Forum JMC

# Contexte

## Fonctionnement général

Il est demandé de concevoir un forum simplifié, dans lequel il est possible :

* Tant qu’aucune connexion n’est réalisée (mode ‘anonyme’) :
  + De consulter la liste des sujets (‘*topic’*) existants ;
  + Puis, sur chaque sujet, la liste des messages (‘*post’*) associés ;
* Lorsqu’un utilisateur s’est authentifié :
  + D’ouvrir un nouveau sujet ;
  + De modifier un sujet existant, pour autant qu’il en soit l’auteur ;
  + De supprimer un sujet existant, pour autant que celui-ci ne comporte aucun message et qu’il en soit l’auteur ;
  + De verrouiller un sujet, qui ne pourra alors plus être modifié ni supprimé ;
  + De déverrouiller un sujet verrouillé ;
  + D’ajouter un nouveau message dans un sujet ;
  + De modifier ou supprimer un message existant dans un sujet, pour autant qu’il en soit l’auteur.

L’utilisateur peut se connecter et se déconnecter à tout moment. On passe alors du mode ‘anonyme’ au mode ‘authentifié’, et réciproquement.

S’il ne possède pas encore de compte, il peut en créer un à partir de la page de connexion. Un nouvel utilisateur est automatiquement connecté.

## Contraintes de sécurité

Un nom d’utilisateur (‘*pseudo’*) doit être composé de 6 à 32 caractères alphanumériques ou tirets (‘-‘ ou ‘\_’), et commencer par une lettre. En outre, un pseudo doit être unique, quelle que soit la casse utilisée : ‘jmi-65’, ‘JMi-65’ et ‘JMI-65’ sont équivalents par exemple, et ne peuvent exister simultanément.

Un mot de passe (‘*passwd’*) doit être composé d’au moins 8 caractères, et comprendre au moins une lettre minuscule, une lettre majuscule, un chiffre et un caractère spécial. Les mots de passe seront enregistrés de manière cryptée dans la base de données.

Tous les champs de saisie sont validés avant utilisation et enregistrement.

Un utilisateur dûment authentifié peut toujours :

* Ajouter un nouveau sujet ;
* Ajouter un nouveau message dans un sujet existant non clôturé.

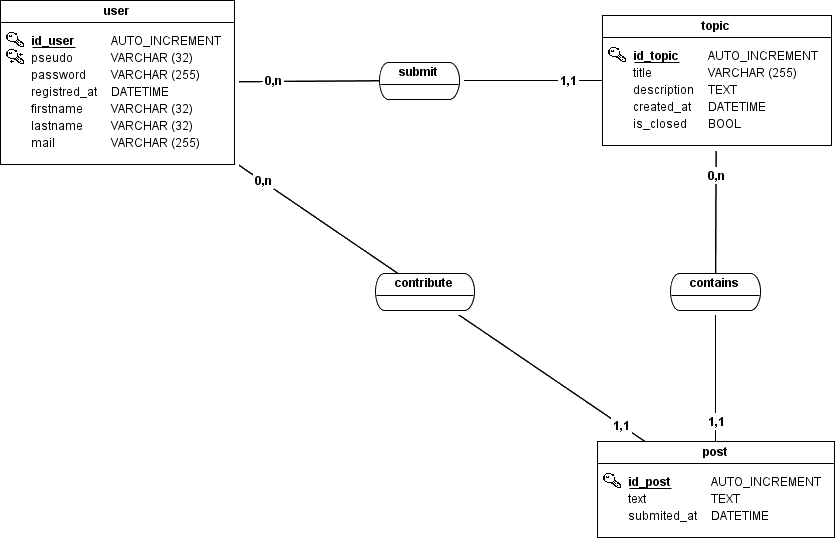
Par contre, il ne peut effectuer les actions suivantes que sur les objets dont il est l’auteur :

* Clôturer un sujet ;
* Modifier ou supprimer un sujet non clôturé ;
* Modifier ou supprimer un message d’un sujet non clôturé.

Selon les cas, l’action sera rendue inactive (bouton grisé) ou invisible (boutons non affichés).

# Modèle conceptuel de donnéées

Le schéma suivant montre le MCD mis en œuvre :



Dans la table ‘**user**’ (ensemble des utilisateurs enregistrés pouvant publier sur le forum) :

* Le champ ‘registered\_at’ reçoit automatiquement le ‘CURRENT\_TIMESTAMP’ à la création ;
* Les champs ‘firstname’ et ‘lastname’ peuvent être non renseignés ;
* Le champ ‘mail’ n’est pas rendu ‘UNIQUE’, autorisant plusieurs utilisateurs à renseigner la même adresse mail.

Dans la table ‘**topic**’ (ensemble des sujets du forum) :

* Le champ ‘created\_at’ reçoit automatiquement le ‘CURRENT\_TIMESTAMP’ à la création ;
* Le champ ‘description’ est un texte libre (limité à 65 536 caractères, et pouvant comporter des sauts de ligne).

