

BTS SERVICES INFORMATIQUES AUX ORGANISATIONS Épreuve E5 - Conception et développement d'applications (option SLAM) ANNEXE 7-1-B : Fiche descriptive de réalisation professionnelle (recto)	SESSION 2023
---	---------------------

DESCRIPTION D'UNE RÉALISATION PROFESSIONNELLE		N° réalisation : 01
Nom, prénom : BOUTEVIN SANCÉ Lucas		N° candidat : 01948505199
Épreuve ponctuelle <input checked="" type="checkbox"/>	Contrôle en cours de formation <input type="checkbox"/>	Date : / /
Organisation support de la réalisation professionnelle La Maison des Ligues de Lorraine (M2L)		
Intitulé de la réalisation professionnelle Projet de modification du site de la M2L		
Période de réalisation : 02/09/2022 au 18/11/22 Lieu : Lycée Gustave Eiffel, Bordeaux Modalité : <input type="checkbox"/> Seul(e) <input checked="" type="checkbox"/> En équipe		
Compétences travaillées <input checked="" type="checkbox"/> Concevoir et développer une solution applicative <input type="checkbox"/> Assurer la maintenance corrective ou évolutive d'une solution applicative <input checked="" type="checkbox"/> Gérer les données		
Conditions de réalisation¹ (ressources fournies, résultats attendus) <u>Voici les ressources fournis au commencement du projet :</u> <ul style="list-style-type: none"> - Le sujet - L'application existante - Le MEA à compléter - Le script de création et de remplissage des tables "commune" et "département". <u>Objectif du projet :</u> La Maison des Ligues de Lorraine dispose d'un site statique destiné à informer les internautes, mais elle souhaite le compléter avec différentes fonctionnalités. La M2L souhaite faire évoluer son site afin de permettre : <ul style="list-style-type: none"> - La gestion dynamique des informations concernant les ligues (infos pratiques sur les ligues et les clubs affiliés) - La gestion et la consultation des informations relatives aux intervenants (bulletins de salaire, contrat, info diverses ...) - La gestion des formations proposées aux membres des ligues affiliées à la M2L. Ce travail ayant été fait en groupe, je me suis occupé de la partie : Gestion des formations proposées aux membres des ligues affiliées à la M2L.		

¹ En référence aux *conditions de réalisation et ressources nécessaires* du bloc « Conception et développement d'applications » prévues dans le référentiel de certification du BTS SIO.

Voici la description des fonctionnalités attendus dans ma partie :

- Les intervenants, après authentification, pourront consulter la liste des formations proposées.
- Les intervenants peuvent déposer une demande d'inscription pour les formations de leur choix. Ils pourront aussi supprimer une demande.
- Les intervenants doivent pouvoir à tout moment aussi consulter l'état de leur(s) demande(s) d'inscription aux formations (en attente, refusée, acceptée).
- Un intervenant n'a accès qu'à ses propres demandes d'inscription.
- Le responsable formation pouvoir, ajouter, modifier, consulter et supprimer les formations.
- Le responsable formation pourra aussi consulter la liste des demandes d'inscription, triée pour chaque formation ou pour chaque intervenant.
- Le responsable formation pourra modifier l'état des demandes d'inscription (accepter ou refuser). Il devra alors faire en sorte que le nombre d'inscriptions acceptées pour une formation ne dépasse pas l'effectif maximum négocié pour cette formation.

Description des ressources documentaires, matérielles et logicielles utilisées²

Ce projet a été réalisé avec les outils suivant :

- Un ordinateur, un écran, un clavier, une souris
- Un navigateur web
- Une connexion internet
- Visual Studio Code
- Wampserver64
- GitLab

Modalités d'accès aux productions³ et à leur documentation⁴

Ouvrir l'explorateur de fichier, faire clique droite puis "Git Bash Here" et exécuter les commandes suivantes :

1. `git clone https://gitlab.com/lucas66166/m2l_groupe2.git`
2. Aller dans le dossier créer m2l_groupe2 puis faire : `git checkout formation`

Ensuite lancer Wampserveur et faite :

1. Cliquez gauche, Vos VirtualHosts, Gestion VirtualHost
2. Rentrer un nom pour votre VirtualHost puis le chemin d'accès en dossier m2l_groupe2 puis faite "Démarrer la création du VirtualHost (Peut prendre un certain temps)"
3. Retourner sur le logo Wamp vert et faite, Cliquez droit, Outils, Redémarrage DNS
4. Ensuite, dans PhpMyAdmin créer une base de données sous le nom "apphpformation" ; cliquer sur apphpformation, puis Importer et glisser le script qui se situe à m2l_groupe2/bdd/ScriptBDD (A choisir pour l'épreuve).sql (Dans le code modeles/DAO/param.php mettre vos identifiant et mdp PhpMyAdmin)

Lancer le projet en allant sur le logo de wamp vert: Cliquez gauche, Vos VirtualHosts, "nom de votre virtualhost".

Pour se connecter en tant que **responsable de formation** rentrez en identifiant "Eren" et en mdp "123". Pour se connecter en tant qu'**internaute** rentrez en identifiant "bob" et en mdp "123".

