

Projet NFE114
Juin 2016

Activons-nous

Réseau social de partage d'activités

Auteurs : Thomas Swank
Chrislin Mpika

Table des matières

	Numéro de page
Présentation du projet	3
Description de la méthode UWE	4
Description générale de l'application	6
Développement de cas d'utilisation	
Ajouter un commentaire	12
Lire un message privé	22
Diagrammes UWE	26
Informations complémentaires sur le site	35
Références	36

1. Présentation du projet

1.1 Contexte

L'objectif initial du projet est de mettre en place une plate-forme qui permet à des utilisateurs de proposer ou de participer à des activités.

Certains sites de ce type existent déjà, il s'agit donc de proposer des fonctionnalités supplémentaires ainsi qu'une bonne ergonomie.

La recherche d'activité doit être aussi simple que possible. Pour cela, nous utiliserons un système de filtres et de tri permettant de sélectionner certaines catégories ou villes par exemple.

Il faut également que le site favorise l'interactivité via des systèmes de commentaires, de messagerie, de groupes, d'utilisateurs favoris ou d'appréciation des activités auxquelles ils participent.

1.2 Règles d'accès

On distingue trois catégories d'utilisateurs :

- Utilisateur non connecté
- Utilisateur connecté (membre)
- Administrateur

1.3 Choix méthodologiques

Méthode UWE : ArgoUWE, ArgoUML(v. 0.34) et Inkscape

1.4 Choix techniques

Langages serveur : PHP (5.5+), MySQL

Client : HTML/CSS, JavaScript (jQuery)

Template : Twig

Nous souhaitons mettre en place une structure de type MVC, afin de bien séparer la gestion des données, les vues et la distribution des traitements.

L'utilisation d'un framework (Zend, Symfony, etc.) aurait été un choix logique, mais la prise en main d'un tel système peut être assez longue.

C'est pourquoi, nous décidons de créer une architecture personnalisée.

Afin de gérer l'import des données dans les vues, nous choisissons d'utiliser un générateur de template.

Des recherches permettent d'identifier Twig comme étant un bon candidat :

- Nombreuses fonctionnalités intéressantes
- Bien documenté
- Rapide

2. Description de la Méthode UWE

2.1 Récapitulatif

La méthode UWE s'appuie sur le langage UML pour décrire une application web orientée objet. Elle permet d'établir les spécifications de l'application à partir d'une série de diagrammes et de descriptions textuelles.

Nous utiliserons les diagrammes suivants :

2.1.1 Diagramme des Cas d'utilisation (UML)

Établit la liste des fonctionnalités pour chaque acteurs.

2.1.2 Description textuelle d'un cas d'utilisation (UML)

Consiste à développer sous forme de scénarios l'ensemble des interactions (entre l'acteur et le système) impliquées dans un cas d'utilisation.

Un même cas d'utilisation peut comporter plusieurs scénarios :

- Un scénario nominal : correspond à un déroulement normal
- Éventuellement, un ou plusieurs scénario(s) alternatif(s) : décrit une variation du scénario nominal lorsqu'une condition spécifique est remplie.

2.1.3 Diagramme d'activités (UML)

Représente graphiquement l'enchaînement des activités impliquées dans un cas d'utilisation. Permet de décrire l'ensemble des scénarios (nominal et alternatifs) dans un seul diagramme.

2.1.4 Diagramme de séquences (UML)

Précise les interactions entre l'acteur et les objets du système impliqués d'un point de vue chronologique dans un scénario.

On retrouve notamment les messages échangés ainsi que la durée de vie des objets.

2.1.5 Diagramme de classes (UML)

Le diagramme de classes donne une représentation de la structure interne du système et des échanges entre objets, permettant de répondre aux cas d'utilisation.

2.1.6 Diagramme de contenu (UWE)

C'est un sous-ensemble du diagramme des classes. Il représente les classes entités pertinentes au modèle de navigation.

2.1.7 Modèle de navigation (UWE)

Indique les chemins de navigation entre les classes du diagramme de contenu. Ajoute les classes frontières.

2.1.8 Modèle d'accès (UWE)

Précise le diagramme de navigation avec les structures d'accès (menu, index, etc.) entre les éléments.

2.1.9 Diagramme de présentation (UWE)

Décrit la structure des éléments du modèle d'accès. Il correspond à une base visuelle pour la mise en forme des pages de l'application.

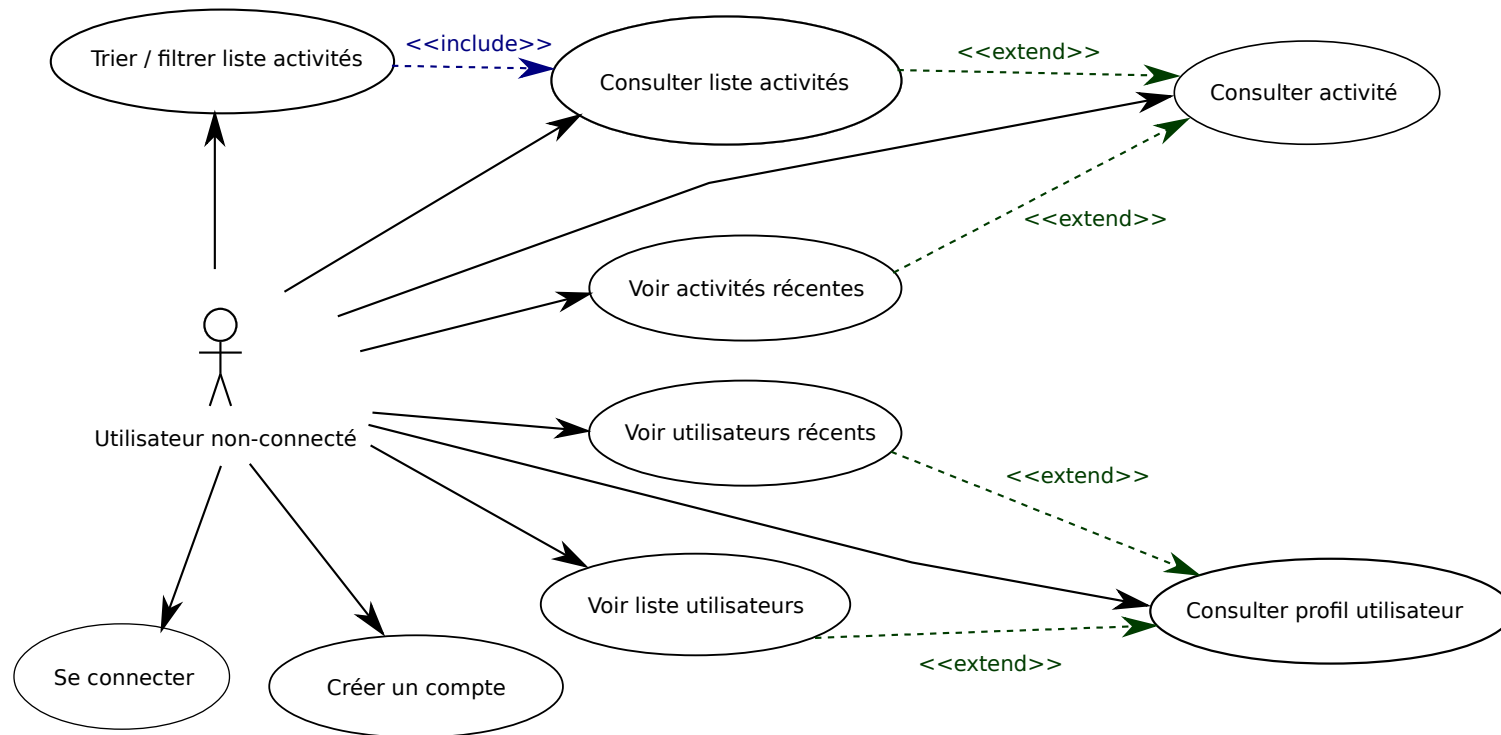
Nous appliquerons la méthode de la manière suivante :

- Présentation générale des cas d'utilisation et des classes
- Déroulement de plusieurs cas d'utilisation jusqu'au diagramme de contenu correspondant
- Rassemblement des diagrammes de contenu en un seul et application des étapes de la méthode UWE pour décrire la navigation, les accès et la présentation des pages.

