

Développement du Réseau Social de L'Emploi :

Plateforme web de recherche d'emploi avec l'ajout des concepts tels l'intégration des réseaux sociaux, le développement de la visioconférence, le développement d'un moteur de pertinence.

Sommaire

Sommaire	1
Introduction générale.....	3
Insertion	5
A. OPENIKA SAS	6
I. Organigramme.....	9
II. Objectifs.....	11
B. Le projet « OPENIKA » : le réseau social de l'emploi.....	12
I. Le modèle OPENIKA.....	13
II. Missions assignées.....	21
III. La démarche employée	21
Modélisation du système	22
A. Analyse et modélisation du système.....	23
I. Les acteurs du système	23
II. Les cas d'utilisation « Use Cases ».....	26
III. Diagrammes de séquences.....	28
IV. Diagrammes de classes.....	32
B. Vue générale du système d'information.....	36
Développement.....	37
A. Environnement de développement	38
I. Equipement matériel.....	38
II. Equipement logiciel	38
III. Langages et technologies employés.....	39
B. Le Framework « SYMFONY 2 »	40
I. Contrôleur et routeur.....	41
II. La vue.....	43
III. Le modèle (base de données)	45
IV. « Bundle » dans SYMFONY 2	51
C. Quelques modules développés	52
I. Simulateur d'appariement sémantique	52
II. L'import de profil LinkedIn	54
III. Le crawler	55
IV. Autres modules	55
Expérience acquise	56

Glossaire	57
Conclusion générale	58
Annexes	61

Introduction générale

A l'heure où le monde tend à resserrer ses distances, à rapprocher le maximum de personnes qui ont très peu de choses en commun, les réseaux en général constituent sans aucun doute un catalyseur très visible et actif à cet effet. Parmi ces réseaux, nous retrouvons les communautés virtuelles ou web qui avec le canal de l'internet « le réseau des réseaux », occupent la quasi-totalité des activités dans le quotidien de nombreuses personnes et divers domaines. Mais tout ceci dans quels objectifs ? Pourquoi rapprocher des personnes ?

Les réponses à ces interrogations certes loin d'être exhaustives, se trouvent nous pouvons dire en elles mêmes. La société, le monde des idées, les découvertes et inventions, le travail ont toujours eu un dénominateur commun, en l'occurrence la collaboration, l'échange, le partage. Mais intéressons-nous plus particulièrement au monde virtuel, donc de l'internet, dans ce processus de réseaux, de communautés.

Actuellement, la portée des réseaux sociaux tels « **Facebook** » et « **Twitter** » n'est plus à démontrer ainsi que leur impact sur nos vies. Ils concourent à créer entre leurs membres une « *amitié virtuelle* », qui peut parfois devenir réelle à long terme. Les distances n'existent quasiment plus, le partage de l'information (actualités de tous genres, campagnes de marketing, publicités diverses,...) utilisent ces réseaux comme moyen de diffusion pour atteindre le maximum de cibles. Si vous n'avez aucune idée sur la manière de vous y prendre pour débiter, plusieurs moyens sont à votre disposition dans ces réseaux afin de vous faire correspondre à des individus grâce par exemple aux calculs de « **Matching** », « **Ranking** » selon vos infos personnelles ou encore à l'envoi d'invitations à vos contacts par mails avec votre accord bien sûr.

Depuis peu, une autre vague de réseaux est arrivée. Certains vont parler de réseaux professionnels pour marquer leur différence, mais au fond l'objectif est le même : rapprocher l'offre et la demande. Dans le but d'avoir une réelle interactivité entre les protagonistes (candidats aux offres et recruteurs), une touche communautaire est ajoutée selon le modèle d'un **Facebook**. Parmi les précurseurs, on note « **LinkedIn** », « **Viadeo** » ..., qui proposent en plus du

jobboard, une approche « réseau social » à travers la mise en relation des individus par divers outils : groupes de discussion, forums, hubs, recommandations, mise en relation ...

Le projet « **OPENIKA** » s'inscrit donc dans cette lignée mais en apportant une réelle et moins négligeable valeur ajoutée. Tout au long du présent document, il sera question de parler plus en détails du périmètre de ce projet, de l'analyse et de la conception du projet à l'implémentation de la solution finale tout en passant par les objectifs fixés au départ et la démarche employée pour y arriver.

Insertion

Le stage **ST40 « Assistant Ingénieur BAC+4 »** de l'Université de Technologie de Belfort - Montbéliard à l'origine de la rédaction du présent document a certes débuté le 6 février 2012 à 9h30 mais le projet « OPENIKA » du moins dans les idées date de bien avant avec, à la base une entreprise : « **OPENIKA SAS** » alimentée par des ressources humaines hautement qualifiées.

Durant cette partie, principaux axes explorés ici seront successivement la présentation de l'entreprise « **OPENIKA SAS** » dans son histoire, ses objectifs, une vue générale du projet à travers son modèle et plus-value, la démarche employée pour atteindre les objectifs de notre mission de stage vis-à-vis du projet.

A. OPENIKA SAS

Parler d'« **OPENIKA SAS** » revient d'abord à évoquer le projet « **OPENIKA** » car c'est derrière celui-ci que s'est greffée toute l'équipe actuelle. Mais comment est née l'idée de projet ?

C'est assez courant mais toujours surprenant à la fois, les idées qui amènent à des projets ou réalisations impressionnantes, innovantes et grandes. « **OPENIKA** » n'échappe pas à cela bien au contraire, il en est un exemple bien placé. En effet, l'idée est partie d'un vécu personnel. Monsieur **Pascal ALANGAROM** actuellement **Président** de l'entreprise, s'est retrouvé sans emploi en août 2010. Cette période lui a donc été propice - on peut le dire maintenant bien sûr - puisqu'elle lui a permis de suivre différentes formations au cours desquelles il a pris conscience que les difficultés qu'il éprouvait pour retrouver un emploi étaient partagées avec d'autres personnes :

- **Trop de sites d'emploi**
- **Trop de temps à consacrer sur ces sites**
- **Manque d'opérationnalité de ces sites**
- **Manque de suivi etc....**

Dès lors, l'idée est née... initialement de concevoir le "**Meetic de l'Emploi**", pour évoluer ensuite vers "**LE Réseau Social de l'Emploi**".

Il n'est en aucune manière question d'arrivisme, mais bel et bien, d'une réelle volonté d'apporter une réponse à l'ensemble des demandeurs d'emploi, isolés dans leurs démarches, et trop souvent en manque de considération. Après réflexion, il apparaît que cette solution permettrait également aux salariés en poste de rester en veille.

L'indicateur de succès d' « **OPENIKA** » sera sa présence à l'international. En effet, l'entreprise ambitionne de devenir le N°1 français en un an, puis le N°3 européen en 3 ans. Le plan de développement à l'international sera basé sur le rachat de « challenger » dans les différents pays, imprégné de la culture économique locale, et de leur proposer la puissance d'une marque, un business model novateur et donc une alliance permettant de bâtir le leader national. Bien entendu, les objectifs de rentabilité seront en permanence à l'esprit, ainsi que les indicateurs de fréquentation du site.

Enfin, « **OPENIKA** » émet le souhait de limiter au maximum les immobilisations et ainsi favoriser, dans une logique de bon sens, les prestations ponctuelles ou encore celles qui nécessitent des effectifs conséquents (Développement de la V1, web Marketing, ...). L'objectif est donc d'avoir une structure légère, saine, et générant du cash--flow avec une excellente rentabilité.

« **OPENIKA SAS** » entreprise a, quant à elle, officiellement vu le jour juridiquement parlant le **16 avril 2012** avec un capital de **5000 euros** détenu par les co-fondateurs ci-après :

- **Pascal ALANGAROM** (40%)
- **Marc CERLI** (20%)
- **Betty BOUSQUET** (20%)
- **Patrick DUSSART** (5%)
- **Bouali DHIA** (5%)
- **Georges TRAN DU PHUOC** (5%)
- **Christophe RAYMOND** (5%)

Et s'est établie dans des locaux de « **Promopole** », une pépinière d'entreprise située au « **12 avenue des Près, Saint Quentin en Yvelines 78180 Montigny Le Bretonneux** ».

« **OPENIKA SAS** » projette une levée de fonds de la part d'investisseurs à la hauteur de **400 000 euros** afin de mener à bien le projet « **OPENIKA** ». La forme juridique de l'entreprise est une « **Société par Actions Simplifiées** » (**SAS**) pour permettre une plus grande aisance dans son développement. Le

conseil retenu pour accompagner l'entreprise dans cette démarche est **ORSAY Law** (www.orsaylaw.fr), 146 Avenue des Champs Elysées, Paris 8, en la personne de Maître **D'URZO Louis**. **ORSAY Law** est en effet un cabinet d'avocats, spécialiste en levée de fonds et en propriété intellectuelle : il pilote la levée des fonds des startups ancrées dans l'univers de l'internet et IT.

La nomination de l'entreprise a, quant à elle été confiée à « **BENEFIK** » sis au **14 place de l'église 92500 Rueil-Malmaison**, entreprise spécialisée dans la **création des noms**, la **vérification linguistique internationale** (moyen de s'assurer que vos noms n'ont aucune connotation gênante dans les langues et autres cultures), **l'architecture de marque** entre autres. Trois noms ont été proposés au départ : « **wanajob** », « **dessinemoiunjob** », « **openika** » pour le projet ; au final « **OPENIKA** » a été retenu pour :

- « **Open** » : pour matérialiser **l'ouverture à toutes les technologies (web, mobile), l'ouverture entre les gens**.
- « **Ika** » : Il s'agit d'un mot d'une langue des Caraïbes, qui veut dire les "Gens", la "Population".

La volonté était de ne pas utiliser le mot "**Job**" ou "**Emploi**" pour ne pas apparaître au fond de la page de résultats de recherche d'un moteur tel « **Google** », mais aussi de faire d'**OPENIKA** une marque que l'on peut décliner à l'international.

Enfin « **OPENIKA** » est une marque déposée, un projet validé par la Commission Européenne avec qui l'entreprise est en collaboration dans le modèle « **EURES ICT Skills** » dont nous reviendrons plus tard dans ce document lorsque que nous présenterons le **modèle OPENIKA**.

I. Organigramme

« **OPENIKA SAS** », comme toute société, ce sont aussi des ressources humaines qualifiées avec des missions bien précises. En effet, on rencontre 4 grands niveaux de responsabilité dans la hiérarchie de la société :

➤ La direction Générale :

Elle est constituée de **Pascal ALANGAROM**, président d'**OPENIKA SAS** et de **Christophe RAYMOND** actionnaire.

➤ Le Marketing :

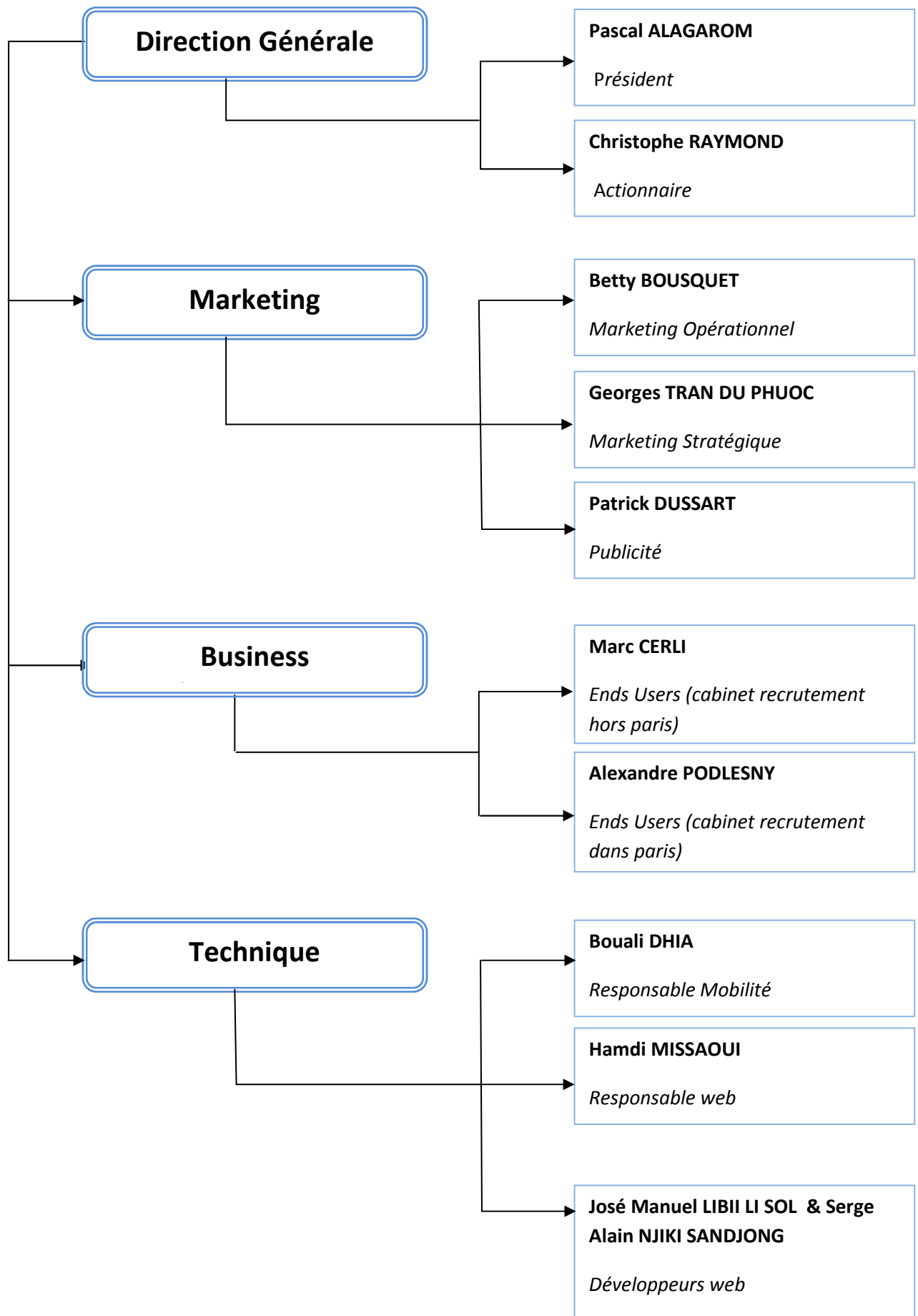
Il est subdivisé en plusieurs spécialités à savoir le **marketing opérationnel** à la charge de **Betty BOUSQUET**, le **marketing stratégique** pour **Georges TRAN DU PHUOC**, et enfin « **publicité** » piloté par **Patrick DUSSART**.

➤ Le Business Développement :

Sous la responsabilité de **Marc CERLI** et **Alexandre PODLESNY**.

➤ La Technique :

Ici nous retrouverons **Hamdi MISSAOUI** responsable web, **Bouali DHIA** responsable Mobilité, **José Manuel LIBII LI SOL** et **Serge Alain NJIKI SANDJONG** en tant que développeurs web.



En complément de cette équipe dirigeante à forte expérience et valeur ajoutée, **OPENIKA SAS** a su en parallèle s'entourer de partenaires et mentors de qualité lui apportant soutiens et conseils avisés pour la construction et le développement du projet « **OPENIKA** ». On peut citer à cet effet :

- **Béatrice COLLET**, *DRH WW South EMEA Groupe Smith Medical (US)* : pour conseils avisés quant aux besoins des DRH de grands groupes industriels.
- **Bernard GIRBAL**, *Executive Assessment Consultant, Groupe ADECCO Ex VP EMEA PACKETEER (US) --- Ex VP EMEA 3COM (US)* : pour conseils avisés quant aux besoins des Cabinets de recrutement.
- **Cédric BANNEL**, *Fondateur des sites Internet Caradisiac.com et Leboncoin.fr* : pour conseils avisés quant à notre concept de site communautaire de l'emploi.
- **Thierry GUERIN**, *Directeur du Développement International, NICE SYSTEMS* : pour conseils avisés quant au développement à l'international.

II. Objectifs

« **OPENIKA SAS** », à travers ce projet, poursuit plusieurs objectifs étalés sur toute la palette de services qu'il propose, services qui seront décrits plus en détail lorsque nous aborderons la présentation générale du projet. Il s'agit en effet :

- De proposer un réseau social dédié spécifiquement à l'Emploi.
- D'identifier rapidement et simplement les profils pertinents par rapport à une offre d'emploi (**Matching** et **Ranking** de candidats par rapport à une offre d'emploi).
- D'améliorer le niveau d'analyse, de productivité et d'efficacité des actions des candidats et recruteurs.
- De permettre à un candidat de donner un aperçu global de son parcours, diplômes et réalisation.

- D'attirer les meilleurs candidats, et, valoriser l'image du recruteur (Marque Employeur).
- S'affranchir des distances et gagner du temps grâce notamment à la visioconférence comme outil de communication pour effectuer des entretiens de recrutement.
- Établir des shortlists de candidats plus rapidement.

