Vers un système d'informations et de gestion du Centre de Ressources Biologiques d'agrumes de San Giuliano

Mickaël Broutta

Centre INRA de Corse, Unité Expérimentale Citrus

Vers un système d'informations et de gestion du Centre de Ressources Biologiques d'agrumes de San Giuliano

Stage de fin d'études, Master Informatique S2I

Broutta Mickaël

Maître de stage : Floriant Guéniot
Responsable pédagogique : Jean-François Santucci

14

Remerciements

Je veux remercier Olivier Pailly, directeur d'unité de la structure GEQA (Génétique et Ecophysiologie de la Qualité des Agrumes), pour m'avoir accueilli dans l'Unité Expérimentale Citrus ;

Emmanuel Bloquel, pour m'avoir aidé à la compréhension des bases de données utilisées ;

Gilles Costantino & François Varamo pour leur soutien dans la conception et le développement du site web et de la base de données ;

Florian Guéniot pour m'avoir accueilli dans son bureau, m'avoir aidé dans le stage tant aux niveaux conception et développement, qu'au niveau de ce rapport ;

Et enfin les autres employés au LRDE de Corte, pour leur accueil chaleureux.

Résumé

Le CRB (Centre de Ressources Biologiques) Citrus est en procédure de certification à la norme AFNOR (NF S 96-900) relative aux CRB végétaux. Pour arriver à l'obtenir, le CRB a besoin d'un outil afin de gérer les données passeport et de caractérisation, les contrôles sanitaires, le processus de production de semences, la diffusion du matériel végétal et la gestion parcellaire.

Donc l'Unité Expérimentale Citrus a décidé de construire cet outil sous forme de site web, qui aidera aussi à terme à promouvoir la collection génétique du CRB Citrus.

Ce rapport traite de mon travail effectué autour de cet outil pendant six mois de stage.

Abstract

The Citrus BRC (Biological Resource Center) is in the process of getting the compliance certification under the french norm AFNOR (NF S 96-900) related to vegetal BRC standards. To achieve this result, the BRC needs a tool able to manage the passport data and features of plants, and track down sanitary controls, the producting trees production process, the broadcast of vegetal materials, and do plot of land management.

Thus, the Experimental Unit Citrus decided to build the tool as a website, which would also help to promote the BRC's genetic ressources collection.

This report deals with my work done around this tool during a six-month internship.

REMERCIEMENTS 1

RESUME / ABSTRACT 2

INTRODUCTION 5

CONTEXTE 6

L'INRA 6
LE CENTRE INRA DE CORSE 6
L'UE CITRUS 7
LE SYSTEME D'INFORMATIONS DE L'UE CITRUS 7

PARTIE I – MATERIEL ET METHODE 11

CHOIX TECHNIQUES 11
GIT 11
GIT-FLOW ERREUR ! SIGNET NON DEFINI.
ARCHITECTURE AUTOUR DE GIT 12
MYSQL WORKBENCH 13
REUNIONS « CODAGE » ERREUR ! SIGNET NON DEFINI.

PARTIE II – CONCEPTION ET MIGRATION DE BASES DE DONNEES 14

LES BASES DE DONNEES DE DEPART 14
LA BASE DE DONNEES MYSQL 15

PARTIE III – LE SITE WEB DU CRB CITRUS 23

ARCHITECTURE 23
STRUCTURE DES FICHIERS 26
LES CLASSES 26

LES MODULES 29

LE MODULE « CATALOGUE » 29
LE MODULE « GESTION DE COMMANDES » 29
LE MODULE « GESTION DES VARIETES » 29
LE MODULE « ADMINISTRATION » 29
LE MODULE « RECOLTE » 29
LE MODULE « EXTRACTION » 29
LE MODULE « UTILISATEUR » 30
LE MODULE « COMMANDE » 30
LE MODULE « FNC » 31

PERSPECTIVES 33

CONCLUSION 34

LEXIQUE 35

ANNEXES 36

ANNEXE 1 : MEMO GIT REDIGE POUR LES AUTRES DEVELOPPEURS 36
ANNEXE 2 : MCD DE LA BASE DE DONNEES EGID 37
ANNEXE 3 : FORMAT D'ECHANGE DE FICHIERS DE SIREGAL 38
ANNEXE 4 : MCD DU CRB CITRUS VERSION EGID 39
ANNEXE 5 : INSERTIONGENERIQUEINTERFACE.PY 40
ANNEXE 6 : INSERTIONGENERIQUE.PY 43
ANNEXE 7 : MCD CRB CITRUS SIREGAL 45

Introduction

Il y a un an, l'INRA de Corse avait créé une proposition de poste en CDD sur 9 mois afin d'aider à la conception et la réalisation d'un site web pour l'Unité Expérimentale Citrus. Le sujet du poste à l'époque était « Optimisation d'un outil de gestion informatique appliqué au CRB Agrumes ». La problématique est de proposer un outil de gestion de CRB répondant à la norme AFNOR (NF S 96-900) relative au système de gestion d'un CRB et de la qualité des ressources biologiques, rédigée en septembre 2011. A terme, l'outil doit permettre de gérer les données passeport et de caractérisation, les contrôles sanitaires, le processus de production de semences, la diffusion du matériel végétal et la gestion parcellaire, à travers un outil disponible *via* un site web. Cette proposition de poste a été infructueuse.

Cette proposition s'est transformée en offre de stage de 6 mois pour un étudiant de niveau Master (Master Systèmes d'Informations et Internet (S2I)), dans le cadre de son stage de fin d'études. J'ai accepté ce stage pour lequel la problématique et le résultat à terme sont restés les mêmes, à savoir le respect de la norme AFNOR (NF S 96-900), à travers un outil web.

J'ai concentré mes efforts sur la coordination des forces de développement à travers des outils de développement collaboratif, puis sur la conception et la migration de la base de données du CRB Agrumes (renommé plus tard CRB Citrus) en partant de deux autres bases de données existant précédemment, et enfin j'ai appuyé la conception et le développement du site web permettant de mettre en œuvre cette base de données du CRB.

Contexte

L'INRA

Premier institut de recherche agronomique en Europe, deuxième dans le monde, l'Inra mène des recherches au service d'enjeux de société majeurs : l'alimentation, l'agriculture et l'environnement. L'INRA est composée de près de 8500 agents titulaires dans 17 centres de recherche régionaux et un centre siège, 13 départements scientifiques et 6 grands programmes transversaux de recherche, avec un budget prévisionnel de 881,61M€.

« Dans un contexte climatique, démographique et énergétique complexe, la recherche agronomique doit étudier des enjeux majeurs à des échelles variées. Imaginer la disponibilité et la sécurité alimentaire mondiale en 2050, contribuer à la limitation du gaz à effet de serre d'origine agricole, favoriser l'adaptation de l'agriculture et des forêts au changement climatique non réversible sont autant de préoccupations mondialement partagées. Elles impliquent, entre autres, de connaître les comportements des individus à l'échelle des territoires ou des marchés, d'étudier les liens entre la santé des plantes, des animaux et des hommes, de rechercher de nouvelles voies pour la production d'énergie et de matériaux issus de l'agriculture et d'en limiter en général l'impact environnemental… Pour cela, l'Institut national de la recherche agronomique (Inra) produit des connaissances scientifiques et accompagne l'innovation économique et sociale dans les domaines de l'alimentation, de l'agriculture et de l'environnement. »

Source : institut.inra.fr

Le centre INRA de Corse

Le centre Corse est composé de 50 agents répartis sur quatre unités, trois étant localisées à San Giuliano et une à Corte.

A Corte, le LRDE (Laboratoire de Recherches sur le Développement de l'Elevage) concentre son activité afin de donner au secteur pastoral corse les moyens nécessaires pour concurrencer l'élevage de masse, tout en préservant les traditions locales.

A San Giuliano :

- L'UE Citrus (Unité Expérimentale Citrus) effectue ses recherches autour des agrumes, leur conservation en verger et cryoconservation, et la recherche autour de ces agrumes afin d'optimiser la ressource génétique et réduire les coûts, assurer la pérennité de leur conservation, et d'enrichir la collection à bon escient, sans augmenter les surfaces plantées. L'UE Citrus héberge et gère le centre de ressources biologiques Agrumes qui est une copropriété INRA-CIRAD.
- L' UMR AGAP (Unité Mixte de Recherche « Amélioration Génétique et Adaptation des Plantes méditerranéennes et tropicales » sous la tutelle du CIRAD, de l'INRA et de Montpellier SupAgro) œuvre dans l'amélioration génétique des plantes tropicales et méditerranéennes en vue d'une meilleur adaptation de celles-ci au contraintes environnementales locales. De fait, l'UMR gère des collections importantes de ressources génétiques d'espèces végétales (*Medicago truncatula,* vigne, riz, sorgho, coton, arachide,…), soit près de 60 000 accessions à ce jour. Ces collections sont constituées de différentes catégories de ressources génétiques : collections nationales et ressources patrimoniales

françaises, matériels originaux, espèces apparentées aux espèces travaillées d'origine cultivée ou sauvage, ressources à caractère scientifique. A San Giuliano, l'UMR a une antenne, l'unité de recherche AGAP-Corse qui effectue ses recherches sur les agrumes. Elle étudie la diversité génétique des complexes d'espèces d'agrumes, évalue les déterminismes génétiques de la qualité des agrumes, et créée et sélectionne des variétés innovantes pour l'agrumiculture.

- Les SDAR (Services Déconcentrés d'Appui à la Recherche) regroupent l'ensemble des moyens humains et financiers délégués au Centre pour assurer la mise en œuvre des fonctions d'intérêt collectif relevant de la responsabilité locale.

L'UE Citrus

L'Unité Expérimentale (UE) Citrus de San Giuliano effectue ses recherches autour des agrumes, et est composée de 20 personnes. Elle travaille en coopération avec l'UMR AGAP-Corse afin de créer et de maintenir une collection de ressources génétiques d'agrumes, le CRB (Centre de Ressources Biologiques) Citrus. Ce faisant, cette collection s'intègre à l'effort des autres CRB en France. On compte à ce jour dans le CRB Citrus plus de 1300 entrées de ressources génétiques d'agrumes, environ 6000 arbres plantés en verger et sous serres.

Depuis son origine, l'Unité a toujours valorisé ses ressources génétiques dans le cadre d'échanges scientifiques, d'appui à des projets de développement dans différents pays, mais également de manière purement commerciale traduisant la notoriété internationale de l'Unité dans ce domaine. Le matériel caractérisé, conservé et contrôlé sanitairement est diffusé, dans une trentaine de pays et territoires, sous forme de semences pour les porte-greffes en raison de leur caractère polyembryonné qui permet leur multiplication à l'identique par semis, et sous forme de greffons en ce qui concerne les variétés. L'unité est officiellement reconnue comme producteur de semences de porte greffe d'agrumes dans le cadre de la certification fruitière française. L'unité Citrus s'est naturellement engagée dans une démarche de validation de certification selon la norme AFNOR (NF S 96-900) sur les Centre de Ressources Biologiques végétaux.

Cependant, jusqu'à présent dans le CRB Citrus, la gestion des ressources, des ventes, leur traçabilité étaient faits sur papier, et l'UE souhaite moderniser le CRB à travers l'outil informatique.

Le système d'informations de l'UE Citrus

L'UE Citrus a donc décidé de mettre en place un système d'information, afin de faciliter l'accès aux variétés, et la diffusion des ressources génétiques, ainsi que d'apporter les fonctionnalités de suivi et de gestion de ces ressources, :

Gestion de la collection d'agrumes

La collection d'agrumes est importante et demande une gestion des variétés disponibles, mais aussi une organisation des actes techniques (taille, fauchage, récolte, …)

1. Introduction de ressources biologiques

Il s'agit de mettre en place un système d'ajout de nouvelles variétés ou sous-variétés (accessions, cf. lexique) dans la base de données par le gestionnaire de la collection d'agrumes, de la réception de la demande jusqu'au transfert au processus conservation.

