MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE
LA RECHERCHE SCIENTIFIQUE
UNIVERSITE ABDELHAMID IBN BADIS MOSTAGANEM

Faculté des Sciences Exactes & Informatique
Département d'Informatique

MEMOIRE DE FIN D'ETUDES
Pour l'Obtention du Diplôme de Master en Informatique
Option : Ingénierie des Systèmes d'Information

Thème

Vers un développement sécurisé d'une application Web pour l'ANEM

Présenté par :

- **ZERGUELAINE Nadia**
- **DJAAFAR Zahra**

Encadré par:

- **CHEHIDA Salim**

Année Universitaire 2011/ 2012

Résumé

La vie économique est aujourd'hui étroitement liée à Internet, ce dernier est le terrain d'une nouvelle délinquance qui se développe de plus en plus ces dernières années. La toile, riche de connaissances, d'échanges sociaux et commerciaux, est aussi devenue riche en vulnérabilités et certains n'ont pas perdu de temps pour exploiter ces failles nombreuses du Web à des fins malveillantes.

Afin d'assurer un niveau de sécurité acceptable pour une application Web, il convient donc aujourd'hui de ne pas s'arrêter à la sécurisation des couches réseau (filtrage, segmentation,…etc). Mais d'aller au-delà, notamment au niveau système (*Serveurs Web et SGBD*) et surtout au niveau conception et application.

Ce travail présente une étude de cas du système *e-ANEM (Application Web pour l'Agence Nationale d'Emploi*) qui a pour objectif d'ouvrir un portail vers le Web pour le demandeur d'emploi et l'employeur considérant les différents risques liés à cette ouverture. En effet Cette étude couvre toutes les phases de cycle de développement en utilisant des méthodologies et technologies récentes telles que : *UML, UMLsec, Java, JSF, Netbeans, Glassfish, et ORACLE*.

Sommaire

Liste des figures

Liste des tableaux

Introduction générale

L'informatique et en particulier l'Internet joue un rôle grandissant dans notre société. Ce dernier étant un espace ouvert ; il est possible pour n'importe quel internaute ou organisation, de mettre à disposition un propre site ou application web, le plus souvent à but commercial, de communication ou de promotion.

L'***A***gence ***N****ationale de l'****E****mploi*, (***A.N.E.M***) est un établissement national public qui a pour mission d'organiser le marché de l'emploi et de la main d'œuvre et d'assurer la coordination entre le demandeur d'emploi et les entreprises publiques et privées. Avec l'évolution du monde du travail et aux besoins continues des demandeurs d'emploi, l'*ANEM* à son tour nécessite une ouverture vers le Web. Les avantages de cette ouverture ne sont plus aujourd'hui à démontrer et l'adoption massive par les entreprises ou par les demandeurs de ces modes de diffusion de l'information est un fait indéniable. L'utilisation d'une application web pour l'*ANEM* offre des possibilités nouvelles et prometteuses, mais elle introduit également un certain nombre de risques dont nous devons prendre conscience, mesurer les conséquences éventuelles et, en connaissance de cause, prendre les mesures adéquates.

Dans les dernières années, la sécurité des applications web devient une problématique essentielle tant pour les individus que pour les entreprises ou les états. La plupart de ces applications sont victimes d'attaques en raison de la possibilité de les accéder à distance ce qui augmente énormément les utilisateurs et par la suite augmente les points d'attaques. En effet, la sécurité d'une application se pratique à tous les stades de sa vie : elle commence au moment de la conception, quand on met en place les choix stratégiques, elle est constante durant le développement, et enfin, elle se concrétise par la surveillance des opérations journalières lorsque l'application est mise en production. **[SG07]**

Notre travail s'intitule sur la conception et la réalisation de l'application web ***e-ANEM*** pour l'agence nationale d'emploi dont le but est d'assurer la communication entre le demandeur d'emploi et l'employeur en tenant compte des différentes exigences de sécurité. Ce travail aborde la nouvelle approche à priori de la sécurité qui propose d'intégrer les solutions de sécurité au niveau processus de développement.

Le standard ***UML*** offre un ensemble des mécanismes pour modéliser des vues multiples d'un système. La version ***UMLsec*** proposée par *Jan Jürjens* (Munich University of Technology) et d'autres extensions ont été utilisée pour la prise en charge de la sécurité dans la phase de conception.

Après la conception vient le développement, ***Java*** a été choisie comme le langage de développement où la sécurité fait partie intégrante du système d'exécution et du compilateur, ***NetBeans*** comme étant un excellent *IDE* moderne de développement, la technologie ***JSF*** qui facilite le développement des applications web dynamiques, ***Glassfish*** comme serveur d'application web et enfin ***Oracle,*** le meilleur *SGBDR* qui permet d'assure l'intégrité, la cohérence et la sécurité des données et qui répond bien aux différents besoins de volume, de distribution et d'exploitation.

Notre mémoire est structuré en trois chapitres :

Le premier chapitre présente les différents types d'attaques qui peuvent menacées les applications Web et les contre-mesures face à ces attaques.

Le second chapitre illustre la modélisation détaillée de notre système ***e-ANEM*** suivant une Démarche de Conception Sécurisée (***DCS*)**.

Le troisième chapitre donne une présentation générale des outils et technologies de développement utilisé ainsi qu'une description de l'application ***e-ANEM***.

Chapitre 1 :

Applications web, vulnérabilités et contre-mesures

1. Introduction

Le web étant en constante évolution ; de plus en plus de sites et d'applications web apparaissent chaque jour. De ce fait, ces applications, quand bien même protégées par une authentification, sont des cibles potentielles pour les pirates. C'est pourquoi il est indispensable de considérer les vulnérabilités lors de la publication d'une application sur la toile.

Ce chapitre présente les différentes attaques [1] utilisées pour voler des informations sur les applications web, ou encore modifier, voire détruire ces portails et les données qu'ils contiennent. Nous avons aussi présenté quelques contre-mesures [2] face à ces attaques.

2. Architecture des applications web

Une application web est l'ensemble des pages web, des fichiers, des données et des services web situés sur le serveur web de compagnie. Souvent, une application web est basée sur une architecture client-serveur 3-tiers qui comprend un client web, un serveur web qui contient les applications web, et un serveur de base de données.

Une application web est une application qui n'a besoin que du protocole *HTTP* pour être pilotée par un utilisateur. Celui-ci utilise un simple navigateur web ou bien une application propriétaire utilisant le protocole http.

[1] Action ou exploit entreprise pour nuire à une ressource.

[2] Mesure de protection qui contre une menace et atténue un risque.

Les données saisies par l'utilisateur sont envoyées à l'application web via les deux fonctions *POST* [3] et *GET* [4]. **[MYA12]**

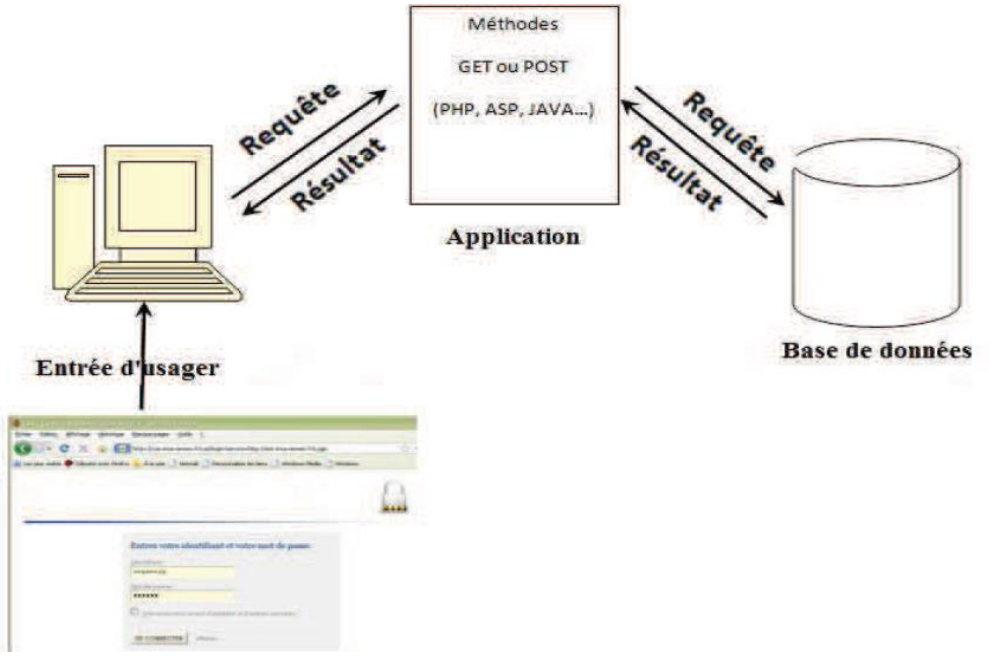

Figure 1 : Architecture d'une application web

3. Points d'attaques

L'avantage majeur d'une application web est qu'elle est utilisable à distance sans la nécessité d'installer ni de ne déployer aucun logiciel sur le poste client. Les visiteurs des applications web sont nombreux. Avec cette variété des utilisateurs et ses différents comportements, il est impossible de savoir qui sont légitimes ou non. Les visiteurs inconnus peuvent représenter un risque pour les applications web et donc altérer les informations, la qualité des traitements et les opérations qui y sont conduites.

Les différentes points d'attaques possible sur les applications web sont apparait dans la figure ci-dessous : **[PCH02]**

[3] POST : La valeur qui permet l'envoi de données stockées dans une requête

[4] GET : Permet l'envoi des données codées dans l'URL

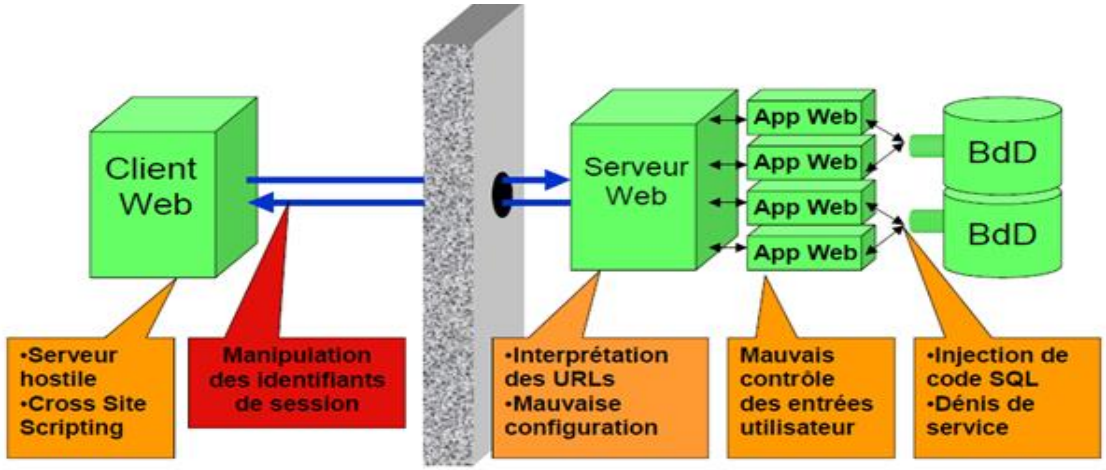

Figure 2 : Les vulnérabilités [5] d'une application Web

Parmi les objectifs des attaques :

- Atteinte à la confidentialité en récupérant les informations confidentielles (Secrets industriels, annonces commerciales, résultats, données clientes…)
- Atteinte à l'intégrité en modifiant le contenu (Résultats incorrects/incohérents, Comptes faussés, Défiguration, BDD corrompues)
- Atteinte à la disponibilité en rendant le service indisponible (Déni de Service – DoS) pour les utilisateurs légitime pendant une durée temporaire ou permanente (perte financière, force de travail bloquée,…). **[EJA08]**

4. Interprétation des URLs

Les *URLs* (Uniform Ressource Locator**)** utilisées dans le cadre d'une application web sont du type : ***http:// www.monserveur.com /chemin/ fichier.ext? param1=toto & param2=alaplage***.

Chacun des composants d'URL est susceptible d'être attaqué : **[CL03]**

- Par exemple, afin de désactiver une authentification par certificat client, l'attaquant va essayer de remplacer **le protocole (https://) par (http://)**.
- Afin d'avoir accès à d'autres parties du site, l'attaquant peut remplacer **l'adresse du serveur** (www.monserveur.com) par son adresse *IP* ou par les noms des domaines des autres sites hébergés sur le même serveur.

[5] Faiblesse qui rend possible une menace

- Comme il peut tenter de naviguer dans **l'arborescence** (chemin) pour accéder à des parties du site non autorisées ou pour remonter dans l'arborescence par l'utilisation de « /.../.../ ».
- L'extension du **Fichier** va déterminer de quel type d'exécutable il s'agit : *CGI, scripts ASP,* ou autre code exécutable, Plusieurs types de fichiers ont connu des vulnérabilités attachées à leur mode d'exécution.
- **Paramètres** : La manipulation des noms de paramètres et de leur contenu peut conduire à des effets dangereux.

Les conséquences liées aux ces attaques sont :

- la divulgation des informations importantes.
- l'accès au code source des scripts ou au contenu des fichiers.
- ou l'exécution de commandes sur le serveur.

Par exemple, URL suivante prend comme paramètre un fichier afin d'en afficher la taille : *http://www.maisjemesoigne.com/cgi-bin/getsize.cgi?fichier=test.txt*
En modifiant l'URL ainsi : *http://www.maisjemesoigne.com/cgi-bin/getsize.cgi?fichier=**, l'attaquant peut donc obtenir la liste des fichiers contenus dans le répertoire *cgi-bin* !

Pour contrer les attaques de ce type, il convient de prendre en compte les recommandations de sécurisation suivantes : **[PCH02]**

- Sécuriser le système d'exploitation et le serveur web (appliquer en particulier les derniers patches).

