

# Collec : description des services web

Eric Quinton

27 juin 2017

# Chapitre 1

## Besoins nécessitant l'utilisation de services web

### 1.1 Définitions

**uid** : identifiant unique numérique au sein d'une base de données d'un échantillon ;

**guid** : identifiant de type UUID<sup>1</sup>, qui garantit de manière certaine l'identification d'un échantillon ;

**identifier** : identifiant « métier » d'un échantillon ;

**données « métier »** : données permettant de caractériser un échantillon selon les critères nécessaires à son exploitation : contexte spécifique d'acquisition, taxon, données physico-chimiques ou biologiques, etc.

**instance, serveur, base de données, application** : implémentation d'une solution de gestion d'échantillons capable soit de fournir des services web, soit d'interroger des services web pour récupérer des informations.

### 1.2 Présentation

La gestion des échantillons de laboratoire (ou autres) est réalisée par divers logiciels, chacun s'appuyant sur une base de données différente. Dans un contexte d'échange des informations entre laboratoires, il est apparu nécessaire de proposer

---

1. Les codes de type GUID ou UUID sont générés à partir de fonctions aléatoires ou cryptographiques, et garantissent qu'ils sont uniques quelle que soit la base de données qui les ont générés. Ainsi, il n'est pas possible d'obtenir deux codes identiques pour deux échantillons différents, ce qui permet de les identifier de manière sûre, comme pourrait le faire l'ADN pour des êtres humains.

une méthode d'interrogation des différentes bases de données, pour connaître ce qui est disponible dans d'autres collections.

Deux cas d'utilisation ont ainsi pu être mis en évidence :

- l'interrogation d'une base distante, pour savoir si des échantillons sont disponibles ;
- l'import d'informations provenant d'une autre base de données, pour répondre à deux cas de figure différents :
  - le prêt d'un échantillon à un autre laboratoire, sans être obligé de resaisir toutes les métadonnées associées ;
  - l'importation de données saisies sur le terrain en mode déconnecté dans une base de données centrale.

La technologie retenue est celle des services web, associée au protocole OAuth v.2 pour la gestion des habilitations.

