BILLET Benjamin
CSII 1ère année

DEVELOPPEMENT D'UNE APPLICATION WEB

Rapport de stage

Stage effectué de juin à septembre 2009

E.P.S.I Montpellier
437 rue des apothicaires
34090 MONTPELLIER

REMERCIEMENTS

I. INTRODUCTION

Effectuer une mission en entreprise représente toujours l'occasion d'enrichir ses connaissances, ce au travers de travaux permettant de les compléter ou d'en acquérir de nouvelles. De mon point de vue, il me paraissait nécessaire, sinon indispensable, d'être confronté à des situations et des technologies nouvelles présentant un grand intérêt dans le secteur de l'informatique.
Ainsi, la mission proposée et les éléments mis en œuvre au sein de celle-ci ont été Des éléments prépondérants dans le choix de l'entreprise.

L'utilisation des clients légers en entreprise est une problématique de plus en plus présente. Aussi ai-je trouvé particulièrement intéressant d'effectuer des travaux dans ce domaine en expansion. De ce fait, comme l'EPSI exploite des systèmes de ce genre à des fins d'interopérabilité[1], j'ai pensé qu'il s'agissait là du partenaire idéal pour aborder une mission dans ce domaine.

L'EPSI est une association à but non lucratif qui s'occupe d'enseignement informatique qui utilise au quotidien des outils dont la logique applicative est centralisée sur des serveurs particuliers. Ces outils sont des biais de communication verticaux et horizontaux ainsi que des solutions de gestion de classes, d'étudiants, de planning, etc.
En d'autres termes, ces outils sont des réponses à des besoins de traitement, de diffusion d'information et de simplification des divers services de fond.

Toutefois, au fil du temps, ces systèmes se sont enrichis de nouvelles choses motivées par des besoins naissants. Les différents ajouts ont fini par complexifier l'utilisation et par alourdir le fonctionnement global. A tel point qu'il fut décidé par l'EPSI que le système actuel devait être repensé pour produire quelque chose de plus robuste, de plus efficace et de plus facile à maintenir.

[1] Il s'agit ici de garantir que n'importe qui puisse accéder aux outils depuis n'importe quel navigateur sous n'importe quel système d'exploitation.

De ce fait, la mission repose sur des besoins réels qui se sont fait ressentir au cours du temps mais qui n'ont pu être étudiés que récemment par l'entreprise.

Etant donné l'ampleur du projet, celui-ci a été considéré dans une optique de long terme. Ainsi, mon rôle dans celui-ci n'est qu'une première itération qui consistera à poser des bases fonctionnelles ainsi qu'à reconcevoir de précédentes fonctionnalités qui seront, à terme, complétées par d'autres développeurs.

Au travers ce document, je vais donc vous présenter les différentes étapes de la réalisation du projet qui m'a été confiée. Tout d'abord, une présentation de l'environnement dans lequel le système actuel évolue permettra d'analyser la situation actuelle de l'EPSI pour en dégager les besoins réels et les enjeux que le projet implique. Cela sera étayé par l'analyse des systèmes existants.
Les méthodes de travail seront ensuite décrites et analysées avant que soient abordés les différentes pistes de recherche et le choix des solutions techniques exploitées lors de la phase de développement.
Enfin, une analyse permettra d'évaluer avec le recul nécessaire la pertinence des différents choix par rapport aux résultats obtenus.

II. L'ENTREPRISE

1. Historique du groupe

L'EPSI (Ecole privée des sciences informatiques) a été fondée en 1961 par le GPNI[1] et son premier centre de formation a été établi à Paris.

Vingt ans plus tard, en 1981, un second centre s'établit à Bordeaux, puis un troisième à Montpellier en 1983.

En 1987, 2002 et 2004 d'autres centres sont mis en place respectivement à Arras, Nantes et Lyon.

En 1994, le titre d'« expert en informatique et système d'informations » délivré par l'école est enregistré au RCNP[2] en tant que niveau 1.

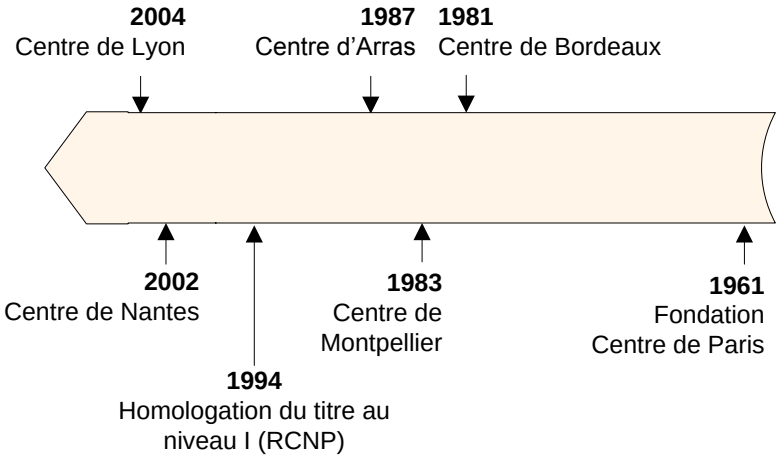


Fig. 1 - Chronologie de l'évolution de l'EPSI.

[1] Le GPNI (Groupement professionnel national de l'informatique) est une chambre syndicale de SSII qui représente, conseille, promeut et défend les TPE et PME du secteur informatique français.

[2] Le RCNP (Répertoire national des certifications professionnelles) enregistre les certifications reconnues sur l'ensemble du territoire.

2. Informations techniques

Données juridiques

Raison sociale : EPSI (Ecole privée des sciences informatiques).

Statut : Association à but non lucratif (Loi 1901).

Implantation : Paris - Nantes - Lyon - Bordeaux - Montpellier - Arras

Hiérarchie

Au niveau du groupe, la hiérarchie est la suivante :

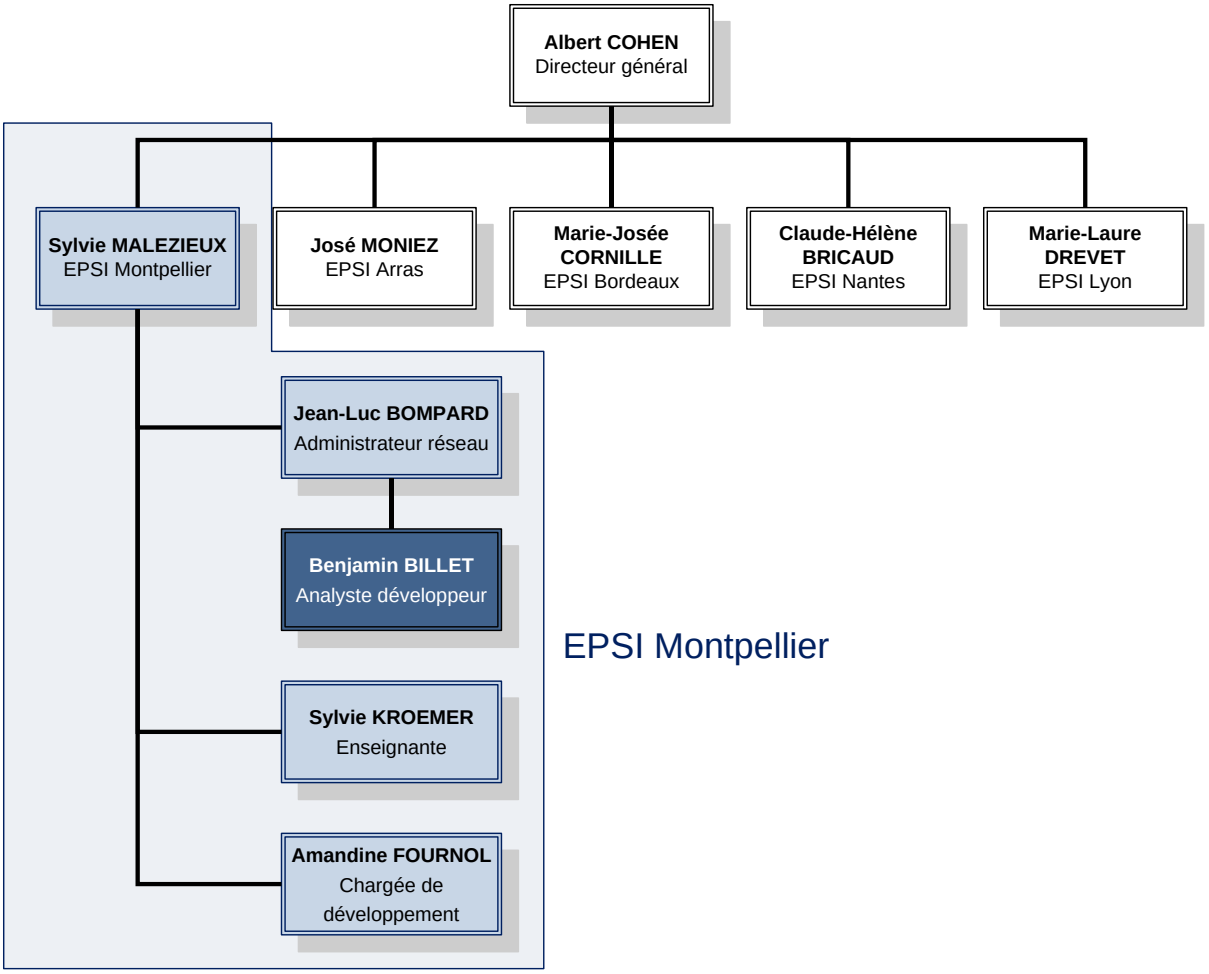


Fig. 2 - Organigramme hiérarchique de la société

Partenaires

Les partenaires de l'EPSI sont multiples, on y retrouve des constructeurs, des opérateurs de télécom, des éditeurs de logiciels, des sociétés de services (SSII), des sociétés de grande distribution, des banques et des assurances ainsi que des entreprises industrielles diverses.

Img. 1 - Quelques partenaires de l'EPSI

En plus des partenariats, certaines entreprises parrainent des promotions particulières. Pour l'entreprise c'est une collaboration sur la veille technologique (au niveau des laboratoires), des propositions de stages et le ciblage des meilleurs éléments d'une promotion sortante.

Historiquement des entreprises comme Dassault, France Télécom, la B.N.P ou encore le Conseil régional du Languedoc-Roussillon ont parrainé des promotions de l'EPSI.

Positionnement

L'EPSI se positionne sur le secteur de la formation aux métiers de l'informatique.

On y distingue plusieurs cursus disponibles :

- Cycle préparatoire intégré (2 ans)

  Il s'agit du cursus préparatoire au cycle ingénieur. Il se termine par l'obtention du BTS Informatique de Gestion.

- Cycle supérieur d'ingénierie informatique (3 ans)

  Il s'agit d'une formation débouchant sur un titre homologué par l'état (niveau 1). Les cours sont ménagés de façon à ce que l'étudiant puisse passer une partie de son temps en entreprise (alternance) s'il le souhaite.

- Formations courtes (< 1 an)

  Le master consultant ERP et le master mainframe architecture specialist sont des formations plus courtes destinées à spécialiser des individus à des domaines techniques particuliers.

  Elles s'effectuent en partenariat avec des entreprises extérieures.

Forces

Au niveau des points forts, j'ai relevé plusieurs points concernant le groupe tout entier.

- Présence

  Le groupe EPSI est présent sur toute la France au travers de ses six sites de formation. Cela lui permet de s'intégrer dans le contexte économique et professionnel régional.

- Diplôme reconnu

  Le diplôme obtenu au bout du cursus d'ingénierie de l'EPSI est le premier à avoir obtenu la certification au niveau I.

- Formation reconnue par les entreprises

  Le nombre de diplômés engagés dès la sortie (CDI, statut cadre) est proche du 100%, notamment grâce aux différents partenariats et à la proximité générale de l'EPSI avec les entreprises. Au niveau des salaires de sortie, ceux-ci tournent en moyenne entre 30 et 36 k€[1].

- Formations particulières

  L'EPSI propose un master consultant ERP en alternance, en partenariat avec SAP et SOPRA Group.

- Proche de l'entreprise

  Les centres de formation EPSI effectuent divers partenariats avec des entreprises de plus ou moins grande envergure. Celles-ci peuvent même parrainer des promotions.

