



Programmation Web

ODIER Valentin - WAKIM
Marie

On News

EFREI
30 av de la République
94800 Villejuif
[N° de télécopie]
29/01/2012

Sommaire

Introduction	4
Objectif du site	5
Diagrammes	5
Besoins fonctionnels	5
Use case.....	5
Diagramme de séquence : en annexe	6
Besoins non-fonctionnels	6
Compatibilité	6
Sécurité.....	6
Besoins liés aux commentaires.....	6
Besoins liés aux membres	6
Besoins liés aux articles	6
Structure du site.....	7
Architecture des pages.....	8
Graphe du site	8
Base de données	9
article	9
membre	9
tag	10
article_tag	10
com	10
Découpage du code.....	10
En 1	11
En 2	12
Entre la balise <html>	12
En 3	12
En 4	12
Explication de certaines parties du code.....	12
connexion.php	12
article.php	13
Améliorations futures	15

CONCLUSION	15
Annexes	16
S'inscrire	16
Commenter	16
Gérer News	0
Désinscrire.....	0
Rechercher.....	1
Lire	1
S'authentifier	2

Introduction

Ce projet nous a permis de nous (re)familiariser avec le langage web. Nous avons découvert le PHP et appris comment gérer une base de données.

Le PHP est semblable au langage C, ce ne fut pas la plus grande difficulté. Notre « problème » le plus notable était que nous ne connaissions pas toutes les fonctions disponibles en PHP ou en SQL. Nous cherchions à coder ces fonctions avant de constater qu'elles existaient déjà ! Exemple nous avons utilisé les variables \$sessions, cependant notre première idée était de passer par les variables globales.

Pour construire les diagrammes nous sommes passés par le logiciel StarUML que nous avons vu l'année passée. Cette étape est très importante puisqu'elle permet de délimiter le projet, de choisir quelles fonctionnalités nous allons créer.

Dans un premier temps, nous vous introduirons l'objectif du site, puis nous vous présenterons nos diagrammes, et les fonctionnalités créées. Puis dans un second temps nous expliquerons l'architecture de notre site, et notre base de données. Enfin nous introduirons des idées d'améliorations futures pour le site.

Un fichier texte « Odier Valentin _ Wakim Marie.txt » a été déposé où sont notés les logins et mots de passe des différents membres.

Objectif du site

Nous avons développé un site évènementiel et d'actualité, qui permet aux utilisateurs d'être au courant des événements dans leur région. Des articles sont postés sur le site en fonction de quatre grands thèmes :

Sport	Cinéma	Concert	Jeux vidéo
-------	--------	---------	------------

Les utilisateurs peuvent rechercher des articles, en ajouter, en commenter...

Les fonctionnalités du site sont au point. Nous avons créé quelques articles, qui ne représentent pas forcément le thème du site.

Nous avons défini trois types d'utilisateurs : visiteur, membre, admin.