- Pour se prémunir des attaques par traversée de chemin, il faut établir une liste de fichiers utilisables et refuser tout autre fichier.
- Contrôler strictement l'arborescence web et supprimer les répertoires inutiles.
- Supprimer tous les filtres, interpréteurs de scripts, *CGI* [6] (*Common Gateway Interface)* et autres exécutables inutiles.
- Appliquer des permissions d'accès sur les fichiers au niveau du serveur web mais aussi du système de fichiers.
- Désactiver le protocole http sur les pages qui nécessitent *HTTPS*.
- Enfin, contrôler les entrées des utilisateurs au niveau des paramètres dans l'*URL*.

5. Mauvais Contrôle des données entrées par l'utilisateur

Les données saisies par l'utilisateur sont envoyées à l'application web via le protocole http en utilisant les fonctions *GET* et *POST*. Le plus souvent au niveau des applications web, le contrôle des entrées par l'utilisateur se fait uniquement côté client. Donc, il faut toujours prendre en considération qu'un attaquant pourra outre passer le contrôle côté client et donc envoyer les données qu'il désire au serveur.

Les entrées de l'utilisateur constituent les points d'attaques les plus utilisés par les hackers pour injecter leurs codes *SQL*. Les pare-feux et les systèmes de détection et de prévention d'instructions ne bloquent pas les attaques provenant des entrées de l'utilisateur. **[MYA12]**
Donc les recommandations nécessaires pour contrer la saisie de données hostile par un utilisateur sont les suivantes :

- Neutralisation des caractères spéciaux.
- Contrôle de la longueur des données.
- Validation du type des données (date, chaîne, nombre).
- Contrôle de l'intervalle de validité des données.
- Vérification de la validité réelle des données (en relatif, dans une base de données).

6. Injection SQL

SQL est un langage de requêtes destinés à interroger ou à manipuler une base de données relationnelle. Le principe des injections *SQL* consiste à manipuler l'application afin d'envoyer

[6] CGI : Permet l'exécution d'un programme extérieure dont la sortie standard d'affichage sera renvoyée au client par le serveur http.

une requête *SQL* non prévue. Un utilisateur malveillant pourra ainsi accéder, modifier ou supprimer des informations d'une base de données.

L'injection *SQL* peut être une conséquence directe d'un mauvais contrôle des données entrées par l'utilisateur. En effet, les caractères « ' » et « ; » peuvent être utilisés pour enchaîner plusieurs requêtes *SQL* à la suite. Cette faille de sécurité est rendue possible lorsque des données envoyées par un utilisateur sont utilisées sans traitement pour construire des requêtes *SQL*. **[RZO12]**

L'attaquant peut donc soumettre l'application à des injections *SQL* afin de passer l'authentification sans s'authentifier ou encore :

- Rendre inopérante l'application web.
- Voler les informations stockées en base de données.
- Usurpation d'identité.
- Altération des informations stockées dans la base de données. **[JSN09]**

Parmi les formes des injections *SQL* : tautologies, requête incorrecte, union de requêtes et requête "Piggy-backed".

6.1. Tautologies

Le but de hacker dans ce type d'attaque est de contourner l'authentification, de se connecter afin d'accéder aux données. Pour cet objectif le hacker va introduire le code *SQL* de manière que la requête demandée soit toujours satisfaisable.

Supposons par exemple un utilisateur identifié par le nom de «Ahmed» avec comme mot de passe « 123 ».

La requête d'interrogation de la base de données prendra la forme :

Select * from Users where UserName = ' Ahmed ' and UserPass = '123'

Maintenant imaginons que sur notre formulaire de saisie un utilisateur, voulant usurper l'identité de « Ahmed » va entrer les informations suivantes : Ahmed, 'or 1=1 --

La requête d'interrogation de la base de données prendra cette fois la forme :

Select * From Users Where UserName = ' Ahmed ' and UserPass = ' 'or 1=1 --'; **[LB11]** (où L'apostrophe indique la fin de la zone de frappe de l'utilisateur, le code "or 1=1" demande au script si 1=1 est vrai, or c'est toujours le cas, et -- indique le début d'un commentaire).

C'est à dire afficher les informations de (utilisateur dont le nom est Ahmed) et (le mot de passe est vide) ou (1=1).

La condition (1=1) est toujours vraie et le code qui suit les deux caractères de commentaire (--) ne s'exécute pas. Le Hacker donc a réussi à contourner l'authentification et d'accéder à l'application en tant Ahmed.

L'attaquant peut alors se connecter sous l'utilisateur Ahmed avec n'importe quel mot de passe. Il s'agit d'une injection de *SQL* réussie, car l'attaquant est parvenu à injecter les caractères qu'il voulait pour modifier le comportement de la requête.

6.2. Requête incorrecte

Dans le cas ou la gestion des erreurs ne sont pas réalisée correctement, le hacker peut déclencher des messages d'erreurs significatifs et par suite obtenir des informations confidentielles sur le système, le serveur et la structure de données.

Exemple : Si le hacker saisit dans le champ d'usager (') et dans le champ de mot de passe (123).

Le code *SQL* correspondant est:

Select * From Users Where UserName= '" and UserPass ='123' et le message d'erreur affiché est:

Msg 102, Level 15, State 1, Line 3; incorrect syntax near '123'.

Msg 105, Level 15, State 1, Line 3 ;Unclosed quotation mark after the character string "

Le Hacker donc a réussi de provoquer un message d'erreur significatif dans lequel il l'analyse et par la suite il peut connaître le *SGBD* de l'application web.

6.3. Union de requêtes

Ici le but de hacker est de réunir une requête avec la requête existante afin de retourner toujours des résultats.

Exemple : Le hacker peut saisir dans le champ d'usager ('UNION Select * From Users --) et dans le champ de mot de passe (123).Alors le code *SQL* correspondant est :

Select * From Users Where UserName= " UNION Select * From Users --' and UserPass ='123'

La première requête retourne nul (Select * From Users Where UserName= "), tandis que la seconde requête (UNION Select * From Users --' and UserPass ='123') renvoie les données de la table *Users*. Le hacker a réussie donc à retourner les informations confidentielles concernant les utilisateurs de l'application web. **[MYA12]**

6.4. Requête "Piggy-backed"

Dans ce type d'attaque le but de hacker est d'injecter une commande malveillante pour s'exécuter sur le serveur.

Exemple : Le hacker saisit dans le champ Nom de la table *Users* (' OR 1=1 ; Drop Table Users --) et dans le champ de mot de passe (123). Alors le code *SQL* est le suivant :

Select * From Users Where UserName ='' OR 1=1; Drop Table Users --'and UserPass ='123'

Si le contrôle d'accès autorise aux utilisateurs qui n'ont pas le droit d'exécuter toute sorte de requêtes, le hacker réussira à écraser la table (*Users*) à l'aide de la commande (*Drop Table*). Le hacker peut aussi mettre le serveur de base de données hors service en injectant la commande (*Shut Down*) au lieu de (*Drop Table*).

6.5. Les contre-mesures avancées de SQLIA

Les techniques simples comme la validation des entrées, le rejet des caractères illégaux, l'acceptation des caractères autorisés, le contrôle d'accès…sont insuffisantes pour contrer toutes les attaques de type *SQLIA* **(**Attaques par Injection *SQL***)**. Plusieurs chercheurs ont proposé des contre-mesures avancées de *SQLIA*.

6.5.1. Black box testing

Le principe de cette approche consiste à utiliser un analyseur web pour tester un programme de l'extérieur en vérifiant que les résultats obtenues sont bien celles qui sont prévues afin d'identifier tous les points d'attaque d'une application web qui peuvent être utilisées pour perpétrer une *SQLIA*, ensuite elle construit des attaques qui ciblent ces points sur la base d'une liste spécifiée de modèles. Le désavantage de cette technique est qu'elle ne peut pas fournir des garanties complètes et elle fournit un rapport des vulnérabilités sans proposer aucune solution.

6.5.2. SQLrand

Comme il indique son nom *SQLrand* (Jeu d'instructions du hasard), cette approche basée sur le jeu d'instructions randomisation. Il fournit un cadre qui permet aux développeurs de créer des requêtes en utilisant des instructions randomisées au lieu des mots clés *SQL* classiques. Un filtre par procuration intercepte les requêtes à la base de données et randomise les mots clés. Le *SQLIA* n'aurait pas été construit en utilisant le jeu d'instructions randomisé. Donc, les codes injectés auraient pour résultat une requête syntaxiquement incorrecte. **[MYA12]**

6.5.3. Analyse statique du code

C'est une approche qui se base sur l'analyse statique de code, citons en deux techniques :

6.5.3.1. JDBC-checker

Le *JDBC-checker* est un outil intégré dans *JDBC* qui permet de vérifier statiquement le type de requête de *SQL* dynamiquement produit. Cette technique peut être utilisée pour détecter et prévenir des attaques qui profitent des types d'erreurs générés par les requêtes dynamiques. Elle est aussi capable de découvrir une des causes de base des vulnérabilités.

6.5.3.2. Approche de Wassermann et Su

L'objectif de cette approche est de vérifier que les requêtes *SQL* produites dans la couche d'application ne peuvent pas contenir une tautologie, pour cela elle propose une utilisation de l'analyse statique combinée avec un raisonnement automatisé. **[MYA12]**

7. Attaques sur les identifiants de session

La plupart des applications web utilisent les protocoles déconnectés [7](*HTTP* ou *HTTPS*). Le serveur ne peut donc pas reconnaître un client qui a déjà commencé une transaction dans l'application web. Pour identifier les clients, l'application conserve des variables sur la machine du client .Comme le client contrôle le stockage des variables de serveur, un hacker pourrait modifier les contenus des variables pour perpétrer une attaque.

Une attaque classique consiste à voler la session d'un utilisateur qui vient de s'authentifier sur le système en essayant de deviner la valeur de son identifiant de session. Si la valeur de celui-ci est découverte, un attaquant peut alors se faire passer pour l'utilisateur légitime en injectant l'identifiant récupéré dans sa propre session, à l'aide d'un proxy intrusif par exemple.

Pour remédier à cela, on utilise un identifiant de session, échangé à chaque page entre le client et le serveur. Le serveur maintient un contexte de transaction pour chaque identifiant de session généré. **[CL03]**

8. Attaque XSS (Cross-Site Scripting)

L'une de ses caractéristiques est de mettre en œuvre une relation tri parties : le pirate, la cible ainsi qu'une troisième partie le piège. Lors d'une attaque *XSS* (*Cross-Site Scripting*), un attaquant force le navigateur d'un utilisateur à exécuter du code fourni par l'attaquant. Un

[7] Entre deux requêtes la connexion entre le client et le serveur est coupée

attaquant insère un bout de code malicieux dans une partie du contenu d'une application web, par exemple dans un commentaire sur un blog. Lorsqu'un utilisateur normal va accéder à la page, et que le blog en question n'a pris aucune mesure pour empêcher ce genre d'attaque, le code malveillant va être exécuté du côté client et va pouvoir accéder potentiellement à des informations sensibles conservées par le blog sur l'utilisateur, comme les valeurs des cookies par exemple. **[LHE10]**

A titre d'exemple, imaginons un site web offrant un simple forum de discussion. Le pirate va tenter de détourner ce site en un piège pour les internautes. Utilisant le formulaire du *Forum*, il va publier un message au titre normal et attractif, mais dont le contenu contiendra également un code caché. Avec la demande de lecture de ce message, le navigateur Internet du visiteur, va recevoir non seulement le texte mais aussi le code inclus dans le message et donc l'exécuter.

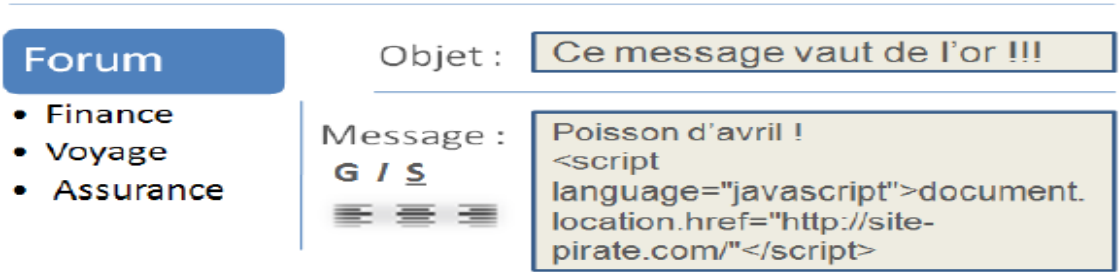

Figure 3 : Exemple d'attaque de type XSS

Cette technique permet au pirate de faire exécuter, à l'insu de l'internaute, différentes actions par son navigateur. Il peut s'agir de choses anodines, comme l'affichage d'un message parasite, ou d'actions beaucoup plus graves telles qu'une redirection (exemple ci-dessus) ou pire encore avec l'ouverture cachée d'une connexion vers le site du pirate permettant à celui-ci d'installer un *cheval de Troie* sur le poste de l'internaute afin d'y dérober des infirmations confidentielles, telles que ses codes d'accès bancaires. **[LB11]**

Les risques liés aux attaques de ce type sont :

- Affichage de contenu subversif pouvant nuire à l'image de marque de l'entreprise propriétaire du site.
- Redirection vers un autre site pour tromper l'utilisateur, récupérer des informations personnelles, …
- Vol de session ou de cookies pouvant mener à l'usurpation d'identité. **[JSN09]**

La vulnérabilité, qui rend l'attaque possible, réside dans le fait que l'application ne contrôle pas le format des données entrées par l'utilisateur. A chaque fois que l'internaute peut

saisir une donnée, la question du *Cross Site Scripting* doit être systématiquement vérifiée. Chaque zone de saisie, chaque paramètre d'*URL*, doivent être contrôlés soigneusement afin de sécuriser au mieux le site web. **[LB11]**

9. Conclusion

Nous avons vu que les attaques qu'elles sont possibles de mener contre une application web sont considérables. Il est donc extrêmement important de prendre en compte la sécurité le plus en amont possible lors du développement d'une application web. L'idéal est de considérer les contraintes sécuritaires dès la conception de l'application.

