




26/08/2021



Projet de création d'un site de petites annonces immobilières

Dossier projet



Abdoulaye CONDE
SABBAH-IMMO

1. Introduction	2
a. Présentation personnelle	2
b. Présentation du projet	2
1. Compétences couvertes par le projet	2
a. CP 5 : Créer une base de données	2, 3, 4, 5, 6
b. CP 6 : Développer des composants d'accès aux données	7
c. CP 7 : Développer la partie backend d'une application web ou web mobile	7
2. Cahier des charges de l'application	8
a. Publics visés	8
b. Fonctionnalités	8, 9
c. User stories du MVP	10
d. Arborescence	11
e. Organisation	11
3. Technologies utilisées	12
a. Frontend	12
b. Backend	12
c. Outil	12
4. Conception	13
a. MCD, MLD, dictionnaire de données	13
b. La sécurité	13
c. Le pattern MVC	14
5. Développement	15
a. Les bundles	15
b. Les entités	16
c. Les controllers	16
d. Les models	17
e. Les views (et le frontend)	18
6. Tests	18
a. Jeu d'essai	18,19,20
b. Test fonctionnel d'une fonctionnalité représentative	21
7. Veille technologique	22
8. Conclusion	22



1. Introduction

a. Présentation personnelle

- Je m'appelle MR CONDE ABDOULAYE
- Titulaire d'une maîtrise en Informatique à l'ISTV (Institut Supérieur des Sciences et Technologies de Valenciennes) en 2010.
- Deux ans Expériences en Informatique
- Trois ans Chez DPD France au sein de l'entité plan et transport
- La formation DEV-PHP -POEC, est un tremplin d'acquérir des nouvelles compétences.
- Je souhaite suivre une formation en alternance dans le domaine des systèmes d'informations

b. Présentation du projet

Je pars du constat que l'univers du web propose des sites annonces, c'est pourquoi je souhaite mettre en place un site de petites annonces immobilières dédié aux particuliers.

L'objectif de cette plateforme est de permettre à chaque particulier qui en aurait le besoin, de déposer des annonces, en quelques clics. Il me fallait donc une interface ergonomique mais également visuellement accueillante pour permettre à n'importe qui de s'y retrouver rapidement.

Les différents utilisateurs du site sont : Le visiteur (utilisateur non connecté), l'utilisateur connecté, l'administrateur du site.

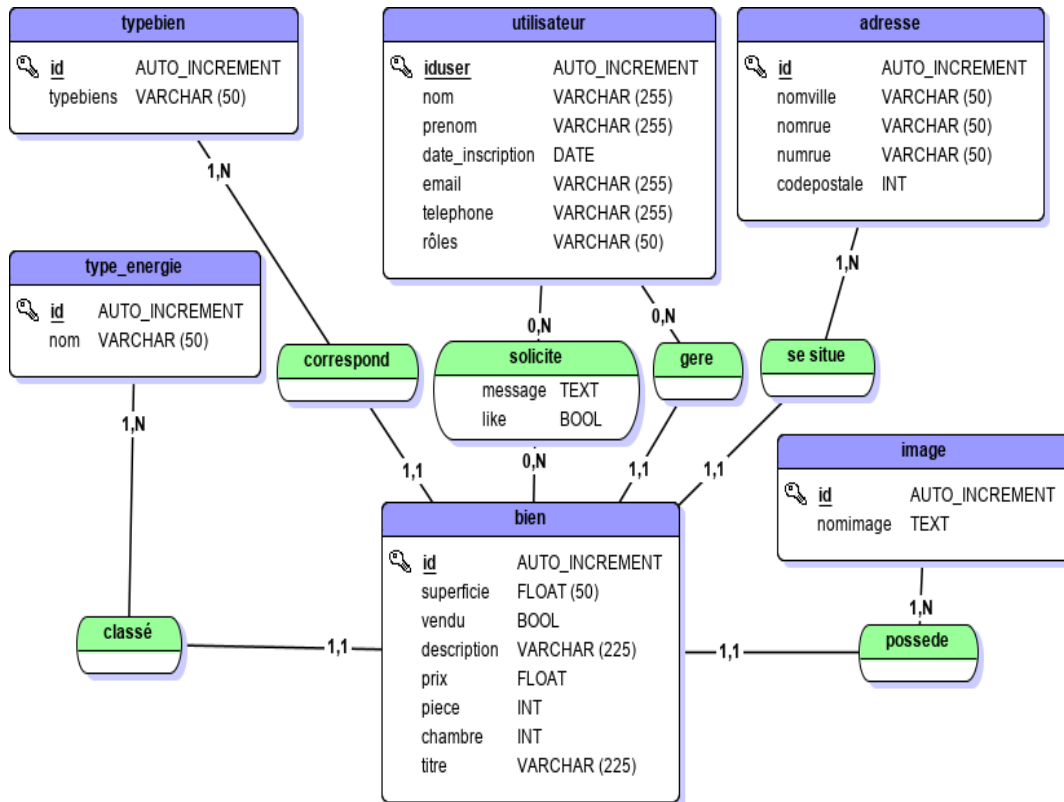
2. Compétences couvertes par le projet

a. CP 5 : Créer une base de données

Les acteurs qui doivent exploiter le site sont définis par des entités qui permettent de décrire des actions à effectuer qui seront stockées en base de données dans des tables, ce processus doit respecter un formalisme de modélisation et de conception base sur la méthode merise.