² Les réalisations professionnelles sont élaborées dans un environnement technologique conforme à l'annexe II.E du référentiel du BTS SIO.

³ Conformément au référentiel du BTS SIO « Dans tous les cas, les candidats doivent se munir des outils et ressources techniques nécessaires au déroulement de l'épreuve. Ils sont seuls responsables de la disponibilité et de la mise en œuvre de ces outils et ressources. La circulaire nationale d'organisation précise les conditions matérielles de déroulement des interrogations et les pénalités à appliquer aux candidats qui ne se seraient pas munis des éléments nécessaires au déroulement de l'épreuve. ». Les éléments peuvent être un identifiant, un mot de passe, une adresse réticulaire (URL) d'un espace de stockage et de la présentation de l'organisation du stockage.

⁴ Lien vers la documentation complète, précisant et décrivant, si cela n'a été fait au verso de la fiche, la réalisation professionnelle, par exemples service fourni par la réalisation, interfaces utilisateurs, description des classes ou de la base de données.

Épreuve E5 - Conception et développement d'applications (option SLAM)

ANNEXE 7-1-B : Fiche descriptive de réalisation professionnelle
(verso, éventuellement pages suivantes)

Descriptif de la réalisation professionnelle, y compris les productions réalisées et schémas explicatifs

Mes DAO, DTO (class singulière et pluriel) **FONCTIONNE TOUTE DE LA MÊME MANIÈRE** et voici leur fonctionnement :

Prenons pour exemple les formations. Il y a une class nommée FormationDAO qui dispose des fonctions allFormation, AjoutFormation, ModifFormation et SupprFormation.

Les méthodes nommées "all" + nom de la classe dans les DAO, servent à retourner un tableau contenant les objets de cette classe, en allant chercher en BDD toutes les valeurs :

(Ce code est dans : modeles/DAO/FormationDAO.php)

```
// Retourne un tableau contenant les objets Formation
1 reference | 0 overrides
public static function allFormation(){
    // Requete pour aller chercher toute les formations
    $requetePrepa = DBConnex::getInstance()->prepare("select * from formation;");
    // Execute la requete
    $requetePrepa->execute();
    // Récupère la reponse
    $list = $requetePrepa->fetchAll(PDO::FETCH_ASSOC);

    // Si la reponse n'est pas vide
    if(!empty($list)){
        // Pour toute les formations
        foreach($list as $formation){
            // Créer un objet formations
            $uneFormation = new Formation(null,null,null,null,null,null,null);
            // Remplie l'objet formation
            $uneFormation->hydrate($formation);
            // Ajoute l'objet à la listes
            $lesFormations[] = $uneFormation;
        }

        return $lesFormations;
    }
    return null;
}
```

Ces méthodes "all" sont utilisées dans mes class pluriels pour aller chercher en BDD toutes les valeurs d'une table, les mettre sous forme objet et les stockées dans la classe plurielle.

Chaque class pluriel contient donc deux méthodes que voici :

(Ce code est dans : modeles/DTO/pluriel/Formations.php)

```
/**
 * Méthode Méthode qui permet d'aller chercher les
 * formations de la BDD et les mettre dans l'attribut lesFormations
 */
4 references | 0 overrides
public static function initializeFormations(){
    // Va chercher tout les formations
    $lesFormations = FormationDAO::allFormation();
    // Stock les formations
    Formations::setLesFormations($lesFormations);
}
```

```
5 references | 0 overrides
public static function getLesFormations(){
    if(self::$lesFormations == null){
        Formations::initializeFormations();
    }
    return self::$lesFormations;
}
```

Par la suite, si on veut toutes les formations, on aura juste à le demander à la class pluriel avec getLesFormation et le code verra si oui ou non il doit faire une requête en BDD.

Je rappelle que cette structure est valable partout dans mon code. Donc si on veut aller chercher tous les utilisateurs on peut utiliser la class pluriel Utilisateurs pour faire getLesUtilisateurs et cela nous renverra tous les objets Utilisateur.

En plus de ces méthodes "all", les DAO ont aussi les méthodes Ajout, Modif et Suppr + "nom de la classe" qui servent à ajouter, modifier et supprimer des valeurs en BDD.

```
// Ajoute une formation
1 reference | 0 overrides
public static function AjoutFormation($intitule, $descriptif, $duree, $dateOuvertInscrip
    $requetePrepa = DBConnex::getInstance()->prepare("INSERT INTO formation (intitule, d

    $requetePrepa->bindParam(":intitule", $intitule);
    $requetePrepa->bindParam(":descriptif", $descriptif);
    $requetePrepa->bindParam(":duree", $duree);
    $requetePrepa->bindParam(":dateOuvertInscriptions", $dateOuvertInscriptions);
    $requetePrepa->bindParam(":dateclotureInscriptions", $dateclotureInscriptions);
    $requetePrepa->bindParam(":effectifmaximum", $effectifmaximum);

    $requetePrepa->execute();
}
```

Une fois qu'on ajoute, modifier ou supprimer des valeurs en BDD, il ne faut pas oublier de modifier les valeurs dans la class pluriel associé comme ceci :

```
FormationDAO::AjoutFormation($_POST["intitule"], $_POST["desc
// Change les élément dans la DTO
Formations::initializeFormations();
```

Il suffit d'appeler la méthode "initialize" qui va réinitialiser la class pluriel en réappelant la méthode "all".