Diagrammes

Besoins fonctionnels

Use case

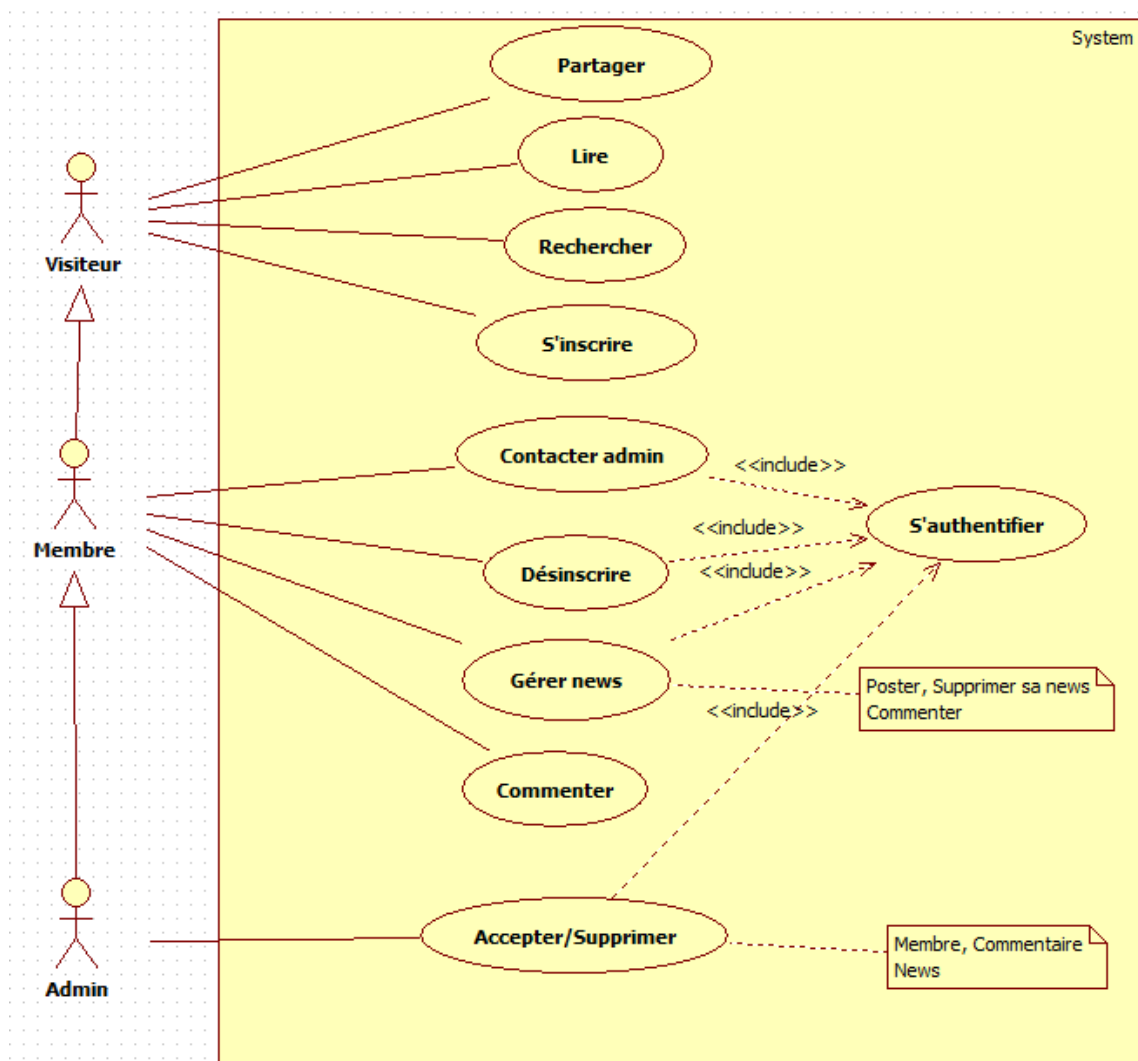


Diagramme de séquence : en annexe

Besoins non-fonctionnels

Compatibilité

Le site web est compatible sous Google Chrome, et Mozilla Firefox.

Sous Internet Explorer les fonctionnalités fonctionnent correctement, seul l’affichage coince à 2 endroits lorsque l’utilisateur effectue une recherche ou quand il veut ajouter un article.

Sécurité

Pour accéder aux fonctionnalités du site il faut s’authentifier par un mot de passe et un pseudo (le pseudo étant unique pour chaque utilisateur). Ainsi il faut être authentifié pour pouvoir ajouter un commentaire.

Besoins liés aux commentaires

Chaque commentaire est lié à un article. Un membre peut ajouter des commentaires, mais il peut aussi en signaler. Lorsqu’un membre signale un commentaire, une variable de la table « com » passe alors à 1. L’administrateur a la possibilité de voir tous les commentaires signalés, il peut alors choisir de laisser le commentaire (la valeur repasse à 0), ou bien de le supprimer.

Besoins liés aux membres

Lors de chaque connexion nous vérifions bien entendu que le mot de passe est associé au bon pseudo. Quand un visiteur souhaite s’inscrire sur le site, nous vérifions que le pseudo choisi n’a pas été déjà entré, pour cela nous vérifions deux choses :

- La présence d’espace à la fin du pseudo, s’il y en a nous les supprimons ;
- Les majuscules. Lors de la vérification nous passons le pseudo entré en minuscule que nous comparons avec notre base de données. Ainsi nous différencions : Dupont de DuPONT, dUPONT...

Besoins liés aux articles

Chaque article est lié à une des quatre catégories. La recherche d’un article se fait en fonction des tags qui définissent un article.

