DELANNOY Jeremy
DEROY Flavien

ANABASE [CONGRES NATIONAL DES MAISONS DES LIGUES]

Mission 5 : Notre mission était de réalisé la conception des tâches suivantes : T1.2, T2.2.

• **T1.2 Gestion des activités :** Cette tâche doit permettre la création, la modification, la suppression d'une activité à partir d'un formulaire. Il permet également d'obtenir la liste des activités sur un formulaire. Il faudra réaliser les maquettes de ces formulaires, les faire valider par le chef de projet avant de commencer le codage.

• **T2.2 Inscription à une activité :** Cette tâche doit permettre d'inscrire un congressiste à une activité, mais également de permettre l'annulation de son inscription. Pour chacun des cas la facture ne doit pas avoir été créée.

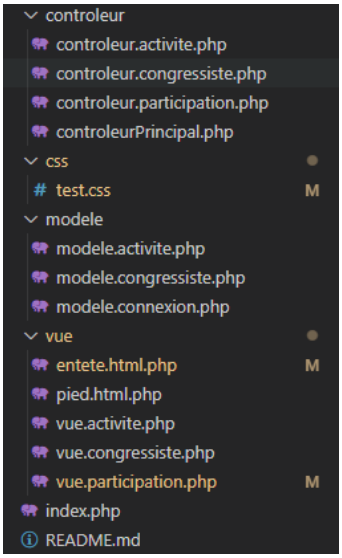
Nous avons effectué ce site web en **modèle MVC** (Modèle Vue Contrôleur) qui comme son nom l'indique, est composé de trois composants logiques principaux : le modèle, la vue et le contrôleur.

La vue est l'interface utilisateur. La vue permet à l'utilisateur d'afficher les données à l'aide d'un modèle et lui permet également de modifier les données.

Un modèle contient les données utilisées par un programme. Il peut s'agir d'une base de données, d'un fichier ou d'un simple objet.

Les contrôleurs agissent comme une interface entre le modèle et la vue, pour traiter toute la logique métier et les requêtes entrantes, manipuler les données à l'aide du composant Modèle et interagir avec les Vues pour rendre le résultat final.

Notre modèle MVC est donc répertorié comme ceci :



Et contiennent à chaque début de nom de fichier leur composant.
(Ex : vue.activite.php dans Vue)

• T1.2 Gestion des activités :

Comme dit précédemment, cette tâche permet la création, la modification, la suppression d'une activité à partir d'un formulaire, tout cela dans la table activité dans la base de donnée ANABASE.

Pour ce faire, nous avons créer des fonctions (en PHP) dans la partie modèle du MVC dans le fichier « *modele.activite.php* » qui regroupe donc toutes requêtes pour la page « Gestion des activités ».

Avant de commencer les fonctions, il faut connecter le fichier à la base de donnée que l'on veut :

```php
    public function __construct()
    {
        $login = "root";                                    /* Connexion au serveur local (ici : Wamp)
          $mdp = "";                                       avec le nom et le mot de passe du compte */

        $bd = "anabase";                                    // Choix de la base de donnée
        $serveur = "localhost";                             // Choix du serveur local utilisé

        try {
            $this->conn = new PDO(
                "mysql:host=$serveur;dbname=$bd",                /* Par la suite, on créé un objet
                $login,                                          PDO avec les données inscrites
                $mdp,                                            auparavant */
                array(PDO::MYSQL_ATTR_INIT_COMMAND => 'SET NAMES \'UTF8\'')
            );
            $this->conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
        } catch (PDOException $e) {
            print "Erreur de connexion PDO ";
            die();
        }
    }
```

Affichage d'une activité : fonction

```php
    public function getActivites()
    {
        $req = $this->conn->prepare("SELECT NUM_ACTIVITE, NOM_ACTIVITE, DATE_ACTIVITE, HEURE_ACTIVITE,
PRIX_ACTIVITE FROM activite ORDER BY NUM_ACTIVITE");
        $req->execute();

        return $req->fetchAll(PDO::FETCH_OBJ);
    }
```

`$this` est une variable qui permet de pointer sur une propriété de la classe.

Le $this pointe donc à la propriété qui permet d'accéder à la bdd voulue. Ainsi il suffit de préparer la requête `SELECT` qui permettra l'affichage dans un tableau de la table et ses colonnes voulues. Tout cela dans une variable que l'on a nommé `$req`.

