ECOLE POLYTECHNIQUE
UNIVERSITAIRE
DE MONTPELLIER

IG
Informatique
et gestion

Projet WEB

Objectif : Créer et mettre en place une application WEB opérationnelle

ECOLE POLYTECHNIQUE UNIVERSITAIRE DE MONTPELLIER
UNIVERSITE MONTPELLIER II SCIENCES ET TECHNIQUES DU LANGUEDOC
Place Eugène Bataillon 34095 MONTPELLIER CEDEX 5
Tél. : 04 67 14 31 60 – Fax : 04 67 14 45 14
E-mail : scola@polytech.univ-montp2.fr

Introduction

L'application Web que j'ai créée dans le cadre du projet SQL IG3 de l'année 2014/2015 est une application permettant de stocker et d'avoir accès à tout moment à des dossiers médicaux de patients inscrits sur la base de données.

Un médecin inscrit peut en effet avoir accès aux dossiers de tous les patients utilisant l'application. En rentrant leur numéro de pièce d'identité. (Qui sert également de Primary Key au niveau de la base de données sur la table patient). Un médecin peut s'inscrire sur le site en accédant à la page NouvelleInscription.php il doit alors fournir des informations relatives à son cabinet (adresse) ainsi que sa spécialité, sa date de naissance et un email qui servira d'identifiant pour pouvoir naviguer sur le site. Concernant le mail, celui-ci sert de clé primaire de la table 'médecins' il doit donc être unique, ainsi lors de chaque nouvelle inscription j'effectue une vérification de l'existance de l'adresse mail rentrée dans la base de donnée. Cette vérification est fait en JavaScript / AJAX. (Explication des fonctions JavaScript/ AJAX dans la partie 5). Un médecin peut également supprimer un patient si celui-ci le réclame. Ou lors d'une consultation rajouter une ordonnance ou un antécédent médical.