B. Le projet « OPENIKA » : le réseau social de l'emploi

Quel est le contexte actuel dans le monde de l'emploi sur la toile ? Comment s'effectue un processus de recrutement ? Combien de temps en moyenne doit-on consacrer soit pour trouver un emploi soit alors être sûr du candidat idéal pour un poste donné ?

D'un côté l'on constate :

- Beaucoup trop de sites dans lesquels je dois déposer un CV et effectuer des recherches.
- Pas d'offres convenant vraiment à mon profil.
- Obligation d'utiliser mon PC pour gérer ma recherche.
- Manque de suivi de mes nombreuses candidatures, opacité plus ou moins totale sur le processus de recrutement.
- Pour cause de mobilité, être parfois contraint à parcourir des centaines de kilomètre juste pour effectuer un entretien d'embauche...

De l'autre côté nous ne sommes pas en reste :

- Trop de sites où déposer des offres d'emploi avec un processus long, fastidieux et souvent coûteux financièrement.
- Répondre à tout prix aux candidatures reçues pour mes offres d'emploi très souvent nombreuses afin de préserver mon image et ce malgré le manque de temps
- Seulement **10%** des CV reçus en moyenne correspondent au profil recherché.
- Besoin d'un outil simple d'usage, rapide et qui sécurise mes recrutements...

Vous l'aurez compris, les attentes, que l'on soit candidat (recherche d'emploi) ou recruteur (recherche du candidat idéal) sont certes opposées mais il n'en reste pas moins qu'elles sont liées. « **OPENIKA** » propose alors, à travers son modèle innovant, un ensemble de services dans le but de combler le fossé se creusant de plus en plus entre candidats et recruteurs dans leurs objectifs respectifs.

I. Le modèle OPENIKA

Il ne s'agit plus ici de créer encore un produit supplémentaire pour le monde virtuel de l'emploi mais de cumuler en un seul package le meilleur du marché actuel tout en ajoutant une réelle plus-value pour se positionner à court et moyen terme comme un incontournable du secteur comme ont pu l'être avant des « **Monster** » et actuellement des « **Viadeo** » ou encore « **LinkedIn** ».

1. Le marché actuel

Que propose le contexte actuel ? Là constitue en effet le principal centre de réflexion de cette partie. Si on ne considère que les **jobboard** présents sur le marché tels « **Monster** », « **Cadre emploi** » ..., on constate une presque quasi absence d'un véritable réseau social, ils se caractérisent de la manière suivante :

- La saisie et/ou l'import de son CV sous format en général « **Word** », ou « **PDF** ».
- La possibilité de candidater aux offres d'emploi moyennant plus ou moins une somme d'argent sous diverses formes (paiement direct pour avoir accès à une offre, abonnement à une offre particulière à travers un changement de compte utilisateur basique vers un compte plus évolué souvent appelé dans la plupart des cas « **compte premium** »).
- La publication des offres d'emploi limitées dans le temps et à un prix au départ sans la garantie de recueillir au terme de la période de l'offre le ou les candidats correspondants à l'offre réellement. Par exemple pour une publication d'une offre sur **30 jours** sur Monster, il faut compter en moyenne **600 euros**.
- Un moteur de pertinence pour la recherche d'emploi correspondant aux candidats ou encore pour rechercher des candidats correspondant aux

offres souvent essentiellement basée sur la syntaxe et non sur la sémantique.

- Des outils de mise en relation plus ou moins efficaces entre recruteurs et candidats très souvent proposés hors du produit principal ou alors quasiment inexistantes comme le « **chat** », la « **visioconférence** ». Dans ce cas les protagonistes sont donc laissés libres d'établir des moyens et stratégies pour se rapprocher les uns des autres en ayant en tête la réduction du temps passé et des coûts.

« **OPENIKA** » propose donc un ensemble de services en rupture totale sur certains points avec le contexte présent. Mais aussi de nouveaux pour marquer un virage par rapport d'une part aux réseaux professionnels dont l'activité sociale à l'image d'un réseau social est très faible et d'autre part, par rapport aux **jobboard** ne constituant au final en générale qu'une image fixée et non interactive du monde de l'emploi.

2. L'offre « **OPENIKA** »

Comment répondre plus efficacement aux attentes des deux communautés présentes ici à savoir « candidats » et « recruteurs » ? La proposition de valeur d' « **OPENIKA** » s'appuie sur un ensemble cohérent de services et de prestations. Cette structure comprend trois principaux blocs : **le JAE, le JSN, le JRM.**

a. Le JAE (Job Affinity Engine)

Considéré comme le cœur du système, le **JAE** est un moteur de pertinence dont l'objectif est de définir une correspondance entre une offre et un profil (candidat). Quels sont ses atouts ?

- Il constitue une innovation majeure grâce à un algorithme inédit.
- Il se base sur une modélisation des arbres de connaissances donc sur la notion d'**ontologie** (outil principal du web sémantique). Cette modélisation dans le cadre du **JAE** utilise une structure de graphes non orientés pour marquer les distances entre les connaissances et donc leur degré de ressemblance. Par exemple, il sera possible de dire : « **UML** » est plus proche de « **MERISE** » que ne l'est « **PHP** ». Par conséquent, contrairement aux moteurs de recherche par mots clés se basant

uniquement sur de la syntaxe et non sur de la sémantique, le **JAE**, à défaut de trouver exactement le profil ou l'offre contenant le ou les critères recherchés, est capable de ressortir des résultats qui s'en rapprochent. « Je cherche **MERISE**, le **JAE** à défaut d'avoir exactement cela, me ressortira dans l'ordre profils ou offres comportant **UML** puis **PHP** ».

- Les résultats « offres » et « profils » seront classés (**RANKING**).
- Il permet d'avoir un taux d'affinité entre offre et profil précis et fiable par la prise en compte et la pondération de nombreux critères tels compétences, expériences, lieu, salaire, niveau de motivation, sémantique...
- Le référentiel de connaissances est sans cesse enrichi grâce au travail collaboratif entre recruteurs eux-mêmes d'une part, mais aussi avec « **OPENIKA** ».

Mais sur quelles bases « **OPENIKA** » tire son référentiel pour modéliser ces connaissances ? Quelle valeur ajoute-t-elle ?

Toutes ces interrogations nous ramènent tout droit au modèle **EURES** de la Commission Européenne et plus précisément des **ICT Skills** (compétences et métiers des technologies de l'information et de la communication).

EURES propose un modèle de compétences et métiers. Chaque métier appartient à une famille et est constitué de compétences pour pouvoir affirmer qu'un individu peut ou ne pas exercer ce dernier. De manière chiffrée, **EURES** c'est **6** familles de métiers, **20** métiers concernés et **36** compétences. La valeur ajoutée d'**OPENIKA** se situe au niveau de sa modélisation de connaissances, de savoir-faire (les méthodes) et savoir-être (les attitudes) greffée au dessus des compétences du modèle **EURES**. L'offre « **OPENIKA** » rajoute aussi un nuage de connaissances autour de chaque connaissance de son modèle.



Valeur ajoutée OPENIKA



Modèle EURES



Nuage de connaissance



Connaissance



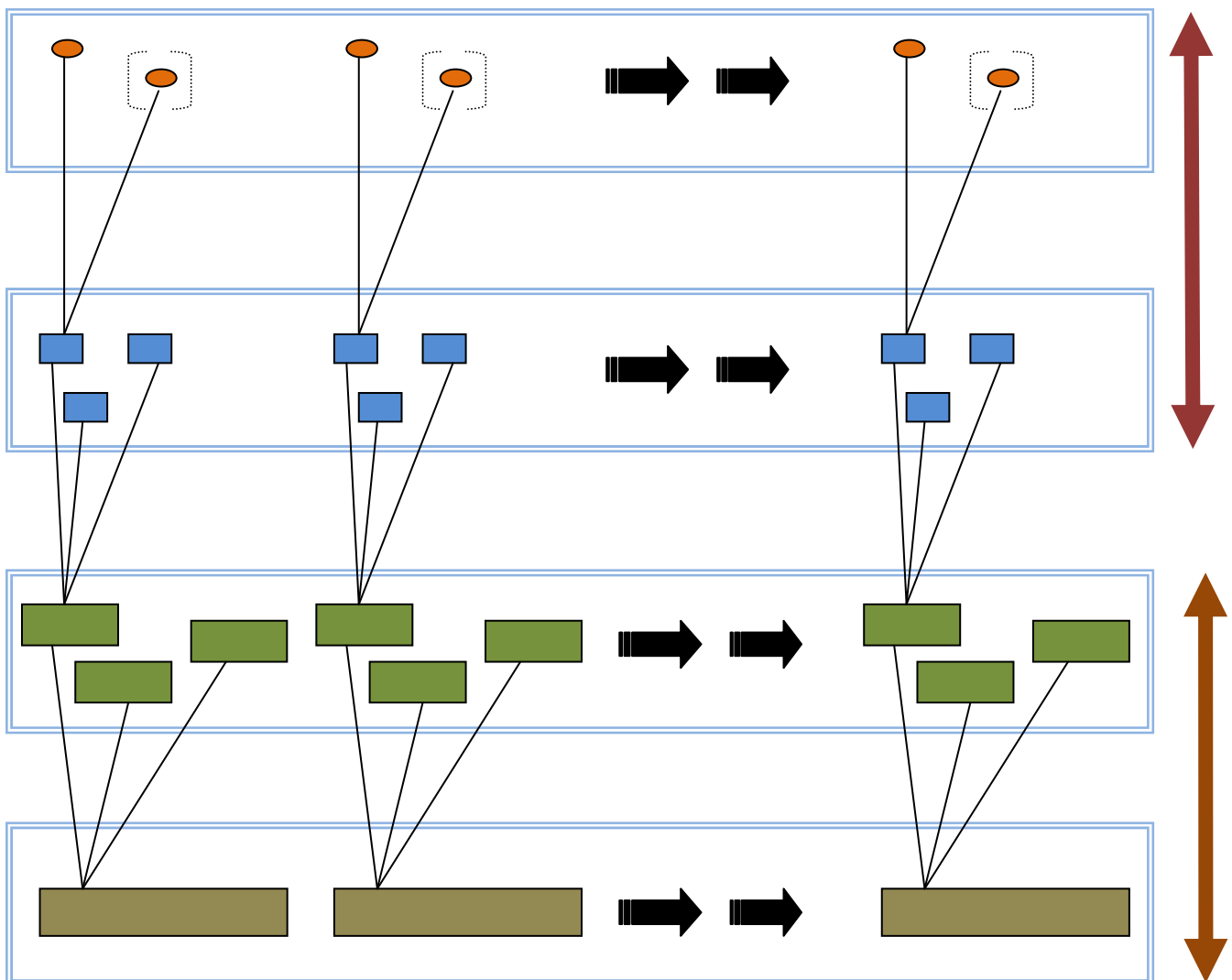
Compétence



Métier



Famille



b. Le JSN (*Job Social Network*)

A l'heure où le **Web 2.0** bat son plein, toute plateforme web se doit désormais, afin de rester au devant de la scène, d'être dotée d'une communauté forte, d'un réseau social qu'elle doit faire vivre par divers moyens et innovations. « **OPENIKA** », afin de ne pas juste être un lieu où l'on vient une fois et puis on repart n'échappe pas à cette mouvance de « réseau social ».

« **OPENIKA** » propose donc le **JSN** son réseau social spécifique au monde de l'emploi. Désormais, « recruteurs » et « candidats » constituent une communauté, un réseau où les échanges ou flux d'informations circulent. Ses atouts sont :

- L'inscription comme dans des réseaux sociaux tels « **facebook** » et « **twitter** » se veut simple et rapide : on s'affranchit alors de redéfinir son profil souvent assez conséquent. C'est la raison pour laquelle, « **OPENIKA** » se veut synchrone avec des réseaux professionnels déjà existants en proposant l'importation dans un premier temps de son profil « **LinkedIn** » (si existant) afin de compléter son compte « **OPENIKA** » et gagner un temps considérable.
- Aux outils de communication traditionnels comme le « **chat** », la « **messagerie interne** », viendra s'ajouter la « **visioconférence** » pour éviter les déplacements aux candidats et aux recruteurs, pour, par exemple planifier un entretien d'embauche.
- Les vies sociales et professionnelles des candidats et recruteurs formeront un véritable portrait de ces entités respectives grâce à une « **timeline** », qui sera elle-même formée à partir d'actions diverses de ces derniers sur la plateforme « **OPENIKA** ».
- Une gestion d'évènements et notifications spécifiques à l'environnement de chacune des entités (Recruteurs & Candidats).
- Une offre d'emploi publiée, par exemple, pourra voir sa cible initiale de candidats multipliée en nombre de façon considérable, grâce au **JSN**. De même, un candidat, grâce au **JSN**, se verra proposer une multitude d'offres et de solutions concernant sa recherche d'offres.

Au final, le candidat se constituera un réseau professionnel suffisamment fourni sur lequel il pourra communiquer tout changement sur sa

vie socioprofessionnelle. Parallèlement, le recruteur pourra avoir la garantie d'étendre son vivier de candidats 2.0, de fidéliser ces derniers, et saura y puiser de potentiels talents lorsqu'il aura des postes à pourvoir.

c. Le JRM (*Job Relationship Management*)

Inspiré des meilleures pratiques des applications CRM, cet outil permettra aux membres d'améliorer l'efficacité et niveau de productivité de leurs actions sur « **OPENIKA** ». Le **JRM**, c'est :

- Des tableaux de bord spécifiques aux recruteurs (gestion de mon vivier de candidats 2.0, de mes différents processus de recrutement...) et candidats (gestions de mes candidatures, mes rendez-vous, mes groupes de discussions...).
- L'automatisation des tâches candidats (centralisation des documents à un même endroit) et des recruteurs (reporting de l'activité recrutement, envoi de réponses positives ou négatives, confirmation de rendez-vous...).

Le recruteur peut donc se concentrer sur l'essentiel : la valorisation du capital humain présent et à venir de sa société. Il peut bénéficier en gain de temps, efficacité, réduction des coûts de recrutement.

3. Les cibles d' « **OPENIKA** »

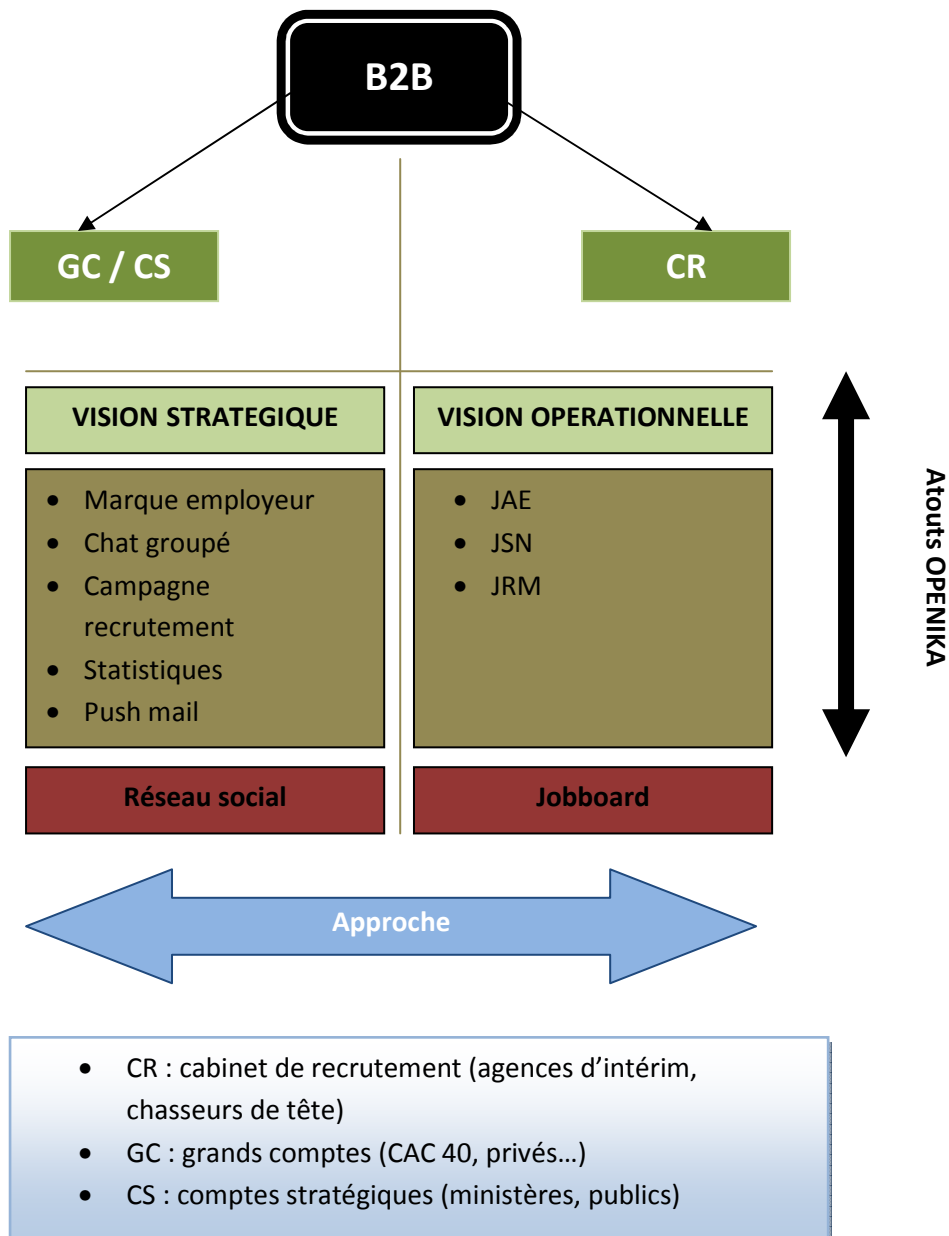
Nos clients cibles sont de deux types : les recruteurs et les candidats.

a. Les candidats

Notre cible correspond aux professions et catégories socioprofessionnelles aisées (**PCS +**), sur les secteurs secondaires et tertiaires, dans la mesure où ces profils seront plus aptes à adopter les nouvelles technologies de communications unifiées.

b. Les recruteurs

DRH (CAC 40, BGPME), Cabinet de recrutements, Chasseurs de tête, Gérant de PME constituent la cible côté recruteurs.