`execute()` permet simplement d'exécuter la variable donc la requête.

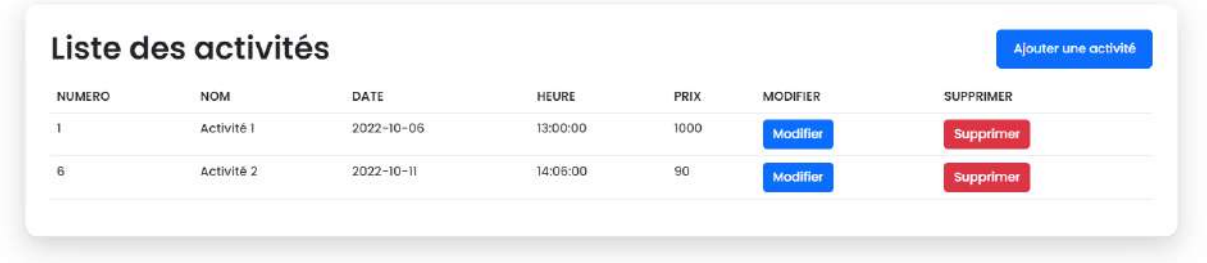
Le but est d'afficher la liste des activités sous forme de tableau, c'est pour cela que nous avons utilisé un `fetchAll()` qui permet de récupérer d'un coup l'ensemble du résultat d'une requête ligne par ligne.

Affichage d'une activité : L'appeler pour l'afficher dans un tableau

Le but est d'afficher les données sélectionnées (NUM_ACTIVITE, NOM_ACTIVITE, DATE_ACTIVITE, HEURE_ACTIVITE et PRIX_ACTIVITE) dans le un tableau sur le site. Le tableau est dans la partie vue (*vue.activite.php*) du MVC. La vue et le modèle sont relié grâce au contrôleur (*controleur.activite.php*) et j'expliquerai comment plus tard.

```
    <div class="liste" style="padding-top:10px;">
        <table class="table table-hover">
            <thead>
                <tr>
                    <th class="header">NUMERO</th>
                    <th class="header">NOM</th>
                    <th class="header">DATE</th>
                    <th class="header">HEURE</th>
                    <th class="header">PRIX</th>
                    <th class="header">MODIFIER</th>
                    <th class="header">SUPPRIMER</th>
                </tr>
            </thead>
            <tbody>
                <?php
                foreach ($c->data["get"] as $ligne) { ?>
                    <tr>
                        <td><?= $ligne->NUM_ACTIVITE; ?></td>                    // On fait donc un foreach qui obtiendra les
lignes, ligne par ligne, du
                        <td><?= $ligne->NOM_ACTIVITE; ?></td>                    // fetchAll et cette ligne est nommée par la
variable $ligne
                        <td><?= $ligne->DATE_ACTIVITE; ?></td>                   // Une ligne contiendra NUM_ACTIVITE, NOM_ACTIVITE,
                                                                                 DATE_ACTIVITE,
                        <td><?= $ligne->HEURE_ACTIVITE; ?></td>                  // HEURE_ACTIVITE et PRIX_ACTIVITE
                        <td><?= $ligne->PRIX_ACTIVITE ?></td>
                        <td><input type="button" class="btn btn-primary" onclick="window.location.href =
'index.php?controleur=activite&idModif=<?php echo $ligne->NUM_ACTIVITE; ?>'" id="btnModif" value= "Modifier" /></td>
                        <td><input type="button" class="btn btn-danger" onclick="window.location.href =
'index.php?controleur=activite&idSuppr=<?php echo $ligne->NUM_ACTIVITE; ?>'" value= "Supprimer" /></td>
```

Ce bout de code donne donc ce rendu :



Création d'une activité : fonction

Pour la création d'une activité, on utilise la requête INSERT INTO qui insère dans la table voulue (ici : activite). On attribue comme valeur des paramètres qui seront reliés grâce à des `bindValue()` qui sont reliés aux variables de la fonction.

```
    public function creerActivite($nom, $date, $heure, $prix)
    {
        $req = $this->conn->prepare("INSERT INTO activite VALUES(NULL, :param1 ,:param2, :param3
,:param4)");
        $req->bindValue('param1', $nom);
        $req->bindValue('param2', $date);
        $req->bindValue('param3', $heure);
        $req->bindValue('param4', $prix);
        $req->execute();
    }
```

