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: *Réseau et Système et Distribué(R.S.D)*

Thème

Sécurisation d'une application web en intégrant Struts 2 et Hibernate

Réalisé par :

- **Saidi Anas Nawfel**

Présenté le 6 juillet 2019 devant le jury composé de :

- *Mr Lehsaini Mohamed* *(Président)*
- *Mr Benmammer Badr* *(Encadreur)*
- *Mr Belhocine Amin* *(Examinateur)*

Année universitaire: 2018-2019

Remerciements

En préambule à ce mémoire nous remercions Allah qui nous a dotés d'une grande volonté qui nous a aidés et nous a donné le courage, la force et la Patience d'accomplir ce travail.

Je tiens tout d'abord à remercier mes parents pour m'avoir encouragé et permise de bien évoluer tout au long de mon cursus universitaire.

Je tiens à remercier mes frères et sœur pour leurs encouragements et conseils qui ont contribué à l'accomplissement de ce travail.

Je tiens également à remercier mon encadrant Monsieur BEMAMMAR B. pour temps qu'il m'a consacré ainsi que ses encouragements. Merci de m'avoir accompagnés tout au long de ce projet.

Je voudrais, aussi, remercier très sincèrement Monsieur *Lehsaini Mohamed* pour l'honneur qu'il me fait en acceptant la présidence de ce jury.

Que l'honorable membre du jury, Monsieur *Belhocine Amin* veuille croire en mes remerciements anticipés pour avoir bien voulu accepter d'enrichir et de faire évoluer ce travail.

J'exprime aussi ma gratitude envers tous ceux qui m'ont accordé leur soutien, tant par leur gentillesse que par leur dévouement.

Table de matières

Liste des figures

Liste des abréviations

AES: Advanced Encryption Standard.

API: Application Programming Interface.

CSS: Cascading Style Sheets.

HTML: HyperText Markup Language.

IMAP: Internet Message Access Protocol.

IPSec: Internet Protocol Security.

J2EE: Java 2 Enterprise Edition.

Java SE: Java Standard Edition.

JDBC: Java Data Base Connectivity.

JMF: Java Media Framework.

JVM: Java Virtual Machine

JPA: Java Persistence API.

JSP: JavaServer Pages.

MVC : Modèle-Vue-Contrôleur.

OWASP: Open Web Application Security Project.

POP3: Post Office Protocol.

PHP: Hypertext Preprocessor.

SGBD : Système de Gestion de Base de Données.

SMTP: Simple Mail Transfer Protocol.

SQL: Structured Query Language.

SSH: Secure Shell.

WASC: Western Association of Schools and Colleges.

WPA2: Wi-Fi Protected Access 2.

XSS: Cross-site Scripting.

LDAP: Lightweight Directory Access Protocol.

Introduction générale

À l'époque actuelle les applications web ont une immense influence sur la société, rares sont les personnes qui ne possèdent pas de comptes dans les réseaux sociaux, ou qu'ils n'ont jamais consulté des blogs, effectuer des achats ou vendre des produits sur des sites e-commerce.

L'utilisation de ces applications a permis l'évolution de notre mode de vie, car elle rend plus simple notre quotidien. En effet, beaucoup de tâches sont devenues facile à faire sans perte de temps.

Les applications web sont devenues accessibles partout et pour tout le monde, par des tablettes ou des Smartphones. Cette montée en puissance de l'internet dans la société a fait de l'application web une cible aux attaques par des personnes ou des logiciels malveillants pour des différentes raisons. Une chose qui a motivé les entreprises à fortifier leurs sécurités pour minimiser les sources de vulnérabilités.

L'objectif principal de la sécurisation est de permettre à l'application de fonctionner correctement. La sécurité d'une application web commence par la sensibilisation à travers l'estimation du risque qu'elle peut subir. Mal conçues, elle permet aux attaquants d'accéder à des informations confidentielles, voire de pénétrer dans le système informatique de l'entreprise. Pour garantir une sécurité idéale cela passe par une étude des besoins en confidentialité et en disponibilité, puis par une évaluation des impacts en cas d'incident, ensuite offrir des solutions afin de s'en protéger contre les attaques les plus courants.

Dans ce PFE, nous avons modélisé, implémenté et sécurisé une application web. La modélisation de l'application est basée sur le design pattern MVC, son implémentation utilise les deux Framworks Hibernate et Struts. Compte à sa sécurité, elle est basée sur trois points essentiels qui sont : l'authentification (des clients et de l'administrateur), la gestion des rôles (la définition de trois rôles différents : Internaute, Administrateur et Client) et le chiffrement des données sensibles (utilisation de l'algorithme AES).

Le reste de ce rapport est organisé comme suit :

- Le premier chapitre se focalise sur l'importance de la sécurité des applications web, par introduire les risques que peut subir une application à travers les attaques qui exploitent les failles. Ainsi montrer les règles et les bonnes pratiques pour maximiser la sécurité.
- Le deuxième chapitre (second c'est juste quand il y a le 1 et le 2, on peut dire premier et second) est consacré à la présentation des outils utilisés pour la conception et la réalisation de notre application.
- L'implémentation de l'application est présentée dans le dernier chapitre.

Le rapport ce termine par une conclusion générale et des perspectives.

Chapitre I: Synthèse sur la sécurité des applications web

I.1 Introduction

De nos jours, le Web est devenu omniprésent, en effet, toute entreprise ou administration se doit maintenant d'avoir une présence sur le Web. Parmi les services les plus répandus dans ce contexte, on peut citer le cas des sites E-commerce. Pour proposer un service sur le web il faut tout d'abord garantir la sécurité, la fiabilité ainsi que l'intégrité des données de l'application.

Dans ce chapitre, nous allons présenter la sécurité, les problèmes ainsi que la méthodologie à suivre afin d'améliorer la sécurité dans le cadre de développement des applications web.

I.2 Enjeu de la sécurité

Il existe plusieurs associations qui ont pour but de prévenir et d'informer les organisations et les personnes sur les problèmes liés à la sécurité des applications web, parmi ces associations nous pouvons citer:

- **L'Open Web Application Security (OWASP)** :

Un organisme à but non lucratif mondial qui milite pour l'amélioration de la sécurité des logiciels. L'OWASP propose un guide de développement pour les applications web dans lequel se trouve les bonnes pratiques à adopter lors de la phase de développement d'un projet web.[1]

- **Le Web Application Security Consortium (WASC):**

Est une association qui publie le rapport <<WASC Threat Classification>> qui décrit et classe les menaces de sécurité sur les applications Web avec des solutions et des bonnes pratiques grâce à des experts internationaux.

I.2.1 Objectifs des attaques

Les attaques ont toujours lieu sur Internet, à raison de plusieurs attaques par minute sur chaque machine connectée, chaque attaquant a un but pour lancer ces attaques, parmi ces objectifs :

-Obtention de privilèges supplémentaires.

-Prise de contrôle du système.

-Vol d'informations confidentielles.

-Corruption des données.

- Instabilité de l'application ou du serveur.

-Accès à des fichiers du serveur http.

-Déni de service.

I.2.2 Failles de la sécurité

Chaque trois ans, l'OWASP publie un document sur les 10 failles de sécurité les plus périlleuses. L'OWASP a établit en 2017 le classement suivant:

- **Injection :**

Des failles de rejet, telles que l'injection SQL, NoSQL et LDAP, se produisent lorsque des données non fiables sont envoyées à un interpréteur dans le cadre d'une commande ou d'une requête. Les données hostiles de l'attaquant peuvent inciter l'interprète à exécuter des commandes inattendues ou à accéder à des données sans autorisation appropriée [2].

- **Broken authentification (gestion incorrecte de l'authentification ou du suivi des sessions :**

Les fonctions d'application liées à l'authentification et à la gestion de session sont souvent implémentées de manière incorrecte, permettant aux attaquants de compromettre les mots de passe, les clés ou les jetons de session, ou d'exploiter d'autres failles d'implémentation afin de prendre l'identité de l'utilisateur temporairement ou définitivement [2].

- **Exposition de données sensibles :**

De nombreuses applications Web et API ne protègent pas correctement les données sensibles, telles que les données financières, de santé et les données personnelles. Les attaquants peuvent voler ou modifier ces données faiblement protégées pour commettre une fraude par carte de crédit, un vol d'identité ou d'autres infractions. Les données sensibles peuvent être compromises sans protection supplémentaire, telle que le cryptage à l'arrêt ou en transit, et nécessitent des précautions particulières lorsqu'elles sont échangées avec le navigateur [2].

- **XML External Entities (XXE) :**

De nombreux processeurs XML anciens ou mal configurés évaluent les références d'entités externes au sein de documents XML. Les entités externes peuvent être utilisées pour divulguer des fichiers internes à l'aide du gestionnaire de fichiers, des partages de fichiers internes, de l'analyse du port interne, de l'exécution de code à distance et des attaques par déni de service [2].

- **Broken Access Control :**

Les restrictions sur ce que les utilisateurs authentifiés sont autorisés à faire ne sont souvent pas correctement appliquées. Les pirates peuvent exploiter ces vulnérabilités pour accéder à des fonctionnalités et / ou à des données non autorisées, telles qu'accéder aux comptes d'autres utilisateurs, afficher des fichiers sensibles, modifier les données d'autres utilisateurs, modifier les droits d'accès, etc. [2].

- **Mauvaise configuration de sécurité :**

La mauvaise configuration de la sécurité est le problème le plus fréquemment rencontré. Cela est généralement dû à des configurations par défaut non sécurisées, à des configurations

incomplètes ou ad hoc, à un stockage en nuage ouvert, à des en-têtes HTTP mal configurés et à des messages d'erreur détaillés contenant des informations sensibles. Tous les systèmes d'exploitation, infrastructures, bibliothèques et applications doivent non seulement être configurés de manière sécurisée, mais ils doivent également être corrigés / mis à niveau rapidement [2].

- **Cross-Site Scripting (XSS) :**

Des failles XSS se produisent chaque fois qu'une application inclut des données non fiables dans une nouvelle page Web sans validation ou échappement correct, ou met à jour une page Web existante avec des données fournies par l'utilisateur à l'aide d'une API de navigateur pouvant créer du HTML ou du JavaScript. XSS permet aux attaquants d'exécuter des scripts dans le navigateur de la victime, ce qui peut détourner des sessions utilisateur, altérer des sites Web ou rediriger l'utilisateur vers des sites malveillants [2].

Le schéma suivant montre le déroulement d'une attaque XSS:

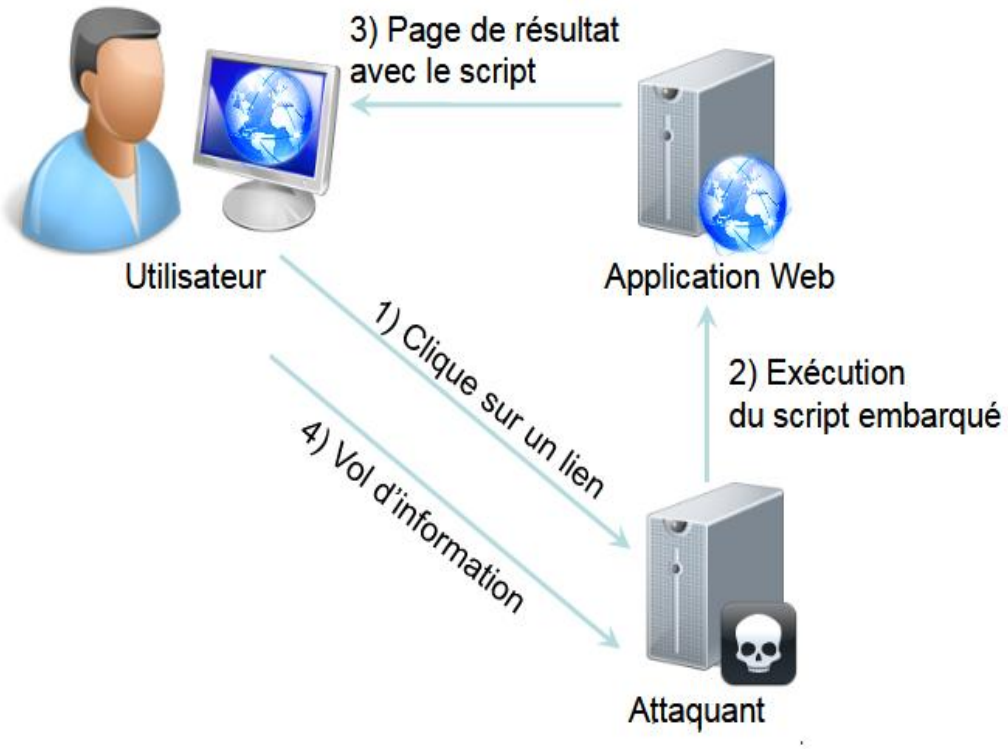


Figure I.1: Cross-Site Scripting (XSS) [6]

- **Références directes non sécurisées à un objet :**

Afin d'éviter les références directes non sécurisées, il convient de suivre une approche permettant de protéger chaque objet mis à disposition des utilisateurs (ex : nom de fichier, etc.) [2].

- **Utilisation de composants avec des vulnérabilités connues :**

Les composants, tels que les bibliothèques, les Frameworks et d'autres modules logiciels, s'exécutent avec les mêmes privilèges que l'application. Si un composant vulnérable est exploité, une telle attaque peut faciliter la perte de données ou la prise de contrôle du serveur. Les applications et les API utilisant des composants avec des vulnérabilités connues peuvent saper les défenses des applications et permettre diverses attaques et impacts [2].

- **Enregistrement et surveillance insuffisants :**

Une surveillance insuffisante, associée à une intégration manquante ou inefficace avec la réponse à un incident, permet aux attaquants d'attaquer les systèmes, de maintenir la persistance, de faire pivoter un plus grand nombre de systèmes et de falsifier, extraire ou détruire des données. La plupart des études sur les brèches montrent que le temps nécessaire pour détecter une brèche est supérieur à 200 jours, généralement détecté par des parties externes plutôt que par des processus internes ou une surveillance [2].

I.3 Règles de sécurité

Il est fortement recommandé de respecter quelques règles de développement pour fortifier le coté sécurité de l'application :

- **Formation et sensibilisation**

Les failles dans les applications web sont dues à un manque de respect des bonnes pratiques par les acteurs lors d'une étape du cycle de développement, qu'il s'agisse de la conception, de l'implémentation ou de l'intégration. Tous les membres d'une équipe projet doivent donc être sensibilisés aux enjeux et risques de sécurité, et formés aux mécanismes de sécurité de base [8].

Tous les acteurs sont importants : la maîtrise d'ouvrage doit être en mesure d'identifier les enjeux de sécurité pour exprimer les besoins. La maîtrise d'œuvre, les développeurs doivent être formés afin de pouvoir mettre en place des mécanismes de sécurité appropriés et efficaces. Cette démarche de sensibilisation et de formation doit couvrir les risques et mécanismes de sécurité spécifiques aux applications et technologies Web. Les mécanismes de sécurité sont en constante évolution et de nouveaux types de vulnérabilités sont découverts chaque année. Il est donc important pour les parties prenantes de suivre régulièrement des formations et d'effectuer une veille sécurité afin de se garder informées des nouvelles techniques d'intrusion et de piratage [8].

- **Minimiser les sources de vulnérabilité**

La pluparts des attaquants exploitent des vulnérabilités dans l'application et profitent parfois de certaines fautes simples. Voici quelques Bonnes pratiques à respecter [6] :

- Mots de passe forts.
- Verrouillage de compte après un nombre d'erreurs consécutives.
- Filtrer les entrées.

- Ne pas autoriser les caractères spéciaux.
- Un système pour réinitialiser le mot de passe (envoi d'un nouveau mot de passe).
- Redemander le mot de passe lors d'un changement de niveau de privilège.
- Rendre les cookies utilisables uniquement par l'application (le code JavaScript ne peut pas accéder au cookie).
- Utilisation de captcha.
- Utiliser un moyen d'identification secondaire comme une solution pour Détournement de session.

- **Les décisions de sécurité seront basées sur le risque**

Les décisions à propos de la sécurité seront prises conjointement avec le Client et le Développeur, selon une compréhension arrêtée des risques impliqués [3].

- **Les vulnérabilités sont attendues**

Tous les logiciels comportent des anomalies et certaines de celles-ci créeront des problèmes de sécurité. Le Client et le Développeur s'efforceront d'identifier les vulnérabilités aussi tôt que possible dans le cycle de vie [3].

- **Validation et codage**

Les exigences doivent préciser les règles pour canoniser, valider et coder chaque entrée à l'application, que ce soit des utilisateurs, des systèmes de fichiers, des bases de données, des répertoires ou des systèmes externes. La règle par défaut doit être que toutes les entrées sont invalides, à moins qu'elles ne correspondent à une spécification détaillée de ce qui est permis. De plus, les exigences doivent préciser l'action à prendre, lorsqu'une entrée invalide est reçue. Précisément, l'application ne doit pas être susceptible aux injections, aux débordements, aux violations, ou d'autres attaques d'entrée corrompue [3].

- **Authentification et gestion de session**

Les exigences doivent préciser comment les authentifiants et les identifiants de session seront protégés à travers leur cycle de vie. Les exigences pour toutes les fonctions reliées, y compris les mots de passe oubliés, les mots de passe changeants, le rappel des mots de passe, la déconnexion et les connexions multiples doivent être incluses [3].

- **Contrôle d'accès**

Les exigences doivent inclure une description détaillée de tous les rôles (groupes, privilèges, autorisations) utilisés dans l'application. Les exigences doivent également inclure tous les biens et fonctions fournis par l'application. Les exigences doivent complètement préciser les droits d'accès exacts de chaque bien et fonction pour chaque rôle. Une matrice de contrôle d'accès est le format suggéré pour ces règles [3].

- **Connexions aux systèmes externes**

Les exigences doivent préciser comment l'authentification et le chiffrement seront gérés pour tous les systèmes externes, comme les bases de données, les répertoires et les services Web. Tous les authentifiants nécessaires pour la communication avec les systèmes externes seront stockés à l'extérieur du code dans un fichier de configuration, sous forme chiffrée [3].

- **L'utilisation des requêtes Post**

L'objectif est que les données du formulaire n'apparaissent pas dans l'URL.

