Ingénieurs 2000
Génie Logiciel

Projet eMagine

Pre-cahier des charges techniques

FICHE D'APPROBATION

Version	Date	Rédacteurs			
1.0		L. BARBISAN	A. OGIER	J.-B. RENAUDIN	C. RONDINI
		Chef de projet	Responsable Documentaire	Responsable Client	
		J.-B. RENAUDIN	L. BARBISAN	R. FORAX	

FICHE DE RÉVISION

Numéro de version actuel :

Version	Date	Auteur	Libellé
0.5	15/10/05	L.BARBISAN	Création du document
0.6	18/10/05	L.BARBISAN	Gestion du développement
0.8	19/10/05	L.BARBISAN	Architecture générals logiciels
1	20/10/05	L.BARBISAN	Relecture et correction mineurs, Validation

SOMMAIRE

INTRODUCTION

Ce document a pour but de décrire les solutions techniques générales pour la mise en place de l'applicatif.

Tout d'abord, avant d'exposer les solutions techniques, les différents besoins sont étudiés afin de permettre une sélection de logiciel adéquate.

La seconde partie propose les principales orientations techniques mises en oeuvre pour eMagine, afin de déterminer les solutions. Pour ce faire une architecture générales d'eMagine sera présentée.

1. BESOINS ET CONTRAINTES TECHNIQUES

Le développement d'une application demande beaucoup d'outils pour permettre un gain de temps et de productivité. En conséquence, une étude des besoins techniques de l'application est effectuée afin de viser des conditions optimums de développement. Ainsi, cela permet de déterminer quels sont les outils qui permettront une productivité maximale. En outre, une étude sur les technologies existantes à mettre en oeuvre sera présentée.

1.1. BESOINS POUR LE DÉVELOPPEMENT DE EMAGINE

Le développement en équipe est plus contraignant que le développement simple par une seul et unique personne. L'inconvénient majeur est de pouvoir effectuer des modifications simultanées sur un même document.

Il est également nécessaire de stocker toutes les informations et les ressources utilisées dans le projet afin de permettre un accès rapide.

De plus, un autre besoin est d'aider le développeur dans les tâches itératives facilement réalisables par un ordinateur.

Enfin, il faut limiter au maximum l'apparition des bogues régressifs.

1.2. BESOINS ET CONTRAINTES DE EMAGINE

Les besoins et contraintes imposées par la mise en place d'eMagine permet de trouver les technologies utilisées dans ce projet. Le chapitre 3 explique l'architecture générale de eMagine et mets en avant les différentes solutions envisageables pour respecter les contraintes eMagine.

Les contraintes sont surtout liées à l'architecture client léger/serveur, qui demande de mettre un place un serveur d'application. Le besoin principal dans cette architecture est d'obtenir une réponse rapide du serveur face aux requêtes du client.

2. GESTION DU DEVELOPPEMENT

Pour le développement de eMagine, plusieurs outils ont été mis en place afin de permettre un développement rapide et efficace. Le plus important à noter est l'IDE[1] Eclipse qui permet de centraliser à peut prêt tout les applicatifs nécessaires au développement. Avant les outils, l'organisation de l'information sera présentée pour expliquer où se trouvent les ressources de développement.

2.1. ORGANISATION DE L'INFORMATION

L'organisation de l'information explique où trouver les ressources nécessaires au développement. La gestion de l'information peut être découpée comme suit:

- **Gestion projet**

  Permet de gérer toutes les données liées aux projet.

