Documentation technique

Contexte:

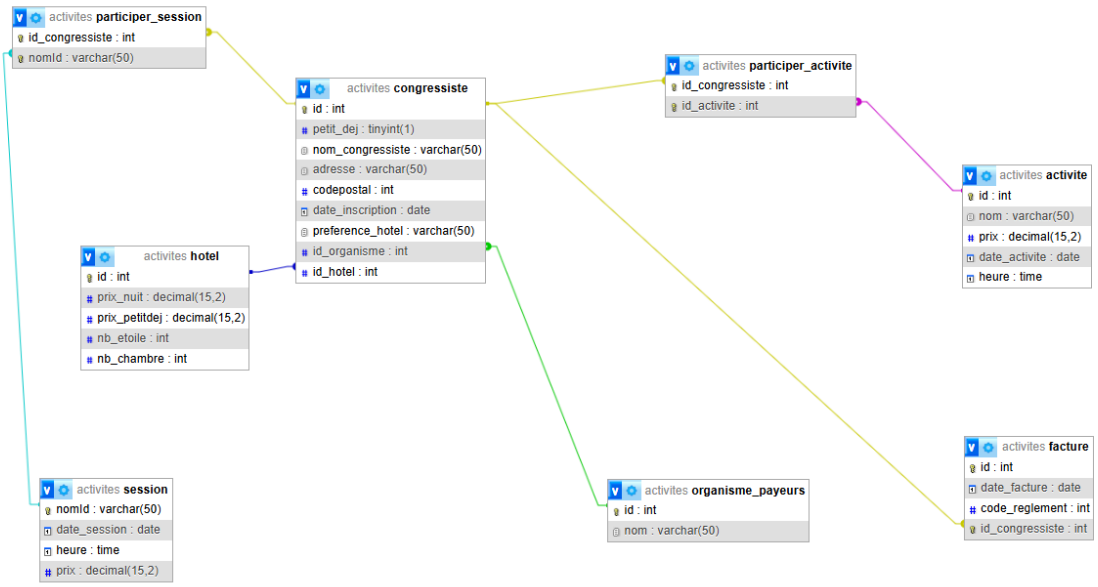
La Maison des ligues de Lorraine souhaite confier aux étudiants de BTS SIO du Lycée Suzanne Valadon la réalisation d'un projet consistant à développer l'application permettant la gestion du congrès (inscription des participants, mise en place de sessions, organisations des activités, facturation, etc.)

Les correspondants de la ligue ont déjà détaché certains domaines importants et ont effectué le découpage de chacun en tâches :

MISSION 5: Activités:

`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.
`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.

BDD:



Ce schéma relationnel représente trois tables reliées par des clés étrangères. La table **"congressiste"** contient les informations des participants avec la clé primaire `id`. La table **"activite"** stocke les détails des activités avec `id` comme clé primaire. La table **"participer_activite"** est une table d'association permettant une relation entre les congressistes et les activités, grâce aux clés étrangères `id_congressiste` (référence à

"congressiste") et `id_activite` (référence à "activite"). Ainsi, un congressiste peut participer à plusieurs activités, et une activité peut accueillir plusieurs congressistes.

Connexion è la BDD:

```php
<?php
    $login = "root";
    $mdp = "";
    $bd = "activites";
    $serveur = "localhost";

    try {
        $pdo = new PDO("mysql:host=$serveur;dbname=$bd", $login, $mdp,
array(PDO::MYSQL_ATTR_INIT_COMMAND => 'SET NAMES \'UTF8\''));
        $pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
        return $pdo;
    } catch (PDOException $e) {
       echo "Erreur de connexion à la base de données : " . $e->getMessage();
        die();
    }
?>
```

Ce code PHP établit une connexion à une base de données MySQL (activites).

Controleur principale:

```php
<?php
function controleurPrincipal($action){
    $lesActions = array();
    $lesActions["activites"] = "controleurActivites.php";
    $lesActions["congressiste"] = "controleurCongressiste.php";

    if (array_key_exists ( $action , $lesActions )){
        return $lesActions[$action];
    }
    else{
        return $lesActions["activites"];
    }
}
?>
```

Ce code PHP définit une fonction `controleurPrincipal($action)` qui redirige les actions vers les fichiers de contrôleurs appropriés. Si l'action existe comme clé dans le tableau (`array_key_exists`), elle retourne le fichier correspondant, sinon elle retourne le fichier par

défaut "controleurActivites.php". Cela centralise la gestion des contrôleurs dans une architecture MVC

Gestion des Activité :

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.

Lorsque que l'on arrive sur la page on peut consulter les différentes activités proposées.

La barre de navigation permet de naviguer entre l'inscription des congressistes via le lien **"Congressiste"**, et la gestion des activités (ajout, modification et suppression) grâce au lien **"Activités"**.

