ÉNONCÉ DU PROJET 2
Version du 28 avril 2020; 1

Travail à effectuer

Dans ce deuxième projet, on vous demande de concevoir une application *PHP* dont le rôle est de créer les équipes pour un projet de session. **Lisez l'énoncé au complet avant de commencer.**

Assurez-vous d'avoir toujours la dernière version de l'énoncé en main.

Pondération et échéance

☐ Ce projet compte pour **20 %** de la note finale du bulletin.

☐ Date de remise de l'énoncé : **Mardi 28 avril 2020**

☐ Date de remise de la solution : **Mardi 19 mai 2020 en fin de journée**

Une pénalité de 10 % par jour de retard, incluant les jours de congé, sera imposée, et ce, jusqu'à concurrence de 5 jours. (Après ce délai, la note attribuée est zéro.)

Fichier à copier

Remettre le fichier compressé **NomFamille-projet02.zip** via LÉA. Vous devrez vous assurer que tous les fichiers nécessaires sont présents sans quoi il ne sera pas corrigé.

Barème de correction

- **Validité de l'exécution** ..... **100 points**
- **Fluidité d'exécution** ..... jusqu'à **-100 points**
- **Respect de la sortie à l'écran** ..... jusqu'à **-40 points**
- **Présentation du code source**
  Identificateurs significatifs, convention de noms respectée,
  bonne indentation des instructions, cohérence et clarté..... jusqu'à **-20 points**
- **Utilisation des deux classes et des fonctions de librairie** ..... jusqu'à **-60 points**
- **Respect des contraintes**..... jusqu'à **-100 points**

Vous serez évalués sur le produit final. (Une application ne produisant pas de résultats tangibles ne vaut rien.) Mieux vaut remettre une application incomplète, mais fonctionnelle qu'une qui semble complète, mais qui ne produit pas les bons résultats.

Copie du dossier *projet02*

1. À partir de *LÉA*, téléchargez le dossier compressé **projet02.zip** dans le dossier **420-W46**, puis décompressez-le.

2. Renommez le dossier *projet02* pour **NomFamille-projet02** (ex. : *brousseau-projet02*).

Fichiers d'inclusion

L'application devra faire référence aux cinq fichiers d'inclusion ci-dessous :

`424w-cgodin-qc-ca.php` .....Contient les infos de connexion à la base de données
`classe-fichier-2020-mm-jj.php` .....Contient la classe **`fichier`**
`classe-mysql-2020-mm-jj.php` .....Contient la classe **`mysql`**
`librairie-generale-2020-mm-jj.php` .....Contient les fonctions de librairie
`fonctions-specifiques-projet02.php` .....Contient les fonctions spécifiques au projet 2

1. Ouvrez le fichier **424w-cgodin-qc-ca.php**, puis entrez les infos de connexion qui vous permettront de vous connecter à la base de données *mySQL*.

2. Copiez la dernière version des fichiers **`classe-fichier-2020-mm-jj.php`**, **`classe-mysql-2020-mm-jj.php`** et **`librairie-generale-2020-mm-jj.php`** dans le dossier *Nom-Famille-projet02*.