Structure du site

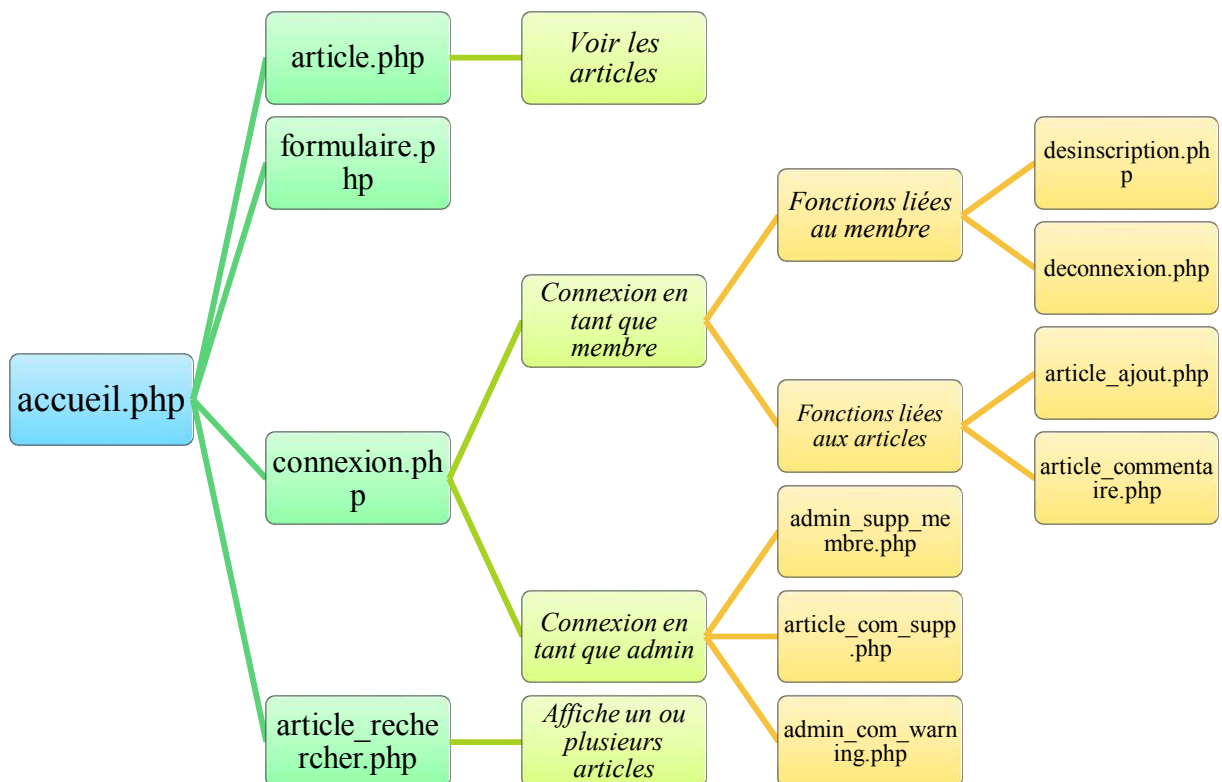
Voici un aperçu du site :



1. Bannière : cliquer sur la bannière permet d'accéder à la page accueil.php
2. Navigation horizontale : permet d'accéder rapidement aux fonctionnalités principales du site
3. Navigation verticale : permet d'accéder rapidement aux catégories principales du site
4. Section : là où l'utilisateur lit les articles, où il peut s'authentifier, s'inscrire... Permet d'afficher les fonctionnalités à l'utilisateur.
5. Footer : pied de page avec ici un lien pour contacter l'administrateur.

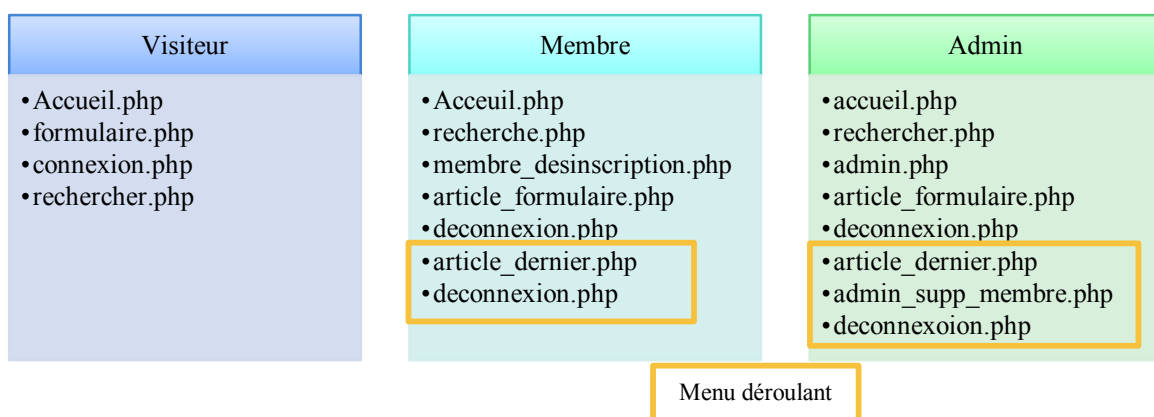
Architecture des pages

Graphe du site

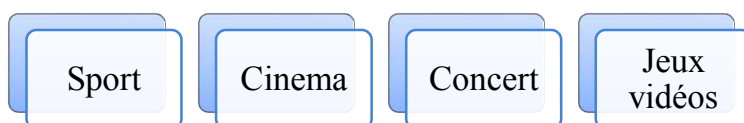


Rappel : admin hérite de membre, et donc des fonctionnalités. Seule la fonction « desinscription.php » est spécifique au membre : le compte administrateur ne peut pas être supprimé du site.