- **Vérifier tous les données avant utilisation**

Il s'agit de :

- Type attendu.
- Nombre d'arguments attendus.
- Valeurs limitées.
- Taille de la donnée.
- Valeur nulle autorisée?
- Valeur suit une expression régulière.
- Type attendu.

- **Chiffrement des données**

Les exigences devront préciser quelles données doivent être chiffrées, comment elles doivent être chiffrées et comment tous les certificats et autres authentifiants doivent être gérés. L'application devra utiliser un algorithme standard implanté dans une bibliothèque de chiffrement largement utilisée et testée [3].

I.4 Méthodes de cryptage

Dans la cryptologie moderne, une distinction fondamentale est faite entre les méthodes de cryptage symétrique et asymétrique. La classification est basée sur la manipulation des clés. [4]

Pour les **crypto systèmes symétriques,** la même clé est utilisée pour le chiffrement et le déchiffrement des données chiffrées. Si deux parties communicantes veulent échanger des données chiffrées, il faut trouver un moyen de transmettre secrètement la clé partagée. Dans les **crypto systèmes asymétriques,** par contre, chaque partenaire de communication génère sa propre paire de clés : une clé publique et une clé privée [4].

Si des méthodes de cryptage symétrique et asymétrique sont utilisées en combinaison, on les appelle des **méthodes hybrides** [4].

I.4.1 Méthodes de cryptage asymétrique

Alors que la même clé est utilisée pour les crypto systèmes symétriques des deux côtés de la communication cryptée, les deux parties d'une communication cryptée asymétrique génèrent

une paire de clés par côté. Chaque abonné de communication dispose ainsi de **deux clés** : une clé publique et une clé privée. Pour communiquer sous forme chiffrée, chaque partie doit divulguer sa clé publique. Ceci se fait généralement via des **serveurs clés**. C'est ce qu'on appelle une **procédure à clé publique**. La clé privée, en revanche, reste secrète. Ceci montre la force des crypto systèmes asymétriques : contrairement au chiffrement symétrique, la clé secrète n'est jamais donnée hors de contrôle à aucun moment [4].

Dans le contexte du chiffrement asymétrique, les clés publiques sont utilisées pour le chiffrement. Ils permettent également de vérifier les signatures numériques et les utilisateurs. Les clés privées, quant à elles, sont utilisées pour le décryptage et permettent de générer des signatures numériques ou de s'authentifier contre d'autres utilisateurs [4].

L'avantage du cryptage asymétrique est que tout le monde peut utiliser la clé publique de l'utilisateur B pour crypter les messages, mais que seul B peut les décrypter avec sa clé secrète. Étant donné que seule la clé publique est échangée, il n'est pas nécessaire d'utiliser un canal protégé contre les effractions [4].

L'un des **inconvénients** de cette méthode de cryptage, cependant, est que B ne peut pas être sûr que le message crypté provient bien de A. En principe, un tiers (C) pourrait également utiliser la clé publique de B pour crypter les messages, par exemple, pour propager des logiciels malveillants. De plus, A ne peut pas être sûr qu'une clé publique est bien la clé de B. C pourrait également créer une clé publique et l'utiliser comme la clé de B pour intercepter les messages de A à B. Dans le cadre du cryptage asymétrique, un mécanisme est donc nécessaire pour qu'un partenaire de communication puisse vérifier l'authenticité de l'autre [4].

I.4.2 Méthodes de cryptage symétrique

Les méthodes de cryptage symétrique classiques fonctionnent au niveau de la lettre pour convertir du texte brut en textes secrets, le cryptage se fait au niveau du bit dans les systèmes de cryptage assistés par ordinateur. On distingue le chiffrement de puissance du chiffrement de bloc :

- Chiffrement de puissance (flux) : chaque caractère ou bit du texte en clair est lié à un caractère ou bit du flux de clés, et ainsi traduit en un caractère de sortie chiffré [4].
- Chiffrement de bloc : les caractères ou bits à chiffrer sont combinés en blocs de longueur fixe et mappés sur un chiffrement de longueur fixe [4].

Dans ce qui suit, nous allons présenter l'algorithme AES.

Advanced Encryption Standard (AES) :

AES est un algorithme de chiffrement symétrique. L'algorithme a été développé par deux cryptographes belges, Joan Daemen et Vincent Rijmen.

AES a été conçu pour être efficace tant sur le plan matériel que logiciel. Il prend en charge une longueur de bloc de 128 bits et une longueur de clé de 128, 192 et 256 bits. [5].

Fonctionnement de l'algorithme AES :

AES divise également le texte brut à crypter en blocs. La norme prend en charge les clés 128, 192 et 256 bits, AES utilise des blocs beaucoup plus grands de 128 bits qui sont encodés en plusieurs cycles consécutifs, à l'aide d'un **réseau de permutation de substitution (SPN).** L'algorithme utilise également une nouvelle clé ronde pour chaque cycle de chiffrement, qui est dérivée récursivement de la clé initiale et liée au bloc de données à chiffrer en utilisant XOR. Le processus de cryptage peut être divisé en quatre étapes [4] :

1. **Expansion des clés** : AES utilise une nouvelle clé ronde dans chaque boucle de cryptage. Ceci est dérivé de la clé initiale par récursion. La clé initiale est étendue à une longueur qui vous permet de mapper le nombre requis de touches rondes de 128 bits. Chaque clé ronde est donc basée sur une sous-section de la clé initiale étendue. Le nombre de touches rondes nécessaires est le nombre de tours de chiffrement (R) y compris le tour final plus une touche ronde pour le tour préliminaire (nombre de touches rondes = R + 1) [4].

2. **Phase préliminaire :** lors de la ronde préliminaire, le bloc d'entrée 128 bits est transféré dans une table bidimensionnelle (tableau) et relié à la première clé ronde à l'aide de XOR (Key Addition). Le tableau contient 4 lignes et 4 colonnes. Chaque cellule contient donc un octet (8 bits) du bloc à crypter [4].

3. **Rondes de cryptage** : le nombre de rondes de chiffrement dépend de la longueur de clé utilisée : 10 rondes pour AES128, 12 rondes pour AES192 et 14 rondes pour AES256 :

   - **Sous-octets** : les sous-octets sont une substitution monoalphabétique. Chaque octet du bloc à crypter est remplacé par un équivalent en utilisant une S-Box (table de subtitution). [4]
   - **Les rangées de quarts** : dans le contexte de la transformation ShiftRow, les octets dans les cellules du réseau (voir le tour préliminaire) sont déplacés cycliquement vers la gauche. [4]
   - **MixColumns** : avec MixColumns, l'algorithme AES inclut une transformation dans laquelle les données sont fusionnées dans les colonnes du tableau. Cette étape est basée sur un nouveau calcul de chaque cellule individuelle. Pour cela, les colonnes de la matrice sont soumises à une multiplication matricielle et les résultats sont liés par XOR. [4]
   - **KeyAddition** : à la fin de chaque cycle de chiffrement, un autre Key Addition a lieu. Comme dans le tour préliminaire, il est basé sur un lien XOR entre le bloc de données et la touche ronde courante. [4]

4. **Final Round :** le dernier round est le dernier round de cryptage. Contrairement aux cycles précédents, il ne contient pas de transformations MixColumns et ne comprend donc que les opérations SubBytes, ShiftRows et KeyAddition. Le résultat du dernier tour est le texte secret. [4]

Le décryptage des données cryptées AES est basé sur l'investissement de l'algorithme de cryptage. En plus de la séquence d'étapes, ceci se réfère également aux opérations ShiftRow, MixColumns et SubBytes, dont la direction est également inversée. [4]

L'AES opère sur des blocs de 128 bits (plaintext P) qu'il transforme en blocs cryptés de 128 bits (C) par une séquence de Nr opérations ou "rounds", à partir d'une clé de 128, 192 ou 256 bits. Suivant la taille de celle-ci, le nombre de rounds diffère : respectivement 10, 12 et 14 rounds [7].

Le schéma suivant décrit succinctement le déroulement du chiffrement :

- BYTE_SUB (Byte Substitution) est une fonction non-linéaire opérant indépendamment sur chaque bloc à partir d'une table dite de substitution. [7]
- SHIFT_ROW est une fonction opérant des décalages (typiquement elle prend l'entrée en 4 morceaux de 4 octets et opère des décalages vers la gauche de 0, 1, 2 et 3 octets pour les morceaux 1, 2, 3 et 4 respectivement). [7]
- MIX_COL est une fonction qui transforme chaque octet d'entrée en une combinaison linéaire d'octets d'entrée et qui peut être exprimée mathématiquement par un produit matriciel sur le corps de Galois (2^8). [7]
- le + entouré d'un cercle désigne l'opération de OU exclusif (XOR). [7]
- Ki est la ième sous-clé calculée par un algorithme à partir de la clé principale K. [7]

Le déchiffrement consiste à appliquer les opérations inverses, dans l'ordre inverse et avec des sous-clés également dans l'ordre inverse. [7]

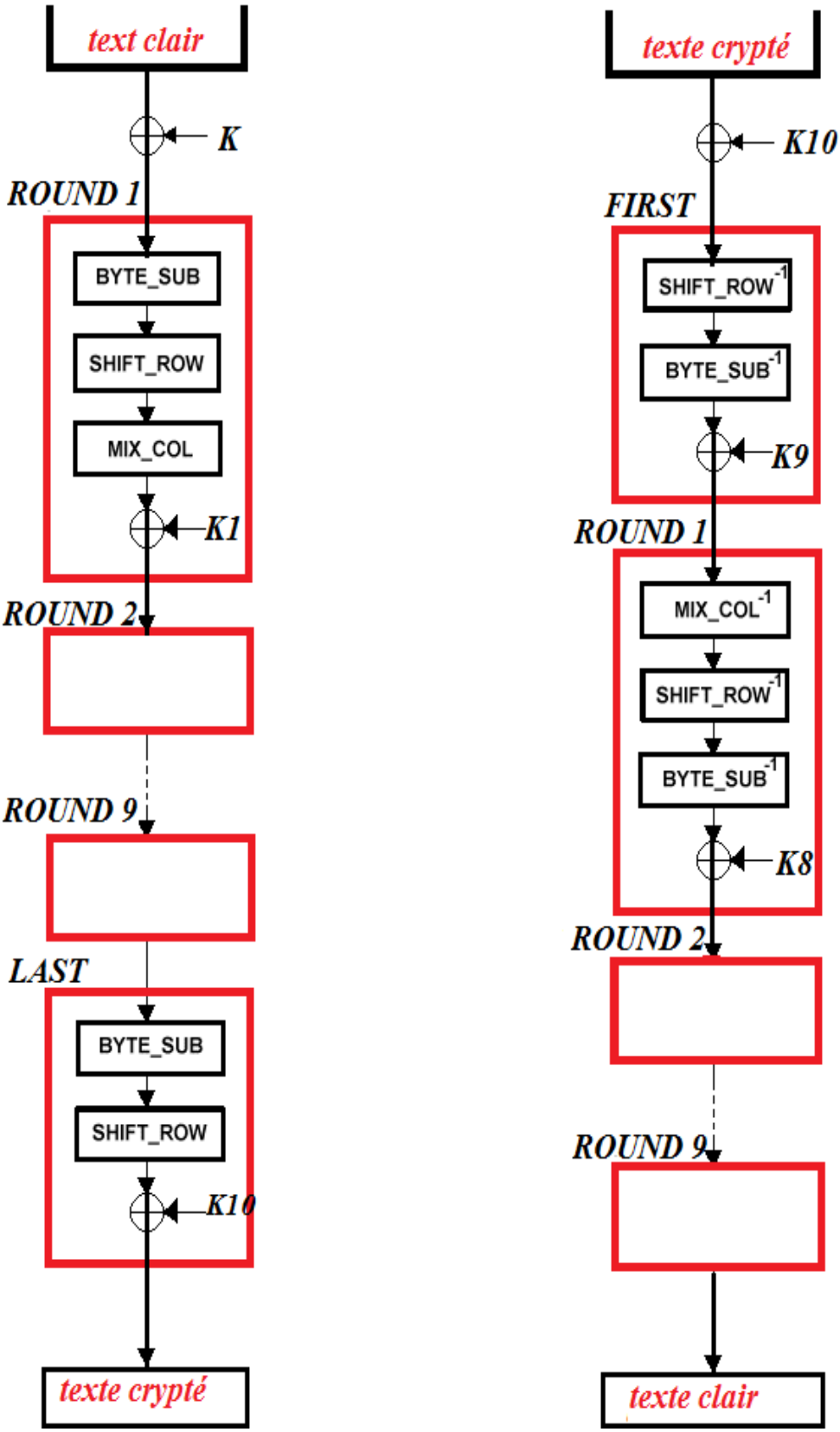


Figure I.2: Le chiffrement et le déchiffrement AES [7]

Avantage :

À ce jour, aucune attaque pratique n'est connue. Les attaques par force brute sont inefficaces en raison de la longueur de clé d'au moins 128 bits. De plus, des opérations telles que ShiftRows et MixColumns garantissent un mixage optimal des bits : dans le résultat, chaque bit dépend de la clé. De plus, le cryptosystème convainc par sa simplicité d'implémentation et sa grande vitesse. AES est utilisé comme norme de cryptage pour WPA2, SSH et IPSec ainsi que comme algorithme de cryptage pour les archives de fichiers compressés telles que 7-Zip ou RAR. [4]

AES résiste aussi aux attaques par cryptanalyse linéaire (LC) où l'attaquant utilise des approximations linéaires pour décrire les opérations conduisant au chiffré. Comme précédemment, plus le nombre d'essais augmente, plus la probabilité de la bonne clé devient forte. [7] ainsi les attaques par cryptanalyse différentielle (DC), l'attaquant choisit des textes clairs présentant une différence fixe, calcule les chiffrés (en ayant accès au système) et leurs différences puis assigne des probabilités à certains types de clés. [7]

En plus on trouve la facilité de calcul, cela entraine une grande rapidité de traitement, avec des besoins en ressources et mémoire très faibles. [7]

Inconvénient :

AES est limitée aux domaines d'application qui ne nécessitent pas d'échange de clés ou qui le permettent via un canal sécurisé. [4]

I.5 Vérification de la sécurité d'une application web

Plusieurs approches peuvent être utilisées pour vérifier la sécurité d'une application Web, chacune ayant ses avantages et ses inconvénients : [8]

I.5.1 Audit des spécifications

Cette approche consiste à considérer des scénarios de menaces et à évaluer comment les architectures techniques et mécanismes de sécurité prévus dans les spécifications sont à même de protéger l'application, ses données et ses traitements. Elle peut être réalisée dès la phase de conception, avant même la phase d'implémentation, indépendamment des technologies utilisées. Elle ne permet toutefois pas de se prémunir contre d'éventuels problèmes d'implémentation.

I.5.2 Audit de code

L'audit de code source consiste à analyser le code source de l'application (code Java, JSP, PHP, .Net, C/C++, etc.), afin de vérifier :

- Que les mécanismes de sécurité permettant de protéger l'application sont bien présents. [8]
- Que les règles de contrôle interne métier sont bien appliquées. [8]

- Que le code source ne contient pas de bugs pouvant permettre à un attaquant de contourner la sécurité. [8]

L'audit de code source a de nombreux avantages. Le fait d'analyser le code source peut en effet permettre :

- De vérifier le respect des bonnes pratiques et du principe de la défense en profondeur. [8]
- D'avoir un niveau d'assurance élevé quant à l'absence de failles. [8]
- D'identifier facilement ce qu'il faut faire pour corriger la faille. [8]

I.5.3 Test d'intrusion

Le test d'intrusion est une approche déjà utilisée dans d'autres environnements que les applications Web. Il consiste à confronter l'application Web à une situation d'attaque réelle, en simulant les actions d'une personne mal intentionnée. Le test peut être réalisé [8]

Sans connaissance et sans habilitations initiales, afin de simuler un attaquant externe (parfois appelé « tests en boite noire »). [8]

Avec les connaissances et habilitations d'un utilisateur de base, pour simuler les attaques d'un utilisateur légitime, mais malveillant (parfois appelé « tests en boite grise »). [8]

Avec les connaissances d'un développeur de l'application (mise à disposition de l'attaquant du code source) : (parfois appelé « tests en boite blanche »). [8]

I.6 Conclusion

Avec tous les progrès réalisés sur le web, la sécurité des applications web reste une des préoccupations majeures qui est difficile à assurer. Pour cela il est primordial d'intégrer la sécurité dés la phase de conception.

Dans le but de maximiser la sécurité d'une application web, il est nécessaire de rester à jour et de s'informer sur les failles les plus connues et les plus exploitées dans le web durant les années récentes. Par conséquent il faut s'adapter et s'améliorer par rapport aux nouveaux risques à travers les bonnes pratiques, avec un bon suivi des règles de développement et avoir un plan de sécurité en définissant tous les besoins de l'application en terme de sécurité.

Chapitre II: Outils utilisés et architecture de déploiement

II.1 Introduction

Notre objectif à travers ce PFE est de mette en œuvre une application Web avec une protection contre les failles de sécurité en se basant sur des mécanismes tel que le contrôle d'accès des utilisateurs, le chiffrement AES des mots de passe et la gestion des rôles.

A travers ce chapitre nous allons présenter les différents outils utilisés pour l'implémentation de notre application, en justifiant notre choix pour les différentes technologies utilisées. Une partie a été consacrée à l'explication du principe de fonctionnement des deux Frameworks Struts 2 et Hibernate que nous avons utilisé dans cette application. Afin d'assimiler le principe de fonctionnement du Framework Struts 2 qui se base sur le pattern de conception MVC, nous avons réservé une partie de ce chapitre à l'explication du principe de ce dernier. A la fin, le chapitre se termine avec les architectures de déploiement supportées par notre application.

II.2 Outils utilisés

Afin de bien mener la mise en œuvre de notre application, nous avons choisi d'utiliser les deux Framework Struts 2 et Hibernate, en se basant sur un langage puissant dans la programmation des applications à savoir le Java.

Ces Frameworks présentent une grande puissance dans le développement des applications web.