  De plus, le cursus ingénieur totalise 12 mois de stages répartis sur les trois ans d'études (3 mois en première et seconde année, 6 mois en fin de cycle).

Localement, en ce qui concerne l'EPSI Montpellier, j'ai observé des points forts particuliers. Tout d'abord, le centre dispose d'une formation spécifique en association avec IBM : le master IBM-Mainframe Architecture Specialist qui forme des individus sur les mainframes fabriqués par IBM.

[1] Source : Site de l'EPSI et magazine l'USINE NOUVELLE.

III. LA MISSION

1. Les portails EPSI

Au fur et à mesure du développement de l'EPSI Montpellier, des outils web ont été conçus pour aider les différents intervenants à communiquer et à effectuer des opérations (communication entre les étudiants et l'administration, outils de gestion pour l'administration, etc.).

On distingue notamment deux portails. L'un est accessible uniquement en interne (Intranet) et l'autre, développé bien après le premier, est consultable depuis l'extérieur (Extranet).
Ces deux outils sont utilisés d'une part par les étudiants qui peuvent, par exemple, s'informer sur les absences des professeurs, consulter des offres de stages, communiquer avec les autres élèves de leur classe ou découvrir ce qu'étaient devenus les anciens, et d'autre part par l'administration pour gérer les étudiants, les classes, les emplois du temps, etc.

Img. 2 - L'extranet EPSI

C'est dans ces buts de gestion et de communication qu'avaient été développés ces portails. Ma mission consistait donc à inscrire ces premiers enjeux dans une optique d'unification, de refonte et d'évolution.

2. Les nouveaux enjeux

Les enjeux initiaux ayant amené au développement des deux précédents portails sont toujours valides. Il s'agit de simplifier la communication entre l'administration et les étudiants, ce en diffusant de l'information (planning, modification d'emploi du temps, documentation, anciens de l'EPSI, offres de stages, etc.), et de mettre à disposition des personnes autorisées des outils de gestion destinés à garantir une certaine efficacité.
Ces enjeux sont désormais étendus à de nouvelles fonctionnalités comme la gestion des quotas imprimante en masse ou le suivi des étudiants lors de la production des rapports de stage ou des mémoires.

On distingue toutefois d'autres enjeux liés à de nouveaux besoins apparus au cours du temps et de l'utilisation des anciens outils.

Disposer d'un système unifié, maintenable et évolutif
L'enjeu principal de la mission est de fusionner les différentes applications web, ce pour simplifier la gestion et offrir un outil unique aux utilisateurs.
En effet, l'utilisation des deux portails est un peu fastidieuse pour les étudiants (authentification différente, informations en double, etc.) et entraîne un désintérêt de ceux-ci vis-à-vis de l'outil.

De plus, il s'agit aussi de fusionner les sources de données (annuaires, bases de données, fichiers, etc.) pour éviter les doublons et les décalages. Cela consiste, en outre, en leur optimisation pour qu'au final l'espace occupé et la complexité des systèmes en soient d'autant réduits.
Cependant, les sources de données doivent être améliorées sans interférer avec le fonctionnement des services logiciels exploitant déjà ces informations (service de filtrage des adresses non autorisées sur le réseau wifi, serveur de travail collaboratif, service de messagerie, etc.). Il s'agit d'une évolution et non d'une révolution du système d'information complet.

Dans le même esprit, la nouvelle application doit reprendre les fonctionnalités des anciens systèmes. C'est une mise à plat de l'existant en vue d'une reconstruction sur des bases saines pour permettre à l'application d'évoluer sans complications car, en l'état actuel des choses, les deux portails sont peu documentés au niveau de leur conception et de leur développement, ce qui rend leur maintenance difficile. Du reste, les choix technologiques effectués par le passé n'étaient pas judicieux (d'où la volonté de refonte) ce qui est désormais un frein à l'évolution des applications.

Ainsi, l'enjeu consiste à disposer d'un nouveau système qui, en plus d'être uniforme, peut être facilement modifié et amélioré par de futurs développeurs. Cela passe par l'utilisation d'un nouveau langage adapté et par l'écriture de documents de supports pour les futurs intervenants amenés à modifier l'application.

⇨ La maintenance et l'évolution sont simplifiées (gain de temps et d'énergie), ainsi que l'utilisation (gain en efficacité).

<u>Déléguer des opérations aux étudiants</u>
L'objectif est de réduire la lourdeur de certaines opérations nécessitant l'intervention du secrétariat ou de l'administrateur réseau. Ceux-ci, étant régulièrement sollicités, perdent du temps à effectuer des opérations qui pourraient être automatisées ou ne nécessiter que le seul concours de l'étudiant (remplir la fiche virtuelle des informations personnelles, par exemple).

L'idée est de donner à l'utilisateur un peu plus de flexibilité vis-à-vis des services techniques mis à disposition par l'EPSI (ceux-ci seront plus amplement détaillés dans l'analyse sur l'existant).
C'est à cette intention que seront implémentées les nouvelles fonctionnalités demandées par l'entreprise.

En second lieu, le nouveau portail doit informer les étudiants des différents services annexes proposés par le système d'information de l'école (gestion de projet, mail, bases de données, etc.) et simplifier l'accès aux documentations.

⇨ Les étudiants deviennent plus autonomes et les autres intervenants peuvent se concentrer sur des tâches plus importantes.

Il est important de garder à l'esprit que la mission s'inscrit dans une optique plus vaste de remaniement du système d'information.
En effet, c'est à cette époque que la direction générale du groupe a décidé d'unifier les différents centres autour de systèmes d'informations communs. Aussi, les outils d'administration et de gestion sont en passe d'être changés, rendant obsolète une grande partie de l'existant.
Le portail étant une solution de gestion interne, celui-ci exploitera des éléments passerelles entre le nouveau système d'information et l'existant sur lequel se basera le nouveau portail.

IV. REALISATION

1. Analyse de l'existant

Au niveau de l'existant, la quantité d'opérations possibles est assez importante, certaines étant plus fastidieuses que d'autres. Toutefois, la cohérence de l'ensemble est incertaine ; une partie des opérations s'effectuent soit par des outils intégrés au portail existant, soit en passant par une tierce personne (l'administrateur réseau ou le secrétariat).
Aussi, dans le premier cas je parlerais d' « outil » tandis que dans l'autre j'utiliserai le terme « protocole ».

Outils existants au niveau utilisateur
Au niveau de l'extranet, l'étudiant peut consulter l'ensemble des offres de stages, des conventions de stage pré-remplies et des informations concernant la vie de l'école (news générales) ou concernant une classe en particulier (news de classe).

Au niveau de l'intranet, les fonctionnalités sont plus nombreuses. Une partie accessible aux étudiants permet de consulter les offres de stages, de prendre connaissance des informations concernant leur classe (planning et changements), de parcourir la liste des anciens de l'EPSI et de contacter d'autres étudiants ou des professeurs.

Chaque étudiant dispose d'un compte unique dont la capacité de stockage est limitée. Ce compte est accessible depuis tous les postes du centre, que ce soit sous Windows et Linux. Les informations relatives aux comptes sont stockées dans un annuaire qui alimente en partie les portails (certaines informations utilisateurs sont stockées dans une base de données à part).
Depuis ce compte, l'étudiant peut accéder aux imprimantes partagées sur le réseau. Les utilisateurs sont soumis à des quotas d'impression (différent en fonction des formations) et d'espace disque.

Il existe aussi un espace public pour les utilisateurs. Celui-ci est structuré pour permettre les échanges au sein d'une même classe, ainsi que pour des échanges inter classes dans des conditions particulières (zones spécifiques). Des emplacements sont réservés aux professeurs et à l'administration.

Protocoles existants au niveau utilisateur

Un certain nombre d'opérations ne sont réalisables que par le biais d'un individu tiers. On distingue notamment un certain nombre d'opérations nécessitant l'intervention de l'administrateur réseau, qui n'est pas toujours facilement accessible.

- L'autorisation d'une adresse MAC sur le réseau wifi.
- L'ouverture d'un espace de travail collaboratif.
- L'activation d'une base de données.
- L'activation d'un compte mail (qui doit ensuite être configuré par l'utilisateur).
- L'activation d'un compte MSDNAA.
- L'ouverture d'un espace de travail collaboratif.

Des documentations présentes sur les espaces publics permettent de prendre connaissance des services disponibles, toutefois celles-ci ne sont pas toujours simples d'accès.

Ensuite, on distingue des opérations nécessitant le concours du secrétariat.

- Accès au document des offres de stages

  Les utilisateurs peuvent demander à consulter une version papier des offres de stages. Toutefois, il peut exister un décalage plus ou moins important entre la version papier (qui résulte d'un document à part) et la version informatique qui résulte de l'ajout manuel des éléments déjà intégrés à la version papier.

- Convention de stage

L'étudiant doit remplir une demande de convention de stage (disponible depuis le réseau ou depuis l'extranet) avant de la transmettre au secrétariat. De là, la convention de stage (3 exemplaires) sera validée et pré-remplie avant d'être remise à l'utilisateur qui devra ensuite s'occuper de les faire signer par son tuteur de stage.

Outils existant en interne

En interne, les informations utilisateurs[1] sont dispersées entre l'annuaire et des bases de données spécifiques.

En effet, le réseau wifi est géré par un service précis, tandis que les comptes (identifiants, droits, quotas) sont gérés par l'annuaire. Enfin, les informations personnelles supplémentaires (adresse physique, numéros de téléphone fixe et portable) sont stockées dans des bases de données propres aux différents portails et alimentent d'autres services de gestion.

Une extension du portail intranet permet aux utilisateurs autorisés (administrateurs) d'effectuer des modifications sur les informations personnelles, de gérer des comptes et des classes, d'envoyer des messages en masse (à toute une classe, à un groupe d'élèves, à tout le monde, aux professeurs, etc.), de créer et d'organiser les plannings, etc.

Cependant, cet outil d'administration travaille sur une surcouche de données distincte de l'annuaire qui gère les comptes utilisateurs. Cette couche supplémentaire est alimentée en début d'année par les nouveaux élèves qui y renseignent leurs informations personnelles depuis un poste autorisé.

Ces données sont aussi utilisées, en partie, par le second portail (extranet) qui s'appuie aussi sur les données de l'annuaire (notamment pour l'identification des utilisateurs).

[1] J'appelle « informations utilisateurs » toute donnée touchant de prés ou de loin à un étudiant. Cela comprend donc les identifiants, les informations personnelles, les droits d'accès et les paramètres d'utilisation des services (adresses MAC autorisées, travail collaboratif, compte mail, et cætera).

Problèmes principaux observés

De l'analyse du système existant ont été relevés des dysfonctionnements dans l'utilisation ou d'ordre technique.

Premièrement, un grand nombre d'opérations requièrent une intervention humaine externe pour être menées à bien. Cela sous-entend des démarches de la part de l'étudiant et nécessite que celui-ci soit informé de l'existence de tel ou tel service ainsi que des méthodes qui peuvent y être liées. En outre, il est nécessaire que les disponibilités des intervenants soient compatibles pour éviter de perdre du temps, ce qui n'est pas forcément simple.

Deuxièmement, l'existence de deux portails au fonctionnement relativement différent est perturbante, d'autant plus que l'un exploite les identifiants de l'annuaire des comptes tandis que l'autre se base sur une source de données autre ce qui force l'étudiant à gérer deux modes d'identification différents.
En grande partie, les informations et opérations disponibles depuis l'un ou l'autre ne sont pas les mêmes. Cependant, celles qui sont communes aux deux sont souvent désynchronisées et il est difficile de déterminer quelle information est la plus à jour.

Troisièmement, l'existence de multiples sources de données rend l'administration complexe. En effet, des informations arrivent d'un côté mais ne sont pas synchronisées avec les systèmes qui sont sensés les traiter. Aussi, l'utilisation par le secrétariat d'un outil ou d'un autre va intervenir sur des jeux d'information mais pas sur d'autres.
Comme des informations de même nature sont dispersées, souvent en doubles exemplaires, entre plusieurs sources de données (annuaires, bases de données, fichiers, etc.), il devient difficile pour un administrateur de gérer les problématiques qui y sont liées.