3. Description générale de l'application

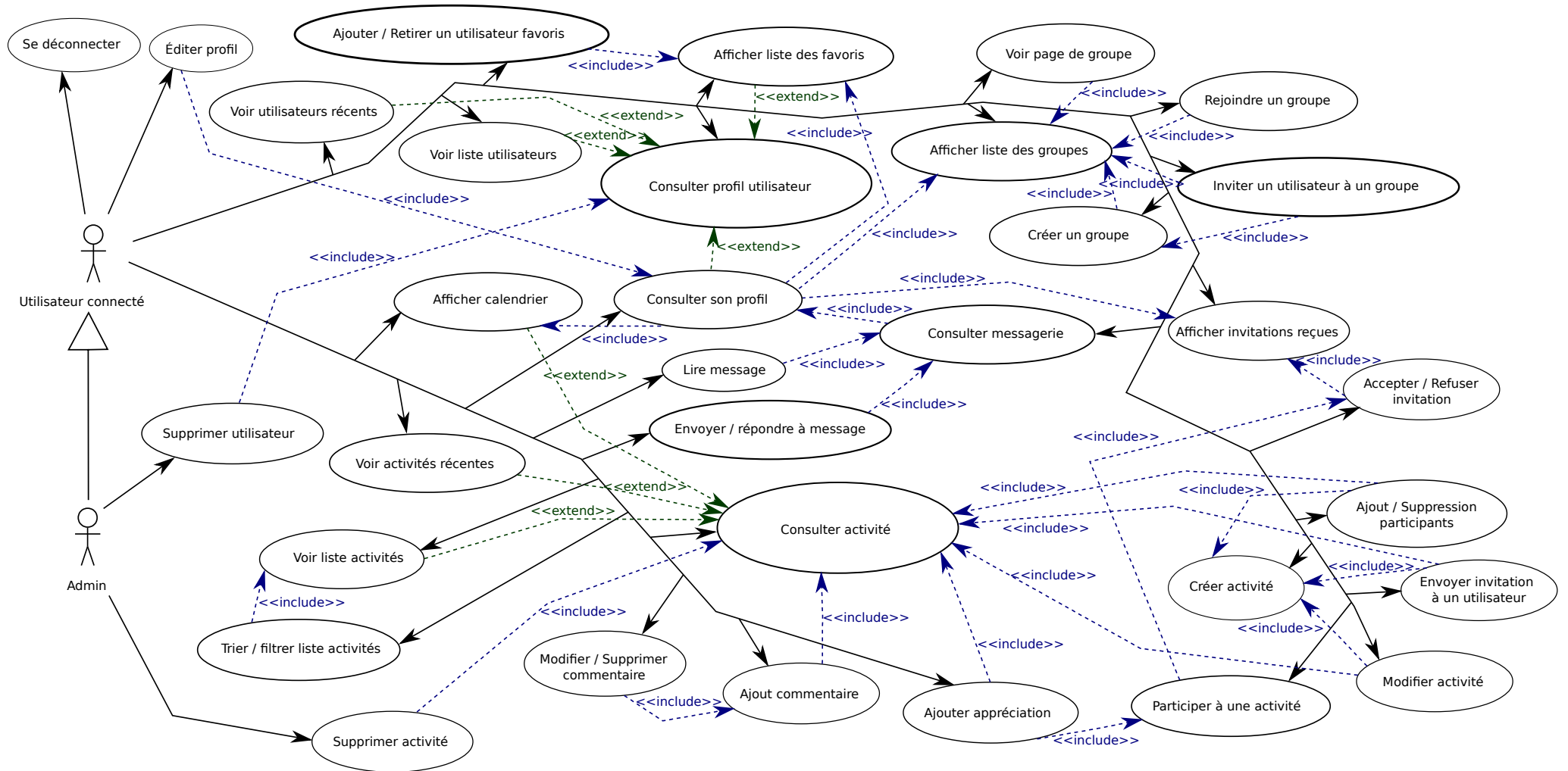
3.1 Diagramme des cas d'utilisation

3.1.1 Utilisateur non connecté



3.1.2 Utilisateur connecté et Admin

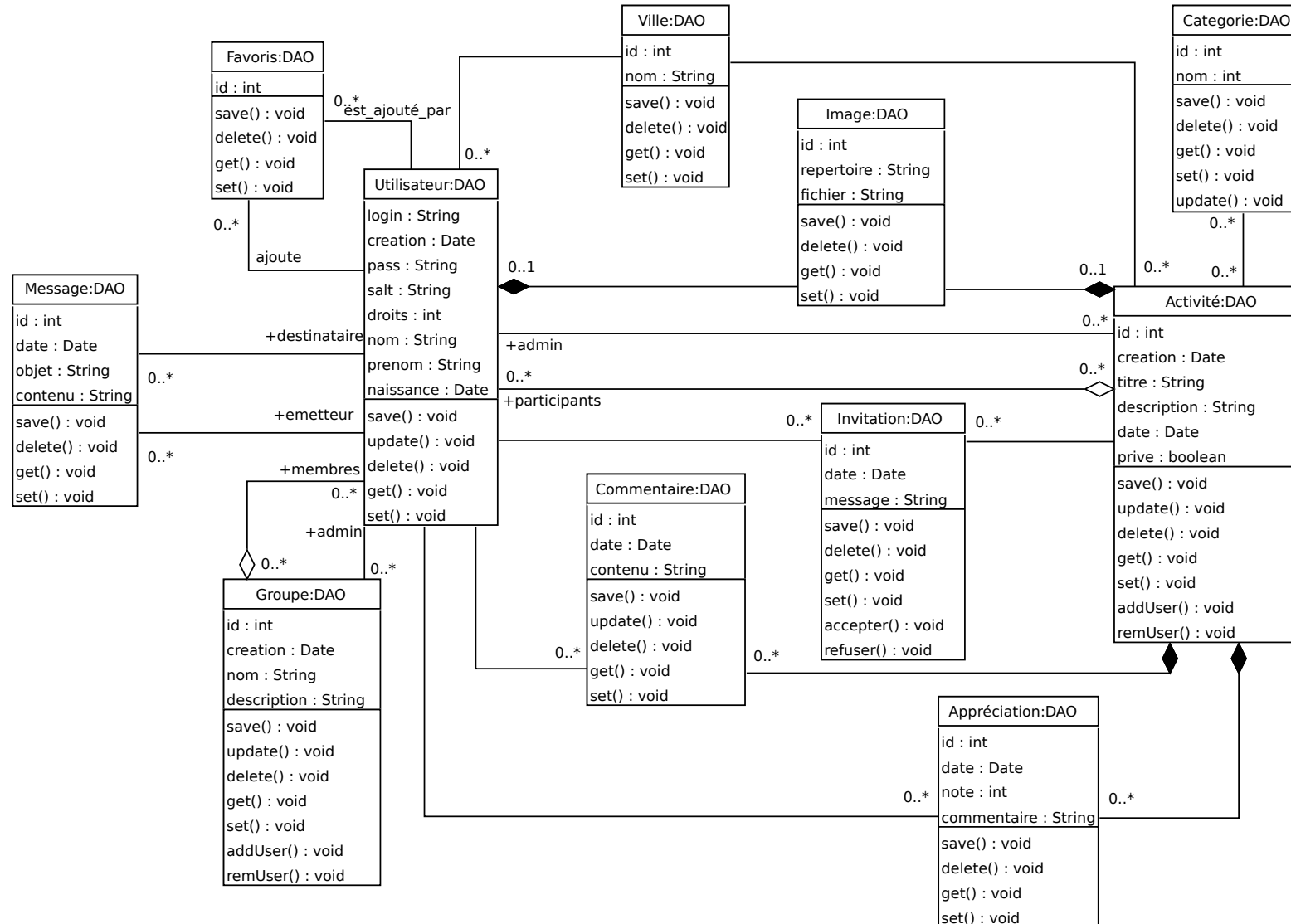
Note : L'utilisateur passe de non-connecté à connecté (ou admin) après avoir réalisé le cas d'utilisation « Se connecter »



3.2 Diagramme de classes

3.2.1 Classes entités

<<Entity>> : Classe représentant des données du système. Information persistante et traitements liés à cette information.



3.2.2 Classes frontières

<<Boundary>> : Classe interface entre le système et l'utilisateur. Gère l'échange d'information entre les deux.

Groupe:Affichage
id : int
creation : Date
nom : String
description : String
/admin : Utilisateur
/listeMembres : Utilisateur
Affichage() : void
AffichageFormulaire() : void

? Messagerie
/listeMessagesReçus : Message
/listeMessagesEnvoyés : Message
Affichage() : void

Message:Affichage
id : int
date : Date
objet : String
contenu : String
/emetteur : Utilisateur
/destinataire : Utilisateur
Affichage() : void
AffichageFormulaire() : void

Utilisateur:Affichage
login : String
creation : Date
pass : String
salt : String
droits : int
nom : String
prenom : String
naissance : Date
/portrait : Image
/ville : Ville
/listeInvitations : Invitation
/listeGroupes : Groupe
/listeFavoris : Favoris
/listeActivités : Activité
Affichage() : void
AffichageFormulaire() : void

? UtilisateurList
/listUtilisateurs : Utilisateur
Affichage() : void

? Accueil
/activitésRécentes : Activité
/utilisateursRécents : Utilisateur
Affichage() : void

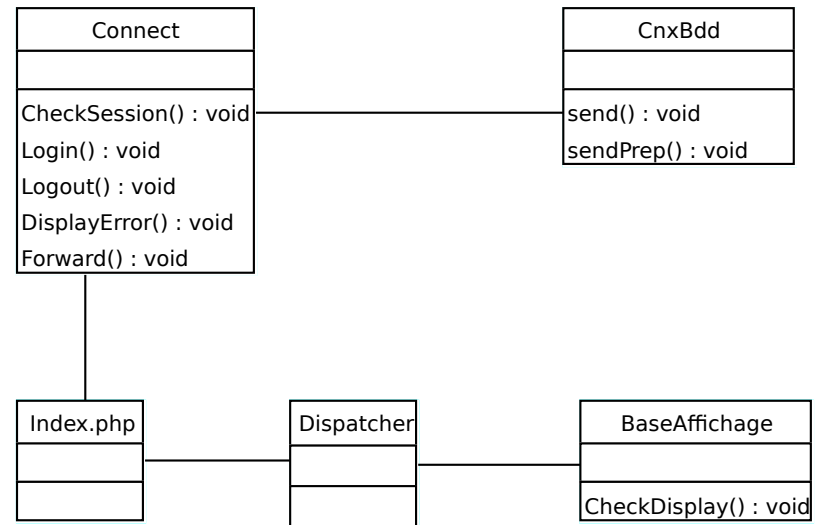
? APropos
Affichage() : void

Activité:Affichage
id : int
creation : Date
titre : String
description : String
date : Date
prive : boolean
/image : Image
/ville : Ville
/listeParticipants : Utilisateur
/listeCatégories : Catégorie
/listeCommentaires : Commentaire
/listeAppréciations : Appréciation
/admin : Utilisateur
Affichage() : void
AffichageFormulaire() : void

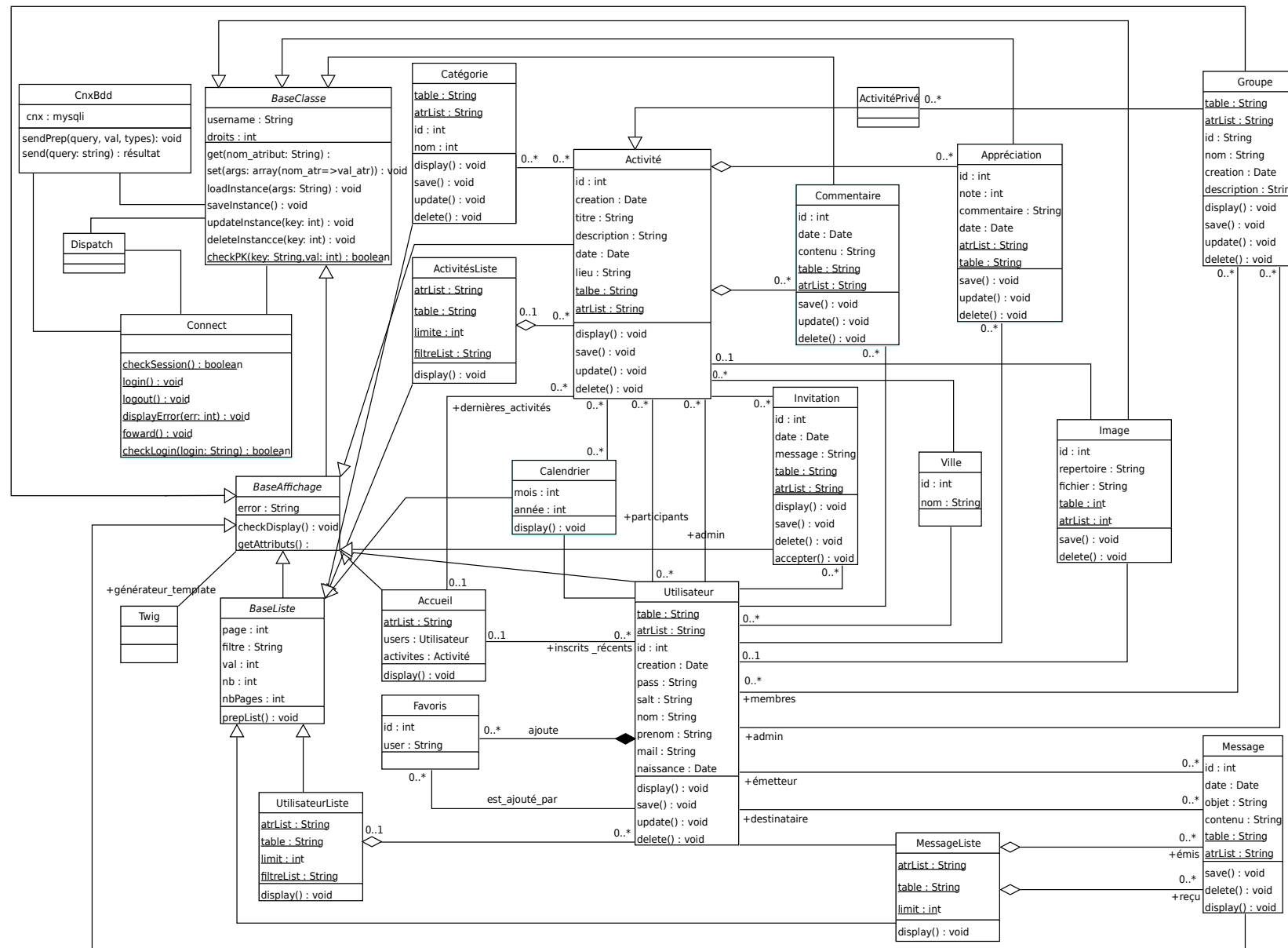
? ActiviteList
/listeActivités : Activité
/listeCatégories : Catégorie
/listeVilles : Ville
Affichage() : void

3.2.3 Classes contrôles

<<Control>> : Fait la jonction entre les deux autres types de classes. Permet de dérouler un cas d'utilisation en distribuant les traitements aux classes appropriées.



3.2.4 Diagramme de classes général

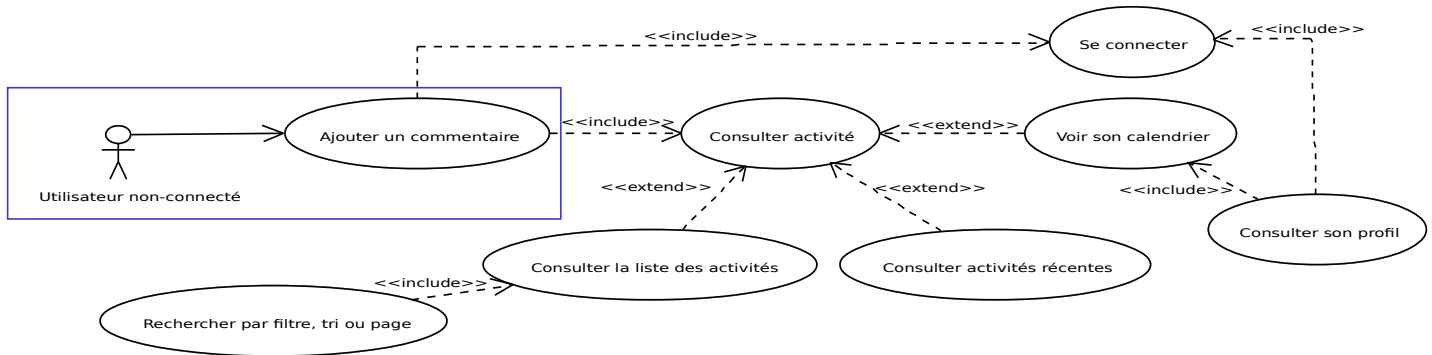


Notes : Ce dernier diagramme représente l'organisation de notre application, au niveau du découpage des traitements et de l'information persistante.