Création d'une activité : l'appeler pour créer une activité par un formulaire

Pour créer une activité, nous utilisons la fonction **todo_Enregistrer** dans *controleur.activite.php* qui vérifie si les champs (nomA, dateA, …) sont remplis grâce à la méthode empty, et s'ils le sont, créer une activité grâce à la fonction `creerActivite()` où ses paramètres ($nom, …) sont reliés donc aux input (type text, type date et type heure) :

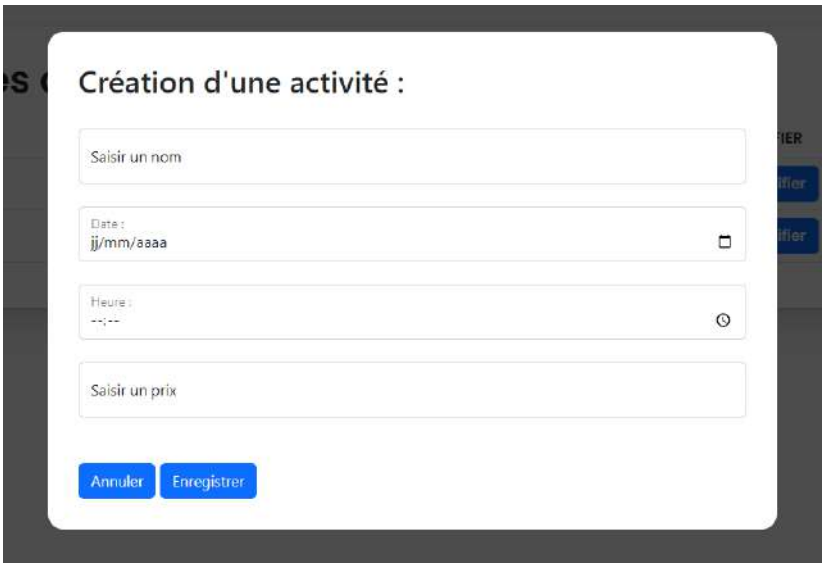
todo_Enregistrer est relier au bouton « Enregistrer » en HTML par son nom et sa valeur :

```
public function todo_Enregistrer()
    {
        if (empty($this->post["nomA"]) || empty($this->post["dateA"]) || empty($this->post["heureA"])
|| empty($this->post["prixA"])) {
            echo "impossible";
            $this->post["nomA"] = "";
            $this->post["dateA"] = "";
            $this->post["heureA"] = "";
            $this->post["prixA"] = "";
        } else {
            $this->modele->creerActivite($this->post["nomA"], $this->post["dateA"], $this-
>post["heureA"], $this->post["prixA"]);
            echo "insertion réussi";
            $this->post["nomA"] = "";
            $this->post["dateA"] = "";
            $this->post["heureA"] = "";
            $this->post["prixA"] = "";
            header("Location:index.php?controleur=activite");
        }
    }
    }
```

relier à … :

```
<input type="submit" class="btn btn-primary" name="todo" value="Enregistrer" />
```

Le rendu du formulaire pour créer une activité donne donc ceci :



Modification d'une activité : fonction

On récupère la ligne par son id (NUM_ACTIVITE) qui permet de modifier uniquement ce qu'on a choisi.

```
    public function modifActivites($numA, $nomA, $dateA, $heureA, $prixA)
    {
        $req = $this->conn->prepare("UPDATE activite SET NOM_ACTIVITE = :NomA, DATE_ACTIVITE = :DateA,
HEURE_ACTIVITE = :HeureA, PRIX_ACTIVITE = :PrixA  WHERE NUM_ACTIVITE = :NumA");
        $req->bindValue(':NumA', $numA);
        $req->bindValue(':NomA', $nomA);
        $req->bindValue(':DateA', $dateA);
        $req->bindValue(':HeureA', $heureA);
        $req->bindValue(':PrixA', $prixA);
        $req->execute();
    }
```

Modification d'une activité : l'appeler pour permettre la modification d'une ligne