Chapitre 2 :

Conception du système e-ANEM

1. Introduction

L'étude que nous considérons dans ce projet porte sur l'***A**gence **N**ationale de l'**E**mploi*, (***A.N.E.M***) a pour mission d'organiser et d'assurer la connaissance de la situation et de l'évolution du marché national de l'emploi et de la main d'œuvre et de garantir à tout demandeur et à toute entreprise un service de recrutement efficace et personnalisé.

Cette étude repose au départ au différents sciences de travail et de consultation que nous avons organisés avec les responsables des différents services , dans le but de réunir toutes les informations nécessaires pour mieux s'informer, comprendre et assimiler les différentes tâches effectuées au niveau de l'***A.N.E.M***.

La conception de notre système ***e-ANEM*** est fondée sur *UML* (*Unified Modeling Language)* suivant une démarche basée sur les principes d'un processus *UP*[1] avec la prise en charge des contraintes de sécurité durant les différentes phases de développement.

2. Démarche et outil

2.1. DCS

***R**ational **U**nified **P**rocess* (**RUP**) est un processus du génie logiciel développé par la société *Rational Software* et basé sur le langage *UML*. Ce processus a comme objectif principal de faire appliquer les bonnes pratiques tout au long du cycle de développement du logiciel, ce

[1] UP (Unified Process) est un processus de développement logiciel « itératif et incrémental, centré sur l'architecture, conduit par les cas d'utilisation et piloté par les risques »

qui confère au produit final une meilleure qualité. La gestion d'un tel processus est organisée selon les quatre phases suivantes: Pré étude, élaboration, construction et transition.

Selon les principes du processus *RUP* et les besoins de notre système, nous avons adapté une Démarche de Conception Sécurisée (***DCS*)** définie par une séquence d'étapes, en partie ordonnées, qui concourent à produire un système logiciel de qualité qui répond aux besoins fonctionnels des utilisateurs ainsi que les différentes exigences de sécurité afin de minimiser les risques liées à l'ouverture du système *e-ANEM* à l'extérieur.

La démarche *DCS* est organisée autour des quatre phases suivantes:

- Expression et spécification des besoins
- Analyse et conception
- Modélisation de la navigation
- Conception technique

L'ensemble des phases de développement sera illustré et détaillé au long de la mise en application de la démarche *DCS* à l'étude de notre système *e-ANEM***.**

2.2. Edraw UML Diagram

Edraw UML Diagram est un outil graphique qui permet de modéliser des Systems d'information ou tout autre système en utilisant le langage *UML*. Il met à disposition beaucoup des fonctionnalités qui permettent de servir de manière optimale, des modèles du langage *UML*.

Edraw UML Diagram est un outil complet pour les concepteurs de software ; il permet d'illustrer avec des diagrammes les flux de données, les relations existantes entre les applications, actions qui sont en cours et connexions entre les différents éléments. Il permet aussi de concevoir, dessiner et créer les différents modèles et diagrammes.

Parmi les avantages de l'outil *Edraw UML Diagram*

- Capacité à apprendre rapidement et de tracer des diagrammes *UML* pour la présentation.
- Utilise pleinement l'interface utilisateur de Windows *XP*.
- Prise en charge de nombreux autres schémas de conception de logiciels, en plus d'*UML*.
- Mise à jour gratuit pour jamais, appui libre de technologie. **[WEB01]**

3. Expression et spécification des besoins

Cette phase a pour but d'étudier les fonctionnalités du système selon les besoins des différents utilisateurs, ainsi que la spécification des exigences de sécurité induites par l'utilisation du système. Elle consiste aussi à partager le système en packages métiers.

3.1. Recueil initial des besoins

Cette étape consiste à effectuer un premier repérage des besoins fonctionnels et de sécurité en utilisant le texte pour définir le cahier des charges (fonctionnel et de sécurité), et des diagrammes de collaboration pour visualiser le contexte du système.

3.1.1. Cahier des charges

3.1.1.1. Objectifs

A.N.E.M (***A****gence* ***N****ationale de l'****E****mploi*) est un établissement public à gestion spécifique régit par les dispositions du décret exécutif n° 06-77 du 19 Moharrem 1427 correspondant au 18 février 2006. Elle est dotée de la personnalité morale et jouit de l'autonomie financière, l'agence est placée sous la tutelle du ministère chargé du travail [**DEX06**], Leur activité principale est le placement des jeunes.

Parmi les objectifs principaux du système *e-ANEM :*

- Assurer la communication et la coordination entre le demandeur d'emploi et l'employeur.
- Assurer le suivi des demandes d'emploi, des offres et des contrats par les responsables de l'agence.

3.1.1.2. Acteurs du système

Un acteur représente un rôle joué par une entité externe (utilisateur humain, dispositif matériel ou autre système) qui interagit directement avec le système étudié. Un acteur peut consulter et/ou modifier directement l'état du système, en émettant et/ou en recevant des messages susceptibles d'être porteurs de données. **[PRO08]**

Les acteurs de notre système sont les suivants :

- **Demandeurs d'emploi :** la personne qui visite le site de l'*A.N.E.M* pour préinscrire afin de rechercher un emploi, Il s'agit bien sûr de l'acteur le plus important, celui pour lequel le site existe.
- **Conseiller du service demandeurs :** Celui ayant le rôle de compléter l'inscription et gère les informations des demandeurs.
- **Employeur :** la personne qui visite le site de l'*A.N.E.M* pour inscrire et déposer ses offres avec les critères recherchés pour les futurs employés.

- **Conseiller du service offres :** celui qui gère les informations des employeurs, il s'occupe aussi du suivi des inscriptions des offres.

3.1.1.3. Fonctions du système

- **Service demandeur:**

Le demandeur ramène son dossier qui est généralement composé d'une carte nationale, ses diplômes ainsi que d'éventuelles attestations de ses précédentes expériences professionnelles.

Une fois que le dossier soit déposé et vérifié, Le conseiller du service demandeur se charge de compléter l'inscription du demandeur (Niveau de qualification, groupe de métiers, ..) et vérifier les informations que ce dernier a déjà mentionnées dans son formulaire d'inscription. Ensuite, le conseiller peut effectuer des recherches plus approfondies sur les profils des demandeurs.

- **Service offre d'emploi:**

Lors de son inscription, l'employeur remplit une fiche d'identification de l'entreprise, ensuite il l'a ramène au service d'offre d'emploi. Eventuellement, le conseiller peut aussi mener des recherches sur les employeurs, choisir l'offre et sélectionner les demandeurs dont le profil correspond le mieux aux offres comptabilisées. Le conseiller de ce service crée aussi un bulletin pour le demandeur.

Le futur système doit faire le rapprochement ; il sélectionne la liste des demandeurs satisfaisant les conditions de l'offre. Lorsque le demandeur est convoqué par le service d'offre d'emploi, il se dirige vers l'entreprise avec le bulletin qui sera ensuite remplis par l'entreprise et renvoyé à l'agence. Le conseiller mentionne après sur l'attestation d'inscription du demandeur que ce dernier occupe un poste.

3.1.1.4. Les Exigences de sécurité

Une fois les données seront publié via l'internet, la possibilité de les attaquées est très élevée. Donc les données sensibles, tel que le mot de passe par exemple sont en tête du risque.

Le système *e-ANEM* doit répondre à certaines exigences de sécurité :

- Assurer aux demandeurs et employeurs qu'ils sont sur le site *e-ANEM* et non pas sur un faux site.
- Assurer aux demandeurs et employeurs que leurs informations personnelles ne sont pas accessibles à des malfaiteurs, lorsqu'ils procèdent à une transaction sur le site.

- Permettre aux demandeurs d'emploi et l'employeur d'inscrire facilement et d'une manière sécurisée.
- Le Système *e-ANEM* doit permettre aux demandeurs d'emploi de consulter leur attestation en toute sécurité.

Ces contraintes permettent de déterminer les principaux objectifs en termes de sécurité pour le système *e-ANEM* :

- Authentification : le système doit assurer l'authentification des demandeurs et des employeurs. Les demandeurs et les employeurs doivent être en mesure de s'assurer qu'ils sont devant le site *e-ANEM* et qu'ils envoient leurs informations privées à une entité réelle et non pas un *spoof* site.
- Confidentialité : la confidentialité des données sensibles doit être assurée par le système tel que les informations personnelles des demandeurs et des employeurs.
- Intégrité des données : les informations des demandeurs et des employeurs ne peuvent pas être modifiées que par les personnes autorisées
- Non répudiation : le responsable demandes ou le responsable offres doit assurer que l'autre partie (demandeurs, employeurs) engagées dans une transaction ne puisse nier qu'une transaction a en lieu.

3.1.2. Modélisation du contexte

Cette étape consiste à élaborer le modèle de contexte fonctionnel et de sécurité en basant sur le cahier des charges. Le diagramme de contexte est représenté par un objet central considéré comme étant une boite noir entouré par un ensemble des objets définissant les différents acteurs sans aucune interaction entre eux.

Les objets sont relies avec le système par des liens .Chaque lien support des messages en sortie de système pour représenter les différents services fonctionnels et de sécurité assurés par le système.

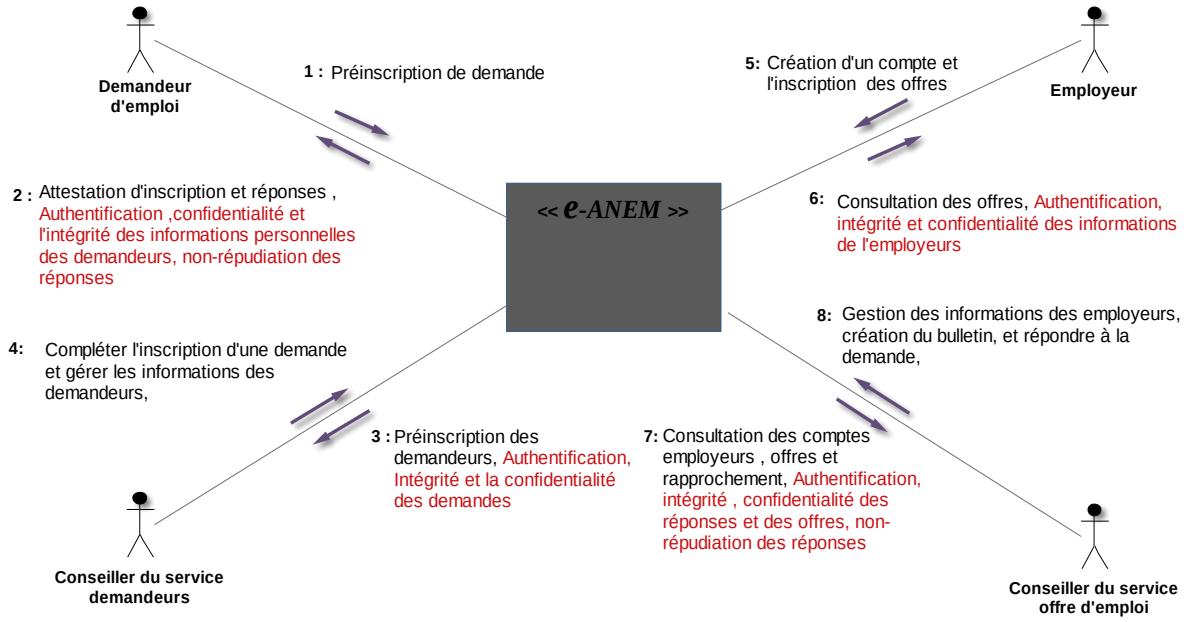

Figure 4 : Modèle de contexte du système «*e-ANEM* »

3.1.3. Découpage en catégorie :

Pour faciliter la conception, nous avons découpé le système en deux catégories, une catégorie de demande, et une autre pour l'offre.

- **Catégorie demande :** s'occupe de tous ce qui a une relation avec la demande d'emploi, les tâches des demandeurs et de conseiller du service demandeurs.
- **Catégorie Offre :** elle s'occupe de tous ce qui a une relation avec l'offre, l'employeur (les comptes et les offres) et le conseille du service d'offre d'emploi.

3.2. Identification et description des cas d'utilisation et des cas de sécurité

Pour chaque acteur identifié précédemment, il s'agit de rechercher les différentes intentions « métier » selon lesquelles il utilise le système.

3.2.1. Identification

Un cas d'utilisation (*use case*) représente un ensemble de séquences d'actions qui sont réalisées par le système et qui produisent un résultat observable intéressant pour un acteur particulier. Un cas d'utilisation modélise un service rendu par le système. Il exprime les interactions acteurs/système et apporte une valeur ajoutée « notable » à l'acteur concerné. **[PRO08]**

Le tableau suivant présente les différents cas d'utilisation de notre système ainsi que leurs acteurs et catégorie.

Cas d'utilisation	Acteur	Catégorie
➢ Préinscrire de demande ➢ Consulter l'attestation d'inscription et réponses	Demandeurs d'emploi	Demande
➢ Compléter l'inscription d'une demande ➢ Gérer les informations des demandeurs	Conseiller du service demandeurs	Demande
➢ Créer compte ➢ Inscrire offre ➢ Consulter les offres	Employeur	Offre
➢ Gérer les informations des employeurs ➢ Créer bulletin ➢ Répondre à la demande	Conseiller du service offre d'emploi	Offre

Tableau 1 : Liste des cas d'utilisation

Un cas de sécurité (*Security case*) représente un service de sécurité rendu par le système pour un ou plusieurs acteurs. Il permet de spécifier un comportement attendu du système pour répondre à des exigences de sécurité sans imposer le mode de réalisation de ce comportement.**[Mem02]** Le tableau 2 présente les cas de sécurité et leurs acteurs et catégories.

