BILLET Benjamin
CSII 2ère année

TRAVAUX LIKEONSITE

TRAVAUX DE RECHERCHES ET DEVELOPPEMENTS APPLIQUES AUX METIERS DE L'AUDIOVISUEL (C/C++, Java, C#, PHP)

Rapport de stage

Stage effectué de juin à septembre 2010

Setinnov	*Travaux de recherche et prototypage*	**Version 0.27**
Tuteur : Cyril FASSIER		**Dernière édition : 24/06/13**
E.P.S.I Montpellier 437 rue des apothicaires, 34090 MONTPELLIER		

REMERCIEMENTS

SOMMAIRE

Introduction 1
L'entreprise 3
1. Setinnov 3
2. OnTech 7
La mission LikeOnSite et ses declinaisons 10
1. L'outil LikeOnSite 10
2. Analyse de l'existant 11
3. Problèmes observés 12
4. Enjeux des divers travaux 13
La realisation des missions LikeOnSite 15
1. Les choix organisationnels 15
2. Refonte LikeOnSite 18
3. Etude du serveur LikeOnSite 24
4. LOS Encoder 38
Resultats 46
1. Faire le point 46
2. Support 47
Discussion 49
Conclusion 51
Glossaire 53
Table des illustrations 54
Sources 56
Annexes 57

INTRODUCTION

Une mission en entreprise, sous la forme d'un stage, représente une chance d'acquérir non seulement de nouvelles compétences techniques et relationnelles mais aussi de découvrir des domaines indirectement liés à l'informatique. Ces domaines possèdent une culture, des problématiques et des contraintes qui leur sont propres et qui, finalement, divergent assez de l'informatique de gestion.
Aussi, de mon point de vue, il est intéressant de s'approprier ces cultures et d'étudier les interactions de celles-ci avec l'outil informatique (parfois utilisé à des stades très expérimentaux).

Cette année, dans un souci de variété, j'ai basé ma recherche et mes choix d'entreprises sur des critères différents, notamment la variété. En effet, il m'a paru intéressant d'effectuer un "retour aux sources" par rapport à ma formation initiale en informatique industrielle car, jusqu'à présent, mon cursus en entreprise s'est toujours trouvé soit lié à l'informatique de gestion soit à l'informatique web.

De plus, j'ai tenu à trouver une mission dans un contexte de recherche et de développement, de préférence en portant l'accent sur la recherche. Certes, la formation reçue au cours de mes études fut assez peu orientée vers les sciences informatiques pures mais, étant mitigé quant à mon orientation possible dans ce milieu, il m'a paru important de mieux connaître ce microcosme très particulier dans le but de me faire une idée réelle de ce que cela pouvait représenter.

De ce point de vue, l'audiovisuel est un choix intéressant car il s'agit d'un domaine récent connaissant une forte expansion et des besoins techniques colossaux. Le passage progressif au numérique et l'évolution massive vers des technologies en réseau en font un terreau fertile pour la recherche et l'innovation.
Qui plus est, il s'agit de technologies gourmandes, déployées sur des périphériques hétérogènes, avec un lien fort au matériel. De facto, le lien avec l'informatique industrielle, les réseaux et l'informatique embarquée se révèle extrêmement fort.

Setinnov est une entreprise innovante (accompagnée par Languedoc-Roussillon Incubation) en cours de création et ses travaux principaux portent sur de la recherche informatique appliquée aux métiers de l'audiovisuel.
Les projets de Setinnov sont multiples mais celle-ci est actuellement focalisée sur la création de son premier produit d'envergure : le LikeOnSite. Il s'agit, en bref, d'un outil de validation collaborative pour les métiers de la vidéo tels que le montage, la réalisation ou la traduction.

Au travers de cet outil, il est possible de travailler (découpage, validation, correction, traduction) sur des vidéos sans devoir les télécharger (streaming[1]), ce qui permet de rationaliser et de simplifier la transmission des données vidéos ainsi que leur traitement (un seul point centralisé).

Actuellement, le produit en est à sa phase de test et des problèmes apparaissent à cause d'une gestion trop floue du projet global (conception, analyse et architecture). De plus, le projet est considérablement délicat à faire évoluer ce qui laisse entrevoir des difficultés futures car Setinnov compte bien adapter son produit à des domaines autres (cours en ligne, imagerie médicale, etc.) que la validation cinématographique.
Autour de cela, le choix de Setinnov est d'amener progressivement une refonte de l'application lorsque celle-ci sera mise sur le marché et rapportera les fonds nécessaires.

Enfin, autour de ce projet principal viennent se greffer d'autres besoins comme, par exemple, la réalisation d'un encodeur logiciel destiné à produire des fichiers directement compatibles avec l'outil LikeOnSite ou encore le dimensionnement des futurs serveurs destinés à héberger l'architecture applicative. Aussi, à l'heure actuelle, les besoins de Setinnov sont fortement orientés étude (faisabilité, fonctionnalités, choix techniques, ressources nécessaires et gestion de projet) et analyse (architecture et modélisation).

Au travers ce document, je vais donc vous présenter les différents projets qui m'ont été confiés ainsi que les différentes étapes qui ont permis d'en mener l'étude et, le cas échéant, leur réalisation. Cependant, comme il y en a un certain nombre, je me focaliserais sur les plus importants.

Pour chacun des projets, je présenterais en premier lieu l'environnement, le contexte et l'existant dans lequel le projet est amené à évoluer. Si nécessaire, je compléterais cela par l'analyse des éléments existants et des solutions précédentes.
Ensuite, j'exposerais les besoins réels ainsi que les enjeux gravitant autour de ceux-ci. De plus, les méthodes d'étude seront 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.
Puis, pour les projets ayant été menés à terme, je mettrais en avant les phases de développement ainsi que les méthodologies de travail employées.
Enfin, une analyse permettra d'évaluer avec le recul nécessaire la pertinence des différents choix par rapport aux résultats obtenus (si ceux-ci ont été obtenus).

[1] Opération consistant à lire un flux audio ou vidéo à mesure qu'il est diffusé. Cela s'oppose à la diffusion par téléchargement de fichier qui nécessite de récupérer l'ensemble des données d'un morceau ou d'un extrait vidéo avant de pouvoir l'écouter ou le regarder.

Il est nécessaire, avant tout, de situer le contexte pratique de l'entreprise Setinnov.

En effet, j'ai indiqué plus haut que l'entreprise Setinnov était en cours de création. Aussi, les projets de l'entreprise sont hébergés par une société tierce appelée OnTech.

L'ensemble du stage s'est déroulé dans les locaux d'OnTech et, comme il s'agit à la fois d'un investisseur de premier plan ainsi que d'un partenaire, il me paraît important de mentionner les deux entreprises dans cette partie. Qui plus est, j'ai réalisé des projets et des études pour OnTech mais je n'en traiterais pas dans la partie technique de ce rapport.

1.Setinnov

1.1.Historique

Setinnov est une entreprise de recherche et développement informatique appliquée aux métiers de l'audiovisuel (montage, tournage, postproduction, etc.).

Celle-ci est en cours de création par Cyril FASSIER. L'idée ayant mené à la création de l'entreprise innovante a été mise à plat en 2008 et le développement de celle-ci se poursuit depuis le début de l'année 2009.

L'entreprise est suivie par le BIC de Montpellier Agglomération, un organisme qui accompagne les entrepreneurs ayant choisi de créer une entreprise porteuse d'un projet innovant, sur le territoire de l'Agglomération de Montpellier.

Il soutient spécifiquement des projets à fort potentiel de développement en privilégiant les secteurs de la santé, des biotechnologies, du tertiaire supérieur et des technologies de l'information / communication.

De plus, Setinnov a été incubée par Languedoc-Roussillon Incubation (LRI), une structure destinée à promouvoir l'innovation d'entreprise à partir des travaux de recherche issus des établissements d'enseignement supérieur ou des organismes de recherche publique.

Ainsi, par ce biais, Setinnov dispose de l'accès à des ressources scientifiques pointues (une collaboration avec la division ICAR[1] du LIRMM[2], par exemple) pour l'aider à réaliser un projet spécifique.

[1] L'équipe ICAR développe son activité selon trois axes scientifiques associant image et interaction pour la manipulation des données visuelles tels que les images, les vidéos et les objets 3D. Ces trois axes sont : Analyse et Traitement (AT), Codage et Protection (CP), et Modélisation et Visualisation (MV).

[2] Le Laboratoire d'Informatique, de Robotique et de Microélectronique de Montpellier (LIRMM) est une Unité Mixte de Recherche de l'Université Montpellier 2 (UM2) et du Centre National de la Recherche Scientifique (CNRS), département Sciences et technologies de l'information et de l'ingénierie (ST2I).

1.2.Informations techniques

1.2.1.Données juridiques et hiérarchiques

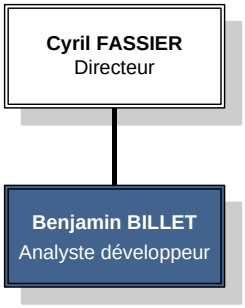


Fig. 1 – Organigramme hiérarchique de Setinnov

Raison sociale : Setinnov

Statut prévu : Société par actions simplifiées

Implantation : Montpellier

1.2.2.Clients et partenaires

Le produit phare de Setinnov intéresse actuellement un certain nombre de structures, principalement des studios de production et de montage audiovisuel d'envergures diverses.

Img. 1 – Quelques clients potentiels pour Setinnov

Les principaux partenaires (hormis la société OnTech) dans la création de l'entreprise Setinnov restent les organismes de subventions et d'accompagnement :

Img. 2 – Les organismes partenaires et/ou suivants Setinnov

1.2.3.Positionnement

Setinnov se positionne sur le secteur de la recherche et du développement informatique pour les métiers de l'audiovisuel.

Plus exactement, Setinnov pense avoir identifié, de par sa forte implication dans l'univers de la post-production, l'émergence d'un nouveau métier : le média manager.

Ce poste nouveau, pour le moment assuré par des personnes issues du tournage ou de la post-production fait intervenir des compétences particulières pour la manipulation, la conversion, le stockage et l'archivage des données vidéos entre le tournage et le montage.

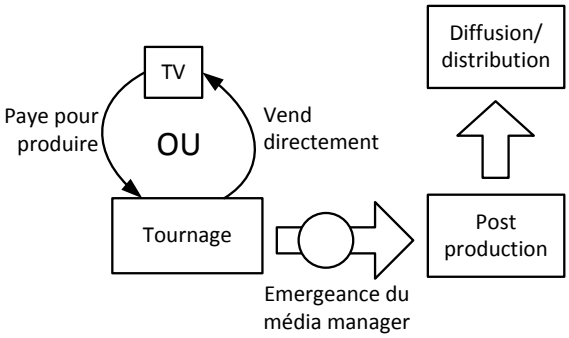


Fig. 2 – Positionnement du média manager

Aussi, Setinnov est destinée à se concentrer spécifiquement sur ce métier émergeant pour lequel tout reste à faire : normalisation des méthodes de travail, création d'outils adaptés (plutôt que la combinaison de multiples utilitaires), etc.

1.2.4.Forces

La principale force de Setinnov se situe sur sa grande expérience des métiers audiovisuels.

Cyril FASSIER, la personne à l'origine de l'entreprise, a exercé la profession de monteur pendant de nombreuses années. Cela lui a permis de bien assimiler la culture et la logique du domaine, pour aujourd'hui la restituer au sein des applications informatiques.

Cette expérience rassure beaucoup les investisseurs et clients potentiels car ceux-ci savent qu'ils n'ont pas affaire à une entreprise d'informatique pure, mais plutôt à une entreprise consciente de leurs pratiques et centrée sur leur métier.

Qui plus est, Setinnov est suivie par un grand nombre d'organismes d'aides à la création, qui lui permettent d'accéder à des ressources spécifiques.

- Le LRI (Languedoc Roussillon Incubation) permet à Setinnov de démarrer un projet en collaboration avec un organisme de recherche (ici, le LIRMM) et de disposer de chercheurs gratuitement.
- Transferts LR permet à Setinnov d'accéder à l'AFC (Aide à la faisabilité commerciale), une subvention destinée à aider l'entreprise à faire face aux dépenses externes liées à la faisabilité commerciale de son projet.
- Le BIC permet à Setinnov de disposer de locaux gratuitement, etc.
- L'organisme Synersud a permis à Setinnov de financer un stagiaire pendant 3 mois.

Enfin, le projet Setinnov intéresse un certain nombre de personnes. De clients d'une certaine envergure sont intéressés (CAPA TV, par exemple) et, de par la grande connaissance de ce milieu, la promotion du produit est facilitée (par exemple, Sonovision, une revue professionnelle qui fait référence dans le métier, a offert deux publications gratuites pour le LikeOnSite).

1.2.5.La culture d'entreprise et le management

L'entreprise Setinnov, tout comme l'entreprise OnTech, est issue du métier de l'audiovisuel.
Ce métier, bien antérieur à l'informatique, est très vaste et s'étend autour d'un processus standardisé dans lequel chaque phase correspond à des métiers spécifiques, aux méthodologies particulières.

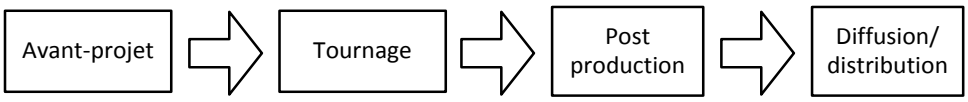


Fig. 3 – Processus de production audiovisuelle

Ces méthodologies, éprouvées et enseignées depuis longtemps, sont perturbées par l'évolution rapide de la technique et l'arrivée en force des logiciels et des réseaux.
Bien entendu, la transition est difficile. Les plus anciens, habitués à travailler d'une certaine façon et formé sur des techniques aujourd'hui menacées d'obsolescence, éprouvent naturellement des difficultés à évoluer dans leur travail. Qui plus est, l'informatique entraîne la disparition de certains postes devenus inutiles (même si effectivement de nouvelles compétences deviennent nécessaires).
De fait, le milieu de l'audiovisuel reste très fermé à l'informatique, à laquelle il reproche le non-respect des méthodologies de travail qui ont fait leurs preuves ainsi qu'une forte complexification des manipulations de média (guerres des formats de fichier, etc.).

Culturellement, il s'agit d'un milieu à cheval entre la technique pure (beaucoup de considérations électroniques, physiques et mécaniques) et le domaine artistique. Selon la phase et les personnes, l'un des deux côtés est plus présent et influe notamment sur la façon de travailler.
Setinnov, à cheval entre la culture audiovisuelle et la culture informatique, possède cette position privilégiée qui lui ouvre des portes généralement fermées aux informaticiens purs.
Il faudra probablement une génération entière avant que les différents intervenants soient pleinement formés, de bout en bout du processus de production audiovisuelle, aux concepts informatiques fondamentaux.

De facto, les méthodes de management de Setinnov (et il en va de même pour OnTech) sont fortement ancrées dans ce monde audiovisuel. Dans cet univers particulier, les équipes de travail sont généralement petites (de l'ordre maximum d'une dizaine de personnes) et il est n'est pas rare de voir de petites entreprises comme OnTech (trois employés) traiter avec de grands groupes comme France Télévision.
La communication au sein des équipes et des entreprises est libre, franche et ouverte. Celle-ci n'est absolument pas formaliste et les notions de hiérarchies restent très floues car le rôle de chacun est

d'une importance égale (l'équipe est vue comme un tout, et il est plus important de chercher des solutions aux problèmes plutôt que de déterminer qui est responsable dudit problème).
Le style de management est délégatif et le travail se base sur une confiance réciproque. Cependant, la rigueur reste de mise car les résultats influent généralement sur des sommes considérables.
De plus, comme il s'agit d'un milieu assez fermé, il n'est pas rare que tout le monde connaisse tout le monde. De fait, le bouche à oreille à un impact important ce qui maximise l'importance de la réputation. En effet, celle-ci peut rapidement se faire et se défaire en cas de problème.

2.OnTech

2.1.Historique

OnTech est une jeune société de service et de vente de marchandise spécialisée dans le domaine de l'audiovisuel et du multimédia en réseau.
Les services proposés par la société sont divers, allant de l'archivage multimédia jusqu'à la diffusion en streaming, en passant par la visioconférence et l'affichage dynamique[1].

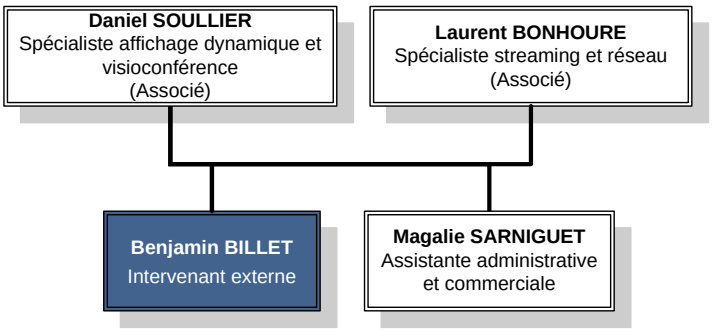
OnTech a été fondée en octobre 2007 par trois personnes, Laurent BONHOURE, Daniel SOULLIER et Renaud SARABIA. Il s'agissait d'un projet commun entre Laurent BONHOURE et Daniel SOULLIER lorsque ceux-ci travaillaient dans la société LTS, disparue depuis.
Ceux-ci considéraient que l'évolution de l'audiovisuel allait tendre de plus en plus vers l'informatique (notamment les réseaux) et c'est sur cette double compétence qu'ils décidèrent d'axer l'entreprise OnTech.

2007 : Création de l'entreprise OnTech.
2008 : Renaud SARABIA quitte l'entreprise. Dans la même année, Magalie SARNIGUET la rejoint.
2009 : OnTech devient redistributeur, pour la France et le Maroc, des produits ECRIN systems.
2010 : Déménagement de la société pour des locaux à la surface plus adaptée.

[1] L'affichage dynamique consiste à diffuser des médias (images, présentations, textes, pages webs, vidéos) en boucle sur des écrans disséminés. Par exemple, les panneaux d'informations numériques dans les hôtels.

2.2.Informations techniques

2.2.1.Données juridiques et hiérarchiques



Raison sociale : OnTech services

Statut : SARL au capital de 10.000€

SIRET : 500 306 725 00010

APE : 6202A

Implantation : Mauguio

Fig. 4 - Organigramme hiérarchique de la société OnTech

2.2.2.Clients et partenaires

Les clients d'OnTech sont multiples, on y retrouve des structures administratives, des sociétés de diffusion, des sociétés de services (SSII), des sociétés de grande distribution, des hôtels, des écoles et de nombreuses autres entreprises variées, de toutes les tailles et de tous les domaines. En effet, de par sa diversification, OnTech intervient à tous les niveaux et possède donc un panel de clients très diversifié.