Maintenant que vous savez comment fonctionne mes DAO/DTO, la suite vous montrera quasiment exclusivement mes contrôleurs.

Voici la liste des fonctionnalités de ma partie et leur fonctionnement détaillé :

- Les intervenants, après authentification, pourront consulter la liste des formations proposées :

Voici la page de connexion :

The image shows a web application interface for 'Maison des Ligues'. At the top, there's a header with the site name and a navigation bar containing 'Accueil' and 'Se connecter'. Below the header, there's a login form with the title 'Veuillez vous identifier'. The form has two input fields: 'Identifiant' with a placeholder 'Entrez votre identifiant' and 'Mot de Passe' with a placeholder 'Entrez votre mot de passe'. A 'Valider' button is positioned below the input fields.

Voici son code visuel (contrôleurs/controleurConnexion.php):

```
// Si l'utilisateur n'est pas connecté
if (!isset($_SESSION['utilisateur'])){
    // Création d'un formulaire
    $formulaireConnexion = new Formulaire('post', 'index.php', 'fConnexion', 'fConnexion');

    $formulaireConnexion->ajouterComposantLigne($formulaireConnexion->creerLabel('Identifiant :'));
    $formulaireConnexion->ajouterComposantLigne($formulaireConnexion->creerInputTexte('login', 'login', '', 1, 'Entrez votre identifiant', ''));
    $formulaireConnexion->ajouterComposantTab();

    $formulaireConnexion->ajouterComposantLigne($formulaireConnexion->creerLabel('Mot de Passe :'));
    $formulaireConnexion->ajouterComposantLigne($formulaireConnexion->creerInputMdp('mdp', 'mdp', 1, 'Entrez votre mot de passe', ''));
    $formulaireConnexion->ajouterComposantTab();

    $formulaireConnexion->ajouterComposantLigne($formulaireConnexion->creerInputSubmit('submitConnex', 'submitConnex', 'Valider'));
    $formulaireConnexion->ajouterComposantTab();

    $formulaireConnexion->ajouterComposantLigne($formulaireConnexion->creerMessage($message));
    $formulaireConnexion->ajouterComposantTab();

    $formulaireConnexion->creerFormulaire();

    // Affiche la vue connexion (Elle inclut la variable $formulaireConnexion donc on la requiert après)
    require_once 'vue/vueConnexion.php';
}
```

Voici le code qui s'exécute lorsqu'on clique sur le bouton "Valider" (contrôleurs /controleurConnexion):

```
// Si on clique sur "valider"
if(isset($_POST['submitConnex'])){

    // Enregistre le login et le mot de passe
    $login = $_POST['login'];
    $mdp = $_POST['mdp'];

    // Va chercher dans la BDD le login et mot de passe
    $unUtilisateur = UtilisateurDAO::verification($login, $mdp);

    // Si le login et mot de passe existe
    if($unUtilisateur != false){

        // Enregistre le login
        $_SESSION['utilisateur'] = $unUtilisateur;

        // Change le contrôleur choisi
        $_SESSION['page'] = "accueil";
        // Rafraîchit la page (On est tout le temps dans index.php, seulement les contrôleurs changent)
        header('location: index.php');

    }
    else{
        // message d'erreur
        $message = "Le nom d'utilisateur ou le mot de passe est incorrect";
    }
}
```

`$_SESSION["page"]` sert à changer de contrôleur mais on reste toujours sur la page index.php donc il faut rafraîchir la page pour pouvoir effectuer le changement du contrôleur.

Voici la méthode vérification de UtilisateurDAO (modeles/DAO/UtilisateurDAO.php):

```

// Prend une table en paramètre et retourne toute les occurences de celle-ci
1 reference|0 overrides
static function verification($login, $mdp) {
    // Requête pour aller chercher l'utilisateur grâce au login et mdp
    $sql = "SELECT idUser, nom, prenom, login, statut, typeUser, idFonct, idLigue, idClub FROM utilisateur where login=:login and mdp=:mdp;";
    // Prépare le requête
    $requetePrepa = DBConnex::getInstance()->prepare($sql);
    // Prépare les arguments de la requête
    $requetePrepa->bindParam(":login", $login);
    $requetePrepa->bindParam(":mdp", $mdp);
    // Execute la requête
    $requetePrepa->execute();

    // Récupère le resultat
    $resultat = $requetePrepa->fetch(PDO::FETCH_ASSOC);

    // Si la requete n'a rien renvoyé
    if(!$resultat){
        // retourne false
        return false;
    }
    // Si la requete à renvoyé quelques choses
    else{
        // Créer un nouvelle objet Utilisateur
        $unUtilisateur = new Utilisateur(null,null,null,null,null,null,null,null,null);
        // Remplie l'objet
        $unUtilisateur->hydrate($resultat);
        // Retourne l'utilisateur
        return $unUtilisateur;
    }
}

```

Les commentaires explique comment le code fonctionne. Pour voir la structure de la BDD aller à la fin du document dans les Annexes.