Bannière horizontale :

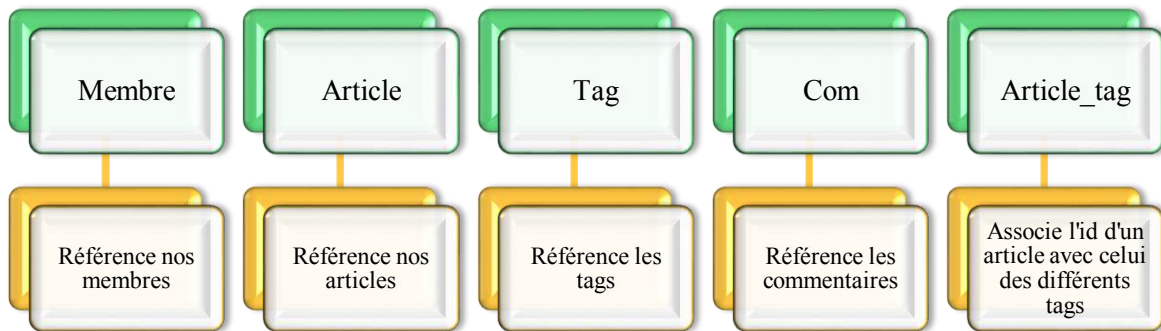


Bannière verticale :



Base de données

Nous avons utilisé cinq bases de données. Les quatre premières possèdent chacune une colonne "id" afin que chaque élément soit unique :



- Chaque base de données possède un "id" pour permettre l'unicité des éléments ;
- Une base pour les membres ;
- Une base pour les articles ;
- Une base pour les tags ;
- Une base pour les commentaires ;
- Et une dernière "article_tag" qui associe les id de l'article avec ceux des tags.

article

5 colonnes :

- id : pour identifier chaque article. De type int, s'auto-incrémente ;
- titre : de type varchar, avec 30 caractères maximum ;
- catégorie : sport/jeux/cinema/concert. De type varchar ;
- date : de la mise en ligne de l'article. De type datetime ;
- texte : contenu de l'article. De type varchar avec 700 caractères maximum.

membre

7 colonnes qui identifient le membre :

- id : pour identifier chaque membre. Comme l'article de type int et s'auto-incrémente ;
- nom : de type varchar, avec 25 caractères maximum ;
- prenom : de type varchar, avec 25 caractères maximum ;
- pseudo : de type varchar, avec 25 caractères maximum ;
- mdp : de type varchar, avec 20 caractères maximum ;
- code : de type int, avec 2 caractères maximum ;
- mail : de type varchar, avec 50 caractères maximum.

tag

2 colonnes :

- id_t : pour identifier les tags. De même il s'incrémente seul ;
- nom_t : le nom du tag qu'entre le membre.

article_tag

2 colonnes :

- id_article : de la base de données "article" ;
- id_tag : de la base de données "tag".

com

7 colonnes :

- id_com : pour identifier chaque commentaire ;
- id_article : id de l'article auquel est attaché le commentaire
- date : de type datetime.
- auteur : le pseudo de l'auteur du commentaire, de type varchar.
- commentaire : contenu du commentaire.
- warning : par défaut à 0, et passe à 1 quand un membre signale un commentaire abusif.

Découpage du code

Notre site est composé de 30 fichiers PHP, 3 fichiers CSS, et de 2 fichiers JPG :

La plupart de nos pages sont codées ainsi :

```
1 <?php
2     session_start();
3 ?>
4
5 <!DOCTYPE html>
6 <html>
7     <head>
8         <meta charset="ISO-8859-1" />
9         <link rel="stylesheet" href="design.css" />
10        <title>Titre de la page</title>
11    </head>
12
13    <body>
14        <?php include("type_banniere.php");?>
15        <div id="central">
16            <?php include("vertical.php");?>
17            <section>
18                <article>
19                    <h1> </h1>
20                    <p>
21                        <?php
22
23                    ?>
24                </p>
25            </article>
26        </section>
27    </div>
28    <?php include("pied_page.php");?>
29 </body>
30 </html>
```

En 1

Pour réaliser ce projet nous nous sommes servis des sessions. Les fonctionnalités liées à ces variables nous ont permis de conserver facilement le pseudo du membre tout au long de sa visite sur le site. Il faut démarrer la session par la fonction : `session_start()`.

Nous avons défini deux variables de session : `type` et `pseudo`.

Lors de la première visite, nous vérifions si « `$_SESSION['type']` » a été défini :

- Si elle existe, il n'y a pas de problème
- Sinon, cela signifie que la session n'est pas déterminée. Nous donnons à nos deux variables la valeur : visiteur.

En 2

Il s'agit du type de document employé tout au long du site.

