Université Abou Bekr Belkaid
Tlemcen Algérie
جامعة أبي بكر بلقايد
تلمسان الجزائر

République Algérienne Démocratique et Populaire
Université Abou Bakr Belkaid– Tlemcen
Faculté des Sciences
Département d'Informatique

Mémoire de fin d'études

pour l'obtention du diplôme de Master en Informatique

Option: *Réseaux et système distribués (R.S.D)*

Thème

Conception et réalisation d'une application web selon le design pattern MVC en intégrant le chiffrement RSA

Réalisé par :

- **Mlle SELKA Hanane**

Présenté le 25 Juin 2015 devant le jury composé de MM.

- *LEHSAINI Mohamed* *(Président)*
- *BENMAMMAR Badr* *(Encadreur)*
- *BENZIANE Yaghmoracen* *(Examinateur)*
- *MANA Mohamed* *(Examinateur)*

Année universitaire : 2014-2015

Remerciements

À Dieu le tout puissant de m’avoir tracé le bon chemin.

À mes parents pour m’avoir encouragée et permise de bien évoluer tout au long de mon cursus universitaire.

À Monsieur BEMAMMAR B. pour l’honneur qu’il me fait en m’encadrant, ses précieux conseils, ses critiques constructives et pour tout le temps qu’il m’a accordé.

Que Monsieur LEHSAINI M. veuille croire à ma profonde reconnaissance pour avoir accepté de présider ce jury.

Que les honorables membres du jury, Monsieur BENZIANE Y. et Monsieur MANA M. veuillent croire en mes remerciements anticipés pour avoir bien voulu accepter d’enrichir et de faire évoluer ce travail.

Je remercie aussi Monsieur le Chef de Département Informatique BENAMAR A. et tous mes enseignants pour la richesse et la qualité de leurs enseignements, pour le suivi et pour les efforts fournis qui m’ont permis d’apprendre et d’acquérir tout ce savoir et ces connaissances.

Dédicaces

À mes très chers parents. J'espère qu'ils trouveront dans ce travail le témoignage de ma reconnaissance éternelle.

À mes sœurs et frère Amel, Ikram et Charaf-Eddine, qui apportent beaucoup de joie à ma vie chaque jour.

À ma petite sœur Esma ; brillante et excellente dans tout ce qu'elle fait. Je lui souhaite beaucoup de succès dans ses études.

À mes oncles, mes tantes, mes cousins et à toutes mes cousines avec qui je partage tant de bons moments.

À tous mes ami(e)s en particulier Fayçal Baghli, Amina Benosman et Amine Benyellesse qui m'ont toujours soutenu et encouragé.

À mon amie Asma Amraoui qui m'a tant encouragé et offert de précieux conseils et critiques.

À mon ami Ahmed Berbar qui a apporté une touche artistique à mon travail.

À mes chères amies Fouzia Baghli et Imène Meziane toujours présentes pour moi et me poussent à aller de l'avant. Je leurs souhaite beaucoup de bonheur et sérénité.

Résumé:

Actuellement les applications web sont largement répandues et elles sont disponibles partout et pour tout le monde. Ces applications sont particulièrement exposées aux attaques par des personnes ou des logiciels malveillants. La sécurité des applications web est devenue primordiale. Nous proposons dans ce travail la conception et la réalisation d'une application web qui met en évidence l'importance de garantir une sécurité idéale et de protéger la confidentialité, l'intégrité et la disponibilité de l'information. « Dispositifs en ligne » a été conçue en suivant le modèle MVC et réalisée avec PHP5 en intégrant plusieurs mécanismes de sécurité : l'authentification pour protéger le contenu, la gestion de rôles pour maitriser l'accès à certaines fonctionnalités et le cryptage asymétrique des données sensibles avec RSA.

Mots clés:

Sécurité des applications web, MVC, PHP5, cryptage asymétrique RSA, authentification, gestion de rôles.

Abstract:

Nowadays web applications are widespread and are available everywhere and for everyone. These applications are particularly exposed to attacks by people or malware. The web application security has become increasingly important. We propose in this work the design and realization of a web application that highlights the importance of ensuring an ideal security and to protect the confidentiality, integrity and availability of information. "Dispositifs en ligne" was designed following the MVC pattern and performed with PHP5 integrating several security mechanisms: authentication to protect the content, roles management master for access to certain features and asymmetric encryption of sensitive data with RSA.

Keywords:

Security of web applications, MVC, PHP5, asymmetric encryption RSA, authentication, roles management.

ملخص:

حاليا تطبيقات الويب منتشرة على نطاق واسع، وتتوفر في كل مكان وللجميع. هذه التطبيقات معرضة بشكل خاص لهجمات من قبل أفراد أو برامج ضارة. لقد أصبح أمن تطبيقات الويب أمر في غاية الأهمية. نقترح في هذا العمل تصميم وتنفيذ تطبيق ويب من أجل ابراز أهمية ضمان الأمن المثالي وحماية سرية وسلامة وتوافر المعلومات.
"Dispositifs en ligne" تم تصميمه على غرار النمط MVC وأنجز بPHP5 مع دمج عدة آليات أمن: المصادقة من أجل حماية المحتوى، تسيير الأدوار للسيطرة على الوصول الى بعض الوظائف وكذلك التشفير الغير المتناظر للبيانات الحساسة بRSA .

الكلمات المفتاحية:

أمن تطبيقات الويب، MVC، PHP5، التشفير الغير المتناظر RSA، المصادقة، تسيير الأدوار.

Table des matières

Introduction générale

De nos jours les applications web sont omniprésentes dans notre vie quotidienne. Grands et petits accèdent à tout moment et où qu'ils soient à plusieurs genres d'applications disponibles sur le plus grand réseau mondiale « Internet ».

Toutes entreprises ou administrations se doit d'avoir une présence sur le web que ce soit via un blog, un site de ventes en ligne ou une offre de services en ligne.

Ces applications deviennent de plus en plus légères et fonctionnent tout aussi bien sur un ordinateur que sur un smartphone ou une tablette. Elles sont accessibles partout et pour tout le monde. Le problème avec ces applications c'est qu'elles sont particulièrement exposées aux attaques par des personnes ou des logiciels malveillants.

La sécurité des applications web est devenue donc une priorité, plus importante même que les fonctionnalités. Le besoin d'assurer une sécurité idéale et de protéger la confidentialité, l'intégrité et la disponibilité de l'information, a nécessité la création de plusieurs organismes. Ce genre d'organismes comme l'OWASP ou L'ANSSI veillent à sensibiliser les entreprises et les personnes sur les risques liés à la sécurité des applications Web. Ces organismes publient très souvent des guides pour mettre en garde et prévenir les personnes des failles et attaques les plus connues et proposent toutes une séries de bonnes pratiques pour s'en prémunir.

Afin de schématiser l'importance de la sécurité dans les applications web, dans un premier chapitre seront présentées les failles et attaques les plus connues dans les applications web mais aussi quelques bonne pratiques a adopter pour garantir un certain niveau d'intégrité et de fiabilité d'une application web, parmi celles-ci le cryptage des données avec RSA. Comme le choix des outils et programmes à utiliser est primordiale pour assurer la sécurité ; dans un second chapitre seront présentés quelques outils et méthodes utilisées pour la conception et la réalisation d'une application nommée « Dispositifs en ligne » qui sera présentée dans un troisième et dernier chapitre.

CHAPITRE I La sécurité dans les applications web

.I Introduction

Le web est devenu depuis plusieurs années un lieu d'échange et de partage d'information mais il est devenu surtout un outil de vente et du e-commerce. Pour proposer la vente d'articles en ligne ou d'un quelconque service, il faut être en mesure de garantir à ces clients une certaine sécurité, fiabilité et intégrité de l'application.

Nous verrons dans ce premier chapitre consacré à la sécurité dans le cadre des applications web, quelles sont les failles et les attaques les plus connues et comment se prémunir contre la plupart des risques de sécurité dans le développement des applications Web.

.II Failles de sécurité

Le besoin d'assurer une sécurité idéale et de protéger la confidentialité, l'intégrité et la disponibilité de l'information, a nécessité la création de plusieurs organismes. Ils ont pour objectif de lutter et de prévenir les risques liés à la sécurité des informations sur le web.

En France :

- L'ANSSI est une agence rattachée au Secrétaire général de la défense et de la sécurité nationale. Elle met à disposition des guides sur la gestion des menaces informatiques et des articles sur les recommandations et bonnes pratiques pour la sécurité des systèmes informatiques. [1]

- Le CLUSIF (Club de la Sécurité de l'Information Français) est une association d'organisations privées et publiques dont le but est de sensibiliser les entreprises et collectivités françaises à la sécurité de l'information. [2]

Aux Etats-Unis :

- Le Web Application Security Consortium (WASC) est une association constituée d'experts internationaux, d'industriels et d'organisations du monde Open Source qui publie des recueils de bonnes pratiques de sécurité pour le Web ainsi que le rapport « WASC Threat Classification » qui décrit et classe les menaces de sécurité sur les applications Web.

- L'Open Web Application Security Project (OWASP) est une association de bénévoles dont l'objectif est de promouvoir la sécurité logicielle et de sensibiliser

les organisations et les personnes sur les risques liés à la sécurité des applications Web. Tous les 3 ans, elle publie le classement des 10 failles de sécurité les plus dangereuses dans le document « OWASP Top 10 ». [1]

.II.1 Types d'attaques

Le WASC établie dans son rapport « WASC Threat Classification » une liste exhaustive des menaces qui pèsent sur la sécurité des applications Web. Elles sont regroupées dans six catégories définies comme suit : [1]

1. Authentification

Cette catégorie regroupe les attaques de sites Web dont la cible est le système de validation de l'identité d'un utilisateur, d'un service ou d'une application. [1]

2. Autorisation

Cette catégorie couvre l'ensemble des attaques de sites Web dont la cible est le système de vérification des droits d'un utilisateur, d'un service ou d'une application pour effectuer une action dans l'application. [1]

3. Attaques côté client

Cette catégorie rassemble les attaques visant l'utilisateur pendant qu'il utilise l'application. [1]

4. Exécution de commandes

Cette catégorie englobe toutes les attaques qui permettent d'exécuter des commandes sur un des composants de l'architecture du site Web. [1]

5. Révélation d'informations

Cette catégorie définit l'ensemble des attaques permettant de découvrir des informations ou des fonctionnalités cachées. [1]

6. Attaques logiques

Cette catégorie caractérise les attaques qui utilisent les processus applicatifs (système de changement de mot de passe, système de création de compte, …) à des fins hostiles. [1]

.II.2 Risques de sécurité

Contrairement au WASC qui décrit toutes les attaques possibles sur une application Web, l'OWASP identifie quelques-uns des risques les plus critiques rencontrés par les entreprises. L'OWASP établit en 2013 le classement suivant :

1. Injection

Une faille d'injection, telle l'injection SQL, OS et LDAP, se produit quand une donnée non fiable est envoyée à un interpréteur en tant qu'élément d'une commande ou d'une requête. Les données hostiles de l'attaquant peuvent duper l'interpréteur afin de l'amener à exécuter des commandes fortuites ou accéder à des données non autorisées. [3]

2. Violation de Gestion d'Authentification et de Session

Les fonctions applicatives relatives à l'authentification et la gestion de session ne sont souvent pas mises en œuvre correctement, permettant aux attaquants de compromettre les mots de passe, clés, jetons de session, ou d'exploiter d'autres failles d'implémentation pour s'approprier les identités d'autres utilisateurs. [3]

3. Cross-Site Scripting (XSS)

Les failles XSS se produisent chaque fois qu'une application accepte des données non fiables et les envoie à un browser web sans validation appropriée. XSS permet à des attaquants d'exécuter du script dans le navigateur de la victime afin de détourner des sessions utilisateur, défigurer des sites web, ou rediriger l'utilisateur vers des sites malveillants. [3]

4. Références directes non sécurisées à un objet

Une référence directe à un objet se produit quand un développeur expose une référence à un objet d'exécution interne, tel un fichier, un dossier, un enregistrement de base de données ou une clé de base de données. Sans un contrôle d'accès ou autre protection, les attaquants peuvent manipuler ces références pour accéder à des données non autorisées. [3]

5. Mauvaise configuration Sécurité

Une bonne sécurité nécessite de disposer d'une configuration sécurisée définie et déployée pour l'application, contextes, serveur d'application, serveur web, serveur de base de données et la plate-forme. Tous ces paramètres doivent être définis, mis en œuvre et maintenus, car beaucoup ne sont pas livrés sécurisés par défaut. Cela implique de tenir tous les logiciels à jour. [3]