4. Développement de cas d'utilisation

4.1 Cas d'utilisation : Ajouter un commentaire

4.1.1 Diagramme de cas d'utilisation



4.1.2 Description textuelle

Scénario nominal :

1. L'utilisateur affiche le bloc de connexion, saisit ses identifiants et valide
2. Le système vérifie les identifiants et lance la session de l'utilisateur
3. L'utilisateur clique sur la page activités
4. Le système affiche la liste des activités
5. L'utilisateur clique sur l'activité de son choix
6. Le système affiche la page de l'activité en question
7. L'utilisateur se place en bas de la page, remplit le formulaire d'ajout de commentaire et confirme
8. Le système enregistre le commentaire et affiche la page de l'activité avec le nouveau commentaire

Scénario alternatif 1 : *L'utilisateur ne dispose pas d'un compte*

(Remplace l'étape 1 du scénario nominal)

1. L'utilisateur clique sur Créer un compte
2. Le système affiche le formulaire de création de compte
3. L'utilisateur remplit le formulaire et le soumet
4. Le système enregistre le compte
5. Reprend l'étape 1 du scénario nominal

Scénario alternatif 2 : *Erreur de saisie dans le formulaire création de compte*

(Remplace l'étape 4 du scénario alternatif 1)

4. Le système détecte une erreur, la notifie à l'utilisateur et renvoie le formulaire
5. Reprend l'étape 3 du scénario alternatif 1

Scénario alternatif 3 : *Erreur de saisie de login ou mot de passe*

(Remplace l'étape 2 du scénario nominal)

2. Le système détecte une erreur de login ou mot de passe et notifie l'utilisateur
3. Reprend l'étape 1 du scénario nominal

Scénario alternatif 4 : *Activité en page d'accueil*

(Remplace l'étape 3 du scénario nominal)

3. L'activité est présente en page d'accueil, l'utilisateur clique dessus
4. Reprend l'étape 6 du scénario nominal

Scénario alternatif 5 : *Activité sur le calendrier de l'utilisateur*

(Remplace l'étape 3 du scénario nominal)

3. L'utilisateur clique sur son pseudonyme
4. Le système affiche le compte de l'utilisateur
5. L'utilisateur clique sur l'activité dans son calendrier
6. Reprend l'étape 6 du scénario nominal

Scénario alternatif 6 : *Activité sur une autre page du calendrier (mois/année différente)*

(Remplace l'étape 5 du scénario alternatif 5)

5. L'utilisateur sélectionne un mois et une année dans le calendrier
6. Le système affiche le compte de l'utilisateur avec le calendrier correspondant
7. Reprend l'étape 5 du scénario alternatif 5

Scénario alternatif 7 : *Activité non présente sur la première page de la liste des activités*

(Remplace l'étape 5 du scénario nominal)

5. L'utilisateur utilise les fonctionnalités de filtre, tri et numéro de pages pour trouver l'activité qui l'intéresse
6. Le système renvoi la liste des activités correspondantes
7. Reprend l'étape 5 du scénario nominal

Scénario alternatif 8 : *Erreur de chargement de la page*

(Remplace l'étape 6 du scénario nominal)

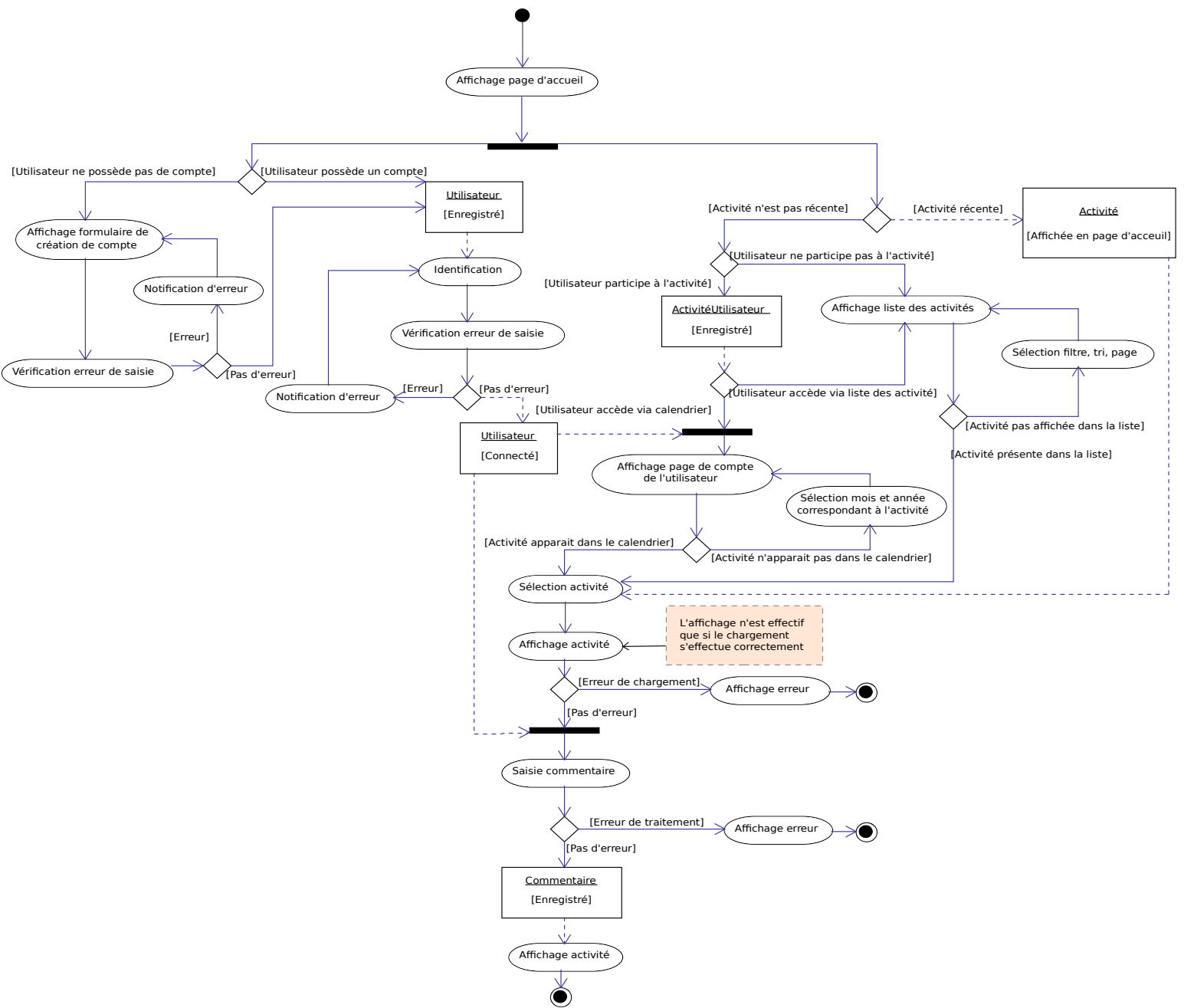
6. Erreur de chargement de la page correspondante, le système affiche une erreur

Scénario alternatif 9 : *Erreur lors de l'enregistrement du commentaire*

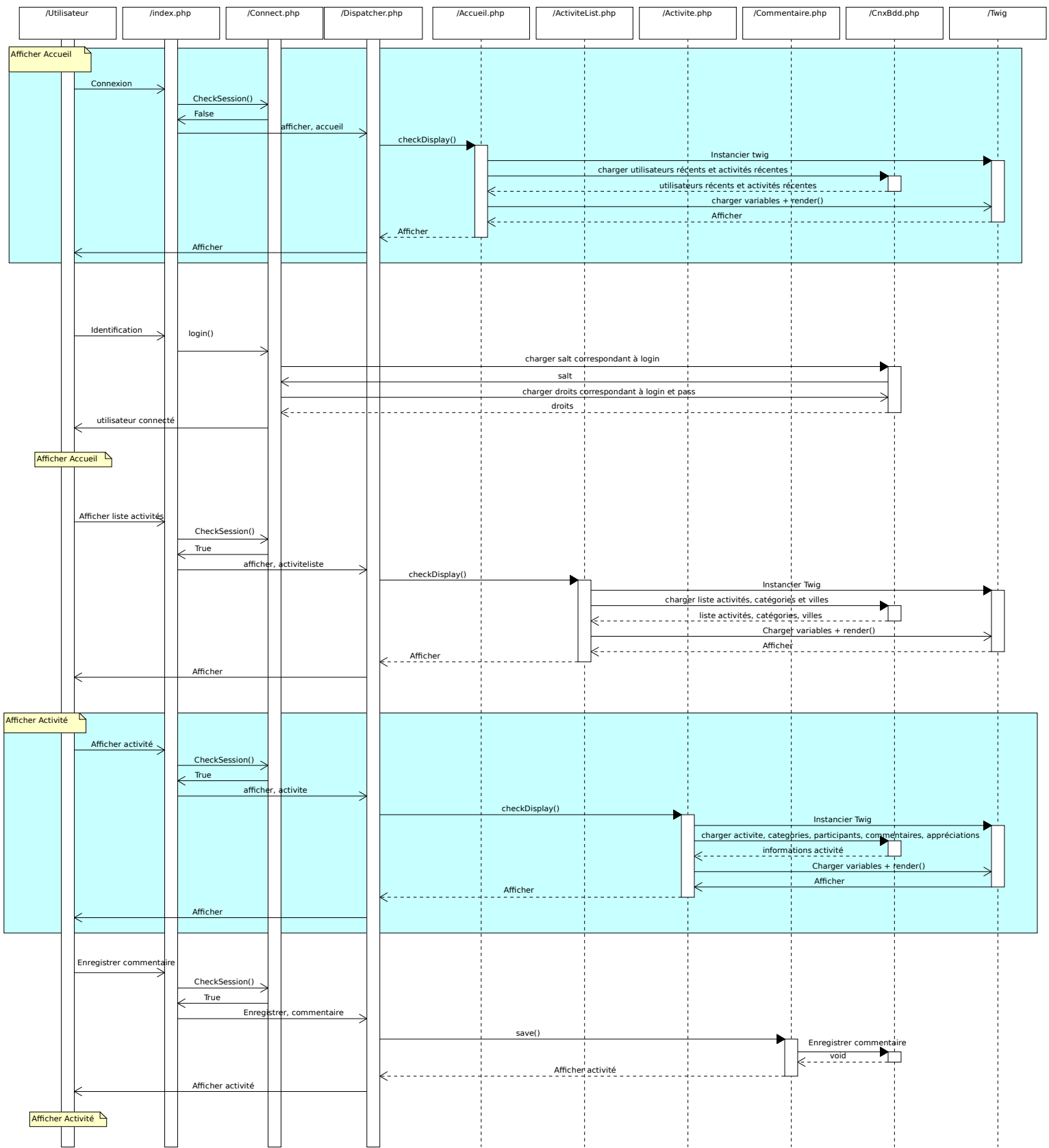
(Remplace l'étape 8 du scénario nominal)

8. Une erreur se produit, le commentaire n'est pas enregistré et le système prévient l'utilisateur

4.1.3 Diagramme d'activité



4.1.4 Diagramme de séquence



Ce diagramme décrit le déroulement du scénario nominal.

Note : Les encadrés et étiquettes servent à indiquer des répétitions de séquences de traitements. Les séquences encadrées sont répétées au niveau de l'étiquette correspondante.

Note2 : Les échanges internes aux classes ne sont pas représentés dans le diagramme. On aurait pu le faire, mais cela en aurait considérablement augmenté la taille.

Nous avons choisi d'utiliser une structure de type MVC pour notre application, il est donc important de préciser le rôle de certains objets.

Contrôleur : Index.php et Dispatcher.php. ils se chargent principalement de récupérer la requête de l'utilisateur et de créer l'objet correspondant du modèle qui se chargera de la traiter.

