**Guide Web Services**

**SLS / Karizma Conseil**

|  |  |  |
| --- | --- | --- |
| DIFFUSION | | |
| ORGANISME | DESTINATAIRES | NBRE. COPIES |
| SLS |  | 1 |

|  |  |  |  |
| --- | --- | --- | --- |
| N° de Versio  n | Date | Auteur de la Modiﬁcation | Nature de la modiﬁcation |
| **1.0** | 22/04/2021 | **Moad KHIDER** | Version initiale |
| **2.0** | 29/04/2021 | **Saad THAIFA FATHI** | Ajout des méthodes  simpliﬁés |
| **3.0** | 07/05/2021 | **Saad THAIFA FATHI** | Ajout possibilité de création de  destinataires |
| **4.0** | 26/05/2021 | **Moad KHIDER** | Ajout le champs state, méthodes de retours de fonds,  méthode get\_colis |
| **4.1** | 07/06/2021 | **Moad KHIDER** | Modiﬁcation base de  données |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Table des matières:**

### AVANT-PROPOS

* 1. Context

### ARCHITECTURE ODOO

* 1. Diagramme de classe
  2. Description des objets

### WEB SERVICES ODOO

* 1. Documentation ofﬁcielle
  2. Comment Exploiter ce template?

### Captures d'écran

1. Méthodes simpliﬁées
2. Instance de test en ligne

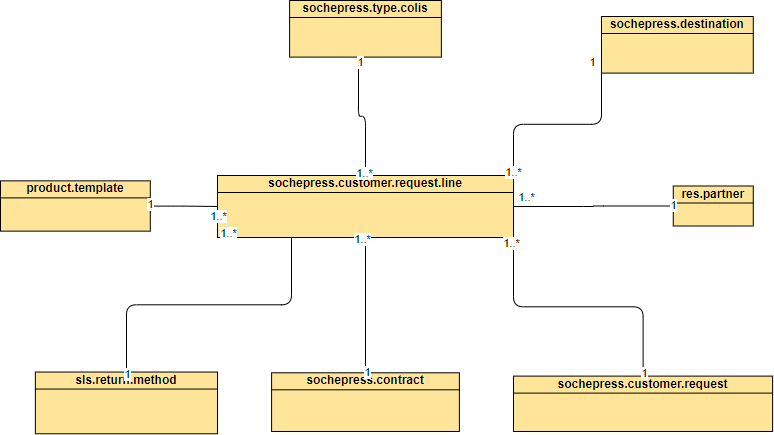
## AVANT-PROPOS

### Contexte

Le but de ce document est de fournir une idée sur l’architecture d’Odoo et les objets principaux qui vont faire l’objet de la synchronisation entre la plateforme web et Odoo.

## ARCHITECTURE ODOO

### Diagramme de classe



|  |  |  |  |
| --- | --- | --- | --- |
|  | **Karizma Conseil** |  | Auteur : Saad THAIFA FATHI |
| **Projet : Mise en place des Web Services plateforme SLS Odoo** | Date de dernière mise à jour : 22/04/2021 |

