (pour tous les visiteurs). Cette personnalisation passe par le menu "Modifier votre site" présent sur toutes les pages lorsque l'administrateur est connecté. On y retrouve trois sous-menu, "images", "couleurs" et "infos". Dans "images" l'administrateur a la possibilité de modifier le logo présent dans le menu en haut de page avec les boutons "modifier le logo", pour ce faire il faut d'abord choisir un fichier grâce au bouton correspondant. En choisissant l'image le visuel est mis à jour sur la page de l'administrateur mais n'est pas encore effectif pour tous les visiteurs, pour enregistrer la modification il faut cliquer sur le bouton "valider". De la même manière l'administrateur peut modifier la bannière de l'accueil et la favicon (icône du site dans l'onglet).

Modifier votre site

IMAGES COULEURS INFOS

Modifier le logo

Choisir un fichier Aucun fichier choisi

Valider

Modifier l'icone de l'onglet

Choisir un fichier Aucun fichier choisi

Valider

L'image doit être de préférence en 32x32

Modifier la bannière

Choisir un fichier Aucun fichier choisi

Valider

Figure 11 : Menu de personnalisation

Dans “couleur” l’administrateur peut modifier les deux couleurs du site, la couleur principale et la couleur secondaire. En cliquant sur “couleur principale” ou “couleur secondaire” l’administrateur a accès à un menu de sélection de couleur. Les changements de couleur se font en temps réel sur l’écran. Pour enregistrer la modification il faut cliquer sur “OK” et pour annuler la modification il faut cliquer en dehors du menu de sélection de couleurs.

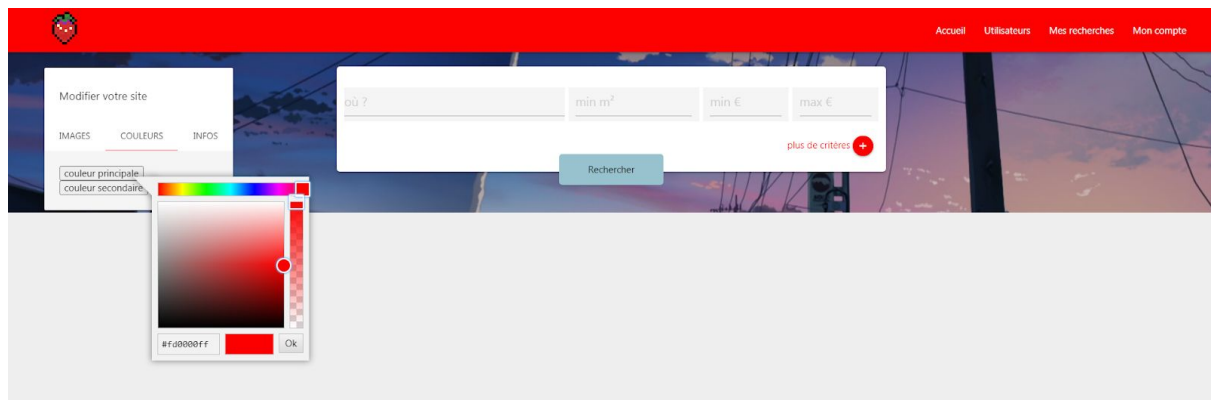


Figure 12 : Modification des couleurs

Dans le menu “infos” l’administrateur a la possibilité de modifier les informations du site, comme l’adresse e-mail et le numéro de téléphone de l’agence. Pour ce faire il faut renseigner les champs correspondant puis appuyer sur entrée. De la même manière l’administrateur peut modifier les liens des profils de l’agence sur les réseaux sociaux, mais pour ces derniers en cochant ou décochant les cases “afficher” l’administrateur peut décider de la présence ou non de ces liens dans le

footer. Si tous les réseaux sociaux sont désactivés la section “Retrouvez nous sur” du footer est supprimée.

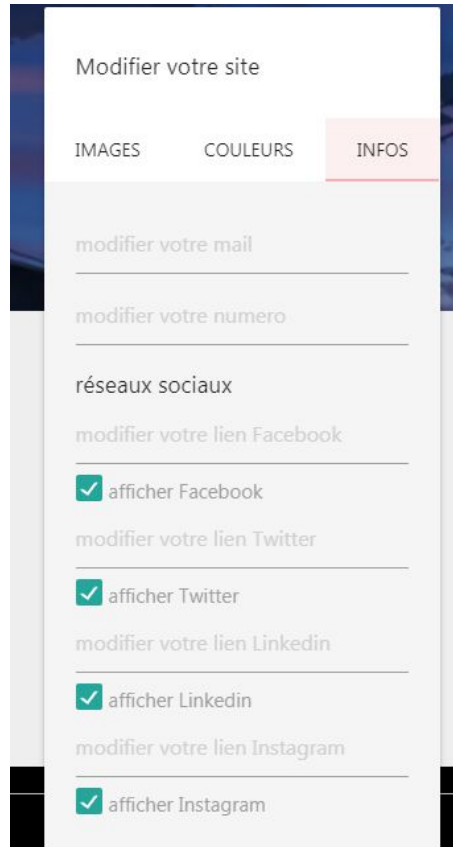


Figure 13 : Modifications des informations

Les comptes administrateurs peuvent accéder à une gestion des comptes en cliquant sur “Utilisateurs” dans le menu. Une fois sur la page des utilisateurs les administrateurs accèdent à la liste des comptes enregistrés, il y a la possibilité de rechercher un utilisateur en particulier en entrant son login (ou une partie) dans le champ “nom d'utilisateur” puis d'appuyer sur entrée.