Une fois connecté de nouveau onglets vont apparaître :

```

// Création du menu
$m2lMP = new Menu("m2lMP");
// Ajout des composant
$m2lMP->ajouterComposant($m2lMP->creerItemLien("accueil", "Accueil"));

// Si l'utilisateur est connecté
if(isset($_SESSION['utilisateur'])){
    // Création d'un onglet Formations
    $m2lMP->ajouterComposant($m2lMP->creerItemLien("formations", "Formations"));
    // Si on est connecté en tant que responsable de formation
    if($_SESSION['utilisateur']->getIdFonct() == "002"){
        // Création d'un onglet inscriptions
        $m2lMP->ajouterComposant($m2lMP->creerItemLien("inscriptions", "Inscriptions"));
    }
    // Création d'un onglet Déconnexion
    $m2lMP->ajouterComposant($m2lMP->creerItemLien("connexion", "Déconnexion"));
}
// Si l'utilisateur n'est pas connecté
else{
    // Création d'un onglet Connexion
    $m2lMP->ajouterComposant($m2lMP->creerItemLien("connexion", "Se connecter"));
}

// Permet de créer le menu avec les composant ajouté
// Le menu créer des lien <a> qui ammene vers index.php?page=...
$menuPrincipalM2L = $m2lMP->creerMenu($_SESSION['page'],'page');

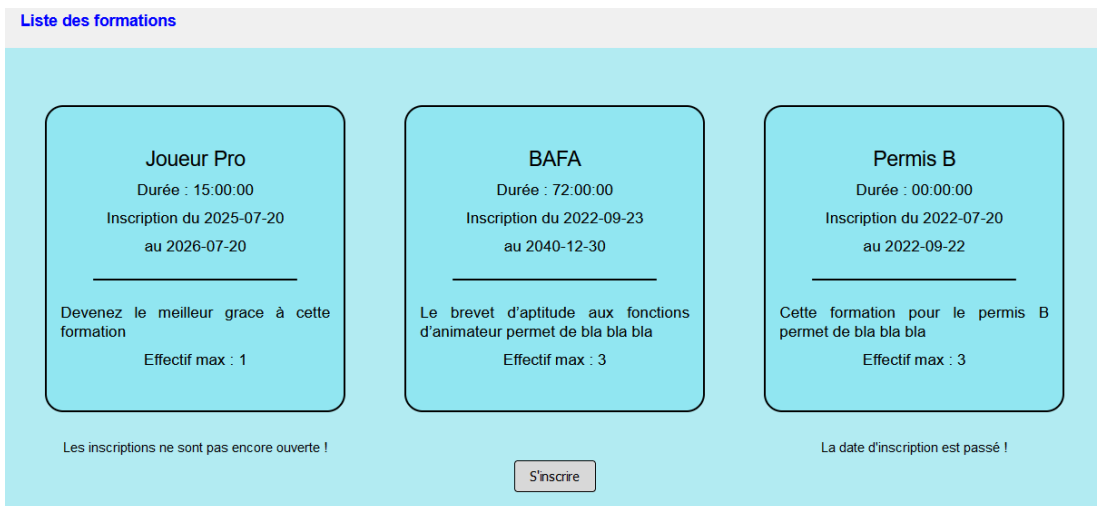
```

Ensuite, on a plus qu'à echo le menu dans la vue. On peut voir que selon que l'on est connecté ou non, qu'on soit un intervenant ou encore un responsable de formation on aura accès à différent onglet dans le menu.

Dans notre cas, en tant qu'intervenant voici les onglets :



Voici la page de l'onglet formation :



Voici le code de la page qui permet d'afficher les formations (contrôleurs/controleurFormation.php):

```
// Pour toute les formations
foreach(Formations::getLesFormations() as $formation){
    // Créer une carte formation
    $html .= $formation->creerCard();
    // Si on est connecté en tant que simple internaute
    if($_SESSION['utilisateur']->getIdFonct() == "001"){
        // Variable pour vérifier si il y a des inscriptions
        $thereIsInscription = false;
        // Récupéré la liste des inscription
        $listeConsulter = Consulters::getLesConsulters();
        if(!empty($listeConsulter)){
            // Pour toute les objet Consulter
            foreach($listeConsulter as $consult){
                // Si l'utilisateur est inscrit à la formation
                if($consult->getIdForma() == $formation->getIdForma()
                && $_SESSION['utilisateur']->getIdUser() == $consult->getIdUser()){
                    // Enregistre qu'il y a des inscriptions
                    $thereIsInscription = true;
                    // Mettre un bouton de Désinscription
                    $html .= $formation->creerButtonDesinscription($consult->getEtat_demande());
                }
            }
        }
    }
}
```



```

// Si il n'y a pas de bouton de désinscription
if(!$thereIsInscription){
    // Si la date pour s'inscrire est passé
    if(Formations::isPassedDate($formation->getDateclotureInscriptions())){
        $html .= $formation->creerMessageInscription("La date d'inscription est passé !");
    }
    // Si le nombre d'inscrit est au max
    else if($formation->getPlacesRestantes() == 0){
        $html .= $formation->creerMessageInscription("Il n'y a plus de place disponible !");
    }
    // Si la date pour s'inscrire est pas encore arrivé
    else if(!Formations::isPassedDate($formation->getDateOuvertInscriptions())){
        $html .= $formation->creerMessageInscription("Les inscriptions ne sont pas encore ouverte !");
    }
    else{
        // Mettre un bouton d'inscription
        $html .= $formation->creerButtonInscription();
    }
}
}