II.2.1 Langages utilisés

II.2.1.1 Java

Java est un langage de programmation moderne développé par Sun Microsystems (aujourd'hui racheté par Oracle). [13]

Une de ses plus grandes forces est son excellente portabilité : une fois votre programme créé, il fonctionnera automatiquement sous Windows, Mac, Linux, etc. [13]

Java permet de réaliser une très grande quantité d'applications différentes, on peut faire de nombreuses sortes de programmes avec Java. [13]

- Des applets, qui sont des programmes Java incorporés à des pages web. [13]
- Des applications, sous forme de fenêtre ou de console. [13]
- et bien d'autres ! J2EE, JMF. [13]

Dans ce qui suit, nous allons montrer la puissance de Java en citant les points forts de ce langage par rapport à d'autres langages de programmation :

- **Exécution de code:**

  Le code source Java est converti en octet qui est exécuté par la machine virtuelle Java (JVM). Comme la machine virtuelle Java est exécutée par le processeur, une application Java s'exécute sans heurts sur divers appareils ou plates-formes sans

recompilation de code. La JVM permet également aux développeurs Java de transférer des applications d'entreprise basées sur Java d'un environnement à un autre.

- **Performance :**

  Un certain nombre d'études suggèrent que Java est beaucoup plus rapide que PHP. Un grand nombre d'employés peuvent accéder à une application d'entreprise écrite en Java simultanément et efficacement. En outre, Java utilise plusieurs threads pour gérer et traiter plusieurs séquences d'instructions rapidement et simultanément.

- **Sécurité :**

  Java propose des fonctionnalités intéressantes: il permet aux développeurs de faire en sorte que le client et le serveur échangent des données via des protocoles sécurisés. Lors du développement d'applications d'entreprise, de nombreux développeurs préfèrent Java à PHP pour tirer parti de ses fonctionnalités de sécurité robustes.

A noter que le terme « Java » fait bien évidemment référence à un langage, mais également à une plate-forme : son nom complet est « Java SE » pour Java Standard Edition, et était anciennement raccourci « J2SE ». Celle-ci est constituée de nombreuses bibliothèques, ou API : citons par exemple java.lang, java.io, java.math, java.util, etc. Bref, toutes ces bibliothèques que vous devez déjà connaître et qui contiennent un nombre conséquent de classes et de méthodes prêtes à l'emploi pour effectuer toutes sortes de tâches. [14]

Le terme « Java EE » signifie Java Enterprise Edition, et raccourci en « J2EE ». Il fait quant à lui référence à une extension de la plate-forme standard. Autrement dit, la plate-forme Java EE est construite sur le langage Java et la plate-forme Java SE, et elle y ajoute un grand nombre de bibliothèques remplissant tout un tas de fonctionnalités que la plate-forme standard ne remplit pas d'origine. L'objectif majeur de Java EE est de faciliter le développement d'applications web robustes et distribuées, déployées et exécutées sur un serveur d'applications. [14]

La plateforme Java EE ajoute les possibilités nécessaires pour fournir une plateforme complète, stable, sécurisé et rapide de java au niveau entreprise.

La plateforme Entreprise fournit un ensemble de services permettant aux composants de dialoguer entre eux :

- **HTTP et HTTPS.**
- **JavaMail :**

  JavaMail est une API Java utilisée pour envoyer et recevoir des courriers électroniques via SMTP, POP3 et IMAP. JavaMail est intégrée à la plate-forme Java EE.

- **Java Data Base Connectivity (JDBC) :**

  JDBC est une API qui est constituée d'un ensemble d'interfaces et des classes qui permettent l'accès, à partir de programmes Java, à des données tabulaires (i.e. triées sous forme de table ou de tableur). Par données tabulaires, on entend généralement des bases de données contenues dans des SGBD relationnelles. Mais, JDBC n'est pas restreinte à ce type de source de données. On peut aussi accéder à des sources de données sous forme de fichiers (fichiers XML par exemple).

- **Entreprise Java Beans (EJB) :**

  Un composant Entreprise JavaBeans (EJB) est une classe ayant des champs et des méthodes pour mettre en application des modules de la logique métier (Business logic), elle permet une approche simplifiée du développement d'applications à plusieurs niveaux, dissimule la complexité de l'application et permet au développeur de composants de se concentrer sur la logique métier. [15]

  - **Les bean session (Session Bean) :** représentent les comportements associés aux sessions client. [15]
  - **Les bean entité (Entity Bean) :** représentent des collections de données telles que des lignes dans une base de données relationnelle et encapsulent des opérations sur les données qu'elles représentent. Les beans Entity sont destinés à être persistants et à survivre tant que les données auxquelles ils sont associés restent viables. [15]
  - **Les bean contrôlés par message (Message Driven Bean) :** permettent aux applications J2EE de traiter les messages de manière asynchrone. [15]

  Les entreprises bean interagissent le plus souvent avec des bases de données.

- **Java Server Pages (JSP) :**

  Les JSPs offrent un moyen simple et rapide de créer du contenu Web dynamique. La technologie JSP permet le développement rapide d'applications Web indépendantes de la plate-forme et du serveur. [10]

- **Servlet :**

  Servlet est une classe de langage de programmation Java utilisée pour étendre les capacités des serveurs hébergeant des applications auxquelles on accède au moyen d'un modèle de programmation requête-réponse. Bien que les servlets puissent répondre à tout type de demande, ils sont couramment utilisés pour étendre les applications hébergées par des serveurs Web. Pour de telles applications, la technologie Java Servlet définit des classes de servlet spécifiques à HTTP. [11]

II.2.1.2 HQL (Hibernate Query Langage)

HQL est un langage de requête orienté objet, similaire au SQL, mais au lieu d'opérer sur des tables et des colonnes, HQL utilise des objets persistants et leurs propriétés. Les requêtes

HQL sont traduites par Hibernate en requêtes SQL classiques, qui à leur tour effectuent des actions sur la base de données.

II.2.1.3 CSS (Cascading Style Sheets)

Le CSS est un langage informatique de feuille de style utilisé pour décrire la présentation d'un document écrit dans un langage de balisage tel que HTML.

II.2.1.4 JavaScript

JavaScript est un langage de programmation **orienté objet** qui permet de dynamiser une page HTML [18].

Voici quelques exemples de ce que l'on peut en faire dans une page Web [18]:

-Ouvrir des pop-up (les petites fenêtres qui s'ouvrent de manière intempestive).

-Faire défiler un texte.

-Insérer un menu dynamique (qui se développe au passage de la souris).

II.2.2 Modélisation utilisée

Modèle-vue-contrôleur ou **MVC** est choisi pour la modélisation de notre application, c'est un pattern de conception logicielle destiné aux interfaces graphiques lancé en 1978 et très populaire pour les applications web qui impose la séparation entre :

- **Modèle :**

  Représente les données, le modèle peut être divers et varié. C'est là où se trouvent les données. Il s'agit en général d'un ou plusieurs objets Java. Ces objets s'apparentent généralement à ce qu'on appelle souvent « la couche métier » de l'application et effectuent des traitements absolument transparents pour l'utilisateur. Par exemple, on peut citer des objets dont le rôle est de gérer une ou plusieurs tables d'une base de données. [17]

- **Vue :**

  Ce que l'on nomme « la vue » est en fait une IHM. Elle représente ce que l'utilisateur a sous les yeux. [17]

- **Contrôleur :**

  Gère l'interface entre le modèle et la vue. Il permet de faire le lien entre la vue et le modèle lorsqu'une action utilisateur est intervenue sur la vue. C'est cet objet qui aura pour rôle de contrôler les données. [17]

Le schéma suivant montre le fonctionnement d'une application sous le modèle MVC :

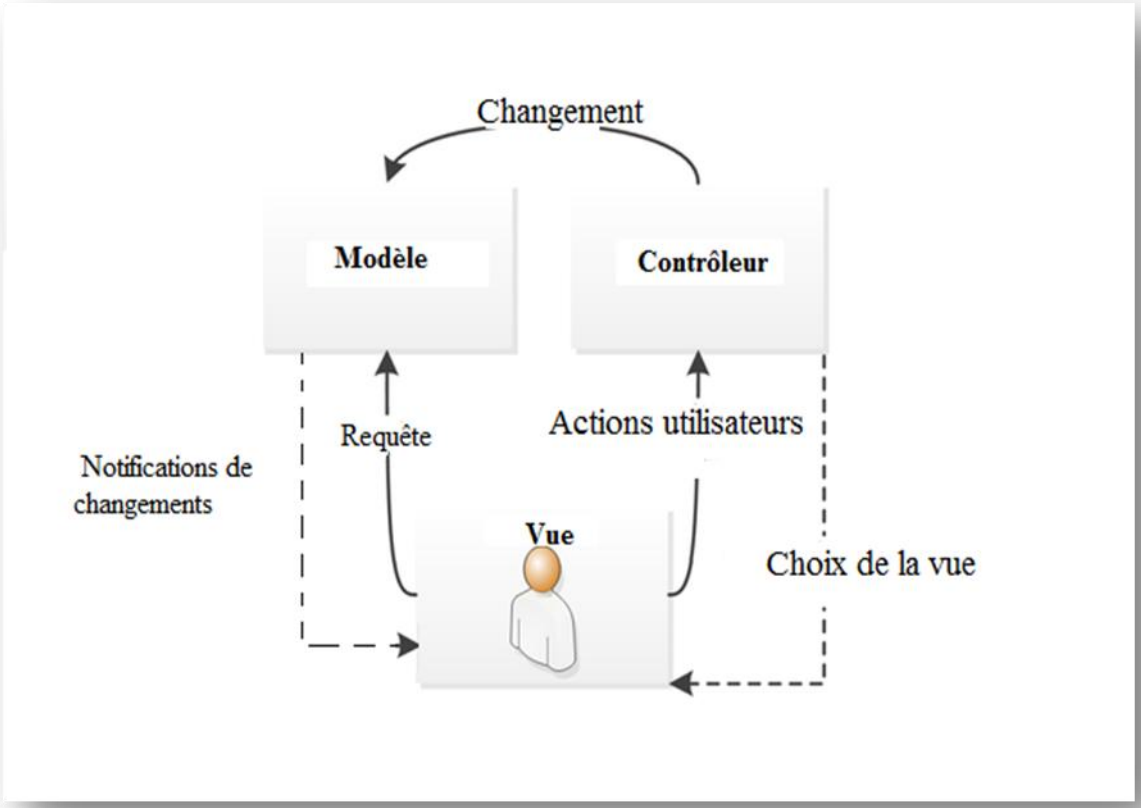


Figure II.1: Schéma du fonctionnement de MVC

Dans ce qui suit, nous allons se focaliser sur le pattern de conception MVC pour la modélisation de notre application web J2EE.

On distinguera deux modèles MVC dans la conception des applications web J2EE :

MVC 1 :

Consiste en une collection de pages JSP. La JSP gère le traitement ainsi que la présentation, ce modèle associe une servelt à chaque vue. [22]

L'Architecture modèle 1 est montrée sur la figure ci-dessous :

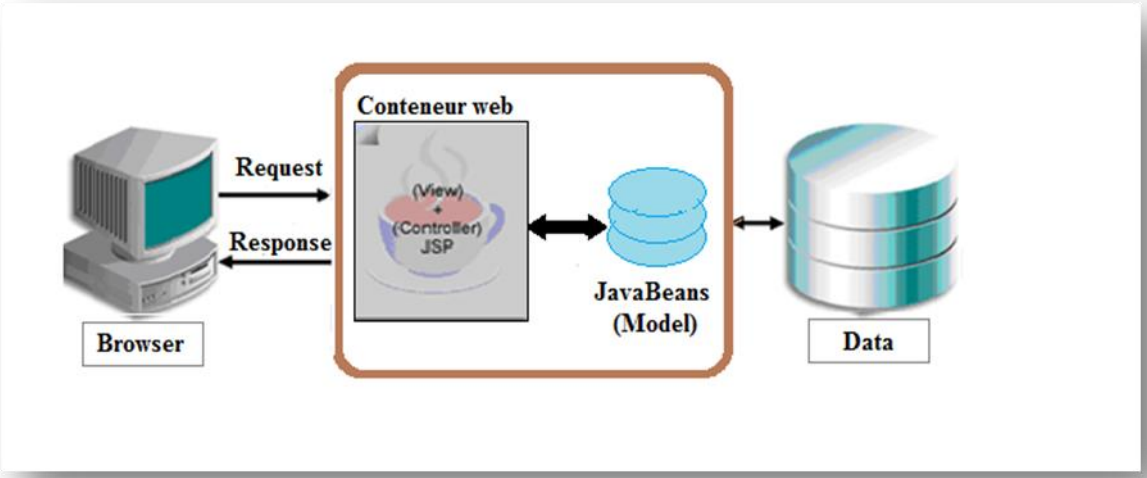


Figure II.2: Architecture Modèle 1 [22]

Autant le modèle 1 est très appréciable par sa simplicité d'application, autant la gestion d'un grand nombre de pages JSP est laborieuse. [22]

Le modèle 1 ne permet pas la construction de grosses applications Web. En effet, les tâches complexes sont difficilement implantées dans ce modèle qui tend à produire du code difficile à étendre et à maintenir. [22]

MVC 2 :

Ce deuxième modèle a une seule servelt de contrôle à l'inverse du modèle 1.

- Le **"Modèle"** contient la logique métier (JavaBean).
- La **"Vue"** est générée par les pages JSP.
- Le **"Contrôleur"**, implanté dans une servlet, apporte une gestion centralisée du traitement.

L'Architecture modèle 2 est montrée sur la figure ci-dessous :

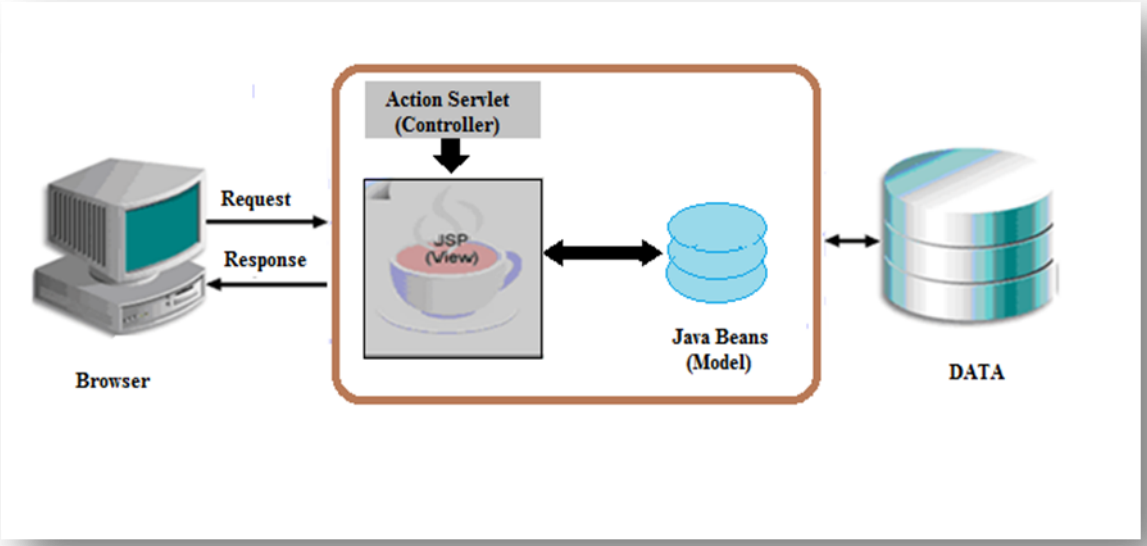


Figure II.3: Architecture Modèle 2 [22]

Le pattern MVC rend le code plus propre et sépare la logique métier (JavaBeans), le traitement côté serveur (Servlet) et l'affichage (JSP) dans des composants distincts. Chaque composant est ainsi réutilisable et remplaçable. Par contre, un Framework complexe est nécessaire pour assembler ces différents composants. [22]

II.2.3 Frameworks utilisés

II.2.3.1 Struts

C'est un Framework open source, permettant de développer des applications Web Java EE, il est basé sur l'architecture MVC2 (modèle-vue-contrôleur), il est composé de :

- **Un contrôleur** : une servlet unique.
- **La vue** : des pages JSP, avec des librairies de tag Struts pour assister le développeur.
- **Le modèle** : des classes JavaBeans pour la logique métier.

Struts 1 :

Le projet OpenSource Jakarta-Struts développé par le consortium Apache permet d'accélérer le développement d'applications Internet. Struts 1 est quasiment devenu le standard de fait pour les projets Java EE. Struts 1 est un environnement agréable et puissant qui gère l'application ainsi que les tâches courantes (routage, actions, validations...). Un autre avantage de son utilisation est le nombre croissant d'utilisateurs qui tendent à pérenniser le projet. [19]

Struts1 est un Framework qui offre des outils de validation des entrées utilisateurs (saisies et formulaires), des bibliothèques de balises JSP pour la création rapide de pages, une technique de routage pour les pages et accès web et un processus de création de formulaires à base de fichiers XML.