6. Exposition de données sensibles

Beaucoup d'applications web ne protègent pas correctement les données sensibles telles que les cartes de crédit, identifiants d'impôt et informations d'authentification. Les pirates

peuvent voler ou modifier ces données faiblement protégées pour effectuer un vol d'identité, de la fraude à la carte de crédit ou autres crimes. Les données sensibles méritent une protection supplémentaire tel un chiffrement statique ou en transit, ainsi que des précautions particulières lors de l'échange avec le navigateur. [3]

7. Manque de contrôle d'accès au niveau fonctionnel

Pratiquement toutes les applications web vérifient les droits d'accès au niveau fonctionnel avant de rendre cette fonctionnalité visible dans l'interface utilisateur. Cependant, les applications doivent effectuer les mêmes vérifications de contrôle d'accès sur le serveur lors de l'accès à chaque fonction. Si les demandes ne sont pas vérifiées, les attaquants seront en mesure de forger des demandes afin d'accéder à une fonctionnalité non autorisée. [3]

8. Falsification de requête intersites (CSRF)

Une attaque CSRF (Cross Site Request Forgery) force le navigateur d'une victime authentifiée à envoyer une requête HTTP forgée, comprenant le cookie de session de la victime ainsi que toute autre information automatiquement inclue, à une application web vulnérable. Ceci permet à l'attaquant de forcer le navigateur de la victime à générer des requêtes dont l'application vulnérable pense qu'elles émanent légitimement de la victime. [3]

9. Utilisation de composants avec des vulnérabilités connues

Les composants vulnérables, tels que bibliothèques, contextes et autres modules logiciels fonctionnent presque toujours avec des privilèges maximum. Ainsi, si exploités, ils peuvent causer des pertes de données sérieuses ou une prise de contrôle du serveur. Les applications utilisant ces composants vulnérables peuvent compromettre leurs défenses et permettre une série d'attaques et d'impacts potentiels. [3]

10. Redirections et renvois non validés

Les applications web réorientent et redirigent fréquemment les utilisateurs vers d'autres pages et sites internet, et utilisent des données non fiables pour déterminer les pages de destination. Sans validation appropriée, les attaquants peuvent réorienter les victimes vers des sites de phishing ou de malware, ou utiliser les renvois pour accéder à des pages non autorisées. [3]

.II.3 Correspondances entre les définitions de l'OWASP et du WASC

Le tableau ci-dessous fait le lien entre les catégories et les attaques définies par le WASC dans le rapport de 2010 et les menaces identifiées par l'OWASP dans le classement de 2010. [1]

Risques identifiés par l'OWASP en 2010	Attaques identifiées par le WASC en 2010	Catégories
Injection	SQL Injection (WASC-19) XML Injection (WASC-23) Null Byte Injection (WASC-28) LDAP Injection (WASC-29) Mail Command Injection (WASC-30) OS Commanding (WASC-31) XPath Injection (WASC-39) XQuery Injection (WASC-46)	Exécution de commandes
Cross-Site Scripting	Cross-Site Scripting (WASC-08)	Attaques côté client
Violation de Gestion d'Authentification et de Session	Insufficient Authentication (WASC-01) Brute Force (WASC-11) Credential/Session Prediction (WASC-18) Session Fixation (WASC-37) Insufficient Session Expiration (WASC-47)	Autorisation, Authentification
Référence directe à un objet non sécurisée	Insufficient Authentication (WASC-01) Insufficient Authorization (WASC-02) Path Traversal (WASC-33) Predictable Location (WASC-34)	Autorisation, Authentification, Révélation d'informations
Falsification de requête inter-site	Cross-Site Request Forgery (WASC-09)	Attaques logiques
Gestion de configuration non sécurisée	Server Misconfiguration (WASC-14) Application Misconfiguration (WASC-15)	Révélation d'informations
Stockage de données non sécurisé	Insufficient Data Protection (WASC-50)	Révélation d'informations
Défaillance de restriction d'accès à une URL	Insufficient Authorization (WASC-02) Denial of Service (WASC-10) Brute Force (WASC-11) Insufficient Anti-automation (WASC-21) Predictable Location (WASC-34)	Autorisation, Authentification, Révélation d'informations
Communications non sécurisées	Insufficient Transport Layer Protection (WASC-04)	Révélation d'informations

Redirection et renvoi non validés	URL Redirector Abuse (WASC-38)	Attaques logiques

Tableau I-1:Tableau de correspondance entre les définitions de l'OWASP et du WASC *[1]*

.III Bonnes pratiques et règles de développement

Il est possible de se protéger de la plupart des attaques expliquées précédemment en suivant quelques règles de développement.

.III.1 Toutes les données doivent être vérifiées

Les valeurs saisies dans un formulaire doivent être validées au niveau du navigateur avec du code JavaScript, car le client n'est pas une source fiable. Les valeurs doivent également être contrôlées au niveau du serveur au moment de la récupération des paramètres, tout comme les paramètres de requête. Il n'est pas certain que ce soit l'utilisateur de l'application qui envoie la requête http. Ainsi pour chaque valeur :

- Vérifier que le type correspond à celui attendu.
- Pour plus de sécurité, caster les données avant de les mettre dans des variables.
- Coder les caractères spéciaux avec le code HTML correspondant.
- Vérifier la présence de tous les arguments attendus.
- Pour les nombres, contraindre la valeur entre deux bornes.
- Pour les listes, vérifier que la valeur appartient à la liste des valeurs autorisées (select, radio, checkbox, …).
- Contraindre la longueur de la valeur saisie avec une taille minimale et une taille maximale.
- Vérifier la valeur avec une expression régulière.
- N'accepter que les lettres de l'alphabet et/ou les chiffres par défaut, tous les autres caractères devant être refusés. Dans le cas où d'autres caractères doivent être autorisés, ils doivent être limités à une liste prédéfinie ou être remplacés par les codes HTML.
- Vérifier si la valeur nulle doit être acceptée.
- Définir le jeu de caractère de la donnée.

Même les données envoyées vers l'utilisateur doivent être vérifiées, avec au minimum les actions suivantes :

- Coder les caractères spéciaux avec le code HTML correspondant.

- Définir le jeu de caractère de la page. [1]

.III.2 Privilégier l'utilisation des requêtes POST

Cela permet de ne pas prendre en compte des paramètres frauduleux dans les adresses. [1]

.III.3 Utiliser les requêtes paramétrées

Les requêtes SQL ne doivent pas être construites dynamiquement. Ainsi, les requêtes SQL ne peuvent pas être parasitées comme c'est le cas dans le cas de l'injection SQL. [1]

.III.4 Ne pas réinventer la roue

Dans la mesure du possible, il est préférable d'utiliser des outils existants plutôt que de développer des fonctionnalités souvent présentes dans les applications, comme les systèmes d'authentification ou de réinitialisation de mot de passe. [1]

.III.5 Sécuriser l'accès aux données

Toutes les pages d'une application Web doivent s'assurer que l'utilisateur s'est authentifié préalablement. Pour les fonctionnalités manipulant des données sensibles, l'application doit demander à l'utilisateur de s'authentifier à nouveau avant de les afficher. [1]

Cela permet de s'assurer qu'il n'y a pas eu usurpation d'identité. Les messages d'erreur renvoyés doivent être générique pour ne pas aiguillé les attaquants potentiels.

Les données sensibles doivent être chiffrées dans les bases de données.

La protection des mots de passe, des identifiants de session et des cookies permet de se prémunir contre l'usurpation d'identité. Pour cela, le mot de passe doit avoir au moins huit caractères, voire même dix pour les accès aux données sensibles. Il doit également contenir au moins trois types de caractères, tels que des chiffres, des lettres minuscules, des lettres majuscules ou des caractères spéciaux. Il doit avoir une période de validité de maximum quatre-vingt-dix jours. Au-delà, il doit être changé. Le compte de l'utilisateur doit être verrouillé, au moins temporairement, si cinq tentatives de connexions consécutives ont échoué. Le mot de passe doit être chiffré ou haché avec un algorithme fort. Seule sa valeur chiffrée sera comparée lors de l'authentification de l'utilisateur. Les identifiants de session doivent avoir une durée de vie limitée au-delà de laquelle il n'est plus utilisable. De même, après une période d'inactivité prédéfinie, l'identifiant de session soit être invalidé. En cas de fermeture du navigateur, l'identifiant de session doit

être supprimé. Ces actions sont équivalentes à un système de déconnexion automatique. Les cookies doivent être protégés en positionnant deux attributs. « Secure » permet d'interdire l'envoie du cookie sur un canal non chiffré. « HTTPonly » permet d'en interdire l'accès à JavaScript. [1]

.IV Protection contre le vol de mot passe par la technique de hashage :

.IV.1 Hash du mot de passe

Pour éviter le vol du mot de passe, il faut limiter, voire proscrire, sa transmission et son stockage en clair. À la place, on va manipuler un hash, beaucoup plus sécurisé car il est (théoriquement) impossible à inverser. En pratique, c'est "réversible". Le hacker utilisera par exemple des dictionnaires pour obtenir le mot de passe en clair. [4]

Le hash est un procédé de chiffrement irréversible. Ceci implique qu'on ne peut appliquer la transformation inverse, et qu'on ne peut donc que comparer les hashs entre eux. L'avantage est que personne, y compris le webmaster, n'a accès aux mots de passe des membres.

Il existe deux principaux algorithmes de hash, le MD5 et le SHA1.

Le MD5 est déprécié et NE DOIT PLUS ETRE UTILISE. En effet, une faille dans l'algorithme a été trouvée, permettant de diminuer considérablement le nombre d'opérations nécessaires à un inversement par rapport à une attaque force brute. De plus, et c'est le plus inquiétant, il a été montré qu'il était possible de créer volontairement des collisions avec un document donné. En effet, les hashs MD5 sont constitués de 32 caractères alphanumériques, soit 3632= 6.3349 combinaisons possibles. Ainsi, le nombre de hashs possibles est limité. Or, nous réalisons des hashs de chaîne de caractères (strings, fichiers...) de taille indéterminée et pour la plupart bien plus grandes que 32 caractères. Par exemple, nous pouvons constituer 21556 chaînes de 1000 caractères alphanumériques.

Nous avons donc une infinité de chaîne de caractères pouvant donner un hash précis et il est maintenant possible de déterminer instantanément les autres chaînes de caractères à partir d'une des chaînes et du hash.

Il est donc recommandé d'utiliser des algorithmes plus robustes, comme le SHA1 ou le SHA-256. Mais il faut bien garder en tête qu'un algorithme est reconnu comme fiable uniquement car il n'a pas encore été craqué. Ainsi, le SHA1 est déjà sur la pente descendante, le nombre d'opérations nécessaires à l'inversion ayant été ramené de 2^80 à 2^63. [4]

.IV.2 Technique du Grain De Sel (GDS)

Un grain de sel, ou "salt", en cryptographie, est appliqué durant le processus de hashage pour éliminer la possibilité d'attaques par dictionnaires (hashages enregistrés dans une grande liste et comparés).

En d'autres termes, un grain de sel est une petite donnée additionnelle qui renforce significativement la puissance du hashage pour le rendre beaucoup plus difficile à cracker. Il existe de nombreux services en ligne qui proposent de volumineux dictionnaires de mots de passe avec leur hash. L'utilisation d'un grain de sel rend ces dictionnaires inutiles. [5]

.V Chiffrement des données :

Les algorithmes de hashage comme MD5, SHA1 et SHA256 sont destinés à être rapides et efficaces. Avec les équipements informatiques modernes, il est devenu facile d'attaquer par force brute la sortie de ces algorithmes pour retrouver la chaîne originale. [5]. C’est la raison pour laquelle les experts conseillent l’utilisation d’algorithmes de chiffrement largement éprouvés.

On trouve principalement deux grandes familles de cryptage : le cryptage symétrique (ou dit à clé secrète) et le cryptage asymétrique (dit aussi à clé publique). [6]

.V.1 Le cryptage symétrique

On parle de cryptage symétrique lorsqu'un texte, document, etc. est crypté et décrypté avec la même clé, la clé secrète, ce procédé est à la fois simple et sûr. On trouvera principalement parmi les algorithmes de cryptage symétrique : AES, qui serait utilisé pour protéger des documents secrets aux États-Unis. Principal inconvénient : étant donné que l'on n'a qu'une clé, si vous la donnez à X pour qu'il puisse vous envoyer des messages cryptés avec celle-ci, il pourra aussi bien décrypter tous les documents que vous avez crypté avec cette dernière. La clé est donc connue uniquement par le destinataire et

l'émetteur et il est plus sûr de faire une clé pour un échange entre X et Y, pour éviter qu'avec une clé on puisse tout décrypter. [6]

.V.2 Le cryptage asymétrique

Contrairement au cryptage symétrique, ici avec l'asymétrique, on a 2 clés.

