Hiver 2018 **Cégep Limoilou** **Département d'informatique** **Professeurs :** **Martin Simoneau**	TP2 **ChitChat** PHP CodeIgniter MVC, BD, authentification, validations **20%** de la note finale

Objectifs

- Utiliser le MVC avec CodeIgniter
- Utiliser des fichiers XML pour tenir les informations du modèle.
- Faire la lecture/écriture dans une base de données
- Gérer l'authentification avec *IonAuth*

Consignes :

- Ce travail fait partie des évaluations sommatives et a une pondération de 20% de la note finale.
- Remettre votre code sur Léa à la date indiquée.
- Vous devez être en équipe de 2 et tenir un **journal des activités** (qui fait quoi et quand).
- Vous devez remettre un **d'état de projet** indiquant
    - Ce qui fonctionne
    - Ce qui ne fonctionne pas et ce qui a été tenté pour corriger le problème
    - Les indications utiles pour faire fonctionner et utiliser votre site.
- Le dossier de votre site et son URL doivent contenir vos initiales ex : **TP2MSD**/page/…

Objectifs :

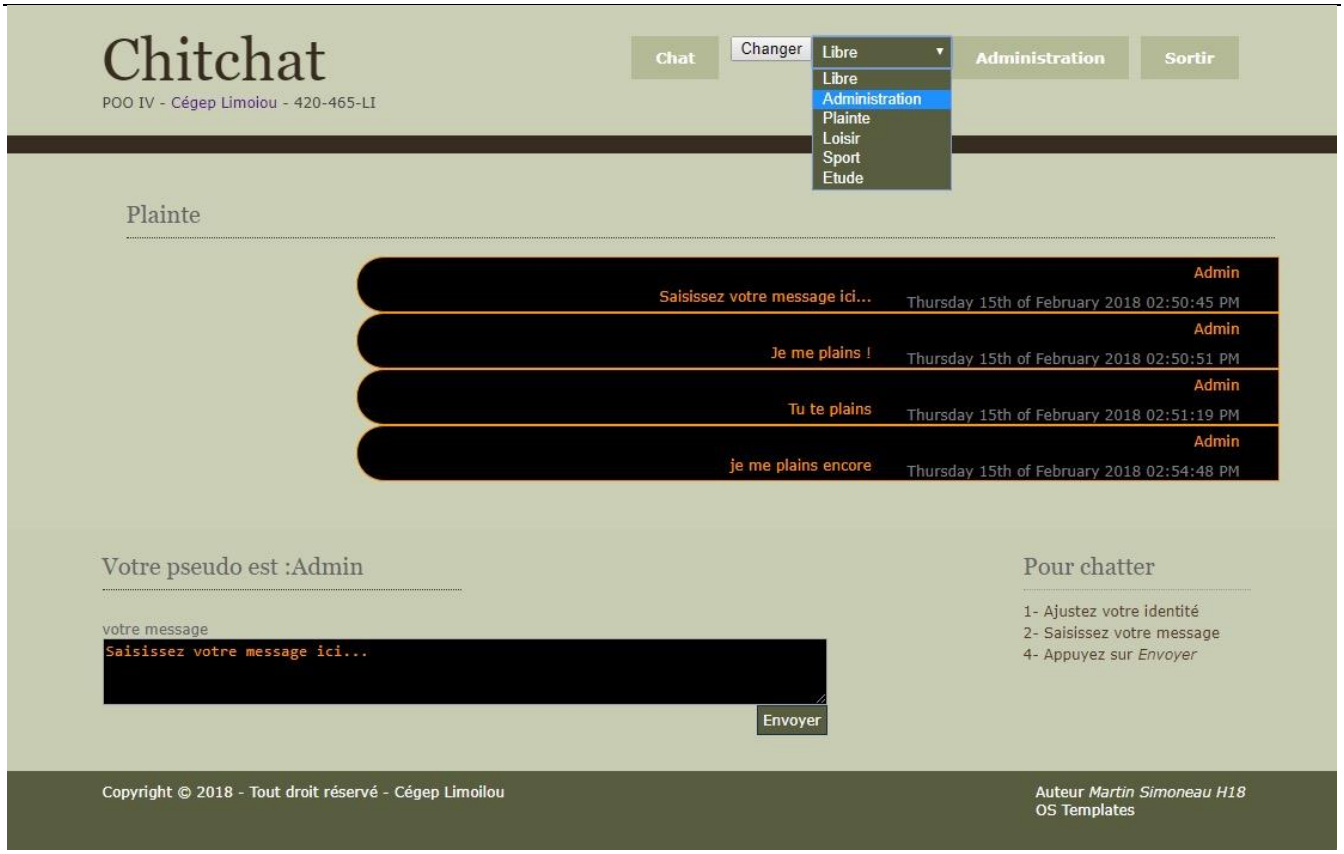


Image à titre indicatif seulement: vous devez utiliser votre propre gabarit

- De vous amener à créer un site web PHP permettant à ses utilisateurs de communiquer dans différentes salles de conversation(chat). Le site affichera l'information contenue soit dans un fichier XML soit dans une base de données. Les utilisateurs s'authentifieront avec le plugin *Ion_auth*. Le site aura 3 niveaux d'authentification (non connecté, connecté et admin).
  - Le non connecté ne peut que lire les messages
  - Le connecté peut écrire des messages
  - L'admin peut en plus gérer les utilisateurs et effacer des messages

Les utilisateurs devront s'authentifier pour accéder au site puis pourront ajouter des messages qui seront écrits dans la base de données ou dans le fichier XML selon la salle choisie.

Contrôleurs (Conversation, Salles, Accueil et Auth)

Ajoutez un contrôleur nommé ***Conversation*** qui servira à gérer les fonctions de chat. Notez qu'on ne vous demande pas de créer des salles dynamiquement ou de les effacer, les salles sont toutes prédéterminées et elles ne changent pas.

