

الجمهورية الجزائرية الديمقراطية الشعبية
وزارة التعليم العالي و البحث العلمي
République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
جامعة محمد الصديق بن يحيى - جيجل-
Université Mohammed Seddik Benyahia-Jijel
Faculté des Sciences Exactes et Informatique
Département d'informatique



Mémoire de fin d'études
En vue de l'obtention du diplôme
Master en Informatique

Option : Réseaux et Sécurité

Thème

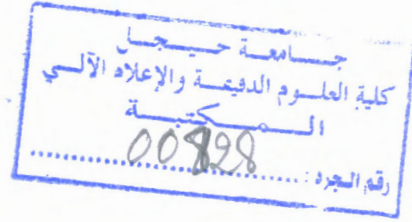
**LA SECURITE DES DONNEES DANS LES
RESEAUX : SAFE WEB APPLICATION**

Réalisé par :
AMIRECHE Souad

Encadrée par :
M. LAABINI MEROUANE

Année Universitaire 2018 - 2019

inf.RS 19/19



Mémoire de fin d'études
En vue de l'obtention du diplôme
Master en Informatique

Option : Réseaux et Sécurité

Thème

**LA SECURITE DES DONNEES DANS LES
RESEAUX : SAFE WEB APPLICATION**

Réalisé par :
AMIRECHE Souad

Encadrée par :
M. LAABINI MEROUANE



Année Universitaire 2018 - 2019

Dédicace

A la mémoire de mon défunt père, que Dieu le tout puissant puisse l'avoir en sa sainte miséricorde,

A ma mère, qui m'a soutenu et encouragé durant toutes ces années d'études,

A toute ma famille et mes proches pour leur soutien tout au long de mon parcours universitaire,

Que ce travail soit l'accomplissement de vos vœux tant allégués, et le fruit de votre soutien infailible,

Merci à vous tous !

Remerciements

Louange à Dieu le tout puissant qui par sa grâce, ce travail a pu voir le jour.

*Je tiens à exprimer toute ma gratitude à mon encadreur, **Monsieur LAABENI Marouane**.*

Merci à tous les enseignants, intervenants durant mes études et toutes les personnes qui par leurs paroles, leurs conseils et leurs critiques ont guidé mes réflexions et ont accepté de répondre à mes questions.

Merci à toute personne qui m'a aidé et soutenu.

Résumé/Summary

Résumé

Le présent mémoire a été rédigé dans le cadre du projet de fin d'études pour l'obtention du diplôme Master en Informatique option : Réseaux et Sécurité de l'université de Jijel.

Notre projet inclut une étude, une conception, une analyse et une implémentation d'un site Web sécurisé pour la gestion d'un journal électronique.

L'objectif majeur de ce site Web est de faciliter la publication sécurisée et protégée des articles rédigés par les différents rédacteurs du journal électronique.

La publication est gérée par un chef rédacteur qui a la responsabilité de bien vérifier le contenu des articles avant de les publier ou les faire revenir à leurs rédacteurs pour une quelconque modification ou correction.

L'architecture de la solution proposée est composée en plusieurs parties. La première étant une partie publique destinée aux lecteurs du site web. La deuxième est une partie destinée aux rédacteurs des articles afin qu'ils puissent produire du contenu pour le site Web. La troisième serait la partie du chef rédacteur qui serait responsable de la structure du journal électronique en gérant la liste des différentes rubriques, aussi il serait responsable de la publication du contenu du site Web. La quatrième consiste à un espace d'administration qui permet à l'administrateur du site Web de gérer les comptes et les profils des utilisateurs.

Pour la mise en place et la gestion de la base de données, MySQL est utilisé comme système de gestion de base de données et PHP comme le langage de programmation côté serveur, HTML 5, CSS 3 ainsi que JavaScript sont utilisés pour le côté client.

Mots clés : Site Web, Sécurité des données, Journal électronique, UML, MySQL, PHP, HTML 5, CSS 3, JavaScript.

Summary

This dissertation was written as part of the graduation project for the Master's Degree in Computer Science option: Networks and Security from the University of Jijel.

Our project includes a study, design, analysis and implementation of a secure website for the management of an electronic journal.

The main purpose of this website is to facilitate the secure and safe publication of articles written by the various editors of the electronic journal.

A chief editor manages the publication. He has the responsibility to check the content of the articles before publication. The chief editor can returned any article to their editor for any changes or corrections.

The architecture of the proposed solution is composed of several parts. The first is a public part for readers of the website. The second is a section for article writers to produce content for the website. The third would be the part of the chief editor who would be responsible for the structure of the electronic journal by managing the list of different sections, so he would be responsible for publishing the content of the website. The fourth is an administration area that allows the website administrator to manage accounts and user profiles.

For setting up and managing the database, MySQL is used as a database management system and PHP as the server-side programming language, HTML 5, CSS 3 as well as JavaScript are used for the client side.

Keywords: Web Site, Data Security, Electronic Journal, UML, MySQL, PHP, HTML 5, CSS 3, JavaScript.

Table des matières

Introduction	1
Chapitre 1 : Sécurité des Applications Web	3
1.1. Introduction	4
1.2. C'est quoi un réseau informatique ?	6
1.2.1. Intranet	6
1.2.2. Extranet	6
1.2.3. Internet	7
1.3. C'est quoi une application Web ?	7
1.3.1. Premier pas, c'est quoi le Web	7
1.3.2. Architecture client/serveur	8
1.3.3. Types d'architectures	8
1.3.4. Types de clients	9
1.4. Les risques à prévenir	9
1.4.1. L'abus de ressources	10
1.4.2. La destruction de données	10
1.4.3. La publication de données confidentielles	10
1.4.4. Le détournement du site Web	11
1.4.5. L'usurpation d'identité	12
1.5. Historique d'attaques pirates	12
1.6. Conclusion	14
Chapitre 2 : Techniques de sécurité Web	15
2.1. Introduction :	16
2.2. Sécurité de l'interface	16
2.2.1. Protection contre les XSS	16
2.2.2. Protection contre les CSRF	22
2.2.3. Protection des formulaires	26
2.2.4. Protection des sessions	31
2.2. Sécurité du code source	33
2.2.1. Protection contre l'injection de code distant	34
2.2.2. Protection des téléchargements	36
2.3. Sécurité de la base de données	37
2.3.1. Protection contre les injections SQL	37
2.3.2. Protection des tables et des colonnes	39
2.4. Protection de l'accès à la base de données	39

2.4.1. Arrangement des accès au serveur MySQL	39
2.4.2. Gestion des droits d'accès au serveur MySQL	40
2.5. Protection des données d'authentification	40
2.5.1. Cryptage des données	40
2.5.2. Camouflage des données	41
2.6. Conclusion	41
Chapitre 3 : Conception et Architecture	43
3.1. Introduction	44
3.2. Outils à utiliser	44
3.2.1. Annotation UML	44
3.2.2. XMind	47
3.2.3. Modelio	47
3.2.4. Toad Data Modeler	47
3.3. La conception du système.....	48
3.3.1. Réalisation d'un journal électronique	48
3.3.2. Description du système.....	48
3.4. Fonctionnement du système	49
3.4.1. Définition des acteurs	49
3.4.2. Interaction utilisateur-système.....	50
3.5. Architecture générale.....	54
3.5.1. Cycle de vie d'un article.....	54
3.5.2. Diagramme de concepts saillants	54
3.6. Architecture détaillée.....	56
3.6.1. Diagramme de classe	56
3.6.2. Augmentation de la sécurité	62
3.7. Conclusion	65
Chapitre 4 : Implémentation	66
4.1. Introduction	67
4.2. Choix technique.....	67
4.2.1. HTML.....	67
4.2.2. CSS	67
4.2.3. JavaScript	67
4.2.4. jQuery	68

4.2.5. Ajax	69
4.2.6. PHP	69
4.2.7. SQL	69
4.3. Environnement logiciel	70
4.3.1. WAMPsServer	70
4.3.2. Visual Studio Code.....	70
4.4. Environnement matériel	71
4.5. Un petit complément de sécurité	71
4.5.1. Gestion des droits d'accès à la base de données	71
4.5.2. Création des vues.....	77
4.6. Interfaces du site Web	80
4.7. Conclusion.....	82
Conclusion et perspectives	83
Bibliographie	85

Table des figures

Figure 2-1 : Exemple de Captcha	29
Figure 2-2 : Exemple de CAPTCHA visuel	30
Figure 3-1 : Diagramme de cas d'utilisation de la gestion des rubriques.....	52
Figure 3-2 : Diagramme de cas d'utilisation de la gestion des comptes	52
Figure 3-3 : Diagramme de cas d'utilisation de la gestion des articles	53
Figure 3.4 : Cycle de vie d'un article	54
Figure 3.5 : Les concepts saillants du système	54
Figure 4-1 : Authentification sous phpMyAdmin	72
Figure 4-2 : la base de données std_journal_db	73
Figure 4-3 : Les utilisateurs de la base de données std_journal_db	73
Figure 4-4 : Ajout d'un compte utilisateur	74
Figure 4-5 : Les privilèges d'un compte utilisateur.....	74
Figure 4-6 : Sauvegarde des données du compte utilisateur	75
Figure 4-7 : La table utilisateur	75
Figure 4-8 : Les pouvoirs d'un compte utilisateur	76
Figure 4-9 : Les pouvoirs d'un compte administrateur	77
Figure 4-10 : Les pouvoir d'un compte visiteur	77
Figure 4-11 : Accueil du site Web « Icosium News ».....	80
Figure 4-12 : Liste des articles par rubrique.....	80
Figure 4-13 : Lecture d'un article.....	81

Liste des tableaux

<i>Table 3-1 : Exemple de séquence de création des diagrammes</i>	46
<i>Table 3.2 : Les acteurs du système</i>	49
<i>Table 3.3 : Associations de la classe "Rubric"</i>	55
<i>Table 3.4 : Associations de la classe "Article"</i>	55
<i>Table 3.5 : Associations de la classe "Account"</i>	55
<i>Table 3.6 : Associations de la classe "Profil"</i>	56
<i>Table 3.7 : Attributs de la classe "Rubric"</i>	57
<i>Table 3.8 : Méthodes de la classe "Rubric"</i>	57
<i>Table 3.9 : Attributs de la classe "Article"</i>	58
<i>Table 3.10 : Méthodes de la classe "Article"</i>	59
<i>Table 3.11 : Attributs de la classe "Account"</i>	60
<i>Table 3.12 : Méthodes de la classe "Account"</i>	61
<i>Table 3.13 : Attributs de la classe "Profile"</i>	61
<i>Table 3.14 : Méthodes de la classe "Profile"</i>	62

Introduction générale

Pendant des années, les experts en sécurité ont mis en garde contre les vulnérabilités des applications Web. Ces avertissements sont malheureusement en train de se concrétiser.

L'actualité de nos jours est dominée par les nouvelles d'un tel ou tel pirate informatique infiltrant avec succès une telle ou telle application Web.

Les forums clandestins sont en pleine effervescence lorsque ces criminels électroniques partagent leurs découvertes de vulnérabilité, leurs réussites et leurs recherches pour leur prochaine cible.

Les pirates informatiques l'ont prouvé, nous ne pouvons sécuriser à cent pour cent les applications Web. Donc que faire pour sécuriser ces applications souvent importantes et sensibles ?

Quand il s'agit d'un crime informatique, l'étendue des dégâts peuvent être des fois qualifiée d'un vrai effet papillon qui commence par une simple opération d'infiltration et qui se termine par le licenciement de milliers d'employés.

Néanmoins, il est primordial avant tout de bien spécifier et comprendre le périmètre de notre étude.

Hormis cette introduction et la conclusion générale, ce mémoire s'articule en quatre chapitres :

- Le premier chapitre abordera des généralités sur la sécurité des applications web.
- Le second chapitre est destiné à la présentation de quelques techniques de la sécurité web.
- Dans le chapitre trois, on prendra en détail la conception de notre site Web qui consiste en un journal électronique, tout en ajoutant des atouts renforçant sa sécurité.
- Le dernier chapitre sera consacré à l'environnement de programmation ainsi que l'ensemble des logiciels que nous avons choisi pour la réalisation de l'application et l'implémentation de la base de données. Ensuite, nous allons présenter en aperçu les interfaces les plus importantes du notre journal électronique.

Chapitre 1 : Sécurité des Applications Web

1.1. Introduction

Il y a à peine dix ans, Internet était encore un gadget pour informaticiens, cher, lent, bruyant et laid. Pourtant, c'était déjà une source incroyable d'informations.

Aujourd'hui, le paysage a bien changé. Et il faut reconnaître qu'Internet a envahi notre vie quotidienne à une vitesse incroyable. 15 % des internautes avouent même qu'ils ont une forme de dépendance à cette technologie. Le Web a envahi le téléphone, la radio, la télévision et lorgne maintenant le cinéma.

Vacances, actualités, technologies, finances en ligne, commerce électronique ou encore échange d'opinions et partage des différences ; notre vie passe de plus en plus par le réseau. On y a une identité propre, des habitudes. Nombre de fournisseurs de services réduisent leurs coûts en supprimant les services traditionnels pour les remplacer par des échanges électroniques. Les gouvernements se rapprochent de leurs administrés en mettant en place des services de proximité, à la fois plus ouverts et plus rapides.

Pourtant, le Web d'aujourd'hui est toujours construit sur une confiance réciproque des internautes telle qu'elle prévalait à ses débuts. Des sites d'importance font encore confiance à leurs utilisateurs pour mettre en lumière les meilleures actualités (<http://www.digg.com>, par exemple), pour répondre à des questions (Yahoo! Answers, par exemple) ou pour rassembler les connaissances à travers le monde (<http://www.wikipedia.org>, par exemple).

Cette approche collaborative permet de transformer Internet en un forum mondial où les internautes les plus éloignés peuvent se retrouver et former des communautés en fonction de leurs goûts et aspirations.

Malheureusement, ce qui rapproche les internautes est aussi ce qui met en péril votre existence numérique. Votre modem est la porte d'entrée de votre demeure et derrière cette porte, tout le monde est voisin. Il n'y a aucune limitation pour solliciter un serveur ou un autre dans le monde. Et certains ne se gênent pas pour aller frapper à toutes les portes afin de voir qui va ouvrir.

La sécurité informatique devient donc un sujet de premier plan puisqu'elle affecte directement le fonctionnement des applications en ligne. Qui dit sécurité fait souvent surgir l'image des ordinateurs de l'armée qui contrôlent les missiles intercontinentaux ou encore des écrans remplis de caractères qui défilent contenant des informations vitales mais chiffrées. Notre

identité numérique n'est pas aussi bien protégée, tout en prenant une valeur qui n'est plus négligeable.

Prenons un exemple. Un guide de restaurants est intéressant si chaque commentaire a pu être déposé par un visiteur qui y a mangé. On peut alors avoir une idée satisfaisante des points forts et des lacunes du service. Cependant, que se passe-t-il quand un spammeur vient proposer des milliers de commentaires élogieux, afin de faire valoir une adresse particulière ? Cela fausse tout l'intérêt des retours d'expérience.

L'intérêt pour ce type de détournement est maintenant décuplé, tant par la variété des applications, que par les participations de chacun. Que devient un forum politique où les partisans d'un camp arrivent à noyer les arguments des autres dans un flot de contre-arguments ? Que devient MySpace quand un million d'utilisateurs choisissent Samy comme leur héros ? Que se passe-t-il si Mastercard se fait soutirer 40 millions de numéros de cartes de crédit ? Et si les impôts se font voler 120 000 dossiers complets ? Le défi des applications web modernes repose sur la maîtrise de ces flots d'informations.

Il faut savoir séparer les informations utiles du bruit ambiant dans lequel on retrouve les déchets classiques de la société de l'information mais aussi tout une population nouvelle d'opportunistes : robots automatiques, vulnérabilités des technologies, armées de zombies et utilisateurs inconscients viennent s'ajouter aux fanatiques, aux mauvaises langues et aux simples d'esprits.

De nos jours, la sécurité d'une application web repose d'abord sur la conscience des développeurs. Avoir opté pour PHP et MySQL est déjà un excellent choix stratégique : la communauté comme le groupe de développement sait à quel point la sécurité est le fer de lance des applications qui réussissent.

Dans ce chapitre, et afin de bien montrer les enjeux de la sécurité Web il serait clarifié les points suivants :

- 1- Le réseau informatique ;
- 2- L'application Web ;
- 3- Les risques qu'il faut prévenir ;
- 4- Un historique des plus importantes attaques pirates.

1.2. C'est quoi un réseau informatique ?

Un réseau informatique est un ensemble d'équipements reliés entre eux pour échanger des informations. Par analogie avec un filet (un réseau est un « petit rets », c'est-à-dire un petit filet), on appelle nœud l'extrémité d'une connexion, qui peut être une intersection de plusieurs connexions ou équipements (un ordinateur, un routeur, un concentrateur, un commutateur).

Indépendamment de la technologie sous-jacente, on porte généralement une vue matricielle sur ce qu'est un réseau.

De façon horizontale, un réseau est une strate de trois couches : les infrastructures, les fonctions de contrôle et de commande, les services rendus à l'utilisateur. De façon verticale, on utilise souvent un découpage géographique : réseau local, réseau d'accès et réseau d'interconnexion.

Un réseau peut être classé en fonction de son utilisation et des services qu'il offre. Ce découpage recoupe également la notion d'échelle. Ainsi, pour les réseaux utilisant les technologies Internet (famille des protocoles TCP/IP), la nomenclature est la suivante [09] :

1.2.1. Intranet

Un intranet est un réseau informatique privé utilisé par les employés d'une entreprise ou de toute autre entité organisationnelle et qui utilise les mêmes protocoles qu'Internet (TCP, IP, HTTP, SMTP, IMAP, etc.). Cette utilisation n'est pas nécessairement locale, un intranet pouvant s'étendre à travers le WAN.

Parfois, le terme se réfère uniquement au site web interne de l'organisation, mais c'est souvent une partie bien plus importante de l'infrastructure informatique d'une organisation.

1.2.2. Extranet

L'Extranet est l'utilisation du réseau internet dans laquelle une organisation structure ce réseau pour s'interconnecter avec ses partenaires commerciaux ou ses parties prenantes.

Par opposition, un réseau intranet, se limite au réseau interne à l'organisation, sans utiliser d'infrastructure tierce (publique).

La liste de sécurité est l'ensemble des données regroupant les identifiants (nom d'utilisateur (login), adresse IP, adresses MAC, clefs logiques ou physiques) autorisés à se connecter.

Réseau informatique à caractère commercial, constitué des intranets de plusieurs entreprises qui communiquent entre elles, à travers le réseau Internet, au moyen d'un serveur Web sécurisé. Par extension, désigne plus généralement les sites à accès sécurisé permettant à une entreprise de n'autoriser sa consultation qu'à certaines catégories d'intervenants externes, ses clients ou ses fournisseurs en général.

Exemple de réseau Extranet : Le réseau automobile European Network Exchange (ENX) destiné à sécuriser les échanges de données entre constructeurs et sous-traitants automobiles en Europe.

1.2.3. Internet

Internet est le réseau informatique mondial accessible au public. C'est un réseau de réseaux, à commutation de paquets, sans centre névralgique, composé de millions de réseaux aussi bien publics que privés, universitaires, commerciaux et gouvernementaux, eux-mêmes regroupés en réseaux autonomes (il y en avait plus de 91 000 en 2019). L'information est transmise via Internet grâce à un ensemble standardisé de protocoles de transfert de données, qui permet des applications variées comme le courrier électronique, la messagerie instantanée, le pair-à-pair et le World Wide Web. Internet a été popularisé par l'apparition du World Wide Web.

Un internaute est une personne qui utilise un accès à internet. Cet accès peut être obtenu grâce à un fournisseur d'accès via divers moyens de communication électronique : soit filaire (réseau téléphonique commuté (bas débit), ADSL, fibre optique jusqu'au domicile), soit sans fil (WiMAX, par satellite, 3G+, 4G, ou 5G).

1.3. C'est quoi une application Web ?

En informatique, une application web (aussi appelée web application, de l'anglais) est une application manipulable directement en ligne grâce à un client web (Les navigateur Web par exemple). Les sites web peuvent être qualifiés comme étant des applications web est généralement installée sur un serveur et se manipule en actionnant des widgets à l'aide d'un navigateur web, via un réseau informatique (Internet, intranet, réseau local, etc.). [09]

Les applications web offrent de nombreux usages dont les plus populaires sont les moteurs de recherche, le webmail, le commerce électronique... etc.

1.3.1. Premier pas, c'est quoi le Web

Le World Wide Web (WWW), communément appelé le Web, est un système hypertexte public fonctionnant sur le réseau mondiale Internet. Le Web permet de consulter, avec un navigateur (client Web), des pages accessibles sur des sites.

Le Web n'est que l'un des services offre via Internet. Nous pouvons citer aussi le courrier électronique, la messagerie instantanée et le partage de fichiers en pair à pair etc. [09]

1.3.2. Architecture client/serveur

L'environnement client-serveur désigne un mode de communication à travers un réseau entre plusieurs programmes : l'un, qualifié de client, envoie des requêtes ; l'autre ou les autres, qualifiés de serveurs. Par extension, le client désigne également l'ordinateur ou la machine virtuelle sur lequel est exécuté le logiciel client (un navigateur Web par exemple), et le serveur, l'ordinateur ou la machine virtuelle sur lequel est exécuté le logiciel serveur. Les serveurs sont des ordinateurs généralement destinés au logiciel serveur qu'ils abritent, et dotés de capacités supérieures qui leur permettent de répondre aux requêtes d'un grand nombre de clients.