2. Caractérisation des accessions

Il s'agit de définir les caractères essentiels d'une accession selon l'INRA, sur ses différents éléments anatomiques comme sa fleur, son fruit ou même les quartiers, ou segments de l'agrume.

3. Suivi sanitaire

La collection d'agrumes fait l'objet d'un suivi sanitaire très strict à l'échelle de l'arbre. En effet, pour être diffusable, le matériel végétal doit-être sain de tous parasites ou maladies de quarantaine pour satisfaire les réglementations internationales en matière de circulation de matériel végétal. Tous les actes techniques de suivi sont ainsi consignés dans la base de données. Ceci inclut les actions sanitaires faites sur les lots de graines, comme par exemple l'utilisation d'antifongiques.

4. Cartographie des champs

Cette rubrique regroupe les plans des parcelles (mis à jour dynamiquement). Il a été déterminé que les plans des parcelles doivent être modifiables depuis l'interface web. Un champ contient plusieurs parcelles, et chaque parcelle est subdivisée en cartouches qui correspondent à une accession, contenant un ou plusieurs arbres (suivant les accessions).

5. Capacité à transférer les données-passeport des accessions à la base de données SIREGAL

L'objectif est de développer une interface de transfert des données-passeport des accessions à la base de données nationale de l'INRA, SIREGAL, développée par l'unité URGI du centre de Versailles. Des scripts seront fournis afin de transformer les informations-passeport de la base de données, et de les écrire dans le format de fichier de transfert vers SIREGAL. Cette technique permet une réalisation plus rapide de ces fichiers.

6. Gestion des problèmes survenant dans le CRB (« non-conformités »)

Ce module permet à n'importe quel gestionnaire du CRB d'éditer des fiches de non-conformité (FNC) rendant compte de situations anormales, à régler. Ceci permet une centralisation des problèmes dans le CRB.

Processus de diffusion des ressources biologiques.

L'UE diffuse auprès de partenaires scientifiques et des professionnels des graines et des greffons. Cette diffusion est actuellement gérée sur des fiches papiers puis reportées sur un fichier Excel partagé en réseau. Il devenait urgent d'informatiser ce processus pour des raisons de traçabilité et pour faciliter le partage de l'information. La diffusion du matériel végétal suit un processus normalisé, allant de la récolte du fruit à l'expédition du matériel végétal en passant par le conditionnement. Pour des raisons de traçabilité, ce processus est suivi étape par étape et consigné dans la base de données.

1. Suivi des récoltes : Les récoltes sont effectuées chaque saison de septembre à juin. Chaque récolte possède un identifiant unique, une date, une accession associée, un format (graines, porte-greffe, fruit…) et une unité (masse de fruits, nombre de feuilles ou de porte-greffes…)
2. Extraction de graines : Une fois le fruit récolté, il est passé dans un extracteur afin d'en retirer les graines. Chaque extraction se voit attribué un identifiant unique, une date, une accession associée, un nombre de cagettes, et la masse de graines extraites.
3. Conditionnement : Après le passage à l'extracteur et le séchage, les graines sont ensuite conditionnées dans des sacs puis mis en chambre froide. Les sacs sont identifiés par un numéro de lot.
4. Précommande : Il existe une grande quantité de variétés disponibles à la diffusion dans le catalogue du CRB. Les utilisateurs ont la possibilité d'établir une liste des accessions et des formats qu'ils souhaitent obtenir, ainsi que la quantité souhaitée.
5. Arbitrage des commandes : Une fois que le client a formulé son souhait de commande, le gestionnaire responsable des commandes détermine les ressources disponibles selon le type de matériel végétal qui est demandé et le type de client (chercheur, professionnel, particulier). L'arbitrage des commandes est nécessaire car le CRB fonctionne à flux tendu sur les lots disponibles. Cet arbitrage est fait au sein du comité de pilotage du CRB.
6. Acheminement : Dans cette étape, le gestionnaire prépare l'envoi du matériel végétal et assure le suivi de l'acheminement par les organismes postaux du courrier ou de colis : les envois sont préparés sur place et se voient attribuer un numéro de suivi (Collissimo ou Chronopost, selon la destination).

La figure 1 montre le diagramme décisionnel du processus de diffusion à partir de la phase 4.

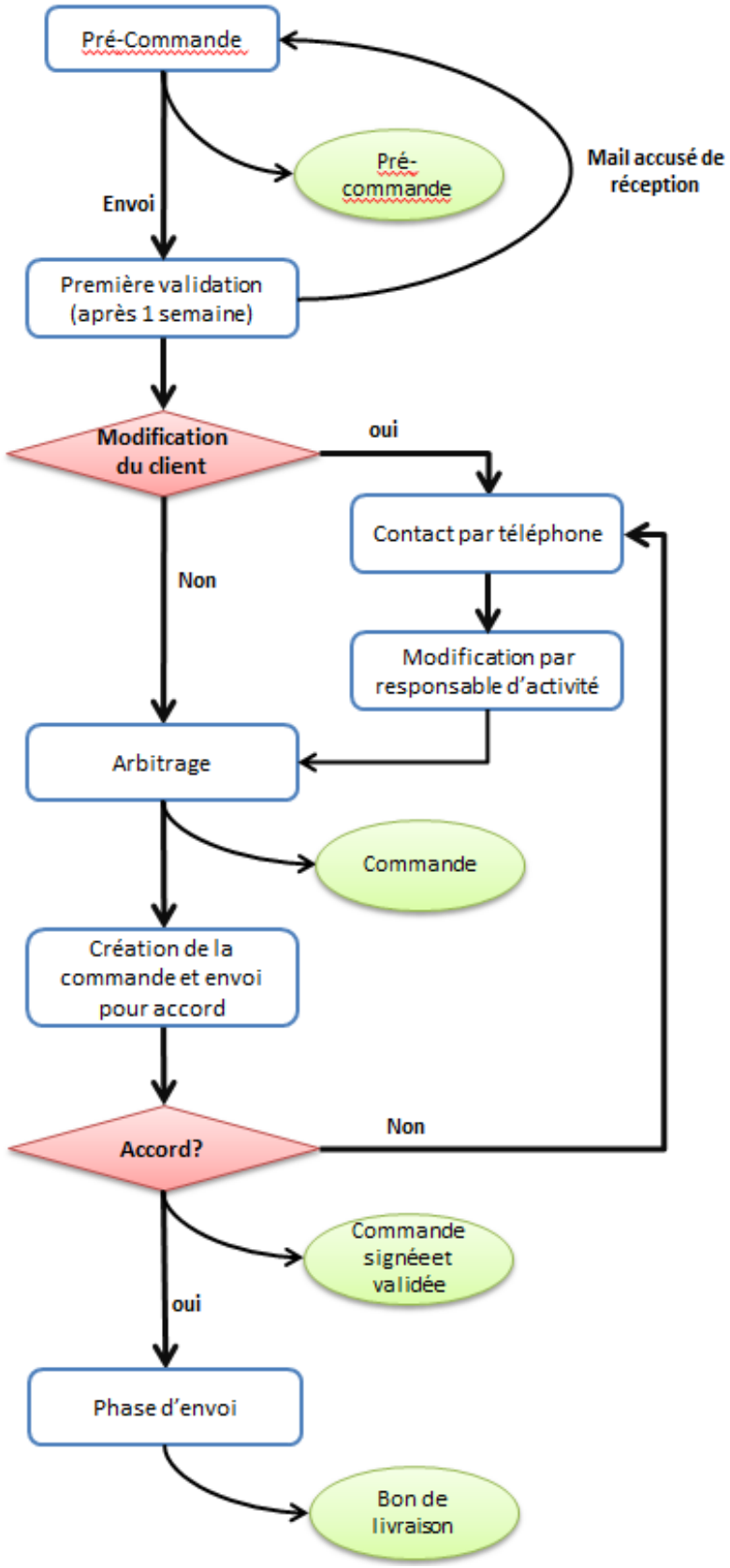


Figure 1 : le processus décisionnel du gestionnaire de précommandes

Partie I – Matériel et méthode

Dès le début du stage, j'ai eu l'occasion de participer à une réunion de conception/codage que les développeurs du site avaient mis en place. J'ai pu ainsi être intégré au projet rapidement, et noter **certains problèmes à régler dans l'immédiat**, comme le besoin d'outils de gestion de projet, et de développement asynchrone.

Choix techniques

Deux options s'offraient à nous : un client applicatif dit lourd ou un client web dit « léger ». Les avantages du client lourd sont sa disponibilité et sa rapidité d'exécution, mais son inconvénient réside dans ses processus d'installation et de mise à jour. A l'inverse, bien qu'un client léger soit plus long à l'exécution (visualisé dans une page web) et requiert bien plus de connaissances à développer (différents langages, différents paradigmes) ses mises à jour et sa disponibilité intranet sont assurées et son accessibilité au grand public sur internet est immédiate, sans nécessité d'installation.

L'architecture choisie a donc été l'architecture web, avec pour base technique, un serveur type Linux / Apache / MySQL / PHP (LAMP) principalement pour les raisons suivantes.

1. La distribution Linux choisie est CentOS, une distribution dérivée de Linux RedHat, standard en tant que serveur web et généralement utilisée à l'INRA.
2. PHP est un langage orienté objet et adapté aux applications web. De plus, deux des développeurs étant débutants, PHP facilite leur apprentissage.
3. MySQL car c'est un système de base de données libre, robuste, et avec une grande communauté, ayant déjà répondu à la majorité des problèmes et bugs aperçus dans le logiciel *via* des sites comme *stackoverflow*.
4. Apache pour les mêmes raisons que MySQL, étant un serveur web robuste et largement utilisé à travers le web.

Git

Git est un outil de développement collaboratif, développé par Linus Torvalds durant la création du noyau Linux. Il permet la gestion de projet à travers l'agrégation de code par différences entre les fichiers, et la traçabilité des modifications apportées au fur et à mesure.

Il s'agissait de coordonner les différentes forces de développement. En effet, jusque-là les différentes modifications apportées au code du site étaient enregistrées à distance *via* FTP, **en synchronisation directe**, ce qui apportait de **nombreux problèmes** dès que deux développeurs changeaient les mêmes fichiers, ou des fichiers impactant d'autres ressources. Il m'a semblé évident qu'un outil de développement collaboratif devait être mis en place. J'ai choisi Git, outil avec lequel j'avais le plus d'expérience comparé à ses concurrents comme Subversion ou Mercurial. Il m'a fallu former les différents développeurs à cet **outil puissant mais difficile d'accès**. J'ai édité un mémo à destination des autres développeurs, il est disponible en Annexe 1, externe.

Git permet entre-autre, de créer des branches de développement, afin de les fusionner en retour à la branche parente. Ce faisant, il a été rendu possible de **développer chacun sa fonctionnalité sans corrompre les fonctionnalités des autres**, tout en réglant les conflits dans les fichiers au moment de la fusion des branches.

Afin de structurer le projet, les développeurs et moi-même avons décidé d'un commun accord et pour des raisons de maintenabilité, de mettre en places ces branches de développement :

1. **Master** : cette branche contient la version en production du projet.
2. **Develop** : cette branche dérive de Master, et contient la version en développement du projet.
3. **Feature** : dérivées de Develop, les branches Feature/nom_de_fonctionnalité permettent le développement d'une fonctionnalité à la fois, sans impacter le développement des unes sur les autres. Quand la Feature est développée et testée, elle est fusionnée à la branche Develop.

Les développeurs utilisent l'IDE Eclipse avec le plugin EGit pour remplir les fonctionnalités de git de manière intégrées et interfacée, ce qui est bien plus efficace en terme d'apprentissage.

