TITRE PROFESSIONNEL DÉVELOPPEUR LOGICIEL NIVEAU III

BONNEFIS Yoan

Sommaire

Remerciements

Je voudrais avant tout remercier l’équipe, Alexandre DENURRA, directeur fondateur de novei formation, Sandrine ORIOL, directrice de l’école e2n, Mickaël NOEL, lead formateur, et Johanne BADIN, chagée de formation, pour m’avoir donner ma chance dans un premier temps, puis pour leur accompagnement et leur bienveillance ainsi que pour leurs conseils.

Merci à tout les apprenant de cette promotions avec qui j’ai partagé six mois, et qui ont su instaurer une très bonne ambiance. 6 mois de partages et d’entraides qu’il fût plus que plaisant de partager avec mes collègues et formateurs.

Merci à tous.

Compétences du référentiel couvertes

N° fiche AT	Activités types	N° fiche CP	Compétences professionnelles
1	Développer une application client-serveur	1	Maquetter une application
		2	Concevoir une base de données
		3	Mettre en place une base de données
		4	Développer une interface utilisateur
		5	Développer des composants d'accès aux données
2	Développer une application web	6	Développer des pages web en lien avec une base de données
		7	Mettre en œuvre une solution de gestion de contenu ou e-commerce
		8	Développer une application simple de mobilité numérique
		9	Utiliser l'anglais dans son activité professionnelle en informatique

Résumés de projet

Mon projet est un projet de plateforme, à but caritatif.

Le but étant, à travers ce site web, de proposer , en échange d'une 'vitrine' , aux professionnels de la construction, de la vente de matériaux et du transport, de participer à la construction (et / ou rénovation) solidaire d'habitations destinées aux sans-abris ainsi qu'aux mal logés.

A travers ce projet, donner la possibilité aux bénévoles de rejoindre un chantier, les professionnels faisant office d'encadrants technique, le contact se feras directement entre eux à travers cette plateforme.

Bien que ce projet, une fois finalisé, ne soit pas voué à ne proposer que du bénévolat, (contrats subventionnés, services civiques etc.), cette plateforme aura pour but de les mettre en avant en d'encourager cette pratique. Elle devra proposer tous les outils nécessaires pour la bonne expérience de ces derniers.

- Recherche de chantiers par localisation.
- Recherche de professionnels (par nom ou localisation).
- Prise de contact direct avec leurs futurs encadrants technique.
- Prise de contact entre bénévoles.
- Mise en avant des leurs accomplissement, sauvegarde de leurs avancés, pouvant être rendus visibles sur leurs profils.

Ainsi que bien d'autres outils leurs permettant une expérience agréable sur cette plateforme.

De plus, rendre accessible ce lieu de partage, à divers autres professionnels, dont nous aurons besoin tout au long de cette aventure, avec leurs propres outils.

- Médecins (addictologues, médecins généralistes ou spécialisés), afin d'assurer des visites dans nos « villages » ou habitations uniques.
- services sociaux (assistantes sociales, etc.).

Enfin permettre l'organisation des services, voir les avancés de nos chantiers, et gérer les approvisionnements (matériels) .

Enfin, c'est un projet auquel je tiens, et que j'espère faire vivre à travers cette plateforme, que je vais vous présenter.

Abstract

My project is a caritative platform.

This website goal is to offer publicity and visibility, to construction professionals, materials selling and transport to ask them, on exchange, to take part of the solidary construction and / or renovation, intended of homeless persons and poor housed.

With this project, offer the volunteer, a join possibility to a construction site.

Professionals would be technicals supervisors. They should can contact each other by the website.

Even if this project, on its finalized version, shouldn't purpose only volunteering, this website will have as goal to showcase and encourage that.

This should have all tools for them good experience.

Besides , make this sharing place accessible for some other professionals we will needed all this adventure long.

Finally, be able to organize work teams, see works in progress and manage supplying (matérials).

All of that, when this project will be finalized, today it's not, unfortunately, but this project is dear to my heart, and i really hope to be able to make this project alive in the future by the website (a part of it) that i will present you now..

Cahier des charges

Descriptif de la demande :

Ce projet part dans un premier temps d'un constat simple, malgré les efforts et les idées de maintes associations, l'accès aux logements pour les personnes en situation de grande précarité ainsi que l'insalubrité sont encore des problèmes trop présents qu'on peine à régler.

L'idée de se projet vient de là, participer à rendre moins présents ces problèmes en proposant des solutions.

Le problème premier pour une association de ce type, est de trouver des équipes techniques compétentes, afin d'encadrer un chantier.

De leurs trouver ensuite de la main-d'œuvre supplémentaire, afin d'avoir des délais raisonnables de livraison, pour des personnes dans le besoins, et permettre à ces dites mains-d'œuvres de trouver, communiquer et s'affilier aux équipes professionnelles.

L'application devra dans un premier temps répondre à ces questions.

Elle devra être responsive.

Spécifications techniques :

