Sommaire

Liste des figures

Glossaire

AGAP	Amélioration Génétique et Adaptation des Plantes Méditerranéennes et Tropicales
AIDA	Agroécologie et Intensification Durable des Cultures Annuelles
API	Application Programming Interface
CRUD	Create, Read, Update, Delete
DBA	Data Bases Administrators
EPIC	Établissement Public à Caractère Industriel et Commercial
PME	Petite et Moyenne Entreprise
UMR	Unité Mixte de Recherche
UPR	Unité Propre de Recherche
Accession	Unité génétique pour laquelle plusieurs lots de semences peuvent être disponibles
Bootstrap	Collection d'outils utile à la création du design de sites web.
CodeIgniter	Framework libre PHP qui suit la conception MVC
Dictionnaire des variables	Catalogue de métadonnées
Dispositif expérimental	Collection d'outils utile à la création du design de sites web.
Echantillon	En expérimentation agronomique au champ, terrain délimité sur lequel se déroule l'essai
Entomologie	Ensemble d'éléments extraits d'une population étudiée de manière à ce qu'il soit représentatif de celle-ci, à des fins statistiques Une branche de la zoologie dont l'objet est l'étude des insectes
Essai agronomique	Expérimentation permettant de comparer et mesurer les effets de différents phénomènes sur une population expérimentale dans des conditions données, à l'aide d'un dispositif expérimental
Miscanthus	Plante herbacée vivace de la famille des Poaceae. Originaire d'Afrique et d'Asie du sud.
Phytopathologie	Une science qui étudie les maladies des plantes

PhpPgAdmin	Outil de gestion des bases de données PostgreSQL
PgAdmin	Application web d'administration pour PostgreSQL
PgModeler	Outil de modélisation des bases de données PostgreSQL
PostgreSQL	Système libre de gestion de base de données relationnelle et objet (SGBDRO)
SCRUM	Cadre de développement logiciel agile dans lequel les équipes pluriciplinaires développent des produits de manière itérative, incrémentielle et adaptative
Sorgho	Plante herbacée de la famille des Poaceae d'origine tropicale. Il possède différentes variétés utilisées pour l'alimentation animale ou humaine ou pour fabriquer des tentures, du papier, des balais...

Chapitre I

Introduction

(Contexte, organisme d'accueil, objectifs)

I. Introduction

Afin de relever les défis à l'échelle mondiale tels que la sécurité alimentaire, la réduction des impacts environnementaux et le changement climatique. Les filières végétales développent des approches agroécologiques pour concevoir des systèmes de production performants qui mettent en valeur et préservent les services écosystémiques.

La diversité des données qui sont générées dans le cadre d'essais agronomiques induit une complexité de gestion de ces dernières (stockage, partage, diffusion). En effet, les jeux de données sont pluridisciplinaires et transdisciplinaires : génétique, agronomie, entomologie, physiologie, phytopathologie. Les jeux de données comprennent des niveaux multiples d'échelles spatiales et temporelles d'observation. Ils peuvent porter sur des variétés ou des mélanges d'espèces différentes. Ils comprennent de nombreuses observations sur la plante (rendement, indice de surface foliaire, biomasse, etc.) et des données correspondant aux conditions environnementales et aux itinéraires techniques (ITK : séries climatiques et pratiques culturales).
En autre, pour un essai Agronomique donnée, le nombre d'observations ne peut généralement pas être fixé a-priori, en effet de nouvelles observations peuvent devoir être effectuées au cours de l'expérimentation en fonction des pressions biotiques ou abiotiques par exemple. Enfin l'ensemble de ces observations sont effectuées au sein de dispositif expérimentaux qui sont très divers.

Afin d'apporter des meilleures solutions pour la gestion des flux d'informations issus de l'expérimentation agronomique, le domaine de l'informatique fait appel au développement de systèmes d'informations cohérents et agiles pour intégrer les besoins de l'entreprise en l'occurrence CIRAD (Centre de coopération internationale en recherche agronomique pour le développement) . Un système d'information est un ensemble d'actions coordonnées de recherche, de traitements, de distributions et de protection des informations utiles. Il a pour objectif la collecte des informations et leur diffusion, l'optimisation des coûts de collecte et de traitement des informations, la mise à jour des bases de données et le partage de données entre plusieurs acteurs.

I.1. Présentation de l'organisme d'accueil

Le CIRAD Centre de coopération internationale en recherche agronomique pour le développement a été créé en 1984, sur la base des instituts de recherche agricole coloniaux. C'est un EPIC, sous la double tutelle du ministère de la recherche et du ministère des affaires étrangères. Il emploie 1650 personnes, dont 800 chercheurs, avec un budget annuel de 200 millions d'euros en 2015.

Le CIRAD comprend 3 départements scientifiques, Performance des systèmes de production et de transformation tropicaux (Persyst), Systèmes biologiques (Bios), Environnement et société (ES), ainsi que 33 unités de recherches et 13 directions régionales à travers le monde (Figure1).

Durant mon stage j'ai été accueillie par l'unité de recherche AGAP, dans l'équipe ID (Intégration de Données), du département Bios et l'unité AIDA du département Persyst. Mon encadrement était composé : d'une informaticienne de l'équipe AIDA, d'une statisticienne de l'équipe Plasticité et Adaptation des Monocotylédones et d'un un généticien de l'équipe Dynamique de la diversité, Environnements et Sociétés, coordinateur d'un des projets nécessitant un système d'information permettant de gérer les données issues d'essais agronomiques.

UMR AGAP

Créée en janvier 2011, cette unité mixte de recherche placée sous la tutelle de trois organismes (Cirad, Inra et Montpellier SupAgro) a pour objectif l'amélioration génétique des plantes tropicales et méditerranéennes.

L'équipe ID développe en interactions fortes avec les biologistes des autres équipes de l'UMR Agap et d'autres UMR partenaires des outils de bio-informatique visant à répondre aux problématiques de la génétique et de la génomique des plantes tropicales et méditerranéennes.

L'équipe PAM (Plasticité Phénotypique et Adaptation des Monocotylédones) analyse les interactions G (génotype) x E (environnement) x C (conduite) et développe des connaissances et des modèles intégrés des processus morphophysiologiques d'adaptation des plantes au milieu agro-écologique.

UPR AIDA

Créée le 1er janvier 2014, cette unité est issue de la fusion des unités Systèmes de Culture Annuels (SCA) et Systèmes et Ingénierie Agronomique (Sia). Elle est structurée en cinq équipes de recherche animées pour mettre en œuvre des démarches interdisciplinaires au service de l'intensification écologique. L'étude, la conception et la proposition de systèmes de culture annuels (riz, blé, sorgho, canne à sucre, cotonnier...), répondant aux exigences de performances

agronomiques et environnementales, caractérisent aujourd'hui la demande sociétale et les besoins du développement.

Avec ses partenaires dans plus de 100 pays, le CIRAD mène des activités autour de six axes scientifiques :

- Agriculture écologiquement intensive ;
- Alimentation durable ;
- Action publique pour le développement ;
- Valorisation de la biomasse ;
- Santé des animaux et des plantes ;
- Sociétés, nature et territoires.

Son objectif principal est celui de développer une agriculture durable adaptée aux changements climatiques au service des êtres humains tout en préservant l'environnement. Avec un objectif premier de développement de l'agriculture dans les pays en voie de développement.

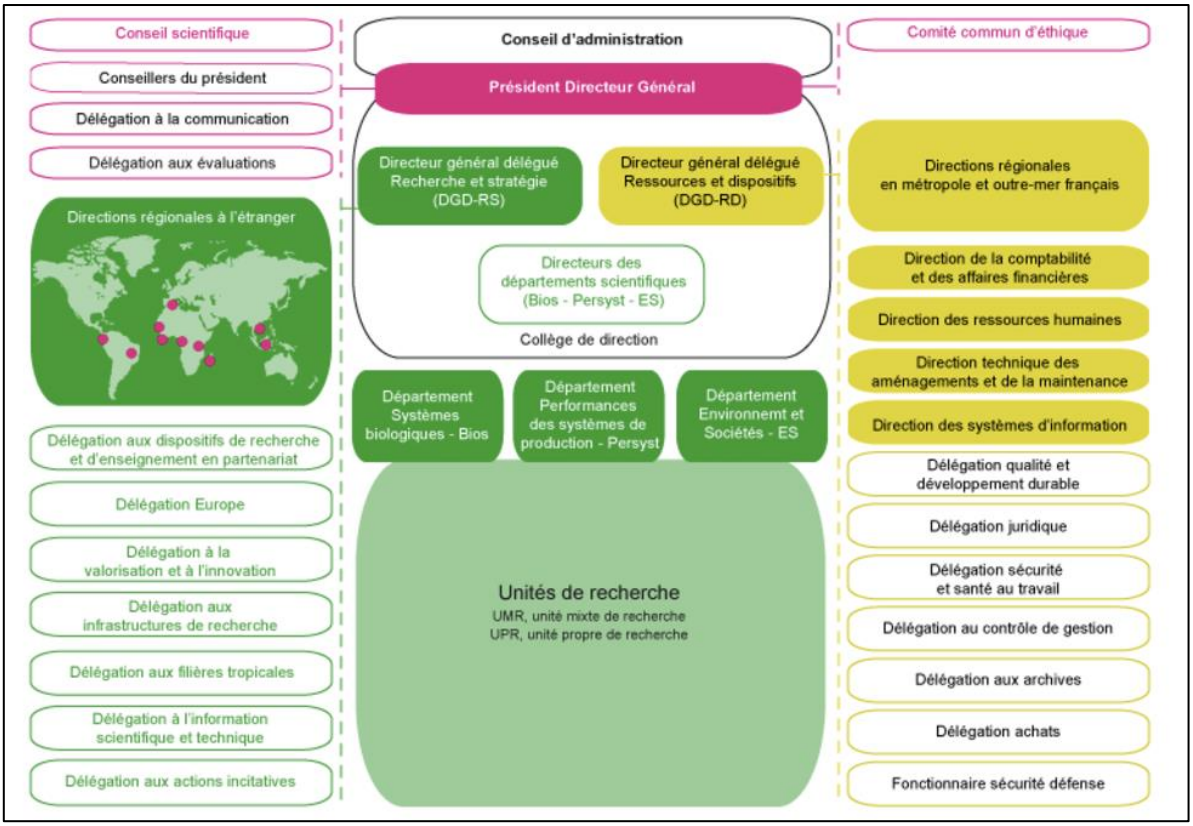


Figure 1: Organigramme du CIRAD (source : www.cirad.fr)

I.2. Contexte du stage

Le projet Biomass For the Future (BFF) (http://www.biomassforthefuture.org/) est une bonne étude de cas pour illustrer les besoins de développement d'un système d'information adapté à la gestion de données issues d'essais agronomiques. Il s'agit d'un projet impliquant 22 partenaires publics et privés, dont 9 laboratoires publics, 1 institut technique, 10 PME ainsi que 2 Communautés d'Agglomération, visant à développer de nouvelles variétés et des systèmes de cultures de deux espèces, le miscanthus et le sorgho, au nord et au sud de la France. Dans ce cadre, chaque année, plusieurs essais agronomiques sur diverses variétés de sorgho et de miscanthus sont conduits sur différents sites expérimentaux. Chaque essai est caractérisé par une question scientifique, un dispositif expérimental associé, des variétés cultivées, des pratiques culturales et des échelles d'observation spatiales et temporelles qui lui sont propres. Le projet ayant débuté en 2012 et s'achevant en 2020, une quantité énorme de données a été et sera encore collectée. Il s'agit de données se rattachant à différents domaines scientifiques tels que l'agronomie, la génétique, l'écophysiologie, l'histologie ou encore la biochimie.

Ces jeux de données de natures hétérogènes gérés d'une manière individuelle sont souvent stockés dans des fichiers éparpillés de divers formats (Excel, Word, CSV…) ou sur différents supports (ordinateurs, clés USB…) ce qui rend difficile leur exploitation et leur partage et peut résulter en perte d'information.

Malgré l'existence des bases de données qui permettent de stocker toutes les données d'une manière harmonisée et sécurisée, d'autres limites liées à l'exportation ou à l'importation des données via cet outil. Les chercheurs ne maîtrisent pas forcément l'utilisation des bases de données, d'où la nécessité de développer des interfaces web faciles à exploiter et qui s'adressent à un large public.

Une veille sur les différents systèmes d'information existants (EPHESYS, Agrobase, PhenomeNetworks, Breeding Magament Systems) a été faite par les responsables du projet mais aucun ne répondait exactement aux besoins spécifiques du projet BFF.

- EPHESIS (https://urgi.versailles.inra.fr/) est un système d'information de URGI (unité de recherche en génomique et bio-informatique) de l'INRA. Ce système permet l'intégration des essais expérimentaux et des données environnementales (interaction génotype-environnement), un dictionnaire de variables liées aux autres ontologies agronomiques est également intégré à ce système.

Cependant au moment de notre analyse portée sur la recherche des outils existant, ce système à montrer ses limites. Premièrement d'un point de vue technique, ce système était en cours de

développement (version instable) donc pas de version exportable (ce qui n'est pas le cas à l'heure actuelle). Deuxièmement d'un point de vue fonctionnel, ce système ne permet le suivi des interventions qu'à l'échelle d'une unité expérimentale (il ne permet pas une gestion simple des caractérisations effectuées à des niveaux intra-parcelles (groupes de plantes, plantes individuelle, organes, tissus, groupes cellulaires).

- Le Breeding Management System(BMS) de l'Integrated Breeding Platform l'IBP est un ensemble d'outils logiciels qui intègre une base de données permettant la caractérisation phénotypique et l'évaluation de matériel végétal. Un dictionnaire de variables incluant des ontologies de 10 espèces dont le sorgho est également disponible. En revanche le dictionnaire de variables utilisées lors du projet BFF n'a pas été fait en utilisant des ontologies existantes.

D'autres systèmes d'information proposent également des fonctionnalités intéressantes (Agrobase, Phenome Networks), néanmoins ces systèmes sont protégés par leurs propriétaires contre toutes modifications possibles et des évolutions menées en partenariat sont difficilement envisageable.

- ECOFI® une base de données développée par le CIRAD afin de gérer des essais agronomiques. Ce système gère parfaitement le stockage et la mise à jour des données phénotypiques de plusieurs espèces issues des domaines différents (agronomie, écophysiologie…) sans altération de la structure de la base grâce à l'utilisation d'un modèle de données génériques. En dépit de cela, le système présente quelques inconvénients à savoir son incapacité à gérer les échantillons, à gérer plusieurs utilisateurs voire plusieurs partenaires.

Au final, l'équipe du projet BFF a choisi de développer un nouveau système de gestion les données phénotypiques et plus largement des dispositifs expérimentaux sur la base de l'outil ECOFI®. Ce dernier doit permettre d'intégrer les données phénotypiques et expérimentales, avec comme perspective d'élargir son utilisation à d'autres projets et espèces.

En 2015 a donc débuté le développement système d'information appelé DAPHNE (Database PHenotype plaNt intEgration) constitué d'une version plus adaptée de la base de données générique ECOFI® et d'une interface utilisateur. La figure2 illustre la page d'accueil de l'interface.