Tout d'abord nous avons la clé publique. Celle-ci, tout le monde peut la posséder, il n'y a aucun risque, vous pouvez la transmettre à n'importe qui. Elle sert à crypter le message. Puis il y a la clé privée que seul le récepteur possède. Elle servira à décrypter le message crypté avec la clé publique. [6]

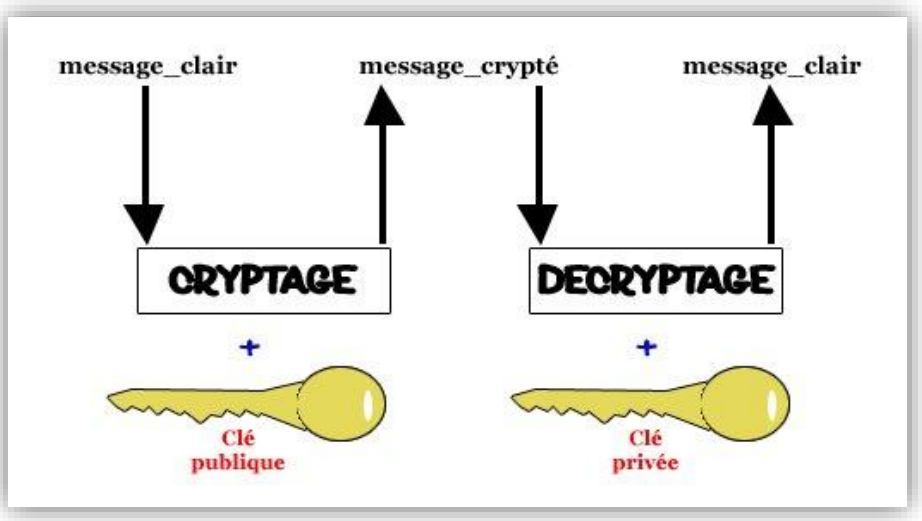

Figure I-1: Cryptage asymétrique *[6]*

Parmi les algorithmes de chiffrement asymétrique, on trouve le RSA (le plus connu), le PGP, le DSA...

.V.3 Le cryptage RSA

Le système cryptographique RSA a été inventé en 1977, et publié en 1978, par trois mathématiciens qui travaillaient alors au MIT (Massachusetts Institute of Technology) : Ronald Rivest, Adi Shamir et Leonard Adleman. Comme vous pouvez le constater, le nom de l'algorithme est simplement tiré des initiales de leurs noms de famille. La grande réussite de leur travail en commun est sans doute due au fait que leurs compétences se complètent.

Rivest est un grand spécialiste de la cryptographie, la discipline qui cherche à protéger les messages d'une lecture inopportune : il est notamment à l'origine de l'algorithme MD5.

Shamir est au contraire un spécialiste de la cryptanalyse, dont le but est de casser les protections mises en place par la cryptographie : il est notamment connu pour avoir cassé en 1982 le cryptosystème de Merkle-Hellman, un algorithme proche de RSA mais reposant sur une base mathématique un peu différente.

Adleman, quant à lui, représentait le grain de folie du groupe : il travaillait à l'époque sur la bio-informatique, en gros, des ordinateurs utilisant de l'ADN pour faire leurs calculs. [7]

.V.3.1 Fonctionnement de l'algorithme RSA

Partie théorique :

Un algorithme de chiffrement est un ensemble d'opérations, généralement mathématiques, à effectuer sur le message que l'on veut protéger, afin de le rendre en théorie incompréhensible par ceux auxquels il n'est pas destiné. Il vient toujours avec un algorithme de déchiffrement, qui permet au récipiendaire du message de retrouver le message d'origine. [7]

Le chiffrement asymétrique est aussi appelé "à clé publique", même si en fait, on manipule une paire de clés : une publique et une privée. [4]

Il est asymétrique car ne fonctionne que dans un seul sens. Par exemple :

- A veut recevoir une information provenant de B.
- A génère une clé publique et une clé privée.
- A envoie la clé publique à B.
- B chiffre son information avec la clé publique.
- B envoie le message chiffré à A.
- A déchiffre en utilisant la clé privée. [4]
    - ❖ Création des clés :
        - Création de la clé publique :

Pour créer notre clé publique il nous faut choisir deux nombres premiers.

Soit **P** et **Q**, ces deux nombres premiers.

P = 53 et **Q = 97**

Ensuite, on va prendre un autre nombre, **N**, tel que : **N = P * Q**

N = 53 * 97 = 5141

On pose[1] **M = (P – 1) * (Q – 1)**

M = (53 - 1) * (97 - 1) = 4992

Pour créer notre clé publique, il ne nous reste plus qu'à choisir un nombre **C**, qui soit premier avec **M**. (Il en existe, là aussi, une infinité.) On prendra **C = 7**.

Au final, notre clé publique est composée de **(N, C)** soit **(N = 5141, C = 7)** comme clé publique. [8]

- Création de la clé privée :

Pour créer notre clé privée, on va calculer un nombre U, qui va nous permettre "d'inverser la fonction de chiffrement" très facilement. C'est ce nombre qui sera important, et secret !

Pour calculer U, on va reprendre les nombres C et M, calculés lors de la création de la clé publique.

Un mathématicien, du nom d'Etienne Bézout, a démontré que deux nombres a et b sont premiers entre eux, si et seulement s'il existe des solutions u et v telles que **a * u + b * v = 1** (u et v étant des nombres entiers). Or, on a dit peu avant que C et M devaient être premiers entre eux. Il existe donc deux nombres entiers u et v qui répondent à l'équation. Nous allons chercher U, tel que C * U + M * V = 1

Pour trouver **U**, il faut utiliser un algorithme spécial. Il faut donc utiliser un petit programme qui permet de trouver la valeur de **U** cherchée.

Le programme est disponible pour Windows, il faut le télécharger et l'installer depuis l'invite de commande. Pour trouver **U** il faut lancer le programme avec comme arguments **C** et **M**.

Nous obtenons alors **U = 4279**.

Voilà, notre clé privée est **(U, N)** soit **(U = 4279 et N = 5141)** comme clé privée. [8]

[1] Ce nombre "M" est appelé indicatrice d'Euler, il correspond au nombre d'entiers naturels (0, 1, 2, 3, etc...) inférieurs ou égaux à N qui lui sont premiers. [8]

La génération de ces clés est fondée sur la difficulté de factorisation des grands nombres. En gros, ça veut dire qu'on peut aisément multiplier 2 énormes nombres entre eux, mais qu'on ne peut pas retrouver si facilement ces 2 nombres en ayant que le résultat de cette multiplication. [4]

En effet, ce n'est pas très sécurisé si P et Q sont de petits nombres premiers. En réalité, P et Q sont des nombres composés d'une bonne centaine de chiffres, ce qui fait qu'il est très difficile de décomposer N pour retrouver P et Q. Même des machines surpuissantes reliées entre elles ne pourraient pas casser N en un temps raisonnable. [8]

Voici un exemple de factorisation :

N = 31074182404900437213507500358885679300373460228427275457201619488232064405180815045563468296717232867824379162728380334154710731085019195485290073377248227835257423864540146917366024776523466090850191954852900733772482278352574238645401469173660247765234660931

Qui est décomposable en :

P = 16347336458092538484431338838650908598417836700330923121811108523893331001045081512121181675115790

Q = 19008712816648221131268515739354139754718967899685154936666385390880271038021044989571912614655710

[8]

- Le chiffrage :

Après avoir créé les clés publique et privée, nous allons voir comment fait-on pour chiffrer un message quelconque (texte /chiffres ...) grâce au système RSA.
B veut envoyer le message « Bonjour ! » à A. Il utilise alors la clé publique de A pour chiffrer le message.

Etape1 : remplacement des caractères par leurs valeurs ASCII :

B⇔66, o⇔111, n⇔110, j⇔106, o⇔111, u⇔117, r⇔114, espace⇔32, !⇔33

Etape2 : premier calcul, la puissance :

Ensuite, on va élever chaque sous-message (nombre) à la puissance C (7, dans notre cas)
$B \Leftrightarrow 66^7$, $o \Leftrightarrow 111^7$, $n \Leftrightarrow 110^7$ ….

Etape3 : deuxième et dernier calcul, le modulo:

On va calculer le modulo du résultat obtenu précédemment par N (le N de la clé publique)

B -> (66^7) modulo (5141) = 386,

o -> (111^7) modulo (5141) = 1858, …

Ainsi avec la clé publique de A (N = 5141, C = 7), le message "Bonjour !" devient "386 1858 2127 2809 1858 1774 737 3675 244"

A partir du moment où le message est chiffré, on ne peut plus le déchiffrer sans l'aide de la clé privée. Même B, qui est l'auteur du message, ne peut le déchiffrer. [8]

- ❖ Le déchiffrage :

Maintenant que A a reçu le message de B "386 1858 2127 2809 1858 1774 737 3675 244". A utilise sa clé privée pour déchiffrer le message.

Etape1 : premier calcul, la puissance :

Il faut élever chaque sous-message (nombre) à une puissance. Cette puissance sera U

386⇔396^{4279}, 1858⇔1858^{4279}, …

Etape2 : Le modulo :

Dans cette étape, on va calculer le modulo des résultats obtenus précédemment par N (Nous avons N = 5141)

386⇔(396^{4279}) modulo (5141), 1858⇔(1858^{4279}) modulo (5141), …

Etape2 : Le remplacement :

Les résultats que nous venons d'obtenir son en réalité la valeur ASCII du caractère original. On va donc se référer à la table ASCII et effectuer les remplacements nécessaires.

Ce qui va nous donner le message en clair : Bonjour ! [8]

- ❖ Le déchiffrage et le chiffrage en une fonction :

Pour le Chiffrage on peut résumer les étapes en une fonction : **$f(x) = x^c \bmod (N)$**. Où x est la valeur ASCII du caractère à chiffrer.

Même chose pour le déchiffrage quasiment la même fonction : **$f(x) = x^u \bmod (N)$**. Où U et N les valeurs de la clé privée, et x le sous-message. Cette fonction renvoie un nombre qui est la valeur ASCII du caractère chiffré. [8]

Partie pratique :

- ***Les bibliothèques d'arithmétique multi-précision :***

En informatique dans un langage comme le C, les types natifs d'entiers permettent de représenter des nombres sur 64 bits. Certaines instructions en assembleur x86 permettent de monter jusqu'à 128 bits. En tout état de cause, on est très loin des clés de 1024 voire 2048 ou 4096 bits qu'un programme implémentant RSA a besoin de pouvoir manipuler. Il faut donc recourir à une bibliothèque d'arithmétique multi-précision. [7]

La plus utilisée en C (et en C++ par l'intermédiaire d'une surcouche) est la GNU Multiple Precision Arithmetic Library ou GMP pour les intimes. Ce n'est pas la seule : en C++, la Class Library for Numbers ou CLN est nettement moins utilisée mais présente l'avantage d'être codée nativement en C++, plutôt que « d'enrober » du C. Mais objectivement, la GMP domine le paysage informatique. En effet, de nombreux langages implémentent l'arithmétique multi-précision dans leur bibliothèque standard (C#, Java, OCaml, Perl, PHP, etc.) voire nativement dans le langage (Erlang, Haskell, Python, Ruby, etc.) ; cependant, la plupart de ces langages font en réalité appel à GMP de manière transparente pour l'utilisateur. [7]

- ***Base64 :***

Le gros inconvénient du chiffrement RSA, c'est qu'une fois chiffré, un simple texte ne ressemble plus à rien, et cela devient excessivement difficile de l'enregistrer dans un fichier ou de le transmettre par mail. De même, les clés publiques et clés privées étant de grands nombres, il n'est pas dit que les interpréter comme du texte donne quoi que ce soit de potable. Ce n'est pas trop gênant pour les conserver, déjà plus pour les transmettre à quelqu'un. [7]

La solution la plus couramment adoptée et la plus économique, consiste à convertir tout ce qui n'est pas du texte brut ou un type de fichier usuel (JPEG, etc.) en base64. Cet encodage consiste à découper les données en tronçons de trois octets soit 24 bits, et à représenter chaque groupe de 6 bits de ce tronçon par un caractère : comme il n'y a plus que 64 possibilités, on peut n'utiliser que des caractères courants. En l'occurrence, les lettres de l'alphabet en minuscule et majuscule, les dix chiffres et les caractères « + », « / » et « = », ou alors, dans une variante destinée à être utilisée dans les URL, « - », « _ » et « = ».

