Trimoz SDK

-- Document d'intégration --

Réalisé par :
Jean-Philippe Boily, ing.

Trimoz Technologies©
28 janvier 2013

Table des matières

1 Introduction

1.1 Présentation générale du produit

La compagnie « Trimoz Technologies » a développé un kit de développement (SDK) de sa technologie de référencement afin de procéder à l'intégration de façon « marque blanche » de celle-ci chez les clients désireux d'en retirer les bénéfices.

Grâce à ce kit de développement, il est possible d'ajouter les fonctionnalités de référencement brevetées par la compagnie à toute entreprise ou organisme possédant un logiciel de recrutement.

Le but du présent document est donc de faire la description de ce kit de développement tant dans son fonctionnement interne que dans ses interfaces en plus de faire état des requis pré-installation, des étapes d'intégration et de la configuration du système.

1.2 Définitions, acronymes, abréviations et conventions

SDK : Le terme *SDK* sera utilisé comme abréviation pour « kit de développement » et fera référence au « kit de développement » Trimoz faisant l'objet du présent document.

PHP : (PHP : Hypertext Preprocessor) Langage de scripts libre principalement utilisé pour produire des pages Web dynamiques via un serveur HTTP, mais pouvant également fonctionner comme n'importe quel langage interprété de façon locale, en exécutant les programmes en ligne de commande.[1]

SQL : (Structured Query Language) Langage informatique normalisé servant à effectuer des opérations sur des bases de données. La partie langage de manipulation de données de SQL

[1] Définition provenant du site web de Wikipedia http://fr.wikipedia.org/wiki/PHP

permet de rechercher, d'ajouter, de modifier ou de supprimer des données dans les bases de données.[2]

TRIIP : (Trimoz References Information Interchange Protocol) Protocol propriétaire permettant l'échange d'information sur les références entre un hôte et le serveur.

SOAP: (Simple Object Access Protocol) est un protocole de RPC orienté objet bâti sur XML. Il permet la transmission de messages entre objets distants, ce qui veut dire qu'il autorise un objet à invoquer des méthodes d'objets physiquement situés sur un autre serveur. 3

RPC : (Remote Procedure Call) est un protocole réseau permettant de faire des appels de procédures sur un ordinateur distant à l'aide d'un serveur d'applications. Ce protocole est utilisé dans le modèle client-serveur pour assurer la communication entre le client, le serveur et des éventuels intermédiaires.[4]

XML : (Extensible Markup Language, « langage de balisage extensible ») est un langage informatique de balisage générique qui dérive du SGML. Cette syntaxe est dite extensible car elle permet de définir différents espaces de noms, c'est-à-dire des langages avec chacun leur vocabulaire et leur grammaire, comme XHTML, XSLT, RSS… Elle est reconnaissable par son usage des chevrons (< >) encadrant les balises. L'objectif initial est de faciliter l'échange automatisé de contenus complexes (arbres, texte riche…) entre systèmes d'informations hétérogènes (interopérabilité).[5]

TCP/IP : La suite TCP/IP est l'ensemble des protocoles utilisés pour le transfert des données sur Internet. Elle est souvent appelée TCP/IP, d'après le nom de deux de ses protocoles : TCP (Transmission Control Protocol) et IP (Internet Protocol), qui ont été les premiers à être définis.[6]

Logiciel hôte : Par logiciel hôte on entend ici le logiciel ou site web du client chez qui le SDK sera déployé.

CSS : (Cascading Style Sheets : feuilles de style en cascade) est un langage informatique qui sert à décrire la présentation des documents HTML et XML. (Cascading Style Sheets : feuilles de style

[2] Définition provenant du siet web de Wikipedia http://fr.wikipedia.org/wiki/SQL
[3] Définition provenant du site web de Wikipedia http://fr.wikipedia.org/wiki/SOAP
[4] Définition provenant du site web de Wikipedia http://fr.wikipedia.org/wiki/Remote_procedure_call
[5] Définition provenant du site web de Wikipedia http://fr.wikipedia.org/wiki/XML
[6] Définition provenant du site web de Wikipedia http://fr.wikipedia.org/wiki/TCP/IP

en cascade) est un langage informatique qui sert à décrire la présentation des documents HTML et XML. [7]

ORM : (Object Relational Mapping) Technique de programmation informatique qui crée l'illusion d'une base de données orientée objet à partir d'une base de données relationnelle en définissant des correspondances entre cette base de données et les objets du langage utilisé. On pourrait le désigner par « correspondance entre monde objet et monde relationnel »[8]

URL : (Uniform Resource Locator) désigne une chaîne de caractères utilisée pour adresser lesressources du World Wide Web : document HTML, image, son, forum Usenet, boîte aux lettres électronique, entre autres. Les URL constituent un sous-ensemble des identifiants uniformisés de ressource (URI).[9]

1.3 Documents de référence

À compléter

[7] Définition provenant du site web de Wikipedia http://fr.wikipedia.org/wiki/Feuilles_de_style_en_cascade
[8] Définition provenant du site web de Wikipedia http://fr.wikipedia.org/wiki/Mapping_objet-relationnel
[9] Définition provenant du site web de Wikipedia http://fr.wikipedia.org/wiki/URL

2 Description générale du produit

2.1 Vue d'ensemble des fonctionnalités du produit

Cette section fait la description des fonctionnalités offertes par le SDK. À noter que la gestion des utilisateurs et des offres doit être faite par l'hôte. Le SDK ne fait que collecter certaines informations non confidentielles afin de faire une bonne gestion des références.

	1. Gestion des utilisateurs
1.1	Ajouter un utilisateur
1.2	Consulter un utilisateur
1.3	Modifier un utilisateur
1.4	Supprimer un utilisateur

	2. Gestion des offres
2.1	Ajouter une offre
2.2	Consulter une offre
2.3	Modifier une offre
2.4	Supprimer une offre
2.5	Fermer une offre

	3. Gestion des références
3.1	Ajouter une référence
3.2	Autoriser une référence
3.3	Vérifier une référence
3.4	Consulter une référence
3.5	Appliquer sur une référence
3.6	Modifier une référence
3.4	Supprimer une référence

	4. Gestion des transactions
4.1	Ajouter une transaction
4.2	Consulter une transaction
4.3	Modifier une transaction
4.4	Supprimer une transaction

5. Génération de statistiques	
5.1	Délai de recrutement
5.2	Candidature reçues
5.3	Emplois comblés
5.4	Emplois comblés par référence
5.5	Emplois offerts

6. Envoi de courriels	
6.1	Le candidat référé a été rejeté par l'employeur
6.2	Le candidat référé a été engagé par l'employeur
6.3	Le candidat référé a autorisé la référence
6.4	Le candidat a postulé
6.5	Une offre référée a été renouvellée
6.6	L'offre est terminée le candidat n'a pas consulté l'offre
6.7	La prime d'une offre référée a augmentée
6.8	X vous invite à appliquer sur une offre
6.9	Avis de fin de parution imminente (Host?)
6.10	Fermeture de l'offre requise
6.11	Rappel de fermeture de l'offre requise
6.12	Confirmation de fermeture d'une offre

7. Génération de tableaux et formulaires	
7.1	Formulaire de clôture d'une offre
7.2	Tableau des références reçues
7.3	Tableau des références envoyées
7.4	Formulaire de référencement

2.2 Pré-requis d'installation

Avant de procéder à l'installation du SDK, nous devons nous assurer que le logiciel hôte est en mesure de nous fournir certaines informations. Voici une liste des fonctionnalités qui doivent exister ou être développées avant l'installation du SDK.

