GeoNature Documentation

Version latest

juil. 06, 2017

Table des matières

Contenu :

CHAPITRE 1

SERVEUR

Cette procédure décrit l'installation de l'application GeoNature seule. Il est aussi possible d'installer plus facilement GeoNature et tout son environnement (UsersHub, TaxHub et GeoNature-atlas) avec le script `install_all` (voir chapitre INSTALLATION GLOBALE).

Prérequis

— Ressources minimum serveur :

Un serveur disposant d'au moins de 1 Go RAM et de 20-25 Go d'espace disque (une sauvegarde complète des bases et des données produites écrasée chaque jour). Prévoir environ 100 Go pour stratégie de sauvegarde plus complète.

— Disposer d'un utilisateur linux nommé `synthese` (par exemple). Dans ce guide, le répertoire de cet utilisateur est dans `/home/synthese`

```
sudo adduser --home /home/synthese synthese
```

Installation et configuration du serveur

Installation pour Debian 7.

notes Cette documentation concerne une installation sur Debian. Pour tout autre environemment les commandes sont à adapter.

notes Durant toute la procédure d'installation, travailler avec l'utilisateur `synthese`. Ne changer d'utilisateur que lorsque la documentation le spécifie.

```
su -
apt-get install apache2 php5 libapache2-mod-php5 php5-gd libapache2-mod-wsgi php5-
↪pgsql cgi-mapserver sudo gdal-bin
usermod -g www-data synthese
usermod -a -G root synthese
```

```
adduser synthese sudo
exit
```

— Fermer la console et la réouvrir pour que les modifications soient prises en compte.
— Activer le `mod_rewrite` et redémarrer Apache

```
sudo a2enmod rewrite
sudo apache2ctl restart
```

— Vérifier que le répertoire `/tmp` existe et que l'utilisateur `www-data` y ait accès en lecture/écriture.

Installation et configuration de PostgreSQL

— Sur Debian 8, PostgreSQL est livré en version 9.4 et postGIS en 2.1, vous pouvez sauter l'étape suivante. Sur Debian 7, il faut revoir la configuration des dépots pour avoir une version compatible de PostgreSQL (9.3) et PostGIS (2.1). Voir http://foretribe.blogspot.fr/2013/12/the-posgresql-and-postgis-install-on.html.

```
sudo sh -c 'echo "deb http://apt.postgresql.org/pub/repos/apt/ wheezy-pgdg main"␣
↪>> /etc/apt/sources.list'
sudo wget --quiet -O - https://www.postgresql.org/media/keys/ACCC4CF8.asc | sudo␣
↪apt-key add -
sudo apt-get update
```

— Installation de PostreSQL/PostGIS pour Debian 8

```
sudo apt-get update
sudo apt-get install postgresql postgresql-client
sudo apt-get install postgresql-9.4-postgis-2.1
sudo adduser postgres sudo
```

— Installation de PostreSQL/PostGIS pour Debian 7

```
sudo apt-get install postgresql-9.3 postgresql-client-9.3
sudo apt-get install postgresql-9.3-postgis-2.1
sudo adduser postgres sudo
```

— Configuration de PostgreSQL pour Debian 8 - permettre l'écoute de toutes les IP

```
sed -e "s/#listen_addresses = 'localhost'/listen_addresses = '*'/g" -i /etc/
↪postgresql/9.4/main/postgresql.conf
sudo sed -e "s/# IPv4 local connections:/# IPv4 local␣
↪connections:\nhost\tall\tall\t0.0.0.0\/0\t md5/g" -i /etc/postgresql/9.4/main/
↪pg_hba.conf
/etc/init.d/postgresql restart
```

— Configuration de PostgreSQL pour Debian 7 - permettre l'écoute de toutes les IP

```
sed -e "s/#listen_addresses = 'localhost'/listen_addresses = '*'/g" -i /etc/
↪postgresql/9.3/main/postgresql.conf
sudo sed -e "s/# IPv4 local connections:/# IPv4 local␣
↪connections:\nhost\tall\tall\t0.0.0.0\/0\t md5/g" -i /etc/postgresql/9.3/main/
↪pg_hba.conf
/etc/init.d/postgresql restart
```

— Création de l'utilisateur PostgreSQL

```
sudo su postgres
psql
```

```
CREATE ROLE geonatuser WITH LOGIN PASSWORD 'monpassachanger';
\q
```

L'utilisateur `geonatuser` sera le propriétaire de la base de données `geonaturedb` et sera utilisé par l'application pour se connecter à celle-ci.

L'application fonctionne avec le mot de passe `monpassachanger` par defaut mais il est conseillé de le modifier !

Ce mot de passe, ainsi que l'utilisateur PostgreSQL créés ci-dessus (`geonatuser`) sont des valeurs par défaut utilisées à plusieurs reprises dans l'application. Ils peuvent cependant être changés. S'ils doivent être changés, ils doivent l'être dans plusieurs fichiers de l'application :

- config/settings.ini
- config/databases.yml
- wms/wms.map

CHAPITRE 2

INSTALLATION DE L'APPLICATION

Cette procédure décrit l'installation de l'application GeoNature seule. Il est aussi possible d'installer plus facilement GeoNature et tout son environnement (UsersHub, TaxHub et GeoNature-atlas) avec le script `install_all` (voir chapitre INSTALLATION GLOBALE).

Prérequis

— Environnement serveur :

Voir le chapitre sur l'installation du serveur (http://geonature.readthedocs.org/fr/latest/server.html)

— Disposer d'un utilisateur linux nommé par exemple `synthese`. Dans ce guide, le répertoire de cet utilisateur est dans `/home/synthese`

— Se loguer sur le serveur avec l'utilisateur `synthese` ou tout autre utilisateur linux faisant partie du groupe www-data.

— Récupérer le zip de l'application sur le Github du projet (X.Y.Z à remplacer par le numéro de version souhaitée), dézippez le dans le répertoire de l'utilisateur linux du serveur puis copiez le dans le répertoire de l'utilisateur linux :

```
cd /home/synthese
wget https://github.com/PnEcrins/GeoNature/archive/X.Y.Z.zip
unzip X.Y.Z.zip
mv GeoNature-X.Y.Z/ geonature/
```

Configuration Apache

— Adaptation des chemins de l'application pour la configuration Apache

Editer les fichiers de configuration Apache : `apache/sf.conf`, `apache/synthese.conf` et `apache/wms.conf` et adapter les chemins à ceux de votre serveur. Pour Apache 2.4 et supérieur, le répertoire de publication web par défaut est `/var/www/html/geonature` ; à changer dans `apache/synthese.conf`.

— Prise en compte de la configuration Apache requises pour Symfony (avant Apache2.4) :

```
sudo sh -c 'echo "Include /home/synthese/geonature/apache/*.conf" >> /etc/
↪apache2/apache2.conf'
sudo apache2ctl restart
```

— Prise en compte de la configuration Apache requises pour Symfony Apache 2.4 et supérieur :

```
sudo sh -c 'echo "IncludeOptional /home/synthese/geonature/apache/*.conf" >> /
↪etc/apache2/apache2.conf'
sudo apache2ctl restart
```

— Pour les utilisateurs d'Apache 2.4 (par défaut dans Debian 8), installer perl et cgi

```
sudo apt-get install libapache2-mod-perl2
sudo a2enmod cgi
sudo apache2ctl restart
```

Configuration de la base de données PostgreSQL

— Se positionner dans le répertoire de l'application ; par exemple `geonature` :

```
cd geonature
```

— Copier et renommer le fichier `config/settings.ini.sample` en `config/settings.ini` :

```
cp config/settings.ini.sample config/settings.ini
```

— Mettre à jour le fichier `config/settings.ini` avec vos paramètres de connexion à la BDD :

```
nano config/settings.ini
```

Renseigner le nom de la base de données, les utilisateurs PostgreSQL et les mots de passe. Il est possible mais non conseillé de laisser les valeurs proposées par défaut.