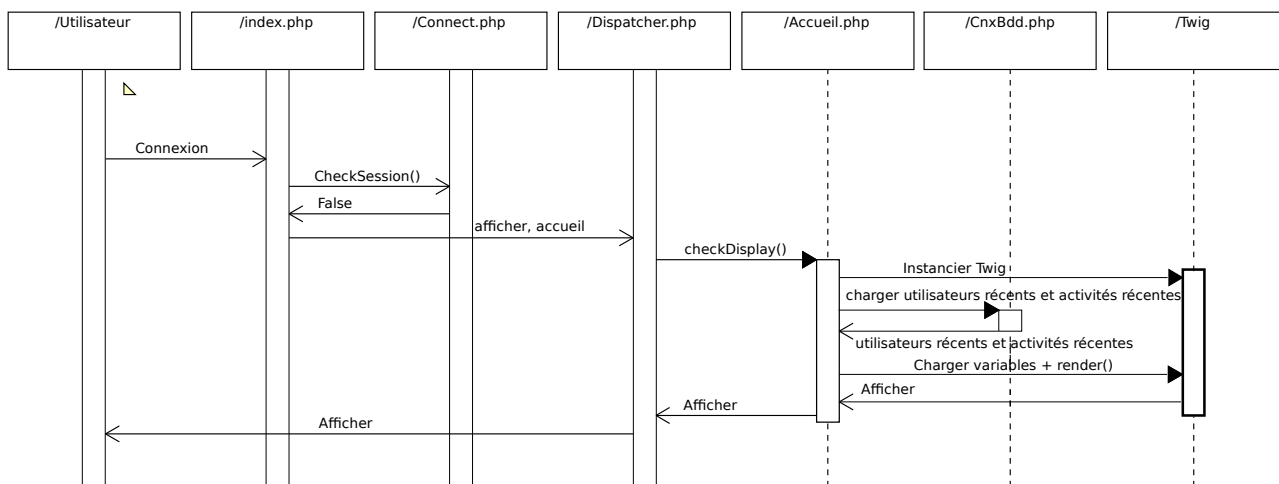
Modèle : Ici, nous avons divisé le modèle en deux catégories de classes :

- Accueil.php, Activite.php, ActiviteList.php, Commentaire.php correspondent à des entités (Données et/ou vues).
- Connect.php, CnxBdd.php sont des classes « transversales », permettant de gérer certaines fonctionnalités communes aux autres classes. L'intérêt de cette démarche est de centraliser la gestion de ces fonctionnalités.

Vue : La classe Twig, correspondant au générateur de template, peut être associée à la vue.

Comparaison de certaines séquences avec le code de l'application :

A. Affichage de la page d'accueil :



Voici le code qui concerne les opérations décrites :

index.php

```
session_start();
```

```
if (Connect::checkSession()) { // Vérification connexion
    $username = $_SESSION['login'];
    $droits = $_SESSION['droits'];
    session_regenerate_id(true);
} else { // L'utilisateur n'est pas connecté
    $username = null;
    $droits = 2;
}
```


classes/Connect.php

```
public static function checkSession() {
    if (isset($_SESSION) && array_key_exists('login', $_SESSION) && array_key_exists('droits', $_SESSION))
        return true;
    return false; // Utilisateur non-connecté
}
```

classes/Dispatcher.php

```
if (isset($_GET)) {
    $display = (array_key_exists('display', $_GET)) ? $_GET['display'] : '';
    $action = (array_key_exists('action', $_GET)) ? $_GET['action'] : '';
} else {
    $display = '';
    $action = '';
}
```

// Choix classe à instancier : ici, Accueil

// Note : Les classes ne sont pas encore toutes implémentées

```
switch ($display) {
    case 'activite' :
        include_once 'Activite.php';
        $page = new Activite();
        break;
    case 'commentaire' :
        include_once 'Commentaire.php';
        $page = new Commentaire();
        break;
    case 'activites' :
        include_once 'ActiviteList.php';
        $page = new ActiviteList();
        break;
    default :
        include_once 'Accueil.php';
        $page = new Accueil();
}
```

// Ajoute nom d'utilisateur et droits aux variables d'instance

\$page -> set(array('username'=>\$username, 'droits'=>\$droits));

// Sélection méthode à utiliser : ici, checkDisplay()

```
switch ($action) {
    case 'enregistrer' :
        $page -> save();
        break;
    case 'modifier' :
        $page -> update();
        break;
    case 'supprimer' :
        $page -> delete();
        break;
    default :
        $page -> checkDisplay();
}
```

classes/BaseAffichage.php

```
public function checkDisplay() {
    // Instanciation twig
    require_once __DIR__.'../twig/lib/Twig/Autoloader.php';
    Twig_Autoloader::register();
    $loader = new Twig_Loader_Filesystem(__DIR__.'../vues');
    $this->twig = new Twig_Environment($loader, array('cache' => false));
    // Vérification renvoi d'erreur (si traitement précédent a échoué)
    if ($_GET && array_key_exists('error', $_GET) && $_GET['error'] >= '0' && $_GET['error'] < 3) {
        $msgError = array('Identifiants incorrects', 'Page introuvable', 'Erreur de traitement');
        $this->error = $msgError[$_GET['error']];
    }
    if (array_key_exists('list', $this::$atrList)) // Page liste
        return $this->display();
    [...]
}
```

/* Les pages de listes sont celles qui n'affichent pas un objet spécifique de la classe (une activité ou un utilisateur par exemple). C'est le cas ici */

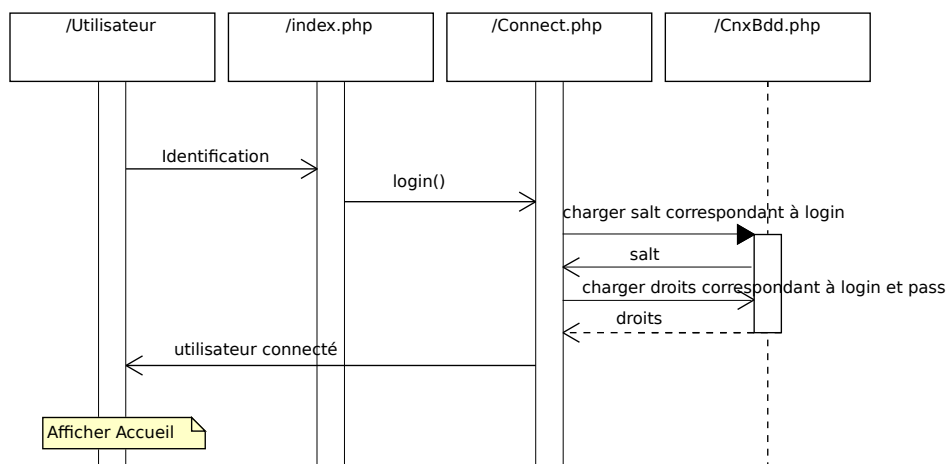
classes/Accueil.php

```
public function display() {  
    // Récupération activités et utilisateurs récents  
    $cnx = new CnxBdd();  
    $this->users = $cnx->send('  
        SELECT U.login, V.nom AS ville, (DATEDIFF(CURRENT_DATE, U.naissance) DIV 365.25) AS age, I.repertoire,  
        I.fichier FROM ville V, utilisateur U LEFT JOIN image I ON U.portrait=I.id WHERE U.ville=V.id ORDER BY  
        U.creation DESC LIMIT 5');  
    $this->activites = $cnx->send('  
        SELECT A.id, A.titre, DATE_FORMAT(A.date, "Le %e/%c/%Y à %kh%i") AS date, A.user, V.nom AS ville FROM  
        activite A, ville V WHERE A.ville=V.id ORDER BY A.id DESC LIMIT 5');  
    // Chargement variables + render()  
    echo $this->twig->render('accueil.html', $this->getAttributs());  
}
```

classes/CnxBdd.php

```
// Envoie requête select et retourne les entrées correspondantes  
public function send ($query) {  
    $result = $this->cnx->query($query);  
    if (!$result)  
        Connect::displayError(2);  
    else {  
        $ret = array();  
        if ($result -> num_rows > 0) {  
            while ($line = $result -> fetch_assoc())  
                array_push($ret, $line);  
        }  
        return $ret;  
    }  
}
```

B. Identification



Le code :

index.php

```
session_start();  
  
if (isset($_GET) && array_key_exists('action', $_GET)) {  
    if ($_GET['action'] == 'logout')  
        Connect::logout();  
    else if ($_GET['action'] == 'login') // Requête connexion
```

```

    Connect::login();
}

```

classes/Connect.php

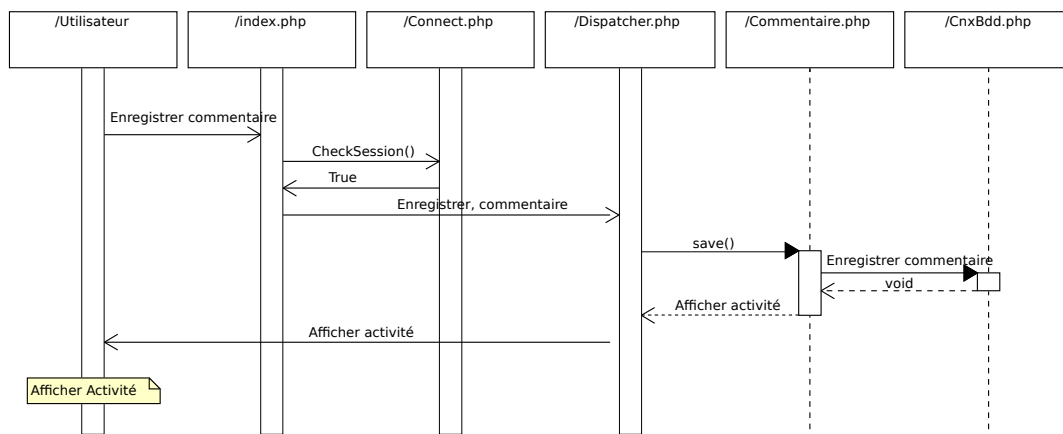
```

public static function login () {
    // vérification paramètres $_POST + format login correct
    if (!isset($_POST) || !array_key_exists('login', $_POST) || !self::checkLogin($_POST['login']) || !array_key_exists('pass',
                                                                    $_POST))

    self::displayError(0);
    $login = $_POST['login'];
    $cnx = new CnxBdd();
    // Récupérer le salt
    $query = "SELECT salt FROM utilisateur WHERE login='$login'";
    $res = $cnx->send($query);
    if (count($res) != 1) // Entrée non trouvée : login n'existe pas ou erreur
        self::displayError(0);
    $pass = hash('sha256', $res[0]['salt'].$_POST['pass']);
    // Vérification couple login / pass
    $query = "SELECT droits FROM utilisateur WHERE login='$login' AND pass='$pass'";
    $res = $cnx->send($query);
    if (count($res) != 1) // couple login / pass incorrect
        self::displayError(0);
    $droits = $res[0]['droits'];
    setcookie(session_id(), NULL, 60*60);
    $_SESSION['login'] = $login;
    $_SESSION['droits'] = $droits;
    self::forward(); // Utilisateur connecté + renvoi à la page précédente : affichage accueil
}

```

C. Enregistrement commentaire



Le code :

index.php, classes/Connect.php, classes/Dispatcher.php

// Le déroulement de ces étapes est le même que pour la séquence d'affichage de la page d'accueil

classes/Commentaire.php

```

public function save() {
    // Vérification droits + variables POST ok
    if ($this->droits > 1 || !$_POST || !array_key_exists('contenu', $_POST) || !array_key_exists('activite', $_POST))
        Connect::displayError(2);
    /* Remplace les sauts de ligne textarea par <br> et ajoute backslash sur guillemets et backslash pour éviter erreur
    d'affichage */
    $contenu = nl2br(htmlspecialchars($_POST['contenu'], ENT_QUOTES, "UTF-8"));
}

```

```

$this->set(array('contenu'=>$contenu, 'user'=>$this->username, 'activite'=>$_POST['activite']));
$this->saveInstance();
Connect::forward(); // enregistrement effectué, chargement de la page précédente (activité)
}

```

classes/BaseClasse.php

```

public function set ($args) {
    foreach ($args as $k => $v) {
        $this->$k = $v;
    }
}

protected function saveInstance () {
    $query = 'INSERT INTO '.$this::$table.' ('; // Requête
    $val = array(); // array(valeur_attributs)
    $types = ''; // Chaîne des types d'attributs
    $finQuery = ''; // Fin de la requête (remplace valeur attributs par ?)
    foreach ($this::$atrList as $k => $v) {
        if ($this->$k != null) {
            $query .= $k.',';
            array_push($val, $this->$k);
            $finQuery .= '?,';
            $types .= ($v == 'int')?'i':'s';
        }
    }
    $query = substr($query, 0, -2).' VALUES ('.substr($finQuery, 0, -2).')';
    $cnx = new CnxBdd();
    $cnx->sendPrep($query, $val, $types); // Envoi de la requête
    unset($cnx);
}