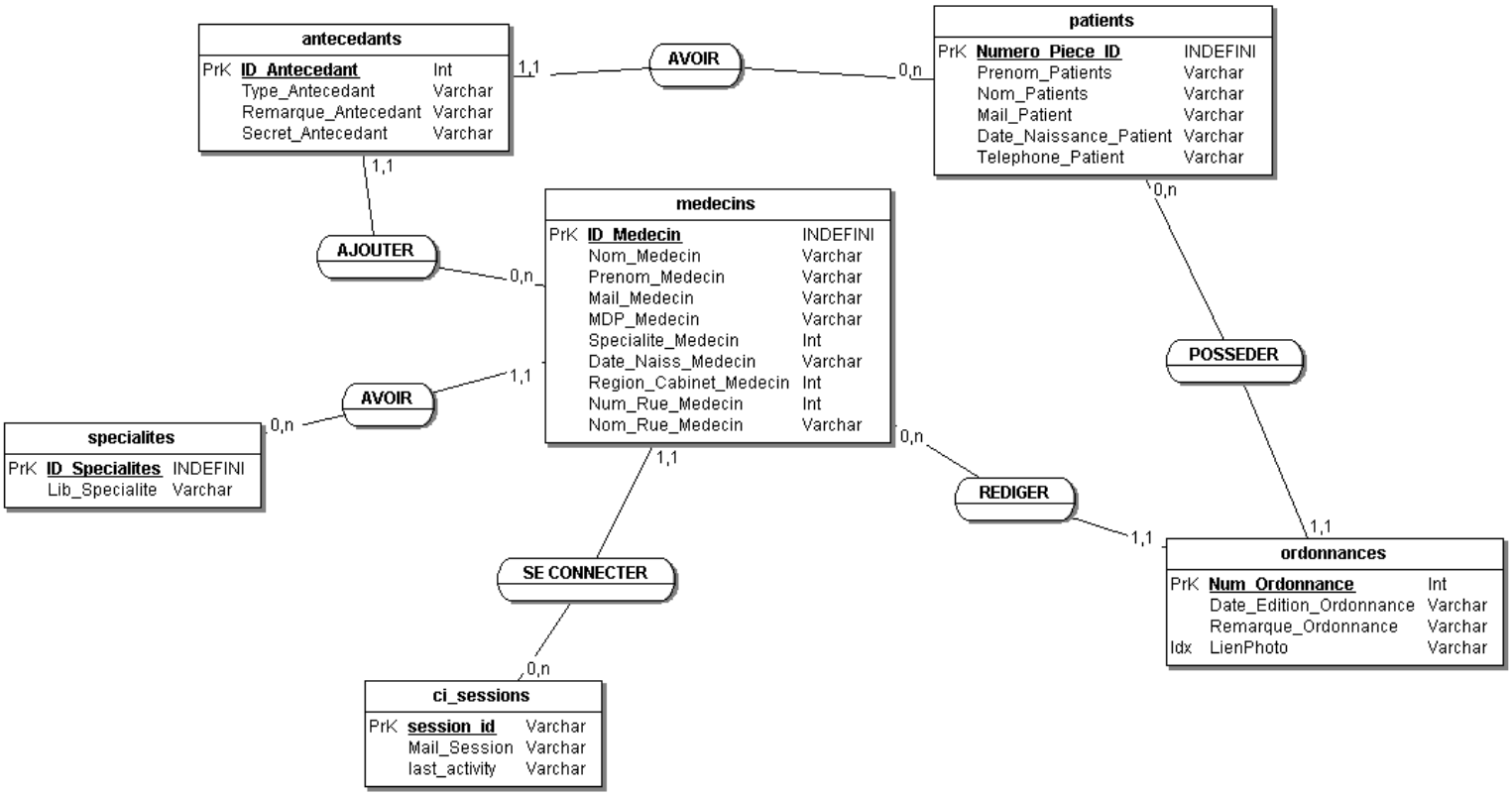
SOMMAIRE

Bootstrap

Bootstrap est un Framework créée par les équipes de Twitter pour encourager tout le monde (en interne) à utiliser le même outil et ainsi diminuer les incohérences. Naturellement, l'initiative Bootstrap a été un vrai succès chez Twitter puisqu'elle a largement augmenté la productivité de l'équipe tout en limitant les incohérences. Le Framework Bootstrap est sorti en août 2011 comme projet open-source sur Git hub. Il est aujourd'hui très largement utilisé car il propose toute une panoplie de solutions et a une documentation très complète. J'ai également dans le cadre de ce projet été amené à utiliser ce Framework. C'est ce qui me permet d'implémenter des fonctions JavaScript sans aucun problème et de manière très efficace.

Base de données

Ma base de donnée, hébergée chez olympe et utilisant PHPMyadmin, est composée de cinq tables : une relative aux sessions, et une pour les spécialités des médecins. Ainsi on ne peut pas insérer un médecin avec une spécialité qui n'existe pas : c'est la contrainte statique Puis une table qui stocke tous les médecins (avec un Identifiant, une adresse mail etc…), une pour les patients (Identifiant, nom, prénom etc…) et enfin deux tables : antécédents et ordonnances.



Des contraintes statiques ainsi que dynamiques ont, bien évidemment, été mises en place. Ces contraintes peuvent être visibles dans le fichier trigger ainsi que dans le script de la base de données. Elles concernent, entre autres, l'unicité des clefs (et de l'adresse mail des médecins) mais également la suppression d'un patient qui entraîne automatiquement la suppression de toutes ses ordonnances et ses antécédents.

Sessions :

Une fois l'inscription effectuée vous êtes redirigé vers l'index pour vous connecter avec vos identifiants fraîchement créés. La connexion se fait à travers un modal Bootstrap (une sorte de pop-up qui est appelé dès que l'on clique sur 'Se connecter '). L'utilisation des sessions PHP ayant été fortement déconseillée par Mr Castelletort, j'ai dû chercher un autre moyen de mettre en place les sessions. La piste des cookies a été empruntée.