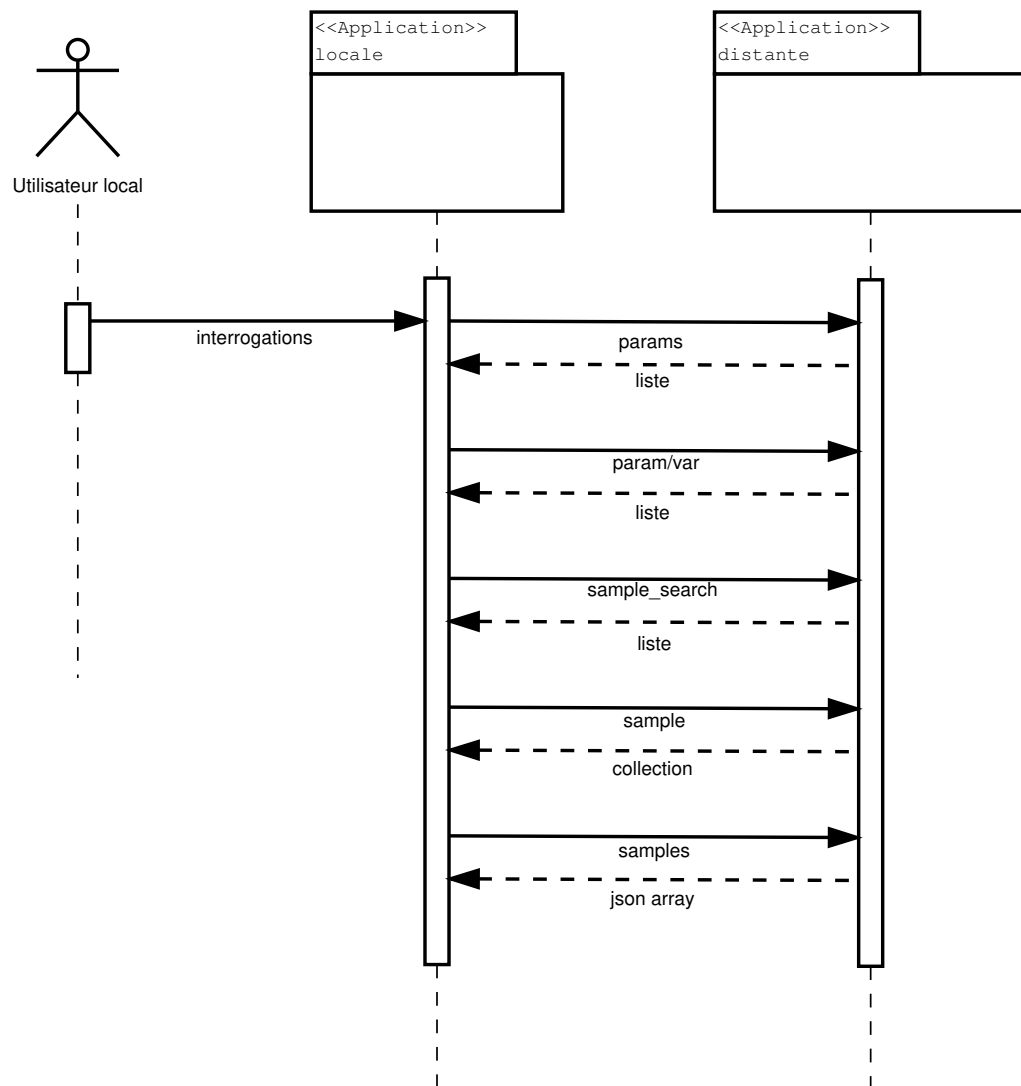
### **1.2.1 Technique employée**

Les services web sont basés sur des requêtes HTTP, et échangent les données selon des formats définis. Dans la version actuelle des services web, seul le format Json est implémenté pour l'échange des informations.

### **1.2.2 Forme des URL**

Les URL sont conçues, dans le cadre des services web, sous la forme d'URL conviviales, par exemple : *http ://collec/sw/v1/sample/4* pour récupérer l'échantillon numéro 4.

### 1.3 Définition des cas d'utilisation couverts par les services web



Cinq services web sont nécessaires :

- les deux premiers permettent de récupérer les listes de référence nécessaires pour faire des recherches selon divers paramètres ;
- le troisième permet de lancer une recherche pour trouver des échantillons ;
- les deux derniers récupèrent les informations correspondant à un ou plusieurs échantillons respectivement.

### 1.3.1 Recherche d'échantillons

La recherche d'échantillons doit pouvoir s'effectuer selon plusieurs critères :

- l'identifiant interne (*uid*) ;
- l'identifiant principal ou des identifiants secondaires ;
- une fourchette de dates de création des échantillons ;
- un type d'échantillons ;
- un projet ou sous-collection ;
- une fourchette d'identifiants internes ;
- un code unique de type GUID ou UUID ;
- le lieu de collecte de l'échantillon ;
- tout autre paramètre traité dans l'application distante.

Elle retourne une liste d'échantillons correspondant aux critères indiqués. Le détail des informations retournées est spécifié dans la section 3.4.

Pour permettre cette recherche, il est nécessaire de récupérer les tables de référence (types d'échantillons, projets ou sous-collections, lieux de collecte...). Les tables utilisables pour la recherche peuvent différer selon les logiciels, et cette liste peut être retrouvée par un service web dédié.

### 1.3.2 Liste des tables de référence

Ce service permet d'obtenir l'ensemble des tables de référence avec lesquelles il est possible de réaliser une recherche.

### 1.3.3 Contenu des tables de référence

Ce service récupère le contenu d'une des tables de référence.

### 1.3.4 Récupération des données d'un échantillon

Ce service permet de récupérer l'ensemble des données concernant un échantillon, y compris les données « métiers » si l'utilisateur dispose des droits nécessaires pour les consulter.

Les données récupérées doivent être suffisamment complètes pour pouvoir être intégrées dans l'application de gestion des échantillons locale.

Elles permettent également une visualisation détaillée de l'échantillon considéré, et contiennent, le cas échéant<sup>2</sup>, les données « métiers » associées.