```

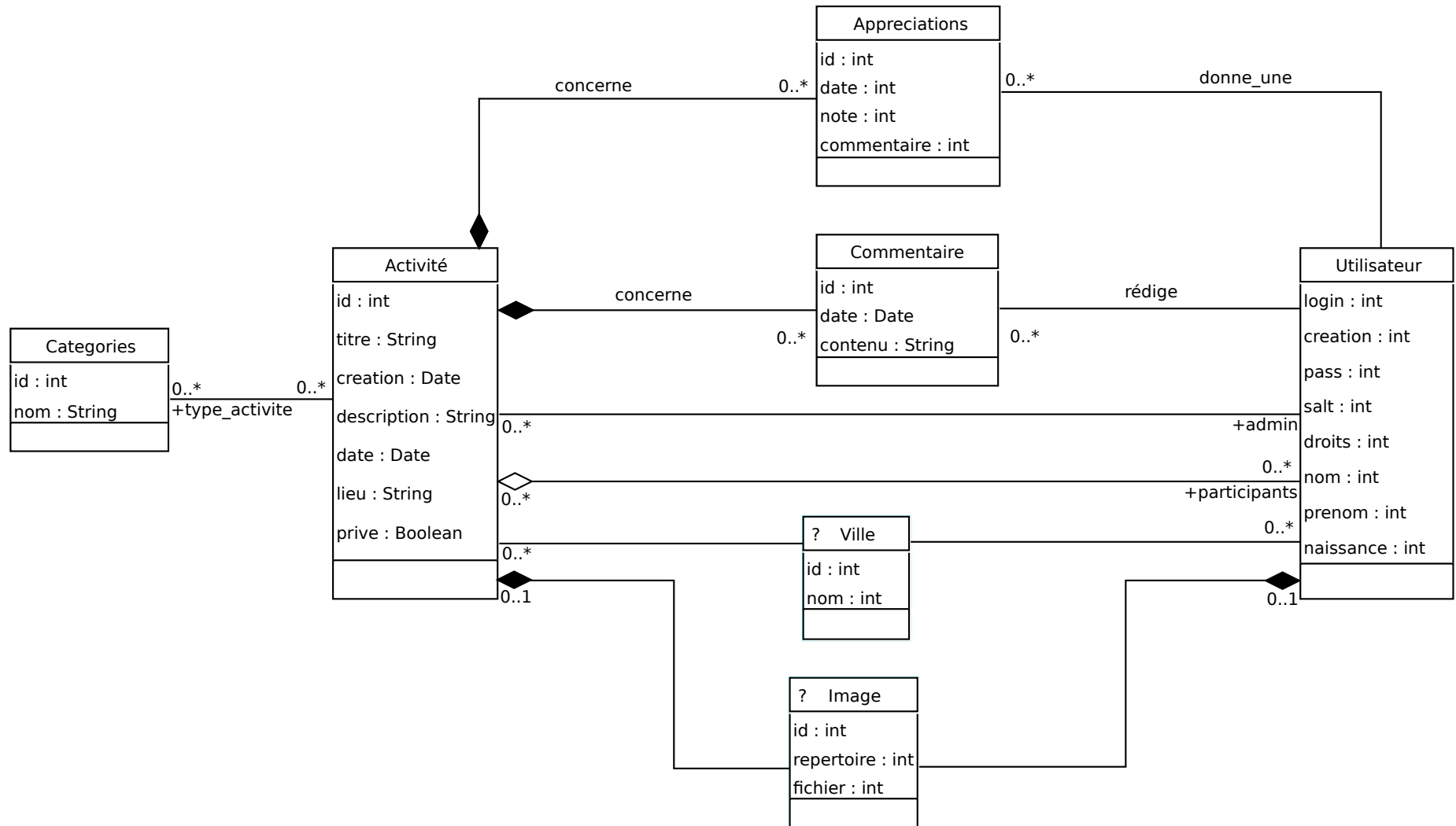
classes/CnxBdd.php

```

public function sendPrep ($query, $val, $types) {
    $stmt = $this->cnx->prepare($query);
    if ($this->cnx->errno)
        Connect::displayError(2);
    $ref = array();
    $ref[0] = $types; // Tableau de référence pour paramètres bind_param dans call_user_func_array
    $len = count($val);
    for ($i=0;$i<$len;$i++) {
        $ref[$i+1] = &$val[$i];
    }
    call_user_func_array(array($stmt, 'bind_param'), $ref); // bind_param dynamique
    if ($this->cnx->errno)
        Connect::displayError(2);
    $result = $stmt -> execute();
    if ($this->cnx->errno)
        Connect::displayError(2);
    $stmt -> close();
}

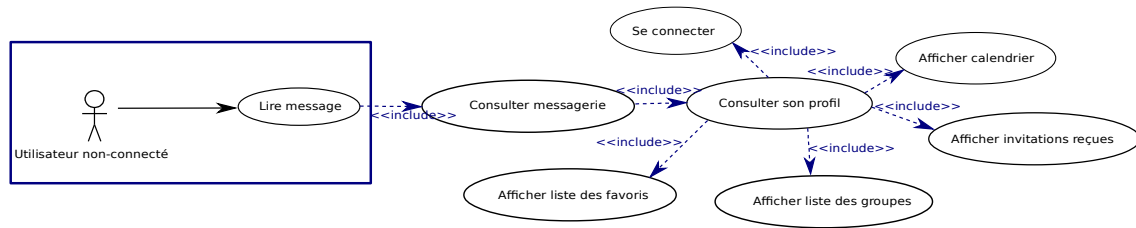
```

4.1.5 Diagramme de contenu



4.2 Cas d'utilisation : Lire un message privé

4.2.1 Diagramme de cas d'utilisation



4.2.2 Description textuelle

Scénario nominal :

1. L'utilisateur affiche le bloc de connexion, saisit ses identifiants et valide
2. Le système vérifie les identifiants et lance la session de l'utilisateur
3. L'utilisateur clique sur son login
4. Le système affiche le profil de l'utilisateur
5. L'utilisateur clique sur l'onglet messagerie
6. Le système affiche la liste des messages
7. L'utilisateur clique sur le message
8. Le système affiche le message

Scénario alternatif 1 : *L'utilisateur ne dispose pas d'un compte*

(Remplace l'étape 1 du scénario nominal)

1. L'utilisateur clique sur Créer un compte
2. Le système affiche le formulaire de création de compte
3. L'utilisateur remplit le formulaire et le soumet
4. Le système enregistre le compte
5. Reprend l'étape 1 du scénario nominal

Scénario alternatif 2 : *Erreur de saisie dans le formulaire création de compte*

(Remplace l'étape 4 du scénario alternatif 1)

4. Le système détecte une erreur, la notifie à l'utilisateur et renvoie le formulaire
5. Reprend l'étape 3 du scénario alternatif 1

Scénario alternatif 3 : *Erreur de saisie de login ou mot de passe*

(Remplace l'étape 2 du scénario nominal)

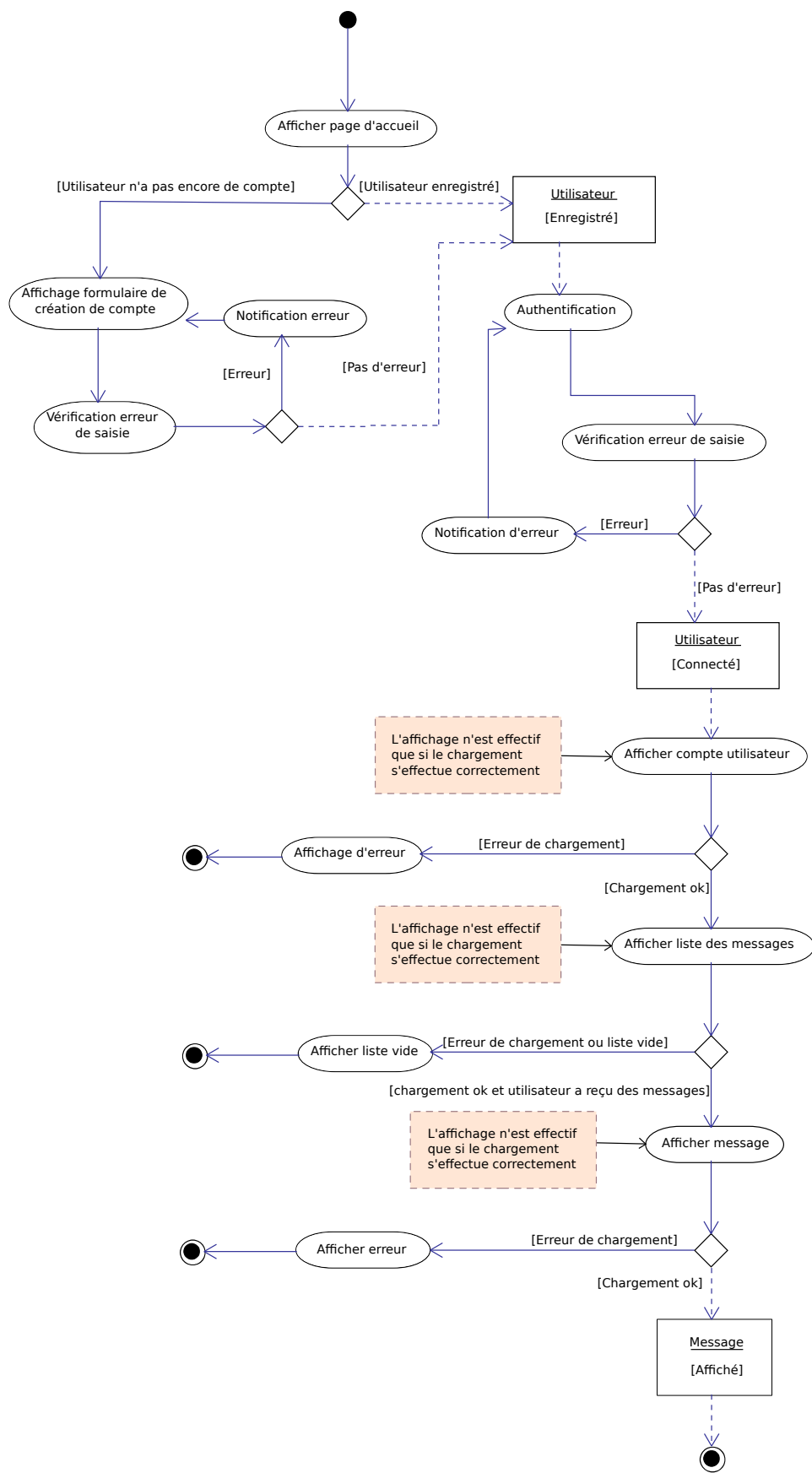
2. Le système détecte une erreur de login ou mot de passe et notifie l'utilisateur
3. Reprend l'étape 1 du scénario nominal

