Faculté de Sciences de Montpellier

3ème année de licence informatique
2015 – 2016

Projet Web :

Technote

Étudiants :
Alexandre CULTY
Vincent IAMPIETRO

Table des matières

Introduction

Le présent rapport détaille la création du site web technote qui est consultable à l'url `technote.calyxe.fr`. La mise en place du site technote s'inscrit dans le cadre de l'UE Architecture et programmation du Web, suivie en 3ème de Licence Informatique à la Faculté de Sciences de Montpellier. Le but du projet a été de concevoir un site web permettant la gestion de petits rapports techniques (appelés *technotes*) qui seraient écrits par les utilisateurs.

Le rapport est divisé en deux parties principales. La première partie s'attarde sur l'architecture logiciel du projet, qui arbore une structure calquée sur la patron de conception MVC. La deuxième partie présente les fonctionnalités du site, et l'accès qu'en ont les utilisateurs selon leur rôle.

Architecture du site

1 Modèle-Vue-Contrôleur

Le *Pattern* MVC a été choisi pour l'implémentation du site technote. Nous avons naturellement opté pour cette architecture logicielle, en grande partie pour garder une structure de programme claire et concise.

Le choix d'un autre protocole de conception, comme l'écriture page par page du site, ne nous aurait pas permis d'atteindre un niveau suffisant de clarté et de concision pour la structure du projet.

La Figure 1 détaille l'arborescence du programme technote qui a été compartimenté en suivant le patron MVC.

- `/` ..... Dossier www du serveur Web Apache
- `assets` .... Contient les différentes librairies CSS et Javascript utilisées pour le site, ainsi que le dossier images des fichiers uploader par les utilisateurs sur le serveur
- `controleurs` ..... Contrôleurs du Patron MVC
- `core` ..... Contient les super-classes PHP dont héritent les classes des dossiers modèles, vues et contrôleurs
- `divers` Contient notamment les fichiers SQL de création et de remplissage de la base de données associée
- `modeles` ..... Modèles du patron MVC
- `vendor` Contient le moteur de template *Twig*. Dossier généré par le gestionnaire de dépendances PHP Composer.
- `vues` ..... Vues du Patron MVC
- `.htaccess` ..... Fichier permettant la réécriture de l'url du site technote grâce au module url rewriting du serveur Apache
- `Autoload.php` ..... Fichier de chargement de l'ensemble des classes PHP
- `config.php` Fichier de configuration pour la communication avec la base de données en utilisant la classe PHP PDO
- `index.php` ..... Fichier de routage du Patron MVC

Figure 1 – Arborescence du site technote sur le serveur Web

2 Contrôleurs

Les contrôleurs sont les éléments du patron MVC faisant le liaison entre les modèles et les vues. Plus spécifiquement, ils possèdent un rôle de routage et de sélection des méthodes à utiliser s'inscrivant dans le processus de rendu de la page HTML.

2.1 Réécriture d'url

Le contenu de la barre d'adresse visible côté client est réécrite grâce au module *url rewriting* du serveur Apache. La réécriture d'url intervient par soucis de présentation, et permet à l'utilisateur de mieux s'orienter dans le site technote.

La Figure 2 est un extrait du fichier .htaccess qui regroupe l'ensemble des règles de réécriture d'url.

```
# Main
RewriteRule ^([a-zA-Z0-9_-]+)/([a-zA-Z0-9_-]+)/([a-zA-Z0-9_-]+)/?$ index.php?
    url_controleur=main&url_page=$1&url_action=$2&url_id=$3 [L,QSA]
RewriteRule ^([a-zA-Z0-9_-]+)/([a-zA-Z0-9_-]+)/?$ index.php?url_controleur=main&
    url_page=$1&url_action=$2 [L,QSA]
RewriteRule ^([a-zA-Z0-9_-]+)/?$ index.php?url_controleur=main&url_page=$1 [L,QSA]
RewriteRule ^/?$ index.php?url_controleur=main [L,QSA]
```

Figure 2 – Règles de réécriture d'url à destination du contrôleur Main

Les *rewrite rules* se divisent en deux parties :

- La première partie est une expression régulière permettant de vérifier la forme de l'url transmise. Les règles de réécriture sont appliquées dans l'ordre jusqu'à ce qu'une correspondance soit établie entre l'expression régulière et la forme de l'url. Après *"matching"* avec l'expression régulière, l'url est décomposée en un tableau contenant les parties marquées entre parenthèses dans la *regexp*.