Quatrièmement, des problèmes d'ordre purement technique ont été observés. Par exemple, les applications web ont été développées en PHP, un langage nécessitant

une rigueur importante et des outils supplémentaires pour pouvoir créer des applications évolutives. En l'état, il est particulièrement difficile de maintenir les logiciels existants et de les faire évoluer.
J'ai aussi remarqué, après un temps, que des corrections et des optimisations relativement simples pouvaient être apportées sur certaines bases de données en fonctionnement.

Au final, ces diverses difficultés ont conduit à envisager des modifications d'une partie du système d'information, plus précisément une fusion et une réécriture des applications web (les portails).

2. Gestion de projet

Méthode de travail

L'application peut être vue comme une suite de modules logiciels distincts ce qui rend plus facile l'utilisation d'une méthode de travail dite « itérative[1] ».

Comme la liste des fonctionnalités initiale était amenée à changer au cours du temps, d'une part à cause des ajouts et modifications émergeant du client et d'autre part à cause de mes propres suggestions, le découpage en module a permis de conserver une certaine simplicité au niveau de l'adaptation.

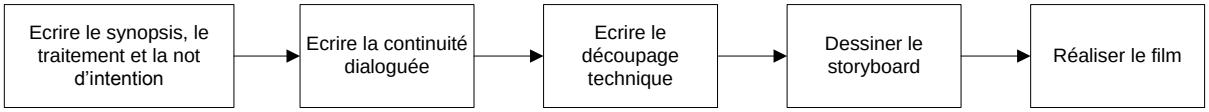


Fig. 3 - Exemple d'application de la méthode itérative

De plus, j'ai appliqué certains des principes édictés par les méthodes dites « incrémentales[2] » car il y a eu beaucoup de remaniement (refactoring) au niveau du code source. Cela est du à ma progression dans la découverte de JEE qui m'a amené à revenir en arrière à plusieurs reprises pour simplifier et améliorer des éléments qui étaient déjà écrits.

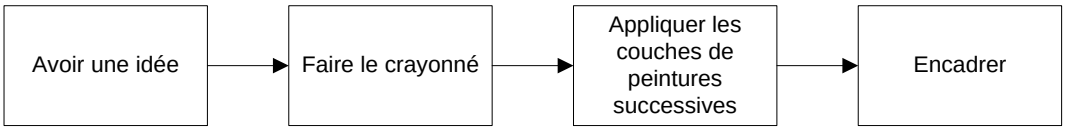


Fig. 4 - Exemple d'application de la méthode incrémentale

Ainsi, j'ai commencé, par rapport aux documents fournis par le client, à découper l'application en parties (thèmes) bien distinctes d'envergures réduites (méthode

[1] Une méthode itérative consiste à découper un projet en phases très courtes dans lesquelles on retrouvera toutes les étapes de la gestion de projet. On peut la comparer à un triptyque pour lequel on commencerait par créer le premier tableau, puis le second et enfin le troisième.

[2] Une méthode incrémentale consiste à procéder par couches successives, comme pour un tableau. On commence par dessiner, puis par appliquer une première couche de peinture et ainsi de suite.

itérative). Puis, pour chaque thème, j'ai travaillé sur les fonctionnalités principales et les ai enrichi (méthode incrémentale) au fur et à mesure des découvertes et des nouveaux besoins communiqués parle client.
Pour chaque itération j'ai ensuite suivi le cycle de développement en V.

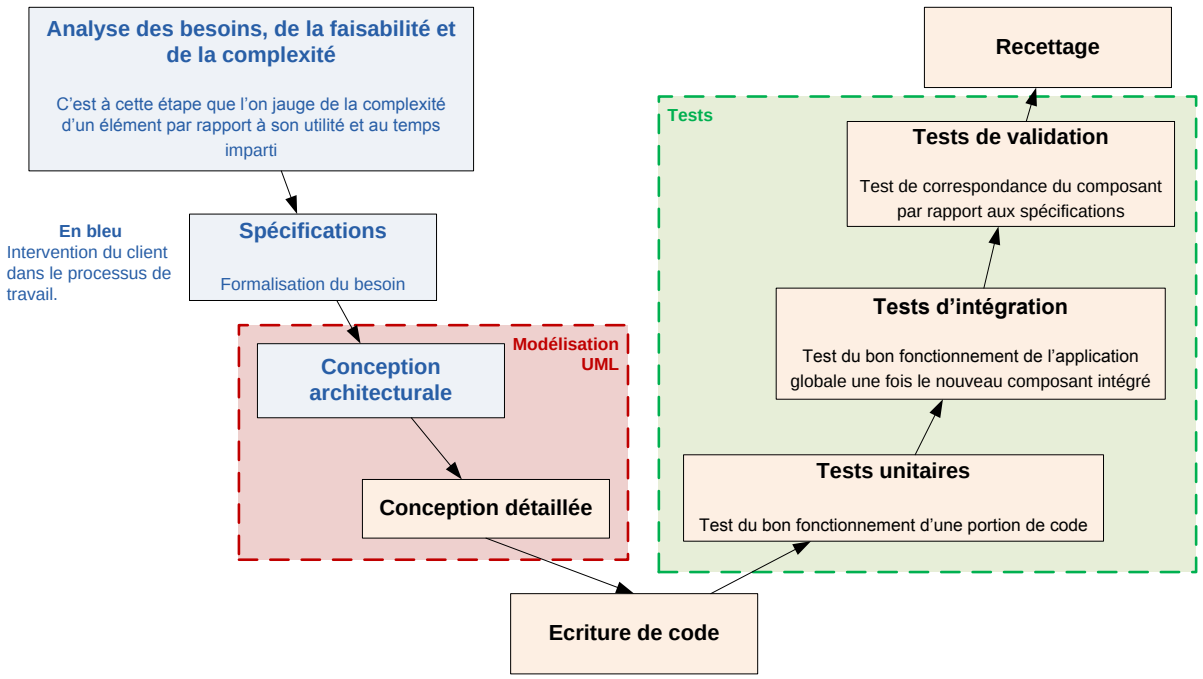


Fig. 5 - Cycle de développement en V

L'intervention du client se fait en début de chaque étape, pour l'analyse et la formalisation du besoin. Comme le temps est limité, on détermine la priorité du module par rapport au système. Ainsi, un module dont l'utilité est moindre sera mis en sommeil à l'étape des spécifications et repris lorsque des modules de priorité plus importante seront achevés.

Comme c'est l'administrateur réseau qui fait office de client, l'étape de conception architecturale sommaire est également effectuée avec son concours car il est nécessaire de tenir compte des éléments déjà établis dans le système ainsi que des interactions futures entre l'application et les services distants (la manipulation du service de travail collaboratif, par exemple).

Il faut noter que le framework de développement rend plus difficile la modélisation UML car une grande partie de son fonctionnement se présente comme une boîte noire qui ne peut pas être rendue sur les documents d'analyse.

J'ai pris connaissance de l'existence des batteries de tests automatisés que très tardivement (JUnit) et ne les ai donc pas utilisées pour les tests unitaires. De ce fait, je n'ai pas opté pour une méthode de développement pilotée par les tests[1] (Test-Driven Development).
Toutefois, l'application génère des rapports réguliers (logs) qui peuvent être activés ou désactivés fonction de leur criticité. Tous les modules rendent ainsi compte de leur activité et des erreurs rencontrées (ainsi que leur cause probable). Cela offre un minimum de suivi aux futurs développeurs / mainteneurs.

Méthode de résolution de problèmes
Aucune méthode de résolution ne m'a été imposée par l'entreprise. Pour chaque cas, j'ai tout d'abord suivi la démarche standard de résolution de problème avant d'exploiter des solutions génériques plus simples. En effet, la méthode étant plutôt réservée à une équipe complète, j'ai simplifié les étapes fonction de la nature des problèmes car, bien souvent, les difficultés rencontrées n'avaient qu'une seule solution possible.

La méthodologie est la suivante :

- Identifier le problème par rapport à l'existant
  Un problème bien posé étant déjà à moitié résolu, il est important de déterminer ce qui est concerné par le problème, qu'est-ce qui est affecté par le problème, où et quand est-il observé, quelle est sa façon de se manifester et quelles en sont les conséquences (dans mon cas, il s'agissait de conséquences en terme de temps).

[1] Le développement piloté par les tests consiste à écrire les tests d'une fonctionnalité avant d'écrire la fonctionnalité elle-même.

- Définir l'idéal

  Il s'agit de comprendre le fonctionnement idéal désiré, de façon rationnelle et réaliste. Cette étape courte peut aboutir à une seule idée et permet de déterminer le chemin à parcourir.

  Par exemple : « Dans deux jours, il faut que l'envoi de mail fonctionne de telle ou telle façon ».

- Rechercher les causes

  Il s'agit de chercher un grand nombre de causes possibles puis de ne retenir que les plus probables.

  Cette phase s'accompagne généralement d'un brainstorming, d'un tri, de validations diverses.

- Chercher les solutions

  Comme la précédente, il s'agit d'une phase de recherche. Une liste des solutions possibles est dressée puis est mise en œuvre.

- Vérifier et valider

  Cette étape consiste à faire le point sur l'action et à vérifier que la solution a correctement résolu le problème sans en avoir soulevé d'autres (batteries de tests).

Etant seul dans le projet, les étapes s'effectuent assez rapidement et tendent souvent vers les mêmes solutions. Ainsi, en fonction de la nature des problèmes, j'ai opté pour des solutions génériques.

Au cours de mon travail, j'ai rencontré et catégorisé les types de problèmes suivants :

- Problème structurel

  L'origine du problème vient d'une ou de plusieurs exigences particulières par rapport à des facteurs particuliers (le problème concerne la structure[1] de l'ensemble).

  Il peut s'agir d'une fonctionnalité irréalisable car trop complexe trop couteuse à mettre en place. Il peut s'agir aussi d'une fonctionnalité ambiguë ou insuffisamment détaillée.

  Solution : Avertir le client (ou le chef de projet, s'il s'agit d'un problème structurel purement technique) et l'informer sur le problème rencontré. Si cela est nécessaire, des réunions peuvent être organisées pour revoir les exigences soit en supprimant la fonctionnalité soit en l'adaptant de façon à simplifier sa mise en œuvre.

- Problème technique informationnel

  L'origine du problème provient d'une connaissance manquante ou incomplète. Il peut s'agir aussi d'une information externe (méconnaissance du langage, par exemple) ou interne à l'entreprise (méconnaissance du fonctionnement de l'existant, par exemple).

  Solution : Acquérir ou perfectionner la connaissance nécessaire par le biais d'une documentation (documentation officielle, livre sur le sujet, contact d'une personne disposant de l'information). Les connaissances peuvent aussi s'acquérir de manière pratique avec des tutoriaux ou, idéalement, des formations.

  Si pour une raison quelconque la connaissance en question ne peut être acquise, le problème n'est plus considéré comme informationnel mais plutôt comme une difficulté structurelle (voir point précédent).

[1] Désigne les informations fonctionnelles (ce que désire le client) et techniques telles que l'architecture logicielle (et matérielle si nécessaire), l'algorithmie, la cohérence de l'ensemble, etc.).

- Problème technique

  L'origine du problème est inhérente à la programmation : à un moment précis, l'applicatif n'a pas le comportement prévu.

  Solution : Il n'existe pas de solution type car les problèmes peuvent varier du plus trivial au plus vicieux.

  En général, une phase de débogage est enclenchée pour tenter de déterminer quelles sont les causes du problème. Cela consiste en l'utilisation d'outils adaptés, la lecture des journaux d'activités, le lancement d'une batterie de tests, etc.

Planification des tâches

La première partie du stage concerne l'acquisition de connaissances car la plupart, si ce n'est leur intégralité, des technologies imposées pour le développement de l'application me sont inconnues.

A ce niveau, le client m'a fourni des documentations diverses concernant JEE et Struts puis m'a orienté vers telle ou telle piste de réflexion.

Certaines de ces documentations et ces technologies évoluant assez vite, une partie des informations n'étaient plus à jour (éléments dépréciés, changements structurels, apparition de nouveaux éléments plus efficaces, etc.). De ce fait, il m'a été nécessaire de les rafraîchir en consultant de nouvelles sources (livres, mises à jour des documentations, documentations techniques officielles), ce qui m'a amené à reconcevoir régulièrement certains aspects architecturaux.