Le contrôleur *Conversation* devra minimalement posséder les éléments suivants: ***afficheSalle()*** qui fait afficher l'ensemble de tous les messages pour une salle donnée et pour un utilisateur donné (tous les messages apparaissent à gauche sauf les messages de l'utilisateur qui consulte le site qui apparaitront à droite). Chaque message doit inclure, le texte, la date de réception du message ainsi que le pseudo de celui qui l'a envoyé.

- ***ajouteMessage***() qui permet à un utilisateur d'ajouter un message dans une salle. Le message doit passer les validations suivantes :

- Ne pas contenir les mots d'une liste de mots vulgaires (environ 10 mots) à vous de les choisir. Elle doit obligatoirement contenir « *merde* » et « *fuck* »). Cette validation doit être faite dans le modèle
- Ne pas dépasser 80 caractères.
- Avoir au moins un caractère
- Si les validations ne passent pas, le message doit être ajouté quand même, mais il contiendra le texte « *message invalide* ou censuré»

Faites également un contrôleur nommé ***Salles*** qui servira à choisir la salle affichée.
Le site comporte 3 salles en BD et 3 salles en XML.

- En BD :
  - Loisir (connecté)
  - Sports (connecté)
  - Études (connecté)
- Sur fichiers XML
  - Administration (admin)
  - Plaintes (public)
  - **Libre** (public)

- **La salle Libre dans le fichier XML est la salle par défaut lorsque rien n'a encore été sélectionné.**
- **Même si un utilisateur ferme son navigateur, lorsqu'il l'ouvre à nouveau il doit se retrouver dans la même salle et avec le même statut de connexion.**

Le contrôleur *Salles* devra avoir les méthodes :

- **changeSalle***()* permet d'entrer dans une salle. L'utilisateur qui entre doit être connu. L'utilisateur devra choisir parmi l'une des 6 salles disponibles. Le contrôleur *Salle* pourra utiliser le modèle *SalleXML* ou *salleBD* selon le choix de la salle. Le contrôleur devra valider qu'il s'agit de l'une des 6 salles permises.
  - Comme validation :
    - La salle est requise
    - Le choix de salle doit être dans la liste des salles.
- ***accueil***() donne un aperçu général du site et don son utilisation. C'est une méthode sans comportement spécifique. Faites en sorte que cette méthode doit être appelée par défaut lorsqu'on accède au site pour la première fois.