  Par exemple, l'url `technote.calyxe.fr/technote/add` correspond à l'expression régulière `^([a-zA-Z0-9_-]+)/([a-zA-Z0-9_-]+)/?$` (le nom de domaine n'est pas considéré comme une partie de l'url). L'url est donc décomposée en un tableau à deux éléments {*"technote"*, *"add"*}.

- Dans la Figure 2, les deuxièmes parties de règles concernent le fichier index.php qui est le *routeur* du site technote. Les arguments en première partie de règle deviennent des valeurs du tableau PHP $_GET, et ces valeurs seront utilisées par index.php afin de rediriger l'utilisateur sur la page web demandée.

  Pour reprendre l'exemple précédent, la deuxième partie de règle est :
  `index.php?url_controleur=main&url_page=$1&url_action=$2`.

  Dans cette partie, $1 représente le premier élément du tableau obtenu après décomposition de l'url (à savoir *"technote"*), et $2 le deuxième élément (*"add"*).

2.2 Le Routeur

Le fichier index.php fait office de routeur dans l'architecture du patron MVC. Après réécriture de l'url, index.php récupère dans les valeurs du tableau $_GET le contrôleur, l'action, la page, et éventuellement un id (si un objet précis de la base de données est demandé).

```
// Recuperation de l'URL
$controleur = ucfirst($_GET['url_controleur']);
$page = !empty($_GET['url_page']) ? $_GET['url_page'] : 'accueil';
$action = !empty($_GET['url_action']) ? $_GET['url_action'] : 'get';
$id = !empty($_GET['url_id']) ? $_GET['url_id'] : NULL;
```

Figure 3 – Extraction des paramètres de l'url dans le fichier index.php

La Figure 3 présente la récupération des valeurs du tableau $_GET dans les variables controleur, page, action, id.

Ces variables seront ensuite passées en paramètre à la méthode *chargerControleurPage* de la classe controleur associée (la variable controleur définissant le controleur appelé).

2.3 Main et Admin

Le site technote dispose de deux contrôleurs Main.php et Admin.php (qui héritent tous les deux de la superclasse Controleur dans laquelle est définie la méthode *chargerControleurPage*)

Usage des contrôleurs Les contrôleurs vont éxecuter une méthode particulière selon la page et l'action données dans l'url.

Par exemple, un utilisateur appelle la page contenant le formulaire d'ajout de technotes :

- index.php (après avoir extrait les informations de routage de l'url) appelle la méthode *chargerControleurPage* du controleur Main avec en paramètre page=*"technote"*, action=*"add"*.

- Le contrôleur Main vérifie qu'il possède bien une méthode nommée *technote*, puis appelle cette méthode avec pour paramètre action=*"add"*.
- Dans la méthode *technote* du contrôleur Main le bloc d'instructions correspondant à l'action *add* est exécuté.

Différences entre Main et Admin La différence entre les deux contrôleurs est simple, le contrôleur Main possède des méthodes pouvant être exécutées par tous visiteurs et membres du site technote, tandis que le contrôleur Admin possède des méthodes dont l'éxecution est strictement réservée aux administrateurs.

3 Modèles

Le modèles permettent de récupérer et de gérer les données de la base technote et contiennent également toutes les méthodes servant à implémenter les fonctionnalités du site technote.

Une partie des modèles repose sur deux super-classes, la classe DAO et la classe TableObject. Ainsi, à chaque table de la base technote correspond un fichier TableDAO.php et Table.php (ou Table est remplacée par le nom de la table en question) dans le répertoire `modeles`.

DAO La super-classe DAO (pour Data Access Object) s'occupe de l'interaction directe avec la base de données via la classe PDO. Elle implémente (elle et ses classes filles) les actions SQL de sélection, d'insertion, de mise à jour et de délétion via les méthodes get, save et delete.