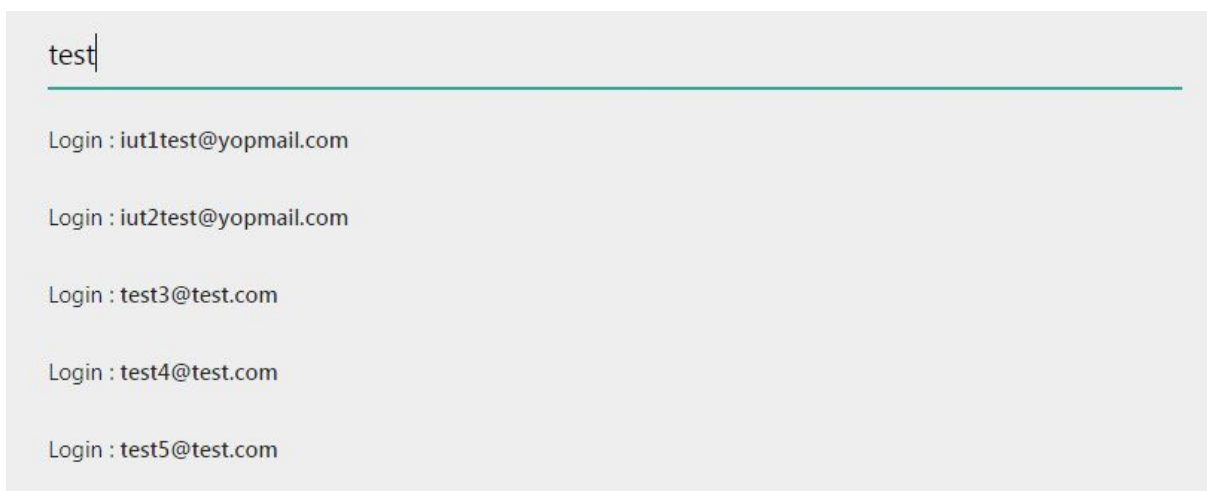


Figure 14 : Recherche d'un utilisateur

L'administrateur peut accéder aux paramètres d'un compte en cliquant sur son login. S'il s'agit d'un simple compte ou d'un commercial l'administrateur a la possibilité de modifier ses informations (nom, prénom, login, mot de passe) sans avoir à renseigner le mot de passe du compte, il peut aussi supprimer ce compte. Il peut aussi modifier le rôle, le passer de simple utilisateur à administrateur ou de commercial à simple utilisateur par exemple.



Figure 15 : Modification de rôle d'un utilisateur

Les administrateurs ne peuvent pas modifier leurs mails et mots de passe entre eux.

1.2.4. Fonctionnalités du super administrateur

Les super administrateurs ont accès aux fonctionnalités des autres rôles et peuvent en plus supprimer et modifier les rôles des comptes administrateurs.

1.2.5. Résumé des rôles et de leurs droits

Ci-dessous un tableau résumant les tous les différents types de rôles ainsi que leurs droits

rôle/action	rechercher des lots	sauvegarder la recherche	modifier son compte	supprimer son compte	accès informations confidentiel de lot	consulter les comptes enregistrés	modifier le compte des autres utilisateurs	personnaliser le site	Modifier le compte des admins
sans compte	oui	non	non	non	non	non	non	non	non
connecté	oui	oui	oui	oui	non	non	non	non	non
commercial	oui	oui	oui	oui	oui	non	non	non	non
admin	oui	oui	oui	non	non	oui	oui	oui	non
superadmin	oui	oui	oui	non	non	oui	oui	oui	oui

Tableau 1 : Tableau récapitulatif des droits des comptes

2. Organisation des fichiers

Dans cette partie nous allons revenir sur l'organisation des fichiers dans le projet.

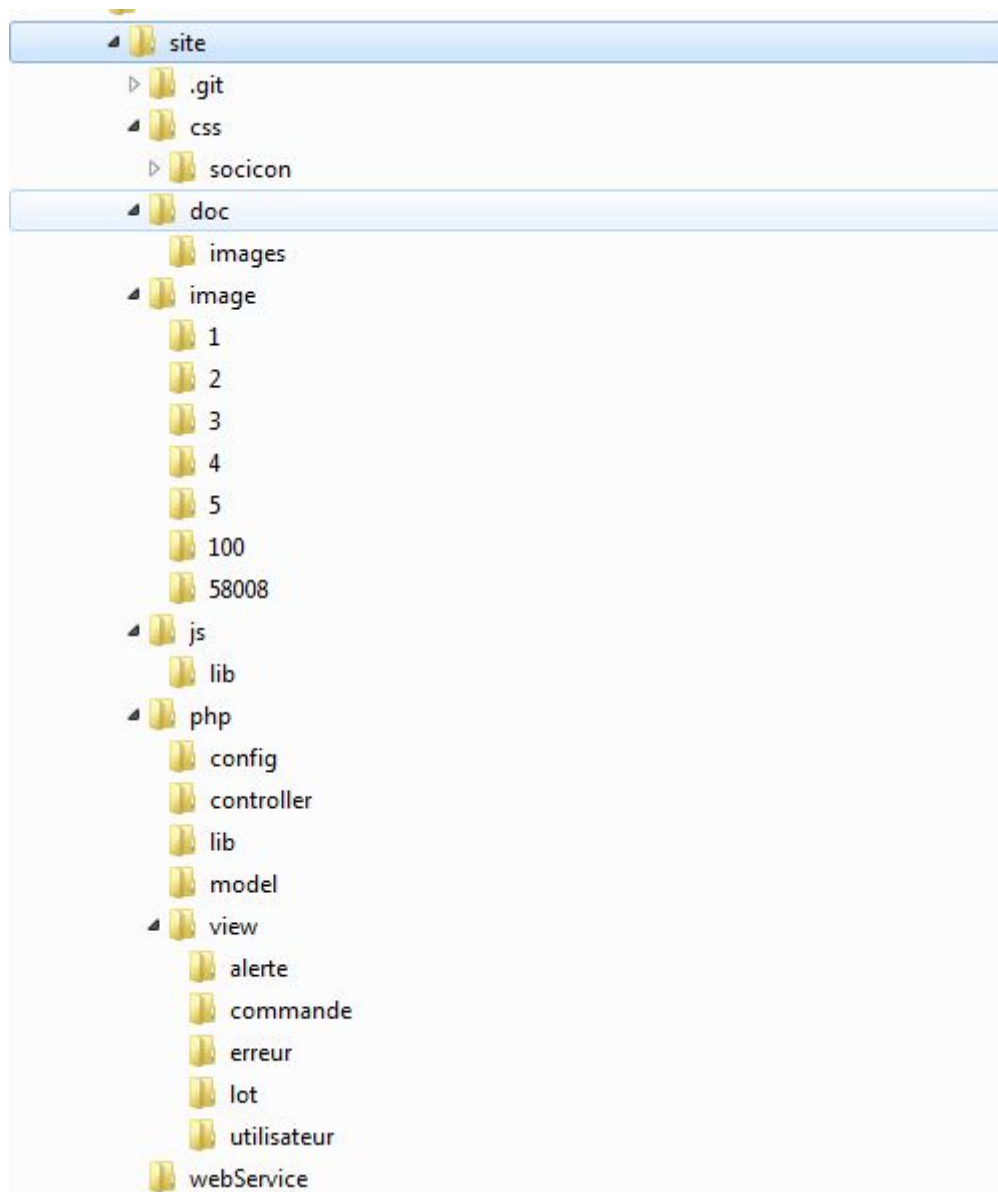


Figure 16 : Hiérarchie des dossiers

Dans le projet nous retrouvons plusieurs dossiers. D'abord, le dossier "css" qui contient les différents styles ainsi que le sous dossier "socicon" qui est une liste d'icônes utilisé dans le projet. Nous retrouvons ensuite le dossier "doc" dans lequel se trouve la documentation. Ensuite, nous avons le dossier "images" dans lequel nous retrouvons les différentes images du site (bannière logo). Les sous dossiers correspondent aux images des différents lot (le nom du dossier correspondant à l'id du lot). Puis se trouve le dossier "js" dans lequel sont contenus les fichiers javascript. Dans le dossier lib se trouve la librairie "Vanilla Picker". Après, nous avons le dossier php divisé en trois principaux sous dossiers "model", "view" et "controller". Dans ces dossiers nous retrouvons les fichiers php propre au design pattern Modèle-vue-contrôleur (MVC). Dans le dossier php nous retrouvons en plus deux dossier, le dossier "config" contenant le fichier de configuration de la base de données et le dossier lib contenant les fichiers php ne s'intégrant pas au pattern MVC.