Pour modifier une activité, même principe que Enregistrer, nous utilisons la fonction **todo_Modifier** dans *controleur.activite.php* qui vérifie si les champs (nomA, dateA, …) sont remplis grâce à la méthode empty, et s'ils le sont, modifier une activité grâce à la fonction `modifActivite()` où ses paramètres ($nom, …) sont reliés donc aux input :

```
    public function todo_Modifier(){
        if (empty($this->post["nomAM"]) || empty($this->post["dateAM"]) || empty($this-
>post["heureAM"]) || empty($this->post["prixAM"])) {
            echo "impossible";
            $this->post["nomAM"] = "";
            $this->post["dateAM"] = "";
            $this->post["heureAM"] = "";
            $this->post["prixAM"] = "";
        } else {
            $this->modele->modifActivites($this->get["idModif"],$this->post["nomAM"], $this-
>post["dateAM"], $this->post["heureAM"], $this->post["prixAM"]);
            header("Location:index.php?controleur=activite");
        }
    }
```

Mais aussi la fonction **todo_initialiser()**, qui appelle la fonction pour supprimer par identifiant.

```php
public function todo_initialiser()
    {
        $this->data["get"] = $this->modele->getActivites();
        if (isset($this->get["idSuppr"])) {
            $this->modele->supprActivites($this->get["idSuppr"]);
            header("Location:index.php?controleur=activite");
        }
        if(isset($this->get["idModif"])){
            $this->data["getby"] = $this->modele->getActivitesbyID($this->get["idModif"]);
        }
    }
```

Après avoir appuyer sur le bouton « Modifier » une nouvelle ligne apparaît donc avec des input qui obtiendront les nouvelles données si nécessaires :

Suppression d'une activité : fonction

On récupère la ligne par son id (NUM_ACTIVITE) qui permet de supprimer uniquement ce qu'on a choisi

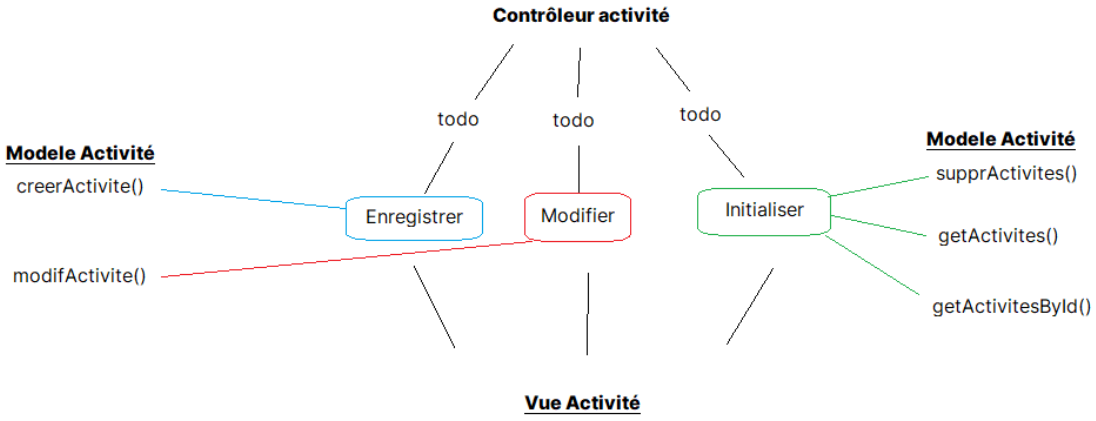
```php
    public function supprActivites($numA)
    {
        try {
            $req = $this->conn->prepare("DELETE FROM activite WHERE NUM_ACTIVITE=:NumA");
            $req->bindValue(':NumA', $numA);
            $req->execute();
        } catch (PDOException $e) {
                echo ' <div class="alert alert-primary" role="alert">

            Impossible de supprimer cette activité car des congressiste y participe! <a
href="./?action=activite" class="alert-link">Cliquer sur le lien pour revenir a la page</a>.
          </div>
            <br>
            </div>';
            die();

        }
    }
```

Suppression d'une activité : l'appeler pour permettre la suppression d'une ligne

Cette fois-ci, nous utilisons le todo_initialiser qui récupère l'id du bouton Supprimer par l'URL (idSuppr qui à la valeur de NUM_ACTIVITE) grâce au $this->get (qui s'apparente à un $_GET) dans la fonction supprActivites(). Cela permet, comme la modification, de supprimer par identifiant pour éviter de supprimer toutes les lignes.

```php
public function todo_initialiser()
    {
        $this->data["get"] = $this->modele->getActivites();
        if (isset($this->get["idSuppr"])) {
            $this->modele->supprActivites($this->get["idSuppr"]);
            header("Location:index.php?controleur=activite");
        }
        if(isset($this->get["idModif"])){
            $this->data["getby"] = $this->modele->getActivitesbyID($this->get["idModif"]);
        }
    }
```