Fonctionnement de Struts 1 :

Il faut tous d'abord connaitre les composants principaux de cette version et leurs tâches :

- **ActionServlet:**

  C'est le contrôleur principal. Il joue le rôle de servlet et reçoit donc les requêtes du client qu'il renvoie vers les sous contrôleurs. C'est le point d'entrée unique du Framework Struts1. [16]

- **Action :**

  Destinés à effectuer des traitements sur les Beans de formulaires. Ils informent le contrôleur quelle page JSP appeler en fonction du résultat des traitements. [23]

- **ActionForm :**

  Destinés à récupérer et à effectuer des traitements de validité sur les données postées par un formulaire. [23]

- **Objets métier, ou Bean:**

  Ce sont les modèles. Ils sont utilisés par les actions lors du traitement. [16]

- **Struts-config.xml:**

  Ce fichier de configuration est le cœur de Struts 1 qui met en relation tous les éléments précédents. En fonction de l'URL il indique quelle action utiliser avec quel formulaire s'il est nécessaire. [16]

Le cycle d'opération de Struts 1 est montré dans la figure ci-dessous :

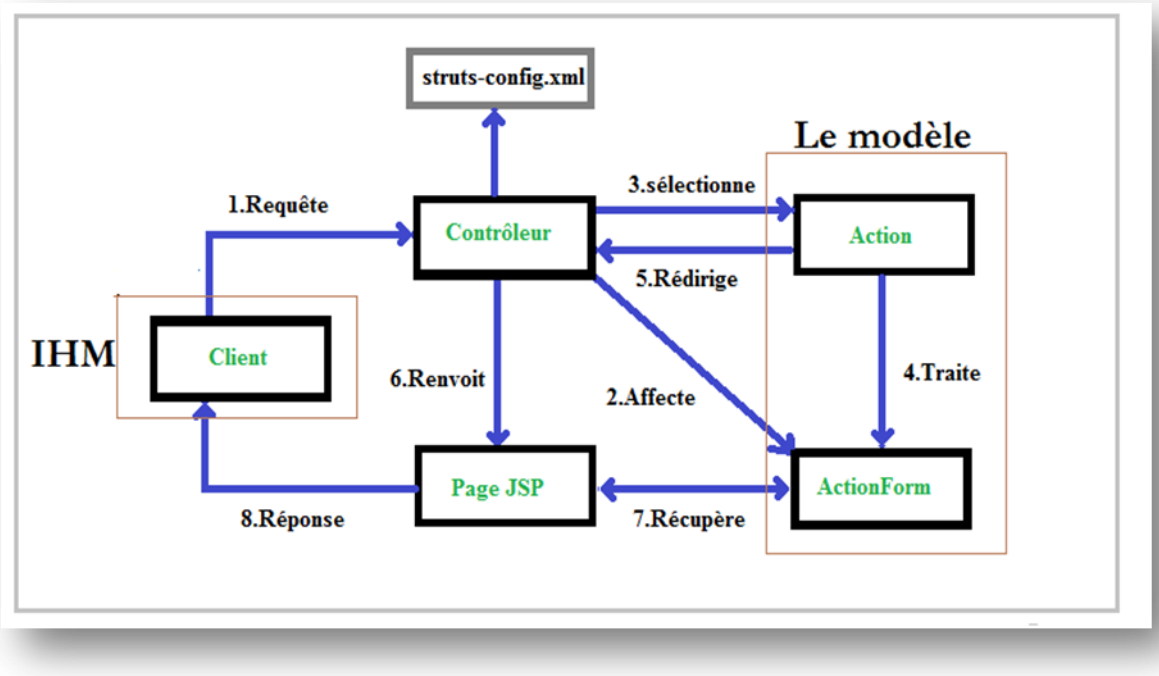


Figure II.4: Cycle d'une opération Struts 1

Le traitement effectué par Struts 1 est comme suit [23] :

1. Le client effectue une requête (via un formulaire par exemple).

2. Le contrôleur, configuré par le fichier **struts-config.xml** affecte le Bean de formulaire **ActionForm** avec les données saisies dans le formulaire.

3. Le contrôle donne la main au Bean d'action **Action.**

4. **Action** effectue des traitements sur l'ActionForm**.**

5. **Action** rend la main au contrôleur en lui spécifiant la réussite ou non de ses traitements et la page de présentation à appeler.

6. Le contrôleur appelle la page de présentation JSP.

7. La page JSP consulte les données de l'ActionForm et les formate.

8. La page JSP envoie une réponse au client.

Toutefois, le Framework Strut 1 a montré des faiblesses (problèmes de flexibilités et dépendances) parmi elles [23] :

- Les actions doivent hériter de la classe Action, donc un fort couplage entre les actions et le Framework Struts1.
- Une action doit surcharger la méthode **execute ()** de la classe Action.

- Les Beans de formulaire doivent hériter de la classe ActionForm.

Struts 2 :

Il s'agit du résultat d'une Fusion de Struts 1 et de WebWork[1], Struts 2 constitue le nouveau Framework de persistance de la communauté Open Source Apache. Son fonctionnement se veut ouvert aux technologies Web et s'intègre parfaitement aux applications J2EE. [19]

Struts 2 intègre la nouvelle architecture MVC 2. Il se trouve être plus flexible et propose plus de fonctionnalités que la version Struts 1 tout en diminuant le couplage entre les objets. [19]

Le Framework Struts 2 a amélioré les faiblesses de Strut 1, par exemple :

- Les classes ActionForm dans Struts 1 ont disparu, elles sont devenues intégrées directement dans les classes actions.
- Les classes Action sont devenues libres d'hériter selon leurs besoins.
- Struts 2 a ajouté la notion d'intercepteur pour effectuer des pré/post traitements.

Fonctionnement de Struts 2 :

Les composants principaux de Framework Struts 2 sont :

- **web.xml :**

  C'est un fichier de configuration qui contient les deux éléments filter et filter-mapping utilisés pour la configuration du FilterDispatcher.

- **FilterDispatcher** :

  C'est la servlet de base du Framework Struts 2, il permet de traiter toutes les requêtes en entrée, il permet l'accès aux éléments de base du Framework pour traiter les requêtes. [23]

- **Struts.xml :**

  Permet de spécifier la relation entre une action, une classe java et la méthode à exécuter dans cette classe et une page de vue selon le résultat du traitement retourné par la méthode.

[1] WebWork : WebWork MVC Framework est un Framework d'applications Web, conçu pour créer des sites Web dynamiques avec un minimum d'effort et une flexibilité maximale.

Le cycle d'opération de Struts 2 est montré dans la figure ci-dessous :

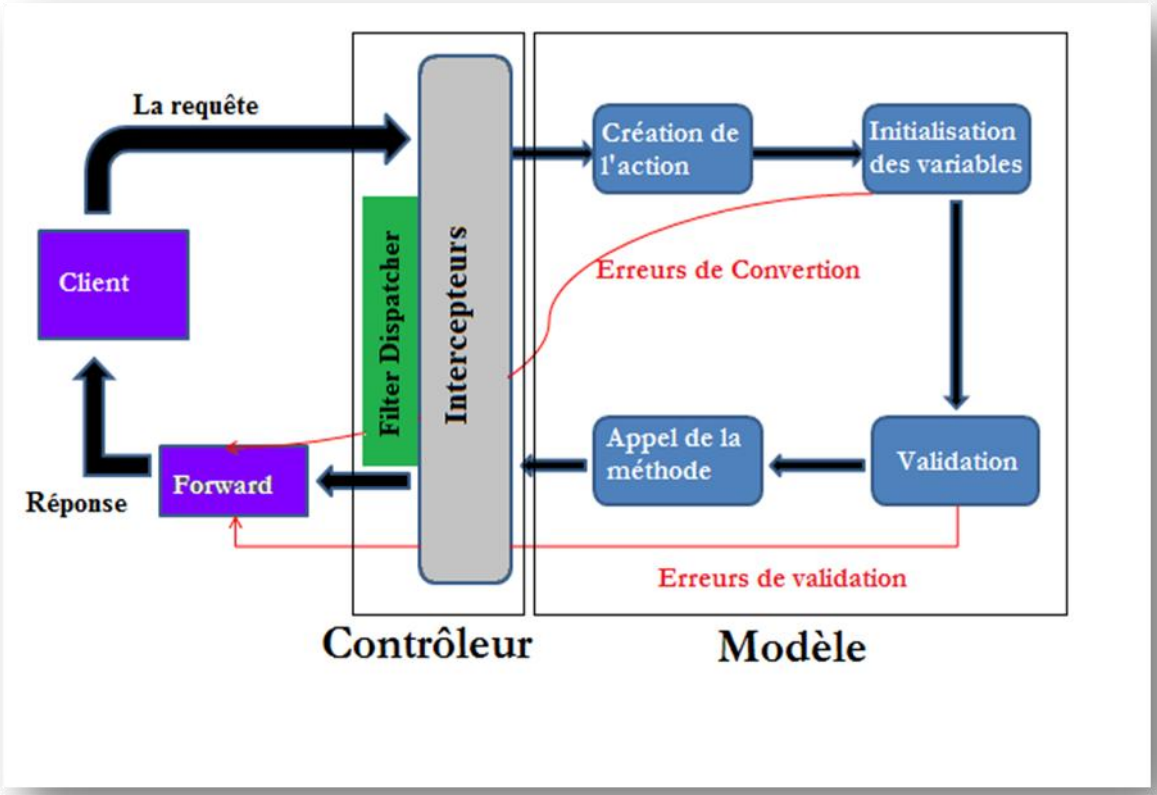


Figure II.5: Cycle d'une opération Struts 2

Le déroulement des étapes d'une requête est comme suit :

1. La requête va passer par le point d'entré du Framework Strut 2 **« FilterDispatcher ».** Les intercepteurs permettent de faire des pré/post traitements sur cette requête.

2. Le Framework va chercher l'action et la méthode à exécuter de cette requête dans le fichier struts.xml.

3. L'action est instanciée, peuplée puis le formulaire est validé, par contre s'il y a une erreur de conversion de variable (la variable de la classe action est un nombre entier et la valeur envoyée par le client est une chaine de caractères), le Framework va retourner une erreur de conversion.

4. Appel de la méthode contenant la logique de l'action pour effectuer les traitements, et après avoir l'achever elle doit retourner le résultat (une chaine de caractères).

5. Délégation de l'affichage à la vue selon le résultat retourné par l'action. [23]

Struts 2 offre beaucoup d'avantages, on peut citer parmi elles [19]:

- Fonctionne avec tous les serveurs Java EE (Tomcat, GlassFish, WebSphere, Weblogic...).
- Propose une architecture solide et stable (projet Apache).
- Adapter aux applications web de grande taille.
- Struts2 permet de décomposer une application complexe en composants plus simples.
- Garantit un développement similaire par les équipes de programmeurs.
- Possède une documentation abondante.
- Permet un développement rapide et peu onéreux.

II.2.3.2 Hibernate

C'est un Framework open source gérant la persistance des objets en base de données relationnelle. Hibernate s'intéresse à la persistance des données telle qu'elle s'applique aux bases de données relationnelles (via JDBC) [12]. Il permet de transformer les tables de base de données en une représentation objet (Objet mappé).

Le mapping objet/relationnel (Object/Relational Mapping : ORM) permet de faire la correspondance entre la base de données et les objets mappés.

Ci-dessous, l'architecture avec laquelle une application utilise le Framework Hibernate pour interagir avec la base de données.

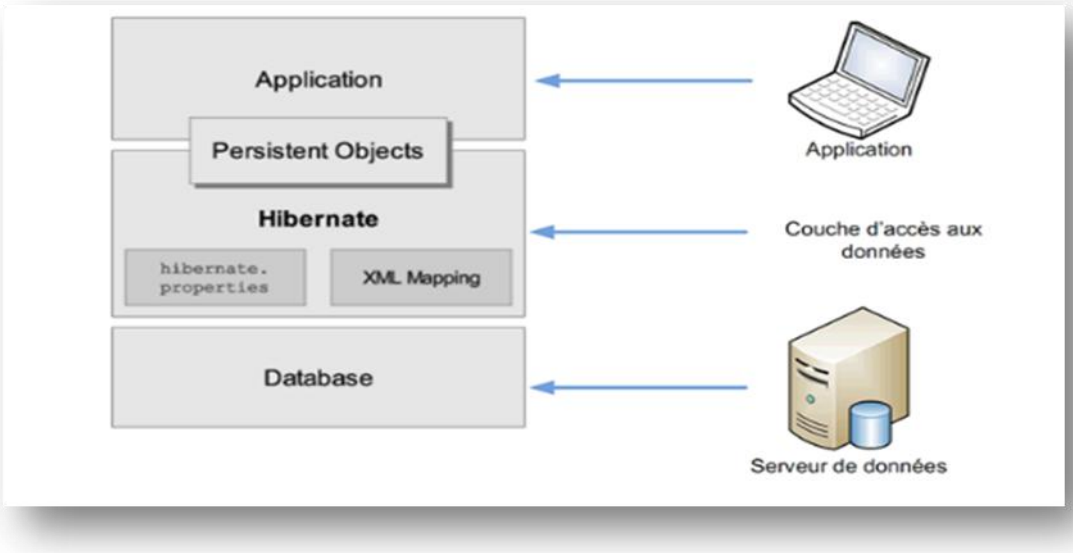


Figure II.6: Architecture Hibernate [12]

Pour une application, java l'utilisation d'Hibernante présente plusieurs avantages par rapport au JDBC, parmi elles [12]:

- **Java Persistance API Provider :**

  En plus de sa propre API "native", Hibernate est également une implémentation de la spécification JPA. En tant que tel, il peut être facilement utilisé dans tout environnement prenant en charge JPA, y compris les applications Java SE, les serveurs d'applications Java EE.

- **L'évolutivité :**

  Hibernate a été conçu pour fonctionner dans un cluster de serveurs d'applications et fournir une architecture hautement évolutive. Hibernate s'adapte parfaitement à tous les environnements qui servent des centaines d'utilisateurs, ou pour des applications critiques desservant des centaines de milliers de personnes.

- **Fiabilité :**

  Hibernate est bien connu pour son excellente stabilité et sa qualité, prouvée par l'acceptation et l'utilisation par des dizaines de milliers de développeurs Java.

- **Extensibilité :**

  Hibernate est hautement configurable et extensible.

- **La persistance est transparente** :

  Hibernate fournit une persistance transparente et le développeur n'a pas besoin d'écrire de code de manière explicite pour mapper des tables de base de données sur des objets d'application lors de l'interaction avec le SGBDR. Avec JDBC, le développeur doit effectuer manuellement cette conversion avec des lignes de code. [21]

- **Code dépendant de la base de données :**

  Une application utilisant JDBC pour gérer des données persistantes (tables de base de données) ayant un code spécifique à la base de données en grande quantité. Le code écrit pour mapper les données de la table aux objets de l'application et inversement consiste en réalité à mapper les champs de la table aux propriétés de l'objet. Au fur et à mesure que la table change ou que la base de données change, il est essentiel de changer la structure de l'objet et de modifier le code écrit pour mapper table à objet / objet à table. Hibernate fournit ce mappage lui-même. Le mappage réel entre les tables et les objets d'application est effectué dans des fichiers XML. S'il y a des changements dans la base de données ou dans une table, il ne sera alors nécessaire que de modifier les propriétés du fichier XML. [21]

- **Prise en charge du langage de requête :**

  Le JDBC prend en charge uniquement Structured Query Language (SQL) natif. Le développeur doit trouver le moyen efficace d'accéder à la base de données, c'est-à-dire de sélectionner une requête efficace parmi plusieurs requêtes à effectuer. Hibernate fournit un langage puissant de requête Hibernate Query Language (indépendant du type de base de données) exprimé dans une syntaxe familière

semblable à SQL et comprenant une prise en charge complète des requêtes polymorphes. Hibernate supporte aussi les instructions SQL natives. Il sélectionne également un moyen efficace d'exécuter une tâche de manipulation de base de données pour une application. [21]

II.3 Architecture de déploiement

L'application a la possibilité d'exécuter ces fonctionnalités sur différentes machines (machine cliente, serveur de base de données, serveur d'applications). Les fonctionnalités de l'application sont décrites en trois couches :

- **Présentation** : représente le client (Interface Homme Machine).
- **Persistance** : représente le coté stockage de données.
- **Service métiers :** représente le traitement ou la partie applicative de l'application.

Certaines couches peuvent être sous découpées comme la couche Service métiers.

II.3.1 Architecture 1-tiers

C'est le modèle centralisé, tous les couches sont sur la même machine.

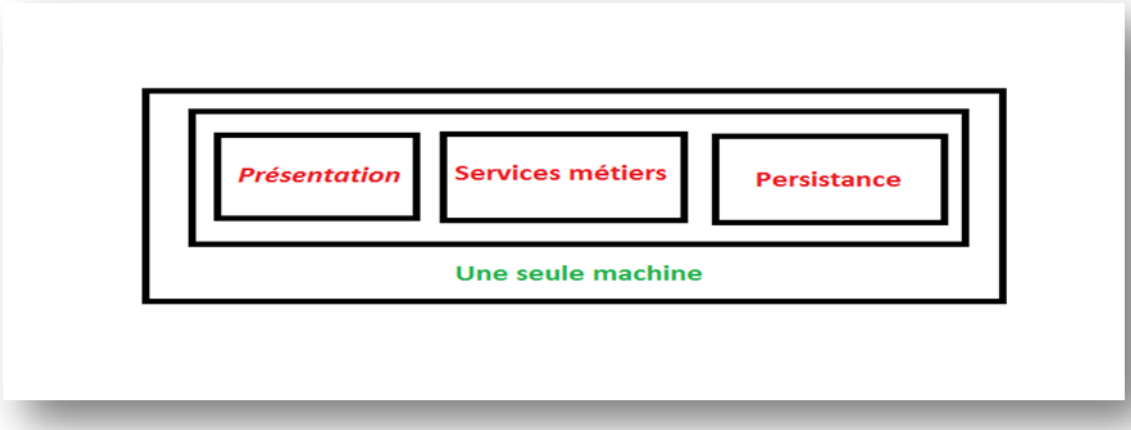


Figure II.7: Architecture 1-tiers

II.3.2 Architecture 2-tiers

C'est une architecture où les fonctionnalités d'une application se trouvent dans deux machines différentes, c'est un modèle typiquement client/serveur :

- **La présentation :** c'est le coté client.
- **La persistance :** c'est le coté serveur.

- **Le service métiers :** soit complètement coté client ou complètement coté serveur, soit découpés entre la partie serveur et la partie client.

II.3.3 Architecture 3-tiers

Dans cette architecture chaque couche s'exécute sur une machine différente :

- La présentation s'exécute sur la machine cliente.
- Le service métiers s'exécute par exemple sur un serveur d'application.
- La persistance s'exécute par exemple sur un serveur de base de données.

Puisque notre application est développée avec J2EE, la figure suivante montre comment notre application va être déployée dans une architecture 3-tiers.

