*Laboratoire 2-1 – Espace membre*

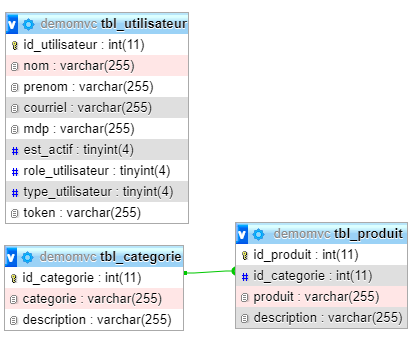
**Remise** : Déposez sur Moodle un zip contenant tous les fichiers du site Web.

Vous partirez du laboratoire précédent pour ajouter l’authentification d’un utilisateur qui peut être fait localement ou via Google. Vous trouverez à la fin du laboratoire un schéma montrant les fichiers impliqués et un diagramme des différentes méthodes à coder.

Dans un prochain laboratoire : Un administrateur pourra ajouter, modifier, supprimer un produit ou une catégorie ainsi que consulter les commandes. Un client pourra effectuer une commande et voir ses commandes.

**Étape préliminaire**

1. Mettre à jour la base de données « demomvc » avec le script fourni dans ce zip.



L’utilisateur présent en BD est :   
Courriel : [admin@gmail.com](mailto:admin@gmail.com)   
mdp : allo123

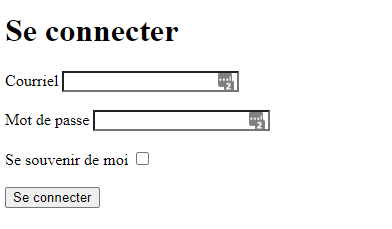
**est\_actif** (0=inactif; 1=actif)  
**role\_utilisateur** (0=utilisateur; 1=admin)  
**type\_utilisateur** (0=normal; 1=Google)  
**token** : Contiendra une chaine de caractère pour la validation post-inscription (à venir)

**Partie 1 - Ajout de l’authentification classique**Dans cette section nous allons ajouter la gestion d’authentification sur le site Web. Comme nous l’avons fait dans le cours précédent, l’utilisateur pourra se connecter avec un courriel et un mot de passe.

1. **La classe « Utilisateur ».**

Dans le répertoire « model », créez un fichier **Utilisateur.php**. Ce fichier contiendra la classe Utilisateur avec les propriétés pour chaque colonne dans la table tbl\_utilisateur ainsi qu’un constructeur et les getters et setters. Codez cette classe.

1. **Gestionnaire d’utilisateur.**

Dans le répertoire « model », créez un fichier **UtilisateurManager.php** qui contiendra la classe **UtilisateurManager**. Nous y reviendrons plus tard pour créer des fonctions.

1. **La vue Authentification**Dans le dossier view, créez un fichier nommé **loginView.php**. Créez cette vue qui permettra l’affichage du formulaire de connexion. Voir le résultat attendu pour le rendu formulaire. Le formulaire doit envoyer ses données à index.php et contenir un champ caché (hidden) nommé « action » qui portera la valeur « authentifier »
2. **Contrôleur Utilisateur**

Dans le dossier contrôleur, créez un fichier nommé **controllerUtilisateur.php**.

En vous inspirant de **controllerProduit.php**, ajoutez un require pour le gestionnaire utilisateur soit **UtilisateurManager** et créez une fonction **getFormConnexion()** qui affiche le formulaire de connexion (loginView.php).

1. **Ajuster le routeur index.php**

Dans le fichier index.php, ajouter une condition elseif qui devra valider si la variable action est égale à « connexion ». Dans cette condition, ajouter un require vers le contrôleur d’utilisateur et un appel à la fonction que vous avez créé dans cette dans ce fichier soit **getFormConnexion()**.

En terminant, ajouter un lien dans la section <nav> permettant d’atteindre la page connexion. Si tout va bien en tapant l’URL <http://localhost/mvc/connexion> vous devriez voir le formulaire de connexion.

1. **Gérer l’authentification**  
   Dans le gestionnaire d’utilisateur, créez la fonction **verifAuthentification()** qui reçoit en paramètre un courriel et un mot de passe. Cette fonction doit valider l’authentification de l’utilisateur dans la BD et retourner un objet utilisateur.

Pour y arriver, l’authentification devra se faire en 2 étapes. La première est de récupérer l’utilisateur en BD qui a le courriel reçu en paramètre. Il serait préférable d’effectuer cette tâche dans une autre fonction, car on la réutilisera plus tard (Créez une fonction **getUtilisateurParCourriel($email)**). Cette fonction devrait retourner un objet Utilisateur s’il y a un résultat *null* sinon. De retour dans la fonction **verifAuthentification()**, si la fonction **getUtilisateurParCourriel()** a retourné un résultat, validez le mot de passe à l’aide de la fonction password\_verify(). Si le mot de passe correspond, on retourne l’utilisateur.