* 1. Description des objets

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Objet** | **Champs** | **Nom technique** | **Description** | **Type de champs** |
| **Colis**  sochepress.customer  .request.line | Reference | ref\_ext | non requis | caractere |
| Modèle d'article | product\_id | non requis | many2one |
| Client | customer\_id | non requis | many2one |
| Retour de fonds | return\_method\_i d | non requis | many2one |
| Ordre | order\_id | non requis | many2one |
| Contrat | contract\_id | non requis | many2one |
| Destination | destination\_id | non requis | many2one |
| **Position actuelle** | **current\_position** | **non requis** | **many2one** |
| Type de colis | type\_colis\_id | non requis | many2one |
| **Modèle d'article**  product.template | ID | id | non requis | integer |
| Categorie d’article | categ\_id | requis | many2one |
| Nom | name | requis | caractere |
| Articles | product\_variant\_i ds | requis | one2many |
| Ligne de commande d’achat | purchase\_line\_w arn | requis | selection |
| Tracking | Type d’article | requis | selection |
| Unité de mesure d’achat | uom\_po\_id | requis | many2one |
| Unité de mesure | uom\_id | requis | many2one |
| Ligne de bons de commande | sale\_line\_warn | requis | selection |
| Longueur | length | non requis | ﬂottant |
| Hauteur | height | non requis | ﬂottant |
| Volume | volume | non requis | ﬂottant |
| Designation Chez Client | designation\_chez  \_client | non requis | caractere |
| **Type de colis**  sochepress.type.col is | ID | id | non requis | integer |
| Nom | name | non requis | caractere |
| Afﬁcher Nom | display\_name | non requis | caractere |
| **Ordre**  soch.transport.order | ID | id | non requis | integer |
| Matricule | matricule | non requis | caractere |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Client | customer\_id | non requis | many2one |
| Société | company\_id | requis | one2many |
| **Retour de fonds**  sls.return.method | ID | id | non requis | integer |
| Code | code | non requis | caractere |
| Name | name | non requis | caractere (Espèces , Aucun ,  Prépayé , Chèque , Carte de crédit) |
| **Demande**  sochepress.customer  .request | Société | company\_id | requis | many2one |
| Client | customer\_id | requis | many2one |
| ID | id | non requis | integer |
| Expéditeur | expeditor\_id | non requis | many2one |
| Destinataire | destinator\_id | non requis | many2one |
| Status de la demande | state | non requis | selection |
| **Destination**  sochepress.destinati on | ID | id | non requis | integer |
| Destination | name | requis | caractere |
| Type de commande | type | non requis | selection |
| **Destinataire**  res.partner | Nom | name | non requis | caractere |
| Société liée | parent\_id | non requis | many2one res.partner |
| Rue | street | non requis | caractere |
| Rue 2 | street2 | non requis | caractere |
| Ville | city | non requis | caractere |
| Etat | state\_id | non requis | many2one res.country.state |
| Code postal | zip | non requis | caractere |
| Destination | destination\_id | non requis | many2one sochepress.destination |
| Telephone | phone | non requis | caractere |
| Mobile | mobile | non requis | caractere |
| Courriel | email | non requis | caractere |
| Type de societe | company\_type | non requis | selection (**person** pour les  particuliers et **company** pour une societe) |
| Type de contact | type | requis | Selection (contact, adresse de  facturation, adresse de livraison, autre adresse, adresse |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  | personnelle) Technical name:  contact, invoice, delivery,oreder, private  Valeur par défaut: contact |

## WEB SERVICES ODOO

### 3.1. Documentation ofﬁcielle d’utilisation des webservices Odoo:

Vous pouvez consultez la documentation de l'API externe de Odoo (XML-RPC) avec Odoo via le lien : <https://www.odoo.com/documentation/13.0/webservices/odoo.html>

### 3.2 Comment Exploiter ce template ?

Ce dépôt à pour but de montrer la possibilité d'attaquer l'API externe de Odoo avec un simple exemple d'une application PHP :

# La connexion vers L’API:

Selon la documentation, la connexion vers l'API peut se faire par plusieurs langages de programmation (Python 3, Ruby, PHP, JAVA) et dans le reste de notre tuto, on va travailler avec PHP :

Constantes requises:

|  |  |  |  |
| --- | --- | --- | --- |
| Python | Ruby | PHP | Java |
| url = <insert server URL> db = <insert database name>  username = 'admin' password = <insert password for your admin user (default: admin)> | url = <insert server URL> db = <insert database name>  username = "admin" password = <insert password for your admin user (default: admin)> | **$url** = <insert server URL>;  **$db** = <insert database name>;  **$username** = "admin";  **$password** = <insert password for your admin user (default: admin)>; | ﬁnal String url = <insert server URL>,  db = <insert database name>,  username = "admin", password = <insert  password for your admin  ***user*** (default: admin)>; |