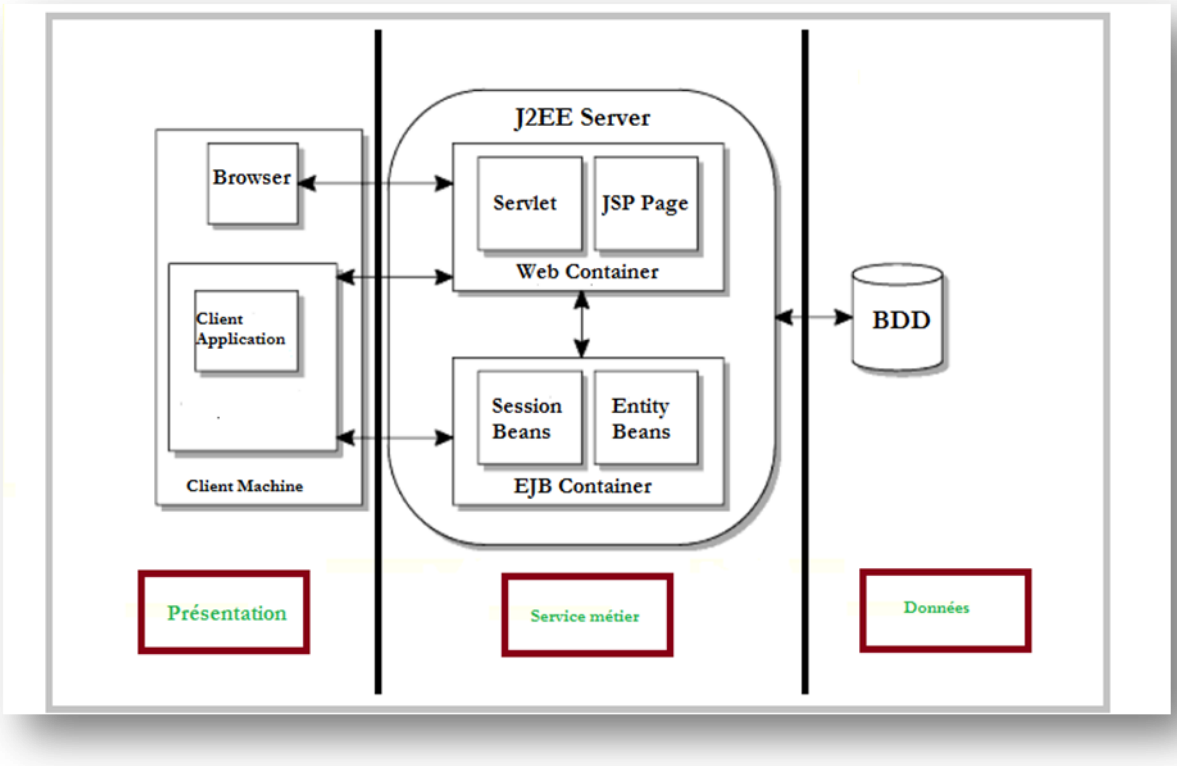


Figure II.8: Architecture 3-tiers J2EE

II.4 Conclusion

Dans ce chapitre, nous avons présenté les différents outils de développement de notre application. Nous avons présenté Struts 1 et Sruts 2 en introduisant le pattern de conception MVC. Ensuite nous avons montré l'intérêt d'utiliser les deux Framework Strut 2 et Hibernate dans notre travail en citant les avantages de leurs utilisations dans le cadre des applications web. Le chapitre s'est terminé par la présentation des architectures de déploiement supportées par notre application (1, 2 et 3 tiers).

Chapitre III: Réalisation de l’application

III.1 Introduction

Ce dernier chapitre est consacré à la partie réalisation de notre PFE. Nous présentons notre application avec l'intégration des deux Framework Struts 2 et Hibernate selon le design pattern MVC. Au début de ce chapitre, nous allons commencer par citer l'objectif de notre application ainsi que la description du fonctionnement de chaque élément de l'application. A noter ici, que nous avons pris en compte durant la phase de l'implémentation le respect des règles et des conseils de sécurité que nous avons présentés dans le premier chapitre.

III.2 Description de l'application

Les **sites e-commerce à travers le monde entiers** ont connu une croissance impressionnante au cours de ses dernières années et les consommateurs achètent de plus en plus via ces sites.

Dans ce PFE, nous avons implémenté une application web e-commerce « E-Shop » avec assez de fonctionnalité qui permettent aux clients de bénéficier de plusieurs possibilités d'usage. Cette application permet aux clients non seulement l'achat des articles mais aussi de gagner de l'argent à travers la vente de leurs produits. Cette utilisation est consolidée par un accès contrôlé à travers une gestion de rôle. En plus, notre plan de sécurité garantie la confidentialité des données sensibles des utilisateurs.

III.3 Modélisation de l'application

Dans un souci de réaliser notre application selon un cadre professionnel garantissant son bon fonctionnement, nous avons opté pour une modélisation avec le design pattern MVC permettant de séparer les trois parties essentielles de l'application (Présentation, Service métiers, Persistance) :

- **Le Modèle (la persistance)** : cette partie gère les données de notre application.
- **La Vue (la présentation)** : cette partie se concentre sur l'affichage.
- **Le Contrôleur (Service métiers)** : cette partie gère les échanges entre les deux parties précédentes.

Pour le développement de notre application, nous avons utilisé Struts 2, un Framework fonctionnant selon le design pattern MVC.

De l'autre coté et afin d'optimiser la partie traitement de l'application, nous avons utilisé le Framework Hibernate. Ce dernier permet d'améliorer les performances de l'application coté persistance.

La figure III.1 montre l'architecture globale de notre application selon MVC et avec l'intégration d'Hibernate.

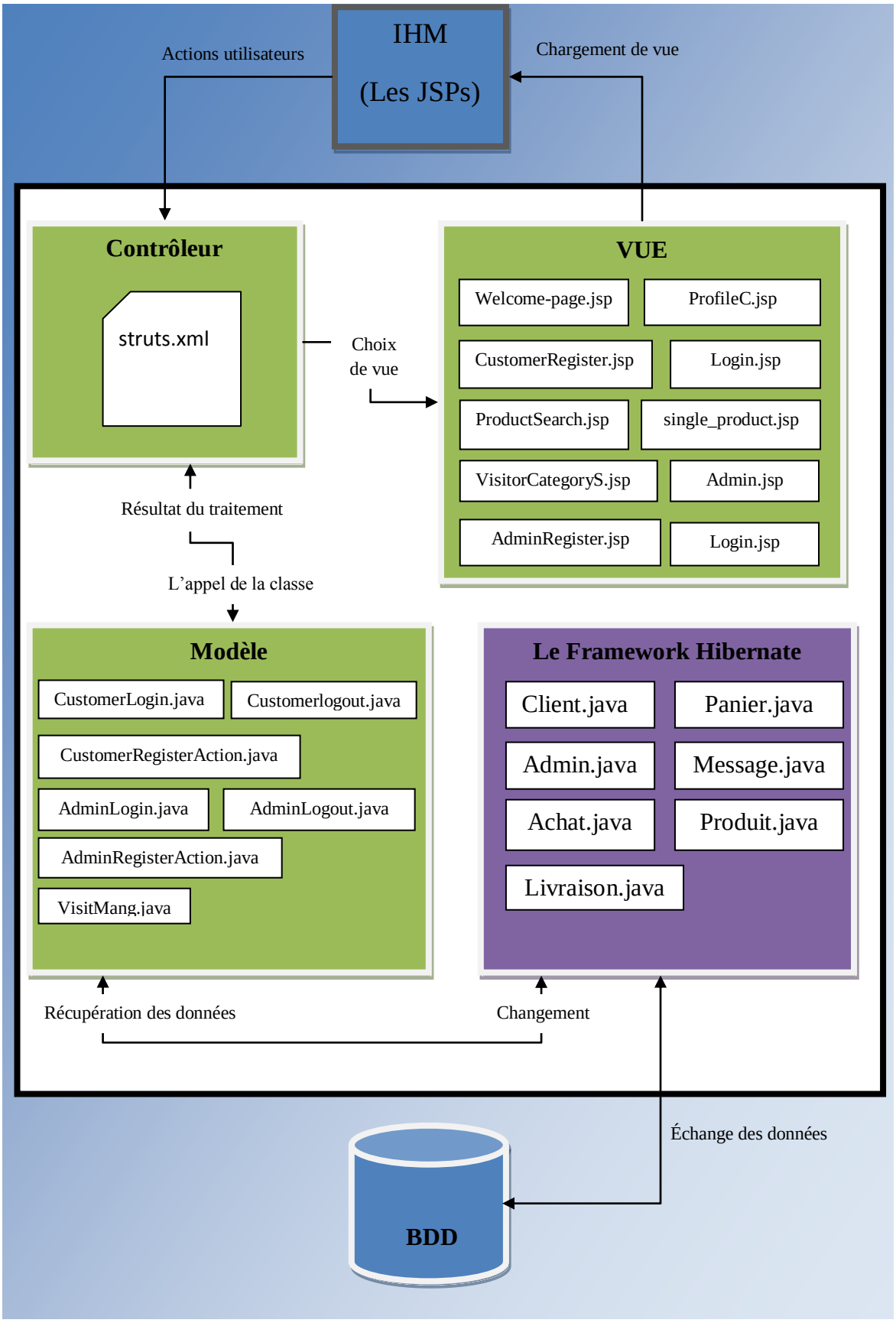


Figure III.1: Architecture globale de l'application

- **Le Contrôleur** : contient le fichier struts.xml qui joue le rôle d'intermédiaire entre le modèle et la vue.
- **Le Modèle** : contient les JavaBeans de l'application.
- **La Vue** : contient les JSPs de l'application.

Ci-dessous, un extrait du Contrôleur :

```
<action name="CRegis"  class="model.CustomerRegisterAction">
        <result name="suc" >Welcome-page.jsp</result>
        <result name="back" >CustomerRegister.jsp</result>
</action>
<action name="logoutcus"  class="model.LogoutCus">
        <result name="suc" >Welcome-page.jsp</result>
</action>
<action name="detail"  class="model.CustomerLogin" method="seedetail">
        <result name="suc" >single_product.jsp</result>
</action>
```

Dans ce qui suit, nous allons montrer avec un exemple (l'inscription d'un client) les cycles d'opération d'une fonctionnalité de l'application en intégrant les deux Frameworks Struts 2 et Hibernate.

1. Pour qu'un utilisateur puisse s'inscrire, il faut qu'il accède à la page CustomerRegister.jsp.
2. Après avoir rempli les données essentielles pour l'inscription du formulaire, l'utilisateur envoie la requête de l'inscription pour l'action.
3. Lorsque le serveur reçoit la requête, il va consulter son fichier de configuration web.xml (descripteur du déploiement du projet).
4. La requête sera routée vers la servlet de base du Framework (FilterDispatcher), c'est le point d'entré unique du Framework.
5. Le Framework consulte le fichier struts.xml, il va chercher la classe à exécuter pour cette action, dans notre cas c'est la classe CustomerRegisterAction.java.
6. Le Framework instancie la classe et remplit les variables de la classe par les valeurs envoyer par l'utilisateur et invoque la méthode, dans notre cas c'est la méthode execute().
7. On arrive à la partie persistance, le Framework Hibernate intervient à cette étape, nous allons échanger les données nécessaire pour l'inscription avec la base de données à travers les objets mappés qui représente les tables de la base de données. Dans ce cas, il s'agit de la classe Client.java.

8. Après avoir effectué son traitement, la méthode execute () retourne le résultat (soit l'utilisateur est inscrit ou il y a un problème) selon lequel struts.xml va choisir la JSP concerné par ce traitement.

9. L'utilisateur reçoit la vue (soit la réussite ou l'échec de son inscription).

III.4 Implémentation de l'application

III.4.1 Présentation de la page d'accueil

La page d'accueil a été accomplie avec un design modern et clair, c'est un modèle absolu de simplicité, avec des couleurs unies et légères qui donnent au client l'envie de découvrir le reste. Cette page d'accueil rend le site facile à parcourir, l'internaute peut accéder à plusieurs pages à travers les liens proposés par cette page, il peut même s'authentifier directement ou rechercher des produits.

Le contenu de cette page est adapté aux trois modes d'affichage : écran téléphone, écran ordinateur, écran tablette.

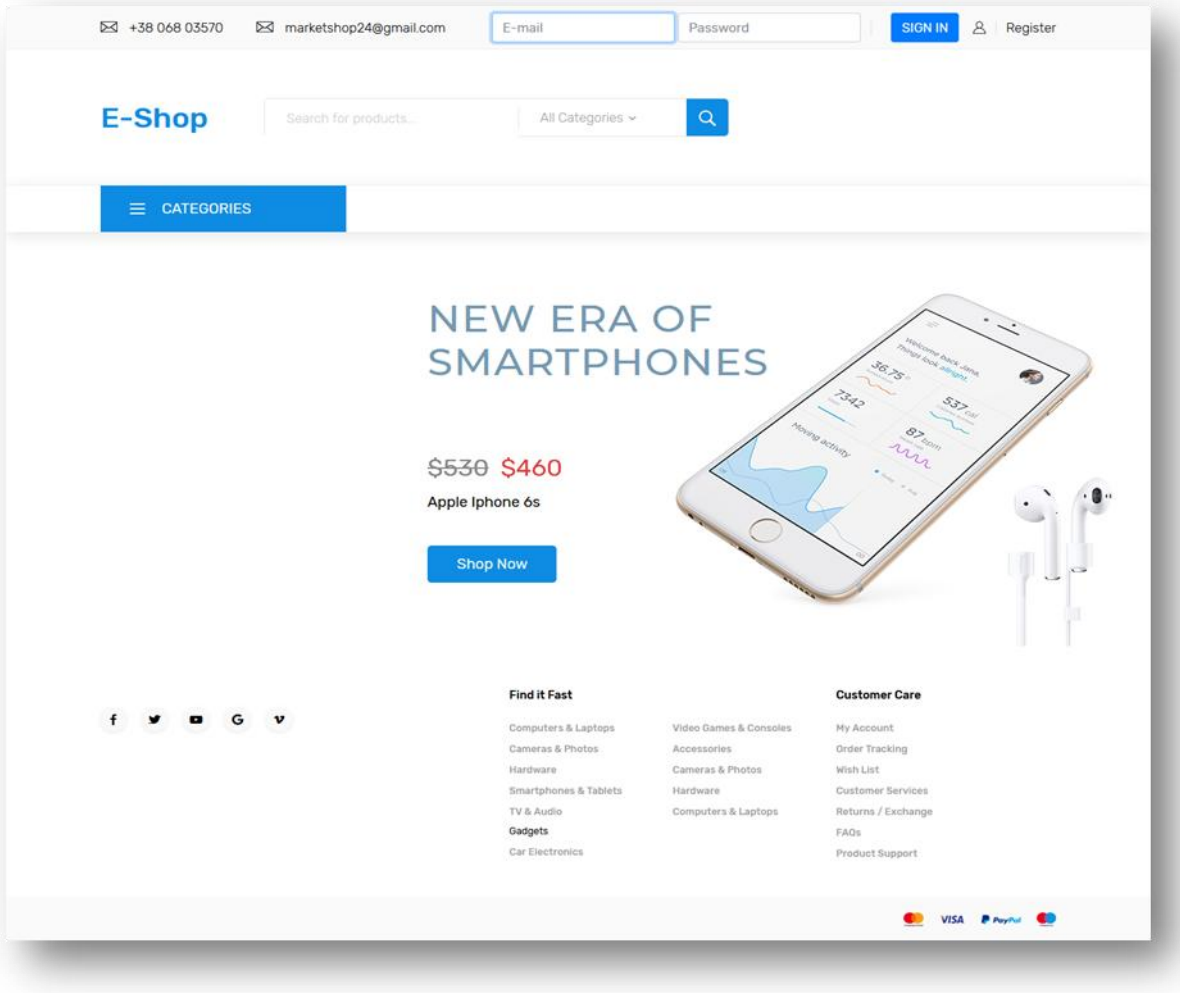


Figure III.2: Page Accueil

III.4.2 Gestion des rôles

Nous avons mis en place trois niveaux de privilège dans notre application qui permettent de faire distinguer trois profils :

- Internaute ;
- Administrateur ;
- Client (Acheteur ou Vendeur).

Dans ce qui suit, nous allons présenter les actions de chaque profil.

Le profil internaute :

- Un internaute peut seulement visualiser les produits qui sont en vente.
- Un internaute peut rechercher un produit avec son nom exact ou effectuer des recherches par catégorie, il peut même visualiser les détails d'un produit.
- Un internaute ne peut pas effectuer des changements sur la base de données (acheter un produit, ajouter un produit, modifier un produit).

Le profil administrateur :

- La page d'authentification de l'administrateur est séparée de celle du client.
- Après avoir s'authentifier, l'administrateur accède à la partie administrative de l'application.
- L'administrateur peut visualiser toutes les données relatives aux achats, aux clients, aux commandes, aux produits à confirmer, aux produits confirmés, aux messages échangés.
- L'administrateur ne peut pas visualiser les données sensibles comme les mots de passe et les numéros de paiement.
- L'administrateur peut confirmer un produit ajouté par un client avant qu'il soit met en vente, il peut aussi le supprimer.
- L'administrateur a la possibilité de bloquer des clients.
- L'administrateur peut consulter les messages envoyés par les clients et les supprimer.

Le profil client :

- Après l'étape d'authentification, le client peut accéder à son compte.
- Il peut visualiser les produits qui sont en vente (les produits des autres clients).

- Il ne peut acheter que les produits des autres clients.
- Il peut ajouter un produit.
- Il peut visualiser ses produits séparément des produits des autres.
- Il peut modifier et supprimer seulement ses propres produits.
- Le client a le droit d'envoyer des messages à l'administrateur.
- Il peut rechercher un produit par son nom exact.
- Il peut rechercher les produits d'une catégorie.
- Il peut filtrer les produits par prix.
- Il peut trier les produits par nom ou par prix.
- Le client a la possibilité de configurer les informations de son compte par exemple :
    - Changer le nom.
    - Changer le mail.
    - Changer le mot de passe.
    - Changer l'adresse.
    - Changer le numéro de paiement.

III.4.3 L'inscription

Notre application permet de gérer l'inscription du client et de l'administrateur de manière séparée, les informations demandées au client sont différentes de celle demandées à l'administrateur pour s'inscrire.

Pour cela, nous avons réalisé deux interfaces différentes.

Le client :

La figure suivante montre la page de l'inscription coté client.