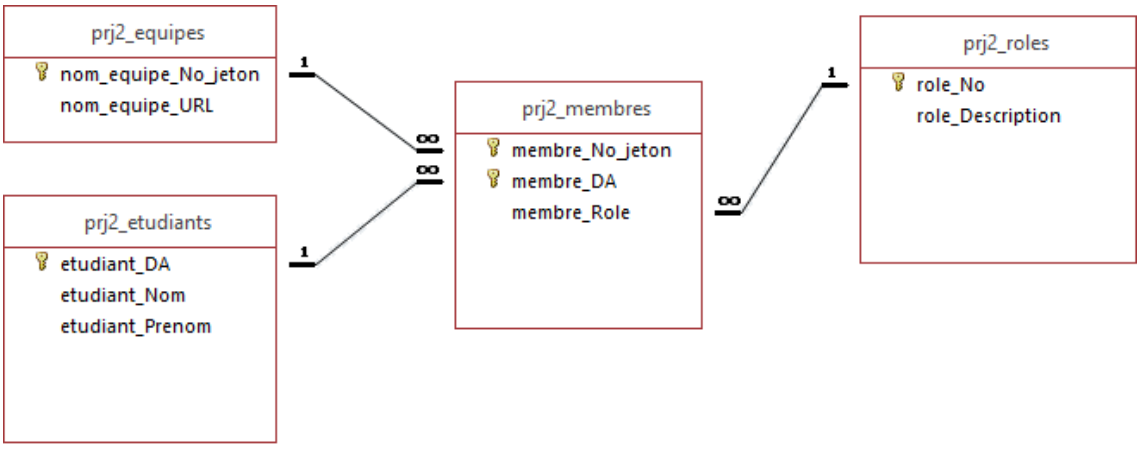
3. Notez la présence du fichier **`fonctions-specifiques-projet02.php`**. (Si vous l'ouvrez, vous constaterez qu'il est vide.)

   Comme indiqué précédemment, ce fichier devra contenir le code source de toutes les fonctions spécifiques au projet 2.

Base de données utilisée pour le projet

C'est la base de données qui vous a été attribuée pour faire l'exercice 4 (**bdh20_Nom-Famille**) qui sera utilisée pour ce projet.

Description des tables nécessaires au projet 2



prj2_equipes

Nom du champ	Type	Remarque
nom_equipe_No_jeton	Entier non nul	Clé primaire
nom_equipe_URL	Max. 15 caractères	

prj2_etudiants

Nom du champ	Type	Remarque
etudiant_DA	Exactement 7 caractères	Clé primaire
etudiant_Nom	Max. 25 caractères	
etudiant_Prenom	Max. 20 caractères	

prj2_membres

Nom du champ	Type	Remarque
membre_No_jeton	Entier non nul	Clés primaires
membre_DA	Exactement 7 caractères	
membre_Role	Entier	

prj2_roles

Nom du champ	Type	Remarque
role_No	Entier non nul	Clé primaire
role_Description	Max. 50 caractères	

Il est important de noter que le nom de chaque table est préfixé par **`prj2`** pour indiquer qu'elles sont utilisées dans le projet 2 et que chaque champ est préfixé du **`nom de la table`** où il déclaré pour faciliter la rédaction des requêtes.

Le fichier texte *etudiants.csv*

Ce fichier contient le **matricule**, le **nom** et le **prénom** des 27 étudiants du cours. Chaque donnée est séparée de l'autre par un point-virgule.

Le contenu de ce fichier doit être copié dans la table *prj2_etudiants* de la base de données au premier démarrage de l'application, puis ne plus être utilisé par la suite.

Le fichier texte *roles.csv*

Ce fichier contient le **numéro** et la **description** de chaque rôle possible dans une équipe. Chaque donnée est séparée de l'autre par un point-virgule.

Le contenu du fichier doit être copié dans la table *prj2_roles* de la base de données au premier démarrage de l'application, puis ne plus être utilisé par la suite.

Les fichiers *equipes.csv* et *membres.csv*

Pour le moment, ces deux fichiers sont vides et il n'est pas prévu de s'en servir dans un avenir rapproché.

Lieu d'entreposage des fichiers *CSV* et des fichiers librairies

Pour des questions de sécurité, il aurait été souhaitable d'entreposer les fichiers *CSV*, comme les fichiers librairie dans des dossiers distincts. Ce n'est que pour une question pratique qu'ils se retrouvent tous dans le même dossier que l'application *index.php*. Il est important que vous le sachiez, même si cela ne change rien au problème à résoudre.

Exécution du démonstrateur

Ne réinventez pas la roue ! Effectuez une copie du code source du démonstrateur pour créer votre fichier **index.php**.

Pour avoir une bonne idée du travail à effectuer, il est suggéré d'exécuter le démonstrateur.