Recevoir votre UID:

|  |  |  |  |
| --- | --- | --- | --- |
| Python | Ruby | PHP | Java |
| uid = common.authenticate(db, username, password, {}) | uid = common.call('authenticate', db, username, password, {}) | **$uid** =  **$common**->authenticate(**$ db**, **$username**,  **$password**, array()); | int uid = (int)client.execute( common\_conﬁg,  "authenticate", asList( db, username,  password, emptyMap())); |

# Méthodes d'appel

Le deuxième point de terminaison est xmlrpc/2/object, est utilisé pour appeler des méthodes de modèles odoo via la execute\_kw fonction RPC.

Chaque appel à execute\_kw prend les paramètres suivants:

* la base de données à utiliser, une chaîne
* l'ID utilisateur (récupéré via authenticate), un entier
* le mot de passe de l'utilisateur, une chaîne
* le nom du modèle, une chaîne
* le nom de la méthode, une chaîne
* un tableau / liste de paramètres passés par position
* un mapping / dict des paramètres à passer par mot-clé (optionnel)

# Liste des enregistrements

Les enregistrements peuvent être répertoriés et ﬁltrés via search().

search() prend un ﬁltre de domaine obligatoire (éventuellement vide) et renvoie les identiﬁcateurs de base de données de tous les enregistrements correspondant au ﬁltre. Pour lister les entreprises clientes par exemple:

|  |  |  |  |
| --- | --- | --- | --- |
| Python | Ruby | PHP | Java |
| models.execute\_kw(db, uid, password,  'res.partner', 'search',  [[['is\_company', '=',  **True**]]]) | models.execute\_kw(db, uid, password,  'res.partner', 'search', [[['is\_company', '=', true]]]) | **$models**->execute\_kw(**$db**,  **$uid**, **$password**, 'res.partner', 'search',  array(  array(array('is\_company', '=', true)))); | asList((Object[])models.exe cute("execute\_kw", asList(  db, uid, password, "res.partner", "search", asList(asList(  asList("is\_company", "=", **true**))) |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  | ))); |

# Lire des enregistrements:

|  |  |  |  |
| --- | --- | --- | --- |
| Python | Ruby | PHP | Java |
| ids = models.execute\_kw(db, uid, password,  'res.partner', 'search', [[['is\_company', '=', **True**]]],  {'limit': **1**}) [record] =  models.execute\_kw(db, uid, password,  'res.partner', 'read', [ids]) *# count the number of ﬁelds fetched by default* len(record) | ids = models.execute\_kw(db, uid, password,  'res.partner', 'search', [[['is\_company', '=', true]]],  {limit: **1**}) record =  models.execute\_kw(db, uid, password,  'res.partner', 'read', [ids]).ﬁrst  *# count the number of ﬁelds fetched by default* record.length | **$ids** =  **$models**->execute\_kw(**$db**,  **$uid**, **$password**, 'res.partner', 'search',  array(array(array('is\_compa ny', '=', true))),  array('limit'=>**1**));  **$records** =  **$models**->execute\_kw(**$db**,  **$uid**, **$password**, 'res.partner', 'read',  array(**$ids**));  *// count the number of ﬁelds fetched by default* count(**$records**[**0**]); | ﬁnal List ids = asList((Object[])models.exe cute(  "execute\_kw", asList( db, uid, password, "res.partner", "search", asList(asList(  asList("is\_company", "=", **true**))),  new HashMap() {{ put("limit", **1**); }}))); ﬁnal Map record =  (Map)((Object[])models.exe cute(  "execute\_kw", asList( db, uid, password, "res.partner", "read", asList(ids)  )  ))[**0**];  *// count the number of ﬁelds fetched by default* record.size(); |

Rechercher et lire:

Comme il s'agit d'une tâche très courante, Odoo fournit un search\_read() raccourci qui, comme son nom l'indique, équivaut à un search() suivi d'un read(), mais évite d'avoir à effectuer deux requêtes et de conserver les identiﬁants.

