Travail de Bachelor réalisé en vue de l'obtention du diplôme HES

CONCEPTION ET RÉALISATION D'UNE APPLICATION DE GESTION POUR AVOCAT À PARTIR D'OPENERP ET GÉNÉRALISATION DE CETTE DÉMARCHE À D'AUTRES PROFESSIONS LIBÉRALES

Par :

Thierry Desbaillet

Conseiller au travail de diplôme :

Thierry Ceillier

Genève, le 11 septembre 2009

Haute École de Gestion de Genève (HEG-GE)

Informatique de Gestion

Remerciements

Je remercie M. Thierry Ceillier pour ses recommandations durant tout le déroulement de ce travail de diplôme ainsi que la société Prolibre et spécialement M. Gilbert Robert pour son soutien et ses conseils pour la réussite de ce projet.

Déclaration

Ce travail de Bachelor est réalisé dans le cadre de l'examen final de la Haute école de gestion de Genève, en vue de l'obtention du titre d'Informaticien de Gestion. L'étudiant accepte, le cas échéant, la clause de confidentialité. L'utilisation des conclusions et recommandations formulées dans le travail de Bachelor, sans préjuger de leur valeur, n'engage ni la responsabilité de l'auteur, ni celle du conseiller au travail de Bachelor, du juré et de la HEG.

« J'atteste avoir réalisé seul(e) le présent travail, sans avoir utilisé des sources autres que celles citées dans la bibliographie. »

Fait à Genève, le 11. septembre 2009

Thierry Desbaillet

Table des matières

Résumé (« Executive Summary »)

Ce rapport illustre le travail réalisé afin de créer un outil de gestion intégré pour la profession d'avocat. Notre décision a été d'utiliser un programme « open source » : « OpenERP » comme base de l'application de gestion intégrée, de l'adapter et de développer les fonctionnalités manquantes.

L'application peut être testée à l'adresse suivante : http://avocat.prolibre.com avec l'utilisateur demo et le mot de passe demo.

Le développement d'application de bout en bout est un concept inadapté pour le marché que nous visons : Les avocats (Indépendants ou en cabinets). Nous avons opté pour des programmes appelés ERPs (Entreprise Ressource Planning) ou PGI (Progiciel de Gestion Intégré) qui sont des programmes généraux que l'on peut personnaliser et auxquels on peut ajouter les fonctionnalités qui nous manquent.

J'ai utilisé OpenERP qui est un ERP Open Source très facilement personnalisable. L'accès à son code source donne un accès à son système de fonctionnement et j'ai pu l'enrichir des fonctions dont j'avais besoin pour qu'il corresponde aux spécificités des avocats.

J'ai tout d'abord réalisé un cahier des charges, afin de connaitre les besoins spécifiques des avocats et savoir comment configurer l'ERP.

Nous avons décidé d'externaliser en Inde le développement des fonctionnalités manquantes. Pour se faire, nous avons engagé cette activité en collaboration avec la société Sisalp.

Il a ensuite fallu définir et mettre en place une infrastructure informatique permettant l'installation et la mise à jour de l'application, en garantissant la meilleure sécurité.

J'ai également réalisé un business plan pour connaitre le marché des avocats, savoir ce dont ils avaient besoin comme fonctionnalités, préciser leurs méthodes de travail, et définir comment formuler une offre afin de vendre notre application.

Ce dossier vous permettra de découvrir

- la problématique de configuration d'un ERP
- l'installation
- la définition d'un cahier des charges pour son intégration
- la gestion de l'outsourcing
- la réalisation d'une offre commerciale pour vendre cette intégration.

Après la réalisation de cette application pour avocat, l'étape suivante est de réutiliser la même méthode pour différentes professions libérales dont le mode de fonctionnement varie peu.

Introduction

Au début de l'informatique, les programmeurs ont développé un programme par client.

Avec le temps, ils ont regroupé les clients par domaine d'activité et réalisé des programmes moins spécifiques et de plus en plus généraux. C'est dans ce contexte que les ERPs (Entreprise Ressource Planning) ou PGI (Progiciel de Gestion Intégré) ont vu le jour.

Un ERP est un programme qui intègre la gestion informatique de l'ensemble des activités de l'entreprise. Il couvre entre autre les achats, les ventes, la gestion des stocks, la gestion des produits, la gestion des projets, les timesheets, la comptabilité, la relation client, bref tout ce qui touche à l'activité de l'entreprise. Il permet d'éviter de ressaisir l'information et la multiplication des programmes de gestion au sein d'une même entreprise. Le monde des ERPs est dominé par SAP qui est destiné aux grandes entreprises mais un petit projet open source essaye de tirer son épingle du jeu.

OpenERP nommé tout d'abord TinyERP (car édité par la société belge Tiny sprl) est un ERP totalement gratuit et open source. Tout comme SAP il essaye de couvrir l'ensemble des besoins des entreprises mais contrairement à lui, il s'attaque au marché des PME[1] et à celui des TPE[2].

J'ai commencé à travailler avec cet ERP pour mon travail de GREP; dans lequel j'ai intégré ce programme avec 2 autres étudiants dans l'entreprise de mes parents, le Domaine des Molards. Une fois cette intégration réalisée, j'ai été voir M. Gilbert Robert de la société Prolibre afin qu'il assure le suivi de ce projet. Il m'a indiqué que ses travaux concernant OpenERP étaient laissés de côtés à cause de restructuration interne et il m'a proposé de m'engager afin de m'occuper des projets OpenERP. Très rapidement, nous nous sommes rendu compte qu'il fallait proposer une solution basée sur OpenERP pour un domaine spécifique afin de gagner du temps dans l'intégration de cet outil et le proposer à un prix très compétitif.

Prendre un programme général comme un ERP et créer une déclinaison spéciale qui correspond à toutes les entreprises d'un secteur d'activité m'a beaucoup attiré car le paramétrage d'un ERP est ce qui coûte le plus cher. En augmentant le nombre de client pour les mêmes développements, on arrive à une situation de base où l'on peut proposer un produit très bien adapté aux besoins, pour un coût très faible par rapport aux fonctionnalités proposées. J'ai été engagé par Prolibre pour créer cette déclinaison pour les avocats et pour réaliser l'offre commerciale correspondante. Je fais mon travail de bachelor sur le sujet car ce type d'implémentation est nouveau et permet de trouver une façon de démocratiser les ERPs et en faire bénéficier le plus d'entreprises possible.

[1]Entreprise entre 10 et 200 employés

[2]Entreprise entre 1 et 10 employés

L'entreprise ProLibre

Prolibre est une Société à responsabilités limitées dont le directeur est monsieur Gilbert Robert. Elle est composée de monsieur Gilbert Robert, d'un employé chargé de l'infrastructure, d'une secrétaire et de moi même qui m'occupe des projets OpenERP.

L'entreprise est spécialisée dans les réseaux et les infrastructures informatiques d'entreprise, mais également dans la gestion de serveurs informatiques et d'applications Open Source.

Depuis fin 2006, Prolibre a travaillé avec la société Tiny sprl pour la mise en place d'OpenERP dans diverses sociétés et a contribuer à créer la « Swiss OpenERP Alliance » fin 2007.

Réaliser le travail de bachelor dans le cadre de cette société m'a beaucoup apporté.

- Au niveau de l'infrastructure informatique, je n'ai pas eu besoin de gérer la problématique d'installation et de maintenance des serveurs. Elle est réalisée par un collègue dont c'est le métier.
- Au niveau de l'expertise : monsieur Gilbert Robert a l'habitude de gérer des projets d'intégration. Il peut donc me conseiller et me guider lorsqu'il y a des problèmes à résoudre.
- Prolibre m'apporte son réseau de partenaires, ce sont d'autres sociétés qui implémentent OpenERP dans d'autres cantons ou d'autres pays. Nous avons des contacts privilégiés avec eux et nous nous entraidons en cas de développements spécifiques ou d'intégration de grande ampleur.
- Finalement, Prolibre m'apporte de la crédibilité, car lorsque l' on est jeune et qu'on sort de l'école, les chefs d'entreprise ne font pas forcément confiance aux jeunes sans beaucoup d'expérience, surtout pour des projets comme ceux d'intégration d'un ERP dont toute l'administration de son entreprise va dépendre. Grâce à mon travail dans cette société, je peux donc plus facilement avoir des rendez-vous et décrocher de nouveaux contrats.

L'Open Source

L'open-source regroupe l'ensemble des programmes dont l'utilisation est gratuite, et le code source est ouvert et consultable par n'importe qui. Les programmes open source les plus connus sont Firefox, OpenOffice, Apache, MySQL. Actuellement on peut faire fonctionner un ordinateur entièrement sans payer de licence. Des systèmes d'exploitation Linux comme Ubuntu, Debian, Mandriva, OpenSuse et bien d'autres s'installeront sur votre machine sans avoir besoin de payer la moindre licence. Ensuite les programmes de traitement de texte OpenOffice, le navigateur Firefox, et bien d'autres programmes gratuit vous permettront d'utiliser votre ordinateur en toute liberté. Il est important de signaler que tous les programmes gratuits ne sont pas forcément open source.

Quelques avantages à travailler avec des logiciels open source.

Plus de gestion de licence

Un logiciel libre sous licence GPL (Licence public générale), est en règle générale gratuit.

Nombre illimité d'utilisateurs

Il n'y a plus de restriction d'usage en fonction du nombre d'utilisateurs ou du prix de la licence. Ici seule la limitation de la puissance du ou des serveurs entre en compte.

Maîtrisez et contrôlez l'évolution de votre informatique

Vous maîtrisez les mises à jour du système d'exploitation et du matériel. Vu qu'il n'y a pas de mise à jour payante ou obligatoire, vous pouvez utiliser la version que vous désirez et adapter les logiciels en fonction des ressources que vous avez à disposition.

Sécurité des applications et des données

Il n'y a pas non plus de problèmes de virus, car les systèmes d'exploitation sont plus fiables dans leur architecture et la réalisation de virus est plus complexe que pour un système d'exploitation Windows.

Indépendance vis à vis de l'éditeur

En cas de faillite d'un éditeur de logiciel, bien souvent le code n'est pas libre, donc l'utilisateur ne peut pas continuer le projet. Dans l'open source, le code est libre donc si quelqu'un veut continuer à l'améliorer il peut le faire sans contrainte.

Standards ouvert

Les spécifications qui ont été publiées assurent la pérennité de la solution. Si une personne ou un groupe de personnes désire prendre comme base un programme open source et développer sa version il peut le faire librement.

Dynamique des outils garantie

Mise à jour, rapport de bug, nouvelles fonctionnalités, etc. sont disponibles puisque la communauté des utilisateurs est très importante et que chacun a intérêt à rendre publique sa contribution afin que les autres l'améliorent à leur tour, les projets évoluent très vite et une communauté vient rapidement se créer autour d'un projet.

Adéquation des programmes avec les besoins utilisateurs

Les utilisateurs développent les fonctions dont ils ont besoin, les programmes sont donc en adéquation avec leurs besoins. Les applications ne sont pas développées par des acteurs (Marketing ou chefs de projets souvent déconnectés du terrain).

Quelques inconvénients à travailler avec des logiciels open source.

Pérennité de l'application

Un logiciel qu'il soit open source ou propriétaire n'est nullement éternel, ce qui signifie qu'il faut se renseigner sur la ou les personnes qui le développent afin d'estimer sa pérennité.

Migration entre les versions et stabilité

Dans l'open source, les contributeurs sont rarement tous centralisés dans la même entreprise, ce qui implique une gestion des corrections et des améliorations beaucoup plus difficile à réaliser. Il faut qu'un groupe s'occupe de la modération afin d'assurer la stabilité entre les versions et la suite dans le développement du programme.

Couverture métier

Si les besoins des utilisateurs sont trop spécifiques, il n'y aura sûrement pas une communauté assez importante pour développer le programme dont ils auront besoin. Le recours à un logiciel propriétaire ou le financement intégral du produit sera donc inévitable.

Nombre d'utilisateurs

Ce qui fait la force d'un programme c'est sa communauté. Vu que le nombre d'utilisateurs de programmes libres est moins grand que celle des programmes propriétaire, la recherche d'information est plus difficile et donc l'apprentissage est également plus compliqué. Le bouche à oreille est donc moins grand, mais cette tendance commence à s'inverser grâce à la fondation Mozilla ou d'autre grands éditeurs de solutions open source qui mettent en avant leur produits grâce à des campagnes de publicité par exemple.

Partie Technique

Voici la démarche que j'ai suivie afin de définir quels sont les besoins d'infrastructure informatique afin de mettre en place OpenERP chez les avocats.