Fig. 5 – Quelques clients d'OnTech

2.2.3.Positionnement

OnTech est spécialisée dans les domaines de l'audiovisuel et du réseau. Plutôt polyvalente, elle intervient à de multiples niveaux et sur une large gamme d'activités telles que l'installation (réseau, machine, etc.) et l'assemblage, la formation et le conseil, la maintenance et le dépannage, l'étude de solutions ainsi que diverses prestations sur site.

En bref, dès qu'il s'agit de réseau et/ou de vidéo, OnTech est en mesure de fournir à la fois du matériel et du service.

Comme compétence spécifique d'OnTech on peut citer, par exemple, la visioconférence, l'affichage dynamique, la vidéo à la demande[1] (VOD), le streaming, la web TV (TV sur IP), l'affichage distribué[2], la vidéosurveillance, l'archivage et la sauvegarde.

2.2.4.Forces

Tout d'abord, il faut savoir l'audiovisuel est un secteur très spécifique, nécessitant des compétences pointues non seulement en réseau (du à l'informatisation massive des outils) mais aussi en vidéo classique.

Aussi, il s'agit d'un milieu relativement peu concurrencé et ne souffrant que très peu des effets de la crise. Bouleversé par l'informatique, il est encore très consommateur de matériels et, surtout, de personnes possédant la double compétence informatique/audiovisuel.

En termes de gros partenariat, OnTech est tout d'abord redistributeur exclusif des produits ECRIN systems pour la France et le Maroc.

ECRIN systems est une société fondée en 1976 et fabriquant des machines d'encodage et de décodage de flux vidéo pour des opérations de streaming. Ces machines sont réputées pour obtenir une qualité vidéo incomparable.

Le chiffre d'affaire d'ECRIN systems s'élevait à 12.3 millions d'euros en 2007 et celle-ci employait, cette même année, 65 personnes.

La relation avec une entreprise de cette envergure permet à OnTech de pénétrer des marchés plus grands, comme Eurosports, au travers d'un produit qui leur est exclusif.

De nombreuses compétences d'OnTech sont uniques en France, ce qui entraine une forte mobilité à travers le pays (conseil général de l'Aisne, France 3 à Paris, Euromed à Marseille, etc.).

Par exemple, OnTech est certifié sur les produits Samsung, une certification rare en France (Samsung ayant tendance à privilégier la qualité de ses clients plutôt que la quantité) qui amène de grands groupes (comme la chaîne d'hôtels Mercure) vers l'entreprise.

Qui plus est, l'entreprise est DELL Partner, ce qui lui permet d'accéder à des formations, à un service client fortement réactif et à des tarifs préférentiels.

[1] La VOD (Video on demand) est une technique de diffusion, temporaire et contre rétribution, de contenus vidéo à travers des réseaux (Internet, 3G). Il s'agit de l'évolution de la location de films.

[2] L'affichage distribué consiste à afficher une image de très grande taille sur une mosaïque composée de plusieurs écrans plus petits.

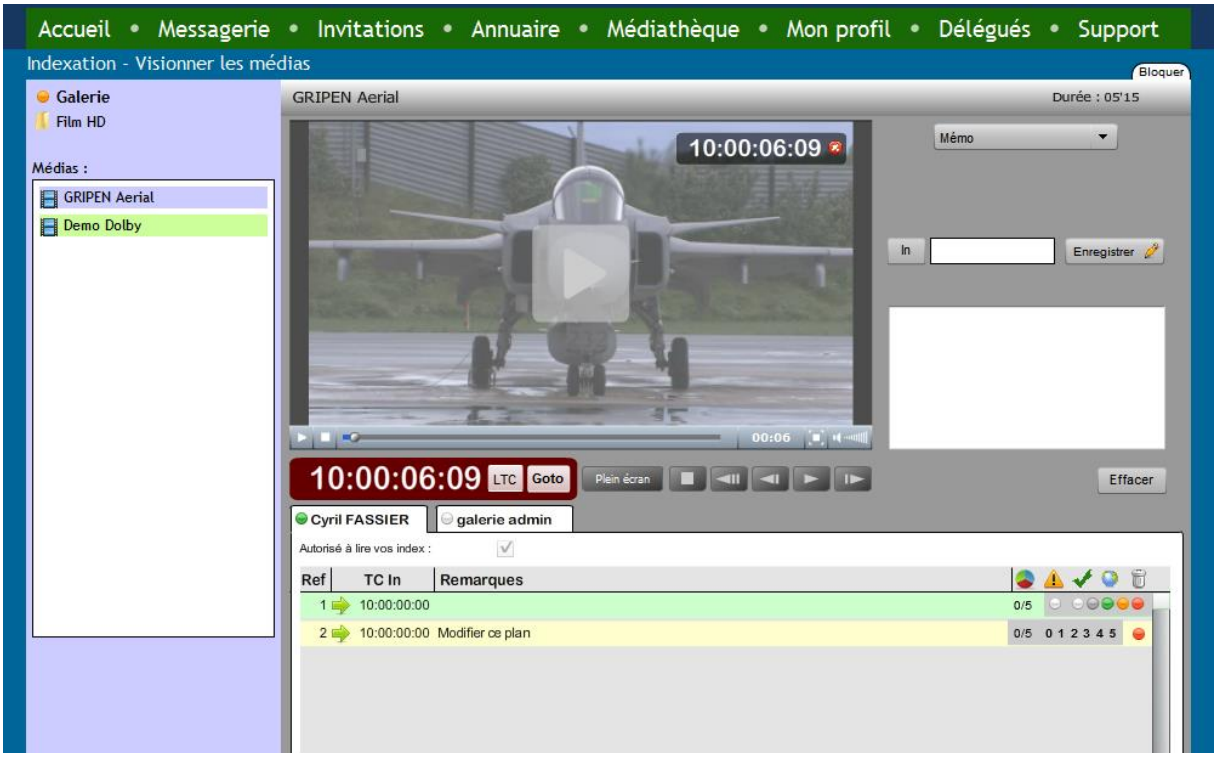
1.L'outil LikeOnSite

L'outil LikeOnSite (LOS) est une plateforme distante de travail collaboratif permettant de visionner et de traiter des fichiers vidéos en streaming dans le but d'effectuer des opérations de validation.

Concrètement, le LikeOnSite affranchit une équipe de tournage / montage de certaines contraintes telles que la duplication, le transcodage et la diffusion des médias entre les différents intervenants de la chaîne de production.

En effet, des problématiques multiples liées aux codecs[1], à la qualité des médias et à la sécurité posent de nombreux problèmes (on citera notamment le cas de ces copies "de travail", ou pilote, qui se perdent et finissent par servir à la création de copies pirates diffusées avant la sortie même du film).

Ainsi, la résolution de ces différentes problématiques unifie le processus de validation, du tournage jusqu'au montage.



Img. 3 - L'outil LikeOnSite (lecteur vidéo)

Ma mission aura été de définir une nouvelle architecture pour l'application, lui permettant d'intégrer des fonctionnalités supplémentaires et impossibles à mettre en œuvre en l'état actuel des applications.

En plus de cela, l'étude s'est aussi étendue à la partie matérielle du LikeOnSite. En effet, celui est actuellement hébergé sur un serveur mutualisé à faible coût. Il était alors intéressant d'étudier les op-

[1] Un codec (CODeur / DECodeur) est un composant permettant d'encoder une vidéo dans un format numérique donné.

tions disponibles pour passer dans un hébergement plus costaud (contenu de serveur, technologies, etc.).

Enfin, autour du LikeOnSite, des outils externes et des prototypes expérimentaux sont à réaliser pour simplifier l'utilisation ou prévoir la faisabilité et les choix techniques sous-jacents à des extensions futures de l'application.

2.Analyse de l'existant

La première phase de la mission consistait effectivement à dresser un bilan de ce qui existait déjà au niveau du LikeOnSite.

2.1.1.L'application LikeOnSite

Le LikeOnSite est écrit dans un certain nombre de langages web. On retrouve, en plus des classiques HTML et PHP, quantité d'éléments destinés à produire du contenu dynamique côté client. Il s'agit notamment d'AJAX, de JavaScript et d'ActionScript (Flash).

L'application exploite, qui plus est, une base de données MySQL.

Pour rentrer dans les détails de fonctionnement du LikeOnSite, le scénario ci-dessous exprime de manière relativement exhaustive l'utilisation classique de l'application :

> Un individu achète un compte LikeOnSite et en devient automatiquement superadministrateur. Il peut y héberger des fichiers vidéos (phase d'envoi des fichiers) pour les visionner ensuite en streaming. Ceux-ci sont organisés dans des dossiers de médias, eux-mêmes stockés dans des productions (représentant concrètement des projets).
>
> Ces productions peuvent être ouvertes à d'autres personnes appelées "délégués". Ces délégués, selon les droits affectés, vont pouvoir visionner, modifier, réorganiser ou rediffuser les données vidéo.
>
> Qui plus est, il est possible d'envoyer une invitation à une personne pour que celle-ci vienne consulter (et uniquement consulter) un ensemble de médias.
>
> Sur un média, les personnes autorisées peuvent placer des mémos (commentaire, éléments à corriger, etc.), des informations de dérush[1], des traductions et des entrées de chapitrage (sectionnement du média en plusieurs chapitres). Ces différents éléments, appelés index, serviront ensuite soit au montage (pour le dérushage), soit à exporter les traductions et les chapitres (dans des formats génériques) soit à effectuer des corrections (d'après les mémos).

2.1.2.Hébergement

Le LikeOnSite est actuellement placé sur un hébergement mutualisé[2]. Il s'agit de l'hébergeur canadien MavenHost, réputé comme étant assez fiable (99.99% de disponibilité).

Il s'agit d'un hébergement simple, offrant les fonctionnalités classiques (700 Mo d'hébergement, un domaine, plusieurs adresses email, trois bases de données MySQL, etc.).

[1] Le dérushage est la première étape du montage d'un programme audiovisuel, il consiste à sélectionner les séquences à utiliser lors du montage, appelées rushes, et à les transférer sur la plate-forme de montage.
Le dérushage est l'étape qui permet de sélectionner les éléments qui seront utilisés lors du montage.

[2] Un hébergement mutualisé revient à héberger les applications/sites webs de plusieurs personnes/entreprises différentes sur un même serveur physique.

2.1.3.Streaming

En ce qui concerne le streaming, celui-ci est sous-traité auprès d'une autre entreprise appelée MédiaTech. Celle-ci propose une solution SaaS[1] appelée StreamLike qui permet d'héberger des fichiers vidéos de manière décentralisée. Le principal avantage se situe dans le fait que StreamLike adapte la qualité des vidéos en fonction de la qualité de la connexion d'un internaute.

Côté StreamLike, du développement spécifique a été effectué pour répondre aux besoins du LikeOnSite. C'est pour cela que MediaTech est, en quelque sorte, un partenaire de l'entreprise Setinnov.

3.Problèmes observés

Les principaux problèmes sont issus de processus de conception et de développement erratiques.

Une conception trop floue, des impératifs de développement court ainsi qu'un budget limité ont imposé la mise en arrière-plan des pratiques de qualité logicielle.

Couplé à une certaine inassiduité dans le suivi de projet (travail combiné de quatre développeurs peu coordonnés et de niveaux variables), au surnombre de technologies employées et à des changements de caps réguliers (nouveaux besoins et nouvelles fonctionnalités) cela a significativement amplifiés les problèmes initiaux.

Aujourd'hui, le produit a pris beaucoup de retard, manque de stabilité (malgré plus d'un an de maturation) et se révèle extrêmement complexe à faire évoluer. L'absence de traces et de documentation rend d'autant plus longue tâche de résolution d'erreurs.

Des fonctionnalités importantes d'un point de vue marketing ne peuvent être intégrées, ou difficilement, du fait d'une architecture monolithique, vaste patchwork de technologies diverses et variées (la plupart réputées pour leur manque de rigueur et d'évolutivité) au travers desquelles sont mélangés des approches impératives[2] et orientées objets[3].

De nombreux problèmes apparaissent au niveau de la gestion des droits. Celle-ci est arrivée tardivement alors qu'il s'agit généralement d'un composant à fort couplage (utilisé par tout le reste de l'application) qu'il est préférable d'intégrer au plus bas niveau possible d'un développement en couche.

Parmi les fonctionnalités importantes rendues aujourd'hui impossibles par la structure même du LikeOnSite on trouve la "marque blanche". Il s'agit d'un principe commercial consistant à mettre à disposition un outil tiers (développé par une autre entreprise) sans en citer la marque ni l'origine.

[1] Le logiciel en tant que service ou en anglais le Software as a Service (SaaS) est un concept consistant à proposer un abonnement à un logiciel plutôt que l'achat d'une licence.

[2] Le paradigme de programmation impérative est le plus répandu. Il décrit les opérations en termes de séquences d'instructions exécutées par l'ordinateur pour modifier l'état du programme.

[3] Le paradigme de programmation orientée objet consiste en la définition et l'interaction de briques logicielles appelées objets.

Concrètement, il s'agit pour le LikeOnSite de permettre à un client de personnaliser l'interface de l'application aux couleurs de sa propre entreprise.
Cependant, le fort couplage entre code métier et interface utilisateur bloque totalement l'intégration d'une telle fonctionnalité, que ce soit en automatique ou au cas par cas.
Un autre problème se situe au niveau de l'hébergement. Des soucis de disponibilité, de fonctionnalité et de charge ont été rencontrés avec MavenHost. Il est devenu pressant d'envisager une solution à base de serveurs physiques propriétés de l'entreprise Setinnov.

Enfin, le LikeOnSite nécessite quelques outils pour favoriser son utilisation. Ces outils sont destinés à être distribués gratuitement, dans un premier temps, pour amener les utilisateurs à découvrir le produit.
Plusieurs outils ont été mentionnés mais un seul d'entre eux a été retenu (pour des raisons temporelles). Son rôle est de transcoder[1] des fichiers vidéos pour une utilisation optimale dans le LikeOnSite. En effet, le LikeOnSite est en mesure de recevoir un média sous trois formes (bas, moyen et haut débit) qui serviront ensuite à le diffuser en fonction du débit de la connexion internet d'une personne visionnant le média.

4.Enjeux des divers travaux

Les enjeux sont variés en fonction des différents problèmes exposés.
Globalement, la refonte permettrait de replacer le système de gestion des droits dans une couche basse et, ainsi, proposer simplement celle-ci aux différents composants de plus haut niveau de l'application. Qui plus est, la refonte permettrait de stabiliser le tout en repartant sur une démarche de gestion de projet saine. Tout d'abord, l'écriture et la fixation des documents contractuels (spécifications et cahier des charges) puis l'étude de l'architecture technique (dossier d'analyse) permettraient d'obtenir un résultat robuste, stable et relativement simple à faire évoluer.
Enfin, la nouvelle architecture pourrait prendre à nouveau en compte des fonctionnalités complexes comme la fameuse marque blanche ou, plus simplement, la traductibilité de l'application.

Le LikeOnSite, actuellement dédié à des applications purement audiovisuelles, pourrait être destiné à d'autres utilisations. En effet, le système de visionnage et de pose d'index pourrait être utilisé pour adapter le LikeOnSite en plateforme de cours en ligne. Les cours, diffusés sous forme de vidéos, pourraient être directement annotés par les différents élèves (les mémos deviendraient des ques-

[1] Le transcodage consiste à transformer un fichier vidéo numérique d'un format d'encodage A vers un format d'encodage B. Dans l'absolu, le format A et le format B peuvent être identiques, mais paramétrés différemment.

tions, par exemple) tout en conservant le principe des index de traductions (pour le sous-titrage des cours).

De même, le LikeOnSite aurait vocation à être adapté pour de l'archivage de fichiers vidéos ou encore de l'annotation pour les systèmes de recherche sémantiques.
Des applications sont prévues pour le domaine médical et quelques autres. Aussi, la nouvelle architecture doit tenir compte de ces modifications probables. Celle-ci doit être modulaire et fortement évolutive (couches multiples).

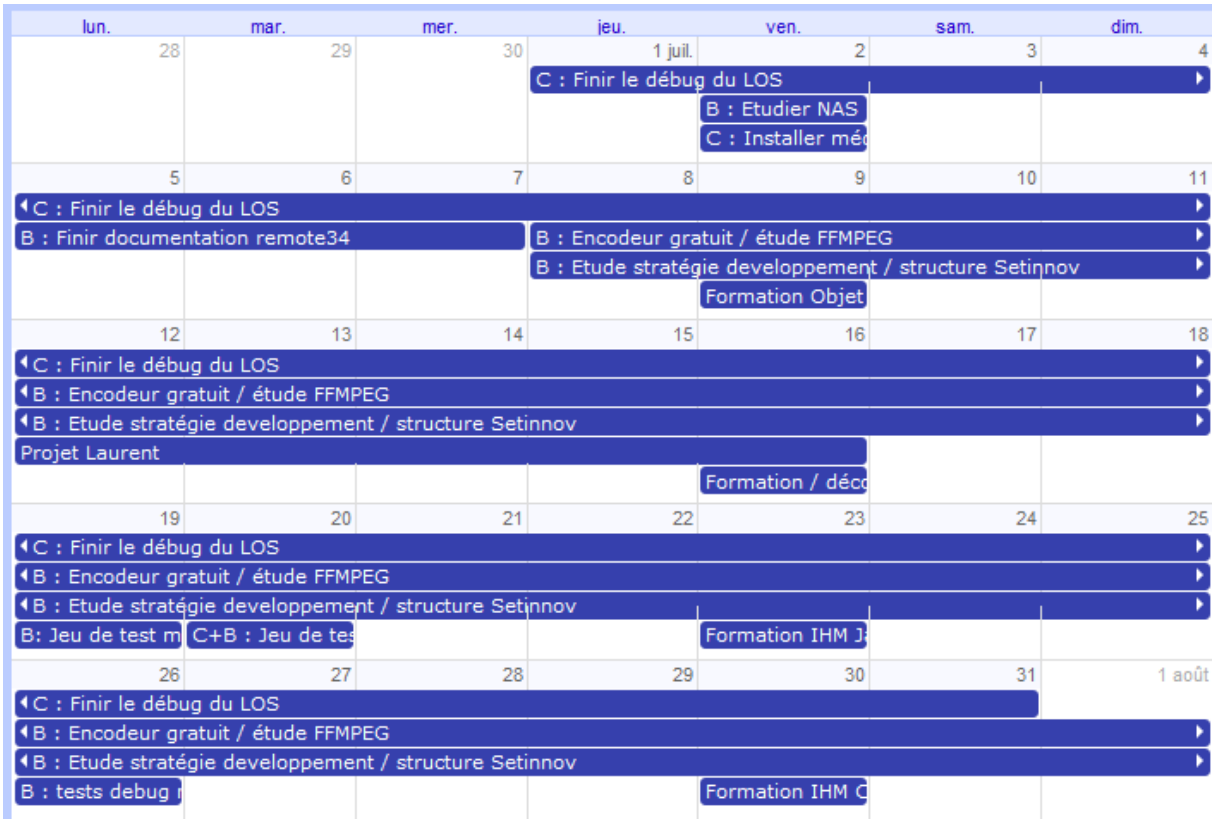
1.Les choix organisationnels

Au vu de la taille de l'entreprise, des choix organisationnels ont dû être fixés en cours de route pour éviter de renouveler certaines erreurs.