Ses arguments sont similaires à search()"s, mais il peut également prendre une liste de ﬁelds(comme read(), si cette liste n'est pas fournie, il récupérera tous les champs des enregistrements correspondants):

|  |  |  |  |
| --- | --- | --- | --- |
| Python | Ruby | PHP | Java |

|  |  |  |  |
| --- | --- | --- | --- |
| models.execute\_kw(db, uid, password,  'res.partner', 'search\_read',  [[['is\_company', '=', **True**]]],  {'ﬁelds': ['name', 'country\_id', 'comment'], 'limit': **5**}) | models.execute\_kw(db, uid, password,  'res.partner', 'search\_read',  [[['is\_company', '=', true]]],  {ﬁelds: %w(name country\_id comment), limit: **5**}) | **$models**->execute\_kw(**$db**,  **$uid**, **$password**, 'res.partner',  'search\_read',  array(array(array('is\_compa ny', '=', true))),  array('ﬁelds'=>array('name', 'country\_id', 'comment'), 'limit'=>**5**)); | asList((Object[])models.execute(" execute\_kw", asList(  db, uid, password, "res.partner", "search\_read", asList(asList(  asList("is\_company", "=",  **true**))),  new HashMap() {{ put("ﬁelds", asList("name",  "country\_id", "comment"));  put("limit", **5**);  }}  ))); |

# Créer des enregistrements:

|  |  |  |  |
| --- | --- | --- | --- |
| Python | Ruby | PHP | JAVA |
| id = models.execute\_kw(db, uid, password, 'res.partner', 'create', [{  'name': "New Partner",  }]) | id = models.execute\_kw(db, uid, password, 'res.partner', 'create', [{  name: "New Partner",  }]) | **$id** = **$models**->execute\_kw(**$db**,  **$uid**, **$password**, 'res.partner', 'create', array(array('name'=>"New  Partner"))); | final Integer id = (Integer)models.execute("execute  \_kw", asList(  db, uid, password, "res.partner", "create", asList(new HashMap() {{  put("name", "New Partner"); }})  )); |

Les enregistrements d'un modèle sont créés à l'aide de create(). La méthode créera un seul enregistrement et retourne son identiﬁant de base de données.

create() prend un mappage des champs aux valeurs, utilisé pour initialiser l'enregistrement. Pour tout champ qui a une valeur par défaut et n'est pas déﬁni via l'argument de mappage, la valeur par défaut sera utilisée.

# Exemple de TEST Web Services ODOO avec PHP:

|  |  |
| --- | --- |
| CONNEXION | require\_once(‘ripcord-master/ripcord.php’);  $url = “URL”;  $db = “DB\_NAME”;  $username = “USERNAME”;  $password = “PASSWORD”; |
| UID | $common = ripcord::client(“$url/xmlrpc/2/common”);  $version = $common->version(); |

|  |  |
| --- | --- |
|  | $uid = $common->authenticate($db, $username,  $password, $version); |
| CRÉATION D’UNE DEMANDE | $models = ripcord::client(“$url/xmlrpc/2/object”);  $id\_colis\_1 = $models->execute\_kw($db,  $username, $password, ‘sochepress.customer.request.line’, ‘create’, array(array(‘name’=> “colis1”, ‘expeditor\_id’ => 1315, ‘type\_colis\_id’ => 1))  $id\_colis\_2 = $models->execute\_kw($db,  $username, $password, ‘sochepress.customer.request.line’, ‘create’, array(array(‘name’=> “colis2”, ‘expeditor\_id’ => 1315, ‘type\_colis\_id’ => 1))  $id\_demand = $models->execute\_kw($db, $uid,  $password, ‘sochepress.customer.request’, ‘create’, array(array(‘name’=>’DEMAND’, ‘customer\_id’=> 1315, ‘request\_line\_ids’ => array($id\_colis1,  $id\_colis\_2)))) |
| CHERCHER ET LIRE TOUS LES COLIS | $colis = $models->execute\_bw($db, $uid,  $password, ‘sochepress.customer.request.line’, ‘search’, array(array(array(‘request\_id.customer.user\_id’, ‘=’,  $uid)))) |

