

Table des matières

| | | |
|----------|--|-----------|
| 1 | Le logiciel Collec | 1 |
| 1.1 | Historique | 1 |
| 1.2 | Fonctionnalités générales | 2 |
| 1.3 | Technologie employée | 3 |
| 1.3.1 | Base de données | 3 |
| 1.3.2 | Langage de développement et framework utilisé | 3 |
| 1.3.3 | Liste des composants externes utilisés | 3 |
| 1.4 | Sécurité | 4 |
| 1.5 | Licence | 4 |
| 2 | Installer le logiciel | 5 |
| 2.1 | Consultez la documentation du framework ! | 5 |
| 2.2 | Configuration du serveur | 5 |
| 2.2.1 | Configuration d'Apache | 5 |
| 2.2.2 | Modules PHP nécessaires | 5 |
| 2.2.3 | Configuration de l'hôte virtuel et de SSL | 6 |
| 2.2.4 | Configuration du dossier d'installation | 6 |
| 2.3 | Configurer l'application | 8 |
| 2.3.1 | Connexion à la base de données | 8 |
| 2.3.2 | Identification des utilisateurs | 9 |
| 2.3.3 | Configuration de l'accès à l'annuaire LDAP | 9 |
| 2.3.4 | Paramètres spécifiques | 10 |
| 2.4 | Créer la base de données | 10 |
| 2.4.1 | Créer les tables de gestion des droits | 11 |
| 2.4.2 | Créer les tables applicatives | 11 |
| 2.4.3 | Login de connexion | 11 |
| 2.4.4 | Droits sur les tables | 11 |
| 2.4.5 | Scripts de modification | 12 |
| 2.5 | Mise en production | 12 |
| 3 | Administrer l'application | 13 |
| 3.1 | Gérer les droits | 13 |
| 3.1.1 | Principe général | 13 |
| 3.1.2 | Créer un nouvel utilisateur | 15 |
| 3.1.3 | Créer un login utilisé dans la gestion des droits | 16 |
| 3.1.4 | Définir les groupes d'utilisateur | 16 |
| 3.1.5 | Créer une application | 17 |
| 3.1.6 | Définir les droits utilisables dans l'application | 18 |
| 3.1.7 | Cas particulier des groupes et des logins issus d'un annuaire LDAP | 18 |
| 3.2 | Droits spécifiques de l'application COLLEC | 19 |

| | | |
|----------|--|-----------|
| 3.2.1 | Droits à positionner | 19 |
| 3.2.2 | Gestion des projets | 19 |
| 3.3 | Configurer les paramètres généraux | 20 |
| 3.4 | Créer ou modifier un modèle d'étiquettes | 20 |
| 3.4.1 | Définir le contenu du QRcode | 21 |
| 3.4.2 | Configuration du fichier XSL | 22 |
| 3.5 | Gestion des traces | 24 |
| 4 | Comment faire pour ? | 27 |
| 4.1 | Générer une liste d'échantillons vides | 27 |
| 4.1.1 | Structure du fichier CSV | 27 |
| 4.1.2 | Procédure d'import | 28 |
| 4.1.3 | Autre usage | 29 |
| A | Structure de la base de données | 31 |
| | Bibliographie | 49 |

Chapitre 1

Le logiciel Collec

Historique

L'unité de recherche *Écosystèmes aquatiques et changements globaux* d'IRSTEA, à Cestas (33), récolte et manipule des échantillons prélevés sur le terrain (ou plutôt, principalement dans l'eau – estuaires, lacs, rivières...), et les stocke, parfois sur des durées très longues : certaines campagnes de collecte ont eu lieu il y a plus de 40 ans.

De plus en plus, des échantillons anciens sont réanalysés (analyses génétiques, étude des ossements des oreilles ou otolithes...), au gré des questions scientifiques à traiter. Le besoin de recourir à un logiciel pour gérer ces matériels est devenu une priorité.

Dans un premier temps, quelques logiciels open-source susceptibles d'être utilisés ont été étudiés. Toutefois, leurs limites sont vite apparues : problème de pérennité, ancienneté du code, modèle de distribution parfois insatisfaisant (une licence open-source est obligatoire pour assurer la pérennité à longue échéance), résistance aux attaques informatiques, fonctionnalités insuffisantes ou inadaptées au besoin.

Dans un second temps, une étude des besoins réels a été menée. De nouvelles fonctionnalités ont été rajoutées, comme la gestion du stock de matériel utilisé sur le terrain, stocké dans un hangar.

L'unité de recherche s'intégrant au niveau régional avec d'autres organismes, des collaborations avec l'Observatoire Aquitain des Sciences de l'Univers (OASU) ou l'université de La Rochelle ont été envisagées. Des échanges productifs ont ainsi pu être mis en place, entre autres sur la gestion de l'étiquetage et le scannage des codes-barres.

Le logiciel a largement évolué suite à ces échanges, de nombreuses fonctionnalités ont été rajoutées ou modifiées pour tenir compte des besoins des partenaires potentiels.

Les délais de développement de la première version opérationnelle se sont étalés sur 9 mois, entre la définition des besoins et le développement proprement dit. Cette seconde phase a été réalisée entre juin et octobre 2016. Environ 160 heures de travail auront été nécessaires pour écrire le logiciel. Le code comprend environ 8900 lignes (commentaires compris), dont 3800 concernent l'affichage des pages web. Il a été écrit en PHP, les pages web générées en HTML et Javascript.

Fonctionnalités générales

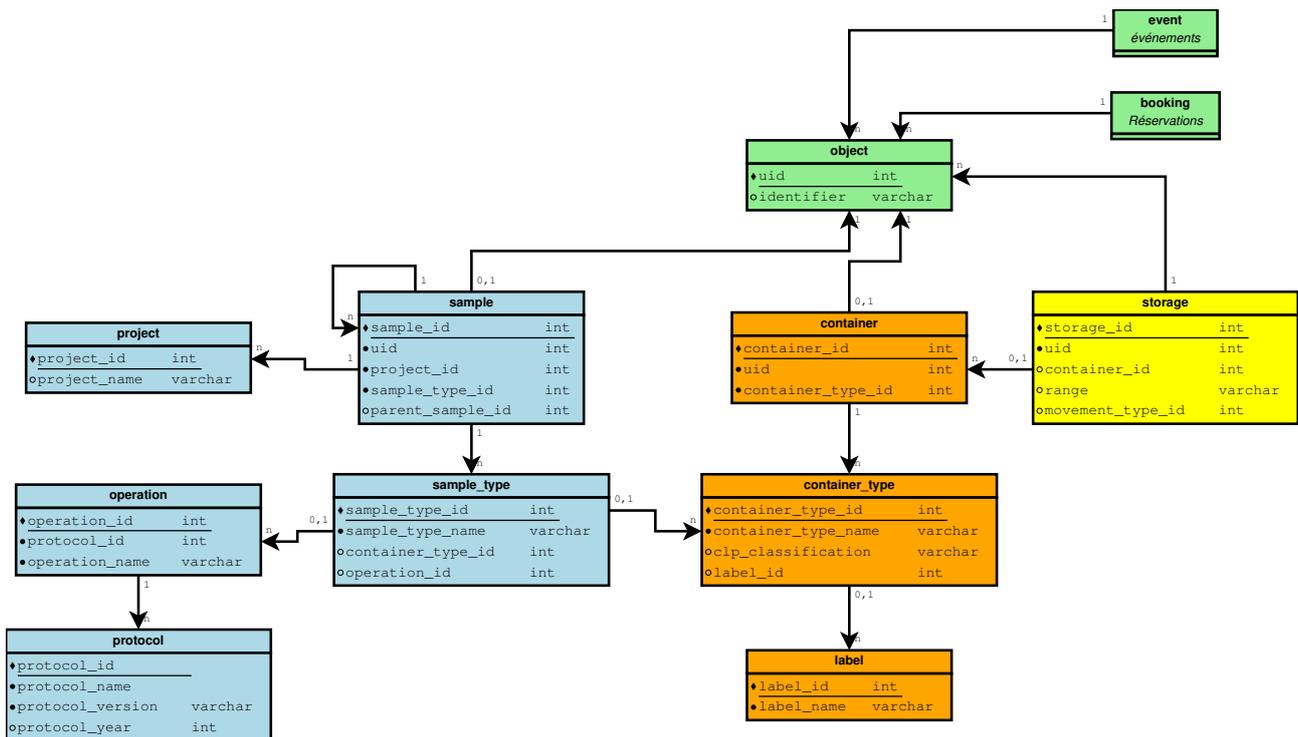


FIGURE 1.1 – Schéma simplifié de la base de données

Deux types d'objets sont manipulés dans l'application :

- des containers, qui peuvent contenir des objets de tout type : d'autres containers ou des échantillons. Ils peuvent être de différentes nature : site, bâtiment, salle, armoire, caisse, éprouvette...
- des échantillons, qui peuvent être associés à un type de container : il y a de nombreux cas où l'échantillon lui-même se confond avec son contenant, par exemple quand le résultat d'une pêche n'est pas trié et est stocké dans un bocal.

Un échantillon ou un container sont issus d'un objet unique, qui est doté :

- d'un numéro unique, l'**UID**, qui sert de référence dans le logiciel ;
- d'un identifiant métier, qui servira à le retrouver facilement (le logiciel permet également de définir d'autres identifiants).

Un objet peut subir un certain nombre d'événements, voire être réservé (fonctionnalité très simplifiée, seul le recouvrement de deux périodes de réservation est signalé).

Tout type de container peut être associé à un modèle d'étiquettes. Les étiquettes peuvent comprendre un code-barre 2D de type QRCode, qui pourra être lu soit à partir d'un terminal dédié (douchette), soit avec une tablette ou un smartphone, l'application disposant d'un module capable d'activer la caméra depuis le navigateur et de scanner le code-barre.

Il est aussi possible d'imprimer des étiquettes tant pour des containers que pour des échantillons, si ces derniers sont d'un type associé à un type de container.

Un échantillon est obligatoirement rattaché à un projet. Seuls les membres du projet considéré peuvent modifier les informations le concernant.

Un type d'échantillon peut être associé à une opération particulière d'un protocole, ce qui permet de mieux tracer l'évolution d'un prélèvement.

Un échantillon peut être subdivisé en d'autres échantillons. Par exemple, des otolithes (os de l'oreille) peuvent être extraits d'un poisson. Deux nouveaux échantillons sont ainsi créés (d'un autre type), qui resteront associés au poisson parent.

Enfin, dans certains cas de figure, un échantillon peut être composé de plusieurs éléments indifférenciés, par exemple plusieurs écailles de poisson prélevées et conservées ensemble. Le logiciel permet d'indiquer les prélèvements et les restitutions réalisées, et affiche le solde (théorique !) restant.

Technologie employée

Base de données

L'application a été conçue pour fonctionner avec Postgresql, en version 9.5. Les versions antérieures peuvent être utilisées, mais seule cette version dispose d'un type de données JSON qui permet de stocker les informations métiers (partie non développée dans la version 1.0).

Langage de développement et framework utilisé

Le logiciel a été écrit en PHP, en s'appuyant sur le framework *Prototypephp* [12], développé parallèlement par l'auteur du logiciel.

Il utilise la classe *Smarty* [14] pour gérer l'affichage des pages HTML. Celles-ci sont générées en utilisant *Jquery* [6] et divers composants associés. Le rendu général est réalisé avec *Bootstrap* [4].

Les étiquettes sont générées en utilisant FOP [2], une classe Java qui crée des fichiers PDF à partir d'un fichier XML contenant les données et un fichier de transformation au format XSL.