```

Pour chaque objet Formation, on va créer une carte grâce à des méthodes ajouter dans la classe singulière de Formation qui permettent de renvoyer du HTML.

Voici la méthode creerCard :

```

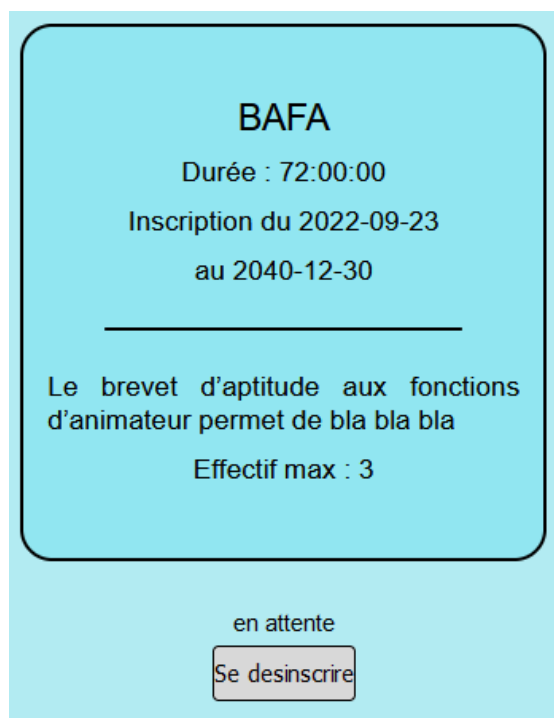
// Méthode qui créer une carte de la formation
0 references | 0 overrides
public function creerCard(){
    $html = "<div class='duo'>";
    $html .= "<div class='card'>";
    $html .= "<p class='titreCard'>". $this->intitule . "</p>";
    $html .= "<p>Durée : " . $this->duree . "</p>";
    $html .= "<p>Inscription du " . $this->dateOuvertInscriptions . "</p>";
    $html .= "<p> au " . $this->dateclotureInscriptions . "</p>";
    $html .= "<div class='trait'></div>";
    $html .= "<p>". $this->descriptif . "</p>";
    $html .= "<p>Effectif max : " . $this->effectifmaximum . "</p>";
    $html .= "</div>";
    return $html;
}

```

Ensuite, on dispose d'autres méthodes qui permettent d'ajouter des boutons ou du texte en dessous. Comme un bouton modifier, supprimer ext...

- Les intervenants peuvent déposer une demande d'inscription pour les formations de leur choix. Ils pourront aussi supprimer une demande. Les intervenants doivent pouvoir à tout moment aussi consulter l'état de leur(s) demande(s) d'inscription aux formations (en attente, refusée, acceptée). Un intervenant n'a accès qu'à ses propres demandes :

Pour faire cette fonctionnalité, j'ai ajouté un bouton en dessous de la carte de chaque formation et un petit texte au-dessus du bouton qui indique l'état de la demande :



BAFA

Durée : 72:00:00

Inscription du 2022-09-23
au 2040-12-30

Le brevet d'aptitude aux fonctions
d'animateur permet de bla bla bla

Effectif max : 3

en attente

Se desinscrire

Voici le code dans contrôleur/contrôleurFormation.php :

```
// Récupéré la liste des inscription
$listeConsulter = Consulters::getLesConsulters();
if(!empty($listeConsulter)){
    // Pour toute les objet Consulter
    foreach($listeConsulter as $consult){
        // Si l'utilisateur est inscrit à la formation
        if($consult->getIdForma() == $formation->getIdForma() && $_SESSION['utilisateur']->getIdUser() == $consult->getIdUser()){
            // Enregistre qu'il y a des inscriptions
            $thereIsInscription = true;
            // Mettre un bouton de Désinscription
            $html .= $formation->creerButtonDesinscription($consult->getEtat_demande());
        }
    }
}
// Si il n'y a pas de bouton de désinscription
if(!$thereIsInscription){
    // Si la date pour s'inscrire est passé
    if(Formations::isPassedDate($formation->getDateclotureInscriptions())){
        $html .= $formation->creerMessageInscription("La date d'inscription est passé !");
    }
    // Si le nombre d'inscrit est au max
    else if($formation->getPlacesRestantes() == 0){
        $html .= $formation->creerMessageInscription("Il n'y a plus de place disponible !");
    }
    // Si la date pour s'inscrire est pas encore arrivé
    else if(!Formations::isPassedDate($formation->getDateOuvertInscriptions())){
        $html .= $formation->creerMessageInscription("Les inscriptions ne sont pas encore ouverte !");
    }
    else{
        // Mettre un bouton d'inscription
        $html .= $formation->creerButtonInscription();
    }
}
```

Consulter est une table qui contient toute les inscriptions (Voir en annexe la structure de la BDD). Je parcours les inscriptions et y ajoute le texte et le bouton nécessaire tout en faisant attention à bien respecter les dates indiquées sur la carte de la formation.