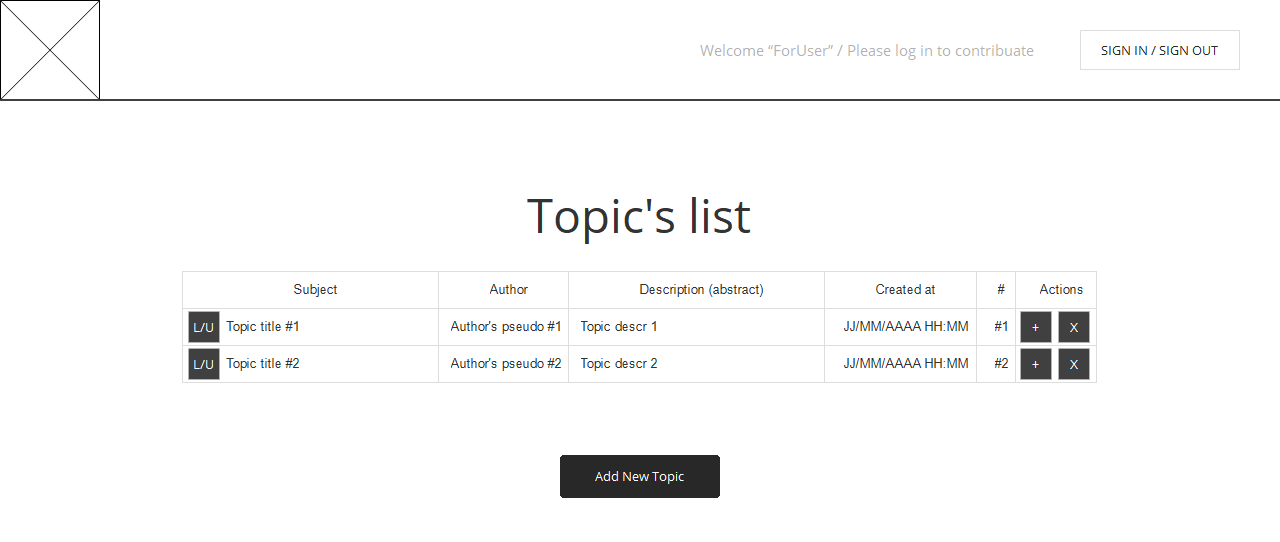
Dans la table ‘**post**’ (ensemble des messages du forum) :

* Le champ ‘submitted\_at’ reçoit automatiquement le ‘CURRENT\_TIMESTAMP’ à la création ;
* Le champ ‘text’ est un texte libre (limité à 65 536 caractères, et pouvant comporter des sauts de ligne).

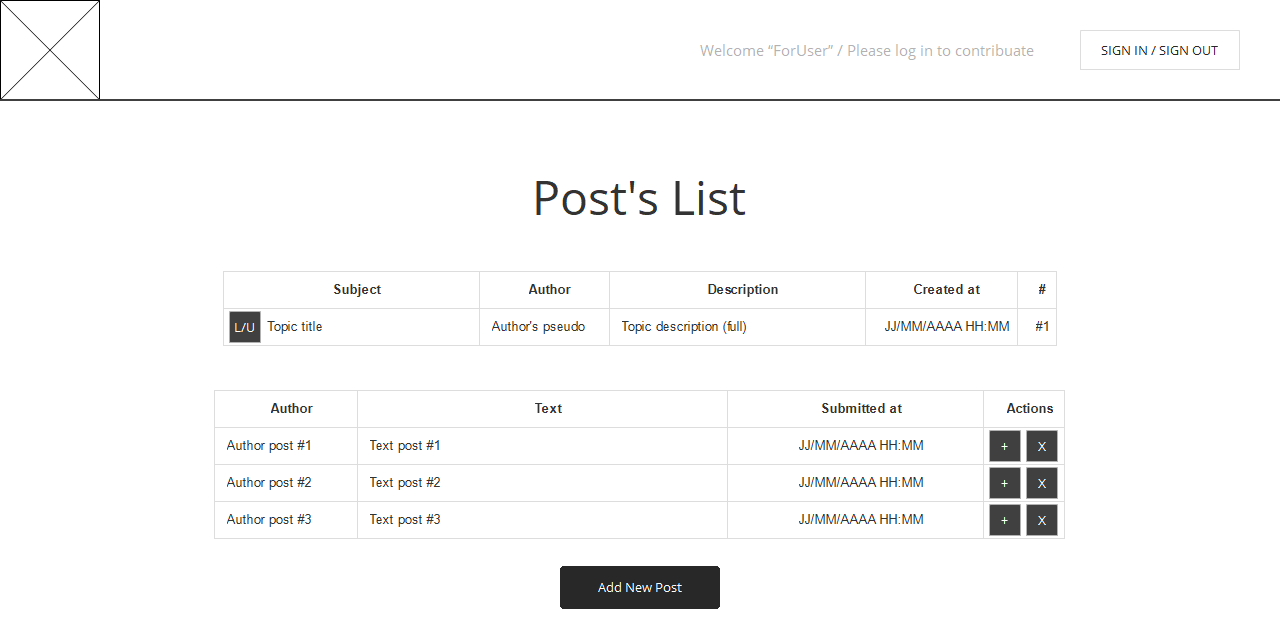
# Maquettage préalable

Les principales pages du forum sont présentées ci-après :