Liste des composants externes utilisés

| Nom | Version | Licence | Usage | Site |
|------------------------|-------------------|------------|--|--|
| PrototypePHP | branche bootstrap | LGPL | Framework | github.com/ equinton/ prototypephp |
| Smarty | 3.1.24 | LGPL | Générateur de pages HTML | www.smarty.net |
| PHPCAS | 1.3.3 | Apache 2.0 | Identification auprès d'un serveur CAS | wiki.jasig.org/ display/ CASC/ phpCAS |
| Bootstrap | | MIT | Présentation HTML | get.bootstrap.com |
| Carhartl-jquery-cookie | 1.5.1 | MIT | Gestion des cookies dans le navigateur | github.com/ carhartl/ jquery-cookie |
| Datatables | 1.10.12 | MIT | Affichage des tableaux HTML | www.datatables.net |
| Datetime-moment | | MIT | Formatage des dates dans les tableaux | datatables.net/ plug-ins/ sorting/ datetime-moment |
| Moment | | MIT | Composant utilisé par datetime-moment | momentjs.com |
| JQuery | 1.12.4 | ≈ BSD | Commandes Javascript | jquery.com |

| Nom | Version | Licence | Usage | Site |
|-------------------------|---------|---------|---|--|
| JQuery-ui | 1.12.0 | ≈ BSD | Commandes Javascript pour les rendus graphiques | jqueryui.com |
| Jquery-timepicker-addon | | MIT | Time picker | github.com/trentrichardson/jQuery-Timepicker-Addon |
| Magnific-popup | 0.9.9 | MIT | Affichage des photos | dimsemenov.com/plugins/magnific-popup/ |
| Smartmenus | | MIT | Génération du menu HTML | www.smartmenus.org |
| Openlayers | 3.12.1 | BSD | Affichage des cartes | openlayers.org |
| Html5-qrcode | | MIT | Lecture des QRcodes | github.com/dwa012/html5-qrcode |

TABLE 1.1: Table des composants externes utilisés dans l'application

Sécurité

L'application a été conçue pour résister aux attaques dites opportunistes selon la nomenclature ASVS v3 [7] de l'OWASP [8]. Des tests d'attaque ont été réalisés en août 2016 avec le logiciel ZapProxy [9], et n'ont pas détecté de faiblesse particulière.

La gestion des droits est conçue pour :

- qu'un utilisateur, membre d'un projet, ne puisse modifier que les échantillons qui y sont rattachés ;
- que tout utilisateur disposant des droits de gestion peut procéder à une entrée ou une sortie d'un objet, quel qu'il soit ;
- que les responsables d'un projet soient les seuls à pouvoir modifier les paramètres comme les types d'échantillons ou de containers.

L'analyse de sécurité a mis en exergue un besoin de ne pas perdre d'information : si un échantillon est étiqueté et rangé, et que l'information est perdue, il y a de gros risques de ne plus pouvoir l'utiliser ultérieurement. Cela impose la mise en place d'un mécanisme de réplication de la base de données, à implémenter – ou faire implémenter par des administrateurs du système – directement dans PostgreSQL.

Licence

Le logiciel est diffusé selon les termes de la licence GNU AFFERO GENERAL PUBLIC LICENSE version 3, en date du 19 novembre 2007 [5].

Chapitre 2

Installer le logiciel

Consultez la documentation du framework !

Le logiciel a été conçu à partir du framework *Prototypephp*. La documentation associée [11] récapitule l'ensemble des informations nécessaires pour réaliser l'installation générale (configuration du serveur, définition des droits d'accès, etc.).

De nombreux passages ont été repris ici, mais il n'est pas inutile de se référer au document d'origine.

Configuration du serveur

La configuration est donnée pour un serveur Linux fonctionnant avec Ubuntu 16.04 LTS Server. Elle peut bien sûr être adaptée à d'autres distributions Linux. Par contre, rien n'a été prévu pour faire fonctionner l'application directement dans une plate-forme windows, même si, en théorie, cela devrait être possible.

Configuration d'Apache

Les modules suivants doivent être activés :

```
a2enmod ssl
a2enmod headers
a2enmod rewrite
```

Modules PHP nécessaires

Modules complémentaires nécessaires :

- *php-mbstring*
- *php-pgsql*
- *php-xdebug* pour les phases de mise au point
- *php-curl* (ou *php5-curl* pour PHP5) pour l'identification via un serveur CAS.

La génération des étiquettes nécessite les paquetages suivants :

- *php-gd* (ou *php5-gd* pour PHP5)
- *fop* (qui inclut des bibliothèques java)

Le stockage et l'affichage des photos nécessite :

- *php-imagick* (ou *php5-imagick* pour PHP5)

Configuration de l'hôte virtuel et de SSL

L'application ne fonctionne qu'en mode SSL, les cookies de session n'étant pas transmis sur des liens non chiffrés. Voici un exemple de configuration à insérer dans le fichier */etc/apache2/sites-available/default-ssl*

```
<Directory /var/www/html>
    Options FollowSymLinks MultiViews
    AllowOverride all
    Order allow,deny
    allow from all
</Directory>

SSLProtocol All -SSLv2 -SSLv3
SSLHonorCipherOrder On
SSLCipherSuite ECDHE-RSA-AES128-SHA256:AES128-GCM-SHA256:HIGH:
!MD5:!aNULL:!EDH:!RC4
SSLCompression off
```

La chaîne *SSLCipherSuite* est donnée à titre d'exemple, d'autres configurations plus modernes peuvent être implémentées (cf. [1]).

Pour activer le mode SSL dans Apache :

```
chmod -R g+r /etc/ssl/private
usermod www-data -a -G ssl-cert
a2ensite default-ssl
service apache2 restart
```

Configuration du dossier d'installation

Mécanisme pour faire cohabiter plusieurs instances avec le même code

Il est possible d'utiliser le même code applicatif pour alimenter des bases de données différentes (ou des données stockées dans des schémas différents). Cette fonctionnalité est basée sur l'attribution d'entrées DNS différentes.

Le mécanisme est décrit dans le schéma 2.2.4, page 7.

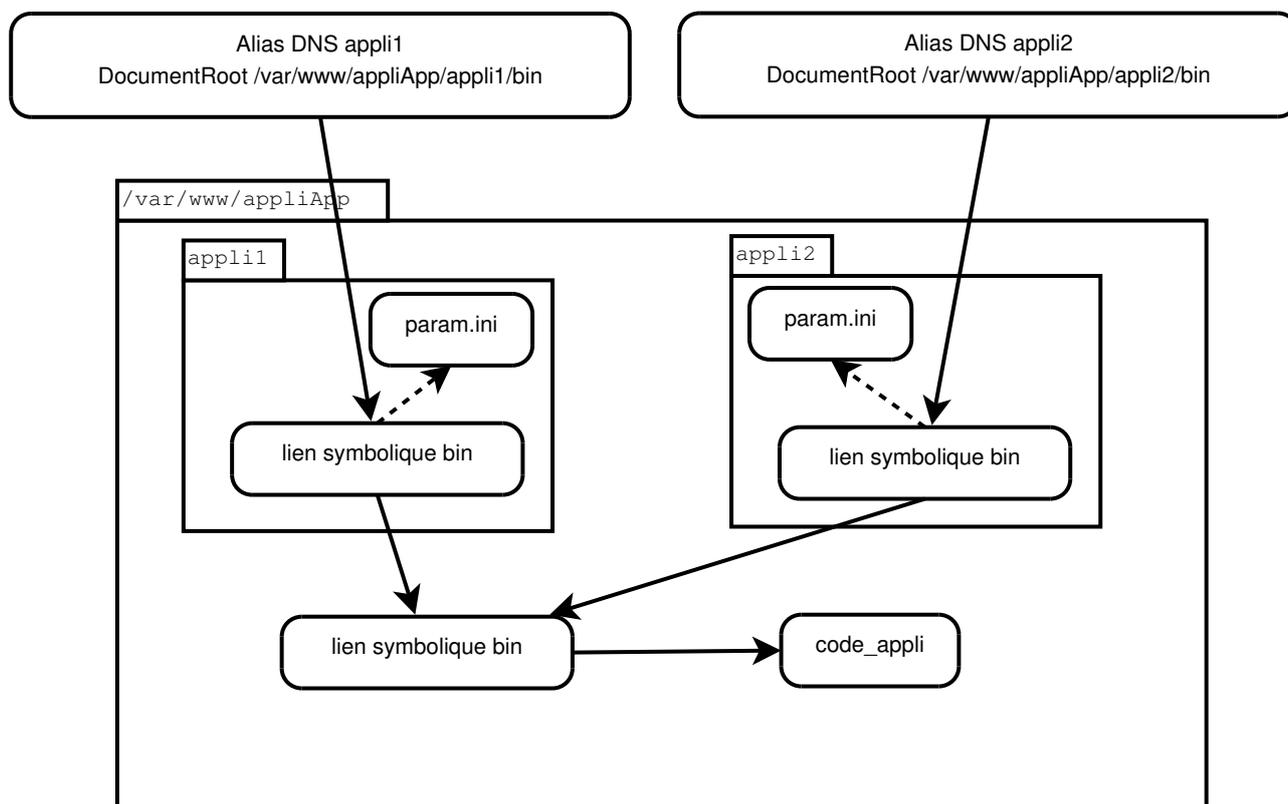


FIGURE 2.1 – Schéma général d’implémentation pour utiliser le même code avec des noms d’application et des jeux de données différents

Dans le paramétrage de l’alias DNS (en principe, dans */etc/apache2/sites-available*), l’application pointe vers le dossier */var/www/appliApp/appli1/bin*. */var/www* correspond à la racine du site web, *appliApp* au dossier racine de l’application, *appli1* au dossier spécifique de l’alias DNS. Ce dossier *appli1* ne contient que deux fichiers : *param.ini*, qui contient les paramètres spécifiques, et *bin*, qui est un lien symbolique vers le dossier *../bin*.

Le dossier *../bin* (donc, dans */var/www/appliApp*) est lui aussi un alias qui pointe vers le code réel de l’application, ici *code_appli*. Le fichier *param.inc.php* décrit l’entrée suivante :

```
$paramIniFile = "../param.ini";
```

Le fichier *param.ini* sera cherché dans le dossier parent du code de l’application, c’est à dire soit dans *appli1*, soit dans *appli2* dans cet exemple. Il suffit qu’il contienne les paramètres adéquats pour rendre l’application utilisable dans des contextes différents à partir du même code initial.

Le fichier *param.ini* est le dernier qui est traité par l’application pour récupérer les paramètres. Ceux-ci sont lus dans l’ordre suivant :

param/param.default.inc.php → param/param.inc.php → ../param.ini

param.ini contiendra les entrées spécifiques liées au DNS utilisé pour accéder à l’application, en principe tout ou partie de celles-ci :

```
APPLI_titre=Gestion des collections EABX
BDD_schema=col, public, gacl
BDD_login=compte_de_connexion
BDD_passwd=mot_de_passe_de_connexion
BDD_dsn=pgsql:host=serveur;dbname=base_de_donnees;sslmode=require
GACL_aco=col
APPLI_code=proto
```

Droits à attribuer au serveur web

Le serveur web doit pouvoir accéder en lecture à l'ensemble des fichiers de l'application, et en écriture à deux dossiers :

- *display/templates_c* : fichier utilisé par Smarty pour compiler les modèles de documents HTML ;
- *temp* : dossier de génération des images et des fichiers temporaires.

Voici un exemple de script utilisé pour positionner les droits :

```
#!/bin/bash
cp ../param/* param/
chmod -R 770 .
setfacl -R -m u:www-data:rx .
setfacl -R -m d:u:www-data:rx .
mkdir display/templates_c
setfacl -R -m u:www-data:rwx display/templates_c
setfacl -R -m d:u:www-data:rwx display/templates_c
setfacl -R -m u:www-data:rwx temp
setfacl -R -m d:u:www-data:rwx temp
setfacl -R -m d:o::- .
rm -Rf database
rm -Rf test
rm -f param.ini
```

Dans cet exemple, le dossier *../param* contient le fichier *param.inc.php*, qui dispose des paramètres spécifiques à l'implémentation.