Joueur Pro

Durée : 15:00:00

Inscription du 2025-07-20
au 2026-07-20

Devenez le meilleur grace à cette formation

Effectif max : 1

Les inscriptions ne sont pas encore ouverte !

BAFA

Durée : 72:00:00

Inscription du 2022-09-23
au 2040-12-30

Le brevet d'aptitude aux fonctions d'animateur permet de bla bla bla

Effectif max : 3

en attente

Se desinscrire

Permis B

Durée : 00:00:00

Inscription du 2022-07-20
au 2022-09-22

Cette formation pour le permis B permet de bla bla bla

Effectif max : 3

La date d'inscription est passé !

On peut voir ici que certaines inscriptions n'ont pas encore été ouverte alors que d'autres sont terminer.

- Le responsable formation pouvoir, ajouter, modifier, consulter et supprimer les formations :

Lorsqu'on se connecte en tant que responsable de formation la page change un peu :

Joueur Pro

Durée : 15:00:00

Inscription du 2025-07-20
au 2026-07-20

Devenez le meilleur grace à cette formation

Effectif max : 1

Modifier

Supprimer

BAFA

Durée : 72:00:00

Inscription du 2022-09-23
au 2040-12-30

Le brevet d'aptitude aux fonctions d'animateur permet de bla bla bla

Effectif max : 3

Modifier

Supprimer

Permis B

Durée : 00:00:00

Inscription du 2022-07-20
au 2022-09-22

Cette formation pour le permis B permet de bla bla bla

Effectif max : 3

Modifier

Supprimer

Ajouter une formation

On peut y voir les boutons ajouter, modifier et supprimer une formation.

(Code de la page controleurs/controleurFormations.php)

```
// Si on est connecté en tant que responsable de formation
if($_SESSION['utilisateur']->getIdFonct() == "002"){
    // Ajoute les bouton modifier et supprimer la formation
    $html .= $formation->creerButtonModSuppr();
}

// Fermé l'emplacement formation
$html .= $formation->finEmplacement();
}

// Si on est connecté en tant que responsable de formation
if($_SESSION['utilisateur']->getIdFonct() == "002"){
    // Créer une formulaire
    $boutonAjoutForma = new Formulaire("post", "#", "boutonAjoutForma", "boutonAjoutForma");
    // Ajouter un bouton "Ajouter une formation"
    $boutonAjoutForma->ajouterComposantLigne($boutonAjoutForma->
    creerInputSubmitWithClass("FormaAjout", "FormaAjout", "Ajouter une formation", "buttonAjoutFormation"), 1);
    $boutonAjoutForma->ajouterComposantTab();
    $boutonAjoutForma->creerFormulaire();
}
}
```

Lorsque les cartes formation sont créés, on regarde quel type d'utilisateur est connecté et on y adapte les boutons en dessous.

Si on clique sur Supprimer :

```
// Si le responsable de formation a cliqué sur le bouton sur supprimer
if(isset($_POST["idFormaSuppr"])){
    // Supprime toute les inscription à cet formation
    ConsulterDAO::supprimerAllInscriptions($_POST["idFormaSuppr"]);
    // Supprime la formation
    FormationDAO::SupprFormation($_POST["idFormaSuppr"]);
    // Change les élément dans la DTO
    Consulters::initializeConsulters();
    Formations::initializeFormations();
}
}
```

Si on clique sur Ajouter ou modifier, la page change :

The screenshot shows a web form for managing training sessions. The form is light blue with rounded corners and contains several input fields and buttons. The fields are labeled: 'Titre' (Title), 'Durée:' (Duration) with a format 'hh:mm:ss', 'Inscription du:' (Registration date) with a format 'yyyy-MM-dd', 'au:' (Until) with a format 'yyyy-MM-dd', 'Description' (a larger text area), and 'Effectif max:' (Maximum capacity) with a format 'Effectif maximum'. There are two buttons at the bottom: 'Valider' (Validate) and 'Annuler' (Cancel).