Scénario alternatif 4 : *L'utilisateur n'a pas reçu de message*

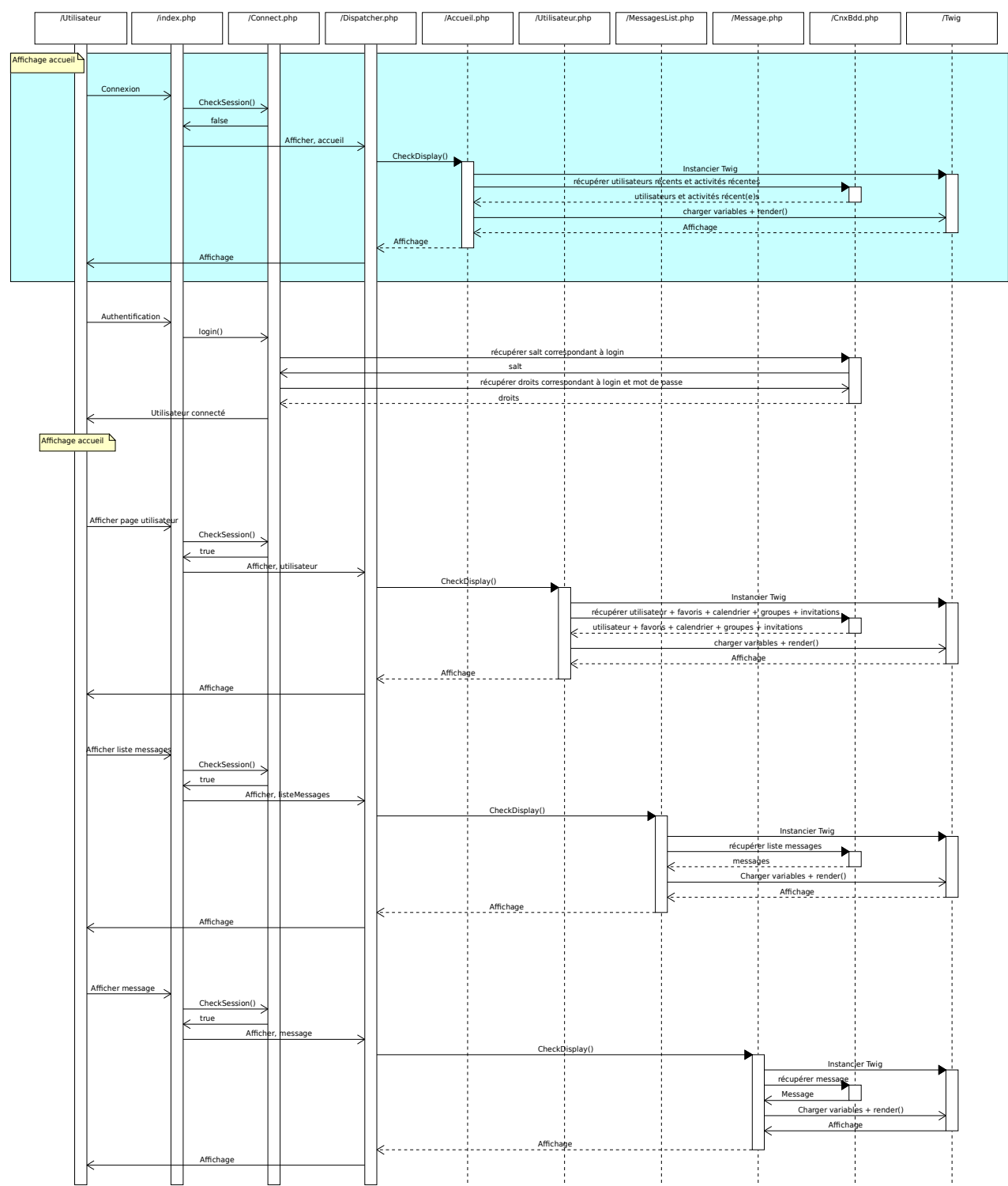
(Remplace l'étape 6 du scénario nominal)

6. Le système affiche une liste vide

4.2.3 Diagramme d'activité

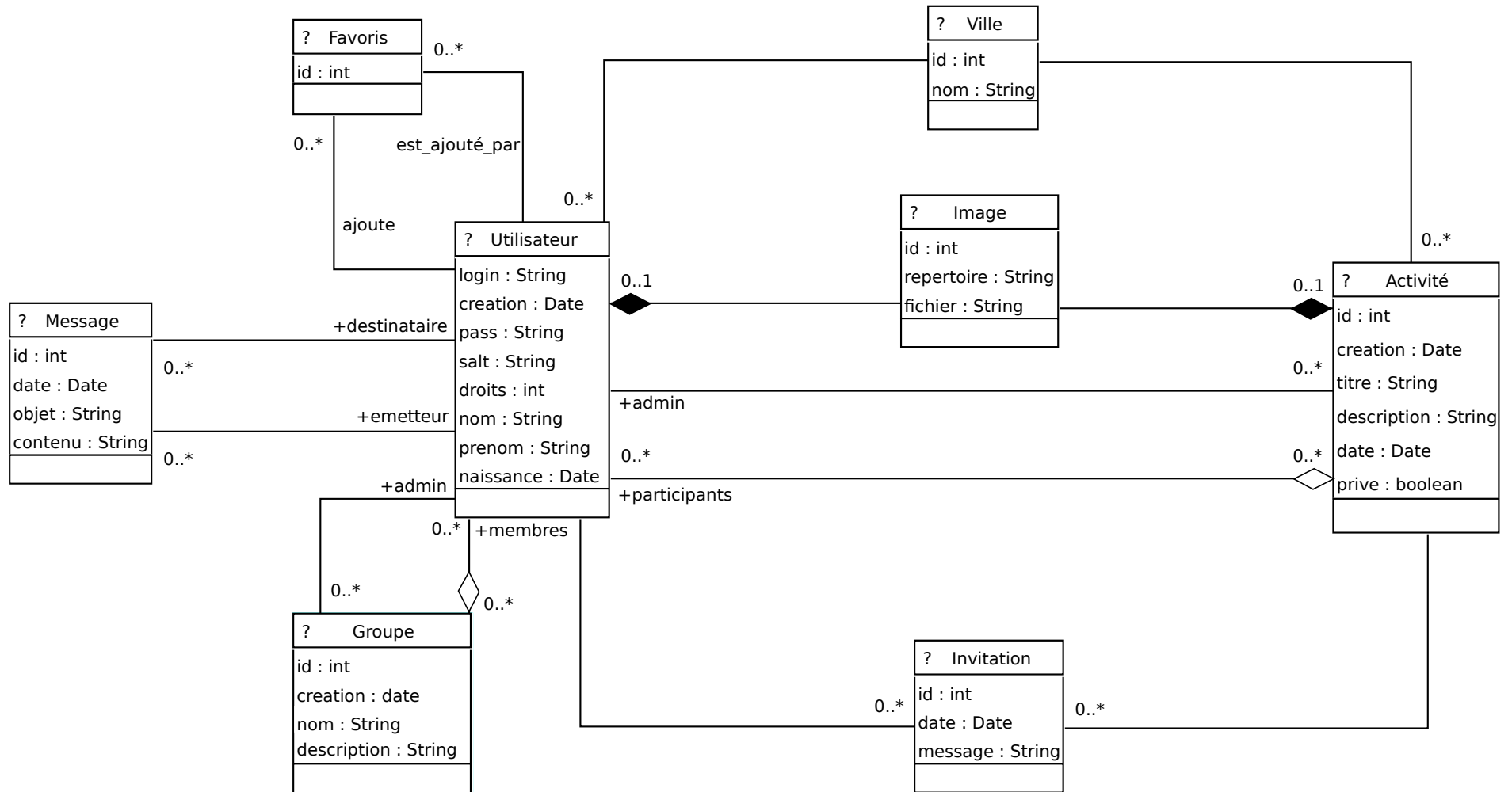


4.2.4 Diagramme de séquence



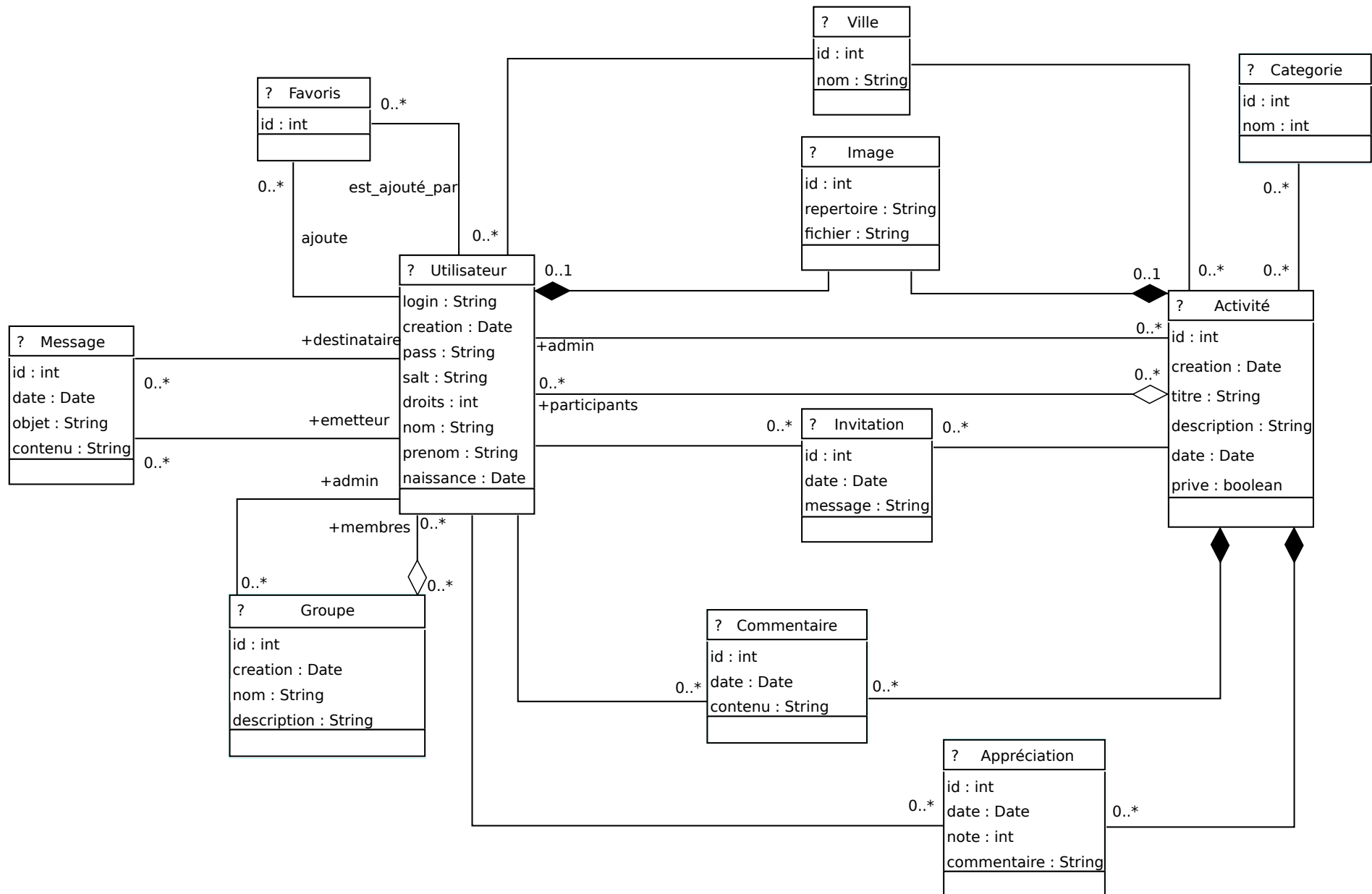
Note : Cette partie du code n'est pas encore développée