- Le logiciel hôte doit permettre l'appel à des fonctions PHP
- Le logiciel hôte doit faire la gestion d'utilisateurs (employeur/chercheur)
- Le logiciel hôte doit faire la gestion d'offres d'emplois
- Le logiciel hôte doit permettre l'envoie de courriel
- Le logiciel hôte doit faire la gestion des applications (C.V., lettres de présentation)
- Le code source du logiciel hôte doit pouvoir être modifié afin d'intégrer les appels aux fonctions du SDK.

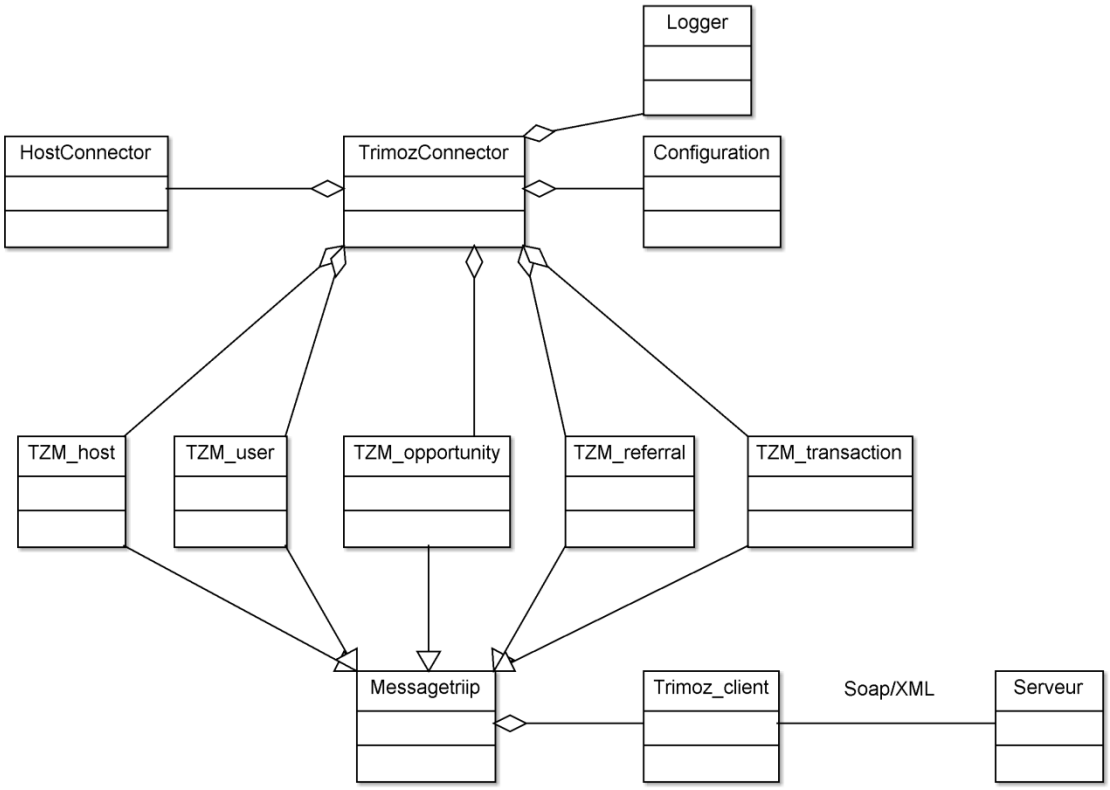
3 Description détaillée

3.1 Structure du logiciel

Le SDK est divisé en deux modules : le module client et le module serveur. La communication entre les deux modules s'effectue par l'échange de messages TRIIP en utilisant le protocole de communication SOAP. Une connection TCP/IP est utilisée pour faire le lien entre la machine client et la machine serveur.

3.1.1 Module client

Le module client du SDK est en charge de faire l'interface avec le logiciel hôte. Pour ce faire, le SDK offre des fonctions d'interfaces qui permettent de communiquer avec le serveur, des fonctions permettant l'affichage de tableaux et de formulaires reliés au référencement (voir section 3.1.3) ainsi que des fonctions permettant soit de recevoir le texte de courriels reliés au référencement ou d'envoyer ces courriels en utilisant les fonctionnalités du logiciel hôte. Voici une image représentant les objets formant la partie client du SDK :



Le point d'entrée du SDK est l'objet « TrimozConnector ». Cette objet est créé lors de l'appel de la fonction `tri_init()` (voir section 4.1.1) et est utilisé dans chacune des autres fonctions `tri_*()` disponible dans le fichier trimoz.php. Cet objet commence d'abord de charger la configuration du SDK. Ensuite, il initalise l'objet « HostConnector », utilisé par le SDK pour accéder aux informations du logiciel hôte (voir section 4.1.2). Il initialise l'objet « logger » qui sera utilisé pour avoir une trace des opérations dans le SDK. Finalement, il initialise les objets TZM_* qui sont utilisés pour faire la communication avec le serveur.

3.1.2 Module serveur

Le module serveur de son côté, reçoit les messages TRIIP en provenance du client. Il débute par décrypter les messages et s'assurer que l'hôte qui l'envoie possède bien les droits d'accès au serveur. Par la suite, il vérifie quel type de message il s'agit. Il peut s'agir d'une demande d'accès de données ou de modification de données. Le serveur fait les requêtes appropriées à la base de données puis retourne un simple message de succès ou d'erreur (dans le cas d'une requête en modification) ou les données demandées (dans le cas d'une requête en accession).

3.1.3 Formulaires et tableaux

Ici nous vous présentons les différents tableaux et formulaires qui sont fournis par le SDK. Il est à noter que le style tel que montré ici peut être modifié pour s'arrimer au style du logiciel hôte grâce au fichier CSS trimoz.css fournit avec le SDK. Une description de ce fichier se trouve à la section 5.3 du présent document.

Tableau des références envoyées :

Date	Reference	Candidate	Job Title	Reward	Status	
2012-10-10 11:42:33	Show	Candidate Name	test_6	$600	applied	delete

Tableau des références reçues :

Date	Reference	Referrer	Job Title	Reward	Status	
2012-10-10 11:42:33	Show	Referrer	test_6	$600	applied	delete

Formulaire de fermeture d'une offre :