3. Rapport Technique

3.1. Back-end

Dans cette partie nous entrerons en détail dans la structure back-end du projet. Elle est divisée en plusieurs parties, tout d'abord nous verrons l'installation nécessaire pour la mise en place du projet, en deuxième nous verrons le fonctionnement global du MVC, puis nous listerons les technologies utilisées, ensuite nous présenterons les différents diagrammes du projet avant de rentrer plus en détail dans le fonctionnement des différentes classes dans la partie suivante, et pour finir nous nous pencherons sur le webservice et son utilisation.

3.1.1. Installation

Pour déployer le site il faut faire quelques modifications, d'abord il faut modifier la balise base dans "view.php".

```
<base href='http://localhost/site/'>
```

Figure 17 : Balise base

Le href doit correspondre au lien sur lequel est accessible le site. Ensuite, il faut vérifier que la variable debug de "errorHandler.php" soit initialisée à false, si elle ne l'est pas, les warnings et les erreurs seront affichés.

Ensuite, dans le fichier "Conf.php" nous devons initialiser les valeurs du tableau database de telle sorte que la connexion à notre base de données peut se faire.

```
static private $databases = array(  
    'hostname' => 'maBaseDeDonnée.fr',  
    'database' => 'maBd',  
    'login' => 'MonLogin',  
    'password' => 'monMdp',  
);
```

Figure 18 : Configuration connexion à la base de données

La création du schéma de la base de données se fait avec le fichier "schemaBDD.sql" dans le dossier "doc" permettant la création des tables et des contraintes.

Certaines modifications doivent être faites dans le fichier "php.ini" :

- supprimer le commentaire de la ligne "extension=php_exif.dll"
- supprimer le commentaire de la ligne "extension=php_soap.dll"
- vérifier que "date.timezone=" ne soit pas commenté et égal à " 'Europe/Paris' "

3.1.2. Fonctionnement global MVC

Le site est développé selon le design pattern Modèle-vue-contrôleur (MVC) schématisé ci- dessous en diagrammes.