Entre la balise <html>

En 3

Cette balise <head> fournit des informations sur la page, ici :

- Le type d'encodage : celui-ci permet de gérer les accents. Nous avons choisi celui-ci mais d'autres existent ;
- Nous avons inclus un fichier CSS pour l'allure générale du site ;
- Le titre de la page.

En 4

Cette dernière partie correspond à notre code, à ce que nous voulons afficher à l'écran. Pour chaque page permettant d'afficher un contenu sur le site, nous incluons 3 fichiers PHP :

- type_banniere.php : correspondant à la bannière horizontale du site, spécifique pour les différents types d'utilisateurs ;
- vertical.php : unique quel que soit l'utilisateur, il s'agit de la bannière verticale ;
- pied_page.php : le footer.

Nous avons séparés le code en plusieurs balises HTML : <div>, <section>, <article>, <h1>, <p>... Afin de bien différencier chaque section.

Ce qui est à l'intérieur de la balise <article> est spécifique à chaque page. Quand l'utilisateur visite le site c'est cette partie qui change, l'affichage des fonctionnalités ce fait ici, a contrario les bannières et le footer restent fixes.

Explication de certaines parties du code

connexion.php

Nous utilisons la fonction WHERE avec l'opérateur AND pour vérifier que le pseudo ET le mot de passe sont corrects. Pour cela nous avons en partie le code suivant :

Tout d'abord un formulaire qui demande le pseudo et le mot de passe choisis lors de l'inscription :

```
<form method="post" action="connexion.php">
  <p>
    <label for="pseudo"> Pseudo :</label> <input type="text" name="pseudo" placeholder="Votre pseudo" /> <br>
    <label for="mdp">Mot de passe :</label> <input type="password" name="mdp" placeholder="Mot de passe" /> <br>
    <input type="submit" value="Connexion" />
  </p>
</form>
```

Puis nous vérifions que l'utilisateur a complété les deux champs :

```
if(isset($_POST['pseudo']) || isset($_POST['mdp']))
{
    try
    {
        $pdo_options[PDO::ATTR_ERRMODE] = PDO::ERRMODE_EXCEPTION;
        $bdd = new PDO('mysql:host=localhost;dbname=bdd_site', 'root', '', $pdo_options);
```

Puis nous recherchons s'il existe bien un membre avec le pseudo et le mot de passe entrés. Si c'est le cas les variables sessions sont mises à jours et la variable \$erreur n'est pas définie l'utilisateur est renvoyé à la page d'accueil. Sinon nous affichons « utilisateur ou mot de passe incorrects » à l'écran.

```
$reponse = $bdd->prepare('SELECT pseudo, mdp from membre WHERE pseudo = ? AND mdp = ?');
$reponse->execute(array($_POST['pseudo'], $_POST['mdp']));

if($reponse->fetch())
{
    if($_POST['pseudo'] == 'admin') // Si le membre est un admin
    {
        $_SESSION['type'] = 'admin'; // Mise à jour de la session
        $_SESSION['pseudo'] = 'admin';
        header('Location: accueil.php');
        break;
    }
    $_SESSION['type'] = 'membre';
    $_SESSION['pseudo'] = $_POST['pseudo'];
}
else // Mdp ou pseudo incorrect
{
    $erreur = 'Utilisateur ou mot de passe incorrects';
}
if(!isset($erreur))
{
    header('Location: accueil.php');
}
else
{
    echo $erreur;
}
```

article.php

Cette page permet d'afficher les articles, le code n'est pas difficile à comprendre mais il y a beaucoup de choses. Car cette page permet d'afficher un ou plusieurs articles, d'ajouter un commentaire, d'en signaler :

Nous commençons par récupérer le titre, le contenu et l'id des différents articles correspondant à la catégorie choisie. La catégorie choisie est envoyé dans l'URL. Par ailleurs nous inversons nos tableaux pour afficher les articles du plus récent au plus vieux.

```
try
{
    $pdo_options[PDO::ATTR_ERRMODE] = PDO::ERRMODE_EXCEPTION;
    $bdd = new PDO('mysql:host=localhost;dbname=bdd_site', 'root', '', $pdo_options);

    $reponse = $bdd->prepare('SELECT * from article WHERE categorie = :categ');
    $reponse->execute(array('categ' => $_GET['categorie']));

    while($donnees = $reponse->fetch() )
    {
        $stab[] = $donnees['titre'];
        $stab_contenu[] = $donnees['texte'];
        $stab_id[] = $donnees['id'];
    }
    $stab = array_reverse($stab); //Inverse le tableau pour afficher les éléments récents d'abord
    $stab_contenu = array_reverse($stab_contenu);
    $stab_id = array_reverse($stab_id);
    $reponse->closeCursor();
}
catch (Exception $e)
{
    die('Erreur : ' . $e->getMessage());
}
```

