Synthèse de compétences

BALANGUE Jean

Maintenance de l'espace membre d'un site web

Titre professionnel développeur informatique
greta 2017

Introduction

Dans le cadre de la formation Javascript du Greta session 2017, nous avons eu la possibilité de présenter le titre professionnel développeur logiciel à l'issue de la formation. Pour cela, il nous fallait réaliser en plus de notre formation et sur notre temps libre une synthèse d'un projet réalisé.

Je vous présente donc un projet déjà en ligne, qui a pour but d'aider les particuliers dans la gestion de leurs biens disponibles en location saisonnière. Deux contrats sont nécessaires pour chaque location hebdomadaire et la rédaction de ce genre de document peut se révéler fastidieuse. J'avais constaté qu'aucun site ne proposait de service interactif et je m'étais donc lancé.

J'ai pour cela réalisé en 2013 sur mon premier site web un formulaire interactif de contrat de location saisonnière qui génère un Pdf avec les informations nécessaires pour la légalité du document, et qui conviendra au plus grand nombre. J'ai également proposé l'enregistrement du contrat sur le site, afin d'éviter d'avoir à remplir une nouvelle fois le document. J'ai pour cela appris les bases des langages Php et Javascript.

Le service a bien démarré et a compté plus de 1000 utilisateurs différents la première année. En 2016, mon hébergeur a décidé de modifier l'encodage des connexions à la base de données sur ses serveurs mutualisés. Mon service d'enregistrement ne fonctionnait alors plus normalement et des erreurs de caractères apparaissaient sur le contrat de location enregistré. Le service d'enregistrement devenait inutilisable et corrompait le contrat des utilisateurs lors de l'édition d'un nouveau contrat. Devant l'ampleur de la tache de maintenance à réaliser, et ne disposant que de très peu de temps, j'ai suspendu le service d'enregistrement et je n'ai rien fait depuis.

Je vous propose donc dans un premier temps d'analyser le site existant, puis de faire le point sur les problèmes, et enfin de suivre le début de la remise en état de ce service.

Abstract :

At the end of our 2017 JavaScript professionnal training, we have the possibilty to present the « Software developer » certificate. To make this possible, I have to carry out, on free time , a summary document of a project made by myself.

I will show you a project already online. The goal is to help property owner for the holidays renting act, giving them an administrative framework. Two contracts are needed for each location, and document's redaction can be tiresome. I have seen there is no interactive service on the web, so i decided to build one.

To do this, i have achieved, in 2013, a contract template who generate a PDF document with all clauses needed to be french law compliant. And, finally, to save time, i offer users to save theirs documents on my database. For all this, I needed to learn PHP and Javascript bases.

More than thousand users generate documents with my template on first year. In 2016, my webhost modify the database connection encoding on his mutualized servers. My records services still don't work normally since this day. A lot of encoding's errors appears on documents and corrupt the database when users try to edit a new contract by clicking the all-in-one button. I decided to close the access to registrered users to their account and stop the registration service. I haven't done nothing more since that.

First, i propose you to analyse the website, then define majors problems, and finally, follow the beginning of the services maintenance.

Remerciement :

Je remercie les formateurs et l'équipe administrative du Greta qui nous ont suivi durant cette formation.

Je souhaite également remercier mes collègues de la formation Javascript du Greta 2017 avec qui j'ai beaucoup appris et progressé.

Et enfin, je remercie la région Nouvelle Aquitaine pour avoir subventionné cette formation.

Sommaire

I - Présentation du projet existant

1- Le site www.locationsaintjeandeluz.fr

Il s'agit au départ d'un site web de présentation de trois locations saisonnières appartenant à des amis. Dans un premier temps j'ai créé la page d'accueil et les locations, dans un deuxième temps, j'ai réalisé le contrat de location saisonnière et l'état des lieux, et dans un troisième temps j'ai réalisé l'espace membre avec l'enregistrement des membres et des contrats. j'ai de plus fait une multitude de modifications non listées ci-dessus mais qui ne nécessitait pas une étape à elles-seules. Rien n'a été pensé depuis le début car j'ai découvert petit à petit les capacités des différents langages. Nous allons voir qui fait quoi un peu plus loin. Nous allons d'abord nous intéresser à mon environnement de travail.

A - Environnement de travail

a – logiciels utilisés

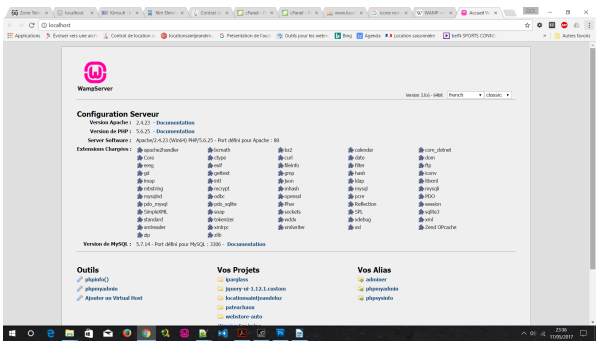
Pour commencer, je vais vous présenter mon environnement de travail qui se compose physiquement d'une tour et d'un double écran. Niveau logiciel, mon OS actuel est Windows 10 alors que c'était Windows 7 lors du début du projet.

Notepad ++

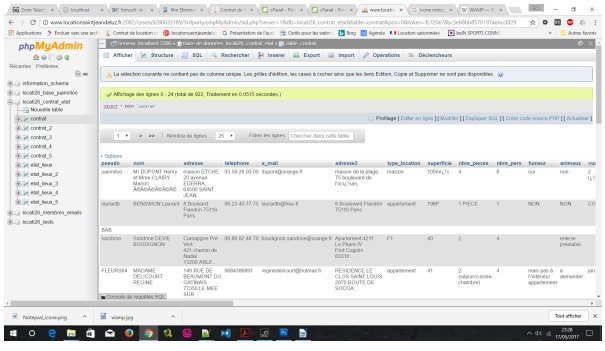
Je me sers principalement de Notepad++ comme éditeur de texte même si j'ai été amené à découvrir Visual Studio Code lors de ma formation. Notepad est un éditeur de texte libre. Il a l'avantage d'être très rapide et de proposer une coloration syntaxique à laquelle je me suis habitué.

Wamp 

Afin de simuler un serveur web en local, je me sers de Wamp, accronyme de Windows, Apache, Mysql, Php. Apache permet de simuler un serveur web sur mon ordinateur, Mysql et un logiciel de gestion de base de données qui me permet de gérer et stocker mes données, Php est le langage de script utilisé côté serveur et enfin l'OS Windows gère tout ça. Wamp me permet de tester mon code avant de le mettre en ligne. Il est également doté d'un outil bien pratique pour la gestion des bases de données, phpMyAdmin qui est une interface graphique pour Mysql.



Wamp



phpMyAdmin

b – hébergement

Le site est hébergé chez Planethoster, il s'agit d'un hébergement mutualisé, c'est à dire que l'on partage un serveur entre plusieurs sites web, c'est très économique et on a rien à paramétrer. En contrepartie, on doit composer avec le paramétrage généraliste.

c – technologies utilisées

Html 5 acronyme d' HyperText Markup Language est la dernière version d'HTML qui est un format de données conçu pour représenter des pages web. c'est le langage structurant l'information. Finalisé fin 2014, il apporte entre autres de nouvelles balises structurantes comme « section » et des API « application programming interface » comme la géolocalisation.

Css 3 signifie Cascading style sheet. C'est le langage de mise en forme. La version 3 a été massivement adopté avec l'apparition de l'internet mobile et la possibilité de faire de l'affichage dynamique selon la largeur de l'écran de l'utilisateur.
C'est l'ère des « media queries » et du « responsive design ».

Php pour Hypertext Preprocessor, Il est apparu en 1994 et permet des créer des pages web dynamiques, c'est à dire personnalisé selon l'utilisateur. C'est un langage interprété seulement par les serveur. En 2016, selon w3techs, c'est le langage serveur utilisé par 82 % des sites dynamiques présent sur le net.

My**SQL** est un système de gestion de base de données. Il permet créer des bases de données et de manipuler les données présentes dans la base. On communique avec lui par l'intermédiaire du langage SQL, « structure query language ».

Javascript est un langage de programmation apparu en 1995 qui est aussi bien « client » que « serveur ». C'est à dire qu'il peut être utilisé aussi bien sur l'ordinateur de l'utilisateur que sur un serveur avec l'utilisation de node.js par exemple. Il m'a principalement permis la manipulation des éléments html présents sur mes pages.

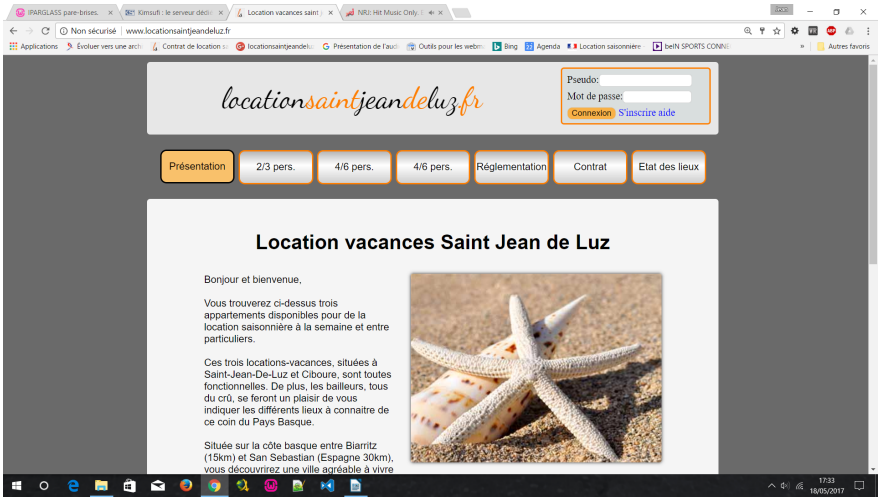
Jquery bibliothèque Javascript libre créée pour faciliter la création de scripts côté client. Je me suis servi principalement de cette bibliothèque pour la création de mes scripts à partir du moment ou je l'ai découverte. Write less, do more !!

Ajax est l'acronyme d'Asynchronous Javascript and Xml. C'est une architecture informatique combinant un langage de programmation serveur ainsi qu'un client. Elle permet d'interroger le serveur pour une requête SQL et de modifier le contenu de la page html en conséquence sans recharger la page du client. Dans mon cas, ce fût dans le but de vérifier lors de la frappe de l'utilisateur si le pseudo était déjà pris ou si l'email était déjà présent dans ma base de données.

B – front-end du site

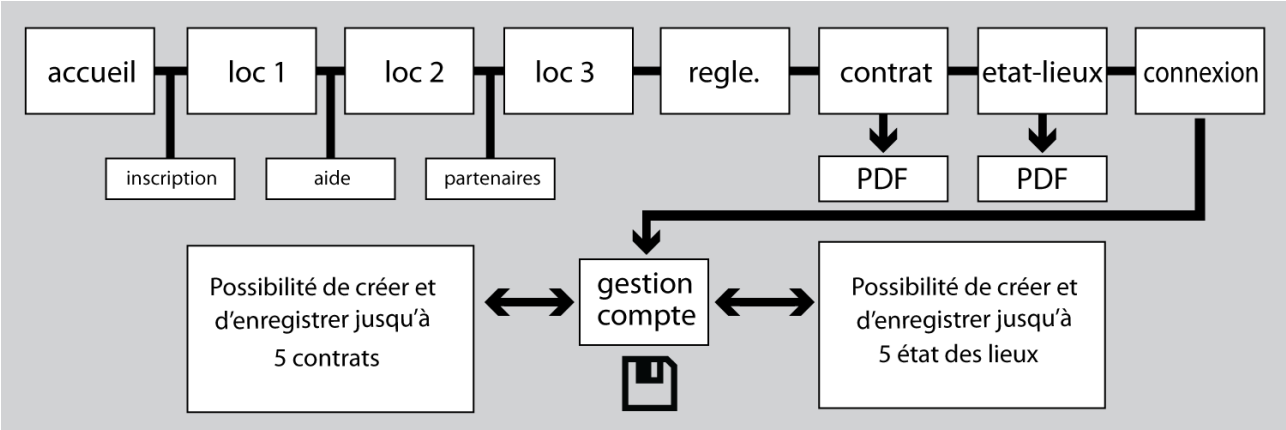
a – structure html

Avant de vous montrer des bouts de codes à proprement parler, je vais vous montrer à quoi il ressemble. C'est un site responsive à deux niveaux, c'est à dire un affichage écran et un affichage téléphone mobile, parce que de part sa conception basique, j'avais naturellement les éléments les uns en dessous des autres, l'adaptation était donc facile.



Full-screen mobile

Voici un petit schéma qui explique le fonctionnement du site :



En ce qui concerne le code Html de la page d'accueil, il est des plus basique. Nous trouvons la nouvelle déclaration de doctype d'Html5. Pour rappel, il s'agit d'une instruction au début du document qui définit les règles et la syntaxe de celui-ci. Nous trouvons ensuite une imbrication classique avec une balise *div* rajoutée pour unifier mes balise *section, nav* et *footer.*

On peut voir 2 menus présents dans le code, l'un pour le site normal, et l'autre pour la version mobile. Ce dernier est masqué par défaut sur le site via la règle CSS *display : block* et la 1ère règle *media queries,* Le bouton menu pour la version mobile apparait si l'écran à la largeur désirée, ici 730 pixels et est géré par un script jquery situé sur la page html et donc présent sur toutes les pages. J'utilise aussi jquery pour la galerie photos, avec 6 photos qui s'affichent à tour de rôle. Script qui ne se lance pas sur la version mobile, où je me contente d'afficher une photo.

A côté du code Html, j'ai un fichier CSS séparé pour la page d'index, les 3 locations et la page de réglementation. Le page de contrat et celle d'état des lieux sont gérées par un autre fichier css car j'ai donné des règles à des balises génériques comme *article* qui empêchait la mise en page désirée.

index.html

```
<!DOCTYPE html>
<html lang="fr" >
    <head>

    <body>
        <div id="page">
            <section class="corps_page1">
                <header>
                    <span class="head1">location</span><span class="head2">saint</span><span class="he
                </header>
                <form method="post" action="gestion-compte.php" class="connexion" >
            </section>

            <nav>
                <ul class="menu-ul"><li class="active-menu"><a href="http://www.locationsaintjeandeluz
                <div id="overlay"></div>
                <div class="menu-mobile"> <p class="menu-mobile-p">Menu</p></div>
                <ul id="menu-mobile-ul"><li class="active-mobile"><a href="http://www.locationsaintjea
            </nav>

            <section  class="corps_page2"><br />
                <article>
                <!--google + et facebook -->
                <div class="recommandation">
            </section>

            <footer>

        <div class="message-cookies">
```

index.html | index.css

```
    color:white;
}

}
/* fin des media queries 5000 px  ----------------

/*----------------------------------------------

/*media queries 730 px ----------------------------

@media screen and (max-width: 730px){

body{
    width: auto;
}

#page{
    width: auto;
    margin: 0 auto;
    margin-top:-2em;
    background-color: rgba(255,255,255,0);
}

header{
    width: auto;
    font-size: 1.8em;
    display: block;
    margin: 0 auto;
}
```

Cascade Style Sheets File

index.html | index.css

```
        setInterval(mesPhotos,21000);
    }

//menu-mobile
    $('.menu-mobile').on('click', function() {
        if ($('#menu-mobile-ul').css('display')=='none'){
            $('#menu-mobile-ul').css('display','block');
            $('#overlay').css('display','block');
            $('.menu-mobile').css('z-index','1');
        }else{
            $('#menu-mobile-ul').css('display','none');
            $('#overlay').css('display','none');
        }
    });

    $('#overlay').on('click', function(){
        $('#overlay').css('display','none');
        $('#menu-mobile-ul').css('display','none');
    });

// menu toujours apparent
    $(window).scroll(function(){
        if(($(window).scrollTop()>90)&&($(window).scrollT
            $('.menu-mobile').css('position','fixed').css
            $('.menu-mobile').css('display','none').fadeI
            $('#menu-mobile-ul').css('position','fixed');
        }else if(($(window).scrollTop()>120)&&(document.d
            $('.menu-mobile').css('position','fixed').css
```

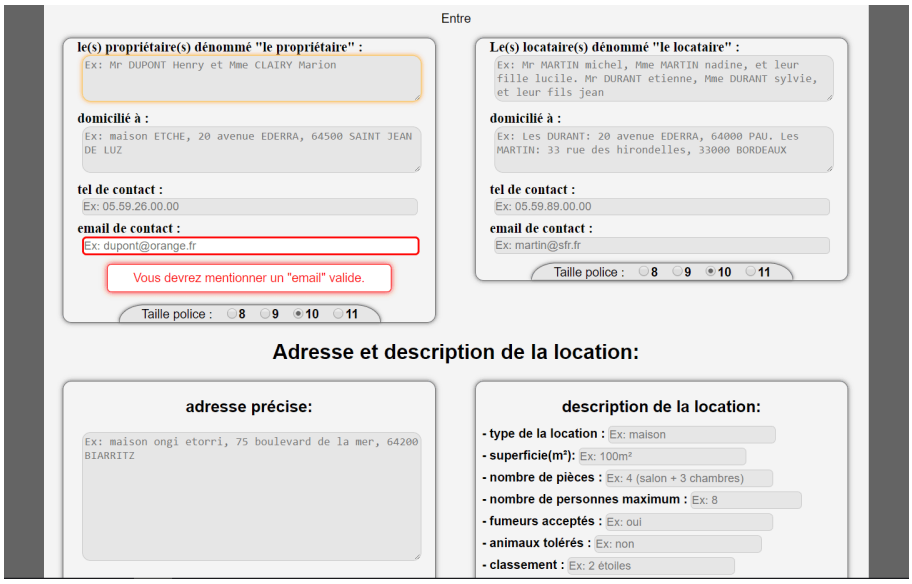
b – swipe pour téléphone mobile

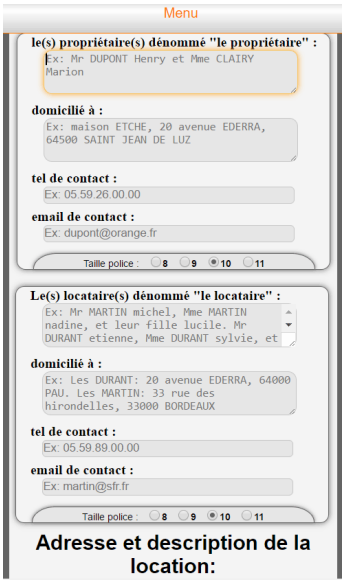
En ce qui concerne les trois pages de location et celle de réglementation, elles ont toutes une structure classique. On peut noter un script photo que j'ai amélioré avec l'apport d'un swipe pour les appareils tactiles. Il est cependant redondant et a été amélioré sur d'autres projets. Je gère tout les cas un par un à l'intérieur des *switch*.

```
// swipe image mobile ////////////////////////////////////////////////////////////
var touchdebutX=$('#photo').on('touchstart', function(e){
    touchdebutX = parseInt(e.originalEvent.changedTouches[0].pageX);
}); //on récupére la coordonnée x de début
    //on récupére la coordonnée x de fin
var touchfinX= $('#photo').on('touchend', function(e){
    touchfinX = parseInt(e.originalEvent.changedTouches[0].pageX);

    var resultX = (touchdebutX-touchfinX);

        if(resultX>=30){//de droite a gauche

            switch($('#photo').attr('src')){

        }else if(resultX<=-30){//swipe de gauche à droite

            switch($('#photo').attr('src')){

        }else{
            return;
        }
});
```

c – formulaire interactif

Nous allons maintenant voir le contrat. L'idée m'est venu lorsque j'expérimentais Php et que je réalisais mes premiers formulaires. J'ai eu l'idée de créer un formulaire afin de permettre aux internautes de générer leur contrat sur le site. J'ai donc tout d'abord pour cela travaillé ce qu'il fallait de réglementation. J'ai ensuite créé un contrat sous libre-office avant de le transposer en formulaire Html. Nous allons pour le moment nous concentrer sur le front-end avant de voir le fonctionnement du back-end.