Architecture autour de Git

J'ai mis en place une architecture vis-à-vis de git, permettant à d'autres informations à l'INRA de pouvoir collaborer sur le projet, en particulier les quatre développeurs principaux. Il existe à présent un serveur contenant le projet sous forme d'un dépôt git dit « bare », sans arbre de travail, et d'un dossier « clone » du dépôt, contenant un arbre de travail, et réglé pour avoir le contenu de la branche « master » mis à jour tous les jours, puis copié sur un autre serveur avec le script suivant :

```
#!/usr/bin/env sh
cd /var/www/CRB_agrumes_git
git checkout master
`which rsync` -avz -e ssh --exclude=".git"
/var/www/CRB_agrumes_git/* root@147.100.149.226:/var/www/CRB_citrus
```

Cet autre serveur est le serveur de production, qui sera disponible au grand public *via* un navigateur web. Y sont copiés les fichiers de la branche « master » directement, sans dépôt (.git) pour ne pas avoir de trou de sécurité à ce niveau-là, avec l'outil rsync et un cron-job, disponible sous Linux sur le serveur de développement.

Enfin, chaque développeur a besoin localement d'un serveur web et d'une base de données MySQL, et de cloner le dépôt du serveur de développement.

La figure 2 résume l'architecture mise en place :

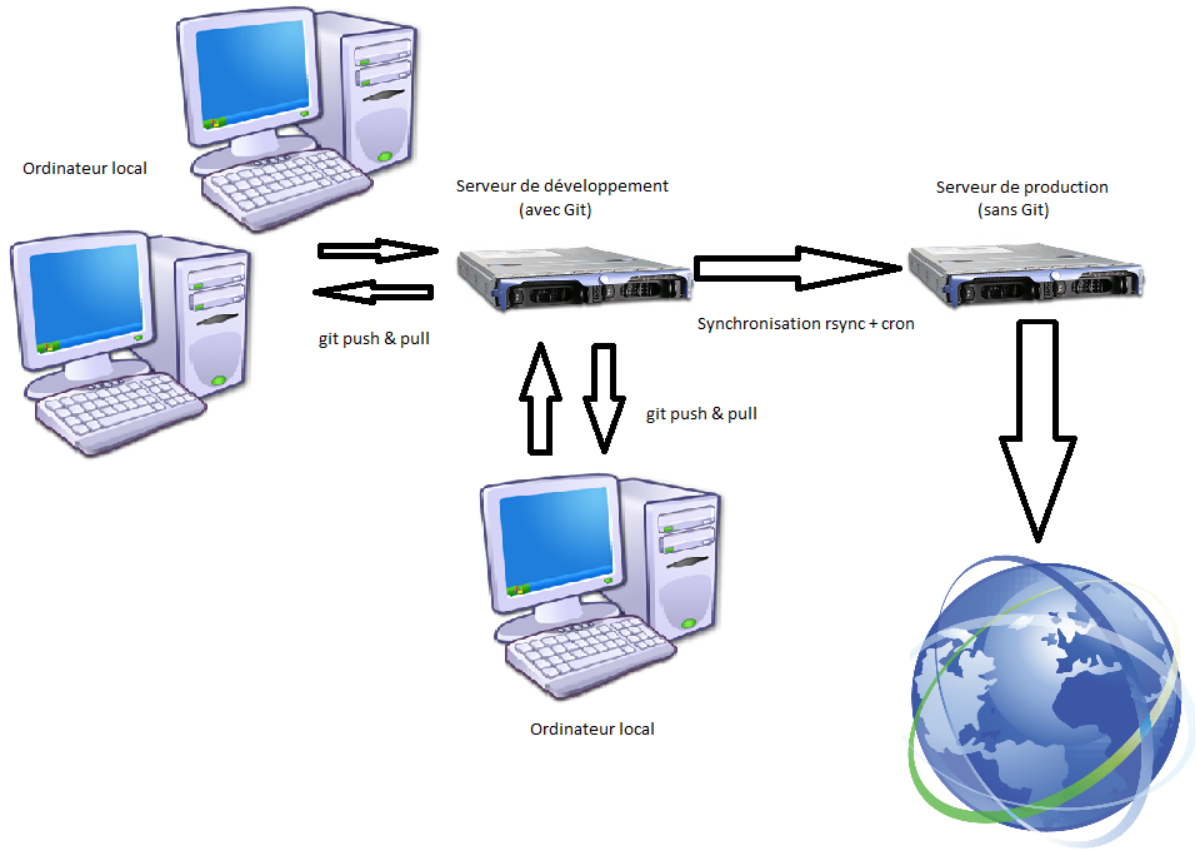


Figure 2 : architecture physique autour de git

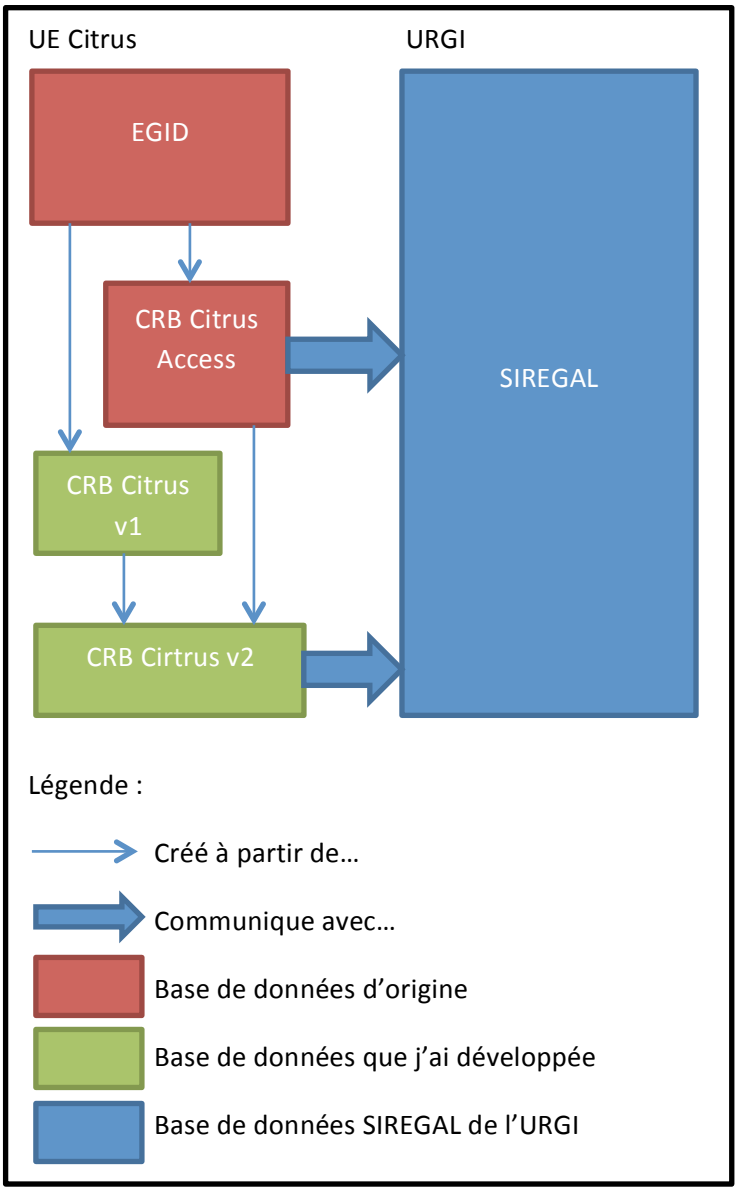
Avoir pu mettre en place Git a permis aux développeurs de collaborer bien plus efficacement, bien que la période d'apprentissage soit relativement longue. Git n'est pas un système accessible au plus grand nombre du fait de sa puissance et de sa complexité.

MySQL Workbench

Afin d'accélérer le développement de la base de donnée, j'ai utilisé le logiciel MySQL Workbench. Il s'agit d'un concepteur de modèles de données, capable de transposer directement le modèle en définitions de base de données dans MySQL. Ce fût un gain de temps considérable, et cet outil m'a permis d'en apprendre plus sur MySQL et notamment sur la gestion des clés étrangères et index.

Partie II – Conception et migration de bases de données

Afin de réaliser le système d'information du CRB Citrus, il m'a tout d'abord fallu effectuer la conception et refonte de la base de données existante EGID développée à San Giuliano, ainsi que l'intégration d'une structure pouvant permettre le transfert vers la base de données nationale SIREGAL. Voici un aperçu du cheminement effectué :



Les bases de données de départ

EGID

EGID, acronyme de « Evaluer, Gérer, Informatiser, Diffuser », est une base de données de ressources végétales créée par Roland Cottin, conçue autour de la « variété » et de ses données. Elle se décompose en plusieurs sous-ensembles de tables :

1. Le cœur (Variety et Data)
2. La taxonomie (cf. Lexique) de la variété

3. La géographique autour de la variété (site de découverte…)
4. La traçabilité des arbres (un arbre est lié à une Accession, qui est un champ de la table Data)
5. La traçabilité des variétés utilisées comme porte-greffes
6. Un ensemble de tables de définitions de descripteurs, nommées Level1, Level2, Level3 et Level4. Ces définitions servent à donner un nom ou des valeurs à tous les champs dont les noms sont composés uniquement de chiffres et de tirets. On notera aussi l'absence de contrôle de cohérence, permettant à certains champs d'exister sans nom ni valeur dans les tables Level* (1-5-15 par exemple).
7. La table UPOV83, taxonomie relative à l'Union internationale pour la protection des obtentions végétales, ainsi que ses deux tables associées Upov1 et Upov2, ayant la même fonction que les tables Level*.

Cette base de données a été conçue avec Microsoft Office Access, la rendant peu adéquate pour une utilisation dans un système d'informations multi-utilisateur. De plus, la convention de nommage, bien que régulière et arborescente, est très singulière, illisible, et pratiquement impossible à maintenir dans des délais raisonnables (exemple de noms de colonnes : 1-2-1, 1-2-2, 1-3-6). De ce fait, il m'est apparu évident que cette base avait besoin d'une refonte. Le MCD de EGID est disponible en Annexe 2.

C'est sur cette base de données que j'ai conçu la première base CRB Citrus version EGID.

SIREGAL

SIREGAL est un projet de l'unité URGI de l'INRA (Versailles) dont le but est de construire et maintenir le système d'information centralisé des ressources génétiques des plantes de l'INRA. Actuellement, ce projet recense 50 espèces de plantes.

Afin de communiquer à SIREGAL les ressources génétiques disponibles dans un CRB, il faut utiliser leur format de fichiers d'échange. Il s'agit de fichiers CSV, dont la structure est claire et disponible en Annexe 3. C'est à partir de ce format de fichiers d'échanges que l'outil d'Emmanuel Bloquel et la base CRB Citrus v2 ont été conçus.

CRB Citrus (sous Microsoft Access)

Un outil de gestion du CRB Citrus de San Giuliano et une base de données basée sur les fichiers d'échange de SIREGAL ont été développés par Emmanuel Bloquel, gestionnaire du CRB, sous Microsoft Office Access. La nouvelle base de données n'est plus centrée autour de la variété pour la raison suivante : la définition de la variété est très instable dans le milieu de la recherche en agronomie. En revanche, la définition de l'accession elle, est très stable. Partant de ce constat, la base de données du CRB Citrus est centrée autour de l'accession.

Encore une fois, l'utilisation de Microsoft Office Access n'est pas adéquate dans un système multi-utilisateur. Cependant, je retiens quand même l'avantage d'avoir pu créer une interface graphique à la base de données, directement dans Access.

La base de données MySQL

Cette partie détaille comme les trois étapes pour construire la base de données du système d'information du CRB en me basant sur les travaux existants.

CRB Citrus v1 (basée sur EGID)

Quand je suis arrivé dans l'Unité, une première version basique du site web en cours de création existait déjà, et disposait d'une base de données simplifiées à des fins de tests. J'ai eu pour première mission, de concevoir et développer la base de données qui sera à terme utilisée par le site web, avec deux objectifs : l'unicité des données, et l'évolutivité de la structure sans toucher au cœur de celle-ci.

Pour commencer, j'ai conçu la nouvelle base de données à partir d'EGID. J'ai rétabli l'intégralité des noms des colonnes à partir de la nomenclature disponible dans les tables Level1 à Level4, ce qui fût un long processus d'allers et retours dans les tables sous Acces et l'outil MySQL Workbench.