Par défaut la variable \$page est définie, elle est transmise dans l'URL, il s'agit du cas où l'utilisateur peut lire plusieurs articles. Nous avons choisi de n'afficher que 6 articles par page :

```
if(isset($_GET['page']))
{
    $page = $_GET['page'];
    for($i=($page-1)*6 ; $i < sizeof($stab) ; $i++)
    {
        $sid = $stab_id[$i];
        echo '<a href="article.php?categorie='.$_GET['categorie'].'&id='.$sid.'">'.$stab[$i]. ' ' </a>' . '<br/>';
        echo $stab_contenu[$i]. '<br/> <br/>';
    }

    $page = (int)($i/6+1);
    if($page>1) echo '<a href="article.php?categorie='.$_GET['categorie'].'&page='.$page.'"> <-- </a>';
    echo "Page : " . ($page) . "/" . (int) (sizeof($stab)/6+1);
    if($page<(int) (sizeof($stab)/6+1)) echo '<a href="article.php?categorie='.$_GET['categorie'].'&page='.$page.'"> --> </a>' . '<br/>';
}
```

Cliquer sur le titre de l'article permet d'afficher seulement cet article (cf. encadré ci-dessus). Nous passons l'id de l'article dans l'URL, ce qui permet de le récupérer facilement par la suite. Si l'utilisateur clique sur le titre, la variable \$page n'existe plus dans l'URL et nous avons le code suivant :

```
else
{
    $reponse = $bdd->prepare('SELECT * from article WHERE id=:_id');
    $reponse->execute(array('_id' => $_GET['id']));
    while($donnees = $reponse->fetch() )
    {
        echo $donnees['titre'] . '<br/>';
        echo $donnees ['texte'] . '<br/> <br/>';
    }
}
```

A chaque article sont liés des commentaires, il s'agit de la quatrième partie de cette page. La table « com » retient l'id de l'article, il n'y a plus qu'à afficher les commentaires où l'id de l'article correspond à celui que nous visualisons : grâce à la fonction WHERE.

Puis si l'utilisateur est un administrateur il a la possibilité de supprimer les commentaires qu'il juge dangereux. Le membre lui peut signaler le commentaire à l'administrateur, nous faisons alors une mise à jour de la valeur « warning » dans la table « com ».

```
/*AFFICHAGE DES COMMENTAIRES */
$sql_com = $bdd->prepare('SELECT * from com WHERE id_article=:id'); // On récupère l'id de l'article associé au commentaire
$sql_com->execute(array('_id' => $_GET['id']));
while($comm = $sql_com->fetch() )
{
    ?> <div id="commentaire"> <p> <?php
    echo "Auteur : " ;
    echo $_comm['auteur'] . '<br/>';
    echo $_comm['commentaire'] . '<br/>';
    ?> </p> </div> <?php

/* SUPPRESSION DU COMMENTAIRE ET MISE A JOUR*/
if($_SESSION['type'] == 'admin')
{
    echo '<a href="article_com_supp.php?id_c='.$_comm['id_com'].'"> Supprimer le commentaire </a>' . '<br/>';
}
elseif($_SESSION['type'] == 'membre' && $_comm['warning'] == 0)
{
    echo '<a href="article_com_update.php?id_c='.$_comm['id_com'].'"> Signaler le commentaire </a>' . '<br/>';
}
if($_SESSION['type'] != 'visiteur' && $_comm['warning'] == 1)
{
    echo "Commentaire signalé" ;
}
}
```

Enfin nous avons un formulaire réservé aux membres (donc aussi à l'administrateur) pour ajouter des commentaires.

Améliorations futures

Nous avons réfléchi à plusieurs fonctionnalités à développer.

Lors de la connexion ajouter un champ « identifiant ou mot de passe oubliés », ce qui amènerait à une page où l'utilisateur rentre l'adresse e-mail choisie lors de l'inscription. Puis nous envoyons un mail avec l'identifiant et le mot de passe.

Une fois le site bien lancé, nous pouvons compter le nombre de visiteurs et le nombre de personnes connectés sur le site.

Nous pourrions ajouter un champ « région », et/ou « département », aux articles pour qu'ainsi l'utilisateur cherche l'article en fonction de sa région. Ainsi nous pourrions développer deux améliorations :