Le script est à lancer à la racine du dossier contenant l'application.

Configurer l'application

L'application est configurable par l'intermédiaire de trois fichiers, comme nous venons de le voir : **param/param.default.inc.php** → **param/param.inc.php** → **../param.ini**

Le premier fichier contient les paramètres par défaut. Il est systématiquement fourni à chaque nouvelle version de l'application.

Le second est spécifique de l'implémentation. Il comprend notamment les informations liées à la connexion à la base de données, à la méthode d'identification, ou à la recherche des attributs dans l'annuaire LDAP.

le troisième est destiné à offrir la possibilité d'accéder, à partir du même code applicatif, à plusieurs bases de données différentes (*cf. 2.2.4 Mécanisme pour faire cohabiter plusieurs instances avec le même code*).

Voici les principaux paramètres utilisés :

Connexion à la base de données

Dans la pratique, deux connexions sont nécessaires : l'une pour accéder à la base des droits, l'autre aux données proprement dites. Voici les paramètres à définir :

| Variable | Signification |
|------------|---|
| BDD_login | compte de connexion à la base de données |
| BDD_passwd | mot de passe associé |
| BDD_dsn | adresse de la base de données sous forme normalisée |
| BDD_schema | schéma utilisé (plusieurs schémas peuvent être décrits, en les séparant par une virgule - fonctionnement propre à PostgreSQL) |

| Variable | Signification |
|---------------|---|
| GACL_dblogin | compte de connexion à la base de données des droits |
| GACL_dbpasswd | mot de passe associé |
| GACL_dsn | adresse normalisée |
| GACL_schema | schéma utilisé |
| GACL_aco | nom du code de l'application utilisé dans la gestion des droits |

TABLE 2.1: Variables utilisées pour paramétrer les connexions

Identification des utilisateurs

| Variable | Signification |
|-----------------------|---|
| ident_type | Type d'identification supporté. L'application peut gérer BDD (uniquement en base de données), LDAP (uniquement à partir d'un annuaire LDAP) LDAP-BDD (d'abord identification en annuaire LDAP, puis en base de données), et CAS (serveur d'identification <i>Common Access Service</i> ¹) |
| CAS_plugin | Nom du plugin utilisé pour une connexion CAS |
| CAS_address | Adresse du serveur CAS |
| CAS_port | Systématiquement 443 (connexion chiffrée) |
| LDAP | tableau contenant tous les paramètres nécessaires pour une identification LDAP |
| privateKey | clé privée utilisée pour générer les jetons d'identification |
| pubKey | clé publique utilisée pour générer les jetons d'identification |
| tokenIdentityValidity | durée de validité, en secondes, des jetons d'identification |

TABLE 2.2: Variables utilisées pour paramétrer l'identification

L'application permet de conserver l'identification plus longtemps que celle définie dans le serveur, en jouant la connexion avec un jeton d'identification chiffré. Cela évite, par exemple, de devoir se ré-identifier toutes les heures si on accède au logiciel à partir d'un terminal mobile (smartphone ou tablette, par exemple).

Les trois dernières variables permettent de configurer ce mode d'identification. Pour plus d'informations, consultez [13].

Configuration de l'accès à l'annuaire LDAP

Les paramètres LDAP sont stockés dans un tableau :

```
$LDAP = array (
    " address " => " localhost " ,
    " port " => 389 ,
    " rdn " => " cn=manager , dc=example , dc=com " ,
    " basedn " => " ou=people , ou=example , o=societe , c=fr " ,
    " user_attr " => " uid " ,
    " v3 " => true ,
    " tls " => false ,
```

1. serveur externe gérant l'identification des utilisateurs, et renvoyant à l'application le login utilisé

```

" groupSupport "=>true ,
" groupAttrib "=>" supannentiteaffectation " ,
" commonNameAttrib "=>" displayname " ,
" mailAttrib "=>" mail " ,
' attributgroupname ' => "cn" ,
' attributloginname ' => "memberuid" ,
' basedngroup ' => 'ou=example ,o=societe ,c=fr '
);

```

L'application peut non seulement identifier les utilisateurs auprès de l'annuaire LDAP, mais également récupérer les groupes auxquels ils appartiennent dans celui-ci.

Voici les paramètres à indiquer dans ce cas de figure :

| Variable | Signification |
|-------------------|--|
| address | adresse de l'annuaire |
| port | 389 en mode non chiffré, 636 en mode chiffré |
| rdn | compte de connexion, si nécessaire |
| basedn | base de recherche des utilisateurs |
| user_attrib | nom du champ contenant le login à tester |
| v3 | toujours à <i>true</i> |
| tls | <i>true</i> en mode chiffré |
| groupSupport | true si l'application recherche les groupes d'appartenance du login dans l'annuaire |
| groupAttrib | Nom de l'attribut contenant la liste des groupes d'appartenance |
| commonNameAttrib | Nom de l'attribut contenant le nom de l'utilisateur |
| mailAttrib | Nom de l'attribut contenant l'adresse mail de l'utilisateur |
| attributgroupname | Attribut contenant le nom du groupe lors de la recherche des groupes (cn par défaut) |
| attributloginname | attribut contenant les membres d'un groupe |
| basedngroup | base de recherche des groupes |

TABLE 2.3: Variables utilisées pour paramétrer l'accès à l'annuaire LDAP

Paramètres spécifiques

| Variable | Signification |
|------------|---|
| GACL_aco | nom du code de l'application utilisé dans la gestion des droits (<i>cf.</i> 3.1 <i>Gérer les droits</i> , page 13) |
| APPLI_code | Code interne de l'application. Ce code est essentiel : il sera inscrit dans les codes-barres générés, pour s'assurer qu'un échantillon est bien issu de la base de données concernée |

TABLE 2.4: Variables spécifiques

Créer la base de données

La base de données est composée de deux schémas : l'un pour stocker les informations d'identification, les droits d'accès et les traces, l'autre pour les données proprement dites.

Le schéma *public* ne devrait jamais être utilisé pour stocker l'information : réservez-le pour les composants communs, comme Postgis si c'est nécessaire.

Les tables de gestion des droits peuvent être communes à plusieurs jeux / applications différentes : la variable *GACL_aco* permet de séparer la gestion des droits pour chaque application, tout en travaillant à partir des mêmes utilisateurs (répartis le cas échéant dans des groupes différents selon le jeu de données considéré).

Les scripts de création des schémas dans la base de données sont stockés dans le dossier *install*.

Créer les tables de gestion des droits

Script à utiliser : *gac_create.sql*. Les tables nécessaires vont être créées dans le schéma *gac* (ne modifiez pas le nom du schéma).

Le script crée un compte d'administration par défaut :

— login : **admin**

— mot de passe : **password**

Il devra être supprimé quand un autre compte d'administration aura été créé.

Les droits par défaut sont positionnés pour le projet *appli* (variable *\$GACL_aco* dans les fichiers de paramètres de l'application).

Créer les tables applicatives

Script à utiliser : *col_create.sql*.

Par défaut, le script crée un schéma appelé *col*. Il est possible de créer plusieurs schémas différents, si l'application supporte plusieurs jeux de données (cf. 2.2.4 *Mécanisme pour faire cohabiter plusieurs instances avec le même code*, page 6). Dans ce cas de figure, remplacez *col* par le nom du schéma voulu dans les deux premières lignes du script.

Login de connexion

Il est fortement conseillé de créer deux logins de connexion, un pour le schéma des droits, l'autre pour les schémas applicatifs. Ces logins ne doivent pouvoir être utilisés que depuis le serveur web hébergeant l'application.

Cette opération est possible en modifiant le fichier */etc/postgresql/9.5/main/pg_hba.conf* selon ce principe :

```
# Connexions pour les serveurs web
host nom_database userGacl adresse_serveur/32 md5
host nom_database userData adresse_serveur/32 md5
```

et en rechargeant ensuite la configuration de Postgresql avec la commande :

```
service postgresql reload
```

Droits sur les tables

Le compte utilisé pour la connexion au schéma des droits doit pouvoir modifier les informations présentes dans l'ensemble des tables de *gac*. Il ne doit pas pouvoir accéder aux autres schémas (hormis *public*).

Le compte utilisé pour accéder aux schémas des données doit pouvoir modifier l'ensemble des informations dans les schémas de données, et lire la table *gac.aclgroup*.

Le plus simple est d'utiliser le logiciel *pgAdmin* [10] pour attribuer les droits.

Scripts de modification

Lors de la livraison de nouvelles versions, il est possible que des scripts de modification soient livrés pour mettre à niveau la base de données. Ces scripts doivent être exécutés dans tous les schémas contenant des données applicatives.

Mise en production

Une fois l'application configurée, et après avoir créé un nouveau compte d'administration :

- supprimez le compte *admin*, livré par défaut, qui ne doit pas être conservé. Sa désactivation n'est pas suffisante : si pour une raison ou pour une autre le compte est réactivé, n'importe qui pourra récupérer les droits totaux ;
- supprimez le dossier *install* qui contient les scripts de création des tables ;
- déplacez le dossier *database*, qui contient la documentation d'installation et de configuration (elle n'a pas à rester accessible depuis le site web) ;
- faites une revue des droits, pour vous assurer que tout est correctement configuré.

Vous pouvez également tester si la configuration du serveur est correcte en recourant à ZAProxy [9], qui analysera la communication entre le serveur et un navigateur et identifiera les problèmes éventuels de non conformité (mauvaise réécriture des entêtes HTML suite à une mauvaise configuration du serveur Apache, par exemple).

Chapitre 3

Administrer l'application

Gérer les droits

Principe général

Les droits sont gérés selon le principe initialement utilisé dans la bibliothèque PHPGACL [3], aujourd'hui obsolète.

Les logins sont déclarés dans des groupes organisés de manière hiérarchique : un groupe hérite des droits attribués à ses parents.

Les droits utilisés dans le logiciel sont associés à des groupes. Il est possible d'attribuer plusieurs droits à un même groupe, et un droit peut être détenu par des groupes différents.

Si le paramètre *\$LDAP["groupSupport"]* est positionné à *true*, les groupes dont fait partie le compte LDAP sont également récupérés. Si ces groupes se voient attribués des droits, les comptes associés les récupéreront automatiquement.