- **Gestion communication extérieur**

  Permet de communiquer l'état d'avancement du projet au public

- **Gestion des ressources**

  Permet de gérer toutes les ressources utilisées pour créer le logiciel

- **Gestion documentation**

  Permet d'effectuer le suivi de documentation

- **Gestion sources**

  Permet d'effectuer les suivis des sources

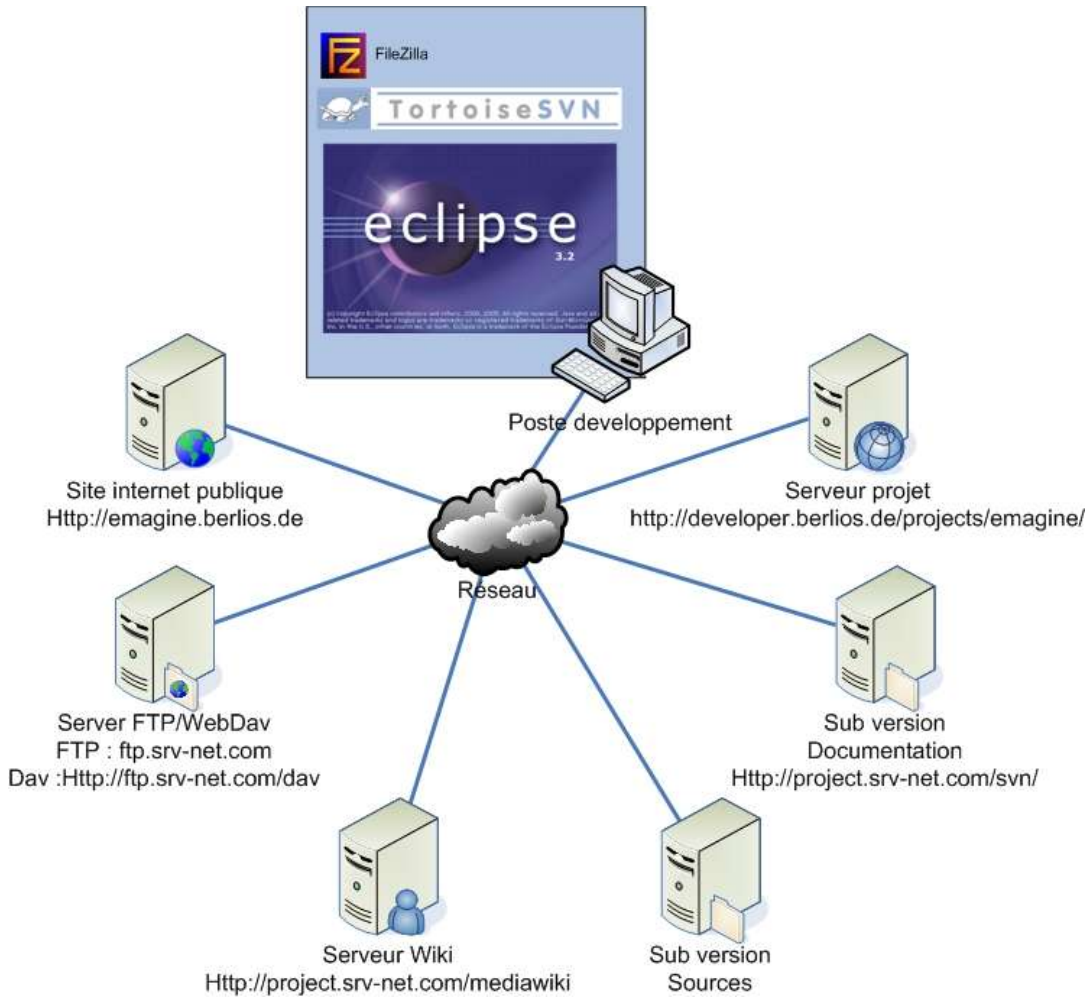
- **Gestion communication Equipe**

  Permet de faire communiquer les membres de l'équipe entre eux.

Le diagramme ci-après montre la répartition des différentes informations

1 **Environnement de Développement Intégré** (*EDI* ou *IDE* en anglais pour *Integrated Development Environment*)

a. Diagramme



b. Gestion de projet : Serveur projet

Le serveur projet est hébergé par berlios.de. Il permet de gérer les tickets de bug, afin de répertorier et historiser chaque problème relevé par les programmeur. Un forum est aussi à disposition des programmeur pour poser des questions, résoudre des problèmes, etc. Il permet aussi de gérer le versionning de sources.

c. Gestion de la communication extérieure : Serveur Site internet

Le serveur internet est hébergé chez Berlios.de. Le site internet permet de suivre le déroulement du développement du projet par le client. Il contient les informations officielles et les téléchargements divers. C'est l'interface entre les acheteurs potentiels et eMagine.

d. Gestion des ressources : Serveur FTP/WebDav

Le serveur Ftp est utilisé pour stocker les ressources qui n'ont pas besoin d'être gérées à l'aide de Sub Version. Notamment les documents figés, comme le sujet du projet, les logiciels utilisés (fileZilla, etc).

e. Gestion documentation : Serveur Subversion documentation

Le serveur de Sub Version pour la documentation permet de stocker tous les documents livrables (les cahiers des charges entre autres). Cela permet d'avoir un historique complet de la rédaction du fichier et de gérer les problèmes de modifications simultanées.

L'accès au serveur se fait via l'utilitaire Tortoise SVN.

f. Gestion sources : Serveur Sub version sources

Le serveur de Sub Version pour les sources permet de stocker toutes les sources d'eMagine. Cela permet d'avoir un historique complet des sources et de gérer les problèmes de modifications simultanées.