Vous devrez également adapter le contrôleur ***Auth*** pour qu'il fonctionne harmonieusement avec votre site.

Modèles

Vous devrez fabriquer 3 modèles, dont 2 qui feront la même chose, enregistrer et retourner les messages pour les différentes salles en BD. Le premier modèle s'appellera ***SalleXML,*** le second Modèle ***SalleBD*** *et le troisième* ***SalleInfo***. Plus loin, 2 sections sont consacrées à la description de la BD et des données XML. Les salles XML et BD devront avoir les méthodes :

- ***ajouteMessage***() Cette méthode ajoute un nouveau message dans une salle
  - En BD chaque message de la table *messages* est associée à une salle de la table *salles*.
  - *Avec les fichiers XML, chaque salle correspond à 1 fichier différent. Tous les fichiers sont dans le même dossier (application/donnees/salles/)*

- ***retrouveTousLesMessage****($salle)* cette méthode retrouve tous les messages pour une salle donnée.
- Les salles en BD doivent être créées dans votre script SQL

Le modèle ***SalleInfo*** est la classe qui connaît toutes les salles et qui fait les validations. Elle contient 3 tableaux en attributs :

- ***SalleBD*** qui contient les 3 noms de salles en BD
- ***SalleXML*** qui contient les 3 noms de salles sur fichier XML
- ***motsInterdits : la liste des mots interdits***

Elle devra contenir les méthodes:

- ***isBDorXML***() : indique si le nom reçu est contenu en BD ou dans les fichiers XML. Retourne faux si le nom n'est contenu dans aucun des deux systèmes.
- **isTextAllowed**() : retourne faux si le texte reçu contient un mot interdit (voir *ajouteMessage* dans le contrôleur Conversation).

Vue

Pour ce TP vous devrez utiliser un gabarit de site (voir exercice 9). Vous pouvez réutiliser celui que vous avez fait avec l'exercice 9. (Vous ne pouvez pas utiliser celui qui a été remis avec la solution de l'exercice9, utilisé dans l'image du début).
Le site comporte les vues suivantes :

- ***Accueil*** (lancer par le contrôleur *Salles*)
- ***AfficheTout*** (la page principale qui affiche le chat)
- ***Login*** (de Ion_auth)
- Les pages d'erreurs de CI ou de votre gabarit HTML
- Les gabarits ***header*** et **footer**.

AfficheTout est la vue la plus importante. Sans dans cette dernière que se retrouve :

- L'affichage du chat
  - Le nom de la salle en cours.
  - Les messages provenant de l'utilisateur connecté devront apparaître à droite alors que les autres messages apparaitront à gauche. Chaque message doit contenir le texte la date et le pseudo de l'utilisateur.
  - Les messages doivent être triés. Le dernier message reçu doit être en bas de la liste.
- L'affichage d'un petit formulaire pour saisir un nouveau message. ATTENTION, si l'utilisateur n'est pas connecté, le formulaire de saisie doit être remplacé par un lien qui permet de se connecter (vers le login de ion_auth).
- Un texte statique qui indique les étapes à suivre pour utiliser le chat (doit être clair et très concis).
- Si vous n'utilisez pas un script AJAX pour recharger votre site automatiquement, vous devrez rajouter un bouton ***refresh*** pour permettre à l'utilisateur de mettre à jour le contenu du chat.