La projection locale peut être modifiée si vous n'êtes pas en métropole. Attention : les couches SIG ainsi que le jeu de données fournis avec l'application sont tous en lambert 93 (2154). Pour ne pas les insérer lors de la création de la base, vous devez mettre les paramètres `install_sig_layers` et `add_sample_data` à `false`.

Si vous êtes en métropole, il est conseillé de laisser la projection officielle en Lambert 93 (2154) et d'insérer au moins les couches SIG fournies.

ATTENTION : Les valeurs renseignées dans ce fichier sont utilisées par le script d'installation de la base de données `install_db.sh` ainsi que par le script d'installation de l'application `install_app.sh`. Les utilisateurs PostgreSQL doivent être en concordance avec ceux créés lors de la dernière étape de l'installation du serveur (Création de 2 utilisateurs PostgreSQL).

Création de la base de données

— Création de la base de données et chargement des données initiales

```
sudo ./install_db.sh
```

— Vous pouvez consulter le log de cette installation de la base dans `log/install_db.log` et vérifier qu'aucune erreur n'est intervenue. **Attention, ce fichier sera supprimé** lors de l'exécution de `install_app.sh`

— Vous pouvez intégrer l'exemple des données SIG du Parc national des Ecrins des tables `layers.l_unites_geo`, `layers.l_aireadhesion` et `layers.l_secteurs`) :

```
export PGPASSWORD=monpassachanger; sudo psql -h localhost -U geonatuser -d␣
↪geonaturedb -f data/pne/data_sig_pne_2154.sql
```

Configuration de l'application

— Lancer le fichier d'installation et de préparation de la configuration de l'application

```
./install_app.sh
```

— Adapter le contenu du fichier `web/js/config.js`
  — Changer `mon-domaine.fr` par votre propre URL (wms_uri, host_uri)
— Adapter le contenu du fichier `web/js/configmap.js`
  — Renseigner votre clé API IGN Geoportail,
  — l'extent max de l'affichage cartographique, le centrage initial, le nombre maximum de niveau de zoom de la carte, la résolution maximale (en lien avec le paramètre précédent et le tableau `ign_resolutions`)
  — Renseigner le système de coordonnées et la bbox des coordonnées utilisables pour le positionnement du pointage par coordonnées fournies (GPS)
— Adapter le contenu du fichier `lib/sfGeonatureConfig.php`. Il indique notamment les identifiants de chaque protocoles, lots et sources de données.
— Pour tester, se connecter à l'application via http://mon-domaine.fr/geonature avec l'utilisateur et mot de passe : `admin / admin`
— Si vous souhaitez ajouter des données provenant d'autres protocoles non fournis avec GeoNature, créez leur chacun un schéma dans la BDD de GeoNature correspondant à la structure des données du protocole et ajoutez un trigger qui alimentera le schéma `synthese` existant à chaque fois qu'une donnée y est ajoutée ou modifiée. Pour cela vous pouvez vous appuyer sur les exemples existants dans les protocoles fournis (`contactfaune` par exemple).
— Si vous souhaitez ajouter des protocoles spécifiques dont les formulaires de saisie sont intégrés à votre GeoNature, référez-vous à la discussion https://github.com/PnEcrins/GeoNature/issues/54
— Si vous souhaitez désactiver certains programmes dans le critère de recherche COMMENT de l'application Synthèse, décochez leur champs `actif` dans la table `meta.bib_programmes` (https://github.com/PnEcrins/GeoNature/issues/67)
— Si vous souhaitez ne pas afficher tous les liens vers les formulaires de saisie des protocoles fournis par défaut avec GeoNature, décochez leur champs `actif` dans la table `synthese.bib_sources` (https://github.com/PnEcrins/GeoNature/issues/69)

Clé API IGN Geoportail

L'API IGN Geoportail permet d'afficher les fonds IGN dans GeoNature directement depuis le Geoportail.

Si vous êtes un établissement public, commandez une clé IGN de type : Licence géoservices IGN pour usage grand public - gratuite.

Selectionner les couches suivantes :
— Alticodage,
— WMTS-Géoportail - Cartes IGN,
— WMTS-Géoportail - Limites administratives
— WMTS-Géoportail - Orthophotographies
— WMTS-Géoportail - Parcelles cadastrales

Pour cela, il faut que vous disposiez d'un compte IGN pro (http://professionnels.ign.fr).

Une fois connecté au site :
— Aller dans "Nouvelle commande"

- Choisir “Géoservices IGN : Pour le web” dans la rubrique “LES GÉOSERVICES EN LIGNE”
- Cocher l’option “Pour un site internet grand public”
- Cocher l’option “Licence géoservices IGN pour usage grand public - gratuite”
- Saisir votre URL. Attention, l’adresse doit être précédée de `http://` (même si il s’agit d’une IP)
- Finir votre commande en selectionnant les couches utiles

Une fois que votre commande est prête, saisissez la valeur de la clé IGN dans le fichier `web/js/configmap.js`.

CHAPITRE 3

MISE A JOUR DE L'APPLICATION

Les différentes versions sont disponibles sur le Github du projet (https://github.com/PnEcrins/GeoNature/releases).

— Télécharger et extraire la version souhaitée dans un répertoire séparé (où `X.Y.Z` est à remplacer par le numéro de la version que vous installez).

```
cd /home/synthese/
wget https://github.com/PnEcrins/GeoNature/archive/X.Y.Z.zip
unzip X.Y.Z.zip
mv geonature geonature_old
mv GeoNature-X.Y.Z/ geonature
rm X.Y.Z.zip
```

— Copier les anciens fichiers de configuration et les comparer avec les nouveaux. Attention, si de nouveaux paramètres ont été ajoutés, ajoutez les dans ces fichiers.

```
cp geonature_old/wms/wms.map geonature/wms/wms.map
cp geonature_old/web/js/config.js geonature/web/js/config.js
cp geonature_old/web/js/configmap.js geonature/web/js/configmap.js
cp geonature_old/lib/sfGeonatureConfig.php geonature/lib/sfGeonatureConfig.php
cp geonature_old/config/databases.yml geonature/config/databases.yml
cd geonature
```

— Si vous l'avez personnalisé, récupérez votre bandeau de l'application

```
cp ../geonature_old/web/images/bandeau_geonature.jpg web/images/bandeau_
↪geonature.jpg
```

— Lire attentivement les notes de chaque version si il y a des spécificités (https://github.com/PnEcrins/GeoNature/releases). Suivre ces instructions avant de continuer la mise à jour.

— Si vous avez ajouté des protocoles spécifiques dans GeoNature (https://github.com/PnEcrins/GeoNature/issues/54), il vous faut les récupérer dans la nouvelle version.

Commencez par copier les modules Symfony correspondants dans le répertoire de la nouvelle version de GeoNature.

Il vous faut ensuite reporter les modifications réalisées dans les parties qui ne sont pas génériques (module Symfony `bibs`, le fichier de routing, la description de la BDD dans le fichier `config/doctrine/schema.yml` et l'appel des JS et CSS dans `apps/backend/modules/home/config/view.yml`).

CHAPITRE 4

WEB API

La web API GeoNature est un service web REST. Les informations sont transmises au format JSON ou GeoJSON.