PHP possède nativement les fonctions base64_encode() et base64_decode(). [7]

.VI Conclusion

Que l'on soit débutant ou spécialiste, il est particulièrement difficile de concevoir des applications web sécurisées et fiables. Il est d'ailleurs fortement recommandé par l'OWASP d'intégrer la sécurité dans une application dès sa conception plutôt que d'identifier les failles et les corriger après.

Afin de produire une application web sécurisée il est important de définir les besoins de l'application en termes de sécurité. Il faut nécessairement être au courant des problèmes connus et des failles identifiées au cours des dernières années pour être en mesure de s'en protéger et cela en adoptant au mieux un plan de sécurité maintenu à jour.

Il faut savoir aussi que ces associations comme l'OWASP et le WASC proposent des outils gratuits et des guides pour justement aider les développeurs à réduire les risques au sein de leurs applications web.

CHAPITRE II

Outils utilisés et modélisation

.I Introduction

Pour comprendre toute la puissance et l'importance de la sécurité dans le cadre des applications web ; nous avons mis en place un exemple d'application qui illustre la prévention et la protection contre les attaques et les failles de sécurité à l'aide d'un mécanisme de contrôle d'accès des utilisateurs, de chiffrement RSA des mots de passe et de gestion des rôles.

Ainsi tout au long de ce chapitre, nous verrons comment l'application a été conçue en suivant le modèle MVC que nous avons conçu (proposé). Nous aborderons en détails le choix des outils utilisés dans l'implémentation pour assurer un niveau optimal de sécurité.

.II Description des outils utilisés

Pour l'implémentation de l'application, notre choix s'est arrêté sur PHP et MySql en suivant le modèle MVC.

Il faut savoir que l'une des caractéristiques intéressantes de PHP et de MySQL tient à ce qu'ils fonctionnent avec tous les systèmes d'exploitation les plus connus et avec la plupart des autres. [9]

Un script PHP peut, dans la plupart des cas, être écrit de façon à être portable entre les systèmes d'exploitation et les serveurs web. [9]

En plus de ces caractéristiques ; PHP et MySql présentent de nombreux points forts qui justifient notre choix et notre intérêt pour ces produits. C'est ce que nous verrons dans ce qui suit.

.II.1 PHP

.II.1.1 Description de PHP

PHP (officiellement, ce sigle est un acronyme récursif pour PHP: Hypertext Preprocessor) est un langage de scripts généraliste et Open Source, spécialement conçu pour le développement d'applications web. Il peut être intégré facilement au HTML. [10]

PHP a été conçu en 1994 par Rasmus Lerdorf. Il a ensuite été adopté par d'autres personnes talentueuses et réécrit quatre fois avant de devenir le produit abouti que nous connaissons aujourd'hui. En novembre 2007, il a été installé sur plus de 21 millions de domaines et sa croissance est rapide.

PHP (PHP Hypertext PreProcessor) est un langage de programmation. Sa principale application se situe au niveau de la gestion des sites Web dynamiques. On peut par exemple lui faire créer le contenu de pages HTML suivant différents paramètres : l'âge d'un visiteur, sa catégorie socioprofessionnelle, des mots-clés qu'il aura indiqués dans un moteur de recherche, des actualités du jour, etc. [11]

Figure II-1: Gestion de pages dynamiques avec PHP *[12]*

Les capacités de PHP ne s'arrêtent pas à la création de pages Web. Il est aussi possible de manipuler des images, de créer des fichiers PDF, de se connecter à des bases de données ou des serveurs LDAP, et même d'instancier des objets Java. Un module annexe lui permet également de fournir des interfaces graphiques classiques (client riche, sans navigateur ou serveur Web), via GTK. [11]

Nouveautés de PHP 5

Le développement de PHP 5 a été entamé en 2002, mais c'est l'année 2003 qui a été la plus active. L'objectif était double : d'une part, rendre PHP plus professionnel, mais également le rendre plus simple. [11]

La première version stable de PHP 5 a fait son apparition en 2004. Par rapport à la version 4, ses principales nouveautés sont : [11]

- l'intégration du *Zend Engine* 2, qui amène une prise en charge complète de la programmation orientée objet ;
- la refonte de la prise en charge de XML ;
- l'intégration de la base de données SQLite ;
- la simplification des principales tâches courantes. [11]

.II.1.2 Quelques avantages de PHP

Les principaux concurrents de PHP sont Perl, ASP.NET de Microsoft, Ruby (avec ou sans Rails), Java Server Pages (JSP) et ColdFusion. [9]

Globalement, il faut garder en tête qu'à chaque problème correspond sa solution et qu'il est difficile de dire que tel langage ou tel autre est meilleur de façon générale. Cependant, PHP 5 dispose par rapport à ses concurrents de quelques particularités et avantages significatifs. [11]

- les performances ;
- l'adaptabilité ;
- des interfaces vers différents systèmes de bases de données ;
- des bibliothèques intégrées pour la plupart des tâches web ;
- un faible coût ;
- la simplicité d'utilisation et d'apprentissage ;
- un bon support orienté objet ;
- la portabilité ;
- la souplesse dans le processus de développement ;
- la disponibilité de son code source ;
- la disponibilité du support et de la documentation. [9]

Allons voir plus en détails quelques-uns de ces avantages :

a- Intégration avec les bases de données

PHP dispose de connexions natives vers la plupart des systèmes de bases de données. Outre MySQL, vous pouvez vous connecter directement aux bases de données PostgreSQL, Oracle, dbm, FilePro, DB2, Informix, InterBase et Sybase, pour ne citer qu'elles. PHP 5 possède également une interface SQL intégrée, SQLite, pour gérer les fichiers plats.

En fait, grâce au standard ODBC (Open Database Connectivity), vous pouvez vous connecter à n'importe quelle base de données possédant un pilote ODBC, ce qui est le cas des produits Microsoft, notamment.

Outre ces bibliothèques natives, PHP dispose d'une couche d'abstraction pour l'accès aux bases de données : PDO (PHP Database Objects) autorise ainsi une certaine cohérence et facilite la prise en compte de la sécurité lors de l'accès aux bases. [9]

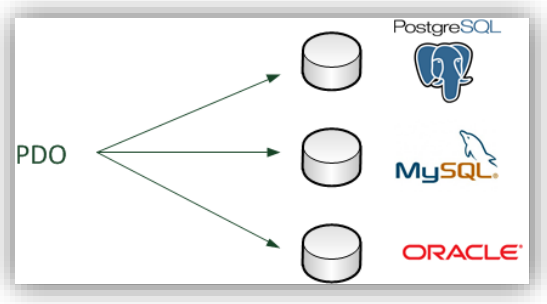

Figure 0II-2: PDO permet de se connecter à n'importe quel type de base de données *[13]*

b- Bibliothèques intégrées

PHP ayant été conçu pour être utilisé sur le Web, il possède de nombreuses fonctions intégrées permettant d'effectuer la plupart des tâches de programmation web.
Vous pouvez ainsi générer des images en temps réel, vous connecter à des services web et à d'autres services réseaux, analyser des documents XML, envoyer du courrier électronique, manipuler les cookies et produire des documents PDF avec seulement quelques lignes de code. [9]

c- Support orienté objet

PHP 5 possède des fonctionnalités orientées objet bien conçues. Si vous avez appris à programmer en Java ou en C++, vous disposerez en PHP des fonctionnalités (et, généralement, de la syntaxe) dont vous avez l'habitude : l'héritage, les attributs et les méthodes privées et protégées, les classes et les méthodes abstraites, les interfaces, les constructeurs et les destructeurs, notamment. Vous découvrirez également des mécanismes moins courants, comme les itérateurs. Certaines de ces fonctionnalités étaient disponibles dans PHP 3 et 4, mais le support orienté objet de la version 5 est bien plus complet. [9]

d- Souplesse dans le processus de développement

PHP permet d'implémenter simplement les traitements simples, mais vous pouvez tout aussi facilement implémenter de grosses applications à l'aide d'un framework reposant sur les patrons de conception, comme Modèle-Vue-Contrôleur (MVC). [9]

.II.2 MySQL

.II.2.1 Présentation de MySQL

MySQL est un système de gestion de bases de données relationnelles (SGBDR) robuste et rapide. Une base de données permet de manipuler les informations de manière efficace, de les enregistrer, de les trier, de les lire et d'y effectuer des recherches. Le serveur MySQL contrôle l'accès aux données pour s'assurer que plusieurs utilisateurs peuvent se servir simultanément d'une même base de données pour y accéder rapidement et pour garantir que seuls les utilisateurs autorisés peuvent accéder aux données. MySQL est donc un serveur multi-utilisateurs et multithread. Il utilise SQL (*Structured Query Language*), le langage standard des requêtes de bases de données. MySQL est disponible depuis 1996, mais son développement remonte à 1979. Il s'agit de la base de données open-source la plus employée au monde. [9]

.II.2.2 Quelques avantages de MySQL

Les principaux concurrents de MySQL sont PostgreSQL, Microsoft SQL Server et Oracle. [9]

MySQL possède sur eux plusieurs avantages :

- des performances élevées ;
- un coût réduit ;
- une simplicité de configuration et d'apprentissage ;
- sa portabilité ;
- l'accessibilité de son code source ;
- la disponibilité du support. [9]

.II.3 SQL

SQL signifie « Structured Query Language » c'est-à-dire « Langage d'interrogation structuré ».

En fait SQL est un langage complet de gestion de bases de données relationnelles. Il a été conçu par IBM dans les années 70. Il est devenu le langage standard des systèmes de gestion de bases de données (SGBD) relationnelles (SGBDR). [14]

C'est à la fois :

- Un langage de manipulation des données (LMD).

  - Un langage d'interrogation (ordre SELECT).
  - Un langage de manipulation des données (UPDATE, INSERT, DELETE).
- Un langage de définition des données (LDD : ordre CREAT, ALTER, DROP).

Un langage de contrôle de l'accès aux données (LCD : ordre GRANT, REVOKE). [14]

.II.4 MVC

Il y a des problèmes en programmation qui reviennent tellement souvent qu'on a créé toute une série de bonnes pratiques que l'on a réunies sous le nom de *design patterns*. Un des plus célèbres *design patterns* s'appelle MVC, qui signifie **Modèle - Vue - Contrôleur**. [15]

Le pattern MVC permet de bien organiser son code source. Il va vous aider à savoir quels fichiers créer, mais surtout à définir leur rôle. Le but de MVC est justement de séparer la logique du code en trois parties que l'on retrouve dans des fichiers distincts, comme l'explique la description qui suit. [15]

- **Modèle** : cette partie gère les *données* de votre site. Son rôle est d'aller récupérer les informations « brutes » dans la base de données, de les organiser et de les assembler pour qu'elles puissent ensuite être traitées par le contrôleur. On y trouve donc les requêtes SQL. [15]

- **Vue** : cette partie se concentre sur *l'affichage.* Elle ne fait presque aucun calcul et se contente de récupérer des variables pour savoir ce qu'elle doit afficher. On y trouve essentiellement du code HTML mais aussi quelques boucles et conditions PHP très simples, pour afficher par exemple la liste des messages des forums. [15]

- **Contrôleur** : cette partie gère la logique du code qui prend des *décisions*. C'est en quelque sorte l'intermédiaire entre le modèle et la vue : le contrôleur va demander au modèle les données, les analyser, prendre des décisions et renvoyer le texte à afficher à la vue. Le contrôleur contient exclusivement du PHP. C'est notamment lui qui détermine si le visiteur a le droit de voir la page ou non (gestion des droits d'accès). [15]

.II.4.1 *Architecture MVC d'une application web*

Le modèle MVC cherche à séparer nettement les couches présentation, traitement et accès aux données. Une application web respectant ce modèle sera architecturée de la façon suivante : [16]