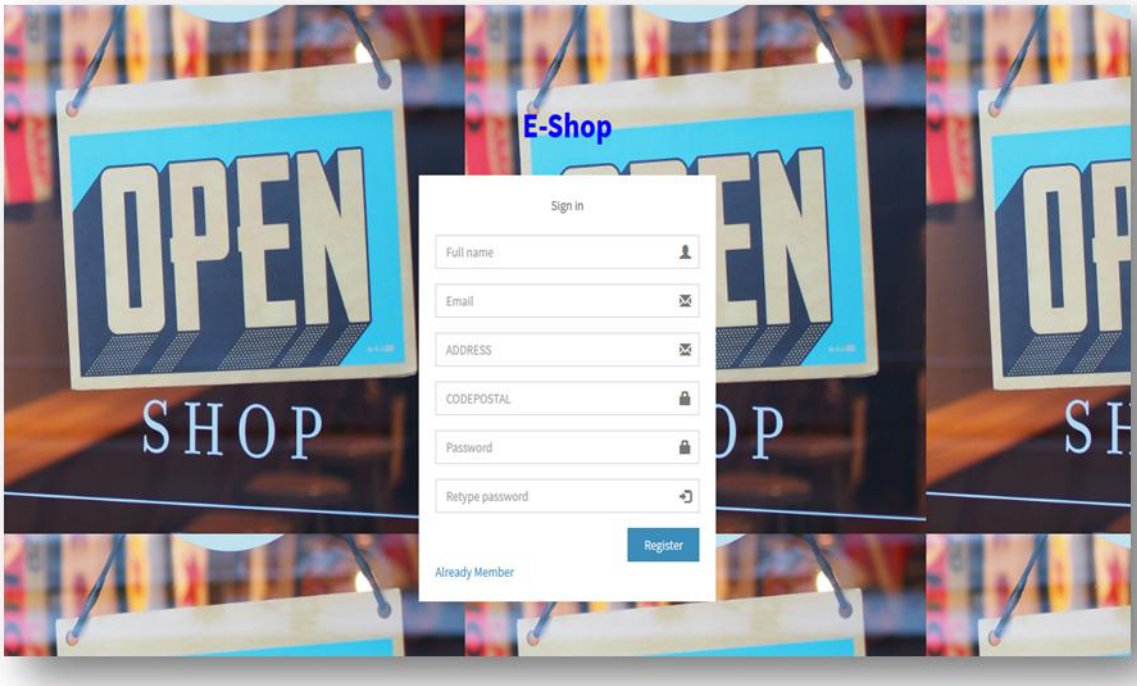


Figure III.3: Page Inscription du client

Dans ce qui suit, nous présentons les traitements effectués avant l'inscription d'un client.

1. A la soumission du formulaire, nous vérifions à l'aide de JavaScript que :
    - Le champ mot de passe a été bien rempli avec 8 caractères.
    - Le champ Confirmation a été rempli avec le même mot de passe.
    - Le champ mail a été rempli avec une adresse email valide.
    - Le champ nom et prénom ne dépassent pas les 30 caractères, et ne contiennent que des alphabets.
    - Le champ adresse a été rempli, et ne contient que les alphabets et des chiffres.
2. Coté Serveur, nous allons vérifier que la personne n'est pas déjà inscrite par la vérification de l'existence de son email dans la base de données à travers la table Client.
    - Si la personne est déjà inscrite, un message de notification lui sera envoyé.
    - Si non, on passe à l'étape suivante.
3. Maintenant que toutes les données ont été vérifiées et validées, nous allons :
    - Générer un nouveau mot de passe aléatoirement.
    - Envoyer un email de confirmation qui contient le nouveau mot de passe pour l'utiliser avec l'aide de L'API JavaMail.

- Crypter le nouveau mot de passe avec la méthode encrypt() de la classe Java Chiff1 qui utilise le Cryptage AES et l'enregistrer dans la base de données.

Dans ce qui suit, nous présentons la classe java Chiff1 permettant de crypter et de décrypter le mot de passe en utilisant le cryptage symétrique AES.

```
package model;
import java.io.UnsupportedEncodingException;
import java.security.NoSuchAlgorithmException;
import java.util.Arrays;
import java.util.Base64;
import javax.crypto.Cipher;
import javax.crypto.spec.SecretKeySpec;


public class Chiff1 {
private static SecretKeySpec secretKey;
private static byte[] key;

public static void setKey(String myKey)
{
MessageDigest sha = null;
try {
key = myKey.getBytes("UTF-8");
sha = MessageDigest.getInstance("SHA-1");
key = sha.digest(key);
key = Arrays.copyOf(key, 16);
secretKey = new SecretKeySpec(key, "AES");
}
catch (NoSuchAlgorithmException e) {
e.printStackTrace();
}
catch (UnsupportedEncodingException e) {
e.printStackTrace();
}
}
public static String encrypt(String strToEncrypt, String secret)
{
try {
setKey(secret);
Cipher cipher = Cipher.getInstance("AES/ECB/PKCS5Padding");
cipher.init(Cipher.ENCRYPT_MODE, secretKey);
return Base64.getEncoder().encodeToString(cipher.doFinal
(strToEncrypt.getBytes("UTF-8")));}
catch (Exception e) {
System.out.println("Error while encrypting: " + e.toString());   }   return null;}
```

```
public static String decrypt(String strToDecrypt, String secret) {
try {
setKey(secret);
Cipher cipher = Cipher.getInstance("AES/ECB/PKCS5PADDING");
cipher.init(Cipher.DECRYPT_MODE, secretKey);
return new String(cipher.doFinal(Base64.getDecoder().decode(strToDecrypt)));

 } catch (Exception e)   {  System.out.println("Error while decrypting: " + e.toString());     }

}

}
```

La classe Chiffl contient 3 méthodes pour permettre le chiffrement d'une variable de type String.

La première méthode **setKey (String key)** prend comme paramètre une clé secrète de type **String** pour l'utiliser dans la création d'une clé de taille 128 bits (16 octets*8) de type **SecretKeySpec** qui va être utilisé dans le chiffrement et le déchiffrement.

La méthode **encrypt (String strToEncrypt, String secret)** retourne une chaine de caractères cryptée, en utilisant la classe java **Cipher** de la Cryptographie API de Java pour chiffrer.

1. Nous allons commencer par créer une instance **Cipher** en appelant la méthode **getInstance()** avec un paramètre indiquant le type d'algorithme de cryptage souhaité.

2. Initialiser l'objet **Cipher** par sa méthode appelante **init()** avec deux paramètres. Le premier paramètre sert à mettre l'objet en mode chiffrement, le deuxième paramètre indique la clé secrète de chiffrement.

3. Chiffrer la chaine de caractères dans un encodage UTF-8 avec la méthode **doFinal()** de la classe **Cipher** avec un paramètre indiquant la chaine de caractère à chiffrer .

La méthode **decrypt (String strToDecrypt, String secret)** retourne une chaine de caractères en claire, en utilisant la classe java **Cipher** de la Cryptographie API de Java pour déchiffrer.

1. Nous allons commencer par créer une instance **Cipher** en appelant la méthode **getInstance()** avec un paramètre indiquant le type d'algorithme de cryptage souhaité.

2. Initialiser l'objet **Cipher** par sa méthode appelante **init()** avec deux paramètres. Le premier paramètre sert à mettre l'objet en mode déchiffrement, le deuxième paramètre indique la clé secrète de déchiffrement.

3. Déchiffrer la chaine de caractères cryptée avec la méthode **doFinal()** de la classe **Cipher** avec un paramètre indiquant la chaine de caractère à déchiffrer.

Ci-dessous, le code java de la méthode **execute()** :

```
Public  String  execute()  throws  Exception {
        String req="from Client where emailCus='"+mail+"'";
        Client c=H.executeHQLQuery(req);
        if (c !=null) {reason="the mail already exist"; return"back";}
       ////pour créer un nouveau mot de passe
        generte_password gen=new generte_password(String password);
        password1=gen.generate(8);        ///fin
       ///pour envoyer un mail vers le client
        Envoyer_email envoi=new Envoyer_email();
       envoi.setTo(mail);
       envoi.setMessageText("Hey !!!"+name+" This  your new password: "+password1);
       envoi.send_msg();          //// fin
      /// instanciation de la classe Chiff1 et le chiffrement AES
       Chiff1 HASH=new Chiff1();
      String hash=HASH.encrypt(password1, secretKey) ;
      String hash1=HASH.encrypt(code, secretKey) ;    ///fin
      Client client=new Client();
      client.setEmailCus(mail);
      client.setPasswordCus(hash);
      client.setNameCus(name);
      client.setAdress(address);
      client.setPayNum(hash1);
     H.executeHQLQuery3(client);  ////enregistrer le nouveau client
     return "suc";
}
```

L'administrateur :

L'inscription en tant qu'administrateur va être traitée de la même manière que celle d'un client. Par contre, il faut avoir un code fournit par l'administration pour s'assurer que cette personne peut s'inscrire avec les privilèges d'un administrateur.

Le mot de passe de l'administrateur sera crypté avec la classe Chiff1 qui utilise le Cryptage AES et enregistrer dans la base de données.

III.4.4 Authentification

La page d'authentification d'un client est séparée de celle de l'administrateur.

Le client :

Pour qu'un internaute puisse accéder à son compte, il doit passer par l'étape de l'authentification. Notre application ne propose pas qu'une seule page pour se connecter, elle offre plusieurs possibilités, parmi ces pages:

- La page de L'accueil Welcome-page.jsp.
- Une page qui visualise les détails d'un produit.
- Une page qui visualise une recherche d'un produit par son nom.
- Une page qui visualise une recherche d'une catégorie des produits.

Un utilisateur qui a été bien inscrit, peut se connecter à son compte en utilisant son email d'inscription ainsi que son mot de passe.

Nous allons vous montrez par la suite comment nous traiterons une demande de connexion d'un utilisateur.

1. A la soumission du formulaire, nous vérifions à l'aide de JavaScript que :
    - Le champ login (email) n'est pas vide.
    - Le champs mot de passe n'est pas vide, et contient 8 caractères.
2. Coté Serveur, lorsqu'on reçoit le login et le mot de passe, nous allons vérifier que :
    - L'utilisateur de ce login est inscrit.
    - Si non, on envoi un message qui indique l'échec de connexion.
    - Si oui, on passe à la comparaison des mots de passe.
3. On récupère le mot de passe crypté de la base de données, après nous allons crypter le mot de passe envoyer par l'utilisateur avec la méthode encrypt() de la classe Java Chiff1 qui utilise le Cryptage AES et comparer les deux chaine de caractère crypté.
    - Si le mot de passe est faux, on envoi un message qui indique l'échec de connexion.
    - Si le mot de passe est juste, on passe à la création de la session (http Session) pour garder les informations de ce client tout au long de sa période de connexion.
4. Pour crée une session, il faut :
    - Instancier un objet de la classe Client.java qui représente la table Client de la base de données (Un objet mappé de Hibernate) avec les données de ce client récupérées de la base de données.
    - Nous utilisons l'interface «SessionAware» pour enregistrer l'objet Client.

5. Maintenant le client accède à son compte.

L'administrateur :

La demande d'un utilisateur pour se connecter en tant qu'administrateur va être traitée de la même manière qu'un client. Sauf pour la création de la session, dans ce cas, nous allons instancier un objet Administrateur de la table Administrateur de la base de données.

III.4.5 Le compte client

Accueil :

Après avoir s'authentifier, le client accède à la page principale de son compte, qui permet de :

- Ajouter un produit à son panier en cliquant sur le bouton «ADD TO CART».
- Voir les détails d'un produit en cliquant sur son nom.
- Chercher un produit par son nom exact.
- Chercher les produits d'une catégorie.
- Envoyer un message en cliquant sur le bouton «Send Message».
- Un lien «LOG OUT» pour se déconnecter.
- Un lien «Configuration »pour accéder à la page de configuration du compte.
- Ordonner les produits par nom ou par prix, filtrer les produits par prix.

La figure III.4 montre la page profil client après la réussite de l'authentification.

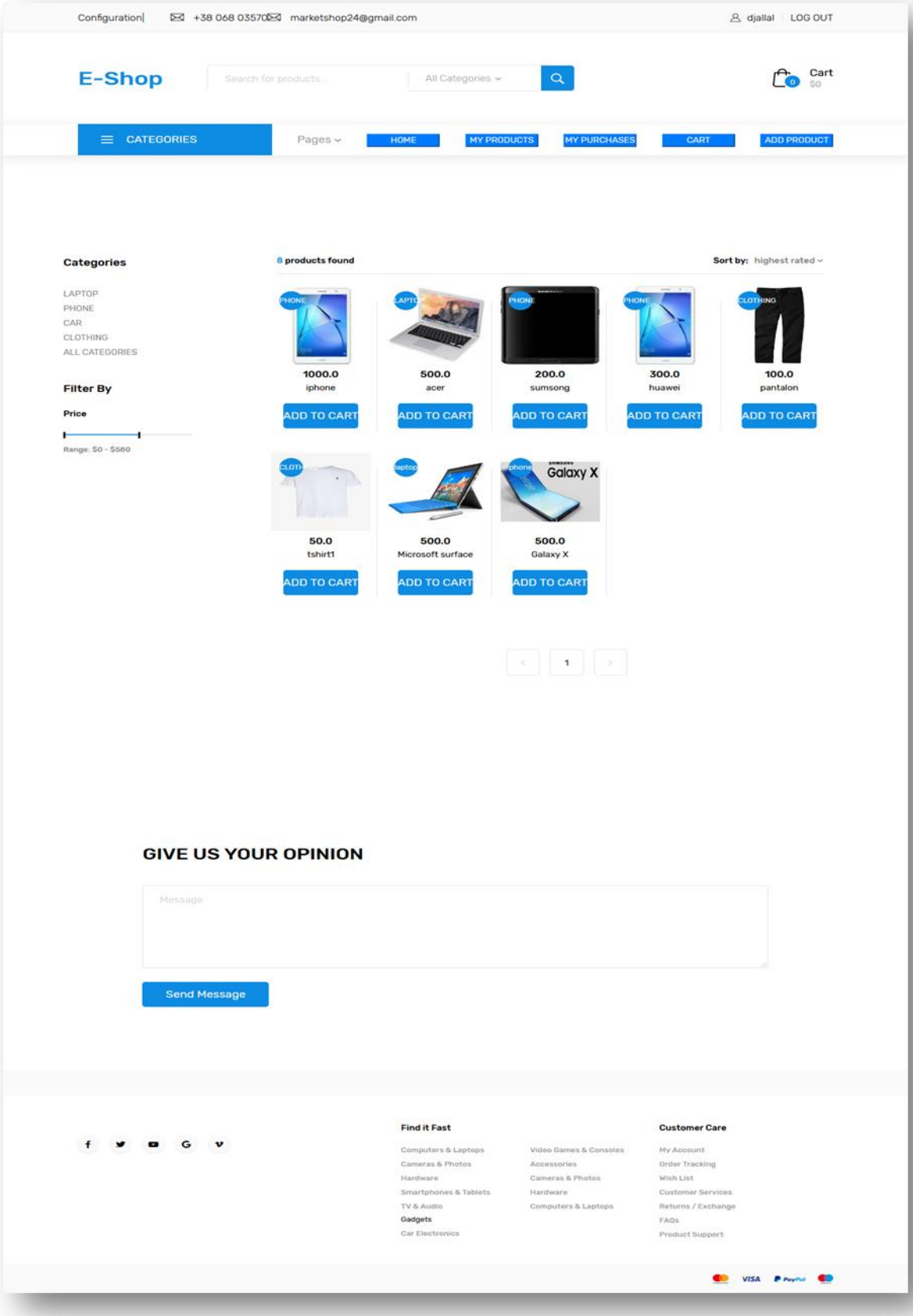


Figure III.4: Compte du client

Les produits du client :

En cliquant sur le bouton «MyProducts », la partie qui contient les produits des autres va être remplacée par les produits du client connecté. Parmi les fonctionnalités :

- Le client peut modifier la quantité et le prix de son produit en cliquant sur le bouton «Modify».
- Le client a la possibilité aussi de supprimer son produit en cliquant sur le bouton «remove».

La figure suivante montre la partie qui affiche les produits du client connecté.

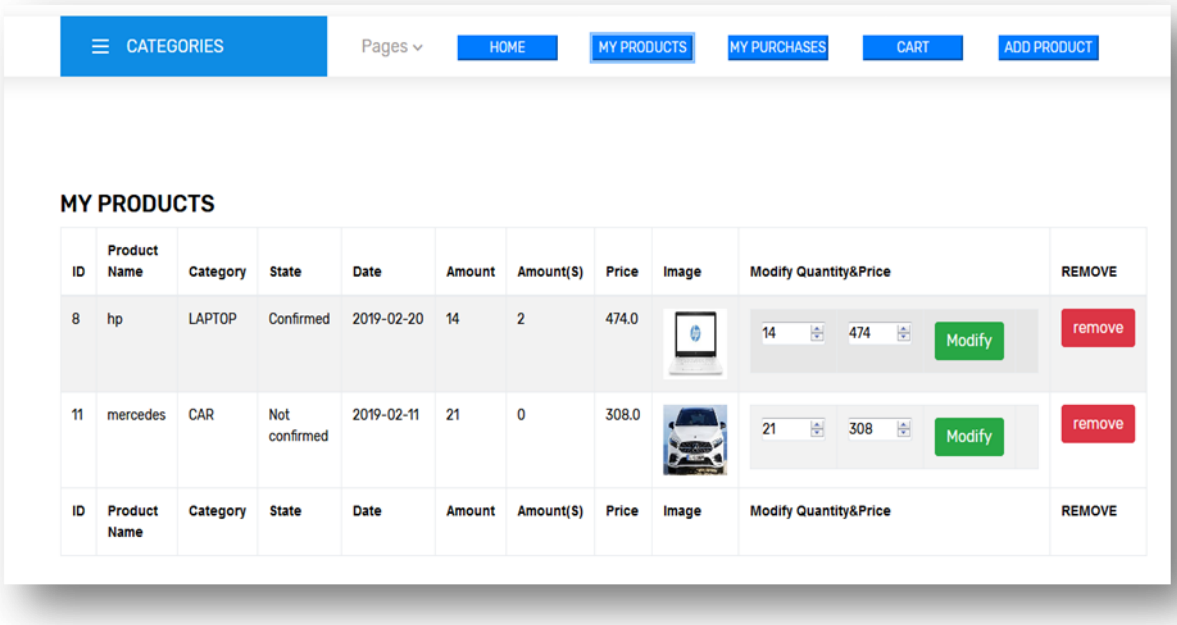


Figure III.5: Les produits du client connecté

Historique des achats :

En cliquant sur le bouton «My Sales », le client peut visualiser l'historique de ses achats.