Ce système d'information doit intégrer et gérer :

- La description des dispositifs expérimentaux
- La gestion des observations à toutes les échelles de l'unité expérimentale (blocs, parcelles, places, placettes, etc.)
- La gestion des échantillons

- La gestion des analyses
- La gestion des itinéraires techniques
- La gestion des équipements et produits utilisés dans le cadre des itinéraires techniques
- La gestion des données météorologiques
- La gestion d'un dictionnaire des variables (intégrant des notions d'ontologies)

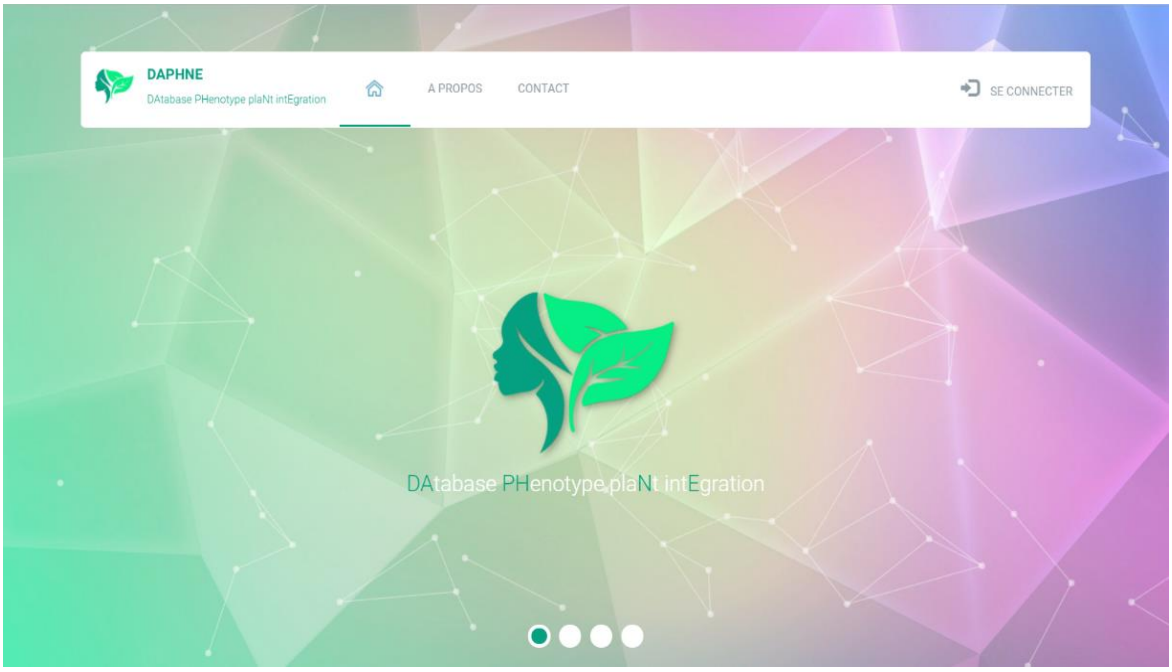


Figure 2: La page d'accueil « DAPHNE »

Les dispositifs expérimentaux sont un ensemble de parcelles ou d'unités expérimentales. Sur chacune de ces unités expérimentales s'applique un ou plusieurs traitements (figure 3). Chaque unité expérimentale est traitée individuellement, elle fait l'objet d'au moins une observation et des examens ou analyses.

A_n, B_n, C_n : parcelles

M_n : modalités

Bloc : répétition

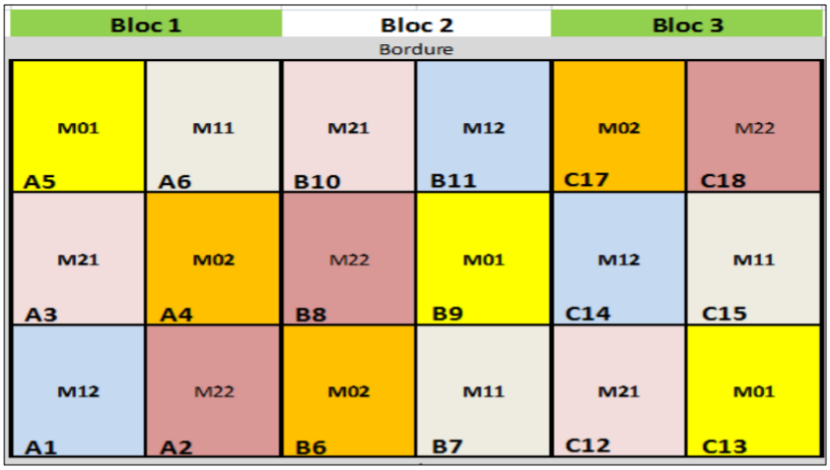


Figure 3: Représentation d'un dispositif expérimental

Une fois le site d'implantation et le type de dispositif choisis, on met en place l'essai.

Le suivi de ces essais nécessite l'acquisition de données hétérogènes :

- Les pratiques culturales associées (fertilisation, travail du sol, irrigation, etc.)
- Les conditions pédo-climatiques (type des sols, la profondeur, température, rayonnement, vent, précipitations, humidité, etc.).
- Les caractéristiques phénotypiques des plantes (la hauteur, diamètre, nombre d'entre nœud, nombre de feuilles…) sont décrites dans un dictionnaire des variables
- Les échantillons récoltés à toutes les échelles de l'unité expérimentale et à différents stades de développement de la plante
- Les résultats issus des analyses de laboratoires des échantillons prélevés.

A mon arrivée au CIRAD, après avoir discuté avec l'équipe de travail de DAPHNE, j'ai constaté que le développement de certains modules avait était achevé à savoir :

- La gestion des utilisateurs (inscription et authentification) et accès aux jeux de données,
- La création d'un projet par un utilisateur,
- Le système qui gère l'envoi des demandes d'ajout à un projet,
- Le système de réception des notifications et des mails.

L'équipe de travail de DAPHNE est composée :

- D'une informaticienne spécialiste des bases de données,
- Une biostatisticienne impliquée dans la définition et l'analyse des dispositifs expérimentaux,
- Un généticien qui pilote le projet, il a le rôle de « client » et
- Moi-même en charge de la poursuite du développement de l'interface web.

I.3. Objectifs de stage

L'objectif global consiste à poursuivre le développement du système d'information DAPHNE capable de gérer des données issues de la gestion des essais agronomiques. Afin de mieux répondre à ce besoin j'ai tracé plusieurs objectifs :

1. Etudier l'existant :
   - Réviser le du schéma de la base de données ;
   - Tester les fonctionnalités existantes de l'application DAPHNE.
2. Finaliser la gestion des confidentialités / droits d'accès

3. Contribuer au développement de l'interface d'importation des données :
    a. Projet,
    b. Essai,
    c. Dispositif,
    d. Observations sur le terrain,
    e. Récolte des échantillons,
    f. Observation sur les échantillons.

4. Contribuer au développement d'une interface pour le dictionnaire de variables
5. Basculement de l'application dans le serveur du Cirad et réalisation des tests.

Chapitre II

Problématique, Méthodologie, Outils

II. Problématique

Lors du développement d'applications informatiques destinées aux système d'information qui gère des flux d'informations, on est amené à répondre à un ensemble de questions. À savoir

- Comment rassembler et stocker les données d'une manière homogène et harmonisée, afin d'éviter les redondances et la perte d'information ?
- Comment sécuriser l'accès et le partage de ces jeux de données ?
- Comment faciliter leur exploitation pour une meilleure capitalisation en termes de temps et de qualité ?

Dans le cas de la poursuite du développement du système d'information DAPHNE j'ai relevées les problèmes suivants :

1) Après l'identification des fonctionnalités existantes dans l'application, j'ai constaté que la majorité des formulaires développés gèrent seulement la saisie ou l'entrée des données.

Solution proposée

Développer des formulaires permettant la mise à jour et la suppression des données avec des restrictions sur les droits d'accès en fonction du statut de l'utilisateur.

2) Par la suite et lors des réunions hebdomadaires organisées avec l'équipe de travail de DAPHNE, des changements sur la base de données ont été effectués. Ces changements ont affecté certaines fonctionnalités développées précédemment.

Ces changements concernent :

- La gestion des jeux de données et leur partagent avec d'autres utilisateurs
- La gestion d'un essai au sien des projets.

Au début de mon stage pour partager un jeu de données, on procédait à la création d'un groupe dans lequel on associait tous les utilisateurs avaient les droits en écriture sur ce jeu de données. Mais cette notion de partage de données par groupes d'utilisateurs a été supprimée et remplacée par une gestion individuelle d'un jeu de données. A présent, chaque utilisateur crée son jeu de données et s'il souhaite le partager, il ajoute des membres avec des accès différents, soit en lecture seulement soit en lecture et écriture.

Le besoin défini au départ concernant l'association d'un essai agronomique à un projet mentionne que dans chaque projet un essai est créée. Cependant la mise en place d'un essai

coûte très cher, donc l'idéal est de pouvoir exploiter le même essai dans plusieurs projets. À noter aussi que dans un seul projet on peut trouver plusieurs essais. Une nouvelle table a été créée dans la base de données pour exprimer cette nouvelle relation « essais-projet » et stocker les données s'y rapportant.

Solution proposée

Refaire les développements concernant la saisie d'un jeu de données et son partage ainsi que la création d'un essai et son association aux projets afin d'adapter l'application aux changements effectués.

3) Face à la grande quantité des données collectées depuis le début des travaux dans le cadre du projet BFF (voir annexe 3) et les données à collecter dans le futur par d'autres projets. L'entrée des données par une simple interface de saisie devient très compliquée et peu efficace.

Solution proposée

Développement d'une interface d'import massif des données via des fichiers excel, s'ajoutant aux formulaires de saisie en ligne.

1) Les observations relevées lors des suivis effectués sur des essais agronomiques, implantés dans des champs, concernent les plantes cultivées, le sol, le climat et les pratiques culturales. Pour stocker les variables permettant de définir précisément les observations effectuées, les chercheurs n'emploient pas forcément les mêmes termes (synonymes, polysèmes) ce qui rend difficile l'exploitation des données. Dans le but d'harmoniser l'entrée des données dans DAPHNE la déclaration des variables observées est basée sur des ontologies existences (Plant ontology, crop ontology).

Solution proposée

Développement d'une interface d'importation des variables (dictionnaires de variable) sous forme d'un arbre (treeview). Cet arbre permet de visualiser les variables déjà déclarées dans la base et donnera la possibilité d'en rentrer d'autres. Il permettra aussi en sélectionnant les variables et en les mettant dans un panier de constituer le fichier d'import automatiquement.

III. Les technologies utilisées

Avant la phase de l'exploitation de l'existant (code ou base de données). La configuration et l'installation des outils sur la machine, est une étape à effectuer.

III.1. Le Framework CodeIgniter

La décision du choix d'utilisation de ce framework, revient à l'équipe de développement de DAPHNE en 2016. Par conséquent, il était nécessaire de continuer le développement de l'interface en utilisant la même technologie.

CodeIgniter un framework PHP qui s'exécute côté serveur. Son architecture est basée sur le patron de conception MVC (Modèle –Vue – Contrôleur). Ce dernier, permet de séparer le code d'accès aux données de celui servant à l'affichage et de celui destiné à gérer les requêtes des utilisateurs, ainsi que les interactions entre l'affichage et les données. Cette figure nous illustre les interactions qui existent entre les différents éléments du patron MVC

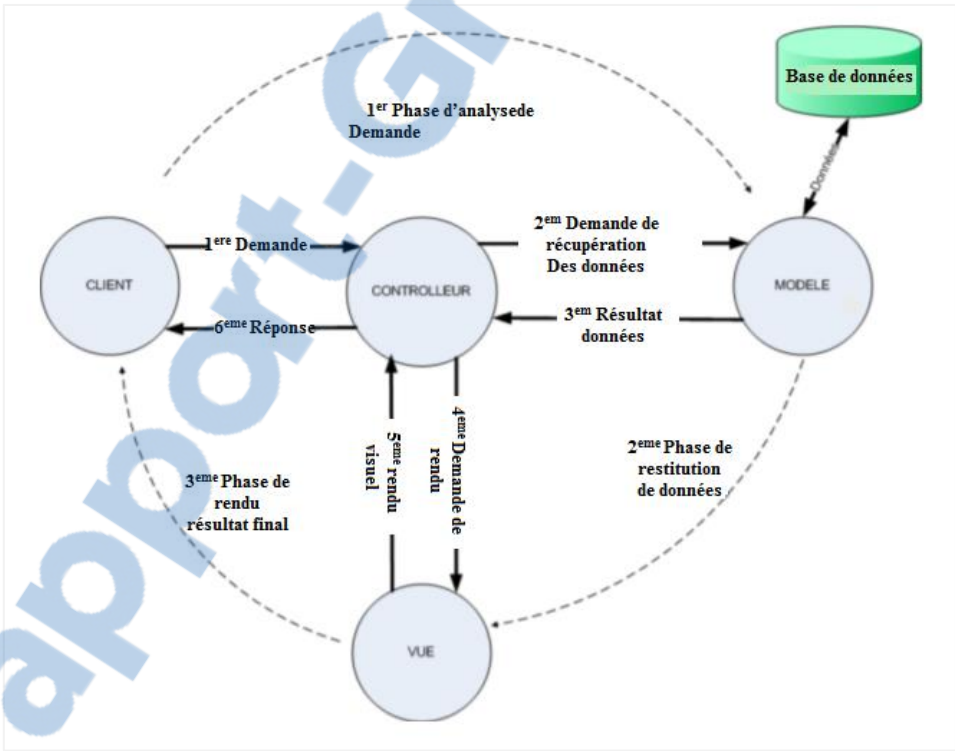


Figure 4 : Architecture globale d'une application

- *Le contrôleur* : représente l'organe de contrôle du système. Il reçoit les requêtes du client (utilisateur), permet de charger les modèles nécessaires ainsi que les vues adéquates et retourne le résultat.

- *Le Modèle* : représente l'ensemble des données stockées dans une base de donnée. C'est un organe(classe) qui sert à effectuer les actions de création, de lecture, de mise à jour et de suppression des données.
- *La vue* : représente l'organe qui produit la présentation des résultats en fonction de données qui lui sont fournies.

Dans la même logique de fonctionnement MVC, CodeIgniter comporte sa propre architecture, illustrée dans la figure.

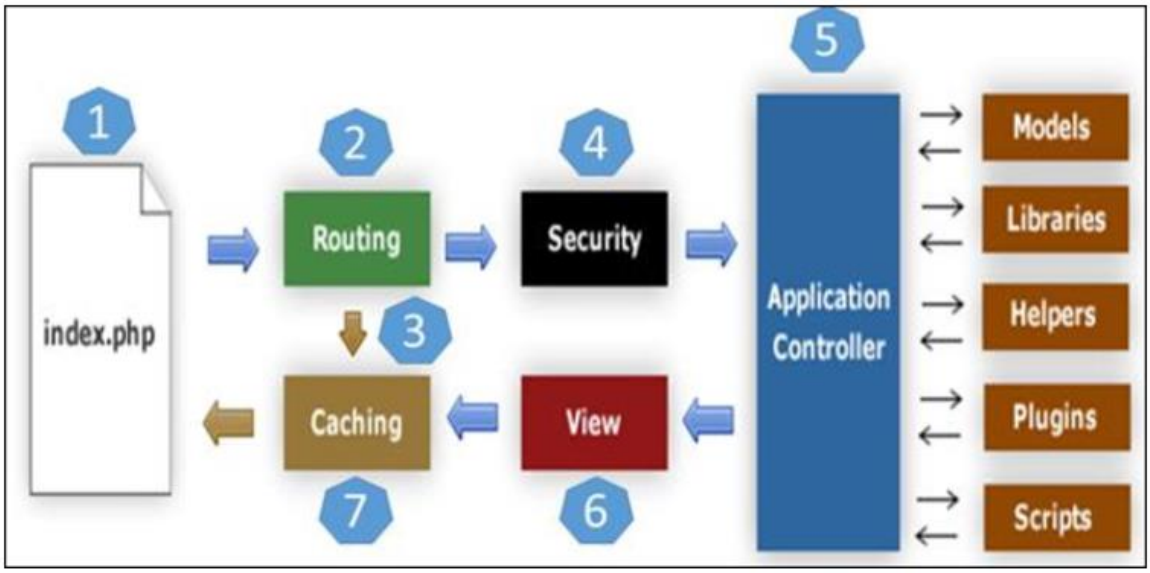


Figure 5: Architecture de CodeIgniter

- *La page index.php* : il s'agit du fichier de vue appelée par défaut à chaque fois que l'internaute se connectera à un site. Dans ce cas, la page affichée sera la page initiale de codeIgniter.
- *Les routes, cache et sécurité* : *au cours de* l'étape «2 » le routing décidera de passer la demande de la page à l'étape 3 pour la mise en cache ou à l'étape 4 pour la vérification de sécurité.

Si la page demandée existe déjà dans le cache alors le routing passera la requête à l'étape 3 et la réponse retournera à l'utilisateur, dans le cas contraire passera la page demandée à l'étape 4 pour les contrôles de sécurité.

- *Le Contrôleur* : après la vérification de sécurité, le contrôleur charge les modèles, les bibliothèques, les helpers, les plugins et les scripts nécessaires et transmet tout à la vue.

- *La Vue* : rendra la page avec les données disponibles. Si la page demandée n'a pas été mise en cache avant, cette fois elle sera mise en cache, afin de traiter rapidement les demandes futures.

❖ L'arborescence de CodeIgniter

La figure 5 montre l'arborescence des dossiers du framework CodeIgniter.

La structure du répertoire CodeIgniter est divisée en 3 dossiers :

a. Application

Ce dossier contient tout le code de l'application web, lui-même contient plusieurs sous dossiers :

▪ **config** : contient la configuration de notre site.

▪ **controllers** : la liste de nos contrôleurs.

▪ **models** : la liste de nos modèles.

▪ **views** : la liste de nos vues.