Voici le schéma des tables utilisées pour gérer les droits :

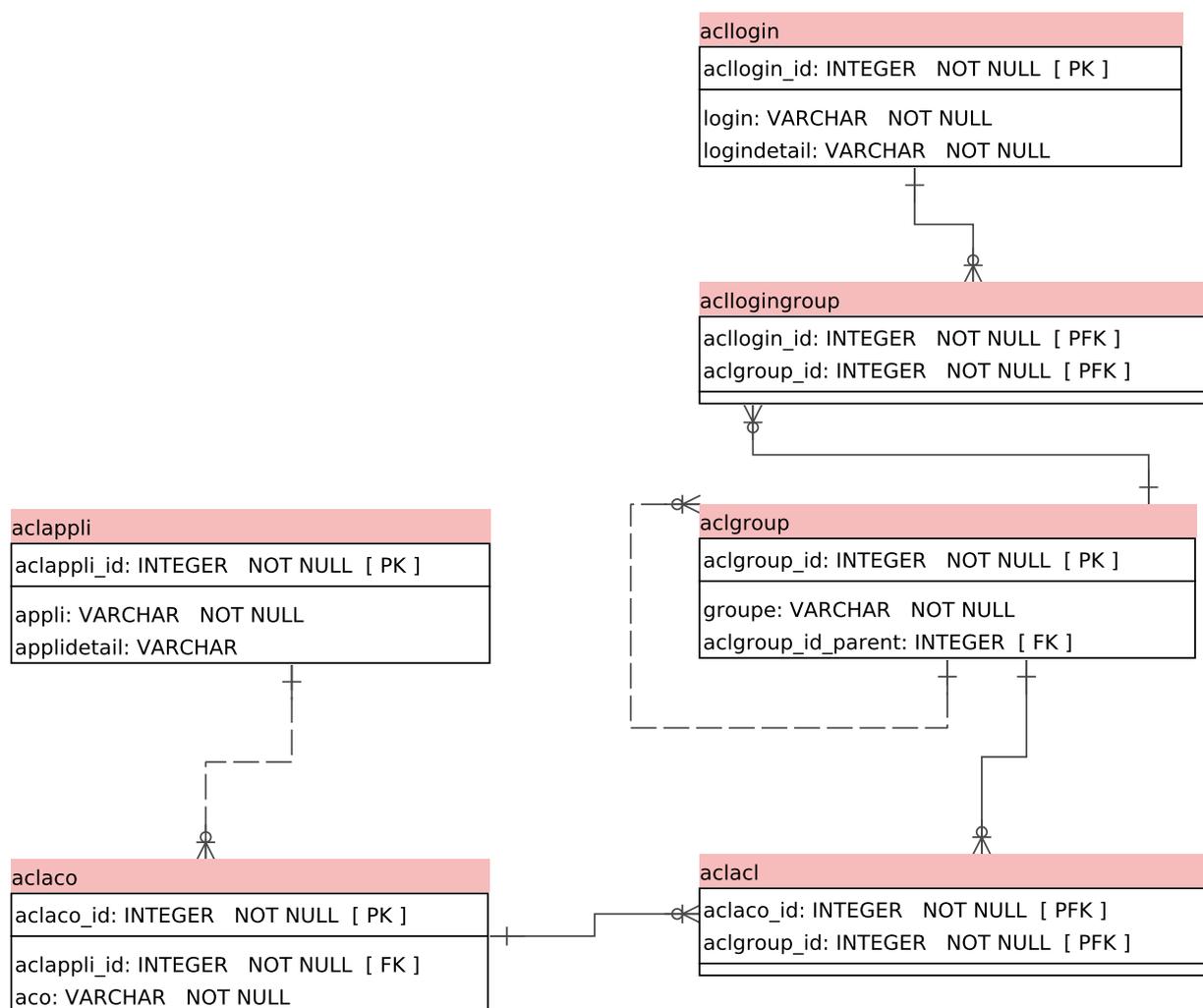


FIGURE 3.1 – Schéma des tables utilisées pour gérer les droits

Voici la description des tables :

acllogin : liste des logins utilisés. Si un compte est créé dans la base locale d'identification, un enregistrement est également créé dans cette table. Pour les identifications LDAP ou CAS, ils doivent être identiques. Si seuls les groupes LDAP sont utilisés pour un compte, il n'a pas besoin d'être décrit ici ;

aclappli : liste des applications gérées. Il est possible de gérer, à partir de la même base de données, plusieurs ensembles de droits, qui utilisent les mêmes logins.

aclaco : liste des droits déclarés dans l'application ;

aclgroup : liste des groupes contenant les logins, et qui détiennent les droits. Un groupe peut hériter d'un autre groupe. Les droits associés au groupe parent sont également attribués au groupe hérité ;

aclloggingroup : table permettant de déclarer les logins associés à un groupe ;

aclacl : table décrivant les droits détenus par un groupe.

Le module d'administration permet de saisir toutes ces informations. Il faut que l'utilisateur dispose du droit *admin*, c'est à dire qu'il fasse partie du groupe *admin* (configuration par défaut à l'initialisation de la base des droits) pour pouvoir accéder à ces fonctions.

Créer un nouvel utilisateur

Les utilisateurs peuvent être issus soit de l'annuaire LDAP, soit de la base interne. Pour créer un nouvel utilisateur dans la base locale :

- *Administration* → *Liste des comptes*
- *Nouveau login*
- renseignez au minimum le login.



Login * :

Nom :

Prénom :

Mèl :

Date :

Mot de passe : 

Répétez le mot de passe : 

Générez un mot de passe aléatoire

actif oui non

FIGURE 3.2 – Écran de saisie d'un login de connexion

Pour créer le mot de passe, vous pouvez cliquer sur le bouton *Générez*, qui en générera un automatiquement. Envoyez-le par mél à son destinataire (par *copier-coller*), en lui demandant de le modifier à la première connexion (icône en forme de clé, dans le bandeau, en haut à droite).

Les mots de passe doivent respecter les règles suivantes :

- ils doivent avoir une longueur minimale de 8 caractères ;
- ils doivent comprendre trois types de caractères différents parmi les minuscules, majuscules, chiffres et caractères de ponctuation ;
- ils ne peuvent pas être réutilisés pour le même login ;
- les mots de passe n'expirent pas.

Les mots de passe sont stockés sous forme d'empreinte, calculée en rajoutant un sel¹ et encodés en SHA256 : ils ne peuvent pas être retrouvés en cas de perte.

1. chaîne de caractère rajoutée au mot de passe – en général le login ou un identifiant – qui permet d'éviter que deux mots de passe identiques, associés à deux logins différents, aient la même empreinte

L'application n'intègre pas de module permettant de régénérer automatiquement un mot de passe en cas de perte : c'est au responsable applicatif d'en fournir un nouveau.

La création d'un compte entraîne la création d'une entrée identique dans la table des *acllogin*, utilisée pour attribuer les droits.

Pour désactiver temporairement un compte, sélectionnez *non* dans la zone *actif*. Si le compte ne doit plus être utilisé, supprimez-le.

Attention : si le compte disposait des droits d'administration, assurez-vous que vous avez toujours un compte disposant des mêmes droits avant la suppression.

Créer un login utilisé dans la gestion des droits

Indépendamment du compte de connexion, qui peut être soit issu de la base interne, soit récupéré auprès d'un annuaire LDAP ou d'un serveur CAS, l'application a besoin de connaître les utilisateurs pour pouvoir leur attribuer des droits.

À partir du menu, choisissez *Administration* → *ACL - logins*.

Vous pouvez modifier un login existant ou en créer un nouveau. Dans ce cas, vous devrez indiquer au minimum le login utilisé (identique à celui qui est employé pour la connexion à l'application : base de données interne, annuaire LDAP, serveur CAS).

Modification d'un login (module de gestion des droits)

[Retour à la liste des logins](#)

Nom de l'utilisateur * : test

Login utilisé * : test

Valider Supprimer

*Donnée obligatoire

Droits attribués

consult

param

FIGURE 3.3 – Écran de modification d'un login dans le module de gestion des droits

Sous l'écran de saisie figurent la liste des droits attribués à un login (en modification, le calcul n'est réalisé qu'à l'affichage de la page).

Définir les groupes d'utilisateur

Les groupes d'utilisateurs sont gérés selon un mécanisme d'héritage. Un groupe de haut niveau hérite des groupes précédents : si des droits ont été attribués à un groupe de niveau inférieur, un login associé à un groupe de niveau supérieur les récupère également.

Pour définir les groupes, dans le menu, choisissez *Administration* → *ACL - groupes de logins*.

Nouveau groupe racine...

| Nom du groupe | Nombre de logins déclarés | Rajouter un groupe fils |
|---------------|---------------------------|-------------------------|
| admin | 2 | + |
| consult | | + |
| EABX | | + |
| aloston | | + |
| gestion | | + |
| projet | | + |
| param | 1 | + |

FIGURE 3.4 – Liste des groupes de logins

Ainsi, le login déclaré dans le groupe *param* récupérera les droits attribués aux groupes *projet*, *gestion* et *consult*.

Pour créer un groupe, deux possibilités :

- soit le groupe est à la base d'une nouvelle branche : utilisez alors *Nouveau groupe racine...* ;
- soit le groupe hérite d'un autre groupe : cliquez sur le signe + (*Rajouter un groupe fils*).

Vous pouvez indiquer les logins qui sont rattachés à ce groupe.

Créer une application

Le moteur utilisé pour faire fonctionner le logiciel COLLEC permet de gérer des droits différents pour des jeux de données différents, à partir du même code applicatif. Chaque couple *logiciel* ↔ *base de données* constitue donc une *application*, au sens de la gestion des droits.

Il est ainsi possible, à partir de la même base de données, de définir des droits différents selon les jeux de données utilisés (un jeu de données correspond à un schéma de base de données comprenant l'intégralité des tables applicatives).

À partir du menu, choisissez *Administration* → *ACL - droits* :

| Modifier  | Nom de l'application  | Description  |
|--|--|---|
|  | col | Gestion des collections d'échantillons |

FIGURE 3.5 – Liste des applications déclarées

Pour créer une nouvelle application, choisissez *Nouvelle application....*

*Donnée obligatoire

FIGURE 3.6 – Écran de saisie d'une application

Le nom de l'application doit impérativement correspondre à la valeur `$GACL_appli` dans les fichiers de paramètres : c'est ce qui permet au framework de savoir quels droits appliquer.

Définir les droits utilisables dans l'application

À partir de la liste des applications, cliquez sur le nom de celle pour laquelle vous voulez définir les droits utilisables. À partir de la liste, sélectionnez *Nouveau droit...*

FIGURE 3.7 – Écran de saisie des droits associés à une application

Le nom du droit doit être celui défini dans le corps de l'application (les droits sont positionnés dans les fichiers `param/actions.xml`, qui contient la liste des modules utilisables, et `param/menu.xml`, qui sert à générer le menu).

Indiquez les groupes d'utilisateurs qui seront associés au droit courant.

Cas particulier des groupes et des logins issus d'un annuaire LDAP

Si vous avez paramétré l'application pour qu'elle s'appuie sur un annuaire LDAP pour gérer l'affectation des utilisateurs dans les groupes, vous n'êtes pas obligés de les déclarer explicitement dans le module de gestion des droits.

Droits attribués à un groupe LDAP

Tous les utilisateurs d'un groupe héritent d'un droit dans l'application.

— définissez le nom du groupe (en respectant la casse) dans le tableau des groupes d'utilisateurs ;

- sélectionnez le nom de ce groupe dans les droits utilisables ;
- tous les utilisateurs de l'annuaire LDAP récupéreront automatiquement les droits attribués à ce groupe.

Droits attribués à un utilisateur particulier de l'annuaire LDAP

Un utilisateur s'identifie auprès de l'annuaire LDAP, mais dispose de droits particuliers.

- créez son login dans la gestion des droits ;
- rajoutez-le dans le groupe d'utilisateurs adéquat.

Droits spécifiques de l'application COLLEC

Droits à positionner

Voici les groupes nécessaires pour faire fonctionner correctement l'application :

| Droit | Usage |
|---------|---|
| admin | Gestion des utilisateurs et des droits |
| param | Définition des tables de paramètres généraux, gestion d'un projet |
| projet | rajout des types d'échantillons ou de conteneurs, import de masse |
| gestion | Ajout d'un échantillon pour les projets autorisés, entrée/sortie. Droit attribué par défaut si l'utilisateur fait partie d'au moins un projet |
| consult | Consultation des informations, sans possibilité de modification. Le droit de consultation doit être indiqué volontairement |

TABLE 3.1: Liste des droits utilisés

Ces droits doivent être définis pour chaque application (couple *logiciel* ↔ *base de données*) gérée par la base de gestion des droits.

Gestion des projets

Les échantillons étant obligatoirement rattachés à un projet, vous devrez en créer au minimum un à partir du menu d'administration. Un utilisateur avec les droits de gestion ne peut modifier que les échantillons pour lesquels il est autorisé (les projets qui sont rattachés au groupe dont il fait partie).

Voici le principe de gestion des droits pour les projets :

- Dans *Administration > ACL - Groupes de logins*, déclarez les groupes adéquats. En cas d'utilisation des groupes LDAP, les saisir avec la même casse que dans l'annuaire (EABX p. e.). Il est possible de définir une hiérarchie des groupes, quelle que soit l'origine de l'affectation (base de données ou annuaire Ldap). Dans le cas où l'annuaire Ldap n'est pas utilisé pour gérer les groupes, renseignez les logins en face des groupes dans le même écran ;
- Dans les projets, sélectionnez les groupes autorisés ;
- les utilisateurs faisant partie des groupes autorisés disposeront des droits de *gestion* pour le projet considéré.