Un fichier session.php contient trois fonctions relatives aux sessions : check_auth(mail,mdp), check_login(), logout().

check_auth(mail,mdp) prend en paramètre un mail et un mot de passe et vérifie leur existence dans la base de donnée. S'ils existent bel et bien elle crée un cookie chez l'utilisateur et y stocke l'identifiant de session ainsi que le mail et le prénom de l'utilisateur puis retourne True. Sinon c'est que l'utilisateur s'est trompé de mot de passe ou de mail elle retourne False.

Check_login() Ne prend pas de paramètre. Elle permet de vérifier si une session existe déjà. Pour cela elle récupère les cookies et vérifie la valeur de l'ID du cookie avec la table ci_session s'il y a une correspondance elle donne l'accès a la page demandée sinon elle renvoie vers la page d'accueil.

Logout() Ne prend pas de paramètre non plus. Elle permet de détruire les cookies ainsi que la ligne qui leur correspond dans la table ci_session. Puis elle renvoie vers la page d'accueil.

A chaque demande d'accès à une page privée qui nécessite une authentification, j'appelle la fonction check_login (). A chaque appel de cette fonction on actualise la table ci_session en ajoutant une heure de temps de connexion supplémentaire, et on actualise également le temps d'expiration des cookies (+ 1h également). La fonction check_auth (mail, MDP) n'est appelée que lors de la connexion et logout () que lors de la déconnexion. Vous avez sûrement remarqué que dans la page index.php, la div qui permet de se connecter est tout simplement cachée. Un utilisateur malveillant pourrait la faire réapparaître grâce à la console en enlevant la class css qui la fait disparaître (ClassThatHides que j'ai créée). Un utilisateur pourrait ainsi se connecter deux fois (deux cookies, deux lignes dans la table ci_session) cela peut s'avérer problématique. Raison pour laquelle avant d'ajouter une ligne dans ci_session ou un cookie je commence par les supprimer au cas où ils existeraient déjà. J'aurais pu également, au lieu de simplement cacher la div de connexion par une classe css, passer par un if en PHP pour vérifier si l'utilisateur est déjà connecté et afficher une page sans cette div. Au bout d'une heure sans activité la session est détruite puisque le cookie est détruit, un trigger temporel détruit également la session coté serveur (sur la table ci_session).

JavaScript :

Grace au cours de web, j'ai pu approfondir mes connaissances en JavaScript. Je l'ai ainsi beaucoup utilisé dans mon site notamment pour éviter de systématiquement rafraîchir la page à chaque accès à la base de données. Ajax est un concept de programmation qui utilise JavaScript pour communiquer avec le serveur. Ainsi j'ai mis en place de l'Ajax à deux reprises dans mon site sur la page NouvelleInscription.php et EspaceMedecin.php.

NouvelleInscription.php : Ici l'Ajax sert à savoir si l'adresse mail rentrée dans le formulaire est déjà utilisée par quelqu'un (auquel cas elle existerait déjà dans la base).

Sans Ajax : On aurait dû attendre que l'utilisateur submit le formulaire pour pouvoir savoir si cette adresse mail existe déjà. Puis renvoyer une erreur grâce à un if. Donc perte de temps puisqu'on doit rafraîchir la page puis revenir sur le formulaire pour saisir une nouvelle adresse mail.

```
<script type="text/javascript">

  document.forms["formulaireinscription"].style.display="block";
  var f=document.forms["formulaireinscription"].elements; // On recupere tout les element du formulaire

  function verifMail(){  // On défini une fonction verifMail() qui prend rien parametre
    var regEmail = new RegExp('^[0-9a-z._-]+@{1}[0-9a-z.-]{2,}[.]{1}[a-z]{2,5}$','i'); // Expression réguliere du'une adresse mail
    var mail = f['mail'].value; // On recupere le premier mail
    var mail2 = f['mail2'].value; // Et le mail de vérificatio
    if (!regEmail.test(f['mail'].value) || !regEmail.test(f['mail2'].value)){ // S'ils correspondent pas à l'expression définie plus haut
          document.getElementById("maildejapris").innerHTML = "Rentrez une adresse mail"; // On met un message d'erreur
          document.getElementById("maildejapris1").innerHTML = "Rentrez une adresse mail";
    }
    else if(mail != ''){ // Si les deux mail correspondent à une vraie adresse mail
          $.post('/checkInscription.php',{ mail: mail, mail2: mail2 }, function(data) { // On envoie ces deux mail à /checkInscription.php qui vérifie leur existence dans la base
          $('#maildejapris').text(data); // On rajoute à la div #maildejapris ce que nous retourne la page /checkInscription.php
          $('#maildejapris1').text(data);
          });
    }
  } // Fin de la fonction
</script>
```