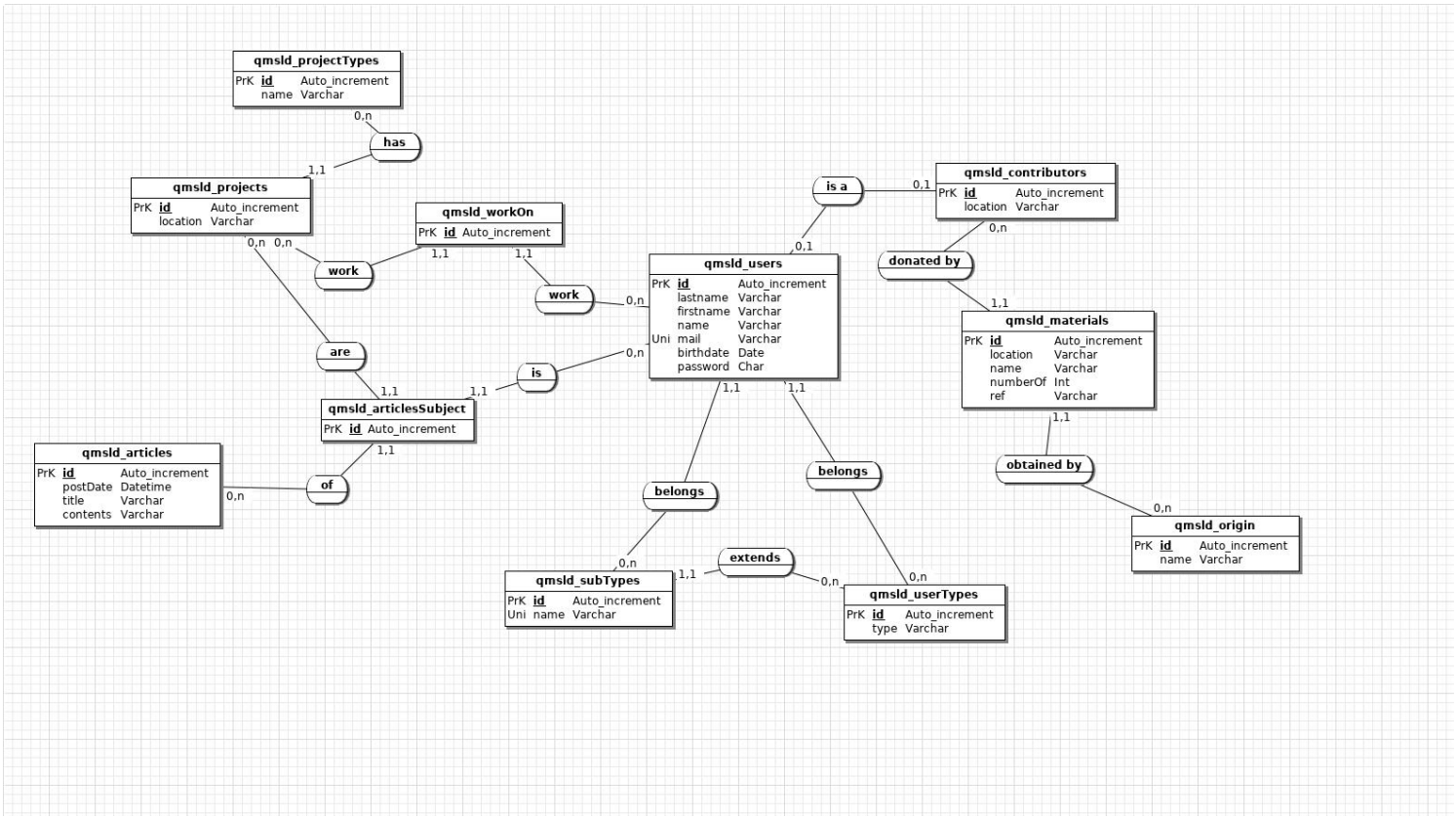
Langages client	HTML5
	CSS3
	JavaScript

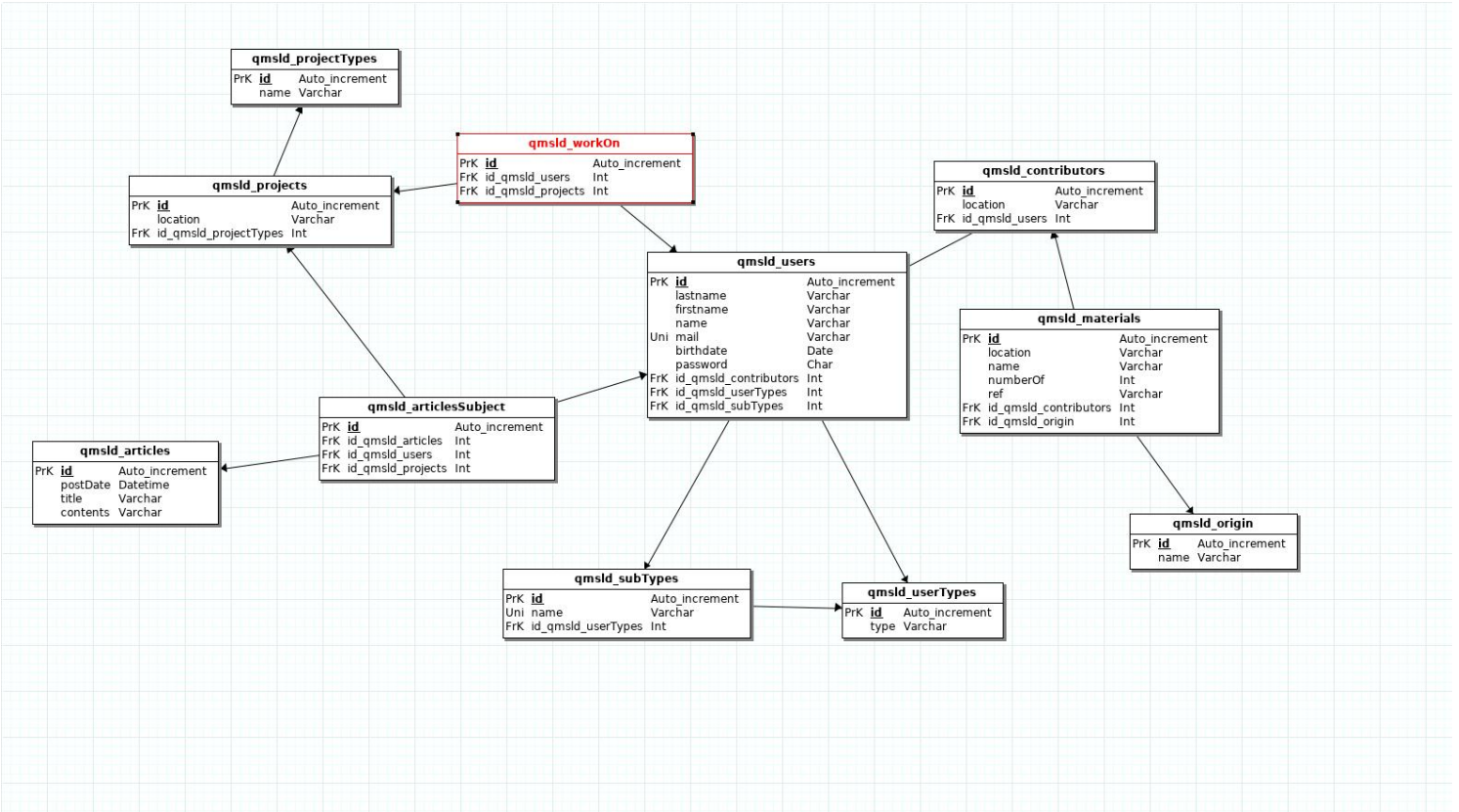
Langages serveur	PHP
	MySQL

Frameworks et librairies	Bootstrap 3
	jQuery
	jQueryUI

Concevoir une base de données

MCD et MLD





L’utilisateur (users) appartient à un type d’utilisateur (userTypes) :

Le type d’utilisateur définira le rôle de l’utilisateur. Il sera demander dès l’inscription.

l’utilisateur pourra choisir entre deux types « particulier » et « professionnel ».

- L’utilisateur à forcément un type (particulier ou professionnel).

- Le type d’utilisateur peut ne pas appartenir a un utilisateur comme appartenir à plusieurs d’entre eux.

Le sous type d’utilisateurs (subTypes) est une extension du type d’utilisateurs (userTypes) :

Le sous type d’utilisateur dépend directement du type, chaque type d’utilisateurs ayant leurs sous types propres.

- Un sous type d’utilisateurs prolonge forcément un type d’utilisateurs et ne peut être l’extension que d’un seul type .

Étant définit par son type principale, le sous type ne peut donc pas avoir plusieurs types.

- Le type d’utilisateurs peut ne pas avoir de sous type comme en avoir plusieurs.

L’utilisateur (users) appartient donc à un sous type d’utilisateurs (sub types) :

Ils seront sélectionnables pour les professionnels afin de définir directement leurs domaine de compétences.
Les particuliers se verront affecter leur sous type par un administrateur, car il dépendra du type de « contrats » les liants a nos projets.

- L’utilisateur à forcément un sous type et ne peut en avoir qu’un seul (à la fois).

- Un sous type d’utilisateurs peut ne pas appartenir a un utilisateur comme appartenir plusieurs d’entre eux

L’utilisateur (users) travail sur des projets (projects) :

Définit la relation entre l’utilisateur et le projet. Ceci permettras de pouvoir savoir qui a travaillé sur un projet ainsi que de lister les projets ou un utilisateur à travailler afin d’avoir un suivi, générale et personnel.

- L’utilisateur peut ne travailler sur aucuns projets et peut travailler sur autant de projets qu’il le souhaite

- Un projet peut n’avoir aucuns utilisateurs travaillant dessus comme en avoir plusieurs.

Étant une relation n / n, la table workOn fait la relation entre ces deux tables .

Les projets (projects) ont un type (projectTypes) :

Les projets seront définis par un type, en effet ce projet auras pour but de gérer les interventions sur deux types de chantier « construction » et renovation.

- Le projet a forcément un type (construction ou rénovation), et ne peut en avoir qu'un.
- Un type peut ne définir aucun projet comme en définir plusieurs.

L'article (articles) parle du projet (projects) :

Les articles parleront, entre autre, des projets. Que se soit pour des appels à main d'œuvres, suivi des travaux ou mise en avant après les travaux.

- L'article peut ne parler d'aucuns projets comme parler de plusieurs d'entre eux.
- Un projet peut n'apparaître dans aucuns articles apparaître dans plusieurs.

Étant une relation n / n, la table articlesSubject fait la relation entre ces deux tables .

L'article (articles) parle de l'utilisateur (users) :

Les articles parleront, aussi, des utilisateurs. Qu'ils soit lier a un projet, en tant que publicité pour les professionnel ou même pour faire des portraits.

- L'article peut ne parler d'aucuns utilisateurs comme parler de plusieurs d'entre eux.
- L'utilisateur peut n'apparaître dans aucuns articles comme dans plusieurs.

Étant une relation n / n, la table articlesSubject fait la relation entre ces deux tables .

Mettre en place une base de données

Création de la table « users » préfixé.

```
CREATE TABLE qmsld_users(
        id                  int (11) Auto_increment  NOT NULL ,
        lastname              Varchar (30) ,
        firstname             Varchar (30) ,
        name                Varchar (25) ,
        mail                Varchar (40) NOT NULL ,
        birthdate             Date ,
        password              Char (60) NOT NULL ,
        id_qmsld_contributors Int NOT NULL ,
        id_qmsld_userTypes    Int NOT NULL,
        id_qmsld_subTypes     Int ,
        PRIMARY KEY (id ) ,
        UNIQUE (mail )
)ENGINE=InnoDB;
```

Création de la table « projects » préfixé.

```
CREATE TABLE qmsld_projects(
        id                  int (11) Auto_increment  NOT NULL ,
        location              Varchar (40) NOT NULL ,
        id_qmsld_projectTypes Int NOT NULL ,
        PRIMARY KEY (id )
)ENGINE=InnoDB;
```

Création de la table « userTypes » préfixé.

```
CREATE TABLE qmsld_userTypes(
        id   int (11) Auto_increment  NOT NULL ,
        type Varchar (25) NOT NULL ,
        PRIMARY KEY (id )
)ENGINE=InnoDB;
```