	Temps estimé (jours)	Temps réel (jours)
Rafraichissement des connaissances sur le langage Java	2	2
Découverte des spécifications JEE et expérimentations	5	8
Découverte de Struts et expérimentations	5	4
Etude comparative des serveurs d'applications	4	4
Total	16	18

Tableau 1 - Planification de la phase d'acquisition de connaissances

Dans un second temps, le client et moi-même avons étoffé le cahier des charges pour déterminer quels étaient les besoins et les solutions proposées. Ce cahier des charges a évolué au fil du temps car de nouveaux besoins apparaissaient, sans toutefois perturber les anciens.

Au niveau du développement (analyse, écriture de code, tests), la planification fut difficile car il était entendu à l'avance que le projet était trop vaste pour être mené à son terme en trois mois. Aussi, l'estimation du temps de développement s'est faite non pas en jugeant du temps nécessaire mais en calculant le temps restant après l'estimation des autres étapes.

	Temps estimé (jours)	Temps réel (jours)
Mise au point sur le cahier des charges et les fonctionnalités	3	3
Réflexion architecturale globale	7	6
Réalisation d'une maquette de l'application générale	3	3
Développement itératif de l'application (voir cycle en V)	57	55
Présentation du produit	1	1
Total	76	74

Tableau 2 - Planification de la phase d'analyse - développement

L'installation du logiciel sur les machines serveur n'a pas été prise en compte dans l'estimation car il a été décidé que ce serait le client qui prendrait en charge les opérations de déploiement finales.

3. Choix techniques

Les choix techniques sont divers du fait de la nature de l'application. Il était nécessaire de déterminer un langage de développement, un framework étendant ce langage pour le web, un serveur d'applications pour faire fonctionner le programme final, etc.

Le langage de développement

Il existe une grande quantité de langages conçus pour développer des applications web. Tous sont adaptés à des situations particulières, privilégiant des aspects particuliers comme la simplicité, la robustesse, la légèreté, le dynamisme, etc.

Bien qu'ils soient tous conçus pour fonctionner dans une optique client/serveur, on distingue généralement les langages conçus pour fonctionner plutôt côté serveur et les langages conçus pour fonctionner du côté du client.

Le choix du langage de développement a été fait par l'entreprise. Il s'agit du langage Java et des spécifications JEE. Ce choix a été motivé car le langage java est portable et car les outils respectant les standards édictés pour JEE sont gratuits.

Bien que JEE permette nativement de développer des applications web, l'entreprise a jugé bon d'y joindre un framework, Struts, conçu pour permettre au programmeur de créer des applications web de manière très structurée.
Ce framework a été choisi car le couple JEE / Struts a déjà fait ses preuves en entreprise, privilégiant la modularité et facilitant l'amélioration et la maintenance du code source produit.

Le serveur d'applications

Pour s'exécuter côté serveur, l'application écrite en java nécessite un serveur d'applications ; un logiciel chargé d'héberger des applications accessibles et de gérer les requêtes des clients.

Un serveur d'applications intègre généralement un serveur HTTP.

Le choix du serveur d'applications m'a été confié. J'ai tout d'abord commencé par chercher quels serveurs d'applications étaient compatibles avec la dernière norme JEE. De même, certains serveurs d'applications se limitent à certaines parties de la spécification (Tomcat, par exemple).

- JBoss
- Websphere
- GlassFish / Sun Java System Application Server
- Apache Geronimo
- JOnAS
- Oracle Weblogic
- SAP Netweaver
- JEUS 6

J'ai ensuite recherché des serveurs d'applications libres et gratuits. Ainsi, Weblogic et Websphere ont été éliminés en raison de leur coût élevé.

Les autres critères qui m'ont paru essentiels étaient la fréquence de mise à jour ainsi que la qualité de la documentation. J'ai donc éliminé JEUS dont la dernière mise à jour datait de 2007 et JOnAS qui n'a été certifié JEE 5 que depuis mars 2009.

Ensuite, j'ai cherché un serveur d'applications qui réponde spécifiquement aux besoins de l'application (JSP et EJB) et ai éliminé Netweaver qui m'a semblé un peu trop vaste[1] pour ce dont j'avais besoin.

Au final, j'ai opté pour Sun Java System Application Server (basé sur Glassfish). En effet, j'ai pensé qu'étant donné que Sun mettait au point les spécifications JEE, il s'agissait de l'entreprise toute indiquée pour fournir un serveur d'applications intégralement certifié et régulièrement mis à jour. De plus, ce serveur d'applications est entièrement pris en charge par l'environnement de développement NetBeans que j'avais pour habitude d'utiliser auparavant.

Application riche et services web

Une petite partie de l'application devait fonctionner sous forme de service web, pour pouvoir charger des données de manière dynamique depuis les clients. En effet, certaines informations étant longues à obtenir côté serveur, il était nécessaire de prévoir un mécanisme permettant à l'internaute de continuer à naviguer pendant le chargement desdites informations.

Pour cela, l'idée d'utiliser un service web a été retenue. Cela fut d'autant plus judicieux puisque JEE propose nativement des outils pour concevoir rapidement et efficacement ce genre de choses.
Toutefois, si la création des services web paraissait relativement simple, il fallait qu'une partie des pages côté client intègre un composant dit « riche » pour pouvoir y accéder.

Une Rich Internet Application (RIA) est une application web présentant les mêmes caractéristiques qu'une application traditionnelle (dynamisme, continuité des actions, etc.).

[1] Le produit est aussi compatible avec la technologie Microsoft .NET et est axé développement d'applications orientées services (SOA).

En général, les applications web classiques fonctionnent sur une architecture de client léger où le serveur effectue toutes les opérations et où le client ne s'occupe que de la présentation statique.
Nominalement, le client envoie ses données au serveur qui les traite avant de renvoyer un résultat sous forme de pages. L'opération d'envoi de données (requête HTTP) est limitée par deux types d'évènements :

- La demande d'une page.
- L'envoi de formulaire.

Les RIA s'efforcent donc de rapatrier en local une partie des traitements jadis effectués par le serveur. De plus, les frameworks de développement adaptés aux applications riches offrent des éléments pour travailler avec de multiples services (et non plus uniquement avec le protocole HTTP), pour gérer un grand nombre d'évènements (mouvement de souris, pointage, clic, saisie clavier), pour concevoir des interfaces graphiques très dynamiques, etc.

L'exemple le plus probant de l'avantage des RIA est celui de la vérification d'éléments saisis par un utilisateur. Ces opérations importantes, qui constituent une bonne partie du code écrit par un développeur, peuvent être effectuées de façon statique (application classique) ou dynamique[1] (application riche) comme le décrit la figure 6 ci-après.

[1] Vérification à la volée.

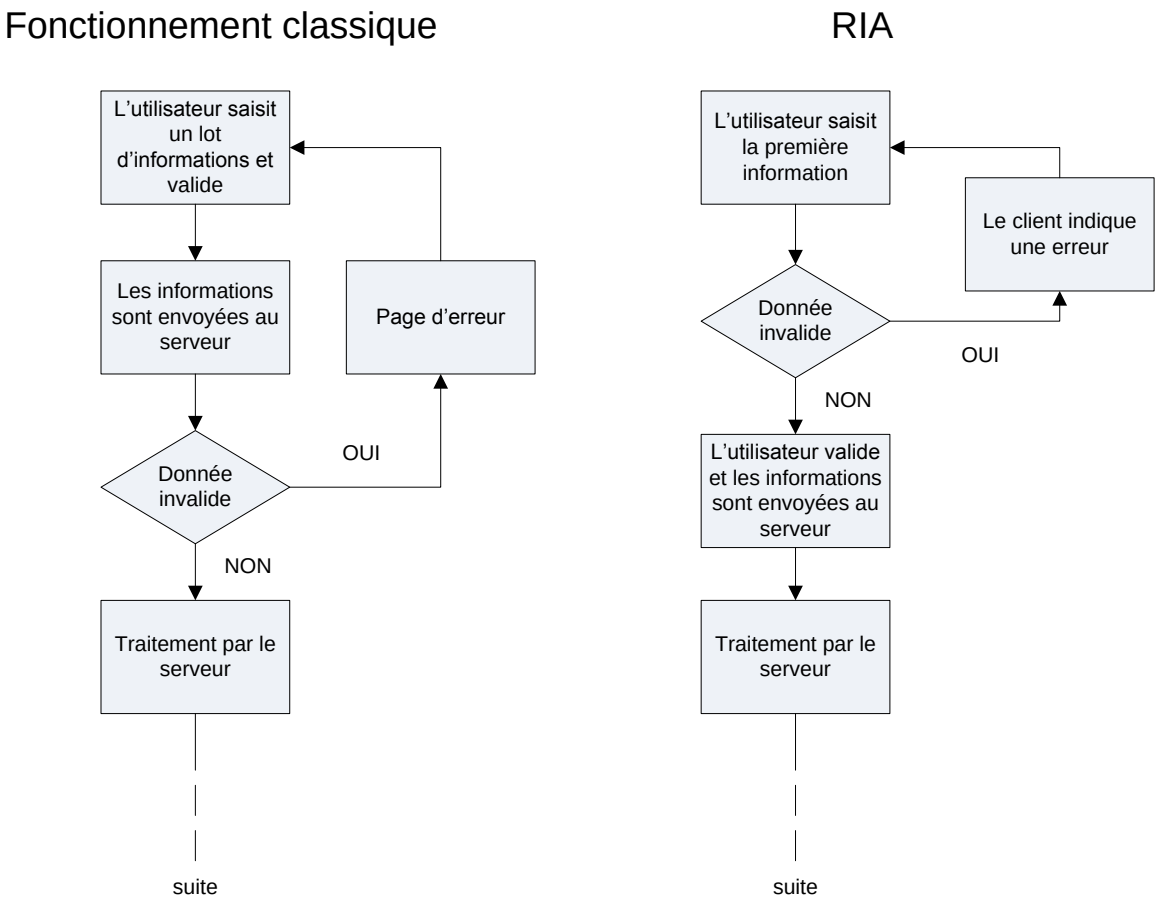


Fig. 6 - Application web classique et RIA

Ainsi, dans le premier cas, le serveur va refuser le lot de données (qui peut être important) à cause d'une seule information invalide tandis que l'application riche va détecter l'erreur lors de la saisie et indiquer au client qu'il doit la corriger avant de valider.

Ainsi, ce dont j'avais besoin c'était d'un framework capable de consulter le service et de présenter le résultat sous forme graphique (diagramme circulaire, par exemple).
Pour ce faire, plusieurs produits spécialisés dans le développement d'applications riches s'offraient à moi :

- AJAX (Asynchronous Javascript And XML)
- Flex / Flash
- Silverlight
- Java FX

Le Javascript ne me paraissait pas adapté car il ne s'agit pas d'un langage spécialement conçu pour créer un code source propre et maintenable (langage à faible verbosité). Ainsi, j'ai rapidement écarté AJAX. Toutefois, il présentait l'avantage de ne pas imposer l'installation d'un module supplémentaire pour l'utilisateur.

De là, j'ai réfléchi au côté pratique et j'ai appris que le plug-in Flash Player est installé, selon les sources d'Adobe, sur plus de 98% des machines.
Dès lors, étant donné que Flex ne nécessite que ce plug-in, les utilisateurs n'ont rien à installer de plus pour faire fonctionner un composant écrit dans ce langage, ce qui n'aurait pas été le cas avec Silverlight ou Java FX.
De ce fait, décidé de me servir de Flex pour développer le composant destiné à communiquer avec le service web de consultation des quotas disques.

Je n'ai pas jugé utile, de par la faible complexité du composant à développer, d'utiliser un framework particulier pour le développement Flex.

4. Les technologies

Java Enterprise Edition

Java Enterprise Edition (JEE) est un ensemble de recommandations et de spécifications techniques destinées à étendre le framework Java Standard Edition (JSE) de Sun Microsystems. JEE est particulièrement destiné aux applications d'entreprise.
Toutefois, il ne s'agit que de spécifications qui sont ensuite supportées en tout ou partie par les serveurs d'applications.

Les spécifications JEE portent sur quatre éléments particuliers :

- **Un ensemble d'API** fournissant au développeur des opérations génériques qu'il n'aura donc pas à redévelopper ou à rechercher. La plupart de ces librairies sont disponibles aussi avec JSE (Collections, etc.).
  Les opérations fournies peuvent être des opérations sur des bases de données (JDBC), des opérations sur des annuaires (JNDI), des opérations pour la gestion d'email (JavaMail), des opérations pour traiter des données XML (JAXP), des opérations pour la communication synchrone (JAX-RPC) ou asynchrone (JAXM, Java RMI, JMX), etc.