---

2. si l'utilisateur dispose des droits adéquats et si l'échantillon dispose de ces informations

### **1.3.5 Récupération des données d'une liste d'échantillons**

Il s'agit d'une variante du service web précédent. La liste des échantillons à récupérer est fournie dans une collection Json, soit en utilisant l'UID, soit en utilisant le GUID.

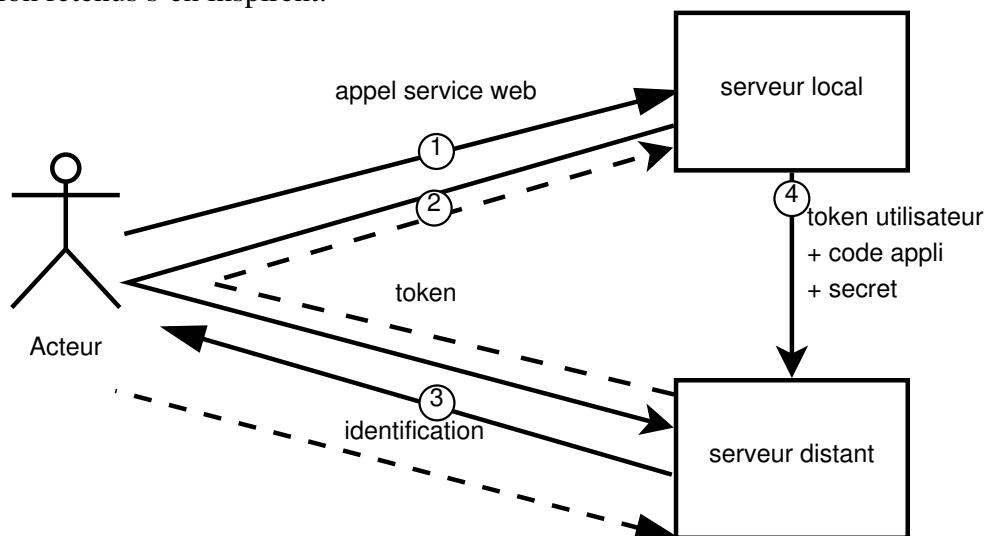
## **1.4 Contraintes liées à la sécurité**

L'accès aux données nécessite une identification préalable de l'utilisateur. Celle-ci est basée sur le protocole OAuth v2 (*cf.* chapitre 2).

## Chapitre 2

# Gestion des habilitations – principes retenus pour le logiciel Collec

Le mécanisme d'identification prévu pour les services web s'appuie en grande partie sur le protocole OAuth v2, au moins dans ses principes. Ce n'est pas le protocole qui est effectivement implémenté (OAuth est prévu pour que l'utilisateur définisse les droits d'accès de l'application cliente), mais les principes d'identification retenus s'en inspirent.



Pour accéder à des informations distantes, le dialogue entre les deux serveurs nécessite plusieurs phases :

1. l'utilisateur veut déclencher l'appel à un service web distant ;
2. aucun jeton n'a été récupéré par le serveur local : l'utilisateur est redirigé vers le serveur distant pour obtenir un jeton d'identification ;

3. le serveur distant ne connaît pas l'utilisateur : il lui demande de s'identifier en utilisant la procédure adéquate (login/mot de passe dans la plupart des cas) ;  
une fois identifié, le serveur distant renvoie le navigateur de l'utilisateur vers le serveur local, en fournissant un jeton chiffré ;
4. l'appel au service web peut maintenant être déclenché, en fournissant :
  - le jeton de l'utilisateur ;
  - le code de l'application locale ;
  - le secret fourni par l'application distante pour l'application locale.

## 2.1 Opérations préalables à l'interrogation d'une base de données distante

### 2.1.1 Identification réciproque des applications

Les applications clientes doivent se faire identifier par les applications distantes avant tout échange.

Cette étape est manuelle : l'administrateur de l'application distante crée un enregistrement dans la table *instance*, dont le détail est décrit dans la section 4.2.2.

Le code d'identification, ainsi que le secret associé, est envoyé par mail par l'administrateur de la base distante.

Ces informations sont stockées dans la table *instance* locale, en indiquant en outre les adresses des différents services web disponibles pour le serveur distant.

### 2.1.2 Identification des utilisateurs

Les utilisateurs qui peuvent utiliser les services web distants sont enregistrés dans le serveur distant et disposent d'un code d'identification. Une fois connectés, ils doivent hériter du droit *consultsw*, indispensable pour accéder aux services web.