1.1.Planification des tâches

La planification des tâches s'est effectuée au fur et à mesure. Chaque début de mois, les tâches étaient planifiées pour le mois complet.

Ce découpage a permis d'obtenir une réactivité plus fine et d'intégrer les changements sans trop de difficulté.



Img. 4 - Exemple de planning (mois de juillet)

Chaque phase du planning était ensuite détaillée en sous-tâches journalières sous forme de tableaux prévisionnels. Ceux-ci, intégrés dans un logiciel de gestion de projet (MS Project) permettaient d'obtenir un diagramme de Gantt prévisionnel.

1.2.Réunion et suivi de projet

Chaque vendredi, une réunion permettait de vérifier si le planning était respecté et, si ce n'était pas le cas, de prendre les mesures nécessaires.

Qui plus est, cette réunion permettait d'aborder les problèmes rencontrés ou à venir, ainsi que les solutions plausibles à envisager.

Les nouvelles choses à intégrer au planning étaient passées en revue et priorisées. Des éléments pouvaient disparaître (car n'étant plus d'actualités) et d'autre apparaître (car issus d'une demande urgente).

1.3.Méthodologie des études

En premier lieu, un certain nombre d'éléments contractuels doivent être mis au point avant de commencer à aller en profondeur.

Une description sommaire du projet, en collaboration avec le tuteur, permet de poser quelques bases (analyse du vocabulaire) et d'extraire rapidement un dictionnaire de données[1] pour communiquer sans ambiguïté. Cette phase est d'autant plus importante car le vocabulaire et la culture associée à l'audiovisuel diffèrent de ceux associés à l'informatique.

De ces premiers éléments est issu un cahier des charges. Celui-ci se compose de la façon suivante :

- Contexte d'étude et d'application : décrit l'existant, le contexte économique, les enjeux, les domaines et contraintes d'application.
- Arbre des fonctions : représente l'ensemble des fonctions de l'application, sous forme hiérarchique, du plus général au moins général

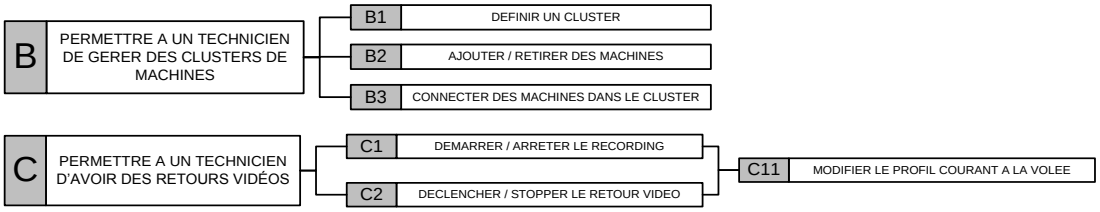


Fig. 6 – Exemple d'arbre des fonctions

- Description de chaque fonction : expose en détail les enjeux des fonctions, fournit une description détaillée de celles-ci et définit des critères de performance[2]. Chaque fonction peut alors logiquement être chiffrée en temps et en coût probable (il ne s'agit que d'une étude préliminaire).
- Hiérarchie d'importance : chaque fonction est confrontée à toutes les autres, pour déterminer son importance relative. Ces importances sont ensuite totalisées et placées sous forme de graphe.

Lorsque le tuteur (et le client, s'il existe) a validé le cahier des charges, un dossier d'analyse est rédigé. Il contient les éléments suivants et respecte la méthodologie UML :

[1] Document décrivant le vocabulaire du domaine étudié. Il permet à tous les intervenants de "synchroniser" leur vocabulaire pour une meilleure compréhension.

[2] Ces critères sont de toute nature, ils peuvent porter sur la rapidité, l'évolutivité, l'ergonomie, etc. Ils sont accompagnés d'une classe de flexibilité (F0 à F2) qui définit l'importance qu'on leur accorde (par exemple F2 signifie que l'on peut se permettre des libertés quant au respect ou non de cette contrainte).

- Dictionnaire de données et analyse du vocabulaire
- Maquette de l'interface graphique (réalisée avec le concours du tuteur / client)
- Choix techniques
- Etude du contexte (mise en situation des acteurs sous-systèmes les uns par rapport aux autres, ainsi que des acteurs humains et matériels). Accompagné par le diagramme de déploiement (architecture matérielle)
- Cas d'utilisation et descriptions textuelles (algorithmes rapides)
- Séquences systèmes (décrit les interactions entre les différents acteurs)
- Diagrammes d'activités (décrit les algorithmes de manière plus fine, en situant les éléments par acteurs)
- Diagramme d'état, si nécessaire
- Diagramme de classes et diagrammes de séquences détaillés
- Diverses informations (schéma de base de données, formats de fichiers, protocoles réseau, etc.)

1.4. Méthodologie pour les projets de développement

J'ai utilisé, pour mener à bien les projets complets, la méthodologie 2TUP, vue cette année en cours de gestion de projet.

Les processus UP s'articulent autour de six activités fondamentales : Capture des besoins, analyse, conception, implémentation, test et déploiement.

Il s'agit d'un processus de travail itératif[1] et incrémental[2] piloté par les risques, c'est-à-dire se focalisant sur l'élimination rapide des risques majeurs, et piloté par les tests (les tests définissent les fonctions et non plus l'inverse). De plus, il est très orienté utilisateur car ceux-ci participent au processus de développement.

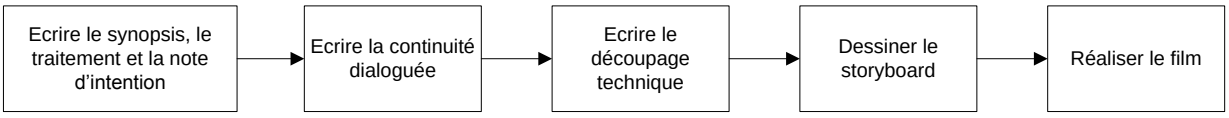


Fig. 7 – Exemple d'application de la méthode itérative

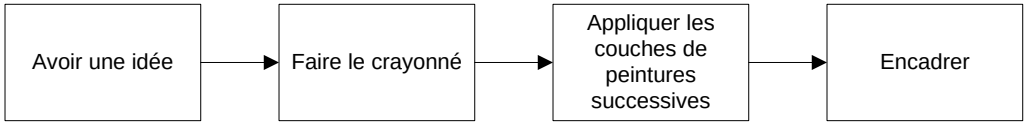


Fig. 8 – Exemple d'application de la méthode incrémentale

[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.

2TUP (Two Track UP) apporte quelques éléments supplémentaires à la méthode UP, notamment un cycle en Y particulier. Le début du projet se découpe en deux axes parallèles : l'axe fonctionnel et l'axe technique.

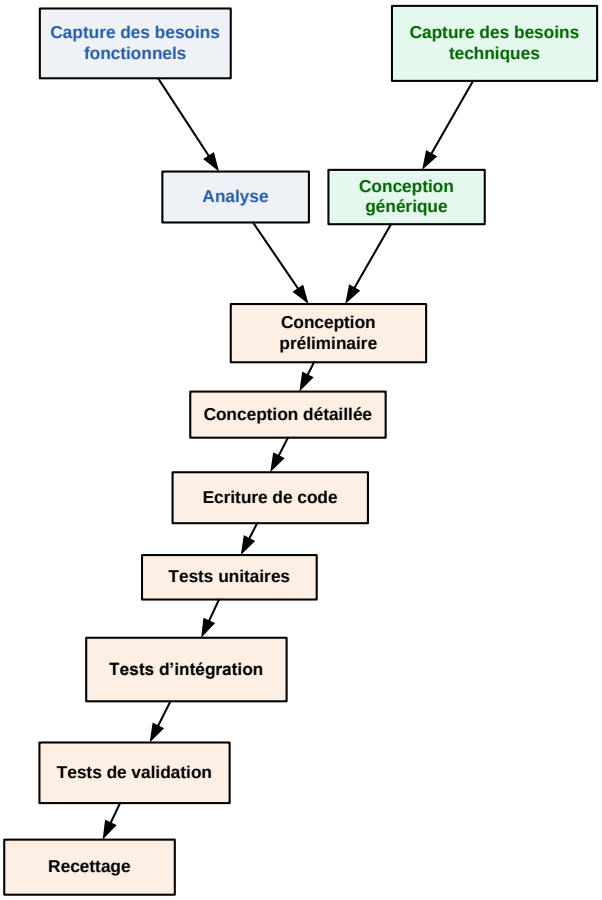


Fig. 9 – Cycle de développement en Y

L'axe fonctionnel (gauche) se focalise sur les métiers de l'utilisateur (dans mon cas, l'audiovisuel) pour identifier et éliminer les risques liés à la production d'un système inadapté. Dans cette branche, les besoins doivent être précis, cohérents et exhaustifs, indépendamment de toute contrainte technique.

L'axe technique (droit) traite de la réalité des besoins techniques. Les contraintes et les dimensionnements sont recensés et déterminent le choix de l'architecture technique.

Cette parallélisation de l'aspect métier et de l'aspect technique est fondamentale pour écarter à la fois les risques fonctionnels et les risques techniques.

Cependant, les deux branches doivent rester les plus indépendantes possible.

1.5.Résolution des problèmes

La méthode de résolution de problèmes était très simple ; lorsqu'une difficulté intervenait, une réunion avec le tuteur était organisée. Le problème était alors identifié ainsi que ses conséquences sur la planification des tâches.

Ensuite, si une solution était envisageable rapidement, celle-ci était privilégiée. Sinon, fonction des cas, la nature de la fonctionnalité était repensée ou revue à la baisse. Parfois, la fonctionnalité était mise de côté pour privilégier des éléments plus importants.

Ces décisions étaient prises après plusieurs discussions. Parfois, un brainstorming était démarré pour tenter de déterminer la meilleure solution.

2.Refonte LikeOnSite

Il s'agit d'une étude (et non d'un développement complet) destinée, dans un premier temps, à définir les besoins et les concepts liés au LikeOnSite (exposés précédemment).

Dans un second temps, cette étude porte sur les choix techniques (langages) et architecturaux pour se terminer par l'analyse complète de l'application.
Bien entendu, ces documents ont vocations à servir pour un développement futur. Ils sont là pour guider une équipe de développeurs, bien que cela ne dispense pas d'une gestion de projet suivie et efficace.

2.1.Choix techniques

2.1.1.Architecture réseau

Il faut bien comprendre que le LikeOnSite pourrait, en cas de succès, accueillir une grosse communauté. Aussi, l'architecture réseau elle-même doit permettre de distribuer et de répartir facilement l'application lorsque ce sera nécessaire.

Aussi, j'ai pensé qu'il pourrait être judicieux de séparer le noyau métier (logique applicative) de l'interface graphique, ce directement au niveau réseau, comme s'il s'agissait de deux composants autonomes (architecture client-serveur).
Ainsi, il devient possible d'écrire une autre interface graphique pouvant être hébergée sur les serveurs de l'entreprise cliente (le concept de marque blanche).

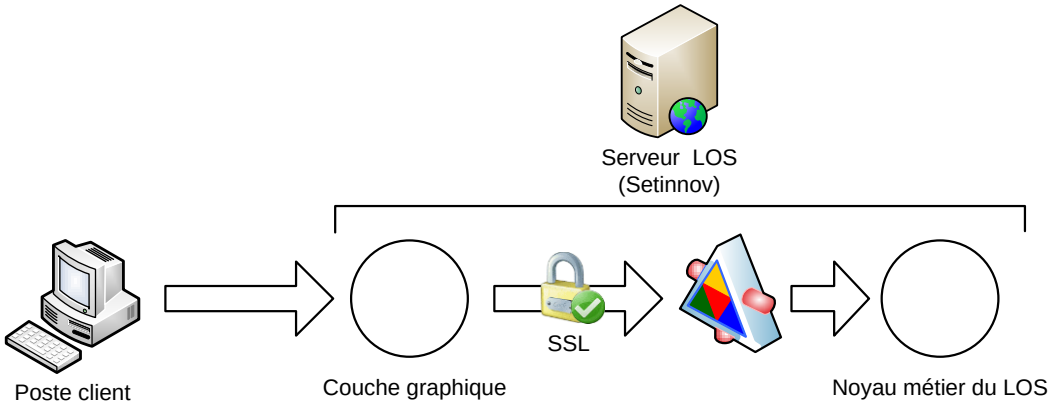


Fig. 10 – Connexion d'un client standard

Dans le schéma ci-dessus, le serveur de Setinnov héberge à la fois la couche graphique et le noyau métier. Cette couche graphique est standardisée pour tous les utilisateurs du LikeOnSite qui n'ont pas souscrit à une offre de personnalisation avancée de l'interface.

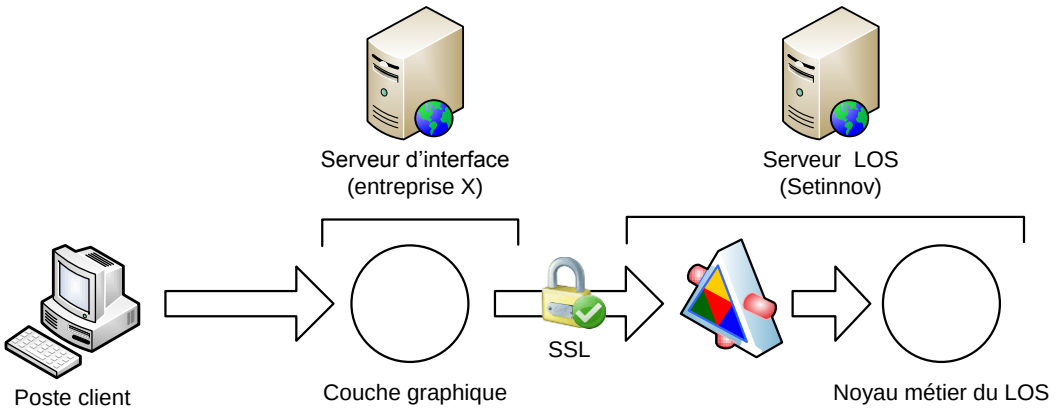


Fig. 11 – Connexion d'un client au travers de la marque blanche

Cette fois-ci, dans le schéma ci-dessus, le serveur de Setinnov héberge toujours le noyau métier. Cependant, c'est le serveur d'une autre entreprise qui héberge une couche graphique redésignée aux couleurs de l'entreprise.

Ainsi, quels que soient les utilisateurs finaux, le noyau dur reste sur les serveurs de Setinnov. De plus, toutes les modifications induites côté métier se répercutent sur les différents noyaux graphiques.

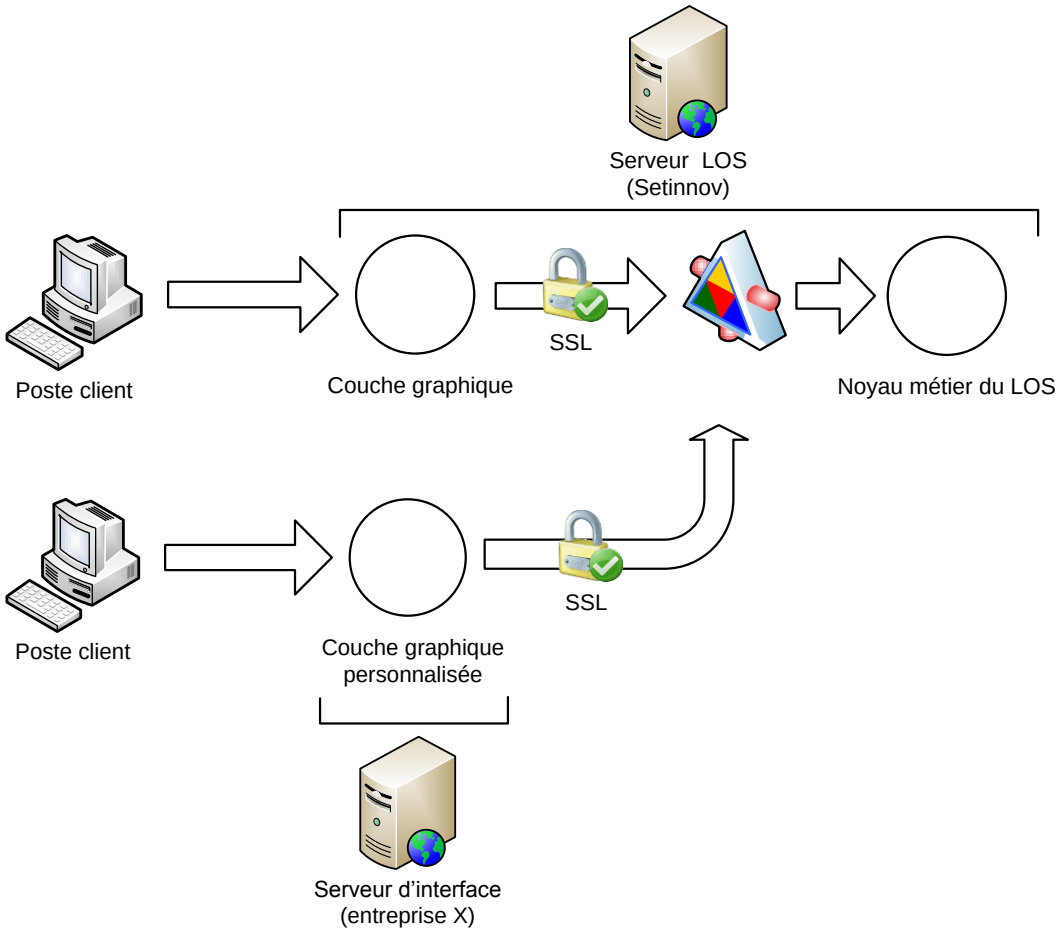


Fig. 12 – Récapitulatif de l'architecture globale

De plus, une telle architecture disposera d'un avantage non négligeable : il devient possible de développer une alternative lourde à la couche graphique.

2.1.2.Le noyau dur

Nous avons vu au point précédent qu'il était possible de séparer la couche métier, conservée côté Setinnov, et la couche graphique, placée n'importe où ailleurs.

C'est ce noyau dur qui est voué à être éclaté sur plusieurs serveurs si un jour la charge venait à fortement augmenter.

Il existe plusieurs moyens de mettre en réseau des morceaux d'applications :

- Application mise en réseau au moyen d'une socket[1]
- RMI (Remote Method Invocation)
- CORBA (Common Object Request Broker Architecture)
- Services webs XML-RPC (Remote Procedure Call), SOAP (Simple Object Access Protocol), REST

L'application mise en réseau possède l'avantage d'être très performante et très personnalisée, mais oblige le développeur à écrire toute une logique de communication. En l'état, cela représenterait une perte de temps. Aussi cette solution a été mise de côté très rapidement.