Configurer les paramètres généraux

L'ensemble des paramètres sont accessibles à partir du menu *Paramètres*.

Par défaut, tous les utilisateurs qui disposent du droit de consultation peuvent visualiser les paramètres. La modification n'est possible que pour ceux qui disposent des droits suivants :

| Nom | Description | Droit nécessaire |
|------------------------|---|------------------|
| Projets | Liste des projets et droits associés | admin |
| Protocoles | Protocoles de prélèvement des échantillons | projet |
| Opérations | Opérations rattachées aux protocoles | projet |
| Type d'événement | Événements survenant aux objets | param |
| Familles de conteneurs | Mécanisme pour retrouver les conteneurs selon leur nature (pièce, caisse...) | param, projet |
| Conditions de stockage | Mécanisme de conservation (lyophilisation, p. e.) | param, projet |
| Motifs de déstockage | Raisons invoquées pour sortir un objet du stock | param, projet |
| Types de conteneurs | Modèles de conteneurs (porteurs des étiquettes, entre autres) | param, projet |
| Statut des objets | Liste des statuts que peut prendre un objet | param |
| Type d'échantillon | Modèles des échantillons (rattachables à un type de conteneur) | param, projet |
| Sous-échantillonnage | Pour les échantillons composés d'éléments indifférenciables, unité utilisée pour réaliser le sous-échantillonnage (nombre, volume...) | param |
| Étiquettes | Modèles des étiquettes imprimables | param |
| Types d'identifiants | Types d'identifiants complémentaires des objets | param |

TABLE 3.2: Liste des paramètres et droits de modification associés

Créer ou modifier un modèle d'étiquettes

Les étiquettes sont créées en recourant au logiciel FOP [2], écrit en Java. Voici les opérations réalisées par l'application pour générer les étiquettes :

- pour chaque objet concerné (des containers ou des échantillons associés à un type de container, et si le type de container est rattaché à un modèle d'étiquettes), une image du QRcode est générée dans le dossier *temp* ;
- dans le dossier *temp*, un fichier au format XML est généré, contenant les informations à imprimer sur l'étiquette ;
- un fichier au format XSL, qui contient les ordres de création de l'étiquette, est également créé dans le même dossier. Le contenu de ce fichier est issu d'un enregistrement provenant de la table *label* ;
- le programme PHP fait appel à FOP pour générer, à partir du fichier XML et en utilisant le fichier XSL, un fichier PDF. Une page du fichier correspond à une étiquette (mécanisme utilisé par les imprimantes à étiquettes pour les séparer).

La configuration du modèle d'étiquettes revient à définir à la fois le contenu des informations qui seront insérées dans le QRCODE et la forme que prendra l'étiquette, c'est à dire les informations qui seront imprimées, le format, etc. Cette forme reprend la syntaxe XSL comprise par FOP.

Nom de l'étiquette* :

Bocaux transect

Transformation XSL* :

```
<?xml version="1.0" encoding="utf-8"?>
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:fo="http://www.w3.org/1999/XSL/Format">
<xsl:output method="xml" indent="yes"/>
<xsl:template match="objects">
  <fo:root>
    <fo:layout-master-set>
      <fo:simple-page-master master-name="label"
        page-height="5cm" page-width="10cm" margin-left="0.5cm" margin-top="0.5cm" margin-bottom="0cm" margin-
        right="0.5cm">
        <fo:region-body/>
      </fo:simple-page-master>
    </fo:layout-master-set>

    <fo:page-sequence master-reference="label">
      <fo:flow flow-name="xsl-region-body">
        <fo:block>
          <xsl:apply-templates select="object" />
        </fo:block>
      </fo:flow>
    </fo:page-sequence>
  </fo:root>
</xsl:template>
</xsl:stylesheet>
```

Champs à insérer dans le QRcode (séparés par une virgule, sans espace)* :

uid,id,clp,db,tag,cab,prj

Champs utilisables dans le QRcode et dans le texte de l'étiquette :

- uid (obligatoire) : identifiant unique
- db (obligatoire) : code de la base de données (utilisé pour éviter de mélanger les échantillons entre plusieurs bases)
- id : identifiant général
- prj : code du projet
- clp : code de risque
- pn : nom du protocole
- et tous les codes d'identifiants secondaires - cf. paramètres > Types d'identifiants

FIGURE 3.8 – Écran de saisie d'un modèle d'étiquette

Définir le contenu du QRcode

Le QRcode est un format de code barre normalisé en deux dimensions, qui permet de stocker jusqu'à 2000 caractères en 8 bits.

Le principe retenu dans l'application est de stocker l'information au format JSON. Pour limiter la taille du code barre, les noms des balises doivent être les plus petites possibles. Voici les balises obligatoires à insérer systématiquement dans une étiquette :

| Nom | Description |
|-----|--|
| uid | Identifiant unique de l'objet dans la base de données |
| db | Identifiant de la base de données. C'est la valeur du paramètre <i>AP-PLI_code</i> (cf. 2.3.4 Paramètres spécifiques, page 10) |

TABLE 3.3: Liste des balises à insérer obligatoirement dans les QRcodes

D'autres informations peuvent être également insérées :

| Nom | Description |
|--------------|--|
| id | Identifiant métier principal (champ <i>identifiant ou nom</i> , en saisie) |
| prj | Code du projet (pour les échantillons) |
| clp | Code du risque associé au container, en raison du produit de conservation utilisé |
| pn | Nom du protocole de collecte des échantillons |
| autres codes | tous les codes d'identification secondaires définis dans la table de paramètres <i>Types d'identifiants</i> (cf. 3.3 Configurer les paramètres généraux, page 20). |

TABLE 3.4: Liste des balises facultatives insérables dans les QRcodes

Configuration du fichier XSL

La syntaxe particulière du fichier XSL ne doit être modifiée qu'en conservant la version initiale (recopie dans un bloc-notes, par exemple), pour éviter de perdre une configuration opérationnelle suite à un mauvais paramétrage.

Voici la description du contenu du fichier et les zones modifiables.

Entête du fichier

Elle permet de modifier la taille de l'étiquette (largeur et hauteur maximale). Vous ne devriez changer que les attributs *page-height* et *page-width*. Pour les marges (attributs *margin-*), soyez prudents et vérifiez notamment que les QRcodes ne soient pas rognés à cause de marges insuffisantes.

```
<?xml version="1.0" encoding="utf-8"?>
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:fo="http://www.w3.org/1999/XSL/Format">
  <xsl:output method="xml" indent="yes"/>
  <xsl:template match="objects">
    <fo:root>
      <fo:layout-master-set>
        <fo:simple-page-master master-name="label"
          page-height="5cm" page-width="10cm"
          margin-left="0.5cm"
          margin-top="0.5cm"
          margin-bottom="0cm"
          margin-right="0.5cm">
          <fo:region-body/>
        </fo:simple-page-master>
      </fo:layout-master-set>

      <fo:page-sequence master-reference="label">
        <fo:flow flow-name="xsl-region-body">
          <fo:block>
            <xsl:apply-templates select="object" />
          </fo:block>
        </fo:flow>
      </fo:page-sequence>
    </fo:root>
  </xsl:template>
</xsl:stylesheet>
```

```

    </fo : page-sequence >
  </fo : root >
</xsl : template >
<xsl : template match=" object ">

```

Format de l'étiquette

Le contenu de l'étiquette est décrit sous la forme d'un tableau (balises *fo :table*). La première colonne contient le QRCode, la seconde le texte associé.

Ici, deux colonnes de taille identique (4 cm chacune) sont définies.

```

<fo : table table-layout="fixed" border-collapse="collapse"
border-style="none" width="8cm"
keep-together . within-page="always">
  <fo : table-column column-width="4cm"/>
  <fo : table-column column-width="4cm" />
<fo : table-body border-style="none" >

```

Les cellules (*table-cell*) sont insérées dans une ligne (*table-row*) :

```

  <fo : table-row>

```

Insertion du QRcode

Le QRcode est inséré dans un bloc. Les seules informations modifiables sont celles concernant la hauteur (attribut *height* et la largeur (attribut *content-width*)). Veillez à ce que la hauteur et la largeur soient identiques, et ne modifiez pas les autres informations.

```

    <fo : table-cell >
      <fo : block >
        <fo : external-graphic >
<xsl : attribute name="src">
  <xsl : value-of select="concat (uid , '.png') " />
</xsl : attribute >
<xsl : attribute name="content-height">
  scale-to-fit
</xsl : attribute >
<xsl : attribute name="height">4cm</xsl : attribute >
  <xsl : attribute name="content-width">4cm</xsl : attribute >
  <xsl : attribute name="scaling">uniform</xsl : attribute >
</fo : external-graphic >
      </fo : block >
    </fo : table-cell >

```

Contenu textuel

Les autres informations sont affichées dans des blocs, avec une ligne par catégorie d'information. L'étiquette commence ici par indiquer l'établissement (ici, IRSTEA), écrit en gras.

```

  <fo : table-cell >
    <fo : block >
      <fo : inline font-weight="bold">
        IRSTEA
      </fo : inline >
    </fo : block >

```

Chaque information est affichée dans un bloc, comprenant un titre (par exemple, *uid*), associé à une ou plusieurs valeurs. Ainsi, la première ligne affiche sur la même ligne, et en gras (attribut *font-weight="bold"*), le code de la base de données (`<xsl:value-of select="db"/>`) et l'UID de l'objet (`<xsl:value-of select="uid"/>`).

```

<fo:block>uid:
<fo:inline font-weight="bold">
<xsl:value-of select="db"/>:
<xsl:value-of select="uid"/></fo:inline>
</fo:block>
<fo:block>id:
<fo:inline font-weight="bold">
<xsl:value-of select="id"/></fo:inline>
</fo:block>
<fo:block>prj:
<fo:inline font-weight="bold">
<xsl:value-of select="prj"/></fo:inline>
</fo:block>
<fo:block>clp:
<fo:inline font-weight="bold">
<xsl:value-of select="clp"/></fo:inline>
</fo:block>

```

Fin de l'étiquette

Une fois toutes les informations affichées, le tableau est fermé, et un saut de page est généré systématiquement :

```

</fo:table-cell>
</fo:table-row>
</fo:table-body>
</fo:table>
<fo:block page-break-after="always"/>

```

Enfin, le fichier XSL est correctement fermé :

```

</xsl:template>
</xsl:stylesheet>

```

Il est possible de créer des étiquettes avec des formats différents, par exemple en créant plusieurs lignes. Pensez à fermer vos balises, et qu'elles soient correctement imbriquées, pour éviter tout souci.

Pour aller plus loin dans la mise en page, consultez la documentation du projet FOP.

Gestion des traces

Tous les appels lancés par les utilisateurs vers les modules de l'application sont enregistrés dans la table *gacl.log*, qui ne doit être accessible qu'aux personnes dûment autorisées. Les traces sont supprimées au bout d'un an (script de nettoyage exécuté lors de la connexion d'un utilisateur).