Application List

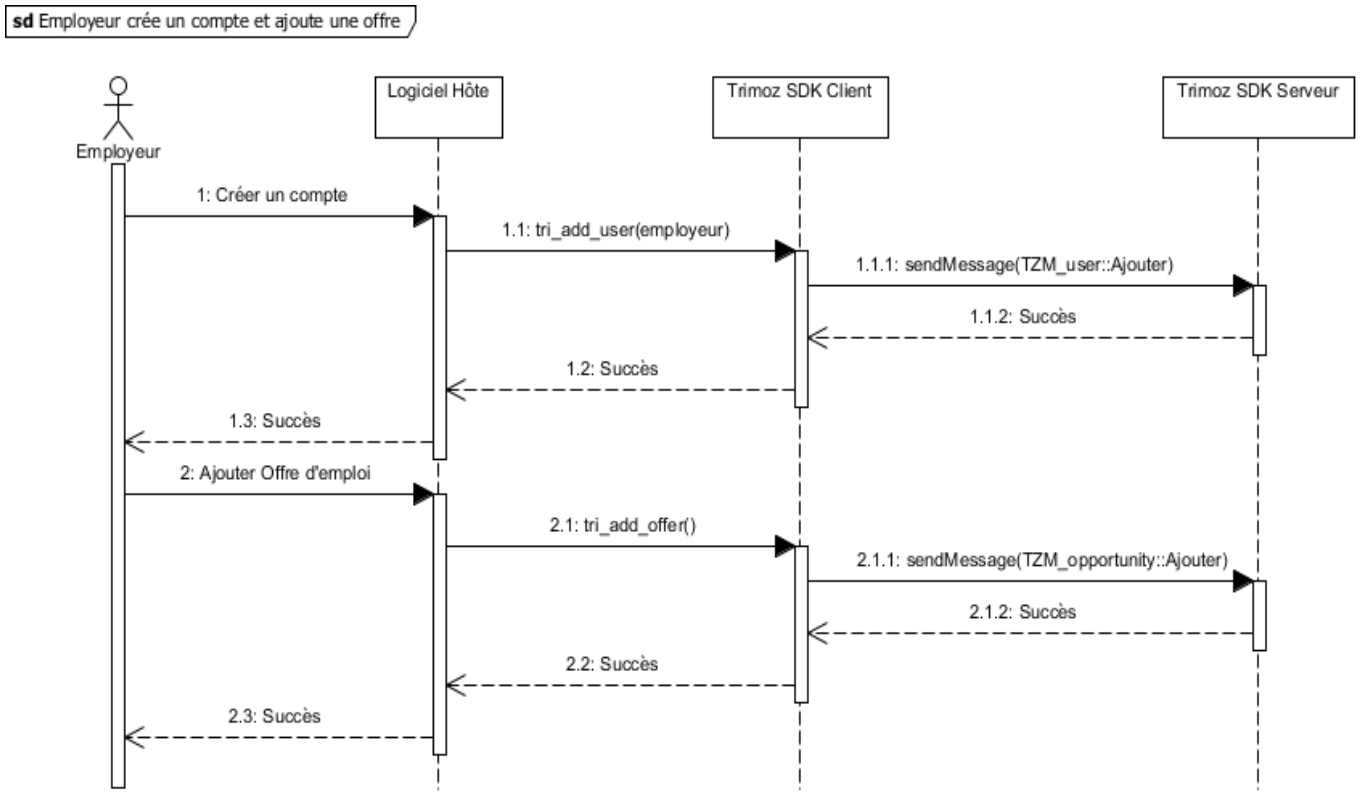
Date	Candidate	Referrer	Hired?
2012-10-16 15:45:18	Candidate	Referrer ***	○
2012-10-16 15:48:24	Candidat sans reference		○

Close Job

3.2 Étapes de référencement

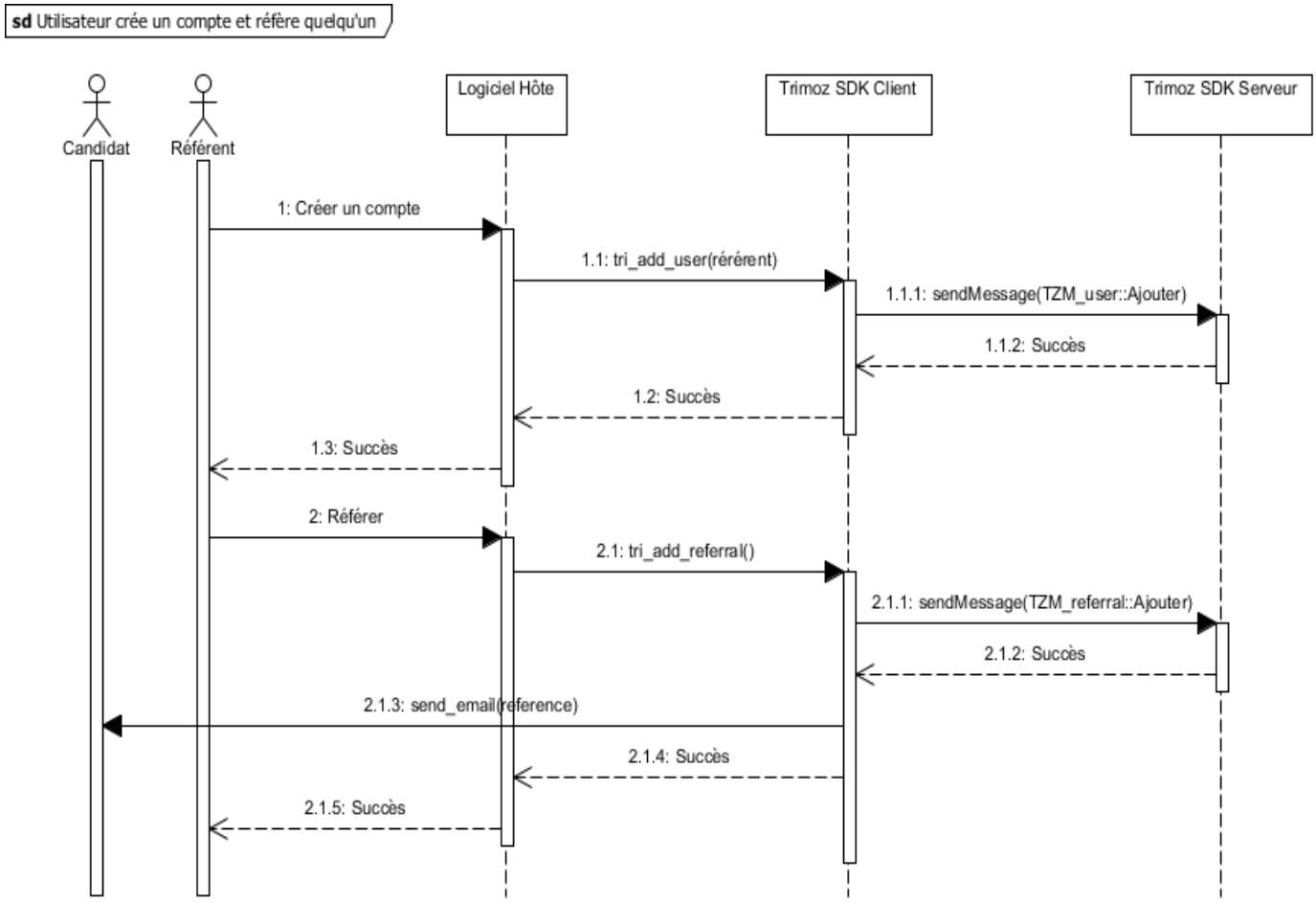
Afin de bien comprendre le processus de référencement, voici une liste des étapes de ce processus ainsi qu'un diagramme illustrant les actions prisent lors de chacune de ces étapes.