Création de la table « subTypes » préfixé.

```
CREATE TABLE qmsld_subTypes(
        id                  int (11) Auto_increment  NOT NULL ,
        name                Varchar (30) NOT NULL ,
        id_qmsld_userTypes Int ,
        PRIMARY KEY (id ) ,
        UNIQUE (name )
)ENGINE=InnoDB;
```

Création de la table « projectTypes » préfixé.

```
CREATE TABLE qmsld_projectTypes(
      id   int (11) Auto_increment  NOT NULL ,
      name Varchar (20) NOT NULL ,
      PRIMARY KEY (id )
)ENGINE=InnoDB;
```

Création de la table « workOn » préfixé.

```
CREATE TABLE qmsld_workOn(
      Id             int (11) Auto_increment  NOT
NULL ,
      id_qmsld_users    Int NOT NULL ,
      id_qmsld_projects Int NOT NULL ,
      PRIMARY KEY (id )
)ENGINE=InnoDB;
```

Création de la table « articles » préfixé.

```
CREATE TABLE qmsld_articles(
      id      int (11) Auto_increment  NOT NULL ,
      postDate Datetime NOT NULL ,
      title    Varchar (60) NOT NULL ,
      contents Varchar (255) NOT NULL ,
      PRIMARY KEY (id )
)ENGINE=InnoDB;
```

Création de la table « articlesSubject » préfixé.

```
CREATE TABLE qmsld_articlesSubject(
      id             int (11) Auto_increment  NOT NULL ,
      id_qmsld_articles Int NOT NULL ,
      id_qmsld_users    Int NOT NULL ,
      id_qmsld_projects Int NOT NULL ,
      PRIMARY KEY (id )
)ENGINE=InnoDB;
```

Création de la table « contributors » préfixé.

```
CREATE TABLE qmsld_contributors(
      id           int (11) Auto_increment  NOT NULL ,
      location      Varchar (30) NOT NULL ,
      id_qmsld_users Int NOT NULL ,
      PRIMARY KEY (id
```

Création de la table « materials » préfixé.

```
CREATE TABLE qmsld_materials(
      id                int (11) Auto_increment  NOT NULL ,
      location           Varchar (30) NOT NULL ,
      name               Varchar (30) NOT NULL ,
      numberOf             Int NOT NULL ,
      ref               Varchar (8) NOT NULL ,
      id_qmsld_contributors Int NOT NULL ,
      id_qmsld_origin      Int NOT NULL ,
      PRIMARY KEY (id )
)ENGINE=InnoDB;
```

Création de la table « origin » préfixé.

```
CREATE TABLE qmsld_origin(
      id   int (11) Auto_increment  NOT NULL ,
      name Varchar (30) NOT NULL ,
      PRIMARY KEY (id )
)ENGINE=InnoDB;
```

Ajout des clefs étrangères.

```
ALTER TABLE qmsld_projects ADD CONSTRAINT FK_qmsld_projects_id_qmsld_projectTypes
FOREIGN KEY (id_qmsld_projectTypes) REFERENCES qmsld_projectTypes(id);

ALTER TABLE qmsld_contributors ADD CONSTRAINT FK_qmsld_contributors_id_qmsld_users
FOREIGN KEY (id_qmsld_users) REFERENCES qmsld_users(id);

ALTER TABLE qmsld_users ADD CONSTRAINT FK_qmsld_users_id_qmsld_contributors FOREIGN
KEY (id_qmsld_contributors) REFERENCES qmsld_contributors(id);

ALTER TABLE qmsld_users ADD CONSTRAINT FK_qmsld_users_id_qmsld_userTypes FOREIGN
KEY (id_qmsld_userTypes) REFERENCES qmsld_userTypes(id);

ALTER TABLE qmsld_users ADD CONSTRAINT FK_qmsld_users_id_qmsld_subTypes FOREIGN
KEY (id_qmsld_subTypes) REFERENCES qmsld_subTypes(id);

ALTER TABLE qmsld_materials ADD CONSTRAINT FK_qmsld_materials_id_qmsld_contributors
FOREIGN KEY (id_qmsld_contributors) REFERENCES qmsld_contributors(id);


ALTER TABLE qmsld_materials ADD CONSTRAINT FK_qmsld_materials_id_qmsld_origin
FOREIGN KEY (id_qmsld_origin) REFERENCES qmsld_origin(id);

ALTER TABLE qmsld_articlesSubject ADD CONSTRAINT
FK_qmsld_articlesSubject_id_qmsld_articles FOREIGN KEY (id_qmsld_articles) REFERENCES
qmsld_articles(id);

ALTER TABLE qmsld_articlesSubject ADD CONSTRAINT FK_qmsld_articlesSubject_id_qmsld_users
FOREIGN KEY (id_qmsld_users) REFERENCES qmsld_users(id);

ALTER TABLE qmsld_articlesSubject ADD CONSTRAINT
FK_qmsld_articlesSubject_id_qmsld_projects FOREIGN KEY (id_qmsld_projects) REFERENCES
qmsld_projects(id);

ALTER TABLE qmsld_workOn ADD CONSTRAINT FK_qmsld_workOn_id_qmsld_users FOREIGN
KEY (id_qmsld_users) REFERENCES qmsld_users(id);

ALTER TABLE qmsld_workOn ADD CONSTRAINT FK_qmsld_workOn_id_qmsld_projects
FOREIGN KEY (id_qmsld_projects) REFERENCES qmsld_projects(id);

ALTER TABLE qmsld_subTypes ADD CONSTRAINT FK_qmsld_subTypes_id_qmsld_userTypes
FOREIGN KEY (id_qmsld_userTypes) REFERENCES qmsld_userTypes(id);
```

Développer des composants d’accès aux données

Description du code mise en place pour l’accès aux données (database) :

```
/*
 *  Inclusion du fichier contenant les constantes de configuration.
 */
include_once 'configuration.php';

/**
 * Classe de connection à la base de données
 */
class dataBase {
    /*
     * Je définit l'attribut protégé $db qui seras accessible depuis la classe et ses enfants
     */
    protected $db;
    /**
     *  __construct Crée une instance PDO qui représente une connexion à la base
     */
    public function __construct() {
        /*
         *  Le try catch récupère l'éxception généré par une errreur de connexion à la base
         */
        try {
            /*
             *  J'instancie un nouvel objet PDO et le stocke dans l'attribut $db
             *  Les informations de connexion à la base de données ont été précedement stockés dans des constantes
( inclues plus haut ).
             */
            $this->db = new PDO('mysql:host='.HOST.';dbname='.DBNAME.';charset=utf8', DBLOG, 'projetpro');
            $this->db->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

        } catch (Exception $e) {
            die('Erreur : ' . $e->getMessage());
        }
    }
    public function __destruct() {

    }
}
```

Les paramètres sont remplis grâce à des constantes définies dans le fichier configuration.

```
define('HOST','localhost');
define('DBNAME','projetpro');
define('DBLOG','usr_projetpro');
```

Développer des pages web en lien avec une base de données

Pour le développement des pages web, le patron de conception (design pattern) seras le modèle MVC (Model, view, controller).

Première vue : index.php

localisation : a la racine du dossier de projet.

```
include 'views/header.php';
include 'views/footer.php';
```

Sur cette page sont seulement visible les deux includes « header » et « footer ».

Seconde vue : header.php

localisation : /views

```
<?php
if (session_status() == PHP_SESSION_NONE) {
    session_start();
}
if ($_SERVER['PHP_SELF'] == '/index.php') {
    include_once 'controllers/headerFeatures.php';
    include_once 'controllers/userRegController.php';
    include_once 'controllers/userLoginController.php';

} else {
    include_once '../controllers/headerFeatures.php';
    include_once '../controllers/userRegController.php';
    include_once '../controllers/userLoginController.php';
}
?>
```

Dans un premier temps je démarre la session si elle n'est pas déjà demarré.

J'inclus ensuite mes différents contrôleurs nécessaire a l'enregistrement, la connexion, et la recherche, suivant la page dans laquelle le header est inclus.

```
<!DOCTYPE html>
<html>
   <head>
      <meta charset="utf-8">
      <title>Titre pro</title>
      <meta name="viewport" content="width=device-width, initial-scale=1">
      <link href="https://use.fontawesome.com/releases/v5.0.4/css/all.css" rel="stylesheet">
      <link rel="stylesheet" href="../assets/lib/bootstrap/dist/css/bootstrap.css">
      <link rel="stylesheet" href="../assets/css/style.css">
      <link rel="stylesheet" href="../assets/css/header.css">
      <link rel="stylesheet" href="../assets/css/register.css">
      <?php
      if ($_SERVER['PHP_SELF'] == '/views/account.php'){
         ?><link rel="stylesheet" href="../assets/css/account.css"><?php
      } ?>
   </head>
```

Après avoir déclarer le doctype et placer mes balises html et head, puis définis un titre

J'ajoute une balise meta qui ce nommera viewport pour contrôler la mise en page sur les navigateurs mobiles.

