# Collec : description des services web

Eric Quinton

13 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 [1], qui garantit de manière certaine l'identification d'un échantillon ;

**identifier** identifiant « métier »d'un échantillon.

## 1.2 Présentation

Collec est un logiciel de gestion de collections d'échantillons, dont l'objectif principal consiste à permettre de retrouver rapidement 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. Il ne gère pas initialement les URL conviviales (implémenté à partir de la version 1.1).

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.

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 [2]). 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).

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. Pour harmoniser les échanges ou les interrogations, la technologie des services web s'impose, en raison de la normalisation qu'elle apporte.

### 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.

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

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

### 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.

Elle retourne une liste d'échantillons correspondant aux critères indiqués. Le détail des informations retournées est spécifié dans la section 3.2.

Pour permettre cette recherche, d'autres services sont nécessaires, notamment pour récupérer la liste des types d'échantillons ou la liste des projets.

### 1.3.2 Liste des projets ou sous-collections

Ce service doit permettre de récupérer la liste des projets ou sous-collections autorisés pour un utilisateur, pour qu'ils puissent servir dans le cadre de la recherche.

### 1.3.3 Liste des types d'échantillons

Ce service récupère la liste exhaustive des types d'échantillons utilisés dans l'instance distante, pour une utilisation dans le cadre de la recherche.

### 1.3.4 Liste des types d'identifiants secondaires

Ce service récupère la liste exhaustive des types d'échantillons secondaires, pour une utilisation dans le cadre de la recherche des échantillons.

### 1.3.5 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 sont suffisamment complètes pour être intégrées dans l'instance *Collec* courante, par exemple pour éviter de les ressaisir suite au prêt d'un échantillon par un laboratoire.

Elles permettent également une visualisation détaillée de l'échantillon considéré, et contiennent, le cas échéant [3], les données « métiers » associées.

### 1.3.6 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

Les instances Collec sont prévues pour donner un accès en lecture à toutes les données des échantillons disponibles, dès lors que l'utilisateur s'est identifié. Cela permet ainsi de connaître, par exemple, le produit utilisé pour la conservation ou d'autres informations nécessaires pour la gestion quotidienne des collections.

Toutefois, les données « métiers »ne sont accessibles qu'aux personnes habilitées à les consulter. Dans la pratique, les échantillons sont associés à un projet, et seuls les utilisateurs rattachés à celui-ci peuvent prendre connaissance de ces informations. Cela impose une gestion particulière des accès lors de l'interrogation par l'intermédiaire des services web, qui sera décrite dans le chapitre 2.

Le protocole d'identification des utilisateurs dans l'instance distante est basé sur le protocole Oauth v2.

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

# Chapitre 2

# Gestion des habilitations

# Chapitre 3

# Description des services web

## 3.1 Remarques générales

### 3.1.1 Format des données transmises ou reçues

### 3.1.2 Codes d'erreur

## 3.2 Recherche des échantillons

### 3.2.1 Variables de recherche

Les variables peuvent être indiquées soit directement dans la requête GET, soit dans le champ *jsonval*, encodé 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 |

| Code | Type | Description |
|---|---|---|
| projectid | integer | identifiant du projet ou de la sous-collection (code issu du résultat du service web 3.5) |
| sampletypeid | integer | identifiant du type d'échantillon (code issu du résultat du service web 3.6) |
| idtype | integer | type d'identifiant (code issu du résultat du service web 3.7) |
| idval | varchar | identifiant recherché selon le type spécifié dans *idtype* |

Exemple de fichier :

```
{"uidstart": 25,
"uidend": 50,
"datestart": "2017-01-01",
"dateend": "2017-06-30",
"idtype": 2,
"idval": "AB01"
}
```

### 3.2.2 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 |
| DATAMETIER | 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 |

## 3.3 Lecture d'un échantillon

## 3.4 Lecture d'un jeu d'échantillons

## 3.5 Liste des projets ou sous-collections

## 3.6 Liste des types d'échantillons

## 3.7 Liste des types d'identifiants

# Chapitre 4

# Implémentation technique dans Collec

## 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* [1] ;
- 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*.

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

### 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 [2], le code est généré à partir des 12 premiers chiffres hexadécimaux de la fonction de calcul d'empreintes SHA-1 :

```
contenu...
```

Cette valeur est calculée « à la volée ».

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).

### 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).

Ainsi, 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 ;

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

— l'URI correspondante ;
— le type d'identification prévu : *OauthV1*, *OauthV2*, pas d'identification.

# Table des matières