3.2.1 Employeur ajoute une offre



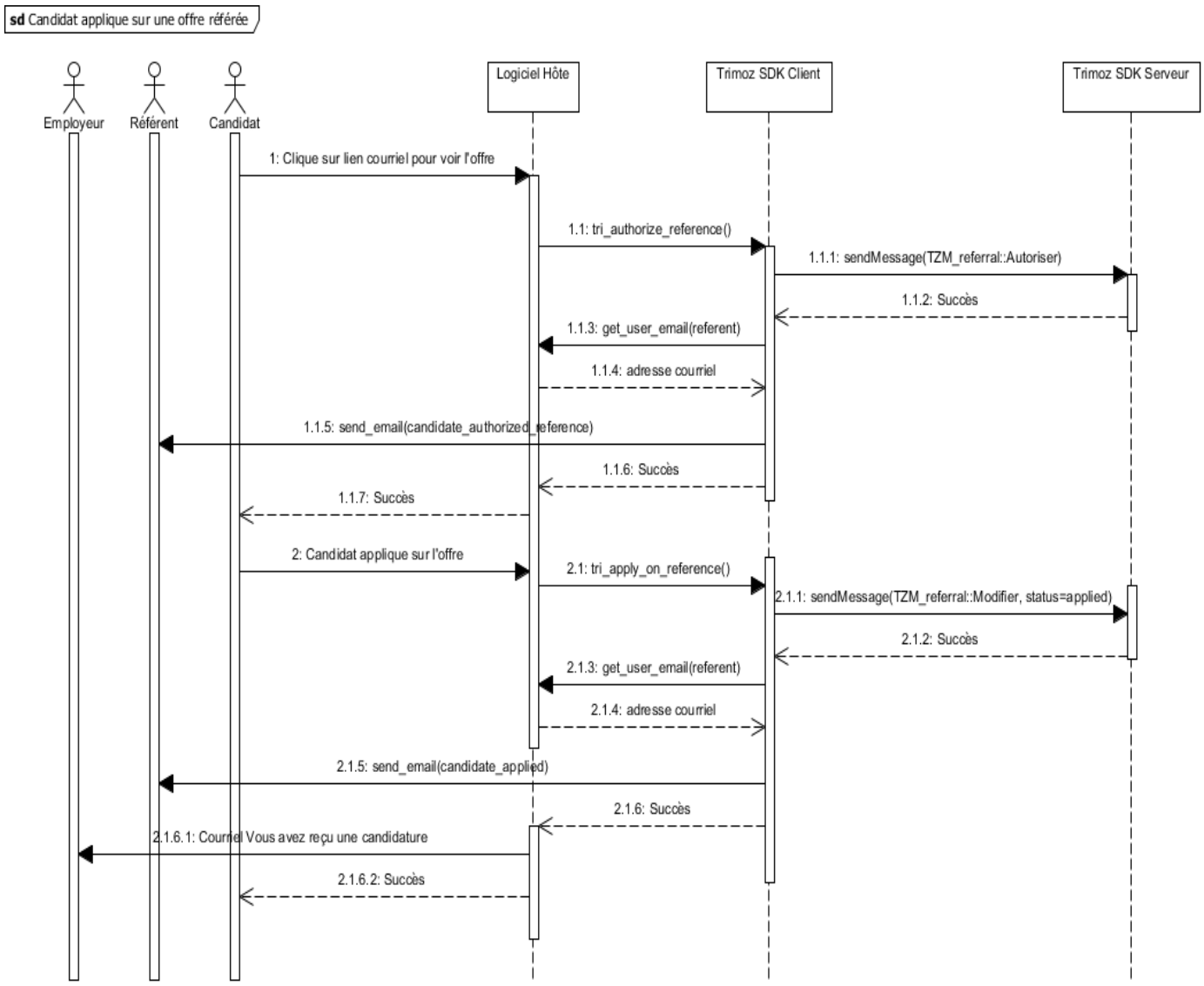
La première étape du référencement consiste à recevoir une offre d'un employeur. Celui doit au préalable avoir créé un compte dans le logiciel hôte, puis il ajoute son offre, toujours dans le logiciel hôte. Ce dernier effectue les appels au SDK qui permettent d'ajouter l'utilisateur nouvellement créé et la nouvelle offre dans la base de données du SDK du côté serveur.

3.2.2 Utilisateur réfère une offre



Lorsqu'une personne voit une offre et qu'elle pense connaître le candidat idéal pour cette offre, elle clique sur le bouton « Référer un candidat ». Si elle n'a pas de compte, elle doit en créer un. Si elle en a un, elle doit se connecter avec son compte. Ensuite, elle choisi par quel moyen elle veut rejoindre ce candidat (courriel, LinkedIN, Facebook, …). Une fois la méthode choisie, elle remplie le formulaire approprié et envoie l'information. Le logiciel hôte appel la fonction d'ajout de référence dans le SDK et celui-ci s'occupe d'ajouter l'information à la base de donnée et se charge de notifier le candidat qu'une personne lui recommande de postuler sur une offre. La notification se fait par courriel ou à travers les médias sociaux selon le moyen choisi par le référent.

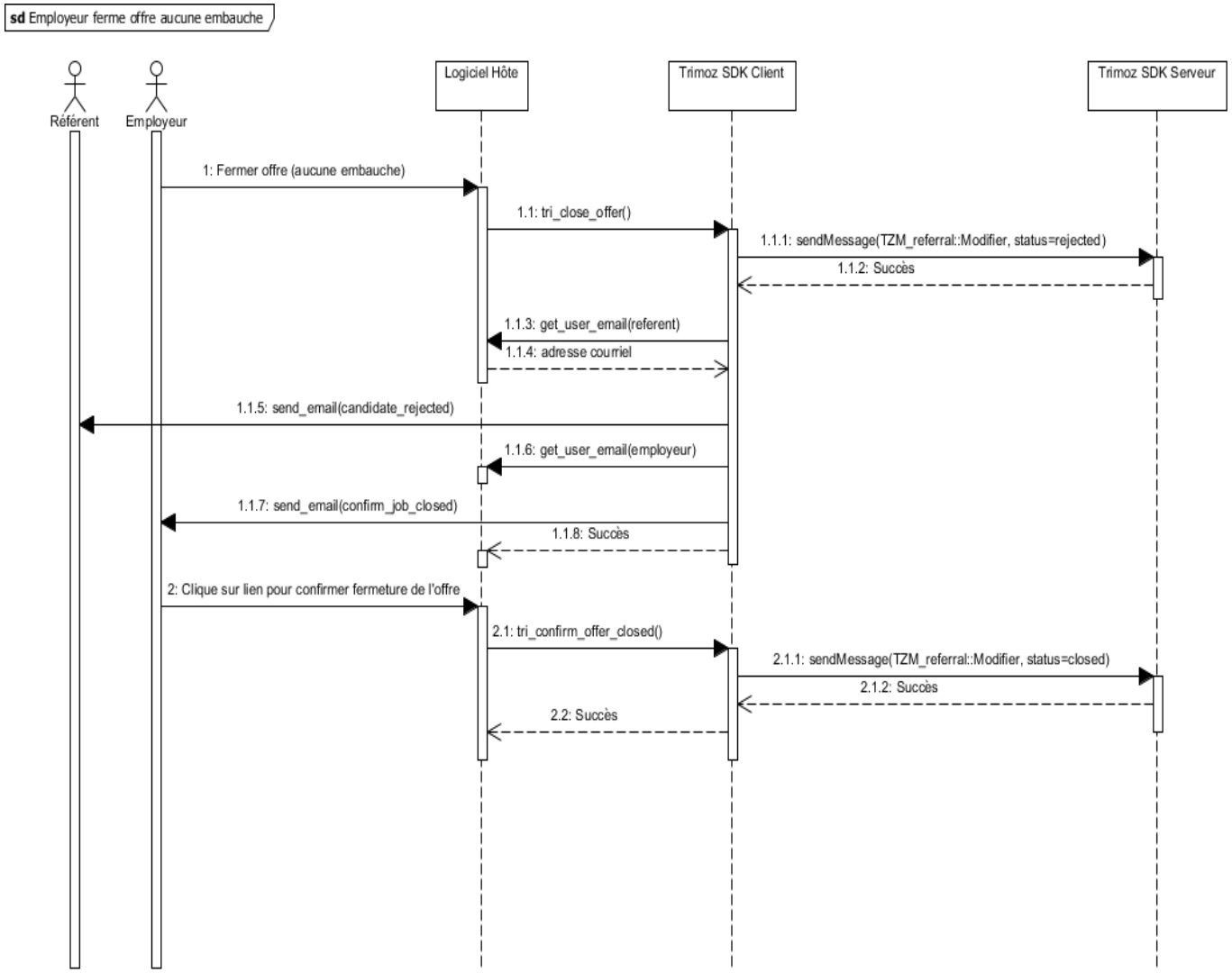
3.2.3 Candidat applique sur une offre référée