Je ne vais pas m'amuser à vous afficher 200 lignes de codes Html. Ce que je peux vous en dire, c'est que j'assiste les propriétaires bailleurs dans la rédaction de leurs contrats en leur faisant apparaître des

astuces en rouge par l'intermédiaire de balises *p* qui disparaissent une fois la lecture effectuée, grâce à la méthode *setTimeOut,* qui exécute une action une fois le délai de temps déterminé écoulé, et la propriété jquery *fadeOut,* qui apporte une touche d'animation.

L'utilisateur a également la possibilité de choisir entre 4 tailles de polices pour la rédaction de son contrat. Vous pouvez aussi voir les *placeholder* qui me permettent de donner un exemple à l'utilisateur pour chaque champ. Ces *placeholder* disparaissent au début de la saisie utilisateur *et* sont gérés par un script Modernizr afin de les faire apparaître sous IE8. Modernizr est une bibliothèque Javascript qui permet la conversion d'éléments Html5 en éléments appréhendables par les navigateurs web ne supportant pas la cinquième version d'Html. En résumé, IE8 n'affiche pas les *placeholder*, Modernizer crée un élément par l'intermédiaire d'un script Javascript qu'IE8 sera capable d'afficher.

J'en profite pour préciser que l'intégralité du site est disponible sous IE8 par l'intermédiaire de feuilles Css qui lui sont spécialement destinées, et que je déclare à l'intérieur de mon header. Les versions antérieures ne sont pas pris en charge.

d - script de validation du formulaire

```
    //modif

    valid2.onclick=function(){
        if(case2.checked){
            if(/^[a-z0-9._-]+@[a-z0-9._-]+\.[a-z]{2,6}$/.test(email.value)){
////////////////////// modif liés a l'erreur d'encodage ////////////////////////////////////////////////////////////
                //confirmContrat.style.display = 'block';
                //overlayContrat.style.display = 'block';
                //return false;
            //soumission
                $('#contrat-en-ligne').submit();
            }else{
                alert('Votre e-mail de contact doit avoir une forme valide :\n\n                exemple@exemple.fr ');
                email.focus();
                window.scrollTo(0,yyy);
                return false;
            }
        }else{
            alert('Vous devez lire et accepter les conditions générales d\'utilisation !');
            return false;
        }
    }
```

Lors du clic sur le bouton de validation du formulaire, je vérifie avec les 2 conditions *if else* si la case des conditions générales a bien été coché et si l'e-mail a une forme valide à l'aide de la fonction *test()* et d'une expression régulière, qui permet de faire de la recherche texte, et éventuellement du remplacement comme pour le formatage d'une date.

Si toutes les variables sont bonnes, alors j'envoie le formulaire à mon script Php. Sinon, soit les conditions générales n'ont pas été accepté et je sors de la soumission en affichant un message via la fonction *Alert()* , soit l'e-mail n'a pas la bonne forme et je renvoie, via l'instruction *return false*, sur la case e-mail via les fonctions *focus()* et *scrollTo(x,y)*.

e - Appel Ajax pour la vérification du pseudo

On peut voir dans le script au dessus en commentaire que j'ai du modifié le script de validation à cause de l'erreur d'encodage, pour le remettre à son état initial. En effet une fois l'espace membre créé, j'avais modifié le script de la page de contrat pour permettre ce que j'ai appelé « l'inscription à la volée » .

Au moment de la soumission du formulaire, je faisais apparaître une *div* grâce à la propriété *onclick.* où je demandais à l'utilisateur s'il ne voulait pas s'inscrire avant de générer son contrat. S'il acceptait, je faisais apparaître le formulaire d'inscription en sur-impression, et lors de la validation, je l'inscrivais en tant que membre, lui envoyais le mail de confirmation, insérais son contrat dans la base et générais le pdf d'un seul coup.

Durant ce processus, je faisais un appel Ajax pour déterminer la disponibilité du pseudo en temps réel lors de la saisie. Le but étant d'éviter tout conflit avec un utilisateur existant et de bloquer la soumission

du formulaire à un endroit précis de mon script sans perturber l'ergonomie afin qu'il puisse corriger l'erreur. Vous pouvez voir le script sur la page suivante.

Je ne vais pas vous détailler l'intégralité du script mais plutôt vous décrire son fonctionnement.
Je commence par initialiser deux variables textes vides, *Pseudo* qui me servira à initialiser la longueur du pseudo à 0 lorsque le champ est vide au début, et *retourAjax qui me permettra de* recupèrer une valeur de retour pour valider la condition de soumission du formulaire final, ici « ok ».

Ensuite j'associe un événement au soulèvement d'une touche avec la fonction *keyUp()* lorsque l'utilisateur est dans le champ pseudo. Cet événement consiste à envoyer un appel Ajax à une page Php qui lance alors une requête Sql afin de vérifier si le pseudo est disponible dans la base de données. Le script Php se charge alors de renvoyer une valeur, dans mon cas, « *dispo* » ou « *pas dispo* », qui me permet ensuite d'afficher la bonne information à l'utilisateur.

L'appel Ajax n'est éxecuté que si le nombre de caractères du pseudo est différent du précédent essai, afin de ne pas lancer d'appel lors de l'appui sur la touche entrée par exemple.

```
719  //ajax pseudo
720  var Pseudo = '';
721  var retourAjax = '';
722  $('#pseudo-inscription').on('keyup',function(){
723      //  si il y a des caractères spéciaux
724      if(/[^a-zA-Z0-9_.\-@]/.test($('#pseudo-inscription').val())){
725          $('#mess-erreur4').text('Vous ne pouvez pas utiliser de caractères spéciaux.').
726          $('#pseudo-inscription').css('border-color', 'red');
727          retourAjax = 'pas bon';  //bloquage de la soumission du form
728      }else{
729          var longueurPseudo = Pseudo.length;
730          var longueurPseudoModif = $('#pseudo-inscription').length;
731          var valeurPseudoModif = $('#pseudo-inscription').val();
732
733          if (longueurPseudoModif != longueurPseudo){
734              $.ajax({
735                  url : 'verif-pseudo-ajax.php',
736                  type : 'GET',
737                  data : 'pseudo=' + valeurPseudoModif,
738                  datatype : 'text',
739                  success : function(text, statut){
740
741                      if ( (text=='dispo') &&  ($('#pseudo-inscription').val().length < 3
742                          $('#mess-erreur4').text('Veuillez choisir un pseudo d\'au moins
743                          $('#pseudo-inscription').css('border-color', 'rgb(255,50,50)').
744                          retourAjax = 'pas bon';
745                      }else if ( (text=='dispo') &&  ($('#pseudo-inscription').val().lenc
746                          $('#mess-erreur4').text('Ce pseudo est disponible').css('color'
747                          $('#pseudo-inscription').css('border-color', 'rgb(00,150,00)').
748                          retourAjax = 'ok';
749                      }else{
750                          $('#mess-erreur4').text('Ce pseudo n\'est pas disponible').css(
751                          $('#pseudo-inscription').css('border-color', 'red').css('box-sh
752                          retourAjax = 'pas bon';
753                      }
754
755                      if(valeurPseudoModif==''){
756                          $('#mess-erreur4').text('Veuillez choisir un pseudo d\'au moins
757                          $('#pseudo-inscription').css('border-color', 'red').css('box-sh
758                          retourAjax = 'pas bon';
759                      }
760                  }
761              });
762          }
763          Pseudo == valeurPseudoModif;
764      }
765  }); //fin ajax pseudo
```

Nous allons maintenant nous intéresser à la structure Ajax de Jquery.

L'Ajax permet de faire des requêtes asynchrones, c'est à dire que l'on a pas à attendre la fin de l'instruction précédente pour que le script général continue. On y accède grâce à une fonction native de JavaScript, XmlHttpRequest.

Afin d'instancier un objet XmlHttpRequest, il suffit de lancer l'instruction $.ajax({ …}) ;

Il ne nous reste plus qu'à utiliser les différents paramètres à transmettre à notre requête http.

Le paramètre ***URL*** me permet de spécifier quelle est la ressource ciblée sur le serveur.
Le paramètre ***type*** me permet de spécifier le type de transfert de données de la requête, GET ou POST. N'ayant qu'un paramètre à ajouter à ma requête, le pseudo à tester, je choisis la méthode GET.
Le paramètre ***data*** me permet de spécifier le (ou les) paramètre de ma requête, ici le pseudo. Je fais donc une concaténation pour pouvoir accéder à mon paramètre en Php.
Le paramètre ***datatype*** me permet de spécifier le type de données renvoyées par le serveur et donc attendu par le script, pour moi du texte vu qu'il ne s'agit que d'une réponse boléenne en fait. Sinon, on pourrait trouver par exemple du Html, du Xml ou Json selon les données attendues en retour. Le Json est à la mode, il signifie JavaScript Object Notation.
Le paramètre ***success*** utilise une fonction de retour pour me renvoyer le résultat uniquement si l'appel a réussi. Ce résultat est mentionné en tant que 1er argument de la fonction et on peut donc l'utiliser dans le code en dessous, il s'agit ici de « text » qui renvoie la valeur « dispo » ou « pas dispo ». Le paramètre statut renvoie une chaîne de caractère automatiquement généré par Jquery permettant de suivre le déroulement.
Le paramètre ***error*** prend trois argument et permet d'afficher un message à l'utilisateur si l'appel n'a pas marché. Il n'est pas utilisé ici. Enfin, beaucoup d'autres paramètres sont disponibles si nécessaire. Dans mon script de validation, s'il y a eu une erreur, c'est à dire que je n'obtiens pas la valeur « ok » pour la variable *retourAjax*, je redemande un nouveau pseudo.

Je vous ai dit dans la présentation d'Ajax que c'était une architecture qui combinait un langage client et un langage serveur, il est donc tant de s'intéresser au back-end du site, en commençant par le petit script Php correspondant à l'appel Ajax.

```php
<?php
$pseudo = htmlspecialchars($_GET['pseudo'], ENT_IGNORE,'ISO-8859-15');
try{
    $bdd = new PDO('mysql:host=localhost;dbname=test','root','');   //  host=localhost;dbname=locat
}catch(Exception $e){
        die('Erreur : ' . $e->getMessage('Connection à la base de donnée impossible.
        Veuillez fermer cette page et réessayer s\il vous plait'));
}
if (isset($pseudo)){
    $verification_pseudo = $bdd -> prepare('SELECT pseudo FROM membres WHERE pseudo = ?');
    $verification_pseudo->execute(array($pseudo));
    $donnee = $verification_pseudo->fetch();

    if ($donnee['pseudo'] != $pseudo){
        echo "dispo";
    }else{
        echo "pas dispo";
    }
}
$verification_pseudo-> closecursor();
?>
```

Nous trouvons un document Php avec une architecture procédurale, bien que la connexion se fasse avec l'objet PDO. On trouve sur la page tout les éléments nécessaires au fonctionnement du script, les uns en dessous des autres. Détaillons un peu le fonctionnement :

- je récupère et sécurise ma variable

Par l'intermédiaire de la fonction *GET* de Php, je récupère la variable *pseudo* envoyée par l'appel Ajax, que je sécurise immédiatement afin d'éviter toute injection sql. C'est la fonction *htmlspecialchars()* qui est utilisée pour sécuriser ma variable à laquelle je passe 2 paramètres, *ENT_IGNORE* qui ignore les séquences de caractères invalides, et *ISO-8859-15* qui est l'encodage de mon élément, nous verrons ça après. Cette fonction échappe les chevrons html <> . En lisant la documentation Php, je m'aperçois que le paramètre ENT_IGNORE est maintenant fortement déconseillé pour des raisons de sécurité !

- je me connecte à la base de donnée

J'utilise la structure try catch pour la connexion à la base de données, qui me permet soit d'instancier un objet PDO dans la variable *$bdd*, soit de récupérer une exception *$e* si jamais ça n'a pas marché et d'afficher un message d'erreur, sans dévoiler mes identifiants de connexion.
PDO, pour PHP Data Object, est une classe maintenant native de PHP qui a pour but de remplacer les anciennes méthodes mysql_connect() et mysqli_connect(). Elle permet aussi de se connecter à plusieurs SGBD différents avec les mêmes instructions. La méthode *PDO::query()* retourne un objet *PDOStatement* issus de la classe du même nom. PDOStatement représente donc soit une requête préparée, soit le résultat associé à cette requête. Les exceptions sont lancées par la méthode *throwable()* de la classe *Exception.*

- je fais ma requête

Bien qu'il n'y est qu'une variable, je fais quand même une requête préparée par habitude et pour plus de sécurité. j'utilise une condition pour tester que la variable à tester *pseudo* existe avec *isset()*. Je prépare donc une requête à la base de données pour lui demander de me retourner tous les pseudos contenus dans la base et qui correspondent au pseudo à tester. Je mets cette requête dans la variable *verification_pseudo*. J'exécute ensuite cette requête via l'instruction *execute()*. Enfin, j'attribue la réponse à cette requête à la variable *donnee.* Enfin, je récupère les entrées une à une avec l'instruction *fetch()*.

- je renvoie ma valeur

Le but de la demande précédente est de pouvoir comparer si le pseudo utilisateur correspond à un pseudo existant dans la base données. j'utilise une condition *if else* pour tester la présence ou non du pseudo dans la base. S'il n'est pas présent, j'affiche avec l'instruction *echo* « dispo » et s'il est présent j'affiche « pas dispo ». C'est cette valeur affichée qui sera retourné au script Ajax par la fonction de retour contenu dans l'objet $.ajax() et qui me permettra d'informer l'utilisateur en Javascript.

Nous avons maintenant fait le tour de mon appel Ajax et pouvons étudier plus en détail le back-end.

C – Back-end

a – Structure

C'est mon premier site, je n'avais pour ainsi dire aucune connaissance au début du projet et j'ai été amené à améliorer le site au fur et à mesure de l'acquisition de nouvelles connaissances.

L'organisation des fichiers est donc des plus sommaire. J'ai un dossier pour les photos de chaque locations, un répertoire pour les images, et c'est tout. Le reste des fichiers est dans la racine du site, c'est le bazar ! Ce sera donc un objectif de maintenance.

b – Persistance des données

Une base de données est un outil permettant de stocker de manière durable des données, tels que des chiffres ou des mots. Les informations sont triées avant d'être stockées afin de pouvoir les manipuler facilement par l'intermédiaire du système de gestion de base de données (SGBD). Une base de données est dite relationnelle lorsqu'elle est composée d'un ensemble de tables (les *relations*) dans lesquelles sont placées les données. Je me sers de Mysql comme SGBD

Afin d'assurer la persistance des données sur mon site, j'ai eu recours à un système de base de données. Pourquoi un système ? Parce que je n'utilise pas 1 mais 2 bases de données. Je me rends bien compte de l'hérésie maintenant. Je me sers donc de la variable de session *Pseudo* pour lier mes tables quand nécessaire. On touche donc là un deuxième gros problème.

Qu'est ce qu'on trouve dans ces deux bases ?