Schéma pour mieux comprendre le todo :

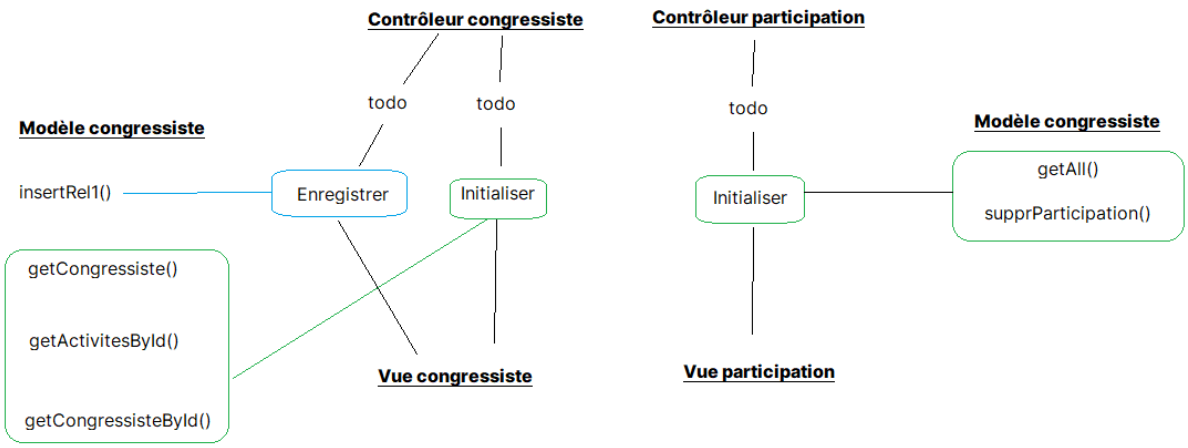


• T2.2 Inscription à une activité :

Comme dit précédemment, l'inscription permet d'inscrire un congressiste à une activité, mais également de permettre l'annulation de son inscription.

Pour ce faire, nous avons créer des fonctions (en PHP) dans la partie modèle du MVC dans le fichier « *modele.congressiste.php* » qui regroupe donc toutes requêtes pour la page « Gestion des activités ».

Comme auparavant, la structure est gérer grâce au todo_name. Voici la conception pour le congressiste :



Afficher les inscriptions (congressiste et activité liée) : fonction

On souhaite afficher simplement le numéro d'activité ainsi que le numéro du congressiste dans la même table (rel_1) pour savoir dans quelle(s) activité(s) seront plusieurs congressistes.

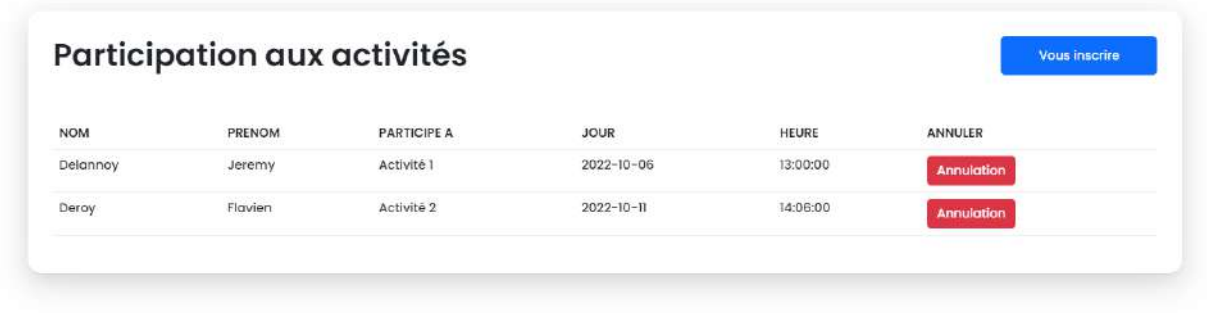
```
public function getAll(){
        $req = $this->conn->prepare("SELECT * FROM rel_1 r INNER JOIN congressiste c ON
c.NUM_CONGRESSISTE = r.NUM_CONGRESSISTE INNER JOIN activite a ON r.NUM_ACTIVITE = a.NUM_ACTIVITE");
        $req->execute();

        return $req->fetchAll(PDO::FETCH_OBJ);
    }
```

On récupère odnc tous les congressistes et activités auquel ils participent pour remplir mon tableau.