TableObject La super-classe TableObject sert à formater les données récupérées grâce à la classe DAO en objets PHP propices à la manipulation. TableObject comprend un attribut *fields* de type *array* (qui stocke les attributs extraits des tuples de la base de données), ainsi que des méthodes d'accessions et de mutations des champs contenus dans le tableau *fields*.

La Figure 4 exemplifie l'utilisation des super-classes DAO et TableObject au travers des classes Technote et TechnoteDAO.

```
1 $technoteDAO = new TechnoteDAO(BDD::getInstancePDO()); // Creation d'une instance de
      la classe TechnoteDAO (classe fille de la super-classe DAO).
2
3 $technote = technoteDAO->getOne('12'); // Creation d'une instance de la classe
      Technote (classe fille de la super-classe TableObject).
4
5 echo 'ID de l\'auteur : ' . $technote->id_auteur; // Affichage de la valeur du champ
      id_auteur contenu dans le tableau fields (attribut de la classe Technote herite de
       TableObject).
```

Figure 4 – Exemple d'utilisation de modèles : Les classes Technote et TechnoteDAO

A la ligne 1, le constructeur de la classe TechnoteDAO prend en paramètre un objet de classe PDO (servant à la communication avec le SGBD MySQL) retourné par la méthode statique *getInstancePDO()* de la classe *BDD*.

La classe BDD contient l'unique méthode *getInstancePDO()*, qui lui permet d'instancier un objet PDO spécifiquement lier à la base de données associée au site technote.

Le paramétrage de la classe PDO (structures des données renvoyées, gestion des erreurs, ...) est également effectuée dans la classe BDD.

A la ligne 3, la méthode *getOne()* de la classe TechnoteDAO renvoie un objet Technote dont l'attribut *fields* est rempli avec le tuple vérifiant $id\_technote = 12$.

Remarque Les méthodes de la classe TechnoteDAO permettant l'accès à plusieurs tuples renverront un tableau d'objets Technote (contenant autant de champs que de tuples).

A la ligne 5, un accesseur en lecture récupère la valeur du champ *id_auteur* de l'objet Technote.

Remarque Il est à noter que les méthodes de la classe Technote (comme pour les autres classes filles de TableObject) ne se limitent à l'accès et à la mutation des champs de l'attribut *fields*. Elles permettent également la vérification de la forme et du contenu de l'information qui va être enregistrée dans la base de données (méthodes de *checking*).

Classes particulières Certaines classes du répertoire `modeles` ne sont ni filles de la classe DAO ni de TableObject. Ces classes permettent l'implémentation de fonctionnalités qui ne sont pas liées à la base de données. Par exemple, la classe *Mail* permet la gestion de l'envoi des mails (lors de l'inscription d'un membre, ou via le formulaire de contact) sur le site technote. La classe *Mail* est un modèle mais n'est pas liée à la base de données (les adresses mail des membres du site technote étant stockées dans la table membre et donc gérées par les classe Membre et MembreDAO).

4 Vues

Les vues définissent les gabarits HTML du site technote. Elles définissent la forme et le contenu des pages qui vont être présentées à l'utilisateur.

Afin de remédier au problème de l'itération (par exemple pour l'affichage de listes de technotes ou de commentaires) et de la réutilisation (chaque page du site possède la même entête) des contenus HTML, nous avons utilisé le moteur de template *Twig*.

La super-classe Vue admet un objet *Twig* en attribut permettant la manipulation des fichiers PHP (possédant l'extension .twig) contenus dans le répertoire `vues`.

Twig définit son propre langage qui permet une identification claire des objets PHP au sein des gabarits HTML.

Par exemple, la Figure 5 présente l'écriture d'une boucle PHP (écrite en respectant la syntaxe *Twig*) pour afficher tous les mots clés pouvant définir une technote à sa création. Les variables PHP manipulées dans un fichier .twig sont identifiées par des doubles accolades.

```
1 <select name="id_mot_cle[]" class="form-control" id="mots-cles" multiple="multiple">
2   {% for motCle in motsCles %}
3     <option value="{{ motCle.id_mot_cle }}">{{ motCle.label }}<option>
4   {% endfor %}
5 </select>
```

Figure 5 – Boucle sur des éléments HTML avec TWIG

5 La Base de Données

Les parties suivantes détaillent la conception de la base technote, depuis sa schématisation s'appuyant sur les instances du monde réel jusqu'à son implémentation sous le SGBD MySQL.