Il existe une grande variété de logiciels serveurs et de logiciels clients en fonction des besoins à servir : un serveur Web publie des pages Web demandées par des navigateurs Web ; un serveur de messagerie électronique envoie du courriel à des clients de messagerie ; un serveur de fichiers permet de partager des fichiers sur un réseau ; un serveur de base de données permet de récupérer des données stockées dans une base de données, etc. [09]

1.3.3. Types d'architectures

i) Architecture à deux niveaux

Une architecture à deux niveaux ou une architecture deux tiers (two-tier architecture en anglais) est un environnement client-serveur où le client demande une ressource au serveur qui la fournit à partir de ses propres ressources.

ii) Architecture à trois niveaux

Une architecture à trois niveaux ou une architecture trois tiers (three-tier architecture en anglais) ajoute un niveau supplémentaire à l'architecture à 2 niveaux, permettant de spécialiser les serveurs dans une tâche précise, ce qui donne un avantage de flexibilité, de sécurité et de performance : un client qui demande une ressource via une interface utilisateur (généralement un navigateur web) chargée de la présentation de la ressource ; un serveur d'application (appelé

middleware) qui fournit la ressource, mais en faisant appel aux ressources d'un autre serveur ; un serveur de données qui fournit au serveur d'application les ressources requises pour répondre au client.

iii) Architecture à N niveaux

Une architecture à N niveaux ou architecture N tiers (N-tier architecture en anglais) n'ajoute pas encore des niveaux supplémentaires à l'architecture à 3 niveaux, mais introduit la notion des objets qui offre la possibilité de distribuer les services entre les 3 niveaux selon N couches, permettant ainsi de spécialiser les serveurs davantage.

1.3.4. Types de clients

i) Le client léger

Un client léger est une application où le traitement des requêtes du client (applications Web n'utilisant pas ou peu de JavaScript côté client, terminaux Terminal Services, Secure Shell, Apple Remote Desktop, Citrix XenApp, TeamViewer, etc.) est entièrement effectué par le serveur, le client recevant les réponses « toutes faites ».

ii) Le client lourd

Un client lourd est une application où le traitement des requêtes du client (applications de bureau, applications mobile) est partagé entre le serveur et le client.

iii) Le client riche

Un client riche est une application où le traitement des requêtes du client (applications Web utilisant beaucoup de JavaScript côté client) est effectué majoritairement par le serveur, le client recevant les réponses « semi-finies » et les finalisant. C'est un client léger plus évolué permettant de mettre en œuvre des fonctionnalités comparables à celles d'un client lourd.

1.4. Les risques à prévenir

Créer un site web peut sembler être une opération anodine et quotidienne : plus un objet est utile, moins il est perçu comme une source de danger. Les risques sont pourtant réels de voir son site abusé et détourné de son utilisation initiale : cela arrive aux grandes entreprises, aux sites populaires et même aux sites personnels. C'est là que la sécurité entre en jeu.

1.4.1. L'abus de ressources

L'abus de ressources est généralement le premier souci des webmasters. Cette attaque consiste tout simplement à accaparer tout ou partie des ressources d'un site web pour en bloquer le fonctionnement. Cela conduit au déni de service : les ressources sont devenues tellement rares que le serveur ne peut plus assurer sa fonction habituelle.

Les abus de ressources se produisent suite à une sollicitation trop importante de la mémoire, du processeur, des connexions aux bases de données, de la bande passante, du nombre de processus du serveur web, ou encore de l'espace disque. En fait, tous les aspects du serveur peuvent être saturés de l'extérieur.

Ce type d'abus est généralement facile à identifier, car il pose immédiatement des problèmes à l'administrateur. C'est aussi un des abus les mieux connus et pris en compte, aussi bien au niveau de la configuration PHP que du développement en PHP et MySQL.

Toutefois, il existe aussi des situations où l'abus est plus discret : par exemple, un robot IRC (est *un ensemble de scripts ou un programme indépendant permettant d'utiliser les fonctions du protocole IRC de manière automatisée*) [1] qui veille sur le serveur, en attente d'instructions spécifiques. Durant toute la veille, il est tellement discret qu'on a du mal à le repérer, jusqu'au moment où il se révèle. Mais à ce moment-là, il est déjà trop tard.

1.4.2. La destruction de données

La destruction de données arrive en deuxième sur la liste des risques de sécurité les plus connus. Il s'agit de s'attaquer aux données, en utilisant une faille de l'application web. Par exemple, à l'aide d'une injection SQL, il est possible d'effacer le contenu d'une table, ou bien de modifier des données dans cette table, et de rendre l'ensemble du site web inutilisable (remplacer tous les mots de passe de la table des utilisateurs par le même mot, par exemple).

Comme l'abus de ressources, la destruction de données est un phénomène largement identifié par les programmeurs et généralement bien défendu. Tout au moins, quelques défenses sont mises en place pour parer aux problèmes les plus évidents. Il s'agit là aussi d'un type d'attaque facile à repérer sur le serveur. [1]

1.4.3. La publication de données confidentielles

La publication de données confidentielles entre dans la catégorie des risques qui ne mettent

plus en péril l'application web elle-même. Il s'agit simplement d'un accès par un pirate à des données auxquelles il ne devrait pas pouvoir accéder : par exemple, lire le profil d'un utilisateur qui souhaite rester confidentiel, le dossier médical d'un patient ou encore le fichier fiscal d'une entreprise.

La publication de données ne perturbe pas l'utilisation d'un site. Prenons l'exemple de MasterCard qui s'est fait voler quelques millions de numéros de cartes de crédit en 2005. MasterCard est toujours en possession des données, mais désormais, une autre personne en dispose également. Or, la sécurité d'une carte de crédit tient beaucoup à la confidentialité du nom du porteur, du numéro de la carte et de la date d'échéance. Avec ces trois informations, il est possible d'utiliser une carte à la place du propriétaire légitime. [1]

Toute la difficulté de ce type de risque est de savoir identifier le vol. Si le pirate a réussi à détourner une requête SQL pour afficher toutes les lignes d'une table, cela va laisser beaucoup moins de traces sur le serveur que l'effacement total d'une table.

1.4.4. Le détournement du site Web

Détourner un site revient à s'en servir dans un but différent de son objet initial. Prenons l'exemple du blog : beaucoup l'utilisent comme un journal public où sont consignés leurs coups de cœur ainsi que les sujets qui les révoltent ou qu'ils désapprouvent. Si le nombre de visiteurs du blog est suffisamment important pour lui assurer un bon référencement, ce dernier devient crédible et respecté. Mais si un pirate s'introduit sur le système pour insérer son propre message, le site sera détourné : le pirate va exploiter la crédibilité du blog pour diffuser à grande échelle un message de spam, une promotion éhontée ou encore d'autres messages illégaux... C'est un problème que rencontrent fréquemment les blogs, les forums ou les sites d'actualités.

Un autre type de détournement survient quand une fonctionnalité est exploitée autrement que pour son but initial. Par exemple, une interface web proposant le service Whois (*est un service de recherche fourni par les registres Internet permettant d'obtenir des informations sur une adresse IP ou sur un nom de domaine*) [1] peut facilement être détournée par un pirate pour collecter des informations. Le pirate demande au site le Whois d'un domaine et c'est le site qui effectue la demande auprès des registres Internet, masquant ainsi l'identité du véritable demandeur. Si un tel site est mal protégé contre les abus, il va faire le relais entre les volumineuses demandes du pirate et les serveurs Internet, pour se voir finalement interdit

d'accès. Le pirate, lui, ne sera pas inquiété...

Les moteurs de recherche peuvent aussi être victime de ce type d'attaques : en effet, ils suivent toutes les URL qui leur sont fournies afin de pouvoir indexer le Web. Malheureusement, si une URL a été créée dans le but de nuire à un autre site, le moteur de recherche va être l'instrument d'une catastrophe sans le savoir, ni pouvoir remonter jusqu'à l'auteur. Les serveurs peuvent également se faire détourner. Dans ce cas, un simple site web peut se transformer en un site FTP pirate ou bien un zombie utilisé dans un déni de service distribué.

1.4.5. L'usurpation d'identité

L'usurpation d'identité est un problème croissant sur Internet. Le respect de l'anonymat est toujours un débat récurrent, mais de nombreux sites requièrent une forme d'identification avant de donner un accès plus complet à leurs services. C'est encore plus vrai pour les sites commerciaux, où le client paie pour un service qui lui est exclusif. [1]

L'identité que l'on utilise sur un site web devient donc un instrument important de sécurité, d'autant plus que différents droits sont attachés à cette identité. Parfois même, l'identité est la seule protection possible sur un site. Il suffit de voler l'identifiant et le mot de passe d'une personne pour prendre sa place. Dans d'autres cas, c'est simplement le cookie ou la session d'un utilisateur qui suffit.

1.5. Historique d'attaques pirates

Nous citons ici quelques attaques récentes [10] :

2007 (23 août) : Un homme britannique est arrêté pour l'utilisation non autorisée d'une connexion sans fil à Chiswick, Londres.

2007 (18 décembre) : Aux Etats-Unis, HarioTandiwidjojo, un ancien consultant informatique, plaide coupable pour avoir accédé sans autorisation à des ordinateurs protégés, après avoir piraté plus d'une soixantaine de stands commerciaux et dérobé les données des cartes de crédits utilisées.

2008 (11 juin) : Aux Etats-Unis, Robert Matthew Bentley est condamné à 41 mois de prison et à une amende 65 000 dollars en compensation pour s'être introduit dans des systèmes informatiques en Europe (notamment dans ceux de Rubbermaid) et de les avoir utilisés pour former un botnet.

2008 (11 juillet) : En Chine, Yang Litao est condamné à deux ans de prison pour avoir piraté le site Web de la Croix-Rouge et essayé de détourner des dons d'aide humanitaire vers un compte bancaire qu'il contrôlait (suite tremblement de terre dans la région du Sichuan).

2008 (05 Novembre) : Aux Etats-Unis, Ivan Biltse, Angelina Kitaeva et Yuriy Rakushchynets (connu sous le nom Yuriy Ryabinin) plaident coupable pour complot et fraude, et reconnaissent avoir participé à un projet qui utilisait les données de cartes bancaires volées de Citibank afin de dérober 2 millions de dollars. Le groupe, qui comprenait sept autres personnes accusées plus tôt dans l'année, s'était introduit dans un serveur qui traitait les opérations de distributeurs automatiques 7-eleven.

2009 (05 mars) : Le gang qui avait tenté de dérober, sans succès, 229 millions de dollars dans les locaux londoniens de la Sumitomo Bank en 2004 est envoyé en prison. Grâce à un contact interne, les hackers s'étaient introduits illégalement dans la banque et avaient installé un logiciel d'enregistrement de frappe pour obtenir des coordonnées bancaires et pouvoir transférer de l'argent sur des comptes à l'étranger. Les deux hackers, Jan van Osselaer et Gilles Poelvoorde, sont condamnés respectivement à trois ans et demi et quatre ans de prison. Kevin O'Donoghue, leur contact interne, est condamné à quatre ans et quatre mois de prison. Hugh Rodley et David Nash, qui avaient organisé les comptes bancaires internationaux, sont condamnés respectivement à huit et trois ans de prison.

2009 (28 août) : Albert Gonzalez accepte de plaider coupable pour 19 chefs d'accusation de fraude, complot, vol d'identité aggravé et blanchiment d'argent en lien avec le vol de plus de 170 millions de données bancaires à partir de comptes TJX, Barnes & Noble, Office Max et autres. Selon les termes de l'accord, Albert Gonzalez effectuera entre 15 et 25 ans de prison et devra payer une amende de plus de 2,8 millions de dollars.

2010 (17 février) : A Moscou, des hackers remplacent des vidéos commerciales par des vidéos pornographiques sur un panneau d'affichage.

2010 (24 février) : Des hackers divulguent des informations relatives aux finances de banques lettones et d'entreprises nationales à la télévision lettone.

1.6. Conclusion

La sécurité d'une application web est étroitement liée à l'environnement qui l'héberge. Des failles telles le cross-site scripting peuvent mettre l'application en péril si le développeur n'a pas pris de précaution pour sécuriser son code. Différentes façons d'attaquer une application Web peuvent être utilisées, par exemple l'injection SQL ou JavaScript dont nous allons les aborder avec plus de détails dans le chapitre suivant.

Chapitre 2 : Techniques de sécurité Web

2.1. Introduction :

Avant de concevoir un système informatique, il est nécessaire d'effectuer une étude préliminaire dont le but est l'identification des besoins et exigences liées à la sécurité de notre futur système.

PHP et MySQL constituent la plate-forme la plus populaire pour produire des applications web. Celle-ci est bien sécurisée contre les attaques. Elle a la charge de produire une application web et d'en sécuriser le code, c'est-à-dire que les pages HTML créées à partir de PHP et des données de l'utilisateur ne doivent donner que le résultat attendu.

Les vulnérabilités web ont été identifiées comme les plus nombreuses en 2006 et seront encore en tête dans les prochaines années : entre la facilité d'accès via le réseau Internet, la popularité des applications et la négligence de certains webmestres, il y a de nombreuses occasions pour exploiter une application en ligne.

Dans ce chapitre, nous passerons en revue les risques encourus par les applications web. Nous présenterons les protections possibles à mettre en place avec PHP et MySQL.

2.2. Sécurité de l'interface

2.2.1. Protection contre les XSS

XSS est un sigle anglophone, qui signifie Cross-Site Scripting, pour lequel il n'y a pas encore d'équivalent français reconnu. C'est dommage, car cela aiderait peut-être à mieux comprendre ce qu'est un XSS et donc à mieux s'en défendre. La définition la plus représentative qui existe actuellement est « injection HTML ».

Historiquement, les premières attaques XSS remontent à l'époque des balises <frame>. Les frames, ou cadres, servent à découper une page web en plusieurs zones, chacune d'entre elles étant gérée par une page web indépendante. Pour garder une unité à la page, JavaScript se charge de faire communiquer les zones entre elles.

Avec cette structure, il est difficile de voir si une telle page provient du site consulté ou bien d'un site externe. En effet, la page principale définit les cadres et leurs URL respectives. Comme pour les images, un cadre peut être local ou bien placé sur un site distant.

Après le chargement, chaque cadre gère sa propre navigation, ses images, son code JavaScript. On peut donc naviguer sur un site et charger des pages sur autre site de façon totalement transparente.

C'est sur ce camouflage que les premières attaques XSS étaient basées. En modifiant le code de la page web contenant les cadres, un pirate parvenait à charger une page de son propre site. L'utilisateur sans méfiance croyait être toujours sur le site qu'il visitait initialement, alors qu'en fait il naviguait sur les pages malintentionnées. Le pirate utilisait

alors la crédibilité du site web attaqué et la crédulité de l'utilisateur pour amener ce dernier à lui confier des informations confidentielles, comme des identifiants ou des coordonnées bancaires.

Le terme XSS a été forgé à partir de ces premières techniques : scripting pour l'utilisation de JavaScript ou de techniques HTML et cross-site pour indiquer le mélange qui était réalisé entre le site web vulnérable et le site pirate.

a. Prototype d'une injection HTML

Le concept initial d'une XSS est la possibilité de faire une injection de code HTML ou JavaScript dans une page HTML. Le site est vulnérable dans la mesure où il permet à un utilisateur externe de modifier le comportement d'une page web à l'aide des arguments qui sont envoyés à cette dernière. L'archétype d'une vulnérabilité XSS est celui-ci :

```
<html>
  <head>
    <title>Une page vulnérable</title>
  </head>
  <body>
    <?php echo "Bonjour ".$_GET['nom'] ; ?>
  </body>
</html>[6]
```

Vous reconnaissez facilement la structure de base d'une page HTML très dépouillée. La variable « nom » est passée via l'URL comme ceci :

```
http://www.monsite.com/index.php?nom=damien
```

La variable « nom » est affichée directement dans la page, via la concaténation.

`$_GET['nom']` est fournie par PHP à partir des données de l'URL et, dans ce script d'illustration, il est utilisé brut : il n'y a aucune modification entre la valeur externe et celle qui est introduite dans la chaîne affichée. Généralement, une telle page est utilisée comme action d'un formulaire et permet d'afficher le nom de l'utilisateur et de personnaliser la page.[6]

La vulnérabilité présentée dans cette page réside dans le fait qu'il est possible d'utiliser des caractères HTML spéciaux, tels que `<` et `>`, pour modifier le comportement de la page HTML produite. Le code peut faire plus que simplement afficher la variable envoyée et le nom de l'utilisateur. Observez cette URL, pointée sur le script ci-dessus :

```
http://www.monsite.com/index.php?nom=%3Cb%3Emonsieur%3C%2Fb%3E
```

Cette URL inclut maintenant deux balises HTML. Elles sont masquées ici, car `<` et `>` ne sont pas des caractères valides dans une URL : il faut utiliser leur représentation hexadécimale, sous forme de `%3C` et `%3E`. Toutefois, un navigateur moderne saura se débrouiller avec une URL telle que :

```
http://www.monsite.com/index.php?nom=<b>monsieur</b>
```

Nous avons maintenant une attaque possible pour le script. Grâce à la vulnérabilité que nous avons identifiée, celui-ci va maintenant afficher le nom de l'utilisateur en gras. En effet, la balise `` fait apparaître le texte encadré en gras. On a donc modifié le comportement initial de la page pour lui faire exécuter une fonctionnalité inattendue. À ce stade, il est certain que vous n'êtes pas encore impressionné par les injections HTML : une vulnérabilité qui met en gras du texte dans une page web n'est pas un gros problème. En fait, nous avons mis le doigt sur la vulnérabilité et nous pouvons maintenant l'exploiter pour réaliser des attaques plus complexes. Pour cela, il suffit d'envoyer des balises HTML, qui réalisent des opérations plus complètes, que simplement changer la graisse d'une police. Parmi les candidats, il y a bien sûr les images, qui seront chargées sur un site distant ou bien l'inclusion de JavaScript. Risques liés aux applications web

Reprenons notre exemple :

```
http://www.monsite.com/index.php?nom=monsieur<script src="http://autre.site.com/xss.js">
```



Cette nouvelle attaque va injecter une balise JavaScript qui demande au navigateur de charger un fichier JavaScript complet, sur un autre site que le site vulnérable. Du point de vue du navigateur, c'est parfaitement valide. Une page HTML permet de combiner des contenus en provenance de différents sites. Généralement, un site web fournit la totalité des contenus qu'il affiche, mais ce n'est pas forcément toujours le cas. Par exemple, les services de statistiques demandent aux webmestres d'inclure un fichier JavaScript ou une image dans toutes leurs pages : ainsi lorsqu'un visiteur charge la page, il charge aussi des informations sur le site de statistiques, qui peut ainsi compter le nombre de visites. Il est possible de charger de nombreuses ressources externes : des images, des animations Flash, des applets Java, des frames ou iframes, du code JavaScript, des feuilles CSS, etc.[01]

Normalement, les liens vers les ressources externes sont gérées par le programmeur de la page : c'est lui qui sait où sont placées les ressources affichées dans la page et qui établit les références là où elles sont utiles. C'est une fonctionnalité immémoriale du Web et sûrement un atout pour le partage d'informations. Néanmoins, en ce qui concerne la sécurité, cela ouvre la porte aux injections les plus dévastatrices : avec une vulnérabilité telle que celle que l'on vient de voir, une partie des ressources de la page est configurée par un auteur externe arbitraire.

Avec un fichier JavaScript externe, il est désormais possible de réaliser de nombreuses opérations distinctes avec le navigateur victime :

- charger du contenu arbitraire : en ajouter, en supprimer, en modifier dans la page en cours ;
- forcer l'utilisation de formulaires : aussi bien ceux de la page en cours que les formulaires distants ;
- détourner des formulaires vers un autre site : l'autre site peut alors s'insérer dans les communications entre le navigateur et le site légitime ;
- voler les cookies : cela conduit directement à l'usurpation d'identité ;
- rediriger vers un autre site ;
- faire exécuter au navigateur de nombreuses opérations au nom de son utilisateur.

b. Savoir protéger les pages web

La stratégie de protection principale contre les injections HTML est la même que pour toutes les injections : il faut neutraliser les données pour la technologie à laquelle elles sont

transmises.

1) Neutralisation des caractères spéciaux :

Dans le cas des pages web, il faut neutraliser les caractères spéciaux HTML. Il y a deux fonctions de protection fournies par PHP :

- `htmlentities()`
- `htmlspecialchars()`

`htmlspecialchars()` remplace tous les caractères qui ont une signification spéciale en HTML par leur entité HTML : le terme « entité HTML » est un anglicisme pour désigner une séquence représentant un caractère spécial HTML. Par exemple, le caractère `<` est remplacé par la séquence `<` (en anglais, `lt` signifie `less than`, c'est-à-dire « plus petit que »). Cette règle s'applique ainsi aux caractères suivants :

- `&`, le « et commercial », qui commence les séquences HTML (telles que `&`) ;
- `'`, les guillemets simples, utilisés dans les attributs ;
- `"`, les guillemets doubles, utilisés dans les attributs ;
- `<` et `>`, les signes « inférieur à » et « supérieur à », qui délimitent une balise

`htmlentities()` est une version plus complète de `htmlspecialchars()`, elle remplace dans une chaîne tous les caractères possibles par leur séquence HTML.[6]

Cela ajoute les caractères spéciaux à la liste précédente, comme les caractères accentués français, ce qui a le double avantage de protéger la chaîne de caractères et de rendre son contenu plus sûr à interpréter pour un navigateur web.

```
print htmlentities('Damien Séguy & Philippe Gamache');  
Damien S&eacute;guy & Philippe Gamache [01]
```

Le revers de la médaille est que la chaîne est allongée par ces protections et qu'elle est rendue moins lisible à un être humain.

Il y a par ailleurs deux autres points à prendre en compte avant de s'en remettre aveuglément à ces deux fonctions.

Le premier aspect est le choix du jeu de caractères utilisé, dont dépend directement la valeur nominale d'un caractère. PHP utilise un jeu en interne, le plus souvent ISO-8859-1

ou latin-1 : c'est un jeu qui couvre le français. Les protections qui sont appliquées à une chaîne de caractères dépendent directement du jeu de caractères utilisé. Voici la même protection que dans l'exemple précédent, appliquée cette fois à une chaîne de caractères entrante de type Unicode, au lieu de latin-1 :

```
print htmlentities('Damien Séguy & Philippe Gamache');  
Damien S&Atilde;&copy;guy &amp; Philippe Gamache
```

Le deuxième argument des fonctions htmlspecialchars() et htmlentities() indique le nom du jeu de caractères utilisé. Utilisez donc le bon jeu pour ne pas laisser passer de problèmes. Au besoin, les fonctions iconv() de PHP vous aideront à vérifier et convertir les chaînes de caractères :

```
print htmlentities(iconv('UTF-8','ISO-88590-1','Damien Séguy & Philippe  
Gamache'));  
Damien S&eacute;guy &amp; Philippe Gamache
```

Par ailleurs, il faut savoir que les injections de code JavaScript n'ont pas toujours besoin d'utiliser les caractères spéciaux de HTML pour s'exécuter correctement. En effet, JavaScript utilise d'autres caractères spéciaux, qui sont neutres pour HTML : c'est le cas des parenthèses, des accolades et du point-virgule. De plus, JavaScript peut être exécuté à partir d'attributs de nombreuses balises HTML. Nous étudierons cela dans les sections qui suivent.