Voici un exemple de trace générée :

| log_id | login | nom_module | log_date | commentaire | ipaddress |
|--------|--------------|------------------|----------|-------------|-----------|
| 523437 | eric.quinton | col-Sample-write | | 2016-10-25 | 14:57:01 |
| 16 | ::1 | | | | |

```

523438  eric . quinton      col-sampleDisplay      2016-10-25 14:57:01
        ok             ::1
523436  eric . quinton      col-sampleWrite 2016-10-25 14:57:00      ok
        ::1
523435  eric . quinton      col-sampleChange      2016-10-25 14:56:58
        ok             ::1
523434  eric . quinton      col-sampleDisplay      2016-10-25 14:56:55
        ok             ::1
523433  eric . quinton      col-sampleList 2016-10-25 14:56:52      ok
        ::1
523431  eric . quinton      col-default      2016-10-25 14:53:05      ok
        ::1
523430  eric . quinton      col-connexion 2016-10-25 14:53:04      ldap-ok
        ::1
523429  unknown col-connexion 2016-10-25 14:52:57      ok             ::1

```

La colonne *commentaire*, pour la ligne 523437, contient l'identifiant modifié (l'enregistrement 16 a été traité par le module `sampleWrite` – `Sample` (majuscule du S) correspond au nom de la classe qui a été utilisée pour réaliser l'écriture vers la base de données). L'adresse IP est théoriquement celle de l'utilisateur (ici, connexion locale), y compris en prenant en compte le passage par un serveur Reverse-proxy².

Parallèlement, les messages d'erreur sont envoyés au processus Linux SYSLOG, qui enregistre les traces dans le fichier `/var/log/apache2/error.log`.

2. serveur mis en entrée du réseau privé, qui permet de masquer les adresses internes et de contrôler les accès depuis Internet

Chapitre 4

Comment faire pour ?

Générer une liste d'échantillons vides

Objectif : préparer des bocaux d'échantillons avant de partir en campagne de collecte. Ces bocaux doivent être étiquetés.

Le logiciel propose une procédure d'import de masse, qui permet de répondre à cette question.

Voici la méthode à utiliser :

- générez un fichier au format CSV (créé par exemple à partir de LibreOffice OpenDataSheet – ODS), qui comprend une ligne par échantillon ;
- lancez la procédure d'import : le programme vous indiquera les UID générés ;
- recherchez les UID générés, et déclenchez l'impression des étiquettes.

Structure du fichier CSV

Toute opération d'import présente des risques : il est difficile de revenir en arrière une fois celle-ci terminée. Pour les limiter, le logiciel va procéder en deux étapes. D'abord, la structure du fichier va être analysée, et la cohérence des informations indiquées vérifiée. Ensuite, si aucune anomalie n'est détectée, l'import pourra être déclenché.

La première ligne du fichier doit comporter le nom des colonnes. Leur nom est normalisé et ne doit en aucun cas être modifié. Si une colonne n'existe pas, l'import du fichier sera rejeté.

Les identifiants numériques (*project_id* par exemple) doivent être recherchés dans les tables de paramètres de l'application.

Voici la liste des colonnes utilisables :

| Colonne | Description | Obligatoire |
|-----------------------|--|-------------|
| sample_identifieur | identifiant de l'échantillon | X |
| project_id | identifiant du projet de rattachement | X |
| sample_type_id | identifiant du type d'échantillon | X |
| sample_status_id | identifiant du statut à attribuer | X |
| sample_date | date de création de l'échantillon, au format dd/mm/yyyy | |
| sample_range | emplacement de rangement de l'échantillon dans le conteneur, si le numéro de conteneur est indiqué dans le fichier | |
| sample_multiple_value | nombre total (ou volume total...) de sous-échantillons associés à l'échantillon, si celui-ci est sous-échantillonnable | |
| container_identifieur | identifiant du container | X |
| container_type_id | identifiant du type de container | X |

| Colonne | Description | Obligatoire |
|------------------------------|---|-------------|
| container_status_id | identifiant du statut à attribuer au container | |
| container_range | emplacement de rangement du container dans son container parent | |
| container_parent_uid | UID du container dans lequel le container courant est rangé | |
| identifiants complémentaires | une colonne par code supplémentaire (menu <i>Paramètres</i> → <i>Types d'identifiants</i>) | |

TABLE 4.1: Liste des colonnes utilisables lors d'un import de masse

Les champs obligatoires ne le sont que si l'identifiant de l'objet considéré – échantillon ou container – a été renseigné. Une ligne doit contenir au minimum soit un numéro d'échantillon, soit un numéro de container.

Procédure d'import

À partir du menu, choisissez *Objet* → *import de masse*. Seuls les utilisateurs qui disposent du droit *projet* pourront réaliser l'opération.

The screenshot shows a light blue interface for mass import. At the top, it says 'Nom du fichier à importer (CSV)* :'. To the right, there is a button 'Choisissez un fichier' and the text 'Aucun fichier choisi'. Below this, there are two dropdown menus: 'Séparateur utilisé :' with 'Virgule' selected, and 'Encodage du fichier :' with 'UTF-8' selected. At the bottom center, there is a dark blue button labeled 'Lancer les contrôles'.

FIGURE 4.1 – Sélection du fichier pour un import de masse

Sélectionnez le fichier à importer, vérifiez le séparateur utilisé. Préférez, si possible, les données au format UTF-8.

L'import sera réalisé ainsi :

1. si *sample_identifieur* est renseigné : création de l'échantillon
2. si *container_identifieur* est renseigné : création du container
3. si *container_parent_uid* est renseigné : création du mouvement d'entrée du container
4. si l'échantillon et le container ont été créés, création du mouvement d'entrée de l'échantillon dans le container

Si des anomalies sont détectées lors du contrôle, un tableau récapitulant les problèmes rencontrés sera affiché, ressemblant à ceci :

| N° de ligne | Anomalie(s) détectée(s) |
|-------------|--|
| 11 | Le numéro du projet indiqué n'est pas reconnu ou autorisé. Le type de container n'est pas connu. |
| 12 | Le statut du container n'est pas connu. L'UID du conteneur parent n'existe pas. |
| 13 | Aucun échantillon ou container n'est décrit (pas d'identifiant pour l'un ou pour l'autre). |

FIGURE 4.2 – Exemples d’anomalies détectées lors du contrôle de l’import

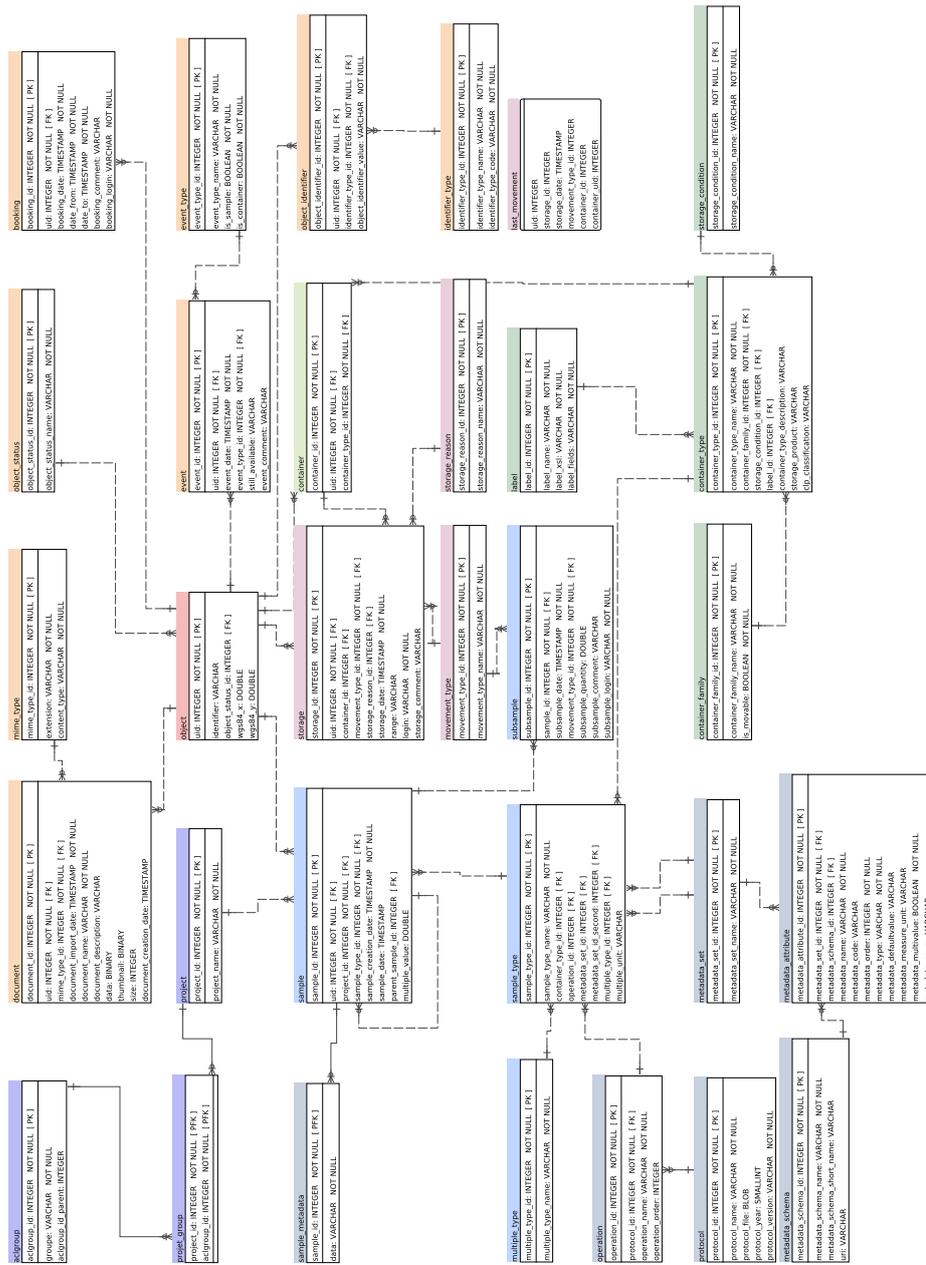
Si les contrôles se sont bien déroulés, le programme proposera alors de déclencher l’import, et affichera en retour les valeurs *mini* et *maxi* des UID générées.

Autre usage

Cette fonctionnalité peut également être utilisée pour déclencher l’import de listes d’échantillons pré-existants, et de créer automatiquement les mouvements adéquats pour les ranger dans leurs containers de stockage.

Annexe A

Structure de la base de données



eabxcol

List of tables

- [aclgroup](#)
- [booking](#)
- [container](#)
- [container_family](#)
- [container_type](#)
- [document](#)
- [event](#)
- [event_type](#)
- [identifier_type](#)
- [label](#)
- [last_movement](#)
- [metadata_attribute](#)
- [metadata_schema](#)
- [metadata_set](#)
- [mime_type](#)
- [movement_type](#)
- [multiple_type](#)
- [object](#)
- [object_identifier](#)
- [object_status](#)
- [operation](#)
- [project](#)
- [projet_group](#)
- [protocol](#)
- [sample](#)
- [sample_metadata](#)
- [sample_type](#)
- [storage](#)
- [storage_condition](#)
- [storage_reason](#)
- [subsample](#)

aclgroup (Physical Name: aclgroup)

Groupes des logins

| Logical Column Name | Physical Column Name | Type | PK | Nullable |
|---------------------|----------------------|------------|----|----------|
| aclgroup_id (PK) | aclgroup_id | INTEGER | PK | NOT NULL |
| groupe | groupe | VARCHAR(0) | | NOT NULL |
| aclgroup_id_parent | aclgroup_id_parent | INTEGER | | |

Referenced By

- [projet_group](#) referencing (aclgroup_id)

booking (Physical Name: booking)

Table des réservations d'objets

| Logical Column Name | Physical Column Name | Type | PK | Nullable |
|---------------------|---------------------------------------|------------|----|----------|
| booking_id (PK) | booking_id | INTEGER | PK | NOT NULL |
| uid (FK) | uid | INTEGER | | NOT NULL |
| booking_date | booking_date | TIMESTAMP | | NOT NULL |
| | Date de la réservation | | | |
| date_from | date_from | TIMESTAMP | | NOT NULL |
| | Date-heure de début de la réservation | | | |
| date_to | date_to | TIMESTAMP | | NOT NULL |
| | Date-heure de fin de la réservation | | | |
| booking_comment | booking_comment | VARCHAR(0) | | |
| | Commentaire | | | |
| booking_login | booking_login | VARCHAR(0) | | NOT NULL |
| | Compte ayant réalisé la réservation | | | |