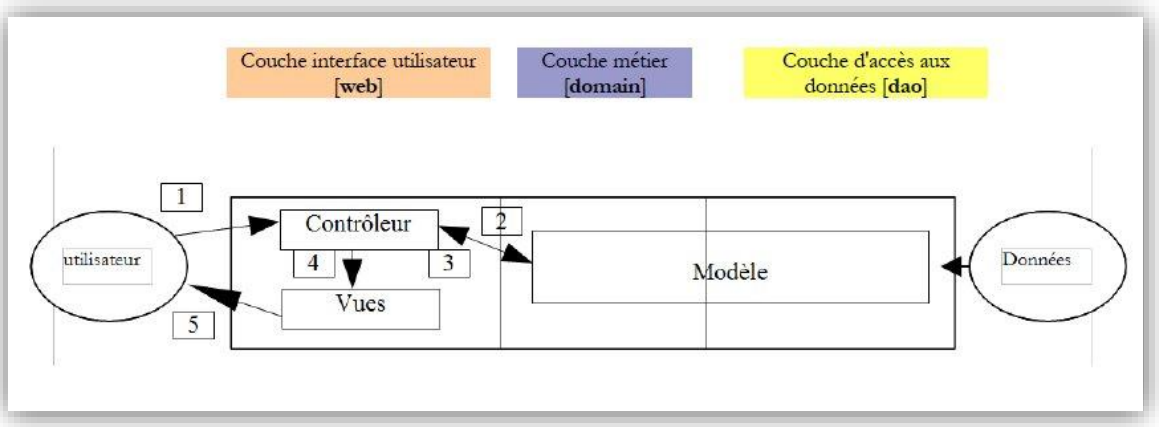

Figure II-3: Architecture MVC d'une application web/php *[16]*

Le traitement d'une demande d'un client se déroule selon les étapes suivantes :

1. le client fait une demande au contrôleur. Ce contrôleur voit passer toutes les demandes des clients. C'est la porte d'entrée de l'application. C'est le C de MVC.
2. le contrôleur traite cette demande. Pour ce faire, il peut avoir besoin de l'aide de la couche métier, ce qu'on appelle le modèle M dans la structure MVC.
3. le contrôleur reçoit une réponse de la couche métier. La demande du client a été traitée. Celle-ci peut appeler plusieurs réponses possibles. Un exemple classique est :
    a. une page d'erreurs si la demande n'a pu être traitée correctement
    b. une page de confirmation sinon
4. le contrôleur choisit la réponse (= vue) à envoyer au client. Celle-ci est le plus souvent une page contenant des éléments dynamiques. Le contrôleur fournit ceux-ci à la vue.
5. la vue est envoyée au client. C'est le V de MVC. [16]

.II.4.2 Les frameworks PHP

De nombreux *frameworks*[2] PHP vous permettent de mettre rapidement en place les bases d'une architecture MVC sur votre site. Ce sont des outils appréciés des professionnels qui nécessitent cependant un certain temps d'adaptation. [15]

Il en existe beaucoup dont :

- CodeIgniter.
- CakePHP.
- Symfony.
- Jelix.
- Zend Framework.

Ce sont des outils puissants et complets. Ils sont tous basés sur une architecture MVC et proposent en outre de nombreux outils pour faciliter le développement de son site web. [15]

Pour utiliser ce genre d'outils, il faut avoir de bonnes connaissances en PHP et en programmation orienté objet. Il faut aussi avoir bien compris l'architecture MVC et même avec tout cela ; il faut un certain temps d'adaptation au framework choisi et surtout accepter de suivre une certaine manière d'organiser ses fichiers.

Dans notre cas, il est plus intéressant de comprendre déjà le fonctionnement du MVC et donc de le concevoir soit même, mais aussi de mettre en pratique le côté orienté objet de la programmation PHP au cœur du fonctionnement de l'application : Appeler le bon contrôleur, charger le modèle requis, afficher la vue à l'utilisateur avec les données qu'il a demandé…etc.

.III Conception de notre application selon le design pattern MVC

.III.1 Pourquoi MVC ?

Au lieu de se lancer tête baissée dans la programmation de notre application en écrivant le code de façon intuitive sans nous soucier de l'organisation, nous nous retrouverons rapidement avec un code très mal structuré ; des requêtes SQL mélangées à des appels de fonctions PHP et de l'affichage en HTML tout cela dans un même fichier. Encore qu'avec

[2] Un ***framework*** est un ensemble de bibliothèques, une sorte de *kit* prêt à l'emploi pour créer plus rapidement son site web, tout en respectant des règles de qualité. (Ré. : open class room)

un site web simple cela ne poserait pas problème mais avec une application web qui intègre des interfaces d'édition de produit et de contrôle d'accès ; il devient très vite difficile de maintenir le code, de le mettre à jour et surtout de garantir un certain niveau de sécurité et de fiabilité.

.III.2 Architecture MVC proposée

Afin de comprendre tout l'intérêt d'utiliser l'architecture MVC dans l'élaboration d'une application web, nous proposons ici notre propre MVC ce qui va nous aider à mettre en place notre application avec un code qui soit facile à maintenir et à faire évoluer. Cela prendra beaucoup plus de temps en programmation et il y'aura plus de fichiers mais au final le code sera beaucoup plus clair et mieux découpé.

L'architecture MVC est bien adaptée à des applications web écrites avec des langages orientés objet [16]. Le langage PHP 5 est orienté objet, ce qui nous arrange parfaitement dans l'élaboration du MVC de notre application.

Nous organisons nos fichiers de la manière suivante :

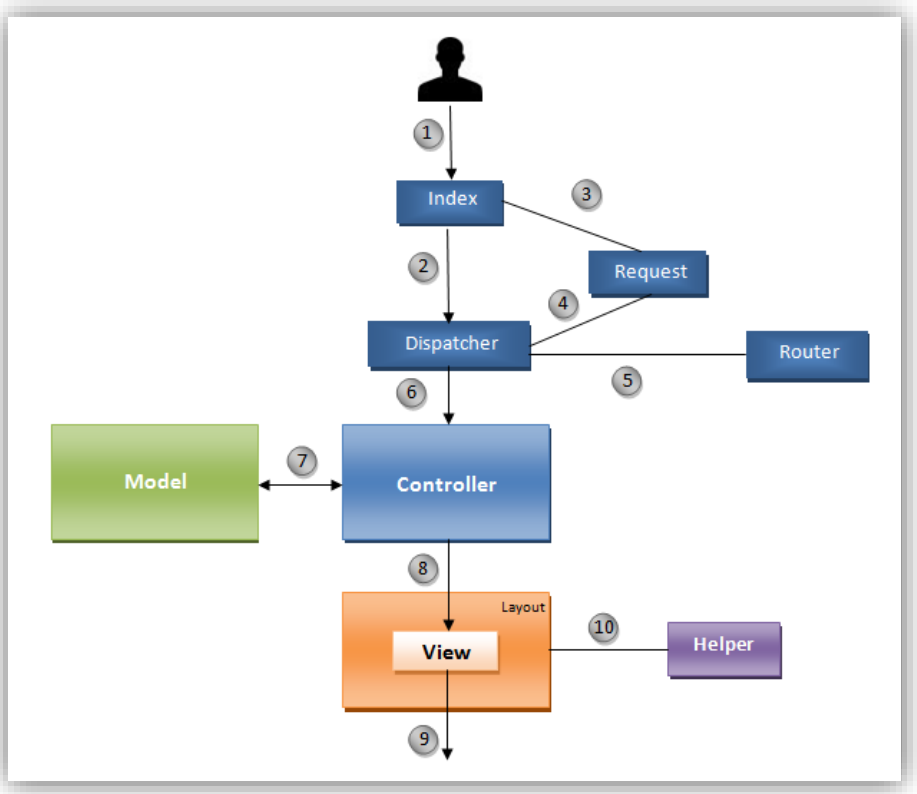

Figure 0II-4: L'architecture MVC proposée

Sur le schéma ci-dessus, nous pouvons voir le déroulement des étapes lorsqu'un utilisateur essaye d'accéder à un contenu du site :

1- Toutes les URLs demandés par l'utilisateur redirigeront vers un fichier index.
2- index inclus tous les fichiers dont on aura besoin pour le fonctionnement de l'application. On instancie dans index la class Dispatcher. Dispatcher est un objet qui va nous permettre de rediriger et initialiser le bon composant en fonction de l'URL tapée.
3- Disptacher récupère un objet Request. Cet objet stocke toutes les informations concernant la requête qui a été faite par l'utilisateur. Exemple : (URL tapée, paramètres, …etc.)
4- Le Dispatcher récupère l'URL grâce à l'objet Request mais il ne sait toujours pas comment l'interpréter.
5- Le Dispatcher fait appel à un objet Rooter qui lui va analyser l'URL dans les deux sens. Dans un premier sens il va interpréter l'URL tapée par l'utilisateur et dans un deuxième sens il va générer et réécrire l'URL qu'on envoi à l'utilisateur.
6- Le Rooter découpe l'URL tapée par l'utilisateur et prépare les variables Controller, action et paramètres s'il y a lieu. Le Disptacher peut maintenant savoir quel Controller appeler et quelle action exécuter. Enfin le Dispatcher initialise le bon Controller et appel l'action.
7- Dans notre Controller on fait appel au Model requis.
8- Après traitement, le Controller envoie les résultats à la vue.
9- L'utilisateur reçoit donc le résultat de sa requête et peut voir la vue qu'il a demandée.
10- On utilise ce qu'on appellera des Helpers pour afficher par exemple les données enregistrées dans les sessions ou générer rapidement des formulaires.

.IV Conclusion

Durant ce chapitre nous avons essayé de présenter en quelques lignes les outils utilisés pour la conception et la réalisation de l'application. Nous avons aussi essayé de justifier ces choix.

Afin de réaliser l'application, nous avons, dans un premier temps, essayé de comprendre l'intérêt d'utiliser une telle architecture dans le développement web, ensuite nous avons

mis en place l'architecture MVC de l'application et schématiser la structure des fichiers afin de permettre une implémentation efficace de l'application.

La conception de notre application avec PHP5 et MySQL dans une architecture MVC promet fiabilité, simplicité et souplesse ; c'est ce que nous verrons dans le chapitre III.

CHAPITRE III

Réalisation de l'application

.I Introduction

Nous voici arrivé à la dernière partie, la partie pratique de ce projet. En ce dernier chapitre, nous verrons comment l'application a été réalisée selon le design pattern MVC que nous avons proposé dans le chapitre 2. Nous verrons aussi toute la souplesse de PHP et son côté orienté objet qui va nous permettre de coder notre application en respectant le modèle MVC. Nous verrons dans un premier temps, l'objectif de l'application et ensuite nous procéderons à la description de chaque élément pour comprendre le fonctionnement et le côté sécurité. Tout au long de la programmation, nous avons tenu à respecter au mieux les conseils des experts en sécurité vu précédemment dans le chapitre 1. Nous mettrons en pratique le chiffrement RSA pour voir qu'il est possible de sécuriser ses données dans les applications web côté client et côté serveur.

.II Description de l'application

Le but de ce projet est d'étudier la sécurité en générale dans les applications web. Pour ceci, nous avons schématisé un des aspects de sécurité qui est l'authentification et la gestion des rôles ainsi que le chiffrement RSA dans une application qu'on appellera « Dispositifs en ligne ».

Ces dernières années ont vu l'arrivée en masse d'applications web disponibles sur plusieurs types de systèmes et de machines. Aujourd'hui tout le monde veut que son site web soit affichable ; tout aussi bien sur un ordinateur de bureau que sur un téléphone ou sur une tablette… etc. Nous avons donc imaginé que l'université offrirai à ses étudiants et enseignants un service en ligne de location ou d'empreint de dispositifs électroniques comme des ordinateurs portables, des smartphones (téléphones intelligents), des tablettes …etc. pour faire des tests de leurs travaux ou les utiliser dans le cadre de leurs études en général.