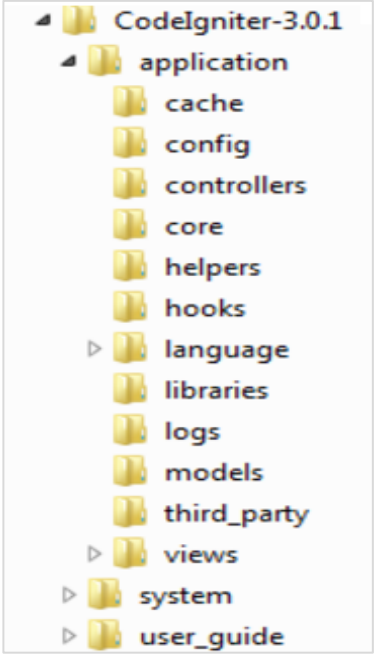


Figure 6: Arborescence des répertoires de CodeIgniter

▪ **helpers** : la liste de nos helpers créée pour une application.

▪ **libraries** : la liste de nos bibliothèques créée pour une application.

▪ **Third_party** : Dans ce dossier, on peut placer n'importe quel plugin, qui sera utilisé pour notre application.

✓ Le dossier « **application/config** »

1. Le fichier « **config.php** » contient la configuration globale de notre application CodeIgniter.

```
<?php
defined ('BASEPATH') OR exit ('No direct script access allowed');
//(…)
$config['base_url'] = 'http://localhost/daphne.cirad.fr/';
$config ['index_page'] = ' index.php ';
$config['language'] = 'french';
```

La variable ['base_url'] permet de spécifier l'url de base de notre application. La variable ['language'] permet de spécifier la langue de notre application car par défaut, la langue de notre application est l'anglais.

2. Le fichier « **database.php** » contient les éléments de configuration de votre base de données.

```
<?php
defined ('BASEPATH') OR exit ('No direct script access allowed');
//(…)
$db['default']['hostname'] = 'localhost';          // Nom d'hôte
$db['default']['username'] = 'root';               // Nom d'utilisateur
$db['default']['password'] = ' ';                  // Mot de passe
$db['default']['database'] = 'daphne_dev';     // Nom de base de données
$db['default'][' dbdriver '] = 'postgre';        // type de base de données
```

Les 4 premières lignes nous renseignent sur la base de données utilisées dans notre application.

3. Le fichier « **autoload.php** » permet de charger automatiquement certains éléments sur toutes les pages de l'application.

```
<?php
defined ('BASEPATH') OR exit ('No direct script access allowed');
//(…)
$autoload['libraries'] = array('database', 'session', 'email');     // Les librairies
$autoload['helper'] = array('url', 'assets', 'html', 'security');     // Les helpers
$autoload['config'] = array();                  // Les fichiers de configuration
$autoload['language'] = array();                // Les fichiers de langue
$autoload['model'] = array();                  // Les modèles
```

✓ Le dossier « **application/contrôllers** »

Dans le dossier « controllers » on trouve tous les dossiers de l'application qui chargent les librairies, les vues, les modèles et les helpers. On trouve aussi les règles de validation des formulaires. Dans CodeIgniter un controller est une classe qui en hérite par default de la classe CI_Controler.

Le bloc de code ci-dessous montre un exemple d'un contrôleur de notre application, « Welcome.php » qui charge la page d'accueil de notre interface web.

```php
<?php
defined('BASEPATH') OR exit('No direct script access allowed');
class Welcome extends MY_Controller {

    public function __construct() {
        parent::__construct();
        $this->load->model(array('users_model'));
        $this->load->library('form_validation');
    }
    public function index() {
        $this->view();
    }
    public function about() {
        $this->view('welcome/about', 'A Propos...');
    }
    public function contact() {
        $this->view('welcome/contact', 'Nous Contacter');
    }
```

Le contrôleur est la classe qui est appelée en premier lorsque l'utilisateur émet une requête http. Toutes les méthodes publiques définies dans le contrôleur sont autant de pages accessibles via l'URL. En fait, CodeIgniter utilise un système de routes très simple. Le contrôleur est directement renseigné dans l'url (voir figure 7).

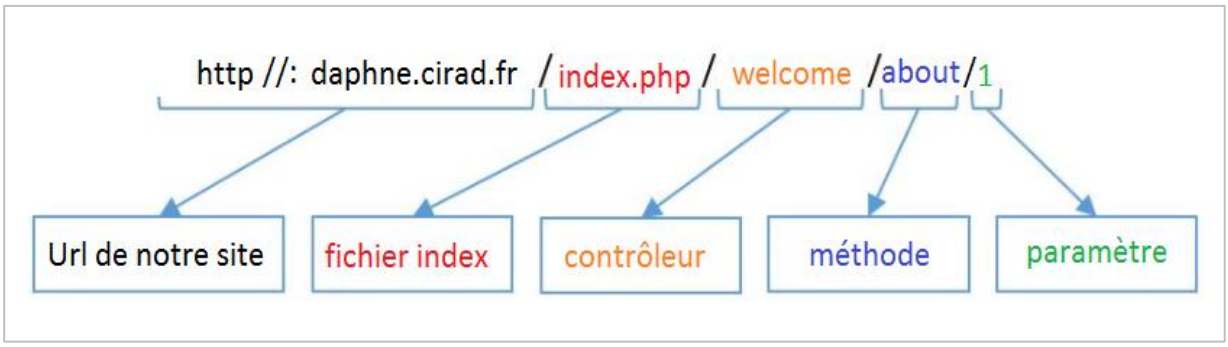


Figure 7: Exemple d'une url avec CodeIgniter

Le slash permet de séparer les urls en plusieurs segments à savoir le domaine du site, l'index, le contrôleur appelé, la méthode appelée dans le contrôleur welcome et le paramètre que recevra la méthode.

✓ Le dossier « **application/views** »

Une vue est tout simplement un fichier php qui contient le code html de notre page, responsable de tout affichage à l'écran. Afin de faciliter le développement et gérer le design de l'application, on a utilisé le Framework Bootstrap.

L'affichage d'une vue est assez simple, l'appel de cette ligne suffit dans le contrôleur.

```
$this ->load-> view ('welcome/about');
```

✓ Le dossier « **application/models** »

Un modèle est une classe représentant une partie des données d'une application, il fait le lien entre l'application web et la base de données. Les modèles héritent de la classe MY_Modele qui propose des fonctionnalités de type CRUD (Create, Read, Update, Delete) facilitant l'exploitation de la base de données.

b. Système

Ce dossier contient le code de base ou le cœur de CodeIgniter. Le contenu de ce dossier ne doit jamais être modifié. Il est constitué de différents dossiers à charger et à utiliser dans le développement de l'applications Web :

- **Core :** contient la classe principale de CodeIgniter.
- **Database :** contient des pilotes de bases de données.
- **Fonts :** contient des informations liées à la police.
- **Helpers :** contient des helpers standard de CodeIgniter (date, cookie, URL, etc.).
- **Libraries :** contient des bibliothèques standard de CodeIgniter (courrier électronique, les calendriers, les téléchargements de fichiers, etc.).
- **Language :** contient des fichiers de langue.

NB : Pour plus de détails sur les helpers (utilitaires) et les libraries (bibliothèques) de CodeIgniter, consulter les Tableaux de synthèse représentés dans l'annexe1 et 2.

c. User guide

La version hors ligne du guide de l'utilisateur, est la même documentation que celle disponible en ligne. C'est un dossier qui peut être supprimé.

III.2. PostgreSQL

PostgreSQL est un système de gestion de base données relationnelle et objet (SGBDRO). Grâce à sa licence libre, PostgreSQL peut être utilisé, modifié et distribué librement.

Malgré l'existence des versions plus récentes, de PostgreSQL, j'ai opté pour l'installation de la version 9.3 car c'est celle qui est installée sur le serveur de la base de données de la plateforme bio-Informatique du Cirad.

Lors de l'installation de postgres, on choisit un login et un mot de passe permettant d'accéder à la future base de données.

- **pgAdminIII**

Pour gérer des bases de données PostgreSQL, on lance pgAdmin depuis le menu "Démarrer" de notre système. Se connecter grâce au mot de passe préalablement choisi et créer une base de données. Pour établir la connexion aux serveurs disponibles un double-clic permet, si les serveurs fonctionnent correctement, de consulter la liste des bases de données et les tables existantes. Comme illustrée dans la figure8.

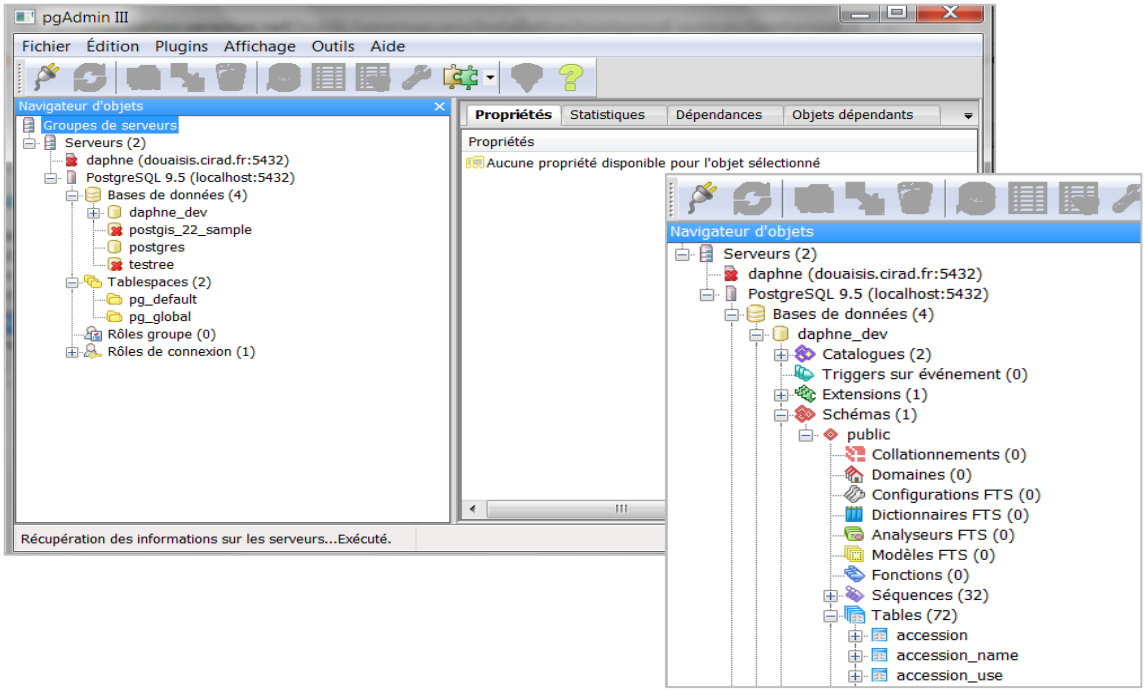


Figure 8: Exemple de connexion à la base de données(daphne_dev) de l'application

- **Installation de PhpPgAdmin**

PhpPgAdmin est un outil d'administration basé sur le Web pour PostgreSQL (figure 9). C'est un outil parfait pour les DBA (Database Administrators).

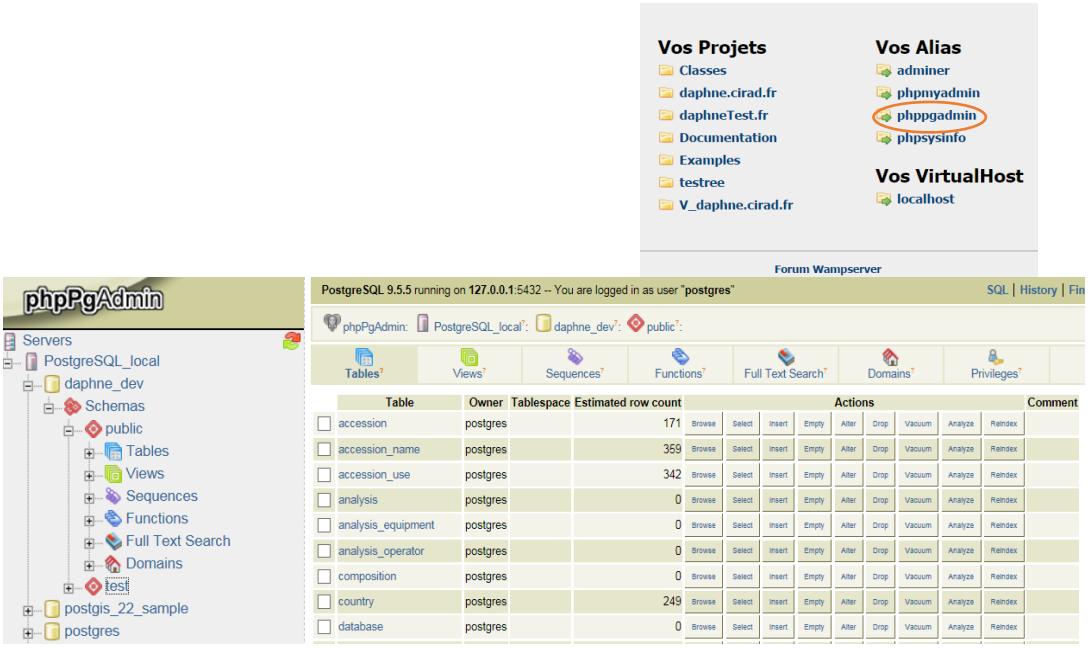


Figure 9: L'interface PhpPgAdmin

Son installation sur le WampServeur nécessite quelques configurations. Après récupération du fichier d'installation on extrait l'archive téléchargée dans "C:\wamp\apps" à côté du répertoire phpMyAdmin. Pour le configurer, il faut éditer le fichier "C:\wamp\apps\phpPgAdmin\conf\config.inc.php".

Voici les lignes à modifier dans le bloc de code suivant :

```
/** Central phpPgAdmin configuration.  As a user you may modify the ...*/

// An example server.  Create as many of these as you wish,
// indexed from zero upwards.

// Display name for the server on the login screen
//$conf['servers'][0]['desc'] = 'PostgreSQL';
$conf['servers'][0]['desc'] = 'PostgreSQL_local';
$conf['servers'][0]['host'] = '127.0.0.1';
$conf['servers'][0]['pg_dump_path'] = 'C:\\PostgreSQL\\bin\\pg_dump.exe';
$conf['servers'][0]['pg_dumpall_path'] = 'C:\\PostgreSQL\\bin\\pg_dumpall.ex';
$conf['extra_login_security'] = false;
```

Par la suite:

- On configure le chemin en modifiant la variable d'environnement PATH
- On configure l'alias via un clic sur l'icône de WampServer : Apache→ Répertoire Alias→créer un alias
- On configure le model Php pour qu'il tienne compte de Postgresql : PHP→Extensions PHP→php_pgsql

A la fin on redémarre WAMP pour s'assurer que PostgreSQL est démarré et on teste sur le localhost (http://localhost/phppgadmin/).

- **PgModeler**

C'est un logiciel multiplateforme permettant de modéliser, configurer et aussi déployer des bases de données PostgreSQL (figure 10) qui respecte les standards UML.

PgModeler permet :

1. De se connecter directement à une base postgres
2. Du générer du code SQL pour créer une base de données PostgreSQL
3. D'exporter des schémas de bases de données PostgreSQL en PNG.

Bien que PgModeler soit Open Source, il faut payer pour télécharger les fichiers d'installation officiels pour Windows. Si non il faut compiler à partir des sources. Il est important de noter qu'avant la compilation de PgModller il est nécessaire de télécharger et d'installer la librairie Qt (la version 5.6.x est la plus recommandée).