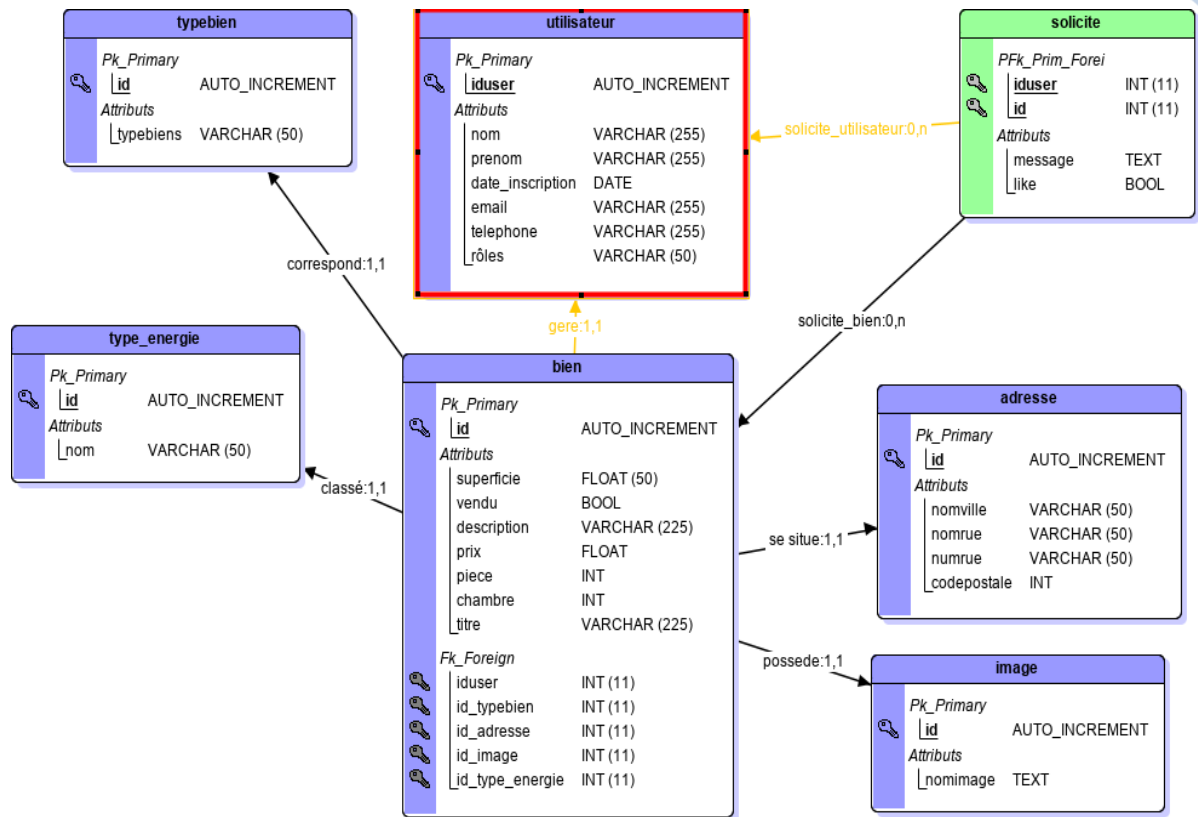
- Le MCD : Model Conceptuel de Données.





- Le MLD : Le model logique de données





- Le dictionnaire de données :

Nom	Code	Type	Taille	Commentaire
Roleid	ROLEID	AUTO_INCREMENT		Identifiant du rôle
iduser	IDUSER	AUTO_INCREMENT		Identifiant utilisateur
Nom	NOM	VARCHAR	50	
prenom	PRENOM	VARCHAR	50	
Email	EMAIL	VARCHAR	50	
password	PASSWORD	VARCHAR	50	
Telephone	TELEPHONE	VARCHAR	50	
Id	ID	AUTO_INCREMENT		
adresse	ADRESSE	VARCHAR	50	
superficie	SUPERFICIE	FLOAT	50	
Loyer	LOYER	FLOAT	50	
typebien	TYPEBIEN	VARCHAR	50	Type de bien



nomquartier	NOMQUARTIER	VARCHAR	50	Nom du quartier
nomrue	NOMRUE	VARCHAR	50	Nom de la rue
numrue	NUMRUE	VARCHAR	50	Numéro de la rue
nomville	NOMVILLE	VARCHAR	50	Nom de la ville
codepostale	CODEPOSTALE	VARCHAR	50	Code postale
message	MESSAGE	TEXT		
nomimage	NOMIMAGE	TEXT		Nom image
Like	LIKE	BOOL		
date_inscription	DATE_INSCRIPTION	DATE		Date d'inscription
username	USERNAME	VARCHAR	225	Le pseudo utilisateur
etatbien	ETATBIEN	BOOL	225	L'état du bien (vendu, disponible)
status	STATUS	BOOL		Activer ou désactiver
description	DESCRIPTION	TEXT		
prix	PRIX	FLOAT		
Piece	PIECE	INT		
chambre	CHAMBRE	INT		
titre	TITRE	VARCHAR	50	
nomrôles	NOMRÔLES	VARCHAR	50	

- Le passage à Doctrine avec le maker :

Doctrine est un **ORM** (couche d'abstraction à la base de données) pour PHP. Il s'agit d'un logiciel libre sous licence GNU LGPL.

Doctrine est l'ORM par défaut du framework Symfony (depuis la version 1.3 de ce framework). Cependant son utilisation dans le cadre d'un projet développé avec Symfony est optionnelle.

Un mapping objet-relationnel (en anglais object-relational mapping ou ORM) est un type de programme informatique qui se place en interface entre un programme applicatif et une base de données relationnelle pour simuler une base de données orientée objet. Ce programme définit des correspondances entre les schémas de la base de données et les classes du programme applicatif. On



pourrait le désigner par-là, « comme une couche d'abstraction entre le model objet et model relationnel ».

La commande make entity de Symfony permet de créer une entity, dans le projet Symfony, l'entité prendra la signature de Class et sera annotée par la signature de l'ORM (ORMEntity).

- Quelques commandes utiles :

Création d'une entity :

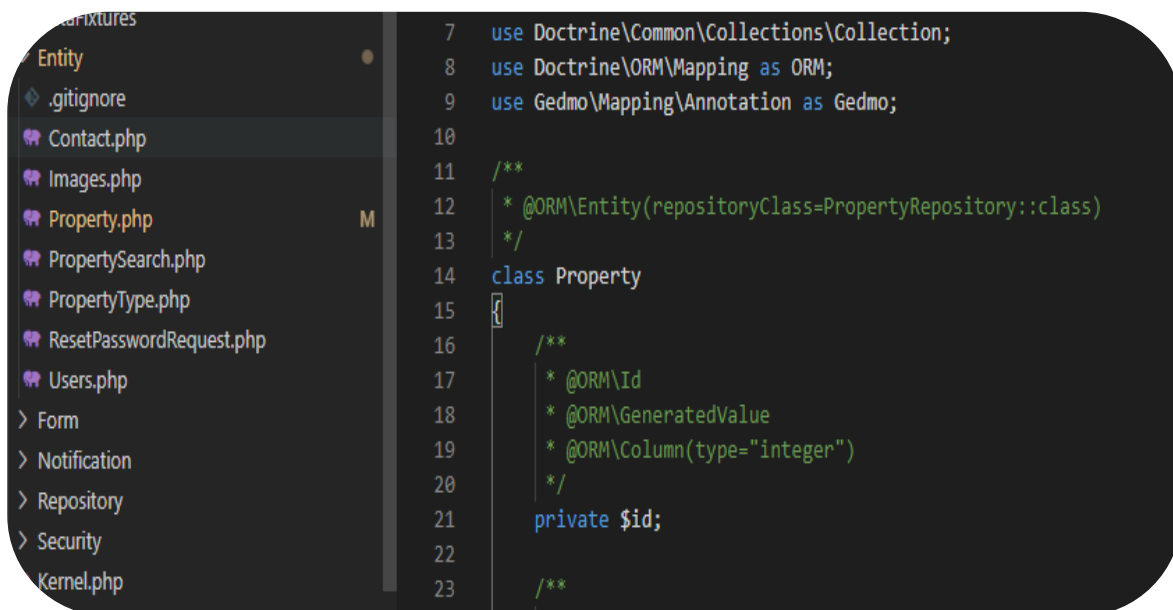
```
C:\wamp64\www\gestion-immo>php bin/console make : entity
Class name of the entity to create or update (e.g. GrumpyGnome):
> Property
Your entity already exists ! So let's add some new fields!
New property name (press <return> to stop adding fields):
```