La figure suivante montre la partie qui affiche l'historique des achats du client connecté.

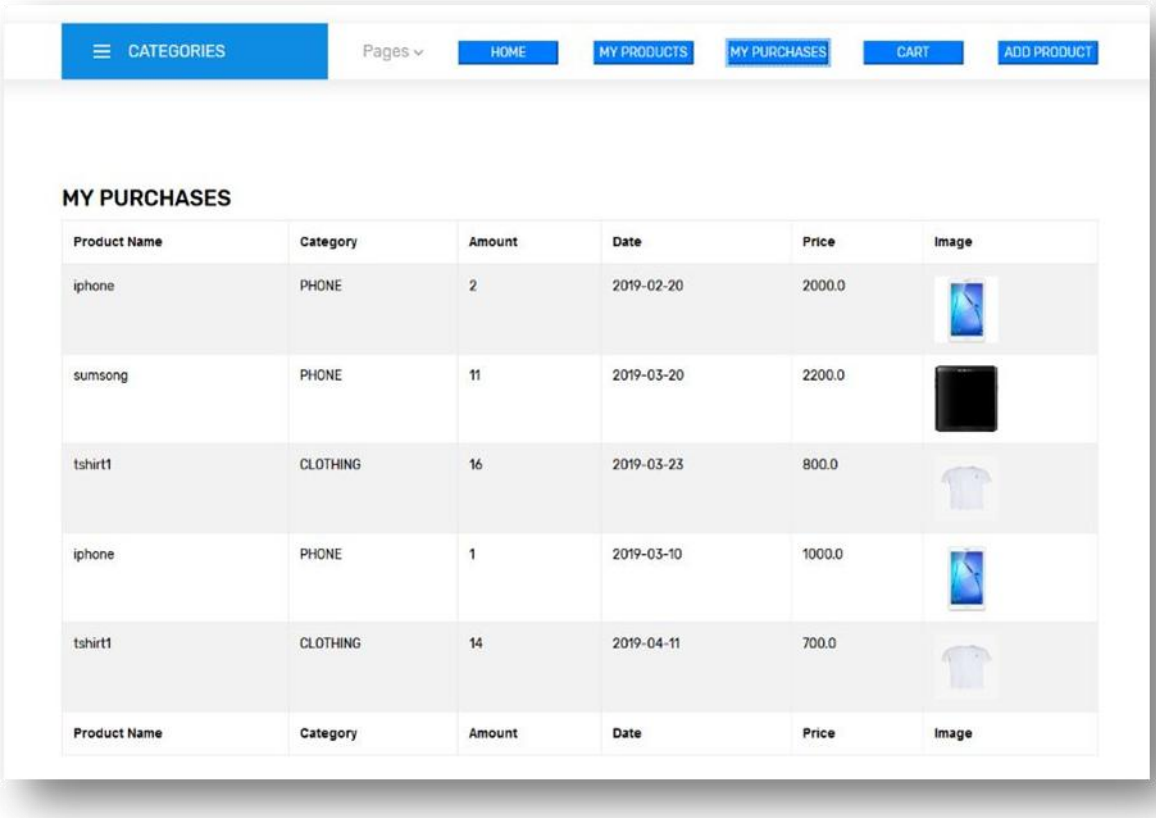


Figure III.6: Historique des achats

L'ajout d'un produit :

Le client peut visualiser la partie où il peut ajouter un produit en cliquant sur le bouton «ADD PRODUCT».

La figure III.7 montre la partie qui donne au client la possibilité d'ajouter un produit.

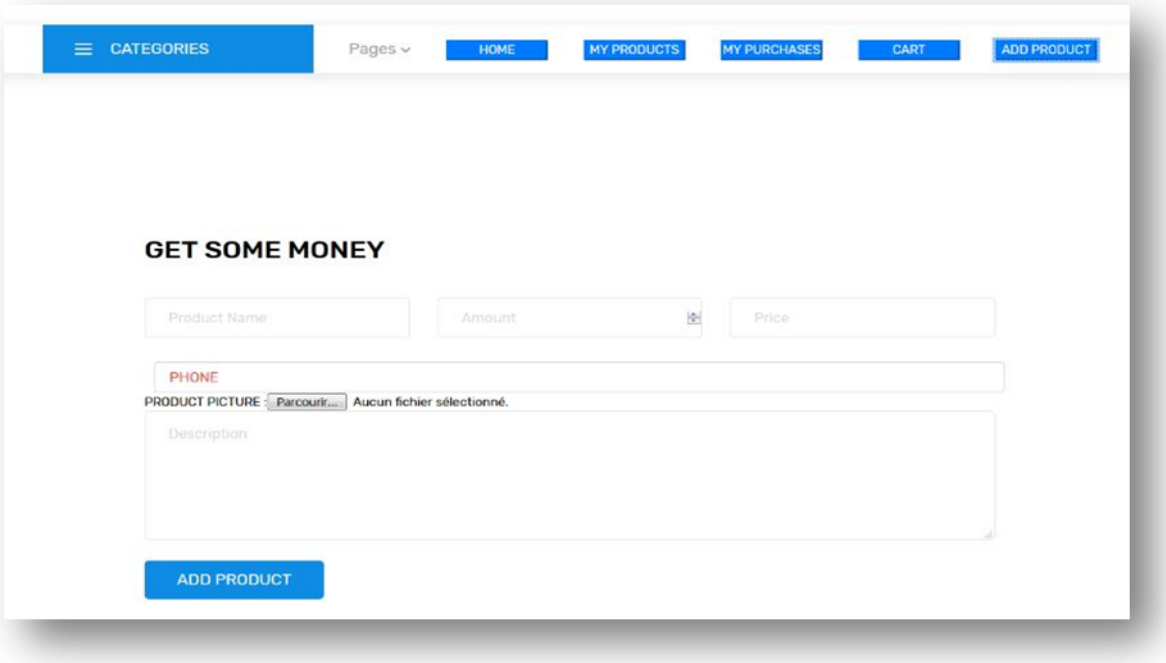


Figure III.7: L'ajout d'un produit

Le panier du client :

Le client peut consulter son panier en cliquant sur «Cart ». Il a la possibilité de supprimer des produits du panier par en cliquant sur «Delete » ou passer à l'étape suivante de l'achat en cliquant sur «BUY ».

La figure III.8 montre la partie qui affiche les produits ajoutés au panier.

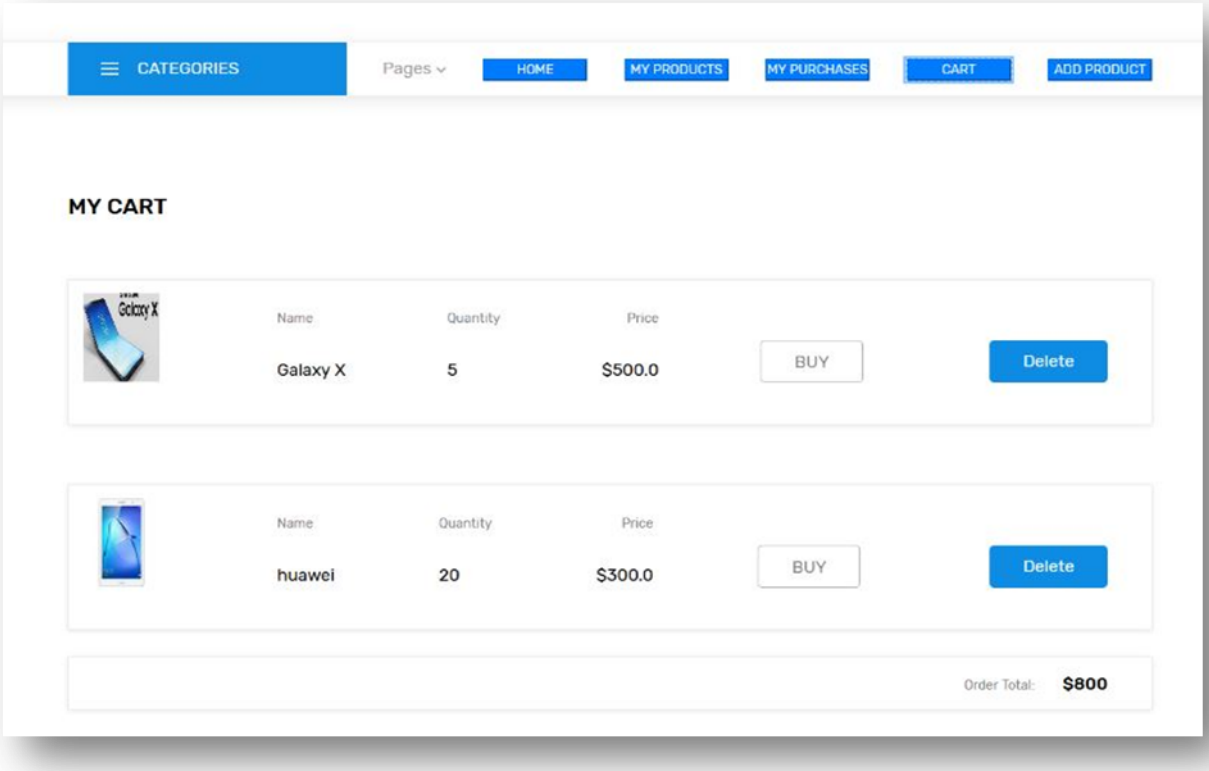


Figure III.8: Le panier

Confirmation d'achat :

Après avoir ajouté le produit dans son panier, le client et en cliquant sur le bouton «BUY» il peut passer à la page de confirmation.

Le client est obligé de remplir la quantité à acheter. Par la suite, il clique sur le bouton «Confirm».

La figure III.9 montre une page qui permet la confirmation d'achat d'un produit.

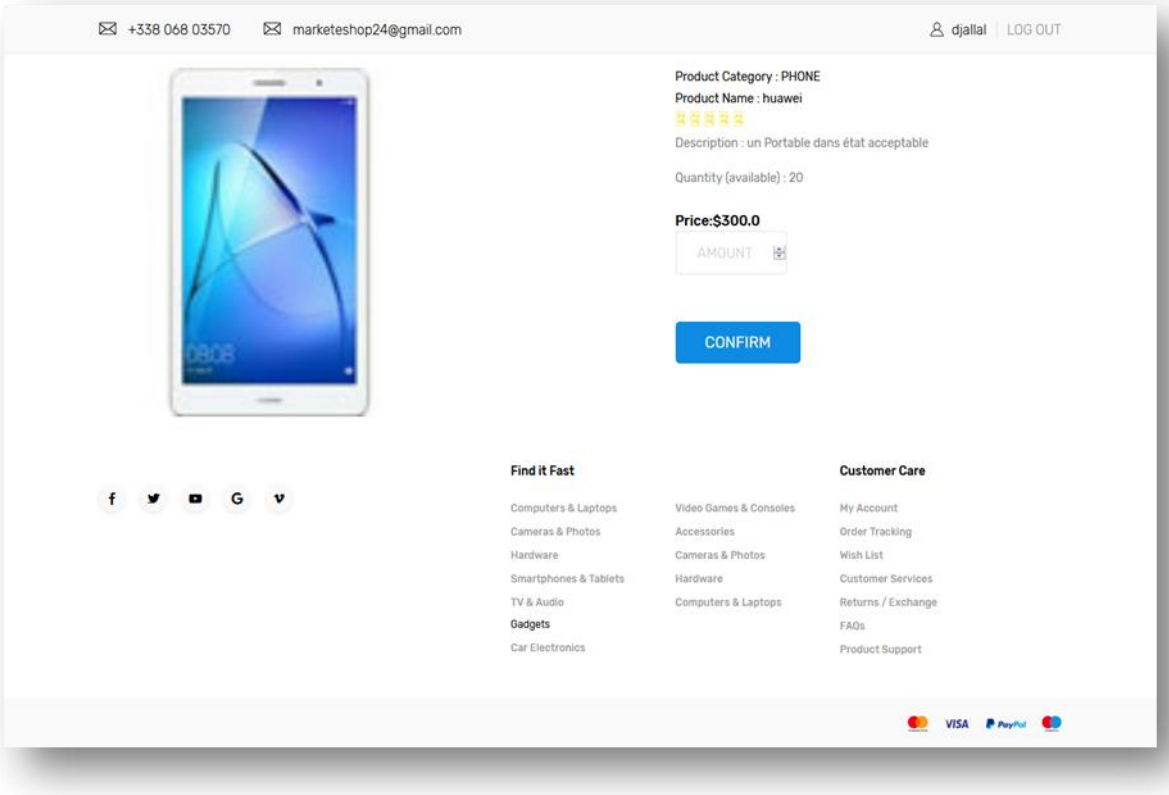


Figure III.9: Confirmation d'achat

III.4.6 Le compte administrateur

Après avoir s'authentifier, l'administrateur accède à son compte. Il peut donc supprimer les messages envoyés par les clients.

L'administrateur peut visualiser toutes les données relatives au (client, achats, produits en vente, produits qui sont en attentes de confirmation, livraisons).

Accueil :

Les figures III.10, III.11 et III.12 montrent la page du profil administrateur.

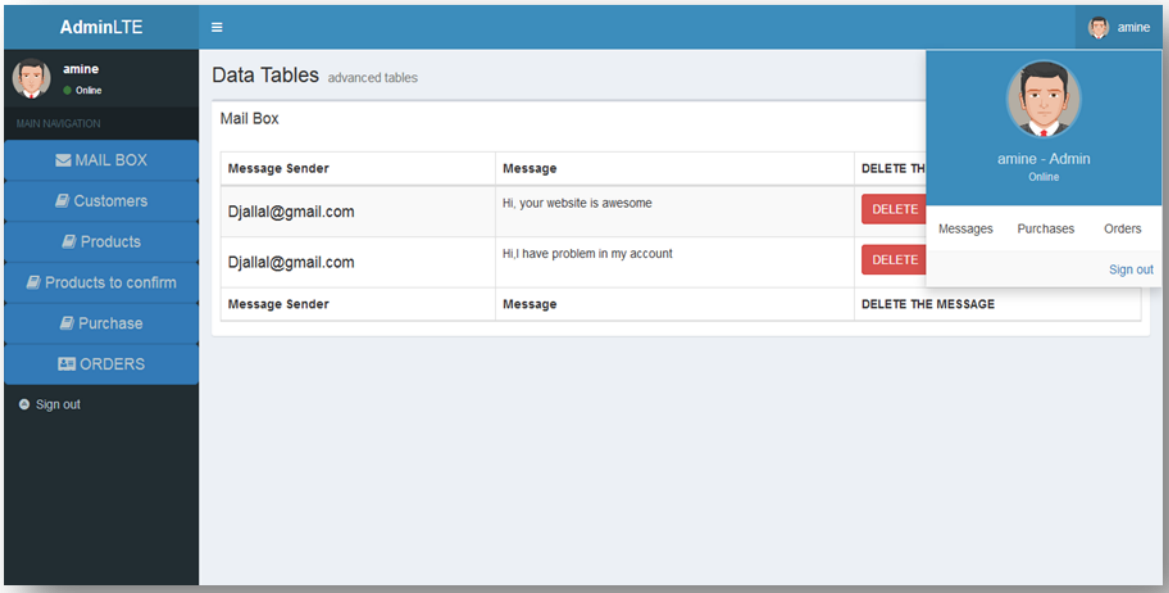


Figure III.10: Le profil administrateur

Elle permet d'apercevoir les données de tous les clients.

L'administrateur peut aussi supprimer un compte de client.

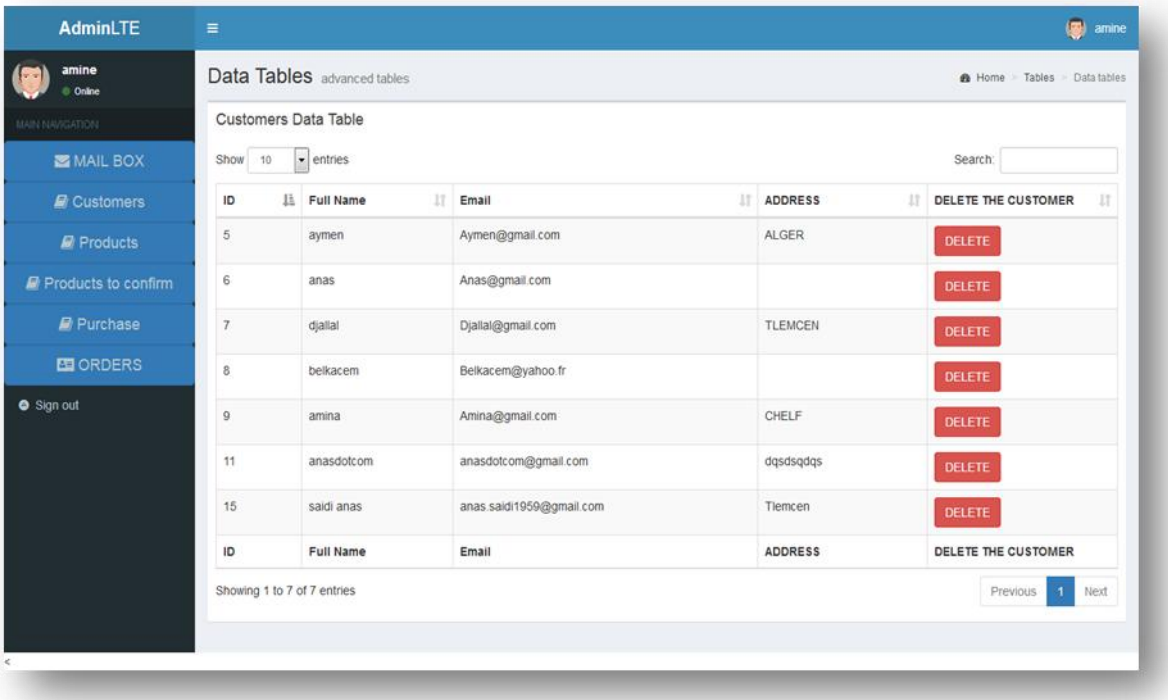


Figure III.11 : Aperçu des clients

La confirmation des produits :

L'administrateur peut confirmer un produit ajouté par le client pour le mettre en vente, ou le supprimer.