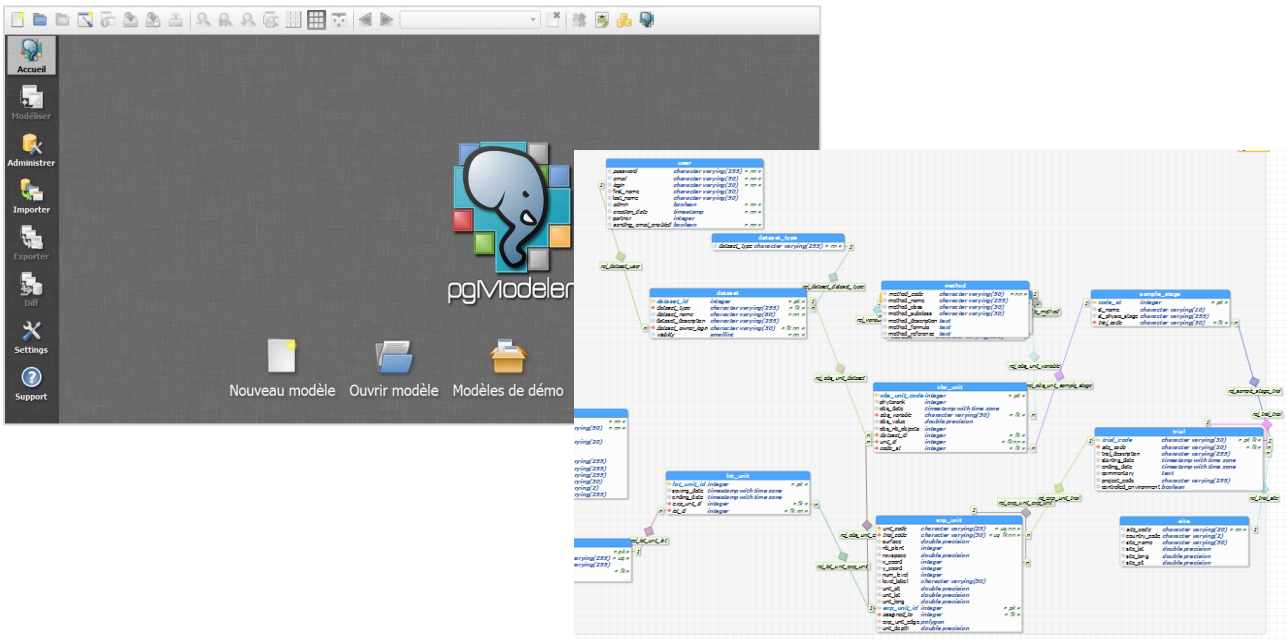


Figure 10: L'interface PgModeler et Exemple de modélisation

IV. Approche Agile « SCRUM »

L'approche de développement adoptée par l'équipe de travail « DAPHNE » s'inscrit dans la logique des méthodes agiles. Parmi ces méthodes on trouve la méthode Scrum, son processus de déroulement est illustré dans la figure

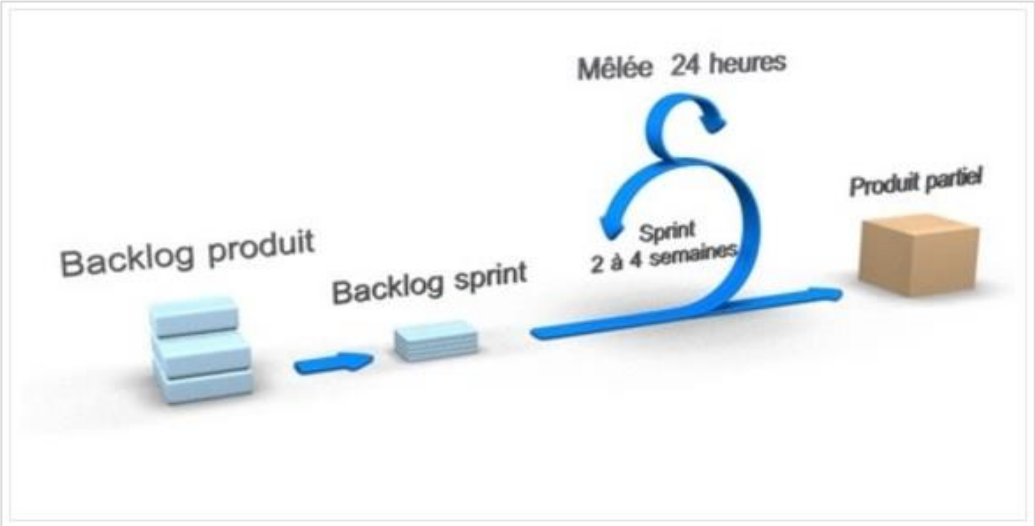


Figure 11: Processus Scrum

Scrum est considéré comme un cadre de gestion de projet, il consiste à définir des rôles, des réunions et des « artefacts[1] ».

Trois rôles sont définis dans cette méthode. Le « Product Owner » représente le client dans mon cas c'est le centre de recherche agronomiques (CIRAD), le « Scrum Master » c'est le chef du projet, il garantit l'application de la méthodologie Scrum et enfin « l'équipe de développement » qui réalise le développement.

D'une manière générale, le cycle de vie d'un projet Scrum est rythmé par des réunions quotidiennes, appelées également « mêlées quotidiennes », de 15 minutes maximum.

Dans le projet DAPHNE des réunions hebdomadaires ont été réalisées durant lesquelles j'ai listé les exigences du client vis-à-vis des fonctionnalités à rajouter dans l'application, pour de produire le « Backlog produit ». Ensuite avec l'équipe de DAPHNE, les éléments prioritaires du « Backlog produit » sont identifier pour pouvoir planifier le sprint et passer à sa réalisation.

[1] Artefact est un terme général désignant toute sorte d'information créée, produite, modifiée ou utilisée par les travailleurs dans la mise au point du système. Il existe principalement 2 types d'artefacts : les artefacts d'ingénierie et les artefacts de gestion. (http://www.cahierdescharges.com).

Lors des réunions on présente également les fonctionnalités terminées au cours du précédent sprint et note les remarques du « Product Owner » concernant le travail réalisé. C'est aussi une occasion pour anticiper le prochain sprint.

Pour mettre en place le processus Scrum on a fait appel à un outil nommé « Trello ». Cet outil nous permet de suivre l'état d'avancement du développement de l'application, car on peut consulter clairement les taches à prévoir, les tâches en cours et les tâches terminées ainsi que la programmation des réunions hebdomadaires **(**figure 12).

Figure 12: Représentation de l'outil Trello (source : https://trello.com)

V. Identification des Packages

La première étape pour concevoir un système d'information est la modélisation de la base de données. Cette base vise à gérer un ensemble de données agronomiques.

Au préalable, un travail de modélisation de la base de données a été élaboré par Sandrine Auzoux (tutrice de stage). Cependant avant l'étape de modélisation, 5 packages (sous-systèmes) ont été définis en amont afin de définir ce que le système devra gérer comme données.

A mon arrivée au Cirad et lors des réunions hebdomadaires d'autres packages ont été ajoutés comme celui des équipement et produits, des variables et des opérateurs, en fonction des besoins recensés auprès des clients (Cirad et partenaires). Chaque package répond à un ou plusieurs cas d'utilisations (Figure 13).

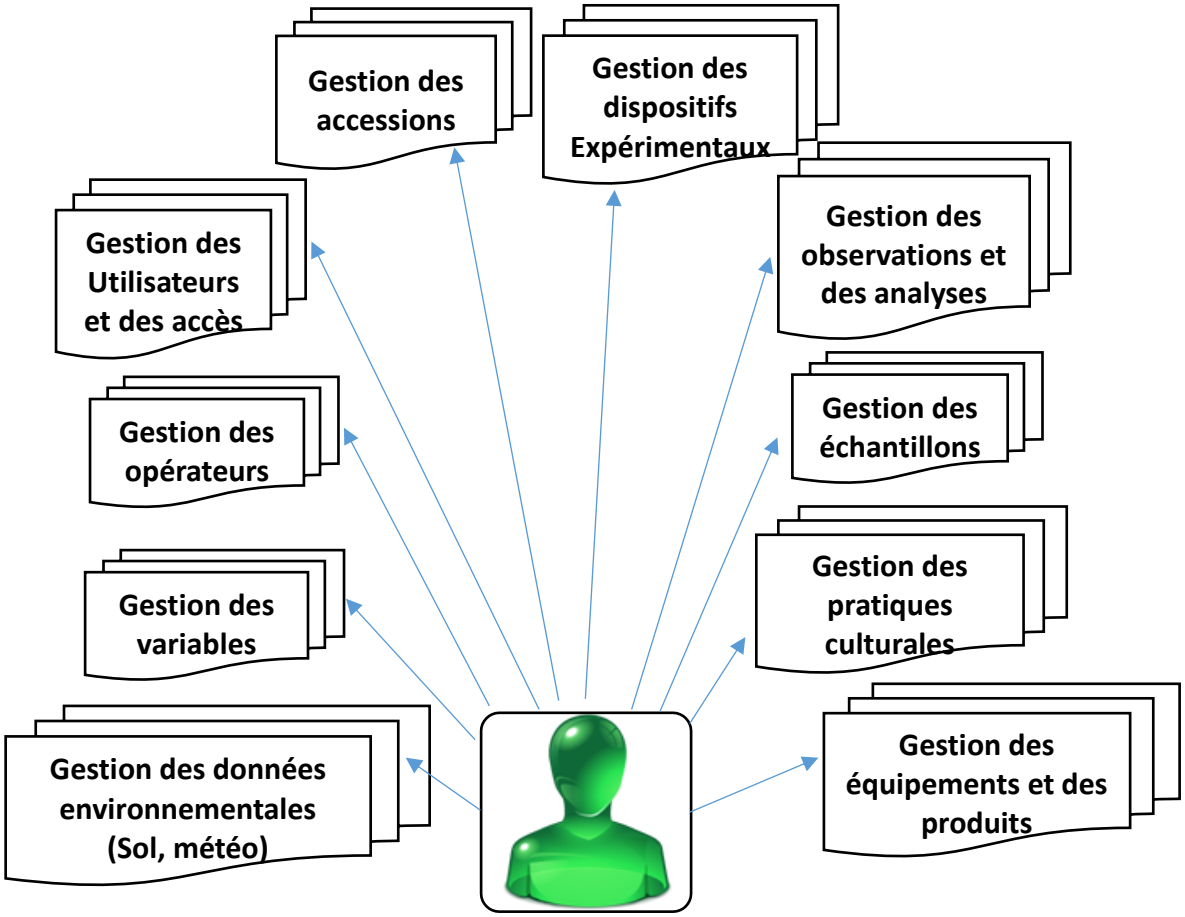


Figure 13: Représentation des packages de système d'information

VI. Diagrammes des cas d'utilisations

1) Gestion des utilisateurs et des accès aux jeux de données

Le diagramme de cas d'utilisation (figure14), montre la manière dont les différents droits d'accès aux données (CRUD) sont distribués par rapport aux utilisateurs du système d'information DAPHNE. Cela nécessite au préalable une authentification de l'utilisateur à l'aide d'un identifiant et d'un mot de passe saisi lors de l'inscription.

2) Gestion des importations de données vers la base

❖ *Accessions*

Un utilisateur doit ajouter des accessions. Chaque accession est définie par un taxon, le site de production de la variété et son pays d'origine.

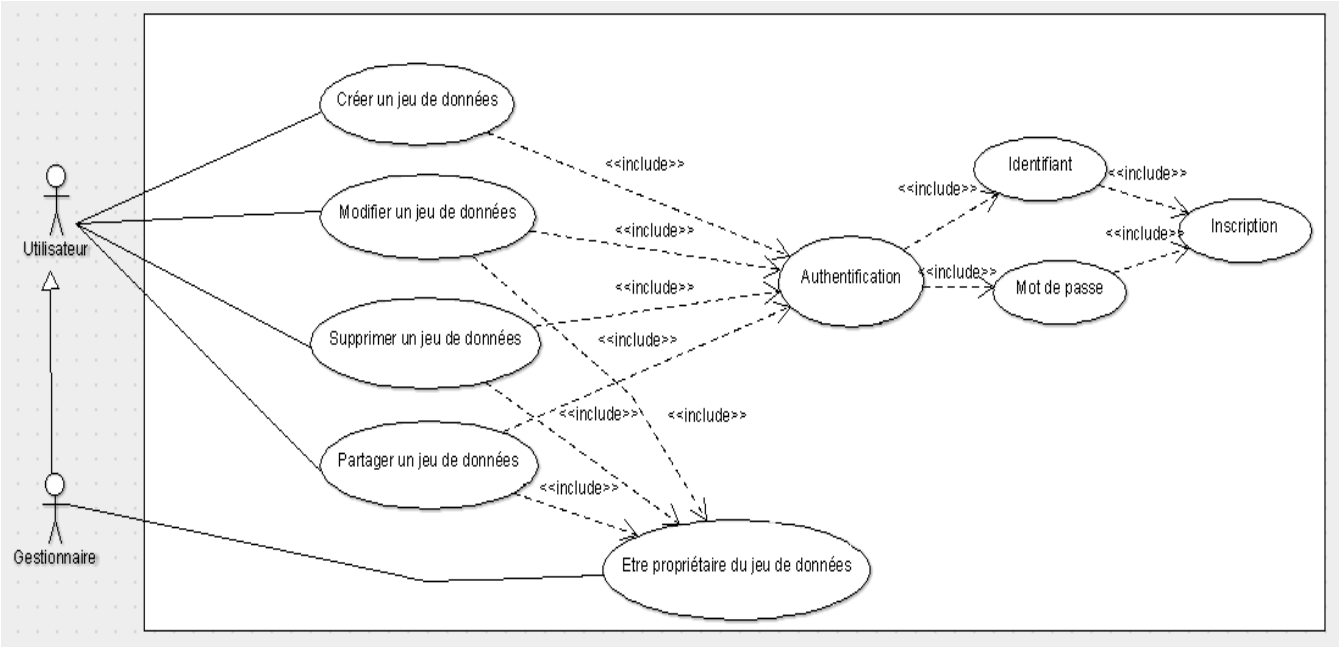


Figure 14: Diagramme de cas d'utilisation d'authentification et de gestion des jeux de données

❖ *Gestion des dispositifs expérimentaux*

Un utilisateur doit ajouter un dispositif expérimental d'un essai planté dans un champ. En revanche, il doit d'abord préciser le nom du projet auquel appartient cet essai et les modalités affectées à ce dispositif.

❖ *Observations et analyses*

Un utilisateur doit pouvoir stocker tous les résultats d'analyses effectuées aux laboratoires et toutes les observations liées aux essais (plante, groupe de plantes, placette, parcelle, etc.) Il est important qu'il puisse déclarer ses propres variables si elles n'existent pas déjà dans la base de données.

Pour importer l'ensemble de ces données via l'interface DAPHNE vers la base de données, on a mis en place deux cas d'utilisation (Figure 15)

3) *Gestion du dictionnaire des variables (trait dictionnary)*

Un utilisateur doit pouvoir sélectionner des variables afin de les utiliser pour la génération des fichiers d'importations de données liée aux observations et analyses, aux données pédoclimatiques et aux itinéraire techniques.

Chaque variable est constituée du nom de l'entité[2], de la méthode et de l'unité de mesure. Le cas d'utilisation du dictionnaire des variables sont illustrés dans la figure 16.