Il est possible d'utiliser les méthodes HTTP `POST`, `PUT`, `DELETE`.

Les données manipuler par le service se trouvent dans la table `synthese.syntheseff`.

Toutes les méthodes http doivent être transmise avec les 2 paramètres `token` et `json` sauf la méthode `DELETE` qui peut fonctionner sans le paramètre `json`.

token :
- description : clé secrète. Si non conforme ou non fournie le service retourne une erreur.
- type : string
- obligatoire : oui
- valeur par défaut : none

json :
- description : paramètre au format JSON ou GeoJSON .
- type : string
- obligatoire : oui
- valeur par défaut : none

Insertion d'une observation

Méthode HTTP à utiliser : POST

URL : synthese/observation ?token=[votre_token]&json=[votre_geojson]

Paramètres :

token : requis (voir ci-dessus) json :

format : GeoJSON contenu :

type : requis. Seul le type "Feature" est implémenté. Ne pas utiliser de geoJSON avec le type FeatureCollection. geometry : requis.

type : requis. valeurs possible Point, LineString, Polygon type : string obligatoire : oui valeur par défaut : none

coordinates [requis.] type : tableau de valeur. Exemple [6.5, 44.85] pour un point. description : les coordonnées doivent être fournies avec le SRID 4326. Exemple [6.5, 44.85] pour un point. Voir les spécifications sur http://geojson.org/ obligatoire : oui

properties :

id_source : description : identifiant de la source de la donnée. Cet identifiant est une clé étrangère et doit être présent dans la table `synthese.bib_sources`. Au besoin, il faut le créer préalablement à tout enregistrement. Si l'id_source n'est pas fourni, l'enregistrement est créé avec l'id_source = 0. type : integer obligatoire : oui valeur par défaut : 0

id_fiche_source : description : identifiant correspondant à la clé primaire dans la table d'origine. Cet identifiant correspond au champ `id_fiche_source`. Couplé au `id_source` il forme une valeur unique correspondant à la clé primaire de l'enregistrement présent dans la table d'origine (=enregistrement présent dans une base distante). type : varchar(50) obligatoire : oui valeur par défaut : none

code_fiche_source : description : cet identifiant peut être utilisé pour tracer une source dont les enregistrements figurent dans plusieurs tables (concaténation de l'id de plusieurs tables). type : varchar(50) obligatoire : non valeur par défaut : none

id_organisme : description : identifiant de l'organisme propriétaire de la donnée. Cet identifiant est une clé étrangère et doit être présent dans la table `utilisateurs.bib_organismes`. type : integer obligatoire : oui valeur par défaut : none

id_protocole : description : identifiant du protocole ayant servi au recueil de la donnée. Cet identifiant est une clé étrangère et doit être présent dans la table `meta.t_protocoles`. type : integer obligatoire : oui valeur par défaut : none

id_precision : description : identifiant du niveau de précision à l'origine de la production de la donnée. Cet identifiant est une clé étrangère et doit être présent dans la table `meta.t_précisions`. type : integer obligatoire : oui valeur par défaut : none

id_lot : description : identifiant a un lot de données. Cet identifiant est une clé étrangère et doit être présent dans la table `meta.bib_lots` ; Les lots sont regroupés dans des programmes (`meta.bib_programmes`). Dans geoNature les programmes correspondent aux filtres de recherche dans le "comment ?" de l'interface. type : integer obligatoire : oui valeur par défaut : none

id_critere_synthese : description : identifiant du critère ayant permis l'observation. Cet identifiant est une clé étrangère et doit être présent dans la table `synthese.bib_criteres_synthese`. type : integer obligatoire : oui valeur par défaut : none

cd_nom : description : identifiant du taxon observé (voir taxref). Cet identifiant est une clé étrangère et doit être présent dans la table `taxonomie.taxref`. type : integer obligatoire : oui valeur par défaut : none

effectif_total : description : nombre d'individus observés. type : integer obligatoire : non valeur par défaut : none

insee : description : insee de la commune correspondant à la localisation de l'observation. La liste des communes est présente dans la table `layers.l_communes`. type : varchar(5) obligatoire : non valeur par défaut : none

dateobs : description : date de l'observation. format "aaaa-mm-jj". Exemple : 2015-07-28. type : date obligatoire : oui valeur par défaut : none

observateurs : description : le ou les observateur(s) de la donnée. Format libre (string) limité à 255 caractères. type : varchar(255) obligatoire : oui valeur par défaut : none

determinateur : description : le déterminateur de la donnée. Format libre (string) limité à 255 caractères. type : varchar(255) obligatoire : non valeur par défaut : none

altitude : description : altitude correspondant à la localisation de l'observation. type : integer obligatoire : non valeur par défaut : none

remarques : description : Champ libre permettant de fournir toutes information utile relative à l'observation. Pas de limite de taille. type : text obligatoire : non valeur par défaut : none

Données des tables liées : Lorsqu'il est fait référence au contenu des tables liées : "Cet identifiant est une clé étrangère et doit être présent dans la table ...". Ces données étant susceptibles d'être modifiées par l'administrateur de GeoNature, vous devez vous référer au contenu des tables liées en consultant le contenu des ces tables dans votre base de données de GeoNature.

Exemple de GeoJSON compatible pour une insertion de données :

```
{
    "type": "Feature"
    ,"geometry":
    {
        "type": "Point"
        ,"coordinates": [6.5, 44.85]
    }
    ,"properties": {
        "id_source" : 18
        ,"id_fiche_source" : "36513"
        ,"code_fiche_source" : "oc36513"
        ,"id_organisme" : 1
        ,"id_protocole" : 2
        ,"id_precision" : 1
        ,"dateobs" : "2015-11-30"
        ,"cd_nom" : 67111
        ,"effectif_total" : 10
        ,"insee" : "05006"
        ,"altitude" : 1000
        ,"observateurs" : "Paulo l'observateur"
        ,"determinateur" : "Paulo le déterminateur"
        ,"remarques" : "une remarque de test"
        ,"id_lot" : 2
        ,"id_critere_synthese" : 2
    }
}
```

Return :

format [JSON] success : boolean - true ou false message : string - Information concernant l'erreur rencontrée. id_synthese : integer - Identifiant nouvellement créé dans la table synthese.syntheseff. Peut constituer un lien entre la donnée d'origine et la donnée enregistrée dans geoNature. id_source : integer - Identifiant de la source référençant la donnée nouvellement créé dans la table synthese.syntheseff id_fiche_source : integer - Clé primaire dans la table d'origine de la donnée nouvellement créé dans la table synthese.syntheseff. Peut constituer un lien entre la donnée d'origine et la données enregistrée dans geoNature.