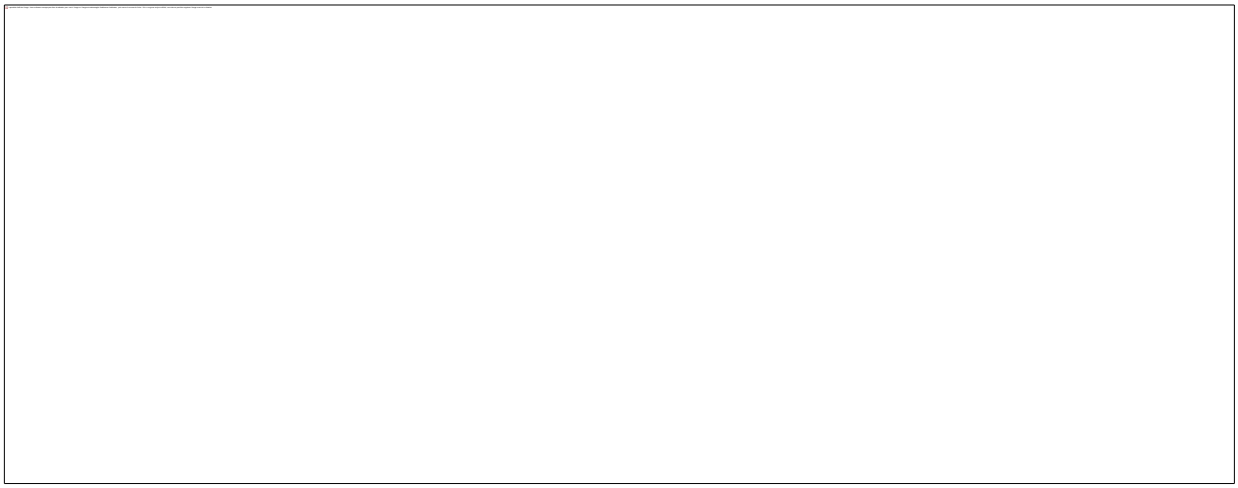
Tout d'abord, j'ai défini ce qui existait dans OpenERP pour la gestion d'un cabinet d'avocat. Quels modules étaient nécessaires afin de pouvoir gérer des timesheets, des rendez-vous, des dossiers, etc. J'ai ensuite pris rendez-vous avec plusieurs avocats pour connaitre leur façon de travailler et finalement j'ai défini un cahier des charges pour développer les fonctionnalités manquantes à un cabinet d'avocat.

Mais avant tout, je vais vous décrire ce qu'est OpenERP et quelle est l'infrastructure informatique nous avons mise au point afin de l'installer et de gérer les serveurs.

Définition d'OpenERP

OpenERP est un progiciel de gestion intégré Open Source édité par la Société belge Tiny sprl. Il est disponible en licence GNU/GPL. Il dispose d'un réseau de développeurs mondial et plus de 350 modules. Tous les développements sont coordonnés par Tiny sprl et intégrés dans le système de gestion des versions de Launchpad.

OpenERP est basé sur le framework de la Société Tiny sprl OpenObject. Il permet d'avoir une architecture client-serveur, d'avoir des workflow, une interface et des rapports personnalisables, une gestion de l'internationalisation[3], et une interface xml-rpc.



OpenERP à besoin pour fonctionner du langage de programmation Python, de quelques librairies spécifiques et d'une base de données PostgreSQL. Pour se connecter à cet ERP, les utilisateurs peuvent soit utiliser un client web avec les navigateurs Internet Explorer ou Mozilla Firefox par exemple, ou alors utiliser les clients GTK[4] pour Windows, Linux, ou Macintosh. Les protocoles de connexion sont soit xml-rpc soit net-rpc.

OpenERP est disponible pour plus de 45 pays, comporte plus de 350 modules, et plus de 700 installations par jour. Sa couverture fonctionnelle est très importante et la vérification de la qualité des modules est effectuée par Tiny sprl. Cela permet aux utilisateurs de savoir quel type de module ils utilisent. Un module peut être soit officiel (développé par Tiny), un module certifié (développé par une société tierce mais validé par Tiny), ou un module issu de la communauté (développé par une société tierce et non

[3]Système de gestions des traductions d'une application

[4]Application s'installant sur l'ordinateur de l'utilisateur voulant se connecter au serveur OpenERP

validé). Ces derniers ne bénéficient pas forcément de support ou de mises à jour. Il faut donc en tenir compte lors de l'utilisation de tel module dans des projets.

Raisons de l'utilisation d'OpenERP

Cela fait 2 ans que je travaille maintenant avec OpenERP et au départ j'ai du choisir un ERP avec 2 collègues de la HEG afin de mettre en place un système de gestion pour l'entreprise de mes parents le Domaine des Molards qui doit gérer la comptabilité financière et analytique, différents produits, des timesheets, des stocks, et les achats et ventes.

Nos critères principaux à ce moment là étaient entre autre chose la couverture fonctionnelle, le dynamisme de la communauté et la pérennité de la société éditrice du logiciel.

Après cette expérience, le développement d'application de bout en bout m'a vraiment semblé dépassé comme principe. Il est bien plus facile de prendre une application existante et de développer les fonctionnalités manquantes. Cela nous donne un gain de temps, d'argent, et un réseau de partenaires nous permettant de collaborer pour faire évoluer notre offre fonctionnelle à moindre coût.

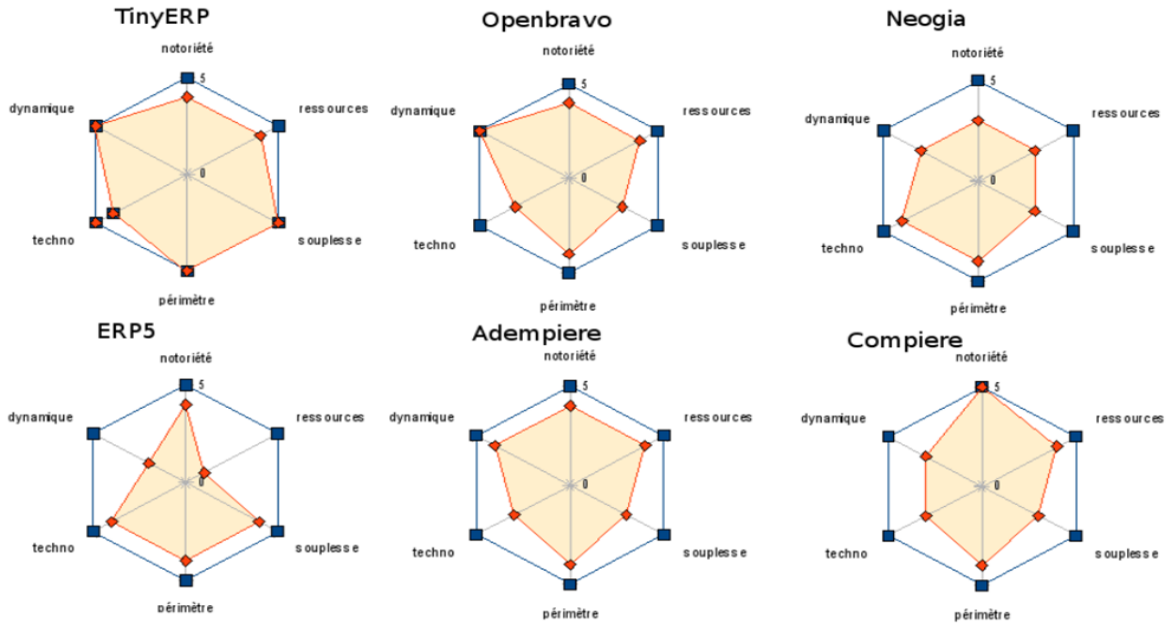
C'est donc fort de cette expérience que j'ai été engagé chez Prolibre pour mettre en place OpenERP qui fonctionnait parfaitement avec tous types de sociétés.

L'utilisation d'OpenERP est donc la base du projet et le choix d'un autre ERP a été étudié mais très vite abandonné.

Comme on peut le remarquer grâce au travail de diplôme de M. Bjarne SORENSEN intitulé ERP Open Source ou Commercial, l'offre fonctionnelle est équivalente voir supérieure pour OpenERP par rapport aux ERPs propriétaires comparés, et sa fiabilité, il y a 2 ans, .était déjà au rendez-vous.

Grâce au livre blanc de la Société Smile[5] intitulé ERP Open Source, on peut remarquer qu'OpenERP est l'ERP qu'ils conseillent. Voici un graphique récapitulatif des ERPs Open Source qu'ils ont comparés :

[5] Livre blanc de la société Smile :
http://openerp.com/images/discover/white_paper_smile.pdf



On peut remarquer qu'OpenERP dénommé encore par son ancien nom TinyERP est très dynamique, très souple et a une très bonne pérennité d'après eux. Il obtient 4 point sur 5 dans les domaines de la notoriété, des ressources et de la technologie utilisée.

Ces différentes sources nous confortent dans notre choix d'ERP et nous pouvons donc affronter la concurrence qu'elle soit propriétaire ou Open Source avec différentes références et études afin de convaincre nos clients.

Avantages et inconvénients

L'inconvénient qui vient à l'esprit quand nous pensons à notre application pour avocat est l'énorme dépendance de ce programme pour la mise en place de notre offre et c'est un fait, mais comme nous disposons de son code source et que la communauté qui est derrière est importante, cet inconvénient n'est pas vraiment un problème car en cas de l'arrêt du développement d'OpenERP, rien ne nous empêcherait pas de continuer à développer notre offre.

Le deuxième inconvénient est l'évolution d'OpenERP qui nous oblige à adapter nos modules et nos personnalisations afin que ceux-ci fonctionnent avec les nouvelles fonctions d'OpenERP. Ceci garantit à nos clients le dynamisme de notre offre et leur garantit de nombreuses évolutions. D'un point de vue client ce n'est donc pas un problème.

Les avantages pour notre offre est que nous bénéficions des nouvelles fonctions d'OpenERP et les corrections de bugs de ce dernier. Nous intégrons de nouvelles fonctionnalités sans avoir besoin de les financer.

Gestion des versions et validation des modules tiers

Le système de gestion des versions permet à l'équipe de Tiny de gérer plusieurs branches officielles et plusieurs branches de contributeurs dans le système de gestion de Launchpad.

Ce système permet à Tiny de coordonner les développements du serveur OpenERP, du client GTK OpenERP, du client Web OpenERP et des différents modules officiels. Il permet aux contributeurs de créer leurs branches pour leurs modules afin de les développer et de les intégrer aux modules d'OpenERP par la suite.

N'importe quel contributeur peut créer sa branche et développer son module afin de la fusionner par la suite dans les branches officielles s'il passe par la phase de validation faite par Tiny ou dans la branche communautaire. Ceci permet d'avoir toutes les contributions dans le même système et de faciliter la modération et la validation des modules par Tiny. De se fait, les modules proposés par l'éditeur sont garantis au niveau qualité du code source et des nouvelles fonctionnalités.

Documentation

La documentation d'OpenERP est faite sous forme de différents wikis. Elle couvre l'utilisation, l'installation, le développement de modules, l'organisation de la communauté, les fonctionnalités, les différents modules et la Business Intelligence[6].

Pour la documentation de notre offre pour avocat, nous pouvons nous appuyer sur la documentation d'utilisation faite pour OpenERP. Nous avons donc moins de documentation à réaliser pour la mise en place de notre offre. Nous avons juste à centraliser la documentation d'OpenERP, la traduire vu qu'elle est uniquement en anglais, et la compléter par nos modifications qui concernent les modifications que nous avons effectuées.

Pérennité

La pérennité est la faculté d'une application à durer dans le temps.

Celle d'OpenERP est mieux garantie que celle d'un logiciel propriétaire car si la société Tiny arrête le développement de ce produit, les partenaires pourraient très bien continuer à le développer étant donné que le code source est disponible.

La pérennité de notre offre est un second aspect à prendre en compte. Si nous arrêtons de la développer, un autre partenaire peut très facilement reprendre l'offre vu que nos contributions sont publiques, donc nos clients auront toujours un moyen de bénéficier de support, ils ne sont liés ni à un éditeur, ni à un intégrateur.

Mise en place technique d'OpenERP

Mettre en place un serveur OpenERP peut paraître simple au premier abord. Il faut installer PostgreSQL pour la base de données, Python et les différentes librairies, puis installer les fichiers du serveur OpenERP, et lancer le serveur. Sous Windows, un Wizard[7] s'occupe de tout installer et en quelques clics, n'importe quel utilisateur peut tester le produit. Sous Linux l'installation est un petit peu plus difficile mais cela reste réalisable grâce aux tutoriels du site doc.openerp.com.

Pour un serveur de test, il n'y a aucun problème de gérer l'infrastructure de cette manière, mais pour des serveurs de production qui doivent être fiable et facilement maintenable, l'infrastructure est légèrement plus complexe

Gestion des versions d'OpenERP

Lors de la réalisation d'un mandat pour un client, tous les modules spéciaux et toutes les modifications des fichiers originaux d'OpenERP sont enregistrés dans un dossier spécifique. De cette façon, en cas de nouvelle version, nous appliquons nos modifications sur les fichiers de base afin de conserver nos spécificités.

[6]Terme anglais désignant l'informatique décisionnelle

[7]Assistant d'installation qui aide l'utilisateur à configurer le programme.

Les modifications sont soit un ajout de fonctionnalité, soit une personnalisation des rapports, ou encore certains champs qui deviennent obligatoires et d'autres qui sont supprimés car inutile au client.

Pour migrer d'une version à l'autre, nous testons d'abord la base de données du client sur un serveur de test puis si tout est fonctionnel, nous faisons la mise à jour du serveur de production du client. Nous appliquons ensuite ses personnalisations et vérifions qu'elles se sont toutes installées sans problème.

Cette gestion manuelle des migrations entre les différentes versions garantit un fonctionnement optimal des nouvelles versions.

Puppet

Puppet est un logiciel qui permet d'installer des services ou des fichiers, de manière distante et centralisée. Il est composé d'un serveur permettant l'administration des clients Puppet installés sur les différents ordinateurs clients.

Ce logiciel a été choisi car il permet de gérer de façon centralisée les installations ou les mises à jour des serveurs OpenERP. Il suffit de remplir un fichier de configuration confectionné par nos soins afin d'installer un nouveau serveur sur une machine locale ou distante. L'installation se déroule par rapport à ce qui a été décidé. Par la suite si une mise à jour est à faire sur les machines, une simple modification du fichier de configuration permet la mise à jour des serveurs contrôlés.