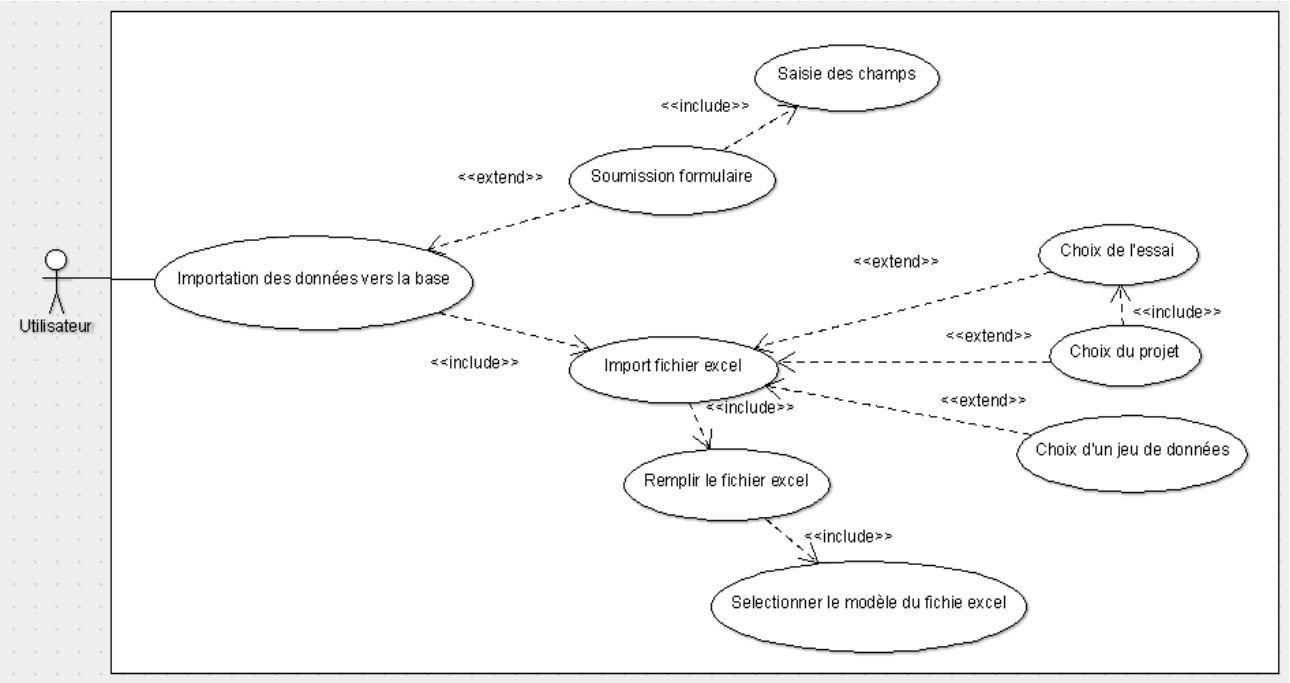


Figure 15: Diagramme de cas d'utilisation de gestion des importations

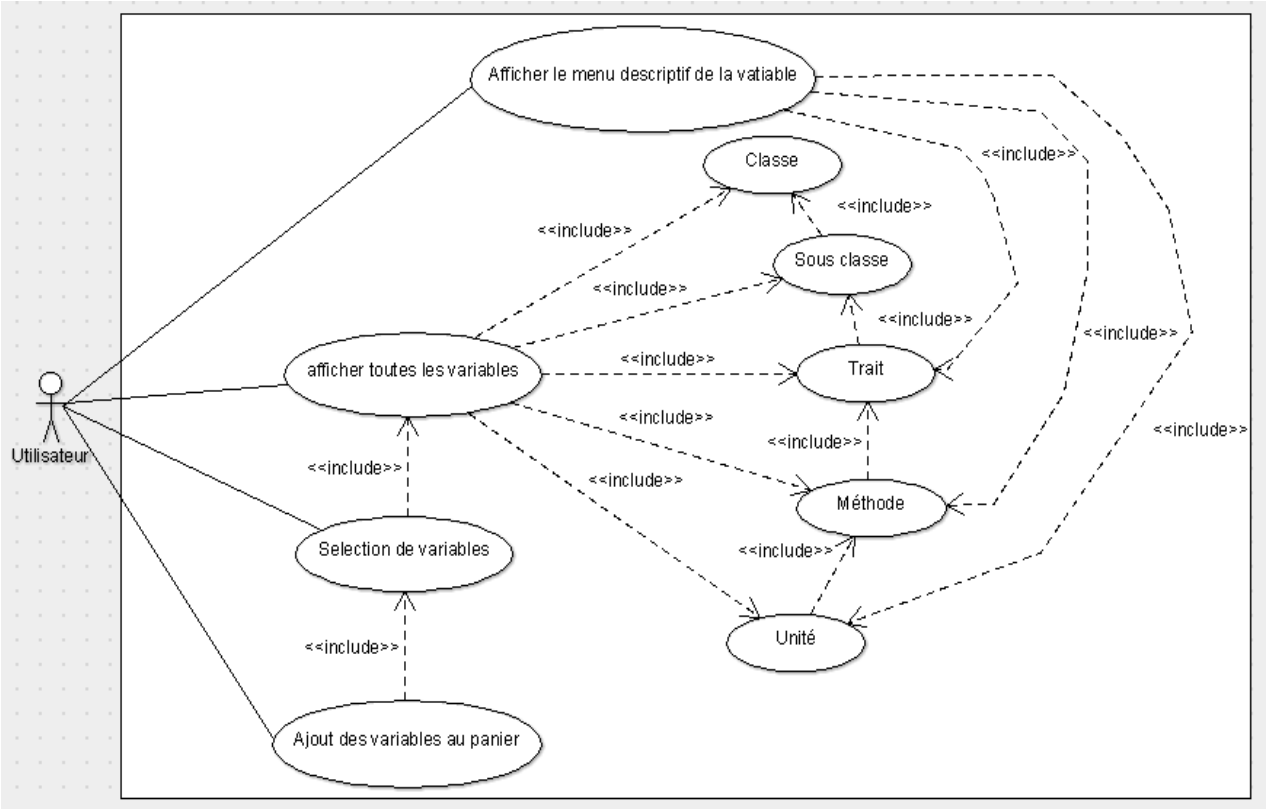


Figure 16: Diagramme de cas d'utilisation de gestion des variables

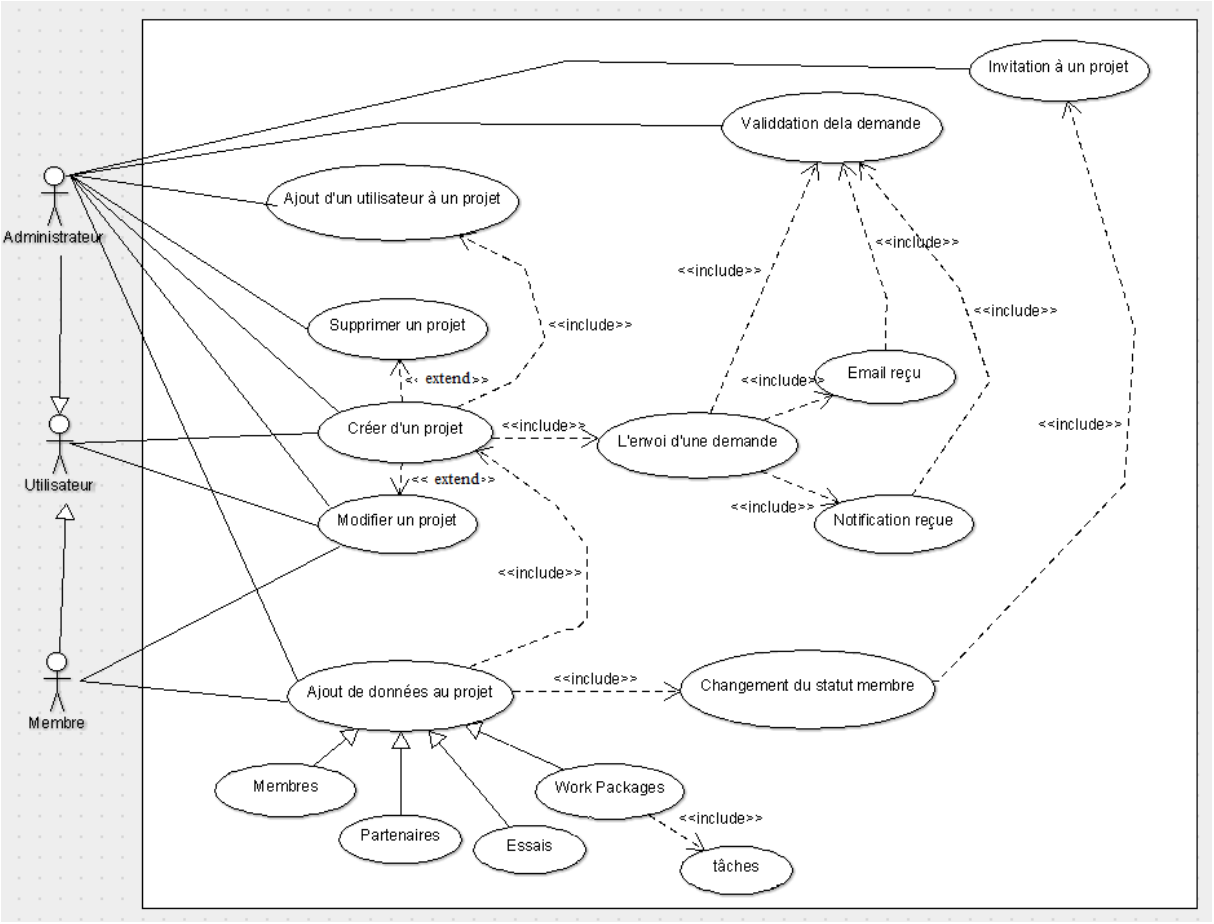


Figure 17: Diagramme de cas d'utilisation de sécurisation des données

[2] c'est le type de l'objet qui est considéré : ce peut être une plante, un organe d'une plante, un produit transformé à partir de la plante, le sol, le climat), du caractère qui est mesuré (longueur de l'entité feuille, profondeur de l'entité sol…

Chapitre III

Résultats et discussions

VII. Implémentation de la base de données

Les différents packages représentés dans le précédent chapitre, ont servi comme support pour implémenter la base de données DAPHNE. Cette base contient 72 tables réparties en 8 grandes thèmes. Pour représenter ces différents thèmes, j'ai fait le choix d'utiliser le modèle conceptuel de données sous format PgModeler.

1. La partie « gestion des utilisateurs, accès et administration ». Cette partie de la base de données est construite autour de 3 tables principales

   - La table « public_user » stocke les informations relatives aux utilisateurs (ex : Nom, prénom, identifiant, mots de passe, adresse email, etc.) et indique si ces derniers sont administrateurs du système. Les mots de passe sont hachés avant l'enregistrement dans la base de données par l'application web via une fonction PHP.

   - La table « public_dataset » stocke les informations relatives à un jeu de données (ex : nom, type, propriétaire, etc.) et contrôle les accès. Ces accès sont définis soit de manière standard en choisissant de rendre le jeu de données privé ou publique. Soit de manière plus précise en invitant un utilisateur à rejoindre un jeu de données, soit en lecture ou en écriture.

   - La table « public_dataset_user » est une table intermédiaire qui permet de stocker les jeux de données partagés avec d'autres utilisateurs, l'identifiant de la personne avec qui le jeu de données a été partagé, ainsi que les permissions qui lui ont été attribuées.

Plusieurs tables peuvent être utilisées pour un même jeu de données, ces dernières stockent des informations liées aux observations, analyses, itinéraires techniques, les facteurs (types de traitements et niveau de traitements) et les données pédoclimatiques (Figure 18).

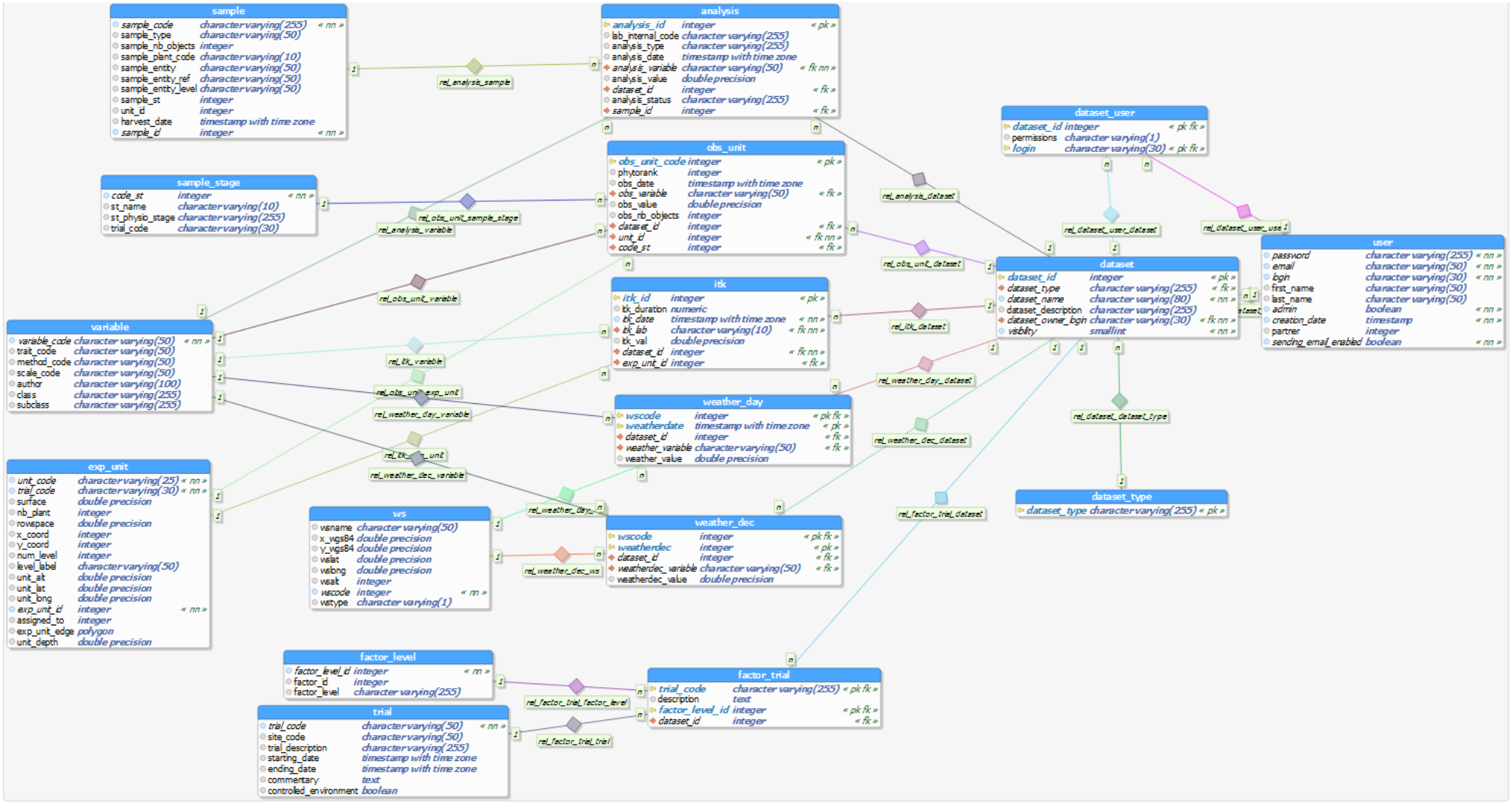


Figure 18: Modèle conceptuel du module de gestion des utilisateurs et des accès aux jeux de données plus les tables appartenant à un jeu de données

2. La partie gestion des accessions stocke les informations liées aux génotypes mis au champ dans les différents essais définis dans la partie "dispositif expérimental" (Figure 19).

- La table « public_accession » permet de définir un génotype produit à un lieu donné et à une date donnée.

3. La partie dispositif expérimental décrit le design d'un essai implanté dans un champ (Figure 20).

- La table « exp_unit » permet de définir autant de niveaux hiérarchiques (num_level) que l'on veut pour décrire un dispositif. Un même dispositif expérimental peut être utilisé dans plusieurs projets de recherche.

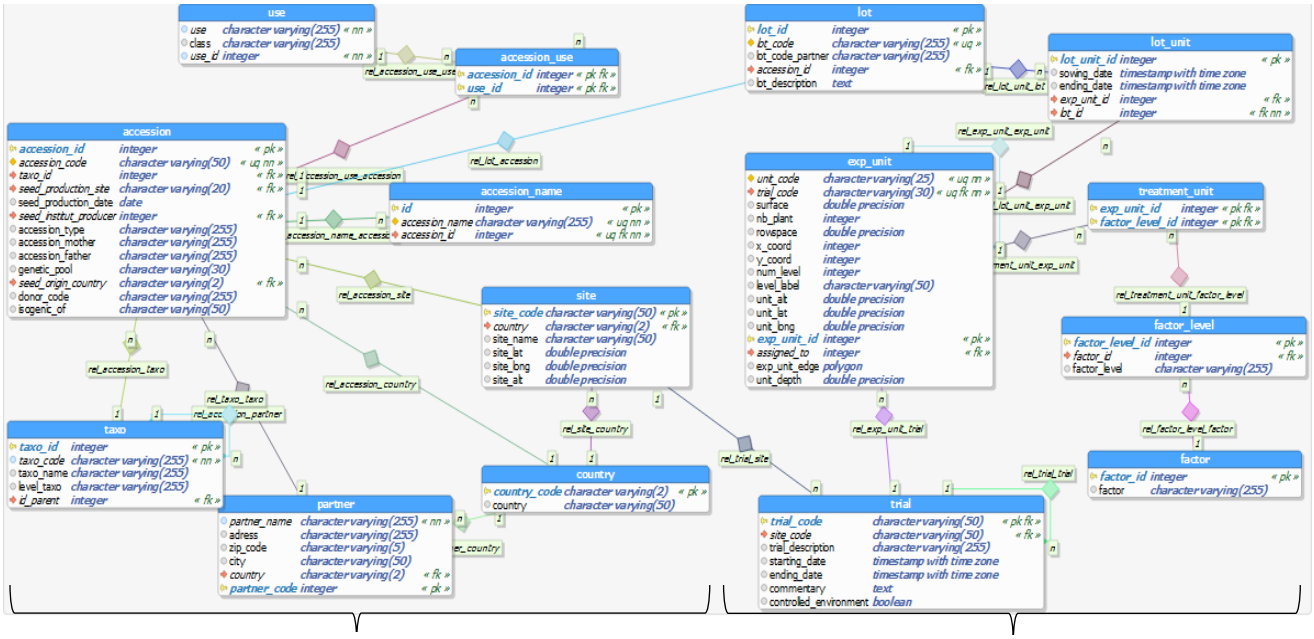


Figure 19: Modèle conceptuel du module de gestion des données des accessions et des dispositifs expérimentaux

4. La partie gestion des observations et analyses permet d'enregistrer les données liées aux observations et les analyses réalisées au champ à tous les niveaux (plante, placette, parcelles, etc.) (figure 20).

- Les tables « public_obs_unit » et « public_analysis » permettent de stocker les données des observations et d'analyses réalisées à des stades différents des plantes semées dans une ou plusieurs unités expérimentales. Afin d'harmoniser le stockage de

ces observations et analyses faites sur les plantes, on fait appel à des variables qui existent déjà dans la base.