4.2.5 Diagramme de contenu

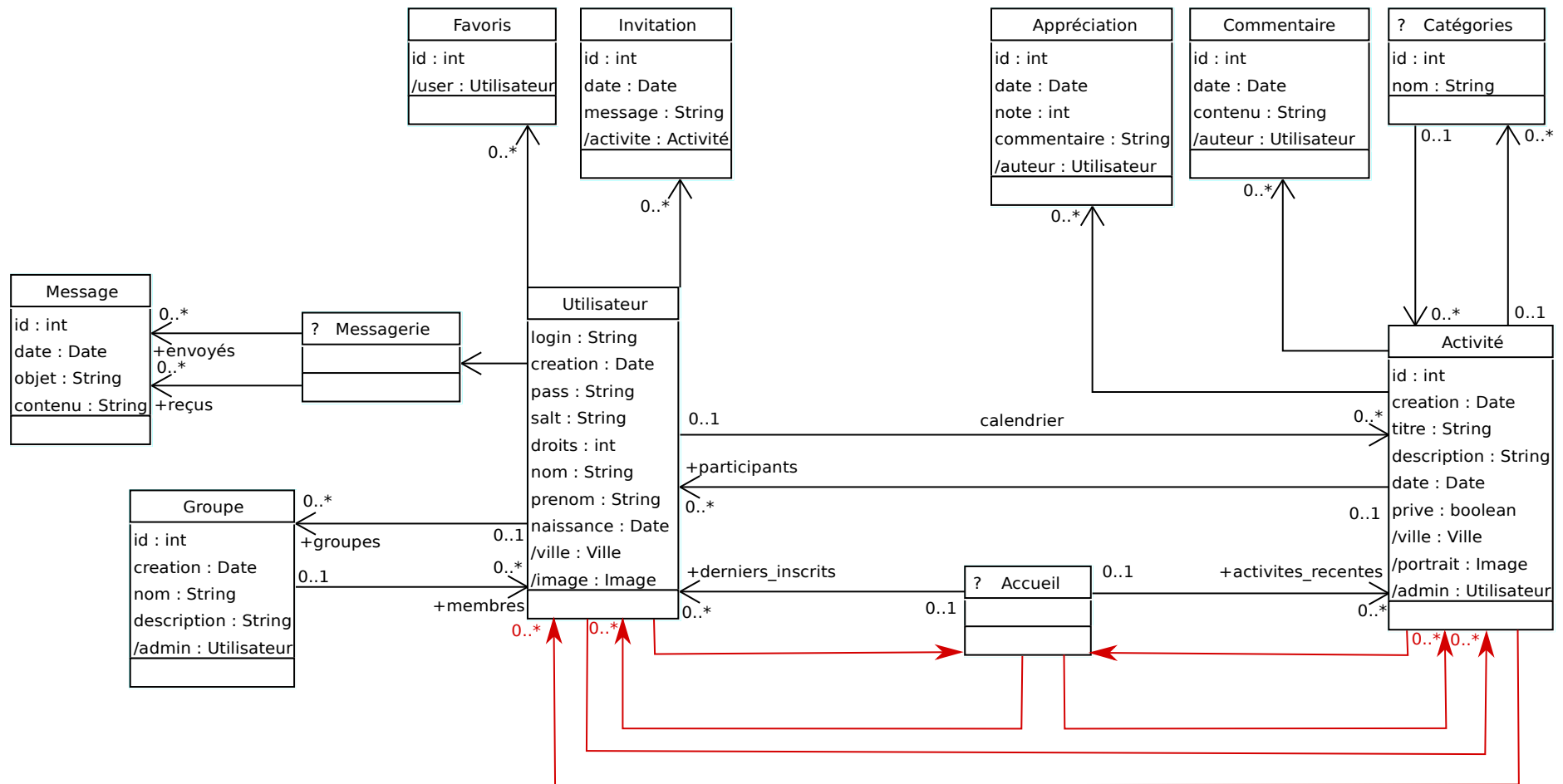


5. Diagrammes UWE

5.1 Diagramme de contenu global pour les deux cas d'utilisation



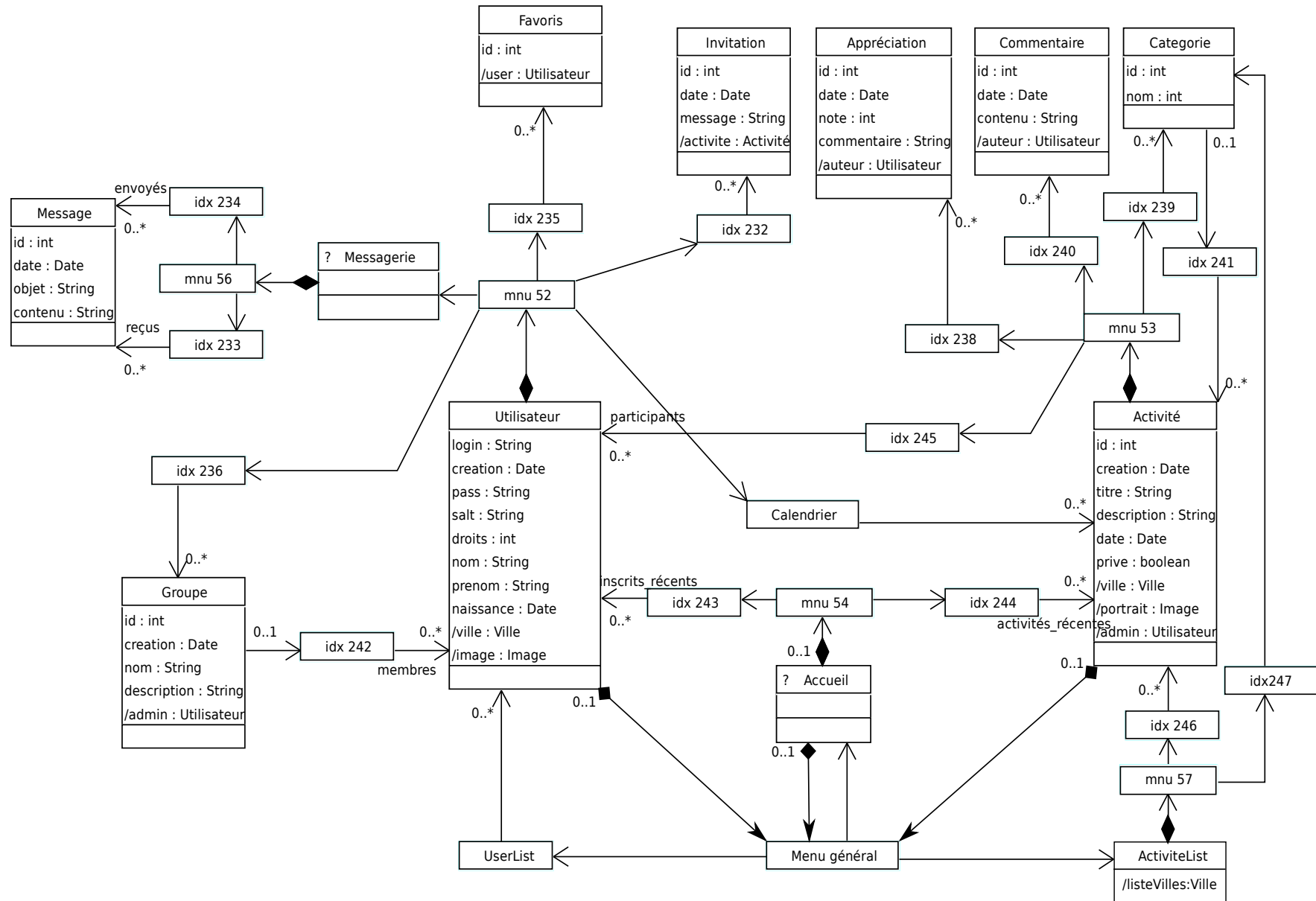
5.2 Modèle de navigation



Le modèle de navigation représente les liens effectifs entre les classes ainsi que les attributs dérivés (notés /nom_attribut) et rajoute les classes frontières (en l'occurrence Accueil et Messagerie).

Notes : Les classes Ville et Image ont été retirées du modèle et remplacées par des attributs dérivés dans Utilisateur et Accueil. Les liens représentés en rouge correspondent aux liens accessibles par le menu général des pages. Celui-ci apparaît sur chaque pages du site.

5.3 Modèle d'accès



Le modèle d'accès rajoute les primitives d'accès au modèle de navigation.

Note : Nous avons remplacé les liens correspondant au menu général par un objet menu. Cela n'est théoriquement pas autorisé par UWE, car un menu n'est censé être lié qu'à une seule classe, mais cette représentation semble mieux correspondre à notre application.

5.4 Diagramme de présentation

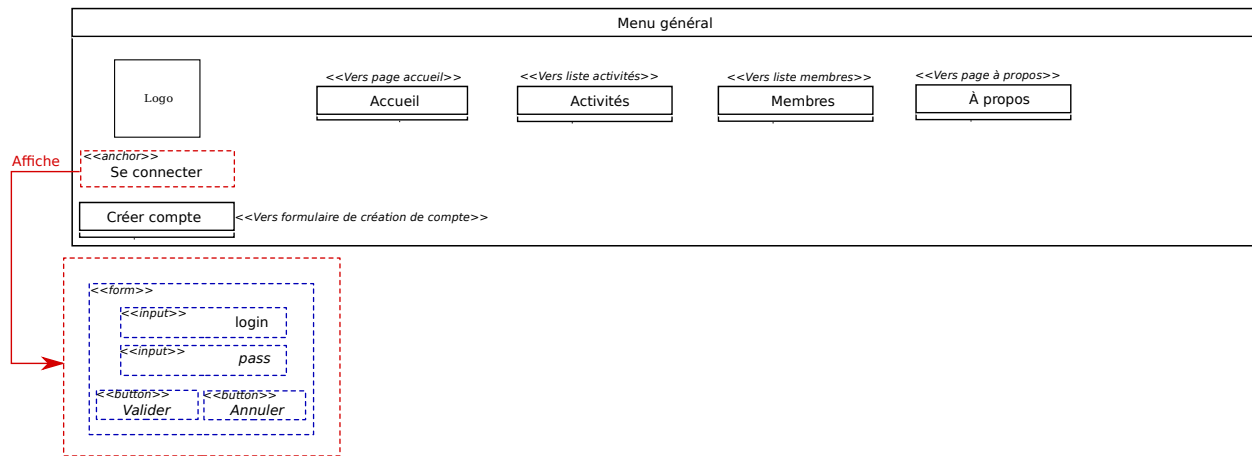
Le diagramme de présentation donne la structure des éléments de navigation, en particulier le type et l'emplacement des éléments principaux.

5.4.1 Canevas



5.4.2 Menu général

Utilisateur non-connecté



Accueil

Activités

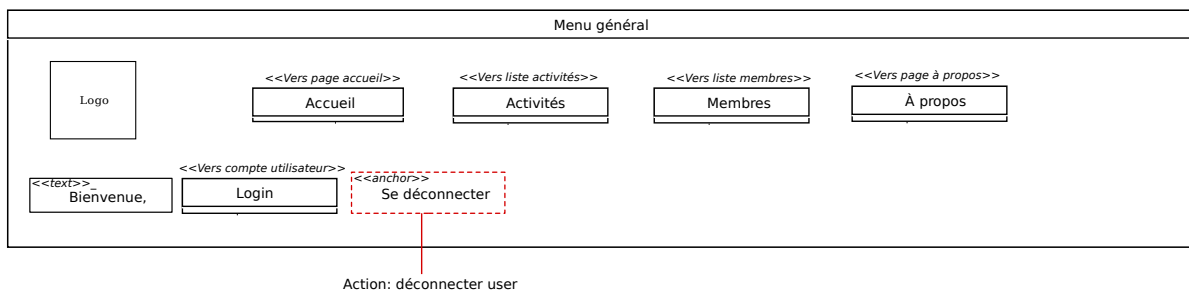
Membres

À propos

Se connecter
[Créer compte](#)

Nom d'utilisateur	
Mot de passe	
<input type="button" value="Valider"/>	<input type="button" value="Annuler"/>

Utilisateur connecté et admin



Accueil

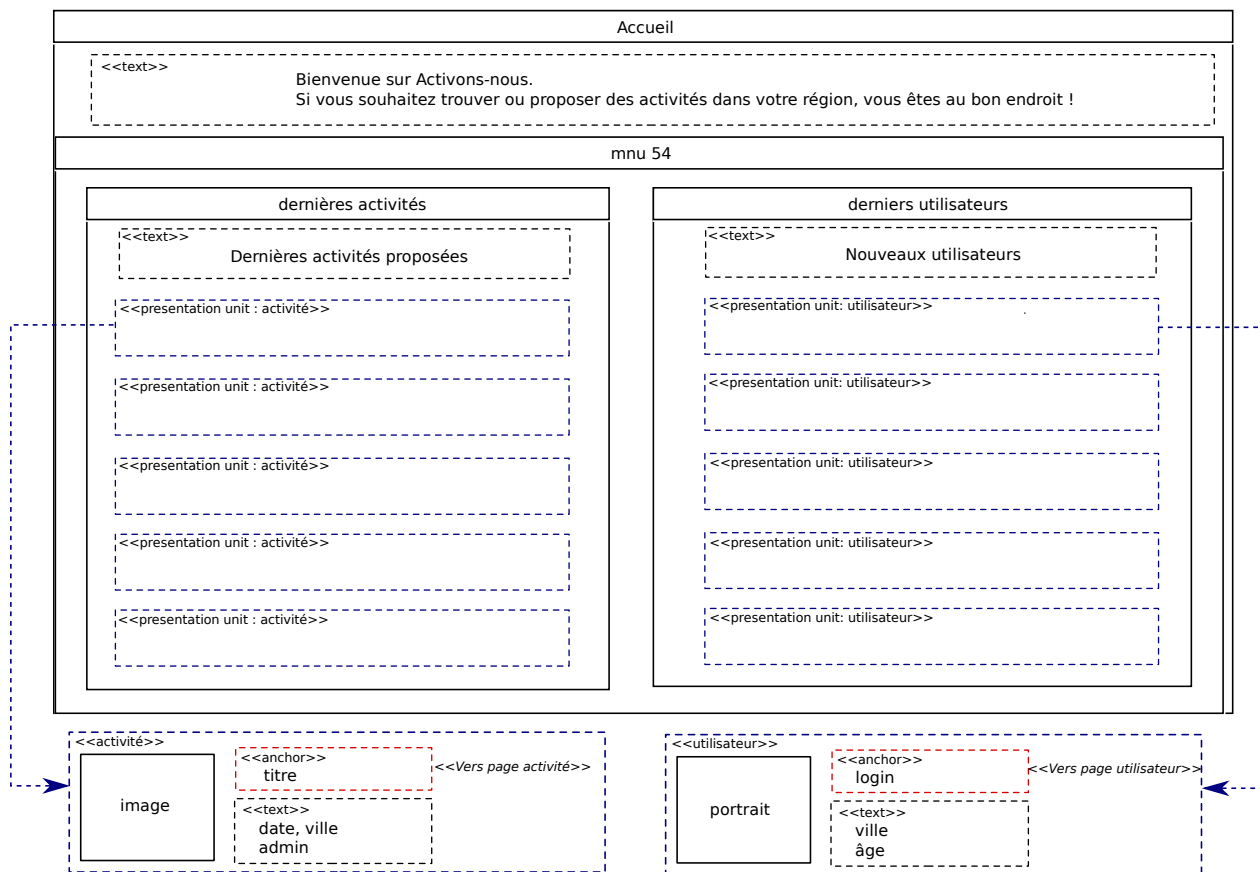
Activités

Membres

À propos

Bienvenue: [admin](#) [Se déconnecter](#)

5.4.3 Contenu Accueil






Bienvenue sur *Activons-nous*.