L'accès au serveur se fait le plug-in Eclipse SubEclipse.

g. Gestion communication Equipe : Serveur Wiki

Un wiki est un site Web dynamique permettant à tout individu d'en modifier les pages à volonté. Il permet non seulement de communiquer et de diffuser des informations rapidement (ce que faisait déjà Usenet), mais aussi de structurer cette information pour permettre d'y naviguer commodément. Il réalise donc une synthèse des forums Usenet, des FAQ dans le World Wide Web en une seule application intégrée (et hypertexte). Actuellement mediawiki est utilisé par l'equipe eMagine.

2.2. Outils de développement

Les outils développement permettent d'augmenter la rentabilité. Les principaux outils sont les bibliothèques qui permettent de réutiliser du fonction écrite par un autre programmeur.

a. Langage de programmation

Le langage utilisé pour la programmation sera JAVA. Java est un langage de programmation orienté objet qui permet un développement rapide grâce à l'aspect objet. De plus, l'engouement des programmeurs pour le langage java engendre la création de nombreuses bibliothèques pour simplifier les développements. En effet, l'utilisation d'une bibliothèque évite le développement de l'outil nécessaire, et donc tous les tests.

b. Logiciels

- **Logiciel du développement**

Pour l'IDE (Integrated Developpement Environnement), c'est le logiciel eclipse qui a été retenu, c'est un logiciel développé intégralement en java. Il offre une interface très intuitive, qui permet de gagner du temps dans toutes les tâches fastidieuses de développement. eclipse est le logiciel de référence dans le monde du développement java et intègre de nombreux pluggin pour le rendre évolutif.

- **Logiciel de gestion des sources**

La gestion des sources est faite à l'aide de Sub Version. Sub Version est le successeur de CVS déjà très réputé dans le monde du développement Open Source. Sub Version Comble la plupart des failles de CVS. La gestion des sources permet de garder un historique complet des développements. Ainsi s'il est nécessaire de revenir en arrière Sub Version le permet.

c. Plug-ins Eclipse

- **Déploiement de l'applicatif : Ant**

Le déploiement de l'applicatif est géré avec Ant. Ant est un projet développé sous l'égide de Apache par la communauté Jakarta. Jakarta est un des acteurs les plus actifs dans le monde Java. Il est connu pour de nombreuses participations. Notamment POI (gestion des outils Excel et Word). Ant permet d'automatiser les tâches de compilation, de distribution, et d'exécution. Mais Ant permet aussi d'effectuer de nombreuses autre action.

- **Mise en place des tests unitaire: JUnit**

Les tests unitaires permettent de tester l'applicatif, et d'éviter les régression. Il existe une bibliothèque intégrer à Eclipse qui permet de générer et d'exécuter des test unitaires : Junit. Junit permet pour chaque classe java de créer une classe de test associée. Lors du lancement des test unitaires, tous les tests sont lancés. Les tests sont valides si tous les tests se sont bien déroulés sinon les tests ont échoué.

- **Mise en place des test de performance : JMeter**

  Les tests de performance seront réalisés avec Jmeter. Jmeter est aussi une application développée par Jakarta.

- **Aide au débugage : Common Logging**

  La bibliothèque common Logging permet de gérer un système de log très pointue, qui permet d'aider au débogage.

3. ARCHITECTURE GÉNÉRALE APPLICATIF EMAGINE

Afin de trouver des solutions techniques, une description générale de l'architecture eMagine est nécessaire.

3.1. DIAGRAMME

eMagine sera une application du type client/serveur avec une base de données relationnelle afin de permettre le stockages des données. Chaque utilisateur à l'aide d'un navigateur pourra accéder au serveur pour faire ses modifications. Le diagramme suivant montre le principe de base :

