

SUIVI SIO Tutorat

Documentation technique

Pré-requis:

HTML/CSS, PHP, JS,JQUERY (et l'objet AJAX), le Framework bootstrap

Introduction:

L'application de gestion de tutorat est codée en HTML/CSS, Javascript, JQuery, PH, Ajax. Elle permettra d'entrer des rapports de rendez vous avec des élèves, ou sa famille. Elle utilise le Framework [bootstrap version 4](#) pour gérer le responsive design ainsi que pour les boutons, tableaux, input, ...

Une grande partie des classes utilisées sont issues de bootstrap. Pour plus d'informations sur ces classes et leurs rôles, visitez le site web de bootstrap.

Les éléments de bootstrap intégrés aux sites sont:

- bouton
- input
- select
- label
- fenêtre modal
- navbar
- tableau

Le site est stocké sur le serveur **172.16.255.14/~tutorat**.








L'identifiant et le mot de passe du compte ftp est **tutorat/sousfifre**.

L'identifiant de la BDD est **tutorat** et le mot de passe est **tutorat**.

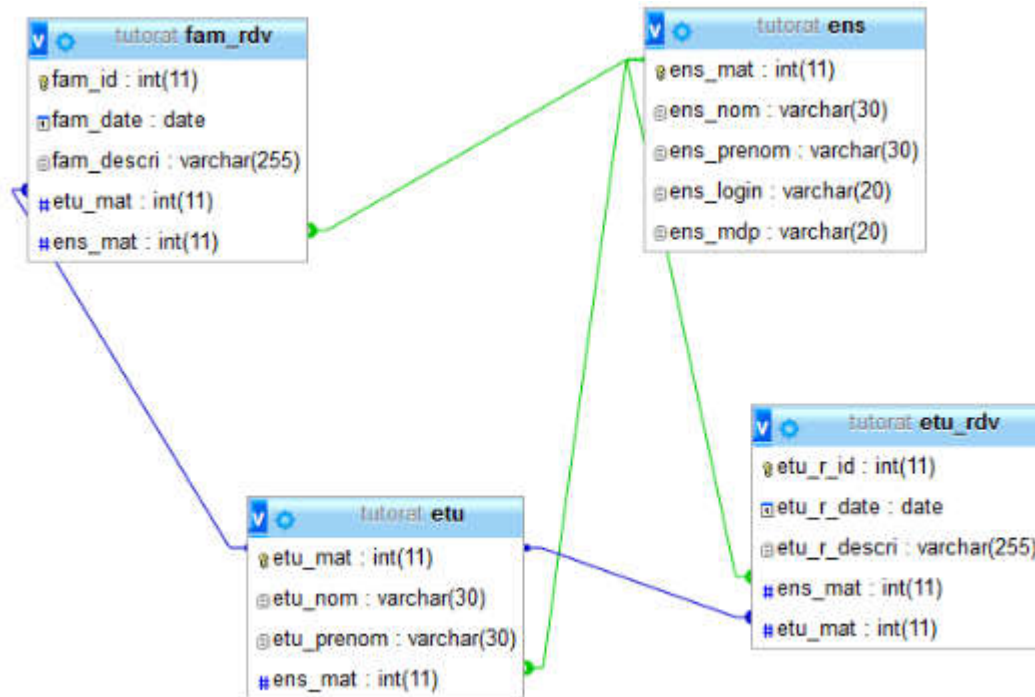
Pour accéder à l'application, il faut un compte enseignant (**gachea/gachea**).

Lien github: <https://github.com/AlexandreCharles/tutorat-projet>

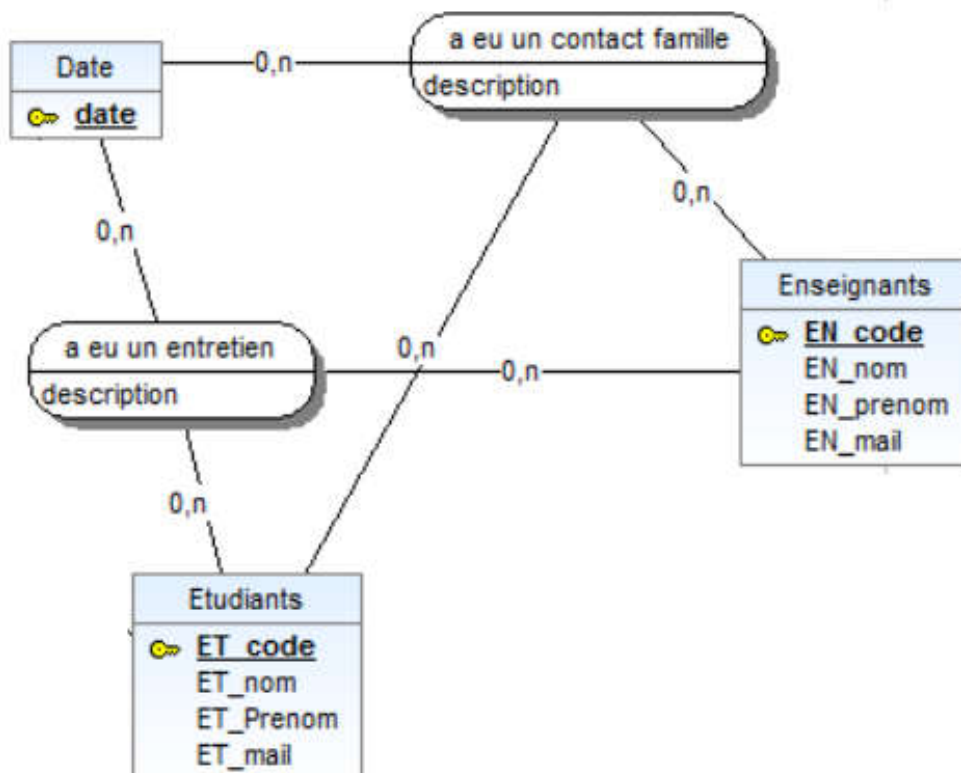
Voici l'arborescence des fichiers de l'application.