Lorsqu'un candidat reçoit un message l'informant qu'une personne l'invite à postuler sur une offre, celui-ci clique sur le lien inclus dans le message. Ce faisant, le logiciel hôte appel la fonction `tri_authorize_reference()` du SDK pour lui signalé que le candidat a autorisé la référence. Celui-ci modifie le status de la référence et envoie un courriel au référent pour l'avertir que le candidat a consulté l'offre. Lorsque le candidat applique, il doit créer un compte avec le logiciel hôte ou se connecter si c'est déjà fait. Le logiciel hôte envoie les informations du candidat au SDK en appelant la fonction `tri_apply_on_reference()`. Le SDK s'occupe de modifier le status de la référence dans sa base de donnée à « applied » puis envoie un courriel au référent pour lui signaler que son candidat à postulé.

Il est possible que le candidat vienne directement sur l'offre d'emploi sans passer par la référence. Dans ce cas, lors de la postulation, le logiciel hôte demande au SDMK si une personne avec les mêmes informations a déjà reçu une référence qui n'a pas été autorisée. Si c'est le cas (le nom et le courriel correspondent), le logiciel hôte doit informer le client qu'une ou des référence(s) coresspondant à ces informations a été trouvée dans le système et lui demander de confirmer si c'est bel et bien le cas et s'il veut utiliser autoriser la référence. Dans l'affirmative, afficher le nom du ou des référent(s) et lui demander de sélectionner celui qu'il veut autoriser. Ensuiste, le logiciel hôte fera un appel à `tri_authorize_reference()` avant l'appel à `tri_apply_on_reference()`.

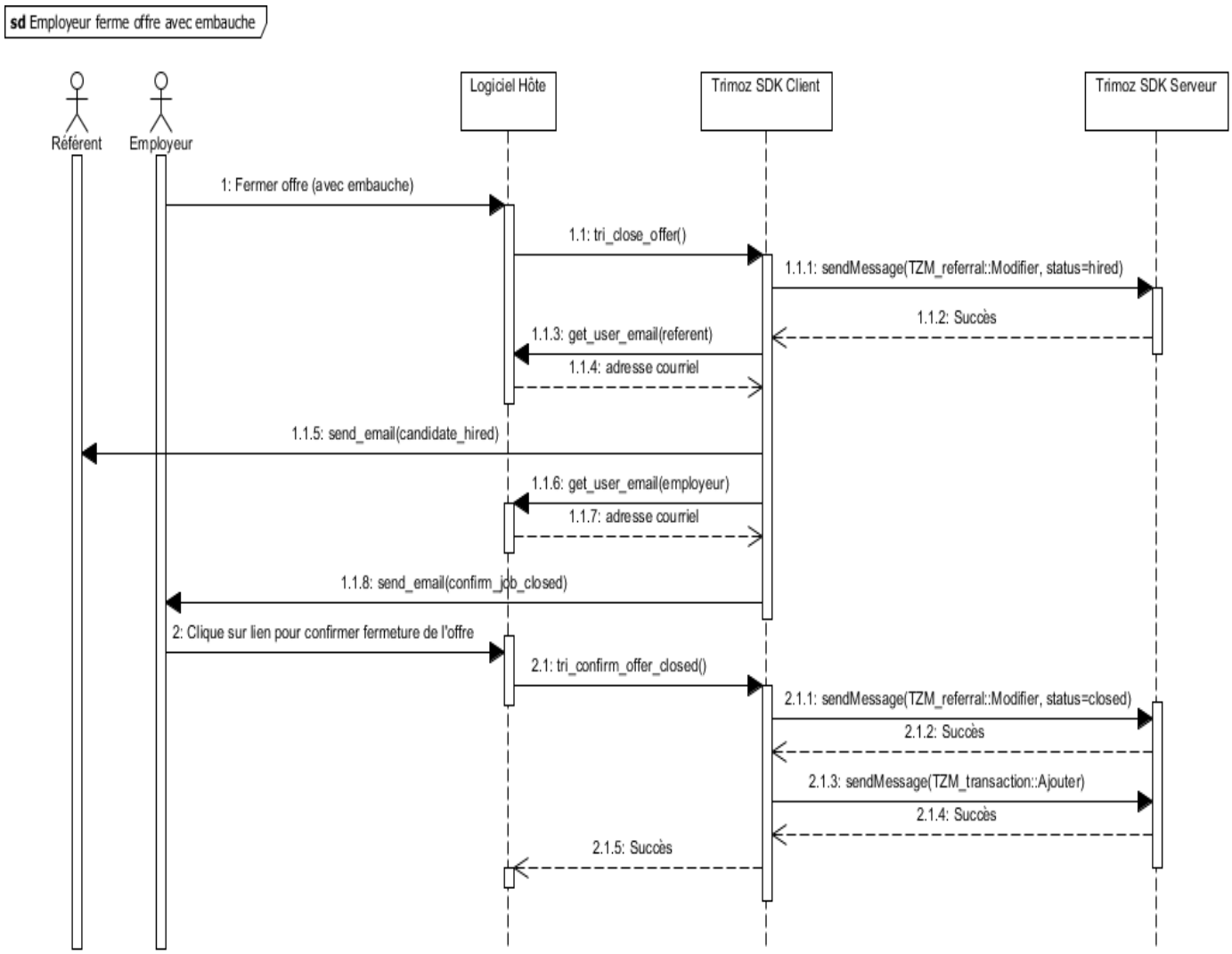
3.2.4 Employeur ferme une offre sans embauche



Une fois l'offre terminée, l'employeur doit aller dans la section « Fermez l'offre » ou la liste des candidats ayant appliqué est affichées. Il doit ensuite spécifier s'il a embauché quelqu'un ou non. Lorsqu'aucune embauche n'a été faite, toutes les références vont changer de status pour « rejected » et des courriels mentionnant que leur candidat n'a pas été retenu sera envoyé aux référents. Tout ce processus s'effectue par l'appel à la fonction `tri_close_offer()`. La fonction envoie également un courriel de confirmation de fermeture à l'employeur. Ce n'est qu'une fois qu'il aura cliqué sur le lien dans ce courriel que l'offre sera considérée comme fermée.

Il est aussi possible que l'employeur est embauché quelqu'un qui n'a pas appliqué par le logiciel hôte. Dans ce cas, une nouvelle référence sera créée sans qu'on puisse y voir le nom du candidat ou l'identifiant du référent. Cette référence « vide » servira lors du calcul de statistiques.