Avec Ajax : On commence par créer une fonction JavaScript, cette fonction commence par vérifier la syntaxe du mail, si sa syntaxe est correcte elle envoie le mail à /CheckInscription.php

/CheckInscription.php

```
<?php include('./fonctions.php');


$mail=$_POST['mail'];
$mail2=$_POST['mail2'];

$connect = ConnexionDB(); // Je me connecte à la base de donnée
$espace =" ";

    if ($mail != $mail2){
        echo 'Les 2 adresses mail sont differentes.';
    }
    else{
        // on recherche si ce login est dÃ@jÃ  utilisÃ@ par un autre membre
        $sql1 = "SELECT count(*) FROM medecins WHERE Mail_Medecin='$mail'";
        $req = $connect->query($sql1);
        $data = mysqli_fetch_array($req);

        if ($data[0] == 0) {
            echo 'Personne n\'utilise cette adresse mail.';
        }
        else{
            echo 'Un membre est déjà  inscrit avec cette adresse mail.';
        }
    }

?>
```

Ce script vérifie l'existence du mail dans la base de données et renvoie soit 'Personne n'utilise cette adresse mail' soit 'un membre est déjà inscrit avec cette adresse' et c'est ceci qui sera affiché dans le formulaire sans avoir besoin de rafraîchir la page. Ainsi nous avons effectué une vérification de manière asynchrone sur la base de données. L'expérience coté utilisateur est plus agréable, plus dynamique.

La même méthode est utilisée également dans EspaceMedecin.php pour vérifier l'existence du patient dans la base de données, avant de valider et donc sans rafraichir la page.

PHP

Pour la construction de cette application Web les principaux langages utilisés sont bien sûr l'HTML et le PHP. Pour mieux expliquer comment a été utilisé le PHP je vais expliquer le fonctionnement d'une page exploitant du PHP (formulaire etc..) : RecupererDossier.php

RecupererDossier.php :

Cette page ne doit être accessible que si l'utilisateur est connecté on doit donc appeler la fonction check_login (). Le script suivant nous permet de faire cela :

<?php include ('./session.php'); check_login(); ?>

Ensuite il va falloir afficher toutes les ordonnances et les antécédents d'un certain patient donc on récupérer le numéro du patient dans le formulaire de la page /EspaceMedecin.php qui incluait une méthode post. Aussi je peux passer en paramètre le numéro du patient en URL (Pratique pour les redirections). Donc en début de page je vérifie les deux (POST et GET verbes HTML)

<?php if(!empty($_GET['numPatient'])){$id=$_GET['numPatient'];}else{$id = $_POST['numPatient'];}?>

Remarque:

Un antécédent peut relever du secret médical raison pour laquelle dans ma table 'antécédents' il y a une colonne secret qui peut être soit à True soit à False. Si elle est à True seul le médecin qui a édité l'antécédent a le droit de le voir sinon tous les médecins peuvent voir cet antécédent.