## METHODES SIMPLIFIEES:

### Methode get\_list\_colis:

Cette méthode permet de retourner une liste de colis, on passe l'état de colis comme paramètre (“new”, “delivered”…)

Le type de retour est une liste de dictionnaire, dans la façon suivante:

[{“name”: “NOM”, “id”: ID, “step”: “ETAT”, “weight”: 10) }]

Exemple sous PHP:

|  |  |
| --- | --- |
| Methodes | Resultats |
| $id = $models -> execute\_kw($db, $uid,  $password,customer.request’, ‘**get\_list\_colis**’, array(“new”)) | [{“name”: “HC3GLUX”, “id”: 883, “step”: “new”, “weight”: 4},...] |

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Karizma Conseil** |  | Auteur : Saad THAIFA FATHI |
| **Projet : Mise en place des Web Services plateforme SLS Odoo** | Date de dernière mise à jour : 22/04/2021 |

### Methode create\_destinator:

Cette méthode se charge de créer un destinataire (client récurrent) que vous pouvez récupérer son id et l’affecter à une demande.

Le paramètre saisie est un dictionnaire qui se compose des éléments suivants:

{

"nom\_destinataire": // ici vous entrez le nom du destinataire “rue2”: // rue

“ville”: // ville “pays”: // pays “region”: // region “zip”: // zip code

“destination”: // destination “phone”: // phone

“mobile”: // mobile “email”: // email

“type”:// delivery

“company\_type”: // ‘person’ pour les particuliers et ‘company’ pour une société.

}

Le type de retour est l’id du destinataire créé.

Exemple d'Implémentation sous php:

|  |  |
| --- | --- |
| Methodes | Resultats |
| $dict\_destinator = [ “nom\_destinataire” => “Mohamed El Kbir”, “rue” => “Rue 2 BV AIT WAKRIM”,  “rue2” => “Poste 24 Hay Addakhla”, “ville” => “Agadir”,  “pays”=> “Maroc”,  “region” => “Tanger-Tetouan”, “zip”=> “23003”,  “destination” => “Tanger”, “phone” => “0654653489”,  “mobile” => “0654653489”,  “email” => “[mohamedelkbir@gmail.com](mailto:mohamedelkbir@gmail.com)”  ]  $id = $models->execute\_kw($db,  $uid,$password,customer.request’,  **‘create\_destinator’**, [$dict\_destinator]) |  |

### Méthode create\_demand:

Cette méthode se charge de créer une demande client et ses colis.

Le paramètre saisie est un dictionnaire qui se compose des éléments suivants:

{

“type”: // “normal” si vous voulez passez une commande normale “customer\_id” // veuillez entrer votre customer\_id

“date”: // a laisser vide si vous voulez passer la date actuelle

“**generate\_barcode**”: // **1** (si vous souhaitez générer les codes barres des colis.) “**colis**”: // liste des colis

}

Le **colis** dans ce dictionnaire est une liste de nombreux colis qui sont créés automatiquement lors de l’appel de cette méthode.

Pour créer un colis il sufﬁt de le déclarer de la façon suivante:

Si vous précisez un destinataire, et que vous passez son id le système va s'en charger de toutes les infos prédéﬁnies par ce dernier.

colis = {

“ref\_externe”: // référence externe du colis “customer\_id” // veuillez entrer votre customer\_id “type\_colis”: // type de colis

“expediteur”: // expediteur “**destinataire**”: // id du destinataire “source”: // source

“destination”: // destination “poids\_colis”: // poids du colis

“methode\_contre\_remboursement”: // méthode contre remboursement; Espèces , Aucun , Prépayé , Chèque , Carte de crédit.