Cas de sécurité	Acteur	Catégorie
➢ Assurer l'authentification	Demandeur, Employeur, Conseiller du service demandeurs, Conseiller du service offre d'emploi	Demande, Offre
➢ Assurer la non-répudiation des réponses	Demandeur, Conseiller du service offres d'emploi	Demande, Offre
➢ Assurer la confidentialité et l'intégrité des informations personnelles des demandeurs	Demandeur	Demande
➢ Assurer la confidentialité et intégrité des informations des employeurs	Employeur	Offre

➢ Assurer l'intégrité et la confidentialité des réponses et des offres	Conseiller du service offres emploi.	Offre
➢ Assurer l'intégrité et la confidentialité des demandes.	Demandeur, Conseiller du service demandeurs	Demande, offre

Tableau 2 : Liste des cas de sécurité

3.2.2. Description textuelle

Dans cette étapes on va présenter une description textuelle de chaque cas d'utilisation afin d'avoir une idée plus détaillée sur le fonctionnement de chacun d'eux. La description est structurée comme suit :

- Titre du cas d'utilisation
- Pré-conditions
- Scénario nominal
- Exceptions
- Post-conditions

➢ **Cas d'utilisation « Préinscrire de demande»**

Pré-condition : L'acteur doit s'authentifier.

Acteur : demandeur.

Scénario nominal :

- Le demandeur doit inscrire en saisissant ses informations personnels comme : son nom, prénom, civilité, date de naissance, lieu de naissance, Adresse, situation familiale, nombre enfants, service nationale…etc. Il doit introduire aussi les informations de sa pièce d'identité (la carte nationale) : numéro de la carte, date de délivrance, la commune et wilaya, les informations sur son expérience s'il a bien sur : poste occupé, nombre d'année expérience, les informations sur la scolarité le niveau d'instruction et le diplôme.
- Le système doit vérifier la validité des données et afficher des messages d'erreur s'ils sont invalides.
- Après la préinscription du demandeur, le système envoie les informations au service des demandeurs.
- Pour que la préinscription soit validé, le demandeur doit se présenter à l'Agence

(*A.L.E.M*) ***A****gence* ***L****ocale de* ***l'E****mploi* concernée avec son ou ses diplômes, sa carte d'identité nationale ainsi que les attestations de son expérience professionnelle s'il a déjà travaillé, pour compléter sa préinscription.

Exception :

- Si le demandeur dépose son dossier, l'inscription devient valide, Sinon elle est annulée.

Post-condition :

- Compléter inscription

➢ **Cas d'utilisation « Créer compte » et « Inscrire offre»**

Pré-condition : L'acteur doit s'authentifier

Acteur : Employeur

Scénario nominal :

- Pour que l'employeur puisse créer un compte, il doit s'inscrire et remplir la fiche d'identification, saisir son raison sociale, secteur juridique, activité principale, activité secondaire, statut juridique, secteur d'activité, branche d'activité, adresse, wilaya, commune ... etc.
- Après l'inscription de l'employeur, le système envoie les informations au service d'offre d'emploi.
- Pour que l'inscription soit validée, l'employeur doit se présenter à l'Agence ***A.L.E.M*** concernée avec fiche d'indentification d'entreprise.
- Après la création de compte, il peut créer une offre. Pour le faire, il doit saisir les informations suivantes : poste proposé, nombre de poste, type de contrat ... etc.
- Le système doit vérifier la validité des données et afficher des messages d'erreurs s'ils sont invalides.

Exception :

- Si l'employeur dépose son dossier, l'inscription devient valide, sinon elle est annulée.

Post-condition :

- L'employeur consulte les offres.

- **Cas d'utilisation « Répondre à offre»**

Pré-condition : L'acteur doit s'authentifier.

Acteur : Conseiller du service offre d'emploi.

Scénario nominal :

- Le conseiller du service introduit l'identifiant et mot passe au début.
- Le système vérifie l'identifiant de connexion et le mot passe saisis par le conseiller, s'ils sont valides, donc le système fait l'ouverture de l'espace, sinon le système affiche un message d'erreur.
- Le conseiller du service construit les conditions de l'offre et le système fait le rapprochement suivant les conditions et affiche la liste des demandeurs.
- Le conseiller du service d'offre d'emploi sélectionne des candidats à partir de la liste des demandeurs, ensuite il crée un bulletin à chaque demandeur sélectionné.
- Le conseiller du service offre valide la réponse et le système envoie le bulletin au demandeur.

Exception :

- L'identifiant ou mot passe invalide : le conseiller du service doit réintroduire à nouveau l'identifiant et mot passe.

3.2.3. Description graphique

La description graphique des cas d'utilisation et des cas de sécurité consiste à documenter ces cas en utilisant les diagrammes dynamiques et à présenter les interactions critiques. Dans cette partie, on présente la description des scénarios :

- Répondre à l'offre en utilisant le stéréotype d'*UMLsec* « *provable* ».
- Compléter inscription en utilisant le stéréotype d'*UMLsec* « *fair exchange* ».
- Création d'un compte et d'une offre en utilisant *le modèle des scénarios critiques.*

A) Répondre à l'offre

Le stéréotype « *provable* » permet d'assurer la non répudiation dans les transactions de e-commerce, il garantie que si une action est exécuté, elle ne peut pas être niée. Un S/système peut être marqué prouvable avec l'utilisation des étiquettes *action* et *cert.* **[JJU04]**

Pour le cas d'un diagramme d'activité : Un état d'activité contenu dans le tag *cert* {Réception du bulletin de demandeur, Réception du bulletin de l'employeur} n'est atteint que si l'état d'activité contenu dans le tag *action* {Valider les réponses} est atteint avant lui.

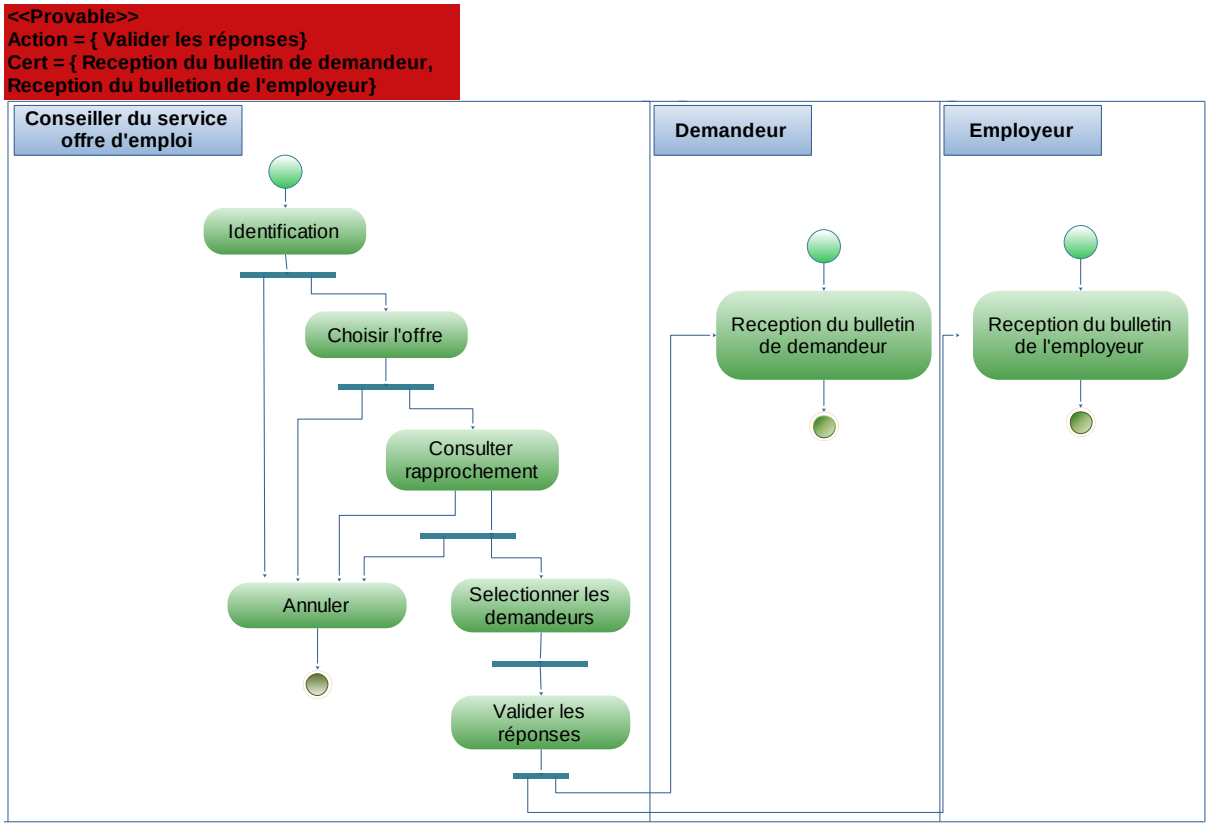

Figure 5 : Diagramme d'activité pour le cas utilisation « *Répondre à l'offre* »

B) Compléter inscription

Le stéréotype « *fair exchange* » définit deux étiquettes {*start*} et {*stop*}. Pour tous scénarios d'un diagramme d'activité : si un état d'activité contenu dans le tag {*start*} est exécuté par un scénario donc l'état d'activité contenu dans le tag {*stop*} sera aussi exécuté. **[JJU04]**

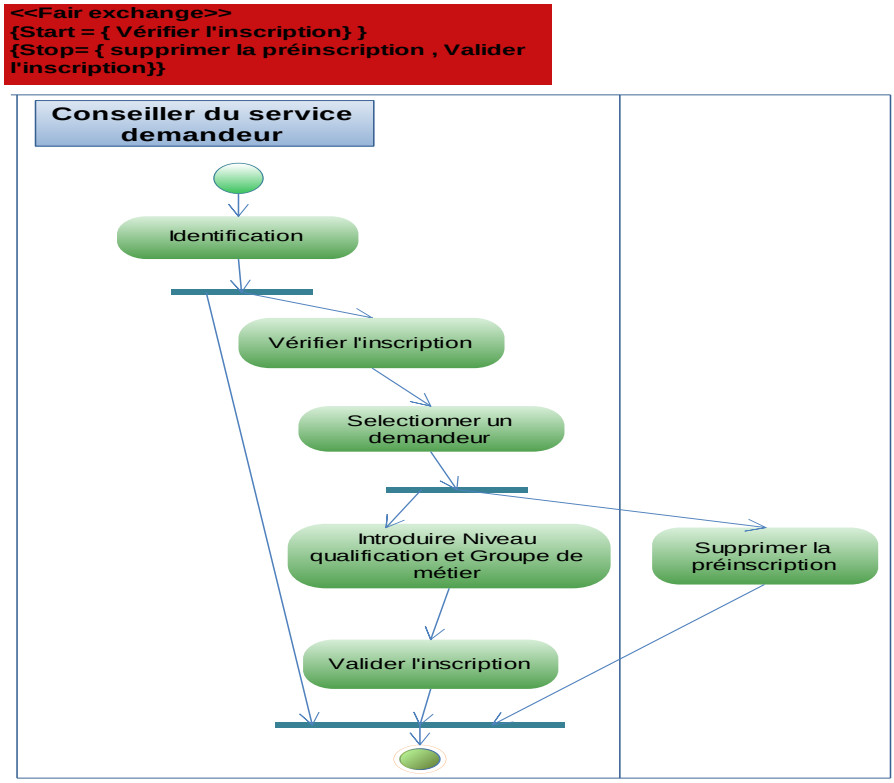

Figure 6 : Diagramme d'activité pour le cas d'utilisation « *Compléter inscription* »

C) Création d'un compte et d'une offre

Dans le modèle des scénarios critiques, les contraintes *secrecy*[2] et *integrity*[3] sont utilisés successivement pour assurer la confidentialité et l'intégrité des interactions entre le système et ses acteurs et la contrainte *identity*[4] pour assurer l'identité des parties lors de l'exécution d'une action d'interaction entre un acteur et le système. **[Mem01]**

[2] {Integrity}: pour assurer l'intégrité des interactions

[3] {Secrecy}: pour assurer la confidentialité des interactions

[4] {Identity}: pour assurer l'identité des parties lors de l'exécution d'une action d'interactions entre un acteur et le système.