Gabarit Header, est aussi une pièce très importante de votre site. Il contiendra :

- Le nom du site ***ChitChat*** doit apparaître de façon importante
- Le numéro du cours et le nom du cégep avec un hyperlien qui pointe vers le site du cégep
- Un menu contenant :
  - Un lien vers le chat (en tout temps) Ce menu permet aussi de rafraichir la page.
  - Un mini formulaire pour demander un changement de salle. Il devrait se mettre à jour automatiquement en fonction des tableaux contenus dans le modèle *SalleInfo*. Il contient :

- Un menu déroulant avec les différentes salles (seules les salles permises en fonction de la connexion voir la description des salles plus haut).
- Un bouton pour envoyer le formulaire

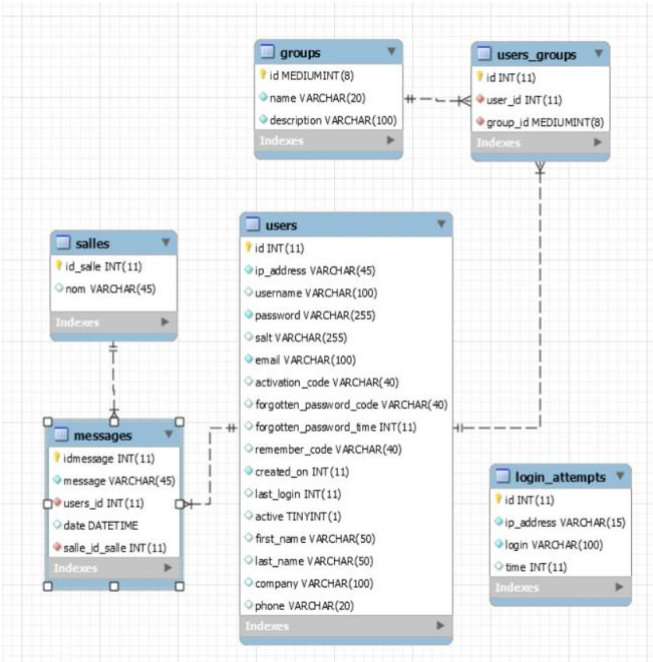
- Un menu qui permet de sortir (si l'utilisateur est connecté). Il suffit d'appeler le *logout* de Ion_auth.
- Un menu qui permet d'administrer le site (fonction Administration de Ion-Auth).(Seulement pour les administrateurs).

Modèle de la BD

- Le script BD vous est fourni. Vous devez utiliser ce script. le modèle en BD est le suivant:
  - Les tables **groups**, **users_groups**, **users** et **login_attemps** sont fournies par le script SQL de ion_auth
  - Vous devrez ajouter les tables **salles** et **messages**
  - La table *messages* est liée à la table *users* par: ***messages.users_id=users.id***
  - La table *messages* est liée à la table *salles* par: ***messages.salle_id_salle=salles.id_salle***
  - C'est le champ ***first_name*** de la table *users* qui servira implicitement de pseudo pour l'usager
  - IMPORTANT, gardez une copie de votre **script** de création de BD dans le dossier **TP2 /application/SQL**
  - C'est le *first_name* Ion-auth qui servira de pseudo pour le site. On peut l'obtenir avec :

```
$this->ion_auth->user ()->row ()->first_name ;
```

  - Les 3 salles doivent être créées avec le script de la BD



Modèle de la XML

Le modèle XML est relativement simple.

- Le dossier application*data**XMLdata*\ de votre projet doit contenir tous les fichiers XML requis. Le programme doit les créer s'ils ne sont pas déjà présents.
- Chaque salle est contenue dans un seul fichier dont le nom est le nom de la salle avec l'extension .xml.
- Le format du fichier doit être comme suit:

```
<?xml version="1.0" encoding="UTF-8"?>
<salle-de-chat>
   <nom>Libre</nom>
   <messages>
         <message date="Sunday 11th of February 2018 06:55:16 PM"
               pseudo="Admin">
               <contenu>allo zozo</contenu>
         </message>
         <message date="Sunday 11th of February 2018 06:55:57 PM"
               pseudo="pikabou">
               <contenu>allo</contenu>
         </message>
```

```
    </messages>
</salle-de-chat>
```

- Notez que la date actuelle peut être obtenue avec la fonction:
  `date('l jS \of F Y h:i:s A')`