Pour qu'ils puissent récupérer les données « métier », les utilisateurs doivent également être rattachés à un projet. Dans la pratique, dans la gestion des groupes de logins, on trouvera l'arborescence suivante (par exemple) :

consultsw	
projet1sw	
projet2sw	



Les utilisateurs distants sont associés aux groupes *projet1sw* ou *projet2sw*, et ces groupes sont associés aux projets (ou sous-collections) adéquats pour leur donner les droits d'accès en lecture.

# Chapitre 3

## Description des services web

### 3.1 Remarques générales

Les services web sont destinés à fournir des informations aux clients. Leur implémentation dépend des logiciels, ainsi que la manière de traiter les données.

Pour des raisons de compatibilité entre les différents logiciels, si un argument fourni figure dans la requête mais n'est pas utilisable par le service web, ce dernier ne doit pas rejeter la requête, sauf si aucun argument ne permet de discriminer la recherche demandée.

#### 3.1.1 Format des données transmises ou reçues

Les données sont échangées au format JSON, encodées en base 64. Toutes les données JSON sont encodées en base 64, que ce soit dans les paramètres transmis ou dans les données retournées.

Les requêtes pour obtenir des informations sont de type GET. Actuellement, aucun service web n'est prévu pour mettre à jour des informations dans la base distante.

#### 3.1.2 Codes d'erreur

Voici la liste des codes d'erreur génériques que les services web peuvent retourner :

Code	Description
400	Bad argument : un des arguments fournis n'est pas valide, ou aucun argument utilisable n'a été indiqué <sup>1</sup>

1. Dans le cas où des arguments sont fournis, mais non utilisables par le service web considéré, l'exécution de la requête conduisant à renvoyer trop d'informations

Code	Description
401	Unauthorized : les droits d'accès sont insuffisants pour permettre d'obtenir les informations
404	Not found : la ressource demandée est introuvable
408	RequestTimeout : la requête n'a pas abouti dans le temps imparti, ou elle a été annulée par le client
500	InternalServerError : une erreur d'exécution ne permet pas de retourner le résultat de la requête
503	ServiceUnavailable : le service est temporairement indisponible

## 3.2 Table des listes techniques

Ce service web permet de récupérer la liste des tables de référence qui peuvent être utilisées pour lancer une recherche.

### 3.2.1 url type

`https://sitedistant.fr/sw/v1/params`

### 3.2.2 Données en entrée

Le service web ne nécessite pas de paramètre d'entrée.

### 3.2.3 Données en retour

La requête retourne la liste demandée sous la forme d'une collection Json contenant les informations suivantes :

Code	Type	Description
code	varchar	Code de la table de référence. C'est ce code qui sera utilisé pour obtenir la liste des valeurs disponibles dans le service web de récupération du contenu
val	varchar	Texte affichable dans un masque de recherche
valfr	varchar	Texte affichable dans un masque de recherche, en français
valen	varchar	Texte affichable dans un masque de recherche, en anglais

Exemple :

```
[
{code:"project",val:"Projets", valen:"projects"},
{code:"sampletype",val:"Types d'échantillons", valen:"samples
types"},
{code:"identifiertype",val:"Identifiants metier", valen:"
business identifiers"},
{code:"place",val:"Lieux de prelevement des echantillons",valen
:"sampling places"}
]
```

### 3.3 Listes techniques

Ce service web permet de récupérer une liste de référence nécessaire pour comprendre ou rechercher les échantillons de la base distante.

#### 3.3.1 url type

`https://sitedistant.fr/sw/v1/reference/project` ou `https://sitedistant.fr/sw/v1/reference/var=project`.

La valeur fournie (*project* ici) est une des valeurs récupérée dans le service web précédent (rubrique *code*).

La requête est de type GET.

#### 3.3.2 Données en retour

La requête retourne la liste demandée sous la forme d'une collection Json contenant les informations suivantes :

Code	Type	Description
id	integer	Identifiant interne
val	varchar	Libellé
comment	varchar	Description ou commentaire

Exemple :

```
[
{id:1,val:"projet1",comment:"Description du projet 1"},
{id:3,val:"projet3",comment:"Descripton du projet 3"}
]
```

## 3.4 Rechercher des échantillons

### 3.4.1 url type

`https://sitedistant.fr/sw/v1/sample_search`

### 3.4.2 Variables de recherche

Les variables doivent être fournies sous la forme d'une variable JSON *val*, encodée en base 64.