Puppet est également configuré pour lancer des commandes de sauvegardes automatiques des bases de données

Monit

Monit est un logiciel de gestion de service et de monitoring. Il permet de démarrer ou redémarrer les serveurs OpenERP et de surveiller leur bon fonctionnement.

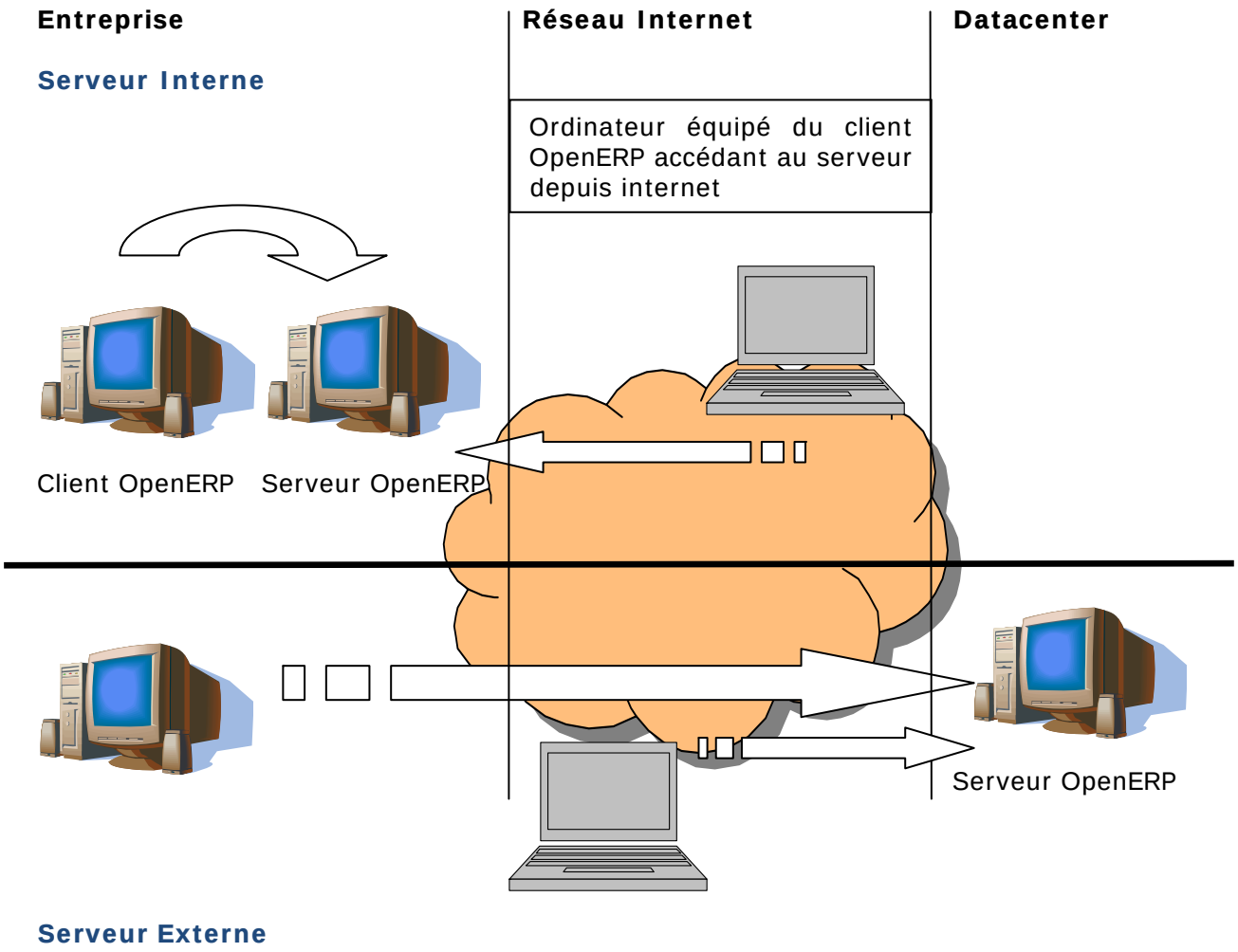
Grâce à ce logiciel nous pouvons connaitre à tout moment l'état de nos serveurs et être averti en cas de dysfonctionnement de l'un d'eux. Nous savons donc en général avant nos clients que leur serveur à des problèmes et remédier à cela dans les délais les plus courts.

Architecture Informatique

Pour le fonctionnement d'OpenERP nous avons un client installé sur chaque poste de travail. Ce client se connecte au serveur qui peut se trouver à l'intérieur ou à l'extérieur de l'entreprise. Sur ce serveur se trouve le client web OpenERP permettant au programme d'être accessible depuis internet, le serveur OpenERP ainsi que la base de données PostgreSQL.

Un firewall est nécessaire pour la sécurité du serveur afin de permettre seulement aux personnes autorisées l'accès à l'application depuis l'extérieur de l'entreprise.

En cas de serveur externalisé, le client OpenERP se trouve sur chaque poste de travail et le Serveur OpenERP ne se trouve plus à l'intérieur de l'entreprise mais dans un Datacenter. Grâce à l'illustration suivante, on peut remarquer les différentes architectures.



Gestion des sauvegardes et reprise d'activité

Garantir une disponibilité de 10 0% des serveurs est quasiment impossible, il faut donc avoir un bon système de backup pour tout remettre en route dans un délai le plus court possible.

Les bases de données sont ainsi sauvegardées quatre fois par jour sur un serveur externe. Si un serveur venait à s'arrêter pour une raison ou une autre, le redémarrage du serveur se fait de façon manuelle afin de vérifier que tout redémarre normalement. Notre temps de reprise d'activité garanti est de 4h durant les jours ouvrables. Des contrats d'intervention dans un délai plus rapide sont possibles pour les clients les plus exigeants.

Dans peu de temps, nous allons mettre en place un deuxième serveur externalisé afin de basculer les serveurs de notre machine située dans nos locaux vers une deuxième machine hébergée à l'extérieur. Grâce à cette nouvelle infrastructure, nous pouvons garantir un taux de disponibilité proche de 100% et une reprise d'activité dans des temps très court.

Pour les serveurs hébergés chez nos clients, nous pouvons également envisager une telle infrastructure. Les coûts varient en fonction du lieu géographique et de la puissance du deuxième serveur.

Avantage d'une telle infrastructure

Installer une telle infrastructure est un gros investissement de départ, car tous ces logiciels obligent les administrateurs à définir très clairement le travail qui doit être fait et

comment configurer chaque partie de l'installation. Pour un petit réseau, le besoin ne s'en fait pas forcément ressentir, mais on se rend vite compte que grâce à des programmes permettant d'automatiser la gestion de l'infrastructure informatique, les administrateurs gagnent très rapidement un temps précieux.

Une fois que le travail est réalisé, il n'y a plus besoin de l'effectuer à nouveau, mais uniquement de renseigner les fichiers de configuration en cas de nouvelle version ou de nouveau client à installer. Le tout est suivi et permet de garder un œil sur les serveurs garantissant leur bon fonctionnement.

Méthodologie d'implémentation d'OpenERP.

Toutes les méthodes de projet disponibles afin d'intégrer un ERP sont basées sur des intégrations de grosses entreprises avec des programmes du type SAP.

Pour les petites intégrations il n'y a pas de méthode toute faite. Il est impossible de faire un cahier des charges figé avec une liste de spécifications à respecter scrupuleusement.

Ceci est le cas parce que la PME ne sait généralement pas ce dont elle a besoin, elle travaille par expérience, sans avoir formalisé ses processus. Une deuxième chose à prendre en compte c'est l'évolution rapide de la façon de travailler d'une PME. Dans 6 mois, elle peut changer de façon de travailler et avoir besoin de nouvelles fonctionnalités, ce qui nous oblige à être rapides dans nos temps de réaction, de développements, et d'implémentation.

La méthode MODEC[8] essaye de résoudre ces problèmes et je m'en suis inspiré en partie pour la réalisation de notre méthodologie d'implémentation.

Définition d'un cahier des charges

Le cahier des charges doit être le plus précis possible afin de cadrer le client et l'inciter à formaliser la façon dont il travaille. Pour ce faire, nous allons rédiger une liste de questions type qui nous permettra de savoir comment le client travaille, quels sont ses processus clés. Nous allons par exemple lui demander ce qu'il fait quand il a une nouvelle affaire, comment il traite son affaire, quels sont les cas spéciaux qui peuvent survenir, et ce qu'il fait quand une affaire est terminée. Nous allons également lui demander comment collaborent les différentes personnes qui travaillent dans un cabinet, qui a le droit de voir quelle information. Toute cette problématique de travail interne et cette gestion des contacts externe doit être prise en compte afin de garantir le succès de la mise en place d'OpenERP.

Méthode utilisée

La méthode utilisée doit être itérative et incrémentale afin de permettre cette souplesse et cette évolutivité. Une bonne méthode pour imaginer celle dont nous aurons besoin est la Roue de Deming. Dans cette méthode de gestion de projet, il faut définir ce qui va être réalisé, ensuite réaliser ce qui a été prévu, puis contrôler si ce qui a été prévu est effectivement réalisé, et finalement corriger et améliorer si besoins ce qui a été développé.

Dans notre méthode, nous allons définir les besoins du client et définir un cahier des charges, puis une offre dans une phase zéro. Une fois cette offre acceptée, la première phase aboutissant à une version prototype comprenant toutes les fenêtres finales du programme est réalisée. De cette façon, le client peut se rendre compte exactement de l'application finale. Ce qui n'est pas développé à cette étape est le traitement qui se

[8] Méthode mise au point avec la Swiss OpenERP Alliance afin de mettre en place OpenERP dans une petite ou moyenne entreprise

trouve derrière ces écrans. Une fois le prototype accepté, il faut réaliser l'ensemble des fonctionnalités et réaliser les traductions manquantes, les valeurs par défaut, tout ce qui améliore l'ergonomie du produit. Dans un troisième temps il faudra vérifier avec le client que l'application fait bien ce qu'il souhaite et ce qui a été défini dans le cahier des charges. Finalement il faut ajuster si besoins l'application afin qu'elle corresponde à ce que souhaite le client.

Pour chaque phase, il faudra définir des rapports à remplir avec le client afin d'avoir sa confirmation par écrit que ce qui va être ou est développé correspond bien à ce qu'il souhaite. De cette façon, nous limitons au maximum les développements inutile ou dans une fausse direction.

Si toutefois cela arrivait, nous étudierions le contexte, au cas par cas, afin de savoir pour quelles raisons le projet n'est pas couronné de succès et définir les mesures à entreprendre pour remédier à ces dysfonctionnements.

Une fois que cette première itération est effectuée, la société peut utiliser le programme qui correspond aux besoins qu'elle a exprimés avant le projet. Si des nouveaux besoins se font sentir, il suffit de refaire une itération avec un nouveau cahier des charges.

De cette façon, l'intégrateur peut personnaliser le produit suivant les besoins du client et évoluer ensemble avec le temps. Si le budget du client est limité en début de projet, celui-ci peut définir ses priorités, et ne choisir que les modules indispensables à son activité dans un premier temps.

Par la suite, il pourra financer l'intégration d'autres modules et ainsi améliorer l'application avec le temps et capitaliser ses investissements.

Sécurité

Quelle entreprise aimerait voir sa comptabilité, ses documents, ses factures, et toutes ses informations publiées sur internet ?

La réponse est : aucune, c'est pour cette raison que la sécurité est le facteur le plus important lors d'une installation d'ERP. Pour OpenERP, la connexion entre le client et le serveur se fait avec le protocole xml-rpc sécurisée et pour la gestion documentaire elle se fait en SFTP (connexion FTP sécurisée). Grâce à cette sécurisation du client au serveur, seule les personnes autorisées peuvent se connecter à l'ERP.

La sécurité de l'ERP se retrouve aussi dans la vitesse de correction des bugs. Tiny est très réactif sur ce point, j'ai par exemple remonté un bug qui concernait la gestion

documentaire et celui-ci a été corrigé en moins de 24 heures[9]. De plus dès que Tiny sprl a été informé d'un bug de sécurité dans la version 5.0.2 ils ont tout de suite sorti la version 5.0.3 une fois cette erreur corrigée. Ils sont donc très réactifs et en tant qu'intégrateur nous pouvons compter sur leur réactivité pour fournir à nos clients une application sûre et fiable.

Définition de la cible

Pour mettre en place un ERP, il y a 3 phases principales à faire avant de proposer une version de démonstration qui correspond au besoin du client.

1. Comprendre le métier du client et définir un cahier des charges afin de valider avec lui quels sont ses **besoins** ; Ensuite il faut chercher quels modules et quelle configuration d'OpenERP nous permettrait de répondre à ses besoins et si un tel module existe ou alors définir **un cahier des charges** pour en réaliser un le cas échéant.

2. Il faut ensuite donner un devis au client afin qu'il connaisse le prix de l'intégration d'OpenERP pour sa société.

3. Une fois le devis accepté, nous commençons à **réaliser** ce qui a été défini dans le cahier des charges. Finalement il faut **valider** que les réalisations correspondent bien à ce qu'attendait le client, le former, et assurer le support en cas de question ou de bugs rencontrés.

Comprendre le métier du client et notre marché cible

Quel est le lien entre une agence de voyage, une entreprise de développement informatique et un commerce exploitant une boutique en ligne basée à Genève ?

La réponse est un fichier client et une comptabilité Suisse. Tout le reste doit être défini précisément et rapidement afin de connaitre l'activité et les processus de la société en question.

Notre but est donc de trouver une profession dans laquelle les méthodes de travail varient peu afin de proposer des modules et une configuration standard nécessitant peu de modifications d'une entreprise à l'autre et un temps d'intégration très court.