1. Dans la barre d'adresse du navigateur, entrez…

   Votre **matricule** ou **DA**

   `http://424w.lmbrousseau.info/brousseau-projet02/?Matricule=9999999&Trace=0`

   où **`9999999`** corresponds à votre matricule
   `Trace=0` pour afficher les équipes déjà formées et les étudiants qui n'ont pas encore d'équipe; « **`Trace=`** » (sans valeur) masque la trace.

   Un message d'erreur s'affichera si vous ne spécifiez pas de matricule ou que celui-ci n'est pas reconnu.

   Note : Vous n'avez pas à programmer ces deux paramètres dans votre solution.

2. Si vous exécutez le démonstrateur pour la première fois, l'application prend quelques secondes avant d'afficher quoi que ce soit. Ce délai s'explique par la création des quatre tables et de l'établissement des relations entre elles.

   Il est possible à n'importe quel moment de l'exécution de cliquer sur le lien **Réinitialisation des tables** pour reconstruire les tables. Inspirez-vous du code source fourni dans l'énoncé pour créer ce lien, ainsi que le lien **Déconnexion**.

3. Notez qu'à chaque fois que vous cliquez sur un bouton, la page Web est rechargée.

   Dans le cas du démonstrateur seulement, les paramètres **`Matricule=`** et **`Trace=`** apparaîtront à chaque rechargement de la page. Ce ne doit pas être le cas dans votre application.

   D'autres paramètres s'afficheront seulement en cas d'erreur de saisie (Ex. **`tbNoJeton`** ou **`tbNomEquipeCreer`**). Notez que ceci est normal, puisque c'est la méthode **`get`** qui est présentement utilisée. (En mode production, nous devrons remplacer la méthode pour **`post`**.)

4. À chaque (re)démarrage, l'application effectue les étapes présentées dans la section *Déroulement de l'exécution*.

Vous n'avez pas à programmer l'assignation des rôles de chaque équipier, ainsi que produire la trace d'exécution.

Déroulement de l'exécution (10 points)

Important : Chaque clic de bouton force le rechargement de l'application en mémoire.

À des fins pédagogiques, vous devez effectuer toutes les opérations en *PHP*, même si le *JavaScript* aurait dû être mis à contribution pour la validation.

Partie *PHP* (exécutée avant le chargement de la page *Web*)

1. Détection de la présence de la table **`prj2_etudiants`**. En cas d'absence…
   a. Création de la structure de la table à partir des spécifications.
   b. Remplissage de la table à partir du fichier **`etudiants.csv`**.

2. Détection de la présence de la table **`prj2_roles`**. En cas d'absence…
   a. Création de la structure de la table à partir des spécifications.
   b. Remplissage de la table à partir du fichier **`roles.csv`**.

3. Détection de la présence de la table **`prj2_equipes`**. En cas d'absence…
   a. création de la structure de la table à partir des spécifications.

4. Détection de la présence de la table **`prj2_membres`**. En cas d'absence…
   a. création de la structure de la table à partir des spécifications.

5. Établissement des relations entre les tables si nécessaire.

6. Au premier démarrage de l'application …

   L'étudiant saisit son **matricule** (*numéro de DA*), puis clique sur le bouton **VALIDER**. Dès que c'est fait, il rechargement de la page, puis authentification.

   - Si l'étudiant est reconnu, il y a création de la variable de session *InfosEtudiant*, puis entreposage du **DA**, **nom** et **prénom** de l'étudiant dans cette dernière.
   - Si l'étudiant n'est pas reconnu, un message d'erreur s'affiche et le traitement s'arrête.

   Pour chaque redémarrage de l'application, cette dernière teste uniquement la présence de la variable de session.

7. Dès que l'étudiant est authentifié, le panneau *Confirmation de l'identité* s'affiche pour lui indiquer son nom complet. (Cette donnée est disponible via la variable de session.)

8. L'application détermine si l'étudiant fait partie d'une équipe ou non (un enregistrement est-il associé à ce dernier dans la table *prj2_membres* ou y a-t-il un numéro de jeton associé dans la variable de session ?).