5.1 Modèle conceptuel de données

En partant des indications du cahier du charges, le modèle entité-association présenté en annexe 2 a été conçu. Les tables et leurs raisons d'être dans la base technote sont détaillées en sous-section 5.3. Une représentation complète et visuelle de la base de donnée technote se trouve en annexe 1.

5.2 Modèle relationnel

Membre	(id.membre, *pseudo*, *email*, *date.inscription*, *date.connexion*, *cle.reset.pass*, *bloquer*, #*id.groupe*)
Technote	(id.technote, *titre*, *date.creation*, *contenu*, *url.image*, *date.modification*, *description*, *visible*, *publié*, #*id.membre.auteur*, #*id.membre.modificateur*)
Mot cle	(id.mot.cle, *label*, *actif*)
Question	(id.question, *titre*, *question*, *date.question*, *date.modification*, *visible*, *resolu*, #*id.membre.auteur*, #*id.membre.modificateur*)
Reponse	(id.reponse, *reponse*, *date.reponse*, *date.modification*, *visible*, #*id.reponse.parent*, #*id.membre.auteur*,#*id.membre.modificateur*,#*id.question*)
Token	(id.token, *clé*, *date.expiration*, *ip*, *actif*, #*id.membre*)
Groupe	(id.groupe, *libellé*)
Action	(id.action, *libellé*, *date.action*, #*id.membre*)
Commentaire	(id.commentaire, *commentaire*, *date.creation*, *date.modification*, *visible*, #*id.membre.auteur*, #*id.membre.modificateur*,#*id.commentaire.parent*, #*id.technote*)
Clarifier	(#*id.mot.cle*, #*id.question*)
Décrire	(#*id.technote*, #*id.mot.cle*)
Droit Membre	(#*id.membre*, type, cible, *autoriser*)
Droit Groupe	(#*id.groupe*, type, cible, *autoriser*)

Figure 6 – Modèle relationnel de la base technote

5.3 Description des tables

Membre. La table Membre est centrale. Elle permet de stocker des informations sur les utilisateurs après leur inscription sur le site technote.

Groupe. La table Groupe représente les différents rôles pouvant être associés à un utilisateur, à savoir, *visiteur*, *membre*, *modérateur* et *administrateur*. L'appartenance à un groupe promulgue à l'utilisateur des droits, c'est à dire la possibilité d'effectuer certaines actions sur le site technote.

Technote. La table technote sert à stocker les "articles techniques" écrits par les utilisateurs du site.

Mot clé. Une technote est décrite par un ensemble de mots clés (ou aucun) contenu dans la table Mot Clé. De nouveaux mots clés peuvent être ajoutés par l'utilisateur, ainsi l'attribut booléen actif de la table mot_cle est à vrai après qu'un administrateur du site a validé l'ajout.

Droit Membre. La table Droit Membre permet le stockage des droits (actions pouvant être effectuées) pour chaque membre du site technote. L'attribut type de la table se décompose en quatre action *add*, *edit*, *drop*, *get*. Chaque action peut être appliquée à une cible qui correspond à une méthode de contrôleur. Par exemple, si le membre numéro 12 a perdu le droit d'écrire des technotes, la ligne suivante sera insérée dans la table Droit Membre :

id_membre	type	cible	autoriser
12	*add*	*technote*	false

Figure 7 – Tuple de la table Droit Membre

Les droits de la table Droit Membre sont prioritaires par rapport à la table Droit Groupe. Pour reprendre l'exemple précédent, le membre numéro a perdu le droit d'écrire des technotes, mais il appartient au groupe *Membre* qui lui a le droit d'écriture sur les technotes. Les droits du groupe Membre seront cachés par les droits individuels du membre numéro 12.

Droit Groupe. Les droits individuels des membres prévalant sur les droits des groupes, la table Droit Groupe permet uniquement d'initialiser les droits d'un membre lors de la création d'un compte sur le site technote. Aussi, la table Groupe contient un attribut *id_groupe_parent* qui permet d'établir une hiérarchie entre les groupes. Les droits sont hérités depuis le groupe le plus général (le groupe *visiteur*) jusqu'au groupe le plus spécifique (le groupe *administrateur*).