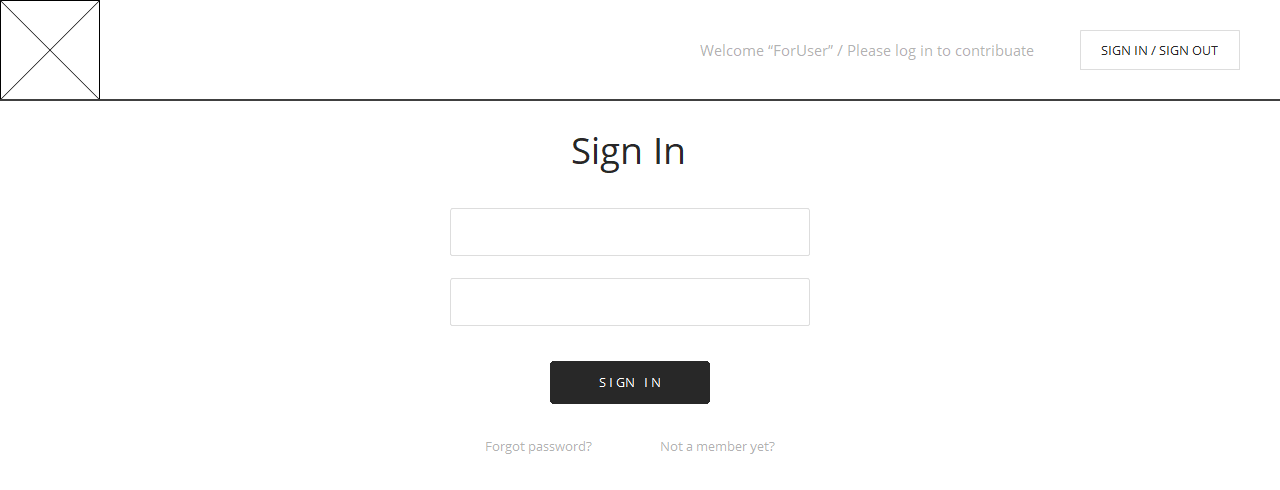
## Page d’accueil



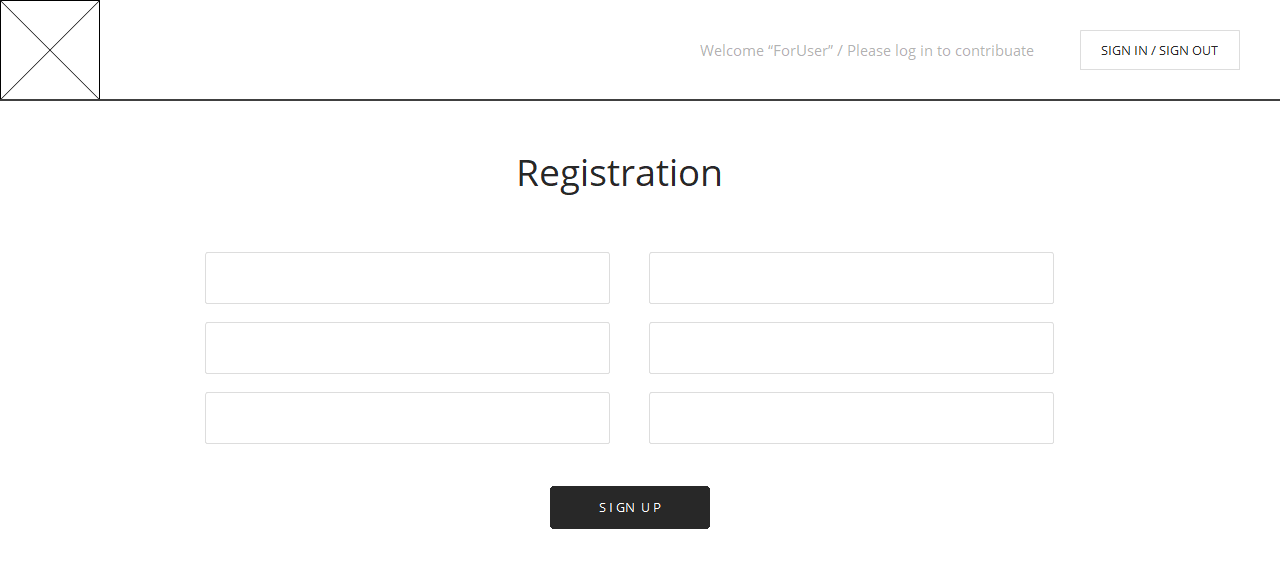
## Détail d’un sujet



## Connexion d’un utilisateur existant



## Inscription d’un nouvel utilisateur



# Modèle de conception mis en œuvre

## Principe général

Le *design pattern* mis en œuvre est du type « Modèle-Vue-Présentation ».