J'effectue trois requêtes :

```php
<?php $medecinNom = $_COOKIE['sessionCookie'][User]; ?>
<?php $medecinMail = $_COOKIE['sessionCookie'][Mail]; ?>

<?php
    $connect = ConnexionDB(); // Je me connecte à la base de donnée

    $IDMedecin = "SELECT ID_Medecin
                  FROM medecins
                  WHERE Mail_Medecin='$medecinMail'"
                  or die("Erreur lors de la consultation de données (Verif ID)" . mysqli_error($connect));

    $ListeID = $connect->query($IDMedecin);
    $IDMedecinFinal = mysqli_fetch_array($ListeID);

    $requeteOrdonnace = "SELECT Date_Edition_Ordonnance, Remarque_Ordonnance, ID_Medecin, LienPhoto, Num_Ordonnance
                         FROM ordonnances WHERE Numero_Piece_ID=$id"
                         or die("Erreur lors de la consultation de données (Ordonnaces)" . mysqli_error($connect));

    $listeOrdonnances = $connect->query($requeteOrdonnace); // Je récupere les ordonnances du Mr
    $requeteAntecedant = "SELECT Type_Antecedant, Remarque_Antecedant, Secret_Antecedant, ID_Medecin
                          FROM antecedants
                          WHERE (ID_Patient=$id AND Secret_Antecedant='False') OR (ID_Patient=$id AND ID_Medecin=$IDMedecinFinal[0])"
                          or die("Erreur lors de la consultation de données (Ordonnaces)" . mysqli_error($connect));

    $listeAntecedants = $connect->query($requeteAntecedant); // Puis les antecedants
    $espace=" ";
?>
```

La première ($IDMedecin) pour récupérer l'ID du médecin à partir du cookie, la deuxième ($requeteOrdonnance) pour récupérer toutes les ordonnances du patients et la dernière ($requeteAntecedant) pour récupérer tous les antécédents qui ne relèvent pas du secret médical OU qui ont été écrite par le médecin qui est connecté.

Puis le résultat de ces requêtes est affiché dans un tableau dynamique qui contient autant de lignes que de résultat de la requête :

<?php while ($row = mysqli_fetch_array($listeOrdonnances)) { ?>

***** Le tableau en HTML*****

<?php } ?>

Les ordonnances peuvent également être scannée, l'affichage de l'ordonnance se fait dans ce cas dans un modal avec en source le lien de la photo. Une vérification en PHP lors de l'ajout d'une ordonnance vérifie qu'il y a soit un lien photo soit une remarque textuel dans l'ordonnance mais jamais les deux. Cette vérification est aussi faite grâce à un trigger au niveau de la base de données.