Affichage des inscriptions : L'appeler pour l'afficher dans un tableau

```
    <table class="table table-hover">

<th>NOM</th>
<th>PRENOM</th>
<th>PARTICIPE A</th>
<th>JOUR</th>
<th>HEURE</th>
<th>ANNULER</th>
    <?php foreach($c->data["All"] as $a){
?><tr>
    <td><?php echo $a->NOM_CONGRESSISTE;?></td>
    <td><?php echo $a->PRENOM_CONGRESSISTE;?></td>
    <td><?php echo $a->NOM_ACTIVITE; ?></td>
    <td><?php echo $a->DATE_ACTIVITE; ?></td>
    <td><?php echo $a->HEURE_ACTIVITE; ?></td>
    <td><a href="index.php?controleur=participation&idSupprC=<?php echo $a-
>NUM_CONGRESSISTE;?>&idSupprA=<?php echo $a->NUM_ACTIVITE;?>" class="btn btn-danger">Annulation</td>
    </tr>
<?php } ?>
```

Voici le résultat obtenu :



Mon bouton suppression me permet de récupérer l'id de congressiste et l'id de l'activité par l'URL me servant plus tard pour la suppression.

Suppression d'une inscription : fonction

L'annulation se base sur le DELETE qui supprime donc la participation contenant les 2 id récupérés précédemment.

```
 public function supprparticipation($ida, $idc){

        $req = $this->conn->prepare ("DELETE FROM rel_1 WHERE NUM_ACTIVITE=:ida AND
NUM_CONGRESSISTE=:idc");
        $req->bindValue(':ida', $ida);
        $req->bindValue(':idc', $idc);
        $req->execute();
```

Suppression d'une inscription : l'appeler pour permettre la suppression d'une ligne

Si nous obtenons les 2 id voulus, on effectue l'appelle à la fonction avec pour paramètre ces id et ensuite recharger la page pour actualiser le tableau.

idSupprA récupère le numéro d'activité, idSupprC récupère le numéro de congressiste.

```
public function todo_initialiser(){
        $this->data["All"] = $this->modele->getAll();
        if(isset($this->get["idSupprC"]) && isset($this->get["idSupprA"])){
            $this->modele->supprparticipation($this->get["idSupprA"], $this->get["idSupprC"]);
            header("Location:index.php?controleur=participation");
        }
    }
```

Afficher le menu d'inscriptions : fonction

```php
public function getCongressiste(){
        $req = $this->conn->prepare("SELECT * FROM congressiste");
        $req->execute();

        return $req->fetchAll(PDO::FETCH_OBJ);
    }
```

Nous sélectionnons tous les congressistes pour obtenir le tableau ci-dessous.

Html du tableau :

```php
<form method="post" action="">
<table class="table table-striped table-hover">
<th>NOM</th>
<th>PRENOM</th>
<th>ADRESSE</th>
<th>SELECTION</th>
    <?php foreach($c->data["idC"] as $unCongressiste){
?>
<tr>
    <td><?php echo $unCongressiste->NOM_CONGRESSISTE;?></td>
    <td><?php echo $unCongressiste->PRENOM_CONGRESSISTE;?></td>
    <td><?php echo $unCongressiste->ADRESSE_CONGRESSISTE?></td>
    <td><input type="button" class="btn btn-primary" onclick="window.location.href =
'index.php?controleur=congressiste&idChoix=<?php echo $unCongressiste->NUM_CONGRESSISTE; ?>'"
value="Choisir" /></td>

    </tr>
<?php } ?>
</table>
<br>
</form>
```

Afficher le menu d'inscriptions : Choix d'activité

Nous débutons l'explication de la conception la plus « compliqué » du projet qui est le choix de une ou plusieurs activités pour un congressiste.

Le bouton « Choisir » récupère l'id du congressiste choisi, dans l'url, et recharge la page.