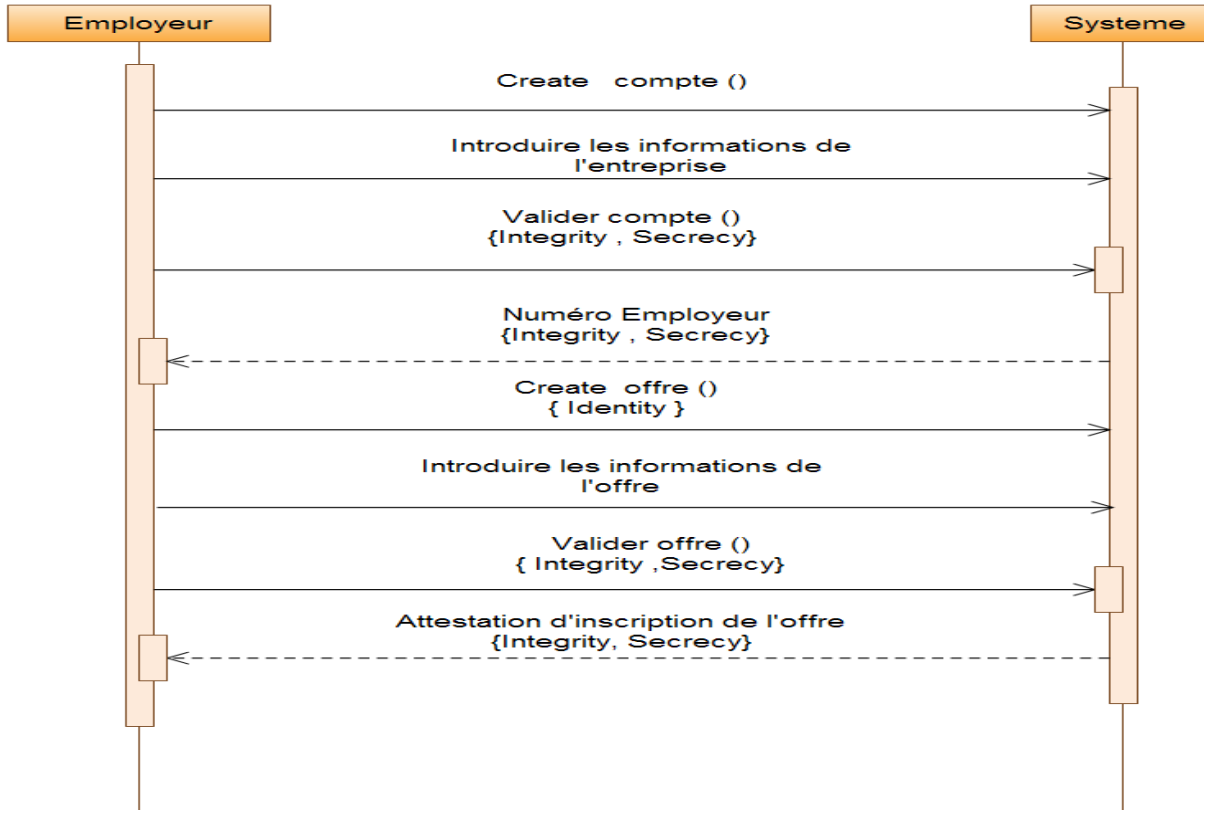

Figure 7 : Diagramme de séquence pour le cas utilisation « *Créer un compte et une offre* »

3.2.4. Représentation

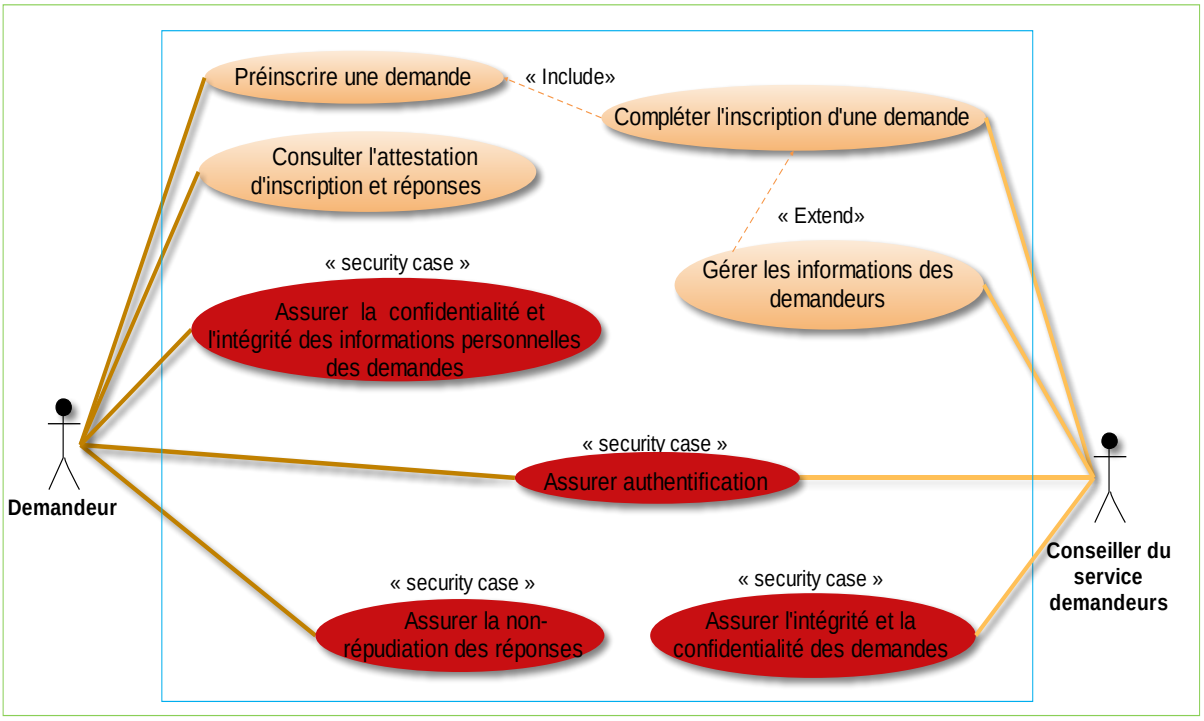

Figure 8 : Diagramme des cas d'utilisation et de sécurité pour la Catégorie « *Demande* »

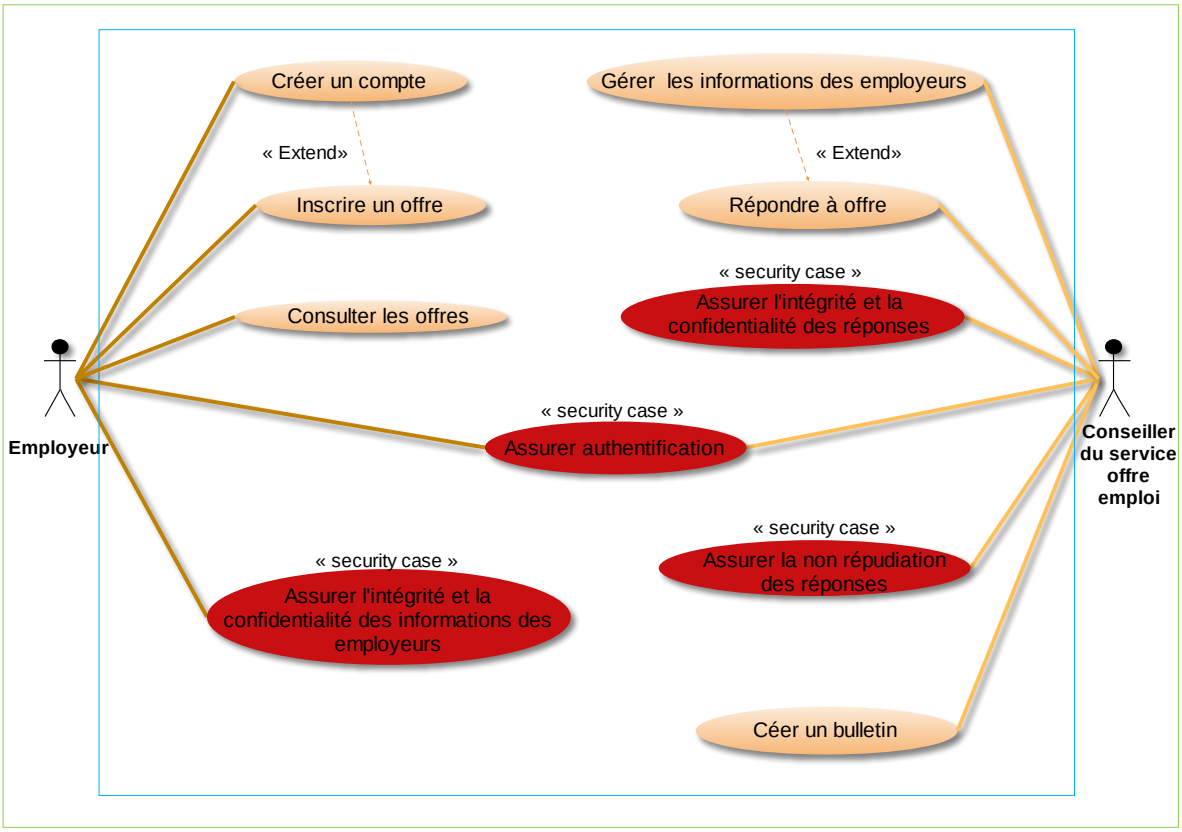

Figure 9: Diagramme des cas d'utilisation et de sécurité pour la Catégorie « *Offre* »

4. Analyse et conception

L'objectif de cette phase est de déterminer les objets et les classes du système à construire, ainsi que leur structure et leurs relations .Chaque objet doit décrire selon deux axes : axe statique (Structure de l'objet) et axe dynamique (Etats et messages de l'objet). **[WEB02]**

4.1. Identification des classes candidates

Il s'agit de décomposer le domaine étudié en classes conceptuelles représentant les entités significatives de domaine à partir de la spécification de chaque cas d'utilisation.

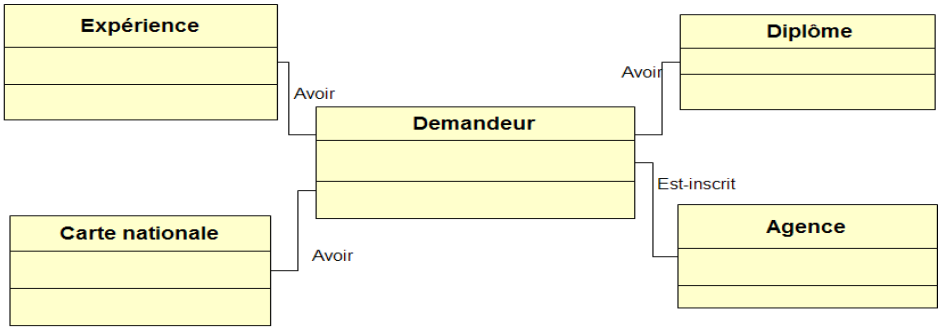

Figure 10 : Diagramme des classes participantes pour le cas d'utilisation « *Préinscrire de demande* »

4.2. Analyse dynamique

Cette phase consiste à représenter les interactions entre les objets qui peuvent être décrites au moyen de deux types de diagrammes : le diagramme de séquence et le diagramme de collaboration. Dans cette étape, nous avons montré les propriétés de sécurité sur les interactions du système.

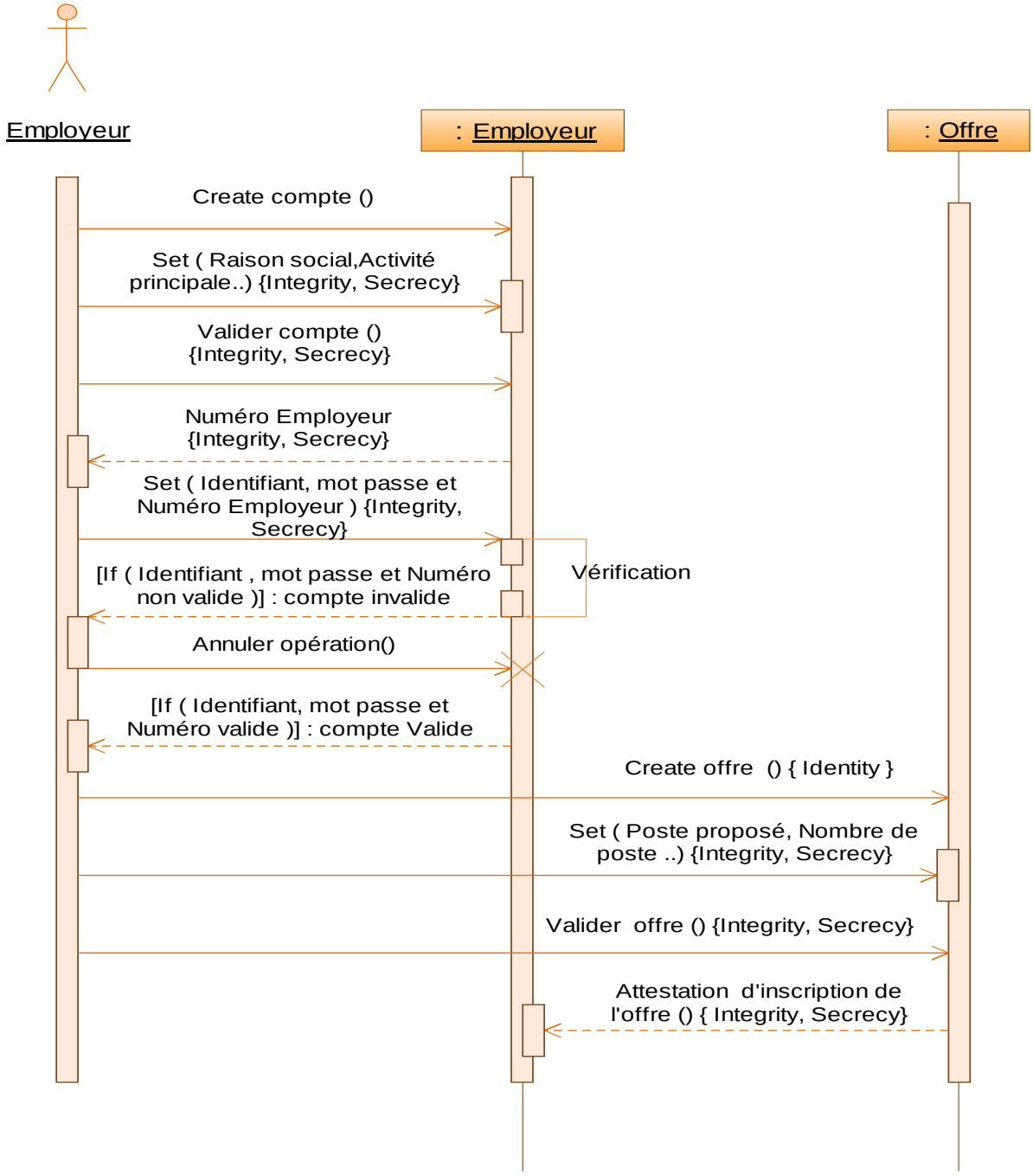

Figure 11 : Diagramme des interactions d'objets sécurisées pour le scénario « *Créer compte et inscrire offre* »

4.3. Diagramme de classes finales

Cette étape consiste à mettre à jour les différentes classes du système en profitant de l'analyse réalisée avec les diagrammes dynamiques. La figure suivante présente le diagramme de classe du système *e-ANEM*.