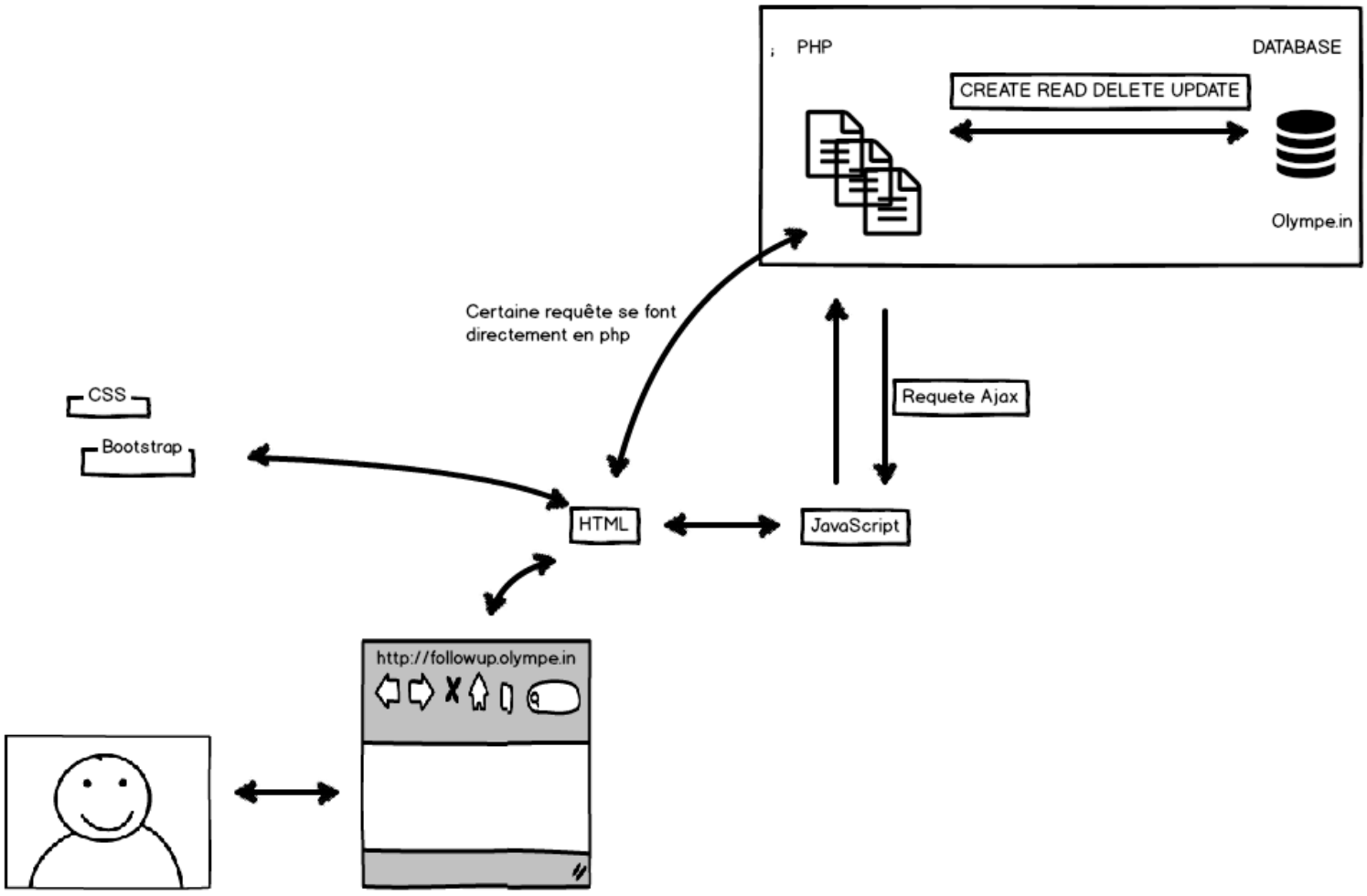
Le PHP m'a également offert la possibilité d'optimiser mon code : J'ai défini certaines pages communes (header, footer) que je peux appeler à chaque page. Ainsi si je veux modifier un lien dans le header je ne ferai cette modification qu'une seule fois !

Fonction MD5 :

MD5 peut aussi être utilisé pour calculer l'empreinte d'un mot de passe avec la présence d'un sel permettant de ralentir une attaque par force brute. Cela a été le système employé dans GNU/Linux. Ainsi, plutôt que de stocker les mots de passe dans un fichier, ce sont leurs empreintes MD5 qui sont enregistrées, de sorte que quelqu'un qui lirait ce fichier ne pourrait pas découvrir les mots de passe.

Schéma d'architecture

Le schéma architectural de l'application web est représenté dans l'image qui suit. Un client accède au site web il récupère une page en HTML. La mise en forme de cette page a été géré en amont par du css et Bootstrap. Des scripts JavaScript effectuent des requêtes Ajax sans rafraichir les pages web et d'autre requête sont faite directement en PHP (rafraichissement de la page nécessaire). Cela peut s'assimiler à une architecture MV (Modèle vue). Le choix de ne pas utiliser un Framework MVC (tel que codeigniter par exemple) a été fait car tout au long du projet il a fallu faire des choix puisque nous disposions que de peu de temps pour la réalisation du projet.



Conclusion

Ce projet a été très constructif, malgré le temps imparti un peu court, car il nous a permis de maitriser un nouveau langage très répandu : JavaScript. Avec la mise en place d'Ajax qui permet de faire des requêtes asynchrones. Les cours de WEB que nous avons eu avec Mr Castelletort ont été très bénéfiques car cela nous a permis de nous orienter au début de notre projet. C'est en effet lors de ces cours-là que j'ai compris l'utilisation d'Ajax à travers du JavaScript et l'utilité de Bootstrap.