	XML-RPC	CORBA	RMI
Disponibilité	Utilise des technologies bien ancrées dans internet (protocole HTTP, langage XML, etc.)	Simples spécifications implémentées par des produits tiers.	Intégré à JEE.
Mise en oeuvre	Facile à implémenter, ne nécessite aucune couche supplémentaire.	Utilisation de nombreuses surcouches.	Simple à mettre en œuvre mais fortement couplé aux technologies Java.
Langages supportés	Plus de 40 langages supportés.	Un langage (IDL) permet de décrire les éléments et de les convertir dans de nombreux langages. Portabilité maximale.	Fortement orienté Java.
Ecriture	Seules les fonctions distantes sont à écrire, le reste est totalement transparent.	Le langage IDL permet de décrire les objets.	Ecriture des fonctions en Java. Sert de base aux EJB.
Données	Supporte très bien les types primitifs (entier, flottants, etc.). Des difficultés potentielles pour les objets.	Supporte très bien les types primitifs (entier, flottants, etc.). Supporte aussi des collections d'objets ainsi que le transfert d'un flux binaire (volume réduit).	Le Java possède nativement des outils de sérialisation, le transfert de n'importe quel objet est donc facilité.
Standards	Technologies standards pour RPC, XML et HTTP.	Standards définis par l'OMG.	Maintenu par Sun Microsystems (Oracle aujourd'hui).

Table 1 – Comparatif des technologies d'exécution de code distant

Globalement, l'utilisation des services webs est extrêmement simple. Le découpage de l'application sous forme de services webs multiples est aisé lui aussi.

[1] Dispositif permettant à deux applications de communiquer en réseau.

XML-RPC est assez vieux et est désormais beaucoup moins populaire depuis qu'un de ses équivalents, SOAP, depuis que celui-ci a été standardisé par le W3C (organisme de standardisation des technologies webs).

REST, quant à lui, n'est pas un protocole mais une architecture. Il s'agit du style architectural du web tout entier.

2.1.3.Langages de développement

Nous avons vu qu'il existait deux grands composants de l'architecture LikeOnSite : le noyau et l'interface graphique. Un autre avantage d'avoir séparé les deux se situe au niveau des langages. Il devient possible de développer le noyau et l'interface graphique dans deux langages différents.

Côté client, Flex a été choisi. Il s'agit d'une solution de développement relativement récente (2004) permettant de créer et de déployer des applications à fort potentiel dynamique grâce à la technologie Flash, sans avoir à passer par l'éditeur graphique jusqu'à lors nécessaire pour utiliser le langage.

Ce langage offre une forte interactivité (richesse de manipulation d'interfaces graphiques) et permet de réutiliser les composants déjà écrits en Action Script au niveau du LikeOnSite (notamment le lecteur de fichiers vidéo).

Côté serveur, tout est possible car n'importe langage peut-être utilisé.

Langage	**Java**	**C#**	**PHP**
OS	Tous ceux pour lesquels la machine virtuelle Java est disponible.	Windows uniquement.	Tous ceux capables d'accueillir un serveur web compatible.
Nécessite	Serveur d'application.	Serveur d'application.	Serveur web.
Gestion de la charge	Fine.	Fine.	Grossière.
Performances	Constantes.	Constantes (meilleures que Java).	Excellentes si faible charge. Va dépendre du serveur web.
Complexité d'administration	Moyenne	Moyenne	Faible
Mutualisable	Difficile	Difficile	Oui

Table 2 – Comparatif des langages

Le choix a été fixé au Java car l'utilisation d'un système d'exploitation libre était indispensable. Le PHP, lui, a été éliminé d'office car inadapté aux fortes charges.

2.1.4.Traductibilité

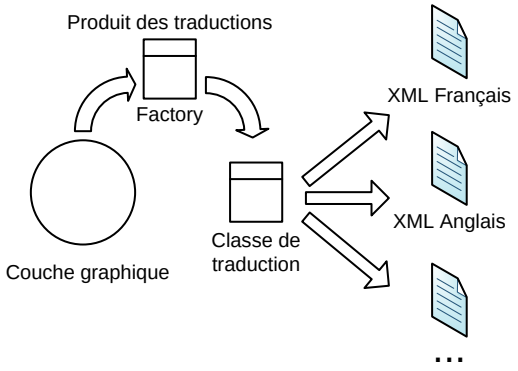


Fig. 13 – Schéma de mise en œuvre de la traduction

Les interfaces graphiques doivent être pourvues d'un système permettant de les gérer en plusieurs langues.

Très simplement, les couches graphiques externalisées embarqueront des fichiers XML de langues décrivant les différents textes de l'interface, traduits dans les langues disponibles.

Ces fichiers XML seront lus et mis à disposition par un composant "Factory" chargé de fournir les bonnes langues aux bons utilisateurs fonction, par exemple, de la langue du navigateur ou d'un choix de l'utilisateur.

2.1.5.Architecture en couches du noyau dur

Au niveau du noyau, la réception d'une information se fait au travers des étapes suivantes.

- Une requête HTTP est reçue.
- Un composant XMLCommunicator décode cette requête et construit un DTO (Data Transfer Object) correspondant.
- Le DTO est transmis à un contrôleur qui va effectuer les opérations demandées.
- Le contrôleur peut utiliser une DAL (Data Access Layer) obtenue au moyen d'une classe de fabrication.
- Si une réponse est nécessaire de la part du noyau, le contrôleur crée un nouveau DTO (si nécessaire) et le renvoie au XMLCommunicator.
- L'XMLCommunicator construit la réponse HTTP et l'envoie au client.

Qu'est-ce qu'une DAL ?

Une Data Access Layer (couche d'accès aux données) est une couche composée de plusieurs composants pouvant accéder à des données identiques, mais stockées sur différents supports.

C'est cette couche qui a connaissance de la manière d'accéder aux données et qui les distribue de manière transparente aux couches supérieures.

Qu'est-ce qu'un DTO ?

Un Data Transfer Object (objet de transfert de donnée) est un objet contenant des informations figées issues d'une ou plusieurs sources. Il réunit des ensembles de données provenant de diverses sources en divers points du programme. Grâce à lui, on peut transmettre facilement un bloc d'information, puisqu'il en est une copie unifiée.

De plus, il connaît la logique de transformation nécessaire au transfert de données. Si nécessaire, il peut être sérialisé[1], transformé en XML, etc.

Dans l'exemple ci-contre, le DTO associe, au sein d'un même objet, les informations concernant un morceau de musique ET celles concernant l'album auquel le morceau appartient.

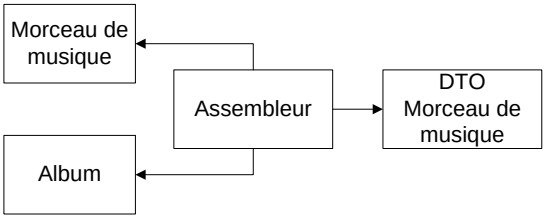


Fig. 14 – Fonctionnement d'un DTO

Représentation schématique de l'architecture

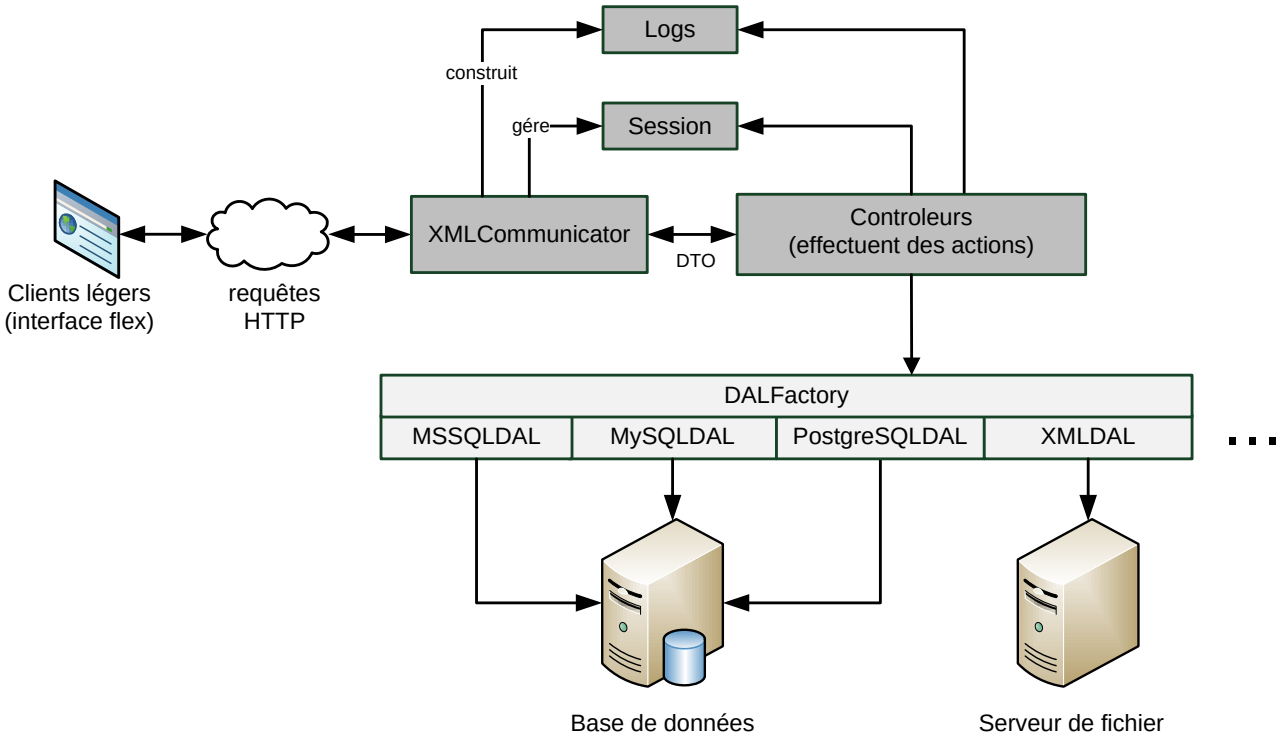


Fig. 15 – Architecture globale du LikeOnSite

3.Etude du serveur LikeOnSite

Il s'agit d'une étude (et non d'une installation) destinée, tout d'abord, à analyser les besoins en ressources de l'application LikeOnSite pour déterminer les solutions adaptées.

Ces documents sont conçus pour servir d'aide à l'estimation du coût nécessaire pour quitter un hébergement mutualisé.

3.1.Choix techniques

Pour cette étude, les choix techniques sont fondamentaux puisqu'il s'agit de déterminer quelles sont justement les solutions techniques (matérielles et logicielles) permettant d'obtenir de bonnes performances ainsi qu'une grande tolérance aux pannes.

[1] La sérialisation est un processus visant à encoder l'état d'une information qui est en mémoire sous la forme d'une suite d'informations plus petites (dites atomiques) le plus souvent des octets voire des bits.

Il faut rappeler aussi que ce serveur (ou architecture de serveur) sera amené, tôt ou tard, à héberger les fichiers vidéos (c'est-à-dire à se passer d'une solution tierce, comme StreamLike). Là aussi, de fortes contraintes de performances sont à prendre en compte.

Concrètement, plusieurs axes ont été abordés séparément :

- **Serveur** : architecture et sécurité "matérielle", performance.
- **Applicatif** : sécurité d'accès et d'utilisation, serveur web, résistance aux pannes.
- **Sauvegarde** : protection des données de la base, protection de l'applicatif.
- **Base de données** : architecture de la base de données, clusters et réplications.

3.1.1.Serveur physique

Maitriser le matériel de son serveur ainsi que le système d'exploitation apporte l'avantage du choix des technologies que l'on y embarque (Python, PHP, MySQL, PostgreSQL, Apache, nginx, GlassFish, etc.) et permet de contrôler la façon dont les différences opérations (maintenance, sauvegarde, évolution) sont effectuées.
Malgré tout, si le système est totalement maîtrisé, il nécessite néanmoins de disposer en interne des compétences techniques nécessaires (à moins de sous-traiter cette partie).

Il existe deux manières principales de procéder :

- Utiliser plusieurs petits serveurs et installer les applications en dur
- Utiliser un gros serveur d'entreprise et virtualiser

Les deux solutions vont permettre de distribuer les ressources à travers les serveurs. D'autant plus que l'architecture du LikeOnSite (à base de services web) permettrait de répartir aisément les morceaux de l'application.

3.1.1.1.Premier point : l'accès physique

L'aspect environnemental est le premier élément à prendre en compte. Il n'est pas pensable, dès lors que le nombre de serveurs s'accroît, de conserver physiquement les machines en interne de Setinnov car la logistique nécessaire serait délicate à mettre en œuvre (sécurité des accès physiques, salle dédiée, climatisation, bande passante, alimentation électrique, etc.).

L'accès physique doit être confié à un hébergeur (salle blanche) qui se chargerait de simplement stocker les machines appartenant à Setinnov, garantissant un accès sécurisé (mais permanent) ainsi que des dispositifs destinés à garantir la stabilité de l'alimentation électrique, la climatisation ainsi qu'une bande passante suffisante et disponible.

3.1.1.2.Second point : le matériel

Côté matériel, qu'il s'agisse de plusieurs petits serveurs ou d'un gros serveur d'entreprise, il est important de mettre en place des solutions matérielles de sécurité.

Quelques remarques d'ordre général par rapport au matériel :

- Les composants les plus importants sont : la mémoire, le processeur, les disques et, bien sûr, les cartes réseau.
- L'utilisation de grandes quantités de mémoire (supérieure à 4 Go) nécessite une architecture de processeur en 64 bits.

3.1.1.3.Troisième point : la résilience[1] des disques

La technologie RAID (Redundant Array of Independant Disks) permet d'apporter une résistance aux pannes et/ou des performances accrues directement au niveau des disques.
Un système RAID peut être intégré au matériel directement, ou se situer dans une couche logicielle (elle-même pouvant être intégrée dans un contrôleur de la carte mère, ou au niveau du système d'exploitation).

Le RAID matériel a pour avantage de ne pas solliciter de ressources systèmes, cependant les contrôleurs de RAID matériel peuvent être moins performants (plus lents) et sont généralement incompatibles entre eux (nécessite de prévoir une carte mère de rechange). De plus, le RAID matériel limite le type de disque pouvant être utilisé (soit SCSI, soit SATA, soit SAS, etc.).
A contrario, le RAID logiciel monopolise des ressources système (notamment le bus de communication entre les disques) mais les contrôleurs de RAID logiciel sont plutôt bien intégrés au niveau des systèmes d'exploitation. Ils permettent de traiter avec n'importe quel type de disque, sont souples à administrer et sont peu coûteux.
Par contre, d'un système d'exploitation à un autre, ils ne sont pas compatibles et, bien souvent, le RAID n'est pas applicable sur le disque qui héberge le système d'exploitation (puisque c'est le système d'exploitation qui s'occupe de mettre en place le RAID).

Remarque : Il existe un type de RAID pseudomatériel (assez répandue aujourd'hui au niveau des cartes mères bon marché) qui consiste à exporter les routines RAID dans des puces particulières.
Cette solution combine les défauts des deux approches, mais permet d'effectuer le RAID même sur le disque qui héberge le système d'exploitation.

[1] Capacité d'un système ou d'une architecture réseau à continuer de fonctionner en cas de panne.

a)RAID 0 : volume agrégé par bandes

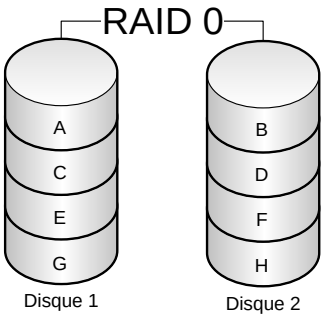


Fig. 16 – RAID 0

Le RAID 0 distribue les données sur les différents disques. Ainsi, les lectures et écritures sont distribuées à travers les disques, ce qui permet d'accroître significativement les performances, en plus d'augmenter la capacité de stockage. De plus, l'espace des deux disques est disponible.

Cependant, si l'un des disques tombe en panne, alors toutes les données sont perdues.

b)RAID 1 : Disques en miroir

Le RAID 1 duplique, à chaque instant, les données sur les différents disques.

Pour que les données soient perdues, il faudrait que tous les disques tombent en panne. Cela offre donc un haut niveau de fiabilité.

Cependant, la taille totale du volume est celle du plus petit disque. Les coûts de stockage sont élevés puisque l'ajout d'un nouveau disque ne change rien à la capacité totale.

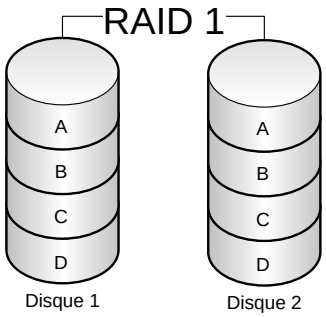


Fig. 17 – RAID 1

c)RAID 5 : volume agrégé par bandes à parité répartie

Le RAID 5 combine les propriétés du RAID 0 et du RAID 1. Les données sont réparties à travers plusieurs disques et des données de sauvegarde sont stockées sur un disque supplémentaire.

Si un disque de données vient à tomber, le RAID 5 peut le restaurer à partir du disque de sauvegarde.

Si le disque de sauvegarde tombe, aucune donnée n'est perdue. Il suffira juste de le remplacer pour que RAID 5 réécrive les données de sauvegarde.

Bien entendu, de par ce fonctionnement, le RAID 5 ne peut supporter la perte que d'un seul disque à la fois. Pour limiter le risque il est courant de dédier un disque dit de *"spare"*. En fonctionnement normal il n'est pas utilisé. Toutefois, en cas de panne d'un disque il prendra automatiquement la place du disque défaillant. Cela nécessite une phase de recalcul (soit des données, soit des sauvegardes) qui peut s'avérer assez longue (deux heures environ pour des disques de 300 Go contre une dizaine d'heures pour 1 To). Bien sûr pendant tout le temps du recalcul le disque est disponible normalement pour l'ordinateur qui se trouve juste un peu ralenti.

d)Conclusion

Ce que peut faire le RAID :

- réduire les risques de pertes de données en cas de défaillance d'une unité de stockage

- améliorer les performances

Ce que ne peut pas faire le RAID :

- protéger totalement des défaillances matérielles (éventualité de pannes successives de plusieurs disques ou du système RAID lui-même).
- protéger les données des défaillances du système d'exploitation (intrinsèques ou dues à un problème de configuration ou d'un conflit de composants) .
- protéger les données des erreurs humaines (suppression accidentelle de fichiers).
- protéger l'utilisateur des risques extérieurs au système (surcharge électrique qui grillerait l'ensemble des disques, incendie, vol, inondation, vandalisme).
- protéger les données des virus et des spywares qui pourraient corrompre les données.

Remarques :

- multiplier le nombre de disques multiplie les risques de panne. De plus, les disques utilisés par une grappe RAID sont souvent de même type et de même âge. Ils auront donc une durée de vie similaire, c'est pourquoi il faut faire attention à utiliser des disques qui soit du même modèle mais pas de là même série.
- la complexité du système RAID ajoute des risques technologiques, de rares mais possibles dysfonctionnements du contrôleur RAID ou du logiciel sont à craindre. Un disque défectueux peut aussi perturber le fonctionnement du contrôleur, logiquement ou électriquement ; ce qui peut causer la perte de plusieurs unités ; dans le cas d'un dysfonctionnement répété la perte totale d'une grappe voire de toute l'unité de stockage est possible.