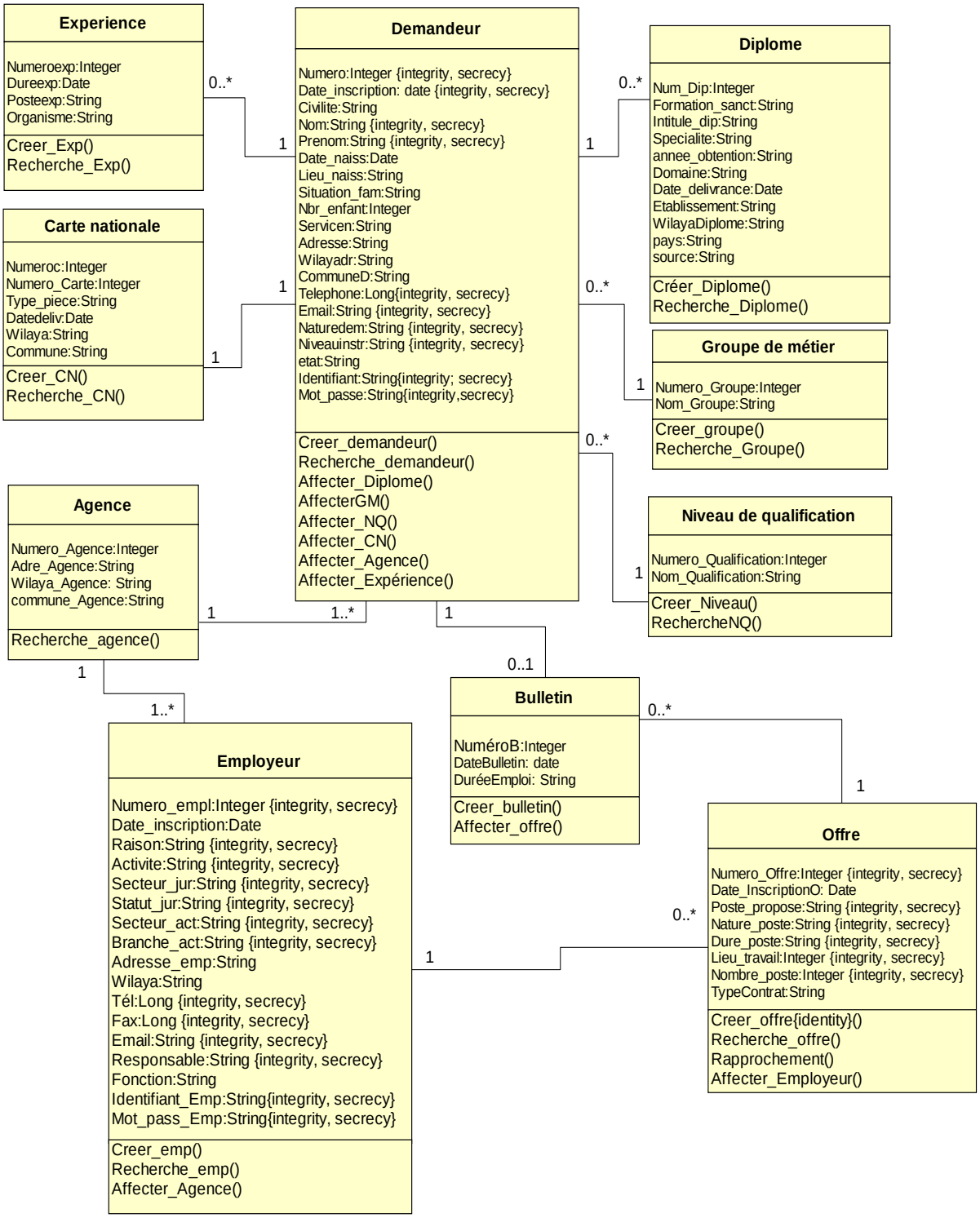

Figure 12 : Diagramme des classes finales du système *e-ANEM*

5. Modélisation de la navigation

Dans cette étape, les diagrammes d'activité sont utilisés pour modéliser avec précision la navigation dans l'application web. On utilise des éléments standards (activités, transitions avec ou sans conditions, branchements,...) mais aussi des conventions propres à la navigation dans un site web.

Les conventions graphiques suivantes seront utilisées :

- «page» pour une page complète du site.
- «action» pour une action simple (ex : classement d'une liste de courriers).
- «exception» pour une erreur ou un comportement inattendu du système.

La figure suite présente le modèle de la navigation pour le demandeur d'emploi.

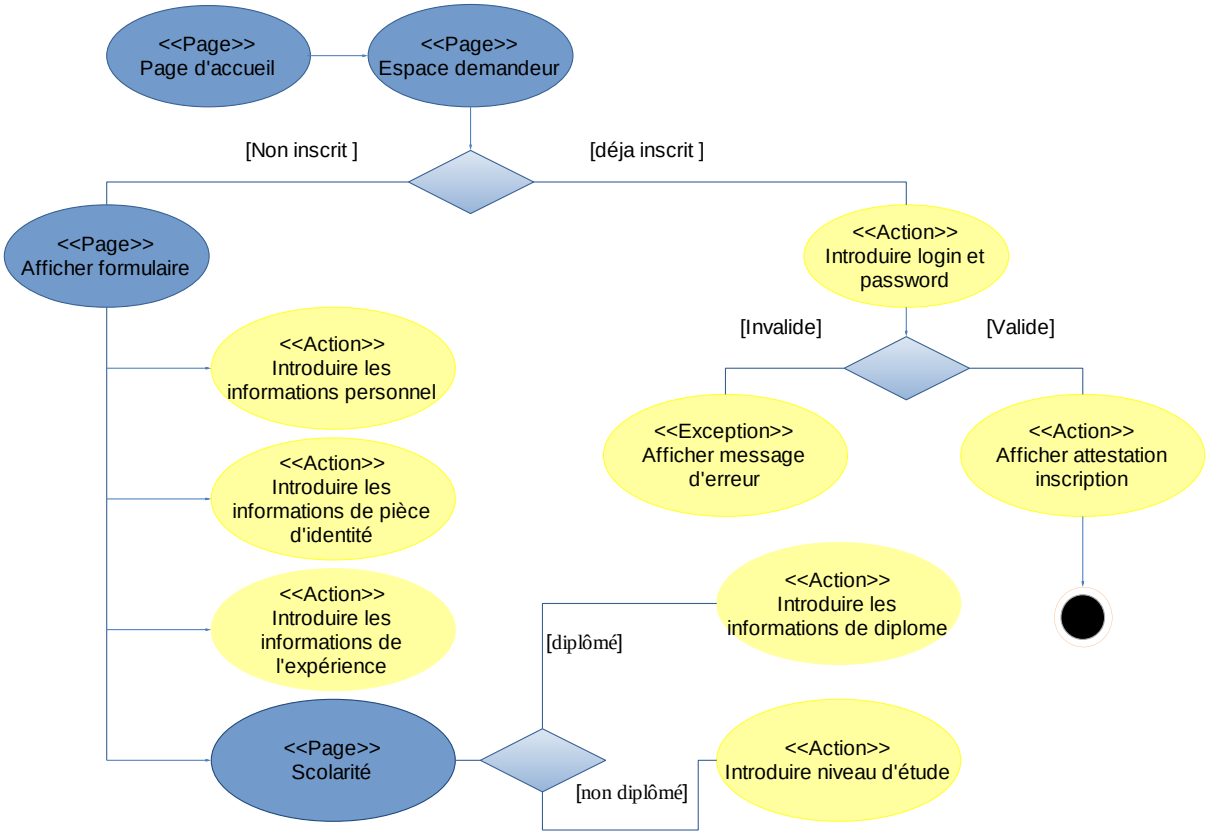

Figure 13: Diagramme d'activité pour modéliser la navigation du système *e-ANEM*

6. Conception technique

Elle consiste à apporter l'architecture technique et la configuration matérielle du système en utilisant le diagramme de composant et le diagramme de déploiement avec la prise en charge des exigences de sécurité. **[OBO12]**

6.1 Composants d'exploitation

Un diagramme de composants est un diagramme permet de représenter l'organisation et les dépendances liant les éléments logiciels d'un système. En effet, ce diagramme propose une vision statique de l'organisation des éléments et il montre les dépendances existant entre les composants.**[YEL06]**

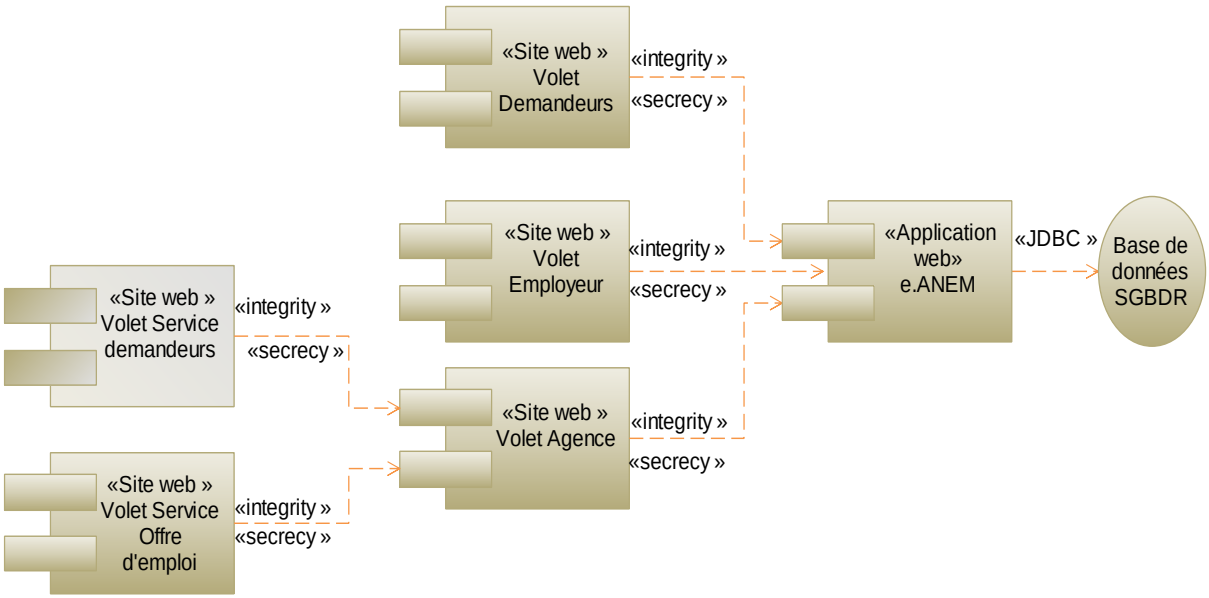

Figure 14 : Diagramme de composant du système *e-ANEM*

6.2 Configuration matérielle sécurisée

Pour assurer une bonne protection des données d'une entreprise, différents outils sont disponibles (firewall, pare-feu applicatif, *DMZ*…). Ils ont en général utilisés ensemble, de façon à sécuriser les différentes failles existantes dans un système.

Pour notre système nous proposant à utiliser :

- Un pare-feu applicatif (Web Application Firewall), en mode reverse-proxy, qui permet de : **[CLU11]**
  - Ploquer des attaques en implémentant des filtres au niveau des données transportées par *HTTP/HTTPS*.
  - Limiter l'exposition à des vulnérabilités détectées dans la logique applicative en attendant une correction en profondeur (par exemple modification du comportement prévu de l'application web).
  - Authentifier et/ou autoriser l'accès à tout ou partie de l'application.
  - Tracer pour apporter de la visibilité supplémentaire sur l'environnement applicatif (facilité l'investigation, apporter des preuves de tentatives d'intrusions,….).

- L'utilisation de mode reverse-proxy permet d'assurer une protection contre : **[PCH03]**
  - Les attaques déjà connues (fonction type *IDS* [5] à l'aide de signatures).
  - Les attaques par saisies hostiles dans les formulaires web.
  - Les attaques par l'utilisation de méthodes *HTTP* particulières (*OPTIONS*, *PUT*,…).
  - Les attaques par le protocole *http*.
  - Les attaques par déni de service.
  - Les attaques par modification du contexte de transaction (champs cachés, champs pré-saisie, cookies,…).
  - Les attaques de type injection de code *SQL*.

En plus le choix du déploiement en reverse-proxy (intelligent) est que le *WAF* [6] peut être mis dans une *DMZ* [7] publique alors que le reste de l'infrastructure web est localisé dans une *DMZ* privée à laquelle aucun système externe n'est autorisé à accéder c.-à-d le seul point d'accès étant le reverse proxy, l'utilisateur n'a par conséquent aucune visibilité de cette infrastructure.