NB La BD peut mettre elle-même la date du message

Authentification et affichage

- Intégrer **ion_auth** à votre code.
  - Copiez tous les fichiers du dossier ion_auth vers les dossiers correspondants de votre application. Au départ il existe déjà un utilisateur (admin@admin.com avec pw :password).
  - Assurez-vous que chaque méthode de vos contrôleurs est protégée par ion_auth
  - Le contrôleur *Auth*, vous fournit déjà tout le code pour gérer les utilisateurs. Mais vous devez:
    - Changer l'affichage pour qu'il respecte le style du TP. L'affichage est assuré par la fonction *_render_*page() à la fin de la classe Auth.
    - Modifier toutes les vues pour n'afficher que ce qui est accessible en fonction du niveau de connexion.
      - **Non connecté** :
        - On voit la page qui permet de se connecter (page Ion_Auth). Toute tentative d'Accéder au site sans être connecté devront retourner l'utilisateur à cette page.
      - Une fois **connecté,** on verra, en plus :
        - un bouton sortir
        - Les salles en BD
      - Si c'est **l'admin** qui est connecté, il aura accès au contenu complet du site
        - Gestion des comptes (Ion_auth)
        - La salle *Administration*

Améliorations générales (15%)

- Vous devez effectuer les améliorations sur l'architecture de votre site en utilisant un contrôleur maison pour :
  - Éviter de recopier du code :
    - Lorsqu'il faut appeler successivement la vue pour l'entête, le fichier voulu et le pied de page
    - Pour l'authentification de chaque méthode
- Saisir toutes les requêtes qui entrent sur le serveur dans le fichier de log du serveur. Vous devez saisir
  - Le nom de l'utilisateur (ou invité s'il n'est pas connecté)
  - La fonctionnalité demandé (afficher le chat ou l'accueil, ajouter un message, administrer le site).
- Utiliser les routes pour faire disparaître les noms des contrôleurs *Salle* et *Conversation* dans les URL de votre site.

Bonus (10%)

En utilisant le decorateur qui vous a été remis avec le projet, en décorant les classes de modèle des salles, faites en sorte que la couleur des messages d'un utilisateur change à la troisième visite (seulement à la troisième visite).

Informations complémentaires

- **Redirect** : pour faire afficher la page d'un autre contrôleur. Par exemple si votre contrôleur effectue un travail qui ne demande pas d'affichage différent (par exemple changer de salle). Codeigniter possède https://www.codeigniter.com/user_guide/helpers/url_helper.html

```
redirect('Conversation/index',"reload");
```

- **Entité :** Vos fichiers XML seront en UTF-8 et vous aurez peut-être de la difficulté avec les accents. Penser à utiliser les utilitaire de Codeigniter pour encoder les entités !

Critères d'évaluation :

- Respect des exigences
- Clarté et qualité du code
  - Noms de variables et de méthodes clairs et complet
  - PHPDoc pour les méthodes importantes
  - Code facile à lire et bien découpé
  - Commentaires pertinents
  - Algorithme clair, robuste et efficace.
- **Respect du modèle MVC**
  - **Affichage uniquement dans les vues**
  - **Le moins de traitement possible dans les contrôleurs et vues**
  - **Toutes les connexions passent par les contrôleurs**
  - **Validations demandées dans les modèles**
  - **Session et gestion des entrées uniquement dans les contrôleurs**
- Programme fonctionnel (sans bogue)
- Interface usager propre et sans ambiguïtés.
- Usage efficace et maximal des librairies et helpers de CI.
- Site bien gérer en fonction des niveaux d'accès.
- Documentation PHP (PHPDoc) pour chacune des méthodes que vous avez programmées.
- Qualité du journal des activités et du rapport d'état de votre projet
- La qualité de votre architecture
  - Factorisation du code en commun
  - Utilisations appropriées des classes
  - Algorithmes sans détour et efficaces.
  - **Ne pas charger du code qui ne sert à rien**
  - **Ne pas remettre du code qui ne fonctionne pas en commentaire**