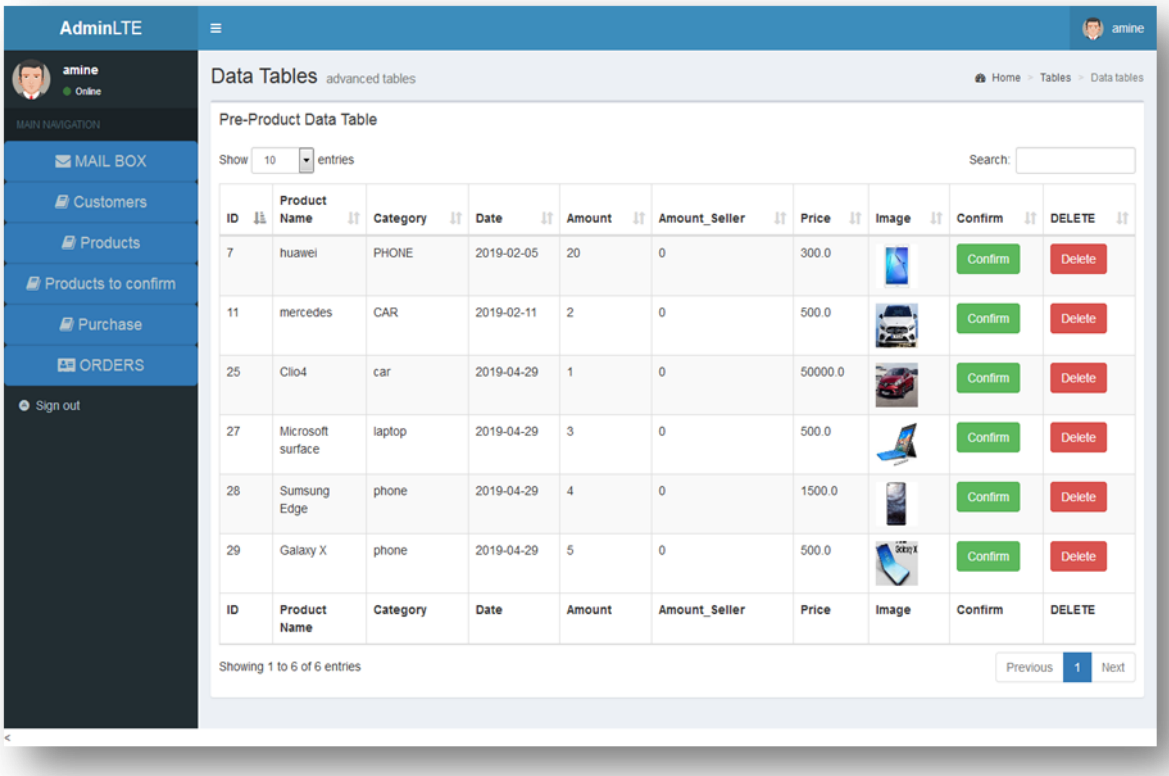


Figure III.12: Confirmation d'un produit

III.5 Déploiement de l'application

III.5.1 Architecture 2-tiers

Le 2-tiers est typiquement le modèle client/serveur, donc nous avons utilisé deux machines pour le déploiement de notre application dans cette architecture :

- La première machine représente le client, c'est la partie présentation de l'application.

- La deuxième machine représente le serveur, une machine contenant les deux autres parties de l'application (la persistance et le service métiers). Nous avons utilisé un serveur d'application J2EE « GlassFish version 4.1.1 » pour le service métiers qui communique avec le serveur MySQL Server de la base de données locale (vérifie c'est le sens est bon ici).

Nous avons mis donc l'adresse IP de la machine serveur pour faire appel à l'application sur la machine du client.

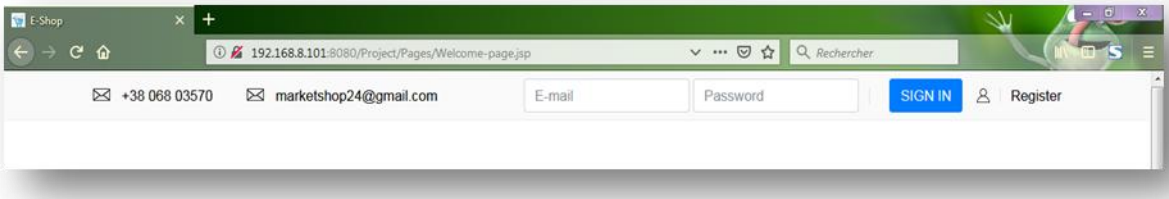


Figure III.13 : Déploiement 2-tiers

III.5.2 Architecture 3-tiers

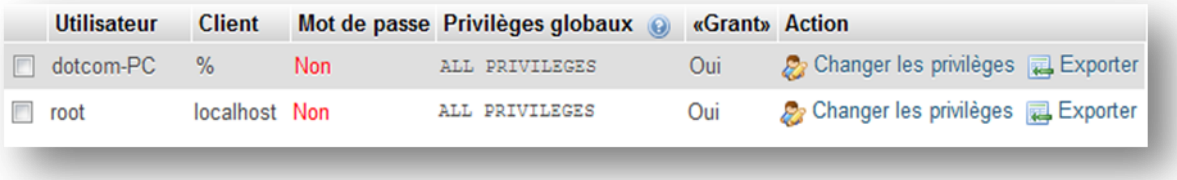
Dans cette architecture, nous avons séparé le serveur d'application et le serveur MySQL sur deux machines différentes. Le déploiement dans ce cas nécessite trois machines.

- La première machine représente le client, c'est la partie présentation de l'application.

- La deuxième machine représente le serveur d'application J2EE « GlassFish version 4.1.1 » pour le service métiers.

- La troisième machine représente le serveur MySQL pour la base de données.

Pour la configuration coté serveur MySQL, nous avons crée un utilisateur à distance dans le serveur MySQL, pour autoriser la machine qui contient le serveur d'application d'accéder à la base de données.



Utilisateur	Client	Mot de passe	Privilèges globaux	«Grant»	Action
dotcom-PC	%	Non	ALL PRIVILEGES	Oui	Changer les privilèges Exporter
root	localhost	Non	ALL PRIVILEGES	Oui	Changer les privilèges Exporter

Figure III.14: Configuration du serveur MySQL

Pour faire communiquer notre serveur d'application avec le serveur MySQL, nous avons besoin de :

- Changer l'utilisateur puisque « root » c'est pour le fonctionnement local de l'application.
- Fournir l'adresse IP de la machine qui contient le serveur MySQL.

La configuration coté serveur d'application est montrée dans la figure suivante :

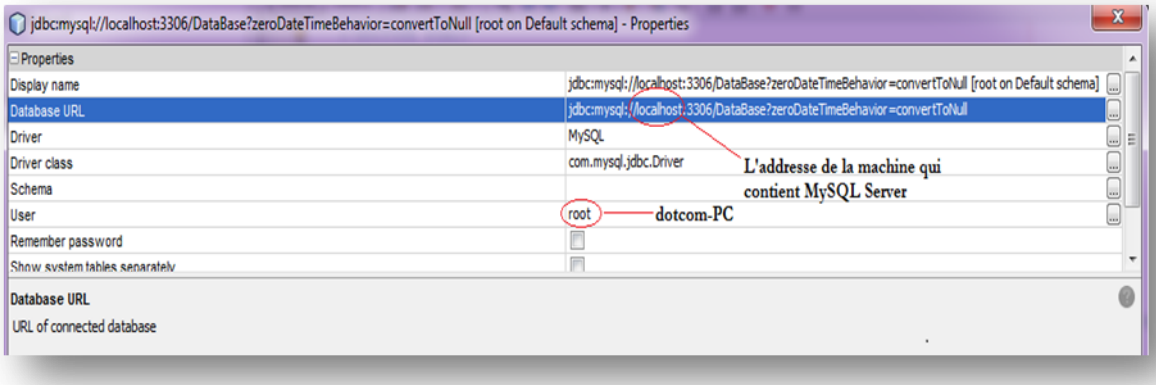


Figure III.15: Configuration du serveur d'application

III.6 Conclusion

A travers ce PFE, nous avons implémenté une application Web « E-Shop » en intégrant des mécanismes de sécurité comme l'authentification (des clients et de l'administrateur), la gestion des rôles (la définition de 3 rôles différents : Internaute, Administrateur et Client) et le chiffrement des données sensibles (utilisation de l'algorithme AES). En plus de l'aspect sécurité, « E-Shop» propose beaucoup de fonctionnalités intéressantes aux clients, en particulier l'achat et la vente de leurs produits. A noter que nous avons modélisé notre application à l'aide du design pattern MVC à travers l'utilisation du Framework Struts 2, un Framework qui nous a offert plus de souplesse et de facilité pour le développement de notre application Web coté serveur. Pour la partie persistance des données, nous avons intégrée le Framework Hibernate, un Framework trop puissant mais souple et facile d'utilisation afin de gérer les données de notre application. Nous avons aussi déployé notre application dans deux architectures différentes qui sont le 2-tiers et le 3-tiers. Cette dernière architecture permet de séparer réellement les trois couches MVC sur trois machines différente et ainsi envisagé le passage à l'échelle.

Conclusion générale

Dans le but d'appliquer notre connaissance acquise sur la sécurisation des applications web, nous sommes parvenus à mettre en œuvre une application e-commerce sous le nom « E-Shop» qui propose plusieurs fonctionnalités, cette application permet aux clients un usage facile et rapide.

Le plan de sécurité de notre application permet de garantir la confidentialité, la disponibilité de l'application. Avec l'intégration de plusieurs mécanismes comme la gestion des rôles, l'authentification pour protéger le contenu des clients et le chiffrement des données sensibles avec l'algorithme de cryptage symétrique AES qui est répandu dans les entreprises et les sociétés informatiques.

La modélisation de notre application se base sur le pattern de conception MVC, ce pattern est utilisé dans le monde professionnel de développement des applications web. L'architecture MVC permet de construire des grosses applications web, les tâches complexes sont facilement implantées.

L'utilisation des deux Framework de développement Struts 2 et Hibernate nous a permis de rendre notre application flexible et évolutive. Nous avons aussi bénéficié de la puissance de ces Frameworks qui facilite au développeur d'éditer et de modifier les fonctionnalités de l'application.

Ce travail nous a permis d'acquérir une expérience professionnelle, car nous avons eu de la chance d'améliorer notre savoir faire dans les domaines de la sécurité, la conception et la réalisation des applications web, ce projet nous a offert la possibilité d'avoir de nouvelles connaissances en matière de programmation et de développement.

Pour finir, il y aura toujours des améliorations à envisager pour rendre l'application plus performante par exemple :

- Ajout des services techniques de sécurité Comme (Serveur Radius pour l'authentification).
- Ajout d'un service qui permet d'étudier le comportement des clients, pour proposer un contenu qui lui ressemble (recommandations).
- Ajout du service de paiement en ligne.

Bibliographie

[1] HTTPCS, « **OWASP** Top 10 des failles de sécurité,», [En ligne]. Available: https://www.httpcs.com/fr/top-10-owasp. [Consulté le 15/04/2019].

[2] OWASP, «OWASP Top 10-2017 Les Dix Risques de Sécurité Applicatifs Web les Plus Critiques,» OWASP The Open Web Application Security Project, [En ligne] https://www.owasp.org/index.php/Top_10-2017_Top_10 . [Consulté le 15/04/2019].

[3] OWASP, « ANNEXE À L'ENTENTE DE DÉVELOPPEMENT DE LOGICIEL SÉCURISÉ,» OWASP The Open Web Application Security Project.

[4] Digital Guide, «Aperçu des procédures de cryptage,», [En ligne] https://www.ionos.fr/digitalguide/serveur/securite/apercu-des-diverses-procedures-de-cryptage/. [Consulté le 16/04/2019].

[5] L'AES : Advanced Encryption Standard. Disponible sur : https://aesencryption.net/l. [Consulté le 26/04/2019].

[6] Guillaume Harry, Failles de sécurité des applications Web Principes, parades et bonnes pratiques de développement. 2012.

[7] Digital Guide, «L'AES : Advanced Encryption Standard,», [En ligne] https://www.securiteinfo.com/cryptographie/aes.shtml. [Consulté le 17/04/2019].

[8] CLUB DE LA SECURITE DE L'INFORMATION FRANÇAIS, «Sécurité des applications Web, Comment maîtriser les risques liés à la sécurité des applications Web ?,», [En ligne] https://repo.zenksecurity.com/Programmation/Securite%20des%20applications%20Web.pdf . [Consulté le 17/04/2019].

[10] ORACLE, «Java Server Pages Technology,», [En ligne]. Disponible sur: https://www.oracle.com/technetwork/java/index-jsp-138231.html. [Consulté le 20/04/2019].

[11] ORACLE, «What Is a Servlet?,», [En ligne]. Disponible sur: https://docs.oracle.com/javaee/7/tutorial/servlets001.htm#BNAFE. [Consulté le 20/04/2019].

[12] HIBERNATE, «Hibernate ORM,», [En ligne]. Disponible sur: http://hibernate.org/orm/. [Consulté le 22/04/2019].

[13] Cyrille Herby, «Apprenez à programmer en java,» [En ligne]. Disponible sur: https://openclassrooms.com/fr/courses/26832-apprenez-a-programmer-en-java. [Consulté le 22/04/2019].

[14] Médéric Munier ,« Introduction au Java EE [En ligne]. Disponible sur : https://openclassrooms.com/fr/courses/626954-creez-votre-application-web-avec-java-ee/618989-introduction-au-java-ee. [Consulté le 23/04/2019].

[15] Oracle, «Java 2 Platform, Enterprise Edition (J2EE) Overview,» [En ligne]. Disponible sur: https://www.oracle.com/technetwork/java/javaee/appmodel-135059.html. [Consulté le 22/04/2019].

[16] Crequis Fabien / Mastroianni Valentin, «Le Framework STRUTS,» [En ligne]. Disponible sur: http://users.polytech.unice.fr/~rey/cours/supports/lpsil/2012/rapport_g11.pdf. [Consulté le 22/04/2019].

[17] Cyrille Herby, « Structurez mieux votre code avec le pattern MVC!,» [En ligne]. Disponible sur : https://openclassrooms.com/fr/courses/26832-apprenez-a-programmer-en-java/25552-structurez-mieux-votre-code-avec-le-pattern-mvc. [Consulté le 23/04/2019].

[18] Mathieu Nebra, « Tout sur le JavaScript!,» [En ligne]. Disponible sur : https://openclassrooms.com/fr/courses/146276-tout-sur-le-javascript/142043-presentation. [Consulté le 23/04/2019].

[19] Bruce Bujon- Aurélien Colladant- Vincent Van Houtte, «Présentation du Framework MVC Struts2,» Disponible sur :http://gardeauxvincent.eu/Documents/ProjetJEE/BVC_Hibernate Struts2/Struts/index.htm. [Consulté le 23/04/2019].

[21] Dipti Phutela, Mindfire Solutions, « Hibernate Vs JDBC,» [En ligne]. Disponible sur : http://www.mindfiresolutions.com/mindfire/Java_Hibernate_JDBC.pdf. [Consulté le 23/04/2019].

[22] Badr Benmammar, «Programmation Web côté serveur : Exécution d'un programme,». Module « construction d'applications réparties ». Master 2 réseaux et systèmes distribués. Département d'informatique, université de Tlemcen. 2018/2019.

[23] Badr Benmammar, «Struts: Framework pour applications Web,». Module « construction d'applications réparties ». Master 2 réseaux et systèmes distribués. Département d'informatique, université de Tlemcen. 2018/2019.

Résumé:

À l'heure actuelle l'utilisation des applications web est devenue omniprésente dans notre vie quotidienne. En effet, ces applications occupent une place importante à cause de la facilité proposée pour effectuer certaines tâches. C'est ce qui a conduit ces applications à être exposées aux différents types d'attaques par des personnes ou des logiciels malveillants. L'objectif principal de notre travail consiste à réaliser une application web sécurisée pour garantir la confidentialité et la disponibilité de l'information. Une application e-commerce a été conçue en suivant le design pattern MVC, et réalisée avec les deux Frameworks Struts 2 et Hibernate en intégrant plusieurs mécanismes de sécurité tel que : l'authentification pour protéger le contenu, la gestion de rôles pour maitriser l'accès à certaines fonctionnalités et le cryptage symétrique des données sensibles avec AES.

Mots clés:

Application web, Struts 2, Hibernate, Java, Authentification, Gestion de rôles, Chiffrement.

Abstract:

Currently, the use of web applications has become ubiquitous in our daily lives; Indeed, these applications occupy an important place because of the ease offered to perform certain tasks. This has led these applications to be exposed to different types of attacks by people or malware. The main goal of our work is the realization of a secure web application to ensure the confidentiality and availability of information. An e-commerce application was designed following the MVC model, and realized with the two Frameworks Struts 2 and Hibernate integrating several security mechanisms such as: authentication to protect the content, role management to control access to certain features and symmetric encryption of sensitive data with AES.

Keywords:

Web Application, Struts 2, Hibernate, Java, Authentication, Role Management, Encryption.

ملخص:

في الوقت الحالي أصبح استخدام تطبيقات الويب في كل مكان في حياتنا اليومية ، تحتل هذه التطبيقات الآن مكانًا مهمًا بسبب السهولة التي توفرها لأداء مهام معينة .وقد أدى ذلك إلى تعرض هذه التطبيقات لأنواع مختلفة من الهجمات من قِبل الأشخاص أو البرامج الضارة .الهدف الرئيسي من عملنا هو تحقيق تطبيق ويب آمن لضمان سرية المعلومات وتوافرها .تم تصميم تطبيق للتجارة الإلكترونية وفقًا لنموذجMVC ، وتم تحقيقه من خلال دمج إطاري العمل Struts 2 و Hibernate في عدة آليات أمنية: المصادقة لحماية المحتوى، وإدارة الأدوار للتحكم في الوصول إلى بعض الميزات والتشفير المتماثل للبيانات الحساسة مع AES .

الكلمات المفتاحية:

تطبيقات الويب ، Struts 2 ، Hibernate ، Java ، التشفير المتناظر AES ، المصادقة ، تسيير الأدوار .