- **Les Servlets**. Il s'agit d'applications ou de fragments d'application qui s'exécutent au sein d'un serveur HTTP (plus précisément dans un conteneur de servlets). Elles peuvent traiter les requêtes HTTP reçues par le serveur et retransmettre des données dans les réponses HTTP. Etant donné qu'il s'agit du langage Java, les servlets peuvent exploiter toutes les API fournies en standard.
  Le comportement obtenu est à rapprocher de celui d'un script CGI à ceci près qu'une servlet est exécutée au sein d'un processus léger au lieu d'un processus lourd.

- **Les JSP (Java Server Pages)**. Il s'agit d'une technologie destinée à générer dynamiquement des pages web (HTML, XML, etc.). Ces pages se présentent sous la forme d'un document web classique dans lequel il est possible d'intégrer des portions de code Java qui seront traitées par le serveur avant l'envoi vers le client.

  Les évolutions des spécifications JEE ont fait apparaître des éléments supplémentaires destinés à minimiser l'injection de code java. Il est possible, en effet, de créer des balises personnalisées (Taglibs) ou d'accéder facilement à des composants java (Expression Language).

  Il est intéressant de noter que les JSP sont une extension des servlets. En effet, les pages sont converties en servlet et compilées avant d'être traitées.

- **Les Enterprise JavaBeans (EJB)**. Il s'agit de composants logiciels distribués (i.e pouvant être déployés sur des serveurs distants) dont la logique de communication (RMI, CORBA, RPC) est laissée au serveur d'applications (et plus spécifiquement à un conteneur d'EJB).

  On distingue trois sortes d'EJB :

  - Les EJB Entité, qui permettent de représenter des données et sur lesquels un langage de requête proche du SQL (EJBQL) permet d'effectuer des opérations.
  - Les EJB Session, qui permettent de proposer des services et peuvent être convertis rapidement en service web. Ils peuvent aussi enregistrer des états (Stateful).
  - Les EJB Message, qui permettent d'accomplir des tâches de manière asynchrone (nouveau processus côté serveur).

Au final, ces grandes fonctionnalités s'articulent entre elles d'une façon particulière comme décrit sur la figure 7. Des éléments supplémentaires (API, frameworks, etc.) peuvent être ajoutés pour accroître l'aspect modulaire des

applications et faciliter la distribution des différentes couches sur des machines distinctes.

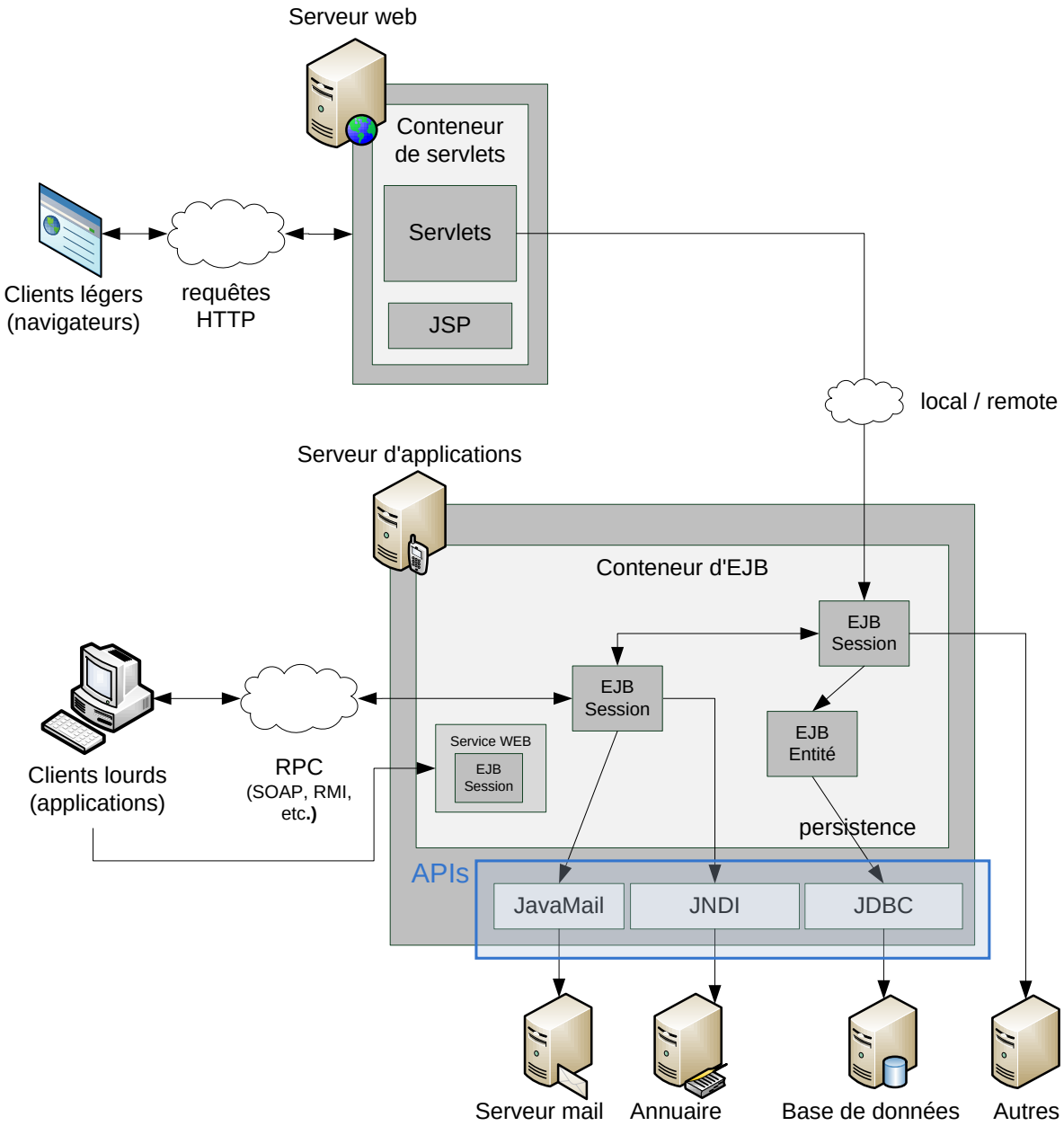


Fig. 7 - Description brève de l'architecture JEE

Struts, le framework de développement web

Comme indiqué plus haut, le choix de Struts a été effectué avant mon arrivée. En effet, il s'agit d'un framework mature et plutôt bien documenté, qui se révèle plutôt adapté aux applications d'une certaine envergure.

Pour information, on distingue notamment, au niveau des concurrents les frameworks suivants :

- Spring MVC
- Tapestry

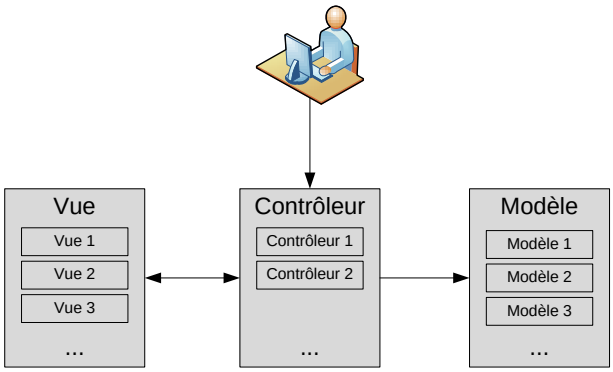


Fig. 8 - L'architecture MVC

Struts aide le développeur à respecter une architecture de type Modèle-Vue-Contrôleur version deux (MVC2) qui découple simplement les données (le modèle) de l'interface utilisateur (la vue). Une troisième couche logicielle (le contrôle) concentre l'ensemble de la logique métier.

Le principal avantage de cette architecture est de pouvoir travailler sur une couche sans interférer sur les autres.

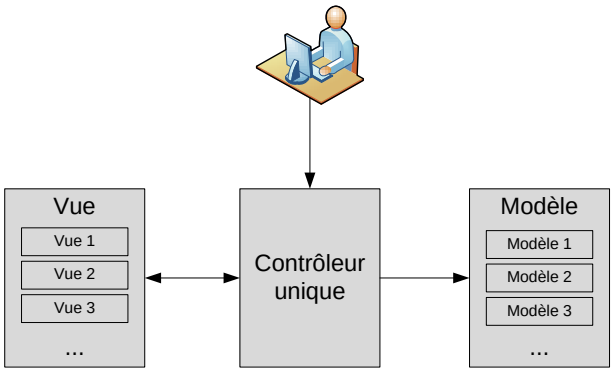


Fig. 9 - L'architecture MVC2

L'architecture MVC2 exploitée par Struts est très proche et consiste simplement en la fusion de tous les contrôleurs en un seul pour faciliter le développement des applications.

Struts intègre dans ce modèle un ensemble de fonctionnalités particulières. Par exemple, les pages et leurs traitements sont décrits sous forme d'actions de différents types. Struts permet aussi d'automatiser certains aspects redondants du développement web comme, par exemple, la vérification des données reçues depuis un formulaire (validateurs) ou la gestion d'une session utilisateur.

Struts met aussi à disposition des technologies développées séparément. Il s'agit notamment de Tiles, un moteur de templates permettant de définir des pages génériques pré-remplies et réutilisables d'une vue à une autre.
Le fonctionnement de Tiles est particulier et diffère du comportement des taglibs d'inclusion fourni par la JSTL. Un template est alimenté par Tiles puis est converti en une servlet supplémentaire qui va générer le code lié au template.
Ce système présente l'inconvénient majeur de ralentir le premier affichage d'une page du fait de la compilation à la volée de la servlet.

Struts fournit aussi un ensemble de taglibs pour répondre à un grand nombre de besoins différents. Il est d'ailleurs possible de ne plus utiliser d'HTML mais uniquement des taglibs fournis par Struts (l'intérêt étant d'étendre ou de simplifier le fonctionnement de certains mécanismes, notamment au niveau des formulaires).

L'intégration des fonctionnalités dans le MVC2 sont décrite ci-après par la figure 10.