Modification d'une observation

Méthode HTTP à utiliser : PUT

URL : synthese/observation/[id_synthese] ?token=[votre_token]&json=[votre_geojson]

Deux manières de modifier un enregistrement :

1/ en fournissant le `id_synthese` dans l'url. Par exemple synthese/observation/68 ?token=mon ;token !hyper#complexe

2/ en fournissant le `id_source` et le `id_fiche_source` dans le paramètre `json` (voir ci-dessous). Dans ce cas, l'url ne contient pas l'id_synthese –> synthese/observation ?token=mon ;token !hyper#complexe

param : id_synthese : optionnel token : requis (voir ci-dessus) json :

format : GeoJSON contenu : Les informations de l'objet `properties` ne doivent pas forcement être toutes fournies, de même que les informations concernant l'objet `geometry`

type : optionnel. Requis avec la valeur "Feature" et l'objet `geometry` si la géometrie doit être mise à jour. geometry : optionnel. Requis avec l'objet `type` si la géometrie doit être mise à jour.

type : requis. valeurs possible Point, LineString, Polygon type : string obligatoire : oui valeur par défaut : none

coordinates [requis.] type : tableau de valeur. Exemple [6.5, 44.85] pour un point. description : les coordonnées doivent être fournies avec le SRID 4326. Exemple [6.5, 44.85] pour un point. Voir les spécifications sur http://geojson.org/ obligatoire : oui

properties [requis]

id_source : description : identifiant de la source de la donnée. Cet identifiant doit être présent dans la table `synthese.bib_sources`. type : varchar(50) obligatoire : optionnel (si non fourni, fournir le id_synthese dans l'url) valeur par défaut : 0

id_fiche_source : description : identifiant correspondant à la clé primaire dans la table d'origine. Cet identifiant correspond au champ `id_fiche_source`. Couplé au `id_source` il forme une valeur unique correspondant à la clé primaire de l'enregistrement présent dans la table d'origine (=enregistrement présent dans une base distante). type : varchar(50) obligatoire : optionnel (si non fourni, fournir le id_synthese dans l'url) valeur par défaut : none

code_fiche_source : description : cet identifiant peut être utilisé pour tracer une source dont les enregistrements figure dans plusieurs tables (concaténation de l'id de plusieurs tables). type : varchar(50) obligatoire : non valeur par défaut : none

id_organisme : description : identifiant de l'organisme propriétaire de la donnée. Cet identifiant est une clé étrangère et doit être présent dans la table `utilisateurs.bib_organismes`. type : integer obligatoire : non valeur par défaut : none

id_protocole : description : identifiant du protocole ayant servi au recueil de la donnée. Cet identifiant est une clé étrangère et doit être présent dans la table `meta.t_protocoles`. type : integer obligatoire : non valeur par défaut : none

id_precision : description : identifiant du niveau de précision à l'origine de la production de la donnée. Cet identifiant est une clé étrangère et doit être présent dans la table `meta.t_précisions`. type : integer obligatoire : non valeur par défaut : none

id_lot : description : identifiant a un lot de données. Cet identifiant est une clé étrangère et doit être présent dans la table `meta.bib_lots` ; Les lots sont regroupés dans des programmes (`meta.bib_programmes`). Dans geoNature les programmes correspondent aux filtres de recherche dans le "comment ?" de l'interface. type : integer obligatoire : non valeur par défaut : none

id_critere_synthese : description : identifiant du critère ayant permis l'observation. Cet identifiant est une clé étrangère et doit être présent dans la table `synthese.bib_criteres_synthese`. type : integer obligatoire : non valeur par défaut : none

cd_nom : description : identifiant du taxon observé (voir taxref). Cet identifiant est une clé étrangère et doit être présent dans la table `taxonomie.taxref`. type : integer obligatoire : non valeur par défaut : none

effectif_total : description : nombre d'individus observés. type : integer obligatoire : non valeur par défaut : none

insee : description : insee de la commune correspondant à la localisation de l'observation. La liste des communes est présente dans la table `layers.l_communes`. type : varchar(5) obligatoire : non valeur par défaut : none

dateobs : description : date de l'observation. format "aaaa-mm-jj". Exemple : 2015-07-28. type : date obligatoire : non valeur par défaut : none

observateurs : description : le ou les observateur(s) de la donnée. Format libre (string) limité à 255 caractères. type : varchar(255) obligatoire : non valeur par défaut : none

determinateur : description : le déterminateur de la donnée. Format libre (string) limité à 255 caractères. type : varchar(255) obligatoire : non valeur par défaut : none

altitude : description : altitude correspondant à la localisation de l'observation. type : integer obligatoire : non valeur par défaut : none

remarques : description : Champ libre permettant de fournir toutes information utile relative à l'observation. Pas de limite de taille. type : text obligatoire : non valeur par défaut : none

Données des tables liées : Lorsqu'il est fait référence au contenu des tables liées : -Cet identifiant est une clé étrangère et doit être présent dans la table "schéma.table"-. Ces données étant susceptibles d'être modifiées par l'administrateur de GeoNature, vous devez vous référer au contenu des tables liées en consultant le contenu des ces tables dans la base de données de GeoNature.

Exemples de GeoJSON compatible pour une modification de données :

```
{
    "type": "Feature"
    ,"properties": {
        "id_synthese" : 53
        ,"dateobs" : "2014-10-27"
    }
}

ou

{
    "type": "Feature"
    ,"properties": {
        "id_source" : 0
        ,"id_fiche_source" : "36513"
        ,"effectif_total" : 12
        ,"altitude" : 1020
        ,"observateurs" : "Gaston l'observateur"
    }
}

ou

{
    "type": "Feature"
```

```
    ,"geometry":
    {
        "type": "Point"
        ,"coordinates": [6.58217, 44.84799]
    }
    ,"properties": {
        "id_source" : 18
        ,"id_fiche_source":"99"
    }
}
```

Return :

format [JSON] success : bool - true ou false message : string - Information concernant l'erreur rencontrée. id_synthese : integer - Identifiant nouvellement créé dans la table synthese.syntheseff. Peut constituer un lien entre la donnée d'origine et la données enregistrée dans geoNature. id_source : integer - Identifiant de la source référençant la donnée nouvellement créé dans la table synthese.syntheseff id_fiche_source : integer - Clé primaire dans la table d'origine de la donnée nouvellement créé dans la table synthese.syntheseff. Peut constituer un lien entre la donnée d'origine et la données enregistrée dans geoNature.

Test :

avec curl : curl -i -X PUT –header 'Accept :application/json' 'http ://92.222.107.92/geonature/synthese/observation/68 ?token=mon ;token !hyper#complexe' -d 'json={"type" : "Feature","properties" : {"dateobs" : "2013-01-18"}}' curl -i -X PUT –header 'Accept :application/json' 'http ://92.222.107.92/geonature/synthese/observation ?token=mon ;token !hyper#complexe' -d 'json={"type" : "Feature","properties" : {"id_source" : 18, "id_fiche_source" :"99", "dateobs" : "2013-01-18"}}'

Suppression d'une observation

Méthode HTTP à utiliser : DELETE

URL : synthese/observation/[id_synthese] ?token=[votre_token]&json=[votre_json]

Deux manières de supprimer un enregistrement :

1/ en fournissant le `id_synthese` dans l'url. Par exemple synthese/observation/68 ?token=mon ;token !hyper#complexe

2/ en fournissant le `id_source` et le `id_fiche_source` dans le paramètre `json` (voir ci-dessous). Dans ce cas, l'url ne contient pas l'id_synthese –> synthese/observation ?token=mon ;token !hyper#complexe

param : id_synthese : oui si le paramètre `json` n'est pas fourni token : requis (voir ci-dessus) json :

format : JSON contenu :

id_source : description : identifiant de la source de la donnée. Cet identifiant doit être présent dans la table `synthese.bib_sources`. type : varchar(50) obligatoire : oui si le paramètre `id_synthese` n'est pas fourni valeur par défaut : 0

id_fiche_source : description : identifiant correspondant à la clé primaire dans la table d'origine. Cet identifiant correspond au champ `id_fiche_source`. Couplé au `id_source` il forme une valeur unique correspondant à la clé primaire de l'enregistrement présent dans la table d'origine (=enregistrement présent dans une base distante). type : varchar(50) obligatoire : oui si le paramètre `id_synthese` n'est pas fourni valeur par défaut : none

Return :

format [JSON] success : bool - true ou false message : string - Information concernant l'erreur rencontrée id_synthese : integer - Identifiant de la donnée supprimée dans la table synthese.syntheseff. id_source : integer - Identifiant de la source référençant la donnée supprimée dans la table synthese.syntheseff id_fiche_source : integer - Clé primaire dans la table d'origine de la donnée supprimée dans la table synthese.syntheseff.