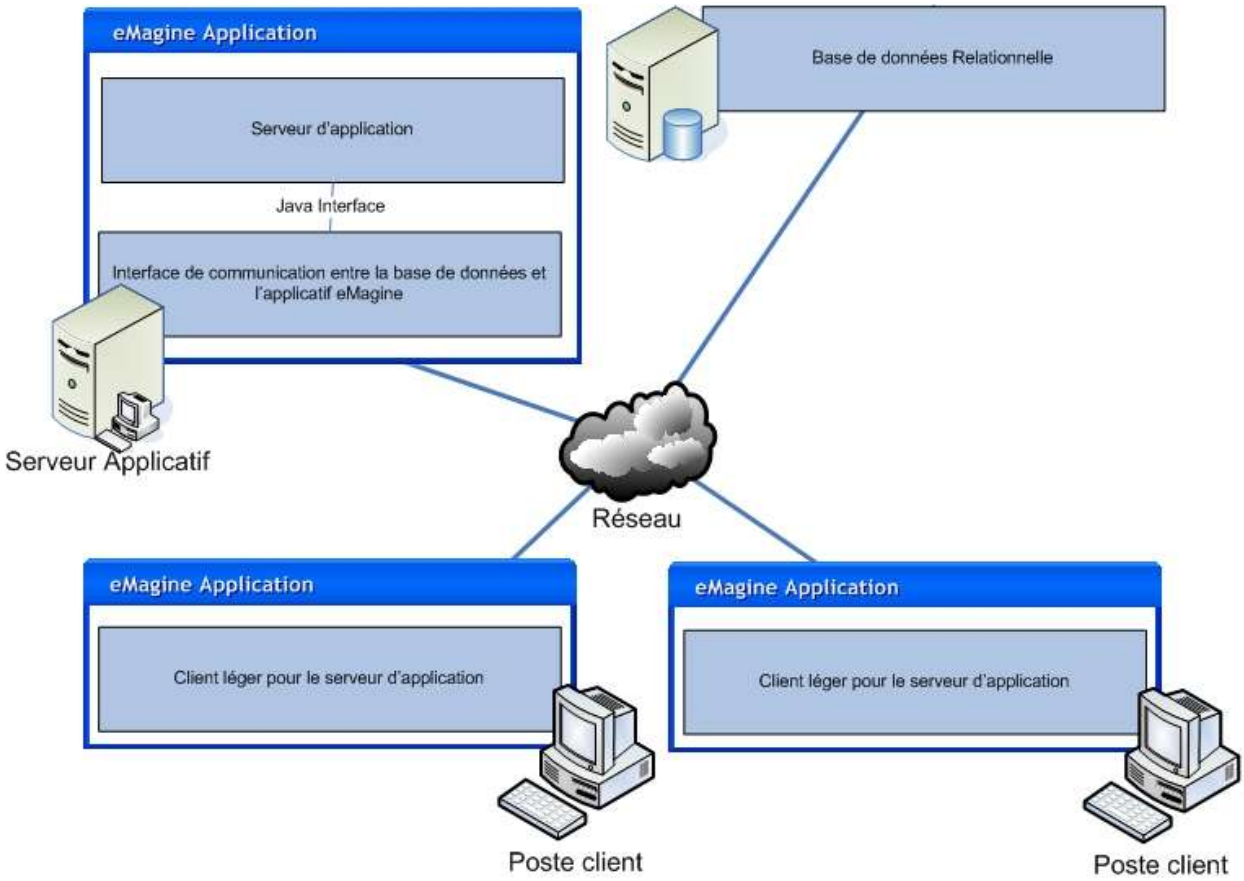


Illustration 1Architecture générale eMagine

3.2. DÉTAILS DE L'ARCHITECTURE

Ce chapitre étudie les différentes technologies utilisables pour répondre à cette architecture.

a. Base de données

La base de données contient les informations du logiciel, il existe sur le marché quatre grand SGBD[1], à savoir :

- **SGBD**

Microsoft SQL Server : Utilisation simple, mais propriétaire, et demande l'achat de licence

Oracle : Utilisation professionnelle, demande des connaissances pointues pour être maîtrisé

MySql : Utilisation orientée web, pas de déclencheur, par exemple

PostGres : Serveur de base de données relationnelle le plus abouti.

- **JDBC**

JDBC permet de dialoguer avec différentes bases de données, en utilisant la même interface, quelque soit la base de données.

b. Serveur d'application

Le serveur d'application permet de mettre en place une architecture client/serveur avec une application centralisée sur un serveur.

- **Il existe sur la marché deux principaux serveurs d'application**

Jboss : JBoss a de nombreuses particularités. La principale concerne son architecture. Elle est de type micro-kernel, tout est développé sous forme de module JMX.
Les différents modules couvrent la norme J2EE : JBossMQ gère les messages Java (Java Messaging System), JBoss MX la messagerie électronique, JBoss TX les transactions JTA/JTS, JBoss SX la sécurité (basée sur JAAS), JBoss CX la connectivité JCA et JBossCMP la persistance CMP.
Chacun de ces modules peut être remplacé par un module au format JMX.
Trois types de configuration sont installés en standard : minumum, default et all.

Tomcat : Le serveur Tomcat est un serveur Open Source qui agit comme un conteneur de servlet J2EE. Il fait partie du projet Jakarta, au sein de la fondation Apache. Tomcat implémente les spécifications des servlets et des JSP de Sun Microsystems. Comme Tomcat inclut un serveur HTTP interne, il est aussi considéré comme un serveur HTTP.