“montant\_contre\_remboursement”: // montant contre remboursement

}

Si vous voulez créer un **destinataire** au moment de la création d’un colis:

colis = {

“ref\_externe”: // référence externe du colis “customer\_id” // veuillez entrer votre customer\_id “type\_colis”: // type de colis

“expediteur”: // expediteur “source”: // source

**"nom\_destinataire": // nom du destinataire “rue”: // rue**

**“rue2”: // rue secondaire**

**“ville”: // ville “region”: // region “pays”: // pays “zip”: // zip code**

**“phone”: // telephone “mobile”: // mobile “email”: // email** “destination”: // destination

“poids\_colis”: // poids du colis

“methode\_contre\_remboursement”: // méthode contre remboursement; Espèces , Aucun , Prépayé , Chèque , Carte de crédit.

“montant\_contre\_remboursement”: // montant contre remboursement

}

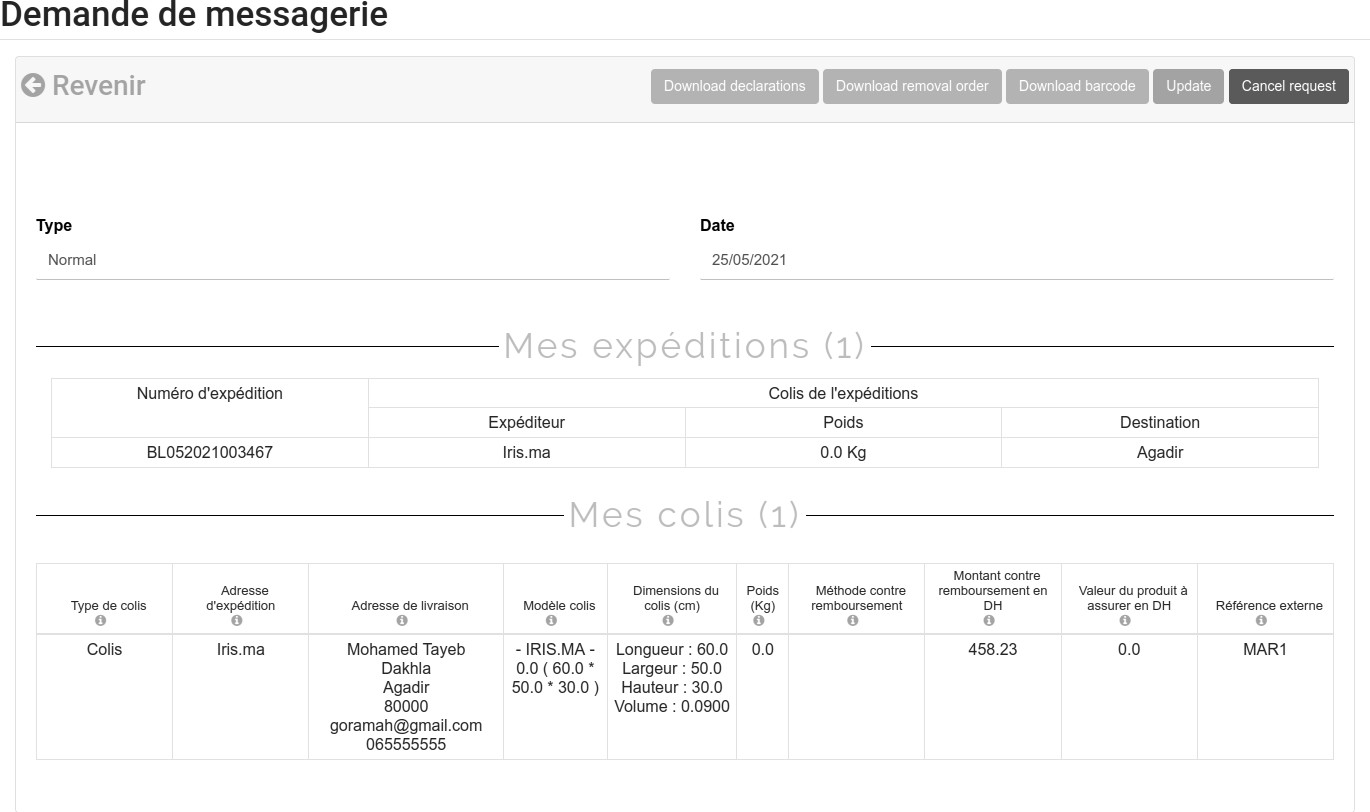
Le type de retour est l’id de la demande créée.