References

- [object](#) through (uid)

container (Physical Name: container)

Liste des conteneurs d'échantillon

| Logical Column Name | Physical Column Name | Type | PK | Nullable |
|------------------------|----------------------|---------|----|----------|
| container_id (PK) | container_id | INTEGER | PK | NOT NULL |
| uid (FK) | uid | INTEGER | | NOT NULL |
| container_type_id (FK) | container_type_id | INTEGER | | NOT NULL |

References

- [object](#) through (uid)
- [container_type](#) through (container_type_id)

Referenced By

- [storage](#) referencing (container_id)

container_family (Physical Name: container_family)

Famille générique des conteneurs

| Logical Column Name | Physical Column Name | Type | PK | Nullable |
|--------------------------|-----------------------|------------|----|----------|
| container_family_id (PK) | container_family_id | INTEGER | PK | NOT NULL |
| container_family_name | container_family_name | VARCHAR(0) | | NOT NULL |
| is_movable | is_movable | BOOLEAN | | NOT NULL |

Indique si la famille de conteneurs est déplçable facilement ou non
(éprouvette : oui, armoire : non)

Referenced By

- [container_type](#) referencing (container_family_id)

container_type (Physical Name: container_type)

Table des types de conteneurs

| Logical Column Name | Physical Column Name | Type | PK | Nullable |
|---|----------------------------|------------|----|----------|
| container_type_id (PK) | container_type_id | INTEGER | PK | NOT NULL |
| container_type_name | container_type_name | VARCHAR(0) | | NOT NULL |
| container_family_id (FK) | container_family_id | INTEGER | | NOT NULL |
| storage_condition_id (FK) | storage_condition_id | INTEGER | | |
| label_id (FK) | label_id | INTEGER | | |
| container_type_description | container_type_description | VARCHAR(0) | | |
| Description longue | | | | |
| storage_product | storage_product | VARCHAR(0) | | |
| Produit utilisé pour le stockage (formol, alcool...) | | | | |
| clp_classification | clp_classification | VARCHAR(0) | | |
| Classification du risque conformément à la directive européenne CLP | | | | |

References

- [container_family](#) through (container_family_id)
- [storage_condition](#) through (storage_condition_id)
- [label](#) through (label_id)

Referenced By

- [container](#) referencing (container_type_id)
- [sample_type](#) referencing (container_type_id)

document (Physical Name: document)

Documents numériques rattachés à un poisson ou à un événement

| Logical Column Name | Physical Column Name | Type | PK | Nullable |
|---|------------------------|------------|----|----------|
| document_id (PK) | document_id | INTEGER | PK | NOT NULL |
| uid (FK) | uid | INTEGER | | NOT NULL |
| mime_type_id (FK) | mime_type_id | INTEGER | | NOT NULL |
| document_import_date | document_import_date | TIMESTAMP | | NOT NULL |
| Date d'import dans la base de données | | | | |
| document_name | document_name | VARCHAR(0) | | NOT NULL |
| Nom d'origine du document | | | | |
| document_description | document_description | VARCHAR(0) | | |
| Description libre du document | | | | |
| data | data | [-2] | | |
| Contenu du document | | | | |
| thumbnail | thumbnail | [-2] | | |
| Vignette au format PNG (documents pdf, jpg ou png) | | | | |
| size | size | INTEGER | | |
| Taille du fichier téléchargé | | | | |
| document_creation_date | document_creation_date | TIMESTAMP | | |
| Date de création du document (date de prise de vue de la photo) | | | | |

References

- [object](#) through (uid)
- [mime_type](#) through (mime_type_id)

event (Physical Name: event)

Table des événements

| Logical Column Name | Physical Column Name | Type | PK | Nullable |
|---------------------|----------------------|-----------|----|----------|
| event_id (PK) | event_id | INTEGER | PK | NOT NULL |
| uid (FK) | uid | INTEGER | | NOT NULL |
| event_date | event_date | TIMESTAMP | | NOT NULL |

Date / heure de l'événement

| | | | |
|---|-----------------|------------|----------|
| event_type_id (FK) | event_type_id | INTEGER | NOT NULL |
| still_available | still_available | VARCHAR(0) | |
| définit ce qu'il reste de disponible dans l'objet | | | |
| event_comment | event_comment | VARCHAR(0) | |

References

- [object](#) through (uid)
- [event_type](#) through (event_type_id)

event_type (Physical Name: event_type)

Types d'événement

| Logical Column Name | Physical Column Name | Type | PK | Nullable |
|---|----------------------|------------|----|----------|
| event_type_id (PK) | event_type_id | INTEGER | PK | NOT NULL |
| event_type_name | event_type_name | VARCHAR(0) | | NOT NULL |
| is_sample | is_sample | BOOLEAN | | NOT NULL |
| L'événement s'applique aux échantillons | | | | |
| is_container | is_container | BOOLEAN | | NOT NULL |
| L'événement s'applique aux conteneurs | | | | |

Referenced By

- [event](#) referencing (event_type_id)

identifiant_type (Physical Name: identifiant_type)

Table des types d'identifiants

| Logical Column Name | Physical Column Name | Type | PK | Nullable |
|--|-----------------------|------------|----|----------|
| identifiant_type_id (PK) | identifiant_type_id | INTEGER | PK | NOT NULL |
| identifiant_type_name | identifiant_type_name | VARCHAR(0) | | NOT NULL |
| Nom textuel de l'identifiant | | | | |
| identifiant_type_code | identifiant_type_code | VARCHAR(0) | | NOT NULL |
| Code utilisé pour la génération des étiquettes | | | | |

Referenced By

- [object_identifier](#) referencing (identifier_type_id)

label (Physical Name: label)

Table des modèles d'étiquettes

| Logical Column Name | Physical Column Name | Type | PK | Nullable |
|---------------------|--|------------|----|----------|
| label_id (PK) | label_id | INTEGER | PK | NOT NULL |
| label_name | label_name | VARCHAR(0) | | NOT NULL |
| | Nom du modèle | | | |
| label_xsl | label_xsl | VARCHAR(0) | | NOT NULL |
| | Contenu du fichier XSL utilisé pour la transformation FOP (https://xmlgraphics.apache.org/fop/) | | | |
| label_fields | label_fields | VARCHAR(0) | | NOT NULL |
| | Liste des champs à intégrer dans le QRCODE, séparés par une virgule | | | |

Referenced By

- [container_type](#) referencing (label_id)

last_movement (Physical Name: last_movement)

| Logical Column Name | Physical Column Name | Type | PK | Nullable |
|---------------------|----------------------|-----------|----|----------|
| uid | uid | INTEGER | | |
| storage_id | storage_id | INTEGER | | |
| storage_date | storage_date | TIMESTAMP | | |
| movement_type_id | movement_type_id | INTEGER | | |
| container_id | container_id | INTEGER | | |
| container_uid | container_uid | INTEGER | | |

metadata_attribute (Physical Name: metadata_attribute)

Table des attributs rattachés à un jeu de métadonnées

| Logical Column Name | Physical Column Name | Type | PK | Nullable |
|---|-----------------------|------------|----|----------|
| metadata_attribute_id (PK) | metadata_attribute_id | INTEGER | PK | NOT NULL |
| metadata_set_id (FK) | metadata_set_id | INTEGER | | NOT NULL |
| metadata_schema_id (FK) | metadata_schema_id | INTEGER | | |
| metadata_name | metadata_name | VARCHAR(0) | | NOT NULL |
| Nom de la métadonnée (creator, name...) | | | | |
| metadata_code | metadata_code | VARCHAR(0) | | |
| Code normalisé de la métadonnée (ex : dcterms:creator) | | | | |
| metadata_order | metadata_order | INTEGER | | NOT NULL |
| Ordre d'affichage des informations dans la grille de saisie | | | | |
| metadata_type | metadata_type | VARCHAR(0) | | NOT NULL |
| metadata_defaultvalue | metadata_defaultvalue | VARCHAR(0) | | |
| metadata_measure_unit | metadata_measure_unit | VARCHAR(0) | | |
| Unité de mesure utilisée | | | | |
| metadata_multivalue | metadata_multivalue | BOOLEAN | | NOT NULL |
| metadata_enum | metadata_enum | VARCHAR(0) | | |
| Liste des valeurs possibles, séparées par ; | | | | |

References

- [metadata_set](#) through (metadata_set_id)
- [metadata_schema](#) through (metadata_schema_id)

metadata_schema (Physical Name: metadata_schema)

Liste des schémas de métadonnées utilisés

| Logical Column Name | Physical Column Name | Type | PK | Nullable |
|--|----------------------------|------------|----|----------|
| metadata_schema_id (PK) | metadata_schema_id | INTEGER | PK | NOT NULL |
| metadata_schema_name | metadata_schema_name | VARCHAR(0) | | NOT NULL |
| Nom complet du schéma | | | | |
| metadata_schema_short_name | metadata_schema_short_name | VARCHAR(0) | | |
| abréviation habituelle (CC, DC...) | | | | |
| uri | uri | VARCHAR(0) | | |
| Adresse URI d'accès à la description du schéma | | | | |

Referenced By

- [metadata_attribute](#) referencing (metadata_schema_id)

metadata_set (Physical Name: metadata_set)

Jeu de métadonnées permettant de décrire précisément un échantillon

| Logical Column Name | Physical Column Name | Type | PK | Nullable |
|----------------------|----------------------|------------|----|----------|
| metadata_set_id (PK) | metadata_set_id | INTEGER | PK | NOT NULL |
| metadata_set_name | metadata_set_name | VARCHAR(0) | | NOT NULL |

Referenced By

- [sample_type](#) referencing (metadata_set_id)
- [metadata_attribute](#) referencing (metadata_set_id)
- [sample_type](#) referencing (metadata_set_id)

mime_type (Physical Name: mime_type)

Types mime des fichiers importés

| Logical Column Name | Physical Column Name | Type | PK | Nullable |
|------------------------------------|----------------------|------------|----|----------|
| mime_type_id (PK) | mime_type_id | INTEGER | PK | NOT NULL |
| extension | extension | VARCHAR(0) | | NOT NULL |
| Extension du fichier correspondant | | | | |
| content_type | content_type | VARCHAR(0) | | NOT NULL |
| type mime officiel | | | | |

Referenced By

- [document](#) referencing (mime_type_id)

movement_type (Physical Name: movement_type)

Type de mouvement

| Logical Column Name | Physical Column Name | Type | PK | Nullable |
|-----------------------|----------------------|------------|----|----------|
| movement_type_id (PK) | movement_type_id | INTEGER | PK | NOT NULL |
| movement_type_name | movement_type_name | VARCHAR(0) | | NOT NULL |

Referenced By

- [storage](#) referencing (movement_type_id)
- [subsample](#) referencing (movement_type_id)

multiple_type (Physical Name: multiple_type)

Table des types de contenus multiples

| Logical Column Name | Physical Column Name | Type | PK | Nullable |
|-----------------------|----------------------|------------|----|----------|
| multiple_type_id (PK) | multiple_type_id | INTEGER | PK | NOT NULL |
| multiple_type_name | multiple_type_name | VARCHAR(0) | | NOT NULL |

Referenced By

- [sample_type](#) referencing (multiple_type_id)

object (Physical Name: object)

Table des objets

Contient les identifiants génériques

| Logical Column Name | Physical Column Name | Type | PK | Nullable |
|---------------------|----------------------|------------|----|----------|
| uid (PK) | uid | INTEGER | PK | NOT NULL |
| identifiant | identifiant | VARCHAR(0) | | |

Identifiant fourni le cas échéant par le projet

| | | | | |
|-----------------------|------------------|---------|--|--|
| object_status_id (FK) | object_status_id | INTEGER | | |
| wgs84_x | wgs84_x | DOUBLE | | |