Comme on peut le voir dans la fonction todo_initialiser(), une fois l'id du choix obtenu, la fonction utilise 2 autres fonctions (getActiviteByID qui récupère le numéro de la ou les activité(s) prochainement sélectionnée(s) ainsi que son numéro de congressiste) pour charger le prochain tableau (choix d'activité) :

```
public function todo_initialiser(){
        $this->data["idC"] = $this->modele->getCongressiste();
        if(isset($this->get["idChoix"])){
            $this->data["idA"] = $this->modele->getActivitesbyID($this->get["idChoix"]);
            $this->data["nom"] = $this->modele->getCongressisteID($this->get["idChoix"]);
            ;
        }
    }
```

La fonction getActivitebyID() va récupérer le numéro de l'activité grâce à son paramètre :

```
public function getActivitesbyID($num){

        $req = $this->conn->prepare ("SELECT * FROM activite WHERE NUM_ACTIVITE NOT IN (SELECT
NUM_ACTIVITE FROM rel_1 WHERE NUM_CONGRESSISTE = :num)");
        $req->bindValue('num', $num);
        $req->execute();

        return $req->fetchAll(PDO::FETCH_OBJ);
    }
```

Même principe pour getCongressisteID() :

```
public function getCongressisteID($num){
        $req = $this->conn->prepare ("SELECT * FROM congressiste WHERE NUM_CONGRESSISTE = :num");
        $req->bindValue('num', $num);
        $req->execute();

        return $req->fetch(PDO::FETCH_OBJ);
    }
```

Nous récupérons ensuite les activités dans lesquelles le congressiste ne participe pas ainsi que les informations (nom, prénom) sur le congressiste sélectionné précédemment pour afficher ce tableau :

Delannoy Jeremy serra présent aux activité :

NOM	DATE	HEURE	PRIX	SELECTION
basket	2022-12-10	17:00:00	15€	☐
Match basket	2023-02-17	21:00:00	10€	☐
match foot	2022-11-30	17:10:00	16€	☐

Enregistrer

Html du tableau :

```
<?php
if(isset($_GET["idChoix"])){
    $nom = $c->data["nom"];
    ?>
    <h3><?php echo $nom->NOM_CONGRESSISTE," ", $nom->PRENOM_CONGRESSISTE;?> serra présent aux activité
:</h3>
    <form method="post" action="">
    </table>



        <table class="table table-hover">
            <th>NOM</th>
            <th>DATE</th>
            <th>HEURE</th>
            <th>PRIX</th>
            <th>SELECTION</th>
            <?php foreach ($c->data["idA"] as $uneActivite) {
            ?>
                <tr>
                    <td><?php echo $uneActivite->NOM_ACTIVITE; ?></td>
                    <td><?php echo $uneActivite->DATE_ACTIVITE; ?></td>
                    <td><?php echo $uneActivite->HEURE_ACTIVITE ;?></td>
                    <td><?php echo $uneActivite->PRIX_ACTIVITE ;?>€</td>
                    <td><input type="checkbox" value="<?php echo $uneActivite->NUM_ACTIVITE; ?>"
name="chkl[ ]"></td>
                </tr>
            <?php } ?>
            <br>
        </table>
        <?php
}
?>

<input type="submit" class="btn btn-primary" name="todo" value = "Enregistrer">
    </form>
```

L'input Checkbox a pour name (chkl[ ]) un tableau car ceci permettra plus tard d'insérer tous les choix de la sélection de l'utilisateur (si l'utilisateur choisi plus d'une option).

Inscription d'un congressiste: fonction

On récupère tous les choix effectués par l'utilisateur précédemment pour inscrire le congressiste a son activité (le congressiste du 1er tableau et les choix d'activité du 2ème tableau).

```php
public function insertRel1( $numC, $numA){
        $req = $this->conn->prepare ("INSERT INTO rel_1 VALUES(:param1, :param2)");
        $req->bindValue('param1', $numC);
        $req->bindValue('param2', $numA);
        $req->execute();
    }
```

Inscription d'un congressiste : l'appeler pour l'inscription d'un congressiste par un formulaire

```php
    public function todo_Enregistrer(){
        {
            if (!isset($this->get["idChoix"]) || empty($this->post["chkl"])) {
                echo "impossible";
                $this->post["selectC"] = "";
                header("Location:index.php?controleur=participation");
            } else {
                foreach($this->post["chkl"] as $chkl){
                $this->modele->insertRel1($this->get["idChoix"], $chkl);
                echo "insertion réussi";
                header("Location:index.php?controleur=participation");
            }
        }
        }
    }
```

La fonction vérifie si l'utilisateur a bien choisi au moins une activité et un congressiste. Si oui, cela fait une boucle pour ajouter chacun des choix/sélections. Un foreach est utilisé pour parcourir l'input type Checkbox grâce à son name qui est donc un tableau (rappel : chkl[ ]). Pour chaque choix, nous passons en paramètre le congressiste et le(s) activité(s) jusqu'à avoir parcouru toutes les activités choisies.