Phase de migration :

```
updated: src/Entity/Property.php
Add another property? Enter the property name (or press <return> to stop adding fi
elds):
>
Success!
Next: When you're ready, create a migration with php bin/console make: migration
```

```
Next: Review the new migration "migrations/Version20210812151300.php"
Then: Run the migration with php bin/console doctrine:migrations:migrate
```

Le rendu du make entité dans VSCODE (Editeur de code) :



```

7  use Doctrine\Common\Collections\Collection;
8  use Doctrine\ORM\Mapping as ORM;
9  use Gedmo\Mapping\Annotation as Gedmo;
10
11 /**
12  * @ORM\Entity(repositoryClass=PropertyRepository::class)
13  */
14 class Property
15 {
16     /**
17      * @ORM\Id
18      * @ORM\GeneratedValue
19      * @ORM\Column(type="integer")
20      */
21     private $id;
22
23     /**

```



b. CP 6 : Développer des composants d'accès aux données

- Le rôle des entités :

Une entité, ce que l'ORM va manipuler et enregistrer dans la base de données, en d'autres termes c'est un objet avec des annotations (ORM\Entity).

La table des entités du projet sous Vscodé .

Noms	Nombre d'attribut	Nombre de clés étrangères
Property	21	2
Images	2	1
Property_Type	2	0
Users	5	0
Reset_password_request	6	1

- Les requêtes customs et les repositories :

Les requêtes customs permettent de répondre à un besoin spécifique qui n'est pas implémentable avec les requêtes génériques définies par Symfony.

Par défaut, le répertoire Repository contient tous les fichiers où sont définies les requêtes customs et génériques de Symfony. (Voir figure ci-dessous)

```

22     parent::__construct($registry, Property::class);
23 }
24 /**
25  * @return Query
26  */
27 //La fonction prend en paramètre la classe PropertySearch et
28 //son type de retour est une Query
29 public function findAllVisibleQuery(PropertySearch $search): Query
30 { // je vais sauvegarder le résultat de la requête de findVisibleQuery dans $query
31     $query = $this->findVisibleQuery();
32     // Si j'ai un getMaxPrice()
33     if ($search->getMaxPrice()) {
34         // Je rajoute à ma requête $query la clause andwhere
35         $query = $query
36             // Je veux que le prix de mon bien (p.price) soit inférieure ou égale
37             // à maxprice (prix maxi saisi dans le formulaire)
38             ->andWhere('p.price <= :maxprice')
39             // La méthode setParameter, maxprice aura pour valeur => $search->getMaxPrice()
40             ->setParameter('maxprice', $search->getMaxPrice());
41     }
42     if ($search->getMinSurface()) {
43         $query = $query
44             ->andWhere('p.area >= :minsurface')
45             ->setParameter('minsurface', $search->getMinSurface());
46     }
47     // Ici je retourne le résultat de la requête
48     return $query->getQuery();
49 }
50 private function findVisibleQuery(): QueryBuilder
51 {
52     return $this->createQueryBuilder('p')
53         ->andWhere('p.active = :active')
54         ->setParameter('active', true);

```

c. CP 7 : Développer la partie backend d'une application web ou web mobile

- La chaîne de traitement MVC dans Symfony :



- **Le Model** c'est la couche qui représente la source de données (SQL, xml,)
- **La Vue** c'est la couche de présentation de l'application (Interface utilisateur)
- **Le Controller** c'est la couche qui définit les directives entre le model et la vue.

Le workflow de connection sur le site : <https://aconde-immo.herokuapp.com/> selon le model MVC avec les méthodes Get et Post :

Lorsque je déclenche l'action de connection dans la vue via ce lien : `Connexion`, le nom de cette route `app_login` fait appel au route (Get /login) via le Controller (SecurityController), et ce controller nous renvoi sur la vue (security/login.html.twig) de connection, notre url sera : : <https://aconde-immo.herokuapp.com/login>.

Lorsque les informations de l'utilisateur sont validées dans le formulaire (**VUE**), ces informations sont envoyées au serveur via la méthode (Post /login) , le serveur fait appel au SecurityController (**Controller**) qui va se charger d'interroger le **model** pour vérifier la cohérence des informations provenant de la vue, dans le cas où les informations issues de la vue ne sont pas cohérentes à travers la méthode POST, le Controller renvoie un message d'erreur à la vue, de type identifiant invalide ou si les informations sont correctes, le Controller nous renvoie sur la page de gestion de compte ou autres pages défini dans le Controller.

- Le rôle du backend et les principes succins du frontend (Twig).

Le rôle du backend :

La partie backend de l'application est en charge du traitement du code s'exécutant côté **serveur**, Il définit la logique de l'application ainsi que la récupération des données depuis une base de données afin de les envoyer vers la partie frontend.

Les principes du frontend :

Le frontend consiste à de développer la partie visible d'une application web ou mobile. C'est à dire ce qui est affiché sur votre navigateur web ou téléphone mobile. Cela correspond généralement au HTML, CSS et Javascript, twig. C'est ce que l'on appelle le côté **client** en opposition au côté **serveur**.

3. Cahier des charges de l'application

a. Publics visés

Les différents types d'utilisateurs :

- Visiteur : utilisateur non identifié.
- Utilisateur : visiteur avec un compte lui donnant accès à des fonctionnalités spécifiques.
- Administrateur : utilisateur inscrit avec des droits supérieurs afin de gérer le backoffice du site.

b. Fonctionnalités

Les différentes fonctionnalités en fonction des types d'utilisateurs.