Dans la première, on trouve 2 tables, celle des membres et celle des e-mails des utilisateurs non-authentifiés. J'ai donc appelé cette base *membres_emails.*

Dans la deuxième on trouve 10 tables, 5 pour les contrats et 5 pour les états des lieux. C'est donc la table *contrat_etat.* Mais les utilisateurs souhaitaient en enregistrer davantage. J'ai donc fait du copier-coller vite-fait pour les satisfaire.

Pour créer et administrer mes bases de données, j'ai utilisé phpMyAdmin, d'abord par l'intermédiaire de Wamp avant de les exporter en ligne. J'ai créé ces 2 bases de données comme j'aurais créé 2 dossiers dans l'explorateur windows, sans en comprendre les conséquences. Nous allons maintenant voir ces tables plus en détail.

Table des utilisateurs non-authentifiés

C'est la table des utilisateurs qui ont réalisés soit un contrat soit un état des lieux sur le site et sans s'enregistrer en tant que membre.

#	Nom	Type	Interclassement	Attributs	Null	Valeur par défaut	Commentaires	Extra
1	**id**	int(11)			Non	*Aucune*		AUTO_INCREMENT
2	**e_mail**	varchar(255)	utf8_general_ci		Non	*Aucune*		
3	**contrat**	varchar(255)	utf8_general_ci		Non	*Aucune*		
4	**etat**	varchar(255)	utf8_general_ci		Non	*Aucune*		
5	**conditions_gene**	varchar(255)	utf8_general_ci		Non	*Aucune*		
6	**date_redaction**	datetime			Non	*Aucune*		

Id : nombre entier (jusqu'à 10^{11}) en clé primaire et auto incrémenté
e_mail : 255 caractères maximum encodés en utf8_general_ci
contrat : j'indique la mention contrat. Le but étant de savoir si l'utilisateur a fait un contrat ou un état des lieux ou les deux.
etat : j'indique la mention etat
conditions_generales : 255 caractères max encodés en utf8_general_ci.
date_redaction : champs au format date + heure a l'anglaise

Le résultat visualisé avec phpmyAdmin :

	id 1	e_mail	contrat	etat	conditions_gene	date_redaction
Modifier Copier Effacer	1	luzien@hotmail.fr	contrat	etat	on	2017-05-19 13:51:01

Table des membres

#	Nom	Type	Interclassement	Attributs	Null	Valeur par défaut	Commentaires	Extra
1	**id**	int(11)			Non	*Aucune*		AUTO_INCREMENT
2	**prenom**	varchar(255)	latin1_swedish_ci		Non	*Aucune*		
3	**nom**	varchar(255)	latin1_swedish_ci		Non	*Aucune*		
4	**sexe**	varchar(255)	latin1_swedish_ci		Non	*Aucune*		
5	**pseudo**	varchar(255)	latin1_swedish_ci		Non	*Aucune*		
6	**mdp**	varchar(255)	latin1_swedish_ci		Non	*Aucune*		
7	**email**	varchar(255)	latin1_swedish_ci		Non	*Aucune*		
8	**cg**	varchar(255)	latin1_swedish_ci		Non	*Aucune*		
9	**ins_volee**	varchar(255)	latin1_swedish_ci		Non	*Aucune*		
10	**date_inscription**	datetime			Non	*Aucune*		

id	prenom	nom	sexe	pseudo	mdp	email	cg	ins_volee	date_inscription
1	jean	BALANGUE	un homme	juannitoo	f1446596023b2b3453b2	luzien@hotmail.fr	on		2013-02-13 00:00:00

Dans cette table, on peut noter surtout l'interclassement en *swedish*, et je dois dire que je ne peux pas vous expliquer pourquoi ! C'était peut-être l'item déjà sélectionné par défaut ? Je pensais que j'étais sur du general_ci. Ce que je peux en dire, c'est que ça n'a pas posé de problème d'affichage.

Je vais aussi vous expliquer que le champ *ins_volee* signifie inscription à la volée, Je l'ai rajouté afin de pouvoir quantifier le nombre d'utilisateurs qui s'inscrivaient seulement quand ils avaient déjà rempli le formulaire.

On peut aussi noter que je hache le mot de passe de l'utilisateur avec la fonction SHA1.

Table du premier contrat

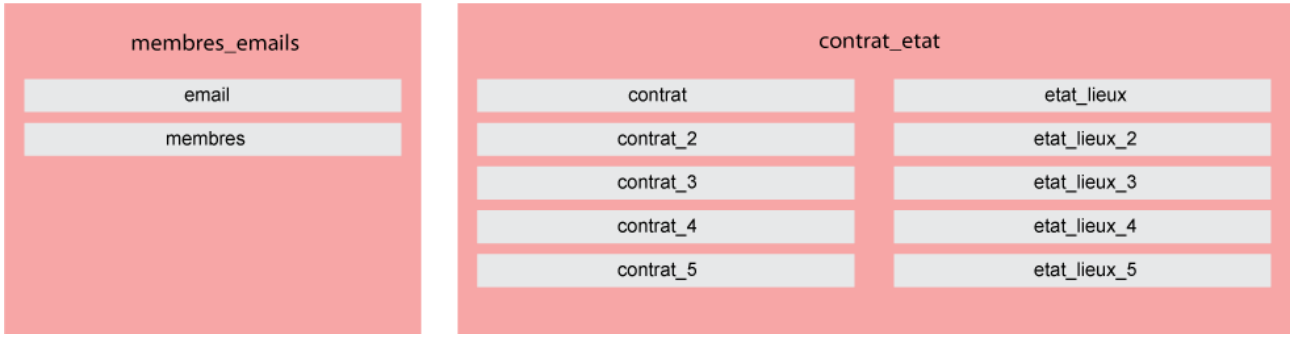
#	Nom	Type	Interclassement	Attributs	Null	Valeur par défaut	Commentaires	Extra	Ac
1	**pseudo**	varchar(255)	latin1_general_ci		Non	*Aucune*			
2	**nom**	text	latin1_general_ci		Non	*Aucune*			
3	**adresse**	text	latin1_general_ci		Non	*Aucune*			
4	**telephone**	varchar(150)	latin1_general_ci		Non	*Aucune*			
5	**e_mail**	varchar(255)	latin1_general_ci		Non	*Aucune*			
6	**adresse3**	text	latin1_general_ci		Non	*Aucune*			
7	**type_location**	varchar(150)	latin1_general_ci		Non	*Aucune*			
8	**superficie**	varchar(150)	latin1_general_ci		Non	*Aucune*			
9	**nbre_pieces**	varchar(150)	latin1_general_ci		Non	*Aucune*			
10	**nbre_pers**	varchar(150)	latin1_general_ci		Non	*Aucune*			
11	**fumeur**	varchar(150)	latin1_general_ci		Non	*Aucune*			
12	**animaux**	varchar(150)	latin1_general_ci		Non	*Aucune*			
13	**vue**	varchar(150)	latin1_general_ci		Non	*Aucune*			
14	**salon**	text	latin1_general_ci		Non	*Aucune*			
15	**cuisine**	text	latin1_general_ci		Non	*Aucune*			

Les autres tables de contrat et d'état des lieux sont construites sur le même modèles, elles comportent autant des champs que j'ai de variables à sauvegarder. 45 pour le contrat et 34 pour l'état des lieux. Je choisis les types en fonction de la longueur du texte à y insérer.

On peut noter que j'insère la variable de session *pseudo* dans la table en clé primaire, elle me sert d'une part à n'avoir qu'une entrée possible pour 1 pseudo, et aussi de lien entre mes tables.

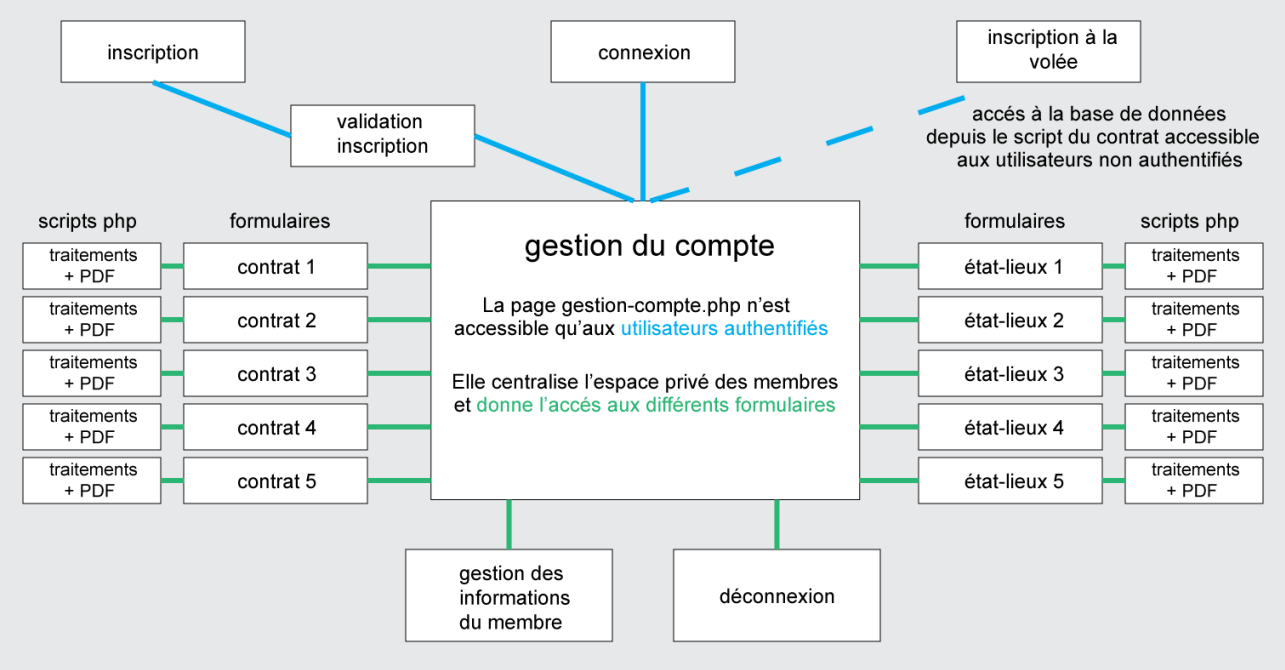
J'ai bien conscience maintenant que ce n'est pas bon. Mais quand on ne sait pas ...

voilà un schéma pour résumer :



2 - L’espace membre

En ce qui concerne le fonctionnement de l’espace membre, on trouve 2 points d’entrée pour accéder à un espace seulement accessible aux utilisateurs authentifiés. Ils peuvent depuis cet espace accéder aux différents formulaires qu’ils ont enregistrés.



Si vous rajouter « .php » à la fin des noms sur le schéma, vous aurez le fonctionnement.

A - Validation de l’inscription

Je ne suis pas en mesure de vous montrer le formulaire d’inscription, que j’ai bloqué et où j’affiche un message aux utilisateurs. Lors de la validation du formulaire d’’inscription, une fois les différents champs remplis et vérifiés en JavaScript, je redirige le futur membre vers ma page « validation-inscription.php » sur laquelle je fais mes différents traitements.

```php
//tests champs formulaire remplis
if (isset($_POST['sexe'])){
    if (!empty($_POST['nom'])){
        if (!empty($_POST['prenom'])){
            if (!empty($_POST['pseudo'])){
                if ((!empty($_POST['mdp'])) OR (!empty($_POST['mdp2']))){
                    if ($_POST['mdp'] == $_POST['mdp2']){
                        if (preg_match("#^[a-z0-9.-_]+@[a-z0-9.-_]{2,}\.[a-z]{2,4}$#" , $_POST['email'])){
                            if (isset($_POST['conditions_gene'])){

                                $nom = strtoupper(htmlspecialchars($_POST['nom']));
                                $prenom = strtolower(htmlspecialchars($_POST['prenom']));
                                $sexe = htmlspecialchars($_POST['sexe']);
                                $pseudo = htmlspecialchars($_POST['pseudo']);
                                $mot_de_passe = SHA1(htmlspecialchars($_POST['mdp']));
                                $password = htmlspecialchars($_POST['mdp']);
                                $e_mail = htmlspecialchars($_POST['email']);
                                $condition= htmlspecialchars($_POST['conditions_gene']);

                                echo '<h3>Votre formulaire d\'inscription a été correctement rempli.</h3>';

                                //requete1: vérification email deja connu
                                $verification_email= $bdd->prepare('SELECT * FROM membres WHERE email = ?');
                                $verification_email->execute(array($e_mail));
                                $donnee = $verification_email->fetch();
                                $verification_email-> closecursor();

                                //requete2: verification si pseudo libre
                                $verification_pseudo= $bdd->prepare('SELECT * FROM membres WHERE pseudo = ?');
                                $verification_pseudo->execute(array($pseudo));
                                $donnee2 = $verification_pseudo->fetch();
                                $verification_pseudo-> closecursor();

                                    if($donnee['email'] != $e_mail){
                                        if($donnee2['pseudo'] != $pseudo){

                                            //requete3: insertion membre
                                            $requete_insertion_membre= $bdd->prepare('INSERT INTO membres (prenom, nom, sexe, pseudo, m
                                                                                      VALUES (:prenom, :nom, :sexe, :pseudo, :mdp, :e
                                            $requete_insertion_membre->execute (array( 'prenom'=>$prenom,
                                                                                       'nom'=>$nom,
```

1- je vérifie les champs
2- je sécurise les variables
3- je vérifie la non présence de l'e-mail dans la table membre
4- je vérifie si le pseudo est libre
5- j'insère le membre dans la table
6- si la requête d'insertion s'est bien passée, j'envoie le mail de confirmation et je fais apparaître un lien de navigation à l'utilisateur pour qu'il poursuivre sa navigation sur la page de gestion de son compte.

```php
// vérification de la bonne insertion.  "si ca c mal passé"
if(!$requete_insertion_membre){
    die( 'les données n\'ont pas pu etre enregistrer. erreur de requete :'.mysql_error());
}else{
```

B - Le mail de validation d'inscription

Lors de l'envoi de mes mails d'inscription ou de récupération du mot de passe oublié, j'ai été confronté au problème des filtres spam, mes e-mails automatiques envoyés avec la fonction *mail()* finissaient dans les courriers indésirables, j'ai donc peaufiné mon mail jusqu'à qu'il passe la plupart des filtres de spam comme SpamAssasin. Découvrons le plus en détail.

Je commence par initialiser le passage à la ligne des différents fournisseurs de boites mails, Microsoft contre le reste du monde. Il faudrait peut-être rajouter Outlook maintenant

```php
//début de l'envoi du mail de confirmation

// on gère le saut à la ligne selon les serveurs
if (!preg_match("#^[a-z0-9._-]+@(hotmail|live|msn).[a-z]{2,4}$#", $e_mail)){
    $passage_ligne = "\r\n";
}else{
    $passage_ligne = "\n";
}
```

Ensuite, je crée le corps de mon message, d'abord sous format texte puis au format html. Je crée donc deux variables, *message_text* et *message_html* qui contiendront 1 format chacune.

```php
$message_text = 'Votre pseudo pour le site locationsaintjeandeluz.fr est : '.$pseudo.', votre mot de passe es
$message_html = "<html>";
$message_html.= "<head>";
$message_html.="</head>";
$message_html.="<body>";
$message_html.="<section style=\"background-color:rgb(235,235,235);\">";
$message_html.="<header>";
$message_html.="<p style=\"font-size:1.5em;\"><span class=\"head1\" style=\"color:orange;\">location</span><s
$message_html.="</header>";
$message_html.="<h1 style=\"font-size:1.2em;\">Vos identifiants de connexion :</h1>";
$message_html.="<p>Bonjour et bienvenue,</p>";
$message_html.="<p>Votre pseudo pour ce site est : <b>$pseudo</b> .</p>";
$message_html.="<p>Votre nouveau mot de passe pour ce site est : <b>$password</b> .</p>";
$message_html.="<p>Si vous avez des problèmes pour vous connecter, ne faîtes pas de copier-coller.</p>";
$message_html.="<p><a href='http://www.locationsaintjeandeluz.fr/contrat.html' title='créer mon contrat de lo
$message_html.="</section>";
$message_html.="</body></html>";
```

Après je crée les variables *sujet* du mail ainsi que la frontière ou *boundary*. Cette frontière sert à délimiter les différents types d'informations envoyées dans le mail, par exemple, type texte, image, html… Elle doit toujours commencer par « -----= » suivi d'une chaîne de caractère. Pour cette chaîne, je choisi un nombre au hasard avec la fonction *rand()* que je hache avec la fonction *md5()*.
Voilà le résultat dans la source du mail : -------=207a70d3267ba997f67f1e461e8e3399

```php
$sujet="Identifiants de connexion";
$boundary = "------=".md5(rand());
```

Enfin, je finis le création de mes variables en finissant avec les headers qui indiquent entre autres l'expéditeur, l'adresse de réponse, ou encore le type mime. Le type mime est un identifiant pour définir le types de données que l'on envoie sur le réseau. En le définissant sur multipart/alternative, j'autorise l'envoie de pièces jointes et l'utilisation des frontières.

```php
//création du header
$headers ='From: "info@locationsaintjeandeluz.fr"<info@locationsaintjeandeluz.fr>'.$passage_ligne;
$headers.="X-Sender: <www.locationsaintjeandeluz.fr>".$passage_ligne;
$headers.="X-Mailer: PHP".$passage_ligne;
$headers.="X-auth-smpt-user: locati26@fornost.planethoster.net".$passage_ligne;
$headers.="X-abuse-contact: webmaster@locationsaintjeandeluz.fr".$passage_ligne;
$headers.='Reply-To: info@locationsaintjeandeluz.fr'.$passage_ligne;
$headers.="MIME-Version: 1.0".$passage_ligne;
$headers.='Content-Type: Multipart/alternative;'.$passage_ligne." boundary=\"$boundary\"".$passage_ligne;
// -----fin header et début message
```

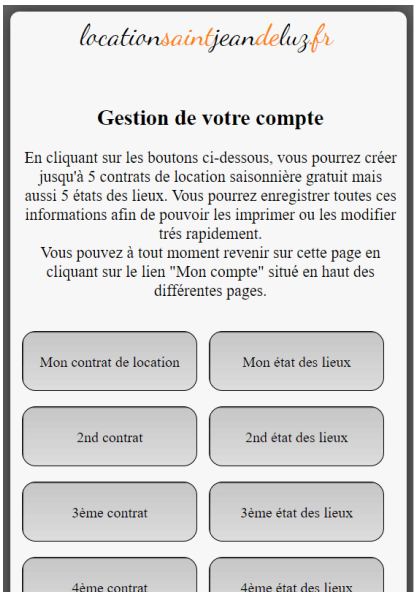
Une fois toutes mes variables crées, je n'ai plus qu'à faire le montage de mes variables *message_texte* et *message_html*, en respectant mes différents types de données et d'encodage, et en séparant mes blocs par des frontières.

```php
//ajout 1ère boundary (frontière)
$message = $passage_ligne."--".$boundary.$passage_ligne;

//=====Ajout message texte
$message.= "Content-Type: text/plain; charset=\"utf-8\"".$passage_ligne;
$message.= "Content-Transfer-Encoding: 8bit".$passage_ligne;
$message.= $passage_ligne.$message_text.$passage_ligne;

//ajout 2ème boundary (frontière)
$message.= $passage_ligne."--".$boundary.$passage_ligne;

//=====Ajout du message au format HTML
$message.= "Content-Type: text/html; charset=\"utf-8\"".$passage_ligne;
$message.= "Content-Transfer-Encoding: 8bit".$passage_ligne;
$message.= $passage_ligne.$message_html.$passage_ligne;

//fermeture boundary
$message.= $passage_ligne."--".$boundary."--".$passage_ligne;
$message.= $passage_ligne."--".$boundary."--".$passage_ligne;

mail($e_mail,$sujet,$message,$headers);
//-----fin du mail.
```

Comme des balises *script*, il est nécessaire de fermer les *bounderies* en rajoutant « -- » après *$boundary* pour signifier qu'elle sont fermées.