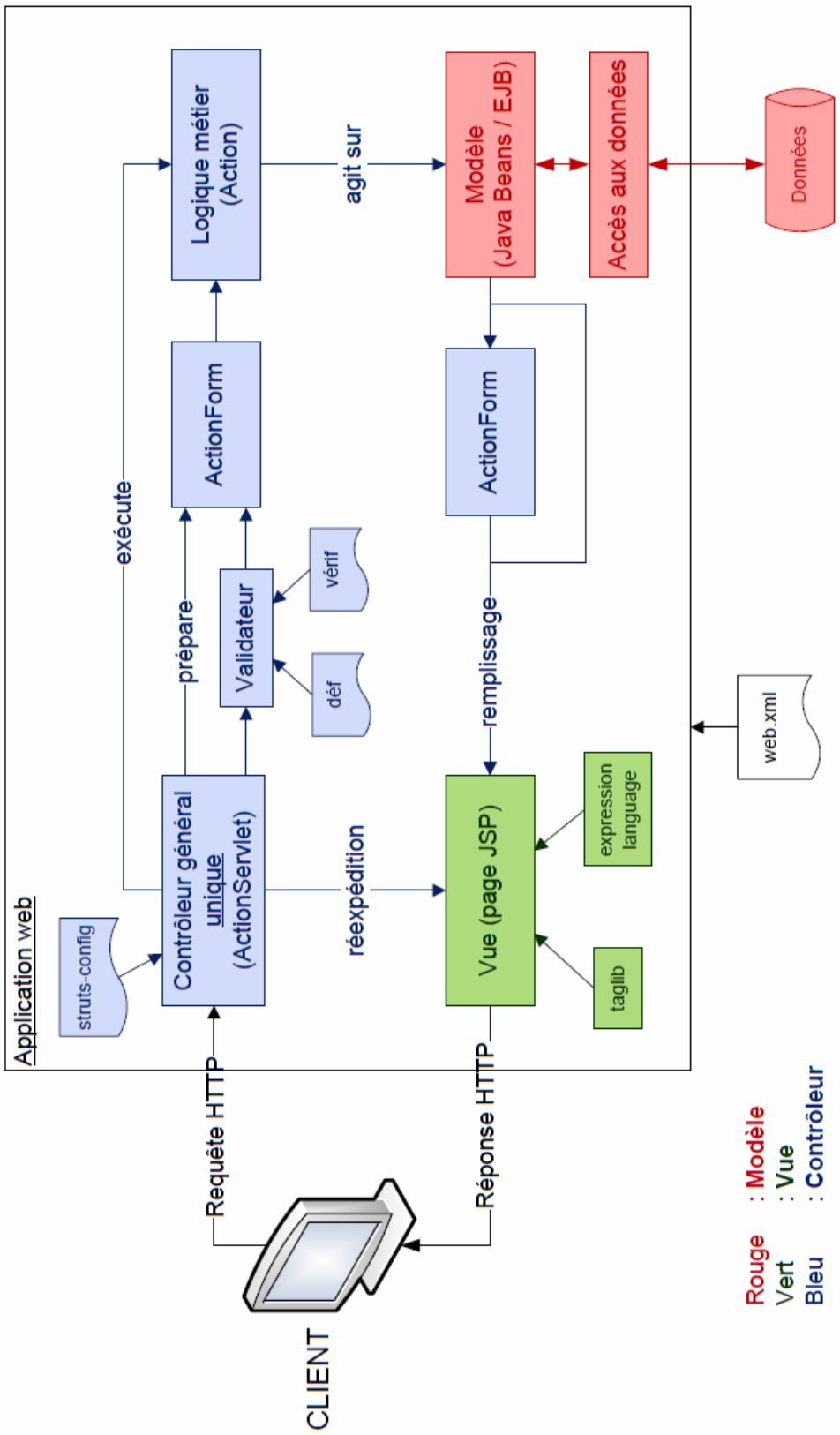


Fig. 10 - L'architecture Struts

Flex et les services web

Flex est une solution de développement relativement récente (2004) permettant de créer et de déployer des RIA grâce à la technologie Flash, sans avoir à passer par l'éditeur graphique jusqu'à lors utilisé.
Flex est basé sur deux éléments clés :

- L'Action Script
  Il s'agit d'un langage de programmation orientée objet dont la syntaxe est plus ou moins proche de Java. Historiquement, il est apparu en même temps que Flash dont il faisait partie intégrante.
  Avec ce langage, il est possible de développer entièrement une application Flex sans passer par du MXML.

- Le MXML
  Il s'agit d'un langage de description pour concevoir des IHM (interfaces graphiques) riches. Celui-ci est basé sur la syntaxe XML et fournit des balises particulières pour représenter un certain nombre de composants propres à l'Action Script.

Un document MXML n'est qu'une autre façon d'écrire de l'Action Script. En effet, le document MXML est converti en code Action Script avant d'être compilé comme une application classique.

En dehors de toute implémentation technique, un service web est caractérisé par quelques principes de base :

- Ils fonctionnent sur le protocole HTTP, un support universel, connu et maîtrisé.
- Ils fonctionnent via un mécanisme d'appel et de réponse.
- Ils communiquent au moyen d'une syntaxe basée sur XML.

L'engouement autour des services web par l'interopérabilité qu'ils apportent, réduisant la dépendance entre les applications et les systèmes d'exploitation.
De plus, ils permettent de créer des applications vues comme un ensemble de services métiers structurés et facilement décrits qui, de plus, dialoguent de manière standardisée et ouverte avec ceux qui les utilisent.

Grâce au découpage apporté par les services web (architectures orientées services), la maintenance et l'évolution sont simplifiées, permettant de travailler sur un service en particulier ou de le changer complètement sans toutefois perturber le reste de l'application.

Néanmoins, les services web possèdent aussi des inconvénients notamment car leurs performances sont réduites vis-à-vis d'autres approches de l'informatique répartie comme RMI, DCOM ou CORBA.
Du reste, l'utilisation du XML et de standards ouverts favorise la compréhension d'éventuels individus malintentionnés, d'autant plus que le protocole HTTP n'est que rarement filtré par les pare-feux.
Enfin, leur jeunesse et leur manque de maturité (normes récentes) font souvent l'objet de reproches de la part des spécialistes.

5. Architecture

Comme évoqué précédemment, l'architecture a été conçue de façon à être modulaire, chaque module suivant le principe de fonctionnement d'une application Struts à part entière (description des actions et des validations, logique métier, pages web, configuration, EJB et contrôleurs, entités).
Tous les modules sont rassemblés au sein de l'application (qui peut être vue comme un module maître) qui va factoriser toutes les opérations communes aux modules (outils d'accès, gestion d'utilisateurs, etc.).

Certains éléments architecturaux ont été imposés dès le départ du projet, notamment l'utilisation la plus judicieuse des EJB Session pour créer une couche supplémentaire de traitement.

Modules
D'après le document fourni à mon arrivée et étoffé au fur et à mesure, j'ai isolé les modules suivants :

- Noyau
  - Démarrage et surveillance du système.
  - Authentification des personnes, gestion des droits et des sessions.
  - Interface(s) d'accès aux outils et aux paramètres pour les autres composants.

- News
  Diffusion d'informations brèves (les news) et catégorisées.

- Administration de comptes
  Opérations relatives à l'administration des comptes, telles que :
  - Quotas
  - Connexion wifi
  - Informations personnelles

- Inscription MSDNAA
- Inscription VMWare
- Bases de données
- Travail collaboratif

- Stages

  Gestion des offres de stages et des demandes de conventions.

- Suivi

  Assurer le suivi des étudiants (rapport de stage pour les CSII1 et CSII2, mémoire pour les CSII3, etc.).

- Cursus

  Informations relatives au cursus scolaire et à l'école (documents statiques).

- Contact

  Contact par mail d'étudiants et de professeurs.

Fonctionnement global

Pour gagner du temps, il ne m'a pas été imposé l'utilisation d'EJB Entités pour le mapping objet-relationnel. Ainsi, j'ai écrit l'accès aux données dans une optique PAC[1], au moyen d'entités (POJO classiques) et de contrôles (manipulateurs d'entités) comme détaillé un peu plus loin sur la figure 11.

Les entités représentent, par exemple, un tuple de base de données, un fichier ou une entrée d'annuaire tandis que les contrôles fournissent les opérations de base sur ces entités (sélection, ajout, édition, suppression) en respectant les contraintes relationnelles.

Les contrôles contiennent donc la logique d'accès aux données, il suffit d'écrire un nouveau contrôle pour changer la source de données sans toucher aux entités.

[1] Présentation – Abstraction – Contrôle

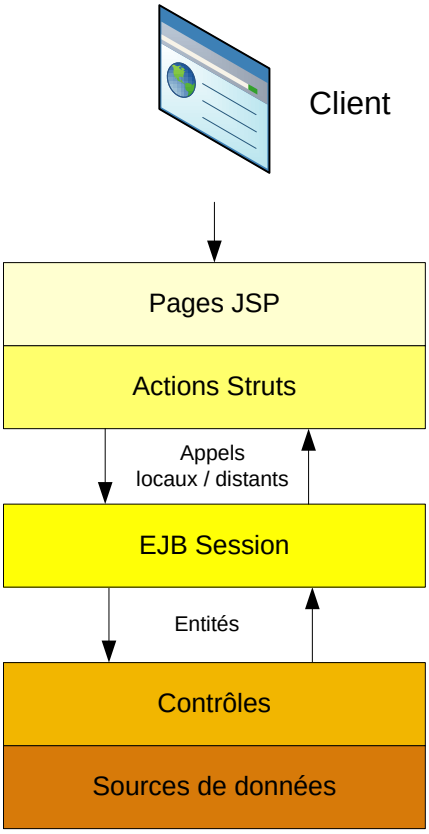


Fig. 11 - L'architecture vue en couche

Dans un souci de développement multicouche, les EJB sont une couche supplémentaire par-dessus les contrôles. Cela permet de rendre les contrôles potentiellement utilisables de manière distante, ou de les transformer facilement en services web.

Au final, de manière schématique, l'architecture d'un module donne ceci :

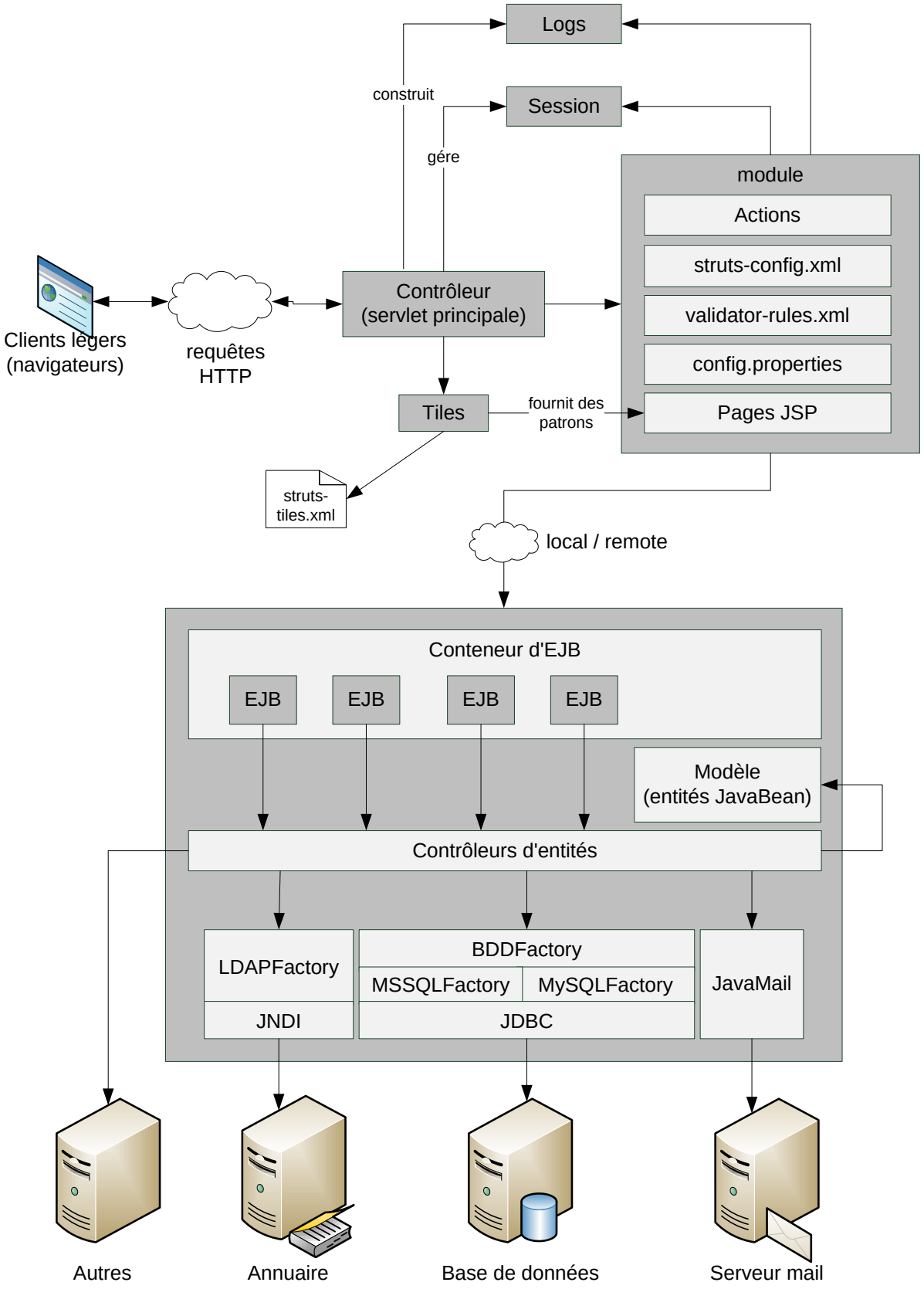


Fig. 12 - Architecture logicielle du portail

V. RESULTATS

1. Faire le point

Pour bien apprécier les résultats, l'idéal est de confronter le nouveau portail à ce qui existait avant.
A l'origine la partie du système d'information concerné par ma mission comprenait les défauts suivants :

- Deux portails dispersés, écrits dans des langages ne favorisant pas leur évolution.
- Plusieurs processus fastidieux, souvent méconnus des étudiants et nécessitant l'intervention de tiers.