Réponse et Commentaire. Les tables réponse et commentaire sont réflexives. Cette propiété symbolise l'imbrication des commentaires ou réponses (indentation à l'affichage représentant la dépendance parent-enfant).

Action. La table action enregistre une historique des différentes actions effectuées par un utilisateur sur le site technote.

Token. La table token est nécessaire à la mise en place de la connexion active pour un utilisateur. La connexion active permet la connexion automatique d'un membre lors de son arrivée sur le site.

Attributs *"d'administration"* L'attribut *visible* des tables technote, question, reponse, et commentaire, l'attribut *bloquer* de la table membre, ou encore l'attribut *actif* de la table mot_cle, sont des attributs servant à l'administration du site technote. En effet, afin de prévoir tous débordements de la part de membres, ces attributs sont là pour pouvoir bloquer certains contenus, qui ne seront plus affichés sur le site.

Les membres du site ont un certain pouvoir d'administration (modification, non publication) sur les contenus qu'ils ont créés, et ce grâce aux mêmes attributs présents dans la base technote.

Fonctionnalités

6 Technologies annexes

6.1 CSS

Le framework CSS Bootstrap a été utilisé pour la définition du style du site technote. Bootstrap nous a permis de donner au site un style simple et élégant sans pour autant s'attarder sur l'écriture des feuilles CSS. Le site technote est responsive, son contenu s'adapte donc à la plateforme de consultation.

6.2 Javascript

Les fonctionnalités du site technote sont dynamisées et fluidifiées par la présence des librairies Javascript (positionnées dans le répertoire `assets`).

L'utilisation de la librairie *jQuery* s'est imposée sur le site technote, pour une question de commodité d'écriture des scripts, mais également pour une question de dépendances (la plupart des librairies Javascript ne fonctionnent pas sans *jQuery*).

Gestion des formulaires Les formulaires du site technote sont traités en Ajax grâce au *jQuery form plugin*, qui facilite la manipulation des formulaires. Le fichier *main.js* (localisé dans `assets/js`) contient les fonctions, entre autres, permettant la gestion des données de formulaires.

L'objet *ajaxForm* du *form plugin* permet d'effectuer des actions avant et après soumission du formulaire (appel de fonctions de callback pre et post submit).

Lors de la soumission, les données du tableau $_POST sont transmises au script PHP associé. Le script PHP va traiter les données (insertion, délétion, mise à jour de la base), puis va retourner une réponse sous la forme d'un objet au format JSON (utilisation de la fonction *json_encode()*).

Ainsi, l'objet ajaxForm va capturer l'objet JSON retourné (qui contient l'annonce du succès ou de l'échec de la soumission du formulaire) et va appelé la fonction de callback post-submit.

Selon le contenu de l'objet JSON retourné, la fonction post-submit peut afficher un message d'échec, de succès, ou ajouter un élément à la page (pour l'ajout de commentaires ou de réponses, par exemple), ...

Ecriture de contenus L'utilisateur peut mettre en forme les technotes, les questions, et les réponses qu'il écrit grâce à l'éditeur de texte fourni par le plugin Javascript *CKEditor*.

CKEditor permet de remplacer un élément de formulaire *textarea* par un éditeur de texte complet. CKEditor a été paramétré pour le site technote et permet d'insérer du code informatique (*code snippet*) au sein des contenus écrits par les utilisateurs.

6.3 Protection contre la faille CSRF

La faille CSRF (Cross-Site Request Forgery) est illustrée dans la situation suivante :

> Supposons que Bob soit l'administrateur d'un forum et qu'il soit connecté à celui-ci par un système de sessions. Alice est un membre de ce même forum, elle veut supprimer un des messages du forum. Comme elle n'a pas les droits nécessaires avec son compte, elle utilise celui de Bob grâce à une attaque de type CSRF.
>
> ◦ Alice arrive à connaitre le lien qui permet de supprimer le message en question.

- Alice envoie un message à Bob contenant une pseudo-image à afficher (qui est en fait un script). L'URL de l'image est le lien vers le script permettant de supprimer le message désiré.
- Bob lit le message d'Alice, son navigateur tente de récupérer le contenu de l'image. En faisant cela, le navigateur actionne le lien et supprime le message, il récupère une page web comme contenu pour l'image. Ne reconnaissant pas le type d'image associé, il n'affiche pas d'image et Bob ne sait pas qu'Alice vient de lui faire supprimer un message contre son gré.