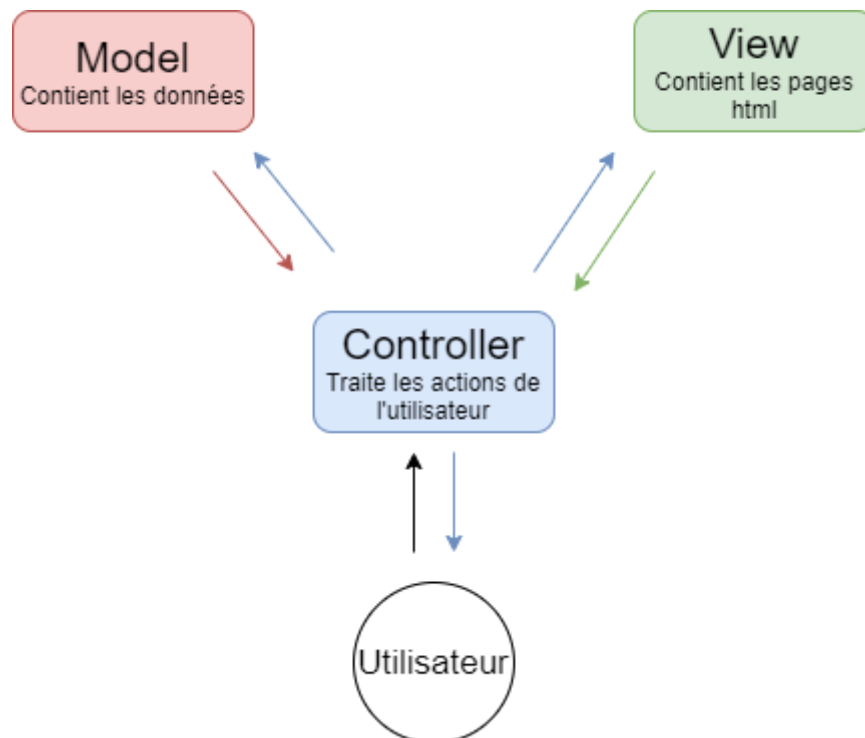


Figure 19 : Schéma MVC

Et ci-dessous en diagramme de séquence

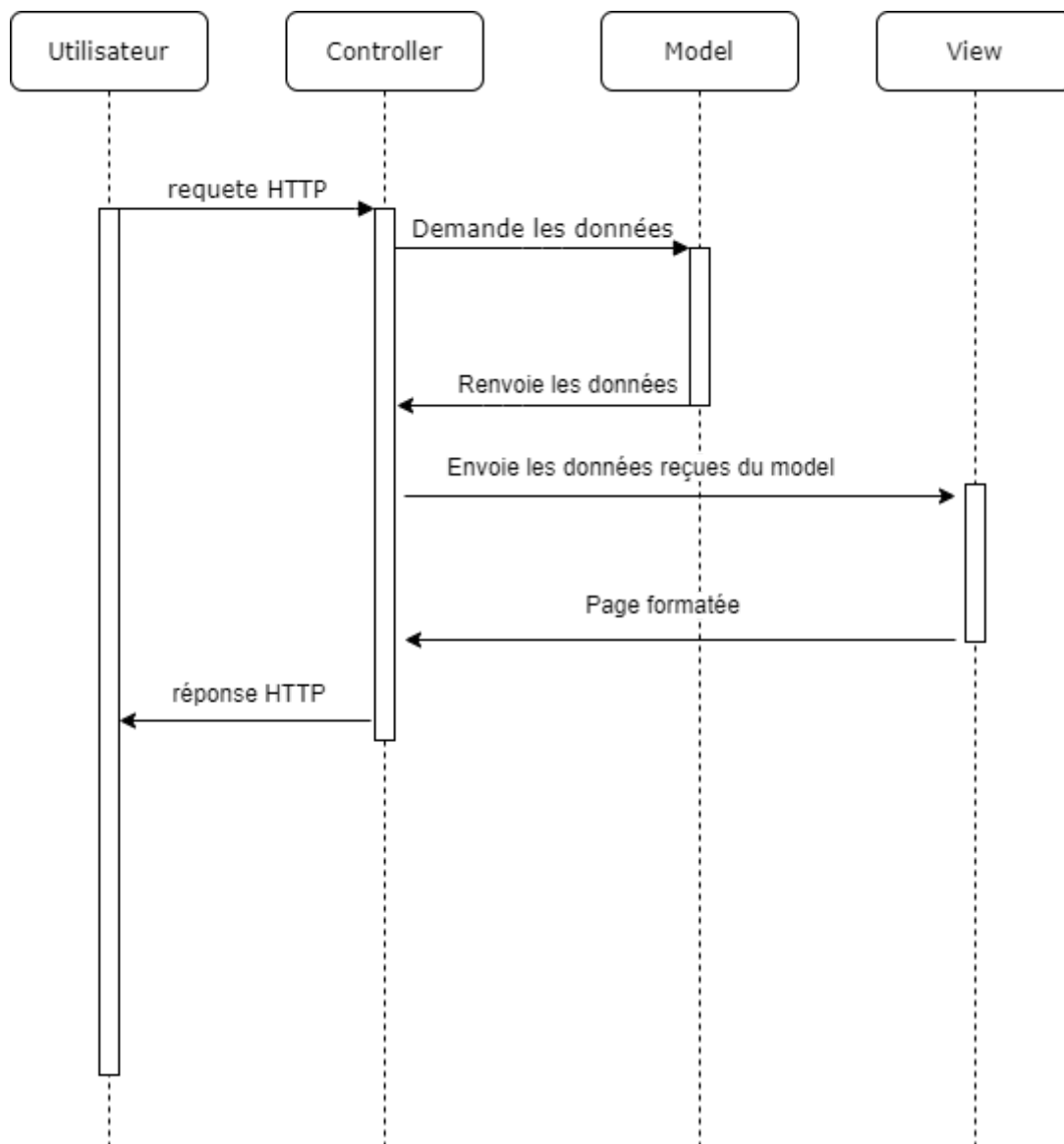


Figure 20 : Diagramme de séquence MVC

3.1.3. Technologies utilisées

Le site a été développé en PHP (version 5.6). Il utilise une base de donnée mySql.

Pour le lien avec le logiciel Priam un webservice a été développé avec l'extension php soap permettant de remplir la base de données ainsi que les fichiers du site depuis le logiciel. Ce même webservice permet aussi au logiciel de recevoir des données. Nous rentrerons dans les détails du webservice dans la partie lui étant dédiée.

3.1.4. Diagrammes de classes

Comme le site suit le design pattern MVC nous pouvons diviser nos diagrammes de classes en plusieurs diagrammes distincts.

D'abord, le diagramme de classes des model

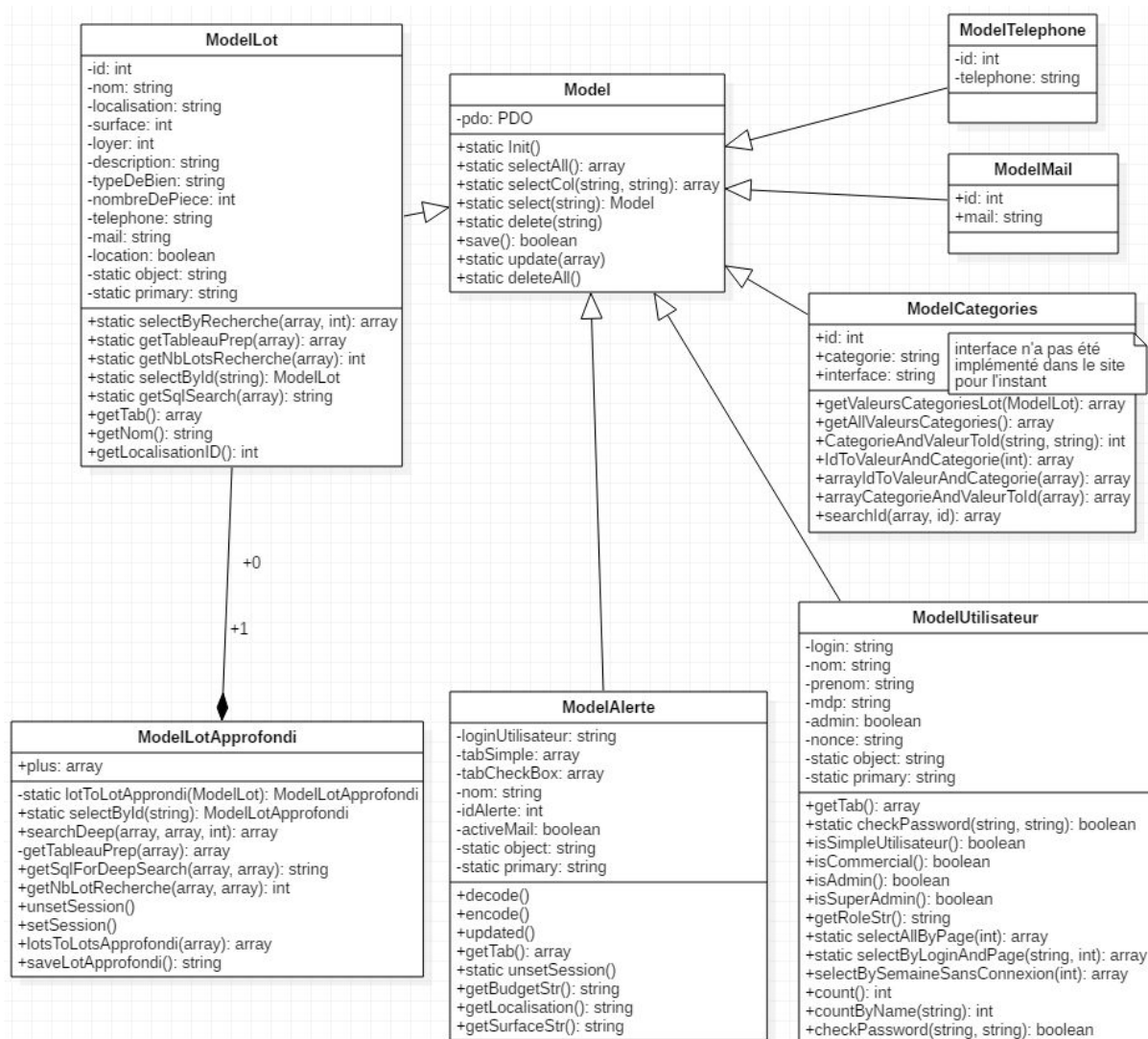


Figure 21 : Diagramme de classes des Model

Nous avons aussi le diagramme de classes des controller.