2) Les balises <iframe> et <frame>

Même si les cadres sont passés de mode, ils sont toujours disponibles dans les navigateurs modernes. On peut toujours utiliser les balises <frame> pour scinder une page en plusieurs parties et charger différentes URL.

Les cadres ont aussi connu une évolution, sous la forme du iframe, le cadre intégré. Le « i » signifie inline, c'est-à-dire intégré dans le corps de la page HTML. Il réserve un espace d'affichage dans une page web et fonctionne comme un cadre traditionnel.

Du point de vue de l'utilisateur, c'est totalement transparent, car aucune information n'est affichée explicitement pour indiquer qu'une page externe est chargée. Comme <iframe> et <frame> font référence à des pages web complètes, il est possible de les utiliser pour charger des applications JavaScript complètes et ainsi avoir accès à toutes les ressources de

la page. Grâce aux cadres, on peut charger un formulaire complet et effectuer des requêtes vers des sites externes, en contournant la politique JavaScript de restriction des accès au domaine en cours.

Du point de vue de l'affichage, il est possible de donner des dimensions très réduites à un cadre : ainsi, il passera totalement inaperçu pour l'utilisateur.

Ce sont donc les balises les plus dangereuses à utiliser, du point de vue de la sécurité.

2.1.2. Protection contre les CSRF

À partir des XSS, il est possible d'élaborer des attaques encore plus vicieuses : les CSRF. Si le fait de pouvoir injecter un peu de JavaScript dans une page web ne vous donne pas de sueurs froides, les paragraphes qui suivent devraient vous montrer la valeur de ces vulnérabilités.

Les CSRF, aussi appelées XSRF (Cross-Site Request Forgeries, se prononce « Sea Surf »), sont une autre forme d'attaque sur le Web. Elles se basent principalement sur le navigateur, qui sert de plaque tournante entre plusieurs sites web et notamment ceux pour lesquels il a obtenu des privilèges particuliers. Pour bien comprendre une attaque CSRF, il faut commencer par identifier les personnes qui jouent dans la pièce. Attention, les acteurs vont être nombreux. Commençons par la victime, celle qui va subir les conséquences directes de l'attaque.[01]

L'application victime d'une CSRF est une application plutôt bien sécurisée : elle a mis en place plusieurs couches de sécurité complémentaires, pour mieux dérouter les pirates.

Par exemple, l'identification des visiteurs, la protection des pages d'administration par identification HTTP, la mise en place d'une session PHP, le verrouillage de l'IP des utilisateurs. Pour obtenir le droit d'utiliser les fonctionnalités de l'application victime, il faut fournir plusieurs mots de passe et disposer d'une IP listée dans les domaines autorisés.

Rien de tout cela n'est en possession du pirate qui organise la CSRF : d'ailleurs, à aucun moment de l'opération, il ne sera en possession de ces informations. Les seuls qui disposent de ces informations sont l'administrateur du site et l'utilisateur dûment affranchi.

Leur utilisation du site est complètement valide et normale, sans histoire jusqu'à maintenant. Après avoir terminé ses opérations quotidiennes, l'administrateur du site

victime se rend sur un autre site web.

Le deuxième site est complètement distinct du premier. Il n'a aucun rapport avec l'application victime. Il est simplement porteur d'une vulnérabilité XSS, qui permet à un pirate d'injecter du code HTML dans ses pages.

Après l'application victime, l'utilisateur inconscient et l'application tierce, voici maintenant l'entrée en scène du pirate. Ce dernier va utiliser l'application tierce et la vulnérabilité XSS pour injecter une image dans la page et la faire lire par l'administrateur du site victime. Une telle image comportera une adresse URL du type :

```

```

Lorsque l'administrateur charge la page avec la vulnérabilité XSS, il charge aussi une image sur le site qu'il administre. En fait, ce n'est pas une image valide. Si vous lisez sémantiquement l'URL présentée en illustration de la XSS, c'est un lien vers une page d'administration du site victime, qui efface un article du site : celui d'identifiant 22.

Ainsi, au lieu de charger une image sur le site qu'il consulte, l'administrateur déclenche en fait l'effacement d'un article. Du point de vue de l'application victime, c'est une commande complètement valide : elle reçoit un ordre provenant d'un utilisateur légitime et dûment identifié comme un administrateur raisonnable.

Si on revoit les techniques qui ont été mises en place pour protéger le site victime, on s'aperçoit que les sécurités ne protègent pas le site : la session PHP est bien celle de l'administrateur et le pirate l'utilise sans même la connaître ; même chose pour la session HTTP ; et la validation par le domaine de l'IP est aussi valide. Au bout du compte, l'ordre est bien transmis au site victime, qui n'a aucun autre choix que de l'accepter. [6]

Le problème principal ici est que l'application victime n'a aucun moyen pour se défendre : une fois qu'un utilisateur est correctement identifié, elle en accepte toutes les commandes.

Or, par souci de commodité pour l'internaute, l'identité est automatiquement envoyée au site par le navigateur quand il sollicite une page : le cookie de session, les identifiants HTTP et son IP sont transmis au site pour vérifications. Ces données sont toujours valables, sauf peut-être la session PHP qui va disparaître au bout de 15 minutes si on utilise la configuration standard de PHP. Ainsi, l'administrateur a conservé ses droits sur le site

victime, alors qu'il l'a quitté et qu'il ne l'utilise plus vraiment.

Du point de vue du camouflage, les CSRF sont d'autant plus vicieuses que le pirate exploite la vulnérabilité d'un autre site pour faire exécuter ses commandes démoniaques. L'application victime aura beaucoup de mal à remonter à la source des commandes pour parer le coup. Elle arrivera à identifier l'administrateur comme origine du problème et peut-être même le site vulnérable à la XSS comme origine de la navigation désastreuse. Mais pour remonter encore au pirate, il faudra d'autres efforts.

La CSRF utilise plusieurs capacités des navigateurs pour arriver à ses fins. D'abord, il y a la possibilité de naviguer simultanément sur plusieurs sites. Notez tout de même que la CSRF pourrait très bien fonctionner séquentiellement : l'administrateur gère son site, puis se rend sur le site vulnérable. La CSRF exploite ainsi la conservation des autorisations.

Comme les droits d'un site sont conservés durant un certain temps, un site tiers peut très bien exploiter les droits acquis par un navigateur. Enfin, pour finir de broser un tableau bien noir, notez bien que l'exemple ci-dessus utilise une attaque par méthode GET : c'est la plus simple à présenter dans le cadre d'un livre. Il serait tout à fait possible d'exploiter la XSS pour réaliser une soumission de formulaire via la méthode POST, que ce soit avec du code JavaScript un peu plus élaboré, ou avec une animation Flash.

Pour terminer, voici une illustration simple d'une CSRF, qui ne fait même pas appel à une vulnérabilité XSS. Une pratique courante sur Internet est de placer sur un site des images en provenance d'un autre site. Quand cela se fait dans le cadre d'un webmail, c'est une pratique reconnue. Quand l'objectif visé est d'épargner au premier site les coûts en bande passante et de les reporter sur le deuxième site, cela s'appelle du vol de bande passante, band width theft ou encore hot linking en anglais.

Une technique de défense des sites victimes de ces pratiques consiste à remplacer l'image ou la ressource fréquemment demandée par une redirection vers une URL du type `http://www.site-appelant.com/logout.php`. De cette manière, quiconque charge la page sur le site original, va aussi chercher l'image sur le site victime, mais sera automatiquement redirigé vers la page de déconnexion. Cela va paralyser tous les comptes qui affichent cette image, jusqu'à ce qu'elle disparaisse du site. Imaginez alors que l'administrateur soit le seul à pouvoir supprimer cette image : comment va-t-il pouvoir le faire s'il est automatiquement déconnecté à chaque fois que l'image s'affiche ?

Dans cette situation, la CSRF fait bien appel aux droits acquis d'un utilisateur pour lui nuire, sans jamais accéder à ses éléments identifiants. On voit alors toute la difficulté qu'il y a à identifier ce type d'attaque et à s'en prémunir.

a. **Se défendre contre une CSRF**

La défense contre une CSRF se fait essentiellement avec deux mesures : il faut compliquer la vie des pirates et s'assurer souvent de l'identité des utilisateurs. [6]

Pour compliquer la vie des pirates, il est recommandé de ne plus utiliser de méthode GET telle que présentée dans l'exemple précédent. Une méthode POST ou bien un enchaînement de plusieurs pages avant d'arriver à l'exécution de l'effacement rendra l'opération plus difficile à transformer en JavaScript et à faire passer par une attaque XSS.

Par ailleurs, l'ajout d'un écran de confirmation avant la finalisation de l'opération est une bonne technique. C'est exactement ce que font les systèmes d'exploitation avant une tâche irréversible comme un effacement de fichiers : un dialogue apparaît avec une demande de confirmation. Les demandes de confirmation HTML sont faciles à contourner, car elles nécessitent simplement l'envoi d'un jeton de confirmation supplémentaire.

D'un autre côté, une confirmation JavaScript est aussi simple à contourner, mais son intrusion dans l'écran pourra surprendre un utilisateur et l'alerter qu'une catastrophe est en cours : pourquoi est-ce qu'une alerte du site victime.application.com apparaît sur le site de vulnerable.site.com ?

En fait, cette approche est contournable, car on n'a pas encore trouvé ce qui bloque réellement une CSRF : ce que cette attaque doit absolument éviter, c'est une demande d'identification de l'utilisateur. En effet, cette information est détenue uniquement par l'utilisateur et pas du tout par le pirate. Pour bloquer efficacement une CSRF, il faut donc valider l'identité de l'utilisateur avant toute opération sensible. Si cette étape échoue, probablement y a-t-il usurpation d'identité, d'une manière ou d'une autre.

Vérifier l'identité de l'utilisateur en permanence finit par nuire à l'ergonomie d'un site. Taper 15 caractères entre deux clics sur des hyperliens est un moyen efficace pour réduire à néant l'ergonomie d'un site : force est de le reconnaître. Il faut donc trouver un compromis acceptable. Vous devez mesurer l'importance stratégique de l'opération qui est demandée à l'application et le niveau de sécurité associé. Une opération irréversible

comme un effacement ou la publication d'un contenu après modération est une opération cruciale pour un site web : il est recommandé de demander une nouvelle authentification avant de l'exécuter. En revanche, l'ajout d'un nouveau contenu sans publication et la consultation des statistiques peuvent être considérés comme des opérations bénignes : il ne sera pas opportun de demander à nouveau l'identité de l'utilisateur.

La bonne pratique consiste donc à identifier les opérations les plus importantes et à les protéger avec une nouvelle authentification systématique.

Une autre solution consiste à implanter un système d'identification aléatoire : après un certain temps, inférieur à la durée d'une session, ou après un certain nombre de clics dans le site, l'application demande automatiquement une identification. Si votre application est critique, vous pourrez réduire le laps de temps entre deux identifications au plus bas possible, tandis que si vous voulez une politique plus souple, vous augmenterez ces valeurs. La politique de sécurité est maintenant paramétrable.

2.1.3. Protection des formulaires

La nature dynamique des sites modernes permet de personnaliser le contenu qui est affiché grâce à des paramètres fournis par l'utilisateur, soit explicitement comme pour un formulaire, soit implicitement via des cookies ou la description du navigateur.

Nous présentons dans ce chapitre tous les risques inhérents à l'échange de données sensibles avec les internautes, que ce soit via les formulaires ou par téléchargement de fichiers. Il est vital pour votre application de vérifier et valider chacune des données entrantes, mais aussi, nous le verrons plus loin, de neutraliser celles qui seront transmises aux applications connexes.

Les formulaires font partie des points les plus vulnérables des applications web : les scripts d'action fonctionnent exclusivement à partir de données provenant de l'utilisateur.

Rares sont les formulaires qui acceptent de marcher sans un minimum de données dynamiques : au pire, des valeurs par défaut sont utilisées.

Un script de formulaire réagit suite à la soumission d'un formulaire. Les formulaires ont traditionnellement une apparence proche de ceux de ceux de la Sécurité sociale, avec des boutons en plus.

Il existe aussi des scripts qui ne passent pas par des formulaires interactifs : pour gérer du contenu en ligne par exemple, il est fréquent d'accéder à un article via un script générique et un identifiant passé en argument dans l'URL, comme ceci :

```
http://www.site.com/afficher.php?id=22
```

En lisant sémantiquement cette URL, on comprend qu'elle va afficher le contenu d'identifiant 22. Sous cette forme, afficher.PHP n'utilise pas de formulaire, mais présente les mêmes caractéristiques : il fonctionne à partir de données choisies dynamiquement par le visiteur, via un lien hypertexte astucieusement présenté. En fait, il ne serait pas difficile de créer un formulaire HTML pour gérer l'accès à cette page.

a. CAPTCHA

CAPTCHA est l'acronyme anglophone pour « Completely Automated Public Turing test to tell Computers and Humans Apart » : un test de Turing complètement automatisé pour faire la différence entre un humain et un ordinateur.[1]

Cette différence se révèle importante, car Internet est peuplé de deux formes de vie : les humains et les robots, applications automatisées qui effectuent les mêmes opérations que les humains, mais bien plus vite et sans se lasser.

Prenons l'exemple des spammers de commentaires : de nombreux blogs ont mis en place des systèmes de commentaires, dans le but de faciliter les échanges entre l'auteur et sa communauté. Malheureusement, cela constitue une cible idéale pour un spammeur : il peut envoyer un message publicitaire en guise de commentaire ou placer un lien vers son site, avec l'espoir de grimper dans les classements des moteurs de recherche. Certains blogs ont ainsi vu des centaines de commentaires inutiles être postés en quelques heures.

Des auteurs finissent par passer plus de temps à effacer les spams qu'à bloguer. C'est un combat qui se répète tous les jours.

Avant les CAPTCHA, il n'y avait aucune parade autre que la suspension des fonctionnalités, la modération de ces centaines de commentaires étant tout bonnement réhibitoire et les filtres par mots-clés trop faciles à contourner.

Un CAPTCHA se présente sous la forme d'un champ de formulaire supplémentaire. Il s'agit d'une question dont la réponse sera aisée pour un être humain, mais impossible pour

un robot (en théorie). Un tel test est appelé un test de Turing. Certains sont très simples, d'autres particulièrement sophistiqués. Voici quelques concepts :

1. Résolution d'une équation mathématique, avec quelques opérations de base :
Additionnez 10 et 12.
2. Réponse à une question qui n'a rien à voir avec le site : quel est le prénom de l'auteur du site ? Ou bien quel jour étions-nous hier ?
3. Lecture d'une chaîne de caractères tordus : c'est la forme la plus classique des CAPTCHA.
4. Écoute d'un texte en format audio, dont il faut stocker les caractères énoncés dans le champ. C'est un complément du test précédent, adapté aux utilisateurs ayant des problèmes de vue.
5. Tests de logique du type : quelle image n'est pas de la même famille que les deux autres ?
6. Tests basés sur l'exécution de JavaScript : les moteurs de spams sont généralement trop rudimentaires pour exécuter correctement ces codes, alors qu'un navigateur complet le fera aisément, tant que JavaScript est activé

Quel que soit le type de test de Turing, la mise en place d'un CAPTCHA fonctionne toujours sur le même principe. Le test est généré sur le serveur web, à partir d'une source aléatoire. La solution est enregistrée sur le serveur dans une session PHP.

Puis, la solution est transformée en question, via un processus de déformation : génération d'une image, d'un son, d'un test, etc. Enfin ce dernier est affiché.

Voyons un exemple de la structure du formulaire :

```
<?php
session_start();

if (!empty($_POST['captcha'])) {
    if ($_POST['captcha'] == $_SESSION['captcha']) {
        $resultat = '<p>Sois le bienvenu!</p>';
    }
    else {
        $resultat = '<p>Va-t-en, méchant robot!</p>';
    }
}
else {
```

```

        $resultat = '<p>Êtes-vous un robot ou un humain?</p>';
    }

$_SESSION['captcha'] = substr(md5(rand(0,100000).time()),0,6);
?>
<html>
<head>
    <meta charset=utf-8">
    <title>CAPTCHA</title>
</head>
<body>
    <?php echo $resultat; ?>
    <form action="index.php" method="post">
        
        <input type="text" name="captcha"><br>
        <input type="submit" value="go" />
    </form>
</body>
</html>[6]

```

Paradoxalement, le script se lit depuis le bas vers le haut, même s'il s'exécute dans l'ordre inverse.

Tout en bas, vous trouverez un formulaire, qui devrait afficher quelque chose de semblable à ceci :



Figure 2-1 : Exemple de Captcha

L'image de CAPTCHA est dans la balise IMG et la source est captcha.php : nous y reviendrons. Le reste du formulaire est très simple et vous pouvez y ajouter tous les champs que vous souhaitez.

Le script PHP au-dessus se charge de préparer le test. Lors de la première sollicitation du formulaire, le script démarre une session PHP et y stocke une valeur produite aléatoirement. Cette dernière est remplacée à chaque utilisation du formulaire, pour éviter les attaques systématiques du formulaire : comme le test est toujours régénéré, il ne sert à rien de tenter plusieurs réponses.

Cette technique permet en outre d'avoir une solution simple pour le cas où le CAPTCHA

est trop difficile : il suffit à l'utilisateur de recharger la page pour obtenir un test différent.

La valeur créée est stockée en session, pour que le client n'ait aucune chance de la deviner à partir des informations dans la page. Lorsque le formulaire est soumis, on peut alors comparer la valeur qui est envoyée avec celle qui est dans notre session. En cas d'égalité, on a identifié un être humain et en cas d'échec, on a identifié assez sûrement un robot.

Le test CAPTCHA est produit dans le script captcha.php : nous allons présenter un test visuel, ce qui est très adapté pour un livre. Nous vous laissons le soin de concevoir votre propre test, adapté à votre audience et au niveau de sécurité que vous souhaitez.

```
<?php
header("Content-Type: image/png");
session_start();
$im = imagecreate(100, 40);
$white = imagecolorallocate($im, 255, 255, 255);
$black = imagecolorallocate($im, 0, 0, 0);
imagefill($im, 0, 0, $white);

if (!empty($_SESSION['captcha']))
{
    for($i = 0; $i < strlen($_SESSION['captcha']); $i++)
    {
        $r = rand(0,255);
        $g = rand(0,255);
        $b = sqrt(100 * 100 - $r * $r - $g * $g);
        $couleur = imagecolorallocate($im, $r,$g,$b);
        imagechar($im, rand(0,4), 0 + 10 * $i + rand(0, 5), rand(0,
10) , $_SESSION['captcha'][$i], $couleur);
    }
}
imagepng($im);
imagedestroy($im);
?>[1]
```



Figure 2-2 : Exemple de CAPTCHA visuel

Couleurs, angle, taille de police, police de caractères, positions : nous avons maintenant un test qui complique la tâche. Il reste encore à ajouter des parasites dans l'image, comme des traits de couleurs ou des effets d'ombrage pour avoir un niveau de protection honorable.

Pour avoir une version audible du CAPTCHA, l'adaptation du script précédent est

relativement simple : ajoutez un lien vers un script appelé `captcha_son.php`, qui reprendra l'image de la même façon, mais la transmettra à un synthétiseur vocal sur votre système d'exploitation.

2.1.4. Protection des sessions

Les sessions répondent à un problème crucial du protocole HTTP : comment conserver des informations entre deux pages web. En effet, le protocole HTTP est sans état : entre deux sollicitations du serveur web, toutes les informations sont perdues.[5]

Dans ce cadre, comment établir des fonctionnalités de base, comme une séance de travail ou une identification ? Sans système de sessions, il n'est plus envisageable de mettre en place une identification, car elle doit être envoyée à chaque fois.

C'est exactement ce que fait l'identification HTTP : en réponse à un code 402 (Unauthorized), le navigateur affiche un dialogue où l'on peut saisir un nom d'utilisateur et un mot de passe. Après cela, l'utilisateur est identifié et il peut naviguer sur les ressources qu'il souhaite. Toutefois, la réalité est masquée : en effet, le navigateur note les informations de connexion lors du dialogue et les transmet à chaque fois qu'il interroge le site web. Il fournit donc une aide non négligeable à l'utilisateur, mais ne résout pas le problème initial : le serveur web oublie tout entre deux requêtes. [6]

a. Fonctionnement des sessions

Les sessions PHP ont pour objectif de résoudre ce problème. Entre deux scripts PHP, des données mises en session sont transmises automatiquement. Le premier script peut ainsi établir une identité et le second fournir des services en fonction de cette identité. Pour y arriver, PHP doit composer avec les contraintes du protocole HTTP.

b. Le Web et la persistance

Il faut résoudre deux problèmes : la persistance des données sur le serveur et l'identité du visiteur auquel sont associées les données sauvées. Les sessions sont donc au confluent de nombreuses technologies et elles exploitent des ressources complémentaires.

Pour résoudre le problème de persistance, PHP doit faire appel à un medium stable : en effet, PHP lui-même n'est pas persistant entre deux requêtes. Par défaut, il fait appel au système de fichiers, mais il peut aussi s'interfacer avec d'autres systèmes pour y stocker

les données, puis les relire : bases de données, mémoire vive, démons, etc.

Les sessions PHP sont donc constituées de trois parties :

- un identifiant de session, unique et aléatoire, qui est confié au navigateur pour une durée spécifique ;
- un stockage de données, appelé par le script et accessible depuis le serveur ;
- un système de communication de l'identifiant à tous les scripts PHP qui en ont besoin.