Ensuite je lie mes fichier css grace a la balise link.

J'ai décider d'utiliser fontawesome depuis le cdn, tandis que j'ai installé depuis grunt bootstrap3.

J'utilise une nouvelle fois PHP_SELF afin de ne charger le fichier account.css que sur la page account.php.

```
<body>
    <div class="container-fluid">
      <div class="navbar">
        <div class="row">
          <div class="col-lg-12 col-md-12 col-sm-12 col-xs-12"><!--COL principale-->
            <div id="top-header" class="row">
              <div id="brand-section" class="col-lg-3 col-md-3 col-sm-4 col-xs-12 text-center">
                <h1>Brand</h1>
                <h2 id="h2-legend">Partage, Entraide, Humanité</h2>
              </div>
              <div id="menu-container" class="col-lg-3 col-md-5 col-sm-2 col-xs-3">
                <button type="button" data-target=".navbar-collapse" data-toggle="collapse"
class="navbar-toggle"><i class="fas fa-bars"></i></button><!--Bouton pour le menu mobile-->
                <nav id="menu" class="collapse navbar-collapse"><!--Menu Collapse-->
                  <ul class="nav navbar-nav">
                    <li class="menu-items"><a href="#"><i class="fa fa-home"></i> Home</a></li>
                    <li class="menu-items"><a href="#"><i class="fas fa-map-marker"></i> Projets en
cours</a></li>
                    <li class="menu-items"><a href="#"><i class="fas fa-comment"></i>
Contact</a></li>
                  </ul>
                </nav>
              </div>
              <div id="button-group" class="col-lg-3 col-md-4 col-sm-6 col-xs-offset-3 col-xs-6">
                <form class="navbar-form" action="#" method="POST">
                  <input id="search-bar" type="search" class="form-control" name="searchbar" />
                  <button id="search-button" type="submit" class="btn btn-default dropdown-toggle"><i
class="fa fa-search"></i></button>
                </form>
                <div id="search-result" class="dropdown-menu"></div>
              </div>
              <div id="session-connect" class="col-lg-2 col-md-3 col-sm-12 col-xs-12">
<?php             if (isset($_SESSION['auth'])){ ?>
                <a href="../views/account.php" name="account-button" class="registration-link"><i
class="fas fa-user-circle"></i>Profil</a>
                <a href="../views/logout.php" name="logout-button" class="registration-link"><i
class="fas fa-user-circle"></i>Déconnection</a>
                <?php  } else {    ?>
                <a name="registration-button" class="registration-link" data-toggle="modal" data-
target="#registration"><i class="fas fa-lock"></i> Inscription</a>
                <a type="button" name="login-button" class="registration-link" data-toggle="modal" data-
target="#login"><i class="fas fa-key"></i> Connexion</a>
              <?php }

                include_once 'userRegistration.php';
                include_once 'userLogin.php'; ?>
```

Après avoir définit un container , je met un place la navbar.

J'arrange les elements grace à bootstrap en utilisant les classes « col ».

Pour les liens de connexion j'utilise la fonction isset sur $_session[‘auth’] , déclaré dans le contrôleur pour afficher des liens differents suivant si nous sommes connectés ou non.

```
                </div>
            </div>
        </div>
    </div>
</div>
<div class="row">
    <div class="col-lg-12 col-md-12 col-sm-12 col-xs-12">
        <div class="container">
            <?php
            if (isset($_SESSION['flash'])) {
                foreach ($_SESSION['flash'] as $type => $message) {
                    ?><div class="alert alert-<?= $type ?>">
                        <?= $message ?>
                    </div><?php
                }
                unset($_SESSION['flash']);
            }
            ?>
        </div>
    </div>
</div>
```

Je referme les balises restés ouvertes et puis je met en place un affichage d'erreurs depuis une session, déclaré dans le contrôleur.

modèle : users.php

localisation : /models

```
/* class users hérité de dataBase
 * Récupération de l'attibut protegé $db
 */
class users extends dataBase {
    /* Je déclare mes attributs de classe
     */
    public $id = 0;
    public $lastname = '';
    public $firstname = '';
    public $name ='';
    public $password = '';
    public $birthdate = '1900-01-01';
    public $mail = '';
    public $confirmation_token = '';
    public $confirmed_at = '1900-01-01';
    public $userTypes = 0;
    public $subTypes = NULL;
    /*
     * J'ajoute un attribut privés afin de gérer plus facilement mon prefixe
de tables (stocké dans des constantes).
     */
    private $tableName = TABLEPREFIX . 'users';
```

La constante correspondante, déclaré dans le fichier configuration,php :

```
define('TABLEPREFIX','qmsld_');
```

```
  public function __construct() {
        parent::__construct();
    }
```

Je met en place le constructeur qui correspond a celui de son parent « dataBase ».

```php
/*
    * Grâce a la méthode getSearchResult je selectionne les champs qui m'interesse afin d'afficher le résultat de la demande entrée dans la barre de recheche
    * Cette méthode prends en paramètre le mot clef.
    */
   public function getSearchResult($keyword) {
      $query = 'SELECT `id`, `lastname` FROM `' . $this->tableName . '` WHERE `lastname` LIKE :motclef';
      $searchResult = $this->db->prepare($query);
      $searchResult->bindValue(':motclef', '%' . $keyword . '%', PDO::PARAM_STR);
      $searchResult->execute();
      return $searchResult->fetchAll(PDO::FETCH_OBJ);
   }
```

Comme je stocke le résultat de cette méthode dans une variable au niveau du contrôleur, je n'hydrate pas de ce coté ci.

Premier contrôleur : headerFeatures,php (barre de recherche).

localisation : /controllers

```php
<?php

if (isset($_POST['searches'])) {
   include_once '../models/dataBase.php';
   include_once '../models/users.php';

   $search = new users();

   $keyword = $_POST['searches'];
   $searchesList = $search->getSearchResult($keyword);
   $encode = json_encode($searchesList);
   echo $encode;

} else {
   if ($_SERVER['PHP_SELF'] == '/index.php'){
      include_once 'models/dataBase.php';
      include_once 'models/users.php';
   } else {
      include_once '../models/dataBase.php';
      include_once '../models/users.php';

   }
   $searchesList = '';
}
```

Si la data de mon fichier ajax existe, j'inclus mes modèles dataBase.php et features.php afin d'avoir le lien avec ma base de données et la requête nécessaire au bon fonctionnement de la barre de recherche.

J'instancie un nouvel objet.

Puis je déclare un mot clef qui correspondras avec ce qui est entré dans la barre de recherche par l'utilisateur.

J'appelle ma méthode avec en paramètre mon mot clef, j'encode en json afin de récupérer le résultat en ajax, puis j'echo, ce qui envois ce résultat.

Sinon, j'inclus mes modèles suivant la page ou je me trouve, toujours grâce a PHP_SELF

Puis je déclare searchList comme une chaîne de caractères vide, afin d'éviter les erreurs quand aucune recherche n'as été faites.

Premier script: search,js

localisation : /assets/js

```
$('#search-button').click(function(e) {
    e.preventDefault();
    var search = $('#search-bar').val();
    if (search.length > 1) {
        $.post("../../controllers/headerFeatures.php",
                {
                    searches: search
                },
                function (searches) {
                    $.each(searches, function (searchArrayPos, profilDetails) {
                        $('#search-result').append('<a class="btn btn-default btn-block" href="views/userProfil.php?
userId=' + profilDetails.id + '">' + profilDetails.lastname + '</a>').slideDown(200);
                    });
                },
                "json");

        $('#search-result').empty();
    }
    $('#search-result').mouseleave(function () {
        $('#search-result').slideUp(400);
    });

 });
```

J'utilise la méthode préventDefault sur l'événement en paramètre afin que si l'événement n'est pas traité explicitement, son action par défaut ne soit pas pris en compte.

Je stocke d'abord ce que je veut récupérer dans ma data dans une variable ,ici, la valeur entré dans la search-bar.

Si la longueur de ce qui à été entré est supérieur à 1 caractère, je commence le traitement.

Par la méthode post j'initialise le traitement des données asynchrone Puis je cible le contrôleur avec lequel je travail.

je déclare ma data en lui passant ma variable préalablement créer.

Je démarre ensuite ma fonction de callback en lui passant en paramètre la valeur entré.

Afin que le retour coïncide bien avec ce qui a été récupérer grâce a la méthode post.

Je parcours le tableau Json que me renvois le contrôleur grace a la méthode each qui prend en paramètre la valeur retourné en post.