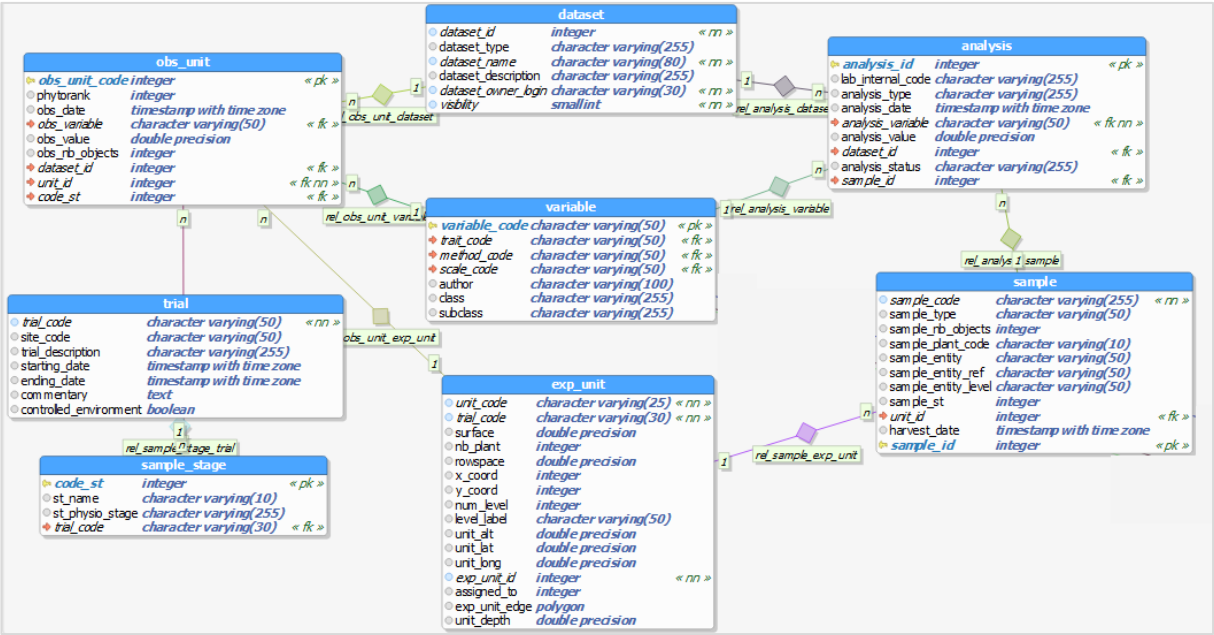


Figure 20: Modèle conceptuel du module de gestion des données d'observations (à gauche) et d'analyses (à droite)

5. La partie de la gestion du dictionnaire des variables. Afin que les données recueillies, tout au long du cycle de vie d'un essai agronomique, puissent être comparées entre elles, il est important de définir leurs natures et propriétés. Le meilleur moyen est celui de faire recours à des ontologies.

Une ontologie est une représentation partagée et consensuelle entre les collaborateurs qui a pour but de se mettre d'accord sur un sujet particulier avec un objectif commun.

Le but est de définir un ensemble de connaissances dans un domaine donné. Elle explicite un vocabulaire en définissant les termes nécessaires pour partager la connaissance liée à ce domaine.

L'ontologie constitue en soi un modèle de données représentatif d'un ensemble de concepts dans un domaine, ainsi que des relations entre ces concepts. Il existe plusieurs types d'ontologies avec des applications diverses. Actuellement, la base de données Daphne mobilise des ontologies d'organes et de de caractères de plantes et prochainement celles des sols, de la météo et des itinéraires techniques.

Ces variables sont définies par le trait observé (composé de l'entité ciblée et du caractère mesuré sur cette dernière), la méthode de mesure et l'unité de mesure (scale) (Figure 21).

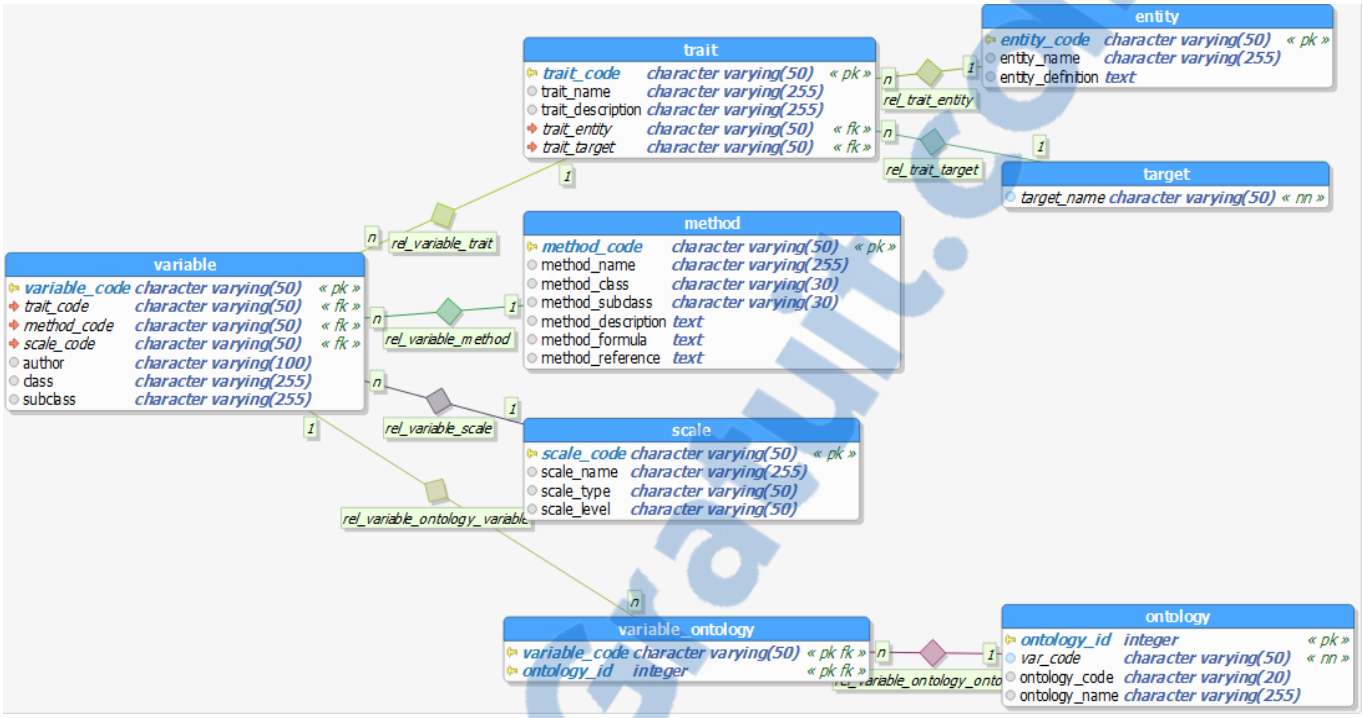
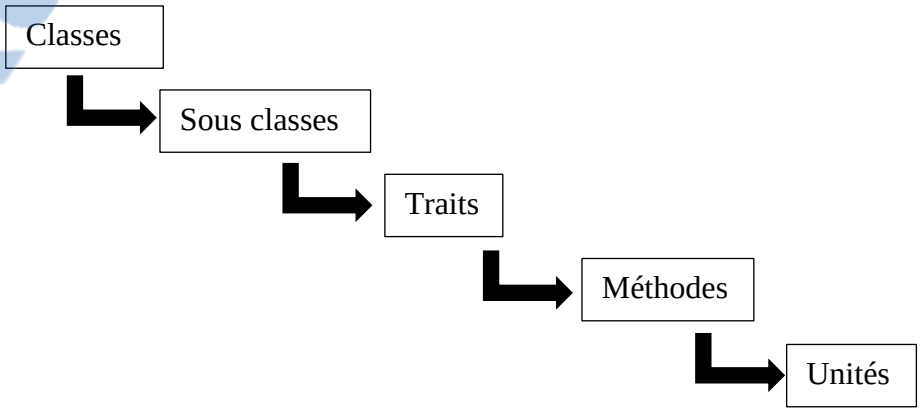


Figure 21: Modèle conceptuel du module de gestion du dictionnaire des variables

Pour mettre en place ce module du dictionnaire des variables, afin de faciliter la sélection des variables observées dans le dictionnaire. J'ai développé un arbre dynamique présentant tous les niveaux de la variable ainsi que les relations récursives existantes entre chaque niveau.
L'arbre construit par programmation peut être plié ou au contraire déplié pour faire apparaître toutes les classes d'intérêt à savoir la plante, le sol, la météo, les itinéraires techniques. Le schéma ci-dessous illustre les activités de clics sur chaque classe pour afficher le niveau du dessous jusqu'à l'obtention du 5ème et dernier niveau. Cet arbre est essentiellement développé en utilisant JavaScript, Ajax et JQuery.



VIII. Implémentation de l'application web

L'application est stockée dans le serveur « Albères » du Cirad (nom de domaine : alberes.CIRAD.fr), ce qui permet d'accéder en ligne à l'interface web (figure 22).

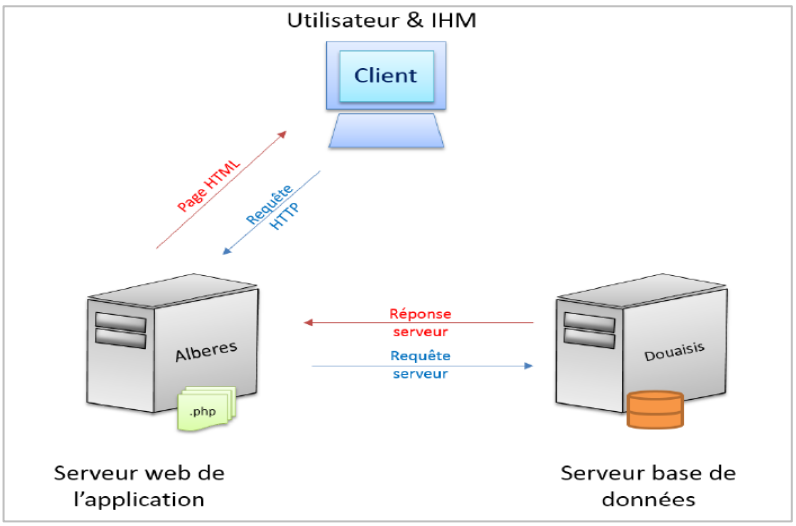


Figure 22: L'architecture à trois niveaux du système d'information (Source: BOULNEMOUR, 2016)

Après l'authentification (figure 23), l'utilisateur a accès à son espace membre (Figure 24), ce qui lui permet d'accéder aux différentes rubriques de l'application.

- ✓ L'accès à la gestion des données, l'utilisateur peut créer son jeu de données et consulter les données auxquelles il a été invité.

- ✓ L'accès à l'espace administration de l'application, cet accès est seulement réservé aux administrateurs de la base de données. A partir de cette rubrique l'administrateur peut gérer les données de DAPHNE (visualisation des tables de la base, gestion des utilisateurs, etc.)

- ✓ L'accès à la rubrique projet (gestion et création). Dans cet espace l'utilisateur peut créer son propre projet, il peut lui associer des essais, des organismes partenaires du projet, des tâches ainsi que d'autres membres. Il peut également voir afficher tous les projets auxquels il a été associé

- ✓ L'accès à la rubrique des notifications. Dans cet espace on retrouve toutes les notifications reçues pour un utilisateur donné (invitation à un projet, un jeu de données, les membres récemment inscrits, etc.).

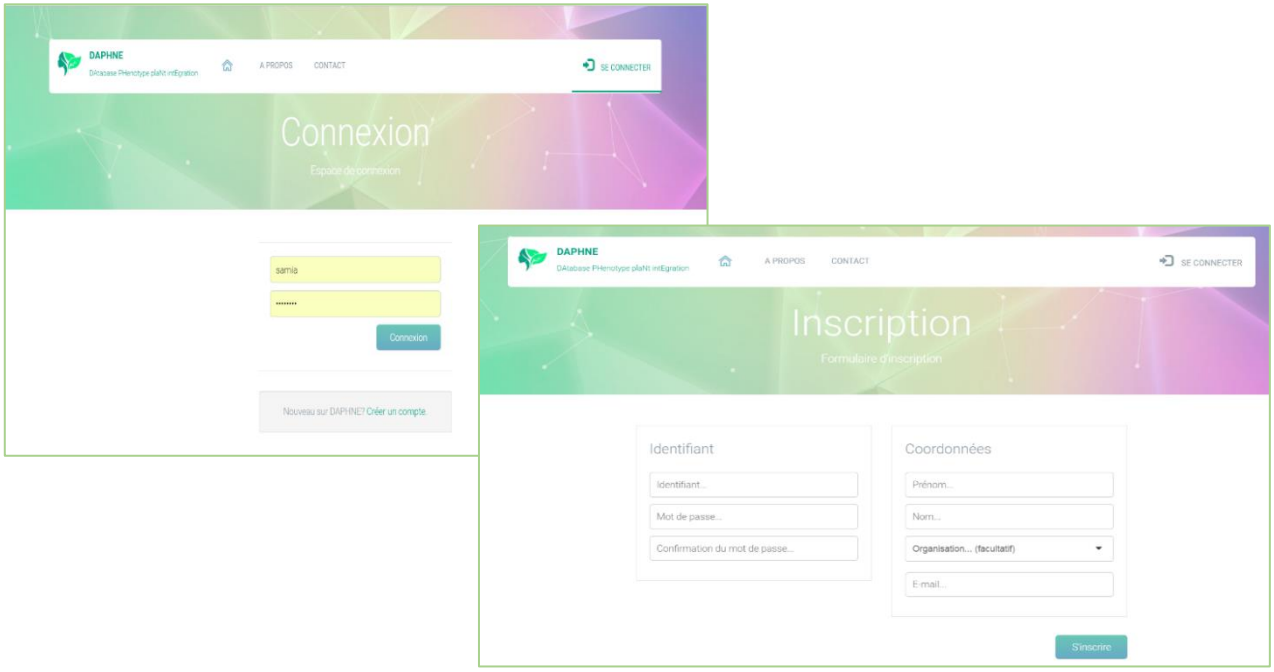


Figure 23: Page de connexion et formulaire d'inscription à DAPHNE

L'importation des données vers la base de données se fait via l'espace « importation/saisie ». Dans cet espace, on affiche des petites rubriques représentant l'ensemble des données à importer (Figure25).

Deux processus d'importation de données ont été mis en place soit par :

- La saisie des champs d'un formulaire en ligne (Figure 26) ;
- Chargement d'un fichier Excel (Figure 27)

Le plus souvent, on choisit l'importation via les fichiers excels car le volume des données à importer dans la base est très important.

Pour mettre en place ce module d'importation, J'ai utilisé une bibliothèque spécifique à ce genre de manipulation « PhpExcel ». Grâce aux fonctionnalités que propose cette bibliothèque, on peut lire notre fichier excel (lignes et colonnes) et stocker par la suite les données de chaque feuille d'excel dans les tables qui correspondent dans la base de données DAPHNE.

- La première étape consiste à télécharger le modèle excel qui correspond aux données qu'on souhaite stocker dans DAPHNE. Ce modèle excel ne contient que les entêtes et il est hébergé dans la racine de notre projet dans un dossier dans le serveur.

.