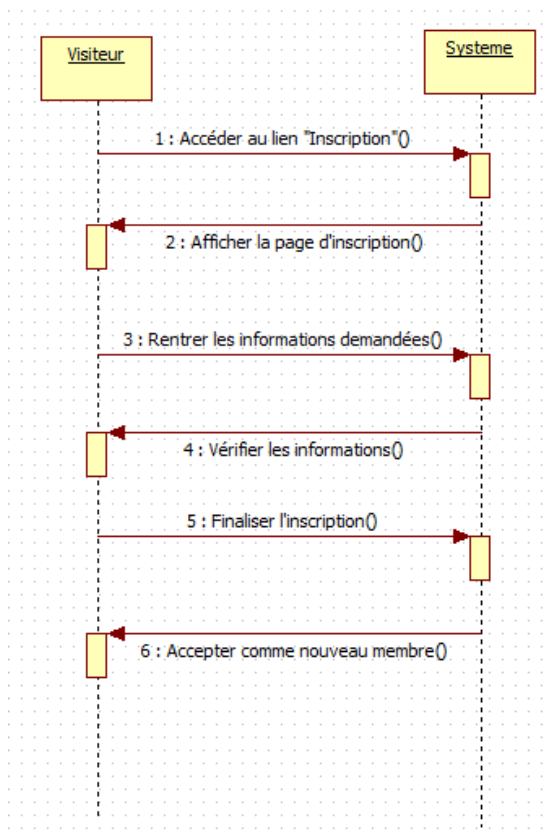
- Un système de géolocalisation, qui permettrait d'afficher une carte avec les événements proches de l'utilisateur ;
- Un calendrier, qui indiquerait les événements à venir. L'utilisateur pourrait choisir de n'afficher que les événements d'une certaine région, ou d'une certaine catégorie.

CONCLUSION

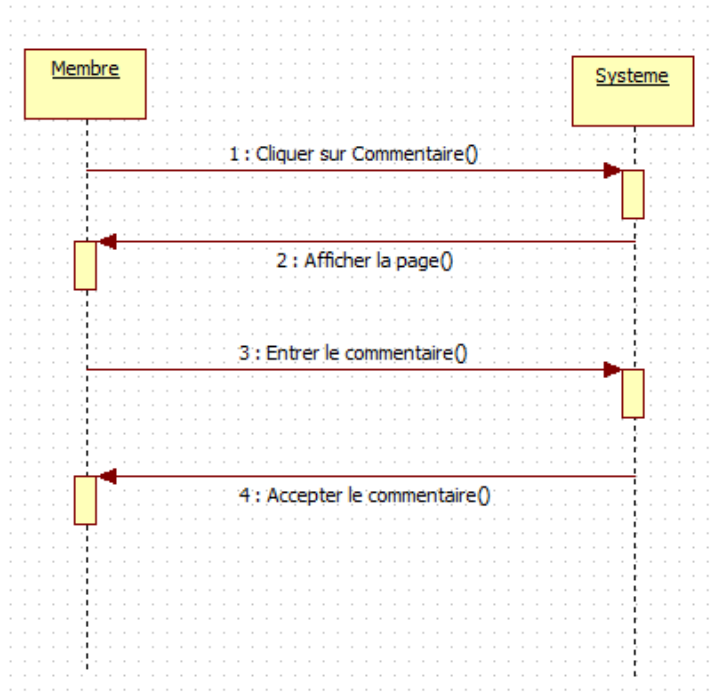
La conception de ce site web fut très intéressante, très enrichissante. Nous avons réussi à développer toutes les fonctionnalités principales que nous avons en tête, ainsi que des secondaires.