Nous avons choisi les cabinets d'avocats car leurs méthodes de travail sont assez similaires d'une Etude à l'autre, parce que c'est une profession avec un pouvoir d'achat assez fort, et que les offres concurrentes ont soit une couverture fonctionnelle plus faible que la notre, ou alors un prix très supérieur. Nous voulons donc proposer une offre intégrée couvrant un maximum de leurs besoins à un prix raisonnable.

Nous allons tout d'abord cibler les petits cabinets d'avocat afin d'avoir des clients qui ont des méthodes de travail simple et ou la communication est plus facile afin de tester notre solution, et une fois que celle-ci sera éprouvée nous allons proposer une offre aux plus grands cabinets d'avocat comprenant la gestion de projets et d'autres options moins nécessaires aux petits cabinets.

[9] Adresse relative au bug en question : https://bugs.launchpad.net/openobject-addons/+bug/409337

Définition du cahier des charges

Dans les cabinets d'avocat d'une dizaine de personnes, il y a en général des secrétaires, des juristes et des avocats. Ils ont des clients privés et des sociétés.

Leur but est de défendre leurs clients contre des parties adverses pour tous les problèmes de la vie courante. Certains sont spécialisés dans certaines activités d'autres non.

Méthodes de travail

Les avocats demandent à leurs clients une provision pour couvrir leurs frais en début d'affaire. Ils peuvent en redemander par la suite si cela est nécessaire.

La facture finale est définie en fonction de plusieurs critères différents d'un cabinet à l'autre comme le temps alloué au dossier, la complexité de l'affaire, l'importance de l'affaire, la situation financière du client et le résultat obtenu.

Pour ce qui est quantifiable, il y a uniquement le nombre d'heures passées sur le dossier. Les autres facteurs influenceront le prix à l'heure demandé par l'avocat à son client. Ces autres facteurs sont par exemple la situation financière du client, la réussite ou non de l'affaire et les retombées de celle-ci par exemple.

Au niveau de la TVA les cabinets d'avocat utilisent les 3 méthodes de taxations en vigueur à Genève. Celles-ci sont intégrées dans OpenERP et notre comptable nous a indiqué qu'il n'y avait aucun problème pour lui de mettre en place l'une ou l'autre de ces méthodes.

Nous proposons également la possibilité de travailler avec notre partenaire, la fiduciaire ComptaCentre sàrl, qui met au service de nos clients son expertise. En effet, grâce à notre offre, il n'est plus nécessaire qu'un comptable re-saisisse les prestations facturées et les charges, les factures clients sont entrées directement et automatiquement dans la comptabilité. Les factures fournisseurs peuvent soit être entrées par une fiduciaire, soit par une secrétaire ou une personne formée à cet effet. Des connaissances comptables ne sont pas requises pour effectuer ce travail.

Gestion des conflits d'intérêts

Les avocats défendent des clients et au fil du temps ils deviennent très proches. De ce fait, il est d'usage dans la profession qu'un cabinet d'avocat ne soit pas partie adverse d'un de ses anciens clients.

Pour un avocat travaillant seul, cette gestion des conflits d'intérêts est relativement simple, mais dès que le cabinet a plusieurs dizaines d'années d'activités ou que plusieurs avocats travaillent ensemble, cette gestion devient tout de suite plus complexe. Les avocats ont donc besoin d'un outil leur indiquant s'il y a des conflits d'intérêts lors de la création d'une nouvelle affaire pour savoir s'il peut l'accepter ou pas. Cette clause doit toute fois pouvoir être outrepassée si l'avocat le souhaite.

Autres fonctionnalités nécessaires

Les avocats ont besoin de gestion documentaire pour gérer les lettres envoyées aux différentes parties d'une affaire et avoir une trace de celles-ci de façon centralisée. Ils ont besoin de la gestion de la comptabilité et celle de la TVA. Ils ont besoin de la gestion des rappels et des fonctionnalités de CRM (Gestion de la relation client).

Choix des modules et configuration

Une fois que nous avons défini une façon standard de travailler qui était la plus commune possible par rapport aux différentes entreprises que nous avons rencontrées, de l'indépendant au cabinet de plusieurs dizaines d'avocats, nous avons regardé la liste des modules que nous avions à disposition parmi la multitudes des modules que comporte OpenERP. Nous avons défini que la liste des modules suivant nous serait indispensable :

- Gestion des partenaires

Elle permet d'avoir la liste des clients, des parties adverse, des fournisseurs, et de tous les contacts d'un cabinet d'avocat.

- Gestion des Timesheets (pointage des heures effectuées par dossier)

Donne la possibilité aux employés d'un cabinet d'avocat de saisir leurs heures dans les différents dossiers afin de savoir le temps investi dans chaque affaire et ainsi savoir s'il faut redemander une provision ou si la provision faite en début d'affaire couvre les frais.

- Gestion de Projet

Dans les plus grosses affaires, quand plusieurs avocats s'occupent d'un dossier, une simple gestion des heures effectuées ne suffit pas. Il faut donc des outils de gestion de projet qui permettent de planifier les différentes tâches d'un projet et leur échéance. Une fois le travail planifié il faut que les différentes personnes indiquent l'avancement de chaque tâche et le temps qu'ils y ont consacré. De cette façon on peut suivre facilement l'avancement de chaque affaire ainsi que le nombre d'heure effectués sur celle-ci.

- Facturation

Il faut la possibilité de faire des factures de provision permettant de couvrir une partie des frais en début de dossier.

Depuis les heures marquées dans les Timesheets ou depuis les tâches des Projets, les cabinets d'avocat vont pouvoir générer des factures automatiquement.

Il faut faire une facture finale tenant compte du ou des provisions demandées et des heures travaillées.

- Comptabilité Suisse

Elle permet depuis les factures de générer les écritures comptables correspondant dans un plan comptable fourni au préalable par le comptable de l'avocat.

- Gestion Documentaire

Permet d'avoir un point central qui regroupe toute la correspondance d'un dossier. Il regroupe donc ses factures, ses lettres, ses mails, toute sa correspondance avec son client.

- CRM

Ce module permet aux avocats de garder une trace directement dans l'ERP de la correspondance e-mail et téléphonique. Il permettrait également de saisir l'état d'esprit des 2 parties, le tout de façon centralisée. Il fournit donc un calendrier avec les dates des différents contacts écrits et des réunions.

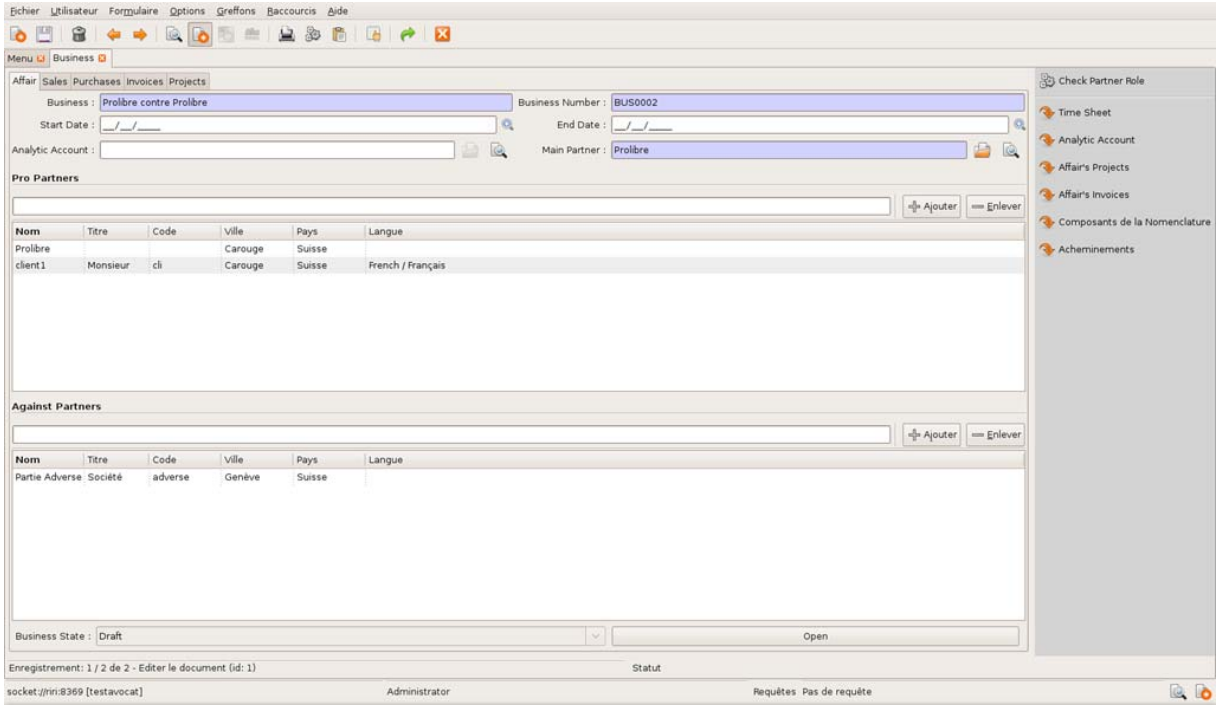
Le module est parfaitement fonctionnel mais le calendrier n'était pas disponible au début de la création du module avocat et il sera donc incorporé à ce dernier si le besoin s'en fait sentir par la suite.

Fonctionnalités manquantes avec les modules de base

Dans OpenERP, les fonctionnalités de gestion par affaire, de gestion des conflits d'intérêts, et quelques améliorations ergonomiques ont été identifiées afin de correspondre au mieux avec les besoins des avocats : elles seront donc développées. Une « verticalisation » sera effectuée afin d'améliorer la vue par affaire.

De base un ERP est organisé en module opérationnel couvrant chacun une fonction bien précise comme la gestion des contacts, la facturation, la relation client ou le timesheet.

Notre but est de regrouper ces différents modules dans une affaire. Nous avons donc une affaire avec ses partenaires, ses projets, ses factures, ses timesheets comme sur l'illustration suivante. Chaque module est situé dans un onglet de l'affaire.



Vue affaire de l'application avocat

Développement des modules nécessaires

Après avoir défini un cahier des charges clair et complet, nous avons discuté avec nos partenaires afin de connaitre leur avis sur notre projet et savoir s'ils avaient déjà fait des mises en place d'OpenERP dans un cabinet d'avocat. Monsieur Dominique Chabord de la société Sisalp basée en Haute Savoie a déjà mis en place OpenERP dans une société française et s'est avéré très intéressé par notre projet. Nous avons décidé de collaborer ensemble pour le développement des modules nécessaire à une bonne intégration.

Nous lui avons donc soumis notre cahier des charges qu'il a étudié et complété par la division de ce qui était spécifique aux avocats et ce qui était spécifique pour une société de gestion d'affaire. Notre cahier des charges était relativement précis car nous avons fait des maquettes d'écran afin que l'entreprise qui devrait réaliser les modules spécifiques sache exactement quoi faire. Nous avons également indiqué quelle fenêtre devait être intégrée dans chaque onglet de la vue par affaire.

Grâce à toutes ces indications, la marge de manœuvre pour l'entreprise qui développera ces modules serait relativement mince.

Choix de l'outsourcing

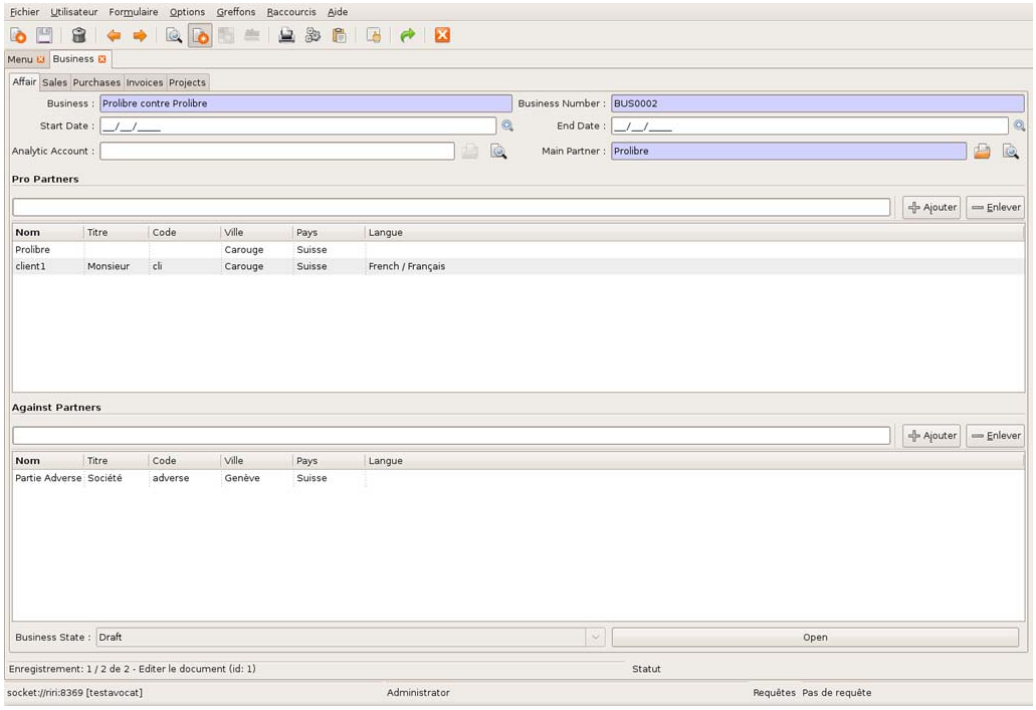
C'est en Inde que nous avons trouvé le partenaire qui pourrait le mieux répondre à nos attentes, et nous avons donc choisi de leur donner le mandat. Nous n'avions jamais travaillé avec eux et la réalisation de ce module était donc une opportunité intéressante afin de voir comment ils travaillaient et savoir si une collaboration serait possible pour le future.