Puisqu'il faut avoir une action en retour, ce seras une fonction, qui prend en paramètre un index et une valeur pour le tableau récupérer depuis le contrôleur.

Mon action seras d'append le résultat (de l'afficher comme dernier enfant du parent sélectionné)

Pour cela je cible directement le champs du tableau a afficher grâce au paramètre de valeur.

Troisième vue : userRegistration,php

localisation : /views

La vue d'inscrption d'utilisateurs est une modal incluse dans le header.

```
<div id="registration" class="modal fade" tabindex="1" role="dialog"><!--INSCRIPTION MODAL-->
        <div class="modal-dialog" role="document">
                <div class="modal-content">
                        <div class="modal-header">
                                <button type="button" class="close" data-dismiss="modal" aria-label="Close">
                                        <span aria-hidden="true">&times;</span>
                                </button>
                                <h5 id="registration-title">Inscription</h5>
                        </div>
                        <div class="modal-body">
```

```
<form class="registration-form" action="#" method="POST">
        <label class="form-radio" for="private-reg">Particulier</label>
        <input type="radio" id="private-reg" name="registerType" checked value="1" />
        <label class="form-radio" for="professional-reg">Professionnel</label>
        <input type="radio" id="professional-reg" name="registerType" value="2" />
        <fieldset class="user-login form-inputs">
                <legend>Informations de connexion</legend>
                <div id="mail-control">
                        <label class="form-labels" for="mail">Adresse e-mail</label>
                         <input class="form-input" type="text" name="mail" placeholder="nom@domaine.fr" />
                 </div>
                 <div id="password-control">
                        <label class="form-labels" for="password">Mot de passe</label>
                        <input class="form-input" type="password" name="password" id="password" placeholder="8
                        caractères minimum" />
                 </div>
                 <p><i id="show-password" class="fas fa-eye"></i>Voir le mot de passe</p>
                 <div id="confirm-password-control">
                        <label class="form-labels" for="confirm-password">Confirmation du mot de passe</label>
                        <input class="form-input" type="password" name="confirm-password" placeholder="Confirmez votre
                        mot de passe" />
                  </div>
                  <p><i id="show-password" class="fas fa-eye"></i>Voir le mot de passe</p>
        </fieldset>
        <fieldset id="user-civil-status">
                <legend>Etat Civil</legend>
                <div id="lastname-control">
                        <label class="form-labels" for="lastname">Nom</label>
                        <input class="form-input" type="text" name="lastname" placeholder="Nom de famille" />

                </div>
                <div id="firstname-control">
                        <label class="form-labels" for="lastname">Prénom</label>
                        <input class="form-input" type="text" name="firstname" placeholder="Prénom" />
                </div>
                <div id="name-control">
                        <label class="form-labels" for="name">Nom</label>
                        <input class="form-input" type="text" name="name" placeholder="Nom ou raison sociale de
                        l'entreprise" />
                 </div>
                <div id="birthdate-control">
                        <label class="form-labels" for="birthdate">Date de naissance</label>
                        <input class="form-input" id="datepicker" type="date" name="birthdate" data-toggle="datepicker"/>

                </div>
                <div id="subtypes-control">
                        <label class="form labels" for="sutypes">Domaine de compétences</label>
                        <select id="subtypes" name="subtypes">
                                <option></option>
                        </select>
                </div>
        </fieldset>
        <div id="validation-group">
                 <button type="button" class="btn btn-secondary" data-dismiss="modal">Annuler</button>
                <button type="submit" id="submit" name="submit" class="btn btn-primary">Valider</button>
        </div>
</form>
```

Modèle : users,php

localisation : /models

```
public function setUser() {
    $correct = FALSE;
    $query = 'INSERT INTO `' . $this->tableName . '` '
            . '(`lastname`, `firstname`, `name`, `password`, `birthdate`, `mail`, `confirmation_token`, `id_qmsld_userTypes`, `id_qmsld_subTypes`) '
          . 'VALUES '
            . '(:lastname, :firstname, :name, :password, :birthdate, :mail, :confirmation_token, :userTypes, :subTypes)';
    $request = $this->db->prepare($query);
    $request->bindValue(':lastname', $this->lastname, PDO::PARAM_STR);
    $request->bindValue(':firstname', $this->firstname, PDO::PARAM_STR);
    $request->bindValue(':name', $this->name, PDO::PARAM_STR);
    $request->bindValue(':password', $this->password, PDO::PARAM_STR);
    $request->bindValue(':birthdate', $this->birthdate, PDO::PARAM_STR);
    $request->bindValue(':mail', $this->mail, PDO::PARAM_STR);
    $request->bindValue(':confirmation_token', $this->confirmation_token, PDO::PARAM_STR);
    $request->bindValue(':userTypes', $this->userTypes, PDO::PARAM_INT);
    $request->bindValue(':subTypes', $this->subTypes, PDO::PARAM_INT);
    if($request->execute()){
       $this->id = $this->db->lastInsertId();
       $correct = TRUE;
    }
    return $correct;
  }
```

La méthode setUser insert dans la base de données ce qui à été renseigner dans le formulaire d'inscription présent sur la vue.

J'initialise dans un premier temps une variable et lui donne False comme valeur.

Je stocke ensuite la requete dans la variable query, puis je la prepare grace à l'attribut db (récupérer de dataBase).

Je lie ensuite les attribut de la classe users aux marqueurs nominatifs grace au PDOStatement bindValue.

Si la requete s'exécute avec succès, je récupère dans l'attribut de classe « id » le dernier id insérer dans la base de données grace à la méthode lastInsertId.

Ceci me permettras de le passer un paramètre d'url du lien de confirmation d'inscription envoyer par mail depuis le controleur.

Enfin je passe la variable initialiser a false, dans un premier temps, a true puis je retourne cette variable.

Ce qui me permet de verifier coté controleur que la méthode s'est bien exécuté.

```
public function checkUserByMail(){
        $query = 'SELECT COUNT(`mail`) AS `mail` FROM `' . $this->tableName . '` WHERE `mail` = :mail';
        $request = $this->db->prepare($query);
        $request->bindValue(':mail', $this->mail, PDO::PARAM_STR);
        $request->execute();
        return $request->fetch(PDO::FETCH_OBJ);
    }
```

La méthode checkUserByMail compte les adresses similaire à celle entré dans le champ mail du formulaire d'inscription mail, dans la table users.

Deuxième controleur : userRegController.php

localisation : /controllers

```
/*Grâce a PHP_SELF je change les chemins des includes par rapport aux pages sur laquelle je me situe
 * si elles se situes dans dans dossiers differents
 * ICI 'index.php' se situe a la racine do mon dossier
 * les autres pages se situant dans le dossier views
 */
if ($_SERVER['PHP_SELF'] == '/index.php') {
    include_once 'models/dataBase.php';
    include_once 'models/users.php';
    include_once 'models/configuration.php';
} else {
    include_once '../models/dataBase.php';
    include_once '../models/users.php';
    include_once '../models/configuration.php';
}
/**
 * Instanciation de ma classe users
 */
$user = new users();
/**
 * Création du tableau d'erreur du formulaire d'inscription
 * ce tableau récuperaras champ par champ les erreur définis, si elles sont générées
 */
$errors = array();
    /**
     * Vérification champ par champ du formulaire d'inscription
     * Les vérification se font en Ajax avex $.post
     */
```

```
if (isset($_POST['checkInput'])) {
        if ($_POST['check'] == 'ajaxlastname') {
            if (empty($_POST['checkInput'])) {
                $errors['lastname'] = 'Veuillez renseigner votre nom de famille';
            } elseif (!preg_match(REGTEXT, $_POST['checkInput'])) {
                    $errors['lastname'] = 'Votre nom ne doit comporter que des lettres';
            }
            if (!empty($errors['lastname'])) {
                echo json_encode($errors);
            }
        }
```

La vérification des champs de formulaire se feras de manière asynchrone, couplé avec le fichier registration,js
Je vérifie d'abord si la valeur du champs selectionné existe, puis que le nom du champs coïncide bien.
Puis si le champs est vide ou s'il ne correspond pas à la regex, je stock un message d'erreur personnalisé puis j'encode en json le tableau s'il n'est pas vide.
Les champs firstname et birthdate se gèrent exactement de la même façon.

```
 if ($_POST['check'] == 'ajaxpassword') {
        $user->password = $_POST['checkInput'];
        if (strlen($_POST['checkInput']) < 8) {
            $errors['password'] = 'Votre mot de passe doit comporter 8 caractères minimum';
        }
        if (!empty($errors['password'])) {
            echo json_encode($errors);
        }
    }
    if ($_POST['check'] == 'ajaxconfirm-password') {
        if (empty($_POST['checkInput'])){
            $errors['confirm-password'] = 'Veuillez confirmer votre mot de passe';
        } elseif ($_POST['checkInput'] != $user->password){
            $errors['confirm-password'] = 'La confirmation de mot de passe à echouée';
        }
        if (!empty($errors['confirm-password'])){
            echo json_encode($errors);
        }
    }
```