4. Le modèle économique

En abandonnant le modèle économique indexé sur la durée de publication d'une offre ou encore indexé sur l'offre unitaire, « OPENIKA » propose une grille tarifaire en rupture.

« OPENIKA » propose une véritable révolution en proposant notamment l'achat d'un contact hors vivier (côté recruteur), les différents packs « all inclusive »,...

a. Tarifs « côté candidat »

	Pack standard	Pack Premium
Prix	gratuit	14,90 € / mois
Durée d'engagement minimum		6 mois
Demandes de mise en relation	illimitée	
Utilisation du Job Affinity Engine	illimitée	
Réponses par offre en dehors du réseau	2 € par contact	illimité
Messagerie, Chat, Visioconférence	Non inclus	inclus

b. Tarifs « coté recruteur »

	Pack standard	Pack Premium	Pack Gold	PackPlatinum
Prix	gratuit	850 € / mois	595 € / mois	360 € / mois
Durée d'engagement minimum	aucune	6 mois	12 mois	24 mois
Demandes de mise en relation	illimitée			
Publication offres d'emploi	illimitée			
Utilisation du Job AffinityEngine	illimitée			
Achat Candidat Hors Réseau	50 €	35 €	35 €	inclus
Visioconférence	noninclus	inclus		

II. Missions assignées

Durant la période de ce stage **ST40** de **6** mois, plusieurs missions sont à remplir :

- Concevoir et réaliser le « **simulateur d'appariement sémantique** » : il s'agit ici de la première brique du **JAE** version back office. Il va permettre d'effectuer les tests du moteur de pertinence sur différentes offres et profils créés manuellement afin de mieux le paramétrer pour l'intégration au projet final « **OPENIKA** ».
- Réaliser un outil de « **crawling** » pour récupérer la base de connaissances déjà présente sur le réseau « **LinkedIn** ».
- Maîtrise du Framework « **SYMFONY 2** » pour l'implémentation de la plateforme web **OPENIKA**.
- Réalisation du module « **import profil LinkedIn** » à partir d'« **OPENIKA** » du **JSN**.
- Réalisation du module « **Chat** » du **JSN**.
- Réalisation du module « **Campagne de recrutement** ».
- Intégration du design.

III. La démarche employée

Comment allons-nous parvenir à notre fin ? Telle est la problématique ici. En effet, pour se faire, plusieurs moyens seront mis en œuvre.

- Pour l'analyse et la modélisation du système, nous aurons une approche objet donc nous pencherons vers **UML** comme langage de modélisation.
- Le système d'information sera supporté sur une base **MySQL**.
- Le principal langage de programmation web sera **PHP 5.3** autour duquel viendront se greffer **JavaScript**, **JQUERY**, **AJAX** (interactivité du site), **XHTML**, **CSS3** (interfaces du site) pour ne citer que ceux-là.
- Comme serveur web local, nous utiliserons **Apache 2.X**.

Modélisation du système

Dans cette partie, nous nous évoquerons la modélisation du système « **OPENIKA** » qui va nous permettre de ressortir le système d'information associé. Pour se faire, le langage de modélisation **UML** sera employé, avec **MySQL** comme **SGBD** (gestion de la base de données).

A. Analyse et modélisation du système

Le système étudié ici se compose de deux grandes entités que sont : le **recruteur**, le **candidat** et le **visiteur** pour ce qui est de la zone visible, et des **administrateurs** pour la zone non visible.

I. Les acteurs du système

1. Le visiteur

Sera considéré comme **visiteur**, tout internaute se connectant sur le site « **OPENIKA** » sans avoir de compte « recruteur », de compte « candidat » ou encore de compte « administrateur. Au sein donc de la plateforme web, cet acteur aura beaucoup plus un rôle consultatif sur les offres « **OPENIKA** », il pourra en outre :

- S'abonner à la newsletter.
- Se créer un compte « **candidat** » ou « **recruteur** ».
- Consulter les actualités du site.
- Partager une offre ou une actualité.
- Effectuer des recherches sur des offres « **OPENIKA** ».

2. Le candidat

Sera considéré comme **candidat**, tout visiteur au préalable s'étant créé un compte « **candidat** » sur le site. Il représente la « **demande** » dans le domaine de l'emploi. Ce sera un acteur très actif sur le système contrairement au visiteur.

Il pourra donc :

- Mettre à jour son profil (ses infos personnelles, ses connaissances, importer son profil « **LinkedIn** », ses expériences professionnelles...).

- Se connecter à son compte, se déconnecter, désactiver / supprimer son compte (gérer son compte).
- Envoyer des demandes de mise en relation à d'autres candidats ou recruteurs.
- S'abonner / se désabonner aux réseaux des recruteurs dont il souhaitera suivre l'activité.
- Rechercher des offres grâce au **JAÉ**.
- Postuler à des offres.
- Consulter sa timeline.
- Gérer ses candidatures.
- S'abonner à différentes newsletter.
- Consulter des actualités du site.
- Participer à des forums (poster des sujets, commenter des sujets...)
- S'inscrire à des campagnes de recrutement.
- Envoyer des messages privés.
- Souscrire à différentes offres d'abonnement sur le site pour avoir accès à plus de services.
- Participer à des « **Tchat** » (messagerie instantanée) avec autres candidats et recruteurs.
- Participer à des « **Tchat LIVE** » avec autres candidats et recruteurs
- Effectuer des visioconférences.
- Gérer son agenda.
- Partager des actualités, des offres...

3. Le recruteur

Le recruteur, tout comme le candidat est un acteur clé du système. Il s'agit de tout visiteur ayant souscrit à un compte « **recruteur** ». Il représente ici la partie « **offre** » dans le domaine de l'emploi. Il pourra donc :

- Gérer son profil.
- Se connecter / déconnecter à son compte, le désactiver.
- Publier des offres.
- Gérer ses offres.
- Consulter sa timeline.

- Envoyer des messages privés.
- Sélectionner ses candidats pour ses offres (grâce au **JAÉ**).
- Publier des campagnes de recrutement.
- Lancer des « **Tchat LIVE** ».
- Participer à des forums.
- Lancer des « **Tchat** » avec d'autres candidats et recruteurs.
- Gérer son agenda.
- Envoyer des demandes de mise en relation à des candidats et recruteurs.
- Partager des offres, actualités.
- Publier des actualités.
- Effectuer des visioconférences.
- Souscrire à différentes offres de compte afin d'avoir accès à davantage de services...

4. L'administrateur

Il constitue ici la face cachée du système pour tout internaute. Son rôle est de gérer dans l'ensemble les différentes fonctionnalités du site dans son ensemble, de surveiller leurs activités, de mettre à jour le site. Il pourra donc :

- Publier des actualités du site.
- Rédiger la newsletter.
- Administrer le référentiel de connaissances « **OPENIKA** ».
- Gérer les membres du site.
- Consulter les différents états d'activités sur le site...

Il s'agit ici tout simplement de schématiser les différentes actions que pourront effectuer les acteurs du système mentionnés précédemment.





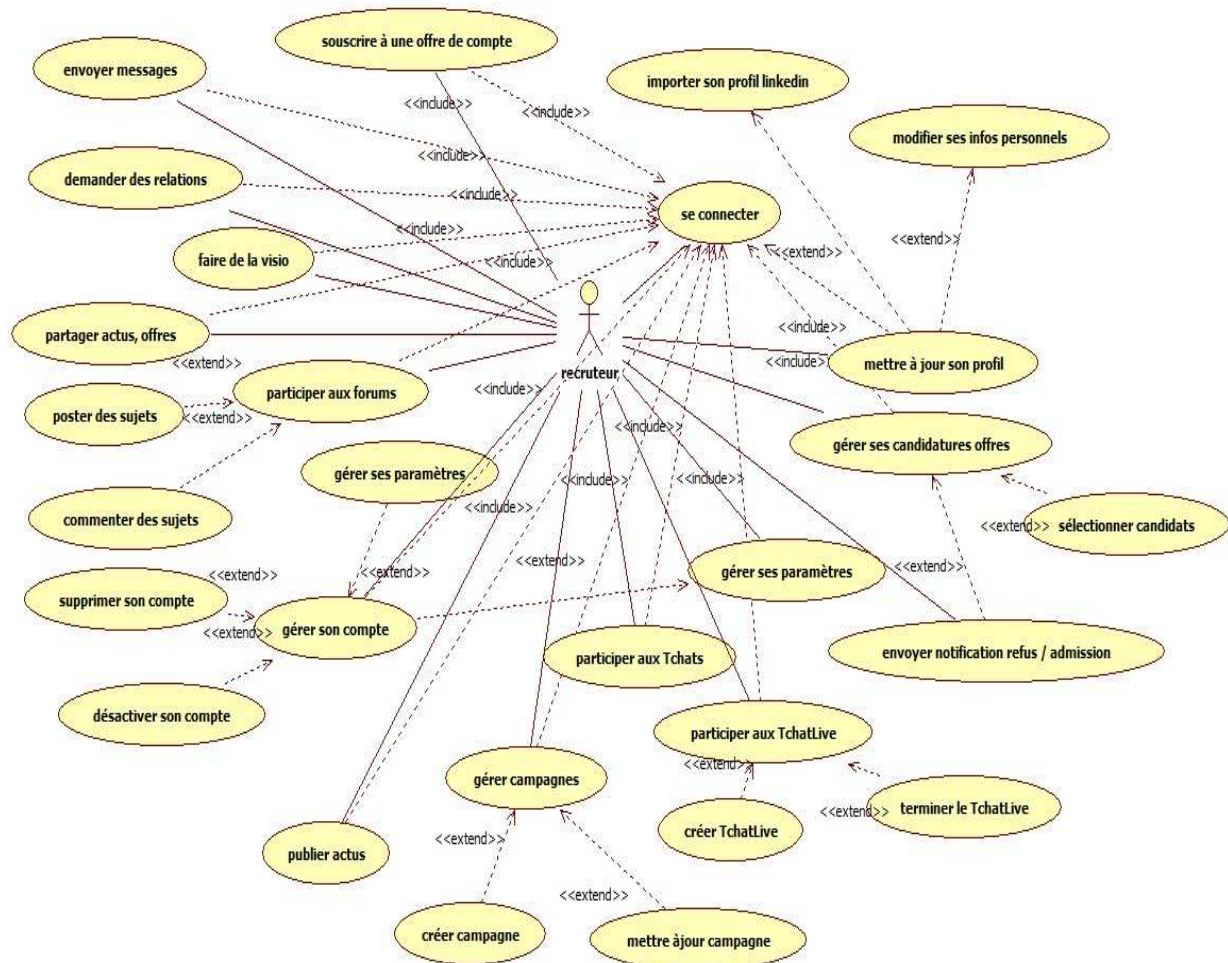


Figure 4: Diagramme cas d'utilisation : le recruteur

III. Diagrammes de séquences

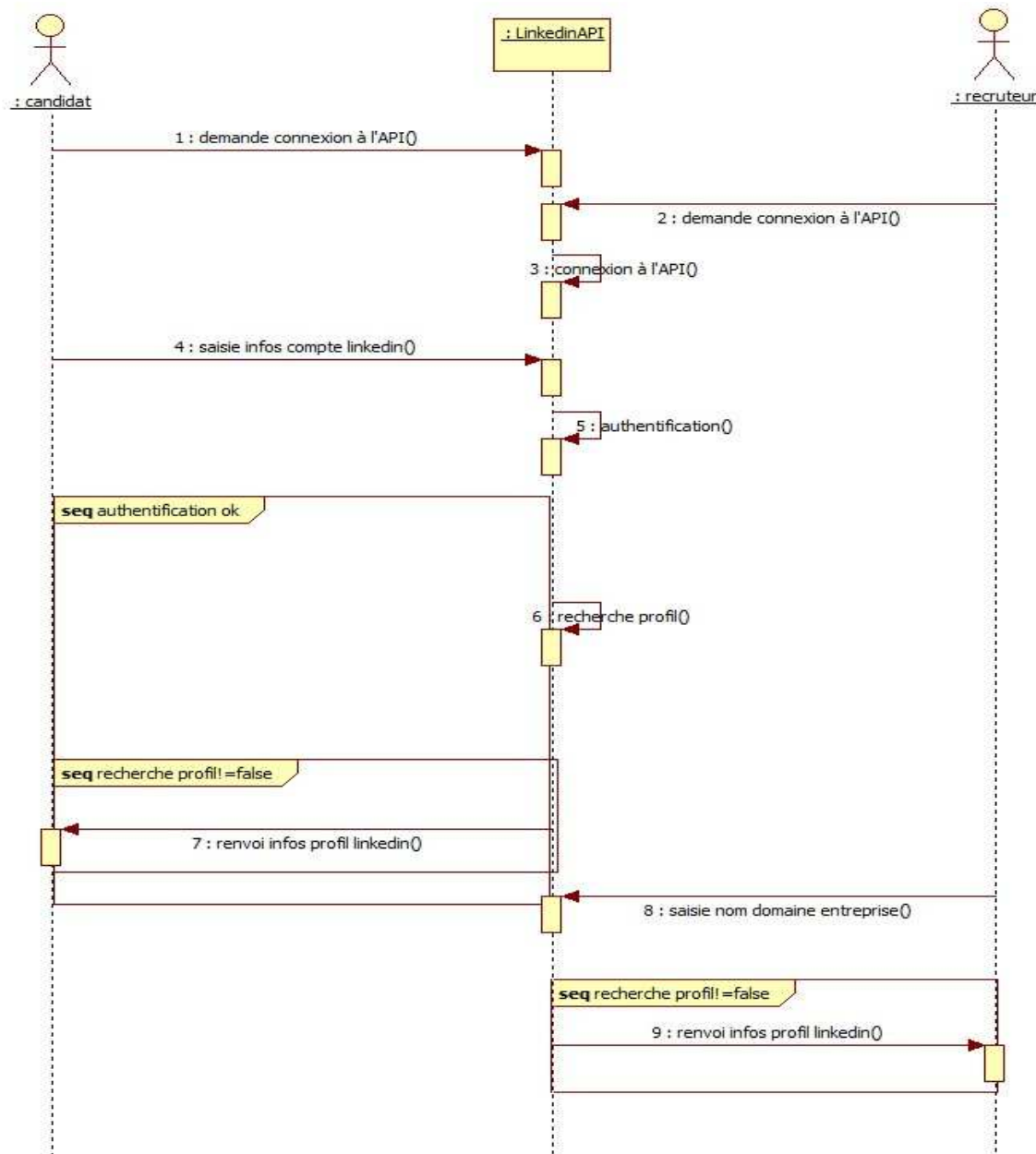
Le diagramme de séquence est une représentation schématique et séquentielle des différentes actions élémentaires concourant à la réalisation d'une tâche précise.

Ce concept est donc important dans le processus de modélisation UML car il permet de ressortir de façon plus explicite les différentes étapes successives menant à la réalisation des cas d'utilisation décrits ci-haut. Nous présenterons donc ici certains diagrammes de séquence **(03)** de notre système que nous avons jugé pertinents de détailler pour cause de leur non simplicité. (D'autres seront mis en annexes du présent document).

1. Diagramme de séquence « Importer profil LinkedIn »

Une fonctionnalité majeure du système « **OPENIKA** » est la possibilité pour le recruteur (ou le candidat) d'importer son profil depuis le réseau professionnel « **LinkedIn** » afin de lui faire gagner en temps. Les différentes étapes conduisant à cela se décrivent comme suit :

- Que l'on soit candidat ou recruteur, le processus débute de la même manière par l'établissement de connexion au réseau **LinkedIn** grâce à son **API**.
- Une fois cette connexion établie, le candidat renseigne ses identifiant et mot de passe **LinkedIn**. Le recruteur, quant à lui, rentre son nom de domaine, son entreprise.
- L'API authentifie le candidat, et renvoie les informations sur le profil du candidat au système « **OPENIKA** » ; le recruteur quant à lui reçoit par le système « **OPENIKA** » les infos sur son profil **LinkedIn**.
- Dès lors le système complète les différents profils « **OPENIKA** » avec ces informations provenant de **LinkedIn**.