Nous avons rencontré des difficultés sérieuses de communication, pour faire comprendre à la Société indienne ce que nous souhaitions exactement. Le cahier des charges est écrit en anglais et toutes nos correspondances se sont faites dans cette langue, mais parfois un dessin ou un schéma était plus parlant qu'un paragraphe descriptif.

Après l'envoie de notre premier cahier des charges, nous avons rencontré un problème de version d'OpenERP. Lors de sa rédaction, nous utilisions OpenERP en version 4.2. La version 5.0 était alors qu'en version béta et sa date de sortie n'était pas connue. Nous avons donc reçu un 1er module pour la version 4.2 et peu après la version 5.0 est sortie en version finale.

Nous avons donc demandé à la Société Indienne de développer un module pour cette dernière version car le nombre de modifications d'OpenERP était trop important et le module précédemment développé n'était pas fonctionnel avec cette nouvelle version. Ils ont accepté sans augmentation de prix et nous avons reçu peu après un module business qui est une « verticalisation » des modules nécessaire pour le travail par affaire ainsi qu'un module avocat qui est une extension du module affaire comprenant la gestion des conflits d'intérêts regroupant les partenaires que nous défendons et ceux de la partie adverse.

Comme on peut le voir sur l'image suivante, les partenaires défendus sont dans la partie « Pro Partners », et ceux de la partie adverse dans « Against Partners ».



Les avantages à travailler avec les Indiens ont étés nombreux. Tout d'abord le prix intéressant qu'ils nous proposaient. Un autre avantage est que la société Indienne avec laquelle nous avons travaillé connaissait parfaitement OpenERP. Cela nous a permis de bénéficier d'un temps de développement très court et un module qui fonctionne parfaitement pour la version d'OpenERP pour laquelle il a été conçu.

Nous avons bien ressenti les différences de culture au niveau communication car par e-mail, ils comprenaient toujours nos spécifications, mais lors de la réalisation, elles ne correspondaient pas toujours à ce que nous souhaitions. Nous nous sommes également rendu compte qu'ils ne connaissaient pas du tout le métier d'avocat comme nous le connaissions. Il a donc fallu bien spécifier la façon suisse de travailler afin qu'ils comprennent ce que qu'il fallait réaliser.

Les problèmes de communication sont bien réels et n'ont pas lieux uniquement avec les Indiens. Nous nous sommes rendu compte que lors de d'échange d'e-mails ou dans une moindre mesure par téléphone, chaque interlocuteur comprenait la situation en fonction de son point de vue mais chaque point de vue ne coïncide pas toujours. C'est parfois le cas avec monsieur Dominique Chabord quand nous parlons de la façon de gérer la comptabilité, en France il n'est pas autorisé de supprimer une facture, il faut faire une facture d'avoir et refaire une nouvelle facture client, alors qu'en Suisse l'annulation d'une facture ne pause aucun problème.

Quand il s'y ajoute le problème de la langue, ces problèmes s'amplifient encore plus et il faut donc être plus vigilant quant à la compréhension de l'autre afin de prendre du recul et formaliser ses façons de travailler.

La seule façon de bien communiquer est d'être en face de la personne et pouvoir décrypter ses expressions afin de savoir si on est sur la même longueur d'onde ou non. Lors de travail à distance, cette façon de travailler devient difficile, il faut être le plus précis possible et donner des schémas, des captures d'écran et si possible des vidéos afin d'être le plus clair possible.

Le seul point négatif de ce développement est la fonctionnalité de saisie des heures par affaire qui n'a jamais été développée parce qu'elle n'a pas été comprise par l'entreprise Indienne.

Globalement nous pouvons dire que nous sommes satisfaits du travail réalisé par les Indiens car le module livré, mis à part le point négatif précédemment cité, correspond à nos besoins. Il a été développé avec du code de qualité et est pleinement fonctionnel.

La société Open-Net établie dans le canton de Vaud a été mandatée il y a peu par un cabinet d'avocat et elle leur à proposé notre application. Si le client désire l'utiliser, il pourra donc participer au financement des dernières fonctionnalités manquantes.

Modifications réalisées

Pour le développement des vues manquantes, j'ai modifié le code du module afin qu'il contienne les fonctionnalités non développées par les Indiens. J'ai donc changé le fichier xml de définition de la vue ainsi que le fichier Python afin d'ajouter la gestion des timesheets. Pour effectuer cette modification, j'ai appris le langage Python ainsi que l'utilisation du Framework OpenObject. Cet apprentissage a été facilité grâce à la documentation développeur disponible sur le site OpenERP.

J'ai quand même rencontré quelques problèmes car les erreurs données par OpenERP ne sont pas toujours compréhensible. Il faut donc parfois deviner d'où peut venir le problème afin de le résoudre.

Il y a également l'ajout d'une erreur sur un autre module d'OpenERP qui fait planter le programme. On peut modifier une fenêtre ou un objet d'un autre module en redéfinissant un champ par exemple. Cette modification peut introduire une erreur. Une fois cette erreur corrigée, il faut réinstaller le module développé et également le module modifié car sinon l'erreur sur ce module persiste.

Déroulement du projet.

Tout le projet s'est effectué durant mon activité chez Prolibre. J'ai donc bénéficié des contacts de la société afin d'avoir des rendez-vous avec des avocats pour définir leurs besoins spécifiques.

- La rédaction du cahier des charges a duré 3 semaines afin de se renseigner sur les méthodes de travail des avocats, la réalisation des maquettes d'écran, et la rédaction du cahier des charges.
- Le développement des modules s'est réalisé en un peu plus d'une semaine mais le temps entre l'envoi du cahier des charges et la réception du module est d'un peu plus de 2 semaines.
- La correction du module et la réalisation de l'offre commerciale définitive a duré une semaine répartie sur un mois.

Le projet d'application pour avocat a commencé durant le mois d'octobre 2008. Les premiers rendez-vous et la création du cahier des charges se sont faits juste après, ainsi que la création des premiers exemples d'offres commerciales.

Nous avons reçu les premières versions de démonstration durant le début de cette année puis nous avons travaillé pour d'autres mandats OpenERP ce qui nous a obligés à mettre ce projet de côté.

Nous pensons sortir une offre fonctionnelle pour la fin de cette année et nous allons mettre sur notre site internet une version de démonstration et des explications sur notre application avocat.

Redistribution des Modules

Tous les modules que nous avons développés seront partagés grâce au système Launchpad avec la société Tiny et tous les autres partenaires.

Dans un premier temps, on peut penser que n'importe quelle société peut prendre nos modules et les redistribuer. C'est effectivement le cas, mais ces autres partenaires ne sont pas vraiment des concurrents car ils se trouvent sur une zone géographique différente. Si un nouveau partenaire venait à s'installer sur Genève, qui utiliserait nos modules et viserait les mêmes clients que nous, ceux-ci ne seraient pas vraiment une concurrence pour nous, car nous aurions beaucoup d'avance et une plus grande maitrise de la mise en place d'OpenERP pour les avocats.

Avantage concurrentiel

Ce qui fait notre avantage concurrentiel, c'est nos méthodes de travail, notre expertise, notre gestion des serveurs et de l'infrastructure information notamment au niveau de la sécurité, notre support et la formation des utilisateurs. La distribution de ces modules permet également de montrer le savoir faire de notre entreprise et ainsi de consolider une meilleure image de marque auprès de nos clients.

Un partenaire qui ne fait que revendre des modules ne peut pas fournir de telles prestations. Comme la mise en place d'un ERP est quelque chose de complexe au niveau métier plus qu'au niveau technique, cela ne nous pause aucun problème.

Au contraire, si un autre partenaire souhaite développer notre offre pour la revendre, nous allons collaborer et ainsi diviser les couts de développements et une telle collaboration sera bénéfiques pour tout le monde.

Business Model

L'application qui va être réalisée à base d'OpenERP va comprendre la gestion des affaires grâce à une « verticalisation » pour avocat qui va regrouper la gestion des partenaires, la gestion des conflits d'intérêts, la gestion des achats, la gestion du timesheet, la gestion des factures, et la gestion budgétaire de l'affaire, et la gestion documentaire.

Les offres concurrentes

La société CS2I établie en France édite un logiciel pour la gestion des cabinets d'avocat qui est distribué par la société Suisse Geste-Info dont le nom est WinLaw. Ils proposent le suivi de dossier, la gestion électronique de documents, la saisie des prestations, la comptabilité et une intégration à la suite bureautique Office de Microsoft. Aucun prix n'est connu, mais vu le type de base de donnée utilisée (Oracle), plus la gestion de licence, leur prix semble bien supérieur au notre. Cette offre est visiblement dédiée aux grands cabinets d'avocat.

L'offre ForenSys de chez Eyetech offre une solution complète pour la gestion de cabinet d'avocat. Leurs tarifs sont de 490 francs suisse pour l'installation puis de 720 francs suisse par avocat ou notaire. Leur offre ne comporte pas la GED par exemple, ni d'accès depuis un navigateur web.

Le logiciel SmartLex édité par la société Smartway couvre beaucoup de fonctionnalités pour la gestion de cabinet d'avocat. Ils ne proposent pas de GED ni de fonction BVR. Leur prix n'est pas connu.

Nous avons identifié d'autres sociétés éditant des logiciels pour la gestion des cabinets d'avocat mais il y avait parfois très peu d'information sur leur site internet ou alors ils n'étaient plus mis à jour. Nous avons trouvé un autre logiciel open source nommé avocat mais son développement semble au point mort.

Les points forts de notre offre sont une couverture fonctionnelle plus importante que celles de nos concurrents et un accès par navigateur web. Nous sommes donc les seuls à proposer une application sécurisée accessible partout et couvrant autant de fonctionnalités.

Segment de marché

Notre cible est les petits cabinets d'avocat sur la région de Genève, ayant besoin d'un programme afin de gérer leur cabinet. Nous allons apporter aux Petites Entreprises un choix de fonctionnalités qui leur était inabordable jusqu'à présent.

Il y a 1'275 avocats du Barreau de Genève d'après une étude faite en 1996[10] et 554 ont répondu à cette étude. Sur ces réponses 206 sont des chefs d'étude. 26% des sondés sont employés dans une étude de 1 à 3 avocats, et 54% dans une étude de 3 à 10 avocats. Ceci correspond bien à notre cible. Nous avons donc environ 600 avocats comme clients potentiels dans un premier temps, et presque 1300 dans un deuxième temps.

[10]Source : http://www.geneve.ch/tribunaux/doc/enquete_95_96.pdf

Innovation

La part d'innovation de notre projet se trouve dans le fait que nous partons d'un logiciel généraliste pour en faire une déclinaison spécialisée pour un secteur d'activité.

Les gains pour les clients sont nombreux. Tout d'abord le nombre de saisie quand tout est fait à la main peut être très importants. Par exemple l'avocat qui note sur un papier des informations à destination de sa secrétaire, qui les saisi dans une feuille de calcul Excel, qui est saisie ensuite par le comptable dans la comptabilité. Tout ceci disparait avec OpenERP, la secrétaire peut saisir directement l'information au bon endroit.

La centralisation de l'information permet à n'importe quelle personne du cabinet d'avoir accès à tout ce qui concerne les informations d'un client dans un seul endroit centralisé. Les gains en termes de temps de recherche sont donc évidents.

La facturation se fait sur la base des heures encodées par les personnes travaillant sur les affaires, ce qui implique que pour facturer une affaire il suffit d'appuyer sur un bouton et la facture brouillon est disponible, il n'y a donc plus besoin de calculer le nombre d'heures effectuées sur un dossier, ressaisir ces heures dans le programme de facturation, tout est fait automatiquement.

Nos clients pourront donc plus facilement avoir accès aux différentes informations sur leurs affaires et améliorer leur efficacité et leurs méthodes de travail. Ils pourront également économiser sur leurs frais de fonctionnement notamment sur leurs frais de secrétaire et de comptable. Ces personnes auront plus de temps pour faire du travail à plus forte valeur ajoutée comme de l'expertise pour le comptable ou soigner la relation client pour les secrétaires.

Retour sur investissements

Notre offre va devenir intéressante dès que nous aurons plusieurs cabinets d'avocat afin de rentabiliser nos serveurs et valider nos méthodes de travail.

Nous avons passé environ 1 mois afin d'effectuer le cahier des charges, communiquer avec les Indiens, valider leurs développements et tester les modules, et établir l'offre commerciale.

Si nous comptons ce mois de travail plus le montant des développements nous arrivons environ à 20'000 francs. Dans ce montant nous ne comptons pas la mise en place de l'architecture informatique OpenERP avec Puppet et Monit ni l'apprentissage de la configuration d'OpenERP car ces 2 éléments ont déjà été intégrés dans notre offre de service sur OpenERP. Afin de rentrer dans nos frais, il nous faudra environ une dizaine de clients.

Ce nombre est approximatif car il dépend du nombre de modules souhaités par nos clients. Nous allons proposer une offre comprenant les modules de base comme la comptabilité, le timesheet, la gestion des partenaires et des conflits d'intérêts et la facturation. Les modules de gestion documentaire, de gestion de la relation client et de gestion de projet seront proposés en supplément. Les personnalisations comprises dans l'offre de base sont celles qui concernent les rapports afin de les adapter au client. Si des modifications d'interface ou de workflow devraient être envisagées, elles seraient facturées en plus également.