Les deux tables principales Variety et Data ont été ensuite décomposées, afin de séparer les informations essentielles permettant de définir une Variety, des non-essentielles ou étant apparentées, comme la caractérisation ou la prospection. Le nombre final des tables issues de cette décomposition atteint 109, classées selon ces catégories :

1. Données essentielles
    1. Variety
    2. Data
2. Caractérisation
    1. Appareil Vegetatif
    2. Feuille
    3. Fleur
    4. Fruit
    5. Segments
    6. Pulpe
    7. Jus
    8. Pepins
    9. Calibres

La dimension du diagramme de base de données ne permet pas de l'intégrer ici, il est disponible en Annexe 4.

Ont été ajouté ensuite les tables relatives à la taxonomie (encart Taxonomie sur le diagramme), les informations géographiques (encart Country), les informations concernant les arbres physiques disponibles dans le CRB, les informations de la taxonomie définie par l'UPOV (Union internationale pour la Protection des Obtentions Végétales), puis finalement certaines des tables liées au site web, autour du processus de Diffusion (encart Commande).

La figure 3 montre pour l'exemple, la table de caractérisation des Feuilles.

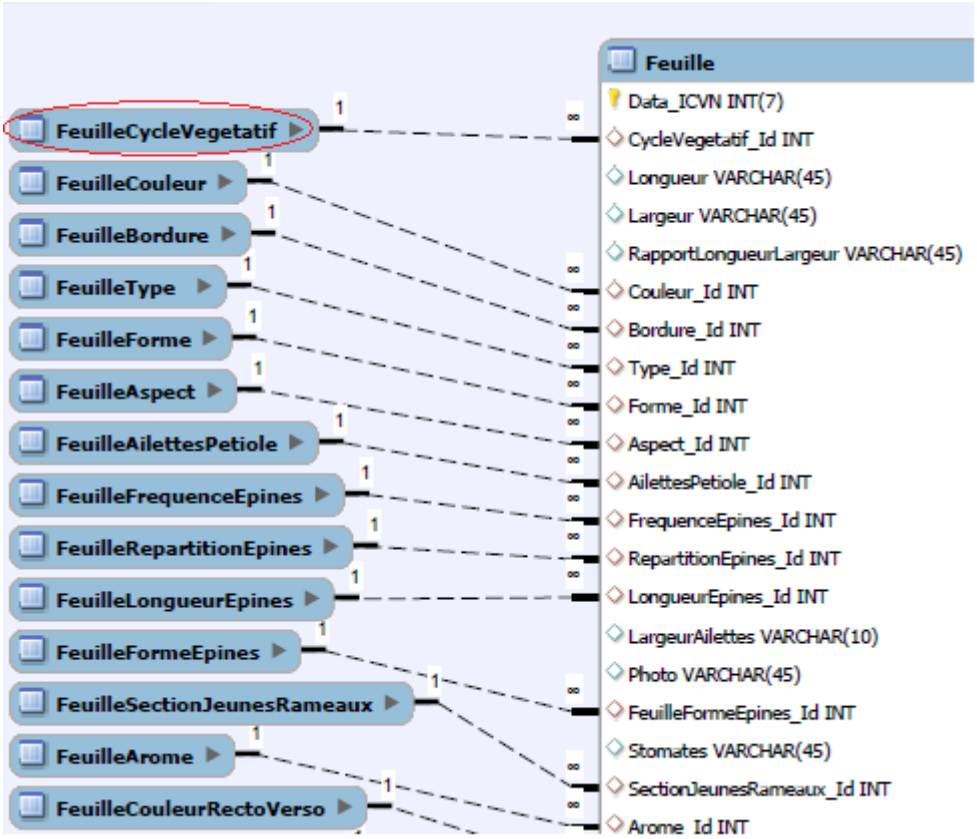


Figure 3 : La table Feuille, à droite, et les tables de valeurs pour la caractérisation, à gauche.

La table entourée en rouge, comme toutes les autres tables réduites, contient simplement deux champs, Id et Valeur. Ces tables ont été conçues ainsi pour plusieurs raisons :

1. La table EGID qui référence ces valeurs est la table Level4. Elle contient l'intégralité de ces valeurs, et ajouter une valeur possible pour une caractérisation correspond à ajouter une ligne dans cette table Level4, ce qui en fait après une tâche plus ardue pour concevoir les requêtes SQL nécessaires au site web final, et ralenti grandement toute forme de recherche manuelle d'information
2. Le système de nommage des champs de EGID soulève la question de l'efficacité en termes de temps de la maintenance de la base, étant obligé de faire des allers et retours avec la documentation pour retrouver les bonnes valeurs
3. Ajouter une valeur possible revient donc dans la nouvelle base, à ajouter un enregistrement à une de ces tables, permettant de retrouver rapidement les valeurs possibles
4. Les requêtes pour MySQL vers ces tables peuvent faire usage de jointures afin de récupérer les valeurs textuelles à afficher.

Les tables de valeurs réduites ont été remplies à la main, et à terme seront remplies et maintenu *via* des formulaires web.

L'action logique après avoir conçue cette base, fût celle de migrer les données depuis la base de données EGID. J'ai décidé de faire cette migration à travers des scripts écrits en Python, car ce langage est très rapide à mettre en œuvre, peut effectuer des traitements à la volée, et a certaines

facilités à se connecter aux différentes bases de données *via* le module pyodbc pour Access, et la librairie ORM peewee pour MySQL.

Un ORM (« Object Relationnal Mapping », ou « Association Relationnelle Objet») est une technique de programmation permettant de donner l'illusion d'une base de données orientée objet, et utilisant une base de données relationnelle. Dans la pratique, cette technique permet de ne plus avoir à taper manuellement les requêtes SQL, à la condition que les requêtes que l'outil ORM forme soient valides.

La librairie *peewee*, développée par « Coleifer », m'a permis de disposer d'un ORM léger pour faire communiquer Python avec MySQL. Cependant, la définition des tables était à faire à la main, une table correspondant à une classe Python, le temps qu'il m'aurait fallu à retranscrire toutes les tables en classes à la main aurait été perdu. Afin de palier à ce problème, j'ai développé un script capable de rétro-concevoir une base de données sous forme de classes ORM Python, utilisant *peewee*. Le script se décompose comme suit :

1- Importer les modules nécessaires
2- Initialiser la connexion à la base de données
3- Supprimer le dossier ORM correspondant à la base de données, s'il existe, afin de faire une éventuelle mise à jour des classes
4- Ecrire le fichier *__metadb__.py*, avec la possibilité de se connecter à distance selon les arguments passés
5- Obtenir la liste des tables de la base de données
6- Pour chaque table, générer une liste des champs de la table, récupérer l'ensemble des informations de chaque champs (type, taille, clé primaire, index, unicité, clé étrangère, actions sur la clé étrangère, valeur par défaut, valeurs possibles, auto-incrément). La liste des champs est transformée en Structures, ensemble de classes d'état disponible pour chaque type de MySQL, et disponibles dans le fichier *peeweemysqldata.py*.
7- Vérifier les index composites et les noms des clés étrangères, les renommer si besoin pour éviter les doublons
8- Ecrire le fichier ORM correspondant à la liste des champs de la table
9- Une fois toutes les tables transformées en fichier ORM, écrire la liste des modules contenant les classes ORM dans le fichier *__init__.py*

Voici le fichier *__metadb__.py*, contenant les informations de connexion à la base de donnée, la définition du type de champs ENUM, et la classe BaseModel, qui fait le lien entre les fichiers ORM et la base de données MySQL à travers la méta-classe :

```
#!/usr/bin/env python2.7
#-*-encoding: utf-8-*-
'''
Meta-informations about the database.
It includes a new types for MySQLDatabase to manage :
    EnumField
'''
import peewee
from peewee import *

dbname = 'crb_citrus_siregal'
login = 'root'
passwd = '****' # Le mot de passe doit évidemment être écrit en clair

class EnumField(Field): # EnumField dérive de Field
    '''
    Enables the enum type for peewee.MySQLDatabase to manage.
    (warning note :
        http://komlenic.com/244/8-reasons-why-mysqls-enum-data-type-is-
evil/ )
    '''
    db_field = 'enum' # Variable de classe
    def __init__(self, *args, **kwargs): # Constructeur
        self.enum_values = None
        if "values" in kwargs:
            self.enum_values = kwargs["values"]
        Field.__init__(self, kwargs)

    def db_value(self, value): # Méthode d'instance (avec "self")
        if self.enum_values is None:
            return str(value)
        if value in self.enum_values or value in
range(len(self.enum_values)):
            return value
        else:
            return ""
    def python_value(self, value):
        return str(value)

# Permettre à MySQLDatabase de gérer l'Enum (peewee a été fait pour MySQL,
PostGRESQL et SQLite, et les deux derniers n'ont pas de type Enum)
setattr(peewee, "EnumField", EnumField)
MySQLDatabase.register_fields({'enum': 'ENUM'})

# Créer une connexion encodée en UTF-8 vers MySQL
db = MySQLDatabase(dbname, user=login, passwd=passwd)
db.get_conn().set_character_set('utf8')

# Créer une classe de base dont toutes les autres classes hériteront, et
qui contient la connexion à la base de données. BaseModel dérive de
peewee.Model
class BaseModel(Model):
    class Meta:
        database = db
```

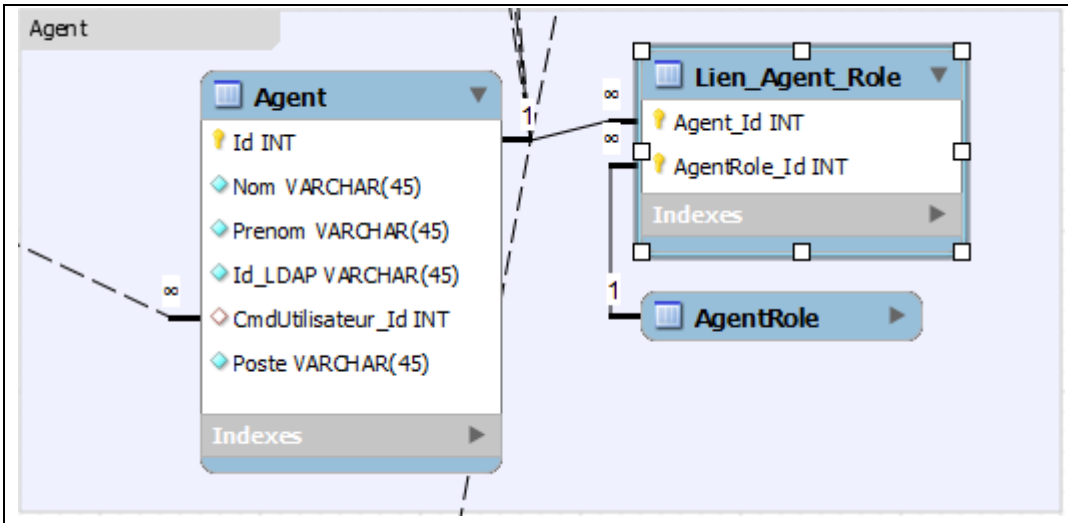


Figure 4 : l'encart Agent, montrant la table Lien_Agent_Role et ses clés étrangères.

Voici le code ORM correspondant généré par *peeweemysqlobjects.py* d'après la table Lien_Agent_Role dans la figure 4, tel qu'il aurait été tapé à la main :

```
#!/usr/bin/env python2.7
#-*-encoding: utf-8-*-
#crb_citrus_siregal/lien_agent_role.py

from peewee import *              # ForeignKeyField, CompositeKey
from __metadb__ import *         # BaseModel
from agent import agent          # L'ORM de la table agent
from agentrole import agentrole # L'ORM de la table agentrole

class lien_agent_role(BaseModel): # Table correspondante
    Agent_Id = ForeignKeyField(agent, # Agent_Id - champ correspondant
        related_name = 'fk_agent_Id_4',
        db_column = "Agent_Id",
        on_update = "NO ACTION",
        on_delete = "NO ACTION")
    AgentRole_Id = ForeignKeyField(agentrole, # AgentRole_Id
        related_name = 'fk_agentrole_Id',
        db_column = "AgentRole_Id",
        on_update = "NO ACTION",
        on_delete = "NO ACTION")
    class Meta:
        primary_key = CompositeKey('Agent_Id', 'AgentRole_Id')
```

La représentation de la table lien_agent_role en tant que classe contient chaque champ défini selon les types disponibles avec peewee (comme ForeignKeyField), ainsi que la clé primaire composite.