Test :

avec CURL : curl -i -X DELETE –header 'Accept :application/json' 'http ://xx.xxx.xxx.xx/geonature/synthese/observation/68 ?token=mon ;token !hyper#complexe' curl -i -X DELETE –header 'Accept :application/json' 'http ://xx.xxx.xxx.xx/geonature/synthese/observation ?token=mon ;token !hyper#complexe' -d 'json={"id_source" : 18, "id_fiche_source" :"99"}'

CHAPITRE 5

MODULE D'EXPORT

Ce module permet d'exporter les données de geoNature.

Il suffit de créer des vues personnalisées dans la base de données GeoNature et de déclarer ces vues dans la variable de configuration du module. Le module créera automatiquement un lien par vue renvoyant les données de la vue **avec tous les champs de la vue**.

Pour des questions de performances sur de gros volumes de données, le fichier est compressé au format zip.

Seuls les utilisateurs de geonature déclarés dans la configuration du module peuvent voir les liens et exporter les données mises à disposition. Vous devez connaitre l'`id_role` des utilisateurs auxquels vous souhaitez attribuer des droits d'export (voir la table `utilisateurs.t_roles`)

Pré-requis

- Commencer par créer la ou les vues renvoyant les données que vous souhaitez mettre à disposition des utilisateurs. Le contenu et le schéma de la vue sont libres.
- Le ou les utilisateurs doivent disposer de droits d'accès à GeoNature. A minima des droits de consultation : `id_droit = 1` dans `utilisateurs.cor role_droit_application`.

Configuration

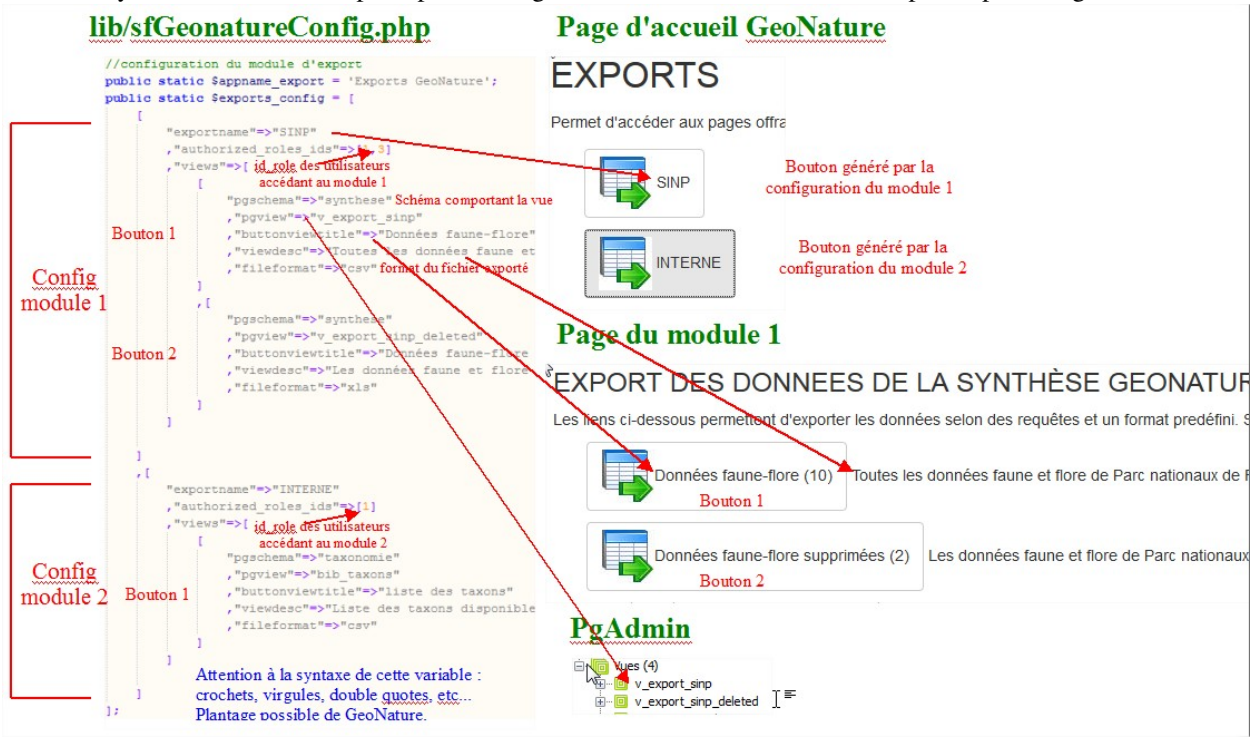
Il est possible de configurer plusieurs modules d'export. A chaque module correspond une page dédiée, automatiquement générée par GeoNature. Le lien (bouton) vers chacune de ces pages est affiché sur la page d'accueil de GeoNature.

Pour qu'un utilisateur puisse voir le ou les liens vers les pages d'export qui le concerne, son `id_role` doit figurer dans le tableau `authorized_roles_ids` du module.

- Editer le fichier `lib/sfGeonatureConfig.php` et recherche la partie `//configuration du module d'export`. Voir `lib/sfGeonatureConfig.php.sample` si besoin.
- La variable générale `$appname_export` défini le titre de la ou des pages du module d'export.
- La variable `$exports_config` contient toute la configuration de la ou des pages du module d'export.

Le schéma ci-dessous résume le rôle de chacun des paramètres de configuration.

- Seuls les formats csv et xls sont possibles. Si un autre format est spécifié, le format csv sera retourné par défaut.
- Le séparateur csv est le ‘ ;’. Le séparateur xls est la tabulation ‘\t’.
- Attention, la syntaxe PHP de cette variable `$exports_config` ne doit pas comporter d’erreur sous peine de plantage. Il s’agit d’un tableau multidimensionnel PHP, veillez à respecter la syntaxe des tableaux [] et objets “key”=>”value”, de les séparer par une virgule et de bien terminer la variable par un point virgule.



CHAPITRE 6

SAUVEGARDES

Sur le serveur, il faut produire des fichiers de backup des bases de données postgreSQL et des répertoires contenant les scripts et les fichiers des applications (médias et configuration notamment).

Pour cela, 2 scripts SH (`gsql-backup.sh` et `internet-backup.sh`) vous sont proposés (à adapter à votre contexte). Ils vont être éxécutés automatiquement et régulièrement grace à des taches cron et vont générés des fichiers `tar.gz` contenant les sauvegardes des bases de données et des fichiers des applications.

Ces fichiers sont ensuite copiés en FTP sur la partie dédiée à cela par l'hébergeur (optionnel) grace à `ncftp`.

Enfin rsync va permettre de récupérer régulièrement ces fichiers sur un serveur local.

Il est recommandé d'exécuter les actions qui suivent avec l'utilisateur `root`.

PostgreSQL

Les sauvegardes sont faites toutes les nuits et conservées un mois (31 fichiers). Une sauvegarde mensuelle est conservée un an (12 fichiers). Cette politique de sauvegarde peut-être adaptée.

Voir le script pgsql-backup.sh qui peut être placé dans `/usr/local/bin/`.

Ce script sauvegarde toutes les bases de données. Dans l'exemple fourni, pour les BDD de GeoNature et de UsersHub, il ne sauvegarde la BDD complete que le premier du mois. Les autres jours, il ne sauvegarde que les schémas "vivants". (option -n). Cette politique de sauvegarde peut également être adaptée.