« Je ne suis pas en mesure à l'heure actuelle de vous montrer ce mail reçu par l'utilisateur, car j'ai bloqué les inscriptions. Vous en verrez un similaire un peu plus bas pour la réinitialisation du mot de passe. »

C - La page gestion-compte.php



Lors de la connexion ou l'inscription d'un membre, je le redirige vers la page « gestion-compte.php » sur laquelle je fais différents traitements :

1) j'initialise le système de session afin de faire suivre les informations de l'utilisateur sur toutes les pages.

2) je crée les variables de session de l'utilisateur, en récupérant soit ses identifiants d'inscription avec $_GET, soit ses identifiants de connexion avec $_POST (validation-inscription.php ou connexion.php). Si je n'obtiens pas de variables de sessions, je redirige vers la page d'accueil avec *header()*.

```php
<?php
session_start();
if(isset($_GET['pseudo'])){
    $_SESSION['login'] = htmlspecialchars($_GET['pseudo']);
}else{
    $_SESSION['login'] = htmlspecialchars($_POST['pseudo']);
}
if(isset($_GET['h'])){
    $_SESSION['hash'] = htmlspecialchars($_GET['h']);
}else{
    $_SESSION['hash'] = SHA1(htmlspecialchars($_POST['mdp']));
}
if (empty($_SESSION['login']) OR empty($_SESSION['hash'])){
header('Location:index.html');
}
?>
```

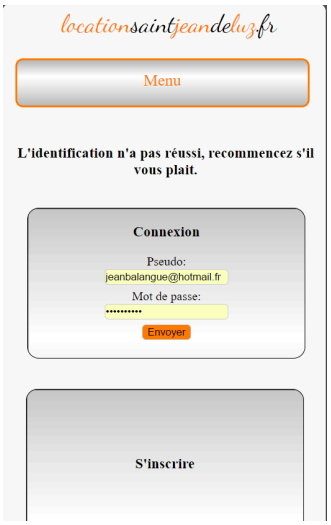
3) je me connecte à la base de données afin de comparer les identifiants récupérés précédemment à ceux stockés dans ma base de données. La connexion se fait avec l'objet PDO que nous avons vu dans l'appel Ajax. Il ne reste plus qu'à tester le mot de passe correspondant au pseudo pour savoir si j'ai bien à faire à un membre présent dans ma base de données.
Si tout va bien, j'affiche les différents liens permettant d'accéder aux différents formulaires.
Sinon, je lui affiche un message d'erreur avec un formulaire pour réessayer et un lien vers la page d'inscription.

```php
try{
    $bdd = new PDO('mysql:host=localhost; dbname=locati26_membres_emails','root','');
}catch(Exception $e){
die('Erreur: connection à la base de donnée impossible, réessayer s\'il vous plait. '.$e->getMessage());
}
$requete = $bdd->prepare('SELECT * FROM membres WHERE pseudo=:pseudo');
$requete->execute(array('pseudo'=> $login));

        $donnee=$requete->fetch();
            if($_SESSION['hash']==$donnee['mdp']){
                include("connexion.php");
                echo'<section><h1>Gestion de votre compte</h1><p>En cliquant sur les boutons ci-dessous, vous pourrez cré
                        mais aussi 5 états des lieux. Vous pourrez enregistrer toutes ces informations afin de pouvoir le
                        imprimer ou les modifier trés rapidement.<br />
                        Vous pouvez à tout moment revenir sur cette page en cliquant sur le lien "Mon compte" situé en ha
                    <article class="bouton"><a href="contrat-membre.php" title="accéder au contrat de location saisonnièr
                    </article>
                    <aside class="bouton"><a href="etat-lieux-membre.php" title="accéder a l\'état des lieux de location
                    </aside>
                    <article><a href="contrat-membre-2.php" title="accéder au contrat de location saisonnière">2nd contra
                    </article>
                    <aside><a href="etat-lieux-membre-2.php" title="accéder a l\'état des lieux de location saisonnière">
                    </aside>
                    <article><a href="contrat-membre-3.php" title="accéder au contrat de location saisonnière">3ème contr
                    </article>
                    <aside><a href="etat-lieux-membre-3.php" title="accéder a l\'état des lieux de location saisonnière">
```

Une partie du code qui affiche les liens vers les formulaires

```php
<h3>L\'identification n\'a pas réussi, recommencez s\'il vous plait.</h3>
    <article class="article-gestion">
        <form method="post" action="gestion-compte.php">
            <h3>Connexion</h3>
                <label>Pseudo:</label><br /><input type="text" name="pseudo" id="pseudo" size="20" /><br />
                <label>Mot de passe:</label><br /><input type="password" name="mdp" id="mdp" size="20" /><br />
                <input type="submit" class="valid-gestion" action="gestion-compte.php" value="Envoyer" />
        </form>
    </article>

    <aside class="aside-gestion">
        <h3 class="bouton-inscription"><a href="inscription.php" title="inscription" style="margin-top:-1em" >S\'inscrire</a>
    </aside>
    <br />
    <p style="text-align:center"><a href="aide.php" title="inscription" >Aide</a></p>
```

Mon message en cas d'erreur



L'affichage du message d'erreur

D - La page d'aide

C'est une page qui permet à l'utilisateur de me contacter au cas ou il aurait un problème, ou de réinitialiser son mot de passe dans le cas où il ne s'en souviendrait plus. Pour se faire, les utilisateurs ont accès à un formulaire ou l'utilisateur renseigne son adresse e-mail. Une fois le formulaire validé, j'effectue mon traitement sur la page php qui réinitialise le mot de passe et qui lui envoie un e-mail lui rappelant son pseudo et son nouveau mot de passe.

La page d'aide

```php
//création mdp aléatoire
$tableau = array('0','1','2','3','4','5','6','7','8','9','A','a','B','b','C','c','D','d','E','e','F','f',
    $valeurs_aleatoires=array_rand($tableau,8);
    $mdp='';
    foreach($valeurs_aleatoires as $i){
        $mdp= $mdp.$tableau[$i];
        $mdp_hash= SHA1($mdp);
    }
```

Je crée une suite de huit caractères que j'enverrai à l'utilisateur par mail et que je crypte pour l'insérer dans ma base de donnée. Pour cela je crée une variable *tableau* composée de minuscules,

majuscules et chiffres, j'utilise la fonction *array_rand()* pour sélectionner 8 caractères au hasard, un tableau des clés d'entrées de *$tableau* est retourné, il est donc nécessaire de passer par une boucle, plutôt que de répéter 8 fois l'instruction, pour créer le mot de passe. Je crée donc le nouveau mot de passe que j'envoie par mail, et son hash que j'insérerai dans la base de données. J'aurais néanmoins dû hacher le mot de passe après la boucle pour éviter d'appeler la fonction à chaque tour.

```
$requete_modif_mdp = $bdd->prepare('UPDATE membres SET mdp=:mdp WHERE email=:email');
$requete_modif_mdp->execute(array('email'=> $email,
                                  'mdp'=> $mdp_hash
                                  ));
$requete_modif_mdp->closecursor();
```

Nous trouvons la requête SQL de mise à jour du mot de passe dans la table membres. J'ai réalisé la connexion à la base de donnée précédemment avec pour variables le mot de passe (le hash) et l'e-mail de l'utilisateur. Enfin, j'envoie un mail à l'utilisateur contenant son nouveau mot de passe.

L'email reçu par l'utilisateur

E - L’accés des utilisateurs à leurs informations personnelles

Vous pouvez voir la page où je permet à l’utilisateur de modifier ses données personnelles, ou de modifier son compte. Il doit cliquer sur les liens prévu à cet effet en bas des cases, afin soit d’être redirigé vers la page de modifications de données personnelles, soit de supprimer son compte après une vérification Javascript.

```php
<?php
    try{
        $bdd = new PDO('mysql:host=localhost; dbname=locati26_membres_emails','root','');//host=localhost;dbname=te
    }catch(Exception $e){
        die('Erreur: Connection à la base de donnée impossible. Réessayer s\'il vous plait'.$e->getMessage());
    }


    $requete = $bdd->prepare('SELECT * FROM membres WHERE pseudo=:pseudo');
    $requete->execute(array('pseudo'=> $_SESSION['login']));
        while($donnee=$requete->fetch()){
         echo '<div class="apparent-mobile">
                    <p>Bienvenue '.$_SESSION['login'].'<br />
                    <a href="gestion-compte.php?pseudo='.$_SESSION['login'].'&h='.$_SESSION['hash'].'" alt="retour
                    <a href="deconnexion.php" alt="se déconnecter" class="compte">Déconnexion</a></p>
                </div>
                <h1>Informations sur votre compte</h1>
                <div class="article-info"><p>Vous etes : <strong>'.$donnee['prenom'].' '.$donnee['nom'].'</strong>.
                        <p>Vous etes <strong> '.$donnee['sexe'].'</strong>.</p>
                        <p>Votre pseudo est <strong>'.$donnee['pseudo'].'</strong>.</p>
                        <p>Votre e-mail de contact est <strong>'.$donnee['email'].'</strong></p>
                        <p><a href="modif-compte.php" alt="modification du compte" class="compte" >modifier mes don
                </div>
                <div class="aside-info"><p style="font-weight:bold">suppression du compte</p><p>Si vous cliquez sur
                            qui y sont liées. Votre état des lieux et votre contrat ne seront plus accéssibles et s
                            définitivement supprimés. Vous serez également supprimés de notre notre de fichier de m
                        <p><a href="suppression-compte.php" alt="suppression du compte utilisateur" class="compte"
                </div>
                ';
        }
?>
```

Le script de cette page.

Je sélectionne, avec Select, tous les éléments de la table Membres qui correspondent au pseudo désiré. J’affiche l’intégralité de ma page dans la boucle while{}.

J’ai surligné en bleu le lien de suppression du compte utilisateur. S’il clique dessus, je m’assure que ce n’est pas une erreur en javascript et je le renvoie vers une page de traitement php qui effectuera l’opération. Vous trouverez un bout du script en dessous.

```php
$requete=$bdd->prepare('DELETE FROM contrat WHERE pseudo=:pseudo');
$requete->execute(array('pseudo'=>$_SESSION['login']));
$requete->closecursor();
$requete2=$bdd->prepare('DELETE FROM etat_lieux WHERE pseudo=:pseudo');
$requete2->execute(array('pseudo'=>$_SESSION['login']));
$requete2->closecursor();
```

F - Modification de leurs données


```php
$requete_modif= $bdd->prepare('UPDATE membres SET prenom=:prenom, nom=:nom, sexe=:sexe, mdp=:mdp, email=:email WHERE id = :id');
$requete_modif-> execute(array('prenom' => $prenom,
                               'nom' => $nom,
                               'sexe' => $sexe,
                               'mdp' => $mot_de_passe,
                               'email' => $e_mail,
                               'id'=> $_SESSION['id']
                               ));
$donnee= $requete_modif->fetch();
$requete_modif-> closecursor();
```

Le script de la mise à jour du membre reprend dans les grandes lignes le script d'inscription. On notera juste que je n'autorise pas la modification du pseudo vu que je m'en sers d'identifiant dans ma base de données des contrats et états des lieux (clé primaire).

3 – Le contrat de location saisonnière

Le but de cette page web est donc de permettre à n'importe quel utilisateur de générer un contrat de location saisonnière et de l'enregistrer sur le site. Il pourra donc le retrouver pour l'éditer facilement. Pour générer ce PDF, je me sers de la classe PHP FPDF.

A – La classe FPDF

FPDF est une classe PHP qui permet de générer des fichiers PDF en pur PHP. Le F de FPDF signifie Free : l'utilisateur est libre de l'utiliser et de la modifier comme il le souhaite.

Une fois cette classe insérer dans le projet, on y accède de la même manière qu'avec une classe classique. Voici un petit exemple détaillé.

```php
<?php
require('fpdf.php');

$pdf = new FPDF();
$pdf->AddPage();
$pdf->SetFont('Arial','B',16);
$pdf->Cell(40,10,'Hello World !');
$pdf->Output();
?>
```

1) On insère la librairie fpdf

2) On crée un objet $pdf par l'intermédiaire du constructeur ***new FPDF()*** avec les valeurs par défaut c'est à dire en mode portrait, avec millimètres comme unité, et A4 comme format (*new FPDF('P','mm','A4');*).

3) On ajoute une nouvelle page à notre document avec ***AddPage()***.

4) On détermine la police de texte utilisée ici arial, en gras (BOLD), et en taille 16 avec ***SetFont()***.

5) On utilise ici la fonction ***Cell()*** qui crée une zone rectangulaire, encadrée ou non, ou l'on insérera notre texte. Les paramètres sont, dans l'ordre, 40 millimètres de long, 10mm de haut, le texte à afficher. D'autres paramètres aurait pu être passé, avec pour valeur 1, pour encadrer le texte d'une bordure., ou encore pour le positionnement du texte dans sa zone, par exemple en position « droite » avec R pour *rigth* ou C pour *center*.

6) Le document est terminé et envoyé au navigateur grâce à la fonction ***Output().***

retour sur la fonction C**ell() :**

cell (largeur cellule, hauteur de la ligne, monTexte, bordure, retour ligne, centrage du texte) ;

largeur : nombre entier; hauteur : nombre entier; monTexte : string ;

bordure : 0 ou 1 pour sans ou avec bordure;

retour à la ligne : 0, 1 ;

0 place le « curseur » à droite de l'élément courant

1 place le « curseur » en dessous de l'élément courant

centrage : L, C ou R pour Left, Center, ou Right

ex : ***Cell(100,7,'Contrat de location saisonnière',0,1,'C');*** représente le titre du PDF

Dans la création de mon PDF, j'utilise en plus la fonction ***multicell()*** qui me permet de définir des zones de texte avec retour à la ligne automatique une fois arriver au bord droit de la cellule, ceci est adapté aux *textarea*.

Multicell (largeur, hauteur, monTexte, bordure, saut de ligne, centrage du texte);

ex : ***MultiCell (89,3.5,$txt,0,1,C);***

J'utilise aussi les fonctions *setX(), setY()* et *setXY()* avec comme paramètre les coordonnées désirés en mm afin de placer mes différents éléments sur la page. Le point d'origine étant le bord haut gauche du document. L'axe Y est donc inversé.

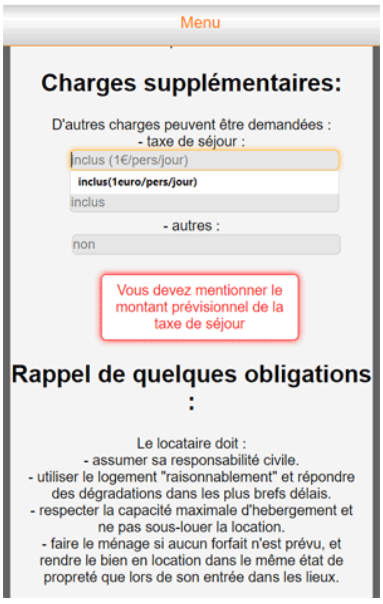
setXY (10,34) ; déplace mon élément de 10mm vers la droite et de 34mm vers le bas.
Enfin la fonction ***In()*** sert de saut de ligne. ***Ln(1)*** saute 1 ligne,

Il est bon de préciser que dans le cas où le PDF est envoyé au navigateur, le script ne doit rien envoyer d'autre, ni avant ni après, pas d'HTML, même pas un espace ni un retour-chariot. Sinon, on obtiendra le message : "Some data has already been output, can't send PDF file".