MODEL

Pour faire cela nous avons besoin d'un model activité dans un premier temp :

```
class Activites {
        private int $Id;
```

```
        private string $Nom;
        private float $Prix;
        private string $Heure;
        private string $Date;

        public function __construct() {

        }
```

On peut y observer que on a créé une class Activités avec un id, nom, prix, heure et date tout cela et dans base de données et nous avons mis un constructeur vide.

Par la suite il y'a les différent get et set.

```
public function getLesActivites() {
        include "bd.php";
        $req = "SELECT * FROM activite";
        $stmt = $pdo->prepare($req);
        $stmt->execute();
        $lesActivites = $stmt->fetchAll(PDO::FETCH_OBJ);
        return $lesActivites;
    }

    public function getActiviteById() {
        include "bd.php";
        $req = "SELECT * FROM activite WHERE id = ?";
        $stmt = $pdo->prepare($req);
        $stmt->bindValue(1, $this->Id);
        $stmt->execute();
        $UneActivite = $stmt->fetch(PDO::FETCH_OBJ);
        return $UneActivite;
    }

    public function ajouterActivite() {
        include "bd.php";
        $req = "INSERT INTO activite VALUES (null, ?, ?, ?, ?)";
        $stmt = $pdo->prepare($req);
        $stmt->bindValue(1, $this->Nom);
        $stmt->bindValue(2, $this->Prix);
        $stmt->bindValue(3, $this->Date);
        $stmt->bindValue(4, $this->Heure);
        return $stmt->execute();
    }
```

```
    public function supprimerActivite() {
        include "bd.php";
        $req = "DELETE FROM activite WHERE id = ?";
        $stmt = $pdo->prepare($req);
        $stmt->bindValue(1, $this->Id);
        return $stmt->execute();
    }

    public function modifierActivite() {
        include "bd.php";
        $req = "UPDATE activite SET nom = ?, prix = ?, date_activite = ?, heure = ? WHERE id = ?";
        $stmt = $pdo->prepare($req);
        $stmt->bindValue(1, $this->Nom);
        $stmt->bindValue(2, $this->Prix);
        $stmt->bindValue(3, $this->Date);
        $stmt->bindValue(4, $this->Heure);
        $stmt->bindValue(5, $this->Id);
        return $stmt->execute();
    }
```

Ceci est les différentes functions de activités elle permette la consultation des activités via getLesActivites, la consultation d'une activité précise via getActivteById, l'ajout d'activite via ajouterActivite, la suppression via supprimerActivite et la modification des activités via modifierActivite.

CONTROLEUR

Le controleurActivites permet de s'occupper de recevoir les données entrées par l'utilisateur et de communiquer les changements aux modèles. Il effectue des opérations CRUD (Créer, Lire, Mettre à jour, Supprimer)

Le controleur est composé de :

```
include "model/activites.php";

$UneActivite = new Activites();
$LesActivites = $UneActivite->getLesActivites();

if (isset($_POST["ajouter"])) {
    $nouvelleActivite = new Activites();
    $nouvelleActivite->setNom($_POST["nom"]);
    $nouvelleActivite->setPrix($_POST["prix"]);
    $nouvelleActivite->setDate($_POST["date"]);
```

```
  $nouvelleActivite->setHeure($_POST["heure"]);

  if ($nouvelleActivite->ajouterActivite()) {
    header("location: index.php?action=");
  } else {
    echo "<p>Erreur lors de l'ajout de l'activité.</p>";
  }
}

if(isset($_GET["supp"])) {
  $nouvelleActivite = new Activites();
  $nouvelleActivite->setId($_GET["supp"]);
  if($nouvelleActivite->supprimerActivite()) {
    header("location: index.php");
  } else {
    echo "<p>Erreur lors de la suppression de l'activité.</p>";
  }
}

if (isset($_POST["modifier"])) {

    $modifierActivite = new Activites();
    $modifierActivite->setId($_GET["modif"]);
    $modifierActivite->setNom($_POST["nom"]);
    $modifierActivite->setPrix($_POST["prix"]);
    $modifierActivite->setDate($_POST["date"]);
    $modifierActivite->setHeure($_POST["heure"]);

    if ($modifierActivite->modifierActivite()) {
      header("location: index.php?action=activites");
    } else {
      echo "<p>Erreur lors de la modification de l'activité.</p>";
    }
}
```

Récupération des activités existantes
Le script initialise une instance de la classe Activites pour charger toutes les activités via la méthode getLesActivites(). Ces données sont préparées pour être utilisées dans la vue.

Ajout d'une activité
Lorsqu'un formulaire avec la clé POST ajouter est soumis, une nouvelle instance de Activites est créée. Les données du formulaire (nom, prix, date, heure) sont définies à l'aide des setters avant

d’appeler la méthode ajouterActivite(). En cas de succès, une redirection est effectuée ; sinon, un message d’erreur est affiché.