Malheureusement, chacun de ces systèmes est sujet à des vulnérabilités et des limitations.

i) Deux méthodes de transport

Du côté Web, il faut un système qui assure le suivi de l'utilisateur en fonction de sa navigation. PHP propose deux techniques pour cela : les cookies ou la réécriture de liens, aussi appelé trans-id.

Les cookies sont la forme la plus adaptée au suivi de session. En effet, comme nous l'avons vu précédemment, ils sont confinés à un navigateur et s'échangent exclusivement entre le navigateur et le serveur. De plus, ils disposent d'un système de limitations dans le temps (la date d'expiration du cookie) et dans l'espace web (les limitations par domaine, par serveur et même par chemin), qui en font l'outil idéal pour gérer les sessions.

Pour suivre la navigation d'un visiteur qui refuse les cookies, il suffit de personnaliser tous les liens d'un site. La valeur qui était stockée dans les cookies est l'identifiant de session : c'est cette valeur que PHP va ajouter à chaque lien utilisé dans une application web, pour pouvoir le retrouver lorsque l'utilisateur du site effectue sa progression. Comme on ne peut pas prévoir à l'avance la prochaine page demandée, il faut que chaque lien soit modifié. Cette méthode est plus universelle que les cookies et ne peut pas être désactivée côté navigateur.

Le choix entre les deux méthodes de transmission de l'identifiant de session est discutable. D'un côté, le cookie est un mécanisme adapté et relativement sûr. Néanmoins, une fraction de la population en ligne n'utilise pas de cookies, notamment les robots de moteurs de recherche. De plus, les cookies sont les victimes désignées des vols et des injections XSS. D'un autre côté, les identifiants dans les URL sont plus faciles à pirater : il suffit de le lire

dans l'URL, ou bien de se la faire envoyer par un utilisateur peu prudent.

ii) L'importance de l'identifiant

Au bout du compte, les sessions mettent en place un système dans lequel des données placées sur le serveur sont reliées à un utilisateur à l'aide d'un identifiant de session. Si un utilisateur présente un tel identifiant, alors PHP charge les données correspondantes dans son medium de stockage. Sinon, une nouvelle session est créée, ou bien l'utilisateur est considéré comme anonyme.

Cette approche fait ainsi peser toute la sécurité sur l'identifiant. De la même façon qu'un mot de passe régit l'accès à de nombreuses technologies, l'identifiant est le sésame qui donne accès à l'identité d'un utilisateur sur un serveur. Si un pirate vole un identifiant de session, il peut usurper l'identité d'un autre utilisateur.

Les données qui sont sur le serveur sont bien protégées, dans la mesure où elles ne sortent pas du serveur lui-même, à moins que cela ne soit prévu par le site et demandé par le visiteur. La protection des données dépend donc directement de la sécurité de l'identifiant.

L'identifiant lui-même ne contient aucune information. Il faut bien comprendre qu'il est complètement aléatoire. Il présente la même valeur qu'une clé : en soi, c'est un bête morceau de métal, mais si elle est utilisée dans la bonne serrure, c'est tout un espace confidentiel qui est ouvert. Toutefois, il existe un nombre incroyable de serrures et les essayer toutes sans se faire repérer est illusoire.

Il est donc primordial de bien protéger l'identifiant, d'autant que les techniques pour le voler sont nombreuses. Les premières sources de problèmes sont les XSS : ces vulnérabilités qui permettent de faire exécuter n'importe quel code JavaScript sont idéales pour exporter un identifiant de session vers un site externe, qui pourra l'utiliser immédiatement.[6]

2.2. Sécurité du code source

La première chose à faire pour protéger une application PHP est de s'assurer que le code qui est exécuté est bien celui que vous avez écrit. C'est une précaution qui vient naturellement à tous les programmeurs, mais qui est parfois battue en brèche par certaines fonctionnalités de PHP détournées de leur utilisation principale.

2.2.1. Protection contre l'injection de code distant

Si les scripts sont généralement bien protégés, comment se fait-il que tant d'applications soient victimes d'injection de code PHP ? Sur les sites de sécurité, on trouve régulièrement des alertes indiquant que du code arbitraire a été injecté dans une application bien connue. C'est à cause d'une fonctionnalité bien pratique de PHP : son navigateur web intégré.[1]

PHP dispose d'un navigateur web interne, qui lui permet d'accéder à des données enregistrées sur le Web:

```
$contenu = file_get_content('/home/web/index.html') ;
```

Cette instruction PHP lit tout un fichier et le verse dans la variable \$contenu. Le fichier cidessus est local au serveur, dans le dossier /home/web/. En ajoutant les informations de protocole et de serveur, on peut lire facilement un fichier distant avec la même instruction :

```
$contenu = file_get_content('http://www.php.net/') ;
```

Cette fois-ci, c'est le fichier disponible sur le site de www.php.net qui est lu. Cette fonctionnalité est contrôlée par la directive allow_url_fopen, qui autorise l'utilisation des protocoles HTTP, HTTPS, FTP et FTPS depuis les fonctions de manipulation de fichiers.

C'est effectivement très pratique pour charger des ressources distantes comme des fils de dépêches, des fichiers RSS ou ATOM, ou encore pour réaliser un robot d'indexation.

Pour revenir aux problèmes de sécurité, il faut savoir que cette fonctionnalité est simultanément disponible avec les instructions include(), require(), include_once() et require_once(). Exemple :

```
include('http://www.autre.site.com/bibliotheque.inc') ;
```

Cette instruction va chercher du code PHP sur un site distant, le ramène sur le serveur local et l'intègre dynamiquement dans le code du script courant, puis l'exécute comme du code local. Fonctionnellement, c'est exactement comme une inclusion de fichier local, mais avec un fichier distant. C'est par ce biais que les pirates effectuent des injections de code.

La technique est simple : il suffit de préparer une bibliothèque PHP sur un site public. Ce code PHP sera inclus dans un site victime et exécutera des commandes qui ouvriront d'autres portes aux pirates : souvent, on commence par phpinfo().

Désormais, il suffit de trouver le moyen pour que l'application victime effectue l'inclusion distante. Pour cela, il suffit de modifier les appels dans les fonctions `include()` et ses cousines. Notez que par la suite, nous utiliserons le nom de `include()` pour représenter `include()`, `require()`, `include_once()` et `require_once()`, car ces quatre fonctions réalisent les mêmes opérations du point de vue de la sécurité. Ainsi, certaines applications web sont construites avec un contrôleur, qui a la charge de définir l'action à mener et de décider de l'inclusion du code PHP nécessaire. L'action est définie dans l'URL appelante et elle est choisie par l'utilisateur au cours de sa visite du site. Par exemple, étudiez l'URL suivante :

```
http://www.site.com/index.php?action=affiche&id=22
```

Sémantiquement, on peut lire que l'action est de type `affiche` et que l'objet de l'affichage sera une ressource d'identifiant `22`. Le contrôleur convertit cette URL en action et inclut dynamiquement le module d'affichage à partir de l'URL, comme ceci :

```
include($_GET['action']. '.inc') ;
```

Avec une telle ligne de code, il devient possible de réaliser l'inclusion de code distant en modifiant `action` pour qu'elle utilise le protocole HTTP :

```
http://www.site.com/index.php?action=http%3A%2F%2Fwww.sitedupirate.com%2F
injection&id=22
```

L'inclusion va maintenant être la suivante :

```
include('http://www.sitedupirate.com/injection.inc') ;
```

L'application va maintenant exécuter le code du site distant comme si c'était son propre code. Le code du site distant est totalement sous le contrôle du pirate, tandis que l'application fonctionne encore avec les droits et les ressources du serveur. Une fois cette première vulnérabilité identifiée, il est facile de lancer un script d'analyse et de raffiner considérablement l'attaque.

Une partie du problème provient bien sûr du fait que les fonctions d'accès aux fichiers et les fonctions de flux sont mélangées avec les fonctions structurantes de type `include()`.

Pour cela, depuis PHP 5.2, une nouvelle directive `allow_url_include` a été introduite : elle permet de gérer séparément les fonctionnalités réseau de ces deux types de fonctions.

Par défaut, `allow_url_include` est désactivée. Il est même recommandé de toujours la

laisser à cette valeur et de trouver d'autres moyens de réaliser les inclusions distantes de code.

En termes de performances, l'inclusion distante ralentit considérablement les scripts PHP [5], car ce dernier doit attendre le rapatriement, l'analyse et l'inclusion du code avant de poursuivre l'exécution. Il est donc recommandé de réaliser une copie locale pour pouvoir y accéder plus rapidement et de profiter des caches de scripts, lorsque c'est possible. De plus, une modération du code distant sera un minimum à réaliser : au fond, vous êtes en train de donner accès à un site extérieur sur votre application !

Si le fichier distant ne contient que des données et pas de code PHP, il existe de nombreuses alternatives qui permettent de lire des données à distance et de les recevoir sous forme native en PHP. Json, WDDX, SOAP, REST sont des solutions adaptées aux échanges de données.

2.2.2. Protection des téléchargements

Le téléchargement de fichiers est une avenue royale pour importer du code PHP pirate sur un site. En effet, pour exécuter un script PHP, il suffit que ce dernier ait les autorisations de lecture et qu'il soit accessible depuis le Web. Il n'y a pas besoin de droits d'exécution ou d'écriture pour pouvoir exécuter un script PHP : simplement les droits de lecture.[1]

PHP est capable de recevoir des fichiers via le Web et de les stocker sur le serveur : c'est le téléchargement (upload) de fichiers. L'application la plus populaire consiste à télécharger des images, que ce soit pour un album de famille ou pour un avatar sur un forum. L'image est validée, puis mise immédiatement en production sur le site.

En termes de sécurité, le commun des programmeurs identifie le cas des fichiers exécutables et des virus : une fois chargés sur le serveur, ces derniers sont effectivement une source de problème, à condition qu'ils soient exécutés. En effet, pour les activer, il faut les charger, mais aussi les exécuter. Or, il est rare qu'un site web utilise des applications tierces.

En prenant même le cas d'un serveur Apache avec PHP sous forme de module, il n'y a qu'un exécutable, Apache lui-même et quelques autres logiciels système incontournables. Au niveau du site web, aucun exécutable n'est disponible et aucun droit d'exécution n'est donné, sauf aux dossiers. Ainsi, les fichiers exécutables et les virus représentent une

menace réelle, mais, en pratique, très rare. En fait, comme les fichiers sont chargés sur le serveur avec uniquement des droits de lecture, il ne reste qu'une seule menace possible : PHP lui-même.

En effet, les scripts légitimes d'une application web sur un serveur sont simplement dotés du droit de lecture : c'est le serveur Apache qui les lit, puis les exécute. Ainsi, en chargeant un script PHP, il n'y a plus qu'une condition pour que ce dernier soit exécuté sur le serveur : pouvoir y accéder depuis un navigateur web.

Dans ce cas, il devient possible de faire exécuter son propre code PHP. On ne parle plus d'injection, mais carrément de téléchargement de vulnérabilité !

2.3. Sécurité de la base de données

Les applications web stockent généralement leurs données dans des bases car ces dernières facilitent le traitement des informations et leur stockage. En 2000, indiquer qu'une application web utilisait telle ou telle base de données était un argument promotionnel.

Ce n'est plus le cas aujourd'hui ; le recours à une base de données pour une application web va de soi. Les bases de données assurent le stockage permanent des informations et la liaison avec de nombreux autres systèmes grâce à une interface standard. Par ailleurs, elles sont compatibles avec le langage SQL qui permet de les manipuler. Elles représentent donc un élément critique dans la sécurité de l'application et des données en général.

2.3.1. Protection contre les injections SQL

Les injections SQL font partie des vulnérabilités les plus courantes dans les applications web. Si les données en provenance de l'internaute ne sont pas protégées avant d'être insérées dans la commande SQL, il devient possible de modifier considérablement le comportement de la requête. [6]

La protection des valeurs, appelée aussi échappement par anglicisme, est la première tactique. Elle est universellement disponible, même avec de vieilles versions de PHP.

Au lieu d'utiliser la valeur de `$_POST` directement dans la commande, on la passe à la fonction `mysqli_real_escape_string()`, qui neutralise tous les caractères susceptibles de perturber le fonctionnement d'une requête, en ajoutant une barre oblique inverse juste

devant ces caractères.

```
$_POST ['passe'] = "' or (1) = '1 "
$mysql_passe = mysqli_real_escape_string($mid, $_POST['passe']) ;
```

Après ce traitement, `$mysql_passe` contient ceci :

```
\' or (1) = \'1
```

Les guillemets sont désormais précédés de barres obliques inverses : on dit qu'ils sont « neutralisés ». Leur valeur SQL est maintenant la même que leur valeur littérale. Ils ont perdu la signification spéciale que leur confère le langage SQL.

Toutefois, pour bien assurer la neutralisation de l'ensemble, il faut que la chaîne fournie soit placée entre guillemets dans la requête. Les types de guillemets, qu'ils soient simples ou doubles, n'ont pas d'importance, car `mysqli_real_escape_string()` sait aussi neutraliser les guillemets doubles.

Cependant, notez que les parenthèses n'ont pas été protégées. Elles introduisent une sous-sélection et peuvent conduire à un déni de service. Toutefois, si les guillemets ont bien été neutralisés, elles garderont leur valeur littérale et ne poseront pas de problème. Il est donc important de bien utiliser les guillemets d'encadrement dans toutes les requêtes SQL.

Or, en SQL, il est possible d'éviter les guillemets lorsqu'on manipule des nombres. Comme la fonction `mysqli_real_escape_string()` laisse passer les parenthèses, on retrouve une vulnérabilité par injection de sous-requête. Si vous devez manipuler des nombres dans une requête, il est recommandé de forcer le type à un format numérique dans PHP, avec `+ 0`, ou bien avec MySQL, en laissant les guillemets et en ajoutant `+ 0` dans la requête :

```
<?php
$_POST['id'] = "' or (1) = '1 " + 0 ;
$requete = "SELECT COUNT(*) FROM utilisateurs
WHERE id = '".
mysqli_real_escape_string($mid, $_POST['id'])."' + 0 " ;
?>
```

En utilisant ce principe de transtypage forcé, voici une astuce pour les mots de passe, ou les valeurs de hashage en général. Les mots de passe ne doivent jamais être stockés en clair ; ils sont toujours signés avec une fonction de hashage, comme MD5 ou SHA. Ces deux fonctions, disponibles simultanément en PHP et MySQL, produisent des chaînes de caractères qui ne contiennent que des chiffres et les lettres allant de A à F. Aucun de ces caractères n'a de signification particulière en SQL, ce qui permet de les utiliser en toute

sécurité. Ainsi, avant de rechercher une correspondance de mot de passe, il est recommandé de le passer à MD5 (ou équivalent), afin d'obtenir une valeur sécurisée pour le serveur SQL.

De cette manière, vos utilisateurs pourront utiliser les caractères qu'ils souhaitent dans leur mot de passe, et vous ne compromettez pas la sécurité de votre système d'identification.[1]

2.3.2. Protection des tables et des colonnes

L'idée est de créer des vues avec un nouveau nommage des tables et des colonnes. L'effet serait que n'importe quelle tentative de piratage ne ferait que se heurter à cette barrière (les tables temporaires des vues) qui n'offre qu'un nommage écran qui ne servirait en rien les injections SQL. [2]

2.4. Protection de l'accès à la base de données

L'accès à la base de données SQL est généralement protégé par un nom d'utilisateur et un mot de passe. Ils sont nécessaires dans l'application PHP pour pouvoir ouvrir la connexion. La gestion de ces mots de passe est une opération délicate : en effet, ils doivent rester secrets.[6]

2.4.1. Arrangement des accès au serveur MySQL

Dans une application web basée sur la plateforme WAMP (Windows, Apache, MySQL & PHP), il faut bien donner au script PHP les moyens de se connecter à MySQL. Donc, il faut les placer à un endroit auquel PHP peut accéder. Si, profitant d'une vulnérabilité, un pirate obtenait un accès au code source, les mots de passe seraient révélés. Il faut par conséquent les ranger judicieusement.

Par ailleurs, les mots de passe sont généralement rassemblés en un seul endroit. Comme ils sont virtuellement nécessaires dans chaque script d'une application, les applications web ont besoin d'un mécanisme pour les centraliser : en les modifiant à un endroit, il est possible de changer la configuration de toute l'application. En termes de sécurité, cela réduit l'exposition des données secrètes. Il ne reste plus qu'un seul point à sécuriser le stockage.[6]

2.4.2. Gestion des droits d'accès au serveur MySQL

L'une des barrières qui peut aider à protéger une application web est la gestion des droits d'accès aux différentes ses bases de données.

Il est établi que dans les nouvelles générations des SGBD, tout accès à une quelconque base de données doit se faire via des comptes utilisateurs.

De ce fait, il est fortement conseillé de créer trois comptes utilisateurs : [6]

- le premier sera attribué à la partie exposée au grand publique de notre site Web (Les internautes),
- le deuxième sera attribué à la gestion des données (Le dessinateur, Le rédacteur, Le chef rédacteur ainsi qu'au commercial),
- le troisième sera attribué à l'administration de la base de données (L'administrateur du site)

2.5. Protection des données d'authentification

Forcer les données d'authentification est légion de nos jours et il est clair qu'il faut envisager la protection de l'authentification dans n'importe quel système.

2.5.1. Cryptage des données

Les notions de sécurité les plus basiques exigent que le stockage, de données sensibles, ne doit jamais se faire dans leur forme brute. Dans le cas de l'authentification, le stockage de données cryptées s'impose.

Comme il est établi que n'importe quel algorithme de cryptage peut être forcé, il est nécessaire d'ajouter une partie intuitive à l'algorithme de base afin de s'assurer que dans le cas d'un décryptage, les données récupérées ne seront pas valide.

Moult solutions sont proposées par les experts :

i) Le Salage

Le salage, est une méthode permettant de renforcer la sécurité des informations qui sont destinées à être hachées (par exemple des mots de passe) en y ajoutant une donnée

supplémentaire afin d'empêcher que deux informations identiques conduisent à la même empreinte (la résultante d'une fonction de hachage). Le but du salage est de lutter contre les attaques par analyse fréquentielle, les attaques utilisant des rainbow tables, les attaques par dictionnaire et les attaques par force brute. Pour ces deux dernières attaques, le salage est efficace quand le sel utilisé n'est pas connu de l'attaquant ou lorsque l'attaque vise un nombre important de données hachées toutes salées différemment.[9]

ii) Le hachage

On nomme fonction de hachage, de l'anglais hash function (hash : pagaille, désordre, recouper et mélanger) par analogie avec la cuisine, une fonction particulière qui, à partir d'une donnée fournie en entrée, calcule une empreinte numérique servant à identifier rapidement la donnée initiale, au même titre qu'une signature pour identifier une personne. Les fonctions de hachage sont utilisées en informatique et en cryptographie notamment pour reconnaître rapidement des fichiers ou des mots de passe.[9]

iii) La profondeur

La profondeur consiste à effectuer le hachage combiné avec le salage mainte fois. Le nombre de répétition doit être aléatoire et caché.

2.5.2. Camouflage des données

Le camouflage est une méthode de dissimulation. Il permet à un organisme visible ou à un objet de passer inaperçu, en se fondant avec son environnement.

Les exemples incluent des rayures d'un tigre, le treillis d'un soldat moderne et un papillon présentant l'apparence d'une feuille. La théorie du camouflage couvre les différentes stratégies qui sont utilisées pour obtenir cet effet. [9]

2.6. Conclusion

Dans ce chapitre, nous avons abordé les défis de la sécurité web, nous avons abordé la protection contre les injections SQL, la protection des tables et des colonnes, la protection de l'accès à la base de données, la protection des données d'accès, la gestion des droits d'accès, ainsi d'autres. Des solutions pour faire face aux menaces qui peuvent menacer le bon fonctionnement des applications web ont été précisées.

Le chapitre suivant sera sujet d'une proposition d'une nouvelle conception ainsi que l'architecture de notre système.

Chapitre 3 : Conception et Architecture

3.1. Introduction

Dans ce chapitre, nous allons aborder la conception d'un site Web tout en ajoutant des atouts renforçant sa sécurité. Avant tout, nous allons préciser les outils aidant à réaliser notre projet, bien décrire son domaine, fournir les différents diagrammes de classes ainsi que leurs détails afin d'aboutir au nécessaire pour débiter l'implémentation.

3.2. Outils à utiliser

3.2.1. Annotation UML

Le Langage de Modélisation Unifié, de l'anglais Unified Modeling Language (UML), est un langage de modélisation graphique à base de pictogrammes conçu pour fournir une méthode normalisée pour visualiser la conception d'un système. Il est couramment utilisé en développement logiciel et en conception orientée objet.

L'UML est le résultat de la fusion de précédents langages de modélisation objet : Booch, OMT, OOSE. Principalement issu des travaux de GradyBooch, James Rumbaugh et Ivar Jacobson, UML est à présent un standard adopté par l'Object Management Group (OMG). UML 1.0 a été normalisé en janvier 1997; UML 2.0 a été adopté par l'OMG en juillet 2005. La dernière version de la spécification validée par l'OMG est UML 2.5.1 (2017). [9]

Utilisation

UML est utilisé pour spécifier, visualiser, modifier et construire les documents nécessaires au bon développement d'un logiciel orienté objet. UML offre un standard de modélisation, pour représenter l'architecture logicielle. Les différents éléments représentables sont :

- Activité d'un objet/logiciel
- Acteurs
- Processus
- Schéma de base de données
- Composants logiciels
- Réutilisation de composants

Grâce aux outils de modélisation UML, il est également possible de générer automatiquement tout ou partie du code d'une application logicielle, par exemple en langage Java, à partir des divers documents réalisés.

La boîte à outils

Les diagrammes sont dépendants hiérarchiquement et se complètent, de façon à permettre la modélisation d'un projet tout au long de son cycle de vie. Il en existe quatorze depuis UML 2.3.