Relancer le script de génération à chaque modification du schéma de la base de données permet d'avoir les définitions des classes Python beaucoup plus rapidement à jour.

Ce script et sa dépendance sont à la disposition de tous à l'adresse http://github.com/Shade2b/PeeweeMySQLObjects.

Suite à la création automatique des fichiers ORM, plusieurs scripts ont été conçus, avec plusieurs spécifications dans le résultat final (script en Annexe 5) :

1. N'avoir à taper que les noms des bases de données, tables et champs relatifs au transfert en cours
2. Utiliser le premier champ pour une table donnée, comme identifiant afin de faciliter la recherche d'un enregistrement déjà existant, dans le cas où le transfert se fait en plusieurs fois
3. Numéroter les champs correspondants d'une table à l'autre
4. Permettre de changer des valeurs à la volée, selon un offset ou une liste de changements à appliquer. En effet, l'auto-incrément commence à 1, et selon les clés étrangères dans la base de données MySQL, certaines valeurs dans EGID étaient illégales, comme 0 et -1 dans les tables de caractérisation.

L'algorithme de transfert utilisé se décompose comme suit (script en Annexe 6) :

1. Construire la requête SELECT nécessaire pour lire les données de la table d'Access, ainsi que d'indiquer la colonne de tri ; exécuter la requête ;
2. Pour chaque enregistrement récupéré dans la base de données Access:
    1. Procéder au traitement des enregistrements, et appliquer les modifications à la volée
    2. Récupérer l'enregistrement MySQL correspondant via peewee. Si l'enregistrement n'existe pas, le créer, et le récupérer ; si une erreur survient, la noter en cache ;
    3. Attribuer tous les champs notés dans la liste des champs donnée, dans la classe représentant la table
    4. Sauvegarder l'enregistrement ; si une erreur survient, la mettre en cache ;
3. Si des erreurs apparaissent, les afficher

La migration des données a pris dix semaines, notamment à cause de la convention de nommage incongrue, des erreurs de clés étrangères et des bugs à corriger.

Arrivé à ce stade, lors d'une « réunion codage », on a pu discuter autour de la base de données obtenue, et arriver à la conclusion que cette base n'était pas compatible avec les fichiers de transferts vers SIREGAL : arriver à en tirer toutes les informations nécessaires était devenu une tâche irréaliste. Une nouvelle version de la base a dû être conçue.

CRB Citrus v2 (basée sur les fichiers de transferts de SIREGAL)

Cette nouvelle version de la base a été conçue pour correspondre au mieux au système de fichiers de transferts de données passeport de SIREGAL. Les définitions de ces fichiers sont disponibles en Annexe 3. Le schéma de base de données du CRB Citrus-Siregal est disponible en Annexe 7.

Afin de concevoir cette nouvelle version de la base de données, l'encart Data a été supprimé, pour laisser place à l'encart Siregal, contenant la table Accession, plus ou moins équivalente à la table Data de EGID. Bien qu'absent dans SIREGAL, j'ai décidé de conserver l'encart Variete. Cela m'a permis de faire la transposition des associations entre Taxonomie et Variete vers les associations entre Taxonomie et Accession. Une définition de la table UPOV existait dans l'outil CRB Citrus version Access, mais ne correspondait pas à celle disponible dans EGID. J'ai préféré conserver les deux formats et attendre de pouvoir discuter à ce sujet avec les agronomes, afin de déterminer si je devais retirer l'ancienne ou non. Ces discussions n'ont pas eu lieu pour le moment.

A mesure que le projet a avancé et lors des réunions « codage », j'ai ajouté les encarts FNC, Agent et Recolte. A l'écriture de ce rapport, ce sont 171 tables, plus de 1300 accessions, 102 043 enregistrements au total qui existent dans cette base de données, pour un poids d'environ 36Mio (dont 4Mio d'index).

Bien que le projet ne soit pas à son terme, la majorité des données présentes dans la base CRB Citrus SIREGAL version Access d'Emmanuel Bloquel sont aussi dans ma base de données MySQL. Comparativement, la base de données Access pèse plus de 270Mo. Ajoutons à cela que la base Access, ainsi que les fichiers de transferts de données doivent être copiés sur tous les postes pour maintenir la cohérence entre les bases, le coût en matière de sauvegarde sur serveur robot de sauvegarde, et de bande passante à l'ouverture d'une session itinérante Windows, on peut dire que la version MySQL de la base de données CRB Citrus SIREGAL est de loin plus légère et donc plus intéressante que la version Access.

On peut argumenter que la base Access intègre aussi les formulaires, interfaces utilisateurs, plans des parcelles au format PDF, et diverses images relatives à la caractérisation des accessions, à des fins d'impression, ce qui prend de la place en mémoire, et sont des fonctionnalités que MySQL ne propose pas nativement. Cependant, en prenant en compte le poids des fichiers du site web au stade de développement présent, on atteint 82Mio, ce qui est toujours bien inférieur aux 270Mio de la base Access.

Partie III – Le site web du CRB Citrus

Le site web de l'UE Citrus permet l'accès à la collection du CRB Citrus depuis un utilisateur extérieur, et la gestion du CRB Citrus par les gestionnaires, en interne. Dans ce chapitre, nous aborderons l'architecture et la décomposition en modules du site web.

Architecture

Dans cette partie, nous verrons la charte graphique, la navigation, la structure de fichiers et l'architecture des classes du site web.

Charte graphique

Le site web utilise un seul fichier CSS pour assurer la cohérence de l'intégralité du site : les tons de verts utilisés représentent les couleurs de l'INRA ; les effets visuels sont minimalistes, apparentés à du Flat Design : de forts contrastes, et des boutons lisibles ; on pourra aussi remarquer la faible présence d'images. La figure 5 montre la bannière commune à toutes les pages.

Figure 5 : la bannière commune du site

Navigation

Le site web est composé d'une page principale, index.php, qui contient la bannière commune à toutes les pages vue en figure 5, et où sont incrustées ensuite les autres pages. Par exemple, la figure 6 montre la page catalogue.php.

Figure 6 : la page catalogue

De plus, l'accès à toutes les fonctionnalités se fait au maximum en suivant deux liens, ce qui est un excellent indicateur en termes d'ergonomie : toutes les informations sont disponibles rapidement.

Des pages supplémentaires permettent de remplir des « shadowbox », qui sont basées sur une librairie Javascript permettant de créer des fenêtres modales (popup) avec un fond noir translucide et d'afficher une autre page dans un cadre par-dessus la page web. La figure 7 montre la shadowbox de la page obtenue en cliquant sur « Plus de détail », un des boutons visibles dans la figure 6.

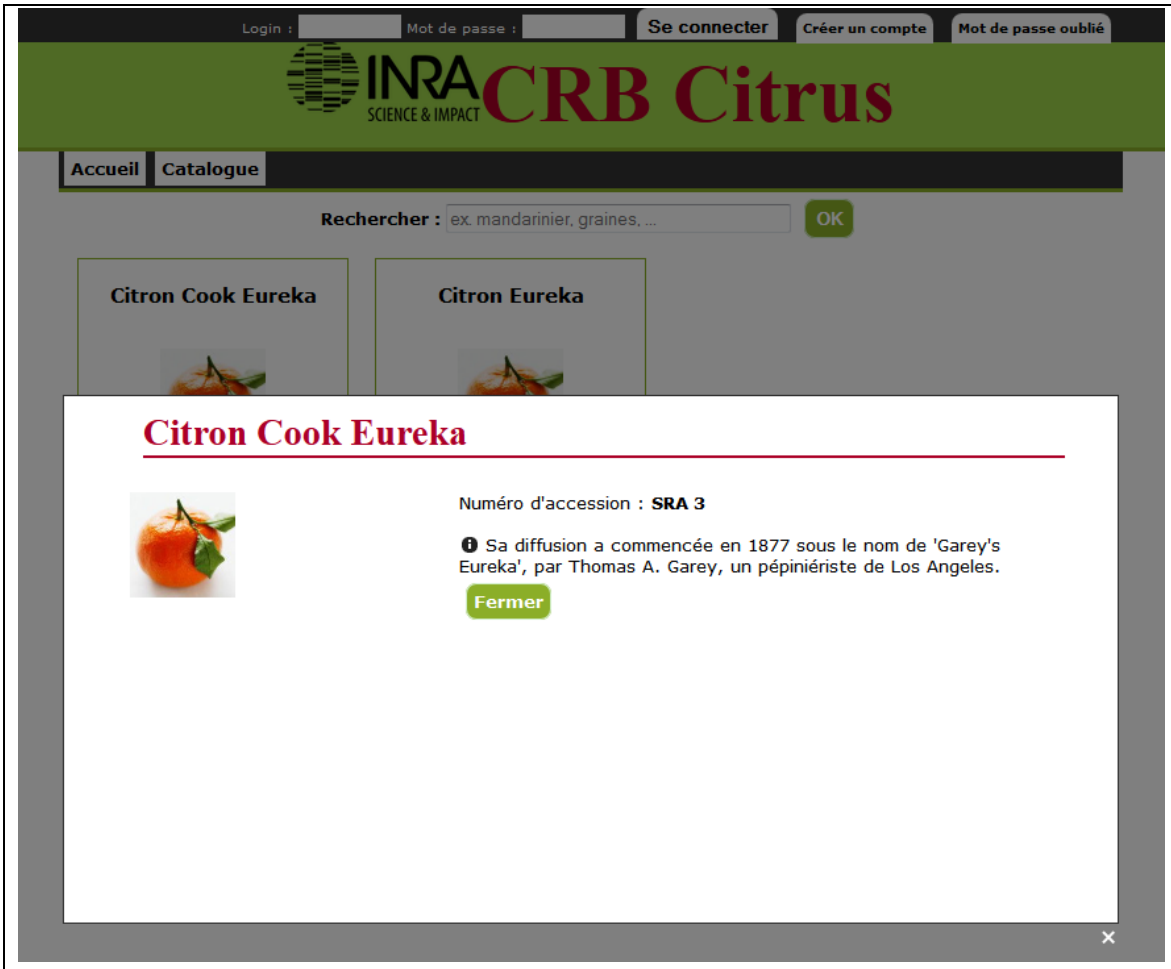


Figure 7 : exemple de « shadowbox »

On remarque la présence d'une zone d'authentification en haut de la bannière, ainsi que deux boutons, « Accueil » et « Catalogue », dans les figures précédentes. Il existe une hiérarchie d'utilisateurs du site, lesquels permettent l'accès à différentes fonctionnalités selon les droits. Les quatre niveaux principaux de navigation sont :

- « Grand public », dont l'accès se limite à la page d'accueil et à l'affichage de la collection du CRB
- « Client » : il peut commander du matériel végétal, suivre sa commande, gérer ses informations personnelles

- « Gestionnaire » : les gestionnaires sont les personnes qui vont gérer en interne le CRB. Ils ont différents rôles, et ne peuvent pas accéder à toutes les fonctionnalités en même temps. Parmi leurs accès, on peut citer la gestion des variétés, la gestion des parcelles, la gestion des précommandes, la rédaction de fiches de non-conformité…
- « Administrateur », qui implique notamment la gestion des utilisateurs, le lien entre utilisateur et gestionnaire, et les rôles des gestionnaires.

Choix techniques

Le site en lui-même a été développé en HTML5, ainsi que plusieurs librairies Javascript. jQuery, une API compatible avec les navigateurs les plus répandus, est utilisée afin de rendre le site plus dynamique à travers AJAX, et de faciliter les modifications du document html à la volée. De plus, l'extension jQuery UI est utilisée, et permet d'ajouter des éléments d'interface utilisateur, comme par exemple un calendrier pour remplir un champ de formulaire.