.III Structure de l'application

.III.1 Fonctionnement pratique de notre MVC avec l'exemple de la page Inscription

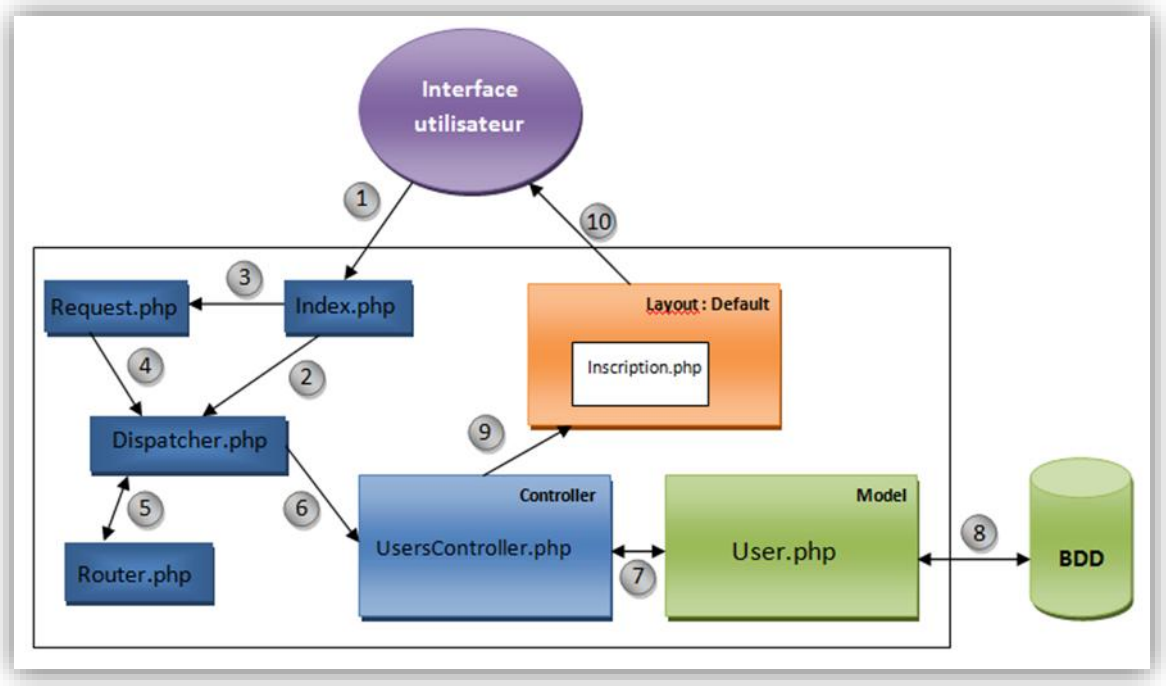

Figure III-1: Cas pratique du fonctionnement de notre MVC

1. Lorsque l'utilisateur veut accéder à la page inscription, il arrive sur index.php.
2. index.php instancie l'objet Disptacher.
3. Request récupère l'url : http://localhost/dispositifs_en_ligne/Inscription ou (http://localhost/dispositifs_en_ligne/users/inscription).
4. Request communique au Disptacher l'url.
5. Dispatcher fait appel au rooter qui lui va interpréter l'url et dire au disptacher que le contrôleur demandé est UsersController et l'action demandée est inscription.
6. A présent Dispatcher sait qu'il faut appeler le contrôleur UsersController.php et l'action inscription ().
7. Le contrôleur peut avoir besoin du modèle User.php.
8. Le modèle échange des données avec la base de données.
9. Le contrôleur après avoir fait les traitements requis envoie les données à la vue Inscription qui sera affichée à l'utilisateur.

10. L'utilisateur peut alors effectuer l'opération d'inscription qui sera détaillée un peu plus loin.

.IV Implémentation de l'application

.IV.1 Présentation de la page d'accueil

L'application réalisée suivant le modèle MVC en utilisant HTML et CSS pour la mise en forme présente une interface très conviviale, moderne et simple. L'internaute en accédant à l'application se retrouvera devant une page d'accueil contenant des liens pour atteindre les autres pages.

Nous avons choisi un design simple et moderne en utilisant des formes et des couleurs qui s'adaptent au contenu du site web. Ce dernier est « responsive » il a été fait de manière flexible, l'affichage s'adapte à différents mode : écran d'ordinateur, tablette ou smartphone.

Figure III-2: Page d'accueil de l'application web « Dispositifs en ligne »

.IV.2 L'inscription

Pour avoir accès à la liste des dispositifs il faut avoir un compte pour s'authentifier. Pour ceci nous avons réalisé une interface web pour l'inscription.

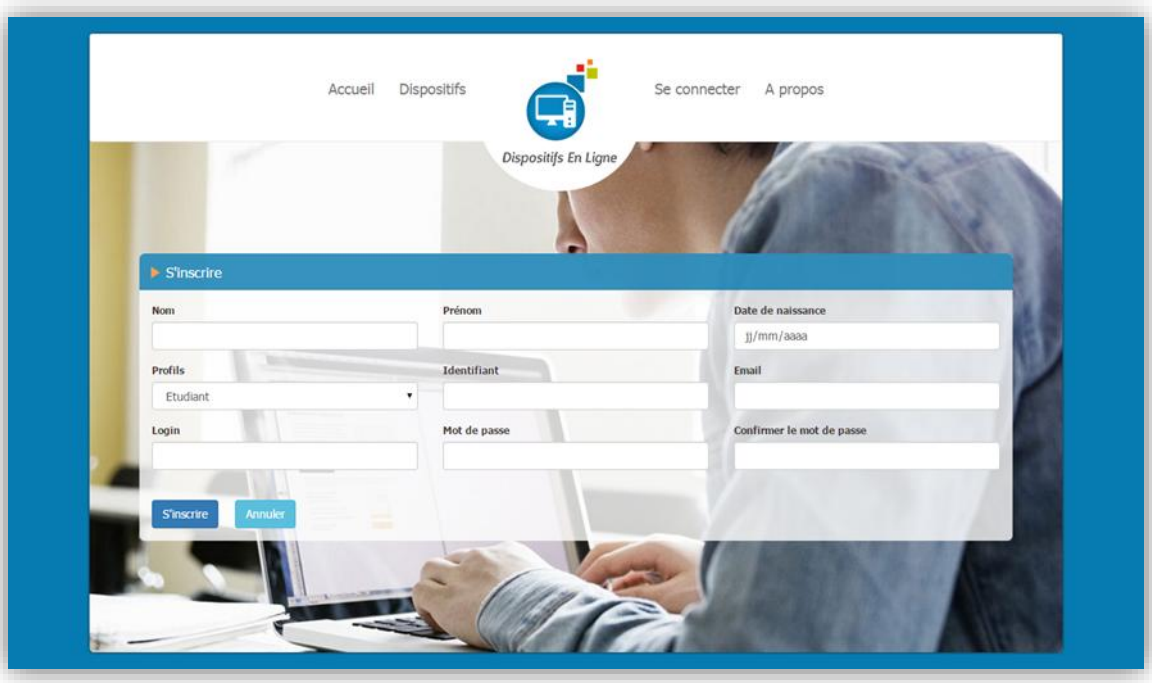

Figure III-3: Page inscription

Côté client :

- A la soumission du formulaire avec une fonction Javascript validateForm(), nous vérifions que le mot de passe à bien été rempli et qu'il n'est pas plus court que 7 caractères et plus long que 15. Ensuite nous vérifions que le champ « Confirmer le mot de passe » a bien été rempli avec le même mot de passe.
- Nous procédons ensuite au hashage du mot de passe en sha1 avec la fonction javascript hash (). Pour cela nous utilisons la bibliothèque pidCrypt[3] crypto javascript qui regroupe plusieurs opérations de hashage mais aussi de cryptage RSA.
- Maintenant nous cryptons en RSA le hashage obtenu et ceci avec la fonction javascript compute() de la même bibliothèque javascript utilisée pour le hashage. Nous avons au préalable envoyé la clé publique (la clé publique a été générée en PHP à part et stockée dans un fichier conf.php) de notre application à la vue inscription pour pouvoir l'utiliser côté client avec javascript.

[3] Voir annexe A

Côté serveur :

Avec le Contrôleur UsersController nous allons précéder à plusieurs étapes avant d'inscrire l'utilisateur :

1. Eviter les failles XSS :

Pour cela nous allons nous assurer que les données récupérées sont fiables en validant chaque donnée grâce aux expressions régulières. Nous faisons alors appel au modèle User qui lui va se charger de vérifier chaque champ.

Ainsi si tous les champs ont bien été validés nous passons à la deuxième étape. Sinon nous affichons un message pour que l'utilisateur sache qu'il faut corriger tel ou tel champ.

2. Comparer les données reçues avec la base de données :

Nous devons nous assurer que la personne qui essaye de s'inscrire n'est pas déjà inscrite. Nous vérifions aussi si la personne n'a pas choisi le même login qu'un autre utilisateur présent en base de données.
Si la personne qui veut s'inscrire choisi un profil étudiant ou enseignant nous comparons ses informations avec les deux tables de notre base de données « Enseignants » et « Etudiants » pour vérifier d'après le profil choisi que c'est bien un étudiant ou un enseignant inscrit à l'université.
Si les informations récupérées du formulaire ne permettent pas l'inscription nous affichons un message d'erreur sinon nous passons à l'étape suivante.

3. Maintenant que toutes les données ont été vérifiées et validées nous pouvons les enregistrer en base de données.
4. Il faut à présent envoyer un email à l'utilisateur pour le prévenir qu'il doit valider son inscription. Pour cela nous utilisons *SwiftMailer* une librairie PHP permettant d'envoyer facilement des emails.
5. Si l'utilisateur valide correctement son inscription, avec la méthode validate () du contrôleur UsersController nous mettons à jour la tables User pour terminer l'inscription sinon nous affichons un message d'erreur.

.IV.3 L'authentification

Un internaute simple ne peut pas accéder à la page « Liste des dispositifs ». C'est là que nous mettons notre système d'authentification pour s'assurer qu'il n'y a que les utilisateurs authentifiés qui peuvent consulter la liste des dispositifs.

Un utilisateur inscrit et qui a bien validé son inscription peut s'authentifier grâce à l'interface d'authentification.

Figure III-4: Page Se connecter

Dans cette partie nous aurons besoin de crypter le mot de passe côté client et le décrypter côté serveur.

Le crypter côté client pour la même raison que pour l'inscription : éviter de faire circuler le mot de passe en clair pour se prémunir du vol de données secrètes.

Côté client :

Même enchainement d'opérations que ce qu'on a vu plus tôt avec l'inscription : hashage du mot de passe en sha1 et ensuite cryptage du hash en RSA avec la clé publique.

Côté serveur :

1- Avec la méthode login() du contrôleur UsersController tout d'abord nous devons nous assurer que l'utilisateur qui a le login par exemple « h_selka » est bien inscrit et qu'il a validé son inscription.
2- Si oui, nous récupérons de la base de données le mot de passe crypté en RSA correspondant au login et nous décryptons en utilisant la bibliothèque phpseclib[4]

[4] Voir annexe A

avec la méthode decrypt() de la class Crypt_myRSA à l'aide de la clé privé de l'application. Cette clé tout comme la clé publique à été générée à part et stockée dans le fichier conf.php.

3- De la même manière nous décryptons le mot de passe crypté récupéré du formulaire de connexion.
4- A présent nous comparons les résultats des deux décryptages. Rappelons que le résultat du décryptage ne donne pas le mot de passe en clair mais plutôt le hashage de ce dernier.
5- Si le mot de passe récupéré du formulaire correspond à celui stocké en base de données nous autorisons l'utilisateur à accéder au contenu du site web.

En plus de vérifier que l'utilisateur est bien authentifier ; nous vérifions son profil.

Selon son profil (Soit étudiant, soit enseignant, soit administrateur comme préalablement préparer dans la base de données) nous afficherons la vue correspondante.

Dans notre base de données nous avons deux tables qui nous permettent de mettre en œuvre notre système d'authentification : la table USERS et la table PROFILS.

Nous utilisons le « helper Session » pour enregistrer dans la session en cours les informations récupérées de la base de données après authentification mais aussi toutes les actions concernant les réservations de l'utilisateur. La session de l'utilisateur connecté expire après un moment si l'utilisateur ne présente aucun mouvement dans l'application.

.IV.4 La gestion des rôles

A l'aide d'un contrôleur « UsersController », des Sessions et des deux tables « USERS & PROFILS » dans la base de donnée ; notre application gère trois profils :

- Enseignant.
- Etudiant.
- Administrateur.

Les profils enseignant et étudiant ont le même niveau (niveau 1), quand à l'administrateur ; il bénéficie d'un niveau supérieur (niveau 2).

Les actions possibles pour les profils enseignant et étudiant :

- Si authentifiés ils peuvent tous deux visualiser la page « Liste des dispositifs ».

- Ils peuvent tous les deux effectuer des réservations tant que le nombre de produits restant est suffisant.
- Ils sont tous les deux limités à trois réservations à la fois.
- Ils peuvent tous les deux voir un aperçu de leur profil et leurs réservations en cours.
- Aucun des deux ne peut accéder à la partie administrative du site web.
- Un enseignant peut voir certains produits que l'étudiant ne peut pas voir et vis versa.
- Un enseignant réserve avec un prix alors que l'étudiant réserve gratuitement le dispositif.