Idéalement, un RAID 5 est fortement recommandé. Il permet non seulement d'accroître les performances mais de bénéficier aussi d'une bonne sécurité au niveau des données mais une lenteur importante dans la reconstruction des données perdues.
Cependant, le RAID 1 n'est pas à écarter. En effet, sa forte fiabilité peut s'avérer utile (fiabilité étant à privilégier vis-à-vis de la performance) même s'il est fortement consommateur de disque.

3.1.1.4.Quatrième point : Evolution du matériel

Lorsque les performances se dégradent (provoquant, de fait, le mécontentement des utilisateurs) et que tous les efforts ont été effectués pour optimiser les logiciels et l'architecture du système, la question de l'évolution du ou des serveurs se pose. Il en va de même en cas de défaillance d'un périphérique physique.

Faire évoluer le matériel est délicat, à plusieurs niveaux :

- Nécessité d'arrêter la machine / la machine virtuelle : peut être résolu en multipliant le nombre de serveurs. Lorsqu'un serveur est mis à jour les autres accusent la charge supplémentaire. Les performances globales sont dégradées mais le service reste utilisable.
- L'évolution du matériel peut être coûteuse et il faut avoir conscience qu'à un moment donné il est plus simple (et moins coûteux) d'acquérir un serveur supplémentaire plutôt que de continuer à mettre à jour ceux qui existent déjà.

3.1.1.5.Cinquième point : Tolérance globale aux pannes

La résistance aux pannes est accrue en multipliant le nombre de serveur.

Pour que l'utilisation des multiples serveurs soit totalement transparente pour les utilisateurs, il faut utiliser des adresses IP virtuelles (VIP ou VIPA). Il s'agit d'une adresse IP non connectée à un ordinateur ou une carte réseau spécifiques. Les paquets entrants sont envoyés à l'adresse IP virtuelle, mais en réalité ils circulent tous via des interfaces réseau réelles.

Ainsi, si un serveur tombe (ou est simplement arrêté pour une réparation), les performances globales du service sont dégradées mais sans aucune interruption.

3.1.1.6.Sixième point : Ce que peut apporter la virtualisation

Ici la virtualisation consisterait à déployer un serveur de grande puissance (coût d'achat élevé) virtualisant les différents petits serveurs prévus dans une architecture classique.

Un dispositif, appelé hyperviseur, héberge les machines virtuelles sur le serveur physique et les met à disposition à travers le réseau.

L'hyperviseur peut être vu comme un petit système d'exploitation, il est installé sur le serveur physique et l'occupe dans son intégralité : le serveur physique lui est entièrement dédié.

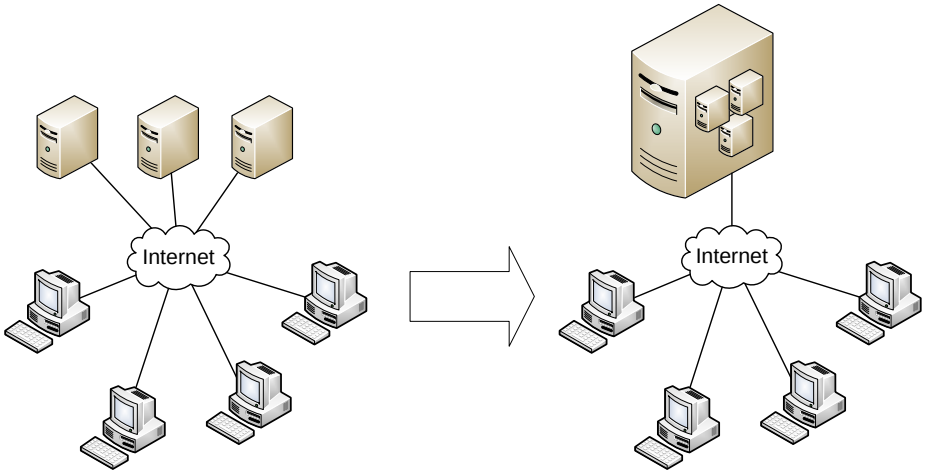


Fig. 18 – Ce qu'apporte la virtualisation

Cette solution de virtualisation possède un certain nombre d'avantages non négligeables :

- Plusieurs machines virtuelles peuvent fonctionner sur un hyperviseur.
- Les serveurs deviennent des fichiers. Ils peuvent être sauvegardés, maintenus, améliorés et déplacés sans difficulté.
- Tout est centralisé sur un serveur de forte puissance, l'administration (configuration, sécurité, etc.) est donc simplifiée.
- Réglages fins sur les ressources allouées à chaque serveur (espace disque, mémoire, nombre processeurs, cartes réseaux, etc.).
- Un unique gros serveur peut, sur le long terme, devenir plus rentable qu'une multitude de petits.

Cependant, d'autres problèmes peuvent apparaître :

- La centralisation de tous les serveurs virtuels sur un seul serveur physique impose de dimensionner avec la plus grande prudence les différents éléments qui le constituent. Toute panne du serveur physique se répercutera sur tous les systèmes virtualisés. De plus, cette centralisation rend le serveur plus vulnérable aux dégâts physiques (incendie, inondation, etc.) potentiels.
- La sauvegarde est simplifiée mais la restauration ne l'est pas. La granularité des sauvegardes est faible (le serveur virtuel entier est sauvegardé), ce qui complique la restauration d'un seul ensemble de fichiers. De plus, les volumes virtualisés sont très lourds et décuplent la charge pendant la sauvegarde.
- Faire évoluer le matériel sous-entend d'effectuer un achat important en une seule fois (les pièces de gros serveurs ne se changent pas aussi facilement que celles des petits).

3.1.2.Applicatif

3.1.2.1.Premier point : le système d'exploitation

Selon le système d'exploitation, le coût est variable (soit gratuit soit compris dans toutes sortes de fourchettes de prix) ainsi que l'ergonomie, la fiabilité et sécurité.

Nous avons choisi de mettre de côté les solutions sous Windows car elles ne sont pas adaptées à un serveur web (même si effectivement la technologie .NET permet d'écrire des services web).

Les solutions basées sur UNIX (Linux et BSD) se valent.

Côté Linux, les distributions généralement utilisées pour les serveurs sont :

- Red Hat (version entreprise, donc payante)
- Suse (version payante aussi)
- Debian (gratuit, réputé pour sa forte stabilité)

D'autres distributions Linux existent, comme Ubuntu Server, mais semblent moins adaptées.

Côté BSD, même si les distributions semblent moins populaires, on peut trouver :

- FreeBSD (gratuit, réputé pour sa forte fiabilité et ses performances)
- OpenBSD (gratuit, réputé pour sa forte sécurité)

Dans tous les cas, les deux sont compatibles avec la norme POSIX. Le choix se situera donc plutôt sur des questions de goûts.

3.1.2.2.Second point : l'applicatif serveur

Au niveau du serveur, les choix de langages du LikeOnSite impacteront le type d'application serveur à utiliser.

Les langages utilisés couramment pour l'écriture de services web sont :

- PHP, à travers un ensemble de librairies
- Java, par l'utilisation des EJB ou au moyen des librairies internes
- C#, mais celui-ci est exclu à cause du choix effectué au niveau du système d'exploitation (bien qu'il existe une implémentation libre du .NET, celle-ci reste limitée et sujette à des difficultés juridiques vis-à-vis de Microsoft).

Si le PHP est choisi, un simple serveur web suffira. Les principaux acteurs sont :

- Apache httpd (66,65 % du marché en 2009).
- IIS (Microsoft), exclu car uniquement sous Windows (18,68 % du marché en 2009).
- nginx (3,06 % du marché en 2009).

nginx est un serveur web récent qui a su tirer parti d'une architecture astucieuse lui permettant d'accroître les performances de manière assez impressionnante.

Par rapport à Apache, la consommation en ressources machine est moindre, les temps de réponse sont nettement moins important (voire constants).

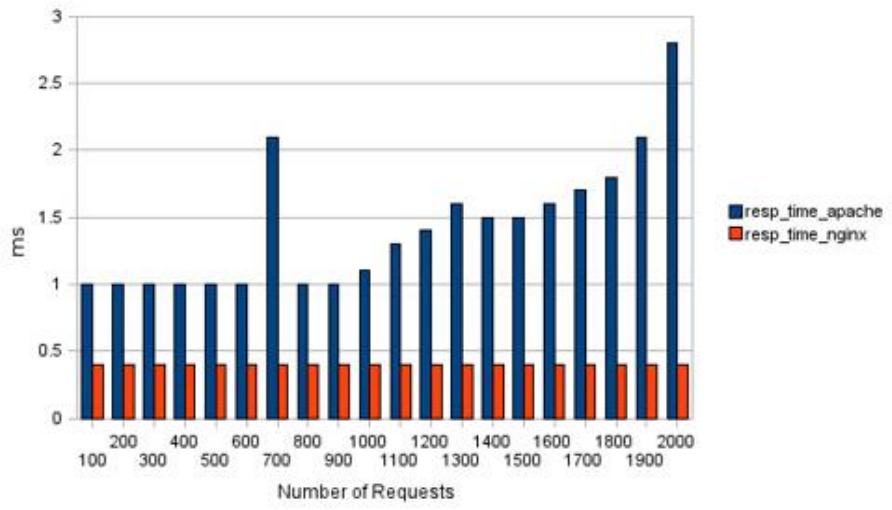


Fig. 19 – Temps de réponse Apache / nginx (source : Joe's Blog)

Toutefois, Apache httpd intègre de nombreux modules utiles pour le développement en PHP ainsi que des configurations plus fines.
De plus, la fondation Apache travaille sur une architecture similaire à celle de nginx, on peut donc tabler sur le fait qu'Apache httpd verra ses performances croître avec le temps.

Remarque : L'utilisation de PHP nécessite d'activer le module PHP sur le serveur web. Il semble que celui-ci dégrade légèrement les performances de nginx, réduisant l'écart de performances avec Apache httpd.

Si le Java est choisi, l'écriture des services sera plus confortable et des possibilités supplémentaires seront offertes par rapport au PHP.
Cependant, il ne faut plus utiliser un simple serveur web mais un serveur d'application. Il s'agit d'un logiciel qui va être en mesure d'exécuter les applications Java pour les faire communiquer avec l'extérieur au travers d'un serveur web (intégré au serveur d'application).

Si les services web java ne nécessitent que des servlets et des pages JSP, il est inutile d'utiliser un serveur d'application complet. En effet, des solutions plus légères (comme Apache Tomcat, qui n'intègre qu'un conteneur de servlet) ne sont pas des serveurs d'applications complets.
Toutefois, si les services web Java sont construits à partir d'EJB, il faudra alors songer à un serveur d'application plus complet (intégrant un conteneur d'EJB).

A ce niveau-là, l'offre est très diversifiée qu'il s'agisse de serveurs d'applications libres ou de serveurs d'applications propriétaires. Parmi les solutions gratuites on trouve les deux leaders suivants :

- JBoss (certifié JEE depuis peu)

- GlassFish / Sun Java System Application Server (mis au point par les concepteurs de Java)

Remarque : L'utilisation d'un serveur d'applications apporte des avantages certains, mais cela reste une solution globalement moins performante pour les petites charges.

3.1.3.Sauvegarde

Nous avons vu que l'utilisation du RAID pouvait garantir une certaine tolérance aux pannes. Cependant, ce n'est en aucun cas un système de sauvegarde fiable. En effet, une fausse manipulation ou un virus peuvent entraîner des dégâts qui seront répliqués sur tous les disques, rendant la sauvegarde inutile.
Il est donc nécessaire de disposer d'un outil de sauvegarde au niveau du système d'exploitation.
Les distributions Linux et BSD sont pourvus d'un grand éventail de solutions de sauvegarde dont un grand nombre est commun.
Ils permettent d'effectuer plusieurs types de sauvegardes (copie en masse), des plus brutes aux plus fines (sauvegarde différentielle, compression). Les opérations sauvegardes peuvent même être scriptées à la main.

Le choix doit être fait en fonction du volume de données et du réseau. Une architecture autour d'un ou plusieurs serveurs de stockage (réplication) communiquant avec le reste des serveurs avec des interfaces réseau dédiées sont une bonne solution.
Cependant, en plus de sauvegarde locale, il est préférable de disposer aussi de copies de sauvegardes distantes, transférées (par exemple) depuis le serveur de stockage vers un serveur distant au travers d'une connexion FTP (de préférence sécurisée).

Enfin, effectuer des sauvegardes est une chose mais encore faut-il qu'elles soient protégées aussi. Les sauvegardes doivent être au moins effectuées en double ou en triple, doivent être datées, etc.
Il est aussi nécessaire de s'assurer qu'elles ne soient pas corruptibles.

3.1.3.1.Premier point : fréquence des sauvegardes

La fréquence des sauvegardes est un point important à débattre. En effet, la restauration d'une sauvegarde entraîne des "retours en arrière" (ou "rollback") qui s'accompagnent inévitablement d'une perte de toutes les données enregistrées pendant le laps de temps écoulé entre la dernière sauvegarde et l'instant présent. Ces rollbacks sont généralement très mal perçus des utilisateurs.
Aussi, au plus les sauvegardes sont fréquentes, au moins la perte de données en cas de rollback est importante. Toutefois, en fonction du volume des données à sauver, la charge au niveau des machines peut être importante. Des sauvegardes trop fréquentes nuiraient aux performances (sans compter l'espace disque des serveurs de stockages) des serveurs mais aussi du réseau.

3.1.3.2.Second point : mise en situation

Même le meilleur système informatique du monde serait inefficace du moment que ceux qui l'utilisent ne sont pas correctement formés / entraînés. En effet, les problèmes graves n'arrivent pas tous les jours (heureusement) et c'est pour cela qu'il faut être correctement préparé lorsqu'ils surviennent.

Disposer d'un jeu de sauvegardes nécessite d'être en mesure de les restaurer efficacement et rapidement, en causant un minimum d'interruption de service.

Le domaine militaire nous donne l'exemple en effectuant régulièrement des manœuvres de simulation. Aussi, il peut être bénéfique de tenter d'effectuer une restauration de sauvegarde régulièrement, pour vérifier que la mécanique est bien huilée.

Cela permet d'identifier les maillons faibles et de prendre conscience de certaines difficultés techniques potentielles qui n'avaient pas été identifiées auparavant.

Ce genre de manœuvre devrait être effectué au moins chaque semestre.

3.1.4.Base de données

Avec l'augmentation des volumes des données et les contraintes liées à la performance ainsi qu'à la sécurité, les SGBD (systèmes de gestion de bases de données) proposent aujourd'hui plusieurs offres et architectures pour résister aux pannes et aux pics d'utilisation.

Remarque : Nous traiterons ici des clusters de bases de données et non pas des clusters de machines (opération consistant à transformer virtuellement plusieurs machines en une seule).

Le clustering consiste à regrouper plusieurs bases de données comme étant une seule. Cela a pour avantage de répartir la charge sur chaque base de données (placées sur des machines différentes appelées "nœuds").

En plus de répartir la charge, ils permettent aussi de minimiser l'impact d'une panne quant à la disponibilité des données.

3.1.4.1.Architecture Share-Nothing

Il ne s'agit pas, à proprement parler, d'une base clustérisée mais d'une base distribuée. Par défaut, cette architecture ne partage aucune information. C'est-à-dire qu'un nœud ne travaille qu'avec une partie de la base de données. De fait, il est plus adapté pour des applications facilement divisibles.

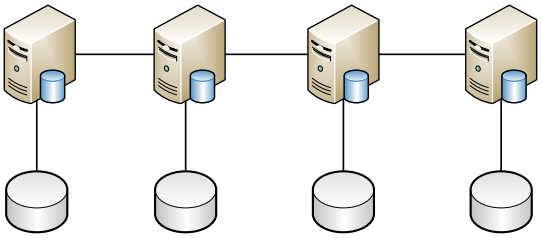


Fig. 20 – Architecture share-nothing sans espace de stockage partagé

Lorsqu'un nœud tombe, les données qu'il traitait deviennent indisponibles.

Cependant, si un nœud A et un nœud B partagent un même espace de stockage et si le nœud A tombe, alors le nœud B est en mesure de prendre en charge les données du nœud A. Cependant, la reprise des données nécessite un certain temps.

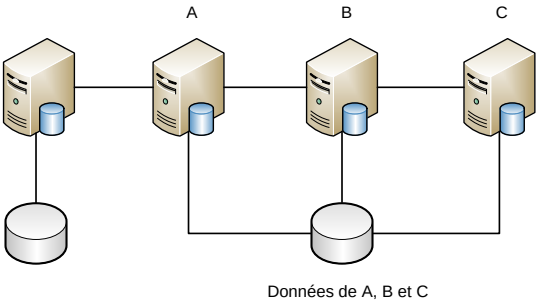


Fig. 21 – Architecture share-nothing avec espace de stockage partagé

De plus, comme le disque partagé va être sollicité par plusieurs machines, il est préférable d'utiliser là aussi un système de stockage distribué en réseau (SAN[1]).

3.1.4.2.Architecture Share-Nothing (avec réplication)

Cette fois-ci, chaque nœud héberge la totalité de la base de données. La synchronisation des modifications s'effectue en interne, de manière synchrone : tout se réplique sur tout ce qui peut altérer les performances de la base lorsque le nombre de nœuds augmente.

De plus, répliquer l'information peut être coûteux en disque (pour n Go de données et y nœuds, il faut n x y Go de disque) ainsi qu'en terme de temps processeur (car toutes les transactions se jouent sur chaque nœud).

Cependant, lorsqu'un nœud tombe, la sécurité est forcément assurée.

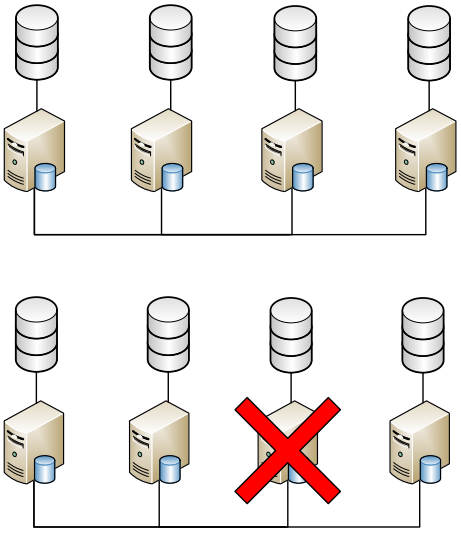


Fig. 22 – Architecture share-nothing répliquée

[1] Un réseau de stockage, ou SAN (de l'anglais Storage Area Network), est un réseau spécialisé permettant de mutualiser des ressources de stockage.