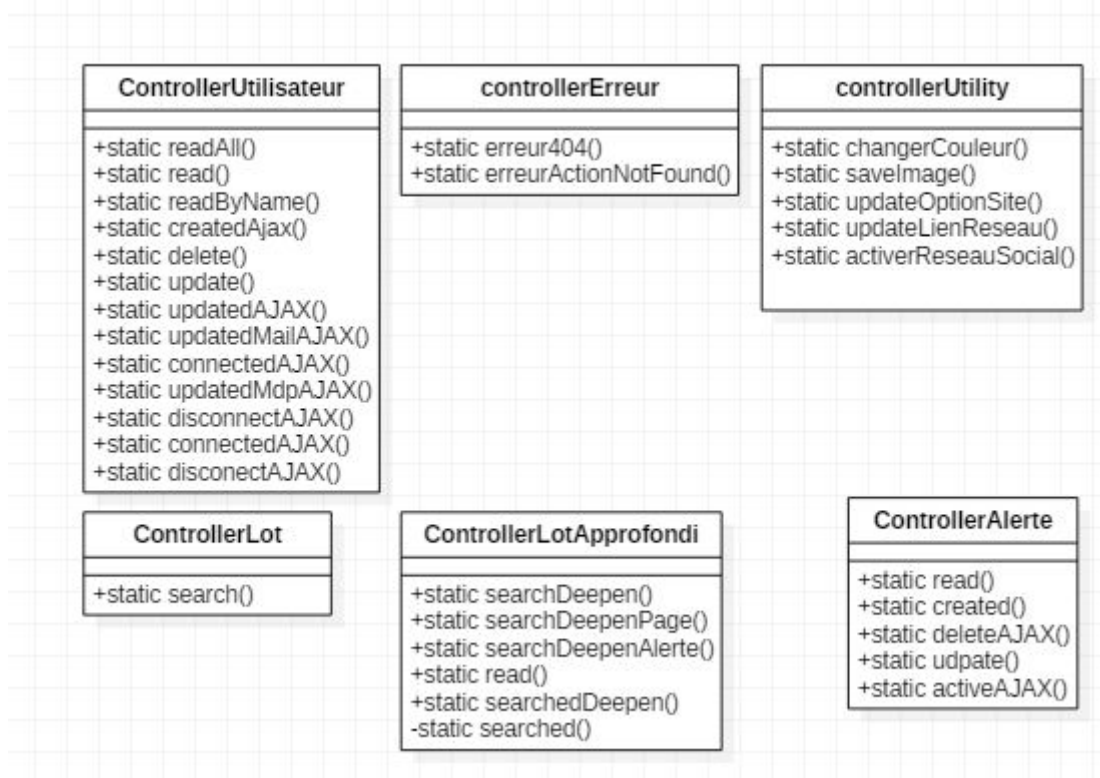


Figure 22 : Diagramme de classe des Controller

Le diagramme de classe des classes lib :

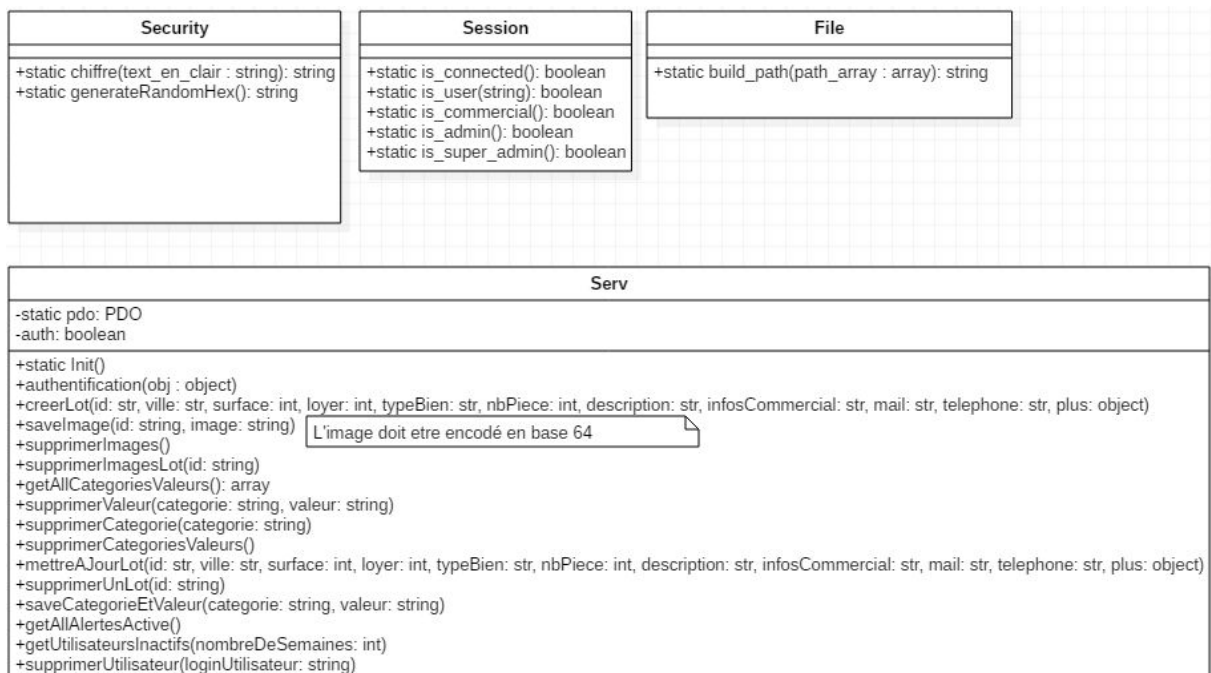


Figure 23 : Diagramme des classes lib

Le diagramme de classes correspondant à la base de données :