Si le champs selectionné coincide avec le nom du champ voulu, ici password.
Je passe a l'attribut password la valeur du champs récupéré en post via ajax.
Si la longueur de cette valeur est inférieur a 8 caractères, je stocke une erreur dans le tableau d'erreur.
Pour le champs de confirmation, après avoir vérifier le nom du champ, je stocke une erreur si le champ est vide.
Ensuite je vérifie si ce qui est entré dans ce champs ci coincide avec ce qui à été entré dans le champ précédent, la valeur ayant été stocké dans l'attribut password.
Puis je stock une erreur si ce n'est pas le cas.

```
if ($_POST['check'] == 'ajaxmail') {
        $user->mail = $_POST['checkInput'];
        $existingMail = $user->checkUserByMail();
        $checkMail = intval($existingMail->mail);
        if (empty($_POST['checkInput'])){
           $errors['mail'] = 'Veuillez renseigner votre adresse email';
        } elseif (!preg_match(REGMAIL, $_POST['checkInput'])) {
              $errors['mail'] = 'E-mail non valide. Exemple : contact@domaine.fr';
        } elseif ($checkMail == 1) {
           $errors['mail'] = 'Cette adresse mail est déjà enregistrée';
        }
        if(!empty($errors['mail'])){
           echo json_encode($errors);
        }
    }
```

Après avec vérifier le nom du champ, je stocke la valeur de ce champs dans l'attribut mail, puis j'appelle ma methode qui va compter les adresse mail similaire et renvoyer 0 ou 1 car on ne peut pas entrer deux adresses similaires dans la base de données.
Je stocke le résultat dans une variable puis je l'intval pour récupérer un résultat de type int et non string et je restocke ce résultat dans une variable.
Je vérifie ensuite si le champ est vide, si il ne correspond pas a sa regex et enfin si une adresse similaire existe dans la base de données.

```
if (!empty($_POST['password']) && !empty($_POST['mail'])) {
   $user->mail = strip_tags($_POST['mail']);
   $user->password = password_hash($_POST['password'], PASSWORD_BCRYPT);
   $setToken = password_hash('connection', PASSWORD_BCRYPT);
   $user->confirmation_token = $setToken;
   $userTypes = $_POST['registerType'];
   $user->userTypes = intval($userTypes);
   if (!empty($_POST['lastname']) && !empty($_POST['firstname']) && !empty($_POST['birthdate']) &&
count($errors) == 0) {
      $user->lastname = strip_tags($_POST['lastname']);
      $user->firstname = strip_tags($_POST['firstname']);
      $user->birthdate = strip_tags($_POST['birthdate']);
   } elseif (!empty($_POST['name']) && count($errors) == 0){
      $user->name = strip_tags($_POST['name']);
   }
   if($user->setUser()){
      $object = 'Confirmation de votre compte';
      $content = 'Afin de valider votre inscription, veuillez cliquer ce le lien qui suit';
      $tokenLink = 'http://titrepro/views/confirm.php?id=' . $user->id . '&token=' . $user->confirmation_token;
      mail($user->mail, $object, $content . ' ' .  $tokenLink);
      $_SESSION['flash']['success'] = 'Afin de valider votre compte, un email vous de confirmation vous a été
envoyé';
      header('location: views/login.php');
      exit;
   }  else {
      $_SESSION['flash']['danger'] = 'Un problème est survenu, veuillez contacter un administrateur';
   }
}
```

Si les champs password et mail ne sont pas vides, je supprime les éventuels balises html ou php de la chaine mail et je donne sa valeur à l'attribut correspondant.

Je hash ensuite le mot de passe grace a password_hash et l'algorithme Crypt_blowfish qui me retourneras une chaine de 60 caractères, puis je donne sa valeur à l'attribut correspondant.

Je génére un token grace a password_hash puis le stocke dans une variable, que je passe ensuite a l'attribut confirmation_token.

Je stocke le type d'utilisateur, (par son id), dans un variable, puis l'intval et le s afin qu'il ne soit pas retourner comme chaine de caractères puis je donne sa valeur à l'attribut correspondant.

Suivant le type d'utilisateur, je vérifie si les 3 champs propres à un particulier (type 1) ne sont pas vides, et si le tableau d'erreur est à 0. Ou si l'unique champ propre à un utilisateur professionnel n'est pas vide et si le tableau d'erreur est à 0.

Je passe la valeur du post à leurs attributs respectifs.

Si la méthode setUser me renvois TRUE, j'envoie un mail de confirmation grâce a l'adresse mail de l'utilisateur précedemment renseigné, avec un lien d'activation qui aura en paramètre l'id de l'utilisateur ainsi que le token. Puis je stocke un message qui prévient l'utilisateur de l'envoi du mail dans une session.

J'utilise la multi dimension, pour pouvoir stocké a l'avenir dans le champ flash, deux type differents de messages différents (success et danger).

Enfin grâce a header je redirige l'utilisateur vers la page login, puis arrete l'execution du code .

```
var verifInputs = (function () {
    $.post('../../controllers/userRegController.php',
            {
                check: 'ajax' + $(this).attr('name'),
                checkType: $('input[name=register-type]').val(),
                checkInput: $(this).val()
            }, function (errors) {
        $.each(errors, function (arrayId, error) {
            var input = 'input[name=' + arrayId + ']';
            $(input).parent().append('<p class="form-errors">' + error + '</p>');
            $(input).css('border-bottom', '2px solid red');
        });
    },
            'json');
    $(this).css('border-bottom', '2px solid #2196F3');
    var empty = $(this).next();
    $(empty).remove();
});
    $('input[name=lastname]').keyup(verifInputs);
    $('input[name=firstname]').keyup(verifInputs);
    $('input[name=password]').blur(verifInputs);
    $('input[name=confirm-password]').blur(verifInputs);
    $('input[name=birthdate]').change(verifInputs);
    $('input[name=mail]').blur(verifInputs);
```

Je fais le traitement du formulaire de manière asynchrone grace a la methode post d'ajax.
Dans ce cas ci j'utilise this pour le ciblage, puis un paramètre de données differents, ciblant plus précisement pour les radios donnant le type d'utilisateur.
Comme precedemment, mon premier parametre me retourneras le nom du champs selectionné avec le mot clef ajax devant.
Le dernier, la valeur de ce champs.
Dans ma fonction de callback, cette fois ci, j'utilise l'index du tableau json afin de récupérer le nom de l'input (c'est pour cela que je n'est qu'un élément par champs dans mon tableau d'erreur, qui change sa valeur suivant la situation) .
J'append ensuite la valeur du tableau d'erreur, enfin, je vide le message, afin d'éviter une supperposition d'erreurs.

Quatrième vue : userLogin.php

localisation /views

```
<div id="login" class="modal fade" tabindex="1" role="dialog"><!--INSCRIPTION MODAL-->
        <div class="modal-dialog" role="document">
          <div class="modal-content">
            <div class="modal-header">
              <button type="button" class="close" data-dismiss="modal" aria-label="Close">
                <span aria-hidden="true">&times;</span>
              </button>
              <h5 id="login-title">Connexion</h5>
            </div>
            <div class="modal-body">
              <form class="login-form" action="#" method="POST">
                  <label class="form-labels" for="mail-connect">Adresse e-mail</label>
                  <input class="form-input" type="text" name="mail-connect"
placeholder="nom@domaine.fr" />
                  <label class="form-labels" for="password-connect">Mot de passe</label>
                  <input class="form-input" type="password" name="password-connect" placeholder="8
cractères minimum" />
                <div id="validation-group">
                  <button type="button" class="btn btn-secondary" data-
dismiss="modal">Annuler</button>
                  <button type="submit" id="submit" name="submit" class="btn btn-
primary">Valider</button>
                </div>
              </form>
            </div>
          </div>
        </div>
      </div><!--INSCRIPTION MODAL FIN-->
```

modèle : users.php

localisation : /models

```
public function connectUser(){
      $correct = FALSE;
      $query = 'SELECT `password` FROM `' . $this->tableName . '` WHERE `mail` = :mail AND
`confirmed_at` IS NOT NULL';
      $request = $this->db->prepare($query);
      $request->bindValue(':mail', $this->mail, PDO::PARAM_STR);
      $request->execute();
      $userExist = $request->rowCount();
      if($userExist == 1){
         $result = $request->fetch(PDO::FETCH_OBJ);
         $this->password = $result->password;
         $correct = TRUE;
      }
      return $correct;
   }
```

La requête selectionne la ligne mot de passe de la table users, la ou le mail correspond a celui entré par l'utilisateur dans le formulaire.
Après avoir exécuter je compte le nombre de ligne retourné par la requête et si le compte est egal a un (si cela me retourne un utilisateur inscrit) je récupère le resultat de cette requête sous forme d'objet puis je donne a l'attribut de classe password, le password retourné par la requête.