Si l'étudiant fait partie d'une équipe, l'application recherche tous les membres qui en font partie.

L'application affiche dans le panneau *Confirmation de l'identité* si l'étudiant fait partie d'une équipe ou non et si c'est le cas, affiche le numéro de jeton associé et les membres la composant.

Pour terminer, un menu s'affiche pour lui indiquer les commandes qu'il est autorisé à effectuer.

À chaque bouton correspond une action soit (1) **CRÉER** une équipe, (2) **JOINDRE** une équipe existante, (3) **RENOMMER** une équipe et (4) se **RETIRER** d'une équipe.

9. En fonction de l'action demandée, la fonction correspondante sera exécutée (**`creeEquipe`**, **`jointEquipe`**, **`renommeEquipe`** ou **`retireEquipe`**). Référez-vous à la section *Fonctions spécifiques à créer pour le projet*.

Fonctions spécifiques à créer pour le projet (90 points)

Comme indiqué précédemment, toutes les fonctions à créer doivent être entreposées dans le fichier **fonctions-specifiques-projet02.php**. Comme pour les fichiers librairies, elles doivent apparaître en ordre alphabétique de noms.

1. Création de la table *`prj2_etudiants`* (2 points)

Prototype : **`creeTableEtudiants($BDProjet02, $strNomTableEtudiants)`**

But : Créer la structure de la table **`prj2_etudiants`**. Référez-vous à la section *Description des tables nécessaires au projet 2* à la page 2.

Contrainte : Vous devez utiliser la méthode *creeTableGenerique*.

2. Chargement de la table *`prj2_etudiants`* (2 points)

Prototype :
```
remplitTableEtudiants($BDProjet02,
                      $strNomTableEtudiants,
                      $strNomFichierEtudiants)
```

But : Charger la table **`prj2_Etudiants`**. Référez-vous à la section *Le fichier texte etudiants.csv* à la page 3.

Contrainte : Vous devez utiliser la méthode *insereEnregistrement* et les méthodes de la classe *fichier*.

3. Création de la table *`prj2_roles`* (2 points)

Prototype : **`creeTableRoles($BDProjet02, $strNomTableRoles)`**

But : Créer la structure de la table **`prj2_roles`**. Référez-vous à la section *Description des tables nécessaires au projet 2*.

Contrainte : Vous devez utiliser la méthode *creeTableGenerique*.

4. Chargement de la table *prj2_roles* (2 points)

Prototype : **remplitTableRoles($BDProjet02, $strNomTableRoles, $strNomFichierRoles)**

But : Charger la table **prj2_roles**. Référez-vous à la section *Le fichier texte roles.csv* à la page 3.

Contrainte : Vous devez utiliser la méthode *insereEnregistrement* et les méthodes de la classe *fichier*.

5. Création de la table *prj2_equipes* (2 points)

Prototype : **creeTableEquipes($BDProjet02, $strNomTableEquipes)**

But : Créer la structure de la table **prj2_equipes**. Référez-vous à la section *Description des tables nécessaires au projet 2*.

Contrainte : Vous devez utiliser la méthode *creeTableGenerique*.

6. Création de la table *prj2_membres* (2 points)

Prototype : **creeTableMembres($BDProjet02, $strNomTableMembres)**

But : Créer la structure de la table **prj2_Membres**. Référez-vous à la section *Description des tables nécessaires au projet 2*.

Contrainte : Vous devez utiliser la méthode *creeTableGenerique*.

7. Authentification de l'étudiant (10 points)

Prototype : **authentifieEtudiant($BDProjet02, $strNomTableEtudiants, $strDA)**

But : Confirmer que l'étudiant est bien autorisé à se connecter.

Si c'est le cas, création de la variable de session **InfosEtudiant**, puis entreposage du **DA**, **nom**, **prénom** et **nom complet** de l'étudiant dans cette dernière. Référez-vous à la section *Création de la variable de session InfosEtudiant* à la page 10.

Retour : **true** si l'étudiant est authentifié; **false** autrement.