Ce script comporte aussi une copie des fichiers de backup vers le serveur de backup-ftp de l'hebergeur (OVH dans notre cas). Pour l'utiliser, votre serveur doit disposer de ce service et il faut installer `ncftp`. Sinon commenter ou retirer les lignes concernées (`ncftpput`).

Ce fichier doit être éxecutable :

```
chmod +x /usr/local/bin/pgsql-backup.sh
```

Attention, l'utilisateur `postgres` doit être le propriétaire de ce fichier ou disposer des droits d'exécution sur celui-ci.

```
chown postgres /usr/local/bin/pgsql-backup.sh
```

Scripts et fichiers des applications

Les sauvegardes sont faites toutes les nuits mais vu la taille potentiellement importante de ce fichier, il est écrasé chaque nuit par la nouvelle sauvegarde. Cette politique de sauvegarde peut-être adaptée.

Voir le script internet-backup.sh qui peut être également placé dans `/usr/local/bin/`.

Ce script comporte également une copie des fichiers de backup vers le serveur de backup-ftp de l'hébergeur. Pour l'utiliser votre serveur doit disposer de ce service et il faut installer `ncftp`. Sinon commenter ou retirer les lignes concernées (`ncftpput`).

Ce fichier doit être executable :

```
chmod +x /usr/local/bin/internet-backup.sh
```

Automatisation des sauvegardes sur le serveur

Ajouter ceci à la fin du crontab de l'utilisateur `postgres` (`crontab -e` pour éditer le crontab).

```
# sauvegarde des bases de donnees postgres à 3h45 du matin
45 3 * * * /usr/local/bin/pgsql-backup.sh
```

Ajouter ceci à la fin du crontab de l'utilisateur `root`

```
# sauvegarde des applications internet
# vers le ftp ovh à 1h15 du matin
15 1 * * * /usr/local/bin/internet-backup.sh
```

Installation et configuration de RSYNC

1. Mise en place de rsync sur le serveur

Voir la documentation rsync_server.md pour la configuration de rsync.

Il existe également plusieurs ressources en ligne pour configurer rsync coté serveur. Demander à Lilo ;-)

2. Récupération des backups sur une machine locale

— Linux

Rsync client doit être présent sur la machine qui récupère les backups. Il y a plusieurs manières de configurer rsync (de façon incrémentielle ou non).

Voir un exemple avec le script rsync_client.sh.

Ce script récupère les fichiers des modules rsync `geonature` et `usershub` configuré sur le daemon rsync (voir documentation rsync_server.md) du serveur et les place dans des répertoires locaux, par exemple : `/home/mylocaluser/svg_geonature/`

— Windows

La partie Windows n'est utile que pour remonter des backups sur une machine Windows locale.

Le script est fourni à titre d'exemple. Usage ancien, non testé récemment. Fonctionnement non garanti.

CHAPITRE 7

INSTALLATION GLOBALE

Cette documentation permet une installation rapide et simplifiée de GeoNature et de ses applications liées : UsersHub, TaxHub et GeoNature-atlas. Pour plus d'informations, référez-vous aux documentations plus détaillées de chaque projet.

Les scripts proposés installent l'environnement logiciel du serveur, téléchargent les applications sur leur dépots github, les installent et les configurent par défaut.

Pré-requis

- Un serveur Debian 8 (Ubuntu 16.04 LTS devrait fonctionner également - non testé)
- Une clé IGN pour l'API Geoportail valide pour le domaine sur lequel votre serveur répond

Installation

notes Votre utilisateur linux doit disposer des droits administrateur avec sudo. Voir https://www.privateinternetaccess.com/forum/discussion/18063/debian-8-1-0-jessie-sudo-fix-not-installed-by-default

Après installation de l'OS avec OpenSSH server, créer un utilisateur linux (nommé `geonatureadmin` dans notre cas) pour ne pas travailler avec le super-utilisateur `root`. Donnez-lui des droits sudo pour qu'il puisse faire les taches d'administration :

```
adduser geonatureadmin sudo
```

L'ajouter aussi aux groupes `www-data` et `root`

```
usermod -g www-data geonatureadmin
usermod -a -G root geonatureadmin
```

Reconnectez-vous au serveur en SSH avec votre utilisateur (`geonatureadmin` dans notre cas) pour ne pas travailler avec le super-utilisateur `root`.

Adapter votre fichier de sources de paquets `/etc/apt/sources.list`.

Pour cela, modifier le fichier

```
sudo nano /etc/apt/sources.list
```

Supprimer tout son contenu pour le remplacer par cet exemple permettant un bon fonctionnement sur Debian 8 :

```
deb http://httpredir.debian.org/debian jessie main contrib non-free
deb-src http://httpredir.debian.org/debian jessie main contrib non-free

deb http://httpredir.debian.org/debian jessie-updates main contrib non-free
deb-src http://httpredir.debian.org/debian jessie-updates main contrib non-free

deb http://security.debian.org/ jessie/updates main contrib non-free
deb-src http://security.debian.org/ jessie/updates main contrib non-free

#Backports
deb http://http.debian.net/debian jessie-backports main contrib non-free
```

Enregistrer et fermer le fichier.

Placez vous dans le répertoire `home` de votre utilisateur et entrez les commandes suivantes :

```
sudo apt-get update
sudo apt-get install -y ca-certificates
```

Récupérer les scripts d'installation (`X.Y.Z` à remplacer par le numéro de la dernière version stable de GeoNature) :

```
wget https://raw.githubusercontent.com/PnEcrins/GeoNature/X.Y.Z/docs/install_all/
↪install_all.ini
wget https://raw.githubusercontent.com/PnEcrins/GeoNature/X.Y.Z/docs/install_all/
↪install_all.sh
chmod +x install_all.sh
```

Mettez à jour le fichier `install_all.ini` avec vos informations. Attention, ne lancez pas les fichiers .sh tant que vous n'avez pas totalement complété ce fichier.

TODO : détailler la procédure pour l'atlas avec :

- install avec données exemple
- mettre à jour les shapes territoire
- relancer le install.db de l'atlas.

Lancez ensuite l'installation des applications :

```
./install_all.sh
```

Le mot de passe sudo vous sera demandé a plusieurs reprise. 2 fichiers de la configuration de Taxhub seront affichés. Il vous est possible de les modifier mais vous pouvez laisser les valeurs par défaut. Pour enregistrer `ctrl + o`, pour sortir et poursuivre l'installation : `ctrl + x`.

Vous devez pouvoir vous connecter à vos applications avec les adresses (adaptez mondomaine.fr à votre nom de domaine ou avec votre adresse IP) :

- http://mondomaine.fr/usershub
- http://mondomaine.fr/geonature
- http://mondomaine.fr/taxhub
- http://mondomaine.fr/atlas

Les 3 premières applications demandent une authentification.

L'utilisateur `admin` avec le mot de passe `admin` est disponible par défaut avec des droits administrateur sur toutes les applications. Vous devez utiliser UsersHub pour gérer d'autres utilisateurs.

L'installation des bases de données est loguée dans le répertoire `log` des applications : `log/install_db.log`.

notes L'application GeoNature-atlas est livrée avec des données exemples. Une fois l'installation de l'atlas terminée, vous devez l'adapter à votre territoire.

- Remplacez les shapes `territoire.shp` et `communes.shp` dans `data/ref` avec celles de votre territoire.
- Relancer le sript `install.db`.