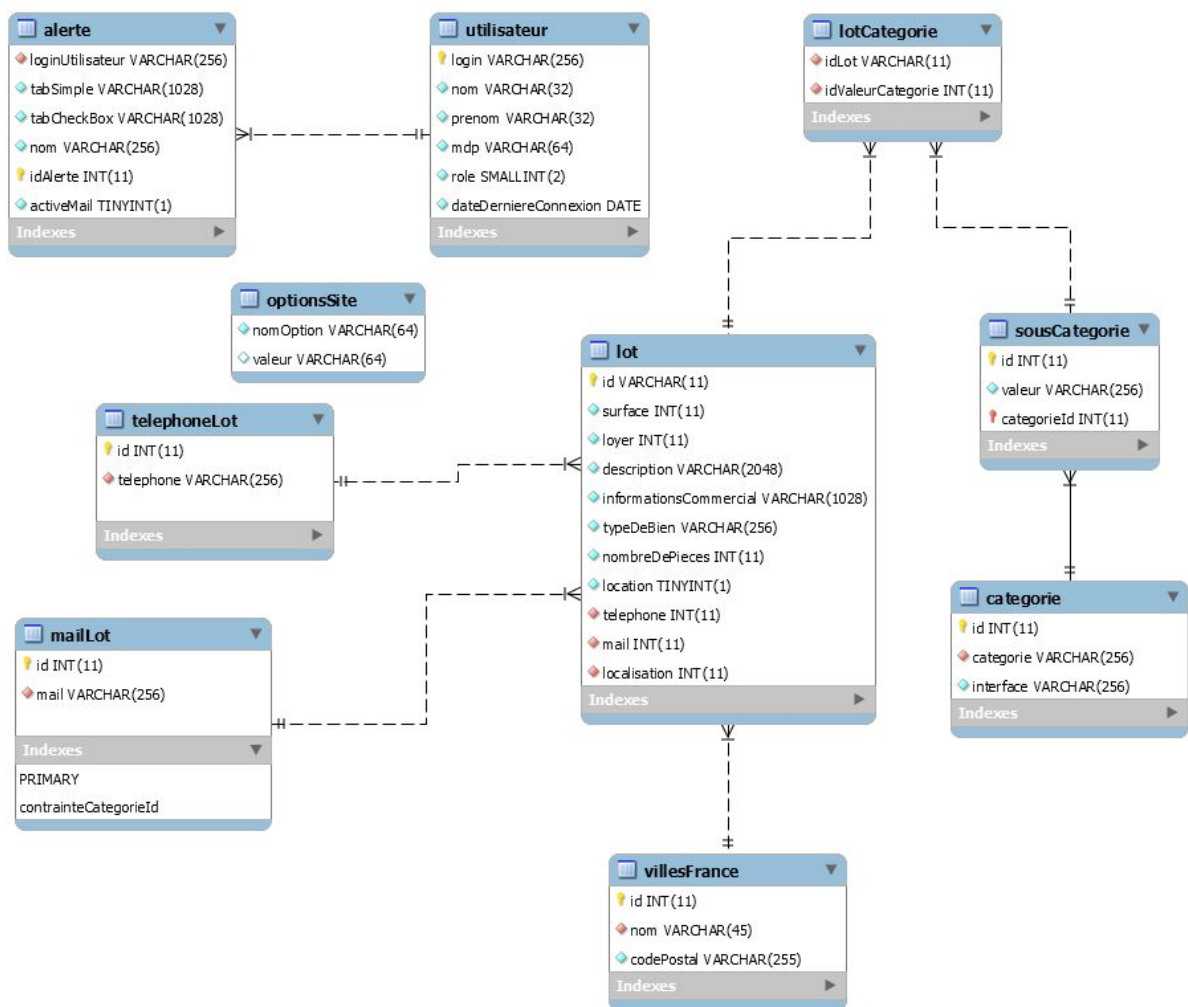


Figure 24 : Diagramme de classe de la base de données

3.1.5. Fonctionnement

Dans cette partie nous allons revenir sur certains fonctionnements du code php importants.

Le premier point sur lequel nous allons nous pencher est le routage, il fonctionne avec deux paramètres GET action et controller, si les deux sont initialisés alors est appelée (si elle existe) la fonction controller::action. Pour faire plus propre avec une règle de réécriture ces deux paramètres sont présents dans l'url comme ceci "index/controller/action/".

Maintenant nous allons revenir sur la manière dont nous stockons les alertes dans la base de données et donc comment cela se répercute sur le ModelAlerte.

alerte	
loginUtilisateur	VARCHAR(256)
tabSimple	VARCHAR(1028)
tabCheckBox	VARCHAR(1028)
nom	VARCHAR(256)
idAlerte	INT(11)
activeMail	TINYINT(1)
Indexes	

Figure 25 : Table alerte

Comme nous pouvons le voir les alertes sont stockées avec un login, deux tableaux, un nom, un id et un flag. Les critères à proprement parlé sont stockés dans les deux champs tabSimple et tabCheckBox. Nous les stockons sous forme de Json. Donc naturellement lorsque nous appelons la méthode select avec le ModelAlerte l'objet renvoyé a dans ses variables tabSimple et tabCheckBox des Json qu'on ne peut pas manipuler tel quel en php. C'est pour cela que ModelAlerte est équipé des fonctions encode et decode qui permettent d'encoder et de décoder les tableaux de l'objet. Lorsqu'on sélectionne un objet il faut appeler la fonction decode pour pouvoir le manipuler et il faut aussi penser à appeler la fonction encode avant d'enregistrer une alerte dans la base de données.

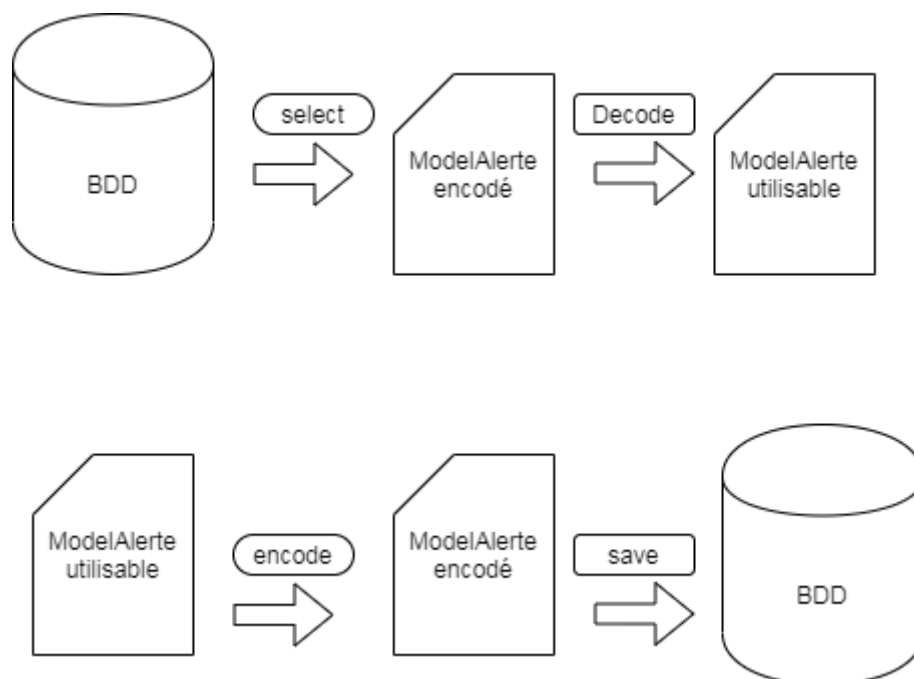


Figure 26 : Encodage et décodage ModelAlerte

Maintenant nous allons expliquer pourquoi dans la fonction searched du ControllerLotApprofondi nous enregistrons les critères dans la session.

```
private static function searched($dataPost, $dataCheckBox){
    $_SESSION["dataFirst"]=$dataPost;
    $_SESSION["dataCheckBox"]=$dataCheckBox;
```