Si nous avons de gros clients en début d'activité, notre offre sera ainsi plus rapidement rentabilisée. Notre prix de base sera aux alentours de 5'000chf. C'est grâce à l'étude faite

lors de la réalisation de cette application pour la profession d'avocat que nous allons pouvoir mettre en place rapidement OpenERP. Tout ce qui correspond aux spécificités de chaque étude sera facturé sur devis sur la base du nombre d'heures passées à faire ces modifications. De ce fait, nous ne prenons aucuns risques si nos clients veulent apporter des modifications à leur version du programme.

Une fois les premières installations effectuées, nous aurons des références et des utilisateurs qui bénéficieront quotidiennement de nos produits et pourront nous donner un retour réel sur les avantages qu'ils tirent de notre solution.

Futures développements et évolution

Dès que nous aurons quelques clients qui auront intégré notre application avocat, nous allons intégrer les fonctionnalités de CRM ou d'autres fonctionnalités souhaitées par nos clients.

La fonction de CRM permettra aux utilisateurs d'avoir une vue calendrier avec sous chaque jour les tâches notées dans les timesheets, les rendez-vous de l'affaire, les jours ou des factures ont été réalisées, en un mot toutes les actions réalisées pour une affaire donnée.

Le module actuel d'OpenERP concernant la gestion documentaire n'était pas assez mature avant le mois d'août 2009, donc nous ne pouvions pas le présenter jusqu'à présent. Il est actuellement opérationnel à 100% et nous allons l'inclure dans notre offre sous peu.

Co-Développements

En cas de nouvelle fonction souhaitée par un de nos clients, nous allons mettre en place un co-développement. Cela signifie que nous allons regarder si cette nouveauté intéresse un ou plusieurs autres clients et leur demander de définir leurs besoins, et nous leur proposons de financer les développements à prix préférentiel, et au final nous pourront revendre cette fonctionnalité et augmenter notre offre. Grâce à cette façon de travailler, nos clients auront les fonctionnalités qu'ils souhaitent à moindre coût et pour nous c'est une augmentation de notre offre sans avoir besoin de faire des investissements supplémentaires.

Élargissement de l'offre avocat à d'autres professions libérales

Une fois que l'offre pour avocat sera mise en place, nous avons comme projet de créer la même offre pour différentes professions libérales.

Dans nos développements, nous avons isolé ce qui était commun aux professions libérales dans un module nommé business, et ce qui ne concerne que les avocats dans un module nommé avocat. De cette façon, pour l'installation de nos personnalisations pour un avocat, nous installons d'abord le module business, et ensuite le module avocat. Nous voulons effectuer la même opération pour chaque profession pour laquelle nous allons réaliser une offre.

De cette façon, les modifications apportées au module business seront générales et se répercuteront sur toutes les autres offres, ce qui nous permettra de bénéficier de la facilité pleinement de personnalisation de l'ERP.

Offre de Serveur 4 en 1

Prolibre propose une offre de Serveur 4 en 1 regroupant dans un serveur physique un serveur OpenERP, un serveur de fichier, un serveur de gestion de courriers, et un serveur d'applications web comme un site internet ou intranet, un ou des blogs ou wikis suivant les besoins des utilisateurs.

Notre offre pour avocat pourra donc très bien fonctionner sur une telle plateforme ce qui permettrait à Prolibre de vendre l'ensemble de ses activités à la société cliente et pour eux de bénéficier d'un seul interlocuteur en cas de question informatique.

Nous étudions actuellement la possibilité de décliner cette offre de serveur 4 en 1 soit sur une machine chez notre client, soit dans un Datacenter accessible depuis internet. Cette deuxième formule permettra au client de ne plus avoir de soucis technique et de bénéficier d'une machine plus performante à moindre coût.

Pour Prolibre, la gestion d'infrastructure hébergée est une nouvelle offre qui répond à un besoin des clients qui se fait de plus en plus ressentir, car bénéficier d'un serveur hébergé en interne avec une architecture garantissant une haute disponibilité est très onéreux. Il faut par exemple mettre deux machines en plus de doubler les disques durs et mettre un système d'onduleur avec batterie afin d'éteindre le serveur en cas de coupure de courant en toute sécurité.

Tout ces problèmes disparaissent avec un serveur hébergé, car il est géré par l'hébergeur, mais de nouveaux surviennent comme par exemple la vitesse de la connexion internet qui devient, car plus il y a d'utilisateurs, plus la connexion doit être rapide.

Il y a également la disponibilité de la connexion qui doit être prise en compte car il faudra peut être prévoir une deuxième ligne afin de continuer à travailler si une connexion était coupée. Il faut donc gérer cette nouvelle problématique avec le client afin de lui fournir l'offre qui correspond le mieux à ces besoins.

Glossaire

Apache : « Apache HTTP Server » est un logiciel de serveur HTTP produit par l'Apache Software Foundation.

Bug : Erreur dans le programme provoquant l'arrêt de celui-ci.

CRM : « Customer Relationship Management » Gestion de la relation client.

Datacenter : Centre informatique de plusieurs serveurs

FireFox : Navigateur Internet de la fondation Mozilla.

Fondation Mozilla : Organisme à but non lucratif établi pour gérer le développement et assurer la publicité des logiciels libres issus de la suite Mozilla.

Framework : Ensemble d'outils et de conventions facilitant de développement d'application informatique.

FTP : « File Transfer Protocol » Protocole de transfert de fichier à travers un réseau informatique.

GREP : travail de groupe réalisé en fin d'étude à la Haute Ecole de Gestion de Genève.

GPL : « Licence publique générale GNU » abrégée GPL donne la liberté d'utiliser, d'étudier, de modifier et de diffuser le logiciel et ses versions dérivées.

Launchpad : Application web qui procure une aide au développement de logiciels comme la gestion des versions, des bugs, des traductions, etc. Elle a été développée par la société Canonical.

MySQL : Serveur de base de données relationnelle libre ou propriétaire très utilisé pour les sites internet développé par Sun qui a été racheté par Oracle dernièrement.

NET-RPC : Protocole de communication à travers le réseau utilisé entre le client et le serveur OpenERP.

OpenOffice : Suite bureautique libre.

Python : Langage de programmation orienté objet sous licence libre.

PostgreSQL : Serveur de base de données relationnelle et objet sous licence libre.

SAP : « Systems, Applications, and Products for data processing » en anglais est un progiciel de gestion intégré développé et commercialisé par l'éditeur de ce produit (SAP AG).

Timesheet : On peut traduire ce terme anglais par feuille de temps. C'est en résumé une liste des tâches réalisées.

Wiki : Système de gestion de contenu collaboratif pour site internet.

Workflow : Modélisation d'un flux de travail définissant les différentes étapes pour réaliser un processus.

XML-RPC : Protocole de communication à travers le réseau utilisé entre le client et le serveur OpenERP.

Bibliographie

Jean-Louis Tomas et Gérard Balantzian, *ERP et PGI - Sélection, méthodologie de déploiement et gestion du changement*, 2007, Dunod/01 Informatique, ISBN-13: 978-2100513734

Fabien Pinckaers et Geoff Gardiner, Tiny *ERP-Open ERP : Pour une gestion d'entreprise efficace et intégrée*, 2 mai 2008, Eyrolles, ISBN-13: 978-2212122619

Tiny sprl, *Site internet d'OpenERP*, http://openERP.com/

Progilibre, Communauté francophone des applications d'entreprise en open source, http://www.prolibre.com/

Tiny sprl, *Sortie de la version 5.0 de OpenERP*, 10 février 2009 http://www.progilibre.com/Sortie-de-la-version-5-0-de-Open-ERP_a778.html

David Larlet, De Windows à Ubuntu : raisons d'un succès, 27 février 2005 http://www.biologeek.com/logiciels-libres,ubuntu/windows-ubuntu-raisons-succes/

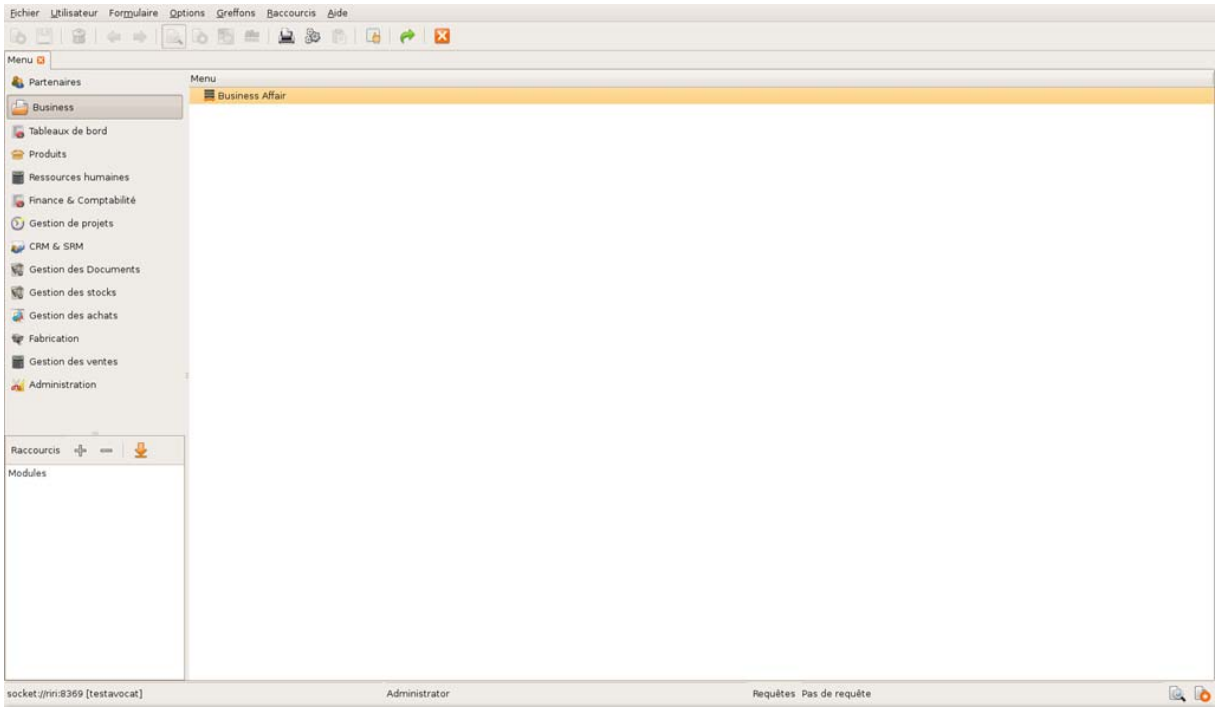
Bjarne SORENSEN, *ERP Open Source ou Commercial*, HEG Genève, 07.12.2007 http://campus.hesge.ch/Daehne/TD/2007/SorensenBjarne.pdf

Raphaël Valyi, *Livre blanc - ERP Open Source*, Smile http://openerp.com/images/discover/white_paper_smile.pdf

Annexe

Captures d'écran montrant l'application finale

Voici quelques captures d'écran afin de mieux visualiser l'application.



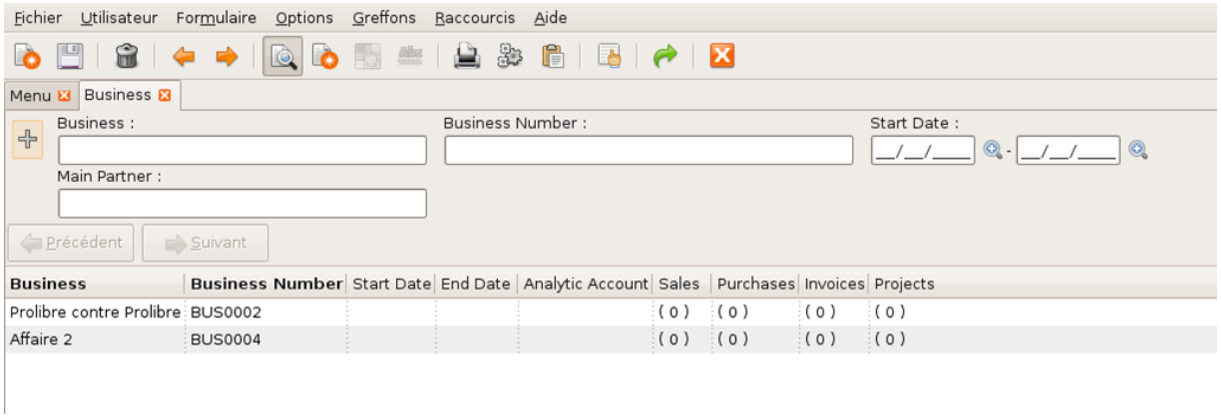
Voici l'écran d'accueil lors de la connexion à OpenERP. On peut voir la liste des différents modules sur la gauche. Ces modules concernant entre autre les partenaires pour la gestion des contacts, le module Business relatif à la gestion des affaires, la comptabilité, la gestion de projet, la gestion des documents ainsi que tous les fonctionnalités disponibles.

Cette liste de module peut être réduite suivant les besoins de chaque utilisateur afin de faciliter l'utilisation de l'application.