(Wikipédia)

Pour remédier à cette faille, un jeton CSRF unique est calculé lors de la création d'une session PHP (donc à l'arrivée d'un utilisateur sur le site).

Ce jeton est ensuite positionné dans les formulaires du site technote dans un champ caché. Ainsi, si un membre envoie un lien permettant la suppression d'une technote à un administrateur, à la comparaison du jeton CSRF le formulaire de suppression sera rejeté.

Les fonctionnalités sont présentées pour chaque rôle que peut entreprendre un utilisateur du site technote. Le statut de l'utilisateur et les droits qui lui sont associés sont déterminés à l'arrivée sur le site (dans le fichier index.php). Après connexion, les droits et le statut de l'utilisateur sont mis à jour. Les actions utilisateur sont divisées en quatre catégories selon le type d'interaction avec la base de données technote : *get* pour la consultation, *edit* pour la mise à jour, *delete* pour la suppression, et *add* pour l'écriture.

7 Visiteur

Le rôle de visiteur est le plus restrictif. Un utilisateur reste visiteur tant qu'il ne s'est pas inscrit sur le site.

Un visiteur du site technote possède des droits de lecture sur les technotes (plus commentaires associés) et sur le forum de question. Il peut également lancer une recherche de technotes ou de questions.

Il ne peut accéder à aucun formulaire permettant l'écriture dans la base de données du site (sauf le formulaire d'inscription bien sûr).

8 Membre

En suivant la hiérarchie des rôles (voir la sous-section 5.3 sur les droits), un membre hérite de toutes les actions pouvant être effectuées par un visiteur. Voici la liste des actions supplémentaires acquises par l'utilisateur après inscription :

1. Modification de son profil de membre (edit).
2. Ajout de commentaires sur une technote (add).
3. Ajout de technotes, avec choix de publier ou non (add).
4. Ajout de questions/réponses (add).
5. Consultation de ses technotes non publiées (get).
6. Suppression par l'auteur des technotes qu'il a écrites (delete).
7. Modification par l'auteur des technotes qu'il a écrites (edit).
8. Ajout de mots clés à l'écriture d'une technote. Les mots clés ajoutés doivent être validés par un administrateur pour ensuite apparaître dans la liste (add).

Le membre accède à plusieurs actions qui vont lui permettre l'écriture de contenu, et sa publication sur le site.

9 Administrateur

Le groupe des administrateurs ont accès à des fonctionnalités supplémentaires concernant la gestion des membres et des contenus ajoutés par les membres. Un compte d'administration a été créé sur le site pour la démonstration des fonctionnalités. Le nom de compte est *admindemo* et le mot de passe *azerty34*.

Une fois connecté, un lien vers les fonctionnalités d'administration apparaît en haut à droite de la page.

1. Consultation de statistiques (get).
2. Recherche/Modification/Blocage/Suppression d'un compte membre (get, edit ou delete).
3. Recherche/Acceptation/Blocage de mots clés ajoutés par les utilisateurs (get ou edit).
4. Modification/Blocage/Suppression d'une technote (edit ou delete).
5. Modification/Blocage/Suppression d'une question/réponse (edit ou delete).
6. Modification/Blocage/Suppression d'un commentaire (edit ou delete).

Conclusion

Le site technote possède la majeure partie des fonctionnalités du cahier des charges. Il est pleinement fonctionnel à l'adresse `technote.calyxe.fr`.

Un utilisateur peut s'inscrire, écrire (ou consulter) des technotes, créer des sujets de discussions, et interagir avec d'autres utilisateurs librement (enfin, dans la limite de ses droits).

Aussi, un utilisateur administrateur/modérateur a accès à des informations statistiques et à des fonctionnalités de gestion du site.

L'architecture MVC procure au site technote une aisance d'extensibilité, ainsi de nombreuses autres fonctionnalités pourront être ajoutées.