Il est basé sur une unique page publique « **index.php** » à la racine du site, qui analyse l’URL reçue, toujours sous la forme « **http://index.php?ctrl=...&action=...&id=...** », pour en extraire :

* Le « contrôleur » qui va traiter la requête (‘*controller*’), en se basant sur modèle (‘*model*’) associé ;
* L’ « action » à déclencher dans ce contrôleur, en lui passant un éventuel argument « id ».

La méthode « action » du « contrôleur » retourne alors un résultat (‘*result*’) qui permettra :

* D’appeler la « vue » (‘*view*’) qui va produire l’affichage souhaité ;
* En lui passant les données (‘*data*’) dont elle aura besoin.

Le code HTML construit par la « vue » est alors transmis à la page d’affichage commune « **layout.php** », ce qui permet de s’assurer d’un visuel unique et cohérent pour l’ensemble du site.

Le fichier « **index.php** », outre le mécanisme décrit ci‑dessus, définit l’ensemble des constantes globales contenant la structure arborescente du site, à savoir :

|  |  |  |
| --- | --- | --- |
|  |  |  |

define("BASE\_PATH",  "src/");

define("CTRL\_PATH",  BASE\_PATH."controllers/");

define("MODEL\_PATH", BASE\_PATH."models/");

define("VIEW\_PATH",  "views/");

define("PUB\_PATH",   "public/");

define("CSS\_PATH",   PUB\_PATH."css/");

define("IMG\_PATH",   PUB\_PATH."img/");

define("JS\_PATH",    PUB\_PATH."js/");

## Exemple appliqué aux sujets

Le modèle précédent, appliqué aux sujets du forum, donnera donc :

* Un unique modèle « **TopicModel.class.php** », présent dans le dossier « **src/models** » ;
* Un unique contrôleur « **TopicController.class.php** », présent dans le dossier « **src/controllers** » ;
* Plusieurs vues associées chacune à une présentation différente de données , toutes dans le dossier « **views** ».

### Le modèle de gestion des sujets

Le fichier « **TopicModel.class.php** » fait l’interface entre le contrôleur et la base de données.

Il contient donc un ensemble de méthodes réalisant chacune une requête SQL, et retournant son résultat.

<?php

    // Modèle de gestion d'un sujet (interface BdD associée au contrôleur de gestion des sujets)

    require\_once MODEL\_PATH."dbcModel.class.php" ;

    final class TopicModel extends DBC {

        public function \_\_construct() {

            parent::dbConnect() ;

        }

        // Obtention de la liste complète des sujets

        public function getTopicsList() {

            $sql = "SELECT t.id\_topic, t.title, t.description, t.created\_at, t.is\_closed, u.pseudo,  
  COUNT(p.id\_post) AS nb\_posts FROM topic t

                    JOIN user u

                    ON t.id\_user = u.id\_user

                    LEFT JOIN post p

                    ON t.id\_topic = p.id\_topic

                    GROUP BY t.id\_topic

                    ORDER BY t.created\_at DESC" ;

            $stmt = self::$\_connexion->query($sql) ;

            return ($stmt->fetchAll()) ;

        }

        // Obtention d'un topic en particulier (par son identifiant)

        public function getTopic($tid) {

            $sql = "SELECT t.id\_topic, t.title, t.description, t.created\_at, t.is\_closed, u.pseudo,   
 COUNT(p.id\_post) AS nb\_posts FROM topic t

                    JOIN user u

                    ON t.id\_user = u.id\_user

                    AND t.id\_topic = :tid

                    LEFT JOIN post p

                    ON t.id\_topic = p.id\_topic" ;

            $stmt = self::$\_connexion->prepare($sql) ;

            $args = array("tid" => $tid);

            $stmt->execute($args) ;

            return ($stmt->fetch()) ;

        }

        // Méthode appelée pour clôturer ou rouvrir un sujet (drapeau binaire <=> bascule)

        public function toggleTopic($tid) {

            $sql = "UPDATE topic

                    SET is\_closed = (1 - is\_closed)

                    WHERE id\_topic = :tid" ;

            $stmt = self::$\_connexion->prepare($sql) ;

            $args = array(

                "tid"   => $tid) ;

            return ($stmt->execute($args)) ;