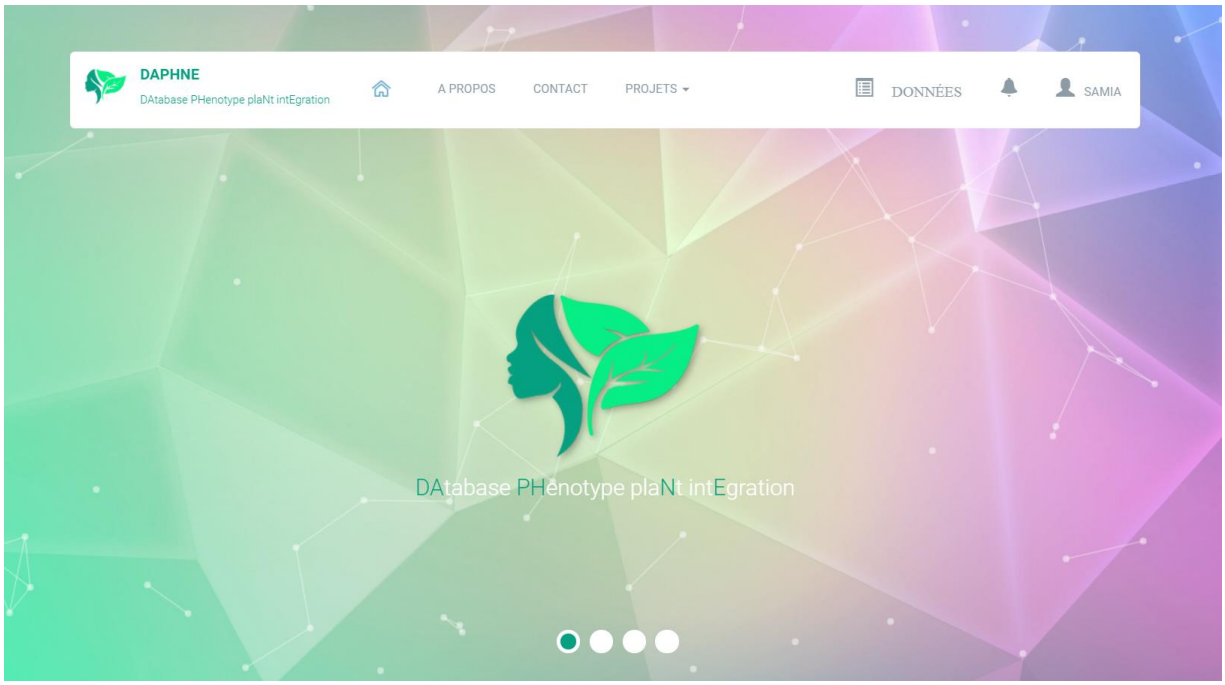


Figure 24: Espace membre matérialisé par de nouveaux liens disponibles dans la barre de navigation

L'icône permet l'accès à la page de saisie et d'importation de données, au dictionnaire des variables ainsi qu'aux données de l'utilisateur.
L'icône permet l'accès aux notification

- Après avoir téléchargé le modèle on le remplit avec nos données, on le charge à nouveau à travers notre interface web et on clique sur importer, les données seront par la suite stockées dans les tables de la base de données.

Figure 25: Espace « importation et saisie des données », chaque rubrique représente les formulaires de données à importer vers la base de données

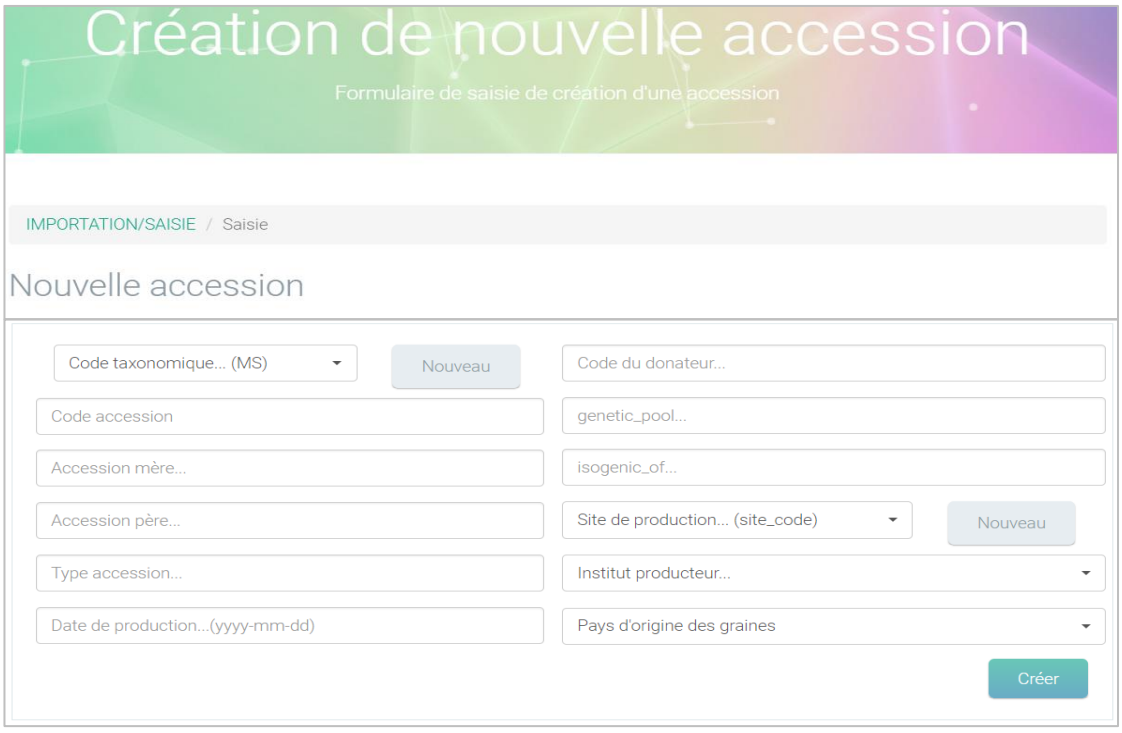


Figure 26: Saisie des données accessions via un formulaire en ligne

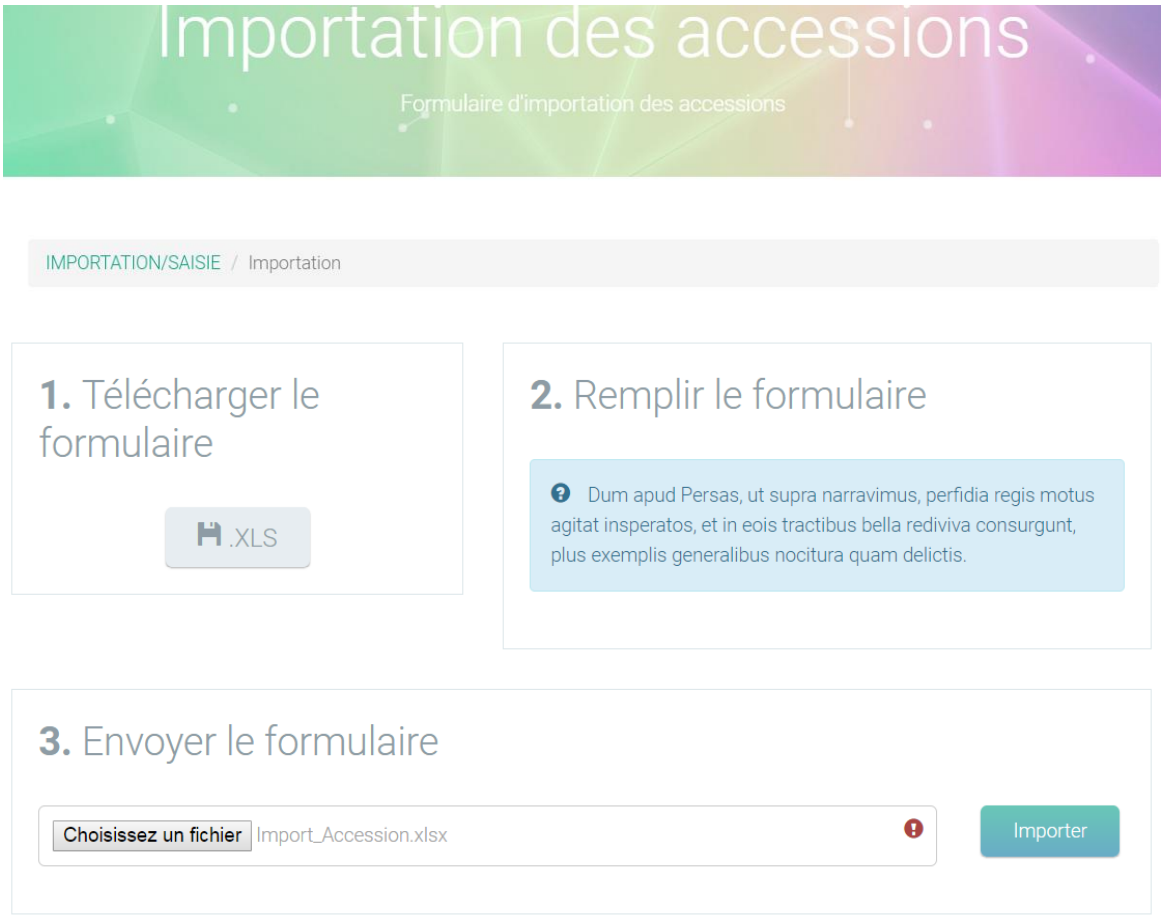


Figure 27: Importation des données accessions vers la base par le chargement d'un fichier excel

L'accès au dictionnaire de variables se fait directement de la barre de navigation, menu « Données » (figure 24) ou à partir de l'interface saisie et importation des données (figure25). Cette interface (figure 28) nous permet de sélectionner les variables sur lesquelles on a apporté des observations, des analyses. Ces variables sélectionnées peuvent être stockées dans un panier pour une utilisation ultérieure surtout dans le cas des imports de données.

Figure 28: Représentation des 5 niveaux que contient le dictionnaire des variables
1.les classes, 2. les Sous classes, 3. Les traits,
4.les méthodes, 5. Les unités

IX. Conclusion et perspectives

DAPHNE est un système d'information conçu pour pouvoir accueillir des données issues du domaine agronomique. A l'heure actuelle ce système permet de gérer des données issues des essais agronomiques d'une ou de plusieurs unités expérimentales et des ontologies agronomiques. Ces données cumulées sur plusieurs années de recherche réparties dans divers fichiers, sont de nature relativement complexes car elles rassemblent plusieurs disciplines, plusieurs espèces, plusieurs sites et plusieurs partenaires scientifiques.

Ce système d'information vise à centraliser, harmoniser et organiser le stockage des données agronomiques, d'une manière à permettre aux chercheurs une exploitation plus flexible de l'ensemble des données. Ce qui va leur permettre d'avancer dans les travaux de recherche, comprenant l'analyse des données et leur partage.

La plupart des modules développés dans l'application sont destinés à importer les données dans la base. Actuellement l'application permet de définir les accès sécurisés aux jeux de données et aux projets dans lesquels les utilisateurs évoluent, la recherches et la sélection des variables plantes stockées dans un dictionnaire, l'ajout, la suppression et la mise à jour des données stockées dans la base de données. A noter que l'intégration de ces données se fait par l'intermédiaire de formulaires de saisie en ligne ou par l'importation de fichiers excel spécifiques dont les modèles sont téléchargeables depuis l'application.

Néanmoins le développement du système d'information DAPHNE est loin d'être terminé. Il reste plusieurs aspects à développer. A savoir, la finalisation des fichiers d'import concernant les échantillons, les pratiques culturales, les équipements et aussi développement des interfaces pour l'extraction des données de la base et leur visualisation.

Réferences bibliographiques

- Boulnemour Mehdi, (2016). Développer un system de gestion de données pour le suivi des dispositifs expérimentaux agronomiques.

Liens internet

- Documentation CodeIgniter et PostgreSQL
  https://codeigniter.com/user_guide/
  http://jc.etiemble.free.fr/abc/index.php/realisations/trucs-astuces/postgresql-wamp
- Documentation librairie PhpExcel
  https://docs.typo3.org/typo3cms/extensions/phpexcel_library/1.7.4/manual.html#_Toc237519893

Annexes

Annexes 1

Tableau de synthèse des classes de helpers CodeIgniter

Nom de la classe utilitaire	Description
Array Helper	Cette classe utilitaire permet de manipuler des tableaux PHP. Fonctions : element() et ramdom_element() Chargement : $this->load->helper('array') ;
Compatibility Helper	Cette classe utilitaire permet de fournir des méthodes de compatibility entre PHP 4 et PHP5. Fonctions : file_put_contents(), fputcsv(), http_build_query(), str_ireplace() et stripos() Chargement : $this->load->helper('compatibility') ;
Cookie Helper	Cette classe utilitaire permet de manipuler des cookies HTTP. Fonctions : set_cookie(), get_cookie() et delete_cookie () Chargement : $this->load->helper('cookie') ;
Date Helper	Cette classe utilitaire permet de manipuler des dates en PHP. Fonctions : now(), mdate(), standard_date(), local_to_gmt(),mysql_to_unix(), unix_to_human(), human_to_unix(), timespan(), days_in_month(), timezones() et timezone_menu() Chargement : $this->load->helper('date') ;
Directory Helper	Cette classe utilitaire permet de manipuler des répertoires. Fonction : directory_map () Chargement : $this->load->helper('directory') ;
Download Helper	Cette classe utilitaire permet de manipuler les entêtes http pour forcer le téléchargement plutôt que l'affichage de données. Fonction : force_download ()
Email Helper	Cette classe utilitaire permet de gérer la validité et l'envoi d'emails. Fonction : email () Chargement : $this->load->helper('email) ;
File Helper	Cette classe utilitaire permet de manipuler des fichiers. Fonctions : read_file(), write_file(), delete_files(), get_filenames(), get_dir_file_info(), get_file_info(), get_mime_by_extension(), symbolic_permissions() et octal_permissions() Chargement : $this->load->helper('file') ;
Form Helper	Cette classe utilitaire permet de générer par programmation des formulaires HTML. Fonctions :form_open(), form_open_multipart(), form_hidden(), form_input(), form_password(), form_upload(), form_textarea(), form_dropdown(), form_multiselect(),form_fieldset(), form_fieldset_close(), form_checkbox(), form_radio(), form_submit(), form_label(), form_reset(), form_button(), form_close(), form_prep(), set_value(), set_select() et set_radio() Fonctions : Chargement : $this->load->helper('form') ;

HTML Helper	Cette classe utilitaire permet de gérer des éléments HTML. Fonctions : br(), heading(), img(), link_tag(), nbs(), ol(), ul(), meta() et doctype() Chargement : $this->load->helper('html') ;
Inflector Helper	Cette classe utilitaire permet de modifier des chaînes de caractères. Fonctions : singular(), plural(), camelize(), underscore() et humanize() Chargement : $this->load->helper('inflector') ;
Language Helper	Cette classe utilitaire permet de gérer les tranductions. Fonction : lang() Chargement : $this->load->helper('language') ;
Number Helper	Cette classe utilitaire permet de traduire des valeurs en octets dans un format lisibles avec unité. Fonction : byte_format() Chargement : $this->load->helper('number') ;
Path Helper	Cette classe utilitaire permet de récupérer le chemin absolu d'une ressource. Fonctions : set_realpath() Chargement : $this->load->helper('path') ;
Security Helper	Cette classe utilitaire permet de faciliter la gestion de la sécurité au sein de votre application. Fonctions : xss_clean(), dohash(), strip_image_tags() et encode_php_tags() Chargement : $this->load->helper('security') ;
Smiley Helper	Cette classe utilitaire permet de traduire des smileys en image à partir de ressources fournies par le projet Code Igniter http://codeigniter.com/download_files/smileys.zip Fonctions : get_clickable_smileys, smiley_js() et parse_smileys() Chargement : $this->load->helper('smiley') ;
String Helper	Cette classe utilitaire permet de manipuler et générer des chaînes de caractères.
String Helper	Cette classe utilitaire permet de manipuler et générer des chaînes de caractères. Fonctions : random_string(), alternator(), repeater(), reduce_double_slashes(), trim_slashes(), reduce_multiples() et quotes_to_entities() Chargement : $this->load->helper('string') ;
Text Helper	Cette classe utilitaire permet de manipuler du texte. Fonctions : word_limiter(), character_limiter(), ascii_to_entities(), entities_to_ascii(), word_censor(), highlight_code(), highlight_phrase() et word_wrap() Chargement : $this->load->helper('text') ;
Typography Helper	Cette classe utilitaire permet de gérer la typographie d'un texte dans la page Web. Fonctions : auto_typography () et nl2br_except_pre () Chargement : $this->load->helper('typography') ;
URL Helper	Cette classe utilitaire permet de manipuler et gérer des URLs. Fonctions : site_url(), base_url(), current_url(), uri_string(), index_page(), anchor(), anchor_popup(), mailto(), safe_mailto(), auto_link(), url_title(), prep_url() et redirect() Chargement : $this->load->helper('url') ;
XML Helper	Cette classe utilitaire permet de gérer les caractères réservés en XML en les convertissant. Fonction : xml_convert () Chargement : $this->load->helper('xml) ;