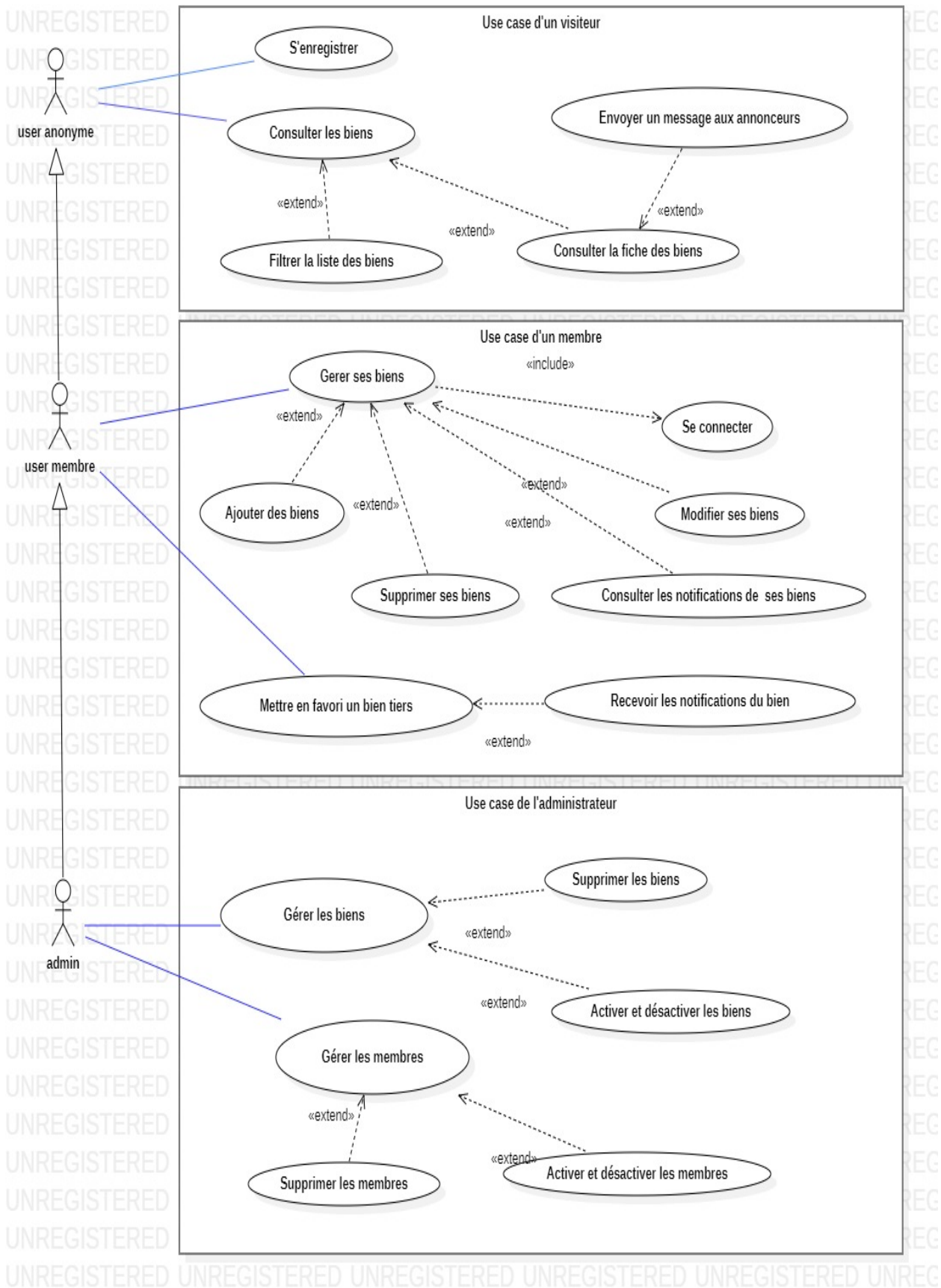


- Le visiteur (utilisateur non connecté) :
 - Visualiser la page d'accueil présentant les derniers biens publiés
 - Visualiser un bien publié en cliquant sur le lien du bien
 - Contacter l'annonceur d'un bien par mail ou téléphone
 - Le visiteur peut s'inscrire et s'authentifier pour accéder à plus de fonctionnalités.
- L'utilisateur connecté (en plus des fonctionnalités précédentes) :
 - Accéder à sa page de profil sur laquelle il retrouvera ses informations personnelles ainsi que les annonces qu'il a publiées.
 - Éditer, modifier ses informations (Username, password, description, etc.)
 - Créer une nouvelle annonce, le publier, le modifier et le supprimer.
 - Voter ou liker des annonces qu'il aime sur la page d'accueil.
 - Se déconnecter et réinitialiser son mot de passe
- L'administrateur :
 - Accède au backoffice pour modifier / supprimer un utilisateur, une annonce.
 - Active ou désactive une annonce.
 - Peut créer des types de biens et des types d'énergies qui sont associés à un bien.



c. User stories du MVP

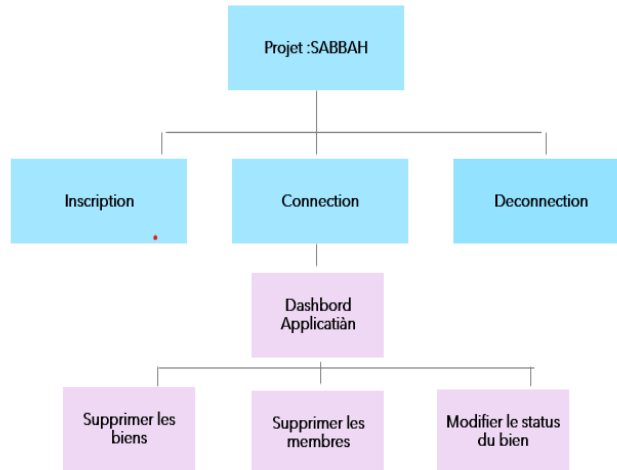
Présentation graphique des fonctionnalités en fonction des rôles.