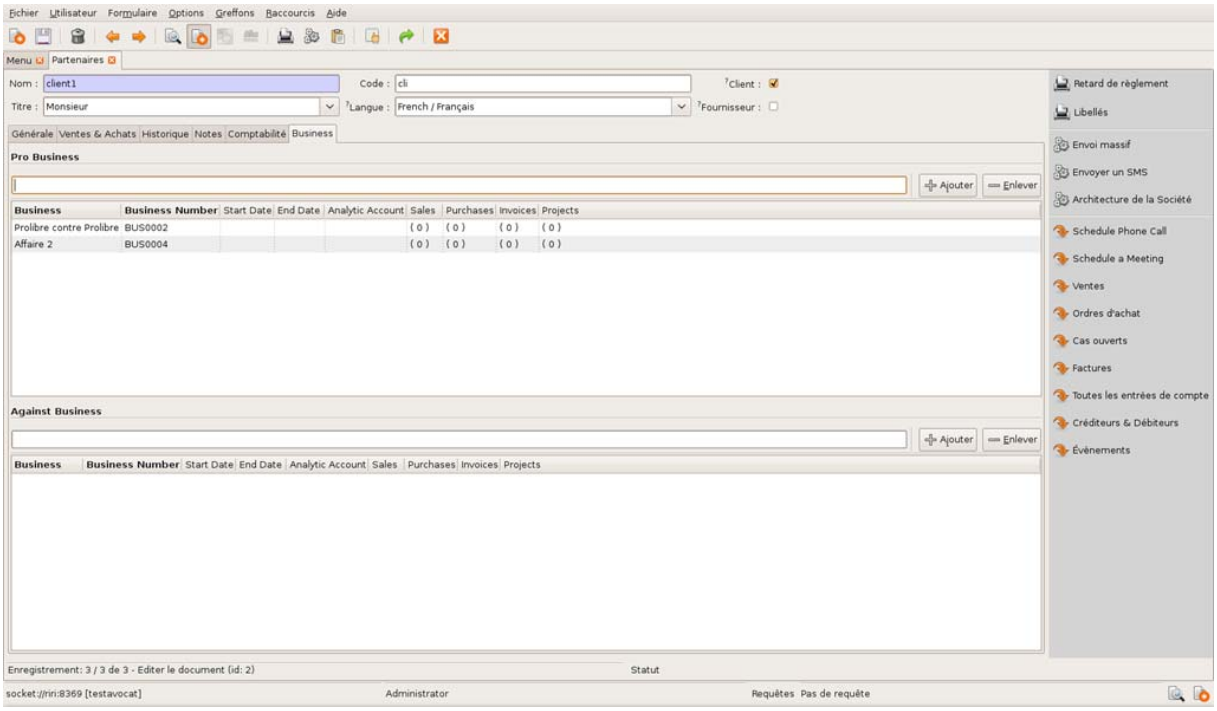
La partie du bas représente les raccourcis utilisateurs permettant l'accès rapide à un sous menu.

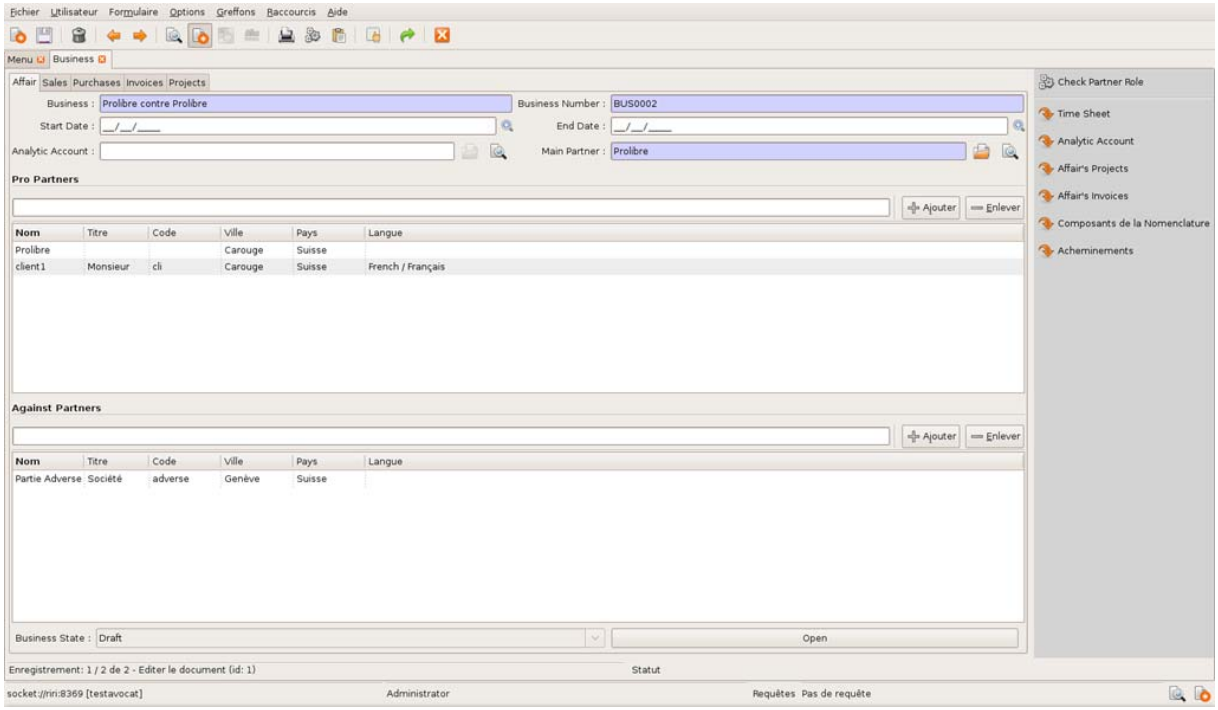
Le menu du haut est commun à toutes les fenêtres et permet de gérer les différentes fenêtres afin de faire un nouvel enregistrement, changer de type de vue, imprimer un document, fermer une fenêtre, etc.

Dans le module Business il y a la liste des affaires disponibles. On peut consulter son numéro, ses dates de début et de fin, ainsi que le nombre de factures, de projets, etc.



L'illustration suivante montre l'onglet Affaire qui a été ajouté dans la fiche de chaque partenaire. Grâce à lui on peut savoir dans quelles affaires notre client a été défendu et dans quelles affaires il a été partie adverse.



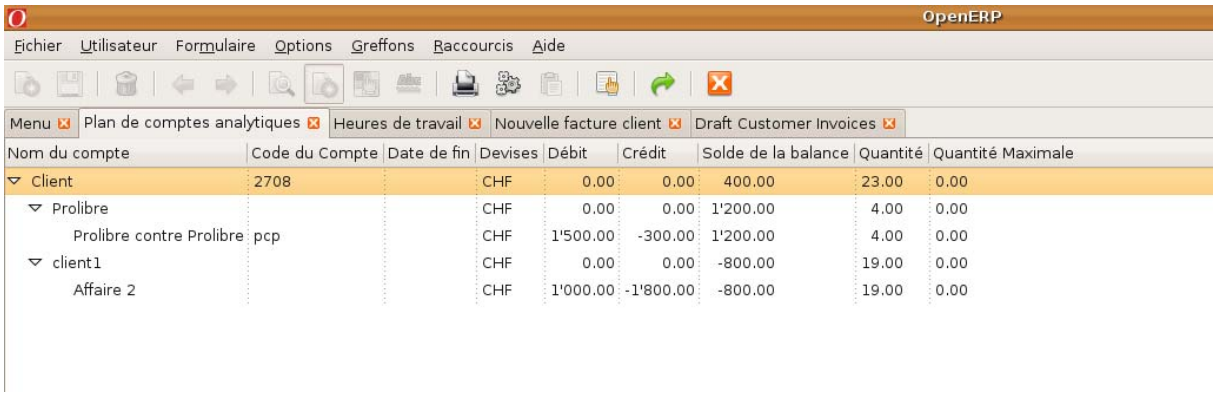


Cette dernière illustration permet d'avoir le détail d'une affaire. Ce premier onglet représente la gestion des contacts de l'affaire avec dans Main Partner le contact principal de l'affaire qui servira pour la facturation de cette dernière.

Les « Pro Partners » sont les clients que nous défendons. Les partenaires dans « Against Partners » sont les parties adverses lors d'une affaire.

Dans les différents onglets on peut consulter les devis, les achats, et les factures de l'affaire.

Ce module est en cours de traduction, c'est pourquoi les champs sont encore en anglais.



Nom du compte	Code du Compte	Date de fin	Devises	Débit	Crédit	Solde de la balance	Quantité	Quantité Maximale
Client	2708		CHF	0.00	0.00	400.00	23.00	0.00
Prolibre			CHF	0.00	0.00	1'200.00	4.00	0.00
Prolibre contre Prolibre	pcp		CHF	1'500.00	-300.00	1'200.00	4.00	0.00
client1			CHF	0.00	0.00	-800.00	19.00	0.00
Affaire 2			CHF	1'000.00	-1'800.00	-800.00	19.00	0.00

Le compte analytique est créé automatiquement si aucun compte n'est sélectionné sous la forme de l'arbre suivant :

Client / Nom du Main Partner / Affaire

Comme on le voit, en un coup d'œil nous avons le détail de chaque affaire ainsi que le détail total pour chaque client ainsi que la quantité d'heures effectuées pour chaque niveau.

Extrait de code du module développé

Voici les 2 fichiers que j'ai modifié afin d'intégré la gestion des timesheets dans notre module Business. Pour rappel, le module business est le module général correspondant au travail par affaire réalisé par les professions libérales, et le module avocat les spécificités de ce module pour la profession d'avocat.

Les modifications sont réalisées en bleu. La difficulté pour les réalisé n'est pas au niveau du code à écrire mais plus de trouver quel module est concerné par la modification, de quel type est-il, et à quel endroit l'intégré.