Le retour des requêtes AJAX se fait soit en XML pour sa verbosité, soit en JSON pour sa simplicité d'utilisation avec Javascript.

JSON signifie « JavaScript Object Notation ». Un objet JSON est un tableau associatif, dont voici un exemple :

```
Tableau = {
    "key1": value1,
    "key2": {
        "key3": value2
        "key4": value3
    }
}
```

Ce tableau associatif s'utilise en Javascript comme un objet classique, en utilisant la même syntaxe :

```
Variable1 = Tableau.key1;
Variable2 = Tableau.key2.key3;
```

Le retour en JSON est utilisé dans le module FNC, et contient plusieurs informations puis la liste des fiches. La figure 8 montre la page d'entrée du module FNC, les zones entourées en rouge étant modifiées quand l'année est changée, sans recharger la page.

Figure 8 : La page d'entrée du module FNC (Fiche de Non-Conformité)

Enfin, la librairie html2pdf est utilisée pour exporter les données au format PDF, format se prêtant bien à l'échange de fichiers d'informations d'un ordinateur à un autre, et à l'impression.

Structure des fichiers

Le développement du site web ayant commencé avant mon arrivée, une structure de fichiers a été mise en place par les autres développeurs. La figure 9 montre l'arborescence de fichiers actuelle, issue de deux refontes :

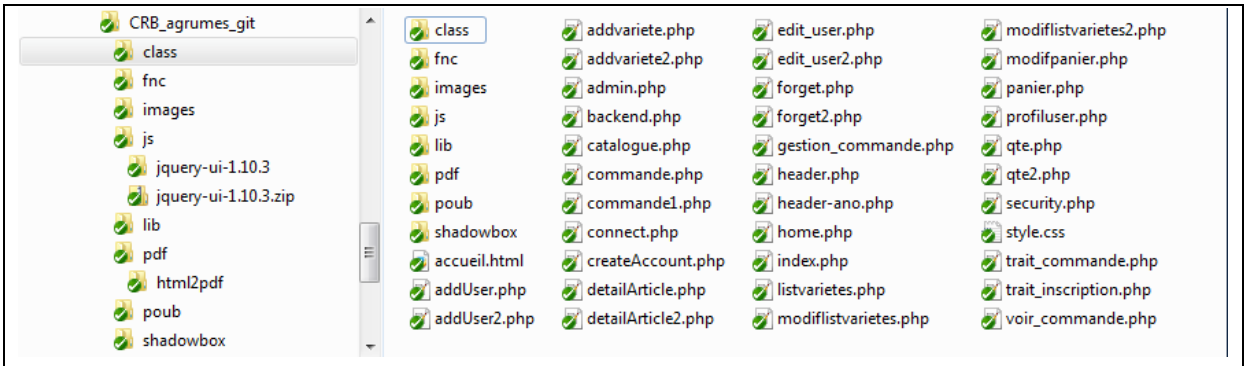


Figure 9 : l'arborescence du site web

Cette arborescence a été pensée comme un MVC, sans en être un. On a un fichier index.html qui agit en tant que pseudo-contrôleur et inclut la page à afficher, un ensemble de classes définissant les modèles, et les autres fichiers PHP qui définissent les vues. Les pages dont le nom termine par « 1.php » ou « 2.php » sont en fait les actions appelées par les formulaires des pages Shadowbox.

Les classes

Le choix de la programmation objet s'est fait autour de la réutilisabilité et de la maintenabilité du code. La réutilisabilité est entrée en ligne de compte quand il a fallu recréer l'état des variables entre les pages, ce qui est impossible en PHP procédural avec une session légère. Il a été préféré la reconstruction des variables sous forme d'objets, avec simplement le passage des identifiants dans la variable de session, entre les pages. Les pages se chargent de recréer les objets, et les objets de recréer leurs objets liés, limitant par la même occasion la réécriture de code d'une

page à l'autre. La figure 10 montre le modèle UML des classes utilisées. Ce modèle est sujet à des changements réguliers, dû au développement itératif du site web : chaque module est développé à part, et voit son lot de classes, méthodes et propriétés ajouté au fur et à mesure.

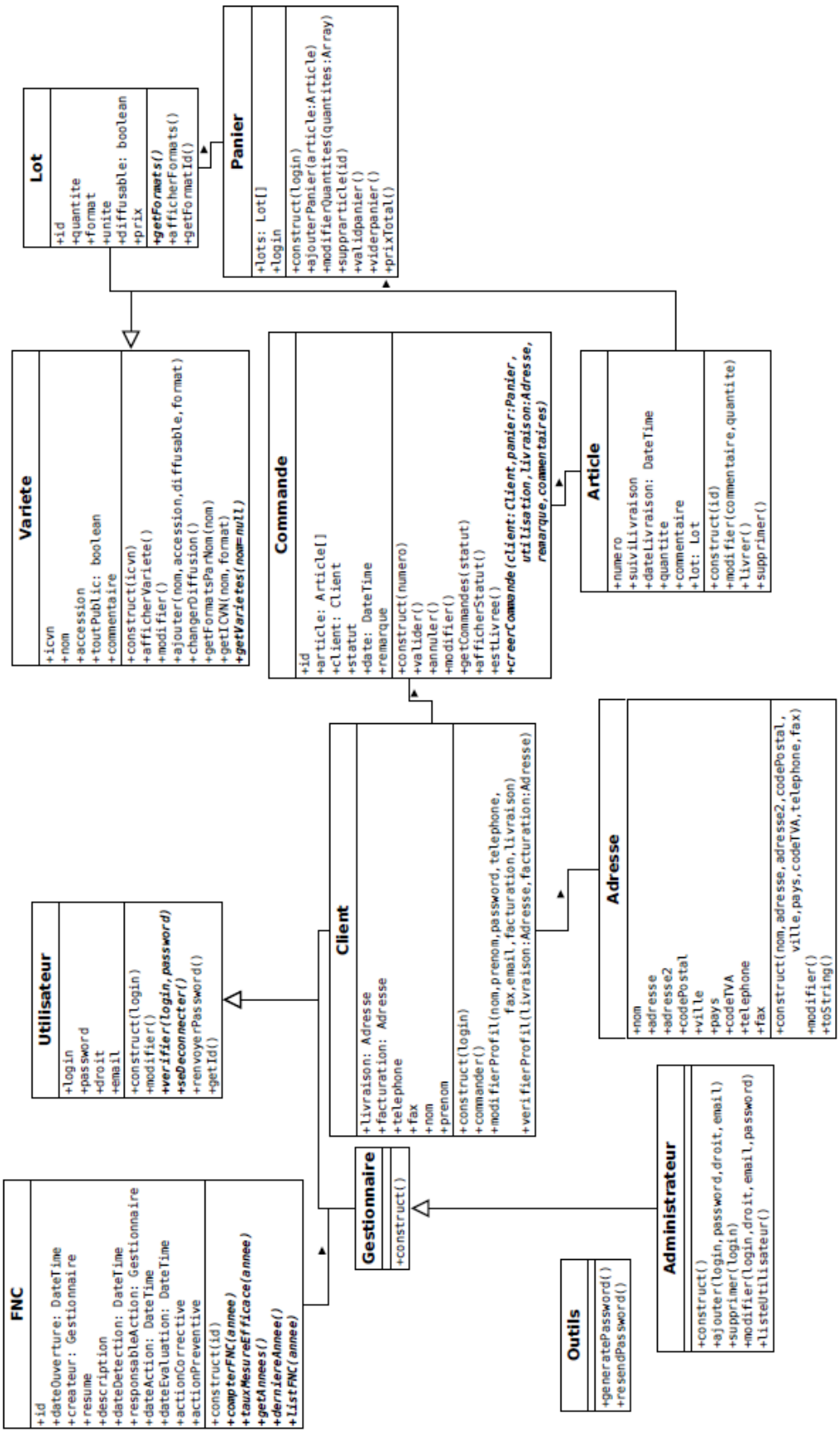


Figure 10 : diagramme de classe du site web du CRB Citrus

Les modules

J'ai développé l'architecture du site web de manière modulaire, augmentant ainsi la flexibilité à travers l'ajout de modules au fur et à mesure. Un module consiste en une fonctionnalité du site web, accessible depuis le menu principal, et se reconnait dans l'architecture de fichier sous la forme d'un sous-dossier.

Le module « Catalogue »

Le site a besoin d'un point central où afficher la collection d'agrumes, mais aussi de limiter les résultats aux accessions disponibles sous au moins un format. Le module Catalogue permet de répondre à ce besoin, et est accessible par tous les visiteurs du site. Les figures 7 et 8 montrent les pages du module « Catalogue ».

Le module « Gestion de commandes »

Le gestionnaire du CRB doit arbitrer toutes les précommandes passées, car les récoltes du CRB ne sont souvent pas suffisantes pour satisfaire toutes les demandes. Ceci impose une certaine flexibilité dans l'arbitrage : permettre d'envoyer les éléments séparément ou non, sous différents organismes d'acheminement, et de choisir une accession plutôt qu'une autre, en résumé satisfaire ou non aux requêtes des commanditaires, puis permettre de trouver un accord. Le module « Gestion de commande » a donc été conçu pour permettre d'arbitrer les précommandes de cette manière.

Le module « Gestion des variétés »

Le site est utilisable en interne comme outil de gestion du CRB. Il faut donc permettre d'ajouter des accessions. Aussi, chaque accession doit être caractérisée, et disponible dans certains formats. Le module « Gestion des variétés » permet pour l'instant de créer une accession, de donner un format à une accession, et de définir si une accession est diffusable ou non. A terme, ce module permettra la caractérisation des accessions.

Le module « Administration »

Le site web permet aux utilisateurs de s'authentifier, et les gestionnaires utilisent le système d'authentification pour gérer le CRB. Il faut donc un espace qui permet de gérer les comptes utilisateurs et gérer les comptes utilisateurs gestionnaires. Voilà à quoi répond le module « Administration ».

Le module « Récolte »

Ce module répond au besoin de consigne des récoltes à des fins de traçabilité. Il permet aussi d'indiquer au module « Extraction » si des caisses de fruits sont disponibles pour extraction.

Le module « Extraction »

Tout comme le module « Récolte », le module « Extraction » permet la traçabilité des extractions de graines des fruits récoltés. Il permet de passer à l'étape du conditionnement.

Le module « Utilisateur »

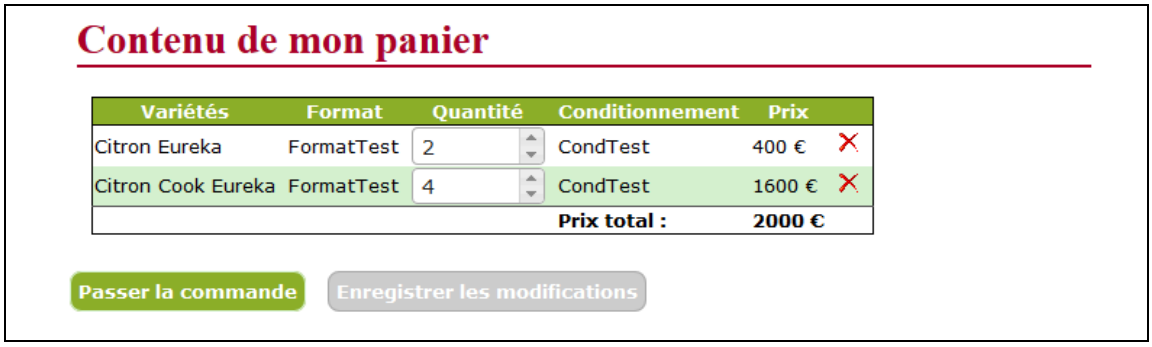
Les utilisateurs ont besoin d'un moyen de s'authentifier sur le site web, afin de profiter des modules Catalogue et Commande. Ils passent donc par ce module « Utilisateur ». Les utilisateurs gestionnaires passent aussi par ce module, mais obtiennent le privilège de l'accès aux modules dont ils sont gestionnaires. Ce module permet aussi aux Utilisateurs de devenir « Clients » quand ils passent une précommande.