B – Interface utilisateur

Je souhaitais que l'utilisateur visualise le contrat sous sa forme finale et j'ai donc tous les champs possibles directement sur la page web. Je l'assiste également lorsqu'il clique sur certains champs. Et je leur permet aussi de pouvoir changer la taille de la police, et implicitement l'espacement de lignes, afin de pouvoir satisfaire le plus de cas possible.
Un appartement studio nécessite plus de texte à insérer dans la case salon qu'un F2 vu que la chambre et le salon sont la même pièce. De plus tout les propriétaires n'amènent pas la même prestations en terme de qualité tant dans l'habitation que dans la rédaction de leur contrat.
Je ne limite pas le nombre de caractères pour chaque champs de type *text* ou *textarea* et laisse gérer ça à l'utilisateur afin que celui qui souhaite un peu plus de place ne soit pas limité. La mise en page sera perturbée mais l'utilisateur aura un contrat avec les mentions qui lui conviennent. Il peut de toute manière générer autant de contrat qu'il le souhaite d'affilé.L'utilisateur peut donc visualiser son contrat à n'importe quel moment lors de la rédaction.

Vous trouverez une réalisation en annexe.



Pour les membres, je teste si des données sont présentes dans la table correspondante. S'il y en a, je les récupère et je les place directement dans le code html, pour les input de type text dans l'attribut value et pour les textarea directement entre les balises. Ainsi ils retrouvent leur formulaire déjà rempli et non plus qu'à inscrire les renseignements du locataire et les dates avant de cliquer sur un seul bouton pour enregistrer et générer le PDF en 1 seul temps. Les propriétaires gagnent beaucoup de temps dans la gestion de leur location avec ce service.

Voilà un bout de code dans une balise *textarea* qui permet d'afficher le nom du propriétaire si ce n'est pas sa première connexion :

```
Mr DUPONT Henry et Mme CLAIRY Marion" rows="3" cols="45"><?php if (isset($donnee['nom'])){ echo ($donnee['nom']);}?></textarea>
```

Ici dans un input de type text :

```
size="40" <?php if (isset($donnee['telephone'])){ echo 'value="'.stripslashes($donnee['telephone']).'"';}?> />
```

C – Script de génération du pdf

Le script fait 660 lignes, il m'est donc difficile de vous en faire des capture d'écrans, je vous propose de suivre plutôt les grandes étapes pas à pas.

a) la déclaration du document et des variables :

```php
<?php
session_start();
header('Content-Type: text/html; charset=ISO-8859-15');
require('fpdf/fpdf.php');
    $nom= str_replace("¤","EUR",htmlspecialchars($_POST['nom'], ENT_IGNORE,'ISO-8859-15'));
    $adresse= str_replace("¤","EUR",htmlspecialchars($_POST['adresse'], ENT_IGNORE,'ISO-8859-15'));
    $telephone= str_replace("¤","EUR",htmlspecialchars($_POST['telephone'], ENT_IGNORE,'ISO-8859-15'));
    $e_mail= str_replace("¤","EUR",htmlspecialchars($_POST['e_mail'], ENT_IGNORE,'ISO-8859-15'));
```

- J'initialise le système de session.
- Je définis le type mime et l'encodage.
- J'inclus la classe FPDF.
- Je commence la déclaration de mes variables

b) Insertion dans la base de données:

je teste la présence du pseudo dans la table sur laquelle il travaille. Si le pseudo n'est pas présent j'insère les données avec l'instruction sql INSERT INTO, sinon je met les informations à jour avec l'instruction sql UPDATE.

```php
try{
catch (Exception $e){
// test  " si  pas login dans la base"
$requete=$bdd->prepare('SELECT * FROM contrat WHERE pseudo = :pseudo');
$requete->execute(array('pseudo' => $_SESSION['login']));
$donnee= $requete->fetch();
//soit j'insère, soit j'update
if (!isset($donnee['pseudo'])){
$requete->closecursor();
// requete insertion bdd
// requete insertion
$requete2=$bdd->prepare('INSERT INTO contrat( pseudo, nom, adresse, telephone, e_mail, adresse3,
                                              type_location, superficie, nbre_pieces, nbre_pers, fu
                                              sanitaires, autres, date1, date2, date3, date4, prix,
                                              condition_paiement, taxe, gaz, autre_taxe, clef, mena
                                              tp1, tp2, tp3, tp4, tp5, tp6, tp7, tp8)
                         VALUES( :pseudo, :nom, :adresse, :telephone, :e_mail, :adresse3,
                                              :type_location, :superficie, :nbre_pieces, :nbre_pers
                                              :sanitaires, :autres, :date1, :date2, :date3, :date4,
                                              :condition_paiement, :taxe, :gaz, :autre_taxe, :ville
                                              :tp1, :tp2, :tp3, :tp4, :tp5, :tp6, :tp7, :tp8)');
$requete2->execute(array('pseudo' => $_SESSION['login'],
                         'nom' => $nom,
                         'adresse' => $adresse,
                         'telephone' => $telephone,
                         'e_mail' => $e_mail,
                         'adresse3' => $adresse3,
                         'type_location'=> $type_location,
                         'superficie'=> $superficie,
                         'nbre_pieces'=>$nbre_pieces,
                         'nbre_pers'=> $nbre_pers,
                         'fumeur'=> $fumeur,
                         'animaux'=> $animaux,
                         'vue'=> $vue,
                         'salon' => $salon,
                         'cuisine' => $cuisine,
                         'chambres' => $chambres,
```

c) création du PDF :

Je commence par créer une classe PDF qui hérite de la classe FPDF et j'utilise les fonctions natives à la classe FPDF ***header()*** et ***footer()*** qui définiront ces deux parties du documents.

Je crée ensuite ma première fonction qui se contente d'écrire le mot « entre » juste en dessous du titre (pour ceux qui se posent la question, « entre » les propriétaires et les locataires).

```
//
//début de la création du pdf
//
class PDF extends FPDF{
function header(){
    $this->SetFont('Arial','U',18);
    $this->Cell(47);  //largeur totale du doc : 195
    $this->Cell(100,7,'Contrat de location saisonnière',0,1,'C');
    $this->Ln(3);
    }
function footer(){
    $this->SetY(-12);
    $this->SetFont('Arial','',8);
    $this->Cell(60,3,'Mis à disposition gratuitement par locationsaintjeandeluz.fr',0,0,'L');
    $this->Cell(60,3,'Paraphes',0,0,'R');
    $this->Cell(60,3,'Page '.$this->PageNo().'/2',0,0,'R');
    }
//"entre"
//titre description
function entre($txt){
    $this->SetFont('Arial','B',8);
    $this->Cell(190,3,$txt,0,1,'C');
    }
```

Nous allons maintenant nous focaliser sur le premier bloc en haut à gauche du contrat, celui du ou des propriétaires, et suivre le script par son intermédiaire. Tout les encadrés présents sur le contrat final sont traités de la même manière. Les propriétaires sont définis par un nom, une adresse, un numéro de téléphone et un e-mail, leur cadre par une position. Il faut aussi penser aux intitulés des différents champs.

```
function info_proprio($txt1,$txt2,$txt3,$txt4){
    $this->intit_proprio('Le(s) propriétaire(s) dénommé "le propriétaire" :');
    $this->nom_proprio($txt1);
    $this->intit_adresse('Adresse :');
    $this->adresse($txt2);
    $this->intit_telephone('Téléphone :');
    $this->telephone($txt3);
    $this->intit_email('E-mail :');
    $this->email($txt4);
    $this->Rect(10,28,90,60);
    }
    $this->SetFont('Arial','B',9);
    $this->Cell(89,3.5,$txt,0,1);
    }
function adresse($txt){
        $this->SetFont('Arial','',$_POST['taille_police_1']);
        $this->MultiCell(89,3.5,$txt,0,1);
    }
function intit_telephone($txt){
    $this->SetY(72.5);
    $this->SetFont('Arial','B',9);
    $this->Cell(89,3,$txt,0,1);
    }
function telephone($txt){
    $this->SetFont('Arial','',10);
    $this->Cell(89,5,$txt,0,1);
    }
function intit_email($txt){
    $this->SetFont('Arial','B',9);
    $this->Cell(89,2.5,$txt,0,1);
    }
function email($txt){
    $this->SetFont('Arial','',10);
    $this->Cell(89,3,$txt,0,1);
    }
```

J’ai commencé par définir les différents champs de texte avant de les réunir dans une seule fonction **info_proprio ()**, j’effectuerai son appel uniquement lorsque l’intégralité de mes champs sont définis. La fonction ***Rect ()*** trace un rectangle et a pour paramètre dans l’ordre, l’abscisse du point d’origine, l’ordonnée du point d’origine, la largeur et la hauteur.

Les paramètres de la fonction **info-proprio ()** seront donc les variables utilisateurs du formulaire récupérés avec la méthode POST.

```
615 // $$$$$$$$$$$$$$$$$$ début phase d'appel $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
616 $pdf = new PDF();
617 $pdf->AddPage();
618 //"entre"
619 $pdf->entre("entre");
620 // info du propriétaire
621 $pdf->info_proprio(stripslashes($nom),stripslashes($adresse),stripslashes($telephone),stripslashes($e_mail));
```

Ici, j’instancie un nouvel objet $pdf, à qui je passe les différents paramètres précédemment créés.

```
660 $pdf->output('contrat-location.pdf','I');
661 ?>
```

C’est la fonction ***output ()*** qui envoie le pdf au navigateur par l’intermédiaire du paramètre « I ». Avec ce paramètre, le visualiseur PDF du navigateur est utilisé par défaut, ou alors Adobe Reader prend le relais du navigateur. J’aurais pu utilisé le paramètre « D » pour forcer le téléchargement sans passer par le visualiseur, mais je préfère l’autre méthode. Le premier paramètre est le nom du PDF généré.

Et voilà, c’est fini pour la présentation de l’existant.

III - Problèmes et objectifs de maintenance

Après avoir analyser l'existant, nous allons nous intéresser aux problèmes rencontrés et nous définirons un plan de bataille pour les solutionner avec un ordre de priorité. Le but étant de remettre le service en route et de s'assurer d'une certaine pérennité sur le long terme.

De mon point de vue, ce qui me fait vous présenter ce site et le reprendre, c'est l'impossibilité pour un utilisateur lambda de pouvoir enregistrer son contrat de location sur le site et donc d'avoir un usage « normal » du site.

D'un point de vue technique, en un sens, ça m'arrange un peu d'avoir à reprendre ce site. Je vais pouvoir mettre en œuvre certaines de mes nouvelles compétences acquises durant de la formation et m'exercer grandeur nature.

1 – Rétablir un usage normal

C'est la priorité. Si l'enregistrement ne fonctionne plus, c'est uniquement parce que j'en ai bloqué l'accès, mais l'espace membre fonctionne correctement. J'ai été contraint de bloquer l'accès aux services en raison d'un problème d'encodage de caractères sur les PDF générés et en conséquence dans la base de données.

Revenons un peu sur la notion d'encodage de caractères :

C'est le fait d'associer le jeu de caractères de nos systèmes d'écritures à un jeu de caractères chiffrés, dans notre cas avec l'objectif qu'il soit compréhensible par l'ordinateur.

Les anglophones étant à l'origine de l'informatique, l'une des premières normes d'encodage a été l'ASCII, American Standard Code for Information Interchange. Apparue en 1960, elle ne comporte aucune lettre accentuée et ne permet pas d'écrire correctement, grammaticalement parlant, dans toutes les différentes langues.

L'organisation internationale de normalisation (ISO) a donc décidé différentes normes d'encodage propres à chaque pays. Pour la France, il s'agit de la norme ISO-8859-15. Le problème, c'est qu'avec l'internationalisation rendu possible par l'informatique, un utilisateur lambda ne pouvait pas toujours afficher correctement les pages encodées dans des encodages différents de son pays.

Une nouvelle norme a donc été créée officiellement en 2000, l'UTF-8, qui signifie Universal character set Transformation Format 8bits et qui résout ce problème.

Mon hébergeur a modifié l'encodage des connexions à la base de données de ses serveurs mutualisés, pour les passer d' ISO-8859-15 à UTF-8. En soi, c'est plutôt positif, sauf que la classe FPDF ne prend en charge que l'encodage en ISO, et que l'intégralité des documents créés produisaient des erreurs. Pour couronner le tout, je permet, en cliquant sur un seul bouton, de créer le PDF et d'insérer ou mettre à jour les données dans la base. En clair, chaque fois qu'un utilisateur génère son contrat, il corrompt ses données. J'ai bien tenté les fonctions utf-8 encode() et décode(), mais ça n'a pas fonctionné.

Pour préserver l'intégrité de la base de données, j'ai préféré bloquer l'accès à l'espace membre.

Il me reste donc à déterminer la solution la plus adaptée à mon problème et à voir s'il est vraiment dans mon intérêt de remettre le service en ligne avant de le structurer.

2 – Structurer le back-end

On récapitule pour ceux qui ont sauté des pages, 2 bases de données, presque tous les fichiers dans la racine du site et 12 formulaires avec presque autant de tables SQL pour deux structures de PDF. Le génie informatique en somme !

Plus sérieusement, Je souhaite revoir l'intégralité du site, mais certainement pas avec vous. Je souhaite néanmoins conserver l'expérience utilisateur, c'est à dire conserver la possibilité offerte à l'utilisateur de générer un contrat sans s'enregistrer en tant que membre, mais aussi la visibilité de l'intégralité du formulaire du contrat sur la page HTML, sans avoir à cliquer pour savoir ce que l'on va faire. Je souhaite aussi conserver la fonctionnalité en 1 clic, PDF-enregistrement.

A – L'architecture de la base de données :

Je vais avoir des cours de SQL avec Merise et UML et je vais donc avoir l'opportunité de m'entraîner. Pourquoi pas une seule base de données avec des jointures et des id? Soyons fous ! Je dois quand même penser à toutes ces données que j'ai en *latin* dans ma base pour des scripts et des pages qui risquent d'être en UTF-8.

B – L'arborescence des fichiers :

Alors là, c'est simple, il faut tout revoir ! Je souhaite créé plusieurs templates, un pour l'index et les pages d'erreurs, un pour les locations, un autre pour les contrats et encore un autre pour l'état des lieux. Je souhaite également mieux structurer mon code en segmentant le Html du Javascript et le plus possible du Php. Je vais m'inspirer d'express.js et d'angular, le but étant une maintenance ultérieure plus facile.

C – Différentes opérations supplémentaires à faire :

Segmenter mes fichiers, utiliser classes et objets, rectifier les conditions « if » pour renforcer la sécurité tant pour l'accès à la base de données que pour les différentes variables, vérifier la longueur des champs attendus, voir le type de données, extraire mon script de connexion à la base et le placer hors du dossier « www ». Je dois également me renseigner sur le protocole https pour savoir comment l'implémenter, savoir si je dois commencer par ça ou pas…. Et le reste auquel je ne pense pas aujourd'hui comme le salage du mot de passe.

En résumé, je souhaite avoir une approche plus professionnelle et plus seulement la problématique : « ça marche ou pas ».

IV – Réalisations des opérations de maintenances

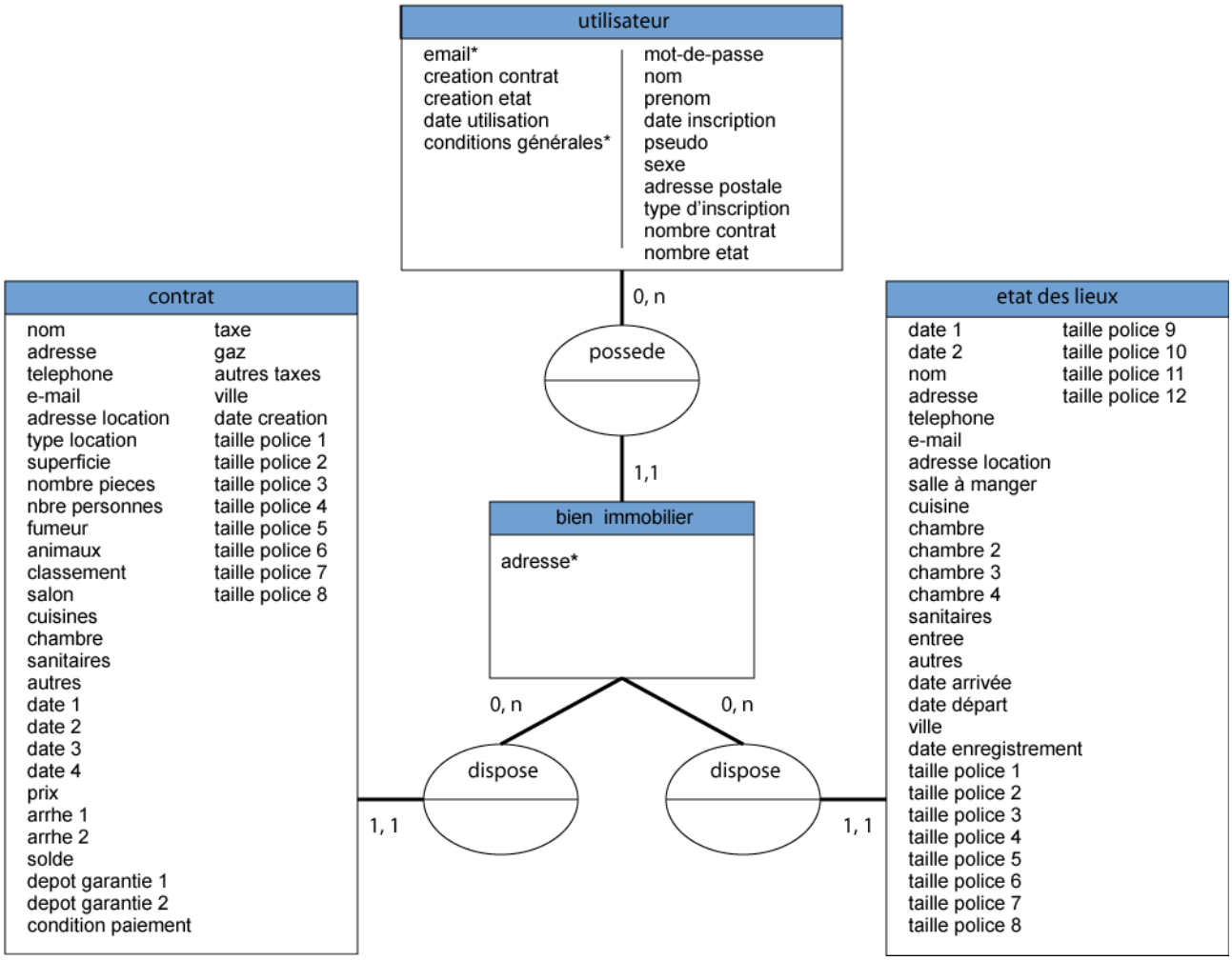
Après analyse des problèmes le temps de préparer cette synthèse, j'ai décidé de redémarrer le back-end depuis zéro. Afin de générer mes PDF, j'ai l'intention de me servir d'une nouvelle classe Php, qui se nomme TFPDF et qui supporte l'utf-8. Je l'ai trouvé sur le site du créateur de la classe originale. Afin de fonctionner, nous devons ajouter des polices unicode dans le dossier mentionné et le tour est joué. Je n'ai pas besoin d'écrire un nouveau script spécifique. Je passerai en https lors de la remise en service de la base de données, mon hébergeur me fournissant le certificat. Je commence donc par la première étape: la refonte de la base de données.