        }

        // Ajout d'un nouveau topic avec ses informations (validées au préalable)

        public function addTopic($title, $descr, $uid) {

            $sql = "INSERT INTO topic (title, description, id\_user)

                    VALUES (:title, :descr, :uid)" ;

            $stmt = self::$\_connexion->prepare($sql) ;

            $args = array(

                "title" => $title,

                "descr" => $descr,

                "uid"   => $uid) ;

            return ($stmt->execute($args)) ;

        }

        // Modification d'un topic existant avec de nouvelles informations (validées au préalable)

        public function updateTopic($tid, $title, $descr) {

            $sql = "UPDATE topic

                    SET title = :title, description = :descr

                    WHERE id\_topic = :tid" ;

            $stmt = self::$\_connexion->prepare($sql) ;

            $args = array(

                "title" => $title,

                "descr" => $descr,

                "tid"   => $tid) ;

            return ($stmt->execute($args)) ;

        }

        // Suppression d'un topic existant (après vérification et confirmation)

        public function deleteTopic($tid) {

            $sql = "DELETE FROM topic

                    WHERE id\_topic = :tid" ;

            $stmt = self::$\_connexion->prepare($sql) ;

            $args = array(

                "tid"   => $tid) ;

            return ($stmt->execute($args)) ;

        }

    }

?>

### Le contrôleur de gestion des sujets

Le fichier « **TopicController.class.php** » fait l’interface entre le contrôleur et la base de données.

Il contient donc un ensemble de méthodes réalisant chacune une des actions attendues (extraite de l’URL), et retournant le résultat sous la forme d’un tableau associatif de deux éléments :

* « **view** » 🡺Le nom de la « vue » que devra appeler le contrôleur principal,
* « **data** » 🡺Et les éventuelles données que celle-ci pourra afficher.

Selon les cas, ces données peuvent être des informations structurées, un simple message d’erreur, voire sans objet (« **null** »).

<?php

    require\_once CTRL\_PATH."ControllerInterface.class.php";

    require\_once MODEL\_PATH."TopicModel.class.php";

    final class TopicController implements ControllerInterface {

        private $\_topic ;

        public function \_\_construct(){

            $this->\_topic = new TopicModel();

        }

        public function indexAction(){

            return array(

                "view" => 'viewTopics.php',

                "data" => $this->\_topic->getTopicsList()

            );

        }

        // Méthode appelée pour initier l'ajout d'un nouveau sujet

        public function newAction() {

            return array(

                "view" => 'viewAddTopic.php',

                "data" => null

            );

        }

        // Méthode appelée pour clôturer ou rouvrir un sujet

        public function toggleAction($tid) {

            $status = $this->\_topic->toggleTopic($tid) ;

            if ($status) {

                // Si ajout OK, on redirige vers la page d'accueil

                header("Location:index.php?ctrl=Topic") ;

            } else {

                // Sinon, on affiche le message d'erreur

                $error = "Unknown error !!! Please try again..." ;

            }

            return array(

                "view" => 'viewTopics.php',

                "data" => $this->\_topic->getTopicsList()

            );

        }

        // Méthode appelée en validation du formulaire d'ajout d'un nouveau sujet

        public function addAction() {

            $title = filter\_input(INPUT\_POST, 'title', FILTER\_SANITIZE\_STRING, FILTER\_FLAG\_STRIP\_LOW) ;

            $descr = filter\_input(INPUT\_POST, 'descr', FILTER\_SANITIZE\_STRING) ;

            if (isset($\_SESSION['user'])) {

                $uid = $\_SESSION['user']['id\_user'] ;

                $status = $this->\_topic->addTopic($title, $descr, $uid) ;

                if ($status) {

                    // Si ajout OK, on redirige vers la page d'accueil

                    header("Location:index.php?ctrl=Topic") ;

                } else {

                    // Sinon, on affiche le message d'erreur

                    $error = "Unknown error !!! Please try again..." ;

                }

            } else {

                $error = "Unregistred user !!! Please sign-in first..." ;

            }

            return array(

                "view" => 'viewAddTopic.php',

                "data" => $error

            );

        }

        public function updateAction($tid) {

            $\_SESSION['topic'] = $this->\_topic->getTopic($tid) ;

            return array(

                "view" => 'viewUpdateTopic.php',

                "data" => null

            );

        }

        // Méthode appelée en validation du formulaire de modification d'un sujet existant

        public function doUpdateAction($tid) {

            $title = filter\_input(INPUT\_POST, 'title', FILTER\_SANITIZE\_STRING, FILTER\_FLAG\_STRIP\_LOW) ;

            $descr = filter\_input(INPUT\_POST, 'descr', FILTER\_SANITIZE\_STRING) ;