Le module « Commande »

Le site doit permettre, depuis l'extérieur, de passer des précommandes de matériels végétaux, avec la même flexibilité dont a fait preuve le gestionnaire du CRB jusqu'à présent. Le module « Commande » répond à ce besoin, et n'est accessible qu'à la condition que l'utilisateur du site soit authentifié.

En plus d'avoir participé à la structuration de la base de données, j'ai développé entièrement ce module, et ce en priorité car le CRB est en phase de validation de la norme AFNOR (NF S 96-900) sur les CRB végétaux. Il se décompose comme suit :

1. Prérequis : l'utilisateur doit être identifié avant de pouvoir utiliser les fonctionnalités d'ajout au panier, et utiliser le module « Catalogue » afin de remplir son panier.
2. Une fois rempli, l'utilisateur peut visualiser son panier, voir un aperçu du prix total, et « Passer la commande ». Ici, la classe Panier récupère les informations du panier de l'Utilisateur, basé sur l'identifiant dans la base de données de l'Utilisateur. L'utilisateur peut modifier les quantités dans cette page, et obtenir un nouvel aperçu du prix total lorsqu'il enregistre les modifications.



3. Un formulaire de précommande est présenté à l'utilisateur. Il doit renseigner les adresses de facturation et livraison si ça n'a pas été fait avant. Il peut ajouter des commentaires à chaque élément du panier de la précommande. En effet, le CRB peut posséder plusieurs accessions pour une même variété, et il était possible, avant de mettre en place l'outil web, de demander une autre accession ou même de prioriser l'envoi d'une accession plutôt que d'autres. Par conséquent, cette demande peut être faite par le biais de ces commentaires. D'autres informations sont prévues pour permettre au gestionnaire du CRB de savoir comment vont être utilisées les ressources, comment envoyer les ressources, et si elles doivent être envoyées en même temps. L'envoi différé permet d'envoyer plusieurs colis pour la même commande, à différentes dates, selon les disponibilités du CRB.

4. Une fois la commande validée, elle est intégrée à une liste des commandes que le comité de pilotage du CRB va arbitrer, c'est-à-dire attribuer les ressources en fonction des disponibilités et des requêtes des demandeurs. Il existe un statut des éléments de chaque commande, permettant ainsi de déterminer si une ressource a été expédiée ou non. De plus, les commandes validées agissent aussi comme un historique, permettant un suivi efficace des commandes. La classe Commande est utilisée dans la page, puis les classes Article, Client, Panier et Lot sont utilisées dans le traitement.

Enfin, il est prévu de développer une fonctionnalité permettant à l'utilisateur ayant reçu ses ressources de déposer un commentaire de retour (« feedback ») sur l'état du colis à la réception (« bonne réception » ou bien « erreur sur l'accession demandée »…)

Le module « FNC »

Des anomalies peuvent se produire dans le CRB, comme un arbre trouvé mort dans une parcelle ou une panne sur l'extracteur de graines. Le module « FNC », pour « Fiches de Non-Conformité », répond au besoin de promptitude et d'efficacité de résolution des anomalies en permettant à n'importe quel Gestionnaire d'éditer des fiches de non-conformité.

Ce module n'est pas de taille conséquente, et je l'ai développé afin d'avoir un autre module à montrer lors d'une éventuelle présentation. Il est composé de deux pages principales :

- La première page affiche la liste des FNC pour l'année sélectionnée. Les changements de la liste et des indicateurs statistiques se fait dynamiquement par une requête AJAX. Il est possible de supprimer une fiche depuis cette page. A terme, il sera possible d'exporter et d'imprimer les fiches. La figure 9 montre la page d'entrée du module FNC.
- La seconde page (figure 11) est la page permettant de rédiger une FNC. Elle est accessible soit par le bouton « Créer », soit par un des boutons « Modifier » dans la liste des fiches.

Le module FNC fait appel aux classes FNC et Gestionnaire. Ajoutons les deux classes FNCObjet et FNCProcessus, qui permettent de lister les objets et processus relatifs aux FNC, et d'en ajouter si le Gestionnaire rédigeant la fiche considère que les options déjà présentes ne sont pas satisfaisantes pour décrire sa fiche.

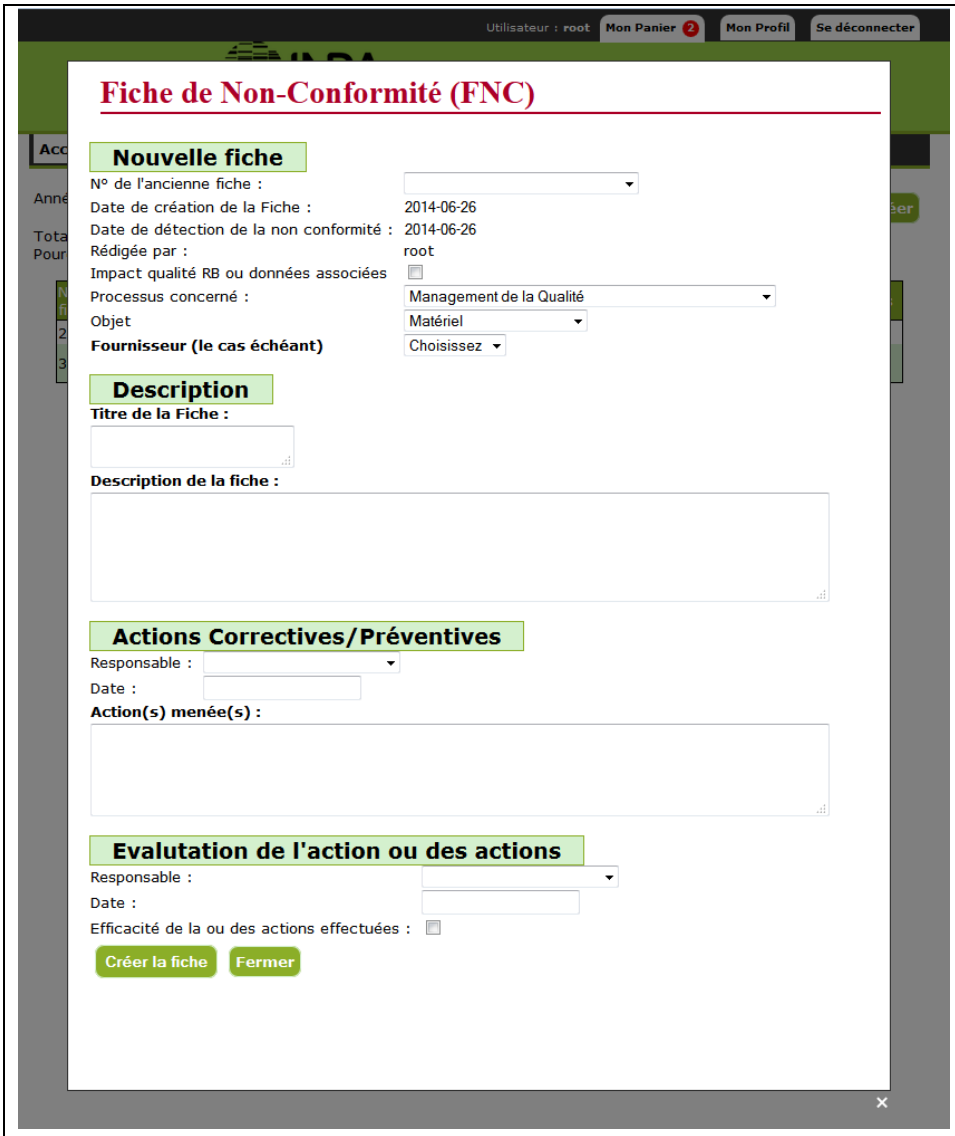


Figure 11 : création d'une FNC

Sa conception a été réalisée à partir de l'outil d'Emmanuel Bloquel : les informations à afficher ou enregistrer ont été listées, les interfaces ont été conçues sur papier avec l'équipe lors d'une réunion « codage », sous forme de brainstorming, et seulement après le module a vu le jour sous forme de code.

Perspectives

Il me reste encore deux mois à travailler à l'INRA de Corse. Le site web du CRB Citrus n'est pas encore terminé, il reste des modules soit à compléter, soit à développer, soit à concevoir :

- Le module Récolte n'est pas de taille considérable, et devrait être vite développé
- Même chose pour le module Extraction
- Le module Utilisateur permet de lister et gérer les comptes utilisateurs, mais pas encore les associations aux gestionnaires
- Le module de gestion de commandes et le module de précommande doivent être débuggés en essayant le plus de cas d'utilisations possible
- Je dois finir la phase de débuggage du module FNC, qui a encore quelques erreurs relatives à MySQL et ses clés étrangères

Les modules sont pour la plupart, conçus et développés à partir de l'outil d'Emmanue Bloquel, qui connait le besoin et a apporté une réponse. Une phase de tests avec les acteurs du CRB est prévue, mais il n'est pas indispensable de les faire immédiatement.

Ensuite viendront les documentations technique et utilisateur à rédiger.

Enfin, le réseau des CRB INRA réfléchit à un outil à adopter au niveau national et dont les besoins coïncident avec le site web développé par l'UE Citrus et moi-même. Leur choix s'était porté sur Grin Global, un outil de gestion de CRB développé par le ministère de l'agriculture américain. Après plusieurs phases de tests, il a été conclu que cet outil n'était pas adapté à la gestion de l'ensemble des CRB de l'INRA de l'INRA compte tenu de la grande variété des espèces conservées. Ceci pourrait permettre à terme, de mutualiser le site web et de le remonter en tant qu'outil de gestion de CRB au niveau national.

Conclusion

Bien que le projet de site web pour le système d'informations du centre de ressources biologiques de l'UE Citrus ne soit pas terminé, le travail que j'ai fait a apporté beaucoup dans ce projet à long terme.

Mon travail a permis à l'unité d'obtenir une base de données solide, évolutive pour les besoins futurs et fonctionnelle pour les besoins actuellement mis en œuvre autour de la collection d'agrumes, et il est possible qu'elle voit une évolution vers le niveau national, et ceci à partir de deux bases de données de gestion de collection d'agrumes peuplées, et manquant d'expertise en matière de conception de base de données en termes informatiques. Cette partie de mon travail m'a permis d'en apprendre plus sur MySQL, comme par exemple à utiliser les jointures ou laisser la base de données contrôler l'intégrité des données par l'intermédiaire des clés étrangères, allégeant ainsi le travail de développement de code de vérifications directement dans le programme. Le fait d'avoir créé une base de données d'une aussi grande taille m'a forcé à m'adapter et à utiliser d'autres outils, comme Peewee.

J'ai aussi apporté mon expertise dans la conception et le développement du site web en lui-même, afin de le rendre modulaire pour permettre l'ajout de fonctionnalités sans accroc avec les autres développeurs et les autres modules notamment grâce à Git, et à terme de l'organiser pour une meilleur maintenabilité.

Le stage n'est pas terminé, et des modules supplémentaires seront développés dans le futur, en plus de consolider l'architecture du site pour permettre aux autres développeurs de continuer à développer une fois ma durée d'embauche terminée.

Ce stage m'a permis de plus me familiariser avec le travail en équipe d'une manière moins conventionnelle : la gestion de projet s'est faite en peer-to-peer, chacun aidant là où il serait le plus utile, avec une hiérarchie « plate » entre les différents développeurs, chacun proposant ses idées à discuter et à adopter d'une décision quasiment unanime.

J'ai pu aussi proposer des améliorations notamment au niveau de la structure du site web, proposant son remaniement sous un micro-framework léger que j'avais choisi, après avoir étudié plusieurs d'entre eux. Proposition qui a été retenue mais qui n'a pas abouti par manque de temps.

Enfin, j'ai eu à apprendre beaucoup sur ce que fait l'INRA. J'ai appris comment un CRB fonctionne, ou encore les termes autour d'une collection de ressources génétiques. Sans ça, il m'aurait été impossible de définir la base de données du CRB et encore moins de développer le site web correspondant.