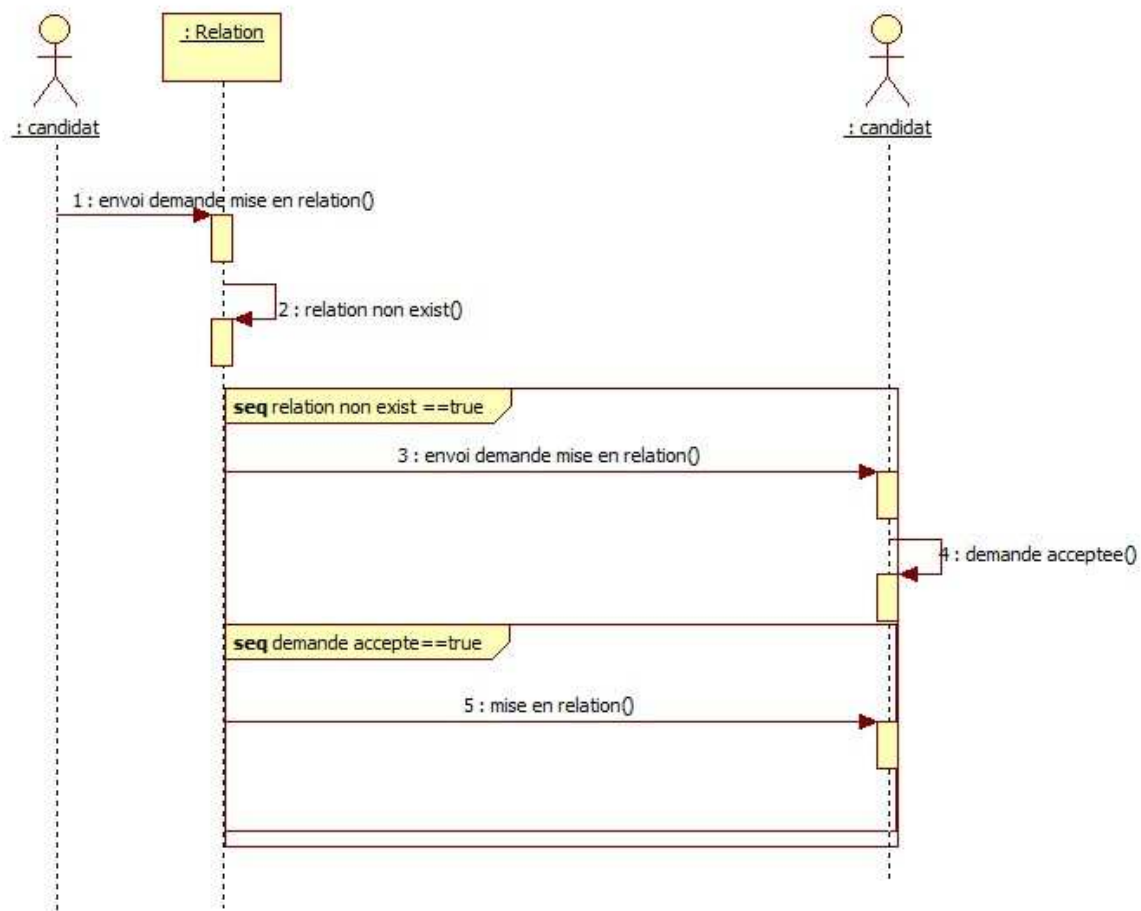


2. Diagramme de séquence « Mise en relation candidat - candidat »

La mise en relation entre deux candidats au sein du système « **OPENIKA** » passe par des envois de demande de mise en relation d'un côté et une acceptation ou refus de l'autre côté. Le processus se déroule ainsi :

- Envoi demande de mise en relation d'un **candidat A** à un **candidat B**.
- Vérification de la demande : il s'agit ici pour le système de voir si une demande n'est pas déjà en cours entre les deux protagonistes.
- Si ce n'est pas le cas, la demande est envoyée au **candidat B**.

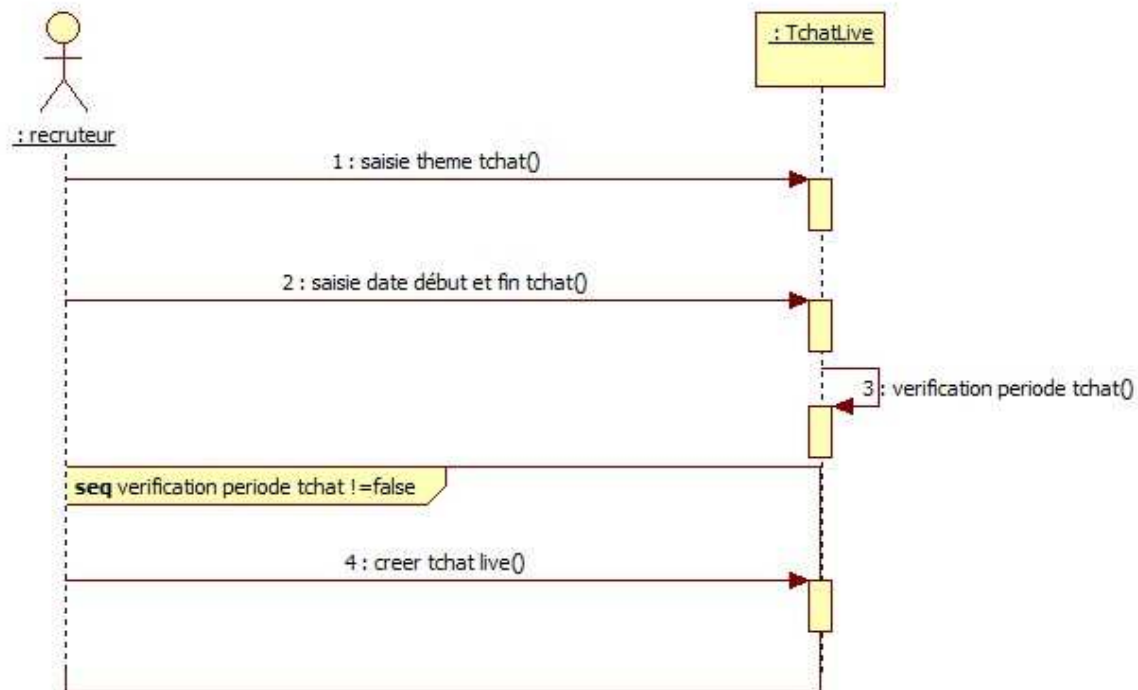
- Si B accepte cette demande, la mise en relation est effectuée.



3. Diagramme de séquence « Créer un TchatLive »

Il s'agit ici de décrire le processus de création d'un Tchat Live dans une campagne de recrutement conçue par un recruteur.

- Le recruteur entre les informations liées au tchat (thème du tchat live, date de début, date de fin)
- Si la date de début du « **tchat** » est postérieure à la date de fin, la création est annulée.
- Sinon le tchat est créé et sera initié plus tard par le recruteur.



IV. Diagrammes de classes

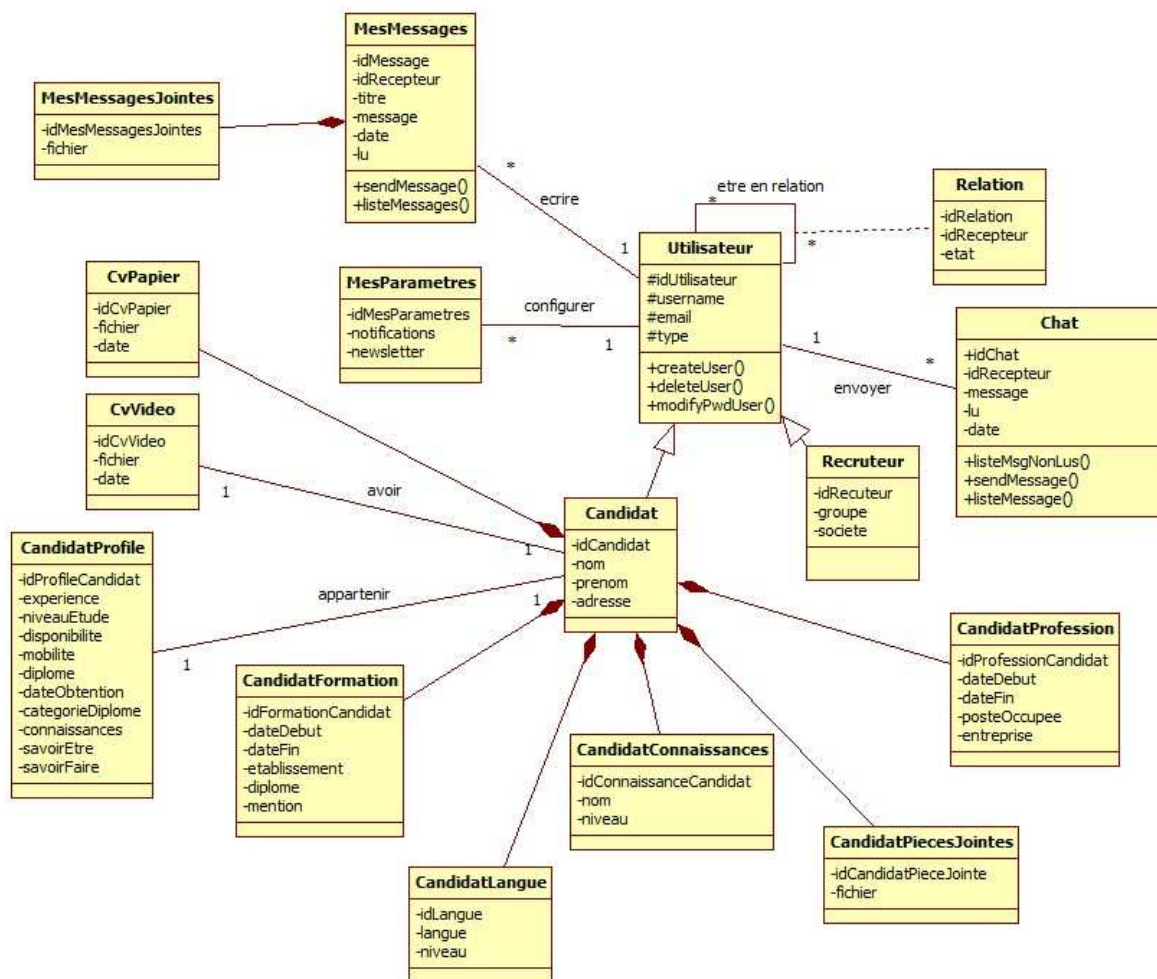
Le diagramme de classes est une collection d'éléments de modélisation statique qui montre la structure d'un modèle. Concernant notre projet, les points suivants ont été mis en exergue dans la mise en œuvre des diagrammes de classes :

- les classes qui participent à un cas d'utilisation,
- les classes associées dans la réalisation d'un scénario précis,
- la structure hiérarchique d'un ensemble de classes.

Vu le caractère complexe de notre modèle, nous avons regroupé les différentes classes qui le constituent dans plusieurs diagrammes de classes en fonction des critères de hiérarchie, de logique entre elles, mais aussi de réalisation des scénarios précis.

1. Espace « Utilisateur »

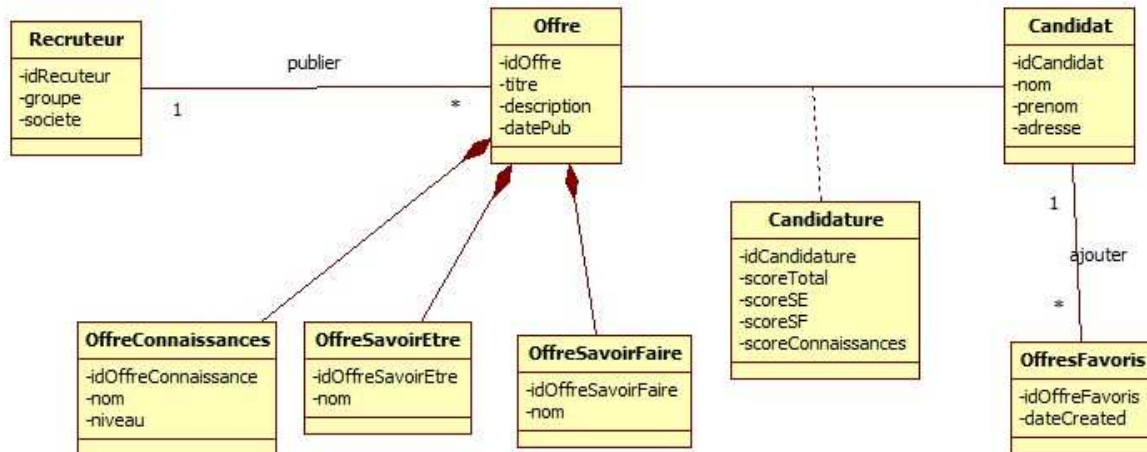
Nous avons regroupé dans ce diagramme de classes, les classes directement rattachées au candidat et au recruteur pour la création (construction) de leur profil « **OPENIKA** », de la messagerie privée, du « **chat** » entre utilisateurs...



2. Espace « Offre »

Nous avons regroupé dans ce diagramme de classes, les classes qui entrent dans la logique de :

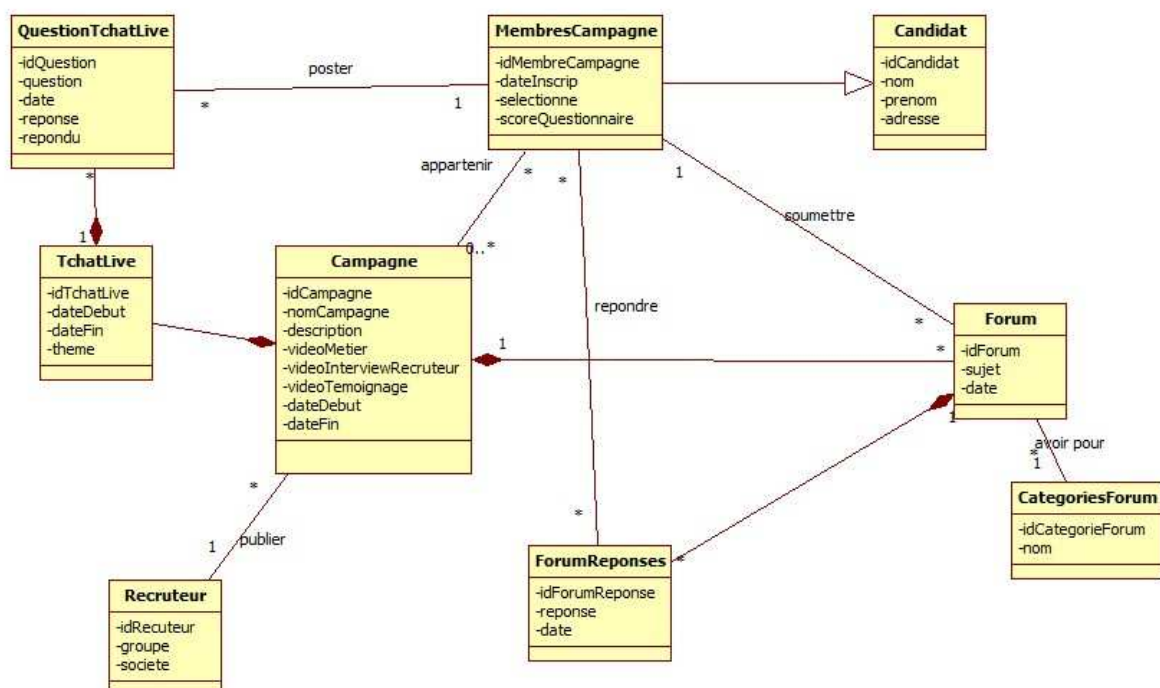
- construction d'une offre par le recruteur,
- candidature pour un candidat à une offre publiée par un recruteur.