1 Système de Gestion de Base de Données

J2EE : Java 2 Platform, Enterprise Edition est un framework pour le langage de programmation Java de Sun plus particulièrement destiné aux applications d'entreprise. Dans ce but, il contient un ensemble d'extension au framework standard afin de faciliter la création d'applications réparties.

c. Client léger

Les clients légers sont des interfaces web accessible via n'importe quel navigateur. Différentes contraintes techniques apparaissent lors de la mise en place.

- **Bibilothèque AJAX**

Le client léger, doit avoir un temps de réponse convenable, c'est pour cette raison que la technologie AJAX va être utilisé, pour éviter d'avoir a tout refaire de nombreux framework permettent d'utiliser des fonctions toutes faites :

- Dojo
- DWR
- JSON-RPC-JAVA
- MochiKit
- Prototype
- Rico
- SAJAX
- Scriptaculous
- Xajax
- Sack

- **JavaMail**

JavaMail est une bibliothèque de gestion des mails, elle supporte le protocole POP et STMP.

- **Java POI**

POI est un projet de Jakarta d'Apache Software Foundation permettant de manipuler de divers types de fichiers créés par Microsoft avec langage Java. Ainsi avec POI, nous avons la possibilité de travailler avec des fichier Excel et word.

- **Corba**

CORBA, acronyme de Common Object Request Broker Architecture, est une architecture logicielle, pour le développement de composants et d'Object Request Broker ou ORB. Ces composants, qui sont assemblés afin de construire des applications complètes, peuvent être écrits dans des langages de programmation distincts, être exécutés dans des processus séparés, voire être déployés sur des machines distinctes.

- **RMI**

Remote **M**ethod **I**nvocation, plus connu sous l'acronyme RMI est une interface de programmation (API) pour le langage Java qui permet d'appeler des objets distants. L'utilisation de cette API nécessite l'emploi d'un registre RMI sur la machine distante. Cette machine héberge les objets que l'on désire appeler et utiliser. Cette API est utilisée très souvent en parallèle avec l'API d'annuaire JNDI ou encore avec la spécification de composants distribués transactionnels EJB du langage Java.

- **Applet**

Une applet est un logiciel qui s'exécute dans la fenêtre d'un navigateur Web.

Cette approche offre moyen de fournir à l'utilisateur, sans installation d'un logiciel ad hoc (souvent appelé client lourd), une application ergonomique et réactive car pour bonne part animée par son navigateur Web plutôt que par le serveur distant.

Un navigateur Web nu n'offre qu'une interface de présentation d'informations, il est incapable de les traiter. Lui assigner certaines tâches réduit les communications réseau ainsi que le volume de travaux (charge) imposé au serveur, donc mobilise les ressources (CPU, mémoire informatique...) du poste client afin d'améliorer la fluidité en réduisant les latences. De surcroît, une applet offre au développeur un moyen d'employer certaines ressources du poste client qui demeurent hors de portée du HTML et améliorent l'ergonomie de l'application, par exemple grâce des éléments d'interface graphique. Tout navigateur contemporain abrite pour cela un environnement d'exécution d'applets.

- **Struts**

Apache Struts est un cadre d'applications open-source pour développer des applications web J2EE. Il utilise et étend l'API Servlet Java afin d'encourager les développeurs à adopter l'architecture MVC (Modèle Vue Contrôleur). Apache Struts a été créé par Craig McClanahan et donné à la fondation Apache en mai 2000.

Un plug-in eclipse est disponible pour l'utilisation de struts.

- **JfreeChart**

JFreeChart est une librairie graphique qui permet de créer des graphiques de formes très variées à partir de vos données.

On notera parmi les fonctionnalités la possibilité d'insérer un deuxième axe ou les divers formats d'export : PNG, JPEG mais aussi PDF ou SVG.

d. Interface Serveur d'application/Base de données

L'interface devra permettre d'éviter les accès intempestifs à la base de données. Il existe actuellement une bibliothèque réputée pour cela : Hibernate. L'autre solution serait de créer l'interface, mais les délais de développement deviendraient beaucoup trop élevés.

- **Hibernate**

Hibernate utilise une base de données et des données de configuration pour fournir un service de persistance (et des objets persistants) à l'application. L'architecture la plus complète abstrait l'application des APIs JDBC/JTA sous-jacentes et laisse Hibernate s'occuper des détails. Cette bibliothèque permet donc de manipuler les éléments d'une base de données sans les contraintes de la base de données.

CONCLUSION

Le pré-cahier des charges a permis d'éclaircir les technologies existantes. Cela permettra lors de la création du cahier des charges techniques de connaître les technologies.