En définitive, ce système améliore les performances en lecture (distribution des requêtes de lecture à travers les bases) et permet d'obtenir une sécurité maximale en cas de perte d'un nœud. Toutefois, les performances en écriture peuvent se dégrader rapidement en fonction du volume et du nombre de nœuds.

3.1.4.3.Architecture Shared-disks

Ici, les données sont stockées sur un espace partagé (qui peut être un disque unique ou un stockage réseau de type SAN).
Sur cette base, tous les nœuds peuvent effectuer des requêtes en lecture, mais un seul d'entre eux est chargé de toutes les écritures.

Lorsqu'un nœud de lecture tombe, les autres sont à même d'assurer son rôle sans aucun problème.
Lorsque le nœud d'écriture tombe, un autre nœud rapatrie la base et devient nœud d'écriture.

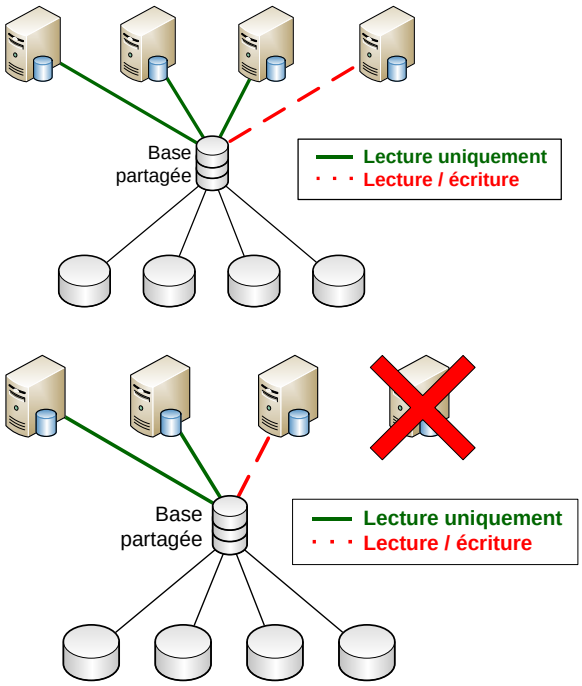


Fig. 23 – Architecture shared-disks

3.1.4.4.Architecture Shared-disks (avec fusion des caches)

Cette fois-ci, en plus de partager les disques, on partage aussi la mémoire de travail des différents nœuds : chaque nœud peut accéder à la totalité de l'information du système.

Un mécanisme sophistiqué de verrous et de synchronisation de blocs permet l'utilisation simultanée d'une donnée par plusieurs nœuds, ce de manière transparente du point de vue des applications.

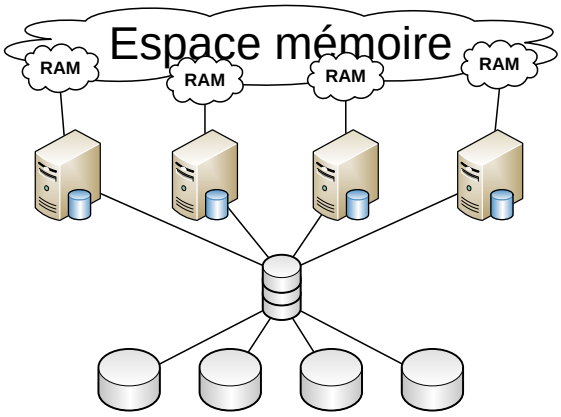


Fig. 24 – Architecture share-disks à fusion de mémoire

Lors de la chute d'un nœud, l'accès aux données n'est pas perdu et l'on peut continuer à travailler (avec toutefois une dégradation des performances, puisqu'un bloc de RAM n'est plus accessible).

3.1.4.5.Récapitulatif et choix possibles

Stratégie	Share nothing with replication	Share nothing	Shared disks	Shared disks with cache fusion
Editeur	MySQL, PostgreSQL	IBM	Sybase	Oracle
Ecriture depuis n'importe quel nœud	Oui	Non	Non	Oui
Performance	Non (en écriture)	Oui	Non	Oui
Interréplication	Oui	Non	Non	Non
Convient pour un grand nombre de nœuds	Non	Oui	Oui	Oui
Open source	Oui	Non	Non	Non
Gratuité	Oui	Non	Non	Non

Table 3 – Comparatif des architectures de résilience pour les bases de données

Du point de vue du LikeOnSite, les deux solutions les plus intéressantes sont :

- MySQL
  - ⇨ Avantages :
    - Déjà en place au niveau du LikeOnSite
    - Architecture share nothing with replication (donc sécurité absolue)
    - Gratuité
    - Relative simplicité de mise en œuvre
  - ⇨ Inconvénients :
    - Performance médiocre si beaucoup de nœuds
    - Forte consommation d'espace disque

Remarque : PostgreSQL serait un meilleur choix. Le clustering se base sur la même architecture mais les performances sont nettement meilleures (temps constant, gestions de grosses bases, etc.). De plus, PostgreSQL respecte la norme SQL.
Il est à noter que MySQL rattrape toutefois son retard.

- Oracle Database
  - ⇨ Avantages :
    - Performance élevée
    - Bonne sécurité des informations
  - ⇨ Inconvénients :
    - Coût élevé
    - Nécessite de disposer d'un spécialiste des produits Oracle

4.LOS Encoder

4.1.Rappel de la mission et éclaircissements

Dans le cadre du service LikeOnSite, plusieurs outils annexes doivent être développés. Cette partie concerne une application exécutable permettant de simplifier l'encodage de fichiers vidéos destinés aux LikeOnSite.

Concrètement, le but est de fournir une interface simplifiée permettant de sélectionner un fichier ou un lot de fichier dans un format quelconque (typiquement QuickTime, Stream DV, AVI) et de l'encoder en trois flux MPEG4/H264.AAC[1] avec des débits, c'est-à-dire des niveaux de qualité, différents (haut, moyen, bas).

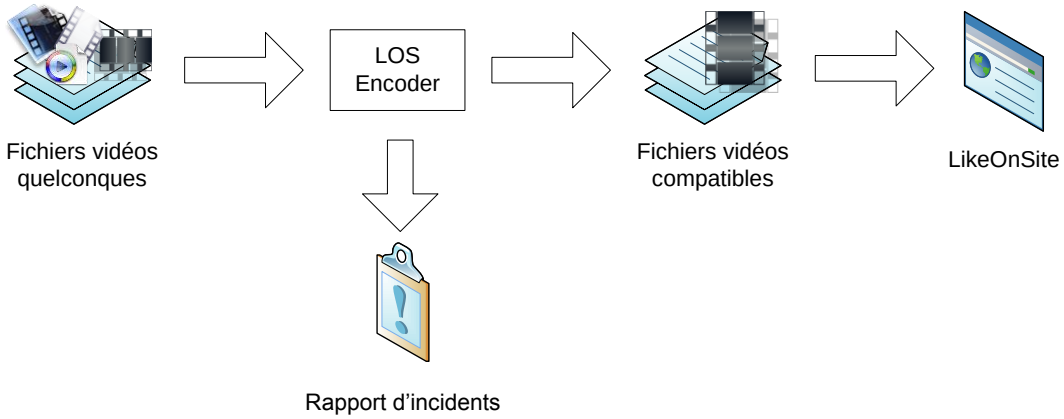


Fig. 25 – Fonctionnement global du LOS Encoder

Ce logiciel à vocation d'être distribué gratuitement et d'être siglé aux couleurs du LikeOnSite, ce à des fins promotionnelles.

4.2.Contraintes de travail

En entrée, les formats conteneurs[2] devant être supportés sont, par ordre d'importance :

- QuickTime (.mov)
- MPEG-4 partie 14 (.mp4)
- Audio Video Interleave (.avi)
- MPEG2TS (.ts)

Le LikeOnSite peut recevoir trois fichiers vidéos de qualité différente (et donc de poids différents) qui seront ensuite utilisés en fonction de la vitesse de la connexion Internet de l'utilisateur :

- Bas débit : ?[3] x 120, 12 FPS, 68 kbps vidéo VBR, 32 kbps audio VBR, f_E = 22 kHz

[1] Le MPEG4 (part 10) est un format conteneur, c'est-à-dire que l'on peut y encapsuler de l'audio, de la vidéo, des sous-titres, etc. L'H.264 est un format d'encodage pour vidéo. L'AAC est un format d'encodage pour l'audio.
[2] Voir note précédente.
[3] Le point d'interrogation représente une valeur proportionnelle à la largeur d'origine comme définie par la formule ci-contre.

$$\text{LargeurFinale} = \text{HauteurFinale} \times \frac{\text{LargeurOrigine}}{\text{HauteurOrigine}}$$

- Moyen débit : ? x 240, 25 FPS, 236 kbps vidéo VBR, 64 kbps audio, f_E = 22 kHz
- Haut débit : ? x 576 en SD et ? x 720 en HD, 25 FPS, 872 kbps vidéo VBR, 128 kbps audio VBR, f_E = 44 kHz

Un ensemble de règles régit les opérations et traitement à effectuer sur la réception d'un fichier vidéos. Celles-ci ont été synthétisées sous la forme d'une table de règle, présentée ci-dessous.

Format d'entrée	Bas débit	Moyen débit	Haut débit
768 x 576 (SD PAL pixels carrés, 4:3)	160 x 120	320 x 240	768 x 576
768 x 576 (SD PAL pixels carrés, 16:9 anamorphosé[1]	213 x 120	427 x 240	1024 x 576
1024 x 576 (SD PAL pixels carrés, 16:9)	213 x 120	427 x 240	1024 x 576
720 x 547 (SD NTSC pixels carrés, 4:3)	160 x 120	320 x 240	720 x 547
720 x 547 (SD NTSC pixels carrés, 16:9 anamorphosé	213 x 120	427 x 240	853 x 486
853 x 486 (SD PAL pixels carrés, 16:9)	213 x 120	427 x 240	853 x 486
1280 x 720 (HD 16:9)	213 x 120	427 x 240	1280 x 720
1980 x 1080 (full HD 16:9)	213 x 120	427 x 240	1280 x 720
Autres avec H <= 120	identique	identique	identique
Autres avec H <= 240	? x 120	identique	identique
Autres avec H <= 576	? x 120	? x 240	identique
Autres avec H <= 720	? x 120	? x 240	identique
Autres avec H > 720	? x 120	? x 240	? x 720

Table 4 – Règles de gestion pour la taille des vidéos

Débit vidéo en entrée	Bas débit	Moyen débit	Haut débit
< 100 kbps	identique	identique	identique
de 100 à 300 kbps	100	identique	identique
de 300 à 800 kbps	100	300	identique
> 800 kbps	100	300	800

Table 5 – Règles de gestion pour le débit vidéo

Débit audio en entrée	Bas débit	Moyen débit	Haut débit
< 24 kbps	identique	identique	identique
de 24 à 48 kbps	24	identique	identique
de 48 à 128 kbps	24	48	identique
> 128 kbps	24	48	128

Table 6 – Règles de gestion pour le débit audio

Fréquence d'échantillonnage audio en entrée	Bas débit	Moyen débit	Haut débit
< 22 kHz	identique	identique	identique
de 22 kHz à 44 kHz	22	22	identique
> 44 kHz	22	22	44

Table 7 – Règles de gestion pour la fréquence d'échantillonnage des signaux audio

Beaucoup d'encodeurs gratuits ne sont que des interfaces couplées à un encodeur de plus bas niveau, qui communiquent par l'intermédiaire des entrées/sorties standard de l'application. Aussi, si l'on veut bénéficier d'un retour, il faut généralement mettre en œuvre de fastidieuses opérations de parsing :

[1] Le 16:9 anamorphosé consiste à embarquer, dans un format 4:3, une image 16:9 déformée.

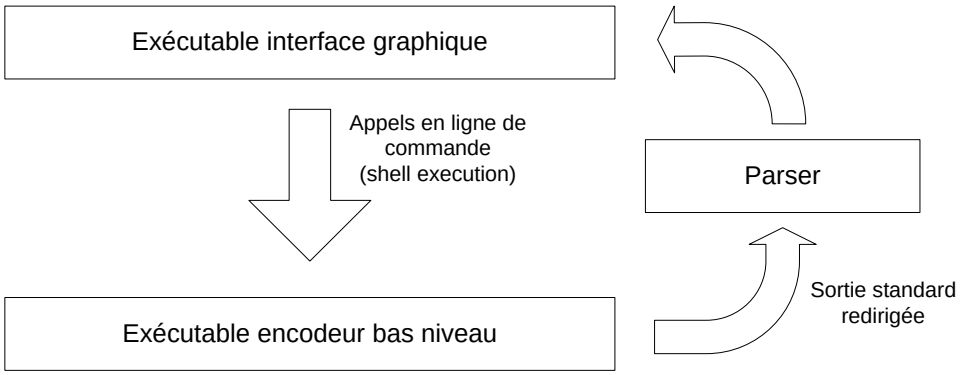


Fig. 26 – Fonctionnement standard des encodeurs basés sur d'autres encodeurs

Cependant, l'encodeur de bas niveau embarque déjà, en principe, la totalité des fonctionnalités de transcodage nécessaire au LOS Encoder. Aussi, il a été décidé de travailler directement avec les fonctionnalités offertes par les bibliothèques de l'encodeur bas niveau, ce pour l'adapter à nos besoins.

4.3.Problèmes classiques liés aux encodeurs vidéo

L'encodage vidéo est, aujourd'hui, plutôt complexe. La multitude des formats informatiques, ainsi que la démocratisation des outils d'encodages ont fait apparaître plusieurs problèmes.

Aussi, l'encodage d'un fichier vidéo pour le LikeOnSite est soumis aux problèmes suivants :

- Les contraintes d'affichage sur un fichier vidéo streamé sont moins fortes que pour de la diffusion cinématographique ou télévisuelle. De fait, en streaming, les standards d'affichages sont peu respectés.
- Un très grand nombre de formats de fichier existe et le LikeOnSite ne traite que du MP4.
- Les encodeurs professionnels sont coûteux et complexes alors que le LikeOnSite n'accepte qu'un seul format d'entrée, avec un nombre défini de tailles / débits possibles.
- Il existe, certes, de nombreux encodeurs gratuits, mais ceux-ci ne permettent pas un monitoring fiable car ils sont généralement basés sur des encodeurs en ligne de commande.

4.4.Choix techniques et architecturaux

4.4.1.Encodage audio / vidéo : FFmpeg

L'application LOS Encoder devant être distribuée gratuitement, l'utilisation d'une technologie d'encodage audio / vidéo libre est préconisée.

De ce côté-là, le choix s'impose de lui-même car il n'en existe qu'un seul. FFmpeg est une collection de logiciels libres écrits en C et dédiés au traitement de flux audio ou vidéo (enregistrement, lecture ou conversion d'un format à un autre).

Ce projet a été créé par Fabrice Bellard en 2000, et est maintenant maintenu par Michael Niedermayer. Il est actuellement hébergé sur les serveurs du projet MPlayer auquel il sert de base (ainsi qu'à VLC par exemple).

FFmpeg est publié sous licence GNU LGPL mais incorpore des modules sous licence GNU GPL. Développé sous Linux, il peut être compilé sur la plupart des systèmes d'exploitation, y compris Windows. Le projet est constitué de plusieurs composants, outils en ligne de commande ou bibliothèques logicielles :

- **ffmpeg** : l'utilitaire éponyme pour convertir des formats vidéo. Il supporte aussi l'encodage en temps réel depuis une carte TV.
- **ffserver** est un serveur HTTP dédié à la diffusion de contenu en continu (*streaming*).
- **ffplay** pour lire un fichier multimédia (il est basé sur les bibliothèques SDL et *FFmpeg*).
- **libavcodec** : bibliothèque contenant les encodeurs et décodeurs audio/vidéo de FFmpeg. La plupart des codecs ont été redéveloppés à partir de zéro (rétro-ingénierie) pour assurer de meilleures performances et la réutilisabilité du code source.
- **libavformat** : bibliothèque contenant un analyseur syntaxique et un générateur pour les formats audio/vidéo les plus communs.

4.4.2.Le langage de développement

De facto, comme les librairies avcodec et avformat sont écrites en C il est préférable d'utiliser soit du C soit du C++.

Cependant, le C n'offre pas le confort et l'intérêt d'un langage objet malgré ses performances notables.

Le C++, moins performant que le C, est toutefois adapté pour travailler avec les librairies susnommées. Qui plus, il est orienté objet et offre donc un confort de développement ainsi que des capacités de modularité et d'évolutivité.

4.4.3.Interface graphique : QT

Le choix de l'interface graphique est lié aux langages choisis. De facto, QT semblait parfaitement adaptée.

QT est un framework orienté objet et développé en C++ par QT Development Frameworks, filiale de Nokia. Elle offre des composants d'interface graphique (widgets), d'accès aux données, de connexions réseau, de gestion des fils d'exécution (thread), d'analyse XML, etc.

Qt permet la portabilité des applications qui n'utilisent que ses composants par simple recompilation du code source, ce qui est un avantage important pour le LOS Encoder (vocation à fonctionner sous Mac OS). Les environnements supportés sont les Unix (dont Linux) qui utilisent le système graphique X Window System, Windows et Mac OS X.

4.4.4.Communication FFmpeg et QT

La communication entre les librairies d'encodage et l'interface graphique a posé beaucoup de problèmes. En effet, la librairie ffmpeg a été développée en C (C99, ce qui le rend impossible à compiler sous Visual Studio).

A contrario, QT a été développée en C++ et apport de nouvelles fonctionnalités au langage (signaux, slots, etc.). De fait, QT exploite des outils de compilation qui lui est spécifique (notamment le qmake).

Aussi, après expérimentation, l'application ffmpeg n'a pas pu être compilée sous QT Creator. Deux solutions ont été envisagées pour rendre la librairie ffmpeg compatible avec QT.

4.4.4.1.Compilation séparée

L'opération consistant à transformer un code source, lisible par un être humain, en du code machine exécutable s'effectue en plusieurs étapes, exposées ci-contre.

L'idée consiste donc à compiler séparément l'application ffmpeg modifiée et l'interface graphique QT.