3. Espace « Campagne de recrutement »

Sont mis en exergue ici, les classes qui participent aux processus de :

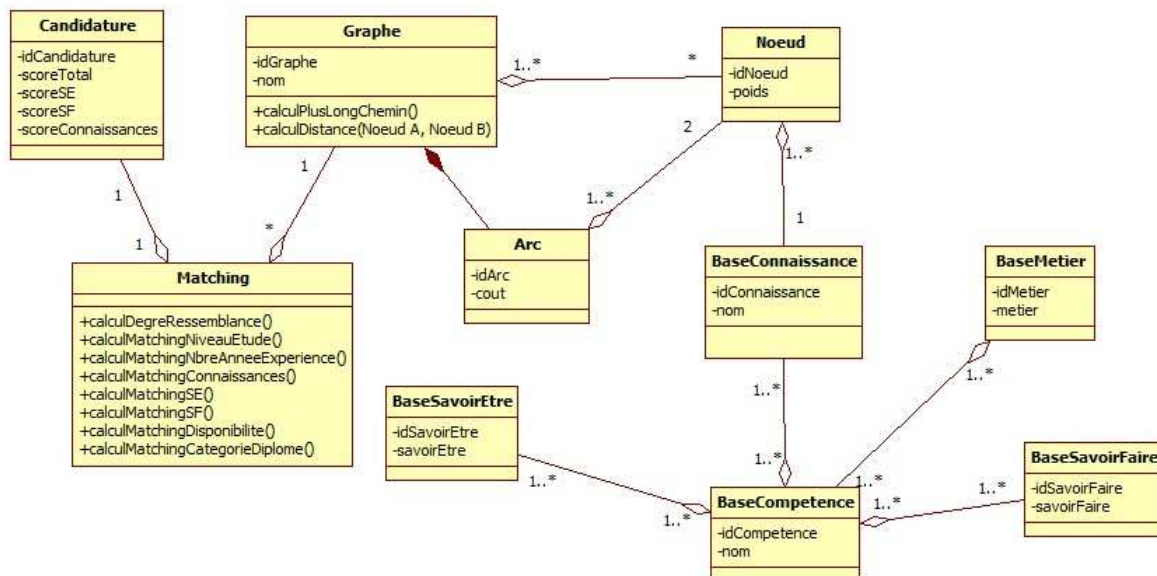
- construction d'une campagne de recrutement par un recruteur,
- gestion d'un « **TchatLive** » dans une campagne de recrutement,
- gestion des différents forums dans une campagne de recrutement.



4. Espace « Simulateur d'Appariement Sémantique »

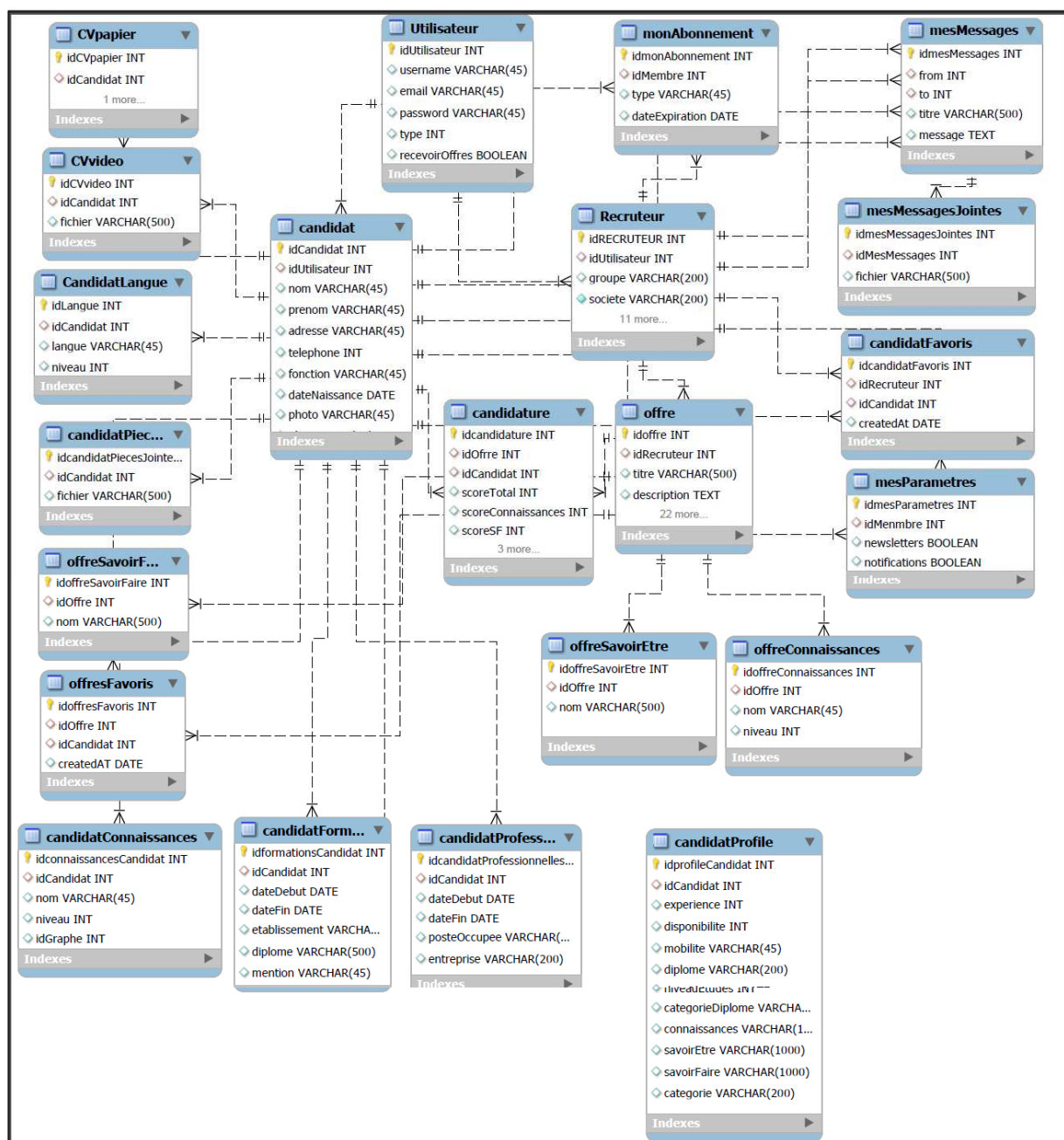
Dans cet espace, nous retrouvons les différentes entités participant à la réalisation du « **Simulateur d'Appariement Sémantique** » ; c'est la première version du JAE développée en interne pour besoin de tests. Les classes ici œuvrent pour :

- la construction de graphes de connaissances pour le calcul des distances entre elles (les connaissances),
- la gestion du référentiel de connaissances, métiers, compétences, savoir-faire et savoir-être d' « **OPENIKA** »,
- le « **Matching** » et le « **Ranking** » des candidatures suivant une offre définie.



B. Vue générale du système d'information

Le système d'information d'« OPENIKA » a été implémenté sur **MySQL** comme SGBD. Il s'agit ici, à ce niveau de réalisation du projet, de passer à la structuration des informations concernant l'application web dans une base de données. Grâce aux règles régissant le passage d'un modèle conceptuel de données matérialisé ici par nos diagrammes de classes, à un modèle logique de données donc à la base de données, nous avons généré le système d'information d'« OPENIKA ».



Développement

La phase de présentation et de modélisation du projet terminée, nous évoquons à présent dans cette partie l'implémentation proprement dite de la solution.

Il s'agira ici de présenter l'environnement de développement dans lequel nous avons évolué pour mener à bien ce projet. Les aspects d'équipements matériels, d'outils logiciels utilisés seront évoqués tour à tour. Nous évoquerons également les langages de programmation et Framework, les diverses technologies employés dans ce cadre. Enfin, nous aborderons certaines briques (modules) implémentées du projet.

A. Environnement de développement

I. Equipement matériel

Notre travail a été effectué sur 2 ordinateurs portables avec les caractéristiques suivantes :

HP EliteBook 8560p - 15.6" - Core i7 2640M - Windows 7 Edition professionnelle 64 bits - 4 Go RAM - 320 Go Disque dur

II. Equipement logiciel

Différents outils ont été employés dans ce projet à des diverses fins :

- **StarUML 5.0** : outil pour la génération des différents diagrammes utilisés pendant l'analyse et la conception.
- **NetBeans 7.1.2** : principal IDE utilisé dans ce projet pour développer l'ensemble des modules (bundles) sous SYMFONY2.
- **Apache 2** : serveur web installé en local pour des tests avant déploiement sur le site principal du projet.
- **MySQL Server 5** : serveur MySQL installé avec Apache2 pour gérer la base de données MySQL sous Linux (UBUNTU distribution).
- **MySQL WorkBench 5.2** : logiciel fourni pour MySQL pour la modélisation système. Dans le cadre de ce projet, il a servi pour générer la base de données sous MySQL.
- **SYMFONY 2** : Framework PHP utilisé dans le cadre de ce projet. Nous reviendrons plus en détails plus tard dans ce document.
- **FireBug 1.9** : outil de débogage de site web.

- **PHOTOSHOP Portable CS4** : outil de traitement d'images. Utilisé dans le cadre de ce projet pour la conception du design des pages web.
- **GIMP** : version plus légère que PHOTOSHOP.
- Les navigateurs **Opera, Google Chrome, Mozilla Firefox, Safari, Internet Explorer** employés ici pour tester le rendu des vues sous différents environnements.

III. Langages et technologies employés

Plusieurs langages et technologies ont été mis en œuvre dans le cadre de ce projet :

- **PHP5** : version 5 du langage web le plus utilisé sur la toile. C'est un langage qui s'exécute côté serveur avec dans cette version, l'implémentation encore plus poussée des concepts de la POO à l'image d'autres langages tels JAVA.
- **JavaScript** : langage de programmation incorporé dans les navigateurs web. Il rend interactif autant que possible le fonctionnement d'une page web. Il s'exécute côté client, ce qui le rend plus rapide par rapport à un langage serveur.
- **JQUERY** : bibliothèque JavaScript. Son but est de simplifier la syntaxe javascript c'est-à-dire « faire beaucoup en écrivant peu » ce qui n'est pas le cas en javascript pur. Il met aussi l'accent sur les effets et animations de tout genre sur une page web.
- **AJAX** : technologie javascript permettant de faire le lien entre javascript et un langage serveur (PHP par exemple). Il modifie le contenu d'une partie de page sans avoir à recharger celle-ci, effectue des requêtes vers le serveur et exécute des callback en fonction des réponses reçues de ce dernier.
- **CSS3** : langage de formatage de pages web.
- **XHTML** : langage de balisage servant à écrire des pages web.
- **JSON** : format de structuration de données en javascript utilisé notamment par AJAX pour retourner les réponses et transmettre des informations au serveur.
- **XML** : langage de balisage pour structurer des données sur le web.

B. Le Framework « SYMFONY 2 »

Afin de garantir :

- Un gain de productivité,
- Un travail en groupe entre développeurs du projet,
- Une clarté dans l'organisation des sources du projet grâce à l'architecture « **MVC** »,
- Une maintenance évolutive et plus aisée du système,
- Une réutilisation des composants sans les réinventer complètement,
- Une robustesse du code, l'utilisation d'un **Framework** est apparue, incontournable pour mener à bien ce projet.

Mais qu'est-ce qu'un **Framework** ? Un **Framework** est un ensemble de composants qui servent à créer les fondations, l'architecture et les grandes lignes d'un système (logiciel, site web...). Il en existe des centaines pour différentes langages de programmation. Mais en revanche, cet outil demande aussi une courbe d'apprentissage plus élevée par rapport au langage natif dont il se sert pour faciliter les tâches du développeur.

Le principal langage de développement utilisé pour ce projet étant « PHP », c'est tout logiquement que nous nous sommes orientés vers un Framework PHP. **SYMFONY (version 2)** est donc un Framework PHP édité par la société française « **SensioLabs** » dont le créateur est **Fabien POTENCIER**. Cet outil étant open Source, il est donc maintenu et amélioré par une grande communauté de développeurs. C'est grâce à cette communauté que la version 2 de SYMFONY a vu le jour en **août 2011**. Ce Framework requiert les concepts de **POO** car ici tout est objet ou classe. La version 2 nécessite **PHP** dans sa **version 5.3** ou ultérieure. Utilisant la couche d'abstraction de données **PDO**, il peut donc s'interfacer avec plusieurs **SGBD (MySQL, ORACLE, MS SQL...)**. Nous utiliserons **MySQL**.

L'installation de SYMFONY2 est assez simple. Il est téléchargeable au sein d'une archive à extraire dans son répertoire local du serveur web dans un environnement de développement. Une fois l'extraction terminée, l'architecture physique comprend les répertoires suivants :

- **app/** : ce répertoire contient tout ce qui concerne notre application sauf son code source. Il s'agit donc de la configuration, le cache, les fichiers logs...
- **src/** : c'est le lieu qui contiendra notre code source organisé en « bundle », briques de notre application.
- **bin/** : ce répertoire ne contient qu'un seul script pour mettre à jour les librairies tierces.
- **web/** : ce répertoire contient tous les fichiers destinés à nos visiteurs : images, fichiers CSS et JavaScript, etc. Il contient également le contrôleur frontal (**app.php** et **app_dev.php**). Ce contrôleur frontal est en quelque sorte la porte d'entrée du Framework, car toute requête passe par lui. C'est une sorte « **index.php** » pour un site web. Son rôle est d'appeler le noyau « **Kernel** » de SYMFONY2 en disant « **On vient de recevoir une requête, transforme la en réponse** ». Il en existe deux en fonction de l'environnement de travail :
 - Dans un environnement de production, le contrôleur est « **app.php** »
 - Dans un environnement de développement, le contrôleur est « **app_dev.php** ».
- **vendor/** : ce répertoire contient toutes les librairies externes à notre application comme, par exemple : **SwiftMailer** (envoi de mails), **Doctrine** (pour la base de données), **Twig** (pour les « Template » ou vues).

Pour l'architecture conceptuelle, SYMFONY2 utilise le **MVC** (Modèle Vue Contrôleur). C'est un découpage très répandu pour beaucoup de langages tels JAVA, PHP. Il permet une séparation en couches des composantes d'un système selon leur rôle logique. Ici tout fichier est caractérisé au tout début par son « **namespace** ».

I. Contrôleur et routeur

Son rôle est de générer la réponse à la requête HTTP demandée par un utilisateur. Il est la couche qui se charge d'analyser et de traiter la requête de l'utilisateur. Le contrôleur contient la logique de notre site Internet et va utiliser les autres composants : les modèles et les vues. Concrètement un contrôleur va récupérer, par exemple, les informations sur l'utilisateur courant,

vérifier qu'il a le droit de modifier tel article, récupérer cet article et demander la page du formulaire d'édition de l'article. Au sein de SYMFONY, les fichiers (classes) servant de contrôleur ont la nomenclature suivante : **«nomcontrôleur»Controller**. Ces classes héritent d'une classe mère de la librairie SYMFONY : « Controller »

Symfony\Bundle\FrameworkBundle\Controller\Controller;

Ex : CandidatController.php, CampagneController.php.

Afin de marquer le fait que ces composants effectuent des actions à l'encontre d'autres composants du Framework, les méthodes de ces classes portent le suffixe « **Action** » à la fin.

Ex : ***public function sauverCampagneAction() {}***

Les contrôleurs sont à leur tour commandés par un composant important de SYMFONY : **le routeur**. Son rôle est, à partir d'une URL, de déterminer quel contrôleur appeler et avec quels arguments. Cela permet de configurer son application pour avoir de très belles URL, ce qui est important pour le référencement et pour le confort des visiteurs. Il est l'équivalent de l'URL Rewriting sauf que cette technique de référencement se fait en PHP et non à travers des **.htaccess** à configurer dans un serveur Apache.

Un routeur est créé dans un fichier au format « **yml** » et se compose de :

- **Le nom de la route** : c'est un nom unique donné par le développeur pour identifier de façon claire sa route.
- **Le pattern** : c'est l'url (avec les arguments éventuels) à fournir pour accéder à la ressource à travers le contrôleur.
- **L'action à exécuter à partir d'un contrôleur (Defaults)** : c'est le contrôleur à capturer pour l'action souhaitée dans l'url fournie.

Ex:

HelloTheWorld: (1)

pattern: /hello-world (2)

defaults: { _controller: SdzBlogBundle:Blog:index } (3)

Dans l'exemple ci-dessus, le nom de la route (1) est : « **HelloTheWorld** », l'url à saisir dans la barre d'adresse pour parvenir au contrôleur devant exécuter l'action souhaiter est : **/hello-word** (2). Pour se faire, le routeur indique (3) qu'il s'agit pour cette action du contrôleur « **Blog** » du bundle « **Blog** » de l'organisation « **Sdz** », il ajoute aussi que la méthode à exécuter pour le contrôleur sera « **indexAction()** »

II. La vue

Son rôle est d'afficher les pages ou ressources que lui ordonne de faire le contrôleur. Désormais donc avec cette séparation vues / contrôleurs, designers et développeurs PHP peuvent travailler ensemble sans se marcher dessus.

A cet effet, SYMFONY2 utilise ici un moteur de Template pour générer ces vues : **Twig**. Un Template sous Twig (une vue) aura donc en plus des codes HTML/XML/TEXT et autres, une syntaxe particulière pour effectuer diverses actions. Désormais le code PHP sera séparé de tous ces autres codes. On pourra par exemple accéder à un objet sur une vue afin d'afficher le contenu, faire des tests, des boucles « **for** » etc. Les fichiers des vues utilisant Twig comme moteur se terminent par l'extension « **.twig** ».