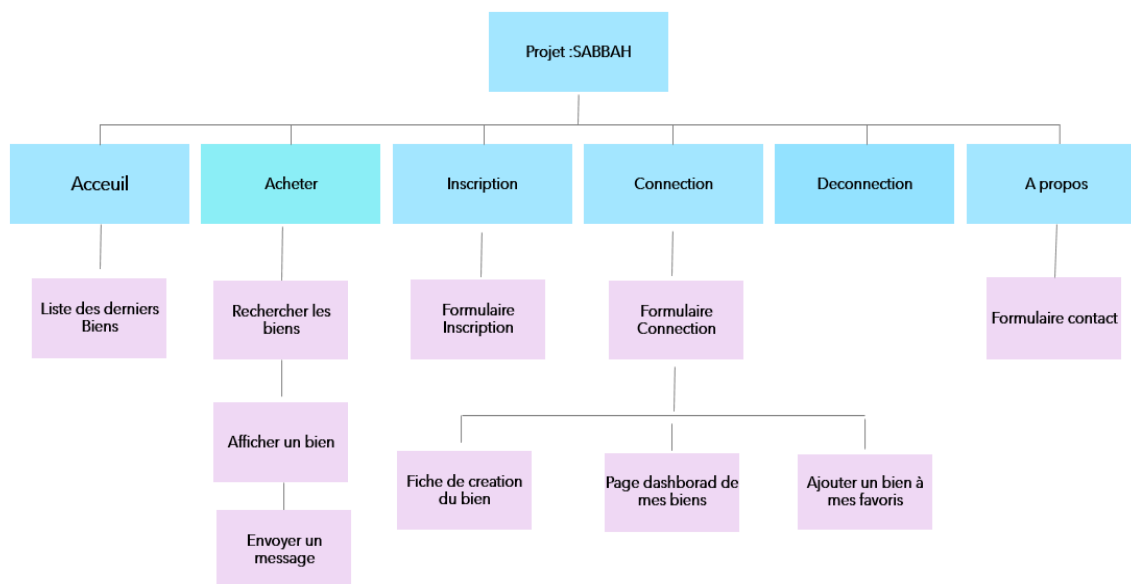
d. Arborescence

Schéma de navigation sur le site internet :

Back office



Le front office :



e. Organisation

Le diagramme de réalisation du projet.

RECUEIL DES INFORMATIONS	01/07/2021	7
DÉFINITION DES BESOINS	08/07/2021	4
RÉDACTION DU CAHIER DES CHARGES	12/07/2021	3
CONCEPTION	15/07/2021	7
CODAGE	22/07/2021	10
PHASE DE TEST	01/08/2021	1
MISE EN PRODUCTION	02/08/2021	1



Etape	Date début	Durée	Date de fin
Recueil des informations	01/07/2021	7	08/07/2021
Définition des besoins	08/07/2021	4	12/07/2021
Rédaction du cahier des charges	12/07/2021	3	15/07/2021
Conception	15/07/2021	7	22/07/2021
Codage	22/07/2021	10	01/08/2021
Phase de test	01/08/2021	1	02/08/2021
Mise en production	02/08/2021	1	03/08/2021

4. Technologies utilisées

a. Frontend

Présenter de façon concise :

- Le template retenu est : startbootstrap-shop-homepage-gh-pages du site <https://startbootstrap.com/>
- Les technologies utilisées par le template :
 - HTML5, CSS3, JS
 - Librairies JS
src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.bundle.min.js"
 - Frameworks CSS
href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.5.0/font/bootstrap-icons.css"

b. Backend

Présentation de l'architecture globale utilisée :

- Wamp : Apache 2.4, MySQL, PHP 7.4
- Symfony 5.3
- ORM Doctrine
- TWIG Moteur de template pour faire le lien avec le frontend

c. Outils

L'environnement de travail :

- Visual Studio Code et les extensions utilisées :
 - Twig Language 2 v0.9.2
 - PHP Tools for Visual Studio Code
 - Getter Setter and Constructor Generator for PHP v0.0.1
 - Git Project Manager v1.7.1
 - PHP Debug v1.17.0



- Github : <https://github.com/condexp/gestion-immo.git>
- Heroku : L'application est déployée chez Heroku et la base de données chez alwaysData, car le déploiement de la base de données MySQL chez Heroku est soumis à l'enregistrement des informations bancaires.
- Trello
- JMerise
- StarUML

5. Conception

a. MCD, MLD, dictionnaire de données

Le rôle des différents modèles.

Le MCD : Le modèle conceptuel des données (MCD) a pour but d'écrire de façon formelle les données qui seront utilisées dans le système d'information. Il s'agit donc d'une représentation des données, facilement compréhensible, permettant de décrire le système d'information à l'aide d'entités.

Le MLD : Le modèle logique des données consiste à décrire la structure de données utilisée sans faire référence à un langage de programmation. Il s'agit donc de préciser le type de données utilisées lors des traitements. Ainsi, le modèle logique est dépendant du type de base de données utilisé.

Le dictionnaire de données : Le dictionnaire des données est un document qui regroupe toutes les données que vous aurez à conserver dans votre base (et qui figureront donc dans le MCD). Pour chaque donnée, il indique toutes les propriétés de l'objet à modéliser et leurs types.

Par exemple, la représentation du dictionnaire de donnée d'un bien immobilier :

Bien :

- Id : auto incrémente, de type entier ou Integer
- Titre : type string et taille (255 caractères)
- Description : type texte.

b. La sécurité

Les failles de sécurité classiques du développement web :

XSS : Le cross-site scripting (XSS) ou script intersites est une attaque par injection de script dans une application Web qui accepte les entrées mais sépare mal les données du code exécutable avant de renvoyer ces entrées vers le navigateur d'un utilisateur.

Pour élever le niveau de sécurité de nos applications de type PHP, on peut utiliser les fonctions addslashes (), strip_tags (), htmlentities (), htmlspecialchars (), strlen (), substr (), pour les projets twig on utilisera cette syntaxe {{}}, on peut activer la directive magic_quotes_gpc dans le fichier php.ini pour échapper automatiquement tous les caractères spéciaux (notamment les simples et doubles quotes) figurants dans les chaînes provenant de l'extérieur. Bien que ce n'est pas suffisant. A retenir qu'il faut faire attention aux données utilisateurs.



CSRF : L'objet de cette attaque est de transmettre à un utilisateur authentifié une requête HTTP falsifiée qui pointe sur une action interne du site, afin qu'il l'exécute sans en avoir conscience et en utilisant ses propres droits. L'utilisateur devient donc complice d'une attaque sans même s'en rendre compte. L'attaque étant actionnée par l'utilisateur, un grand nombre de systèmes d'authentification sont contournés.

Quelques mécanismes de prévention des attaques CSRF :

Utiliser les tokens sur les actions sensibles, voir exemple ci-dessous :

```
<form method="post" action="{{ path('property_delete', {'id': bien.id}) }}" onsubmit="return
confirm('Etes-vous sûr de supprimer ce champ ?');">
<input type="hidden" name="_token" value="{{ csrf_token('delete' ~ bien.id) }}">
<button class="btn">Supprimer</button>                                </form>
```

Éviter d'utiliser des requêtes HTTP GET pour effectuer des actions : cette technique va naturellement éliminer des attaques simples basées sur les images, mais laissera passer les attaques fondées sur JavaScript, lesquelles sont capables très simplement de lancer des requêtes HTTP POST et filtrer les entrées utilisateurs avec la fonction `strip_tags()`.