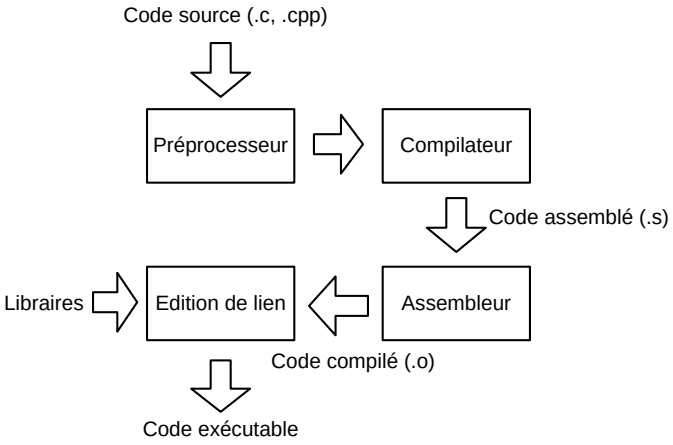
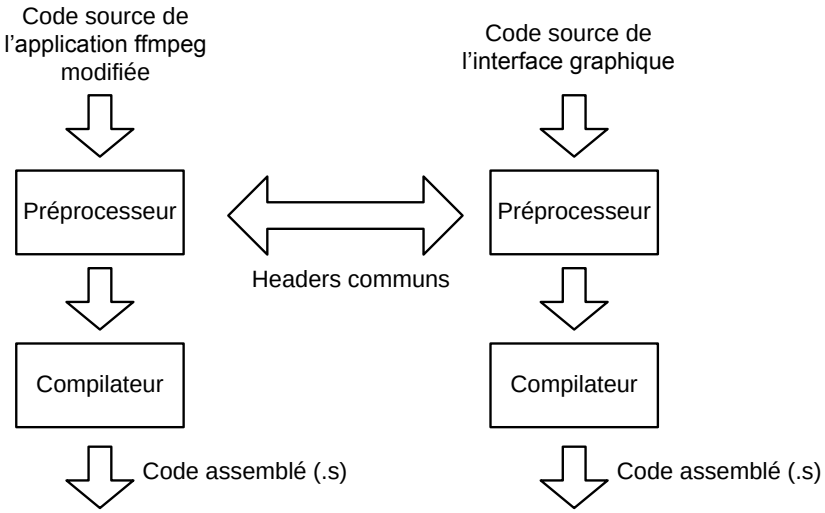


Fig. 27 – La compilation par étapes

Ainsi, les fichiers temporaires de compilation (.o) générés des deux côtés pourront subir une édition de lien commune et produire l'exécutable final.



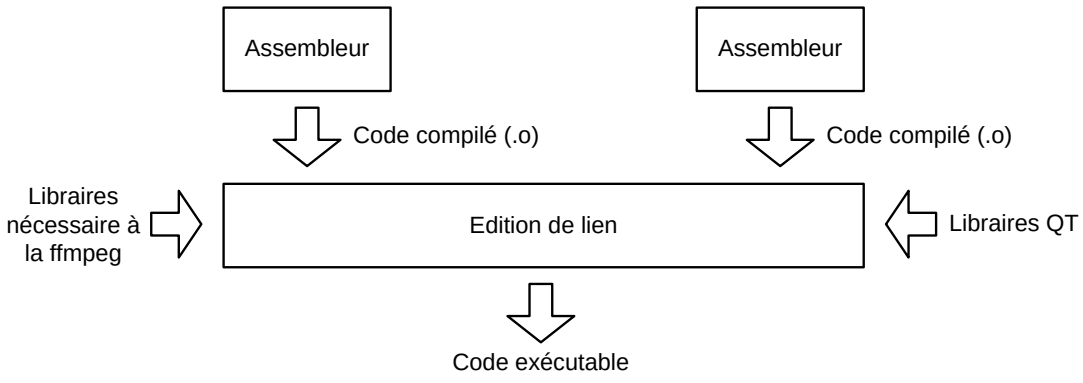


Fig. 28 – La compilation séparée

Avantages :

- Un seul exécutable produit
- Accès à tous les paramètres et toutes les options de l'application ffmpeg

Désavantages :

- Complexité de mise en œuvre (double compilation, édition de lien séparée, etc.)

4.4.4.2.Double processus

Une autre solution consiste à compiler deux exécutables :

- Interface graphique seule
- ffmpeg seule

Le lien entre les deux se situerait au niveau de l'interface graphique qui effectuerait les opérations suivantes :

1. Créer un processus fils
2. Remplacer le code source du processus fils par l'exécutable ffmpeg (exec)
3. Ouvrir des canaux de communications entre les deux processus (signaux, tubes, mémoire partagée, etc.) pour assurer les retours nécessaires depuis l'application ffmpeg.

Cette solution reste proche de l'architecture décrite dans "Technologies retenues", mais a pour avantage de ne pas nécessiter les opérations de parsing.

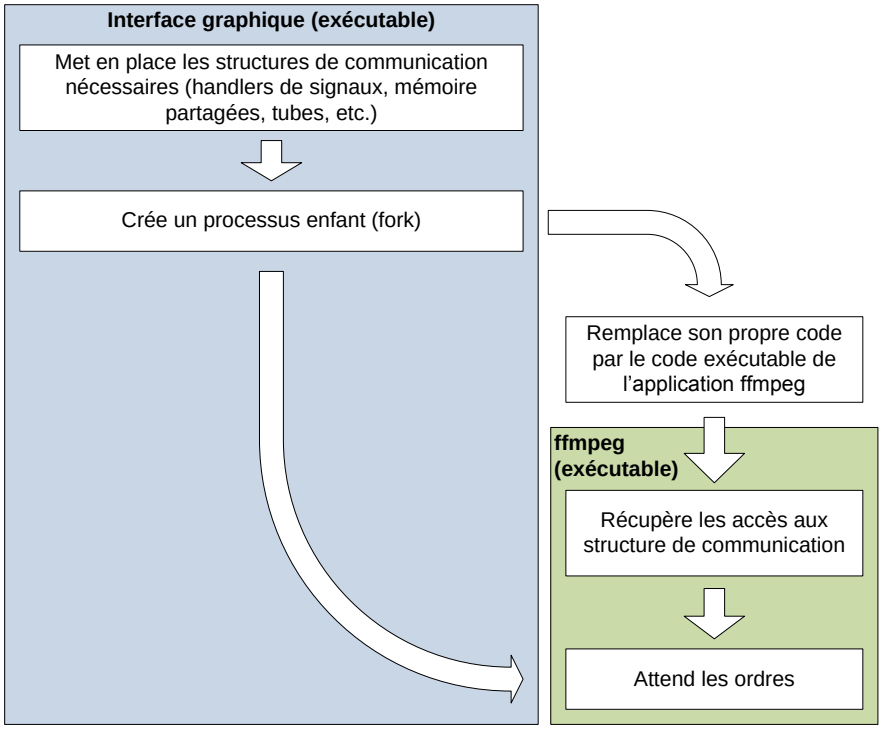


Fig. 29 – Architecture d'instanciation et de communication

Avantages :

- Simplicité de mise en œuvre
- Evolutivité (possible mise à jour / reprogrammation de la ffmpeg séparément)
- Répartition de charge entre les deux processus
- ffmpeg modifiée réexploitable par d'autres applications, selon le même procédé

Désavantages :

- Double processus
- Double compilation avec cette solution aussi (mais sans édition de lien séparée)

4.4.4.3.Solution retenue

La solution du double processus a été choisie à ceci près que les deux processus ne communiqueront pas au moyen de mécanismes systèmes mais de mécanismes réseaux.

Cela a pour avantage majeur de permettre une future extension de l'encodeur, avec un déploiement à travers un réseau.

4.4.5.Architecture de distribution (prototype pour l'avenir de l'encodeur)

Au point précédent nous avons vu que le LOS Encoder serait distribuable à travers un réseau (il s'agit d'une fonctionnalité non implémentée mais étudiée et prototypée pour de futures évolutions).

Le système consiste à transformer l'encodeur en système multiagent piloté par une unique interface graphique (système dit "primus inter pares"[1], où une entité arbitraire contrôle toutes les autres).

Ainsi, les agents de travail (appelés "nanocodeurs") sont dispersés sur un réseau et possèdent un espace de travail partagé. Une entité maîtresse (qualifiée de "nanomaster") reçoit les ordres de la part du technicien (typiquement une opération d'encodage).
Le nanomaster va alors broadcaster[2] une requête à travers le réseau pour prendre connaissance de tous les nanocodeurs présents. Les différents agents vont alors répondre ou non, selon s'ils ont déjà du travail ou pas.
Cette phase de question/réponse est appelée "phase de découverte". A l'issue de celle-ci, le nanomaster a connaissance de tous les agents disponibles. Il va alors découper, dans l'espace de travail, le fichier vidéo à encoder et assigner les différents morceaux de vidéo à chaque agent.
Lorsqu'un agent a terminé, il prévient le nanomaster que son travail est achevé et celui-ci lui donne une nouvelle tâche, s'il reste des morceaux de vidéo à encoder.
Lorsqu'un agent est en échec (problème quelconque), le nanomaster peut choisir de lui faire refaire la même tâche ou une autre. Il peut aussi choisir de délaisser ce nanocodeur et de retransmettre la tâche a un autre agent.
Au fur et à mesure du temps, les nanocodeurs effectuent des statistiques concernant leur taux d'erreur, leur vitesse, etc. De ces statistiques, le nanomaster extrait des "taux de performance" qui lui serviront plus tard à privilégier les agents les plus performants et à leur affecter des morceaux de vidéos plus grands que les autres.

Il a été envisagé d'introduire un troisième type d'entité (nommée "nanodoctor") qui serait chargée de gérer les taux de performances, de rechercher des solutions à des pannes, d'isoler ou de détruire les nanocodeurs trop peu performants (donc inutiles) voire contre-productifs.
De plus, le nanodoctor aurait pour rôle de communiquer avec un nanodoctor situé sur un réseau différent et de lui transmettre des requêtes du nanomaster.

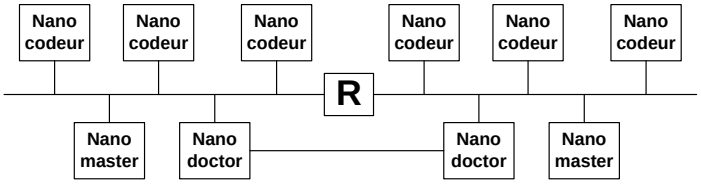


Fig. 30 – Exemple de communication entre deux réseaux d'agents

Cela permettrait de passer outre les routeurs, qui stoppent les communications broadcast, en possédant un agent point à point dans chaque réseau de travail visé.

[1] Par opposition aux systèmes dits "vox populi, vox dei", où l'entité de contrôle est désignée démocratiquement par toutes les autres.
[2] Le broadcasting désigne une méthode de transmission de données à l'ensemble des machines d'un réseau.

1.Faire le point

L'appréciation des résultats est difficile. En effet, au long du stage, pas moins de douze projets ont été étudiés, mais seulement deux ont été accompagnés d'un cycle de développement complet (en raison de leur urgence).
Il s'agit d'un choix de la part de l'entreprise, qui préférait obtenir un grand nombre d'études de projet pour les faire développer ensuite beaucoup plus tard par d'autres équipes.

Pour tous ces projets, autres que ceux menés à terme, il n'existe pas de résultats concrets, uniquement des documents d'analyse.
Ces documents décrivent les besoins inhérents à chaque projet (spécification et cahier des charges fonctionnel), les choix techniques et structurels (dossier d'analyse) ainsi que des indications sur le temps probable de réalisation et les coûts de développement engendrés.

1.1.Refonte LikeOnSite

Au niveau de la refonte du LikeOnSite, celle-ci est sensée apporter des solutions à tous les problèmes exposés. Si l'on confronte l'existant et ses défauts (problèmes de conception et de gestion de projet), les données laissées à l'entreprise sous forme de documents analytiques et de prototypes seront une bonne base pour structurer et gérer la refonte réelle.
Les solutions architecturales apportées répondent à toutes les contraintes exposées lors de la création du projet. De plus, elles intègrent des éléments classiques et surs d'une architecture modulaire (modèle en couche, faible couplage mais forte cohésion, etc.) qui permettront de réadapter l'application à moindre coût.
Les choix techniques effectués paraissent adaptés, tant au niveau des besoins qu'au niveau des compétences techniques à l'entreprise. Qui plus est, le découpage permet à des graphistes (non spécialistes du développement lourd) et à des développeurs de travailler ensemble et en parallèle, ce qui aura un impact positif sur la durée totale du projet.

1.2.Etude du serveur LikeOnSite

Là aussi, il s'agit d'une étude théorique accompagnée d'essais expérimentaux déployés en environnements fermés. Une précédente étude, très rapide, avait mis en évidence les différents points fondamentaux et mon travail a consisté à les développer et à en ajouter d'autres. Cependant, beaucoup de recherches ont été nécessaires pour appréhender les différents aspects qu'implique la création de ce serveur interne à Setinnov car il a fallu, d'une part, définir lesdits aspects fondamentaux et, d'autre part, se documenter sur les solutions pratiques de chacun d'entre eux.

Le document est destiné à servir de référence pour l'aide à la décision lors de l'estimation des coûts d'une telle machine. Le maximum d'options possibles a été envisagé et est exposé de manière simple pour pouvoir être compris par un non-spécialiste.

Bien sûr, il s'agit là d'une projection sur l'avenir (lorsque les fonds nécessaires seront disponibles), mais le document peut être facilement adapté pour la conception et l'aide à la décision autour de n'importe quel type de serveur, du fait de sa relative exhaustivité.

1.3.LOS Encoder

Il s'agit d'un projet mené à son terme (analyse et cycle de développement complet).

Distribué gratuitement sur le LikeOnSite et testé par divers partenaires, il semble remplir correctement son rôle. C'est-à-dire qu'il parvient à consommer correctement les formats demandés en entrée et les convertit dans les formats voulus.

Malgré un peu de retard dans le développement (environ 10%), il est utilisé à plusieurs niveaux. En effet, au départ destiné à rendre des fichiers vidéos compatibles LikeOnSite, il a été étendu à d'autres rôles (encodages de natures différentes).

L'outil semble suffisamment facile à utiliser (paramétrage et traitement de masse simplifié à l'extrême) pour rencontrer un certain succès auprès des non-informaticiens (professionnels du montage, par exemple). Un aperçu de l'interface graphique est fourni en annexe.

2.Support

Au niveau du support, j'ai laissé à l'entreprise une grande quantité de documents.

2.1.Refonte LikeOnSite

- Spécifications & Cahier des charges fonctionnel (89 pages)

  Il expose le contexte, les enjeux, les demandes initiales, les différentes fonctions et leur description détaillée (critères de validité, contraintes, importance, etc.)

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

- Le dossier d'analyse (134 pages)

  Ce dossier décrit, en premier lieu, une maquette de l'application (dessin) ainsi qu'un dictionnaire de données. Il contient aussi les descriptions algorithmiques de chaque fonction du système.

  Ensuite, pour chaque module, il contient les diagrammes concernant les différents composants et leurs interactions réciproques.

- La planification des tâches et les rapports journaliers

  Chaque jour, j'ai rédigé un rapport et l'ai envoyé sur un compte mail particulier. Collationnés ensemble, ces rapports décrivent l'avancée réelle de chaque projet.

Mis en relation avec l'agenda fixé en début de projet, il permet d'identifier les dérives ainsi que leurs causes dans le but d'en identifier les causes et de ne plus les reproduire.

- Plan de test (35 pages)

  Ce document reprend chaque fonction et les découpe en différents cas (fonctionnement normal, cas d'erreurs, sécurité, etc.). Il expose les réponses attendues de l'application pour chaque cas.

  Il est ensuite utilisé pour effectuer des "itérations de tests". Il s'agit de phases pendant lesquelles on reprend chaque test (même ceux effectués lors de la précédente itération) pour remplir un plan de test vierge. Cette fastidieuse revérification permet de s'assurer que le logiciel n'a pas régressé, c'est-à-dire que les nouvelles fonctionnalités n'ont pas perturbé celles qui fonctionnaient déjà.

  Ce plan a été très utile pour tester l'application LikeOnSite déjà existante.

2.2. Etude serveur

- Dossier d'étude (31 pages)

  En premier lieu, ce document fixe les différents points de l'étude, le but de celle-ci et le contexte de travail.

  Puis, point par point, il expose les éléments importants ainsi que les différents choix possibles. Lorsque cela est faisable, il décrit la solution la plus adaptée et entre plus en détail sur ses avantages et ses inconvénients.

  Il est complété par des conseils et par la définition d'un ensemble de bonnes pratiques pour amener à faire les choix les plus adaptés.

2.3. LOS Encoder

- Cahier des charges fonctionnel (29 pages)
- Le dossier d'analyse (49 pages)

  En plus de décrire le même type d'informations que ceux exposés précédemment, il intègre les spécifications autour des formats de fichiers ainsi que les spécifications du protocole de communication théorique des nanocodeurs.
- La planification des tâches et les rapports journaliers
- Plan de test (7 pages)

DISCUSSION

Les différentes missions qui m'ont été confiées ont essentiellement été des projets d'études. Leur but fut de produire des documents de référence (faisabilité, spécifications, cahier des charges, dossier d'analyse, plan de test), des documents de recherche ou des prototypes dans l'idée d'amener un développement futur lorsque le temps et le financement seraient disponibles.
Sur ces projets d'études articulés sur le vaste domaine de l'audiovisuel, il est difficile de dresser un bilan définitif quant à l'adéquation des solutions.
Globalement, les études et les prototypes ont permis de mettre en évidence la faisabilité technique des différents projets et les dossiers d'analyse permettront à l'entreprise de communiquer avec une maîtrise d'œuvre et d'en définir l'orientation technique et conceptuelle. C'est du moins dans ce but là qu'ils ont été écrits.

Cependant, au vu de l'évolution rapide du milieu ainsi que de l'entreprise, certains de ces projets ne seront probablement pas réalisés ou évolueront massivement pour s'adapter à de nouvelles fonctionnalités ainsi qu'à de nouveaux contextes techniques.
En termes de production, les techniques de modélisation utilisées ainsi que les technologies préconisées sont robustes et il est probable qu'elles évolueront peu. Tout du moins ont-elles peu de chance de disparaître. De fait, ces documents resteront lisibles et pourront même, éventuellement, servir de base à d'autres études.
Au niveau des modélisations diverses, j'ai pris soin de construire les architectures et les raisonnements sur des bases les plus génériques possible, pour permettre l'ajout de fonctionnalités, le changement des besoins ou même, plus simplement, la réutilisation de fragments d'architecture pour d'autres projets.

Toutefois, il s'agit d'analyse globale produite d'un bloc. Elle n'est là que pour donner des pistes et des bases pour l'architecture, les choix techniques ainsi que pour le découpage de l'application.
Aussi, elle est sensée s'enrichir au fur et à mesure du développement car chaque itération permettra d'avancer en profondeur sur les différents détails de l'analyse.
Fort heureusement, cette globalité apporte une certaine facilité à évoluer et une forte adaptabilité des documents.

Contextuellement, de nombreuses ressources physiques ont été mises à ma disposition qu'il s'agisse de serveurs, d'espace de stockage, de matériel ou de logiciels.

De même, les différents intervenants de Setinnov et d'OnTech ont su se montrer disponibles pour répondre à mes demandes et à mes questions, malgré un emploi du temps un peu délicat en ces périodes de vacances.