L'intérêt se situe à plusieurs niveaux :

- la syntaxe est plus concise et plus claire. Par exemple pour afficher une variable, **{{ mavar }}** suffit, alors qu'en PHP, il faudrait faire **<?php echo \$mavar; ?>** ;
- il y a quelques fonctionnalités en plus, comme l'héritage de templates. Cela serait possible en PHP, mais il faudrait coder soi-même le système et cela ne serait pas aussi esthétique ;
- il sécurise vos variables automatiquement : plus besoin de se soucier de **<?php htmlentities(); ?>** ou **<?php addslashes(); ?>**

▪ Boucle dans la syntaxe Twig

```
<ul>
```

```
{% for user in users %}
```

```
<li>{{ user.pseudo }}</li>
```

```
{% endfor %}
```

```
</ul>
```

Ici nous avons l'équivalent d'une boucle « foreach » en PHP sur un tableau ou objet transmis car twig aussi est capable d'afficher les informations d'un objet classique défini avec les getters et setters.

▪ Créer une variable dans la syntaxe Twig

`{% set foo = 'bar' %}` on définit la variable « foo » avec « bar » comme valeur.

▪ Effectuer des conditions dans la syntaxe Twig

```
{% if pseudo == 'winzou' %}
```

```
    OK accès autorisé
```

```
{% endif %}
```

▪ Mettre des commentaires dans la syntaxe Twig

```
{# Commentaires... #}
```

Twig c'est aussi de nombreux atouts comme :

▪ l'héritage entre vues :

`{% extends "SdzBlogBundle::layout.html.twig" %}` : il a pour but de permettre à plusieurs vues de redéfinir à leur manière des blocs de vues déjà existants dans une vue mère. Il prend en argument la vue mère.

▪ L'inclusion `{% include "SdzBlogBundle::layout.html.twig" %}` des twig dans d'autres twig : c'est l'équivalent de la fonction « include() » en PHP. Il prend en argument la vue à inclure.

▪ Les filtres sur des variables « | » : `{{ var|upper }}` (mettre « var » en majuscule), `{{ var|length }}` (retourne le nombre d'éléments du tableau si `{{ var }}` est un tableau, et le nombre de caractères si `{{ var }}` est une

chaîne de caractères) , {{ var|date('d/m/Y') }} (formate la date {{ var }} suivant le format donné en argument)...

III. Le modèle (base de données)

1. Définition et rôle

Son rôle est de gérer les données et leur contenu. Au final, le contrôleur peut manipuler les objets sans toutefois savoir comment ils sont stockés, gérés, ... C'est une couche d'abstraction.

SYMFONY, pour se faire, utilise un **ORM (Object-Relational Mapper)** pour la gestion des bases de données. Dans sa version 2, l'ORM « **Doctrine2** » est déployé. Mais que fait concrètement un ORM ?

L'objectif d'un ORM est simple : se charger de l'enregistrement de données en faisant oublier que nous disposons d'une base de données. Fini donc les requêtes souvent très longues à construire, la création des tables manuellement, l'ORM se charge de tout cela. Ici comme dans tout SYMFONY, tout est objet. Doctrine ne déroge donc pas à la règle sauf qu'ici il y a un changement de vocabulaire : l'objet est désormais appelé « **entité** ». Nous notons aussi des notions telles « **persister** » (donner en charge une entité à Doctrine), « **flush** » (enregistrer une entité dans la base de données par Doctrine). Nous aurons donc les méthodes respectives « **persist(\$entity)** » et « **flush(\$entity)** ».

Imaginons un objet « user » avec plusieurs « formations professionnelles » (formation professionnelle ici est un objet) ; imaginons par la suite une méthode pour sauvegarder un « user » : au lieu de sauvegarder d'une part le « user », et d'une autre part ses « formations professionnelles » par de multiples requêtes, l'ORM est capable de sauvegarder en une seule ligne le user entier avec ses formations professionnelles.

```

<?php
// Depuis un contrôleur
Public function saveUserAction(){
    $f1=new FormationPro() ;
    $f2=new FormationPro() ;
    $user=new User() ;
    $user->addFormationPro( $f1) ;
    $user->addFormationPro($f2) ;
    // On récupère l'EntityManager
    $em = $this->getDoctrine()->getEntityManager();
    // Etape 1 : On "persiste" l'entité
    $em->persist($user);
    // Etape 2 : On "flush" tout ce qui a été persisté avant
    $em->flush();
}

```

Exemple : sauvegarde d'un utilisateur avec ses formations

2. Notion d'entité

Comment se crée les entités ? Les entités sous SYMFONY2 se créent aisément en console (ligne de commande) car SYMFONY2 c'est aussi son côté console pour plus de rapidité.

En effet, dans un terminal tout en s'étant au préalable assuré de l'installation de PHP sous notre serveur web, la création d'une entité utilise la commande :

php app/console generate:doctrine:entity et suivez le guide :

- The **Entity shortcut name**: grâce au commentaire juste au-dessus, il faut rentrer le nom de l'entité sous le format **NomBundle:NomEntité**. Dans notre cas, nous entrons donc ***OpenikaUserBundle:Article*** ;
- Configuration format (***yml, xml, php, or annotation***) [***annotation***]: comme précisé, nous allons utiliser les annotations qui sont d'ailleurs le format par défaut. Appuyez juste sur la touche Entrée ;
- **New field name** (press <return> to stop adding fields): nous commençons à saisir le nom de nos champs. Lisez bien ce qui est inscrit avant : Doctrine2 va ajouter automatiquement l'id, de ce fait, pas besoin de le redéfinir ici. Nous entrons donc notre nom : ***nom***;
- **Field type [string]**: c'est maintenant que nous allons va dire à Doctrine à quel type correspond notre propriété « date ». Voici la liste des types

possibles : array, **object**, **boolean**, **integer**, **smallint**, **bigint**, **string**, **text**, **datetime**, **datetimetz**, **date**, **time**, **decimal**, et **float**. Tapez donc **string** ;

Lorsque c'est fini avec tous nos attributs, appuyez sur la touche Entrée ;

- **Do you want to generate an empty repository class [no]? : oui**, nous allons créer le **repository** associé. Entrons donc **yes** ;

Confirmons la génération et pour la procédure est maintenant terminé.

A la fin du procédé, nous obtenons une classe « **NomEntité** » avec ses attributs définis, les getters et setters. Avant chaque attribut sont insérées des **annotations** qui ont pour but de caractériser ce dernier dans la base de données générée plus tard. Des annotations existent également pour caractériser l'entité même. Par exemple lui faire correspondre une table dans la base de données. Une entité est donc une classe avec des « commentaires ». Les annotations commencent toujours par « **@ORM** »

```
<?php
// src/Sdz/Bundle/Entity/Article.php

namespace Sdz\Bundle\Entity;

// On définit le namespace des annotations utilisées par Doctrine2
// En effet il existe d'autres annotations, on le verra par la suite, qui utiliseront un autre namespace
use Doctrine\ORM\Mapping as ORM;

/**
 * @ORM\Entity
 * @ORM\Entity(repositoryClass="Sdz\Bundle\Entity\ArticleRepository")
 */
class Article
{
    /**
     * @ORM\Column(name="id", type="integer")
     * @ORM\Id
     * @ORM\GeneratedValue(strategy="AUTO")
     */
    private $id;

    /**
     * @ORM\Column(name="date", type="date")
     */
    private $date;

    /**
     * @ORM\Column(name="titre", type="string", length=255)
     */
    private $titre;

    /**
     * @ORM\Column(name="contenu", type="text")
     */
    private $contenu;

    // les getters
    // les setters
}
```

Exemple d'entité générée

Dans l'exemple ci-dessus, nous avons donc un extrait de l'entité « Article », nous pouvons donc apercevoir certaines annotations avant la définition des attributs ; en voici quelques exemples :

- **@ORM\Column(name="id", type="integer")** : l'attribut "id" aura pour champ "id" dans la table générée pour l'entité "Article" dans la base de données.
- **@ORM\Id**
@ORM\GeneratedValue(strategy="AUTO") : cet attribut « id » sera clé primaire de la table et sera un champ qui s'auto-génère en s'incrémentant.
- **@ORM\Table(name= "article")** : souvent placée avant la définition de l'entité, elle donne un nom à la table qui sera générée plus tard pour l'entité.

3. La base de données

Les paramètres de configuration de la base de données sont gérés à l'aide d'un fichier situé dans : **app/config/parameters.ini**

C'est dans ce fichier que nous gérons le nom de la base de données, le mot de passe de l'utilisateur ayant accès, son mot de passe, le SGBD (driver) utilisé par PDO pour la connexion à celle-ci. Les commandes ci-après se chargent tour à tour de la création de la base de données, la génération des tables à l'intérieur, l'exécution des requêtes SQL pour le processus entier, la mise à jour de cette dernière (deux dernières requêtes).

- **php app/console doctrine:database:create**
- **php app/console doctrine:schema:update --dump-sql**
- **php app/console doctrine:schema:update --force**
- **php app/console doctrine:schema:update --dump-sql**
- **php app/console doctrine:schema:update --force**

4. Les services sous SYMFONY2

SYMFONY2 ce sont aussi de nombreux services (classes remplissant des fonctions bien définies et accessibles partout dans notre code). Nous y retrouvons :

a. Doctrine et L'EntityManager

Le service Doctrine est celui qui va nous permettre de gérer la persistance de nos objets. Ce service est accessible depuis le contrôleur comme n'importe quel service. Ce service qui va nous permettre de gérer la base de données, notamment :

- Les différentes connexions à des bases de données. Nous pouvons décider tout à fait utiliser plusieurs connexions à plusieurs bases de données différentes. Cela n'arrive que dans des cas particuliers. Le service Doctrine dispose donc, entre autres, de la méthode `<?php $doctrine->getConnection($name)` qui permet de récupérer une connexion à partir de son nom. Cette partie **DBAL** permet à Doctrine2 de fonctionner sur plusieurs types de SGBDR, tels que MySQL, PostgreSQL, etc.
- Les différents gestionnaires d'entités, ou **EntityManager**. C'est la partie **ORM** de Doctrine2. Encore une fois, vous pouvez utiliser plusieurs gestionnaires d'entités, ne serait-ce qu'un par connexion. Le service dispose donc, entre autres, de la méthode dont nous nous servirons beaucoup :
`<?php $doctrine->getEntityManager($name)` qui permet de récupérer un ORM à partir de son nom.

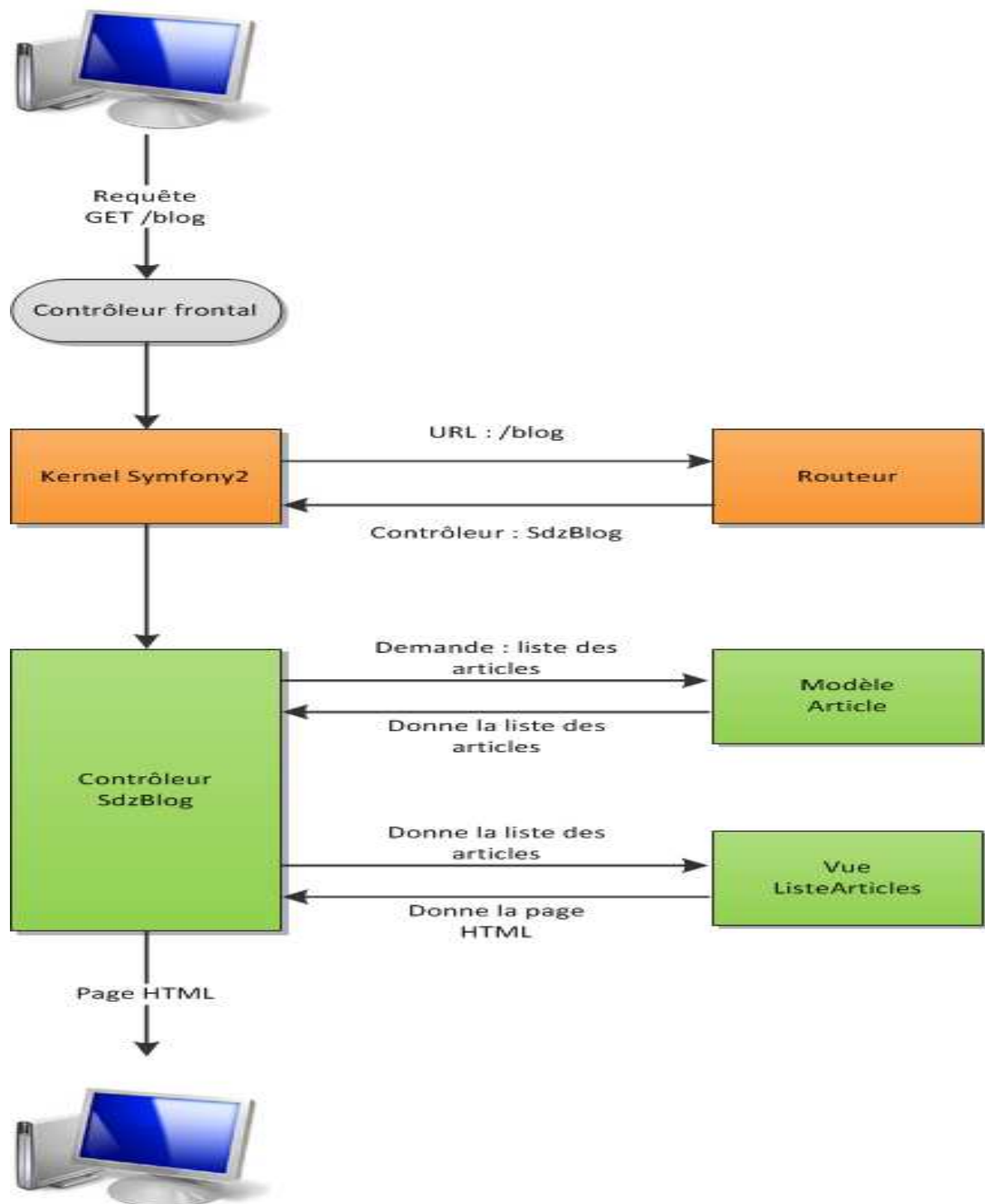
b. Le Repository

Ce sont donc les **Repository** qui nous permettront de récupérer nos entités. Il en existe toujours un par entité. Ainsi, pour charger deux entités différentes, il faut d'abord récupérer leur **Repository** respectif.

```
$em = $this->getDoctrine()->getEntityManager();
```

```
$repository_article = $em->getRepository('SdzBlogBundle:Article');
```

Doctrine c'est aussi la gestion des relations entre entités afin d'implémenter les concepts d'agrégation entre classes, de clés étrangères, ...



Exemple de fonctionnement d'une requête sous SYMFONY2 « source *siteduzero.com* »

IV. « Bundle » dans SYMFONY 2

Depuis le début de cette partie sur l'architecture logique de SYMFONY2, vous rencontrez le terme « **bundle** » assez souvent. En effet, un **bundle** est simplement un module de votre application, une brique. C'est à l'intérieur de ce dernier que l'on trouve les fichiers de routeurs, contrôleurs, les Template Twig, classes personnelles ... organisés dans des répertoires bien définis. Il se structure selon la vue ci après :

- **Controller/** | Contient nos contrôleurs.
- **DependencyInjection/** | Contient des informations sur le bundle (chargement automatique de la configuration par exemple).
- **Entity/** | Contient nos modèles.
- **Form/** | Contient les éventuels formulaires.
- **Resources/**
 - **config/** | Contient les fichiers de configuration de votre bundle (nous placerons les routes ici, par exemple).
 - **public/** | Contient les fichiers publics de votre bundle : fichiers CSS et JavaScript, images, etc.
 - **views/** | Contient les vues de notre bundle, les Template Twig.
- **Tests/** | Contient les éventuels tests unitaires et fonctionnels.

La création d'un bundle se fait aussi par la console à travers la commande :

"php app/console generate:bundle" . Grâce à l'assistant qui se lance par la suite, vous devez définir au bundle :

- **Un namespace (Sdz\ForumBundle par exemple) :** **Sdz** est le répertoire racine, **Forum** est le nom du bundle et **Bundle** est le suffixe obligatoire.
- **Un nom (SdzForumBundle par exemple) :** par convention, on nomme le bundle de la même manière que le namespace, sans les slashes.
- **Un lieu de destination des fichiers du bundle créé**
- **Un format de configuration (PHP, YML, XML...)...**

C. Quelques modules développés

I. Simulateur d'appariement sémantique

Il s'agit de la première version du futur JAE. Cet outil de back-office est destiné à effectuer les tests de matching en interne sur des profils face aux offres « OPENIKA » afin de trouver un paramétrage efficace pour les critères du futur JAE. Il (l'outil), a une interface avec un autre outil développé séparément : le **générateur visuel de graphes** : son but est de permettre le calcul de distances entre les connaissances acquises et requise au moment du matching.