            $status = $this->\_topic->updateTopic($tid, $title, $descr) ;

            if ($status) {

                // Si ajout OK, on redirige vers la page d'accueil

                header("Location:index.php?ctrl=Topic") ;

            } else {

                // Sinon, on affiche le message d'erreur

                $error = "Unknown error !!! Please try again..." ;

            }

            $\_SESSION['topic'] = $this->\_topic->getTopic($tid) ;

            return array(

                "view" => 'viewUpdateTopic.php',

                "data" => $error

            );

        }

        public function deleteAction($tid) {

            $status = $this->\_topic->deleteTopic($tid) ;

            header("Location:index.php?ctrl=Topic") ;

        }

    }

?>

### Les vues associées aux sujets

On peut voir dans le contrôleur précédent que trois vues seront manipulées pour les sujets :

* Une vue « **viewTopics.php** », qui sera chargée d’afficher la liste de l’ensemble des sujets, avec les informations associées : titre, auteur, description, date de création et nombre de messages actuellement associés à ce sujet ;
* Une vue « **viewAddTopic.php** », qui est un formulaire permettant d’ajouter un nouveau sujet ;
* Une vue « **viewUpdateTopic.php** », qui est un formulaire permettant de modifier un sujet existant.

Chaque vue est uniquement responsable de l’affichage d’un contenu, et peut fournir des liens permettant d’enchaîner d’autres actions.

Ci-après, la vue « **viewTopics.php** » :

<h1>Topic's List</h1>

<?php

    $topics=$result['data'] ;

    if (is\_array($topics) && (count($topics) > 0)) {

        // On vérifie si on est en mode anonyme ou connecté (et on mémorise le pseudo en conséquence)

        if (isset($\_SESSION['user']))

            $pseudo = $\_SESSION['user']['pseudo'] ;

        else

            $pseudo = null ;

?>

    <div>

        <table>

            <thead>

                <tr>

                    <th class="center" colspan="2">Subject</th>

                    <th class="center">Author</th>

                    <th class="center">Description</th>

                    <th class="center">Created at</th>

                    <th class="center"># Posts</th>

                    <th class="center">Actions</th>

                </tr>

            </thead>

            <tbody>

<?php

            foreach($topics as $topic) {

?>

        <!-- Si le résultat est un tableau, on génère une ligne par sujet retourné -->

                <tr>

                    <td class="center nb-right">

<?php

                    if ($topic['pseudo'] == $pseudo) {

?>

                        <a href="index.php?ctrl=Topic&action=toggle&id=<?=$topic['id\_topic']?>">

<?php

                    }

?>

                        <i class="fas fa-<?=($topic['is\_closed'] != '0') ? 'lock' : 'lock-open'?>"></i>

<?php

                    if ($topic['pseudo'] == $pseudo) {

?>

                        </a>

<?php

                    }

?>

                    </td>

                    <td class="left nb-left">  
                        <a href="index.php?ctrl=Post&action=list&id=<?=$topic['id\_topic']?>">

                        <?=$topic['title']?>

                        </a>

                    </td>

                    <td class="center"><?=ucwords($topic['pseudo'], "-\_\t\r\n\f\v")?></td>

                    <td class="left">  
                        <a href="index.php?ctrl=Post&action=list&id=<?=$topic['id\_topic']?>">  
                        <?=strlen($topic['description'])>64?substr($topic['description'],0,61)."...":$topic['description']?>  
                        </a>  
                    </td>

                    <td class="left"><?=date\_format(new DateTime($topic['created\_at']), "d/m/Y H:i")?></td>

                    <td class="center"><?=$topic['nb\_posts']?></td>

                    <td class="center">

<?php

                    if (($topic['is\_closed'] == '0') && ($topic['pseudo'] == $pseudo)) {

?>

                        <a href="index.php?ctrl=Topic&action=update&id=<?=$topic['id\_topic']?>">  
                        <i class="fas fa-pen-square"></i>  
                        </a>

<?php

                    }

?>

                        &nbsp;

<?php

                    if (($topic['is\_closed'] == '0') && ($topic['pseudo'] == $pseudo)) {

                        if ($topic['nb\_posts'] == 0) {

?>

                        <a href="index.php?ctrl=Topic&action=delete&id=<?=$topic['id\_topic']?>" class="del-topic">

<?php

                        }

?>

                        <i class="fas fa-trash-alt"></i>

<?php

                        if ($topic['nb\_posts'] == 0) {

?>

                        </a>

<?php

                        }

                    }

?>

                    </td>

                </tr>

<?php

            }

?>

            </tbody>

        </table>

    </div>

<?php

    } else {

?>

        <!-- Sinon, on génère un simple message -->

        <p><strong>No topics yet !</strong></p>

<?php

    }

    if(isset($\_SESSION['user'])) {

?>

        <a class="button" href="index.php?ctrl=Topic&action=new">Add New Topic</a>

<?php

    }

?>

# Résultat obtenu (extrait)

Les écrans suivants présentent les principales pages du site.