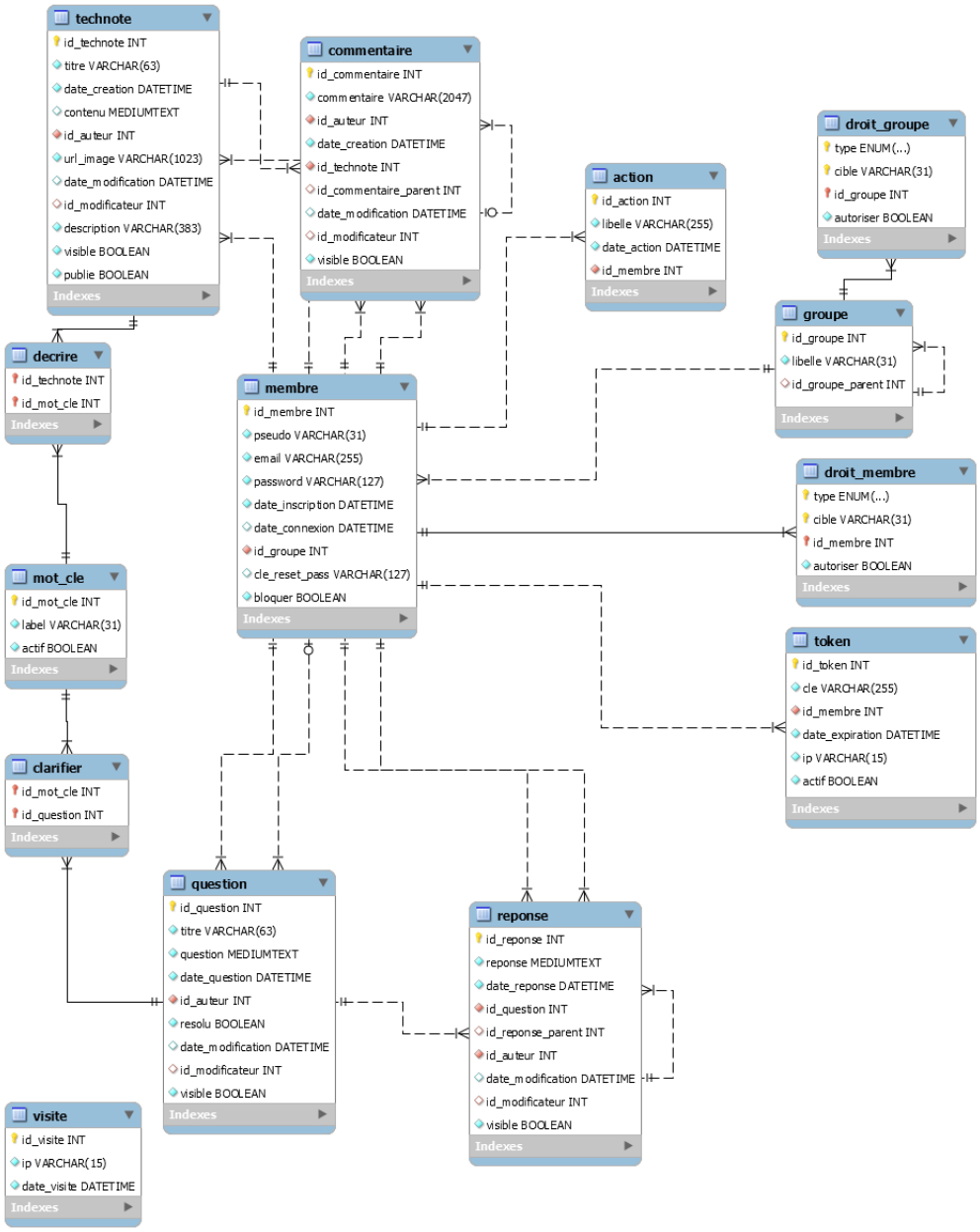
L'utilisation de librairies Javascript et CSS ajoute du dynamisme et de la fluidité à la navigation sur le site, en plus de permettre un déchargement du travail effectué côté serveur.

De plus, un intérêt a été porté à la sécurité des données avec l'implémentation d'une méthode de protection contre la faille CSRF.

Table des figures

Annexes

1 Schéma de la base technote générée avec MySQL Workbench



2 Diagramme Entité-association de la base technote