1 – La base de données

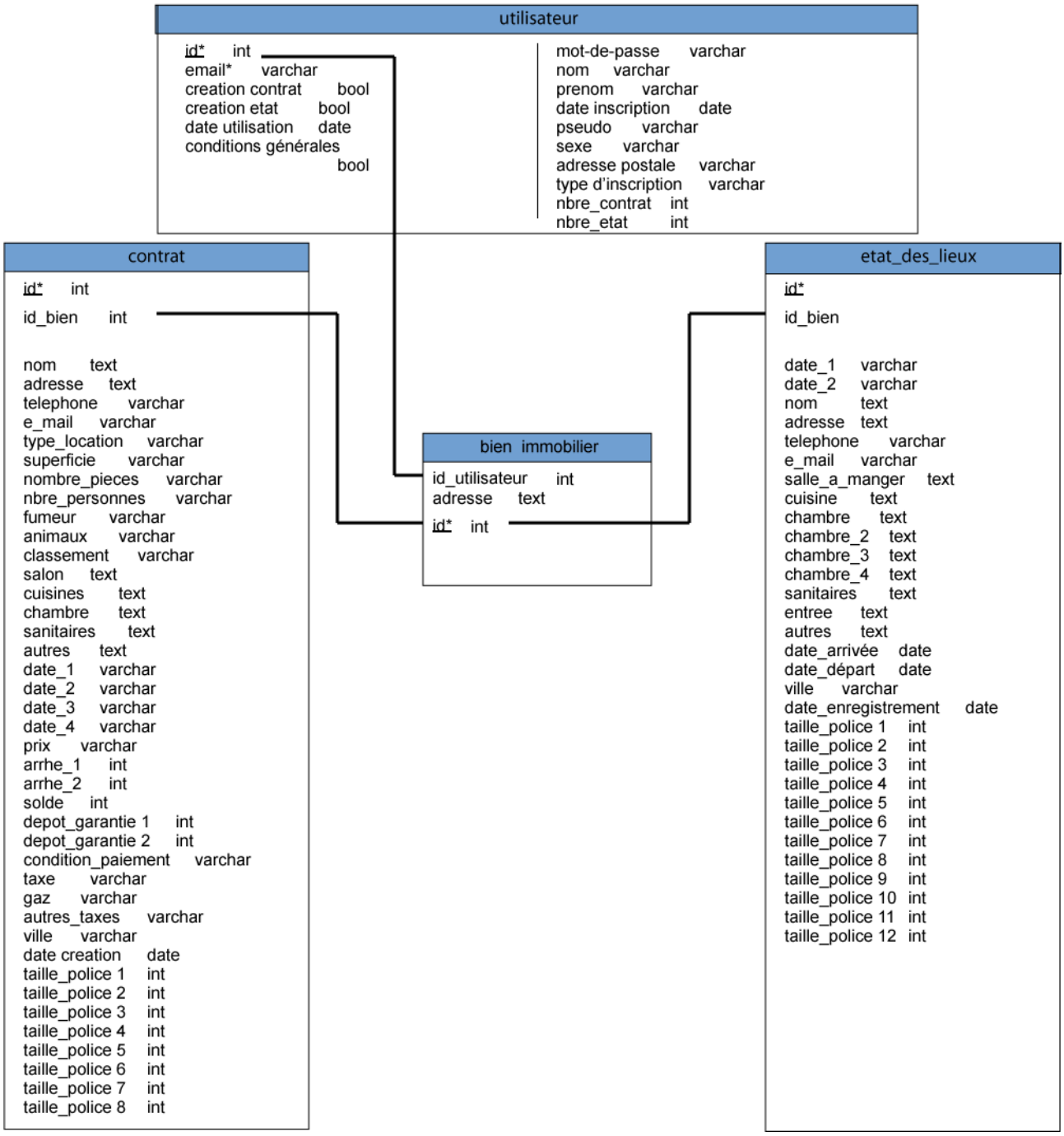
A - Le modèle conceptuel de données :

Après mes premiers cours de SQL ainsi que les conseils de mon formateur et de mes camarades, j'ai pu élaborer le Modèle Conceptuel de Données de la nouvelle base. Vous remarquerez que l'entité utilisateur est segmentée en deux colonnes. Cette future table enregistrera l'e-mail des utilisateurs qui ne souhaitent pas enregistrer leur contrat et les membres qui auront donc des propriétés supplémentaires.

Cette segmentation vient du fait que je souhaite décompter les différents utilisateurs qui ne s'enregistrent pas mais qui créent quand même un contrat. J'efface ces e-mails au bout d'un an maximum.



B – Le modèle logique de données



C - le script de création de la base et des tables :

Ici, un premier problème se pose, j'ai déjà une base de données remplie de données. Outre l'encodage des caractères, j'ai déjà défini les différents types de mes différents champs. Je me pose donc la question de savoir si je dois garder les types existants ou si je peux les modifier/affiner.

J'écris donc le genre de requêtes ci-dessous dans l'onglet SQL de PhpMyAdmin pour tester les champs de la base existante et me faire une idée de ce qu'il est possible de faire. Je décide ensuite pour chaque champ quel type choisir selon les données présentes et la place disponible sur le PDF. Je me servirai de ces différentes valeurs pour faire de nouvelles vérifications sur les variables utilisateurs, en Php et Javascript.

```sql
SELECT nbre_pers, LENGTH(nbre_pers)
FROM contrat
ORDER BY LENGTH(nbre_pers) DESC
```

nbre_pers	LENGTH(nbre_pers) ▼ 1
2 +prêt lit bébé ou chauffeuse 2 pl	35
3 adultes ou 2 adultes + 2 enfants	34
2 (si plus, supplément demandé).	32
3 (dont 1 lit d'appoint)	24

Le script de création de la base et des tables « utilisateur » et « bien_immo » :

```sql
DROP DATABASE IF EXISTS location_st_jean;
CREATE DATABASE location_st_jean CHARACTER SET 'utf8';
USE location_st_jean;

CREATE TABLE utilisateur (
    id INT AUTO_INCREMENT,
    email VARCHAR(70) NOT NULL,
    creation_contrat BOOLEAN,
    creation_etat BOOLEAN,
    date_utilisation DATETIME NOT NULL,
    condition_gene BOOLEAN,
    mdp VARCHAR(255),
    nom VARCHAR(50),
    prenom VARCHAR (50),
    pseudo VARCHAR (50),
    sexe VARCHAR (10),
    date_inscription DATETIME,
    type_inscription VARCHAR(20),
    nbre_contrat_membre TINYINT,
    nbre_etat_membre TINYINT,
    PRIMARY KEY(id)
)ENGINE=InnoDB;

CREATE TABLE bien_immo (
    id INT  AUTO_INCREMENT,
    adresse TEXT NOT NULL,
    id_utilisateur INT NOT NULL REFERENCES utilisateur(id),
    PRIMARY KEY(id)
)ENGINE=InnoDB;
-- CONSTRAINT fk_id_utilisateur FOREIGN KEY id_utilisateur REFERENCES utilisateur(id)
```

J'aurais pu écrire la contrainte d'intégrité référentielle comme ceci (à l'intérieur du create table) mais je préfère la façon raccourci.

La suite du script de création de la base avec la table des contrats et et celle des état des lieux.

```sql
CREATE TABLE contrat (
    id INT AUTO_INCREMENT,
    nom TEXT,
    adresse TEXT NOT NULL,
    telephone VARCHAR (50),
    e_mail VARCHAR(40),
    type_location VARCHAR (50),
    superficie VARCHAR (50),
    nbre_piece VARCHAR (50),
    nbre_pers VARCHAR(35),
    fumeur VARCHAR(50),
    animaux VARCHAR(50),
    classement VARCHAR(60),
    salon TEXT,
    cuisine TEXT,
    chambres TEXT,
    sanitaires TEXT,
    autres TEXT,
    date_1 VARCHAR(50),
    date_2 VARCHAR(20),
    date_3 VARCHAR(25),
    date_4 VARCHAR(35),
    prix VARCHAR(6),
    arrhes_1 VARCHAR(6),
    arrhes_2 VARCHAR(6),
    solde VARCHAR(6),
    depot_garantie_1 VARCHAR(6),
    depot_garantie_2 VARCHAR(6),
    condition_paiement VARCHAR(150),
    taxe VARCHAR(70),
    gaz VARCHAR(70),
    autres_taxes VARCHAR(130),
    ville VARCHAR(75),
    date_enregistrement DATETIME,
    tp1 INT,
    tp2 INT,
    tp3 INT,
    tp4 INT,
    tp5 INT,
    tp6 INT,
    tp7 INT,
    tp8 INT,
    id_bien INT NOT NULL REFERENCES bien_immo(id),
    PRIMARY KEY(id)
) ENGINE=InnoDB;
```

```sql
CREATE TABLE etat (
    id INT AUTO_INCREMENT,
    date_1 VARCHAR(70),
    date_2 VARCHAR(70),
    nom TEXT,
    adresse TEXT NOT NULL,
    telephone VARCHAR (50),
    e_mail VARCHAR(50),
    salon TEXT,
    cuisine TEXT,
    chambre TEXT,
    chambre_2 TEXT,
    chambre_3 TEXT,
    chambre_4 TEXT,
    sanitaires TEXT,
    entree TEXT,
    autres TEXT,
    date_arrivee TEXT,
    date_depart TEXT,
    ville VARCHAR(100),
    date_enregistrement DATETIME,
    tp1 INT,
    tp2 INT,
    tp3 INT,
    tp4 INT,
    tp5 INT,
    tp6 INT,
    tp7 INT,
    tp8 INT,
    tp9 INT,
    tp10 INT,
    tp11 INT,
    tp12 INT,
    id_bien INT NOT NULL REFERENCES bien_immo(id),
    PRIMARY KEY(id)
) ENGINE=InnoDB;
```

2 - Migration des données dans la nouvelle base

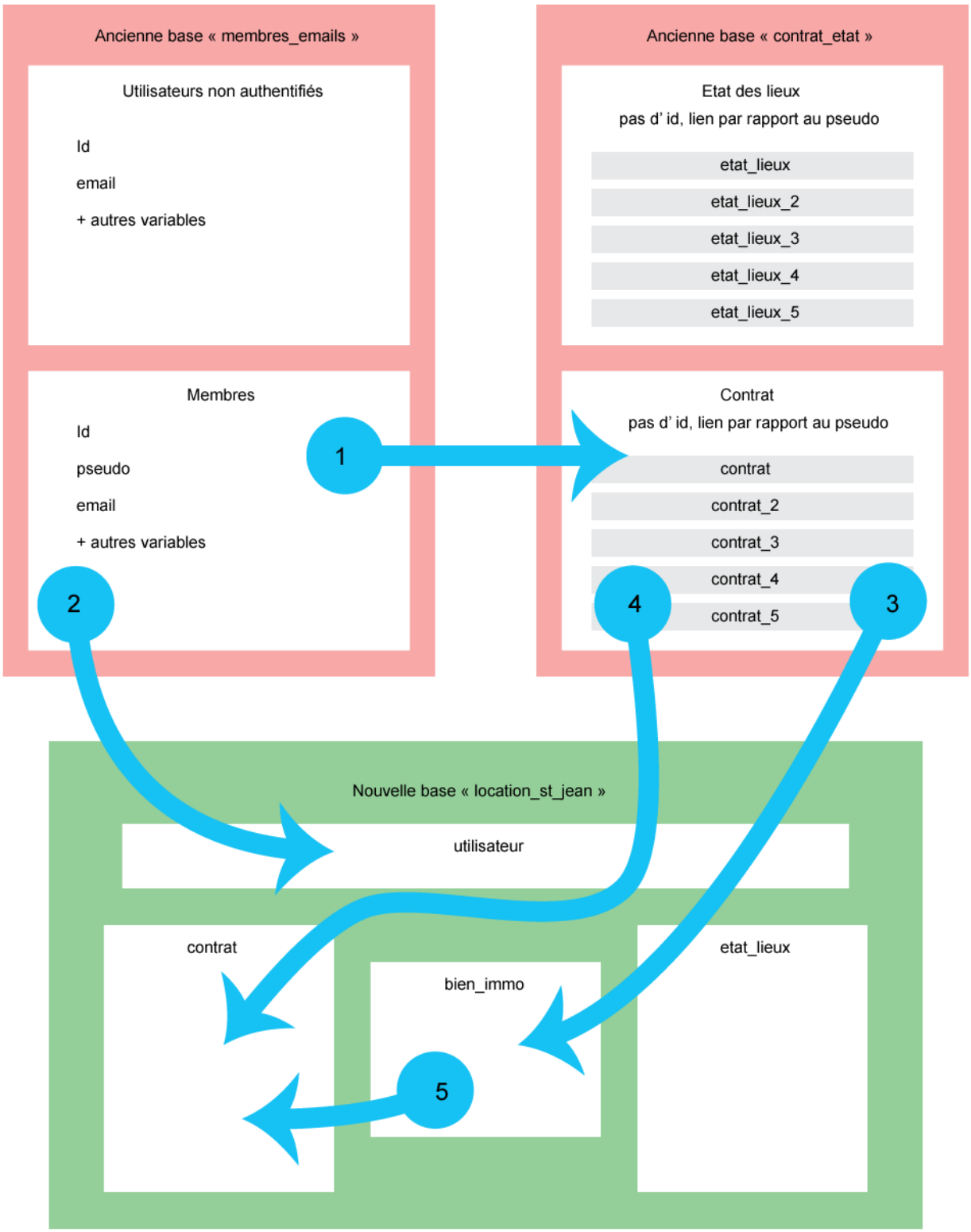
A - Script de migration

Ici, plusieurs problèmes se posent.

- J'ai des problèmes d'encodage mais aussi des nouvelles variables.- J'ai 2 bases de données distinctes en plus de la nouvelle.

- J'ai relié mes tables « contrat » existantes par le pseudo, ces tables ne contiennent donc pas d'id, il n'y a pas de test d'intégrité référentielle fait par le SGBDR et je n'ai pas de lien entre les différents contrats et leur membre respectif hormis le champs « pseudo » dans mes scripts existants.

Je vais dans un premier temps migrer les données des membres et des contrats. Vous trouverez à la page suivante un schéma pour vous en expliquer le déroulement.



1) Insérer l'id du membre dans les contrats existants, j'appelle ce champ « id_pseudo »
2) Insérer les infos des membres existants dans la table « utilisateur » (id du membre = id utilisateur).
3) Insérer « id_utilisateur » « adresse » dans la table bien_immo
4) Insertion des contrats dans la table « contrat » . J'insère également les variables « pseudo » et « id_pseudo » par rapport au mld, dans les tables pour le fonctionnement du script et pour faire des vérifications. Je les effacerai après.
5) Récupérer les id des bien_immo pour les associer à leur(s) contrat(s) respectif(s)

Détaillons le script point par point :

1)Insérer l'id du membre dans les contrats existants, j'appelle ce champ « id_pseudo »

Afin de pouvoir recréer le lien d'association entre, d'une part un contrat,et d'autre part l'id du pseudo du membre (2 champs contenus dans 2 bases distinctes), j'ai choisi d'insérer une colonne *id_pseudo* dans les tables « contrat » et « etat » existantes.

J'ai besoin de ce champ numérique « id_pseudo » qui contiendra l'ID du membre, je l'ai nommé ainsi pour éviter tout conflit (id qui correspond au pseudo). Il me servira à faire le lien avec les ID des nouvelles tables « bien_immo » , « contrat » , « etat » et donc forcement « utilisateur ».

Le script est un peu complexe et m'a donné du fil à retordre, je dois associer un Id à un pseudo contenus dans 2 bases différentes.