Dans cet outil développé sous PHP5 (POO utilisée), quatre espaces ont été implémentés :

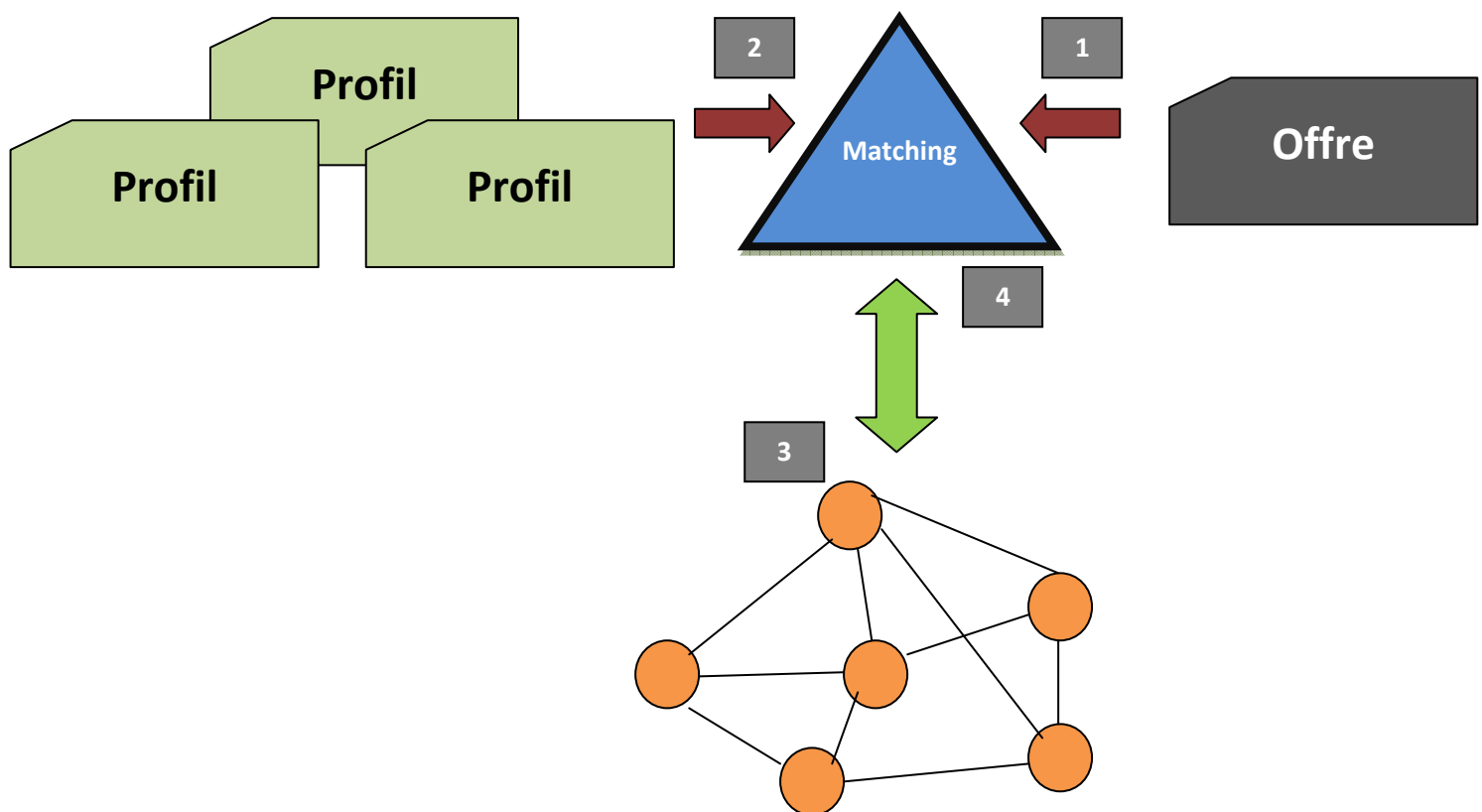
- **Espace candidat**
 - Création de profils.
 - Ajout des connaissances, savoir-être, savoir-faire requis.
- **Espace recruteur**
 - Création profils.
 - Création des offres (définition candidat idéal, connaissances, savoir-être, savoir-faire requis).
- **Espace Matching**
 - Choix du graphe à utiliser pour le matching.
 - Choix de l'offre.
 - Choix des profils candidat à matcher à l'offre.
 - Affichage des résultats.
- **Espace Administration**
 - Création du référentiel « OPENIKA » (métiers, compétences, savoir-être, savoir-faire, connaissances).

Quel est le fonctionnement de ce simulateur ? Le simulateur doit être au préalable configuré c'est-à-dire :

- Choix de l'offre à matcher avec les profils (1).
- Choix des profils (2).
- Choix du graphe de connaissances.

Une fois l'outil configuré, le calcul de matching peut être exécuté. Ce calcul requiert les fonctionnalités du graphe choisi précédemment (3) lors de la configuration de l'outil afin d'effectuer le calcul de distances entre connaissances. En clair, il s'agit ici pour ce simulateur de déterminer à quel degré telle connaissance se rapproche d'une autre, ou alors est similaire à telle autre connaissance.

Le calcul de distance effectué, le graphe renvoie ses résultats au simulateur (4) qui se chargera de les traduire sémantiquement par d'autres calculs sur des critères d'« OPENIKA ». Si les résultats ont lieu d'être affinés, une zone de paramétrage des coefficients alloués aux critères de Matching est disponible.



Vue d'ensemble du fonctionnement du Simulateur d'appariement sémantique

II. L'import de profil LinkedIn

D'abord développé en interne, ce module transformé en bundle SYMFONY, a ensuite été intégré au projet. Il s'agissait ici de se servir de l'**API** LinkedIn pour effectuer des requêtes **REST** (web services) afin de récupérer les informations de ce réseau professionnel pour qu'un utilisateur candidat ou recruteur puisse rapidement, remplir son profil « OPENIKA » avec ses données précédemment saisies sur LinkedIn.

Dans un premier temps, il a fallu créer une application sur LinkedIn afin de permettre au module de s'authentifier pour effectuer ensuite des requêtes sur le réseau LinkedIn. Cette procédure conduit à l'obtention de deux clés de connexion à l'API : une **clé API** et une **clé secrète API**. L'authentification à l'API peut alors débiter avec, dans l'ordre :

- Demande du REQUEST TOKEN en fournissant les clés API et d'autres paramètres tels que les différentes URL qui serviront à l'API à rediriger par exemple l'utilisateur vers une page d'authentification LinkedIn afin que lui-même autorise l'application à récupérer ses informations de LinkedIn pour « OPENIKA »
- Demande de l'ACCESS TOKEN afin de permettre à l'application d'effectuer des requêtes REST.

Rappelons ici que le protocole d'authentification utilisé est **oAuth**. Il a été employé grâce à l'extension **oAuth.so** déployée sur le serveur Apache2 qui, nous a ensuite fourni un ensemble de classes en native donc la classe « oAuth ».

Les requêtes REST pour l'API LinkedIn sont de la forme :

- **http://api.linkedin.com/v1/people/~:[champs à importer]** pour l'importation d'un profil candidat.
- **http://api.linkedin.com/v1/companies: [champs à importer]** pour l'importation d'un profil entreprise (recruteur)

Les champs à importer avaient le format ci-après (chaque champ séparé de son suivant par une virgule, le tout entre parenthèse) :

(first-name,last-name,headline,picture-url,id,main-address,...)

Les résultats sont formatés en XML. Pour traiter et afficher ces résultats, la classe « **simplexml_load_string** » de PHP a été employée afin de convertir une chaîne XML en objet PHP.

III. Le crawler

Le « **web crawling** » est une technique d'exploration récursive des pages web à travers leurs liens hypertextes afin de mémoriser des informations. Le crawler développé ici fut un outil de back-office visant, à partir des liens hypertextes tirés de linkedin.com, à récupérer les connaissances déjà répertoriées dans ce réseau pour la construction du référentiel d'OPENIKA dans un premier temps, puis plus tard des différents graphes de connaissances.

Plus concrètement, l'outil présent à partir d'un lien de départ, effectue un parcours récursif d'autres liens. A partir de chaque lien indexé, il récupère la connaissance LinkedIn grâce à des **regex** conçues à cet effet. La connaissance est ensuite stockée dans la base de données avec le lien pour y accéder et le lien parent qui a permis d'explorer le lien menant à celle-ci. La bibliothèque **cURL** de PHP à travers l'extension **php_curl** a été utilisée pour se faire.

IV. Autres modules

Plusieurs autres modules (bundles) ont été développés dans le cadre de ce projet et notamment sous SYMFONY2 donc :

- **Le Chat** : module réalisé pour permettre aux membres d'OPENIKA de disposer d'une messagerie instantanée.
- **La campagne de recrutement** : module réalisé pour mettre en avant côté recruteur la marque employeur qu'il véhicule. Il comprend d'autres sous-modules tels :
 - **Le TchatLive** : « chat » interactif permettant au recruteur de discuter (répondre aux questions des adhérents de la campagne) avec les internautes sur différents thèmes de campagne.

- **Le Forum** : espace permettant aux adhérents de la campagne de poser leurs problèmes dans le but de trouver des solutions.

Expérience acquise

Durant ces six derniers mois, il ressort une très grande satisfaction des efforts accomplis, mais aussi la fierté d'avoir pu bénéficier de plusieurs aptitudes et savoir-faire notamment concernant le travail en équipe, la maîtrise du Framework SYMFONY2. Nous avons aussi:

- vécu de près les mécanismes qui entourent la mise sur pied d'une start-up,
- assisté aux réunions avec les investisseurs afin d'acquérir certains rouages et méthodes marketing,
- exposé nos travaux de projet devant des personnalités par forcément du monde technique,
- appris à traduire techniquement les besoins d'un client dits en ses propres termes et non en les nôtres...

Glossaire

AJAX Asynchronous Javascript And Xml

API Application Programming Interface

B2B Business To Business

CSS Cascading Style Sheets

JAЕ Job Affinity Engine

JRM Job Relationship Management

JSN Job Social Network

JSON JavaScript Object Notation

MERISE Méthode d'Etude et de Réalisation Informatique pour les Systèmes d'Entreprise

ORM Object-Relational Mapper

PDF Portable Document Format

PDO PHP Data Object

PHP PHP Hypertext Preprocessor

POO Programmation Orientée Objet

SGBD Système de Gestion de Bases de Données

SQL Structured Query Language

UML Unified Modeling Language

URL Uniform Resource Locator

WWW World Wide Web

XHTML eXtensible HyperText Markup Language

XML eXtensible Markup Language

Conclusion générale

Les réseaux sociaux et professionnels sont plus que jamais incontournables à l'heure où 'on parle lorsqu'on évoque le web 2.0. Ils le sont également dans un processus où les frontières et les distances se veulent effacées afin de permettre le rapprochement entre individus ou entités dans un sens plus large. Le « **Matching** » comme le « **Ranking** » sont donc appelés à être de plus en plus employés comme notions ; de technologies nouvelles ou actuellement perfectionnées suivent et suivront le pas pour plus de cohérence dans les résultats, et pour la rapidité à travers la conception d'algorithmes à cet effet.

Le présent projet a permis de mettre en lumière le concept de théorie des graphes avec le calcul de distances dont se sert le JAE pour rapprocher des profils de candidats à ceux des recruteurs à travers leurs offres publiées. Il a aussi permis (**Matching & Ranking**), d'associer au monde de l'emploi des technologies qu'on retrouve déjà depuis quelques temps dans le monde des sites de rencontres par exemple, ce qui montre au final l'étendue de sa portée.

Au terme de ce document, nous noterons que le passage par une entreprise lors de l'apprentissage d'un métier est une nécessité étant donné que la théorie ne permet pas de cerner toutes les difficultés (réalités) des problèmes posés. Malgré de quelques difficultés, nous pouvons affirmer que nous sommes sortis enrichis de cette expérience.

Nous pouvons relever entre autre la découverte d'astuces, l'apprentissage de nouvelles méthodes de programmation, la négociation de marché, la gestion de ressources (hommes, matériel, argent, temps), la prise de décision. Toutefois, des efforts devraient être faits pour maîtriser les techniques apprises et suivre les avancées technologiques. Nous tenons donc enfin à remercier toute l'équipe OPENIKA pour leur soutien sans faille tout au long de notre passage au sein de la structure, notre suiveur UTBM pour avoir toujours été à l'écoute lorsque nous avons des interrogations sur le déroulement du stage, ce présent document est en quelque sorte aussi leur résultat.

Il est à noter que notre travail a été confronté à la réalité du marché, marché qui est, rappelons-le, extrêmement atomisé de par la présence de nombreux acteurs historiques. Les premiers résultats sont plutôt élogieux. En deux mois de démarches commerciales, un portefeuille d'affaires de près d'un

million d'euros a été développé. Parmi ces dossiers, nous pouvons notamment citer de prestigieux prospects comme le Ministère de la Défense (13 000 recrutements), *Manpower* (600 agences, 3200 recruteurs) ou encore *Randstad* (500 agences). Chaque retour clientèle fait l'objet d'une analyse et de correction si besoin est, afin de coller aux exigences du marché. Nous pouvons également souligner qu'une commande de 13.000 euros a également été enregistrée, alors que le site n'ouvrira... que le 10 septembre.

Documentation

Ouvrages :

- *Linux Compétence Micro Le plaisir d'apprendre*, Numéro 7 Avril/Mai 2009.
- *PHP 5 avancé*, Eric DASPET, Cyril Pierre DE GEYER, 4ème Edition, EYROLLES.

Ressources internet :

- Tutoriel « *un tutoriel pour débiter avec le Framework Symfony2* » par winzou, www.siteduzero.com
- www.lafermeduweb.com , Tutoriel « *Bien débiter avec le Framework Symfony2* »
- www.developpez.com
- www.codes-sources.com

Annexes

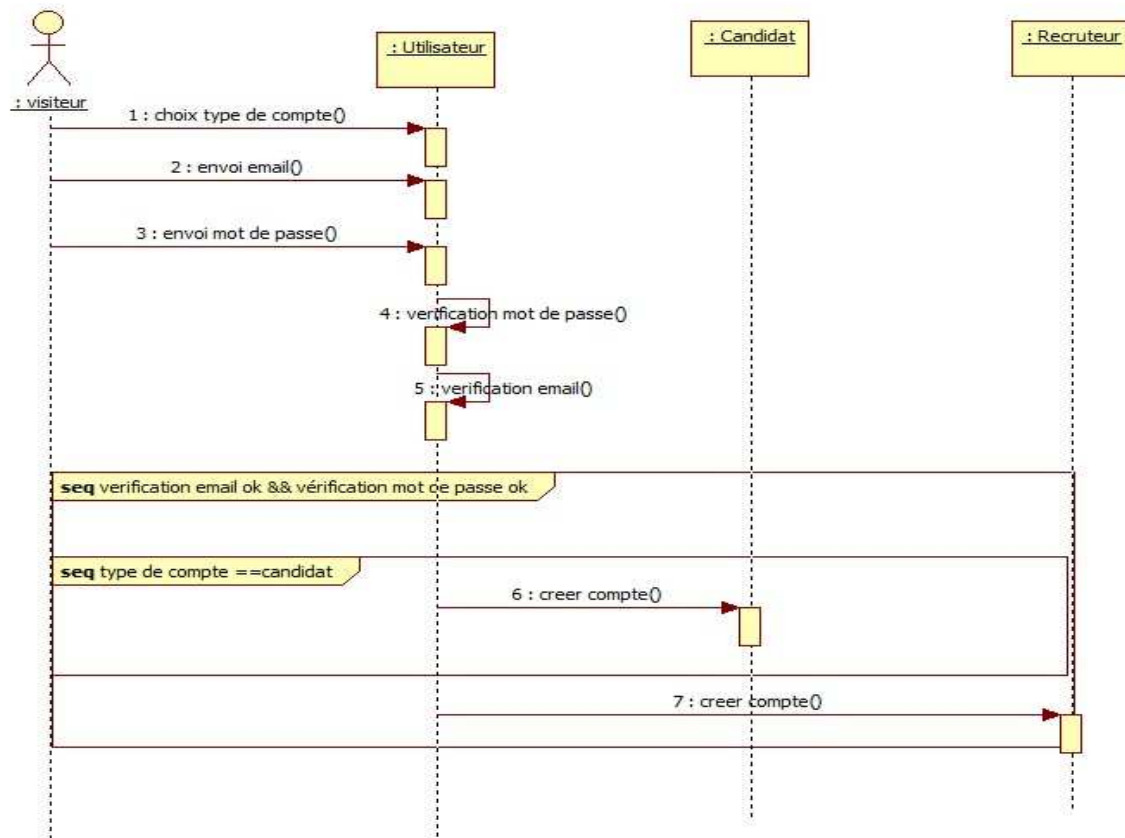


Diagramme de séquence : création compte membre sur « OPENIKA »