Diagrammes de structure ou diagrammes statiques

Les diagrammes de structure (structure diagrams) ou diagrammes statiques (static diagrams) rassemblent :

- Diagramme de classes (class diagram) : représentation des classes intervenant dans le système.
- Diagramme d'objets (object diagram) : représentation des instances de classes (objets) utilisées dans le système.
- Diagramme de composants (component diagram) : représentation des composants du système d'un point de vue physique, tels qu'ils sont mis en œuvre (fichiers, bibliothèques, bases de données...)
- Diagramme de déploiement (deployment diagram) : représentation des éléments matériels (ordinateurs, périphériques, réseaux, systèmes de stockage...) et la manière dont les composants du système sont répartis sur ces éléments matériels et interagissent entre eux.
- Diagramme des paquets (package diagram) : représentation des dépendances entre les paquets (un paquet étant un conteneur logique permettant de regrouper et d'organiser les éléments dans le modèle UML), c'est-à-dire entre les ensembles de définitions.
- Diagramme de structure composite (composite structure diagram) : représentation sous forme de boîte blanche les relations entre composants d'une classe (depuis UML 2.x).
- Diagramme de profils (profile diagram) : spécialisation et personnalisation pour un domaine particulier d'un meta-modèle de référence d'UML (depuis UML 2.2).

Diagrammes de comportement

Les diagrammes de comportement (behavior diagrams) rassemblent :

- Diagramme des cas d'utilisation (use-case diagram) : représentation des possibilités d'interaction entre le système et les acteurs (intervenants extérieurs au système), c'est-à-dire de toutes les fonctionnalités que doit fournir le système.

- Diagramme états-transitions (state machine diagram) : représentation sous forme de machine à états finis le comportement du système ou de ses composants.
- Diagramme d'activité (activitydiagram) : représentation sous forme de flux ou d'enchaînement d'activités le comportement du système ou de ses composants.

Diagrammes d'interaction ou diagrammes dynamiques

Les diagrammes d'interaction (interaction diagrams) ou diagrammes dynamiques (dynamicdiagrams) rassemblent :

- Diagramme de séquence (sequencediagram) : représentation de façon séquentielle du déroulement des traitements et des interactions entre les éléments du système et/ou de ses acteurs.
- Diagramme de communication (communication diagram) : représentation de façon simplifiée d'un diagramme de séquence se concentrant sur les échanges de messages entre les objets (depuis UML 2.x).
- Diagramme global d'interaction (interaction overviewdiagram) : représentation des enchaînements possibles entre les scénarios préalablement identifiés sous forme de diagrammes de séquences (variante du diagramme d'activité) (depuis UML 2.x).
- Diagramme de temps (timing diagram) : représentation des variations d'une donnée au cours du temps (depuis UML 2.3).

Diagramme	Étape du cycle en V
1. Diagramme de cas d'utilisation	Spécification, cahier des charges
2. Diagramme de séquence	
3. Diagramme d'activité (processus métiers)	
4. Diagramme d'activité (cinématique et/ou processus applicatifs)	
5. Diagramme de classe	Conception architecturale
6. Diagramme d'objets	
7. Diagramme de communication	
8. Diagramme de déploiement	
9. Diagramme de composants	

Table 3-1 : Exemple de séquence de création des diagrammes

3.2.2. XMind

XMind est un logiciel de cartographie conceptuelle et de brainstorming développé par XMind Ltd. En plus des éléments de gestion, le logiciel peut être utilisé pour capturer des idées, clarifier la pensée, gérer des informations complexes et promouvoir la collaboration en équipe. En avril 2013, XMind avait été sélectionné comme logiciel de cartographie de l'esprit le plus populaire sur Lifehacker.

Il prend en charge les cartes mentales, les diagrammes en arête de poisson, les arborescences, les organigrammes, les feuilles de calcul, etc. Il est normalement utilisé pour la gestion des connaissances, les comptes rendus de réunion, la gestion des tâches et GTD. Pendant ce temps, XMind peut lire les fichiers FreeMind et MindManager, puis les enregistrer sur Evernote. Pour XMind Pro / Zen, il peut exporter les cartes mentales dans des documents Microsoft Word, PowerPoint, Excel, PDF, FreeMind et MindjetMindManager. [9]

3.2.3. Modelio

Successeur de l'atelier Objectteering, Modelio est un outil de modélisation UML disponible sur les plates-formes Windows, Linux et Mac. Il intègre également la modélisation BPMN, et le support de la modélisation des exigences, du dictionnaire, des règles métier et des objectifs.

Modelio propose une gamme d'outils étendant ses fonctionnalités permettant, entre autres, la mise en œuvre de l'approche MDA. [9]

3.2.4. Toad Data Modeler

Toad Data Modeler est un outil de conception de base de données permettant aux utilisateurs de créer, gérer et documenter de manière visuelle des systèmes de base de données nouveaux ou existants, ainsi que de déployer les modifications apportées aux structures de données sur différentes plates-formes. Il est utilisé pour construire des modèles de données logiques et physiques, comparer et synchroniser des modèles, générer du SQL / DDL complexe, créer et modifier des scripts, et inverser et transférer des bases de données et des systèmes d'entrepôt de données. Le logiciel de modélisation de données de Toad est utilisé pour la conception, la maintenance et la documentation de bases de données. [9]

3.3. La conception du système

3.3.1. Réalisation d'un journal électronique

Un journal en ligne, journal électronique, ou encore e-journal est un journal publié sur internet qui s'intéresse à toutes les informations politiques, économiques, culturelles ou sportives. La plupart des journaux électroniques sont gratuits, mais certains exigent de s'inscrire ou de s'abonner pour accéder à toutes les actualités.

La majorité des journaux en ligne s'appuient sur les formats HTML et PDF parfois utilisés indépendamment. Un faible nombre font appel à Microsoft Word pour la publication de textes. Quant aux formats MP4, Flash Player ou Silverlight ils sont utilisés pour la publication de vidéos.

Une grande majorité des journaux électroniques sont financés grâce aux annonces et aux publicités.

Ce type de publication en ligne s'inscrit dans le concept encore récent de révolution des médias.

Ce type de journaux a fait son apparition dans les années 1990 grâce au développement d'Internet. Ce développement s'est fortement accéléré à partir du début des années 2000 en raison de la popularisation de l'accès à l'ADSL, aux technologies mobiles et au WEB 2.0, jusqu'à toucher aujourd'hui plus de lecteurs que les journaux papier.

3.3.2. Description du système

Le projet consiste à réaliser un support électronique afin de permettre aux internautes de faire le suivant :

1. La visualisation de la liste d'articles les plus récents
2. La visualisation de la liste d'articles par rubrique
3. La visualisation d'un seul article afin de le lire

Ce support électronique doit aussi assurer la gestion de ces articles comme suit :

1. La rédaction des articles qui permet aux différents rédacteurs d'écrire leurs articles et de les sauvegarder sous forme de brouillons jusqu'à terminer la rédaction.
2. La validation des articles qui permet au chef rédacteur de visualiser les articles dont

leur rédaction est terminée afin de les vérifier.

3. La publication des articles qui soit permet au chef rédacteur la publication des articles favorables à la publication, soit lui permet de retourner un ou plusieurs articles à la modification, soit de supprimer des articles.

Le support électronique doit offrir le nécessaire pour une administration fiable comme suit :

1. Une gestion de comptes utilisateurs fiable.
2. Le nécessaire pour intervenir sur les données du support électronique dans le but d'effectuer de la maintenance.

3.4. Fonctionnement du système

3.4.1. Définition des acteurs

Ils sont des entités, externes ou internes, qui interagissent avec le système, comme une personne humaine ou un robot. Une même personne (ou robot) peut être plusieurs acteurs pour un système, c'est pourquoi les acteurs doivent surtout être décrits par leur rôle, ce rôle décrit les besoins et les capacités de l'acteur. Un acteur agit sur le système. L'activité du système a pour objectif de satisfaire les besoins de l'acteur. Les acteurs sont représentés par un pictogramme humanoïde (stick man) sous-titré par le nom de son rôle.[3][4]

En se basant sur les besoins précédemment décrits, nous constatons l'existence des acteurs suivants :

Acteur	Description
Lecteur	un utilisateur du support électronique ne possédant aucun ne compte et qu'il ne fait que consulter des articles
Redacteur	un utilisateur du support électronique possédant un compte qu'il utilise dans le but de faire la rédaction d'articles
ChefRedacteur	un utilisateur du support électronique possédant un compte qu'il utilise dans le but de gérer la publication des articles
Administrateur	un utilisateur du support électronique possédant un compte qu'il utilise pour des fins administratives

Table 3.2 : Les acteurs du système

3.4.2. Interaction utilisateur-système

Ils permettent de décrire l'interaction entre l'acteur et le système. L'idée forte est de dire que l'utilisateur d'un système logiciel a un objectif quand il utilise le système. Le cas d'utilisation est une description des interactions qui vont permettre à l'acteur d'atteindre son objectif en utilisant le système. Les cas d'utilisation sont représentés par une ellipse sous-titrée par le nom du cas d'utilisation (éventuellement le nom est placé dans l'ellipse). Un acteur et un cas d'utilisation sont mis en relation par une association représentée par une ligne.[3][4]

Le diagramme de cas d'utilisation

Les diagrammes de cas d'utilisation sont des diagrammes UML utilisés pour donner une vision globale du comportement fonctionnel d'un système logiciel. Ils sont utiles pour des présentations auprès de la direction ou des acteurs d'un projet, mais pour le développement, les cas d'utilisation sont plus appropriés. Un cas d'utilisation représente une unité discrète d'interaction entre un utilisateur (humain ou machine) et un système. Il est une unité significative de travail. Dans un diagramme de cas d'utilisation, les utilisateurs sont appelés acteurs, ils interagissent avec les cas d'utilisation (use cases).

UML définit une notation graphique pour représenter les cas d'utilisation, cette notation est appelée diagramme de cas d'utilisation. UML ne définit pas de standard pour la forme écrite de ces cas d'utilisation, et en conséquence il est aisé de croire que cette notation graphique suffit à elle seule pour décrire la nature d'un cas d'utilisation. Dans les faits, une notation graphique peut seulement donner une vue générale simplifiée d'un cas ou d'un ensemble de cas d'utilisation. Les diagrammes de cas d'utilisation sont souvent confondus avec les cas d'utilisation. Bien que ces deux concepts soient reliés, les cas d'utilisation sont bien plus détaillés que les diagrammes de cas d'utilisation.

Trois types de relations sont pris en charge par la norme UML et sont graphiquement représentées par des types particuliers de ces relations. Les relations indiquent que le cas d'utilisation source présente les mêmes conditions d'exécution que le cas issu. Une relation simple entre un acteur et une utilisation est un trait simple.

- **Inclusions** : Dans ce type d'interaction, le premier cas d'utilisation inclut le second et son issue dépend souvent de la résolution du second. Ce type de description est utile pour extraire un ensemble de sous-comportements communs à plusieurs tâches, comme une macro en programmation. Elle est représentée par une flèche en pointillé et le terme

include.

- Extensions : Les extensions (extend) représentent des prolongements logiques de certaines tâches sous certaines conditions. Autrement dit un cas d'utilisation A étend un cas d'utilisation B lorsque le cas d'utilisation A peut être appelé au cours de l'exécution du cas d'utilisation B. Elle est représentée par une flèche en pointillée avec le terme extend. Ce type de relation peut être utile pour traiter des cas particuliers ou fonctions optionnelles, préciser les objectifs, ou encore pour tenir compte de nouvelles exigences au cours de la maintenance du système et de son évolution.
- Généralisations : La troisième relation est la relation de généralisation ou spécialisation. Le cas d'utilisation A est une généralisation de B, si B est un cas particulier de A c'est-à-dire lorsque A peut-être substitué par B pour un cas précis. Ces relations sont des traits pleins terminés par une flèche en triangle.

Relations entre acteurs : Il est également possible d'appliquer à un acteur la relation de généralisation. Cela se fait notamment lorsqu'un acteur est un sous-type d'une autre catégorie d'acteurs. Un acteur lié à un autre par un ce type de relation peut interagir avec le système de plus de manières que son parent.

Autres éléments : Les réalisations décrivent un cas d'utilisation par une suite de collaborations d'autres éléments du modèle de données. Ce type de schématisation n'est pas propre à l'UML mais constitue un des éléments du Processus Unifié.

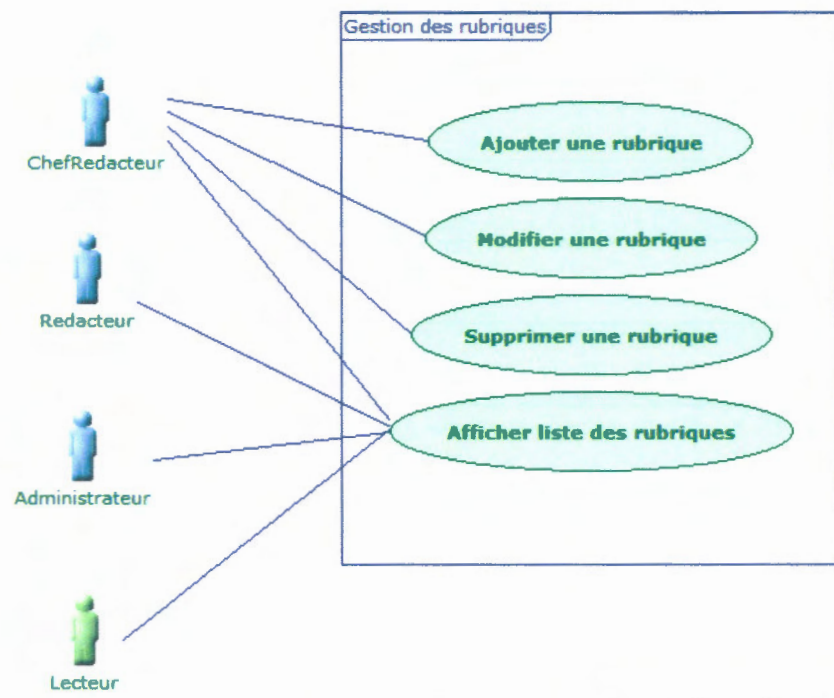


Figure 3-1 : Diagramme de cas d'utilisation de la gestion des rubriques

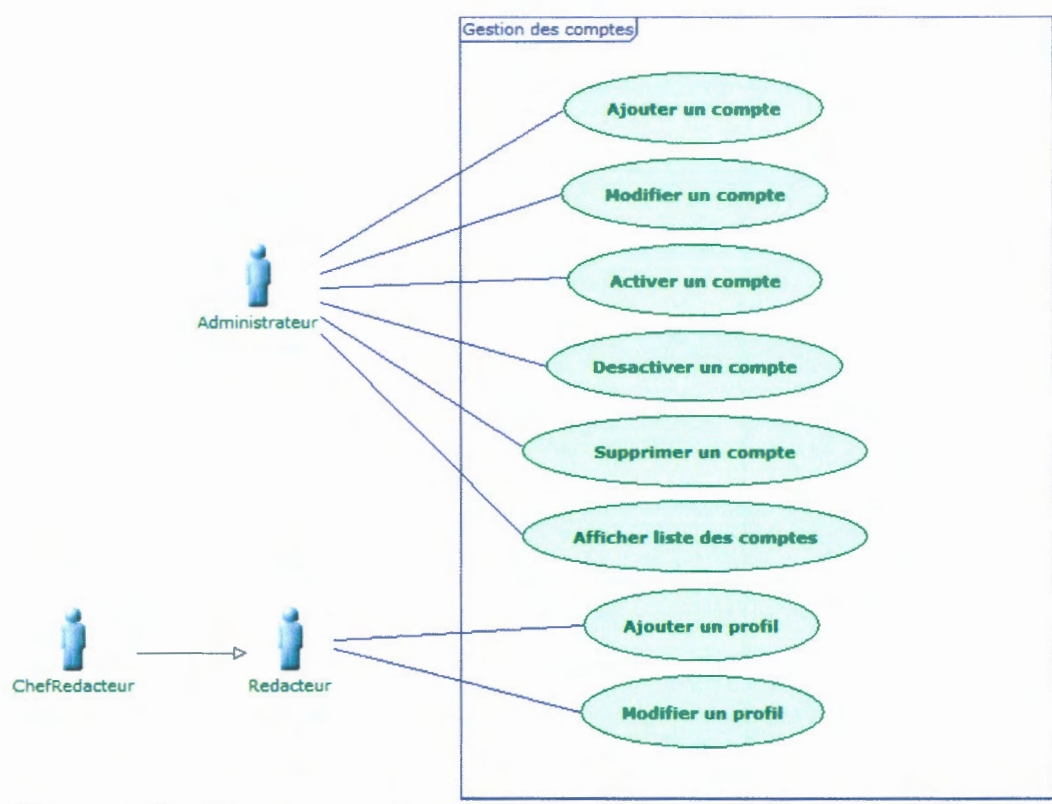


Figure 3-2 : Diagramme de cas d'utilisation de la gestion des comptes

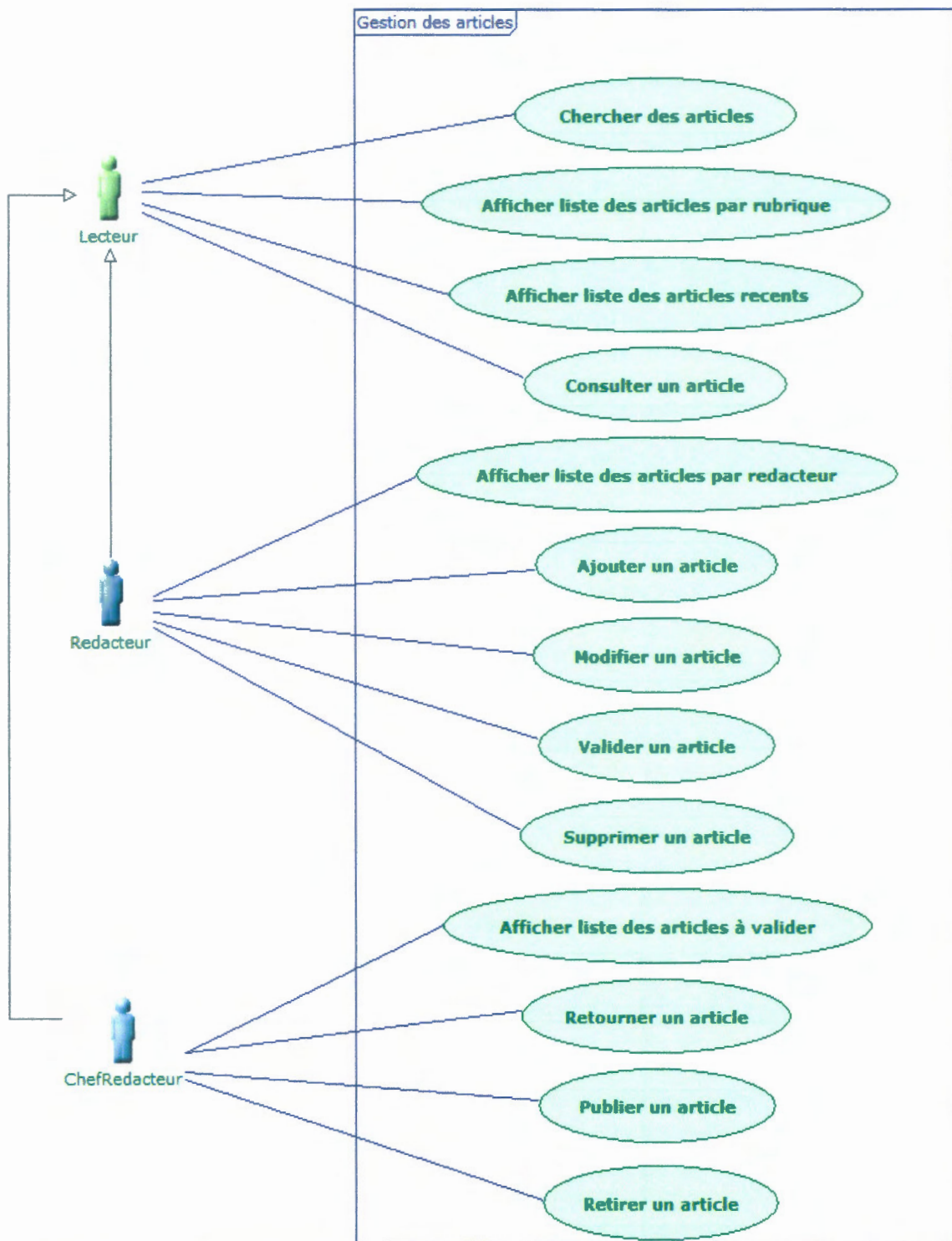


Figure 3-3 : Diagramme de cas d'utilisation de la gestion des articles

3.5. Architecture générale

3.5.1. Cycle de vie d'un article

Le cycle de vie d'un article est illustré à la figure 3-4.