3.2.5 Employeur ferme une offre avec embauche



Lorsqu'une offre est fermée et qu'il y a embauche, le même processus s'applique qu'à la section précédente avec pour seule différence que la référence choisie par l'employeur se vera attribuée le status de « hire » par le SDK et un courriel signifiant au référent que son candidat a été embauché sera envoyé.

3.3 Organisation des fichiers

4 Interfaces et documents d'entrée/sortie

Dans cette section, nous détaillerons les interfaces disponbiles avec le SDK. La première partie consiste en la description des objets ORM (voir section Défintions) utilisé dans le SDK et qui doivent être utilisé par le logiciel hôte pour manipuler les données provenant de la base de données du SDK. Ensuite, il sera question de l'interface utilisée par le logiciel hôte pour communiquer avec le SDK (Interface SDK) et de l'interface utilisée par le SDK pour communiquer avec le logiciel hôte (Interface logiciel hôte). Nous allons également présenter les différents documents de configuration ainsi que les fichiers de « logging » générés par le SDK.

4.1 Description des interfaces

4.1.1 Interface « ORM »

L'utilisation d'objets ORM nous permet d'éloigner le plus possible le client du SDK (logiciel hôte) de la syntaxe SQL utilisée dans la base de donnée et de fournir des objets dont les attributs et les fonctions sont beaucoup plus simple à utiliser et beaucoup plus intuitifs. Voici donc les objets disponibles avec une description de leurs attributs et des méthodes spcéfiques à chacun.

Pour tous les objets ORM, on peut retrouver pour chacun des attributs une méthode pour accéder à la valeur de l'attribut (« getter ») et une méthode pour assigner uner valeur à l'attribut (« setter »). La méthode pour accéder à la valeur d'un attribut consiste toujours en son nom sans le '_' devant. Par exemple, pour accéder à l'attribut `_id`, on peut appeler la méthode `id()`. Pour assigner une valeur à un attribut, la méthode est toujours formée de `set_nom_attribut()`. Par exemple toujours pour _id, la méthode est `set_id()`.

En plus des ces méthodes d'accession et d'assignation, chacun des objets possède une méthode `init_with_db_obj()` et une méhtode `get_params()`. La première est une méthode static qui permet de générer un objet à partir des données reçues de la base de données du serveur. La deuxième génère un tableau d'éléments qui pourra être utilisé lors de l'envoie d'une commande au serveur.

4.1.1.1 Classe TriUser

Les objets TriUser servent à faire la représentation des utilisateurs enregistrés dans le serveur. Ils représentent à la fois les employeurs, les référents, les candidats et les administrateurs. La différentiation entre les types d'utilisateur se fait par l'attribut *_type*.

Liste des attributs de la classe TriUser :

Attributs	
_id	Identifiant unique de l'objet
_name	Nom de l'utilisateur
_type	Type d'utilisateur (employer, referrer,demander, admin)
_url	Adresse web du profil de l'utilisateur (optionel)
_reputation	Cote calculé par l'algorithme de Trimoz permettant de qualifier les référents.
_status	Statut de l'utilisateur (active ou inactive)

4.1.1.2 Classe TriOffer

Les objets TriOffer servent à faire la représentation des offres enregistrées dans le serveur.

Liste des attributs de la classe TriOffer :

Attributs	
_id	Identifiant unique de l'objet
_offerer_id	Identifiant de l'objet TriUser ayant créé cette offre
_title	Titre de l'offre
_description	Description de l'offre
_reward	Valeur de la prime attribuée au référent trouvant le candidat idéal
_reward_description	Description de la prime (optionel). Pourrait par exemple être des indications sur les conditions à respecter pour que la prime soit attribuée
_created_date	Date de création de l'offre
_expiry_date	Date d'expiration de l'offre
_status	Statut de l'offre (active ou inactive)

4.1.1.3 Clase TriReferral

Les objets TriReferral servent à faire la représentation des références enregistrées dans le serveur. Contrairement aux utilisateurs et aux offres, l'identifiant des références est généré par le SDK.

Liste des attributs de la classe TriReferral :

Attributs	
_id	Identifiant unique de l'objet
_referrer_id	Identifinat de l'objet Triuser ayant fait cette référence
_candidate_id	Identifiant de l'objet TriUser ayant postulé sur cette offre
_offer_id	Identifiant de l'objet TriOffer faisant l'objet de cette référence
_offerer_id	Identifiant de l'objet TriUser ayant créé l'offre faisant l'objet de cette référence
_confirmation_number	Numéro de confirmation unique pour un hôte permettant de faire la liaison entre un candidat et son référent
_full_name	Nom du candidat référé
_email	Adresse courriel du candidat référé
_relationship	Relation entre le candidat référé et le référent. Doit être une des valeurs autorisées dans la classe TriRelationship
_msg_personal	Message envoyé par le référent au candidat dans le courriel de référence
_msg_professional	Message envoyé à l'employeur en guise de référence lorsque le candidat référé applique sur l'offre
_initiated_by_id	Identifiant de l'objet TriUser ayant créé la référence
_status	Statut de la référence. Doit être une des valeurs autorisées dans la classe TriRefStatus.
_created_date	Date de création de la référence
_deactivated_date	Date de désactivation de la référence
_updated_date	Date de la dernière modification de la référence
_expiry_date	Date d'expiration de la référence
_authorized_date	Date à laquelle la référence a été autorisée par le candidat

Le champ _relationship et _status font l'objet de validation lors de l'assignation par les fonctions par les fonctions suivantes :

Méthodes	
validateRelationship	Vérifie que la valeur de l'attribut _relationship d'une référence fait partie des valeurs autorisées dans la classe TriRelationship
validateRefStatus	Vérfie que la valeur de l'attribut _status d'une référence fait partie des valeurs autorisées dans la classe TriRefStatus

À noter que c'est fonctions ne font pas partie de l'objet TriReferral mais se situe dans le même fichier.

4.1.1.4 Classe TriRelationship

Cette classe est une liste de constantes définissant les valeurs possibles pour le champ _relationship dans l'objet TriReferral.

- const Friend : ami
- const Student : étudiant
- const Colleague : collègue
- const Acquaintance : connaissance
- const Family : famille
- const ProfessionalContact : contact professionnel
- const Facebook : contact Facebook
- const LinkedIn : contact LinkedIn
- const Twitter : contact Twitter
- const GooglePlus : contact Google+
- const Other : autre contact

4.1.1.5 Classe TriRefStatus

Cette classe est une liste de constantes définissant les valeurs possibles pour le champ _status dans l'objet TriReferral

- const Pending : en attente d'autorisation
- const Authorized : autorisée
- const Cancelled : annulée
- const Applied : le candidat a postulé
- const Rejected : le candidat a été rejetté
- const Hried : le candidat a été embauché
- const Paid : le candidat a été payé
- const Closed : l'offre est fermée

4.1.1.6 Classe TriTransaction

Les objets TriTransaction servent à faire la représentation d'une transaction qui a été effectuée entre un employeur et un référent. Une nouvelle transaction est créée à chaque fois qu'un candidat est choisi alors qu'il a fait l'objet d'une référence sur une offre avec prime.