 bdd	16/11/2016 10:58	Dossier de fichiers
 css	16/11/2016 10:35	Dossier de fichiers
 csv	16/11/2016 10:35	Dossier de fichiers
 image	16/11/2016 10:56	Dossier de fichiers
 js	16/11/2016 10:35	Dossier de fichiers
 php	16/11/2016 10:35	Dossier de fichiers
 index.php	28/09/2016 15:04	Fichier PHP

Voici le MRD de l'application.



Voici le MCD de l'application.



I- Structure du code

Il y a plusieurs fichiers inclus dans les pages du site (config.php, haut.php, bas.php).

Il ne faut pas insérer de l'html dans la page web avant l'inclusion de haut.php !
Header location renvoi des erreurs si du code HTML est déjà présent dans la page.

```

1  <?php
2  $titrePage= "Index";// sous titre de la page
3  include 'config.php';//fichier de conf; test session, timeout session ...
4  //code php
5  include 'haut.php';//code html generale du site
6  ?>
7  <!-- contenu de la page html -->
8  <?php
9  include 'bas.php';//footer du site
10 ?>
11 <!--script js-->

```

Voici config.php:

```

<?php
session_start();
//test session timeout
if (isset($_SESSION['timeout']) && $_SESSION['timeout']+ 100 * 60 < time()) {
    header('Location: deconnexion.php');
}
//test session existence
if (!isset($_SESSION['ens_mat'])) {
    header('Location: login.php');
}
//reinitialiser le timeout
$_SESSION['timeout'] = time();
//la bdd
include 'BDD.php';
$connexion=new BDD('tutorat');
?>

```

Dans config.php, la session est ouverte, puis nous testons si le timeout n'est pas fini. Si c'est le cas nous déconnectons l'utilisateur, sinon nous le remettons à zero puis nous testons l'existence de la session. Enfin nous incluons BDD.php et on l'instancie.

L'ensemble des fichiers js, style, framework, l'ouverture des sessions et la connexion à la base de données sont mis en place dans ces include, il ne faut pas les rajouter !

II- Importation CSV

L'application permet l'importation de fichiers csv, pour créer des étudiants. Le fichier ne doit pas contenir de légende et possédera 2 colonnes (**nom;prenom**).

```
ANDRE;Jonhhy  
CARTIER; Joévin  
CHARLES; Alexandre  
CHÂTEAU; Mathieu  
CHIE BONNE SAN; Bryan  
DAHMANI;Mouna  
DAUPHIN;Florian  
DESSIRIEX;Brian  
FOUCAULT;Antoine
```

Il n'y a pas besoin des noms de colonnes !

La page où se trouve l'importation CSV est **add_etu.php**. Cette page possède deux parties: la création ligne par ligne des étudiants et l'importation csv. Voici le code pour l'importation csv.