Annexe 2

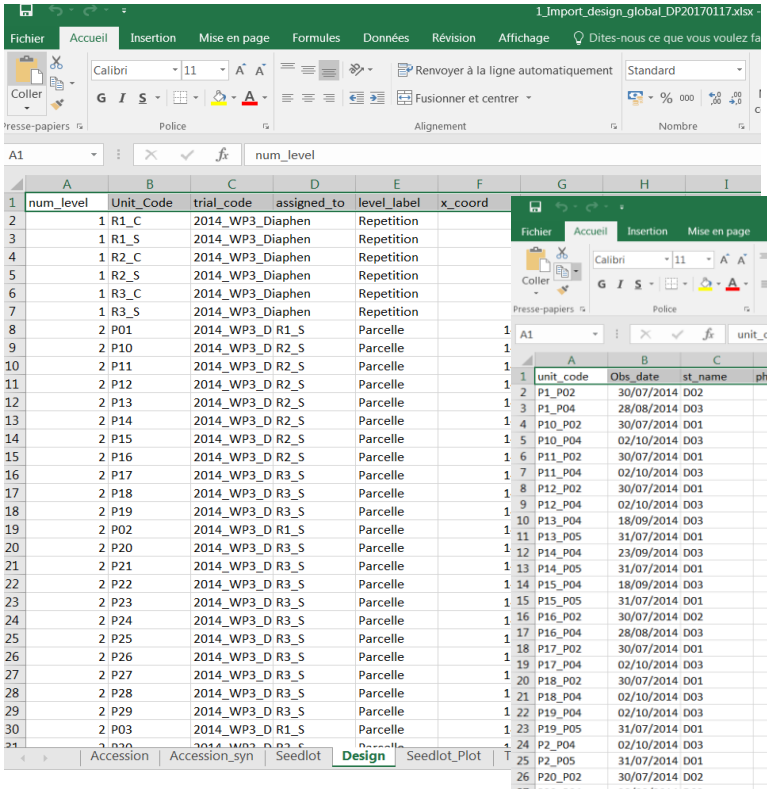
Tableau de synthèse des classes des Librairies de CodeIgniter

Nom de la classe **librairie**	Description
Benchmarking Class	Cette librairie permet de réaliser des mesures de temps de traitements de votre application. Chargement :$this->output->enable_profiler(TRUE);
Calendar Class	Cette librairie permet de fournir des méthodes de gestion de calendrier. Chargement : $this->load->library('calendar') ;
Cart Class	Cette librairie permet de gérer un caddy pour site de ecommerce. Chargement : $this->load->library('cart') ;
Config Class	Cette librairie permet de gérer des fichiers de configuration. Chargement : Automatique
Database Class	Cette librairie permet de manipuler des données dans une base de données. Chargement : $this->load->library('database') ;
Email Class	Cette librairie permet de manipuler et envoyer des emails avec des fonctionnalités avancées tel que les pièces jointes. Chargement : $this->load->library('email') ;
Encryption Class	Cette librairie permet de manipuler le chiffrage et de déchiffrage de données. Chargement : $this->load->library('encrypt') ;
File Uploading Class	Cette librairie permet de gérer l'upload de fichier depuis le navigateur client. Chargement : $this->load->library('upload');
Form Validation Class	Cette librairie permet de gérer la validation des formulaires HTML. Chargement : $this->load->library('form_validation') ;
FTP Class	Cette librairie permet de gérer des transferts de fichiers via FTP. Chargement : $this->load->library('ftp') ;
HTML Table Class	Cette librairie permet de générer des tableaux HTML. Chargement : $this->load->library('table') ;
Image Manipulation Class	Cette librairie permet de manipuler des images, créer des vignettes, retailler, effectuer des rotations et des fusions. Chargement : $this->load->library('image_lib') ;
Input and Security Class	Cette librairie permet de gérer le filtrage et la manipulation sécurisée des données. Chargement : Automatique
Loader Class	Cette librairie permet de gérer le chargement des différents éléments dans Code Igniter. Chargement : Automatique
Language Class	Obsolète et remplacé par la classe utilitaire Language.
Output Class	Cette librairie permet de gérer le résultat final envoyé au client. Chargement : Automatique
Pagination Class	Cette librairie permet de gérer la pagination et la navigation dans vos pages. Chargement : $this->load->library('pagination') ;
Session Class	Cette librairie permet de manipuler et de gérer les sessions. Chargement : $this->load->library('session') ;

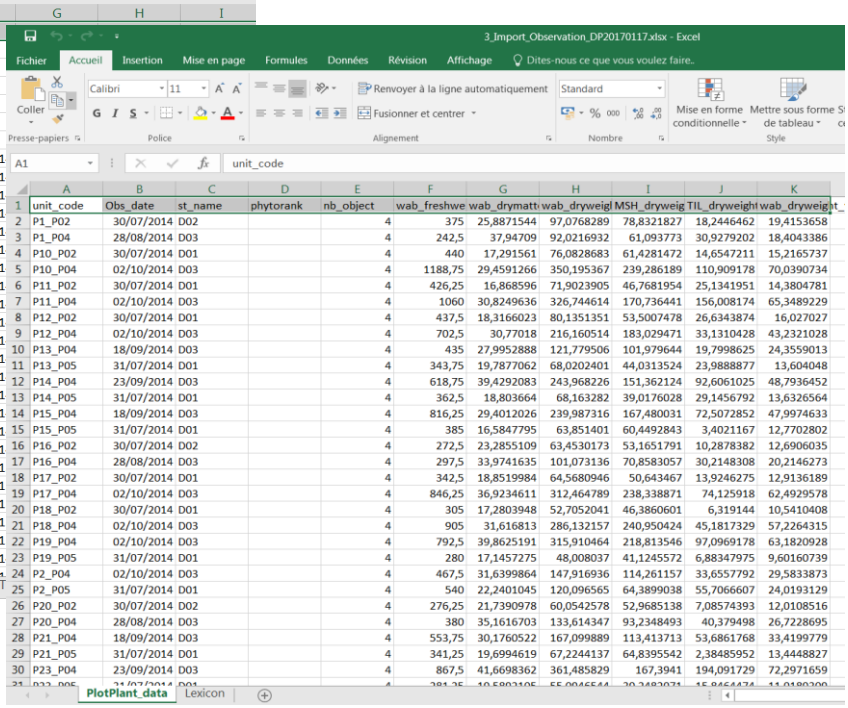
Trackback Class	Cette librairie permet de gérer et manipuler des retroliens ou trackback. Chargement : $this->load->library('trackback') ;
Template Parser Class	Cette librairie permet de supporter un langage simple de template pour vos pages Web. Chargement : $this->load->library('parser') ;
Typography Class	Cette librairie permet de gérer la typographie de vos textes. Chargement : $this->load->library('typography') ;
Unit Testing Class	Cette librairie permet de réaliser et exécuter des tests unitaires. Chargement : $this->load->library('unit_test') ;
URI Class	Cette librairie permet de manipuler des URI. Chargement : Automatique
User Agent Class	Cette librairie permet de manipuler l'entête User-Agent. Chargement : $this->load->library('user_agent') ;
XML-RPC Class	Cette librairie permet de fournir un service XML au format XML-RPC. Chargement : $this->load->library('xmlrpc') ;
Zip Encoding Class	Cette librairie permet de manipuler des données à au format ZIP. Chargement : $this->load->library('zip') ;

Annexe 3

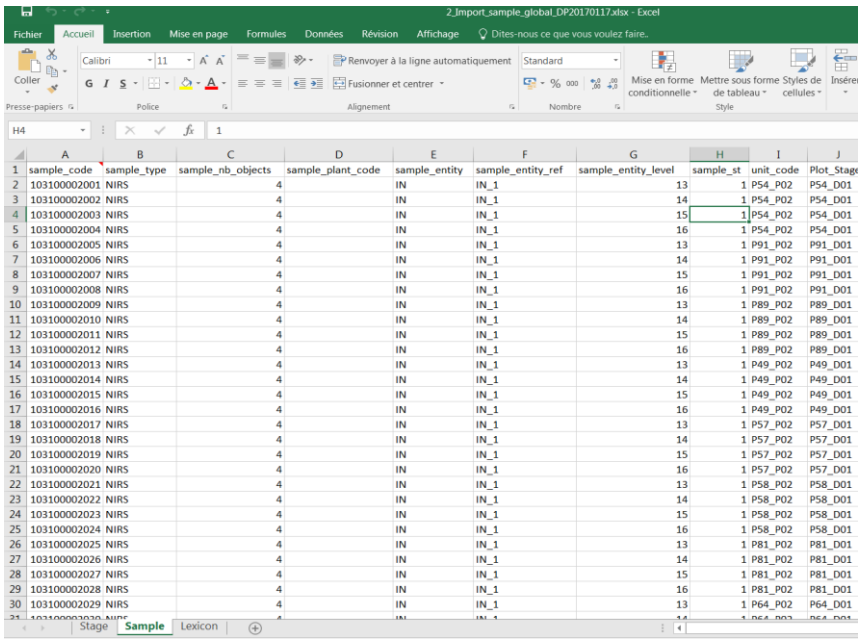
Exemple des données (dispositif expérimental, observations et échantillons) stockées dans des fichiers excels à importer vers la base de données DAPHNE



num_level	Unit_Code	trial_code	assigned_to	level_label	x_coord
1	R1_C	2014_WP3_	Diaphen	Repetition	
1	R1_S	2014_WP3_	Diaphen	Repetition	
1	R2_C	2014_WP3_	Diaphen	Repetition	
1	R2_S	2014_WP3_	Diaphen	Repetition	
1	R3_C	2014_WP3_	Diaphen	Repetition	
1	R3_S	2014_WP3_	Diaphen	Repetition	
2	P01	2014_WP3_D	R1_S	Parcelle	1
2	P10	2014_WP3_D	R2_S	Parcelle	1
2	P11	2014_WP3_D	R2_S	Parcelle	1
2	P12	2014_WP3_D	R2_S	Parcelle	1
2	P13	2014_WP3_D	R2_S	Parcelle	1
2	P14	2014_WP3_D	R2_S	Parcelle	1
2	P15	2014_WP3_D	R2_S	Parcelle	1
2	P16	2014_WP3_D	R2_S	Parcelle	1
2	P17	2014_WP3_D	R3_S	Parcelle	1
2	P18	2014_WP3_D	R3_S	Parcelle	1
2	P19	2014_WP3_D	R3_S	Parcelle	1
2	P02	2014_WP3_D	R1_S	Parcelle	1
2	P20	2014_WP3_D	R3_S	Parcelle	1
2	P21	2014_WP3_D	R3_S	Parcelle	1
2	P22	2014_WP3_D	R3_S	Parcelle	1
2	P23	2014_WP3_D	R3_S	Parcelle	1
2	P24	2014_WP3_D	R3_S	Parcelle	1
2	P25	2014_WP3_D	R3_S	Parcelle	1
2	P26	2014_WP3_D	R3_S	Parcelle	1
2	P27	2014_WP3_D	R3_S	Parcelle	1
2	P28	2014_WP3_D	R3_S	Parcelle	1
2	P29	2014_WP3_D	R3_S	Parcelle	1
2	P03	2014_WP3_D	R1_S	Parcelle	1



unit_code	Obs_date	st_name	phytorank	nb_object	wab_freshwe	wab_drymatt	wab_dryweig	MSH_dryweig	TIL_dryweight	wab_dryweight
P1_P02	30/07/2014	D02		4	375	25,8871544	97,0768289	78,8321827	18,2446462	19,4153658
P1_P04	28/08/2014	D03		4	242,5	37,94709	92,0216932	61,093773	30,9279202	18,4043386
P10_P02	30/07/2014	D01		4	440	17,291561	76,0828683	61,4281472	14,6547211	15,2165737
P10_P04	02/10/2014	D03		4	1188,75	29,4591266	350,195367	239,286189	110,909178	70,0390734
P11_P02	30/07/2014	D01		4	426,25	16,868596	71,9023905	46,7681954	25,1341951	14,3804781
P11_P04	02/10/2014	D03		4	1060	30,8249636	326,744614	170,736441	156,008174	65,3489229
P12_P02	30/07/2014	D01		4	437,5	18,3166023	80,1351351	53,5007478	26,6343874	16,027027
P12_P04	02/10/2014	D03		4	702,5	30,77018	216,160514	183,029471	33,1310428	43,2321028
P13_P04	18/09/2014	D03		4	435	27,9952888	121,779506	101,979644	19,7998625	24,3559013
P13_P05	31/07/2014	D01		4	343,75	19,7877062	68,0202401	44,0313524	23,9888877	13,604048
P14_P04	23/09/2014	D03		4	618,75	39,4292083	243,968226	151,362124	92,6061025	48,7936452
P14_P05	31/07/2014	D01		4	362,5	18,803664	68,163282	39,0176028	29,1456792	13,6326564
P15_P04	18/09/2014	D03		4	816,25	29,4012026	239,987316	167,480031	72,5072852	47,9974633
P15_P05	31/07/2014	D01		4	385	16,5847795	63,851401	60,4492843	3,4021167	12,7702802
P16_P02	30/07/2014	D02		4	272,5	23,2855109	63,4530173	53,1651791	10,2878382	12,6906035
P16_P04	28/08/2014	D03		4	297,5	33,9741635	101,073136	70,8583057	30,2148308	20,2146273
P17_P02	30/07/2014	D01		4	342,5	18,8519984	64,5680946	50,643467	13,9246275	12,9136189
P17_P04	02/10/2014	D03		4	846,25	36,9234611	312,464789	238,338871	74,125918	62,4929578
P18_P02	30/07/2014	D01		4	305	17,2803948	52,7052041	46,3860601	6,319144	10,5410408
P18_P04	02/10/2014	D03		4	905	31,616813	286,132157	240,950424	45,1817329	57,2264315
P19_P04	02/10/2014	D03		4	792,5	39,8625191	315,910464	218,813546	97,0969178	63,1820928
P19_P05	31/07/2014	D01		4	280	17,1457275	48,008037	41,1245572	6,88347975	9,60160739
P2_P04	02/10/2014	D03		4	467,5	31,6399864	147,916936	114,261157	33,6557792	29,5833873
P2_P05	31/07/2014	D01		4	540	22,2401045	120,096565	64,3899038	55,7066607	24,0193129
P20_P02	30/07/2014	D02		4	276,25	21,7390978	60,0542578	52,9685138	7,08574393	12,0108516
P20_P04	28/08/2014	D03		4	380	35,1616703	133,614347	93,2348493	40,379498	26,7228695
P21_P04	18/09/2014	D03		4	553,75	30,1760522	167,099889	113,413713	53,6861768	33,4199779
P21_P05	31/07/2014	D01		4	341,25	19,6994619	67,2244137	64,8395542	2,38485952	13,4448827
P23_P04	23/09/2014	D03		4	867,5	41,6698362	361,485829	167,3941	194,091729	72,2971659



sample_code	sample_type	sample_nb_objects	sample_plant_code	sample_entity	sample_entity_ref	sample_entity_level	sample_st	unit_code	Plot_Stage
103100002001	NIRS	4		IN	IN_1	13	1	P54_P02	P54_D01
103100002002	NIRS	4		IN	IN_1	14	1	P54_P02	P54_D01
103100002003	NIRS	4		IN	IN_1	15	1	P54_P02	P54_D01
103100002004	NIRS	4		IN	IN_1	16	1	P54_P02	P54_D01
103100002005	NIRS	4		IN	IN_1	13	1	P91_P02	P91_D01
103100002006	NIRS	4		IN	IN_1	14	1	P91_P02	P91_D01
103100002007	NIRS	4		IN	IN_1	15	1	P91_P02	P91_D01
103100002008	NIRS	4		IN	IN_1	16	1	P91_P02	P91_D01
103100002009	NIRS	4		IN	IN_1	13	1	P89_P02	P89_D01
103100002010	NIRS	4		IN	IN_1	14	1	P89_P02	P89_D01
103100002011	NIRS	4		IN	IN_1	15	1	P89_P02	P89_D01
103100002012	NIRS	4		IN	IN_1	16	1	P89_P02	P89_D01
103100002013	NIRS	4		IN	IN_1	13	1	P49_P02	P49_D01
103100002014	NIRS	4		IN	IN_1	14	1	P49_P02	P49_D01
103100002015	NIRS	4		IN	IN_1	15	1	P49_P02	P49_D01
103100002016	NIRS	4		IN	IN_1	16	1	P49_P02	P49_D01
103100002017	NIRS	4		IN	IN_1	13	1	P57_P02	P57_D01
103100002018	NIRS	4		IN	IN_1	14	1	P57_P02	P57_D01
103100002019	NIRS	4		IN	IN_1	15	1	P57_P02	P57_D01
103100002020	NIRS	4		IN	IN_1	16	1	P57_P02	P57_D01
103100002021	NIRS	4		IN	IN_1	13	1	P58_P02	P58_D01
103100002022	NIRS	4		IN	IN_1	14	1	P58_P02	P58_D01
103100002023	NIRS	4		IN	IN_1	15	1	P58_P02	P58_D01
103100002024	NIRS	4		IN	IN_1	16	1	P58_P02	P58_D01
103100002025	NIRS	4		IN	IN_1	13	1	P81_P02	P81_D01
103100002026	NIRS	4		IN	IN_1	14	1	P81_P02	P81_D01
103100002027	NIRS	4		IN	IN_1	15	1	P81_P02	P81_D01
103100002028	NIRS	4		IN	IN_1	16	1	P81_P02	P81_D01
103100002029	NIRS	4		IN	IN_1	13	1	P64_P02	P64_D01