Troisième controller : userLogin.php

localisation : /controllers

```php
$connectedUser = new users();
if (!empty($_POST['mail-connect']) && !empty($_POST['password-connect'])) {

    $connectedUser->mail = $_POST['mail-connect'];

    if ($connectedUser->connectUser()) {
        if (password_verify($_POST['password-connect'], $connectedUser->password)) {
            if ($connectedUser->getUserByMail()){
                $_SESSION['auth']['id'] = $connectedUser->id;
                $_SESSION['auth']['confirmed_at'] = $connectedUser->confirmed_at;
                $_SESSION['auth']['mail'] = $connectedUser->mail;
                $_SESSION['auth']['password'] = $connectedUser->password;
                $_SESSION['auth']['name'] = $connectedUser->name;
                $_SESSION['auth']['lastname'] = $connectedUser->lastname;
                $_SESSION['auth']['firstname'] = $connectedUser->firstname;
                $_SESSION['auth']['birthdate'] = $connectedUser->birthdate;
            }
            if ($_SERVER['PHP_SELF'] == '/index.php') {
                header('location: views/account.php');
                exit();
            } else {
                header('location: account.php');
                exit();
            }
        } else {
            $_SESSION['flash']['danger'] = 'L\'identifiant ou le mot de passe saisi est incorrect';
        }
    } else {
        $_SESSION['flash']['danger'] = 'L\'identifiant ou le mot de passe saisi est incorrect';
    }
} else {
    $_SESSION['flash']['danger'] = 'Veuillez remplir tous les champs';
}
```

J’instancie ma classe, puis je verifie si les champs mail et password ne sont pas vides.
Je donne à l’attribut mail la valeur du champs de formulaire mail.
Si ma méthode connectUser renvois TRUE et si le mot de passe entré correspond a celui stocké dans l’attribut password et si la méthode getUserByMail me renvois TRUE aussi je stocke dans la session (auth) tout les champs retourné par cette dernière methode .

modèle : users.php

localisation : /models

```
public function getUserByMail(){
        $correct = FALSE;
        $query = 'SELECT `id`, `lastname`, `firstname`, `name`, `password`, DATE_FORMAT(`birthdate`, \'%d-%m-%Y\') AS `birthdate`, `confirmed_at`, `id_qmsld_subTypes`, `id_qmsld_userTypes` '
                . 'FROM `' . $this->tableName . '` WHERE `mail` = :mail';
        $request = $this->db->prepare($query);
        $request->bindValue(':mail', $this->mail, PDO::PARAM_STR);
        if ($request->execute()){
            $result = $request->fetch(PDO::FETCH_OBJ);
            $this->id = $result->id;
            $this->lastname = $result->lastname;
            $this->firstname = $result->firstname;
            $this->name = $result->name;
            $this->password = $result->password;
            $this->birthdate = $result->birthdate;
            $this->confirmed_at = $result->confirmed_at;
            $this->subTypes = $result->id_qmsld_subTypes;
            $this->userTypes = $result->id_qmsld_userTypes;
            $correct = TRUE;
        }
        return $correct;
    }
```

Dans cette méthode la requête selectionne tout ce qui peut être renseigné à l'inscription dans la table users la ou le mail est celui entré à la connexion.
J'hydrate toute les ligne et donne leurs valeurs aux attributs de classe correspondant.

Quatrième controller : confirmController.php

localisation : /controller

Quand un utilisateur s'inscrit, au-delà de la connexion, une fois confirmer par le clic sur l'url de confirmation, l'utilisateur est automatiquement connecté.

```
$toConfirm = new users();

$toConfirm->id = $_GET['id'];
$token = $_GET['token'];
session_start();
$toConfirm->getTokenById();
```

Après avoir récuprérer l'id et le token de l'utilisateur par la methode get dans l'url de confirmation, je demarre la session afin de pouvoir connecté automatiquement l'utilisateur à la confirmation d'inscription.
J'appelle ensuite ma méthode getTokenById qui selectionne dans la table users le token de confirmation la ou l'id est égal a l'id récupéré plus tôt.

```
if ($toConfirm->confirmation_token == $token) {
    $toConfirm->setConfirmedUser();
    if ($toConfirm->getUserById()){
        $_SESSION['auth']['type'] = $toConfirm->userTypes;
        $_SESSION['auth']['id'] = $toConfirm->id;
        $_SESSION['auth']['mail'] = $toConfirm->mail;
        $_SESSION['auth']['confirmed_at'] = $toConfirm->confirmed_at;
        $_SESSION['auth']['password'] = $toConfirm→password;
        $userTypes = intval($toConfirm→userTypes) ;
        if ($userTypes === 1){
            $_SESSION['auth']['lastname'] = $toConfirm->lastname;
            $_SESSION['auth']['firstname'] = $toConfirm->firstname;
            $_SESSION['auth']['birthdate'] = $toConfirm->birthdate;
        } else {
            $_SESSION['auth']['name'] = $toConfirm->name;
            $_SESSION['auth']['subTypes'] = $toConfirm->subTypes;
        }
    }
    $_SESSION['flash']['success'] = 'Votre compte a bien été validé';
    header('location: account.php');
    exit();
} else {
    $_SESSION['flash']['danger'] = 'Ce token n\'est plus valide';
    header('location: login.php');
    exit();
}
```

Je compare ensuite le token présent dans l'url et celui que me retourne ma requête et si les deux sont similaires,
Par la méthode setConfirmedUser, je passe le token en NULL dans ma table et je renseigne dans la date de confirmation.
Je verifie que ma méthode getUserById retourne bien TRUE, si c'est le cas, cette dernière méthode étant hydraté, je recupères directement toute les lignes renseignés de la table users et stocke leurs valeur dans la session ('auth').
Avec un nouveau if, je stocke les lignes qui en dépendent suivant l'id de leurs type.
Si la récupération par token s'effectue correctement, je redirige l'utilisateur vers la page account.php, sinon j'envoie un message d'erreur par ma session ('flash') et je redirige vers la page de connexion.
La controller offrant une redirection dans tout les cas, la vue de confirmation est vide, seul le controller y est inclus.

modèle : users.php

localisation : /models

```
public function setConfirmedUser() {
        $query = 'Update `' . $this->tableName . '` SET `confirmation_token` = NULL,
`confirmed_at` = NOW() WHERE `id` = :id';
        $request = $this->db->prepare($query);
        $request->bindValue(':id', $this->id, PDO::PARAM_INT);
        return $request->execute();
    }
```

```
public function getUserById() {
      $correct = FALSE;
      $query = 'SELECT `lastname`, `firstname`, `name`, `password`, `birthdate`,
`mail`, `confirmed_at`, `id_qmsld_subTypes`, `id_qmsld_userTypes` '
                . 'FROM `' . $this->tableName . '` WHERE `id` = :id';
      $request = $this->db->prepare($query);
      $request->bindValue(':id', $this->id, PDO::PARAM_INT);
      if ($request->execute()){
         $result = $request->fetch(PDO::FETCH_OBJ);
         $this->lastname = $result->lastname;
         $this->firstname = $result->firstname;
         $this->name = $result->name;
         $this->mail = $result->mail;
         $this->password = $result->password;
         $this->birthdate = $result->birthdate;
         $this->confirmed_at = $result->confirmed_at;
         $this->subTypes = $result->id_qmsld_subTypes;
         $this->userTypes = $result->id_qmsld_userTypes;
         $correct = TRUE;
       }
       return $correct;
   }
```

La méthode getUserById selectionne tout les champs de la table users à part le token par l'id de l'utilisateur.
j'hydrate mon objet en passant a mes attributs, les valeurs renvoyé par ma requête.

Cinquième vue : account.php
localisation : /views

```
<form action="#" method="POST"></form>
```

La page account affiche les données de l'utilisateur par la session (auth).
Un formulaire y est déployer, afin de gérer la modification de profil.

```
<?php if ($_SESSION['auth']['type'] == 'particulier') { ?>
            <div id="profil-lastname-control">
                  <label class="form-labels" for="profil-lastname">Nom : </label>
                  <input class="form-input account-inputs" type="text" name="profil-lastname" id="profil-
                  lastname" readonly value="<?= $_SESSION['auth']['lastname'] ?>" />
            </div>
<?php } ?>
```

L'affichage est gérer suivant si le type stocké dans la session (auth) est égal a particulier ou professionnel.