(Code de la page controleurs/controleurFormations.php):

```
// Si le responsable de formation a cliqué sur le bouton Modifier
if(isset($_POST["idFormaModif"])){
    // Pour toute les formation
    foreach(Formations::getLesFormations() as $formation){
        // Si c'est la formation que l'on veut modifier
        if($_POST["idFormaModif"] == $formation->getIdForma()){
            // Créer une carte modifiable de la formation
            $html .= $formation->creerCardModifiable();
            // Fin de l'emplacement
            $html .= $formation->finEmplacement();
        }
    }
}

// Si le responsable de formation a cliqué sur le bouton "Ajouter une formation"
else if(isset($_POST["FormaAjout"])){
    // Créer une carte ajout de formation
    $html .= Formation::creerCardAjoutEmpty();
}
```

En cliquant sur ajouter une carte vide se crée avec un bouton Valider pour créer la formation et si on clique sur modifier ce sera la même carte mais préremplis avec les informations de la carte sur laquelle on vient de cliquer.

```
// Si le responsable de formation a cliqué sur valider modif
if(isset($_POST["idFormaModifValid"])){
    FormationDAO::ModifFormation($_POST["idFormaModifValid"], $_POST["intitule"], $_POST["descriptif"], $_POST["duree"], $_POST["prix"]);
    // Change les élément dans la DTO
    Formations::initializeFormations();
}
```

Si on clique sur valider on lance une requête (ici la requête pour modifier), grâce à la DAO et on réinitialise la classe pluriel associé.

- Le responsable formation pourra aussi consulter la liste des demandes d'inscription, triée pour chaque formation ou pour chaque intervenant :

Lorsqu'on se connecte en tant que responsable de formation un nouvel onglet apparait "Inscriptions" :

Accueil

Formations

Inscriptions

Déconnexion

Voici la page "Inscriptions" :

Tirées par : Formation Intervenant

BAFA
Places restantes : 3

Bob Dupond | Accepter Refuser

Langues vivantes
Places restantes : 1

Thomas Pevre-Clemens | Accepter Refuser

Voici le code de l’affichage dans contrôleurs/controleurInscriptions.php :

```
// Si on a pas de trie (trie pas default) ou alors si on trie par formation
if(!isset($_SESSION["sortedBy"]) || $_SESSION["sortedBy"] == "formation"){
    // Pour chaque consulter trié par formation
    foreach(Consulters::getLesConsultersSortFormation() as $lesConsulterDuneFormation){
        // Crée la carte de la formation avec ses consultants
        $html .= Consulters::getCardInscriptionForFormation(
            Formations::getFormationById($lesConsulterDuneFormation[0]->getIdForma()),$lesConsulterDuneFormation);
    }
}
// Si on a trie par intervenants
else if($_SESSION["sortedBy"] == "intervenant"){
    // Pour chaque consulter trié par intervenant
    foreach(Consulters::getLesConsultersSortInternaute() as $lesConsulterDunInternaute){
        $html .= Consulters::getCardInscriptionForInternaute(
            Utilisateurs::getUtilisateurById($lesConsulterDunInternaute[0]->getIdUser()),$lesConsulterDunInternaute);
    }
}
```

Selon le tri décidé, on affiche les inscriptions dans des cartes par formation ou par utilisateur.

Voici le code de getLesConsulterSortFormation :

```
/**
 * Retourne un tableau avec des tableau pour chaque formations. Chacun des tableau a la liste des objet Consulter de sa formation.
 */
1 reference | 0 overrides
public static function getLesConsulterSortFormation(){
    // On déclare le tableau a renvoyer
    $lesConsulterByFormation = array();
    // Pour toute les formations
    foreach(Formations::getLesFormations() as $uneFormation){
        // On déclare le tableau
        $pourUneFormation = array();
        // Si il y a des consulter
        if(Consulter::getLesConsulter() != null){
            // Pour toute les consulter
            foreach(Consulter::getLesConsulter() as $unConsulter){
                // Si il y a une inscription à cet formation et qu'elle est en attente d'une reponse
                if($uneFormation->getIdForma() == $unConsulter->getIdForma()
                    && $unConsulter->getEtat_demande() == "en attente"){
                    // Ajoute au tableau de cet formation
                    $pourUneFormation[] = $unConsulter;
                }
            }
        }
        // Si il y a des inscription à cet formation
        if(!empty($pourUneFormation)){
            // Ajoute la liste des inscription de la formation à la liste
            $lesConsulterByFormation[] = $pourUneFormation;
            // Supprime la variable (Pour pas garder ce qu'il y a à l'intérieur)
            unset($pourUneFormation);
        }
    }
    return $lesConsulterByFormation;
}
```

getLesConsulterSortFormation permet de renvoyer un tableau contenant des tableaux pour chaque formation. Chacun de ces tableaux contient la liste des objets Consulter de sa formation.