SQLi : Une injection SQL est une forme de cyberattaque lors de laquelle un pirate utilise un morceau de code SQL (« Structured Query Language », langage de requête structurée) pour manipuler une base de données et accéder à des informations potentiellement importantes

Prévention :

Filtrer les entrées de l'utilisateur avec les fonctions `addslashes()` ou `mysql_real_escape_string()`, `preg_match()`, `strlen()` et utiliser l'objet PDO.

Password : Utiliser un algorithme d'authentification fort telsques : algo, BCrypt, MD5, SCrypt

c. Le pattern MVC

Les principes du MVC et Symfony.

Le pattern MVC permet de bien organiser son code source. Il va vous aider à savoir quels fichiers créer, mais surtout à définir leur rôle. Le but de MVC est justement de séparer la logique du code en trois parties que l'on retrouve dans des fichiers distincts.

Le MVC sous Symfony :

- **Modèle** : cette partie gère les données de votre site. (Répertoire repository, entity)
- **Vue** : cette partie se concentre sur l'affichage. (Répertoire vue)
- **Contrôleur** : cette partie gère la logique du code qui prend des décisions. (Répertoire controller).



6. Développement

a. Les bundles

Liste des bundles utilisés dans le projet et leurs rôles.

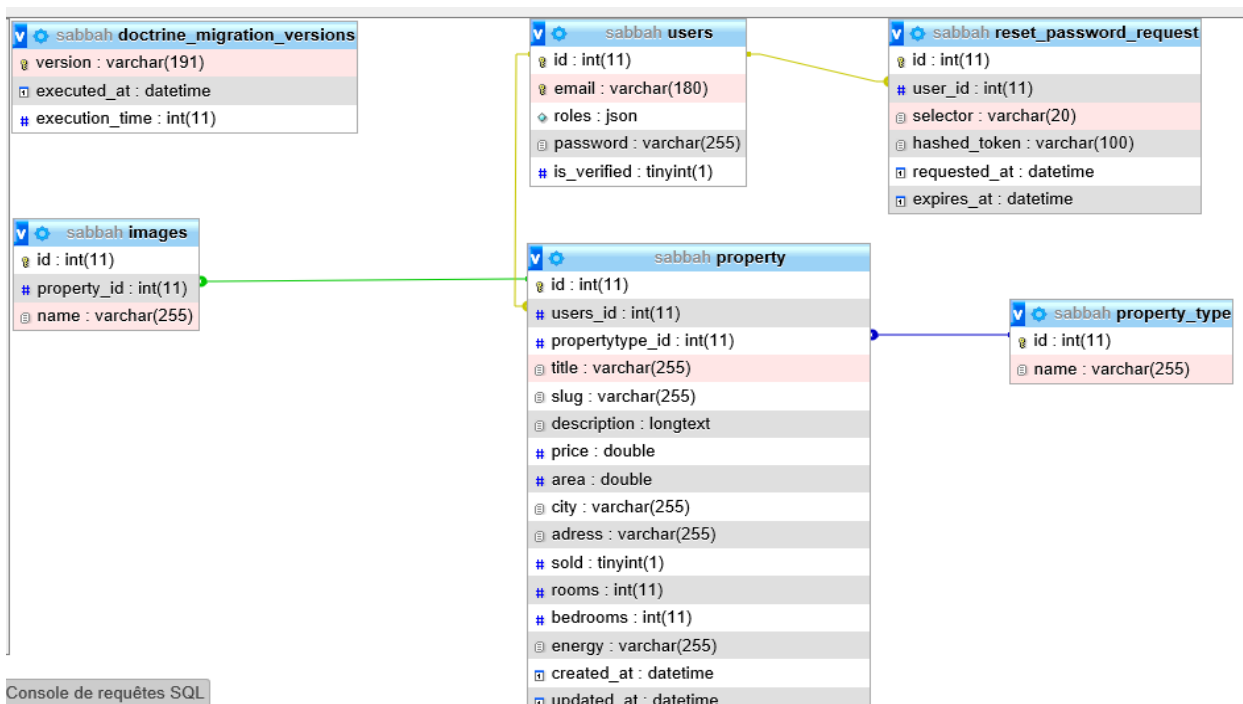
Nom	Rôles
FrameworkBundle	Le FrameworkBundle définit la configuration principale du Framework, des sessions et des traductions, le traitement des formulaires, le routage ...
SensioFrameworkExtraBundle	Le Symfony FrameworkBundle par défaut implémente un framework MVC basique mais robuste et flexible. SensioFrameworkExtraBundle l'étend pour ajouter des conventions et des annotations.
TwigBundle	Twig Configuration Reference (TwigBundle), le TwigBundle intègre la bibliothèque Twig dans les applications Symfony pour rendre les modèles (VUE).
WebProfilerBundle	Le WebProfilerBundle est un outil de développement qui fournit des informations techniques détaillées sur chaque exécution de requête et les affiche à la fois dans la barre d'outils de débogage Web et dans le profileur.
MonologBundle	Gestion des logs dans Symfony
DebugBundle	Fournit une intégration étroite du composant Symfony-Debug dans le framework Symfony full-stack
MakerBundle	Symfony Maker vous aide à créer, des contrôleurs, des classes, des formulaires, des tests à l'aide de commande et afin de vous fournir un code générique et maintenable
DoctrineBundle	Le projet Doctrine abrite plusieurs bibliothèques PHP principalement axées sur le stockage de bases de données et le mappage d'objets. Les projets principaux sont l'Object Relational Mapper (ORM) et la Database Abstraction Layer (DBAL) sur lesquels il repose.
DoctrineMigrationsBundle	Les migrations de bases de données sont un moyen de mettre à jour en toute sécurité le schéma de base de données à la fois localement et en production. Au lieu d'exécuter la commande doctrine:schema:update ou d'appliquer les modifications de la base de données manuellement avec des instructions SQL, les migrations permettent de répliquer les modifications de votre schéma de base de données de manière sécurisée
SecurityBundle	Le SecurityBundle intègre le composant de sécurité dans les applications Symfony. Toutes ces options sont configurées sous la clé de sécurité dans la configuration de votre application.
TwigExtraBundle	Ce package est un bundle Symfony qui permet d'utiliser toutes les extensions "supplémentaires » de twig sans aucune configuration.
DoctrineFixturesBundle	Ce bundle intègre la bibliothèque Doctrine2 Data Fixtures dans Symfony afin que vous puissiez charger les data fixtures par programme dans Doctrine ORM