On peut dire que ces défauts, qui sont à l'origine de la mission, n'existent plus dans le nouveau portail.
En effet, désormais le produit conçu pendant le stage est désormais constitué :

- D'un unique portail (ou du moins d'une base pour celui-ci), développé au moyen de techniques strictement codifiées et connues pour leur capacité d'évolution.
- De nombreux processus automatisés ne nécessitant plus que le concours des étudiants (autoriser une adresse sur le réseau, remplir une demande de convention de stage, créer une base de données ou un espace de travail collaboratif, etc.).
- D'un système plus simple, plus facile à maintenir et en partie allégé ou optimisé (corrections sur des bases de données).

Toutefois, il était convenu dès le départ que trois mois seraient insuffisants pour parvenir à réaliser toutes les fonctionnalités initiales du cahier des charges, d'autant plus que de nouvelles demandes sont apparues au fil du temps.

Malgré tout, je pense que l'objectif principal a été rempli, à savoir la réalisation d'une base non seulement suffisamment aboutie pour être mise en production mais aussi suffisamment claire et structurée pour que d'autres développeurs puissent y travailler ultérieurement.

Ainsi, lors de la phase d'acquisition de connaissances, j'ai pu sélectionner les informations utiles au niveau des documentations fournies et, lorsque cela s'avérait nécessaire les compléter au moyen de nouveaux documents ou d'une bibliographie utile. De ce fait, les programmeurs néophytes dans le développement JEE/Struts perdront moins de temps lorsqu'ils reprendront mon travail.

En terme de développement, une bonne partie des fonctionnalités décrites dans le cahier des charges ont été analysées et mises en place. Pour les autres, j'ai pris soin de créer les maquettes en vue de leur future implémentation.

A l'heure où j'écris ces lignes, le portail n'a pas encore été déployé faute de temps. Toutefois, cette opération est prévue d'ici quelques semaines.

2. Support

Au niveau du support, j'ai laissé à l'entreprise les documents suivants :

- Le cahier des charges

  Celui-ci contient les demandes initiales, les cas d'utilisations et les descriptions textuelles de chaque fonctionnalité, ce pour chaque module.
  De plus, ce document contient les diagrammes de navigation, intégrant les algorithmes, les vérifications et les indications particulières.

  Ce document a été étoffé au fur et à mesure du travail, lors de chaque phase d'analyse préliminaire.

- Le dossier de développement

  Ce dossier contient, pour chaque module, les diagrammes de classe concernant les entités et les contrôles et leurs interactions réciproques.
  De plus, celui-ci contient les diagrammes modélisant les tables de la base de données, ainsi que des comptes rendus des modifications apportées à celles-ci (corrections, optimisations, ajouts et retraits).

  Ce dossier contient aussi la description des taglibs mis en œuvre pour l'application ainsi que des indications sur l'utilisation des outils développés (gestion de bases de données, d'annuaire, de mail ou de fichiers, opérations de conversion, etc.)

  A ceci s'ajoute aussi toutes les informations concernant la structure générique des pages, les manipulations à effectuer pour créer de nouveaux modules (ajout dans le fichier de configuration, architecture, etc.) ainsi que toutes sortes d'informations relatives à des points particuliers du développement.

Enfin, j'ai joint avec ce document une compilation d'autres documentations utiles (création de taglibs, utilisation des EL, mise en œuvre de la librairie standard de JEE, utilisation de Tiles, classes Action pour Struts, etc.) dans le but d'aider les futurs développeurs à prendre plus rapidement en main les technologies.

- La planification des tâches
  Ce document assez court présente une liste détaillée des fonctionnalités et des éléments sous-jacents.
  Il indique ce qui a été implémenté et précise ce qui manque ou ce qui doit être corrigé (erreurs observées à la dernière minute, etc.).

Je n'ai pas rédigé de documentation pour les utilisateurs car, comme l'application est simple à utiliser (aide contextuelle, etc.), j'ai jugé que le temps passé à la rédaction d'une telle documentation était trop important par rapport au bénéfice final que l'utilisateur pouvait en tirer.
Pour les mêmes raisons, je n'ai pas assuré de formation aux utilisateurs.

La documentation purement technique peut être générée automatiquement à partir du code. En effet, j'ai pris soin de renseigner les diverses informations associées à Javadoc, le générateur automatique de documentation pour Java.

VI. DISCUSSION

La mission qui m'a été confiée concernait un certain nombre de besoins réels pour l'entreprise. De ce fait, plusieurs choses ont été mise en œuvre afin d'assurer le bon déroulement de la mission.
Les ressources nécessaires au développement m'ont été accordées par M. BOMPARD qui a su se montrer disponible pour répondre à mes demandes et à mes questions, ce malgré son emploi du temps assez chargé.
De plus, au cours de la mission j'ai bénéficié d'une grande liberté au niveau des solutions techniques et pratiques. Cela m'a forcé à faire des choix et à découvrir beaucoup de nouvelles choses et, de ce fait, à améliorer perpétuellement ma façon de procéder.

Pour ma part, je pense avoir fourni un travail propre et rigoureux dans la mesure du possible en sachant que les vastes techniques employées m'étaient totalement étrangères et en constante évolution. Quoi qu'il en soit, j'ai toujours réfléchi dans une optique de structuration et de découplage des éléments, ce pour pouvoir faciliter la reprise par d'autres personnes.

En ce qui concerne l'application proprement dite, j'ai rempli une grande partie des objectifs initiaux, bien qu'il reste des choses à terminer et à améliorer, ce qui nécessitera encore sans doute plusieurs mois de travail.
De ce fait, j'ai remarqué, avec le recul, que certaines choses auraient sans doute pu être améliorées, notamment au niveau de la gestion de projet.

En effet, malgré une planification (certes sommaire) et l'exploitation de techniques comme le développement itératif incrémental ou le cycle en V, il n'y a pas eu de démarche de suivi de projet proprement dite.
De nombreux imprévus, comme par exemple le remaniement incessant du code au fur et à mesure des découvertes technologiques, l'arrivée de nouveaux besoins perturbant l'ordre de priorité établi ou encore les retards pris sur la fabrication de certains modules, auraient pu être anticipées et gérées plus facilement avec une

gestion de projet efficace. En outre, une bonne gestion de projet aurait permis de mettre en évidence la criticité de certains points et de prévoir des marges de manœuvre relatives aux difficultés potentielles.

De mon point de vue personnel, je pense que cette mission m'a apporté beaucoup de connaissances sur des domaines que je ne maîtrisais pas (c'est notamment pour cela que je l'avais choisie), m'obligeant à m'adapter à de nouvelles connaissances fondamentalement différentes de ce que j'avais pu rencontrer jusque-là. Au niveau purement technique, j'ai travaillé dans un domaine relativement jeune, du moins en comparaison avec les domaines sur lesquels j'avais l'habitude de travailler. Il s'agissait aussi de technologies porteuses très utilisées aujourd'hui en entreprise comme JEE ou les architectures de services web dans leur ensemble.
Ensuite, le fait d'avoir réalisé une application dans un contexte de multiutilisation (application web) m'a permis d'appréhender de nouvelles contraintes de travail (optimisations fines, réflexion sur la sécurité, structuration, etc.).
Au final, cette expérience m'a permis de confronter de nouvelles idées sur la conduite d'un projet.

De plus, la grande autonomie dont j'ai pu jouir tout au long de la mission m'a appris à faire force de proposition, à expliquer et à défendre mes idées auprès d'un client qui possède les idées directrices. Jusqu'à lors j'avais plutôt l'habitude de travailler en équipe, ce qui change du tout au tout les méthodes de travail.
Cependant, j'avais l'habitude des petites structures de travail et des équipes réduites, ce qui m'a permis d'être réactif assez rapidement.

Enfin, malgré tout ce que j'ai pu apprendre pendant la mission, j'aurais toutefois apprécié aller plus loin dans l'application et réaliser un produit fini. Hélas, c'est toujours en progressant que l'on s'aperçoit que la masse des choses à faire est plus importante que prévu.
Malgré tout, la rédaction des documents à destination des autres développeurs m'a aidé à me rendre compte de l'importance réelle du travail accompli.

VII. CONCLUSION

Les trois mois passés à la réalisation de cette mission ont conduit à faire évoluer une situation existante vers quelque chose de nouveau qui semble avoir répondu aux attentes de l'EPSI.
Mon tuteur, M. BOMPARD, a grandement contribué à ce résultat, en m'orientant régulièrement sur des pistes de réflexion et en me fournissant l'aide nécessaire à la découverte des technologies mises en œuvres.

Malgré mes quelques regrets vis-à-vis de la gestion du temps et du projet, les résultats sont concrets et seront très bientôt utilisés en production par l'EPSI. Il est déjà prévu que l'évolution du logiciel soit assurée en fin d'année prochaine par d'autres stagiaires.

Pour ce qui est de l'expérience acquise, je la juge assez importante du fait des technologies que j'ai découvert et longuement étudié. Bien entendu, ce n'est qu'une partie de l'existant - qui, je le rappelle, est en continuel mouvement - mais il s'agit déjà, à mon échelle, d'une réelle progression et d'une ouverture vers des horizons nouveaux.
Ensuite, j'ai pu comprendre, de par l'analyse de l'existant, l'importance de structurer les choses et de laisser des traces à ses successeurs. Cela m'a d'autant plus concerné étant donné que je devais à mon tour laisser des documents concernant mes travaux.

VIII. GLOSSAIRE

Annuaire : En informatique, un annuaire est un système de stockage de données dérivé des bases de données. Celui-ci est utilisé pour stocker des données destinées à être peu modifiées, mais lues régulièrement (coordonnées de personnes, droits utilisateurs, etc.).

API : Ensemble de fonctions, procédures et/ou classes mises à disposition du développeur.

(Design) Pattern : Les motifs de conception décrivent des solutions génériques pour résoudre des problèmes conceptuels et architecturaux régulièrement rencontrés en analyse logicielle.

ERP (Enterprise Ressource Planning) : Un progiciel de gestion intégré est une application permettant de coordonner les différentes fonctions de l'entreprise dans un système d'information unique.

Framework : Ensemble de bibliothèques, d'outils et de spécifications permettant le développement d'applications particulières.

Interopérabilité : On entend par interopérabilité, la capacité à rendre compatibles deux systèmes quelconques entre eux.

Mainframe : Machines dont la puissance de calcul est très élevée. Les mainframes sont aussi appelés *ordinateurs centraux* ou *supercalculateurs*.

Mapping objet-relationnel : Technique de programmation informatique créant l'illusion d'une base de données orientée objet à partir d'une base de données relationnelle au moyen de la définition des correspondances entre cette base de données et les objets du langage utilisé.

POJO (Plain Old Java Object) : Terme utilisé pour désigner un objet java.

« Nous nous sommes demandés pourquoi tout le monde était autant contre l'utilisation d'objets simples dans leurs systèmes et nous avons conclu que c'était parce que ces objets simples n'avaient pas un nom sympa. Alors, on leur en a donné un et ça a pris tout de suite. »

Martin Fowler.

Template : modèle stockant la structure d'une page web pour séparer le fond de la forme. Le template pourra ensuite être réutilisé indépendamment du contenu.

XML (eXtensible Markup Language) : est un langage de balisage qui permet de faciliter les échanges d'informations en prenant en charge l'encodage, les formats, etc.

IX. TABLE DES ILLUSTRATIONS

X. SOURCES

Bibliographie

- *Programmer en Java*, Claude Delannoy.
  Editions Eyrolles – Collection BestOf.
  ISBN 978-2-212-12326-5.

- *Développement Java sous STRUTS*, Jean Charles Félicité.
  Editions ENI France.
  ISBN 2-7460-3105-1.

- *Jakarta Struts*, Chuck Cavaness et Brian Keeton.
  Editions O'Reilly France - Collection Précis&Concis.
  ISBN 2-84177-256-X.

- *EJB 3*, Laboratoire Supinfo des technologies Sun.
  Editions Dunod, Paris.
  ISBN 978-2-10-051831-9.