Lexique

Accession : Une « accession » est une identification combinant simultanément une variété, une date d'introduction et une provenance géographique. Admettons qu'une variété soit trouvée à deux endroits différents, elle aura deux accessions. Autre exemple, admettons que sur un arbre d'une certaine accession, sur une branche en particulier, les fruits murissent remarquablement plus vite que sur les autres, alors la branche est récupérée et cultivée ou greffée, et une nouvelle accession lui est attribuée.
Les accessions peuvent être accessibles depuis le site seulement si elles ont été listées comme tel dans la base de données. Il faut permettre aux gestionnaires du CRB d'éditer la liste des accessions et leur caractérisation dans la base de données, sans y toucher directement. Il faut donc des pages web permettant ceci.

CRB : Centre de Ressources Biologiques. Il s'agit de collections génétiques d'espèces biologiques.

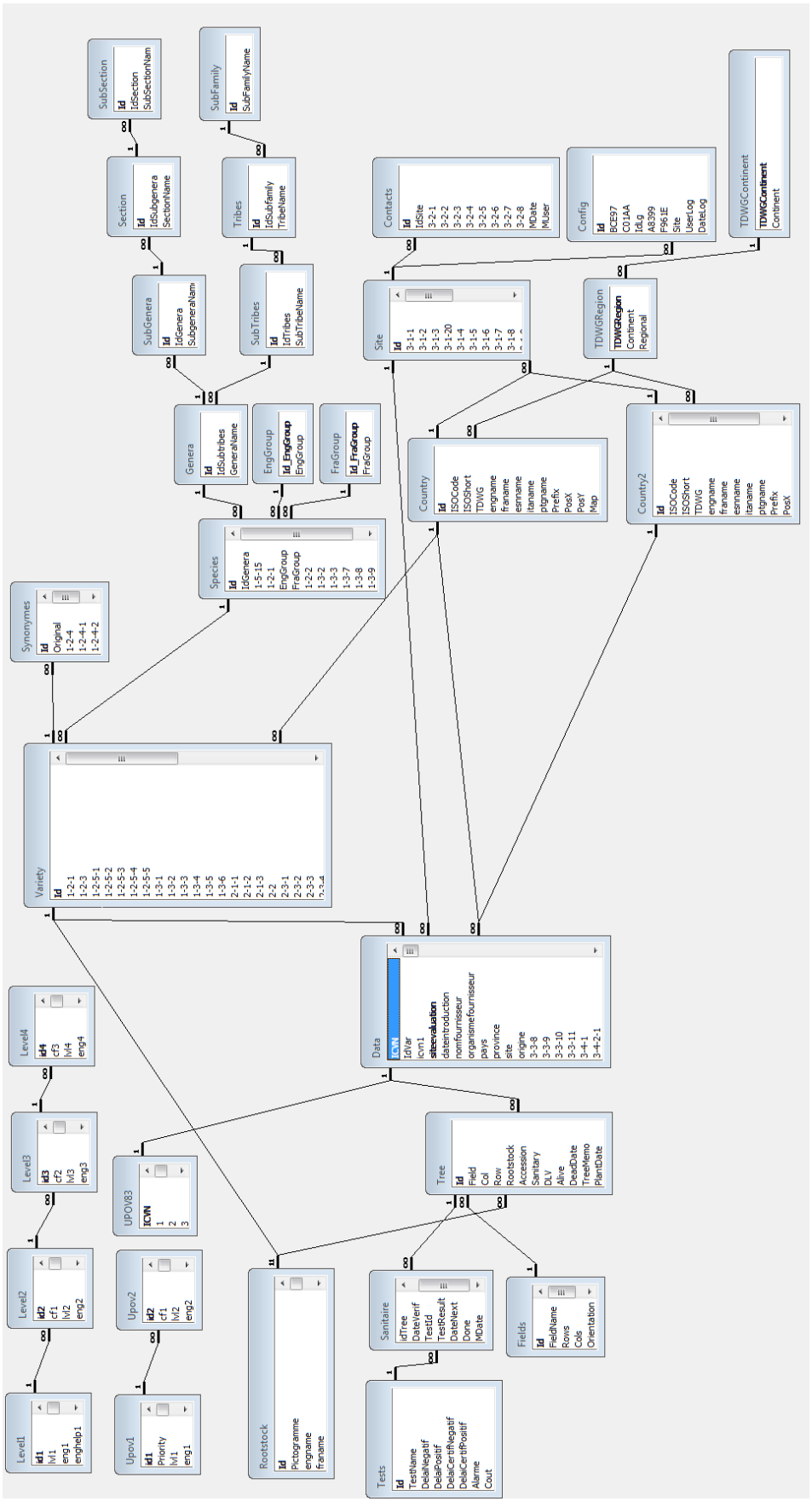
MVC : « Modèle – Vue – Contrôleur », une technique de programmation séparant les différentes couches de l'application, afin d'augmenter sa maintenabilité. Les vues sont les fichiers contenant les informations de mise en forme de l'affichage, les modèles sont les fichiers responsables d'effectuer les traitements sur les données, les contrôleurs sont les fichiers responsables d'associer les vues et les modèles afin de faire le rendu des vues avec les données traitées des modèles. Le fait de séparer les couches permet d'établir des contrats entre celles-ci, et de remplacer un élément d'une couche avec un autre tant que le contrat est respecté.

Taxonomie : science, branche de la biologie, qui a pour objet de décrire les organismes vivants et de les regrouper en entités appelées taxons afin de les identifier, puis les nommer, et enfin les classer et de les reconnaitre via des clés de détermination dichotomiques.

Annexes

Annexe 1 : Mémo git rédigé pour les autres développeurs

Annexe 2 : MCD de la base de données EGID



Annexe 3 : Format d'échange de fichiers de SIREGAL

Annexe 4 : MCD du CRB Citrus version EGID

Annexe 5 : insertionGeneriqueInterface.py

```
#!/usr/bin/env python2.7
#-*-encoding: utf-8-*-

import ast
import os
try:
    import importlib
except ImportError:
    print "This module requires importlib. It is available in Python 2.7+,
or you can '$ pip install importlib' or get it from
https://pypi.python.org/pypi/importlib/ ."
    exit(1)
from insertionGenerique import transfert

accessdb = None
accessTable = None
accessFields = {}
mysqldb = None
mysqlTable = None
mysqlFields = []
matchId = {}

def get_access_db():
    global accessdb
    while True:
        accessdb = raw_input("access db > ")
        if not os.path.isfile(accessdb):
            print "Error"
        else:
            break

def get_access_table():
    global accessTable
    accessTable = raw_input("access table > ")

def get_access_fields():
    global accessFields
    index = 0
    while True:
        field = raw_input("access field %d > "%index)
        if len(field.split(" ")) > 1:
            print "Error"
        elif field == "":
            print accessFields
            break
        else:

accessFields.update(ast.literal_eval("{"+str(index)+":'"+field+"'}"))
            index += 1

def get_mysql_db():
    global mysqldb
    while True:
        mysqldb = raw_input("mysql db > ")
        if not os.path.isdir(mysqldb):
            print "Error"
        else:
            break
```

```
def get_mysql_table():
    global mysqldb
    global mysqlTable
    while True:
        table = raw_input("mysql table > ")
        try:
            mysqlTable = importlib.import_module(mysqldb+"."+table)
            break
        except:
            try:
                mysqlTable = importlib.import_module("test."+table)
                break
            except:
                print "Table not found."

def get_mysql_fields():
    global mysqlFields
    global mysqlTable
    index = 0
    while True:
        field = raw_input("mysql field %d > "%index)
        print field
        if len(field.split(" "))> 1:
            print "Error 1"
        elif field == "":
            break
        elif not hasattr(getattr(mysqlTable,
mysqlTable.__name__.split(".")[1]), field):
            print "Error 2"
        else:
            mysqlFields.append(field)
            index += 1

def get_matches():
    global matchId
    while True:
        match = raw_input("Changes to be done (Id:shift, or
Id:{number:number,...}) NO CHECKING DONE ! > ")
        if match == "":
            print matchId
            break
        try:
            if match.split(":")[1] == "none":
                match = match.split(":")[0]+":{-1:None,0:None}"
                print "Matching",match
            matchId.update(ast.literal_eval("{"+match+"}"))
        except:
            print "Error"

def main():
    get_access_db()
    get_access_table()
    get_access_fields()
    get_mysql_db()
    get_mysql_table()
    get_mysql_fields()
    get_matches()
    print
"##############################################################################
######"
    print "TRANSFERING !"
```

```
    transfert(accessdb, accessTable, accessFields, mysqldb,
getattr(mysqlTable, mysqlTable.__name__.split(".")[1]), mysqlFields,
matchId)
    print "AND IT'S DONE !"

if __name__ == "__main__":
    main()
```

Annexe 6 : insertionGenerique.py

```
#!/usr/bin/env python2.7
#-*-encoding: utf-8-*-

import pyodbc # Used for Access connections. Uses the ODBC model.

def transfert(accessdb, accessTable, accessFields, mysqldb,
mysqlTable=None, mysqlFields=None, matchId=None):
    connectionStr1 = "DRIVER={Microsoft Access Driver (*.mdb)};
DBQ=%s"%accessdb
    connectionStr2 = "DRIVER={Microsoft Access Driver (*.mdb, *.accdb)};
DBQ=%s"%accessdb

    try:
        dbConn = pyodbc.connect(connectionStr1)
    except:
        try:
            dbConn = pyodbc.connect(connectionStr2)
        except:
            "Error db."
            exit(1)
    cur = dbConn.cursor()
    # Generate SQL from accessTable and accessFields
    errors = []
    sql = "SELECT "
    for champ in accessFields:
        sql += "`"+accessFields[champ]+"`"
        if champ != max(accessFields.keys()):
            sql += ", "
    sql += " FROM `"+accessTable+"` ORDER BY "+accessFields[0]
    # The SQL is done.
    print sql
    result = cur.execute(sql)

    # Make the results ready for importation
    for res in result:
        # Process res in order to make it match the numbers in matchId
        try:
            res[0] = int(res[0])
        except Exception, e:
            print e
        if res[0] == None:
            continue
        if res[0] == 0:
            print "%s = 0, passing..."%mysqlFields[0]
            continue
        print res[0]
        for index in xrange(len(res)):
            if matchId is not None and index in matchId:
                if hasattr(matchId[index], "keys"): # matchId[index] is a
dict
                    if res[index] in matchId[index]:
                        buffer = list(res)
                        buffer[index] = matchId[index][res[index]] # Let's
say res[1] == 2, res[1] = matchId[1][2]
                        res = tuple(buffer)
        # Now export the result to MySQL
        mysqlTable._meta.database.set_autocommit(False)
        instance = None
        in_table = 0
        try:
```

```
            instance = mysqlTable.get(getattr(mysqlTable, mysqlFields[0])
== res[0])
            in_table = 1
        except Exception, e:
            print "No row with matching %s %s. Creating
one."%(mysqlFields[0],res[0])
            id_param = {mysqlFields[0]:res[0]}
            try:
                instance = mysqlTable.create(**id_param)
                mysqlTable._meta.database.commit()
            except Exception, e:
                mysqlTable._meta.database.rollback()
                errors.append(str(res[0])+" : "+str(e)+"\nValues :
"+str(res)+"\n")
                continue
        instance = mysqlTable.get(getattr(mysqlTable, mysqlFields[0]) ==
res[0])
        for index in xrange(len(mysqlFields)):
            if index == 0:
                continue
            setattr(instance, mysqlFields[index], res[index])
        try:

            instance.save()
            mysqlTable._meta.database.commit() # TRY TO AVOID THE
AUTO_INCREMENT BUG
        except Exception, e:
            instance._meta.database.rollback()
            errors.append(str(res[0])+" : "+str(e)+"\nValues :
"+str(res)+"\n")

    if len(errors) != 0:
        print "Errors occured :"
        for e in errors:
            print e

    mysqlTable._meta.database.set_autocommit(True)
    cur.close()
    dbConn.close()
```

Annexe 7 : MCD CRB Citrus SIREGAL