Liste des attributs de la classe TriTransaction :

Attributs	
_id	Identifiant unique de l'objet
_offer_id	Identifiant de l'objet TriOffer faisant l'objet de cette transaction
_offerer_id	Identifiant de l'objet TriUser ayant créé l'offre faisant l'objet de cette transaction
_value	Valeur de la transaction
_created_date	Date de création de la transaction
_status	Statut de la transaction (active ou inactive)

4.1.2 Interface SDK

Afin de bien intégrer le SDK dans le logiciel hôte, on se doit de bien comprendre les fonctions disponibles, leur utilité et leurs paramètres. Voici donc une liste de toutes les fonctions qui devraient être utilisées par le logicie hôte. Pour chacune de ses fonctions, nous vous présentons le nom de la fonction, sa description, sa valeur de retour et s'il y a lieu, la liste des paramètres que cette fonction requiert. L'arborsescence des fichiers ou se trouvent ces fonctions peut être consulté dans la section « Organisation des fichiers »

Fonction :	**tri_init**
Description :	Initialise le SDK. Crée l'objet TrimozConnector et s'assure des droits d'accès du logiciel hôte au serveur. DOIT ÊTRE APPELER AVANT TOUT AUTRE FONCTION.
Valeur de retour :	« true » si la création et la validation réussissent; « false » si non

!+! jpboily : **À REVOIR SI ON MODIFIE LA SIGNATURE, PAS BESOIN D'AJOUTER TOUT CES PARAMÈTRES**

Fonction :	**tri_add_user**
Description :	Ajoute un utilisateur à la base de données du serveur
Valeur de retour :	« true » si la création est réussie; « false » si non

Paramètre	**Entrée/Sortie**	**Description**
$user_id		

$name		
$type		
$description		
$url		

Fonction :	**tri_get_user**
Description :	Retourne un objet contenant l'information reliée à l'utilisateur dont l'identifiant est passé en paramètre
Valeur de retour :	Objet TriUser avec l'information si l'utilisateur est trouvé; null si non trouvé

Paramètre	**Entrée/Sortie**	**Description**
$user_id	Entrée	Identifiant unique de l'utilisateur à récupérer

Fonction :	**tri_update_user**
Description :	Modifie les informations d'un utilisateur dans la base de donnée dont l'identifiant correspond au id de l'objet passé en paramètre.
Valeur de retour :	« true » si l'opération a réussi; « false » si non

Paramètre	**Entrée/Sortie**	**Description**
TriUser $user	Entrée	Objet représentant l'utilisateur à être modifié.

Fonction :	**tri_del_user**
Description :	Désactive l'utilisateur dont l'id est passé en paramètre de la base de données.
Valeur de retour :	« true » si l'opération a réussi; « false » si non

Paramètre	**Entrée/Sortie**	**Description**
$user_id	Entrée	Identifiant unique de l'utilisateur à désactiver

tri_add_offer($offer_id, $user_id, $title, $description, $url, $reward_value, $expire_date)

!+! jpboily : **À REVOIR SI ON MODIFIE LA SIGNATURE, PAS BESOIN D'AJOUTER TOUT CES PARAMÈTRES**

Fonction :	**tri_add_offer**
Description :	Ajoute une offre à la base de données.
Valeur de retour :	« true » si la création réussie; « false » si non

Paramètre	**Entrée/Sortie**	**Description**
$offer_id		
$user_id	Entrée	Identifiant unique de l'utilisateur
$title		
$description		
$url		
$reward_value		
$expire_date		

Fonction :	**tri_get_offer**
Description :	Retourne un objet contenant l'information reliée à une offre dont l'identifiant est passé en paramètre
Valeur de retour :	Objet TriOffer avec l'information si l'utilisateur est trouvé; null si non trouvée

Paramètre	**Entrée/Sortie**	**Description**
$offer_id	Entrée	Identifiant unique de l'offre à récupérer

Fonction :	**tri_update_offer**
Description :	Modifie les informations d'une offre dans la base de donnée dont l'identifiant correspond au id de l'objet passé en paramètre.
Valeur de retour :	« true » si l'opération a réussi; « false » si non

Paramètre	**Entrée/Sortie**	**Description**
TriOffer $offer	Entrée	Objet représentant l'offre à être modifiée.

Fonction :	**tri_close_offer**
Description :	Débute le processus de fermeture de l'offre dont l'identifiant est passé en paramètre en modifiant le statut de chacune de ses références ainsi qu'en alertant par courriel les référents du changement d'état.
Valeur de retour :	« true » si l'opération a réussi; « false » si non

Paramètre	**Entrée/Sortie**	**Description**
$offer_id	Entrée	Identifiant de l'offre devant être fermée
$hired_ref_id	Entrée	Identifiant de la référence dont le candidat a été retenu. 0 si aucune candidature référée n'a été retenue.

Fonction :	**tri_del_offer**
Description :	Désactive l'offre dont l'id est passé en paramètre de la base de données.
Valeur de retour :	« true » si l'opération a réussi; « false » si non

Paramètre	**Entrée/Sortie**	**Description**
$offer_id	Entrée	Identifiant unique de l'offre à désactiver

tri_add_reference($refer_id, $offer_id, $offerrer_id, $initiated_by, $fullname, $referrer_id = NULL, $demander_id=NULL, $email='', $relationship='other', $personal_msg='', $professional_msg='', $status='pending') !+! jpboily : **À REVOIR SI ON MODIFIE LA SIGNATURE, PAS BESOIN D'AJOUTER TOUT CES PARAMÈTRES**

Fonction :	**tri_add_reference**
Description :	Ajoute une référence à la base de données.
Valeur de retour :	« true » si l'opération a réussi; « false » si non

Paramètre	**Entrée/Sortie**	**Description**

Fonction :	**tri_validate_reference**
Description :	Vérifie si une référence a déjà été effectuée pour ce candidat.
Valeur de retour :	« true » si aucune référence n'existe pour ce candidat; « false » si non

Paramètre	**Entrée/Sortie**	**Description**
$offer_id	Entrée	Identifiant de l'offre faisant l'objet d'une référence
$candidate_name	Entrée	Nom du candidat référé
$candidate_email	Entrée	Adresse courriel du candidat référé

Fonction :	**tri_authorize_reference**
Description :	Change le statut de la référence à « authorized » pour signifier au système que le candidat confirme avoir reçu la référence et désir consulter l'offre. L'information sur la référence (id du référent et numéro de confirmation) est extraite en décodant le lien envoyé dans le courriel au candidat.
Valeur de retour :	« true » si l'opération a réussi; « false » si non

Paramètre	**Entrée/Sortie**	**Description**
$email_ref_link	Entrée	Lien envoyé dans le courriel du candidat qui doit être décodé pour extraire l'identifiant du référent et le numéro de confirmation de la référence.

Fonction :	**tri_find_reference**
Description :	Essaie de trouver la référence correspondant aux critères spécifiés dans l'objet TriReferral passé en paramètre et retourne l'objet complet si trouvé.
Valeur de retour :	« true » si la référence a été trouvée; « false » si non

Paramètre	**Entrée/Sortie**	**Description**
TriReferral $ref	Entrée et Sortie	Objet contenant les paramètres de recherche de la référence. Cet objet est complété avec les informations de la référence si trouvée et retournée par référence.

Fonction :	**tri_apply_on_reference**
Description :	Modifie la référence passée en paramètre pour lui assigner l'identifiant du candidat ayant appliqué et changer sont statut pour « applied ». Envoie également un courriel au référent pour lui signaler que le candidat a appliqué.
Valeur de retour :	« true » si l'opération a réussi; « false » si non

Paramètre	**Entrée/Sortie**	**Description**
TriReferral $ref	Entrée	Objet contenant les informations de la référence.
$user_id	Entrée	Identifiant de l'utilisateur appliquant sur l'offre.