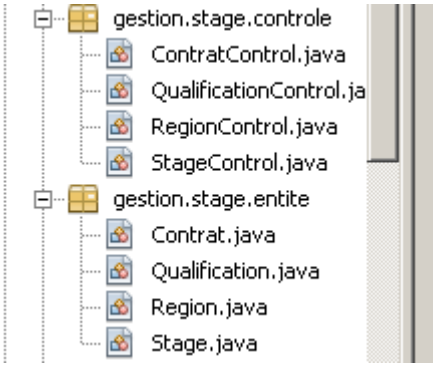
Netographie

- *Programmation web avec Java*, Serge Tahé.
  http://tahe.developpez.com/java/web/
- *Tutoriel Jakarta Struts*, Serge Tahé.
  http://tahe.developpez.com/java/struts/
- *FAQ Struts*, developpez.com.
  http://java.developpez.com/faq/struts/
- *Présentation des JSB Tag Libraries*, F. Martini.
  http://adiguba.developpez.com/tutoriels/j2ee/jsp/taglib/
- *Présentation des Expressions Languages*, F. Martini.
  http://adiguba.developpez.com/tutoriels/j2ee/jsp/el/
- *Présentation de la JSTL*, F. Martini.
  http://adiguba.developpez.com/tutoriels/j2ee/jsp/jstl/

- *Struts Validator Guide*, The Apache Software Foundation.
  http://struts.apache.org/1.2.4/userGuide/dev_validator.html
- *Construire un service web Java EE*, Serge Tahé.
  http://tahe.developpez.com/java/webservice-jee/
- *Javadoc Struts*, The Apache Software Foundation.
  http://struts.apache.org/javadoc.html
- *JEE API Specifications*, Sun Microsystems.
  http://java.sun.com/javaee/5/docs/api/
- *Développons en Java*, Jean Michel Doudoux.
  http://www.jmdoudoux.fr/java/dej/chap000.htm
- *Adobe Flex Tutorial*
  http://www.flex-tutorial.fr/
- *Adobe Flex Language Reference*, Adobe.
  http://livedocs.adobe.com/flex/3/langref/class-summary.html

XI. ANNEXES

Exemple de code source lié à la gestion des stages



Entités et contrôles

```
public class StageControl extends ControleSQL
{
    protected static StageControl instance = null;
    protected final static Long MS_PAR_JOUR = (long)24 * (long)3600 * (long)1000;

    protected final static String SELECT_STAGE = "SELECT DISTINCT * FROM " + Configuration.T
    protected final static String SELECT_N_STAGES = "SELECT DISTINCT offre_stageemploi_id, o

    public static StageControl getInstance()
    throws BDDException
    {
        if(instance == null)
            instance = new StageControl();

        return instance;
    }
```

Un contrôle basé sur la communication avec une BDD

```
 * @author Benjamin BILLET
 */
public class Stage implements Entite
{
    protected Integer id;
    protected String entreprise;
    protected Region region = null;
    public Stage(Integer P_Id, String P_Entreprise, Region P_Region,
    {
        id = P_Id;
        entreprise = P_Entreprise;
        region = P_Region;
```

La classe d'entité Stage

```java
@Remote
public interface EJBGestionStages
{
    public Contrat chargerContrat(Integer P_Id) throws BDDException;
    public List<Contrat> chargerContrats() throws BDDException;

    public Boolean ajouterContrat(Contrat P_NouveauContrat) throws BDDException;
    public Boolean modifierContrat(Integer P_Id, Contrat P_NouveauContrat) throws BDDException

    public Boolean supprimerContrat(Integer P_Id) throws BDDException;
    public Boolean supprimerContrat(Contrat P_Contrat) throws BDDException;

    public Qualification chargerQualification(Integer P_Id) throws BDDException;
    public List<Qualification> chargerQualifications() throws BDDException;

    public Boolean ajouterQualification(Qualification P_NouvelleQualification) throws BDDExcep
    public Boolean modifierQualification(Integer P_Id, Qualification P_NouvelleQualification)

```

L'interface métier pour l'EJB de gestion des stages.

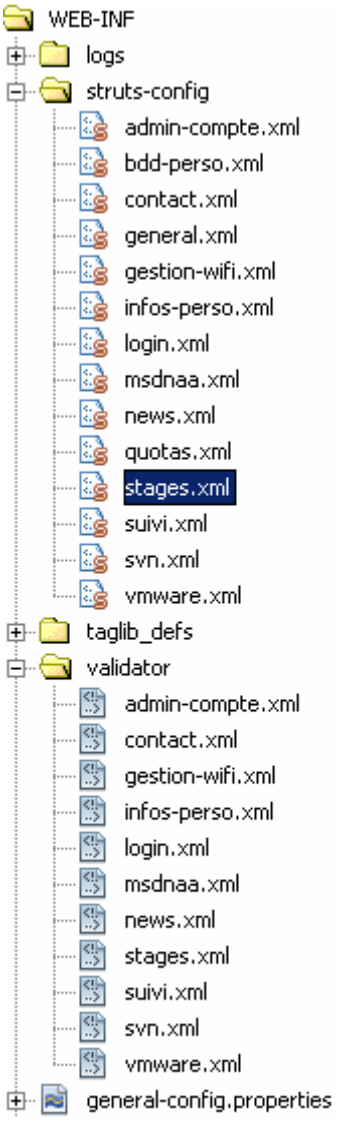
```java
@Stateful
public class EJBGestionStagesImpl implements EJBGestionStages
{
    @Override
    public Contrat chargerContrat(Integer P_Id)
    throws BDDException
    {
        return ContratControl.getInstance().chargerContrat(P_Id);
    }

    @Override
    public List<Contrat> chargerContrats()
    throws BDDException
    {
        return ContratControl.getInstance().chargerContrats();
    }

    @Override
    public Boolean ajouterContrat(Contrat P_NouveauContrat)
    throws BDDException
    {
        return ContratControl.getInstance().ajouterContrat(P_NouveauContrat);
    }

    @Override
    public Boolean modifierContrat(Integer P_Id, Contrat P_NouveauContrat)
    throws BDDException
    {
        return ContratControl.getInstance().modifierContrat(P_Id, P_NouveauContrat);
    }
```

L'implémentation de l'EJB de gestion des stages



Fichier d'actions et de validation

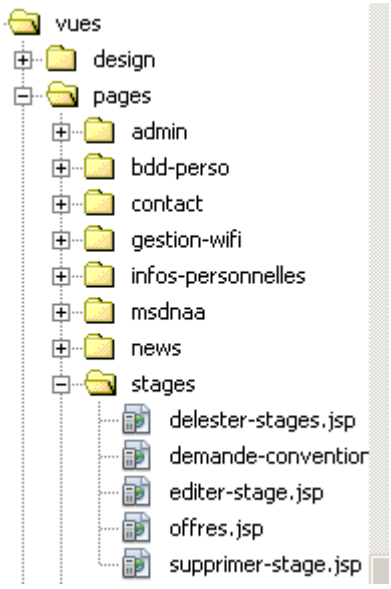
```
        <form-property name="stage_lieu" type="java.lang.String" initial=""/>
        <form-property name="stage_debut" type="java.lang.String" initial=""/>
        <form-property name="stage_fin" type="java.lang.String" initial=""/>
        <form-property name="stage_horaire_debut" type="java.lang.String" initial=
        <form-property name="stage_horaire_fin" type="java.lang.String" initial=""
        <form-property name="stage_gratification" type="java.lang.String" initial=
        <form-property name="stage_description" type="java.lang.String" initial=""
    </form-bean>
</form-beans>

<action-mappings>
    <action
        path="/stages/offres"
        name="FormulaireFiltrageStages"
        scope="session"
        validate="false"
        type="page.stage.StageDispatcher"
        parameter="initPageListeOffres">
            <forward name="offres-stages" path="/vues/pages/stages/offres.jsp"/>
    </action>
    <action path="/stages/offre/ajouter" forward="/stages/offre/editer.html" />
    <action
        path="/stages/offre/editer"
        type="page.stage.StageDispatcher"
        parameter="initFormulaireAjoutEditionStage"
        name="FormulaireAjoutEditionStage"
        scope="request"
        validate="false"
    >
```

Fichier de description d'actions

```
 7  <form-validation>
 8      <formset>
 9          <form name="FormulaireAjoutEditionStage">
10              <field property="titre" depends="required"></field>
11              <field property="entreprise" depends="required"></field>
12              <field property="dates" depends="required"></field>
13              <field property="duree" depends="required"></field>
14              <field property="classes" depends="required, integer, tabTailleMini
15                  <msg name="required" key="generique.erreur.interne"/>
16                  <msg name="integer" key="generique.erreur.interne"/>
17                  <msg name="tabTailleMinimale" key="generique.erreur.interne"/>
18                  <var>
19                      <var-name>seuil</var-name>
20                      <var-value>1</var-value>
21                  </var>
22              </field>
23              <field property="contrat" depends="required, integer, intSuperieurA
24                  <msg name="required" key="generique.erreur.interne"/>
25                  <msg name="integer" key="generique.erreur.interne"/>
26                  <msg name="intSuperieurA" key="generique.erreur.interne"/>
27                  <var>
28                      <var-name>seuil</var-name>
29                      <var-value>0</var-value>
30                  </var>
31              </field>
32              <field property="qualification" depends="required, integer, intSupe
33                  <msg name="required" key="generique.erreur.interne"/>
34                  <msg name="integer" key="generique.erreur.interne"/>
35                  <msg name="intSuperieurA" key="generique.erreur.interne"/>
```

Fichier de description d'un validateur



Vues liées à la gestion de stage

```
45      Region :
46      <html:select property="region">
47          <html:optionsCollection property="valeursRegions" value="id" label="nom"/>
48      </html:select>
49      <html:submit value="Filtrer" />
50  </html:form>
51  <hr/>
52
53  <securite:section-restreinte groupes="Administrateurs">
54      <html:form action="/stages/offres/delester/confirmer" method="post" styleClass="inline
55          <small>
56              <a href="stages/offre/ajouter.html"><img src="vues/design/icones/nouveau_stage
57              - Supprimer les stages vieux de plus de <html:text property="age" size="2" />
58          </small>
59      </html:form>
60      <hr/>
61  </securite:section-restreinte>
62
63  <jstl:choose>
64      <jstl:when test="${requestScope.nbPages > 0}">
65          Page :
66          <jstl:if test="${requestScope.pageCourante > 1}">
68              <a href="stages/offres.html?page=${requestScope.pageCourante - 1}"><img src="v
69          </jstl:if>
70          <jstl:forEach var="numeroPage" begin="1" end="${requestScope.nbPages}">
71              <jstl:choose>
72                  <jstl:when test="${requestScope.pageCourante == numeroPage}">
73                      <strong>[ ${numeroPage} ]</strong>
                    </jstl:when>
```

Code JSP d'une vue

```
#================================================================#
#                        Système de stages                       #
#================================================================#
# Noms des tables (String)
TableStages = stageemploi
TableStagesRegions = region
TableStagesQualifs = qualification
TableStagesContrats = contrat
TableStagesClasses = stageemploi_concerne_classe

# Nombre de stages par page
# Défaut : 20 (Integer)
NombreStagesParPage = 20
```

Configuration de la gestion de stages

```
#Erreurs liées à la gestion des stages
stage.convention.formulaire.telephone.invalide=<li>Le numéro de téléphone doit être de la forme
stage.convention.formulaire.mail.invalide=<li>L'adresse email doit être de la forme x@y.z.</li>
stage.echec.suppression=<li>Impossible de supprimer le stage.</li>
stage.echec.edition=<li>Impossible d'éditer ou d'ajouter le stage.</li>
```

Erreur liées à la gestion des stages (ressources-bundles)

MappingDispatchAction liées à la gestion de stage

```
public class StageDispatcher extends MappingDispatchAction
{
    public ActionForward initPageListeOffres(ActionMapping P_Mapping, ActionForm P_Formulaire,
    throws IOException, ServletException
    {
        try
        {
            DynaActionForm formulaire = (DynaActionForm)P_Formulaire;
            EJBGestionStages gestionStages = (EJBGestionStages)new InitialContext().lookup("ej
            MapAttributs valeurs = Outils.chargerAttributs(formulaire, "classe", "contrat", "q

            List<Classe> classe = null;
            Contrat contrat = null;
            Qualification qualification = null;
            Region region = null;

            /* Récupération des paramètres de filtrage */
            if(valeurs.getInt("classe", 0) > 0)
            {
                classe = new ArrayList<Classe>();
                classe.add(gestionStages.chargerClasse(valeurs.getInt("classe", -1)));
```

Code source d'une MappingDispatchAction