Code	Type	Description
uid	integer	Identifiant unique de l'échantillon dans l'instance distante
ident	varchar	identifiant « métier » principal
guid	UUID	identifiant unique quel que soit la base de données
uidstart	integer	uid inférieur pour une recherche sur une fourchette d'identifiants
uidend	integer	uid supérieur pour une recherche sur une fourchette d'identifiants
datestart	yyyy-mm-dd	date de début pour une recherche sur une fourchette de dates
dateend	yyyy-mm-dd	date de fin pour une recherche sur une fourchette de dates
codefamily	objet Json	objet json permettant de rechercher une valeur sur une table de paramètres. <i>Codefamily</i> doit être remplacé par le code retourné lors de la recherche des listes de paramètres. L'objet JSON accepte, comme informations associées :

Code	Type	Description
		<i>id</i> : identifiant spécifique à rechercher (par exemple, le code de l'identifiant métier). <i>id</i> peut être facultatif, si la recherche ne porte que sur la valeur, par exemple, le lieu de prélèvement ; <i>val</i> : valeur recherchée si uniquement <i>val</i> est indiqué, on peut utiliser un couple classique : <i>codefamily:val</i>

Exemple de fichier :

```
{ "uidstart": 25,
  "uidend": 50,
  "datestart": "2017-01-01",
  "dateend": "2017-06-30",
  "identifiertype": {id:10, val:"AB01"},
  "sampletype": 2
}
```

La recherche portera sur les échantillons dont l'UID est entre 25 et 50, dont la date est entre le 1er janvier et le 30 juin 2017, dont l'identifiant métier connu sous le numéro 10 vaut AB01, et dont le type d'échantillons est 2.

### 3.4.3 Données en retour

Collection Json avec, pour chacun, les informations suivantes :

Code	Type	Description
uid	integer	Identifiant dans l'instance
identifier	varchar	identifiant principal « métier »
guid	uuid	code d'identification global
ids	collection	liste de tous les identifiants secondaires, selon la forme idtype : idval
project	varchar	nom du projet ou de la sous-collection correspondante
createdate	yyyy-mm-dd hh:mm:ss	date de création de l'échantillon dans la base de données d'origine
collectdate	yyyy-mm-dd hh:mm:ss	date de collecte ou de génération de l'échantillon
metadata	objet json	données « métier » rattachées à l'échantillon

Code	Type	Description
sampleparent	objet json	json de même structure que ce tableau comprenant les informations du parent (imbrication des différents parents le cas échéant)
storageproduct	varchar	produit de stockage utilisé
clp	varchar	risques associés aux produit de stockage
subsampletype	varchar	type de sous-échantillonnage
subsampleunit	varchar	unité de sous-échantillonnage
subsampleqty	double	quantité de sous-échantillons présents initialement
samplingplacename	varchar	nom du lieu de collecte

## 3.5 Lire un échantillon

### 3.5.1 url type

`https://sitedistant.fr/sw/v1/sample`

La requête, de type GET, contient soit en dernière valeur l'UID de l'échantillon à lire, soit la variable *param=*, encodée en base 64, contenant un fichier json permettant de spécifier le type d'identifiant utilisé :

Code	Description
Clé	Nom de la variable utilisée – valeur issue du résultat du service web de recherche des listes techniques (type <i>identifiertype</i> )
valeur associée	Valeur correspondante

Les valeurs utilisables dans les champs sont les suivants :

id	Format attendu dans val	Description
uid	integer	Clé utilisée dans la base de données
guid	uuid	Identifiant unique global
id	varchar	Identifiant principal de l'échantillon

Voici un exemple de fichier Json, avant son encodage en base 64 :

```
{"uid":25}
```

Les données en retour sont celles du service de recherche d'un échantillon.