Contrainte : Vous devez utiliser les méthodes *selectionneEnregistrements* et *contenuChamp*.

8. Récupération des informations liées à une équipe (10 points)

Prototype : `function recupereInfosEquipe($BDProjet02, $strNomTableEquipes, $strNomTableMembres, $strDA, &$intNoJeton, &$strNomEquipe)`

But : Récupérer s'il a lieu, le **numéro de jeton** et le **nom d'une équipe** dont l'étudiant fait partie.

9. Récupération du nom complet des membres d'une équipe (10 points)

Prototype : `function recupereMembresEquipe($BDProjet02, $strNomTableEtudiants, $strNomTableMembres, $intNoJeton, &$tListeEquipiers)`

But : Récupérer le nom complet de chaque membre de l'équipe dont l'étudiant fait partie.

Retour : Nombre d'équipiers.

10. Création d'une équipe (10 points)

Prototype : `creeEquipe(paramètres à votre discrétion)`

But : Permettre à l'étudiant de créer une nouvelle équipe.

Validations : Référez-vous à la section *Gestion des consignes et des messages d'erreur* à la page 10 pour avoir une idée des validations à effectuer.

Contraintes : Le **numéro de jeton** (10000 à 99999) est généré par l'application. Même si les chances sont faibles, vous devez vous assurer que l'application ne génère pas un numéro de jeton existant.

Les éléments `["NoJeton"]` et `["NomEquipe"]` de la variable de session sont mis à jour.

Deux tables sont mises à jour (*prj2_equipes* et *prj2_membres*).

Il y a rechargement de la page en cas de réussite de l'opération.

11. Se joindre à une équipe (10 points)

Maximum 3 étudiants par équipe

Prototype : `jointEquipe(paramètres à votre discrétion)`

But : Permettre à l'étudiant de se joindre à une équipe existante.

Validations : Référez-vous à la section *Gestion des consignes et des messages d'erreur*.

Contraintes : Les éléments `["NoJeton"]` et `["NomEquipe"]` de la variable de session sont mis à jour.

Seule la table *prj2_membres* est mise à jour.

Il y a rechargement de la page en cas de réussite de l'opération.

12. Renommer une équipe (16 points)

Prototype : **`renommeEquipe(paramètres à votre discrétion)`**

But : Permettre à l'étudiant de changer le nom de l'équipe dont il fait partie.

Validations : Référez-vous à la section *Gestion des consignes et des messages d'erreur*.

Contraintes : L'élément **`["NomEquipe"]`** de la variable de session est mis à jour.
Seule la table *prj2_equipes* est mise à jour.
Il y a rechargement de la page en cas de réussite de l'opération.

13. Se retirer d'une équipe (12 points)

Prototype : **`retireEquipe(paramètres à votre discrétion)`**

But : Permettre à l'étudiant de se retirer de l'équipe dont il fait partie.

Validations : Référez-vous à la section *Gestion des consignes et des messages d'erreur*.

Contraintes : Les éléments **`["NoJeton"]`** et **`["NomEquipe"]`** de la variable de session sont mis à jour.
Deux tables sont mises à jour (*prj2_equipes* et *prj2_membres*).
Il y a rechargement de la page en cas de réussite de l'opération.

Création de la variable de session *InfosEtudiant*

Au démarrage de l'application, l'étudiant est invité à se connecter en saisissant son matricule (**numéro de DA**). Dès que l'authentification est complétée, une variable de session contenant les données suivantes, entreposées sous forme d'un tableau, est créée :

`$_SESSION["InfosEtudiant"]`
- **`["DA"]`** ..... Numéro de **DA** entreposé dans la BD
- **`["Nom"]`** ..... **Nom** entreposé dans la BD
- **`["Prenom"]`** ..... **Prénom** entreposé dans la BD
- **`["NomComplet"]`** ..... « **Prénom** » « **Nom** »
- **`["NoJeton"]`** ..... **No de jeton** qui sera attribué éventuellement : **`null`**
- **`["NomEquipe"]`** ..... **Nom de l'équipe** dont il fera partie éventuellement : **`null`**

Gestion des consignes et des messages d'erreur

Il est suggéré d'initialiser chaque consigne au démarrage de l'application, puis de la remplacer par un message d'erreur si la condition s'y prête.