Si vous souhaitez trouver ou proposer des activités dans votre région, vous êtes au bon endroit !

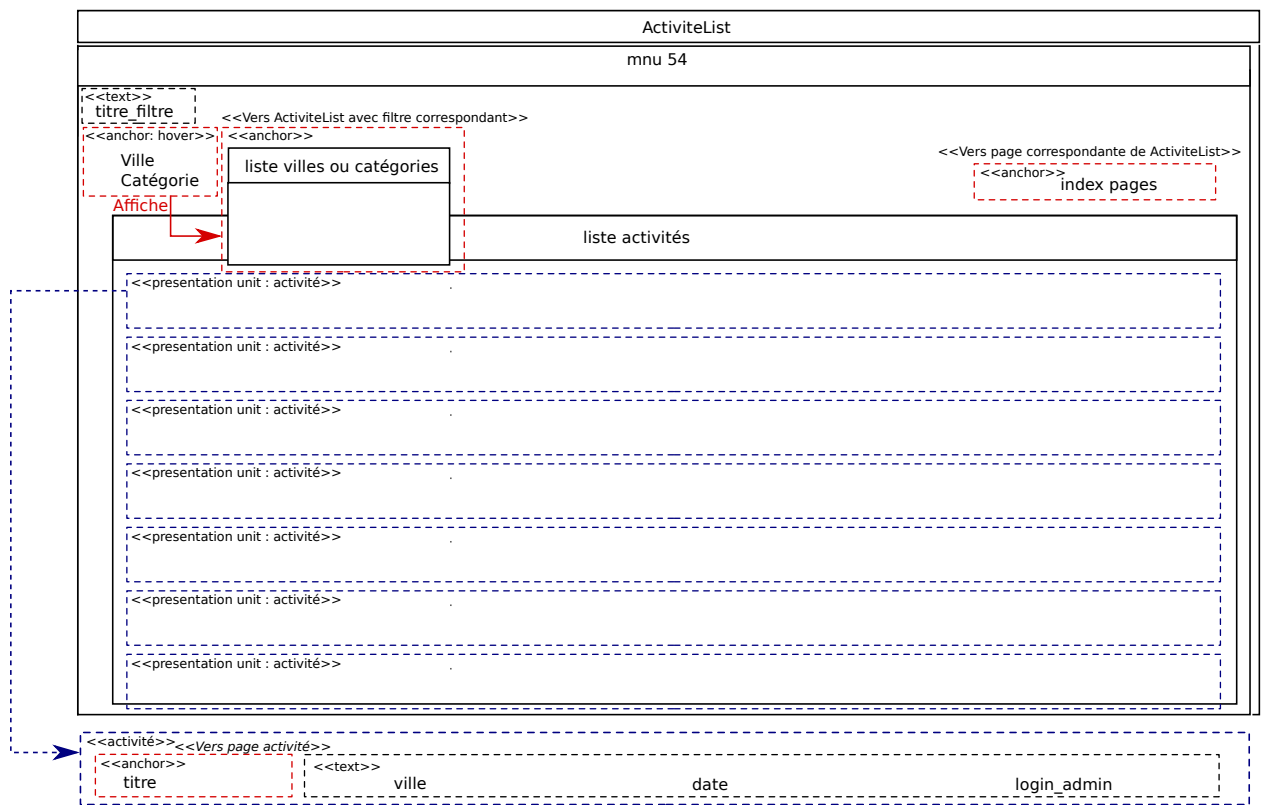
Dernières activités proposées

	Sortie escalade Le 5/7/2016 à 14h30, Lyon Dupont
	Dîner Malgache Le 26/6/2016 à 20h00, Paris PP_Officiel
	Rétrospective Hitchcock Le 2/7/2016 à 19h00, Paris VVG12

Nouveaux utilisateurs

	Dupont Lyon 37
	PP_Officiel Paris 54
	Denis05 Lyon 27
	Mona42 Paris

5.4.4 Contenu ActiviteList



Filtre			
Ville			
Catégorie			
Sortie escalade	Lyon	Le 5/7/2016 à 14h30	Dupont
Diner Malgache	Paris	Le 25/5/2016 à 20h00	PP_Officiel
Retrospective Hitchcock	Paris	Le 2/7/2016 à 19h00	VVG12

Filtre

Ville

Catégorie

Sortie escalade

Cinéma

Diversissement

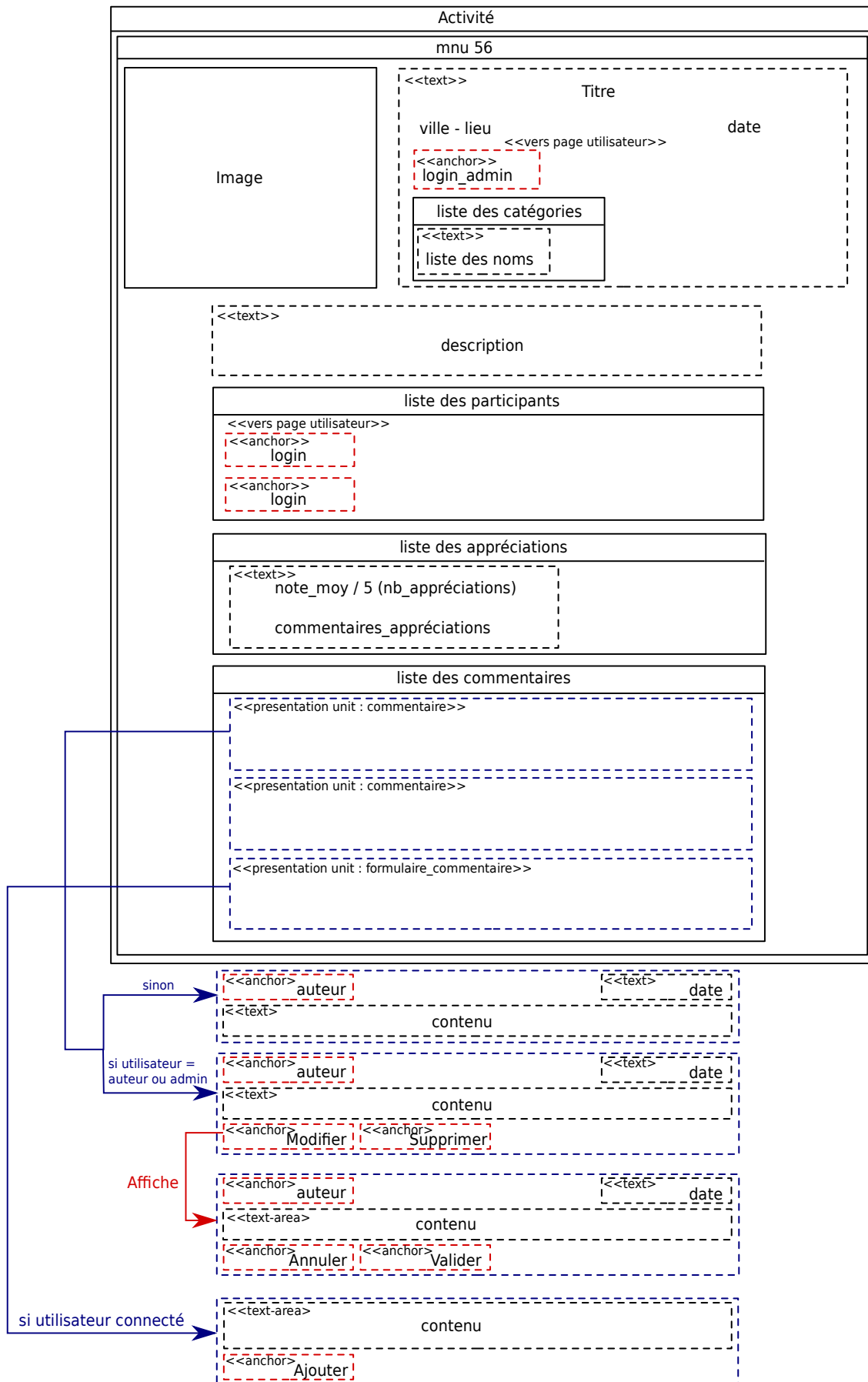
Restau

Sortir

Sport

Lyon

5.4.5 Contenu Activité





Rétrospective Hitchcock

Paris - MK2 Quai de Loire

Le 2/7/2016 à 19h00

Organisé par : [VVG12](#)

- Cinéma
- Divertissement

Programme:

- Les oiseaux
- Psychose

PARTICIPANTS

[VVG12](#)

[Mona42](#)

[PP_Officiel](#)

Appréciations : 4.5/5 (2 appréciations)

Un commentaire
d'appréciation ...

Commentaires :

[Mona42](#)

Un premier commentaire ...

2016-06-06 12:14:23

Modifier

Supprimer

[VVG12](#)

Un deuxième
sur deux lignes !

2016-06-06 12:14:23

[PP_Officiel](#)

Et un troisième
avec des caractères spéciaux: /\'azaz'

2016-06-06 12:14:23

Nouveau commentaire...

Ajouter

[Mona42](#)

2016-06-06 12:14:23

Un premier commentaire ...

Annuler

Valider

Note : Les autres pages ne sont pas encore développées

6. Informations complémentaires sur le site

6.1 Hébergement

Le site est actuellement hébergé et accessible à l'adresse suivante :

<http://vps133660.ovh.net:8083/>

6.2 Situation du développement

Les pages et fonctionnalités suivantes sont disponibles :

- Accueil
- Liste des activités
- Activités
- Système de connexion / déconnexion
- Ajout / modification / suppression de commentaires

6.3 Informations d'authentification

Liste des comptes accessibles :

- Mona42
- Dupont
- VVG12
- PP_Officiel
- Denis05
- Admin

Les mots de passe sont : azeaze

6.4 Informations pour le déploiement

Le site est déployé sur un serveur Nginx mais il devrait être compatible avec un serveur Apache.

Il est conçu pour fonctionner avec MySQL et PHP 5.5+

Voici la procédure pour le lancer :

- Ouvrir classes/init.php
- Éditer les variables \$login, \$pass et \$host pour correspondre au serveur MySQL
- Lancer init.php

6.5 Remarques sur l'organisation

- /index.php et /classes/Dispatcher.php sont des classes contrôleur
- /classes/Connect.php est une classe transversale : gère authentification et renvoi de pages
- /classes/CnxBdd.php est une classe transversale : gère envoi des données à la BDD
- L'ensemble des classes du modèle sont dans /classes/
- Dans /classes/, BaseClasse.php, BaseAffichage.php et BaseListe.php sont des classes abstraites : Elles servent de base aux autres classes du modèle
- Les vues sont dans /vues/

7. Références

Général

- Cours CNAM
- <https://fr.wikipedia.org/wiki/>
- <http://stackoverflow.com/>

UML

- <http://uml.free.fr>
- <http://laurent-audibert.developpez.com/Cours-UML/>
- <https://openclassrooms.com>

UWE

- <http://uwe.pst.ifi.lmu.de/tutorial/Tutoriel-UWE-fr.pdf>

Twig

- <http://twig.sensiolabs.org/documentation>