Les actions possibles pour le profil Administrateur :

- Si authentifié il peut accéder à la partie administrative « Le Backoffice ».
- Il peut visualiser tous les produits au niveau du site web.
- Il peut ajouter et modifier un dispositif.
- Il peut à partir du site web accéder à l'édition d'un produit en cliquant sur le bouton « Editer » qui se trouve à côté du produit.
- Il peut choisir de mettre en ligne un dispositif et donc il sera visible au niveau du site ou alors le masquer pour une raison quelconque (maintenance ou autres).
- Il peut choisir d'afficher le dispositif pour les étudiants seulement ou pour les enseignants ou pour les deux en même temps sinon pour aucun des deux.
- Il peut ajouter une ou plusieurs photos pour chaque dispositif. Il peut aussi supprimer ces photos.
- Il peut voir la liste des réservations en cours en allant à la page Réservations.
- Il peut voir quel dispositif est réservé par quel utilisateur.
- Il ne peut pas modifier ou ajouter une réservation mais il peut en supprimer seulement à partir de la page Réservations.
- Il peut accéder à tout moment à la partie administrative du site en utilisant le lien dans le menu.
- Il peut tout aussi bien accéder à partir du « backoffice » au site web pour voir les dispositifs en ligne.

Nous verrons tout cela en images dans ce qui suit.

.IV.5 La page Liste des dispositifs

Si l'utilisateur est authentifié et a bien un des trois profils « étudiant – enseignant – administrateur) une icône profils sera affichée à côté de son nom et prénom et nous

ajouterons un lien pour qu'il puisse se déconnecter. Même chose dans le menu, nous remplaçons « Se connecter » par « Se déconnecter ».

Figure III-5: Affichage de la page Liste des dispositifs pour le profil Etudiant

Selon qu'il soit étudiant, enseignant ou administrateur le bouton à côté de chaque produit varie.

Figure III-6: Bouton réserver pour le profil "étudiant", réserver avec prix pour "enseignant", éditer pour "administrateur"

.IV.6 La réservation d'un dispositif

Un utilisateur authentifié à la possibilité de réserver jusqu'à trois dispositifs différents. Une réservation ne peut être annulée que par l'administrateur du site web.

Lorsqu'un dispositif est réservé et qu'il n'est plus disponible en stock ; le bouton devient gris et désactivé comme le montre l'image suivante :

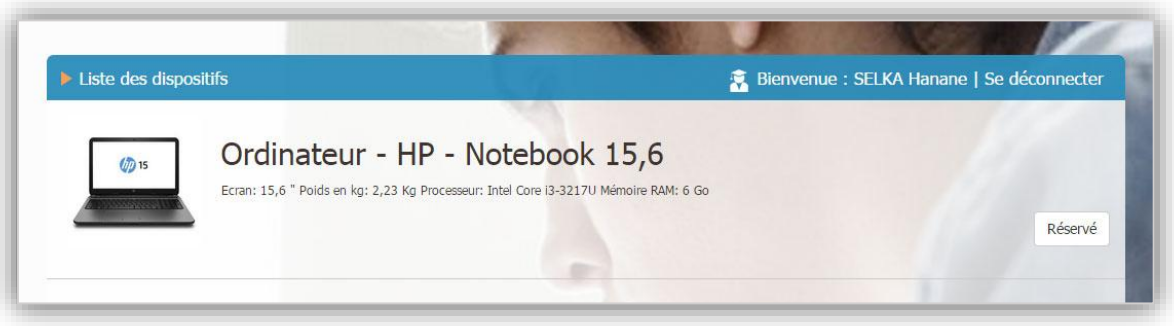

Figure III-7: Bouton Réserver désactivé. Produit déjà réservé

.IV.7 Aperçu profil

A l'accueil du site web, lorsqu'un utilisateur est connecté nous remplaçons l'encadré Se connecter par un encadré Profils. Ainsi l'utilisateur peut avoir un aperçu de son profil et un récapitulatif de ses réservations en cours.

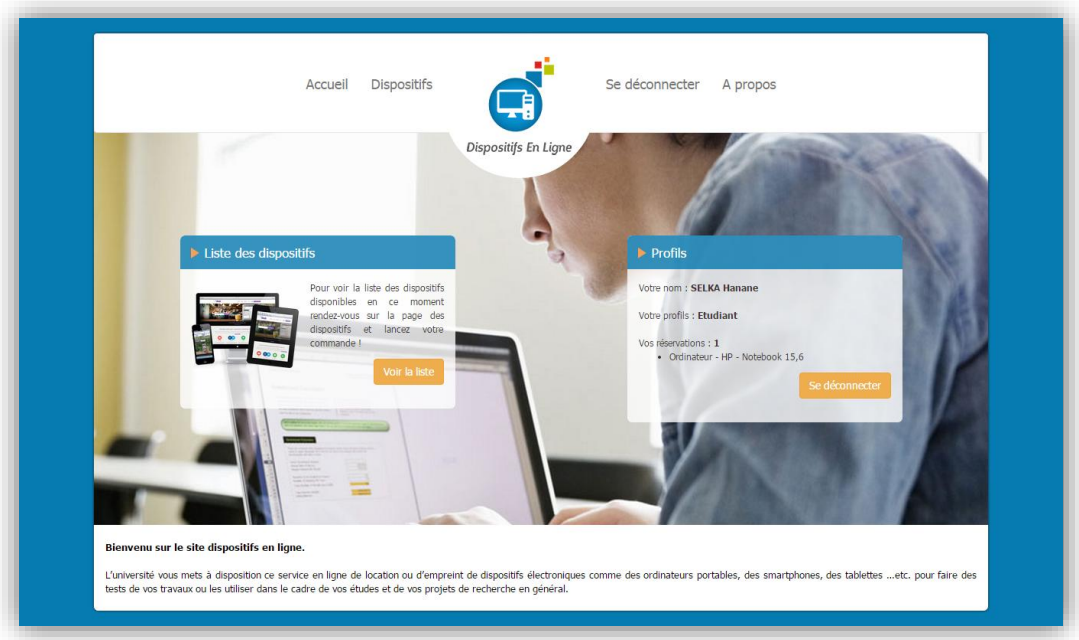

Figure III-8: Aperçu profil

.IV.8 Le Backoffice

Accueil

Lorsque l'utilisateur connecté à un profil Administrateur il est automatiquement redirigé vers le backoffice. Avant toutes actions nous nous assurons que nous avons bien un utilisateur connecté et que son profil lui permet ou non une telle action.

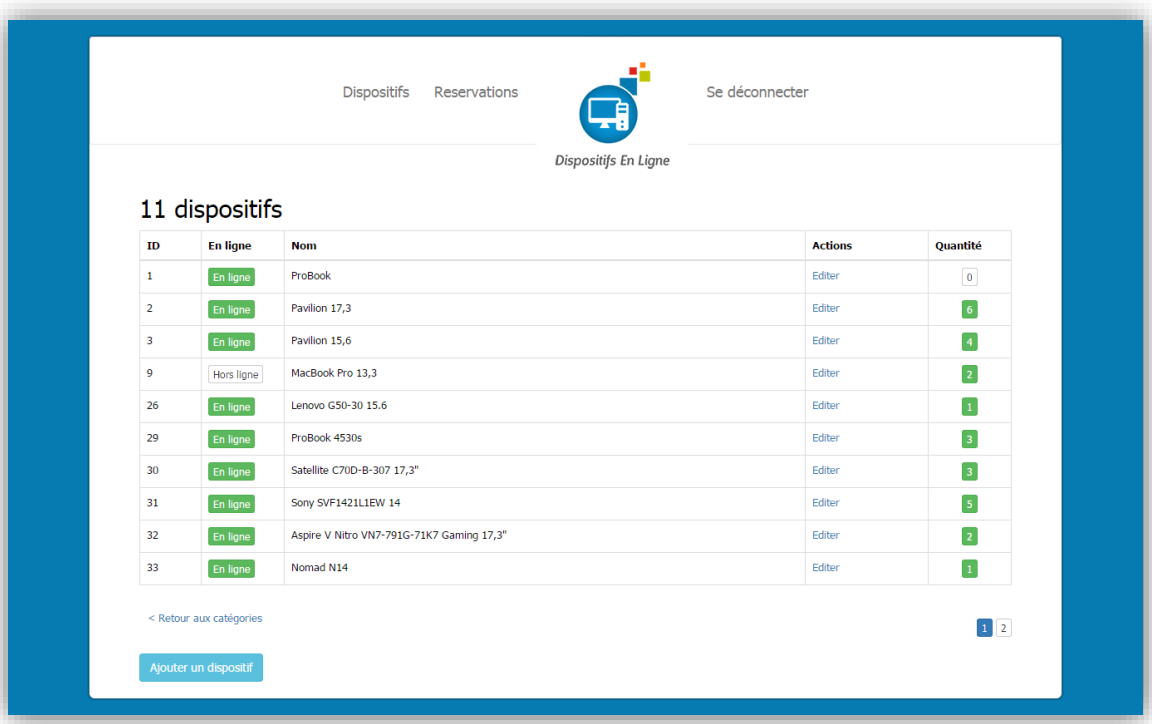

Figure III-9: Liste des dispositifs de la catégorie ordinateur

Edition d'un dispositif

L'administrateur à la possibilité d'ajouter un nouveau dispositif ou de modifier un dispositif existant. Il peut aussi lui ajouter plusieurs photos.

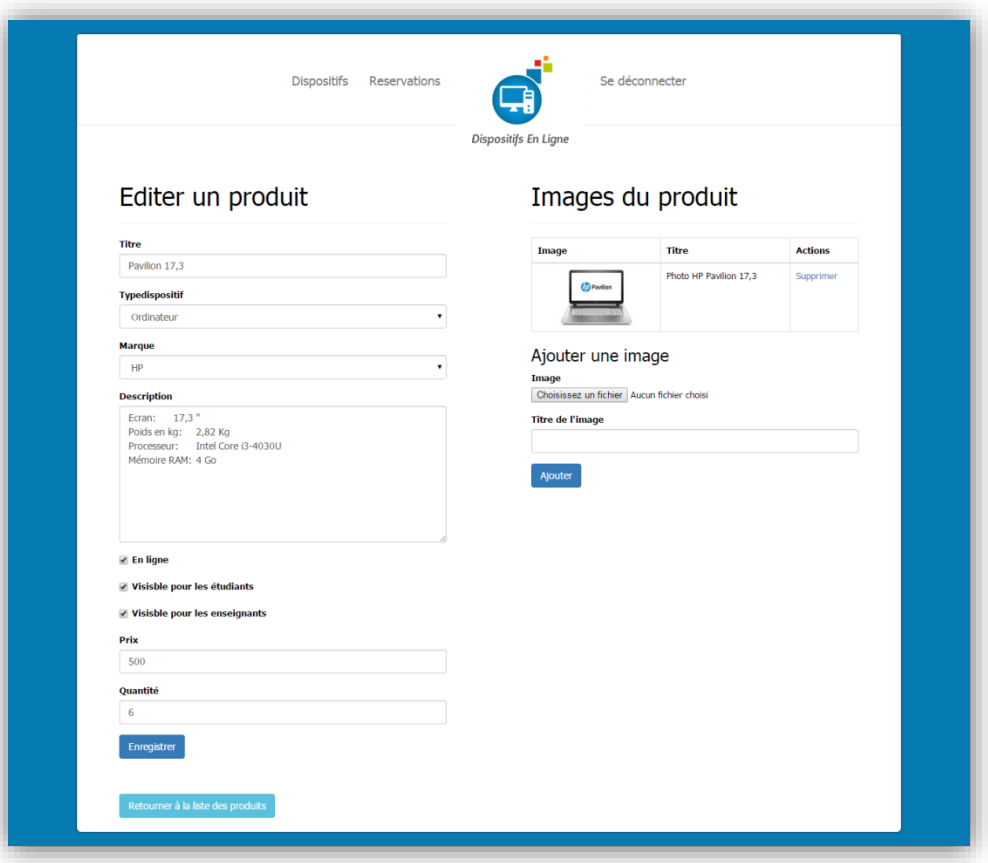

Figure III-10: Edition d'un dispositif

Les réservations

L'administrateur peut visualiser la liste des dispositifs qui sont réservés et voir quel utilisateur a réservé quel dispositif.

Il peut aussi supprimer des réservations.