Cinquième contrôleur : accountController,php

localisation : /controllers

```
$account = new users();
$account->id = $_SESSION['auth']['id'];
$myProfil = $account->getTypesById();
$_SESSION['auth']['type'] = $myProfil->type;
$_SESSION['auth']['subType'] = $myProfil->subTypesName;
```

J'instancie ma classe puis je donne la valeur de l'id stocké dans ma session a mon attribut id.
j'appelle ma méthode getTypesById et je stocke ce qu'elle renvois dans une variable.
Je recupère depuis celle ci le type et le sous type que j'envois dans ma session, sous ('auth').

```
$updateErrors = array();
if ($_SESSION['auth']['type'] == 'particulier'){
    if (empty($_POST['profil-lastname'])) {
        $updateErrors['emptyLastname'] = 'Veuillez renseigner votre nom de famille';
    } else {
        if (!preg_match(REGTEXT, $_POST['profil-lastname'])) {
            $updateErrors['regLastname'] = 'Votre nom ne doit comporter que des lettres';
        }
    }

    if (empty($_POST['profil-firstname'])) {
        $updateErrors['emptyFirstname'] = 'Veuillez renseigner votre prénom';
    } else {
        if (!preg_match(REGTEXT, $_POST['profil-firstname'])) {
            $updateErrors['regFirstname'] = 'Votre prénom ne doit comporter que des lettres';
        }
    }
    if (empty($_POST['profil-old'])) {
        $updateErrors['emptyBirthdate'] = 'Veuillez renseigner votre date de naissance';
    } else {
        if (!preg_match(REGDATE, $_POST['profil-old'])) {
            $updateErrors['regBirthdate'] = 'La date doit être au format : jj/mm/aaaa';
        }
    }
} else {
    if (empty($_POST['profil-name'])) {
        $updateErrors['emptyName'] = 'Veuillez renseigner le nom de votre entreprise';
    }
}
```

Après avoir mis en place mon tableau d'erreurs, je verifie le type d'utilisateur (stocké precedemment dans ma session), et je traite les champs de mon formulaire par rapport à ce type.
Je vérifie ensuite, que les champs soit remplis et qu'ils correspondent aux différentes regexs, un message d'erreur étant envoyé à mon tableau d'erreur si ce n'est pas le cas.

```
else {
        if (empty($_POST['profil-name'])) {
            $updateErrors['emptyName'] = 'Veuillez renseigner le nom de votre entreprise';
        }
}
if (empty($_POST['profil-mail'])) {
        $updateErrors['emptyMail'] = 'Veuillez renseigner votre adresse email';
} else {
        if (!preg_match(REGMAIL, $_POST['profil-mail'])) {
            $updateErrors['regMail'] = 'E-mail non valide. Exemple : contact@domaine.fr';
        }
}
```

Toujours en rapport en type, dans le else je finis le traitement des champs, dans le cas ou l'utilisateur n'est pas un particulier.

```
if (isset($_SESSION['auth']['mail']) && isset($_SESSION['auth']['password'])) {
   $account->mail = $_SESSION['auth']['mail'];
   $account->password = $_SESSION['auth']['password'];
   If (!empty($_POST['profil-mail']) && count($updateErrors) === 0) {
      $account->mail = $_POST['profil-mail'];
      $_SESSION['auth']['mail'] = $account->mail;
   }
   if (isset($_SESSION['auth']['lastname']) && isset($_SESSION['auth']['firstname']) &&
isset($_SESSION['auth']['birthdate'])) {
      $account->lastname = $_SESSION['auth']['lastname'];
      $account->firstname = $_SESSION['auth']['firstname'];
      $account->birthdate = $_SESSION['auth']['birthdate'];
      If (!empty($_POST['profil-lastname']) && !isset($_POST['profil-firstname']) && isset($_POST['profil-
old']) && count($updateErrors) === 0) {
         $account->lastname = $_POST['profil-lastname'];
         $account->firstname = $_POST['profil-firstname'];
         $account->birthdate = $_POST['profil-old'];
         $_SESSION['auth']['lastname'] = $account->lastname;
         $_SESSION['auth']['firstname'] = $account->firstname;
         $_SESSION['auth']['birthdate'] = $account->birthdate;
      }
   }
   if (isset($_SESSION['auth']['name'])) {
      $account->name = $_SESSION['auth']['name'];
      if (isset($_POST['profil-name']) && count($updateErrors) == 0) {
         $account->name = $_POST['profil-name'];
         $_SESSION['auth']['name'] = $account->name;
      }
   }
   $account->updateUser();
   $_SESSION['flash']['success'] = 'Votre compte a été modifié avec succès';
}
```

Je vérifie que mes sessions mail et password existent, si c'est le cas je passe leurs valeurs aux attributs correspondants.
Si l'input correspondant n'est pas vide et que le tableau d'erreur est vide, je donne à l'attribut, la valeur de l'input, puis je met à jour la session depuis l'attribut. Même traitement pour les autres champs.
j'appelle ensuite ma méthode updateUser afin de mettre à jour les donées dans ma table.

modèle : users.php

localisation : /models

```
public function getTypesById() {
    $query = 'SELECT `type`, `subTypesName` '
           . 'FROM `qmsld_users` '
         . 'INNER JOIN `qmsld_userTypes` '
           . 'ON `qmsld_users`.`id_qmsld_userTypes` = `qmsld_userTypes`.`id` '
         . 'INNER JOIN `qmsld_subTypes` '
           . 'ON `qmsld_userTypes`.`id` = `qmsld_subTypes`.`id_qmsld_userTypes` '
         . 'WHERE `qmsld_users`.`id` = :id';
    $request = $this->db->prepare($query);
    $request->bindValue(':id', $this->id, PDO::PARAM_INT);
    $request->execute();
    return $request->fetch(PDO::FETCH_OBJ);
  }
```

Je selectionne le type de la table users à qui je joins la table userTypes depuis la clef étrangère id_userTypes de la table users avec l'id correspondante de la table userTypes .
J'utilise Inner Join qui compare deux tables et retourne tous les enregistrements comportant une concordance.
Je fais une double jointure pour retourner aussi les concordances de la tables subTypes avec sa clef étrangère de la table users.

```
public function updateUser() {
      $query = 'UPDATE `' . $this->tableName . '` '
             . 'SET `lastname` = :lastname, `firstname` = :firstname, `name` = :name, `birthdate` = :birthdate,
`password` = :password '
           . 'WHERE id = :id';
      $request = $this->db->prepare($query);
      $request->bindValue(':id', $this->id, PDO::PARAM_INT);
      $request->bindValue(':lastname', $this->lastname, PDO::PARAM_STR);
      $request->bindValue(':firstname', $this->firstname, PDO::PARAM_STR);
      $request->bindValue(':name', $this->name, PDO::PARAM_STR);
      $request->bindValue(':birthdate', $this->birthdate, PDO::PARAM_STR);
      $request->bindValue(':password', $this->password, PDO::PARAM_STR);
      return $request->execute();
   }
```

Grace a la méthode updateUser je met a jour les enregistrements de la table users .

Fin du controller,

```
if(password_verify($_POST['password-delete'], $account->password)){
        $account->eraseUser();
        session_destroy();
        header('location : ../index.php');
        exit();
    }
```

Méthode de suppression de l'utilisateur

```
public function eraseUser() {
        $query = 'DELETE FROM `' . $this->tableName . '` WHERE `qmsld_users`.`id` = :id';
        $request = $this->db->prepare($query);
        $request->bindValue(':id', $this->id, PDO::PARAM_INT);
        return $request->execute();
    }
```

Destructeur

```
        public function __destruct(){
         parent::__destruct();
        }
```

Conclusion

Cela fais longtemps, bien avant d'entrer en formation à l'école du numérique du noyonnais, que ce projet mûrit dans ma tête. Ce projet me tenant vraiment à cœur , j'y ai longtemps refléchis, côté terrain, n'ayant pas les compétences avant la formation pour développer une plateforme web, vitrine et gestionnaire de projets et d'équipes.

C'est au bout de quelques mois, voyant mes compétences accroître, et à la demande de l'équipe pédagogique de commencer à penser un projet que j'y ai vu l'opportunité de développer sur ce thème, ce qui est ensuite devenu une évidence, faire tourner tout un projet caritatif au tour d'une plateforme d'échange, de partage, et pouvoir y gérer les apport extérieurs, les contributeurs etc..

Ce projet a mûrit, et mûrit encore, et il me tiens à cœur de continuer de le développer, que ce soit en parallèle d'une licence ou d'un contrat, pour espérons pouvoir développer tout cette écosystème.