Figure 27 : Session dans la fonction searched

En fait nous utilisons les variables enregistrées en session au moment d'enregistrer une nouvelle alerte dans le ControllerAlerte.

```
static function created(){
(Session::is_connected() && !empty($_SESSION["dataFirst"])){
    $tabPlus=ModelCategories::arrayIdToValeurAndCategorie($_SESSION["dataCheckBox"]);
    $alerte=new ModelAlerte(null,$_SESSION["login"],$_SESSION["dataFirst"],$tabPlus,"I
    $alerte->encode();
    $alerte->save();
    ModelAlerte::unsetSession();
```

Figure 28 : Session dans la fonction created

Maintenant nous allons rentrer en détail dans la gestion des erreurs. Pour gérer les erreurs les plus courante un Controller spécifique a été créé, le ControllerErreur.

```
<?php

class ControllerErreur{

    public static function erreur404() {
        $controller='erreur'; $view='erreur404'; $pagetitle='Erreur 404';
        require File::build_path(array("view", "view.php")); //"redirige"
    }

    public static function erreurActionNotFound(){
        $controller='erreur'; $view='erreurAction'; $pagetitle='Erreur';
        require File::build_path(array("view", "view.php")); //"redirige"
    }

}
```

Figure 29 : ControllerErreur

Il comporte deux fonctions, erreur404 (aussi utilisé pour les erreurs 403) et erreurActionNotFound appelé lorsque le controller dans la variable ou sous action ne correspondent pas à ceux existants. Ces Deux actions redirige vers des pages d'erreurs. De plus, une classe "errorHandler" a été créée dans le but d'éviter l'affichage d'erreurs et de warnings pour les visiteurs.

Maintenant nous allons expliquer comment sont modélisés le ModelLot et le ModelLotApprofondi.

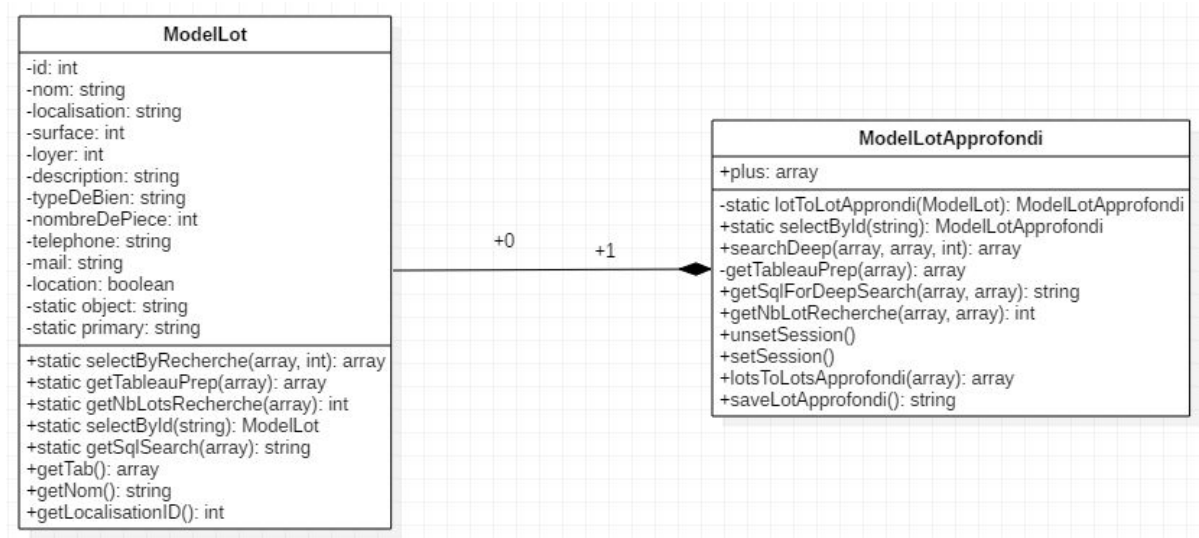


Figure 30 : Diagramme ModelLot

Comme nous pouvons le voir sur ce diagramme il y a une composition entre modelLot et ModelLotApprofondi. Le constructeur de ModelLotApprofondi peut ne prendre qu'un argument, un ModelLot, dans ce cas la variable plus essaye de s'initialiser en cherchant dans la base de données les critères enregistrés pour ce lot. Dans le cas où le lot n'est pas enregistré dans la base de données et que nous voulons que l'attribut plus soit initialisé il faut ajouter un second paramètre, un tableau regroupant les critères du lot. Le tableau doit être sous la forme : [catégorie1=>[valeur1, valeur2], categorie2=>[valeur1]]

```

array(5) { ["Type(s) de bien"]=> array(1) { [0]=> string(6) "Maison" } ["Nombre de pièce(s)"]=> array(1) { [0]=> string(9) "6 et plus" }
["Type(s) de pièces"]=> array(1) { [0]=> string(13) "salle de bain" } ["Commodité(s)"]=> array(1) { [0]=> string(6) "alarme" }
["Orientation(s)"]=> array(1) { [0]=> string(3) "est" } }
  
```

Figure 31 : Exemple de mise en forme de tableau de critères

Nous pouvons aussi voir dans le diagramme que ModelLot a une fonction selectById. En effet cette fonction est nécessaire car la fonction select contenu dans la classe Model ne retournera pas le lot attendue. En effet comme il y a plusieurs clés étrangères dans la table lot la fonction select retournera les id et non pas les valeurs des attributs étrangers. Pour les mêmes raisons une fonction saveLotApprofondi est contenue dans ModelLotApprofondi.

Les vérifications de droits se font grâce aux fonctions contenues dans la classe Session. Elles sont sous la forme "is_typeDeCompte" et renvoie un booléen comme nous pouvons le voir sur le diagramme des classes lib.

Pour une meilleure portabilité du projet les require ne se font pas à partir du nom du fichier lui-même mais passe par la fonction build_path de la classe File.

```
require_once File::build_path(array("model", "ModelLot.php"));
```

Figure 32 : Exemple de build_path

Cette fonction a comme paramètre un tableau de string qui représente le chemin du fichier à partir du dossier php. Cela permet au site d’être hébergé sur des systèmes d'exploitation différent sans avoir à modifier les chemins des fichiers.

Pour générer les vues nous faisons comme cela :

```
$controller='lot'; $view='recherche'; $pagetitle='Recherche de biens';  
require File::build_path(array("view", "view.php")); // "redirige" vers
```

Figure 33 : Exemple de génération de vue

Ensuite dans view.php vous avons un autre require qui ouvre le fichier contenu dans view/"la valeur de la variable controller"/"la valeur de la variable view". De cette manière le header ainsi que le footer sont présents sur toutes les pages sans que nous avons à les écrire dans tous les fichiers.