Figure III-11: Les réservations

Aperçu des dispositifs dans le site web

L'administrateur à la possibilité d'accéder au site web pour voir la liste des dispositifs.

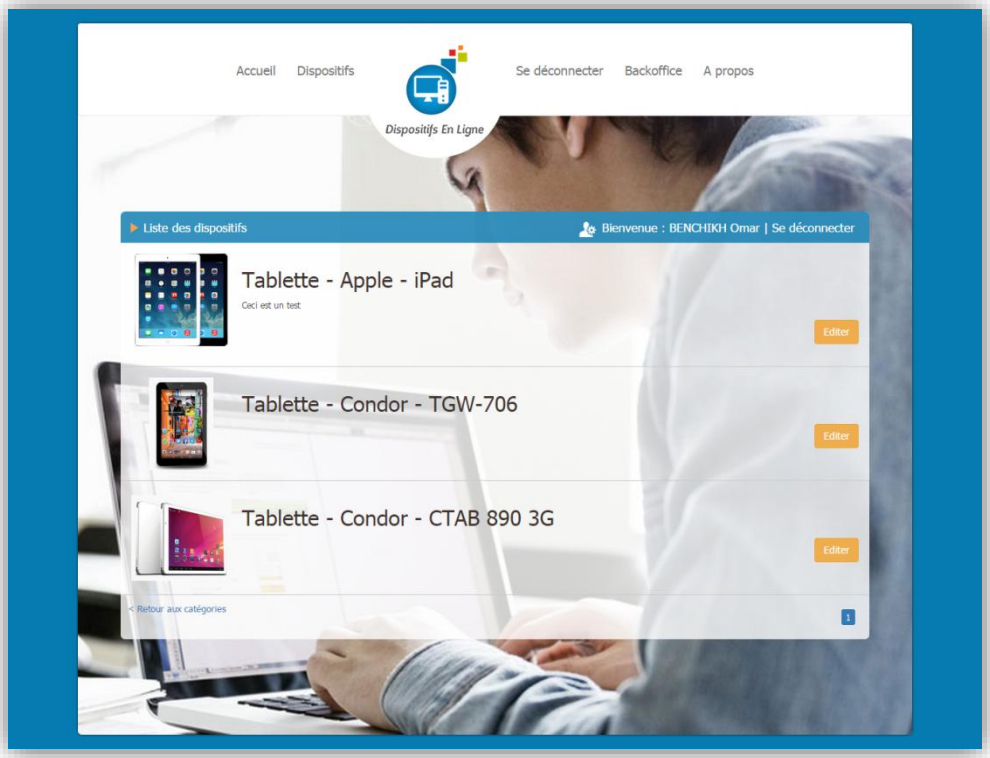

Figure III-12: Aperçu des dispositifs dans le site web

.V Conclusion

L'application « Dispositifs en ligne » a été réalisée dans le but de montrer l'efficacité d'une mise en place d'un système de sécurité tel que le chiffrement RSA avec un système d'authentification et de gestion des rôles dans une application web développée avec PHP selon le design pattern MVC.

Mettre en place un modèle MVC personnalisé nous a permis de comprendre tout l'intérêt d'utiliser une telle architecture dans une application web. Cela semble difficile au départ et ça prend énormément de temps de programmer les bases de son système MVC mais il est beaucoup plus facile par la suite d'ajouter des fonctionnalités et de mettre à jour son application.

Tout au long de la réalisation de l'application « Dispositifs en ligne », il a été intéressant de vérifier et de s'assurer à chaque étape de la sécurité des éléments programmés et leur fonctionnement. Cela nous a permis de réaliser une application sécurisée, fiable et évolutive.

Conclusion générale

Afin de proposer une solution qui répond au besoin des universitaires et leurs enseignants en matériel électronique ; il a été intéressant de pouvoir mettre en œuvre quelques outils informatiques permettant de concevoir des applications web qui intègrent à la fois des fonctionnalités diverses mais aussi des mécanismes de sécurité qui ont permis de proposer du contenu fiable et une utilisation sécurisée.

La conception que nous avons proposée se base sur une architecture MVC, une méthode appréciée des professionnels et des passionnés de la programmation. Ce modèle nous a permis de réaliser une application flexible, facile à éditer et à faire évoluer. Nous avons réussi à intégrer plusieurs fonctionnalités en utilisant la programmation orienté objet avec PHP5. Aussi pour garantir l'intégrité et la fiabilité de l'application nous avons mis en place plusieurs mécanismes de sécurité : l'authentification pour protéger le contenu, la gestion de rôles pour maitriser l'accès à certaines fonctionnalités et plus important encore le cryptage des données sensibles avec RSA ; un algorithme largement utilisé à travers le monde par les entreprises et les banques.

L'application « Dispositifs en ligne » met en lumière l'importance de la sécurité dans les applications et a permis de voir grand et plus loin. En effet il serait judicieux de pouvoir proposer plus de contenus dans l'application. On pourrait imaginer que l'université proposerait d'autres dispositifs qui concerneraient un plus large publique comme les étudiants en chimie ou en génie biomédical. Elargir et proposer plus de services en allant plus loin avec un système d'offre de priorité pour les utilisateurs les plus fidèles ou les plus sérieux. Ajouter un autre profil « Super administrateur » qui s'occuperait de gérer les utilisateurs de l'application…etc. Les idées sont nombreuses et très variées, l'important c'est de ne pas oublier que plus l'application évoluera plus il faudra renforcer la sécurité en adoptant d'avantage de méthodes et de solutions pour se protéger des attaques à différents niveaux.

Bibliographie

[1] Guillaume Harry, Failles de sécurité des applications Web Principes, parades et bonnes pratiques de développement, hal-00736013, 2012.

[2] Guillaume Harry, «Direction des Systèmes d'Information Pôle ARESU CNRS,» [En ligne]. Available: https://aresu.dsi.cnrs.fr/spip.php?article134. [Accès le 10 Novembre 2014].

[3] OWASP, «OWASP Top 10-2013 Les Dix Risques de Sécurité Applicatifs Web les Plus Critiques,» *OWASP The Open Web Application Security Project*, p. 22, 2013.

[4] Guillaume Affringue, «Chiffrement et hash en PHP contre l'attaque Man in the middle,» 25 Janvier 2007. [En ligne]. Available: http://guillaume-affringue.developpez.com/securite/chiffrement/?page=5. [Accès le 11 Février 2015].

[5] The PHP Group, «Hashage de mots de passe sûr,» [En ligne]. Available: http://php.net/manual/fr/faq.passwords.php. [Accès le 30 Mai 2015].

[6] LO1c et Django20, «L'algorithme RSA,» [En ligne]. Available: http://openclassrooms.com/courses/l-algorithme-rsa. [Accès le 30 Mai 2015].

[7] Vayel & Dominus Carnufex, «Le cryptosysteme RSA entre theorie et pratique,» 29 Mars 2015. [En ligne]. Available: http://zestedesavoir.com/tutoriels/663/la-cryptographie-asymetrique-avec-rsa/915/le-cryptosysteme-rsa-entre-theorie-et-pratique/3814/un-peu-dhistoire-et-beaucoup-de-mathematiques/. [Accès le 28 Avril 2015].

[8] Simon Chabot, «La cryptographie asymétrique : RSA,» 18 Septembre 2010. [En ligne]. Available: http://openclassrooms.com/courses/la-cryptographie-asymetrique-rsa. [Accès le 22 Mai 2015].

[9] Luke Welling & Laura Thomson, PHP et MySQL, PEARSON, 2009.

[10] The PHP Group, «Qu'est ce que PHP,» [En ligne]. Available: http://php.net/manual/fr/intro-whatis.php. [Accès le 17 Novembre 2014].

[11] Eric Daspet & Cyril Pierre de Geyer, PHP 5 avancé, EYROLLES, 2004.

[12] Mathieu Nebra, «Introduction à PHP,» [En ligne]. Available: http://openclassrooms.com/courses/concevez-votre-site-web-avec-php-et-mysql/introduction-a-php. [Accès le 29 Octobre 2014].

[13] Mathieu Nebra, «Lire des données,» [En ligne]. Available: http://openclassrooms.com/courses/concevez-votre-site-web-avec-php-et-mysql/lire-des-donnees-2. [Accès le 20 Novembre 2014].

[14] Richar Grin, Langane SQL, Université de Nice Sophia-Antipolis, 1998.

[15] Mathieu Nebra, «Organiser son code selon l'architecture MVC,» [En ligne]. Available: http://openclassrooms.com/courses/concevez-votre-site-web-avec-php-et-mysql/organiser-son-code-selon-l-architecture-mvc. [Accès le 04 Novembre 2014].

[16] Serge Tahé, «Méthodologie de développement MVC d'une application WEB/PHP,» 2008. [En ligne]. Available: http://tahe.developpez.com/web/php/mvc/. [Accès le 15 Novembre 2014].

Annexe A

Déroulement du chiffrement et du déchiffrement du mot de passe

Côté client :

Le formulaire en HTML :

```
<form name="theForm" action="/dispositifs_en_ligne/Connexion" method="post">
…
<label>Mot de passe</label><input type="password" name="password" value=""
onkeyup="hash();">
<input type="text" id="hashed" name="hashed" size="67" style="display:none;" READONLY/>
<TEXTAREA id="crypted" name="crypted" ROWS="4" COLS="65"
style="display:none;"></TEXTAREA>
…
</form>
```

Les fonctions javascript :

```
function hash(){
   var theForm = window.document.theForm;
   var cleartext = theForm.password.value;
   theForm.hashed.value = pidCrypt.SHA1(cleartext);
}
```

```
function compute(mode){
 var theForm = window.document.theForm;
 var input = theForm.hashed.value;
 var crypted = theForm.crypted.value;
 var public_key = '<?php echo $soft_publickey;?>';
 var params = {};
 switch(mode){
  case 'encrypt':
   params = certParser(public_key);
   if(params.b64){
    var key = pidCryptUtil.decodeBase64(params.b64);
    //new RSA instance
    var rsa = new pidCrypt.RSA();
    //RSA encryption
    //ASN1 parsing
    var asn = pidCrypt.ASN1.decode(pidCryptUtil.toByteArray(key));
    var tree = asn.toHexTree();
    //setting the public key for encryption
    rsa.setPublicKeyFromASN(tree);
    crypted = rsa.encrypt(input);

    theForm.crypted.value =
pidCryptUtil.fragment(pidCryptUtil.encodeBase64(pidCryptUtil.convertFromHex(crypted)),64);
```

```
      //vider les champs
      theForm.password.value = '';
      theForm.hashed.value = '';
     } else alert('Le formulaire ne fonctionne pas correctement! Veuillez réessayer plus tard.');
    break;  }}
```

La Librairie pidCrypt crypto javascript est téléchargeable à l'adresse : https://www.pidder.de/pidcrypt/?page=rsa. Dernier accès le 06/06/2015. La version utilisée dans cette application est la version 006 du 15/01/2015.

Côté serveur :

Dans la méthode login du contrôleur UsersController :

```
//je récupère la valeur cryptée
$data->password = $data->crypted;
unset($data->crypted);
//appel des fichier RSA
set_include_path('phpseclib');
require_once 'Net/SSH2.php';
require_once 'Crypt/RSA.php';
require_once 'class/myRSA.php';
$rsa_obj = new Crypt_myRSA();
//Decryptage du mot de passe correspondant en BDD pour comparaison
$decrypt_bdd = $rsa_obj->decrypt($user->password);
//Decryptage du mot de passe récupéré de la form pour comparaison
$decrypt_form = $rsa_obj->decrypt($data->password);
```

La class Crypt_myRSA qui nous permet de comprendre le cryptage javascript récupéré du formulaire.

```
class Crypt_myRSA extends Crypt_RSA{
   function decrypt($strBase64CipherText)    {
      $rsa=new Crypt_RSA();
      // je charge directement ma clé privé que je ramène de Conf
      $rsa->loadKey(Conf::$soft_privatekey);
      $binaryCiphertext=base64_decode($strBase64CipherText);
      $rsa->setEncryptionMode(CRYPT_RSA_ENCRYPTION_PKCS1);
      $strBase64DecryptedData=$rsa->decrypt($binaryCiphertext);
      return base64_decode($strBase64DecryptedData);    }}
```

La Librairie phpseclib est téléchargeable à l'adresse : http://phpseclib.sourceforge.net/ . Dernier accès le 06/06/2015. La version utilisée dans cette application est la version 0.3.10

Liste des figures

Liste des tableaux