Dans le contrôleur, créez la fonction **authentifier()** recevant en paramètre un courriel et un mot de passe. Cette fonctionappellera la fonction verifAuthentification() du gestionnaire. En terminant, cette fonction charge le contrôleur de l’accueil et appelle la fonction **listProduit()** si le résultat de **verifAuthentification()** contient un utilisateur et on ajoute dans la session le courriel et le rôle de l’utilisateur sinon on appelle la fonction qui affiche le formulaire de connexion.

Dans le fichier **index.php**, ajouter une condition qui validera si la variable action est égale à « authentifier » et qu’il y a un courriel et un mot de passe dans $\_REQUEST. Dans cette condition, utilisez le contrôleur d’utilisateur pour appeler la fonction **authentifier()** en lui envoyant le courriel et le mot de passe reçu en $\_REQUEST.   
  
La commande session\_start() irait très bien au début de index.php. Remplacez l’utilisation de $\_GET par $\_REQUEST pour uniformiser les appels. Profitez-en pour déplacer le bloc de débogage qui affiche le contenu de $\_REQUEST dans le fichier template.php. Dans template.php ajoutez avant le bloc de navigation <nav> une condition qui valide s’il existe dans la session une variable contenant le courriel de l’utilisateur signifiant que l’utilisateur est authentifié puis si c’est le cas affichez « Bienvenue *courriel\_de\_l’utilisateur* ». Ajoutez une condition autour du lien redirigeant vers la page connexion. Si l’utilisateur est connecté, modifiez ce lien pour « Se déconnecter » et dirigez ce lien vers **index.php/deconnexion**.

1. **Testez votre système d’authentification. *Rappel du mot de passe de*** [***admin@gmail.com***](mailto:admin@gmail.com)***: allo123***
2. **Déconnexion**Ajoutez dans index.php, une condition qui valide qu’action vaut « deconnexion ». Dans cette condition, ajoutez un require du contrôleur Utilisateur et appelez la fonction deconnexion() que nous créerons à l’instant. Dans le contrôleur Utilisateur créez cette fonction qui a pour but de terminer la session et charger l’accueil.

**Partie 2 - Ajout de l’authentification via Google**

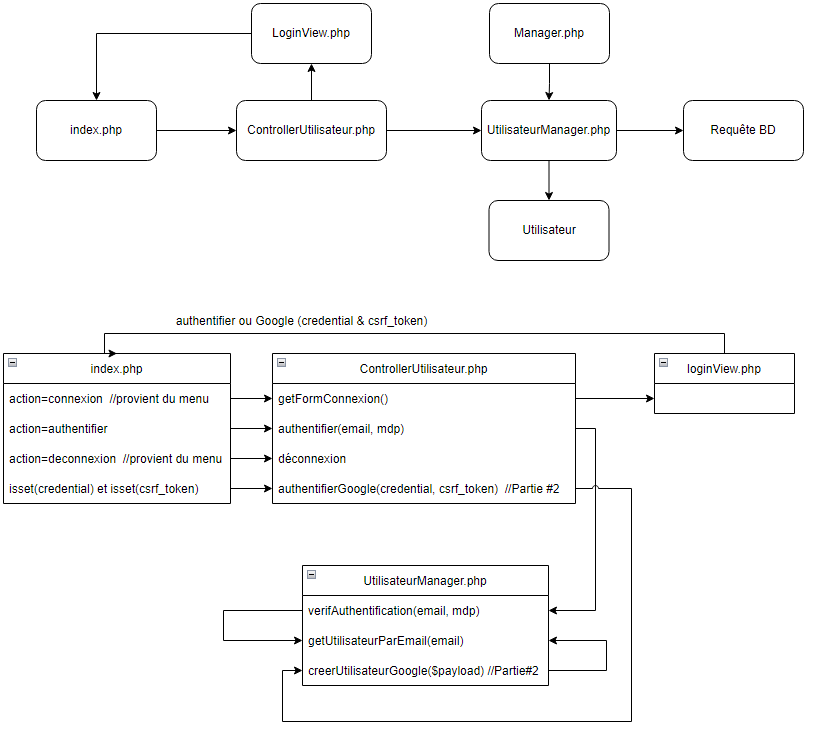
1. Vous aurez besoin de la documentation de Google : <https://developers.google.com/identity/gsi/web/guides/overview>
2. À l’étape des **Prérequis**, vous devrez accéder à la **Console API Google** puis vous aurez à créer un projet (le nom du projet n’a pas d’importance).
3. Une fois le projet créé, vous devrez **Créer des informations d’identifications de type OAuth ID Client**.  
     