Longitude GPS, en valeur décimale

| | | | | |
|---------|---------|--------|--|--|
| wgs84_y | wgs84_y | DOUBLE | | |
|---------|---------|--------|--|--|

Latitude GPS, en décimal

References

- [object_status](#) through (object_status_id)

Referenced By

- [sample](#) referencing (uid)
- [container](#) referencing (uid)
- [event](#) referencing (uid)
- [storage](#) referencing (uid)
- [booking](#) referencing (uid)
- [document](#) referencing (uid)
- [object_identifier](#) referencing (uid)

object_identifier (Physical Name: object_identifier)

Table des identifiants complémentaires normalisés

| Logical Column Name | Physical Column Name | Type | PK | Nullable |
|---------------------------|-------------------------|------------|----|----------|
| object_identifier_id (PK) | object_identifier_id | INTEGER | PK | NOT NULL |
| uid (FK) | uid | INTEGER | | NOT NULL |
| identifieur_type_id (FK) | identifieur_type_id | INTEGER | | NOT NULL |
| object_identifier_value | object_identifier_value | VARCHAR(0) | | NOT NULL |

Valeur de l'identifiant

References

- [object](#) through (uid)
- [identifieur_type](#) through (identifieur_type_id)

object_status (Physical Name: object_status)

Table des statuts possibles des objets

| Logical Column Name | Physical Column Name | Type | PK | Nullable |
|-----------------------|----------------------|------------|----|----------|
| object_status_id (PK) | object_status_id | INTEGER | PK | NOT NULL |
| object_status_name | object_status_name | VARCHAR(0) | | NOT NULL |

Referenced By

- [object](#) referencing (object_status_id)

operation (Physical Name: operation)

| Logical Column Name | Physical Column Name | Type | PK | Nullable |
|---------------------|----------------------|------------|----|----------|
| operation_id (PK) | operation_id | INTEGER | PK | NOT NULL |
| protocol_id (FK) | protocol_id | INTEGER | | NOT NULL |
| operation_name | operation_name | VARCHAR(0) | | NOT NULL |
| operation_order | operation_order | INTEGER | | |

Ordre de réalisation de l'opération dans le protocole

References

- [protocol](#) through (protocol_id)

Referenced By

- [sample_type](#) referencing (operation_id)

project (Physical Name: project)

Table des projets

| Logical Column Name | Physical Column Name | Type | PK | Nullable |
|---------------------|----------------------|------------|----|----------|
| project_id (PK) | project_id | INTEGER | PK | NOT NULL |
| project_name | project_name | VARCHAR(0) | | NOT NULL |

Referenced By

- [sample](#) referencing (project_id)
- [projet_group](#) referencing (project_id)

projet_group (Physical Name: projet_group)

Table des autorisations d'accès à un projet

| Logical Column Name | Physical Column Name | Type | PK | Nullable |
|---------------------|----------------------|------|----|----------|
|---------------------|----------------------|------|----|----------|

| | | | | |
|--|-------------|---------|----|----------|
| <u>project_id</u> (PK) (FK) | project_id | INTEGER | PK | NOT NULL |
| <u>aclgroup_id</u> (PK) (FK) | aclgroup_id | INTEGER | PK | NOT NULL |

References

- [project](#) through (project_id)
- [aclgroup](#) through (aclgroup_id)

protocol (Physical Name: protocol)

| Logical Column Name | Physical Column Name | Type | PK | Nullable |
|-------------------------|------------------------------|------------|----|----------|
| <u>protocol_id</u> (PK) | protocol_id | INTEGER | PK | NOT NULL |
| protocol_name | protocol_name | VARCHAR(0) | | NOT NULL |
| protocol_file | protocol_file | BLOB | | |
| | Description PDF du protocole | | | |
| protocol_year | protocol_year | SMALLINT | | |
| | Année du protocole | | | |
| protocol_version | protocol_version | VARCHAR(0) | | NOT NULL |
| | Version du protocole | | | |

Referenced By

- [operation](#) referencing (protocol_id)

sample (Physical Name: sample)

Table des échantillons

| Logical Column Name | Physical Column Name | Type | PK | Nullable |
|--|--|-----------|----|----------|
| <u>sample_id</u> (PK) | sample_id | INTEGER | PK | NOT NULL |
| <u>uid</u> (FK) | uid | INTEGER | | NOT NULL |
| <u>project_id</u> (FK) | project_id | INTEGER | | NOT NULL |
| <u>sample_type_id</u> (FK) | sample_type_id | INTEGER | | NOT NULL |
| sample_creation_date | sample_creation_date | TIMESTAMP | | NOT NULL |
| | Date de création de l'enregistrement dans la base de données | | | |
| sample_date | sample_date | TIMESTAMP | | |
| | Date de création de l'échantillon physique | | | |
| <u>parent_sample_id</u> (FK) | parent_sample_id | INTEGER | | |
| multiple_value | multiple_value | DOUBLE | | |

Nombre initial de sous-échantillons

References

- [object](#) through (uid)
- [sample](#) through (parent_sample_id)
- [project](#) through (project_id)
- [sample_type](#) through (sample_type_id)

Referenced By

- [sample](#) referencing (sample_id)
- [sample_metadata](#) referencing (sample_id)
- [subsampling](#) referencing (sample_id)

sample_metadata (Physical Name: sample_metadata)

| Logical Column Name | Physical Column Name | Type | PK | Nullable |
|---------------------------------------|----------------------|------------|----|----------|
| sample_id (PK) (FK) | sample_id | INTEGER | PK | NOT NULL |
| data | data | VARCHAR(0) | | NOT NULL |

Champ JSONB pour stockage des données spécifiques de l'échantillon

References

- [sample](#) through (sample_id)

sample_type (Physical Name: sample_type)

Types d'échantillons

| Logical Column Name | Physical Column Name | Type | PK | Nullable |
|---|------------------------|------------|----|----------|
| sample_type_id (PK) | sample_type_id | INTEGER | PK | NOT NULL |
| sample_type_name | sample_type_name | VARCHAR(0) | | NOT NULL |
| container_type_id (FK) | container_type_id | INTEGER | | |
| operation_id (FK) | operation_id | INTEGER | | |
| metadata_set_id (FK) | metadata_set_id | INTEGER | | |
| metadata_set_id_second (FK) | metadata_set_id_second | INTEGER | | |
| Second jeu de métadonnées rattaché au type | | | | |
| multiple_type_id (FK) | multiple_type_id | INTEGER | | |

| | | |
|---|---------------|------------|
| multiple_unit | multiple_unit | VARCHAR(0) |
| Unité caractérisant le sous-échantillon | | |

References

- [container_type](#) through (container_type_id)
- [metadata_set](#) through (metadata_set_id)
- [metadata_set](#) through (metadata_set_id_second)
- [operation](#) through (operation_id)
- [multiple_type](#) through (multiple_type_id)

Referenced By

- [sample](#) referencing (sample_type_id)

storage (Physical Name: storage)

Gestion du stockage des échantillons

| Logical Column Name | Physical Column Name | Type | PK | Nullable |
|--|----------------------|------------|----|----------|
| storage_id (PK) | storage_id | INTEGER | PK | NOT NULL |
| uid (FK) | uid | INTEGER | | NOT NULL |
| container_id (FK) | container_id | INTEGER | | |
| movement_type_id (FK) | movement_type_id | INTEGER | | NOT NULL |
| storage_reason_id (FK) | storage_reason_id | INTEGER | | |
| storage_date | storage_date | TIMESTAMP | | NOT NULL |
| Date/heure du mouvement | | | | |
| range | range | VARCHAR(0) | | |
| Emplacement de l'échantillon dans le conteneur | | | | |
| login | login | VARCHAR(0) | | NOT NULL |
| Nom de l'utilisateur ayant réalisé l'opération | | | | |
| storage_comment | storage_comment | VARCHAR(0) | | |
| Commentaire | | | | |

References

- [container](#) through (container_id)
- [object](#) through (uid)
- [movement_type](#) through (movement_type_id)
- [storage_reason](#) through (storage_reason_id)

storage_condition (Physical Name: storage_condition)

Condition de stockage

| Logical Column Name | Physical Column Name | Type | PK | Nullable |
|---------------------------|------------------------|------------|----|----------|
| storage_condition_id (PK) | storage_condition_id | INTEGER | PK | NOT NULL |
| storage_condition_name | storage_condition_name | VARCHAR(0) | | NOT NULL |

Referenced By

- [container_type](#) referencing (storage_condition_id)

storage_reason (Physical Name: storage_reason)

Table des raisons de stockage/déstockage

| Logical Column Name | Physical Column Name | Type | PK | Nullable |
|------------------------|----------------------|------------|----|----------|
| storage_reason_id (PK) | storage_reason_id | INTEGER | PK | NOT NULL |
| storage_reason_name | storage_reason_name | VARCHAR(0) | | NOT NULL |

Referenced By

- [storage](#) referencing (storage_reason_id)

subsample (Physical Name: subsample)

Table des prélèvements et restitutions de sous-échantillons

| Logical Column Name | Physical Column Name | Type | PK | Nullable |
|---------------------------|----------------------|-----------|----|----------|
| subsample_id (PK) | subsample_id | INTEGER | PK | NOT NULL |
| sample_id (FK) | sample_id | INTEGER | | NOT NULL |
| subsample_date | subsample_date | TIMESTAMP | | NOT NULL |
| Date/heure de l'opération | | | | |
| movement_type_id (FK) | movement_type_id | INTEGER | | NOT NULL |
| subsample_quantity | subsample_quantity | DOUBLE | | |

Quantité prélevée ou restituée

| | | | |
|-------------------|-------------------|------------|----------|
| subsample_comment | subsample_comment | VARCHAR(0) | |
| subsample_login | subsample_login | VARCHAR(0) | NOT NULL |

Login de l'utilisateur ayant réalisé l'opération

References

- [sample](#) through (sample_id)
- [movement_type](#) through (movement_type_id)

Bibliographie

- [1] ANSSI. Recommandations de sécurité relatives à tls, 2016. URL http://www.ssi.gouv.fr/uploads/2016/09/guide_tls_v1.1.pdf.
- [2] Apache-FOP. The apache fop project, 2016. URL <http://xmlgraphics.apache.org/fop/>.
- [3] Mike Benoit and Dan Cech. Phpgacl, generic access control lists, 2006. URL <http://phpgacl.sourceforge.net>.
- [4] bootstrap. Bootstrap is the most popular html, css, and js framework for developing responsive, mobile first projects on the web, 2016. URL <http://getbootstrap.com>.
- [5] Free Software Foundation. Gnu affero general public license, 2007. URL <https://www.gnu.org/licenses/agpl.html>.
- [6] JQuery. Site officiel, 2015. URL <http://jquery.com/>.
- [7] OWASP. Application security verification standard (2014), 2014. URL https://www.owasp.org/images/5/58/OWASP_ASVS_Version_2.pdf.
- [8] OWASP. Site institutionnel, 2015. URL <https://www.owasp.org>.
- [9] OWASP. Zed attack proxy project, 2015. URL https://www.owasp.org/index.php/OWASP_Zed_Attack_Proxy_Project.
- [10] pgAdmin. pgadmin postgresql tools, 2016. URL <https://pgadmin.org>.
- [11] Eric Quinton. Documentation d'utilisation du framework prototype php, 2016. URL <https://github.com/equinton/prototypephp/blob/bootstrap/database/documentation/prototypephp-documentation.pdf>.
- [12] Eric Quinton. Prototypephp, 2016. URL <https://github.com/equinton/prototypephp/tree/bootstrap>.
- [13] Eric Quinton. Ré-identification par jeton, 2016. URL <http://www.linux-professionnel.net/programmation/php---codes-divers/re-identification-par-jeton>.
- [14] smarty. Smarty, php template engine, 2016. URL <http://www.smarty.net>.