VerifyEmailBundle	VerifyEmailBundle génère, valide l'url de manière sécurisée et signée qui peut être envoyée par e-mail aux utilisateurs pour confirmer leur adresse e-mail. Il le fait sans avoir besoin de stockage.
SymfonyCastsResetPasswordBundle	Ce composant fournit une solution sécurisée pour permettre aux utilisateurs de réinitialiser leurs mots de passe oubliés.
StofDoctrineExtensionsBundle	Ce composant permet l'intégration de DoctrineExtensions (Sluggable, Translatable) dans vos projets Symfony
KnplaginatorBundle	Ce composant fournit des paginations en fonction du résultat des requêtes.

b. Les entités

Les différentes entités du projet sous la forme d'un diagramme de classes UML.



Les controllers

- Les différentes routes du projet.



```
C:\wamp64\www\gestion-immo>php bin/console debug:router
-----
Name                                Method  Scheme  Host  Path
-----
_wdt                                ANY     ANY     ANY   /_wdt/{token}
_profiler_home                      ANY     ANY     ANY   /_profiler/
_profiler_search                    ANY     ANY     ANY   /_profiler/search
_profiler_search_bar                ANY     ANY     ANY   /_profiler/search_bar
_profiler_phpinfo                   ANY     ANY     ANY   /_profiler/phpinfo
_profiler_search_results            ANY     ANY     ANY   /_profiler/{token}/search/results
_profiler_open_file                 ANY     ANY     ANY   /_profiler/open
_profiler                           ANY     ANY     ANY   /_profiler/{token}
_profiler_router                    ANY     ANY     ANY   /_profiler/{token}/router
_profiler_exception                 ANY     ANY     ANY   /_profiler/{token}/exception
_profiler_exception_css             ANY     ANY     ANY   /_profiler/{token}/exception.css
admin_home                          ANY     ANY     ANY   /admin
admin_test                          ANY     ANY     ANY   /admin/test
property_add                        ANY     ANY     ANY   /admin/property/add
admin_property_index                ANY     ANY     ANY   /admin/property
admin_property_activate             ANY     ANY     ANY   /admin/property/activate/{id}
admin_property_delete               ANY     ANY     ANY   /admin/property/delete/{id}
admin_property_update               ANY     ANY     ANY   /admin/property/update/{id}
app_register                       ANY     ANY     ANY   /register
app_verify_email                   ANY     ANY     ANY   /verify/email
app_forgot_password_request         ANY     ANY     ANY   /reset-password
app_check_email                     ANY     ANY     ANY   /reset-password/check-email
app_reset_password                  ANY     ANY     ANY   /reset-password/reset/{token}
app_login                           ANY     ANY     ANY   /login
app_logout                          ANY     ANY     ANY   /logout
home_annonce                       ANY     ANY     ANY   /
property_show_guest                 GET|POST ANY     ANY   /{slug}
app_property_list                   GET      ANY     ANY   /member/view/{id}
property_new                        GET|POST ANY     ANY   /member/new/{id}
property_show                       GET      ANY     ANY   /member/{id}/show
property_edit                       GET|POST ANY     ANY   /member/{id}/edit
property_delete                     POST     ANY     ANY   /member/{id}
property_delete_image               DELETE   ANY     ANY   /member/supprime/image/{id}
_preview_error                      ANY     ANY     ANY   /error/{code}.{_format}
-----

C:\wamp64\www\gestion-immo>_
```

- Les différents controllers du projet.

Nom	Rôle
PropertyController	Ce contrôleur définit le mécanisme de crud (création, modification, mis à jour, et la suppression) au niveau de l'administrateur
PropertyHomeController	Ce contrôleur traite les informations qui seront mis à disposition d'un utilisateur non connecté
MemberPropertyController	Ce contrôleur traite le mécanisme de crud au niveau d'un utilisateur inscrit en tant que membre.
RegistrationController	Ce contrôleur traite le mécanisme d'inscription sur le site pour les utilisateurs membres ou administrateurs.
ResetPasswordController	Ce contrôleur traite le mécanisme de réinitialisation du mot de passe.
SecurityController	Ce contrôleur traite le mécanisme de connection sur le site pour les utilisateurs membres ou administrateurs

c. Les models

Présenter les models qui ont des requêtes customs.

Entités	Repositorys	Requêtes customs	Commentaire
Images	ImagesRepository		
Property	PropertyRepository	Private ou Public function findVisibleQuery () : QueryBuilder {return \$this->createQueryBuilder('p') ->andWhere ('p. active = : active') ->setParameter ('active', true) ; }	Cette requête retourne tous les biens dont la valeur de la propriété active vaut vrai (true)
PropertyType	PropertyTypeRepository		
Users	UsersRepository		
	PropertyTypeRepository		



d. Les views (et le frontend)

Exemple de code d'une view avec Twig.

```
{% extends 'base.html.twig' %}
{% block title %} nouveau Bien
{% endblock %}

{% block body %}

    <h1>Déposer une annonce</h1>
    {{include('member/_form.html.twig')}}
    <a href="{{ path('app_property_list', {'id': app.user.id}) }}">Retour à la liste</a>

{% endblock %}
```

7. Tests

a. Jeu d'essai

Les data fixtures : Les fixtures sont utilisées pour charger des données définies par les développeurs dans une base de données. Elles sont très utiles en environnement de développement, car elles permettent d'avoir une application avec plusieurs jeux de données qui correspondent à ce qu'il se passe en production. Ainsi, les tests d'intégration, fonctionnels se basent sur de la donnée concrète et réaliste.