Voici getCardInscriptionForInternaute :

```
1 reference | 0 overrides
public static function getCardInscriptionForInternaute($unUtilisateur, $tableauConsulter){
    $html = "<div class='card widthDuo'>";
    $html .= "<p class='titreCard'>" . $unUtilisateur->getPrenom() . " " . $unUtilisateur->getNom() . "</p>";
    $html .= "<div class='trait margeBottom'></div>";
    foreach($tableauConsulter as $unConsulter){
        // Récupère l'utilisateur
        $uneFormation = Formations::getFormationById($unConsulter->getIdForma());
        $html .= "<div class='unConsulter'>";
        $html .= "<p>" . $uneFormation->getIntitule() . "</p>";
        $html .= "<p>|</p>";
        $html .= "<p>Places restantes : " . $uneFormation->getPlacesRestantes() . "</p>";
        $html .= "<p>|</p>";
        $html .= "<form method='post' class='buttonSortCenter'>";
        $html .= "<input type='hidden' name='inputAccepterInscriptionFormation' value='" . $uneFormation->getIdForma() . "'/>";
        $html .= "<input type='hidden' name='inputAccepterInscriptionUtilisateur' value='" . $unUtilisateur->getIdUser() . "'/>";
        $html .= "<button type='submit' name='buttonAccepterInscription' class='buttonInscription bg-vert'>Accepter</button>";
        $html .= "</form>";
        $html .= "<form method='post' class='buttonSortCenter'>";
        $html .= "<input type='hidden' name='inputRefuserInscriptionFormation' value='" . $uneFormation->getIdForma() . "'/>";
        $html .= "<input type='hidden' name='inputRefuserInscriptionUtilisateur' value='" . $unUtilisateur->getIdUser() . "'/>";
        $html .= "<button type='submit' name='buttonRefuserInscription' class='buttonInscription bg-rouge'>Refuser</button>";
        $html .= "</form>";
        $html .= "</div>";
    }

    $html .= "</div>";
    return $html;
}
```

getCardInscriptionForInternaute permet de créer en html une carte qui contient toutes les inscriptions d'un utilisateur. Il y a une méthode similaire appelé getCardInscriptionForFormation qui permet de créer en html une carte qui contient toutes les inscriptions d'une formation.

L'utilisateur peut donc trier visuellement les inscriptions par formation ou par internaute.

- Le responsable formation pourra modifier l'état des demandes d'inscription (accepter ou refuser). Il devra alors faire en sorte que le nombre d'inscriptions acceptées pour une formation ne dépasse pas l'effectif maximum négocié pour cette formation :

Comme vu précédemment on peut voir à coter de la liste des demandes les boutons accepter et refuser :

The screenshot shows a light blue rounded rectangle representing a user's card. At the top, the name 'Bob Dupond' is centered. Below the name is a horizontal line. Underneath the line, there are two rows of information. Each row consists of a course name, a vertical bar, the number of remaining places, another vertical bar, and two buttons labeled 'Accepter' and 'Refuser'. The first row shows 'BAFA' with 'Places restantes : 3'. The second row shows 'Langues vivantes' with 'Places restantes : 1'.

Bob Dupond					
BAFA		Places restantes : 3		Accepter	Refuser
Langues vivantes		Places restantes : 1		Accepter	Refuser

Voici le code dans contrôleurs/controleurInscription.php :

```
// Si on a cliqué sur accepter une demande d'inscription
if(isset($_POST["buttonAccepterInscription"])){
    ConsulterDAO::accepterInscription($_POST["inputAccepterInscriptionUtilisateur"], $_POST["inputAccepterInscriptionFormation"]);
    // Récupère l'objet formation
    $uneFormation = Formations::getFormationById($_POST["inputAccepterInscriptionFormation"]);
    // Vérifie le nombre de demande restant
    if($uneFormation->getPlacesRestantes() == 0){
        // Refuse toute les autres demandes
        ConsulterDAO::refuserAllInscription($_POST["inputAccepterInscriptionFormation"]);
    }
    // Met à jour la DTO
    Consulters::initializeConsulters();
}

// Si on a cliqué sur refuser une demande d'inscription
if(isset($_POST["buttonRefuserInscription"])){
    ConsulterDAO::refuserInscription($_POST["inputRefuserInscriptionUtilisateur"], $_POST["inputRefuserInscriptionFormation"]);
    // Met à jour la DTO
    Consulters::initializeConsulters();
}
```

Si on clique sur accepter, on lance la requête en BDD puis s'il n'y a plus de place on refuse toute les autres automatiquement. Et on clique sur refuser, on lance simplement la requête.

Voici comment on calcule les places restantes :

(Code dans DTO/singulier/Formation.php)

```
public function getPlacesRestantes(){  
    return $this->effectifmaximum - Consulers::getNbInscrit($this);  
}
```

On retire de l'effectif maximum le nombre de personne déjà inscrite.

(Code dans DTO/pluriel/Consulers.php)

```
public static function getNbInscrit($formation){  
    $nbInscrit = 0;  
    // Si il y a des consuler  
    if(Consulers::getLesConsulers() != null){  
        // Pour chaque consuler  
        foreach(Consulers::getLesConsulers() as $unConsuler){  
            if($formation->getIdForma() == $unConsuler->getIdForma() && $unConsuler->getEtat_demande() == "accepter"){  
                $nbInscrit += 1;  
            }  
        }  
    }  
    return $nbInscrit;  
}
```

On parcourt toutes les inscriptions et regarde celle qui sont accepter pour la formation en paramètre.

Annexe :

Schémas de la BDD :