## **3.6 Lire un jeu d'échantillons**

### **3.6.1 url type**

```
https://sitedistant.fr/sw/v1/samples?param=xxx
```

Il s'agit d'une variante du cas précédent. Le fichier json (encodé en base 64) est organisé pour fournir un identifiant par échantillon retourné. Voici un exemple du fichier :

```
[{"uid":15}, {"id":"A1-B2-C3"}]
```

Les données en retour sont celles du service de recherche d'un échantillon.



## Chapitre 4

# Implémentation technique dans Collec

Collec est un logiciel de gestion de collections d'échantillons, dont l'objectif principal vise à faciliter la recherche d'un échantillon stocké ou de récupérer les informations générales le concernant.

Écrit en PHP, les données sont stockées dans une base de données PostgreSQL. Le code de l'application est disponible à l'adresse <https://gitlab.com/Irstea/collec>. Il est disponible sous licence AGPL.

Le logiciel est bâti sur un modèle MVC, tous les accès étant gérés par l'appel à des modules déclarés dans un fichier spécifique.

La gestion matérielle des échantillons de laboratoire (ou d'expérimentations scientifiques) est une fonctionnalité largement demandée, mais peu couverte jusqu'à présent par les logiciels disponibles, et particulièrement dans le domaine de l'*Open Source*. Collec, dont la première version remonte à l'automne 2016, fait l'objet d'un réel intérêt de la part de la communauté scientifique, ses fonctionnalités et sa facilité d'utilisation le rendant attractif.

Toutefois, il n'est pas conçu comme un système global de gestion de données à la fois techniques – stockage des échantillons – et de résultats d'analyse par exemple (pas d'informations métiers complexes<sup>1</sup>). Il n'est pas non plus prévu de mettre en place un hébergement centralisé qui permettrait de gérer tous les échantillons de la sphère de recherche.

*A contrario*, cette organisation permet de créer autant d'instances que nécessaires, notamment pour gérer des saisies en mode décentralisé (bateau partant en campagne de sondage dans les mers du Sud, collecte d'échantillons depuis des zones non couvertes par Internet, par exemple).

---

1. Dans la pratique, à partir de la version 1.1, il est possible de renseigner quelques informations métiers, mais de manière relativement frustrante et sans permettre la complexité des actions envisageables avec des bases de données dédiées.

Cette souplesse nécessite de prévoir des mécanismes soit d'interrogation de diverses instances, soit de récupération des informations concernant des échantillons provenant d'autres bases de données.

## 4.1 Transformation des URL et appel aux modules

Les URL conviviales sont transformées en noms de modules, selon le fonctionnement suivant :

- les trois premiers éléments de l'adresse sont fusionnés ;
- si le quatrième élément est présent, il est stocké dans la variable de requête *\$id*, et :
  - si la requête est de type GET, le module est suffixé par *Display* ;
  - si la requête est de type POST, le module est suffixé par *Write*<sup>2</sup> ;
- sinon, le module est suffixé par *List*.

Les modules doivent être décrits, comme les autres, dans le fichier *param/actions.xml*, et sont exécutés selon le fonctionnement classique de l'application.

## 4.2 Emplacement du code

Le code spécifique des modules doit être stocké dans le dossier *modules*, en respectant l'arborescence des URL conviviales, par exemple, pour l'adresse *http://collec.local/sw/v1/sample*, dans le sous-dossier *sw/v1*.

### 4.2.1 Données d'identification de l'utilisateur fournies par l'instance locale

Toutes les données concernant l'identification ne sont accessibles qu'aux administrateurs (droit *admin*), et sont stockées dans le schéma *gacl*.

L'identification vers un serveur distant nécessite que le serveur appelant fournisse les informations suivantes :

- un code identifiant de manière sûre l'utilisateur ;
- un secret connu uniquement par les deux serveurs.

Le code d'identification est généré à partir du mail de l'utilisateur, en utilisant une fonction cryptographique. Compte-tenu de l'absence de risque lié à ce code et du faible risque de collisions<sup>3</sup>, le code est généré à partir des 12 premiers chiffres hexadécimaux de la fonction de calcul d'empreintes SHA-1 :

---

2. Dans la version actuelle, l'écriture depuis une instance distante n'est pas implémentée

3. Une collision se produit quand deux chaînes différentes aboutissent à la même empreinte.

contenu...