Dans cette application, nous vérifions d'abord que le fichier est bien un fichier CSV en vérifiant l'extension de celui-ci. Nous vérifions aussi la taille maximale du fichier qui est fixé à 1Mo. On donne au fichier un id créé aléatoirement pour éviter les doublons dans le cas où plusieurs utilisateurs feraient une importation CSV simultanée avec un fichier CSV du même nom. On rajoute à cet id l'extension CSV afin de lui constituer un nouveau nom. Le fichier est ensuite transféré de la machine du client au serveur dans un dossier où il sera temporaire. On vérifie ensuite que le flux est bien ouvert entre le fichier CSV et le serveur avant d'utiliser le fichier CSV pour importer ligne par ligne les données qu'il contient dans la base de donnée grâce à une boucle «tant que» et une requête SQL d'insertion. On ferme ensuite le flux. Si le fichier n'est pas valide, une ligne de texte apparaît indiquant qu'il faut sélectionner un fichier CSV.

```

<!-- importation csv -->
<div id='import' class='col-xs-12'>
  <h2 class='col-md-12'><i class="fa fa-users" aria-hidden="true"></i> Importation CSV</h2>
  <br />
  <div class='row'>
    <form method="post" action="" class='list-group-item col-xs-12' enctype="multipart/form-data">
      <label for="mon_fichier">Fichier (format CSV | Max. 1 Mo) :</label><br />
      <input type="hidden" name="MAX_FILE_SIZE" value="1048576" />
      <input type="file" name="mon_fichier" value="mon_fichier" /><br>
      <input type="submit" name="submit" class="btn btn-info" value="Envoyer" /></br>
    </form>
  </div>
</div>
<br />
<?php
  $etu_nom='';
  $etu_prenom='';

  if(isset($_FILES['mon_fichier'])){
    $extensions_valides = array( 'csv' );
    $extension_upload = strtolower( substr( strrchr($_FILES['mon_fichier']['name'], '.'), 1) );

    if ( in_array($extension_upload,$extensions_valides) ){
      //Vérification de l'extension du fichier
      echo "Extension correcte<br>";

      //mkdir('../csv/1/', 0777, true);
      $id_membre=md5(uniqid(rand(), true));
      $nom = "../csv/{$id_membre}.{$extension_upload}";
      $nonf="{$id_membre}.{$extension_upload}";
      $resultat = move_uploaded_file($_FILES['mon_fichier']['tmp_name'],$nom);
      if ($resultat)
      {
        echo "Transfert réussi<br>";
      }else
      {
        echo "transfert échoué<br>";
      }

      //contrôlfile a faire avec upload
      $row=1;
      $nom=$_FILES['mon_fichier']['name'];

      if (($handle = fopen("../csv/$nonf", "r")) !== FALSE) {
        while (($data = fgetcsv($handle, 1000, ";","")) !== FALSE) {
          $num = count($data);
          $row++;
          $etu_nom=$data[0];
          $etu_prenom=$data[1];
          $requete="insert into etu (etu_nom,etu_prenom,ens_mat) values ('$etu_nom','$etu_prenom',0);";
          $message = $connexion->insert($requete);
          if ($message==""){
            $message="Ligne bien insérée.";
          }
          else{
            $message="Problème d'insertion";
          }
        }
        fclose($handle);
      }
    }else{
      echo "Selectionner un fichier CSV";
    }
  }
}

```

III- La gestion des dates:

Le type date n'étant pas opérationnel sur tous les navigateurs, nous avons dû gérer les dates pour les navigateurs ne l'acceptant pas. Nous demandons aux utilisateurs d'entrer une date en format français. (lorsqu'un calendrier est utilisé, le format est en anglais). Nous testons la date avec les expressions régulières (en JS et en PHP) puis nous transformons la date en format anglais.

Code PHP

```
//date
$date=addslashes($_POST['date']);
$date=htmlspecialchars($date);
$regex = "/^[0-9]{2}\\/[0-9]{2}\\/[0-9]{4}$/";

if(preg_match($regex, $date)){
    $tabDate = explode("/", $date);
    $date =$tabDate[2]."-".$tabDate[1]."-".$tabDate[0];
}
}
```

Code JS

```
//date
var regex = /^[0-9]{2}\\/[0-9]{2}\\/[0-9]{4}$/;
$("#date").change(function(){
    console.log("yop "+Date.parse($('#date').val())+" fin yop");
    console.log($('#date').val());
    var yopDate=$('#date').val();
    if (yopDate==regex){
        var yopDate=String(yopDate);
        console.log(typeof yopDate);
        var tab=yopDate.split('/');
        yopDate=tab[2]+"-"+tab[1]+"-"+tab[0];
        console.log(yopDate);
    }
}
```

IV- La modification des étudiants

Pour modifier les étudiants, nous avons fait le choix d'ouvrir des fenêtres modal lors du clique du bouton d'édition d'un étudiant. Lors de ce clique nous récupérons les données de l'étudiant présent dans le tableau avec JQUERY. Nous les insérons dans le formulaire d'édition. Pour la liste des enseignants, nous créons cette liste dès le lancement de la page, puis nous réactualisons l'attribut **selected** lors du clique d'un bouton édition.

Pour modifier l'étudiant, nous utilisons l'objet AJAX. Cela permet de ne pas recharger la page à chaque édition.

Le code pour le transfert de données vers la fenêtre modale:


```

$(".btn-user").click(function(){
    console.log("yep");
    var nom = "#nom_" + $(this).attr('id');
    console.log($(nom).text());
    var prenom = "#prenom_" + $(this).attr('id');
    $("#nom-name").val($(nom).text());
    $("#prenom-name").val($(prenom).text());

    var classe = '.' + $(this).attr('id');
    var ens = "#tut_ens_" + $(classe).text();
    console.log(ens);
    $("#tuteur-name option:selected").prop("selected", false);
    $(ens).prop("selected", true);

});

```

IV- Autres

L'identifiant d'un rapport est créé comme cela: **idEtu-idEns-Type-numeroDeRapport**

Le **numeroDeRapport** est le nombre de rapport +1 que possède l'étudiant. Le type s'écrit E ou F en fonction du type de rendez vous (famille ou étudiant). Il est stocké en booléen sur la bdd (**true**: famille, **false**: étudiant).

```

//le type avec le matricule
$type=$_POST['type'];
if($type=="TRUE"){
    $mat=$id_etu."-".$id_ens."-F-".$nb;
}else{
    $mat=$id_etu."-".$id_ens."-E-".$nb;
}

```