Logiciel COLLEC

Installation et configuration

11 octobre 2016

Éric Quinton

IRSTEA - Centre de Bordeaux
50, avenue de Verdun, Gazinet
33612 CESTAS Cedex

Table des matières

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 patent.

Dans un premier temps, une rapide recherche à permis de repérer un certain nombre de logiciels susceptibles de répondre au besoin. Toutefois, leurs limites sont vite apparues : problème de pérennité, ancienneté du code, modèle de distribution insatisfaisant (licence OpenSource obligatoire pour le suivi sur le long terme), 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'Organisme 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, en particulier sur la gestion de l'étiquetage et le scan des codes-barres.

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

Les délais de développement de la première version opérationnelle se sont étalés sur environ 9 mois, entre la définition des besoins et le développement proprement dit, mené entre juin et octobre 2016. Environ 160 heures auront été nécessaires pour boucler cette version.

Principes généraux

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 d'un type de container : il y a de nombreux cas où l'échantillon lui-même se confond avec son contenant, par exemple le résultat d'une pêche non triée et stocké dans un bocal.

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

— d'un numéro unique, l'UID, et 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 être 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 un tablette, l'application disposant d'un module capable d'activer la caméra depuis le navigateur et de scanner le code-barre.

Il est ainsi possible d'imprimer des étiquettes tant pour des containers que pour des échantillons, si ceux-ci sont d'un type rattaché à un type de container.

Un échantillon est forcément rattaché à un projet. D'ailleurs, seuls les membres du projet peuvent modifier les informations générales d'un échantillon. Il est également possible de le rattacher à un protocole et à une opération dans le protocole, ce qui permet de suivre son évolution.

Un échantillon peut être subdivisé en d'autres échantillons. Par exemple, un poisson peut se voir extrait des otolithes (os de l'oreille), ce qui permettra de créer 2 nouveaux échantillons d'un autre type, qui pourront toujours être rattachés au poisson parent.

Enfin, dans certains cas de figure, un échantillon peut être composé de plusieurs éléments indifférenciés (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*, développé parallèlement par l'auteur du logiciel.

Il a recours à la classe Smarty pour gérer l'affichage des pages HTML. Celles-ci sont générées en utilisant Jquery et divers composants associés. Le rendu général est réalisé avec Bootstrap.

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

Sécurité

L'application a été conçue pour résister aux attaques dites opportunistes selon la nomenclature ASVS v3 de l'OWASP. Des tests d'attaque ont été réalisés en août 2016 avec le logiciel ZapProxy, 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 ;

— 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.

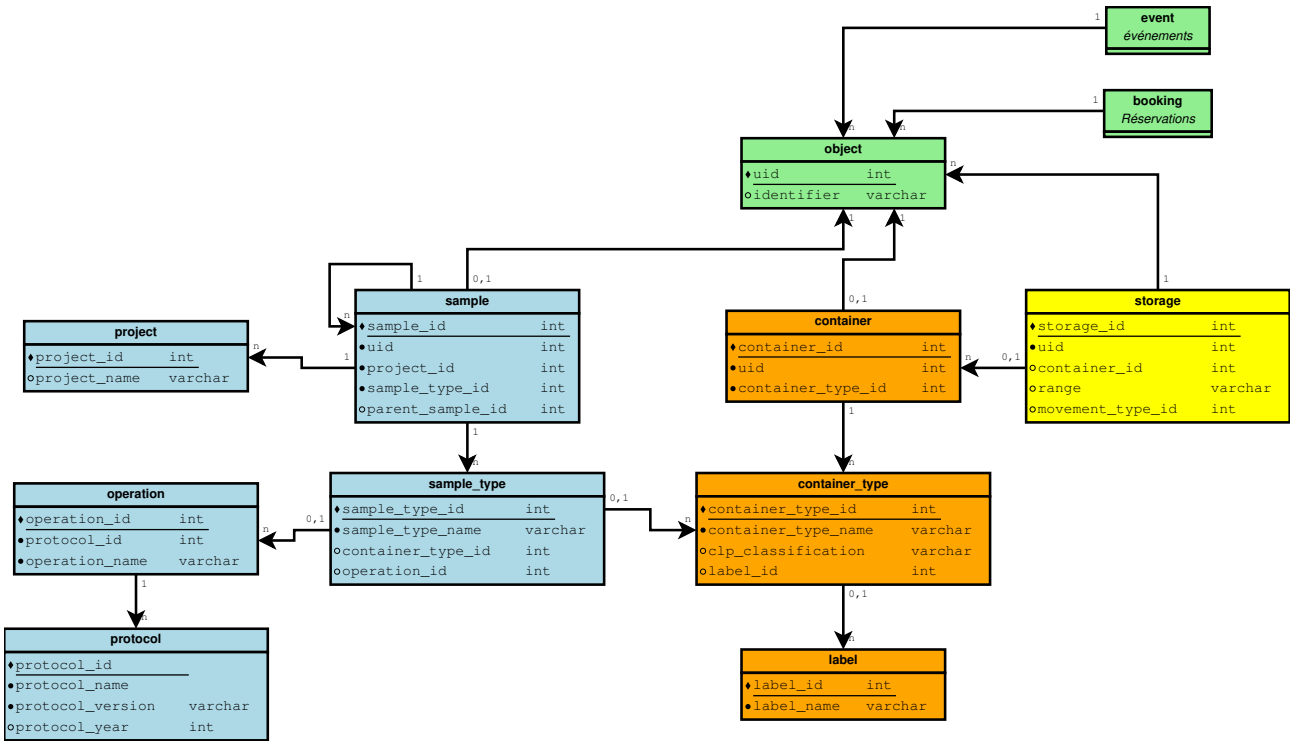

FIGURE 1.1 – Schéma simplifié de la base de données

Chapitre 2

Installer le logiciel

Consultez la documentation du framework !

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

De nombreuses reprises figurent ici, mais il n'est pas inutile de se référer au document d'origine. Seuls les exemples dédiés au logiciel sont présentés, mais d'autres informations utiles pourront y être piochées.

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.

Pré-requis

Configuration de l'hôte virtuel et de SSL

Configuration du dossier d'installation

Mécanisme pour faire cohabiter plusieurs instances avec le même code

Droits à attribuer au serveur web

Première section

Référence bibliographique au livre **?**], à l'article **?**], à la thèse [**?**].
La bibliographie utilise le fichier *irstea.bib* dans cet exemple.

Chapitre 3

Second chapitre

Annexe A

Première annexe

Bibliographie

[1] Quinton Eric. Prototypephp, 2016. URL https://github.com/equinton/prototypephp/tree/bootstrap.

[2] Quinton Eric. Documentation d'utilisation du framework prototype php, 2016. URL https://github.com/equinton/prototypephp/blob/bootstrap/database/documentation/prototypephp-documentation.pdf.