Le secret est généré par l'instance distante, en utilisant une fonction aléatoire :

```
sha256(random()) x 64 caracteres
```

Cette information est stockée dans la table *login*, dans un champ qui n'est jamais transmis au navigateur (mécanisme d'écrasement s'il est modifié depuis un formulaire, notamment pour l'ajouter).

Les codes d'identification et les secrets partagés étant, par construction, différents selon les implémentations, la table *user\_instance* comprend les informations suivantes :

- le mail de l'utilisateur, en tant qu'identifiant unique ;
- la référence à l'instance distante ;
- le code généré à partir de l'adresse mail ;
- le secret ;
- la liste des projets auquel l'utilisateur est associé (accessibilité aux données « métier » ;
- la date d'expiration de l'autorisation d'accès. Par défaut, cette date est fixée à 1 an.

#### 4.2.2 Données d'identification des instances distantes

Pour identifier les instances clientes, la table *instance* contient les données suivantes :

- l'url de l'instance ;
- le nom d'un contact ;
- son mail ;
- le code attribué en utilisant le même mécanisme que pour les utilisateurs, mais en se basant sur l'url ;
- le secret (même mode de calcul) ;
- le type d'instance (cliente ou serveur) ;
- la date de fin d'autorisation d'accès, par défaut, fixée à 5 ans.

Si une instance est à la fois cliente et serveur, elle aura deux lignes, l'une pour chaque sens de communication. Cela permet de maintenir des secrets différents pour chaque canal.

Pour les instances « serveurs », une table complémentaire permet d'indiquer les URI des services web disponibles :

- le type du service web ;
- l'URI correspondante ;
- le type d'identification prévu : *OauthV1*, *OauthV2*, pas d'identification.

### **4.2.3 Structure des tables correspondantes**

# Table des matières

<b>1</b>	<b>Besoins nécessitant l'utilisation de services web</b>	<b>1</b>
1.1	Définitions . . . . .	1
1.2	Présentation . . . . .	1
1.2.1	Technique employée . . . . .	2
1.2.2	Forme des URL . . . . .	2
1.3	Définition des cas d'utilisation couverts par les services web . . .	3
1.3.1	Recherche d'échantillons . . . . .	4
1.3.2	Liste des tables de référence . . . . .	4
1.3.3	Contenu des tables de référence . . . . .	4
1.3.4	Récupération des données d'un échantillon . . . . .	4
1.3.5	Récupération des données d'une liste d'échantillons . . .	5
1.4	Contraintes liées à la sécurité . . . . .	5
<b>2</b>	<b>Gestion des habilitations – principes retenus pour le logiciel Collec</b>	<b>6</b>
2.1	Opérations préalables à l'interrogation d'une base de données distante . . . . .	7
2.1.1	Identification réciproque des applications . . . . .	7
2.1.2	Identification des utilisateurs . . . . .	7
<b>3</b>	<b>Description des services web</b>	<b>9</b>
3.1	Remarques générales . . . . .	9
3.1.1	Format des données transmises ou reçues . . . . .	9
3.1.2	Codes d'erreur . . . . .	9
3.2	Table des listes techniques . . . . .	10
3.2.1	url type . . . . .	10
3.2.2	Données en entrée . . . . .	10
3.2.3	Données en retour . . . . .	10
3.3	Listes techniques . . . . .	11
3.3.1	url type . . . . .	11
3.3.2	Données en retour . . . . .	11
3.4	Rechercher des échantillons . . . . .	12

3.4.1	url type . . . . .	12
3.4.2	Variables de recherche . . . . .	12
3.4.3	Données en retour . . . . .	13
3.5	Lire un échantillon . . . . .	14
3.5.1	url type . . . . .	14
3.6	Lire un jeu d'échantillons . . . . .	15
3.6.1	url type . . . . .	15
<b>4</b>	<b>Implémentation technique dans Collec</b>	<b>16</b>
4.1	Transformation des URL et appel aux modules . . . . .	17
4.2	Emplacement du code . . . . .	17
4.2.1	Données d'identification de l'utilisateur fournies par l'ins- tance locale . . . . .	17
4.2.2	Données d'identification des instances distantes . . . . .	18
4.2.3	Structure des tables correspondantes . . . . .	19