## Page principale (en mode ‘anonyme’)



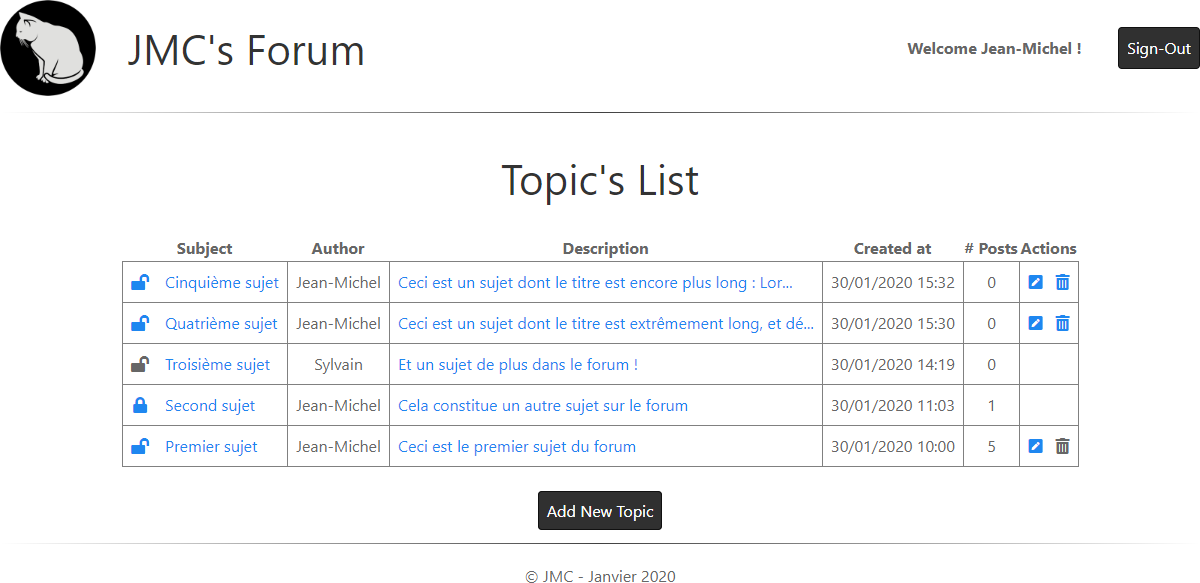
## Connexion utilisateur existant



## Enregistrement nouvel utilisateur



## Page principale (en mode ‘authentifié’)



## Ajout d’un nouveau sujet

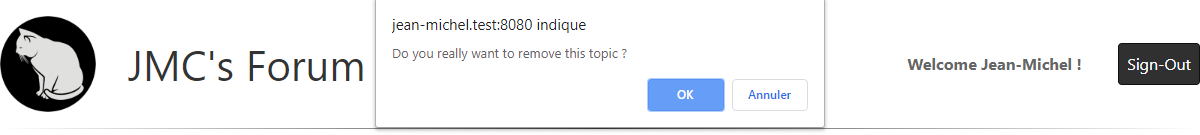


## Modification d’un sujet existant



## Suppression d’un sujet existant

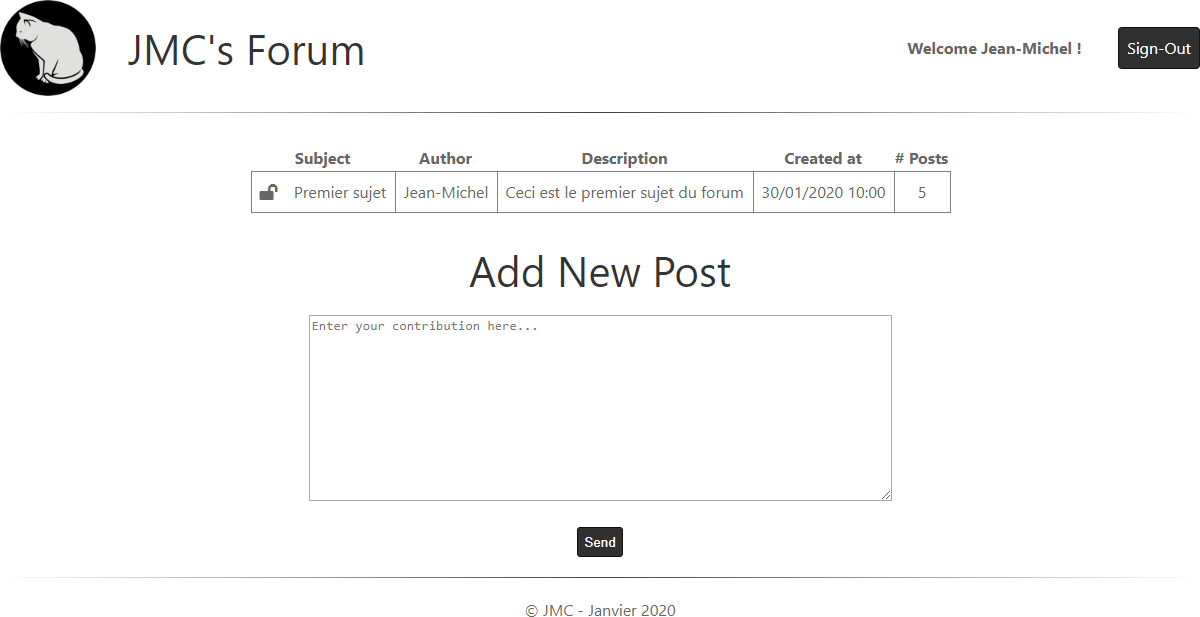
Il n’y a pas de vue de suppression d’un sujet, mais uniquement l’apparition d’une fenêtre demandant confirmation de l’action :