Suppression d’une activité
Si le paramètre GET supp est présent, une activité est supprimée grâce à son ID, transmis via la méthode setId(), avant d’appeler supprimerActivite(). Le script redirige ou affiche un message d’erreur selon le résultat.

Modification d’une activité
Lorsqu’un formulaire avec la clé POST modifier est soumis, une activité est mise à jour en récupérant son ID via le paramètre GET modif. Les nouveaux attributs (nom, prix, date, heure) sont définis et la méthode modifierActivite() est appelée. Le script redirige ou affiche un message d’erreur en fonction de l’opération.

Affichage des activités
La vue vueActivites.php est incluse pour afficher les données des activités récupérées, en exploitant le modèle pour la mise en page.

La VUE

Cette vue est dédiée à l’affichage, la modification et l’ajout des activités dans l'application. Elle propose plusieurs sections fonctionnelles pour interagir avec les données.

```
<h1>Bienvenue à l'inscription aux activités</h1>
    <h1>Liste des Activités</h1>
    <table>
      <tr>
        <th>ID</th>
        <th>Nom</th>
        <th>Prix</th>
        <th>Date</th>
        <th>Heure</th>
        <th></th>
        <th></th>
      </tr>
      <?php foreach ($LesActivites as $activite) { ?>
        <tr>
          <td style='padding: 15px;'><?php echo $activite->id; ?></td>
          <td style='padding: 15px;'><?php echo $activite->nom; ?></td>
          <td style='padding: 15px;'><?php echo $activite->prix; ?> €</td>
          <td style='padding: 15px;'><?php echo $activite->date_activite; ?></td>
          <td style='padding: 15px;'><?php echo $activite->heure; ?></td>
          <td style='padding: 15px;'><a href='index.php?action=activites&supp=<?php echo
$activite->id ?>'>SUPPRIMER</a></td>
```

```
            <td style='padding: 15px;'><a href='index.php?action=activites&modif=<?php echo
$activite->id ?>'>MODIFIER</a></td>
          </tr>
        <?php } ?>
    </table>
    <div class="btn">
      <a href='index.php?action=activites&ajouter='>AJOUTER UNE ACTIVITE</a>
    </div>
    <?php
          if(isset($_GET["modif"])) {

            $nouvelleActivite = new Activites();
            $nouvelleActivite->setId((int) $_GET["modif"]);

            $activite = $nouvelleActivite->getActiviteById();

            ?>
            <h1>Modifier Activite</h1>
            <form action="./?action=activites&modif=<?php echo $_GET['modif']; ?>"
method="POST">
              <input type="text" name="nom" placeholder="Nom de l'Activité" value="<?php
echo $activite->nom; ?>" required/><br />
              <input type="number" name="prix" placeholder="Prix de l'activité" value="<?php
echo $activite->prix; ?>" required/><br />
              <input type="time" name="heure" value="<?php echo $activite->heure; ?>"
required/><br />
              <input type="date" name="date" value="<?php echo $activite->date_activite; ?>"
required/><br />
              <input type="submit" name="modifier" value="Modifier"/>
            </form>
            <?php
          } if(isset($_GET["ajouter"])) {
            ?>
            <h1>Ajouter Activites</h1>
            <form action="./?action=activites" method="POST">
              <input type="text" name="nom" placeholder="Nom de l'Activité" required/><br />
              <input type="number" name="prix" placeholder="Prix de l'activité" required/><br
/>
              <input type="time" name="heure" placeholder="Date de l'Activité" required/><br />
              <input type="date" name="date" placeholder="Heure de l'Activité" required/><br />
              <input type="submit" name="ajouter" value="Ajouter"/>
            </form>
            <?php
          }
```

```
        ?>


  </div>
</div>
```

Cette vue combine du HTML et du PHP pour afficher et gérer dynamiquement les activités. Elle commence par un titre statique et un tableau généré de manière dynamique grâce à une boucle foreach en PHP, qui parcourt la variable $LesActivites. Chaque activité, représentée par un objet, est affichée dans une ligne de tableau avec ses propriétés (id, nom, prix, date_activite, heure). Deux colonnes supplémentaires contiennent des liens générés dynamiquement pour supprimer ou modifier une activité en passant son identifiant en paramètre GET.

Le PHP est également utilisé pour gérer l'affichage conditionnel des formulaires. Si le paramètre GET modif est présent, un formulaire prérempli apparaît, récupérant les données de l'activité via la méthode getActiviteById() de la classe Activites. Les champs du formulaire sont remplis à l'aide des propriétés de l'objet retourné (nom, prix, date_activite, heure).

De manière similaire, si le paramètre GET ajouter est détecté, un formulaire vierge s'affiche pour permettre la création d'une nouvelle activité. Les données saisies dans ce formulaire sont envoyées au contrôleur via une requête POST. Ainsi, le PHP joue un rôle clé pour lier les données du modèle, la logique métier et les éléments dynamiques de l'interface utilisateur.