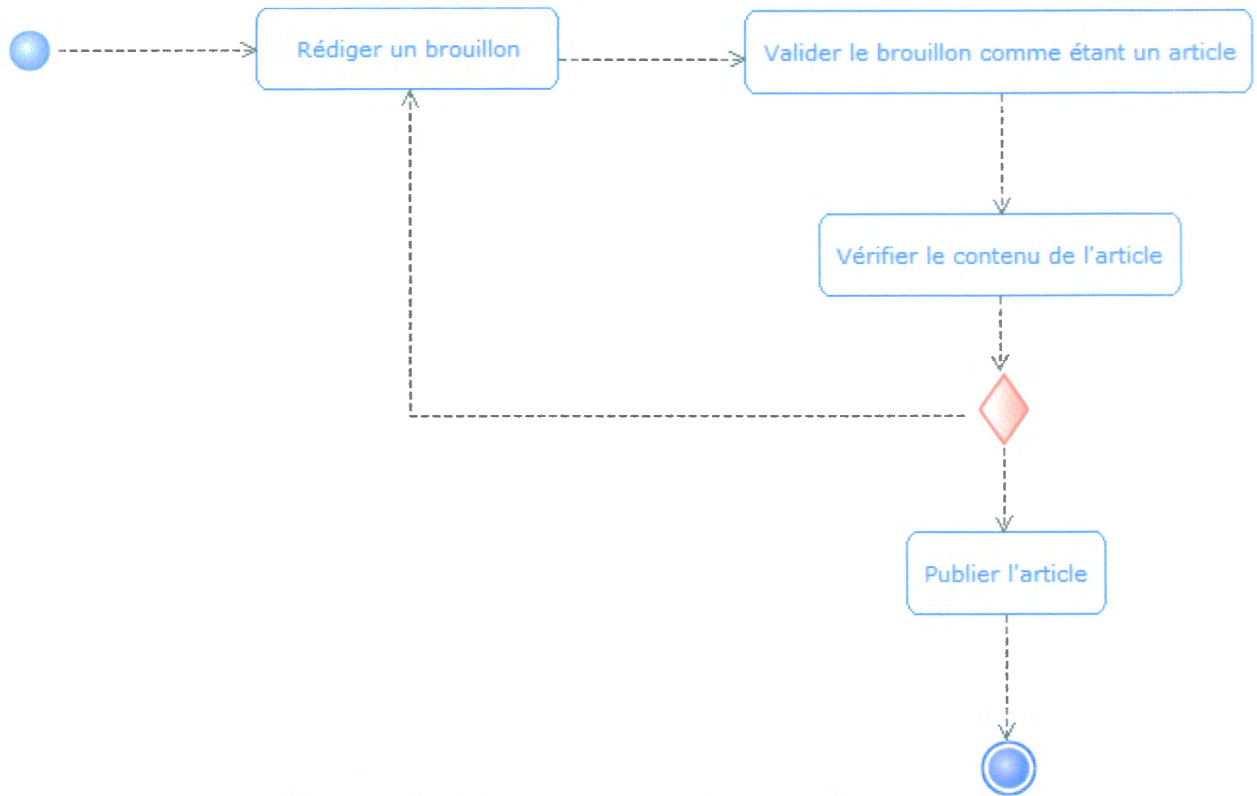


Figure 3.4 : Cycle de vie d'un article

3.5.2. Diagramme de concepts saillants

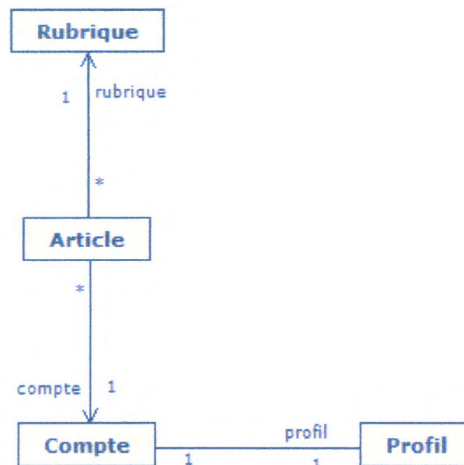


Figure 3.5 : Les concepts saillants du système

Nom	Description
-> : [0..*] <u>Article</u>	Une rubrique peut ne pas avoir d'article comme elle peut avoir plusieurs
	Un article appartient à une et une seule rubrique

Table 3.3 : Associations de la classe "Rubric"

Nom	Description
->account : [1..1] <u>Account</u>	Un compte peut ne pas rédiger d'article comme il peut rédiger plusieurs.
	Un article est rédigé par un et un seul compte.
->rubric : [1..1] <u>Rubric</u>	Une rubrique peut ne pas avoir d'article comme elle peut avoir plusieurs
	Un article appartient à une et une seule rubrique

Table 3.4 : Associations de la classe "Article"

Nom	Description
-> : [0..*] <u>Article</u>	Un compte peut ne pas rédiger d'article comme il peut rédiger plusieurs.
	Un article est rédigé par un et un seul compte.
->profil : [1..1] <u>Profil</u>	Un compte peut ne pas avoir de profil comme il peut en avoir un seul.
	Un profil appartient à un et un seul compte.

Table 3.5 : Associations de la classe "Account"

Nom	Description
-> : [1..1] <u>Account</u>	Un compte peut ne pas avoir de profil comme il peut en avoir un seul. Un profil appartient à un et un seul compte.

Table 3.6 : Associations de la classe "Profil"

3.6. Architecture détaillée

3.6.1. Diagramme de classe

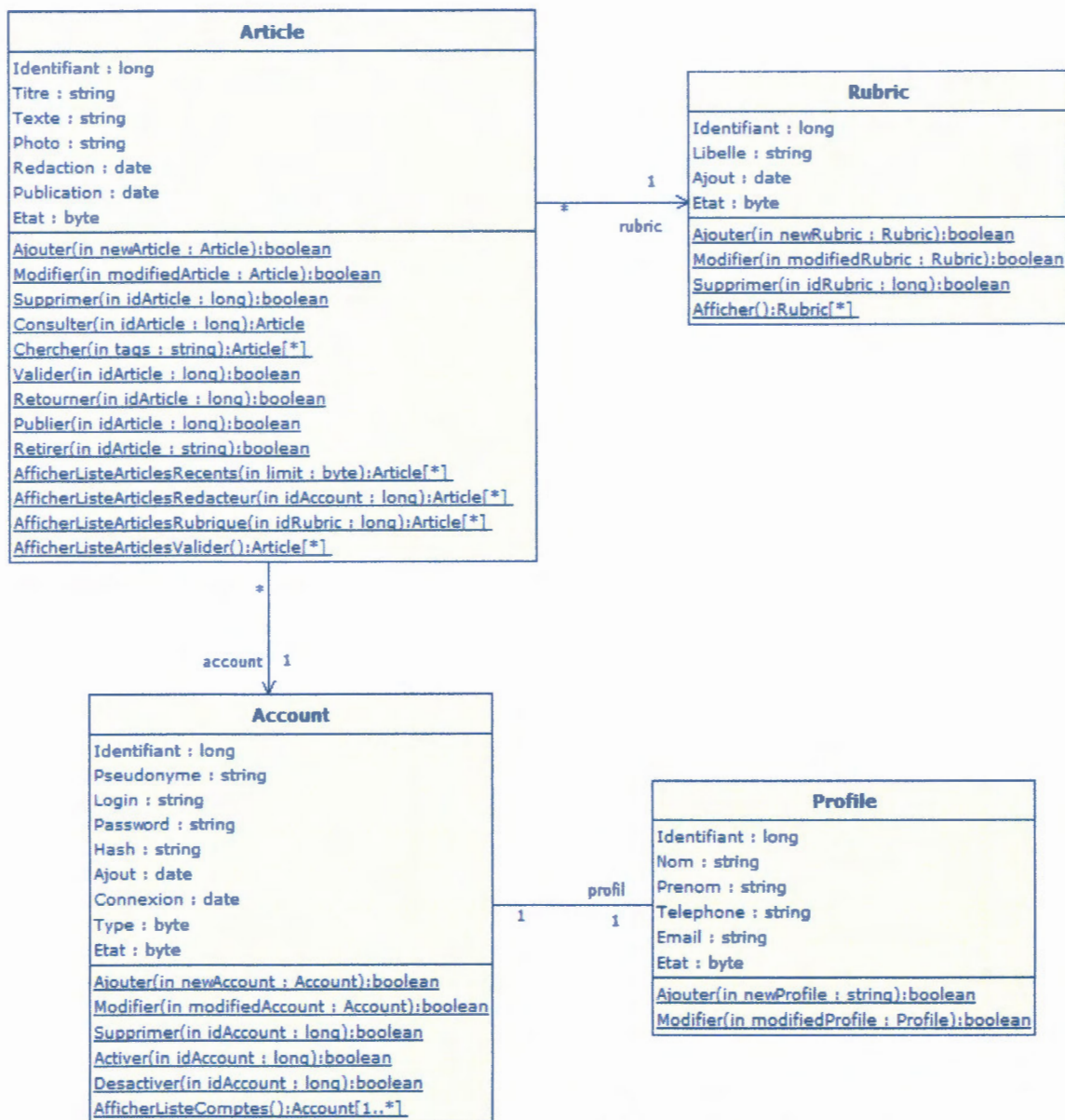


Figure 3.6 : Diagramme de classe de Journal électronique

Classe "Rubric"

Classe contenant la définition du schéma des objets de type Rubrique

Nom	Description
Identifiant : [1..1] long	Identifiant de la rubrique
Libelle : [1..1] string	Libellé ou nom de la rubrique
Ajout : [1..1] date	Date d'ajout de la rubrique
Etat : [1..1] byte	Etat de la rubrique (0: Désactivée, 1: Activée)

Table 3.7 : Attributs de la classe "Rubric"

Nom	Description
booleanAjouter (INnewRubricRubric)	Méthode pour ajouter une nouvelle rubrique.
booleanModifier (INmodifiedRubricRubric)	Méthode pour modifier les données d'une rubrique existante.
booleanSupprimer (INidRubriclong)	Méthode pour supprimer une rubrique existante.
RubricAfficher ()	Méthode pour afficher la liste des rubriques.

Table 3.8 : Méthodes de la classe "Rubric"

Classe "Article"

Classe contenant la définition du schéma des objets de type Article

Nom	Description
Identifiant : [1..1] long	Identifiant de l'article
Numero : [1..1] string	Numéro de l'article

Nom	Description
Titre : [1..1] string	Titre de l'article
Texte : [1..1] string	Texte de l'article
Photo : [1..1] string	Lien de la photo associée à l'article
Redaction : [1..1] date	Date de rédaction de l'article
Publication : [1..1] date	Date de publication de l'article
Etat : [1..1] byte	Etat de l'article : 0: Retiré/Bloqué/Supprimé/Caché 1: Brouillon 2 : Article validé 3 : Article publié

Table 3.9 : Attributs de la classe "Article"

Nom	Description
booleanAjouter (INnewArticleArticle)	Méthode pour ajouter un nouvel article.
booleanModifier (INmodifiedArticleArticle)	Méthode pour modifier un article existant.
booleanSupprimer (INidArticlelong)	Méthode pour supprimer un article existant.
ArticleConsulter (INidArticlelong)	Méthode pour afficher un article existant.

Nom	Description
ArticleChercher (INtagsstring)	.Méthode pour chercher des articles
booleanValider (INidArticlelong)	Méthode pour valider un article ayant l'Etat d'un brouillon.
booleanRetourner (INidArticlelong)	Méthode pour retourner un article à son rédacteur.
booleanPublier (INidArticlelong)	Méthode pour publier un article.
booleanRetirer (INidArticlestring)	Méthode pour retirer un article publié.
ArticleAfficherListeArticlesRecents (INlimitbyte)	Méthode pour afficher la liste des articles les plus récents.
ArticleAfficherListeArticlesRedacteur (INidAccountlong)	Méthode pour afficher la liste des articles d'un rédacteur.
ArticleAfficherListeArticlesRubrique (INidRubriclong)	Méthode pour afficher la liste des articles d'une rubrique.
ArticleAfficherListeArticlesValider ()	Méthode pour afficher la liste des articles à valider.

Table 3.10 : Méthodes de la classe "Article"

Classe "Account"

Classe contenant la définition du schéma des objets de type Compte

Nom	Description
Identifiant : [1..1] long	Identifiant du compte

Nom	Description
Pseudonyme : [1..1] string	Pseudonyme de l'utilisateur du compte
Login : [1..1] string	Login du compte
Password : [1..1] string	Mot de passe du compte
Hash : [1..1] string	La clé d'hashage du mot de passe du compte
Ajout : [1..1] date	Date d'ajout du compte
Connexion : [1..1] date	Dernière date de connexion du compte
Type : [1..1] byte	Type du compte : 1 : Rédacteur 2 : Chef rédacteur 3 : Administrateur
Etat : [1..1] byte	Etat du compte : 0 : Désactivé 1 : Activé

Table 3.11 : Attributs de la classe "Account"

Nom	Description
booleanAjouter (INnewAccountAccount)	Méthode pour ajouter un nouveau compte.
booleanModifier (INmodifiedAccountAccount)	Méthode pour modifier un compte existant.
booleanSupprimer (INidAccountlong)	Méthode pour supprimer un compte existant.

Nom	Description
booleanActiver (INidAccountlong)	Méthode pour activer un compte existant ayant été désactivé.
booleanDesactiver (INidAccountlong)	Méthode pour désactiver un compte activé.
AccountAfficherListeComptes ()	Méthode pour afficher la liste des comptes utilisateurs.

Table 3.12 : Méthodes de la classe "Account"

Classe "Profile"

Classe contenant la définition du schéma des objets de type Profile

Nom	Description
Identifiant : [1..1] long	Identifiant du profile
Nom : [1..1] string	Nom de l'utilisateur
Prenom : [1..1] string	Prénom de l'utilisateur
Telephone : [1..1] string	Numéro de téléphone de l'utilisateur
Email : [1..1] string	Adresse email de l'utilisateur
Etat : [1..1] byte	Etat du profile 0 : Désactivé 1 : Activé

Table 3.13 : Attributs de la classe "Profile"

Nom	Description
booleanAjouter (INnewProfilestring)	Méthode pour créer le profil d'un nouveau compte.
booleanModifier (INmodifiedProfileProfile)	Méthode pour modifier les données d'un profil existant.

Table 3.14 : Méthodes de la classe "Profile"

3.6.2. Augmentation de la sécurité

Premièrement, et dans le but d'augmenter le niveau de la sécurité, nous avons changé le nommage en anglais tout en ajoutant un caractère spécial « _ » à la fin de chaque nom comme illustré à la figure 3.7.

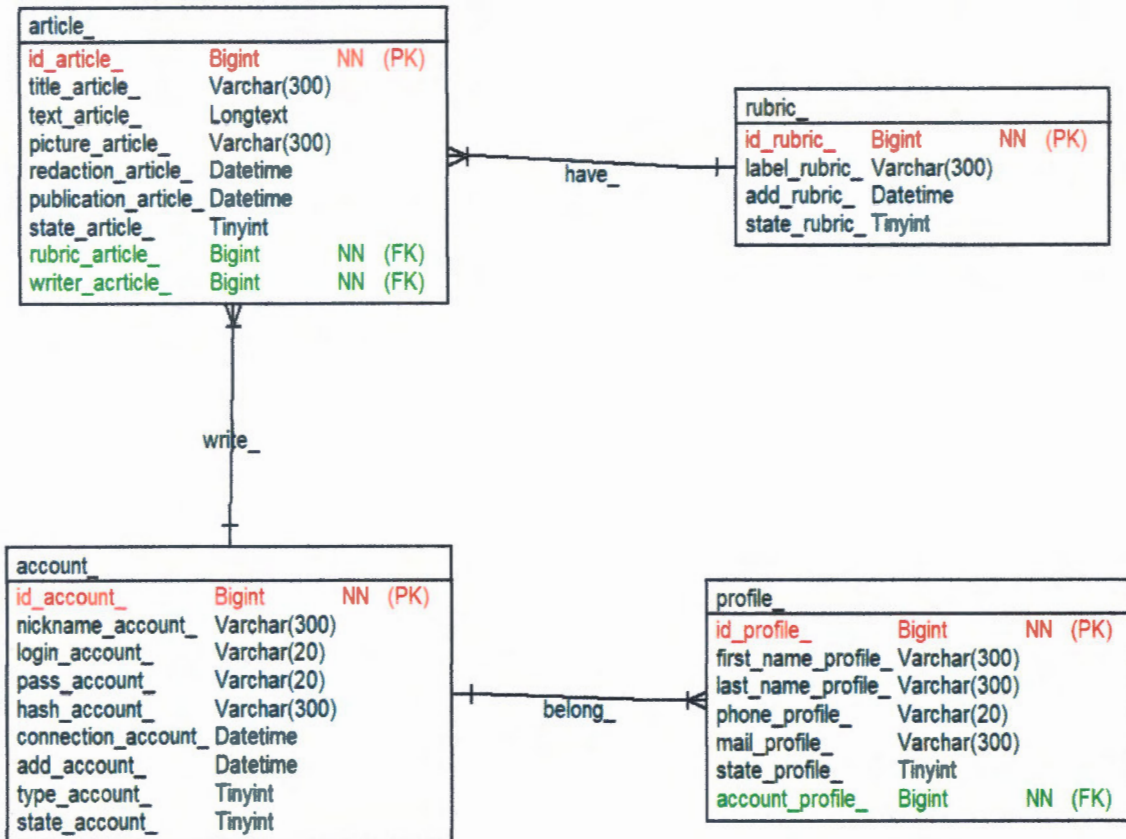


Figure 3.7 : Le nommage en anglais

```

CREATE TABLE `account_` (
  `id_account_` bigint(20) NOT NULL,
  `nickname_account_` varchar(300) COLLATE utf8_unicode_ci DEFAULT NULL,
  `login_account_` varchar(20) COLLATE utf8_unicode_ci DEFAULT NULL,
  `pass_account_` varchar(20) COLLATE utf8_unicode_ci DEFAULT NULL,
  `hash_account_` varchar(300) COLLATE utf8_unicode_ci DEFAULT NULL,
  `connection_account_` datetime DEFAULT NULL,
  `add_account_` datetime DEFAULT NULL,
  `type_account_` tinyint(4) DEFAULT NULL,
  `state_account_` tinyint(4) DEFAULT NULL
) ENGINE=MyISAM DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

CREATE TABLE `article_` (
  `id_article_` bigint(20) NOT NULL,
  `title_article_` varchar(300) COLLATE utf8_unicode_ci DEFAULT NULL,
  `text_article_` longtext COLLATE utf8_unicode_ci,
  `picture_article_` varchar(300) COLLATE utf8_unicode_ci DEFAULT NULL,
  `redaction_article_` datetime DEFAULT NULL,
  `publication_article_` datetime DEFAULT NULL,
  `state_article_` tinyint(4) DEFAULT NULL,
  `rubric_article_` bigint(20) NOT NULL,
  `writer_acrticle_` bigint(20) NOT NULL
) ENGINE=MyISAM DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

CREATE TABLE `profile_` (
  `id_profile_` bigint(20) NOT NULL,
  `first_name_profile_` varchar(300) COLLATE utf8_unicode_ci DEFAULT NULL,
  `last_name_profile_` varchar(300) COLLATE utf8_unicode_ci DEFAULT NULL,
  `phone_profile_` varchar(20) COLLATE utf8_unicode_ci DEFAULT NULL,
  `mail_profile_` varchar(300) COLLATE utf8_unicode_ci DEFAULT NULL,
  `state_profile_` tinyint(4) DEFAULT NULL,
  `account_profile_` bigint(20) NOT NULL
) ENGINE=MyISAM DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

CREATE TABLE `rubric_` (
  `id_rubric_` bigint(20) NOT NULL,
  `label_rubric_` varchar(300) COLLATE utf8_unicode_ci DEFAULT NULL,
  `add_rubric_` datetime DEFAULT NULL,
  `state_rubric_` tinyint(4) DEFAULT NULL
) ENGINE=MyISAM DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

CREATE VIEW `article` AS
select
  `article_`.`id_article_` AS `id_article`,
  `article_`.`title_article_` AS `titre_article`,
  `article_`.`text_article_` AS `texte_article`,
  `article_`.`picture_article_` AS `photo_article`,
  `article_`.`redaction_article_` AS `redaction_article`,
  `article_`.`publication_article_` AS `publication_article`,
  `article_`.`state_article_` AS `etat_article`,
  `article_`.`rubric_article_` AS `rubrique_article`,
  `article_`.`writer_acrticle_` AS `redacteur_article`
from `article_`;

CREATE VIEW `compte` AS

```

```

select
  `account_`.`id_account_` AS `id_compte`,
  `account_`.`nickname_account_` AS `pseudo_compte`,
  `account_`.`login_account_` AS `login_compte`,
  `account_`.`pass_account_` AS `pass_compte`,
  `account_`.`hash_account_` AS `hash_compte`,
  `account_`.`connection_account_` AS `connexion_compte`,
  `account_`.`add_account_` AS `ajout_compte`,
  `account_`.`type_account_` AS `type_compte`,
  `account_`.`state_account_` AS `etat_compte`
from `account_` ;

```

CREATE VIEW `profil` AS

```

select
  `profile_`.`id_profile_` AS `id_profil`,
  `profile_`.`first_name_profile_` AS `nom_profil`,
  `profile_`.`last_name_profile_` AS `prenom_profil`,
  `profile_`.`phone_profile_` AS `telephone_profil`,
  `profile_`.`mail_profile_` AS `email_profil`,
  `profile_`.`state_profile_` AS `etat_profil`,
  `profile_`.`account_profile_` AS `compte_profil`
from `profile_` ;

```

CREATE VIEW `rubrique` AS

```

select
  `rubric_`.`id_rubric_` AS `id_rubrique`,
  `rubric_`.`label_rubric_` AS `libelle_rubrique`,
  `rubric_`.`add_rubric_` AS `ajout_rubrique`,
  `rubric_`.`state_rubric_` AS `etat_rubrique`
from `rubric_` ;

```

CREATE VIEW `all_data_acc_` AS

```

select
  `c`.`id_compte` AS `id_compte`,
  `c`.`pseudo_compte` AS `pseudo_compte`,
  `c`.`login_compte` AS `login_compte`,
  `c`.`pass_compte` AS `pass_compte`,
  `c`.`hash_compte` AS `hash_compte`,
  `c`.`connexion_compte` AS `connexion_compte`,
  `c`.`ajout_compte` AS `ajout_compte`,
  `c`.`type_compte` AS `type_compte`,
  `c`.`etat_compte` AS `etat_compte`,
  `p`.`id_profil` AS `id_profil`,
  `p`.`nom_profil` AS `nom_profil`,
  `p`.`prenom_profil` AS `prenom_profil`,
  `p`.`telephone_profil` AS `telephone_profil`,
  `p`.`email_profil` AS `email_profil`,
  `p`.`etat_profil` AS `etat_profil`,
  `p`.`compte_profil` AS `compte_profil`
from (`compte` `c` join `profil` `p`)
where (`p`.`compte_profil` = `c`.`id_compte`) ;

```

CREATE VIEW `all_data_art_` AS

```

select
  `r`.`id_rubrique` AS `id_rubrique`,

```

```

`r`.`libelle_rubrique` AS `libelle_rubrique`,
`r`.`ajout_rubrique` AS `ajout_rubrique`,
`r`.`etat_rubrique` AS `etat_rubrique`,
`a`.`id_article` AS `id_article`,
`a`.`titre_article` AS `titre_article`,
`a`.`texte_article` AS `texte_article`,
`a`.`photo_article` AS `photo_article`,
`a`.`redaction_article` AS `redaction_article`,
`a`.`publication_article` AS `publication_article`,
`a`.`etat_article` AS `etat_article`,
`a`.`rubrique_article` AS `rubrique_article`,
`a`.`redacteur_article` AS `redacteur_article`,
`c`.`pseudo_compte` AS `pseudo_compte`
from ((`rubrique` `r` join `article` `a`) join `compte` `c`)
where
((`r`.`id_rubrique` = `a`.`rubrique_article`) and
(`a`.`redacteur_article` = `c`.`id_compte`));

ALTER TABLE `account_` ADD PRIMARY KEY (`id_account_`);

ALTER TABLE `article_` ADD PRIMARY KEY (`id_article_`),
ADD KEY `have_` (`rubric_article_`),
ADD KEY `write_` (`writer_acrticle_`);

ALTER TABLE `profile_` ADD PRIMARY KEY (`id_profile_`),
ADD KEY `belong_` (`account_profile_`);

ALTER TABLE `rubric_` ADD PRIMARY KEY (`id_rubric_`);

ALTER TABLE `account_` MODIFY `id_account_` bigint(20) NOT NULL
AUTO_INCREMENT;

ALTER TABLE `article_` MODIFY `id_article_` bigint(20) NOT NULL
AUTO_INCREMENT;

ALTER TABLE `profile_` MODIFY `id_profile_` bigint(20) NOT NULL
AUTO_INCREMENT;

ALTER TABLE `rubric_` MODIFY `id_rubric_` bigint(20) NOT NULL
AUTO_INCREMENT;

```

3.7. Conclusion

Nous avons présenté dans ce chapitre les outils nécessaires pour achever notre projet, ensuite, nous avons donné la conception de notre système et ses différents diagrammes de cas d'utilisation. Nous avons aussi fourni une architecture générale suivie par une autre détaillée en illustrant le mécanisme interne de notre système qui a abouti à la génération d'un script SQL utilisé pour renforcer la sécurité de la base de données de notre système, ce script sera utilisé dans le codage, sujet de notre chapitre suivant.