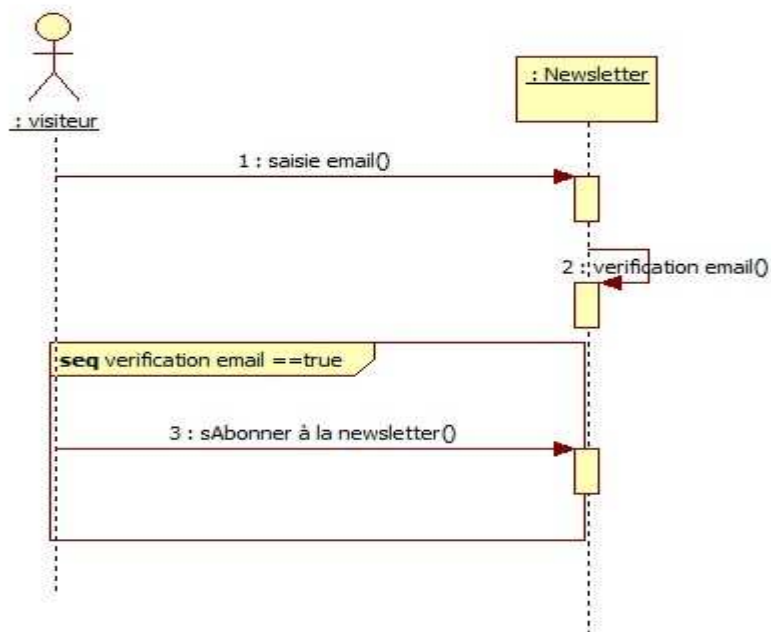


Diagramme de séquence : inscription à la newsletter sur « OPENIKA »

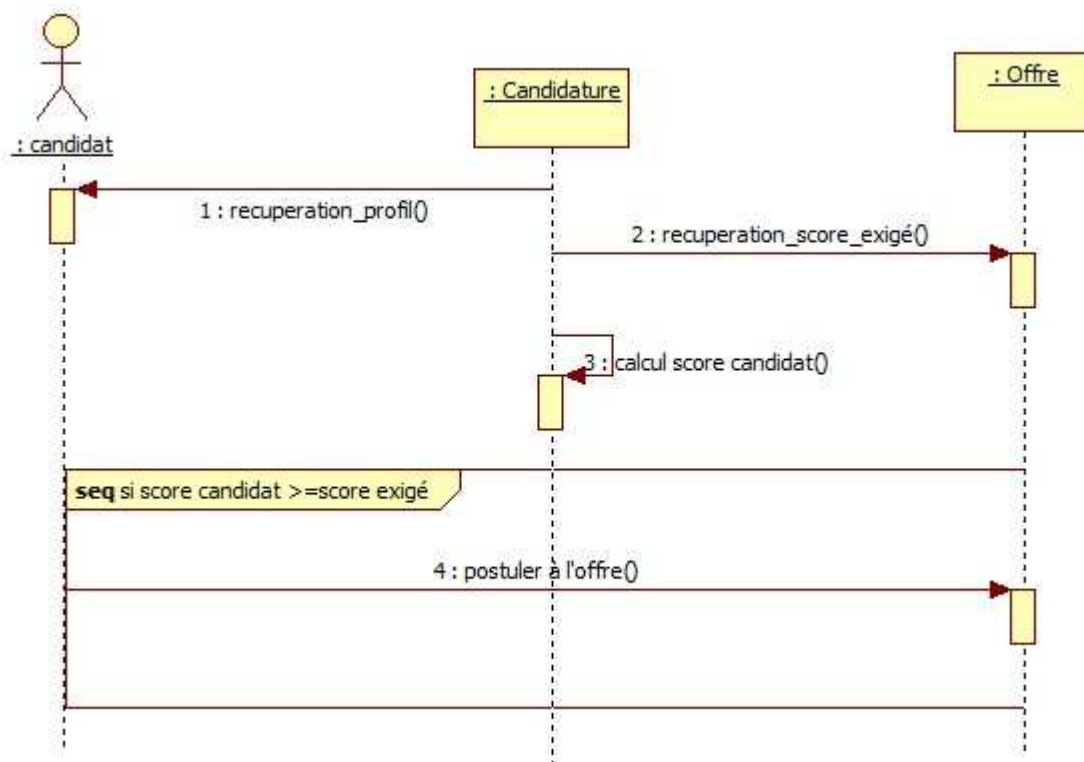


Diagramme de séquence : candidature à une offre d'emploi sur « OPENIKA »

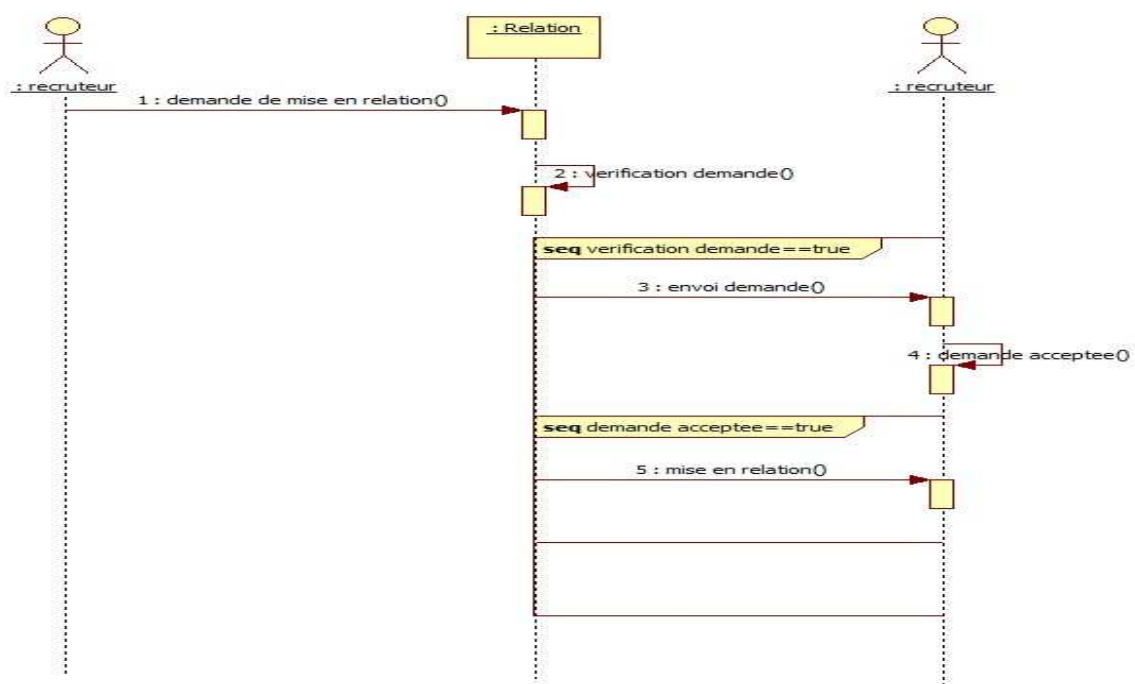
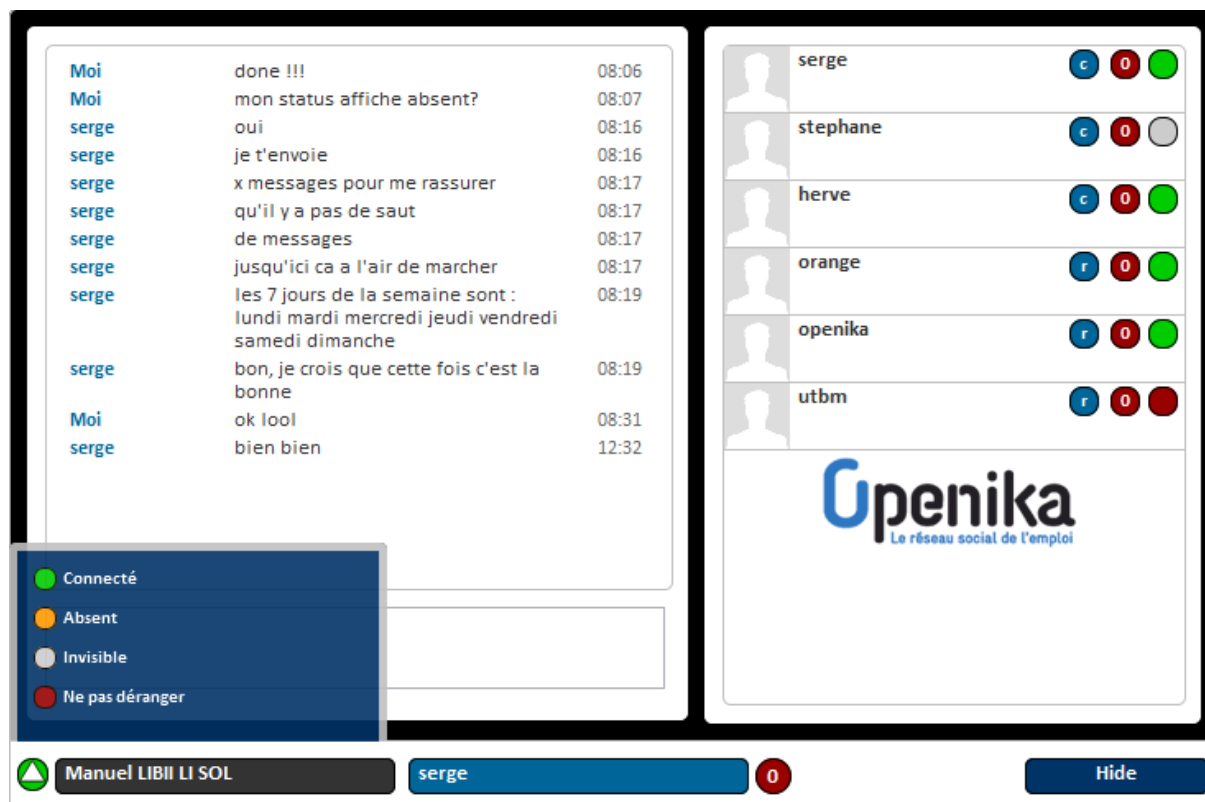
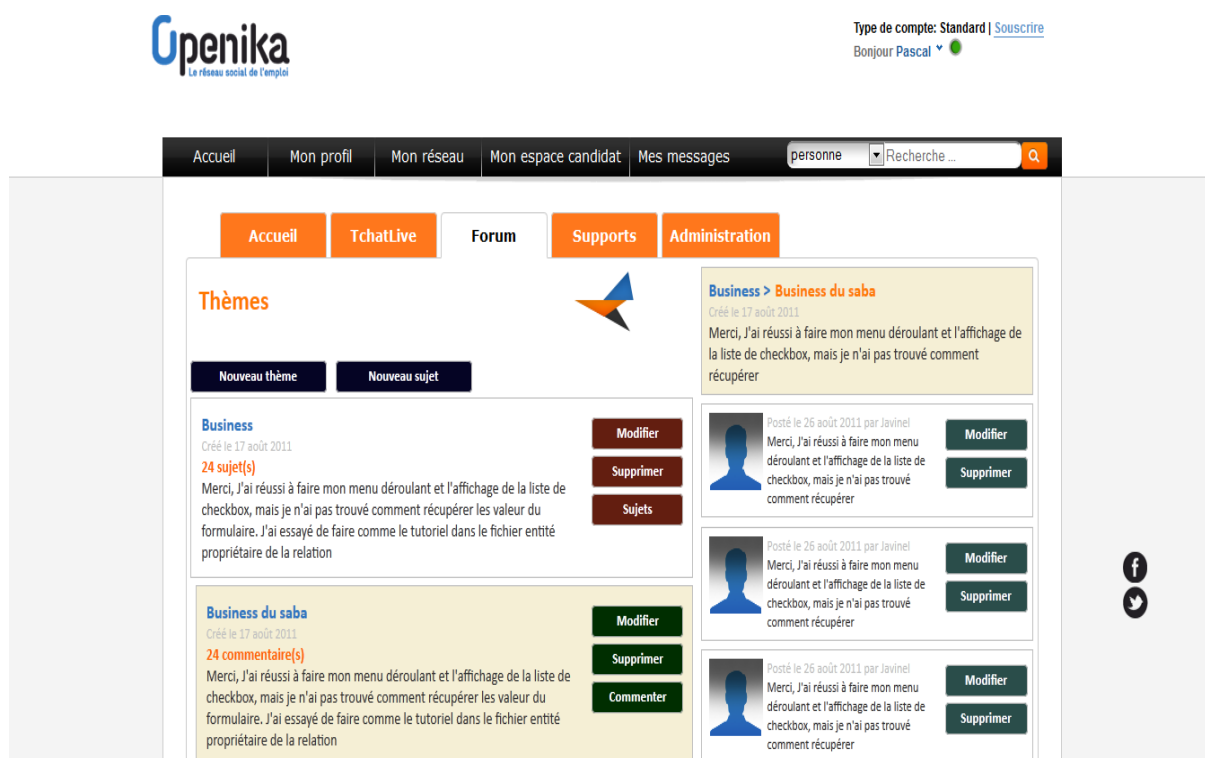


Diagramme de séquence : mise en relation entre recruteurs



Aperçu du module « chat » d' « OPENIKA »



Aperçu du module « forum » - espace campagne de recrutement d' « OPENIKA »

Offre	LIBI LI SOL	Lehrer BOYET
Exigences	21.51 %	23.30 %
Connaissances	91.14 %	73.82 %
Méthodes [SF]	100.00 %	100.00 %
Attitudes [SE]	100.00 %	100.00 %
Score Total	89.95 %	84.75 %

Shell	Shell 100.00 %	discuter 60.00 %
vba	vba 100.00 %	vba 100.00 %
discuter	discuter 100.00 %	discuter 100.00 %
négociier	vba 80.00 %	vba 80.00 %

Paramètres Matching

SCAVB 0.8 ▼

NDC 39 ▼

NDN 4 ▼

EAE 9 ▼

ENE 5 ▼

CMA 17 ▼

CMCD 11 ▼

Aperçu du Simulateur d'Appariement Sémantique : présentation des résultats du Matching

Openika
Le réseau social de l'emploi

Bonjour **Pascal** ▼

Accueil Mon profil Mon réseau Mon espace recruteur Mes messages Personnes Recherche...

Accueil > Profil



Ernestine-Armandine d'avout d'aerstadt
15 rue de la crête de la revinière
67500 Niederschaeffolsheim
FRANCE
01 34 81 08 79
06 42 23 88 57
slangarom@hotmail.fr

actualités book références

vos notes historique

Message laissé mercredi soir
A rappeler
Premier contact le 21/4/12

modifier ajouter supprimer

Le salon des Grandes Écoles
11 & 20 NOVEMBRE 2011
PALAIS BRONCKART - SOURCE DE PARIS

SUIVI d'un candidat

DIRECTEUR COMMERCIAL, 10 ans d'expérience dans le secteur TELECOM/HIGH TECH
(Enregistrement et écoute téléphonique, Visioconférence)

AFFINITÉ 82% CLASSEMENT 7#

Statut de la candidature :

SON HISTORIQUE : **entretien passé** : 03/07/11 **mail reçu** : 23/09/11 **réponse négative envoyée** : 23/09/11

formation

Sup de Gs, Paris, de Septembre 2006 à Juin 2009, ingénieur commercial (Diplôme obtenu)
INSEAD, Fontainebleau, de Septembre 2009 à Juin 2010, MBA Administration Générale des entreprises (Niveau)

entreprises

AREVA, La Défense, Secteur Energie, de Mai 2009 à Juillet 2012 (2 ans 2 mois), ingénieur commercial
POLYCOM, La Défense, Secteur IT, de Mai 2008 à Mai 2009 (1 an), ingénieur commercial

connaissances
compétence technique/métier
compétence comportementale
prix & distinction

PRIX DU MEILLEUR MANAGER AREVA 2011

prix & distinction

LES ANCIENS DE SUP DE CO (www.exsupdeco.com)

envoyer un message
 chatter
 planifier une visioconférence
 signaler un abus

© 2012 Openika | Qui sommes nous | Mentions Légales | Conditions Générales d'utilisation | Politique d'utilisation des données personnelles/Politique de confidentialitéPublicité | Contactez nous | Relations presse | FAQ
Publicité | Contactez nous | Relations presse | FAQ

Suivi d'un candidat dans l'espace gestion d'offre du recruteur

Openika
Le réseau social de l'emploi

Type du compte : Standard | [Souscrire](#)

Bonjour **Pascal** ▼

Accueil Mon profil Mon réseau Mon espace recruteur Mes messages Personnes Recherche...

Accueil > Résultats de recherche


RÉSULTATS de votre recherche

Vos critères de recherche


critère 1 **high resolution**

critère 1 **design**


[32] personnes correspondent à votre recherche




Hors Réseau
Affinité : 94%
acheter ce contact



Marc LUFFILLOL
Affinité : 80%
placer dans ma shortlist



Pierre BARBOZA
Affinité : 45%
placer dans ma shortlist



Cyril MERCIER
Affinité : 30%
placer dans ma shortlist

< 1 2 3 4 5 ... 10 20 30 >

© 2012 Openika | Qui sommes nous | Mentions Légales | Conditions Générales d'utilisation | Politique d'utilisation des données personnelles/Politique de confidentialitéPublicité | Contactez nous | Relations presse | FAQ
Publicité | Contactez nous | Relations presse | FAQ

Affichage des résultats de recherche de profils sur « OPENIKA » avec définition de critères