Les différentes data fixtures du projet :

- UsersFixtures :

```
<?php

namespace App\DataFixtures;

use Faker;
use App\Entity\Users;
use Doctrine\Persistence\ObjectManager;
use Doctrine\Bundle\FixturesBundle\Fixture;
use Symfony\Component\Security\Core\Encoder\UserPasswordEncoderInterface;

class UsersFixtures extends Fixture
{
    private $encoder;

    public function __construct(UserPasswordEncoderInterface $encoder)
    {
        $this->encoder = $encoder;
    }

    public function load(ObjectManager $manager)
    {
```



```

    $faker = Faker\Factory::create('fr_FR');

    for ($nbUsers = 1; $nbUsers <= 20; $nbUsers++) {
        $user = new Users();
        $user->setEmail($faker->email);

        if ($nbUsers === 1) {
            $user->setRoles(['ROLE_ADMIN']);
            $user->setIsVerified(true);
        } else
            $user->setRoles(['ROLE_USER']);

        $user->setPassword($this->encoder->encodePassword($user, '123456'));
        $user->setEmail($faker->email);
        $user->setIsVerified($faker->numberBetween(0, 1));

        $manager->persist($user);
        // Enregistre l'utilisateur dans une référence
        $this->addReference('user_' . $nbUsers, $user);
    }
    $manager->flush();
}
}

```

- PropertyTypeFixtures :

```

<?php

namespace App\DataFixtures;
use App\Entity\PropertyType;
use Doctrine\Persistence\ObjectManager;
use Doctrine\Bundle\FixturesBundle\Fixture;

class PropertyTypeFixtures extends Fixture
{
    public function load(ObjectManager $manager)
    {
        $types = [
            1 => [
                'name' => 'appartement'
            ],
            2 => [
                'name' => 'maison',
            ],
            3 => [
                'name' => 'terrain constructible',
            ],
        ],
    }
}

```



```

4 => [
    'name' => 'studio',
],
5 => [
    'name' => 'parking',
],
6 => [
    'name' => 'local professionnel',
],
7 => [
    'name' => 'terrain non constructible',
],
];

foreach ($types as $key => $value) {
    $propertytype = new PropertyType();
    $propertytype->setName($value['name']);

    $manager->persist($propertytype);
    // Enregistre la propertytype dans une référence
    $this->addReference('propertytype_' . $key, $propertytype);
}
$manager->flush();}}

```

- PropertyFixtures :

```

<?php

namespace App\DataFixtures;
use Faker;
use App\Entity\Images;
use App\Entity\Property;
use Doctrine\Persistence\ObjectManager;
use Doctrine\Bundle\FixturesBundle\Fixture;
use Doctrine\Common\DataFixtures\DependentFixtureInterface;

class PropertyFixtures extends Fixture implements DependentFixtureInterface
{
    public function load(ObjectManager $manager)
    {
        $faker = Faker\Factory::create('fr_FR');
        for ($nbProperty = 1; $nbProperty <= 20; $nbProperty++) {
            $user = $this->getReference('user_' . $faker->numberBetween(1, 19));
            $propertytype = $this->getReference('propertytype_' . $faker->
            >numberBetween(1, 7));

            $property = new Property();
            $property->setUsers($user);
            $property->setPropertytype($propertytype);
            $property->setTitle($faker->realText(20));
            $property->setDescription($faker->realText(300));
            $property->setAdress($faker->address);
        }
    }
}

```



```

$property->setArea($faker->numberBetween(10, 500));
$property->setRooms($faker->numberBetween(2, 10));
$property->setBedrooms($faker->numberBetween(1, 9));
$property->setPrice($faker->numberBetween(100, 900000));
$property->setEnergy('solaire');
$property->setPhone($faker->phoneNumber());
$property->setCity($faker->city);
$property->setPostcode($faker->postcode);
$property->setSold($faker->numberBetween(0, 1));
$property->setActive(true);

// On upload et on génère les images
for ($k = 1; $k <= rand(1, 6); $k++) {
    // $rand = rand(1, 3);
    $img = 'public/uploads/image' . rand(1, 6) . '.jpg';
    $imageProperty = new Images();
    $nomimage = str_replace('public/uploads/', '', $img);
    $imageProperty->setName($nomimage);
    $property->addImage($imageProperty);
}
$manager->persist($property);
}
$manager->flush();
}
public function getDependencies()
{
    return [
        UsersFixtures::class
    ];
}
}

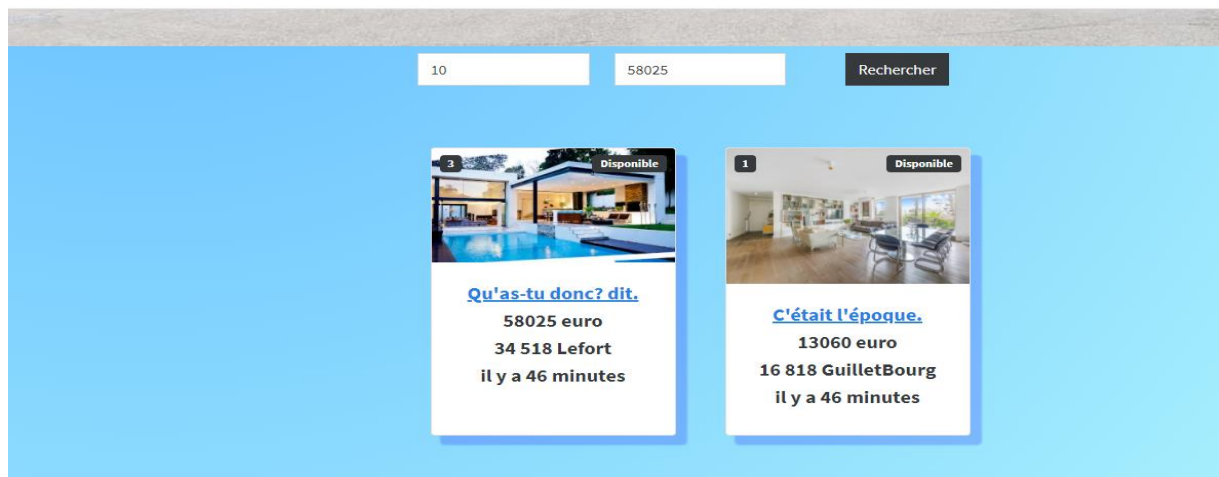
```

b. Test fonctionnel d'une fonctionnalité représentative

Test fonctionnel :

Si l'on définit la superficie minimale à 10 et le budget max à 58025, selon l'état actuel de la base de données, il y'a deux biens immobiliers qui répondent à ce critère de recherche, comme le démontre la figure ci-dessous.





8. Veille technologique

The Upload image in Symfony (Le chargement d'image sous Symfony) :

La gestion des images reste un sujet majeur dans la conduite de ce projet, à l'état actuel, l'application ne gère pas le redimensionnement des images.

9. Conclusion

La version actuelle du projet est testée, et mise en production sur le serveur Heroku sans bug majeur.

En outre, Je prévois de mettre en place la politique de geolocalisation des annonces, avec l'API (Application interface Programm) d'OpenStreetMap.

Ce projet m'a apporté une montée en compétence sur les technologies PHP7.4, le framework Symfony, Html5, CSS3, twig, Heroku, Github et la conception des systèmes d'information.