Chapitre 4 : Implémentation

4.1. Introduction

Dans ce chapitre nous allons commencer d'abord, par une brève illustration de l'environnement de travail ainsi que l'ensemble des logiciels que nous avons utilisé dans la réalisation de l'application et l'implémentation de la base de données d'où nous allons réaliser un dernier complément de sécurité. Puis, nous passons en aperçu les interfaces les plus importantes du journal électronique.

4.2. Choix technique

4.2.1. HTML

L'HyperText Markup Language, généralement abrégé HTML, est le langage de balisage conçu pour représenter les pages web. C'est un langage permettant d'écrire de l'hypertexte, d'où son nom. HTML permet également de structurer sémantiquement et logiquement et de mettre en forme le contenu des pages, d'inclure des ressources multimédias dont des images, des formulaires de saisie et des programmes informatiques. Il permet de créer des documents interopérables avec des équipements très variés de manière conforme aux exigences de l'accessibilité du web.

4.2.2. CSS

Les feuilles de style en cascade¹, généralement appelées CSS de l'anglais Cascading Style Sheets, forment un langage informatique qui décrit la présentation des documents HTML et XML. Les standards définissant CSS sont publiés par le World Wide Web Consortium (W3C). Introduit au milieu des années 1990, CSS devient couramment utilisé dans la conception de sites web et bien pris en charge par les navigateurs web dans les années 2000.

4.2.3. JavaScript

JavaScript est un langage de programmation de scripts principalement employé dans les pages web interactives mais aussi pour les serveurs avec l'utilisation (par exemple) de Node.js. C'est un langage orienté objet à prototype, c'est-à-dire que les bases du langage et ses principales interfaces sont fournies par des objets qui ne sont pas des instances de classes, mais qui sont chacun équipés de constructeurs permettant de créer leurs propriétés, et notamment une propriété de prototypage qui permet d'en créer des objets héritiers personnalisés. En outre, les fonctions sont des objets de première classe. Le langage supporte le paradigme objet, impératif et fonctionnel. JavaScript est le langage possédant le

plus large écosystème grâce à son gestionnaire de dépendances npm, avec environ 500 000 paquets en août 2017.

JavaScript a été créé en 1995 par Brendan Eich. Il a été standardisé sous le nom d'ECMAScript en juin 1997 par Ecma International dans le standard ECMA-262. Le standard ECMA-262 en est actuellement à sa 8e édition. JavaScript n'est depuis qu'une implémentation d'ECMAScript, celle mise en œuvre par la fondation Mozilla. L'implémentation d'ECMAScript par Microsoft (dans Internet Explorer jusqu'à sa version 9) se nomme JScript, tandis que celle d'Adobe Systems se nomme ActionScript.

Avec les technologies HTML et CSS, JavaScript est parfois considéré comme l'une des technologies cœur du World Wide Web⁵. Le langage JavaScript permet des pages web interactives, et à ce titre est une partie essentielle des applications web. Une grande majorité des sites web l'utilisent⁶, et la majorité des navigateurs web disposent d'un moteur JavaScript dédié pour l'interpréter, indépendamment des considérations de sécurité qui peuvent se poser le cas échéant.

4.2.4. jQuery

jQuery est une bibliothèque JavaScript libre et multiplateforme créée pour faciliter l'écriture de scripts côté client dans le code HTML des pages web². La première version est lancée en janvier 2006 par John Resig.

Le but de la bibliothèque étant le parcours et la modification du DOM (y compris le support des sélecteurs CSS 1 à 3 et un support basique de XPath), elle contient de nombreuses fonctionnalités ; notamment des animations, la manipulation des feuilles de style en cascade (accessibilité des classes et attributs), la gestion des événements, etc. L'utilisation d'Ajax est facilitée et de nombreux plugins sont présents.

Depuis sa création en 2006 et notamment à cause de la complexification croissante des interfaces Web, jQuery a connu un large succès auprès des développeurs Web et son apprentissage est aujourd'hui un des fondamentaux de la formation aux technologies du Web. Il est à l'heure actuelle la bibliothèque front-end la plus utilisée au monde (plus de la moitié des sites Internet en ligne intègrent jQuery).

4.2.5. Ajax

Ajax (de l'anglais Asynchronous JavaScript And XML) est un ensemble de techniques découplant l'échange de données entre le navigateur et le serveur web de l'affichage d'une page web, ce qui permet de modifier le contenu des pages web sans les recharger. Grâce à l'objet JavaScript XMLHttpRequest, cette méthode permet d'effectuer des requêtes HTTP sur le serveur web depuis le navigateur web, et permet également de traiter les réponses HTTP du serveur web pour modifier le contenu de la page web. La réponse était en général au format XML qui tend aujourd'hui à être remplacé par le format JSON qui a l'avantage d'être natif en JavaScript. Le script manipule l'ensemble d'objets DOM qui représente le contenu de la page web. Les technologies XMLHttpRequest, XML et DOM ont été ajoutées aux navigateurs web entre 1995 et 2005. La méthode Ajax permet de réaliser des applications Internet riches, offrant une maniabilité et un confort supérieur ; c'est un des sujets phares du mouvement Web 2.0.

4.2.6. PHP

Hypertext Preprocessor, plus connu sous son sigle PHP (sigle auto-référentiel), est un langage de programmation libre⁶, principalement utilisé pour produire des pages Web dynamiques via un serveur HTTP, mais pouvant également fonctionner comme n'importe quel langage interprété de façon locale. PHP est un langage impératif orienté objet.

PHP a permis de créer un grand nombre de sites web célèbres, comme Facebook, Wikipédia, etc. Il est considéré comme une des bases de la création de sites web dits dynamiques mais également des applications web. [7]

4.2.7. SQL

SQL (sigle de Structured Query Language, en français langage de requête structurée) est un langage informatique normalisé servant à exploiter des bases de données relationnelles. La partie langage de manipulation des données de SQL permet de rechercher, d'ajouter, de modifier ou de supprimer des données dans les bases de données relationnelles.

Outre le langage de manipulation des données, la partie langage de définition des données permet de créer et de modifier l'organisation des données dans la base de données, la partie langage de contrôle de transaction permet de commencer et de terminer des transactions, et la partie langage de contrôle des données permet d'autoriser ou d'interdire l'accès à certaines

données à certaines personnes.

Créé en 1974, normalisé depuis 1986, le langage est reconnu par la grande majorité des systèmes de gestion de bases de données relationnelles (abrégé SGBDR) du marché. [8]

4.3. Environnement logiciel

4.3.1. WAMPServer

WampServer (anciennement WAMP5) est une plateforme de développement Web de type WAMP, permettant de faire fonctionner localement (sans avoir à se connecter à un serveur externe) des scripts PHP. WampServer n'est pas en soi un logiciel, mais un environnement comprenant trois serveurs (Apache, MySQL et MariaDB), un interpréteur de script (PHP), ainsi que phpMyAdmin pour l'administration Web des bases MySQL.

Il dispose d'une interface d'administration permettant de gérer et d'administrer ses serveurs au travers d'un tray icon (icône près de l'horloge de Windows).

La grande nouveauté de WampServer 3 réside dans la possibilité d'y installer et d'utiliser n'importe quelle version de PHP, Apache, MySQL ou MariaDB en un clic. Ainsi, chaque développeur peut reproduire fidèlement son serveur de production sur sa machine locale.

4.3.2. Visual Studio Code

Visual Studio Code est un éditeur de code extensible développé par Microsoft pour Windows, Linux et macOS. Il est présenté lors de la conférence des développeurs Build d'avril 2015 comme un éditeur de code cross-platform, open source et gratuit, supportant une dizaine de langages.

Il est basé sur Electron, une structure utilisée pour déployer des applications Node.js pour le bureau exécuté sur le moteur Blink. Bien qu'il utilise le framework Electron, le logiciel n'utilise pas Atom mais utilise le même composant éditeur (nommé "Monaco") utilisé dans Azure DevOps (anciennement appelé Visual Studio Online et Visual Studio Team Services).

Le code source est fourni sous la licence libre MIT (plus précisément la licence Expat) sur le site du projet sur Github. En revanche, l'exécutable est proposé sur le site officiel de Microsoft sous une licence privative

Visual Studio Code prend immédiatement en charge presque tous les principaux langages de

programmation. Plusieurs d'entre eux sont inclus par défaut, par exemple JavaScript, TypeScript, CSS et HTML, mais d'autres extensions de langage peuvent être trouvées et téléchargées gratuitement à partir de VS Code Marketplace.

4.4. Environnement matériel

- Processeur : i7 intel inside
- Mémoire vive : 8 Go
- Disque dur : 1 To
- Système d'exploitation : Windows 10 pro 64 bits

4.5. Un petit complément de sécurité

4.5.1. Gestion des droits d'accès à la base de données

L'une des barrières qui protégeraient notre système est la gestion des droits d'accès à la base de données.

Il est établi que dans les nouvelles générations des SGBD, tout accès à une quelconque base de données doit se faire via des comptes utilisateurs.

De ce fait, il est fortement conseillé de créer trois comptes utilisateurs,

- le premier sera attribué à la partie exposée au grand public de notre site Web (Les internautes),
- le deuxième sera attribué à la gestion des données (Le rédacteur, Le chef rédacteur),
- le troisième sera attribué à l'administration de la base de données (L'administrateur)

Les privilèges de chaque compte

Le compte "std_journal_visitor" : SELECT + UPDATE + SHOW VIEW

Le compte "std_journal_user" : SELECT + UPDATE + DELETE +INSERT + SHOW VIEW

Le compte "std_journal_admin" : Tous les privilèges uniquement à la base de données du site Web nommée : std_journal_db

Création des comptes sous phpMyAdmin

phpMyAdmin (PMA) est une application Web de gestion pour les systèmes de gestion de base de données MySQL réalisée principalement en PHP et distribuée sous licence GNU GPL.

Etape 1 : Connexion à "root" en localhost

Connectez-vous en tant que 'root' avec le mot de passe du root MySQL (généralement vide)



The image shows the phpMyAdmin login page. At the top, it says 'phpMyAdmin' with a logo and 'Bienvenue dans phpMyAdmin'. Below this is a language selection dropdown menu currently set to 'Français - French'. Underneath is a 'Connexion' section with three input fields: 'Utilisateur' (containing 'root'), 'Mot de passe' (empty), and 'Choix du serveur' (set to 'MySQL'). An 'Exécuter' button is located at the bottom right of the form.

Figure 4-1 : Authentification sous phpMyAdmin

N. B. afin de sécuriser le compte administrateur "root" vous êtes dans l'obligation de lui attribuer un mot de passe

Etape 2 : Création des nouveaux utilisateurs

Sélectionnez la base de données existante "std_journal_db"

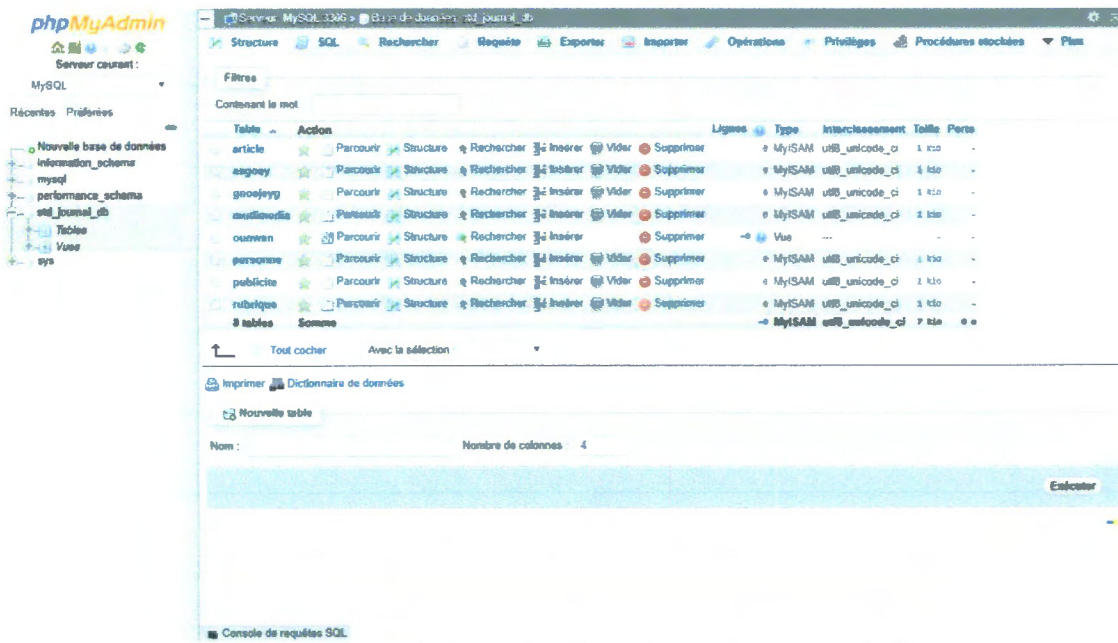
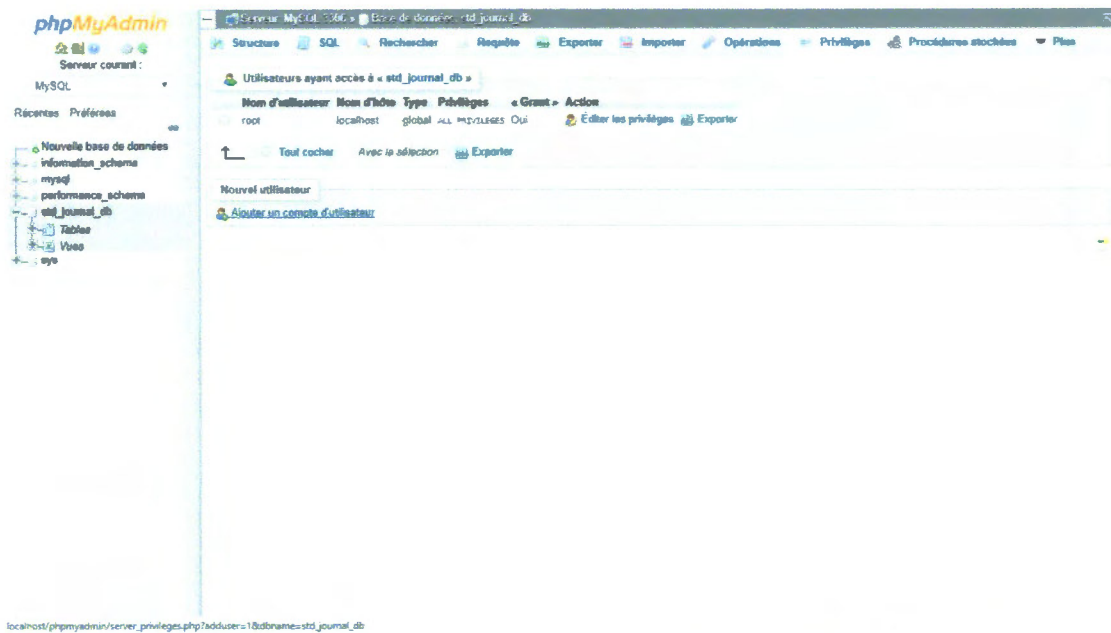


Figure 4-2 : la base de données std_journal_db

Cliquez sur "Privilèges", puis sur "Ajouter un utilisateur"



localhost/phpmyadmin/server_privileges.php?adduser=1&dbname=std_journal_db

Figure 4-3 : Les utilisateurs de la base de données std_journal_db

Entrez les informations de l'utilisateur

(std_journal_visitor/std_journal_user/std_journal_admin)

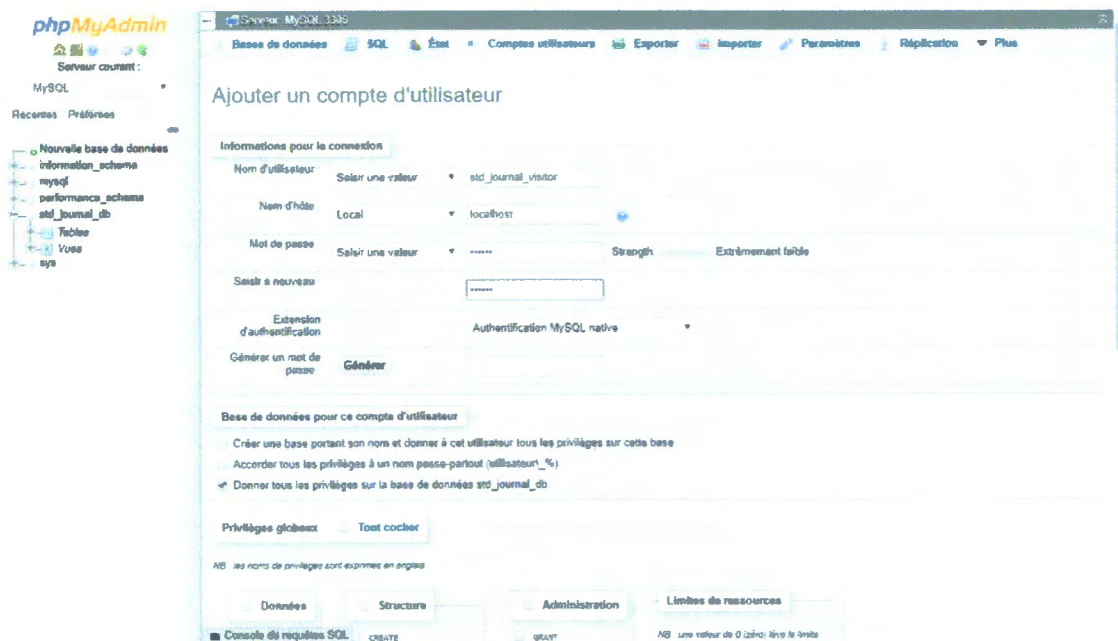


Figure 4-4 : Ajout d'un compte utilisateur

Dans la rubrique "Privileges globaux", ne cochez aucune case afin de restreindre les utilisateurs à la base de données sélectionnée.

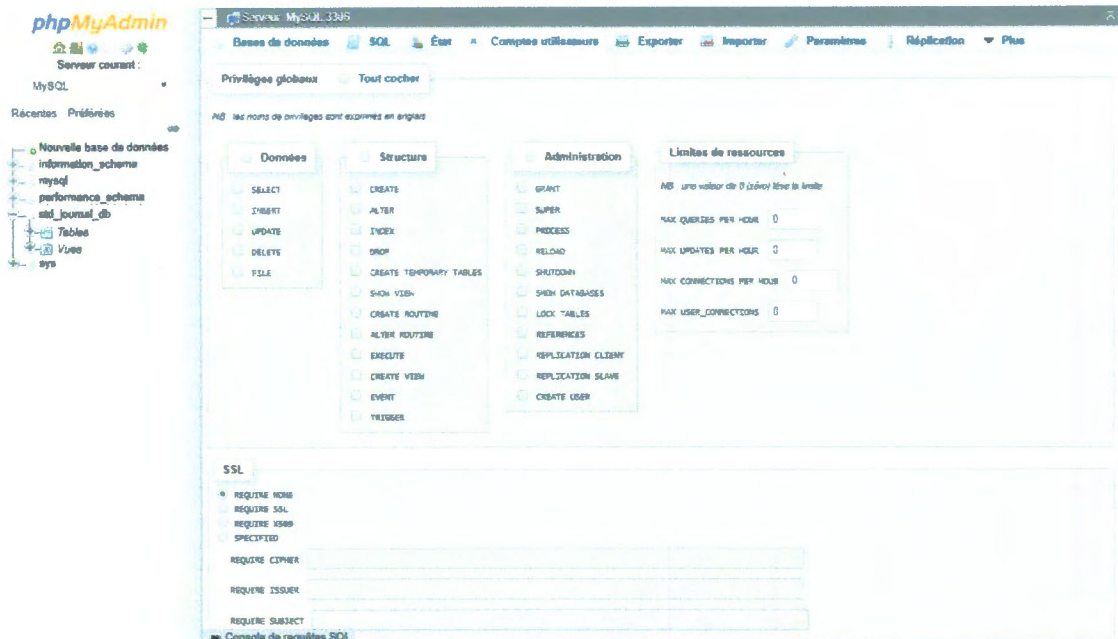


Figure 4-5 : Les privileges d'un compte utilisateur

Validez en cliquant sur "Exécuter" (La partie SSL est facultative).