La philosophie des deux entreprises pour lesquelles j'ai travaillé fut assez peu directive, du moment que les délais et les attentes étaient respectés. J'ai bénéficié d'une grande autonomie et d'une grande liberté dans les choix à effectuer, en dehors des éléments concernant spécifiquement l'audio-visuel, sur lequel j'ai pu acquérir de nombreuses connaissances.
De mon point de vue, les méthodes de travail de ce milieu sont peu conventionnelles (style de management fortement délégatif) mais privilégient l'efficacité, la confiance, la productivité et la bonne entente entre les personnes. Je pense que c'est une façon intéressante de procéder, qui privilégie les rapports entre les individus plutôt qu'une hiérarchisation "à tout prix". Cela semble, en tout cas, faire ses preuves au vu du succès d'OnTech (intervenants réguliers, clients fidèles, bonne réputation et ouverture nationale) et du plaisir que j'ai pris à travailler dans ces conditions.

En dehors de ceci, la compagnie Setinnov était en création et finalisait le projet qui allait aboutir à sa concrétisation définitive. Ce fut très intéressant car cela m'a permis de découvrir un peu mieux, en discutant avec mon tuteur, les mécanismes et les difficultés de la création d'entreprise. Qui plus est, de par les démarches de Setinnov, cela m'a amené à mieux cerner les processus gravitant autour des études de marché.

Tous ces petits facteurs (contexte de travail, contexte de l'entreprise, multiplicité des projets, forts changements, etc.) ont rendu les missions extrêmement stimulantes et ont représentés des sortes de petits "défis" réguliers tant au niveau technique (technologies pointues) que relationnel (remontée de résultat, discussions techniques, formation, recherche commune de solutions, etc.).
Enfin, ces missions étaient orientées recherche et développement, milieu dont je suis fort peu familier et que je désirais explorer. Je constate que cela m'a apporté beaucoup plus de connaissances (découverte de technologie, implémentation dans des prototypes, documentation, débats scientifiques, etc.) et de satisfaction que dans un contexte de développement classique.

CONCLUSION

Lorsque j'ai postulé à Setinnov, j'étais intéressé par la forte orientation recherche (expérimentations, accès au LIRMM, etc.) et mes objectifs principaux se situaient d'une part dans la découverte des technologies exploitées dans le domaine audiovisuel et d'autre part à la découverte d'une structure de recherche.

Ces deux objectifs ont pu être atteints et même dépassés puisque j'ai aussi eu l'occasion d'aborder d'autres concepts passionnants appartenant à l'informatique de demain, comme la distribution d'une application au travers du paradigme de programmation par agents mobiles.

Je regrette seulement de ne pas avoir pu rencontrer les intervenants du LIRMM impliqués dans le projet de recherche de Setinnov, mais Cyril FASSIER a laissé entendre qu'il me recontacterait lorsque le projet se concrétiserait. Cela traduit plutôt bien, je pense, le côté positif de notre collaboration.

Les différents travaux effectués pour Setinnov m'ont finalement conduit à accompagner certains projets de l'entreprise, à les formaliser et à leur apporter une sorte d'existence rationnelle (une possibilité d'existence par la démonstration de leur faisabilité). Bien entendu, je n'ai pas pu aborder l'ensemble des projets imaginés par l'entreprise (ceux-ci pouvant naître et mourir en une même journée) car, bien que de petites tailles, ils étaient très nombreux.

D'autres projets, impliquant des compétences en mécanique et en électronique que je ne possédais pas n'ont pas pu être étudiées non plus.

D'un aspect professionnel, la formalisation effectuée au niveau des méthodes de travail de l'entreprise (conseils de gestion d'équipes, mise en forme des documents contractuels, méthodologies de gestion de projets, etc.) ainsi que les formations dispensées autour des projets devraient permettre à Setinnov de mieux appréhender ses futurs projets et, surtout, de contrôler avec plus de finesse ses processus de production logicielle, de l'idée jusqu'au déploiement.

De fait, j'espère que l'entreprise pourra se concrétiser dans l'avenir et réaliser les projets que j'ai étudiés et analysés.

A tous ces niveaux, travailler pour l'entreprise Setinnov a été une expérience extrêmement enrichissante, car j'ai pu d'une part assister à la naissance d'une entreprise porteuse d'un projet innovant et reconnu par le cœur de métier ciblé et d'autre part apporter des connaissances multiples, directement issues de mon cursus scolaire (gestion de projet, management, méthodes de travail, etc.) et de mon expérience personnelle.

Mon travail à Setinnov et à OnTech semble avoir été apprécié suffisamment pour que l'on fasse appel à moi pour une mission hors stage (contrat de travail de trois jours pour aller installer, avec un technicien d'OnTech, la télévision sur IP au conseil général de l'Aisnes, à Laon).
De plus, des opportunités me sont offertes pour de futurs stages (notamment au LIRMM, dans la division ICAR, pour le projet de recherche de Setinnov) et, peut-être, de futures collaborations.

Au final, ces diverses missions ont développé chez moi un certain goût pour la recherche, l'enseignement et les aspects scientifiques de l'informatique. Ces aspects, très peu abordés dans mon cursus scolaire un peu atypique, m'amènent à m'interroger sur sa pertinence générale et sur l'orientation à lui donner dans mon stage de troisième année.
Ce dernier stage, déterminant, servira d'impulsion initiale à ma carrière professionnelle et c'est par lui que je me dirigerais soit vers le monde du travail soit vers la poursuite de mes études.

GLOSSAIRE

Codec : Un codec (CODeur / DECodeur) est un composant permettant d'encoder une vidéo dans un format numérique donné.

Format conteneur : Il s'agit d'un format de fichier dans lequel on peut encapsuler de l'audio, de la vidéo, des sous-titres, des segments de code exécutable, etc.

Streaming : Opération consistant à lire un flux audio ou vidéo à mesure qu'il est diffusé. Cela s'oppose à la diffusion par téléchargement de fichiers qui nécessite de récupérer l'ensemble des données d'un morceau ou d'un extrait vidéo avant de pouvoir l'écouter ou le regarder.

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

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.

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.

Dérush : Le dérushage est la première étape du montage d'un programme audiovisuel, il consiste à sélectionner les séquences à utiliser lors du montage, appelées rushes, et à les transférer sur la plateforme de montage.

Le dérushage est l'étape qui permet de sélectionner les éléments qui seront utilisés lors du montage.

Transcodage : Le transcodage consiste à transformer un fichier vidéo numérique d'un format d'encodage A vers un format d'encodage B. Dans l'absolu, le format A et le format B peuvent être identiques, mais paramétrés différemment.

SSL : Protocole de sécurité pour les échanges sur internet.

JEE, EJB, JSP, servlet : JEE est une technologie basée sur le langage Java et dédiée aux applications d'entreprises. Elle offre plusieurs fonctionnalités spécifiques comme les servlets (des composants capables de se comporter comme des petits serveurs webs), des pages JSP (des servlets décrites sous forme de balises) et des EJB (des composants capables de communiquer naturellement en réseau).

Les applications JEE s'exécutent au sein d'un serveur d'application, composé généralement d'un serveur web, d'un conteneur de servlet et d'un conteneur d'EJB.

IDM : Ingénierie dirigée par les modèles. Les méthodes de travail IDM consistent à analyser et décrire les problèmes selon des méthodes éprouvées (UML) pour générer ensuite le code source correspondant dans n'importe quel langage.

IDL : Langage de description utilisé en ingénierie dirigée par les modèles.

TABLE DES ILLUSTRATIONS

Fig. 1 – Organigramme hiérarchique de Setinnov 4
Fig. 2 – Positionnement du média manager 5
Fig. 3 - Organigramme hiérarchique de la société OnTech 8
Fig. 4 – Quelques clients d'OnTech 8
Fig. 5 – Exemple d'arbre des fonctions 16
Fig. 6 – Exemple d'application de la méthode itérative 17
Fig. 7 – Exemple d'application de la méthode incrémentale 17
Fig. 8 – Cycle de développement en Y 18
Fig. 9 – Connexion d'un client standard 19
Fig. 10 – Connexion d'un client au travers de la marque blanche 20
Fig. 11 – Récapitulatif de l'architecture globale 20
Fig. 12 – Schéma de mise en œuvre de la traduction 23
Fig. 13 – Fonctionnement d'un DTO 24
Fig. 14 – Architecture globale du LikeOnSite 24
Fig. 15 – RAID 0 27
Fig. 16 – RAID 1 27
Fig. 17 – RAID 01 ..... **Erreur ! Signet non défini.**
Fig. 18 – RAID 10 ..... **Erreur ! Signet non défini.**
Fig. 19 – Ce qu'apporte la virtualisation 29
Fig. 20 – Temps de réponse Apache / nginx 32
Fig. 21 – Architecture share-nothing sans espace de stockage partagé 35
Fig. 22 – Architecture share-nothing avec espace de stockage partagé 35
Fig. 23 – Architecture share-nothing répliquée 35
Fig. 24 – Architecture shared-disks 36
Fig. 25 – Architecture share-disks à fusion de mémoire 36
Fig. 26 – Fonctionnement global du LOS Encoder 38
Fig. 27 – Fonctionnement standard des encodeurs basés sur la ffmpeg 40
Fig. 28 – La compilation par étapes 42
Fig. 29 – La compilation séparée 43
Fig. 30 – Architecture d'instanciation et de communication 44
Fig. 31 – Exemple de communication entre deux réseaux d'agents 45

Img. 1 – Quelques clients potentiels pour Setinnov 4
Img. 2 – Les organismes partenaires et/ou suivants Setinnov 4

Img. 3 - L'outil LikeOnSite (lecteur vidéo) 10
Img. 4 - Exemple de planning (mois de juillet) 15

Table 1 – Comparatif des technologies d'exécution de code distant 21
Table 2 – Comparatif des langages 22
Table 3 – Comparatif des architectures de résilience pour les bases de données 37
Table 4 – Règles de gestion pour la taille des vidéos 39
Table 5 – Règles de gestion pour le débit vidéo 39
Table 6 – Règles de gestion pour le débit audio 39
Table 7 – Règles de gestion pour la fréquence d'échantillonnage des signaux audio 39

SOURCES

Netographie

- *Pixels pas carrés ?*
  http://www.macetvideo.com/technique/Pixels/Pixels1.html
- *Informations techniques [sur la vidéo]*
  http://www.ubikprod.com/mc/informations-techniques.html
- *Définition des vidéos numériques*
  http://forum.mac-video.fr/articles/Definition/Definition.html
- *Tutoriaux sur les formats vidéos*
  http://www.dkdesign.fr/test/tutoriaux_video_formats.php
- *FFmpeg – la vidéo expliquée*
  http://www.jcartier.net/spip.php?article57
- *An ffmpeg Tutorial or How to Write a Video Player in Less Than 1000 Lines*
  http://dranger.com/ffmpeg/
- *FFmpeg officiel documentation*
  http://www.ffmpeg.org/ffmpeg-doc.html
- *FFmpeg (Ubuntu documentation)*
  http://doc.ubuntu-fr.org/ffmpeg
- *FFmpeg source code documentation*
  http://ffmpeg.org/doxygen/trunk/index.html
- *QT Online Reference Documentation*
  http://doc.qt.nokia.com/
- *FAQ QT (developpez.com)*
  http://qt.developpez.com/faq/

Exemples de code source pour le LOS Encoder

```
vector<OperationEncodage*> Projet::getEncodages()
{
    return encodages;
}

unsigned int Projet::getNbEncodages()
{
    return encodages.size();
}

void Projet::supprimerEncodage(int indiceEncodage)
{
    delete encodages[indiceEncodage];
    encodages.erase(encodages.begin() + indiceEncodage);
}

void Projet::vider()
{
    vector<OperationEncodage*>::iterator it;
    for(it = encodages.begin(); it != encodages.end(); ++it)
    {
        delete *it;
    }
    encodages.clear();
}
```

```
void MainWindow::on_startEncodingButton_clicked()
{
    ui->startEncodingButton->setEnabled(false);
    ui->stopEncodingButton->setEnabled(true);
    ui->listeEncodages->setEnabled(false);

    LOSEncoder::getInstance()->demarrerEncodages();
}

void MainWindow::on_destinationPreviousButton_clicked()
{
}

void MainWindow::on_destinationNextButton_clicked()
{
    ui->settingsOnglets->setCurrentWidget(ui->profilTab);
}

void MainWindow::on_destinationApplyButton_clicked()
{
    apply();
}
```

Fragments de code

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<?xml-stylesheet href="los-encoder-profile.xsl" type="text/xsl"?>
<profile xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="los-encoder-profile.xsd">

  <name>Force FPS to 25 (1:1)</name>

  <fps>25</fps>

  <video-codec format="h264"/>

  <container format="mp4"/>

</profile>
```

Un fichier de profil

Capture d'écran du LOS Encoder

LOS Encoder

Nouveau projet | Sauver projet | Ouvrir projet | Ajouter fichier | Démarrer encodages | Stopper encodages | fr | LikeOnSite

	Source	Destination	Profil	Etat
1	G:/dat/LOS Encoder/vidéos de tests/ratz15sDV.mov			
2	G:/dat/LOS Encoder/vidéos de tests/ratz15_Laurent.mov			
3	G:/dat/LOS Encoder/vidéos de tests/BATF1_DV.mov			
4	G:/dat/LOS Encoder/vidéos de tests/BA TF1_4_DV4_3.mov			
5	G:/dat/LOS Encoder/vidéos de tests/BA TF1_3_DV4_3.mov			
6	G:/dat/LOS Encoder/vidéos de tests/BA TF1_2_DV4_3.mov			
7	G:/dat/LOS Encoder/vidéos de tests/BA TF1_2_DV.mov			

Fichier source sélectionné

BATF1_DV.mov

Durée : 00:00:15
Type : SD PAL (ratio non standard)
Taille : 720x576
FPS : 25 (tot. 384)
Vidéo : dvvideo 28800kbps (mov)
Audio : pcm_s16be 1536kbps 48000Hz (mov)

16:9 anamorphosé ? | Forcer 4:3 | Forcer 16:9

Paramètres

Destination | Profil | Résumé

Pour la sélection | Pour tous les éléments

LOS Bas débit (? x 120, 12 FPS, H264 68kbps, AAC 32kbps 22kHz)
LOS Moyen débit (? x 240, 25 FPS, H264 236kbps, AAC 64kbps 22kHz)
LOS Haut débit (? x 576 SD ou ? x 720 HD, 25 FPS, H264 872kbps, AAC 128kbps 44kHz)
Profil personnalisé | Force FPS to 25 (1:1)

Précédent | Suivant | Appliquer

Un échange protocolaire entre le nanomaster et un nanocodeur

```
greys@ubuntu: ~
File  Edit  View  Terminal  Help
Broadcasting discovery packet...
Waiting for response...
Nanocoder detected : MY NAME IS CODER, NANO CODER (192.168.1.39 : grade 6)
Waiting for response... => timeout (total : 1)
Broadcasting mindcontrol to nanocoders found...
Nanocoder [192.168.1.39] : I AM YOUR SLAVE (ready to work).
Splitting file ./video_test1.mp4 [38M, H.264/AAC] in /mnt/nfs_workspace/ => Done (8 subfiles)
Sending work to 192.168.1.39 [subfile1]...
All nanocoders (1) receive task ! 7 subfiles unassigned.
Nanocoder [192.168.1.39] : START ENCODING [filetype = H.264/AAC.MP4, 98s/2941 frames will be processed].
Nanocoder [192.168.1.39] : ENCODING STATUS => logged in file (log454687687213)
Nanocoder [192.168.1.39] : ENCODING STATUS => logged in file (log454687687213)
Nanocoder [192.168.1.39] : ENCODING STATUS => logged in file (log454687687213)
Nanocoder [192.168.1.39] : ENCODING STATUS => logged in file (log454687687213)
Nanocoder [192.168.1.39] : ENCODING STATUS => logged in file (log454687687213)
Nanocoder [192.168.1.39] : ENCODING STATUS => logged in file (log454687687213)
Nanocoder [192.168.1.39] : ENCODING STATUS => logged in file (log454687687213)
Nanocoder [192.168.1.39] : ENCODING STATUS => logged in file (log454687687213)
Nanocoder [192.168.1.39] : ENCODING STATUS => logged in file (log454687687213)
Nanocoder [192.168.1.39] : ENCODING STATUS => logged in file (log454687687213)
Nanocoder [192.168.1.39] : ENCODING STATUS => logged in file (log454687687213)
Nanocoder [192.168.1.39] : ENCODING STATUS => logged in file (log454687687213)
Nanocoder [192.168.1.39] : ENCODING STATUS => logged in file (log454687687213)
Nanocoder [192.168.1.39] : ENCODING STATUS => logged in file (log454687687213)
Nanocoder [192.168.1.39] : ENCODING STATUS => logged in file (log454687687213)
Nanocoder [192.168.1.39] : ENCODING STATUS => logged in file (log454687687213)
Nanocoder [192.168.1.39] : ENCODING STATUS => logged in file (log454687687213)
Nanocoder [192.168.1.39] : ENCODING STATUS => logged in file (log454687687213)
Nanocoder [192.168.1.39] : ENCODING STATUS => logged in file (log454687687213)
Nanocoder [192.168.1.39] : ENCODING STATUS => logged in file (log454687687213)
Nanocoder [192.168.1.39] : ENCODING STATUS => logged in file (log454687687213)
Nanocoder [192.168.1.39] : ENCODING STATUS => logged in file (log454687687213)
Nanocoder [192.168.1.39] : ENCODING STATUS => logged in file (log454687687213)
Nanocoder [192.168.1.39] : TASK ENDED [file : /mnt/nfs_workspace/video_test1__1.avi]
1 subfile ended (Nanocoder 192.168.1.39).
Sending mindcontrol to nanocoder 192.168.1.39...
Nanocoder [192.168.1.39] : I AM YOUR SLAVE (ready to work).
Sending work to 192.168.1.39 [subfile2]...
SIGTERM receveid ! NANOMASTER WILL BE STOPPED !
Sending end request to nanocoders (1)...
```

Le nanomaster

```
greys@ubuntu: ~
File  Edit  View  Terminal  Help
Waiting for tasks...
Receiving : I AM THE MASTER, WHO ARE YOU ? [192.168.1.97]
Waiting mindcontrol confirmation...
Receiving : BREAKING NEWS, I AM EVERYONE ! [master found : 192.168.1.97, d=3600s]
Task stack size : 0 => Sending ready to work status...
Waiting for work...
Task received [/mnt/w/video_test 1__1.mp4(OK)]. Neeed encoded to [XVID/MP3.AVI]...
Working...
Sending status...
Sending status...
Sending status...
Sending status...
Sending status...
Sending status...
Sending status...
Sending status...
Sending status...
Sending status...
Sending status...
Sending status...
Sending status...
Sending status...
Sending status...
Sending status...
Sending status...
Sending status...
Sending status...
Sending status...
Sending status...
Sending status...
Sending status...
Sending status...
Enconding done (2941 frames encoded in 139s)... Confirmation to master...
Waiting mindcontrol confirmation...
Receiving : BREAKING NEWS, I AM EVERYONE ! [master : 192.168.1.97, d=3459s]
Task stack size : 0 => Sending ready to work status...
Waiting for work...
Receiving : EXTERMINATE ! => Master [192.168.1.97] disconnected
Waiting for tasks...
```

Le nanocodeur