Exemple sous PHP:

|  |  |
| --- | --- |
| Methode | Resultat |
| $colis\_one = array( “” =>  "ref\_externe" => 'MAR1', "type\_colis" => "Colis", "expediteur" => "Iris.ma", "source" => "Rabat", "destinataire" =>4532, "destination" => "Agadir", "poids\_colis" => 4,  "methode\_contre\_remboursement" => "Espèces", "montant\_contre\_remboursement" => 65564, ); “modele” => 5;  $colis\_two = array( “”  "ref\_externe" => 'ref\_ext', "type\_colis" => "Colis", "expediteur" => "Iris.ma", "source" => "Tétouan", "destinataire" =>4532, "destination" => "Kénitra", "poids\_colis" => 34,  "methode\_contre\_remboursement" => "Chèque", "montant\_contre\_remboursement" => 80000, ); “height” => 5,  “width” => 4,  “length” => 3,  $dict\_infos = array( "customer\_id" => 358, "type" => "normal", "date" => '', "colis" => array($colis\_one,  $colis\_two) );  $id = $models->execute\_kw($db, $uid,  $password,.customer.request', 'create\_demand', array($dict\_infos,)); |  |

On a ajouter un autre champs modèle pour spéciﬁer l’id du **modèle** de colis sans spéciﬁer le volume qui sera calculé automatiquement:

Vous pouvez également passez les champs hauteur (**height**), longueur (**length**), largeur(**width**) sans spéciﬁer le volume qui sera affecté automatiquement après création le modèle de colis avec les dimensions que vous avez entrer :



### Méthode get\_colis():

Nous avons également ajouté une autre fonction **get\_colis()** qui a pour paramètre un dictionnaire :

$dict\_infos = [

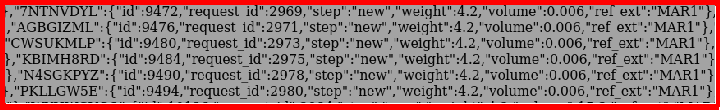
"ref" => "MAR", //1er Critère de recherche (Optionnel),

"colis\_name" => 'OF8PA0QC', //2eme Critère de recherche (Optionnel) "request\_id" => 3000 //3eme Critère de recherche (Optionnel),

];

avec :

**ref** : la référence externe du colis. **colis\_name** : le nom du colis. **request\_id** : l’id de la demande.



Exemple sous PHP:

|  |  |
| --- | --- |
| Methodes | Resultats |
| $dict\_infos = [  "ref" => "MAR", //1er Critère de recherche (Optionnel),  "colis\_name" => 'OF8PA0QC', //2eme Critère de recherche (Optionnel)  "request\_id" => 3000 //3eme Critère de recherche (Optionnel),  ];  $p = $models->execute\_kw($db, $uid,  $password,customer.request', 'get\_colis', [$dict\_infos]); echo json\_encode($p); | Le résultat de la requête sera un dictionnaire ou un ensemble des dictionnaires, ca dépend du nombre des colis retourné : |

5. INSTANCE DE TEST EN LIGNE

Une instance Odoo a été mise à votre disposition pour suivre l’avancement du projet et exécuter la recette client :

|  |  |
| --- | --- |
| **Lien** | http://13.37.208.5/ |
| **Login** |  |
| **Mot de passe** |  |
| **Base de données** | SLM\_PROD\_30\_07 |