3.1.6. Utilisation du webservice

Le webservice utilise la librairie soap de PHP, il est donc nécessaire de l'utiliser avec de l'xml.

Nous retrouvons dans le fichier "webservice/websericeCall.xml" la liste des appels xml à utiliser pour appeler les fonctions du webservice. L'utilisation du webservice se fait sur le fichier "php/lib/soap.php".

Il est cependant important de noter qu'il est nécessaire d'avoir une entête dans le xml qui fait appel à la fonction authentification. En effet pour sécuriser le webservice un système d'authentification a été mis en place. Qui fonctionne comme ceci :

Lors de l'appel xml la fonction authentification est exécutée avec un paramètre login et un paramètre mot de passe. Si les valeurs de ces paramètres ne correspondent pas à ceux prévus alors les fonctions appelées ensuite ne feront rien et retourneront "non connecté". De base le login est "test" et le mot de passe "toast" mais ils sont modifiables dans la fonction authentification de "lib/serv.php". S'ils sont modifiés il faut faire attention de mettre les nouveaux dans l'entête xml.

Ci-dessous la liste des fonctions disponibles dans le webservice :

Fonction	Action
creerLot	créé un lot
savelImage	enregistre une image pour un lot
supprimerImages	supprime toutes les images des lots
supprimerImagesLot	supprime toutes les images d'un lot
getAllCategoriesValeurs	renvoie la liste de toutes les catégories et de leurs valeurs
supprimerValeur	supprime une valeur
supprimerCategorie	supprime toute une categorie
supprimerCategoriesValeurs	supprime toutes les catégories
mettreAJourLot	met à jour un lot
supprimerLots	supprime tout les lots
supprimerUnLot	supprime un lot
saveCategorieEtValeur	ajoute une valeur (si la catégorie n'existe pas elle est créée)
getAllAlertesActive	renvoie toutes les alertes actives
getUtilisateursInactifs	renvoie tout les utilisateur inactifs pendant x semaines
supprimerUtilisateur	supprime un utilisateur

Tableau 2 : récapitulatif des fonctions du webservice

Pour ajouter une fonction au webservice il suffit d'en écrire une nouvelle dans la classe Serv. Néanmoins il faut veiller à débiter la fonction par la vérification de l'authentification.

```
public function saveImage($id,$image){  
    if(!$this->auth){  
        return "pas connecté";  
    }  
}
```

Figure 34 : Vérification authentification

3.2. Front-end

3.2.1. Technologies utilisées

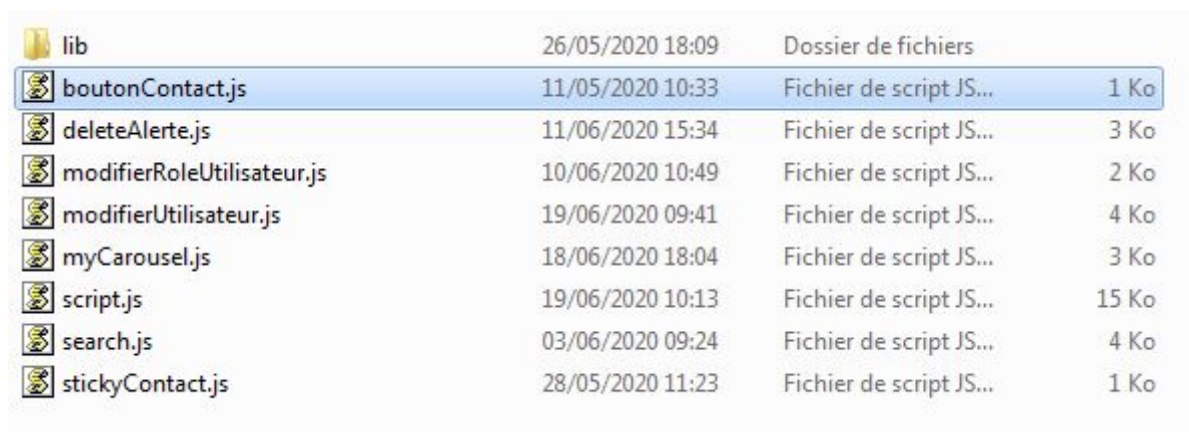
Pour le front end nous avons utilisée de l'HTML/CSS/JS.

Nous avons utilisé le framework Materialize (<https://materializecss.com/>) pour les outils qu'il offre pour le design. Nous avons aussi utilisé la librairie "Vanilla-Picker" (<https://vanilla-picker.js.org/>) pour les color pickers.

3.2.2. Javascript

Dans cette partie décrivons les parties les plus importantes du javascript.

Pour des soucis de réduction de temps de chargement de page la plupart des scripts ont été fusionnés en un seul fichier "script.js". Les script ne pouvant être présent que sur quelques pages sont contenus dans des fichiers individuels.



lib	26/05/2020 18:09	Dossier de fichiers	
boutonContact.js	11/05/2020 10:33	Fichier de script JS...	1 Ko
deleteAlerte.js	11/06/2020 15:34	Fichier de script JS...	3 Ko
modifierRoleUtilisateur.js	10/06/2020 10:49	Fichier de script JS...	2 Ko
modifierUtilisateur.js	19/06/2020 09:41	Fichier de script JS...	4 Ko
myCarousel.js	18/06/2020 18:04	Fichier de script JS...	3 Ko
script.js	19/06/2020 10:13	Fichier de script JS...	15 Ko
search.js	03/06/2020 09:24	Fichier de script JS...	4 Ko
stickyContact.js	28/05/2020 11:23	Fichier de script JS...	1 Ko

Figure 35 : Fichiers javascript

La fonction la plus importante de script.js est la fonction "requeteAJAX" elle prend deux paramètres : url et callback. L'url est le fichier sur lequel la requête ajax va être fait (il peut être donné en relatif). Comme la méthode utilisée pour passer les paramètres est la méthode get les données à envoyées doivent être contenues dans l'url. Le paramètre callback est la fonction qui va être exécutée à la réponse de la requête, cette fonction doit avoir un paramètre xhr. Le paramètre va permettre de récupérer la réponse de la requête en utilisant xhr.responseText.

exemple d'utilisation de la fonction :

```
requeteAJAX("index/utilisateur/connectedAjax/?login="+mail+"&mdp="+mdp,callbackIdentifiantCheck);
```

Figure 36 : Appel de la requeteAJAX

```
function callbackIdentifiantCheck(xhr){  
    console.log(xhr.responseText);  
    if(xhr.responseText=="true".valueOf()){  
        document.location.reload(true);  
    }else{  
        notification("mauvais identifiant ou mot de passe");  
    }  
}
```

Figure 37 : Callback requête ajax