Dans un premier temps, je crée un champ dans ma table contrat

```
ALTER TABLE `contrat` ADD `id_pseudo` INT NOT NULL FIRST;
```

Ensuite, j'initialise les connections PDO et 2 tableaux qui contiendront les résultats de mes requêtes sql.

```
$bdd2 = new PDO('mysql:host=localhost;dbname=location_st_jean','root','');
$bdd3 = new PDO('mysql:host=localhost;dbname=locati26_membres_emails','root','');
$bdd4 = new PDO('mysql:host=localhost;dbname=locati26_contrat_etat','root','');
```

```
//récupération des variables ID et pseudo de la base membres_email
$requete_id_pseudo_membre = $bdd3->query('SELECT id, pseudo FROM membres');

//recupération du pseudo uilisé pour les contrats
$requete_pseudo_contrat = $bdd4->query('SELECT pseudo FROM contrat');


//je crée un tableau associatif vide ou je recupère les id et pseudo de la requete "membres"
$tableau_id_pseudo = array();
while($reponse_id_pseudo_membre = $requete_id_pseudo_membre->fetch()){
    //concaténation tableau = tableau + tableau
    $tableau_id_pseudo[$reponse_id_pseudo_membre['id']] = $reponse_id_pseudo_membre['pseudo'];
}

//je crée un tableau itératif vide ou je recupère les pseudo de la requete "contrat"
$tableau_pseudo_contrat=array();
while($reponse_pseudo_contrat = $requete_pseudo_contrat->fetch()){
    array_push($tableau_pseudo_contrat, $reponse_pseudo_contrat['pseudo']);
}
```

Enfin, je fais une boucle dans une boucle pour associer les jeux de résultats. Je teste l'égalité du pseudo du membre à celui du contrat, s'il y a correspondance, je mets à jour la table.

```
//j'associe mes variables et j'insère dans la base
foreach ($tableau_id_pseudo as $key => $value){

    for ($i = 0; $i<count($tableau_pseudo_contrat); $i++){
        if ( $value == $tableau_pseudo_contrat[$i] ){
            //echo $key .' '.$value.' '.$tableau_pseudo_contrat[$i].'<br />';
            $req = $bdd5->prepare('UPDATE contrat SET id_pseudo = :id WHERE pseudo = :pseudo');
            $req->execute(array(
                'id' => $key,
                'pseudo' => $value
            ));
        }
    }
}

$requete_id_pseudo_membre -> closeCursor();
$requete_pseudo_contrat -> closeCursor();
$req -> closeCursor();
```

2) Insérer les infos des membres existants dans la table « utilisateur » (id_membre = id_ utilisateur)

Ici, c'est beaucoup plus simple, je récupère les données et je les insère dans leur champs respectifs. On voit que j'initialise le nouveau champ « type_inscription ».

```php
//   --------   requete recuperation infos membres  -------------------
$requete_recup = $bdd->query('SELECT * FROM membres');

// boucle pour lire et insérer les données une à une dans la nouvelle table contrat
while ($donnees = $requete_recup->fetch()){

    $id = $donnees['id'];
    $email = $donnees['email'];
    $date_utilisation = $donnees['date_inscription'];
    $condition_gene = true;
    $mdp  = $donnees['mdp'];
    $nom  = $donnees['nom'];
    $prenom  = $donnees['prenom'];
    $pseudo = $donnees['pseudo'];
    $sexe = $donnees['sexe'];

    if ($donnees['ins_volee'] == 'oui'){
        $type_inscription = 'ins_volee';
    }else{
        $type_inscription = 'normale';
    }

    $requete_insertion = $bdd2->prepare('INSERT INTO utilisateur(id, email, date_utilisation, condition_gene, md
                                        VALUES (:id, :email, :date_utilisation, :condition_gene, :mdp, :nom,
    $requete_insertion->execute(array('id' => $id,
                                      'email' => $email,
                                      'date_utilisation' => $date_utilisation,
                                      'condition_gene' => $condition_gene,
                                      'mdp' => $mdp,
                                      'nom' => $nom,
                                      'prenom' => $prenom,
                                      'pseudo' => $pseudo,
                                      'sexe' => $sexe,
                                      'type_inscription' => $type_inscription
                                    ));
}
$requete_recup->closeCursor();
```

3) Insérer « id_utilisateur » « adresse » dans la table bien_immo

Ici c'est comme précédemment, je sélectionne mes entrées et je les insère. On peut noter que j'initialise l'adresse à « non rempli » si le champs est vide, je traiterai ce genre d'erreur après.

```php
// insertion adresses  id_pseudo   pseudo
$req_recup_contrat = $bdd2->query('SELECT id_pseudo, pseudo, adresse3
                                   FROM contrat
                                   ');

while($reponse = $req_recup_contrat->fetch()){

    if($reponse['adresse3'] == NULL){
        $adresse = 'non rempli';
    }else{
        $adresse = $reponse['adresse3'];
    }

    echo  ' <b>'.$reponse['id_pseudo'].' '.$reponse['pseudo'].'</b> '.$adresse.'<br />';

    // insérer les adresses des 5 tables par id_pseudo
    $req_insertion_bien_immo = $bdd3->prepare('INSERT INTO bien_immo (pseudo, id_utilisateur, adresse)
                                              VALUES (:pseudo, :id, :adresse)');
    $req_insertion_bien_immo->execute(array('pseudo' => $reponse['pseudo'],
                                            'id' => $reponse['id_pseudo'],
                                            'adresse' => $adresse
                                            ));

}
```

4) Insertion des contrats dans la table « contrat »

Je sélectionne, j'insère. Vite dit mais moins vite fait, je dois vérifier les champs.

Pour vérifier que toutes mes entrées ont bien été transféré au bon endroit, je décompte le nombre d'entrée avant et après le transfert, et je m'en assure également visuellement via mon affichage et via phpMyAdmin. En travaillant table par table, et en commençant par la table avec le moins d'entrées, je peux identifier les lignes problématiques…

J'ai eu une dizaine de lignes qui ne s'inscrivaient pas du fait d'un nombre de caractères trop important pour le type varchar mentionné dans le script de création de la base. J'ai du élargir la longueur de certain champs de mon script.

Une fois les vérifications faites, je vide le contenu des nouvelles tables « biens_immo » et « contrat », via phpMyAdmin, pour tester la table ancienne suivante.

Voilà le script, on peut voir que j'initialise la valeur id_bien (contrainte d'intégrité referentielle) à 1 sinon le script ne marche pas.

```php
//requete recuperation contrat
$requete_recup = $bdd2->query('SELECT * FROM contrat');

while($donnees = $requete_recup->fetch()){

    //insertion contrat    id_bien = valeur par defaut
    $requete2=$bdd3->prepare('INSERT INTO contrat(id_bien, id_pseudo, pseudo, nom, adresse, telephone, e_mail,
                                type_location, superficie, nbre_piece, nbre_pers, fumeur,
                                animaux, classement, salon, cuisine, chambres, sanitaires, autres,
                                date_1, date_2, date_3, date_4, prix, arrhes_1, arrhes_2, solde,
                                depot_1, depot_2, condition_paiement, taxe, gaz, autres_taxes, ville,
                                date_enregistrement, tp1, tp2, tp3, tp4, tp5, tp6, tp7, tp8)
                            VALUES (:id_bien, :id_pseudo, :pseudo, :nom, :adresse, :telephone, :e_mail,
                                :type_location, :superficie, :nbre_piece, :nbre_pers, :fumeur,
                                :animaux, :classement, :salon, :cuisine, :chambres, :sanitaires, :autres,
                                :date_1, :date_2, :date_3, :date_4, :prix, :arrhes_1, :arrhes_2, :solde,
                                :depot_1, :depot_2, :condition_paiement, :taxe, :gaz, :autres_taxes, :ville,
                                :date_enregistrement, :tp1, :tp2, :tp3, :tp4, :tp5, :tp6, :tp7, :tp8)
                                ');
    //total 43 parametres
    $requete2->execute(array('id_bien' => 1,
                            'id_pseudo' => $donnees['id_pseudo'],
                            'pseudo' => $donnees['pseudo'],
                            'nom' => $donnees['nom'],
                            'adresse' => $donnees['adresse'],
                            'telephone' => $donnees['telephone'],
                            'e_mail' => $donnees['e_mail'],
                            'type_location' => $donnees['type_location'],
                            'superficie' => $donnees['superficie'],
                            'nbre_piece' => $donnees['nbre_pieces'],
                            'nbre_pers' => $donnees['nbre_pers'],
                            'fumeur' => $donnees['fumeur'],
                            'animaux' => $donnees['animaux'],
                            'classement' => $donnees['vue'],
```

5) Récupérer les id_pseudo des contrats pour les associer à bien_immo.id et à id_utilisateur

La dernière étape qui consiste à mettre à jour « id_bien » dans le contrat que je récupère dans la table bien_immo.

```php
// update  id_bien dans contrat
$req= $bdd4->exec('UPDATE contrat
                    INNER JOIN bien_immo ON id_pseudo = id_utilisateur
                    SET contrat.id_bien = bien_immo.id
                    ');
```

Et voilà, le script de migration est presque terminé, les données sont bien rangées à leur place. Reste à traiter les problèmes d'encodage des données et nettoyer les tables des données superflues qui m'ont servi pour mes vérifications.

B - Traitements des problèmes de données

- Pour la table des membres :

J'ai commencé par exporter mes bases depuis mon serveur pour avoir les versions de « production ». J'exporte ces bases en double, une en ISO et une en UTF8. Je choisirai selon le résultat visuel dans mes éditeurs. J'ai modifié les fichiers pour me permettre une importation rapide dans phpMyAdmin

DROP table IF EXISTS et **use old_loc** .

Ensuite j'ai essayé une commande de conversion d'encodage directement en sql via phpMyAdmin mais ça n'a pas fonctionné.

ALTER TABLE `contrat` CONVERT TO CHARACTER SET 'UTF8';

Enfin, j'ai essayé de convertir l'encodage des caractères directement dans notepad et avec une copie du script sql avec toutes les données dedans, et là, ça a quasiment fonctionné. Seul les « è » n'ont pas supportés la conversion, mais tout les autres caractères spéciaux minuscules sont bien passés, é, ç, à.. Pour les majuscules, elles sont remplacées par leur équivalent en minuscules. La fonction php *touppercase* fera certainement l'affaire où je laisserai le soin à l'utilisateur de le modifier.

En résumé :

1) j'ouvre le fichier sql avec encodage ISO avec notepad++ . On peut voir ce qu'il arrive lorsque j'affiche les données à l'utilisateur.

```
(10, 'adÃ©laÃ¯de', 'THÃ©NARD', 'une femme', 'adÃ?le', '1192dfdb037cca453ae9f55c83f6ca85a24
(1287, 'franÃ§oise', '[illegible]', 'une femme', '[illegible]', '0807e3fab94e524d832dfbaf88b3851c5b8d4b
(1445, 'michaÃ«l', '[illegible]', 'un homme', '[illegible]', '2a3b027c80116e9a80bba13d18198526eadbf
```

2) je force directement l'encodage le fichier en utf-8, pas de conversion ni d'étape intermédiaire. C'est ce qui fonctionne le mieux dans mon cas.

```
(10, 'adélaïde', 'THéNARD', 'une femme', 'adxC3?le', '1192dfdb037cca453ae9f55c83f6ca85a24526
(1287, 'françoise', '[illegible]', 'une femme', '[illegible]', '0807e3fab94e524d832dfbaf88b3851c5b8d4bf1
(1445, 'michaël', '[illegible]', 'un homme', '[illegible]', '2a3b027c80116e9a80bba13d18198526eadb
```

3) Un simple rechercher remplacer fera l'affaire pour la finalisation.
Et non, notepad++ ne comprend pas le caractère, je pense au regex mais d'abord j'essaye avec visual studio code, mon autre éditeur. Et là, le caractère est bien reconnu pour le remplacement. Ouf. Parce que pour le contrat et ces nombreuses zones de texte, je vais en avoir besoin.

J'ai refait les 3 étapes pour avoir un fichier prêt à importer avec toutes les données « propres ».

Ici, j'ai utilisé mes éditeurs de textes pour avoir un repère visuel sur ce qui allait être à faire. J'ai seulement 3 champs texte dans cette table, alors que dans les contrats j'ai de nombreuses zones de texte densément fournies …

- Pour les tables des contrats :

Alors, là, je fais l'extraction de la base donnée depuis le script phpMyAdmin de mon serveur avec toutes les tables, contrat et état des lieux en ISO et je répète l'opération précédente.

Et ça ne fonctionne pas aussi bien. Pourquoi ? j'imagine que c'est parce que j'ai un encodage initial en latin_général_ci alors que pour le fichier des membres, il était en latin_swedish.

La solution reste simple, dans notepad++, il faut utiliser « convertir en UTF-8 » et non plus directement « encoder en UTF-8 sans BOM»
(BOM est l'ordre des octets utilisés, ordre inscrit au début du fichier, header() php ne marche plus).

L'opération s'est plutôt bien déroulée mais j'ai quand même une petite trentaine de ligne de mal affichée avec cette méthode. J'ai pu m'en rendre compte grâce à Visual Studio code et sa barre latérale verticale, qui fait apparaître les erreurs en surbrillance une fois sélectionnée pour le rechercher-remplacer.

Ceux sont les lignes des utilisateurs qui ont voulu éditer un contrat entre le moment où mon hébergeur à modifier l'encodage de la connexion à ma base sur mon serveur mutualisé et le moment où j'ai stoppé le service, 3 jours il me semble.

Je fais donc des rechercher remplacer, via notepad++ et VScode:

Ã© => é	159 occurences
Ã§ => ç	4
Ã? => è	40 uniquement avec VScode
Ã¹ => ù	5
Ã® => î	0
Ã» => û	4
Ã¢ => â	1
Ãª => ê	4
Ã´ => ô	2
Ã¯ => ï	0
Ã« => ë	0
Ã¤ => ä	0
Ã¼ => ü	0
Ã => à	16

Mon fichier SQL de contrat et état des lieux est maintenant « propre ».

Avant de l'insérer dans l'ancienne base pour la passer à la moulinette, je dois modifier les bouts du script qui créent les tables parce qu'ils ont l'encodage en latin. Je remplace donc latin1 et latin_general_ci par utf8, et j'indique quelle base doit être utilisée.

Enfin, j'importe mes données pour avoir en local la même configuration actuelle qu'en ligne mais avec mes données réparées et en UTF8.

L'importation a réussi, 24 requêtes exécutées. (locati26_membres_emails - Copie.sql)

L'importation a réussi, 63 requêtes exécutées. (locati26_contrat_etat - Copie.sql)

Et ça marche, après quelques essais bien sûrs.

J'ai l'intégralité de la base bien encodée, en local, avec l'ancienne architecture.

C – migration en local avec la nouvelle architecture

Il est temps de passer à la version 2 de la base de donnée de ce site (en local). Je relance donc mon script de migration avec mes données toutes fraîches.

Et j'obtiens donc ma nouvelle base remplie avec les données traitées.

Il me manque les états des lieux, et leur script de migration, mais le principe est là. J'aurais aussi des variables à ajouter pour limiter les contrats par utilisateurs par exemple. J'émets quand même un petit soupir de soulagement à cet instant même si je crains le pire pour le hash du mot de passe.

Une fois cela réalisé fait, je n'aurais plus qu'à exporter ma base via phpMyAdmin avec les options qui vont bien, avant d'importer ce fichier SQL sur mon serveur toujours via phpMyAdmin.

La fin de la formation arrive, je n'ai pas eu le temps d'en faire plus. Beaucoup de chose reste à faire mais quasiment aucune qui n'est jamais été faite. Mon seul obstacle reste le design pattern MVC que je souhaite mettre en place.

V - Conclusion

La formation se termine, 4 mois vite passé, et je n'ai pas eu le temps de faire grand chose. J'ai préféré commencé par faire le script de migration parce que j'attendais d'avoir mes premiers cours soit de Merise soit d'UML pour démarrer la maintenance. Les deux derniers mois ont été très rude avec en plus de Git Saas et SQL, l'initiation à Angular, Nodejs et au MVC.

Je n'ai donc pas pu donner tout le temps que je désirais, ni à la formation, ni à mon projet. Mais j'ai quand même vu le design-pattern MVC qui manque cruellement dans mon code. J'aimerai l'appliquer dans mon futur code mais avant de me lancer, je souhaite également réfléchir à quelles technologies je dois utiliser. En fait il serait intéressant de le faire deux fois, la première juste côté serveur pour relancer le service, en PHP pour valider le concept de « classe » et « objet » vu avec TypeScript, et la deuxième en validant Nodejs côté Back-end et Angular en Front-end.

Je n'ai pas beaucoup parlé de référencement positionnement dans ce rapport, mais pour mon projet personnel, ça reste un critère de premier plan quant au choix de la technologie. En ce sens, Angular et son credo tout dynamique me fait un peu peur, même si Google se cache derrière.

Compétences validées pour le titre :

- Concevoir une base de données
- Mettre en place une base de données
- Développer une interface utilisateur
- Développer des pages web en lien avec une base de données
- Développer des composants d'accès aux données
- Développer une application simple de mobilité numérique
- Utiliser l'anglais dans mon activité professionnelle en informatique (stackoverflow, devdocs, w3schools, html5rocks… et surtout maintenant github !)

Compétences vu en formation et à valider :

- Utiliser l'anglais dans mon code
- Maquetter une application
- Mettre en oeuvre une solution de gestion de contenu ou e-commerce

- Les plus importantes, le MVC, les classes et autres objets

Annexes

Découpez-les si besoin pour pouvoir suivre plus facilement.