   Une image contenant texte

   Description générée automatiquement
4. Suivez les étapes de **Configurer l’écran de consentement** avec les paramètres suivant :
   1. Écran 0 - User Type
      1. Sélectionnez **Externe**
   2. Écran 1 – Écran de consentement OAuth
      1. Donnez un nom à votre application (peu importe)
      2. Entrez l’adresse courriel de votre compte Google
      3. Entrez ce même courriel dans le champ **Coordonnées du développeur**.
   3. Écran 2 – Niveaux d’accès
      1. Sélectionnez **Ajouter ou Supprimer des champs d’application** puis sélectionnez les 2 premières cases en lien avec le courriel et le profile.
   4. Écran 3 – Utilisateurs test
      1. Sélectionnez **Add Users** et entrez votre courriel
   5. Écran 4 – Résumé
      1. Sélectionnez **Revenir au tableau de bord**.
5. Retournez à la création d’un identifiant de type **ID client OAuth** (Comme le montre l’image ci-dessus). Sélectionnez le type **d’application Web**. Le nom n’a pas d’importance, mais vous devrez ajouter <http://localhost> dans la section **Origines JavaScript autorisées** et **URI de redirection autorisés**. Une fois l’identifiant enregistré, conservez l’ID client et le code secret en note.
6. Passez à l’étape suivante dans la documentation Google soit **Charger la bibliothèque cliente**. Cette étape vous donnera un code JavaScript à ajouter dans le <head> du fichier template.php.
7. Passez à l’étape de la documentation Google nommée **Comprendre le bouton personnalisé**. Lisez cette page puis passer à l’étape suivante. La partie **Rendu des boutons** vous donne le code HTML à insérer dans le fichier **loginView.php** pour ajouter le bouton. Copiez seulement les 2 <div> et ajoutez les sous le formulaire que vous avez codé à la partie 1. Pensez à remplacer **YOUR\_GOOGLE\_CLIENT\_ID** par le code que vous avez conservé précédemment et remplacez l’URL <https://your.domain/your_login_endpoint> par l’URL de votre page d’accueil. Cet URL doit se terminer par un « / ».
8. Avec cette procédure, vous devriez être capable de vous authentifier sur Google, mais nous ne traitons pas encore la réponse que Google nous envoie. Nous pouvons constater que nous recevons en paramètre $\_REQUEST **credential** et **g\_csrf\_token**.  
     
   *N.B. La variable* ***g\_csrf\_token*** *contient un code qui devrait aussi être présent dans vos Cookies. Ceci est une mesure de sécurité afin d’éviter des attaques CSRF (Cross-Site Request Forgery). Pour valider cela, il suffirait de comparer que la variable reçue en $\_REQUEST correspond à celle reçue dans les COOKIEs. Pour sauver un peu de temps nous n’effectuerons pas cette étape.*

*N.B. Pour se déconnecter lorsque vous utilisez Google, il faut vous déconnecter de votre compte Google. Un petit truc, utilisez le mode « Navigation privée » ce sera plus simple.*

1. Dans la documentation de Google, une section intitulée [**Vérifiez le jeton d'identification Google côté serveur**](https://developers.google.com/identity/gsi/web/guides/verify-google-id-token) Utilisez cette procédure pour y arriver. Un exemple PHP vous est fourni plus bas dans cette page. À noter qu’il faudra charger l’API de Google à l’aide de Composer (Voir la documentation sur Moodle en lien avec Composer).  
     
   Une fois l’API Google installé, ajoutez une condition else if dans index.php qui valide que vous recevez en $\_REQUEST un champ nommé **credential** et un autre nommé **g\_csrf\_token**. Dans cette condition, ajoutez un require au contrôler Utilisateur et appelez une fonction nommée **authentificationGoogle** qui recevra en paramètre les 2 variables reçues en $\_REQUEST. Nous créerons cette fonction.  
     
   Dans le contrôler Utilisateur, créez la fonction **authentificationGoogle** qui reçoit les 2 paramètres indiqués précédemment. Insérez dans cette fonction l’exemple PHP fourni par Google en remplaçant **$CLIENT\_ID** par votre code client (le même que vous avez utilisé au point #7) et remplacez **$id\_token** par le paramètre de la fonction correspondant à **credential**.
2. Si tout a bien fonctionné, vous devriez avoir dans la variable $payload les données du client, dont son courriel, son nom et son prénom. Cela confirme que Google a réussi l’authentification de l’utilisateur. Faites un print\_r de cette variable sous la ligne de code $userid = $payload[‘sub’] et valider les informations contenues dans cette variable.
3. Vous devez ensuite identifier cet utilisateur dans votre BD. Il est possible qu’il existe déjà ou que se soit un nouvel utilisateur. Utiliser la fonction du gestionnaire d’Utilisateur pour chercher un utilisateur via son courriel. Si cela fonctionne vous obtiendrai un objet utilisateur. Si le retour est null c’est que l’utilisateur n’existe pas encore. Il faut donc le créer. Ajouter une fonction dans le gestionnaire d’Utilisateur permettant d’ajouter dans la BD un utilisateur avec le courriel, le nom et le prénom contenu dans la variable **$payload**. En créant l’utilisateur assurez-vous qu’il soit actif, que son rôle est normal et que son type est Google (voir le schéma de BD pour connaitre les valeurs à mettre). La fonction qui ajoute l’utilisateur doit retourner un objet de type Utilisateur au contrôleur. Dans le contrôleur, il faudra ensuite ajouter le courriel et le rôle dans la session et charger l’accueil.

*N.B. Pour le moment nous avons authentifié l’utilisateur, mais nous n’avons pas protégé des pages.   
À suivre.*

**