Fonction :	**tri_update_reference**
Description :	Modifie les informations d'une référence dans la base de donnée dont l'identifiant correspond au id de l'objet passé en paramètre.
Valeur de retour :	« true » si l'opération a réussi; « false » si non

Paramètre	**Entrée/Sortie**	**Description**
TriReferral $ref	Entrée	Objet représentant la référence à être modifiée.

Fonction :	**tri_get_referrer_ratio**
Description :	Retourne la cote de l'utilisateur dont l'id est passé en paramètre.
Valeur de retour :	La cote de l'utilisateur; 0 si l'id est invalide

Paramètre	**Entrée/Sortie**	**Description**
$referrer_id	Entrée	Identifiant de l'utilisateur référent.

Fonction :	**tri_get_reference**
Description :	Retourne un objet TriReferral correspondant à la référence dont l'id est passé en paramètre
Valeur de retour :	Objet TriReferral avec l'information si la référence est trouvé; null si non

Paramètre	Entrée/Sortie	Description
$ref_id	Entrée	Identifiant de la référence à récupérer

Fonction :	**tri_get_ references**
Description :	Retourne un tableau de toutes les références correspondant aux critères passés dans l'objet en paramètre.
Valeur de retour :	Un tableau d'objets TriReferral si des références sont trouvées; un tableau vide si une erreur survient ou aucune référence n'est trouvée

Paramètre	Entrée/Sortie	Description
TriReferral $ref	Entrée	Objet TriReferral dont chacun des attributs correspond à un critère de recherche.

Fonction :	**tri_get_application**
Description :	Retourne un tableau de toutes les références effectuées pour l'offre dont l'id est passé en paramètre et dont le statut est « applied ».
Valeur de retour :	Un tableau d'objets TriReferral si des références sont trouvées; un tableau vide si une erreur survient ou aucune référence n'est trouvée

Paramètre	Entrée/Sortie	Description
$offer_id	Entrée	Identifiant de l'offre pour laquelle on veut obtenir les applications

Fonction :	**tri_del_reference**
Description :	Désactive la référence dont l'id est passé en paramètre de la base de données.
Valeur de retour :	« true » si l'opération a réussi; « false » si non

Paramètre	Entrée/Sortie	Description
$ref_id	Entrée	Identifiant de la référence à désactiver

Fonction :	**tri_close_offer_form**
Description :	Affiche le formulaire de fermeture d'une offre en HTML.
Valeur de retour :	Aucune valeur de retour

Paramètre	**Entrée/Sortie**	**Description**
$offer_id	Entrée	Identifiant de l'offre à fermer
$action_url	Entrée	URL à assigner comme action au formulaire une fois soumis

Fonction :	**tri_process_close_offer_form**
Description :	Analyse le contenu soumis par le formulaire généré par tri_close_offer_form() et prend les actions appropriées pour fermer l'offre.
Valeur de retour :	« true » si l'opération a réussi; « false » si non

Paramètre	**Entrée/Sortie**	**Description**
$request	Entrée	Variable contenant les informations envoyées par le formulaire (variable $_REQUEST en PHP)
$offer_id	Entrée	Identifiant de l'offre à fermer
$errors	Sortie	Tableau contenant les erreurs expliquant une valeur de retour de « false »

Fonction :	**tri_refer_form**
Description :	Affiche le formulaire de référencement d'une offre en HTML.
Valeur de retour :	Aucune valeur de retour

Paramètre	**Entrée/Sortie**	**Description**
$action_url	Entrée	URL à assigner comme action au formulaire une fois soumis
$posted	Entrée	Contient les valeurs précédemment entrées afin de pré-remplir le formulaire

Fonction :	**tri_process_refer_form**
Description :	Analyse le contenu soumis par le formulaire génér par tri_refer_form() et prend les actions appropriées pour créer une référence à une offre.
Valeur de retour :	« true » si l'opération a réussi; « false » si non

Paramètre	**Entrée/Sortie**	**Description**
$user_id	Entrée	Identifiant de l'utilisateur effectuant la référence
$offer_id	Entrée	Identifiant de l'offre étant référée
$submit_values	Entrée	Tableau contenant les informations sur le candidat référé (nom, courriel, message, etc.)
$err	Sortie	Tableau contentant les erreurs expliquant une valeur de retour de « false »

Fonction :	**tri_user_reference_form**
Description :	Affiche un tableau contenant les références associées à un utilisateur en HTML. Les références affichées sont soit celles envoyées par l'utilisateur dans le cas où $is_referrer vaut « true », soit celle reçues par l'utilisateur dans le cas où $is_referrer vaut « false ».
Valeur de retour :	Aucune valeur de retour

Paramètre	**Entrée/Sortie**	**Description**
$user_id	Entrée	Identifiant de l'utilisateur pour lequel on doit afficher les références
$is_referrer	Entrée	Si la valeur est « true », le client désir obtenir les références envoyées par l'utilisateur. Si la valeur est « false », le client désir obtenir les références reçues par l'utilisateur.

!+! jpboily : **Ajouter les fonctions liées aux statistiques, aux transactions et aux autres formulaires**

4.1.3 Interface logiciel hôte

La classe *HostConnectorIf* est une classe d'interface devant êre implémentée pour chacun des logicels hôtes qui veulent installer le SDK. Les méthodes définies dans cette classe représentent certaines infromations dont a besoin le SDK pour fonctionner, mais qui ne sont accessible que par le logiciel hôte. Comme la façon d'accéder à ces informations est unique à chaque logiciel hôte, nous devons créer une nouvelle implémentation de cette interface pour chacun des clients du SDK. Voici donc une liste des méthodes de cette interface ainsi que leur description et la description de leurs paramètres :

Fonction :	**get_user_email**
Description :	Comme nous n'avons pas le droit de conserver le courriel des utilisateurs, cette fonction permet à partir d'un id d'utilisateur de recevoir son courriel le temps de lui envoyer un message. Le courriel n'est pas conservé et cette fonction doit être appelée chaque fois que nous voulons lui accéder.
Valeur de retour :	Adresse courriel de l'utilisateur

Paramètre	**Entrée/Sortie**	**Description**
$user_id	Entrée	Identifiant de l'utilisateur pour lequel on veut obtenir une adresse courriel

Fonction :	**send_mail**
Description :	Cette fonction sert à envoyer des courriels en utilisant la méthode préconisée par le logiciel hôte pour les envoyer.
Valeur de retour :	« true » si le courriel a été envoyé; « false » si non

Paramètre	**Entrée/Sortie**	**Description**
$to	Entrée	Adresse courriel du récipiendaire du message
$subject	Entrée	Sujet du courriel
$message	Entrée	Contenu du courriel
$headers	Entrée	Peut contenir des éléments à ajouter à l'entête du

		courriel
$attachments	Entrée	Contient dans un tableau les fichiers à joindre au courriel

public function append_arg_to_url($url, $arg_name, $arg_value);

!+! jpboily: On risque de conserver les tokens dans la BD de trimoz alors nous n'aurions plus besoin de ces méhtodes.

public function get_user_linkedin_token($user_id);

public function set_user_linkedin_token($user_id);

4.2 Documents d'entrée/sortie

5 Installation du produit

5.1 Intégration au système hôte

5.2 Ajout des pages non existantes

5.3 Modification du style CSS du SDK

5.4 Implémentation de l'interface vers l'hôte

5.5 Synchronisation des données

6 Résolution de problèmes