CHAPITRE 8

EXEMPLE D'INSTALLATION GLOBALE

Installation de GeoNature, TaxHub, UsersHub et GeoNature-atlas dans un PNR utilisant SERENA

Décembre 2016 / Camille Monchicourt (Parc national des Ecrins)

Introduction

Ce document décrit le déploiement de tout l'environnement GeoNature à l'aide du script `install_all`.

Il décrit aussi comment l'outil a été paramétré et comment les données historiques du PNR saisies dans SERENA ont été intégrées dans GeoNature.

Il se base sur les versions 1.8.0 de GeoNature, 1.1.1 de TaxHub, 1.2.0 de UsersHub et 1.2.2 de GeoNature-atlas.

Vérifiez les évolutions de ces applications et de leurs procédures si vous utilisez des versions plus récentes.

Suivre la documentation de référence du script de déploiement global : https://github.com/PnEcrins/GeoNature/tree/master/docs/install_all

Voir aussi http://geonature.fr et la documentation de chaque projet pour des informations complémentaires et plus détaillées sur chaque outil.

On travaille ici sur un serveur Debian 8 avec seulement OpenSSH d'installé. Le script se charge d'installer tous les autres éléments sur le serveur.

Installation

On commence la procédure en se connectant au serveur en SSH avec l'utilisateur linux ROOT.

— Mettre à jour la liste des dépôts Linux

```
apt-get update
```

— Installer sudo

```
apt-get install -y sudo ca-certificates
```

— Créer un utilisateur linux (nommé `geonatureadmin` dans notre cas) pour ne pas travailler en ROOT (en lui donnant les droits sudo)

```
adduser geonatureadmin sudo
```

— L'ajouter aussi aux groupes www-data et root

```
usermod -g www-data geonatureadmin
usermod -a -G root geonatureadmin
```

— Récupérer les scripts d'installation (X.Y.Z à remplacer par le numéro de la dernière version stable de GeoNature) :

```
wget https://raw.githubusercontent.com/PnEcrins/GeoNature/X.Y.Z/docs/install_all/
↪install_all.ini
wget https://raw.githubusercontent.com/PnEcrins/GeoNature/X.Y.Z/docs/install_all/
↪install_all.sh
```

— Changer les droits du fichier d'installation pour pouvoir l'éxecuter

```
chmod +x install_all.sh
```

On se reconnecte avec le nouvel utilisateur pour ne pas faire l'installation en ROOT.

On ne se connectera plus en ROOT. Si besoin d'executer des commandes avec des droits d'administrateur, on les précède de `sudo`.

On peut d'ailleurs renforcer la sécurité du serveur en bloquant la connexion SSH au serveur avec ROOT.

Voir https://docs.ovh.com/pages/releaseview.action?pageId=18121864 pour plus d'informations sur le sécurisation du serveur.

Renseigner le fichier de configuration `install_all.ini` (avec WinSCP, clic droit puis Editer, puis enregistrer le fichier une fois modifié)

notes Pour la clé IGN, voir http://geonature.readthedocs.io/fr/latest/installation.html#cle-api-ign-geoportail. Il est conseillé de la créer avant de lancer l'installation. Sinon vous devrez modifier plus tard la clé IGN dans la configuration de GeoNature et de GeoNature-atlas. Si vous accédez aux applications par une IP et non un domaine, il faut quand même la créer en mode referer avec http:// devant l'adresse IP.

— Lancer l'installation

```
./install_all.sh
```

Connexion aux applications avec les données tests par défaut

Tester les applications dans un navigateur web avec l'utilisateur par défaut (admin / admin) :

— http://ip/geonature
— http://ip/usershub
— http://ip/taxhub
— http://ip/atlas

Si une application renvoie une INTERNAL ERROR, les logs d'Apache peuvent fournir des éléments sur l'erreur. Pour les consulter :

```
sudo tail -f /var/log/apache2/error.log
```

Dans UsersHub (voir documentation http://usershub.readthedocs.io) :

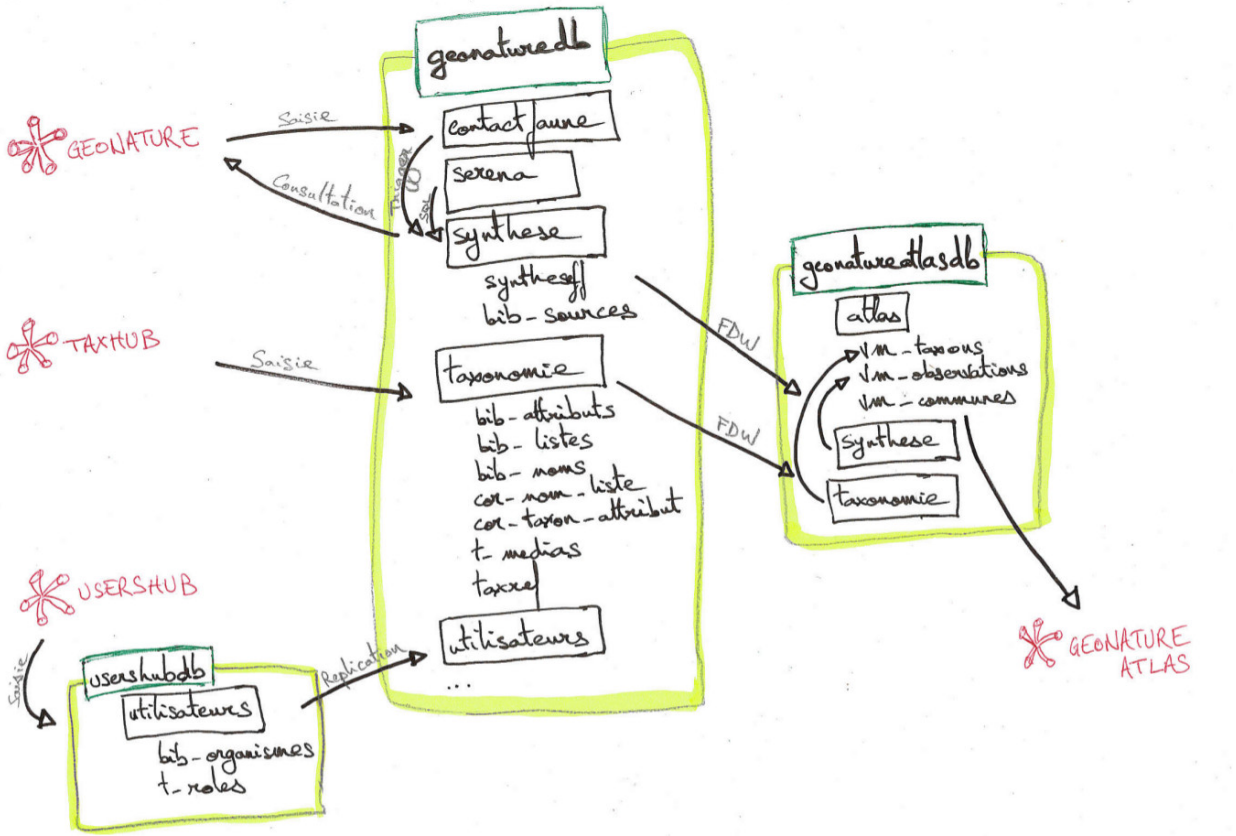
- On ajoute quelques utilisateurs
- On les met dans les bons groupes
- On vérfie que les groupes sont dans les listes d'observateurs souhaitées
- On met les droits aux groupes dans les applications
- On modifie les organismes

Si vous souhaitez continuer à travailler avec les quelques données tests présentes dans l'installation par défaut, celles-ci sont associées par défaut à l'organisme `99`. Or par défaut, l'atlas n'affiche que les données de l'organisme `2`.