Exemple :

```
$strMessageSaisieDA = "Composé de 7 chiffres";
...
if (!$binAuthentificationReussie) {
   $strMessageSaisieDA = "<span class=\"sGras sRouge\">" . ERREUR_01 . "</span>";
}
...
<span class="sConsigne sGras"><?php echo $strMessageSaisieDA; ?></span>
```

Voici la liste des messages d'erreur qui devront être utilisés dans l'application. Même s'il est suggéré de les entreposer en mémoire sous forme de constantes, vous êtes libre de procéder de la façon que vous désirez.

DA

Erreur 01 : DA invalide ou non reconnu !

Équipes

Erreur 11 : Aucun nom d'équipe saisi !
Erreur 12 : Nom d'équipe ne débute pas par une lettre ou n'est pas composé de 5 à 15 lettres minuscules (ou caractères de soulignement) !
Erreur 13 : Nom d'équipe déjà utilisé !
Erreur 14 : Nom d'équipe saisi est identique au nom actuel !

Numéro de jeton

Erreur 21 : Aucun numéro de jeton saisi !
Erreur 22 : Numéro de jeton saisi n'est pas un entier ou dans l'intervalle 10000 à 99999 !
Erreur 23 : Numéro de jeton saisi ne correspond pas à votre numéro de jeton !
Erreur 24 : Numéro de jeton saisi ne correspond à aucune équipe enregistrée !

Autres

Erreur 31 : L'équipe est complète ! (maximum 3 étudiants par équipe)

LIENS *DÉCONNEXION* ET *RÉINITIALISATION DES TABLES*

Le lien Déconnexion (-5 points si absent)

Partie HTML

```
<a class="sConsigne sBleu"
   href="index.php?Deconnexion=Oui&Matricule=<?php echo $Matricule; ?>
         &Trace=<?php echo $binTrace; ?>">Déconnexion</a>
```

Partie PHP

```
if (isset($_GET["Deconnexion"])) {
   $_SESSION["InfosEtudiant"] = null;
   header("Location: index.php?" . (isset($_GET["Matricule"]) ? "Matricule=" . $_GET["Matricule"] : "") .
                                   "&" . (isset($_GET["Trace"]) ? "Trace=" . $_GET["Trace"] : ""));
   die();
}
```

Pour recharger la page courante

1. La fonction **`header()`** en *PHP* :
   http://php.net/manual/fr/function.header.php
2. L'instruction **`window.location`** en *JavaScript* :
   https://www.w3schools.com/js/js_window_location.asp

Le lien Réinitialisation des tables (-5 points si absent)

Partie Javascript

```
function confirmationDemandee() {
   if (confirm("Cliquer sur OK pour reconstruire les tables; autrement ANNULER.")) {
      window.location = 'index.php?Reinitialisation=Oui&Matricule=<?php echo $Matricule; ?>
                                                        &Trace=<?php echo $binTrace; ?>';
   }
}
```

Partie HTML

```
<a class="sConsigne sBleu" href="javascript:confirmationDemandee()">Réinitialisation des tables</a>
```

Partie PHP

```
if (isset($_GET["Reinitialisation"])) {
   $_SESSION["InfosEtudiant"] = null;
   $BDProjet02->supprimeTable($strNomTableMembres); $BDProjet02->supprimeTable($strNomTableEtudiants);
   $BDProjet02->supprimeTable($strNomTableRoles);   $BDProjet02->supprimeTable($strNomTableEquipes);
   header("Location: index.php?" . (isset($_GET["Matricule"]) ? "Matricule=" . $_GET["Matricule"] : "") .
                                   "&" . (isset($_GET["Trace"]) ? "Trace=" . $_GET["Trace"] : ""));
   die();
}
```