Fichier xml de la vue du module Business

```
<?xml version="1.0" encoding="utf-8"?>
<openerp>
  <data>
<!--       order view -->
    <record model="ir.ui.view" id="view_order_tree_business">
      <field name="name">sale.order.tree</field>
      <field name="model">sale.order</field>
      <field name="inherit_id" ref="sale.view_order_tree" />
      <field name="type">tree</field>
      <field name="arch" type="xml">
      <field name="partner_id" position="before" >
         <field name="business_id"/>
         </field>
      </field>
    </record>

    <record model="ir.ui.view" id="view_order_form_business">
      <field name="name">sale.order.form</field>
      <field name="model">sale.order</field>
      <field name="inherit_id" ref="sale.view_order_form" />
      <field name="type">form</field>
      <field name="arch" type="xml">
        <field name="project_id" position="before" >
           <field name="business_id" select="1"
on_change="onchange_business_id(business_id)"/>
         </field>
      </field>
    </record>

<!--    purchase view -->
    <record model="ir.ui.view" id="purchase_order_form_business">
      <field name="name">purchase.order.form</field>
      <field name="model">purchase.order</field>
      <field name="inherit_id" ref="purchase.purchase_order_form"/>
      <field name="type">form</field>
      <field name="arch" type="xml">
         <field name="partner_ref" position="after" >
           <field name="business_id" select="1" />
         </field>
      </field>
    </record>

    <record model="ir.ui.view" id="purchase_order_form_business2">
      <field name="name">purchase.order.form</field>
      <field name="model">purchase.order</field>
      <field name="inherit_id" ref="purchase_order_form_business"/>
```

```
    <field name="type">form</field>
    <field name="arch" type="xml">
       <field name="order_line" position="replace" >
        <field name="order_line" widget="one2many_list" colspan="4" nolabel="1"
context="business_id=business_id"/>
       </field>
    </field>
  </record>

  <record model="ir.ui.view" id="purchase_order_tree_business">
    <field name="name">purchase.order.tree</field>
    <field name="model">purchase.order</field>
    <field name="inherit_id" ref="purchase.purchase_order_tree"/>
    <field name="type">tree</field>
    <field name="arch" type="xml">
       <field name="date_order" position="after" >
        <field name="business_id" select="1"/>
       </field>
    </field>
  </record>

  <record model="ir.ui.view" id="purchase_order_line_form">
     <field name="name">purchase.order.line.form</field>
     <field name="model">purchase.order.line</field>
     <field name="type">form</field>
     <field name="inherit_id" ref="purchase.purchase_order_line_form" />
     <field name="arch" type="xml">
        <field name="account_analytic_id" position="replace" >
           <field
              name="account_analytic_id" colspan="3"
              context="business_id = parent.business_id,"/>
        </field>
     </field>
  </record>

<!--   invoice view  -->
  <record model="ir.ui.view" id="invoice_tree_business">
    <field name="name">account.invoice.tree</field>
    <field name="model">account.invoice</field>
    <field name="inherit_id" ref="account.invoice_tree"/>
    <field name="type">tree</field>
    <field name="arch" type="xml">
       <field name="number" position="after" >
        <field name="business_id" select="1"/>
       </field>
    </field>
  </record>

  <record model="ir.ui.view" id="invoice_form_business">
    <field name="name">account.invoice.form</field>
    <field name="model">account.invoice</field>
    <field name="inherit_id" ref="account.invoice_form"/>
    <field name="type">form</field>
    <field name="arch" type="xml">
       <field name="currency_id" position="after" >
        <field name="business_id" select="1"/>
       </field>
    </field>
```

```
    </record>

<!--    Project view   -->
    <record model="ir.ui.view" id="edit_project_business">
      <field name="name">project.project.form</field>
      <field name="model">project.project</field>
      <field name="inherit_id" ref="project.edit_project"/>
      <field name="type">form</field>
      <field name="arch" type="xml">
         <field name="state" position="after" >
           <field name="business_id" select="1"
on_change="onchange_business_id(business_id)"/>
         </field>
      </field>
    </record>

    <record model="ir.ui.view" id="view_project_business">
      <field name="name">project.project.tree</field>
      <field name="model">project.project</field>
      <field name="inherit_id" ref="project.view_project"/>
      <field name="type">tree</field>
      <field name="field_parent">child_id</field>
      <field name="arch" type="xml">
         <field name="manager" position="after" >
           <field name="business_id" select="1"/>
         </field>
      </field>
    </record>

<!--    Business affair view  -->
    <record model="ir.ui.view" id="business_form">
      <field name="name">business.business.form</field>
      <field name="model">business.business</field>
      <field name="type">form</field>
      <field name="arch" type="xml">
         <form string="Business">
         <notebook>
            <page string="Affair">
            <field name="name" select="1"/>
            <field name="code" select="1"/>
            <field name="date_start" select="1"/>
            <field name="date_end" select="1"/>
            <field name="analytic_account" select="2"/>
            <field name="main_partner" select="1" required="1"/>
            </page>
          <page string="Sales">
            <field name="sales" select="2" nolabel="1"/>
          </page>
          <page string="Purchases">
            <field name="purchases" select="2" nolabel="1"/>
          </page>
          <page string="Invoices">
            <field name="invoices" select="2" nolabel="1"/>
          </page>
          <page string="Projects">
            <field name="projects" select="2" nolabel="1"/>
            </page>
          <page string="TimeSheet">
```

```
          <field name="timesheet" select="2" nolabel="1"/>
          </page>
<!--      <page string="Partners">
            <field name="list_partners" select="2" domain="[('business_id', '=',
business_id)]">
              <tree string="Partners">
                 <field name="partner_id"/>
            </tree>
          </field>
         </page>           -->
       </notebook>
       </form>
    </field>
   </record>

   <record model="ir.ui.view" id="business_tree">
     <field name="name">business.business.tree</field>
     <field name="model">business.business</field>
     <field name="type">tree</field>
     <field name="arch" type="xml">
        <tree string="Business">
           <field name="name" select="1"/>
           <field name="code" select="1"/>
           <field name="date_start" select="1"/>
           <field name="date_end" select="1"/>
           <field name="analytic_account" select="2"/>
<!--       <field name="list_partners" select="2"/>    -->
           <field name="sales" select="2"/>
           <field name="purchases" select="2"/>
           <field name="invoices" select="2"/>
           <field name="projects" select="2"/>
           <field name="timesheet" select="2"/>
        </tree>
     </field>
   </record>


   <record model="ir.actions.act_window" id="business_form_action">
     <field name="type">ir.actions.act_window</field>
     <field name="res_model">business.business</field>
     <field name="view_type">form</field>
     <field name="view_mode">tree,form</field>
     <field name="view_id" ref="business_tree"/>
   </record>

<!-- menu view -->
  <menuitem name="Business"  id="menu_business_root" sequence="1"/>

  <menuitem name="Business/Business Affair" id="menu_business_form_action"
action="business_form_action"/>

<!-- list of views -->
  <act_window name="Affair's Orders "
    domain="[('business_id', '=', active_id)]"
    res_model="sale.order"
    src_model="business.business"
    view_type="form"
    view_mode="tree,form"
```

```
    id="view_order_tree_business"/>

  <act_window name="Affair's Purchases "
    domain="[('business_id', '=', active_id)]"
    res_model="purchase.order"
    src_model="business.business"
    view_type="form"
    view_mode="tree,form"
    id="purchase_order_tree_business"/>

  <act_window name="Affair's Invoices "
    domain="[('business_id', '=', active_id)]"
    res_model="account.invoice"
    src_model="business.business"
    view_type="form"
    view_mode="tree,form"
    id="invoice_tree"/>

  <act_window name="Affair's Projects "
    domain="[('business_id', '=', active_id)]"
    res_model="project.project"
    src_model="business.business"
    view_type="form"
    view_mode="tree,form"
    id="view_project"/>

  <act_window name="Analytic Account "
    domain="[('id', '=', analytic_account)]"
    res_model="account.analytic.account"
    src_model="business.business"
    view_type="form"
    view_mode="tree,form"
    id="hr_timesheet.account_analytic_account_form_form"/>

    <record id="business.hr_timesheet_sheet_form" model="ir.ui.view">
          <field name="name">hr.timesheet.sheet.form</field>
          <field name="model">hr_timesheet_sheet.sheet</field>
          <field name="type">form</field>
          <field name="inherit_id" ref="hr_timesheet_sheet.hr_timesheet_sheet_form"
/>
          <field name="arch" type="xml">
             <field name="name" position="after">
                <field name="business_id" />
             </field>
          </field>
    </record>


  <act_window name="Time Sheet"
    domain="[('business_id', '=', active_id)]"
    res_model="hr_timesheet_sheet.sheet"
    src_model="business.business"
    view_type="form"
    view_mode="tree,form"
    id="afair_time_sheet"/>

    <record id="business_client_analytic_account" model="account.analytic.account">
       <field name="name">Client</field>
```

```
    </record>

 </data>
</openerp>
```

Fichier python du module Business

```
#!/usr/bin/env python
#coding: utf-8

# (c) 2009 Prolibre
# author : desbaillet@prolibre.com

# This program is free software; you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation; either version 2 of the License, or
# (at your option) any later version.
#
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with this program; if not, write to the Free Software
# Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.

from osv import osv, fields
import tools
from mx import DateTime
import netsvc

def _default_purchase_analytic( obj, cr, uid, context):
    #print '_default_purchase_analytic',context
    business_id = context.get( 'business_id', False )
    if business_id:
        res = obj.pool.get( 'business.business' ).browse( cr, uid, business_id, context
).analytic_account.id
    else:
        res = False
    #purchase_order_obj = obj.pool.get('purchase.order').browse( cr, uid, context )

    #print purchase_order_obj
    return res

class business_business(osv.osv):
  _name = "business.business"
  _description = "business affair"

  def make_list_partners(self, cr, uid, ids,business, arg, context):
        for busi in self.browse(cr, uid, ids):
                business_id = busi.name
        res1=[]
        if business_id:
                crit = [('business_id','=',business_id)]
                res2 = self.pool.get('sale.order').search(cr, uid, crit)
                for sale in self.pool.get('sale.order').browse(cr, uid, res2):
                        res1.append(sale.partner_id.id)
                res3 = self.pool.get('purchase.order').search(cr, uid, crit)
                for purchase in self.pool.get('purchase.order').browse(cr, uid, res3):
                        res1.append(purchase.partner_id.id)
                res = unique(res1)
                print res
```

```
        else:
                return False
        return res

  _columns = {
        'name' : fields.char('Business', size=64, required=True),
        'code' : fields.char('Business Number', size=64, required=True),
        'date_start': fields.date('Start Date'),
        'date_end': fields.date('End Date'),
        'analytic_account' : fields.many2one('account.analytic.account', 'Analytic
Account',required=False),
        #'list_partners' : fields.one2many('partner.from.list', 'business_id', 'Sale',
readonly=True),
        'sales' : fields.one2many('sale.order', 'business_id', 'Sales'),
        'purchases' : fields.one2many('purchase.order', 'business_id', 'Purchases'),
        'invoices' :fields.one2many('account.invoice', 'business_id', 'Invoices'),
        'projects' : fields.one2many('project.project', 'business_id', 'Projects'),
        'timesheet' : fields.one2many('hr.analytic.timesheet', 'account_id', 'Time Sheet')
    'main_partner':fields.many2one('res.partner','Main Partner',required=True),
        }
  _defaults={
       'code': lambda obj, cr, uid, context: obj.pool.get('ir.sequence').get(cr, uid,
'business.business'),
   }
  def create(self, cr, user, vals, context=None):
      new_id = super(business_business,self).create(cr, user, vals, context)
      if (new_id):
         client_analytic_id =
self.pool.get('ir.model.data')._get_id(cr,user,'business','business_client_analytic_account')
         res_id = self.pool.get('ir.model.data').browse(cr,user,client_analytic_id).res_id
         main_partner_name =
self.pool.get('res.partner').browse(cr,user,vals['main_partner']).name
         parent_analytic_id =
self.pool.get('account.analytic.account').search(cr,user,[('name','=',main_partner_name),
('parent_id','=',res_id)])
         if not (parent_analytic_id):

parent_analytic_id=[self.pool.get('account.analytic.account').create(cr,user,{'name':main
_partner_name,'parent_id':res_id})]

self.pool.get('account.analytic.account').create(cr,user,{'name':vals['name'],'parent_id':p
arent_analytic_id[0]})
      return new_id
   #end def create(self, cr, user, vals, context=None):
business_business()

class sale_order(osv.osv):
  _name = "sale.order"
  _inherit = "sale.order"
  _description = "Sale order with business affair"

  _columns = {
        'business_id' : fields.many2one('business.business','Business', required=False ,
readonly=True, states={'draft':[('readonly',False)]}),
        }

  def onchange_business_id(self, cr, uid, ids, business_id):
        if not business_id:
```

```
                return {}
        analytic = self.pool.get('business.business').browse(cr, uid,
business_id).analytic_account.id
        return {'value':{'project_id': analytic}}

  def action_invoice_create(self, cr, uid, ids, grouped=False, states=['confirmed','done']):
                res = super(sale_order, self).action_invoice_create(cr, uid, ids, grouped,
states)
                for o in self.browse(cr, uid, ids):
                        inv = self.pool.get('account.invoice').write(cr, uid, [res],
{'business_id': o.business_id.id})
                return res
sale_order()

class purchase_order(osv.osv):
  _name = "purchase.order"
  _inherit = "purchase.order"
  _description = "Purchase order with business affair"

  _columns = {
        'business_id' : fields.many2one('business.business','Business', required=False,
readonly=True, states={'draft':[('readonly',False)]}),
        }


  def action_invoice_create(self, cr, uid, ids, *args):
                res = super(purchase_order, self).action_invoice_create(cr, uid, ids, *args)

                for o in self.browse(cr, uid, ids):
                        inv = self.pool.get('account.invoice').write(cr, uid, [res],
{'business_id': o.business_id.id})
                return res
purchase_order()

class purchase_order_line(osv.osv):
  _name = "purchase.order.line"
  _inherit = "purchase.order.line"
  _description = "purchase order line with business affair"

  _defaults = {
        'account_analytic_id' : lambda *args : _default_purchase_analytic( *args ) ,
      }
purchase_order_line()

class account_invoice(osv.osv):
  _name = "account.invoice"
  _inherit = "account.invoice"
  _description = "invoice with business affair"

  _columns = {
        'business_id' : fields.many2one('business.business','Business', required=False,
readonly=True, states={'draft':[('readonly',False)]}),
        }

account_invoice()
```

```
class project(osv.osv):
  _name = "project.project"
  _inherit="project.project"
  _description = "Project with business affair"

  _columns = {
        'business_id' : fields.many2one('business.business','Business', required=False),
        'category_id': fields.many2one('account.analytic.account','Analytic Account'),
        }

  def onchange_business_id(self, cr, uid, ids, business_id):
        if not business_id:
                return {}
        analytic = self.pool.get('business.business').browse(cr, uid,
business_id).analytic_account.id
        return {'value':{'category_id': analytic}}

project()

class partner_from_sale(osv.osv):
        _name = "partner.from.sale"
        _description = "Partners from business sales"
        _order='partner_id'
        _auto = False
        _columns = {
                'name': fields.one2many('sale.order', 'business_id', 'Sale Order',
readonly=True),
                'business_id': fields.many2one('business.business','Business'),
                'partner_id' : fields.many2one('res.partner','Partner'),
        }
        def init(self, cr):
                cr.execute("""
                        create or replace view partner_from_sale as (
                                select
                                        id, partner_id,business_id,name
                                                from
                                                        sale_order
                                                )""")
partner_from_sale()

class partner_from_purchase(osv.osv):
        _name = "partner.from.purchase"
        _description = "Partners from business purchases"
        _order='partner_id'
        _auto = False
        _columns = {
                'name': fields.one2many('purchase.order', 'business_id', 'Purchase Order',
readonly=True),
                'business_id': fields.many2one('business.business','Business'),
                'partner_id' : fields.many2one('res.partner','Partner'),
        }
        def init(self, cr):
                cr.execute("""
                        create or replace view partner_from_purchase as (
                                select
                                        id, partner_id,business_id,name
                                                from
                                                        purchase_order
```

```
                                        )""")
partner_from_purchase()

class partner_from_invoice(osv.osv):
        _name = "partner.from.invoice"
        _description = "Partners from business invoices"
        _order='partner_id'
        _auto = False
        _columns = {
                'name': fields.one2many('account_invoice', 'business_id', 'Purchase Order',
readonly=True),
                'business_id': fields.many2one('business.business','Business'),
                'partner_id' : fields.many2one('res.partner','Partner'),
        }
        def init(self, cr):
                cr.execute("""
                        create or replace view partner_from_invoice as (
                                select
                                        id, partner_id,business_id,name
                                                from
                                                        account_invoice
                                                )""")
partner_from_invoice()

class partner_from_list(osv.osv):
        _name = "partner.from.list"
        _description = "Partners from business"
        _order='partner_id'
        _auto = False
        _columns = {
                'business_id': fields.many2one('business.business','Business'),
                'partner_id' : fields.many2one('res.partner','Partner'),
        }
        def init(self, cr):
                cr.execute("""
                        create or replace view partner_from_list as (
                                SELECT DISTINCT r.id,r.id as partner_id,b.id as business_id
                                        FROM res_partner r, business_business b,
                                                partner_from_invoice i, partner_from_sale s,
                                                partner_from_purchase p
                                        WHERE b.id=p.business_id and b.id=s.business_id
and b.id=i.business_id
                                                )""")
partner_from_list()


class hr_timesheet_sheet(osv.osv):
   _inherit = "hr_timesheet_sheet.sheet"
   _columns = {
                   'business_id': fields.many2one('business.business','Business'),

           }
hr_timesheet_sheet()
```