1 - Contrat de location saisonnière généré

2 – Schéma fonctionnement du site + schéma espace membres

3 – MLD

4 – Schéma du script de migration

Contrat de location saisonnière

entre

Le(s) propriétaire(s) dénommé "le propriétaire" :

Mr DUPONT Henry et Mme CLAIRY Marion

Adresse :
maison ETCHE,
20 avenue EDERRA,
64500 SAINT JEAN DE LUZ

Téléphone :
05.59.26.00.00
E-mail :
dupont@orange.fr

Le(s) locataire(s) dénommé "le locataire" :
Mr MARTIN Michel, Mme MARTIN Nadine, et leur fille Lucile.
Mr DURANT Etienne, Mme DURANT Sylvie , et leur fils Jean

Adresse :
Les MARTIN : 33 rue des hirondelles, 33000 BORDEAUX.
Les DURANT: 20 avenue des fleurs, 64000 PAU

Téléphone :
05.59.89.00.00
E-mail :
martin@sfr.fr

Adresse et description de la location

Adresse précise :

maison de la plage,
75 boulevard de l'océan,
64200 BIARRITZ

Description de la location :
- Type de la location : maison
- Superficie(m^2) : 100m^2
- Nombre de pièces : 4
- Nombre de personnes maximum : 8
- Fumeurs acceptés : oui
- Animaux acceptés : non
- Classement : 2 étoiles

Salon/salle à manger :
Une table de 8 personnes et 8 chaises, un canapé d'angle, 2 fauteuils, 1 tv lcd, 1 meuble TV, 1 table basse, 1 vaisselier. Vue sur mer.table tout perdu?? l'océan

Cuisine :
Table 4 personnes et 4 chaises, gazinière 4 feux gaz+four, haute aspirante, réfrigirateur+congélateur, four micro-ondes, cafetière, grille pain, évier inox 2 bacs

taille police 8

Chambres :
Chambre 1 : lit king size, armoire, commode table de chevet.
Chambre 2 : 2x2 lits superposés, 2 armoires
Chambre 3 : lit king size, armoire, table de chevet

taille police 9

Sanitaires :
au rez de chaussé : salle d'eau comprenant 1 douche, 1 bidet, un lavabo double vasque, 4 étagères.
à l'étage : salle de bain comprenant baignoire, lavabo double vasque, 4 étagères.
1 WC par étage.

taille police 10

Autres (garage, jardin, situation géographique...) :
Terrasse du salon de 15 m^2 avec vue sur mer, table parasol et 4 transats.
La maison se situe à moins de 500m de la plage et des premiers commerces alimentaires.

taille police 11

Contrat de location saisonnière

(suite et fin)

Date de la location :

Location entendue du 17/07/2013 à 15 heures jusqu'au 24/07/2013 à 10 heures.

Prix, arrhes et dépot de garantie :

Le prix de cette location est de 1000€ par semaine, non inclus les charges mentionnées plus bas.
Les arrhes de cette location sont de 20% soit 200€ et seront versées lors de la signature du présent contrat.
Le solde du prix, soit 800€ sera payé le jour de la remise des clés, en même temps que le dépôt de garantie de 25% soit 250€. Ce dépôt de garantie peut etre encaissé. Il sera rendu en intégralité jusqu'à quinze jours aprés le départ du locataire, excepté en cas de dégâts.

Conditions de paiement :

Le locataire paiera par tout moyen convenu avec le propriétaire et mentionné ici: chèques.

Conditions d'annulation :

Les deux parties ont sept jours pour se rétracter à partir de la date de signature, sans qu'aucune pénalité ne leur soit infligée.
Passé ce délai :
- Le locataire peut se rétracter à tout moment mais perdra l'intégralité de ses arrhes.
- Le propriétaire peut se rétracter à tout moment mais devra remettre le double des arrhes déjà versées par le locataire.

Charges supplémentaires :

D'autres charges peuvent être demandées :
Taxe de séjour : inclus(1euro/pers/jour)
Eau/éléctricité/gaz : inclus
Autres : non

Rappel de quelques obligations :

- Le locataire doit :
- Assumer sa responsabilité civile.
- Utiliser le logement "raisonnablement" et répondre des dégradations dans les plus brefs délais.
- Respecter la capacité maximale d'hébergement.
- Faire le ménage si aucun forfait n'est prévu, et rendre le bien en location dans le même état de propreté que lors de son entrée dans les lieux.

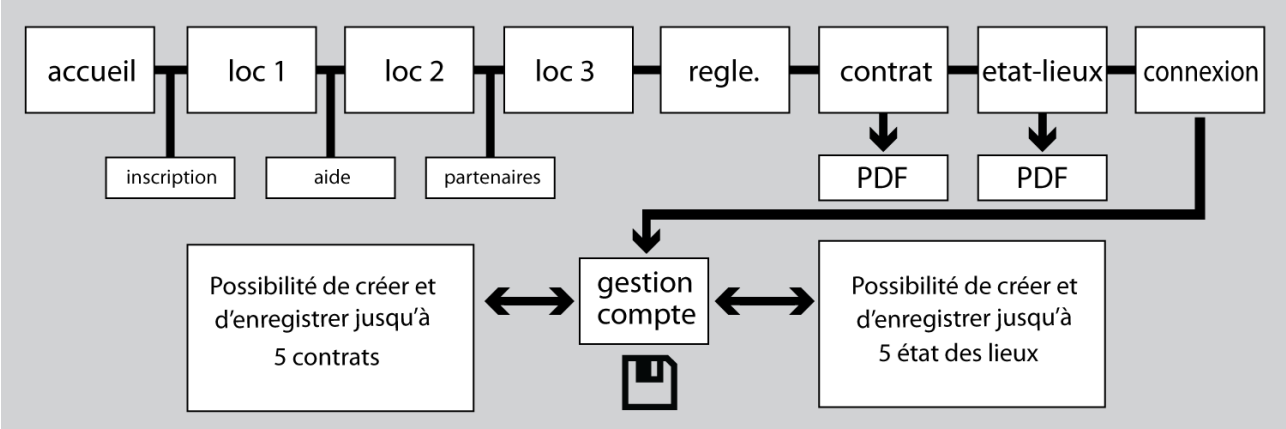
Annexes :

Vous trouverez en annexe à ce contrat le dossier de diagnostic technique comprenant, si nécessaire, un constat de risque d'exposition au plomb, un état des risques naturels et technologiques, ainsi qu'un diagnostic de performances énergétiques.

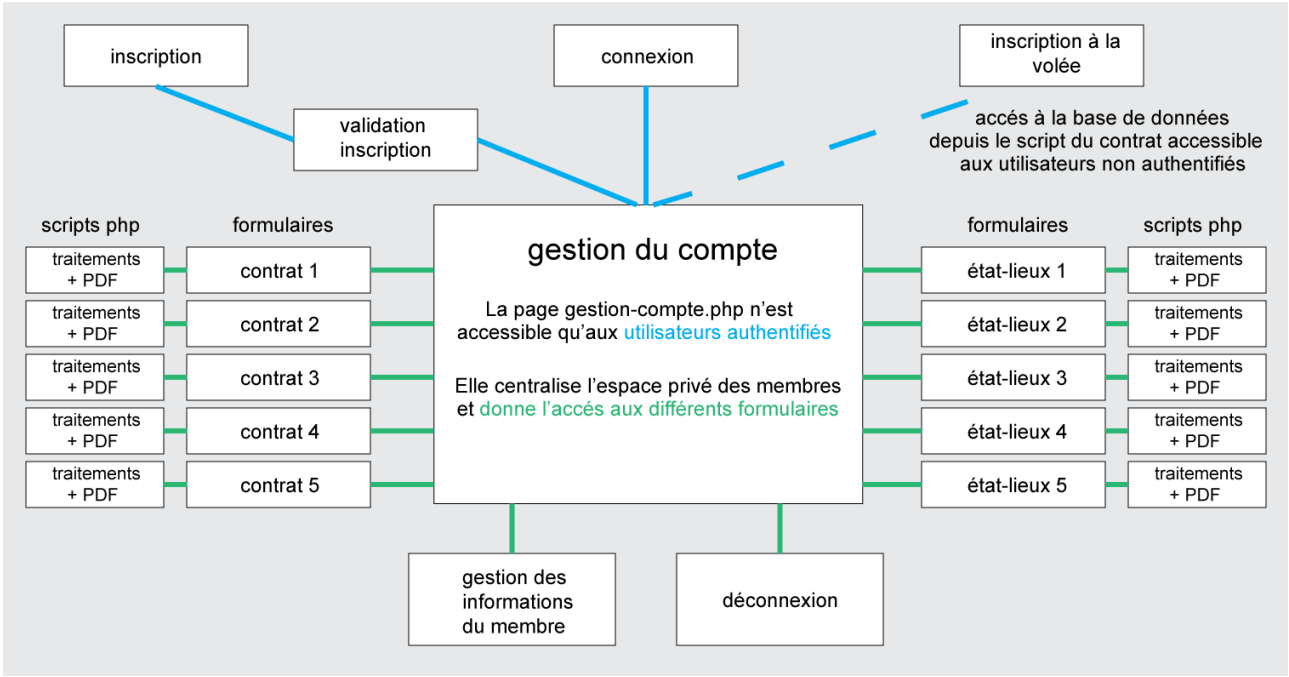
Fait en deux exemplaires à : Saint Jean de Luz le

Signatures des deux parties précédées de la mention "lu et approuvé"
Le(s) propriétaire(s) Le(s) locataire(s)

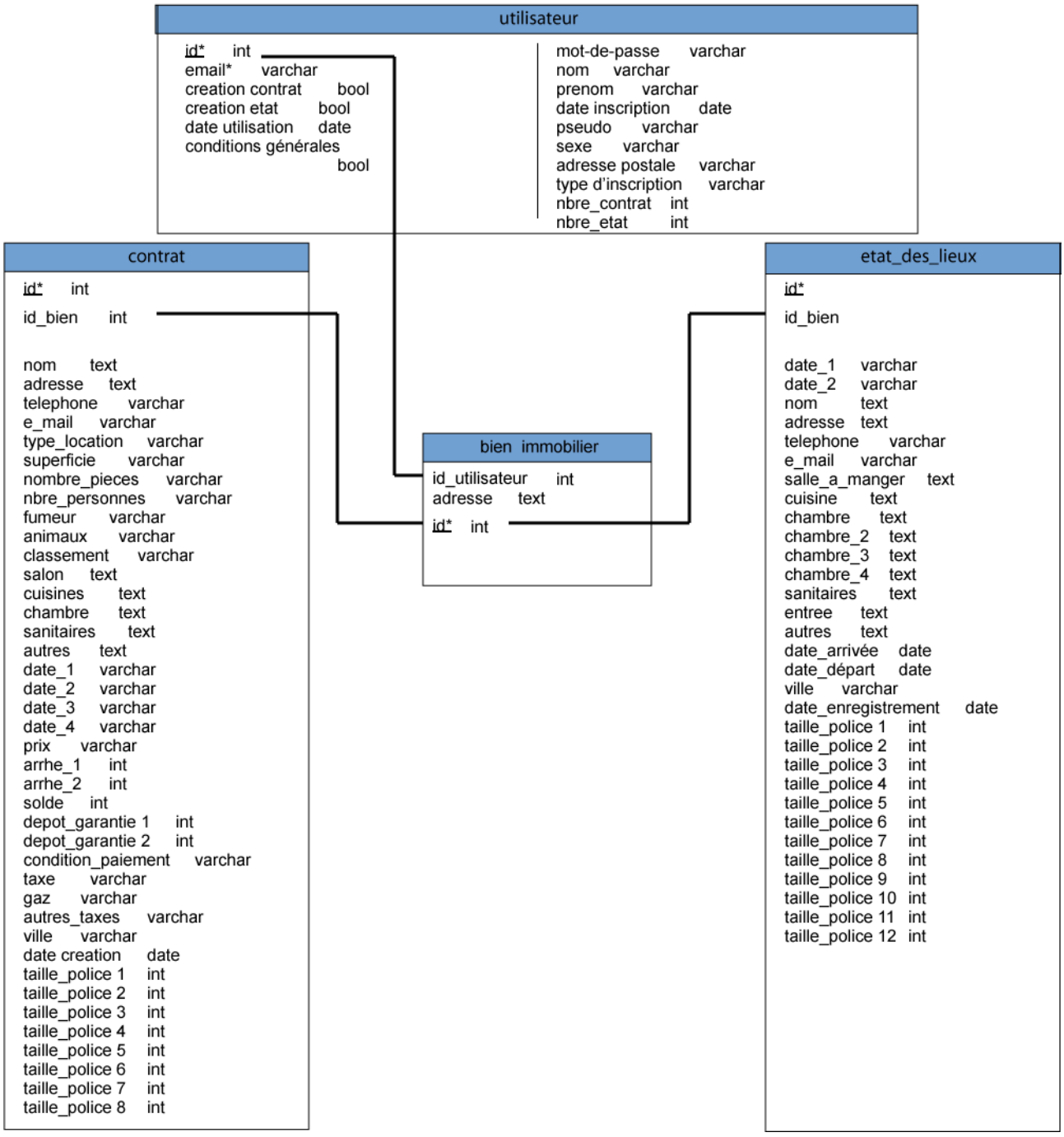
Fonctionnement du site

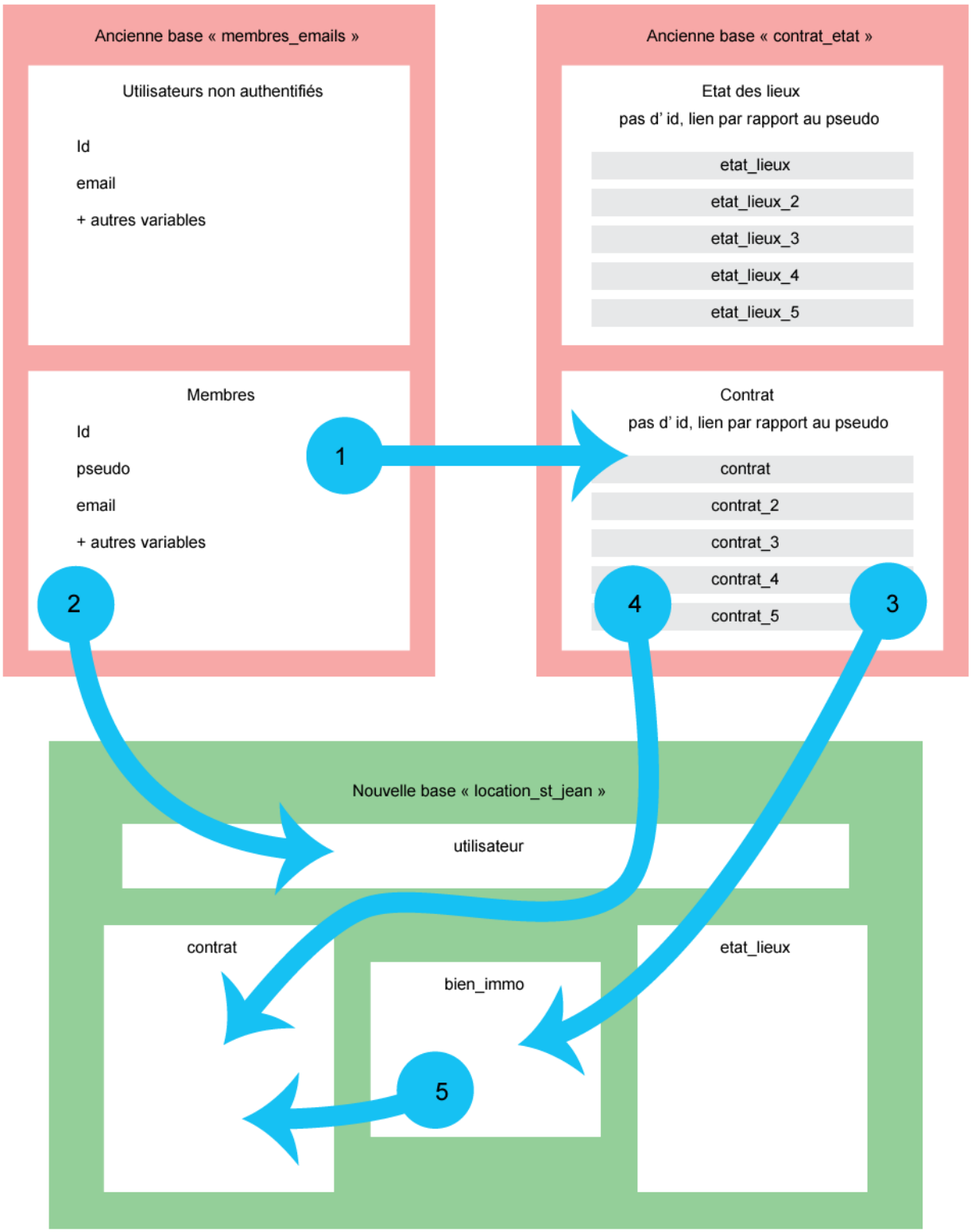


Fonctionnement de l'espace membre



Modèle logique de données





1) Insérer l'id du membre dans les contrats existants, j'appelle ce champ « id_pseudo »
2) Insérer les infos des membres existants dans la table « utilisateur » (id du membre = id utilisateur).
3) Insérer « id_utilisateur » « adresse » dans la table bien_immo
4) Insertion des contrats dans la table « contrat » . J'insère également les variables « pseudo » et « id_pseudo » par rapport au mld, dans les tables pour le fonctionnement du script et pour faire des vérifications. Je les effacerai après.
5) Récupérer les id des bien_immo pour les associer à leur(s) contrat(s) respectif(s)