L'architecture de notre système donc peut se définir comme dans la figure ci-dessous :

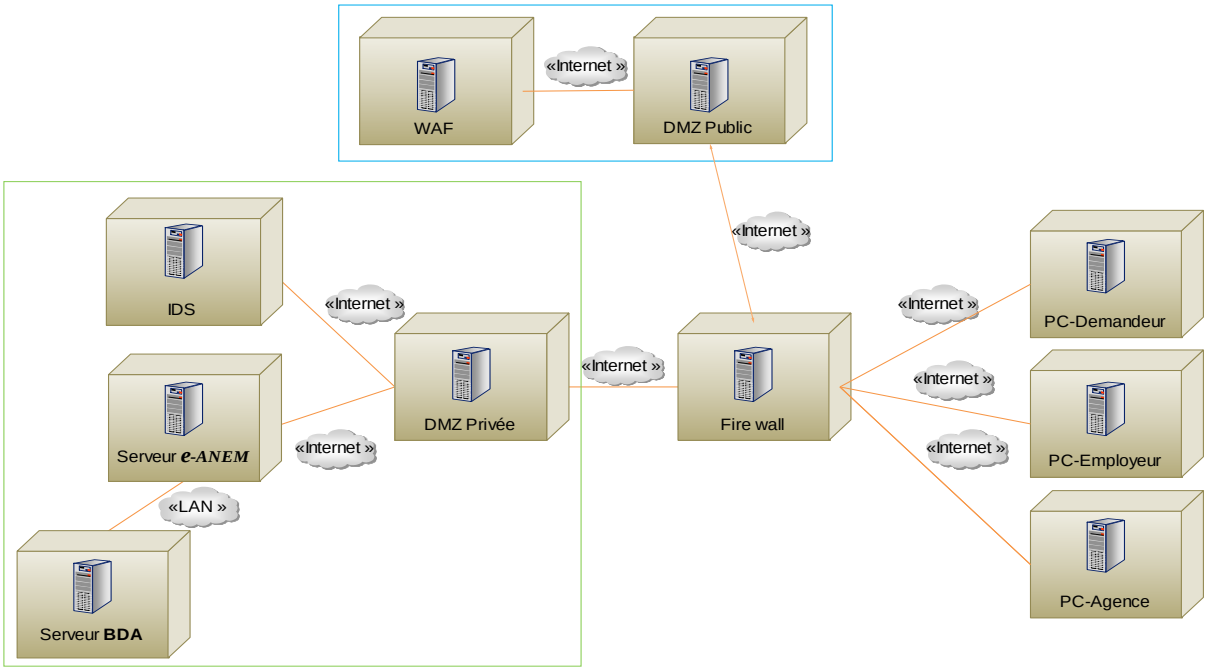

Figure 15 : Modèle de configuration matérielle sécurisée du système *e-ANEM*

[5] Intrusion Detection System

[6] Web Application Firewall

[7] DMZ est une partie du système d'Information présentant un niveau de sécurité homogène.

7. Conclusion :

Dans ce chapitre nous avons défini la spécification des besoins puis l'analyse et la conception objet du système *e-ANEM* en fondant sur une démarche simplifiée basée sur le langage *UML* avec la prise en charge des exigences de sécurité, cette étude conceptuelle nous a permis d'avoir une vision claire sur notre système et qui puisse être par la suite concrétisé sous forme produit logicielle.

Chapitre 3 :

Implémentation du système *e-ANEM*

1. Introduction

Après l'analyse et la conception du système *e-ANEM*, il s'agit dans ce chapitre d'introduire à la phase de sa réalisation. Après la définition des outils et des technologies de développement utilisés pour la réalisation de notre système, ce chapitre présente les solutions de sécurité utilisées au niveau implémentation ainsi que l'application réalisée et ses interfaces.

2. Choix des outils et technologies de développement

Le langage de programmation *Java*, l'EDI *NetBeans*, le serveur web *GlassFish*, la technologie des *JSF*, le modèle *MVC* et le *SGBDR* oracle sont les outils employés pour la réalisation de système *e-ANEM*

2.1. Java

Java est un langage de programmation moderne développé par *Sun Microsystems*, Il est en passe de détrôner le langage *C++* dont il hérite partiellement la syntaxe mais non ses défauts. Le choix de ce langage se justifie par les nombreux avantages offerts : **[RDS05]**

- Java est multiplateformes et permet d'assurer la sécurité aussi bien pendant le développement que pendant l'utilisation d'un programme java (la sécurité fait partie intégrante du système d'exécution et du compilateur).
- Java est algorithmique et orienté objet ; à ce titre il peut effectuer toutes les tâches d'un tel langage (bureautiques, graphiques, multimédias, bases de données, environnement de développement, etc...).

- Une de ses plus grandes forces est son excellente portabilité due à ses bibliothèques de classes indépendantes de la plate-forme: une fois votre programme créé, il fonctionnera automatiquement sous *Windows*, *Mac*, *Linux*, etc.

2.2. NetBeans

Un tel langage de programmation nécessite un environnement dans lequel on développe l'application. *NetBeans* est un environnement de développement intégré (*EDI*) [1] écrit en Java permettant d'écrire, compiler, déboguer et déployer des programmes, il peut supporter n'importe quel langage de programmation comme il est disponible sous *Windows, Linux, Solaris, Mac OS* ou sous toute autre version indépendante des systèmes d'exploitation (requérant une machine virtuelle Java).

NetBeans comprend toutes les caractéristiques d'un *IDE* moderne (coloration syntaxique, projets multi-langage, refactoring, éditeur graphique d'interfaces et de pages web,…).Il embarque également des Frameworks connus pour Java, facilitant ainsi grandement la tâche des développeurs. En plus, *Netbeans* peut prendre en charge l'exécution des serveurs (*Glassfish*, *Tomcat*, etc.).

2.3. JSF

***J**ava **S**erver **F**aces* est une technologie (Framework[2]) de développement d'application web, qui dispense d'écrire du code Java dans les interfaces permettant de respecter le modèle d'architecture *MVC* [**YBE10**], basé sur les technologies *JSP* et *Servlets* dont le but est d'accroître la productivité des développeurs dans le développement des interfaces utilisateur tout en facilitant leur maintenance.

Nous avons opté pour ce choix parce que ce dernier permet : [**ATO05**]

- Une séparation nette entre la couche de présentation et les autres couches d'une application web.
- Mise en place d'un mapping entre l'*HTML* et l'objet.
- Réutilisation d'un modèle riche de composants graphiques.
- Une gestion de l'état de l'interface entre les différentes requêtes.
- Une liaison simple entre les actions coté « Client » et les actions des objets Java coté « Serveur ».
- Création de nouveaux composants graphiques en combinant plusieurs composants pour aboutir à un composant plus complexe.

[1] Integrated Development Environment (IDE) en anglais.
[2] Un ensemble de composants logiciels réutilisables qui permet d'accélérer le développement d'applications

- La séparation des problématiques de construction de l'interface et du rendu de cette interface ce qui permet le support de différents clients (*HTML*, *WML*, *XML*, ...).
- JSF tient en compte les différentes expériences acquises non seulement avec des technologies de type standard comme les *Servlets*, les *JSP*,… mais aussi avec les technologies de type *Framework* comme *Struts*.
- L'utilisation complète des autres technologies web tels que les *servlets*, les *JSP*, les balises JSTL et les langages d'expressions.

2.4. MVC

JSF reposent sur le modèle *MVC* ; Le *Model-View-Controller* est une architecture et une méthode de conception qui a pour objectif d'organiser une application interactive en séparant les données, la représentation des données et le comportement de l'application. [**DLA02**]

Ce modèle d'architecture est constitué de trois parties.

- La première : le contrôleur, reçoit les actions de l'utilisateur et gère la répartition des traitements entre la vue et le modèle. C'est cette partie qui gère toutes les entrées de l'application.
- Le modèle représente le comportement de l'application (traitements des données, interactions avec la base de données, etc). Il décrit ou contient les données manipulées par l'application. Il assure la gestion de ces données et garantit leur intégrité.
- La dernière partie : la vue, est la partie graphique correspond à l'interface avec laquelle l'utilisateur interagit. Sa première tâche est de présenter les résultats renvoyés par le modèle. Sa seconde tâche est de recevoir toutes les actions de l'utilisateur (clic de souris, sélection d'une entrée, boutons, etc). Ces différents événements sont envoyés au contrôleur. [**LZU12]**

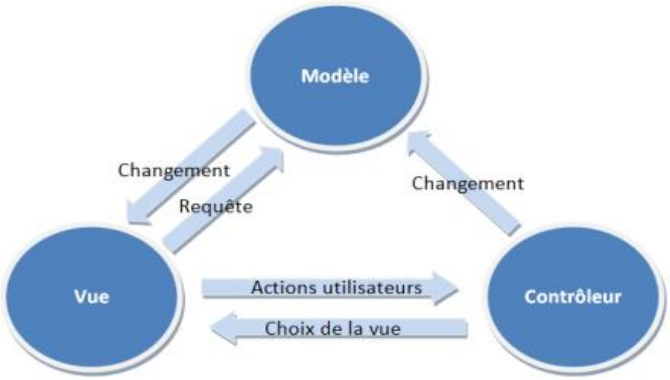

Figure 16 : L'architecture *MVC*

2.5. GlassFish

GlassFish est le nom du serveur d'applications Open Source *Java EE 5* et désormais *Java EE 6* avec la version 3 qui sert de socle au produit *Oracle GlassFish Server*. Sa partie *Toplink* persistante provient d'Oracle. C'est la réponse aux développeurs *Java* désirant d'accéder aux sources et de contribuer au développement des serveurs d'applications de nouvelle génération.
Le choix d'utilisation de ce serveur Web est motivé par les raisons ci après :

- Il apporte un découpage modulaire.
- Leur temps de démarrage est de quelques secondes seulement.
- En plus Au niveau des standards, *GlassFish* est une implémentation complète de la norme *Java EE 5* qui recouvre :
  - JSF (*Java Server Faces*) - *Framework MVC*
  - JSP 2.1 & Servlet 2.5 : Pour générer des pages et du contenu *WEB* dynamiquement.
  - EJB 3 (approche *POJO*, configuration par annotations, injection de dépendance) …..etc.

2.6. ORACLE 10g

Nous avons optés à utiliser Oracle (la version *ORACLE 10g*), puisqu'il est considéré comme l'un des *SGBDR* les plus puissants qui existent. Il se caractérise principalement par :

- Modèle sous jacent simple, bien formalisé et éprouvé,
- Simplicité d'accès via un langage de requêtes *SQL***,**
- Existence de contraintes d'intégrité,
  - Intégrité de domaines (contrôle du type de la donnée)
  - Intégrité de relation (existence d'une clé primaire : unique et toujours définie (non nulle))
  - intégrité de référence (contrôle de la cohérence d'attributs de tables différentes lors des mises à jour) [**WEB04**]
- Un système de droits et de mots de passe très souples et sécurisés. Chiffrement de tous les échanges de mots de passe ce qui garantie une meilleur protection de mot de passe, même lors des connexions.
- Sa disponible sur un grand nombre de plates-formes et apporte de nombreuses nouvelles fonctionnalités.
- Gestion de très grands volumes de données (taille maxi de 65 536 fichiers de 128).

2.7. SQL Developer

Nous avons choisi de travailler avec l'éditeur *Oracle SQL Developer*. Cet outil graphique gratuit libre de *SQL*Plus* permet de simplifie les tâches de développement de bases de données. L'utilisation d'*Oracle SQL Developer* nous aide de parcourir les objets de la base de données, exécuter des instructions et des scripts *SQL*, éditer et déboguer des instructions *PL/SQL*, manipuler et exporter des données, et consulter et créer des rapports.

En plus cet outil permet de connecter aux bases de données *Oracle*, connecter à certains tiers (non-Oracle) des bases de données, afficher les métadonnées et des données, et de migrer ces bases de données d'*Oracle*. [**WEB05**]

3. Solutions de sécurité

- **Cryptographie**

C'est la conception de formules mathématiques permettant de transformé un message clair en un message chiffré dont le but est assurer la confidentialité d'informations sensibles, assurer l'identité d'un individu ou d'une application distante (authenticité), détecter toute altération d'information stockée ou transmise (Le contrôle d'intégrité) et empêcher un expéditeur de pouvoir nier son envoi (La non-répudiation).**[AGU08]**

Figure 17: Principe de cryptographie

L'utilisation de cette solution de sécurité dans notre application est pour assurer la confidentialité et l'intégrité des informations critiques telles que le de mot de passe où il apparut crypté dans la base de données en utilisant pour cela l'algorithme de cryptage *MD5* qui est une fonction de hachage à sens unique et qui peut être calculée sans connaissance d'un secret.

- Parmi les autre solutions qui nous avons intégré, le masquage de l'identifiant et le mot passe où ces derniers ne sont pas apparus ce qui minimise par exemple les attaques de type injection *SQL*.

4. Présentation de l'application

L'application *e-ANEM* est composée de trois parties : Un espace de demandeurs d'emploi, un autre pour les employeurs (les entreprises) et le troisième espace pour l'administration de l'agence.

4.1. Interface page d'accueil

C'est la fenêtre principale de l'application qui offre l'accès au demandeur, employeur, et les représentants de l'agence *ANEM*. Cette fenêtre présente aussi des annonces et publicité sur des nouvelles mécanismes et solutions d'emploi.

Figure 18 : Page principale « *e-ANEM* »

4.2. Volet demandeur

Le cas où un demandeur visite le site, il doit cliquer sur volet demandeurs où l'interface ci-dessous va apparaitre. Dans cette volet, si le cas d'un nouveau demandeur, il doit s'inscrire en cliquant sur inscription où le formulaire ci après s'affiche.

Figure 19 : Interface de préinscription de demandeur

Dans le premier lieu le demandeur doit introduire ses informations personnelles, le demandeur doit choisir un identifiant et mot de passe afin d'utiliser plus tard pour consulter et mettre à jour ses coordonnées. Il est nécessaire après l'introduction des informations concernant son diplôme s'il est diplômé. C'est si ne pas le cas le demandeur doit introduire les informations concernant leur expérience toujours s'il possède. Comme il est obligatoire au demandeur d'introduire les cordonnées de sa pièce d'identité (carte nationale, permis de conduire). Enfin, le demandeur doit valider la préinscription et il reçoit la lettre suivante.