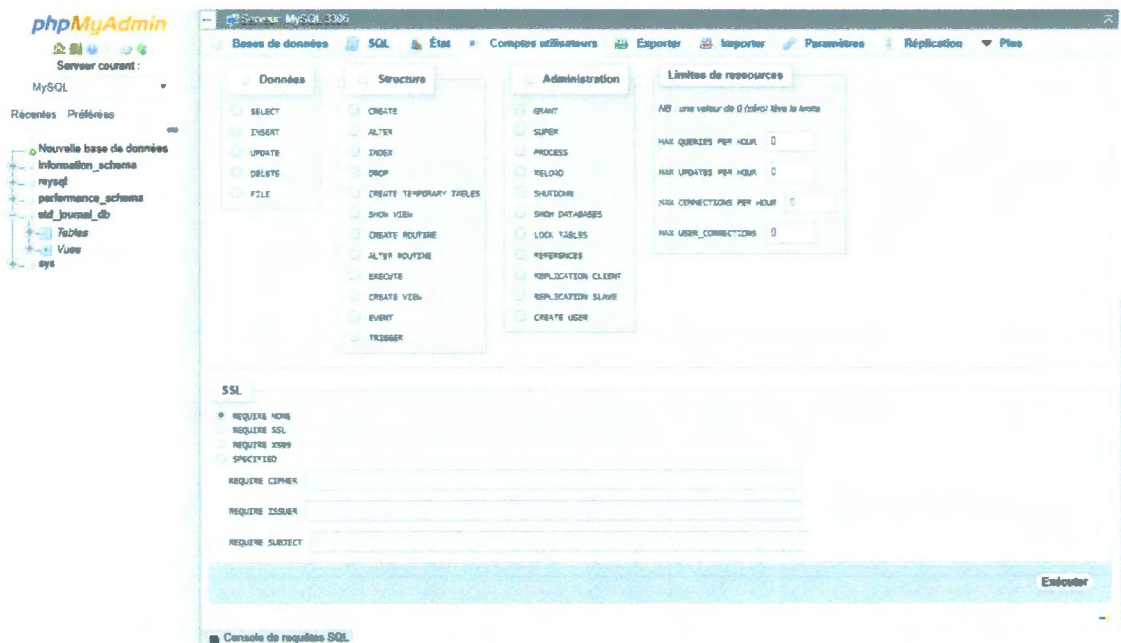


Figure 4-6 : Sauvegarde des données du compte utilisateur

Etape 3 : Ajouter et retirer des privilèges aux utilisateurs

Sélectionnez la table "user" dans la base de données "mysql" et assurez-vous qu'aucun privilège global n'est attribué à nos trois utilisateurs.

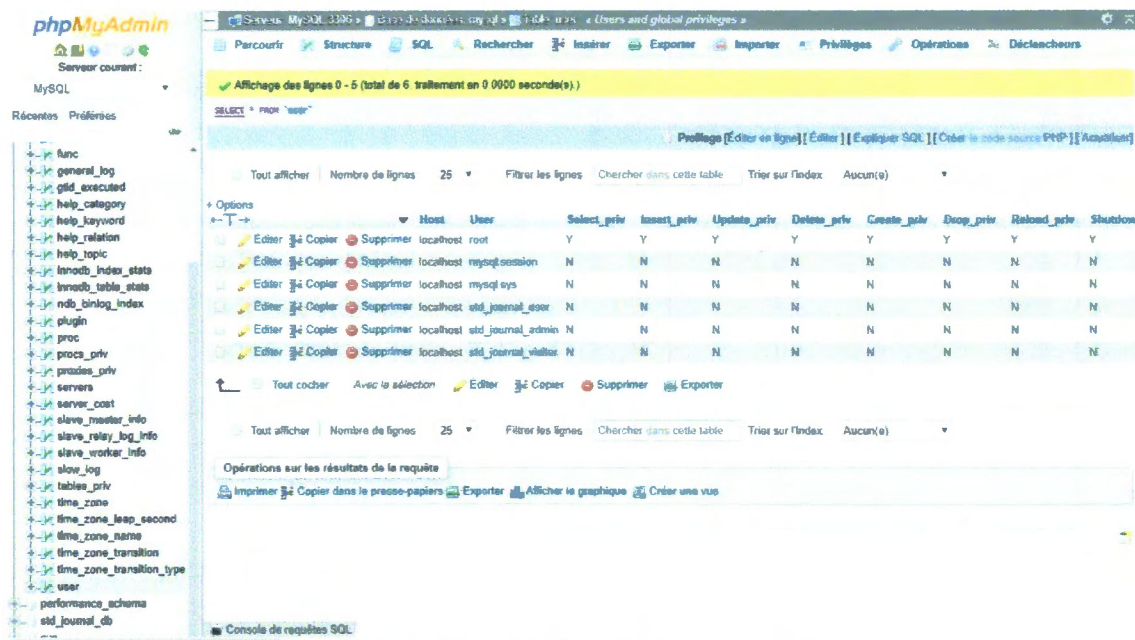


Figure 4-7 : La table utilisateur

Puis sélectionnez la table "db" toujours dans la base de données "mysql" afin d'attribuer des privilèges sur la base de données sélectionnée à chacun des utilisateurs comme suit :

Utilisateur : std_journal_user (Select_priv + Insert_priv + Update_priv + Delete_priv + Show_vieux_priv)

Attribué aux rédacteurs du journal qui ne doivent que créer du contenu et l'ajouter au site Web.



Figure 4-8 : Les pouvoirs d'un compte utilisateur

Utilisateur std_journal_admin : Tous les privilèges sur la base de données std_journal_db

Attribué à l'administrateur du site qui doit avoir le plein pouvoir sur la base de données du site Web.

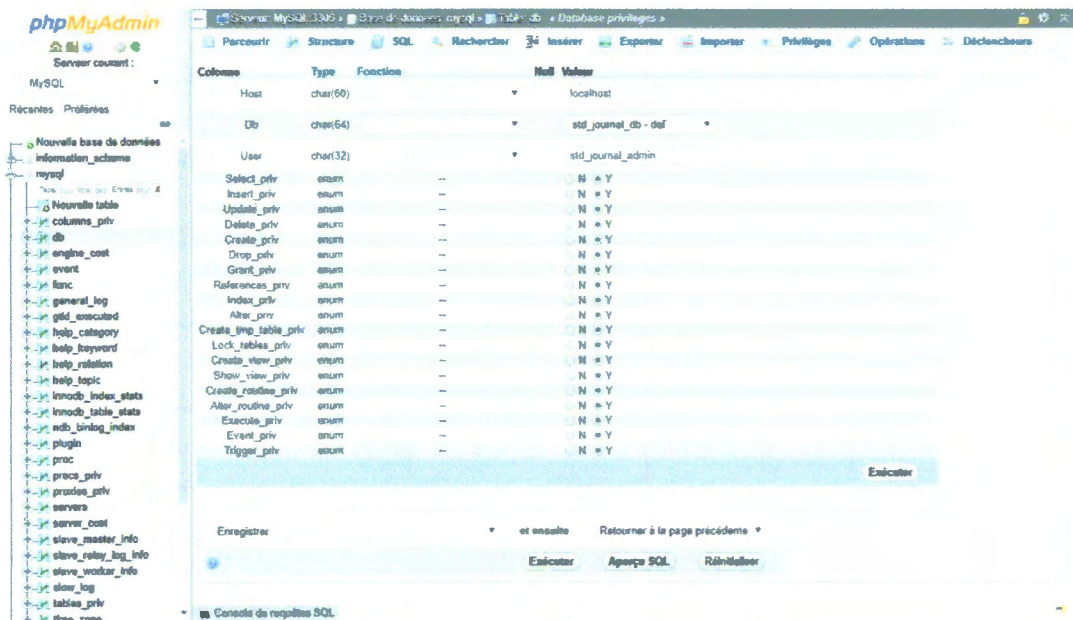


Figure 4-9 : Les pouvoirs d'un compte administrateur

Utilisateur std_journal_visitor : Update_priv + Show_view_priv

Attribué aux visiteurs du site Web, qui n'ont droit qu'à consulter les vues protégées les vraies tables de la base de données ou de modifier le nombre de vues au niveau des tables concernées

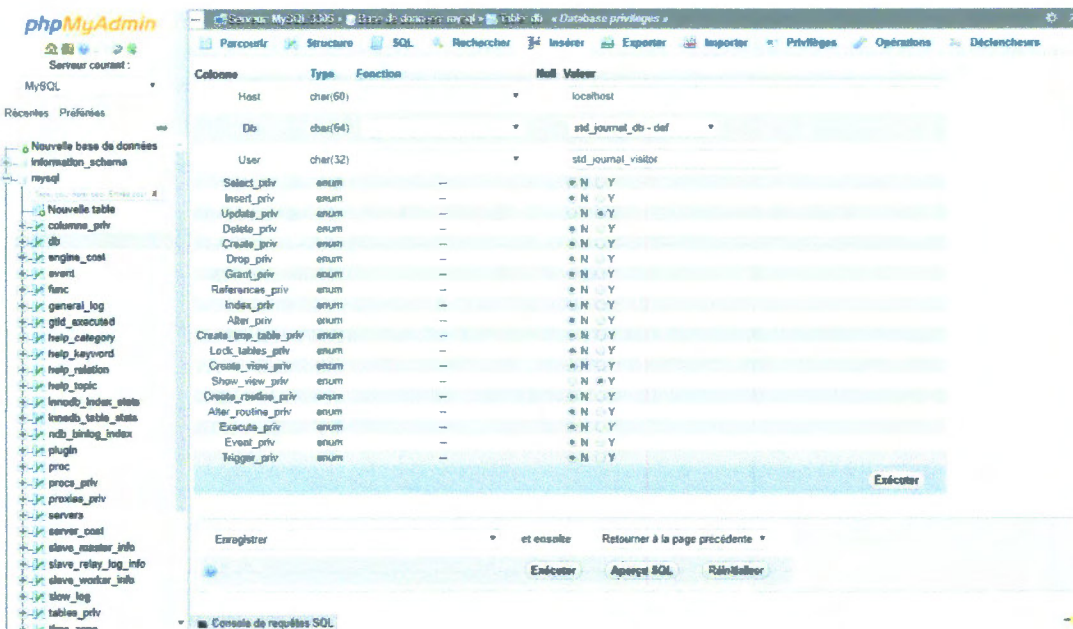


Figure 4-10 : Les pouvoir d'un compte visiteur

4.5.2. Création des vues

Dans le but d'augmenter la sécurité, nous allons faire usage des vues afin de conduire toute

tentative de piratage dans l'erreur. Le pirate ne saura jamais récupérer les vrais noms de l'ensemble des tables et des colonnes l'idée est de faire en sorte que toute partie exposée au grand public ne traite avec la base de données que via le compte d'accès "std_journal_visitor" et de ne traiter qu'avec des vues.

L'avantage est que toute opération de modification ou suppression est interdite ainsi que les noms des colonnes et des tables seront changées. De cette façon toute tentative de suppression va aboutir à un échec car elle ne touchera jamais les tables de la base de données.[8]

a) Vue "rubrique"

```
CREATE VIEW `rubrique` AS
SELECT
  `id_rubric_` AS `id_rubrique`,
  `label_rubric_` AS `libelle_rubrique`,
  `add_rubric_` AS `ajout_rubrique`,
  `state_rubric_` AS `etat_rubrique`
FROM `rubric_`
```

b) Vue "article"

```
CREATE VIEW `article` AS
SELECT
  `id_article_` AS `id_article`,
  `title_article_` AS `titre_article`,
  `text_article_` AS `texte_article`,
  `picture_article_` AS `photo_article`,
  `redaction_article_` AS `redaction_article`,
  `publication_article_` AS `publication_article`,
  `state_article_` AS `etat_article`,
  `rubric_article_` AS `rubrique_article`,
  `writer_acrticle_` AS `redacteur_article`
FROM `article_`
```

c) Vue "compte"

```
CREATE VIEW `compte` AS
SELECT
  `id_account_` AS `id_compte`,
  `nickname_account_` AS `pseudo_compte`,
  `login_account_` AS `login_compte`,
  `pass_account_` AS `pass_compte`,
  `hash_account_` AS `hash_compte`,
  `connection_account_` AS `connexion_compte`,
  `add_account_` AS `ajout_compte`,
  `type_account_` AS `type_compte`,
  `state_account_` AS `etat_compte`
FROM `account_`
```

d) Vue "profil"

```
CREATE VIEW `profil` AS
```

```

SELECT
  `id_profile_` AS `id_profil`,
  `first_name_profile_` AS `nom_profil`,
  `last_name_profile_` AS `prenom_profil`,
  `phone_profile_` AS `telephone_profil`,
  `mail_profile_` AS `email_profil`,
  `state_profile_` AS `etat_profil`,
  `account_profile_` AS `compte_profil`
FROM `profile_`

```

e) Vue "all_data_art_"

```

CREATE VIEW `all_data_art_` AS

SELECT r.*, a.*, c.`pseudo_compte`

FROM `rubrique` r, `article` a, `compte` c

WHERE
  r.`id_rubrique` = a.`rubrique_article` AND
  a.`redacteur_article` = c.`id_compte`

```

f) Vue "all_data_acc_"

```

CREATE VIEW `all_data_acc_` AS
SELECT c.*, p.* FROM `compte` c, `profil` p
WHERE p.`compte_profil` = c.`id_compte`

```

g) Vue "all_data_ope_"

```

CREATE VIEW `all_data_ope_` AS
SELECT o.`*`, c.`pseudo_compte` FROM `operation` o, `compte` c
WHERE o.`compte_operation` = c.`id_compte`

```

4.6. Interfaces du site Web

Afin de bien présenter l'interface de notre site Web, nous proposons le déploiement du jeu d'essai suivant, généré grâce au site Web www.generatedata.com, dans le but de remplir la base de données comme suit :

- Au niveau de la table « account_ » : Insertion automatique de vingt (20) comptes utilisateurs.
- Au niveau de la table « rubric_ » : Insertion automatique de dix (10) rubriques d'articles.
- Au niveau de la table « article_ » : Insertion automatique de cent (100) articles.

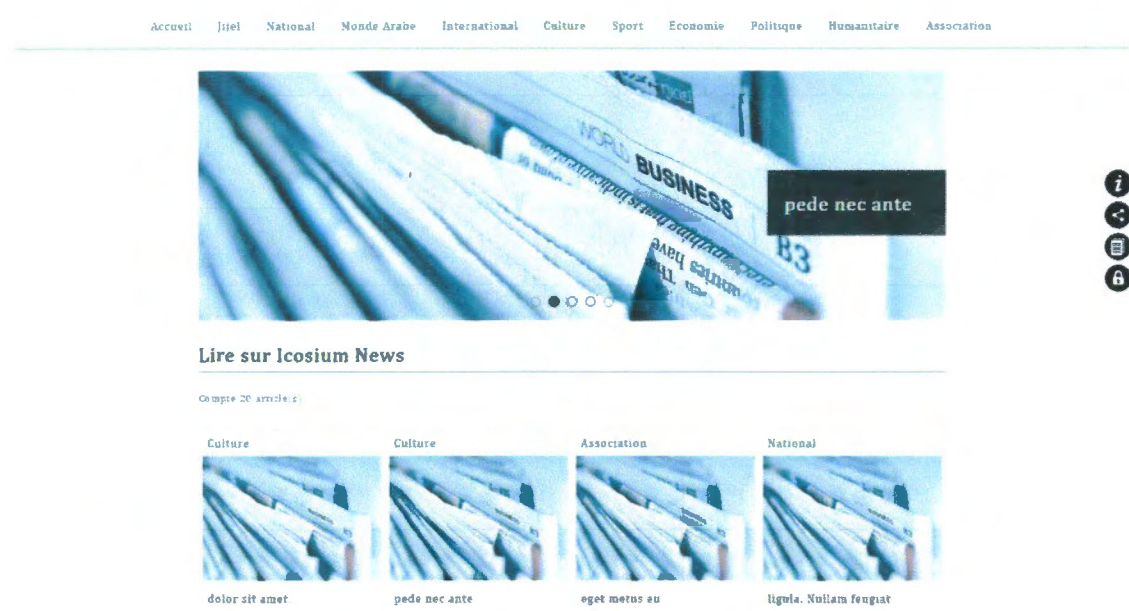
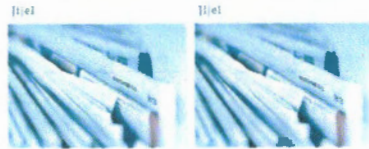


Figure 4-11 : Accueil du site Web « Icosium News »

Jijel

Compre 2 article(s)



vel, faucibus id.

Imperdiet nec leo.

Publié le 2019-06-29

Publié le 2019-05-21



Figure 4-12 : Liste des articles par rubrique



vel, faucibus id.

Écrit par : aute. Vivamus | Rubrique : Jijel | Publié le : 2019-06-29

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nullam suscipit posuere risus, at suscipit arcu aliquam non. Mauris est turpis, auctor vel tincidunt ut, vulputate id purus. Proin ac ultrices ipsum. Suspendisse potenti. Vestibulum vehicula velit felis, sed feugiat nulla vehicula sed. Quisque hendrerit ultrices felis, non accumsan neque sagittis id. Cras vulputate nisi a congue efficitur. Ut vel finibus risus. Vivamus sed metus maximus, congue est ut, interdum magna. Morbi risus turpis, vestibulum nec consectetur sit amet, iaculis at neque. Nunc interdum porta sapien ac dapibus. Phasellus id scelerisque nunc. Phasellus sodales, lectus quis ornare bibendum, urna mauris dignissim odio, eget rutrum dolor lectus et massa. Donec pharetra, felis a venenatis tristique, augue orci fermentum ante, eu gravida tellus est quis orci.

Aliquam ligula ligula, luctus ut commodo id, vehicula eget felis. Nulla ut magna cursus, hendrerit risus non, facilisis erat. In viverra, enim nec tempus pretium, turpis ex accumsan justo, et laoreet duis nisi sed nisi. Curabitur mauris nisi, iaculis in eros ut, placerat aliquet velit. Nulla auctor eu lectus non efficitur. Class aptent taciti sociosqu ad



Figure 4-13 : Lecture d'un article

4.7. Conclusion

Dans ce chapitre nous avons fait, le tour des choix techniques qui ont été fait afin de spécifier les différents langages et technologies à utiliser, une vue sur l'environnement logiciel d'où nous avons spécifié les outils et les utilitaires à utiliser, la spécification de l'environnement matériel qui fait la description de l'ensemble des PC utilisés.

La précision d'un jeu d'essai de taille imposante était nécessaire. Généré automatiquement grâce au site Web www.generatedata.com, le nouveau jeu d'essai permettrait à l'interface de fonctionner en plein régime, la présentation de quelques interface de la partie grand publique « lecteur » pour montrer le fonctionnement du site Web côté visiteur.

Conclusion et perspectives

Ce travail a été réalisé dans le cadre d'un projet de fin d'études et qui a pour objectif le développement d'un site Web sécurisé pour gérer un journal électronique.

Dans un premier temps, les différentes notions et techniques de sécurité ont été exposées en détails, ensuite une étape d'analyse et de conception a été entamée afin de bien spécifier l'architecture de notre système afin de la modéliser grâce à des diagrammes UML (diagramme de cas d'utilisation, diagramme d'activité et diagramme de classe)

L'aboutissement de l'analyse et la conception été le modèle physique de données qui a été dérivé directement du diagramme de classe et qui a donné naissance au premier script de la base de données grâce à l'option de la génération automatique de Toad Data Modeler.

L'implémentation ainsi que tous les tests nécessaires à la réalisation du journal électronique ont été possible d'où la dernière version de la plate-forme WAMPServer qui a été utilisée avec Visual Studio Code.

Cependant des perspectives d'améliorations de la sécurité du site Web restent envisageables pour être enrichie par des fonctionnalités avancées tel que :

- Le développement de la traçabilité des opérations effectuées côté serveur applicatif, ce qui ferait que n'importe quelle tentative de piratage peut être pistée afin de dévoiler la faille qu'elle exploite.
- La génération de fichiers Log qui offrent une vue chronologique des opérations exécutés par les utilisateurs.

A la fin de ce travail, l'espoir est là pour qu'il soit utilisé et enrichi dans le futur par d'autres fonctionnalités telles que la gestion de fichiers multimédias, la gestion de la publicité, et pourquoi pas la gestion d'enquêtes qui peuvent contenir plusieurs photos et vidéos, etc.

Bibliographie

- [1] Damien Seguy et Philippe Gamache, *Sécurité PHP 5 et MySQL*, EYROLLES, 2007.
- [2] Georges Gardarin, *Bases de données*, EYROLLES, 2003.
- [3] Joseph Gabay et David Gabay, *UML2 analyse et conception*, DUNOD, 2008.
- [4] Pascal Roque, *UML2 par la pratique étude de cas et exercices corrigés*, EYROLLES, 2006.
- [5] Mathieu NEBRA, *Concevez votre site web avec PHP et MySQL*, OpenClassrooms, 2019, <https://openclassrooms.com/fr/courses/918836-concevez-votre-site-web-avec-php-et-mysql>
- [6] Julien MOULIN, *PHP : La sécurité des applications*, Video2Brain, 2014, https://fr.linkedin.com/learning/php-la-securite-des-applications/bienvenue-dans-php-la-securite-des-applications?autoplay=true&trk=course_preview&upsellOrderOrigin=default_guess_t_learning
- [7] Support de cours et manuel en PHP, <https://www.php.net/>, Consulté le 01/08/2019.
- [8] Support de cours en SQL, <https://sql.sh/>, Consulté le 01/08/2019.
- [9] Wikipédia l'encyclopédie libre, https://fr.wikipedia.org/wiki/Wikipédia:Accueil_principal, Consulté le 01/07/2019.
- [10] Encyclopédie de Kaspersky, <https://encyclopedia.kaspersky.fr/knowledge/a-brief-history-of-hacking/>, Consulté le 17/05/2019.