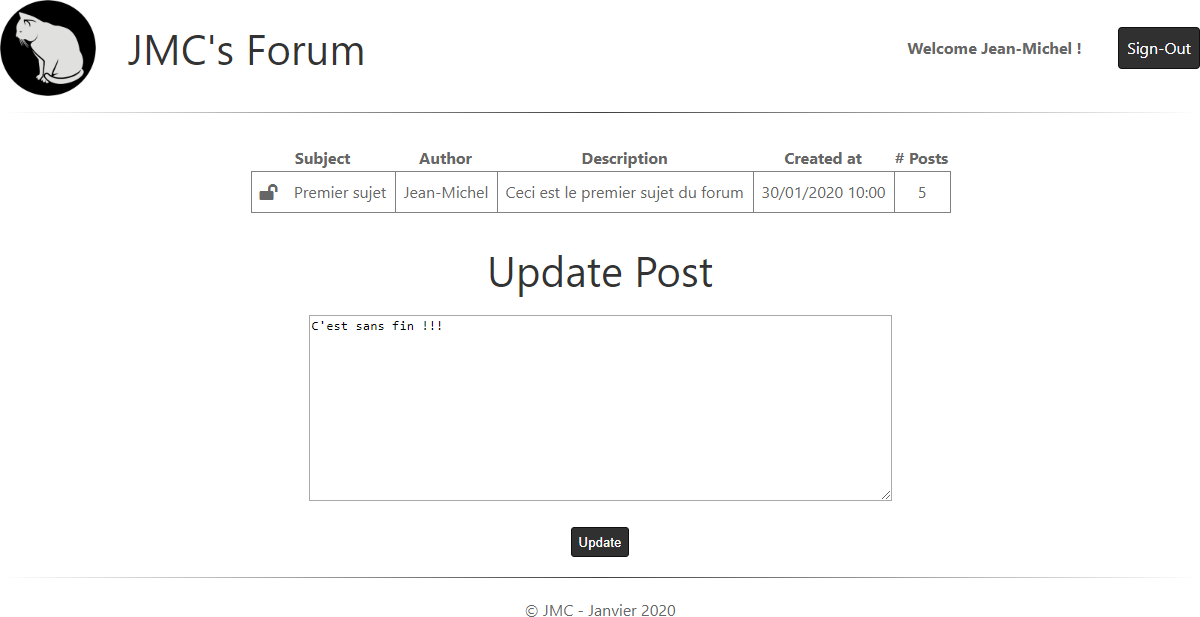
## Détails d’un sujet (⬄ Liste des messages sous ce sujet)



## Ajout d’un message (sous un sujet)

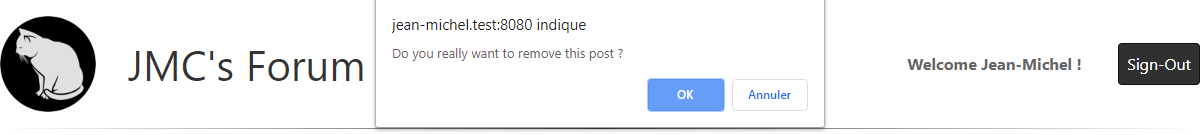


## Modification d’un message (sous un sujet)



## Suppression d’un message (sous un sujet)

Il n’y a pas de vue de suppression d’un sujet, mais uniquement l’apparition d’une fenêtre demandant confirmation de l’action :



## Fonctionnalité non encore implémentée

Le lien « Forgot password ? » présent dans la fenêtre de connexion n’est à l’heure actuelle pas réactif.

À terme, il devra permettre d’accéder à une vue demandant à l’utilisateur de saisir son pseudo et l’adresse mail qu’il a associée à son compte (pour vérification), puis provoquer l’envoi d’un mail à cette adresse avec un lien vers une autre vue permettant de saisir un nouveau mot de passe (avec double saisie).