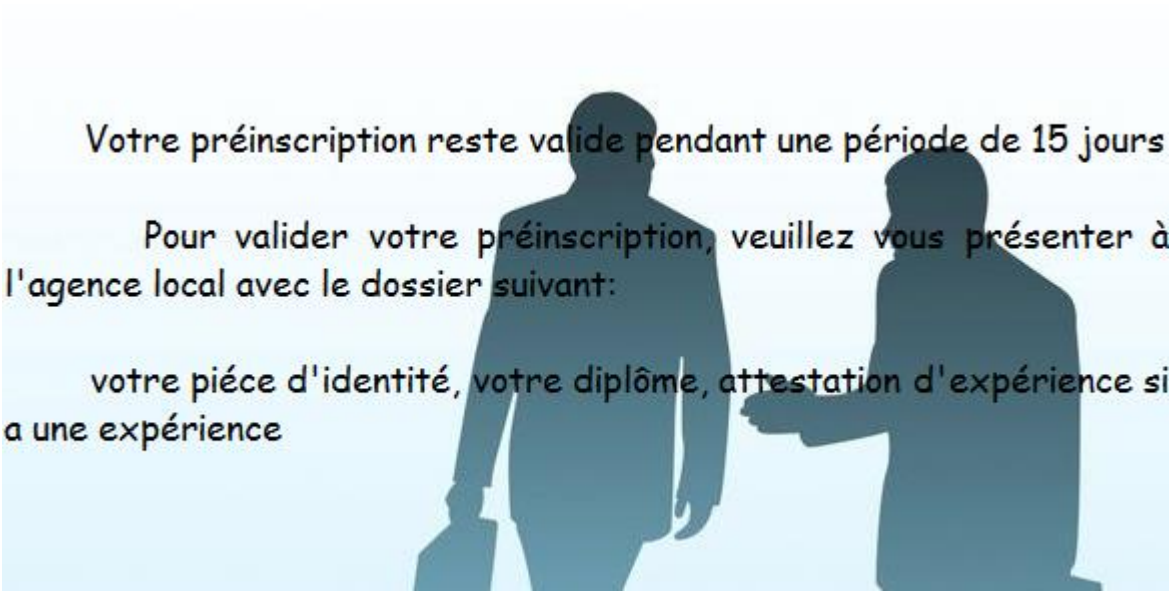

Figure20: Interface montre la validation initiale de préinscription du demandeur

Le cas où le demandeur déjà inscrit, il peut visiter le site afin de consulter et mettre à jour ses informations, comme il peut consulter ses réponses et télécharger son attestation d'inscription en insérant l'identifiant et le mot passe.

4.3. Volet employeur :

Le cas d'un employeur qui visite le site *e-ANEM*, il doit cliquer sur volet employeur. Dans ce volet l'employeur a la possibilité de créer un compte s'il est nouveau en cliquant sur « inscrire compte » où la fiche d'identifiant d'entreprise va s'affichera.

Figure21: Interface d'inscription de l'employeur

L'employeur doit remplir ce formulaire pour être accepter à l'agence. Dans le cas où l'employeur a déjà inscrit, il peut créer et consulter ses offres en introduisant l'identifiant et le mot de passe.

Pour la création d'une nouvelle offre l'employeur doit remplir un formulaire (les informations de l'offre) concernant leur entreprise

Figure 22: Inscription des offres

4.4. Volet agence :

Cette volet est repartie en deux espaces : le premier concernant le service demandeurs et l'autre pour le service offre d'emploi.

Chaque responsable de service doit être s'authentifié pour accéder à son espace en introduisant son propre identifiant et mot de passe.

Figure23: Interface service offre d'emploi

4.4.1. Volet service offre :

Comme nous avons dit précédemment ce service à son tour doit s'authentifier afin de connecter à son espace. Après l'authentification, le conseiller de service offre doit remplir le formulaire ci-dessous pour faire la recherche de l'offre concerné.

Figure 24: Interface montre lancement de rapprochement

Donc le résultat de recherche est:

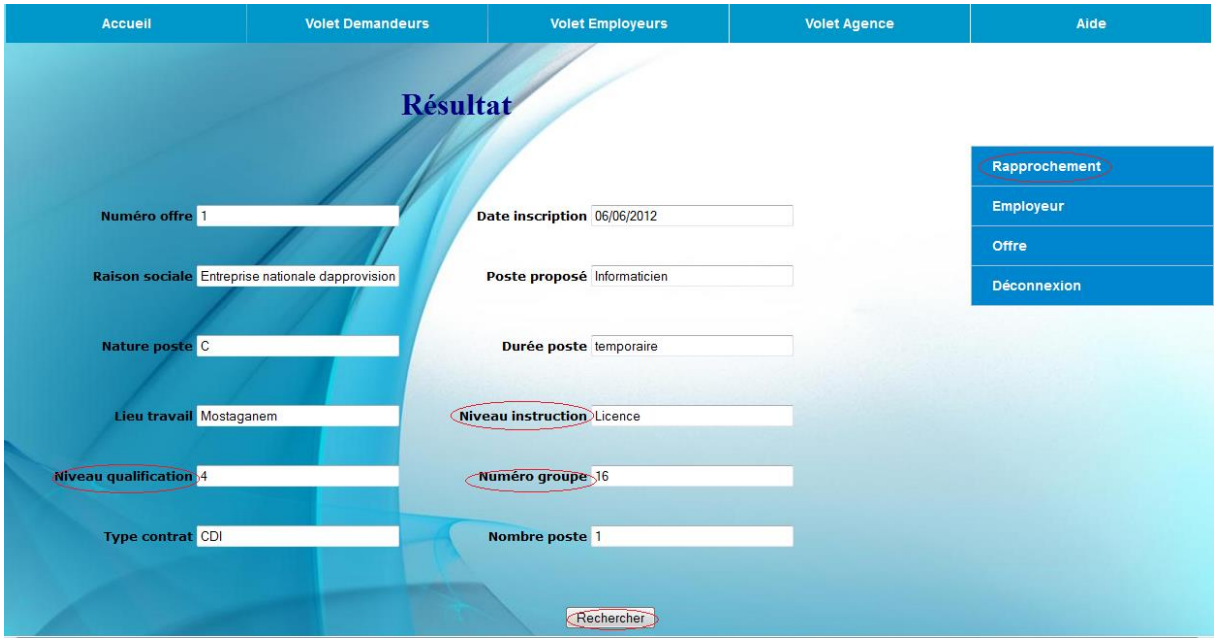

Figure 25 : Les résultats de rapprochement

Après la recherche, la liste des demandeurs répondre aux conditions de cette offre sera affiché. Le conseiller de service offre crée un bulletin pour chaque demandeur sélectionné. Il peut faire ensuite des recherches sur les bulletins.

5. Conclusion

Au cours de ce chapitre nous avons présenté les étapes de la réalisation du système *e-ANEM*, le choix des différents outils de développement : le langage *Java*, la technologie *JSF*, le *SGBD* oracle et le serveur *GlassFish* nous a permet de réaliser une application Web qui répond aux besoins de qualité et d'intégrer quelques solutions de sécurité durant l'implémentation de notre produit logicielle.

Conclusion générale

Presque toutes les organisations, aussi bien publiques que privées, possèdent aujourd'hui une vitrine sur le Web. Ces vitrines sont majoritairement devenues interactives, c'est-à-dire qu'elles stockent des données, organisent des échanges d'information, réalisent des processus vitaux pour le bon fonctionnement de ces organisations ou le business de ces entreprises.

Du fait de cette importance croissante que connait le monde de télécommunication et la progression remarquable des technologies Web qui ouvrent la porte à de nombreux types d'attaques visant les applications Web, soit à les rendre inutilisables (*Deny of Service*), soit à accéder aux données sensibles qu'elles contiennent et cela d'autant plus grave que les applications Web manipulent parfois des données confidentielles (*mots de passe, numéros de cartes bancaires, les informations personnelles*), la problématique de la sécurité des données et des processus prend une grande importance. Donc, il est indispensable de considérer la sécurité d'une application Web dans tous les stades de développement ; au moment de la conception, durant le développement, et enfin, après le développement en utilisant des dispositions de prévention.

Afin d'assurer le mieux possible la sécurité des applications Web, il est important de connaitre tout d'abord les risques liées aux ces derniers pour les maîtriser**.** Les divers risques possibles contre les applications web ont été étudiés en montrant comment ils peuvent être facilement visant les données d'une application.

Dans ce travail, nous avons présenté une étude de cas détaillée du système *e-ANEM* après un stage passé dans l'agence local de l'emploi, cette étude couvre tous les points de vie nécessaires aux différentes phases de cycle de développement et en tenant compte les besoins fonctionnels et les exigences de sécurité du système.

La réussite des projets réside en premier lieu dans l'existence d'une démarche de développement bien définit et bien géré. Dans ce contexte, nous avons proposé une démarche de conception sécurisée (*DCS*) basé sur le processus *RUP*. Cette démarche permet une bonne modélisation des systèmes d'information en satisfaisant les besoins fonctionnels et pourra trouver une ouverture vers la maitrise des contraintes de sécurité (Disponibilité, Authentification, Intégrité, Confidentialité, Non-Répudiation, etc.) dès la conception.

La nouvelle démarche *DCS* est composée de quatre phases : *Expression et spécification des besoins* où le modèle de contexte et les cas d'utilisation et de sécurité s'expriment les différents besoins fonctionnels et de sécurité du système, *Analyse et conception* qui consiste à identifier le modèle statique qui définit les classes et les objets de système et le modèle dynamique qui décrit les scenarios des interactions d'objets sécurisées, *Modélisation de la navigation* qui permet de modéliser avec précision la navigation dans l'application web, et enfin pour *la conception technique*, le diagramme de composant présente l'organisation et les dépendances entre les différents éléments logiciels du système, et le modèle de configuration matérielle sécurisée exprime les contraintes de mis en œuvre au niveau physique avec l'intégration des dispositions de prévention pour répondre aux exigences de sécurité.

La réalisation d'un produit logiciel de qualité nécessite un bon choix des technologies et des outils de développement. Les *JSF* ont été choisi comme technologie de développement car elles proposent un Framework puissant en termes de sécurité et de performance, *Netbeans* comme support de développement du langage *Java*, des *JSF* et serveur web *Glassfish*, et enfin pour la base de données, l'utilisation d'un *SGBDR* tel qu'*ORACLE* a été indispensable.

Bibliographie

[AGU08]

A.Guetat, *« Cryptographie»*, Licence Pro ATC, (2008)

[ATO05]

« Introduction à JSF, Java Server Faces », Atol , Site: http://www.atolcd.com, (2005).

[CL03]

P.CHAMBET, E.LARCHER, *« Vulnérabilités et solutions de sécurisation des applications Web »*, EdelWeb , RSSI Accor Services,(2003).

[CLU11]

CLUSIF, *«Défense en profondeur des applications web »*, Site :clusif@clusif.asso.fr, (2011).

[DEX06]

« Décret exécutif n° 06-77 », ANEM, (2006).

[DLA02]

D.Laurent, *«Module UV java »*, Cours, (2002).

[EJA08]

E.JAMTEL, « *Attaques sur les applications web : Pourquoi un firewall et un antivirus ne suffisent plus …* », Silicomp-AQL, (2008).

[JJU04]

J.Jürjens, «*Security Requirements Analysis and Modeling of Distributed Systems*», Munich University of Technology, (2004).

[JSN09]

J.Snow, *«Quelques exemples basiques d'attaque sur une application web»*, site: http://www.blogoergosum.com, (2009).

[LB11]

« Applications Web et Cyber- attaques », Livre Blanc, Bee Ware, Site : http://www.bee-ware.net, (2011).

[LHE10]

Le Henaff Loi, « *Détection d'attaques contre les données dans les applications web »*, Supélec, (2010).

[LZU12]

L.Zunarelli, *«Java server faces et struts*», Exposer, Synthèse réalisée dans le cadre de l'U.E : Etude et Projet d'Intergiciels, (2012).

[Mem01]

S.CHEHIDA, *« La modélisation des aspects de sécurité avec UML : Elaboration des extensions, d'une démarche et d'un outil »*, Mémoire magister en informatique, Université Abdelhamid Ibn Badiss de Mostaganem, (2010).

[Mem02]

F.KREDOUDA, Z.CHERGUI, *« UML pour la sécurité à priori des Applications Web : Développement sécurisé d'un système de paiement électronique »*, Mémoire master en informatique, Université Abdelhamid Ibn Badiss de Mostaganem, (2011).

[MYA11]

M.Yassine, *« Protection des applications web contre l'attaque par injection de code SQL »*, Université du Québec à Montréal, Site : http://www.mohamedyassine.com, (2011).

[OBO12]

O.BOUSSAID, *«Une méthodologie de conception »*, Cours UML; univ-lyon, (2012).

[PCH02]

P.CHAMBET, *«Vulnérabilités et solutions de sécurisation des applications Web : pourquoi les firewalls ne couvrent pas tous les aspects de protection ? »*, EdelWeb, (2002).

[PCH03]

P.CHAMET, *«Vulnérabilité et sécurisation des applications web : pourquoi les firewalls sont impuissants face a certaines attaques »*, EdelWeb, (2003).

[PRO08]

P.ROQUES ,«*Les Cahiers du programmeur UML2 ;modéliser une application web »* ; Eyrolles, 4 éme édition, (2008).

[RDS05]

R. di Scala, *« Java2, Les fondements du langage Java »*, Site : http://www.berti-editions.com, (2005).

[RZO12]

R.ZOUARI, *« Les attaques dans le web »*, Cours, Ecole Supérieure des Technologies et d'Informatique ESTI, (2012).

[SG07]

D.SEGUY, P.GAMACHE, **«***Sécurité PHP5 et MySQL*»,Eyrolles , (2007).

[YBE10]

Y.Bekkers, *« JSF Java Server Faces »*, L'AWT-SWING du WEB, (2010).

[YEL06]

Y.ELMAZOURI, *« UML : Diagramme de composants, Diagramme de déploiement »*, Cours, Site : www.freewebs.com/fresma, (2006).

Webographie

[WEB01]

http://edraw-uml-diagram.programmesetjeux.com/

[WEB02]

http://www.adullact.org/documents/grc_modelisation_developpement_v1.3

[WEB03]

http://www.latrach.net/les-caracteristiques-de-java

[WEB 04]

http://www.tondeurh.fr/docs/intro-oracle.pdf

[WEB 05]

http://docs.oracle.com/cd/E25259_01/appdev.31/e24285.pdf