Annexes

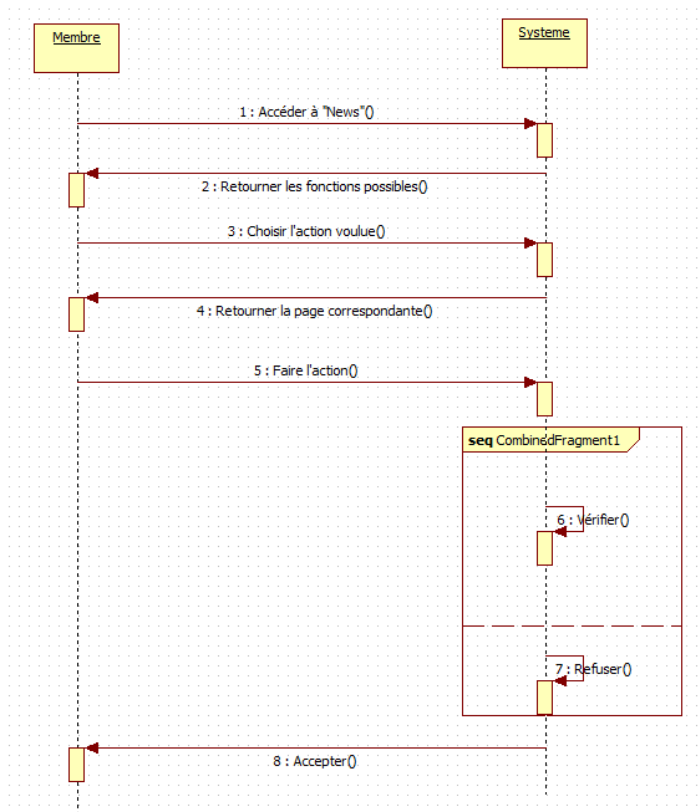
S'inscrire



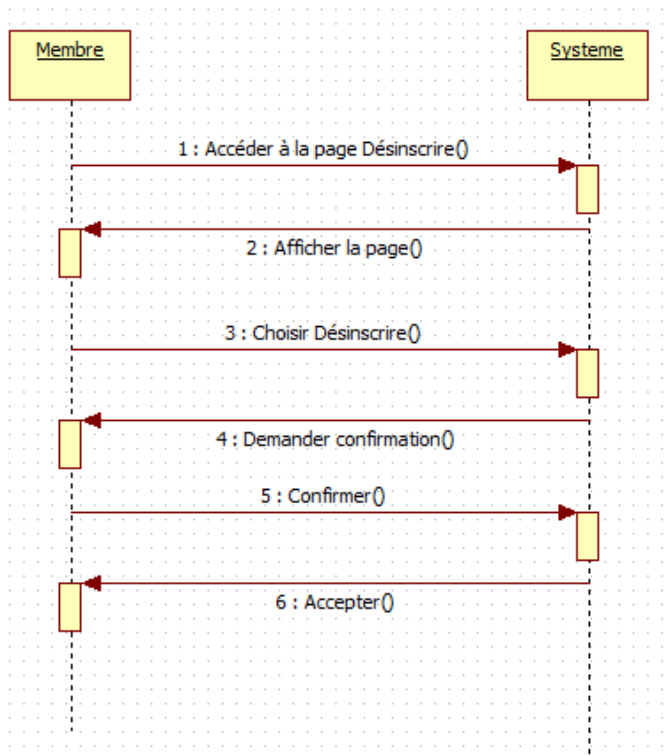
Commenter



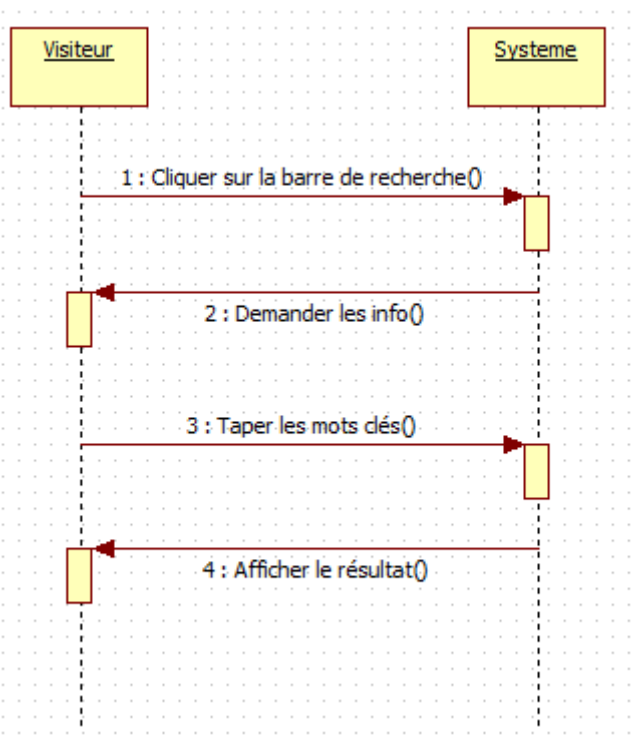
Gérer News



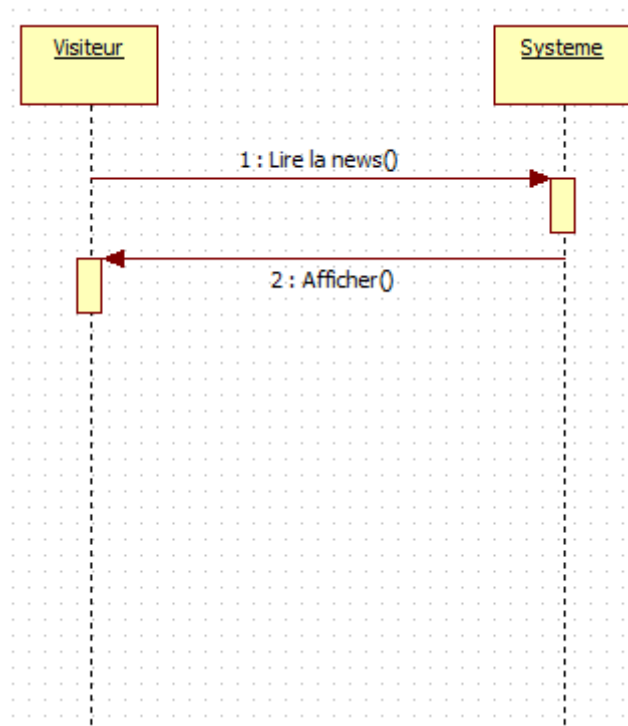
Désinscrire



Rechercher



Lire



S'authentifier