Mettez donc les données en cohérence pour qu'elles apparaissent dans l'atlas. Pour cela on va modifier l'organisme associés à ces données dans les protocoles sources. Dans les données Contact Faune et Flore Station, on change les `id_organismes` des données tests pour être en cohérence avec la table `utilisateurs.bib_organismes`.

Rafraichir les VM de l'atlas pour faire apparaitre les modifications faites dans GeoNature et/ou TaxHub (https://github.com/PnEcrins/GeoNature-atlas/blob/master/docs/vues_materialisees_maj.rst)

Fonctionnement général



USERSHUB

- L'application UsersHub dispose de sa propre BDD `usershubdb`. Chaque modification dans cette base de données faites avec UsersHub est répliquée dans les BDD filles utilisant son schema `utilisateurs`.
- `bib_organismes` contient la liste des organismes. `t_roles` la listes des utilisateurs et groupes. `cor_roles` permet d'associer des utilisateurs à des groupes.
- Il est conseillé de donner des droits dans des applications à des groupes plutôt qu'à des utilisateurs

TAXHUB

- L'applications TaxHub permet de gérer le contenu du schéma `taxonomie` de `geonaturedb`.
- Celui-ci contient le référentiel taxref complet mais il permet d'y selectionner les taxons utilisés, d'y ajouter des informations et de créer des listes de taxons pour les différentes applications
- `bib_noms` contient la liste de tous les taxons utilisés par la structure. Cette table s'alimente dans TaxHub en ajoutant des taxons depuis l'onglet TaxRef.
- `bib_attributs` permet d'associer des informations complémentaires à chaque taxon. Chaque structure peut créer autant d'attributs qu'elle souhaite.
- Certains attributs sont obligatoires au fonctionnement de GeoNature. `Patrimonialité` et `protégé` sont requis pour la synthese
- Les attributs `Description`, `Commentaire`, `Milieu` et `Chorologie` sont utilisés par l'atlas
- `cor_taxon_attribut` permet de stocker les valeurs des attributs pour chaque taxon
- `bib_listes` et `cor_nom_liste` permettent de créer des listes de taxons pour les différents protocoles. Il est important de mettre chaque taxon dans les bonnes listes pour qu'il soit possible de les saisir dans les protocoles correspondants
- `t_medias` contient les medias locaux (chemin) ou distants (URL) de chaque taxon pour l'atlas. Il peut s'agir de photos, audios, vidéos ou d'articles

GEONATURE

- Chaque protocole dispose de son propre schéma correspondant à son modèle de données.
- Il est possible d'ajouter autant de schémas que souhaité
- Certains schémas liés à des protocoles intégrés sont fournis (`contactfaune`, `contactflore`, `contactinv`, `florestation`...).
- A chaque fois qu'une donnée est saisie dans un de ces protocoles, un trigger alimente automatiquement la synthèse de GeoNature
- Pour chaque donnée, on renseigne une source, un lot, un programme et un protocole

GEONATURE-ATLAS

- L'application GeoNature-atlas dispose sa propre BDD `geonatureatlasdb` pour pouvoir être installé sur un autre serveur
- GeoNature-atlas se base uniquement sur des vues matérialisées pour pouvoir être totalement indépendante de GeoNature et pouvoir être alimenté par n'importe qu'elle autre source de données
- Dans notre cas GeoNature-atlas est alimenté par les données présentes dans la synthèse de GeoNature
- Pour disposer des données de la synthèse ainsi que des informations taxonomiques sans les répliquer, un mécanisme de Foreign Data Wrapper (FDW) est utilisé.
- Les vues matérialisées nécessaires à GeoNature-atlas s'appuient dans notre cas sur les tables filles utilisant ces FDW
- Il est nécessaire de rafraichir les vues matérialisées pour que GeoNature-atlas prenne en compte tout changement dans la synthèse ou la taxonomie de `geonaturedb`.
- Ce rafraichissement peut-être réalisé manuellement ou automatiquement

Consultez le MCD complet pour en savoir plus : https://github.com/PnEcrins/GeoNature/blob/develop/docs/2017-01-mcd_geonaturedb_1.8.2.png

Intégration des données existantes dans GeoNature

On va maintenant copier les données de SERENA dans la BDD de GeoNature.

Cela pour les stocker et y accéder sous leur forme brute mais aussi pour les intégrer dans la synthèse de GeoNature et dans l'atlas.

Dans notre cas, les données ont été copiées de la BDD Access de SERENA vers une BDD PostGIS locale dans un schéma spécifique.

La structure de ce schéma ainsi que les données ont été exportées dans 2 fichiers SQL séparés.

Ces fichiers sont copiés sur le serveur puis éxécutés dans la BDD `geonaturedb`.

- Création du schéma `serena_affo_pnr` et de ses tables qui accueilleront les données SERENA brutes

```
export PGPASSWORD=MONPASSACHANGER;psql -d geonaturedb -U geonatuser -h localhost␣
↪-f serena_affo_pnr_schema.sql  &>> geonature/log/install_db_serena_1.log
```

— Intégration des données SERENA brutes dans le schéma `serena_affo_pnr`

```
export PGPASSWORD=MONPASSACHANGER;psql -d geonaturedb -U geonatuser -h localhost␣
↪-f serena_affo_pnr_donnees.sql  &>> geonature/log/install_db_serena_2.log
```

— Idéalement on devrait créer une vue matérialisée (VM) basée sur ces données mais par manque de temps on va repartir de la table à plat contenant les geométries générées par le PNR.

```
export PGPASSWORD=MONPASSACHANGER;psql -d geonaturedb -U geonatuser -h localhost␣
↪-f serena_affo_pnr_vm_schema.sql  &>> geonature/log/install_db_serena_6.log
export PGPASSWORD=MONPASSACHANGER;psql -d geonaturedb -U geonatuser -h localhost␣
↪-f serena_affo_pnr_vm_donnees.sql  &>> geonature/log/install_db_serena_7.log
```

C'est cette table que l'on utilisera pour remplir la table `synthese.syntheseff`.

Les éléments suivants sont éxécutés en SQL avec l'utilisateur propriétaire des BDD (`user_pg`), en utilisant pgAdmin.

— Mettre à jour de la couche des communes de GeoNature (à partir des départements dans notre cas) :

```
UPDATE layers.l_communes SET organisme = true
WHERE inseedep IN ('14','50','53','61','72')
```

— Pour alléger la BDD et les traitements, on supprime toutes les communes en dehors de ces 5 départements :

```
DELETE FROM layers.l_communes
WHERE inseedep NOT IN ('14','50','53','61','72')
```

On va maintenant préparer le schéma `taxonomie` pour y intégrer les taxons observés par le PNR et les mettre dans les bonnes listes (voir documentation de TaxHub)

Vider la table `taxonomie.bib_noms` et ses tables liées pour supprimer les taxons exemples.

Idem avec les autres tables de geonaturedb qui contiennent quelques données exemple (`synthese.syntheseff`, `contactfaune.t_fiches_cf`,...).

— Peupler `taxonomie.bib_noms` (liste des espèces du territoire) à partir des espèces observées dans les observations SERENA :

```
INSERT INTO taxonomie.bib_noms (cd_nom,cd_ref,nom_francais)
SELECT DISTINCT rnf.taxon_mnhn_id, t.cd_ref, t.nom_vern FROM serena_affo_pnr_vm.
↪rnf_obse_geom rnf
JOIN taxonomie.taxref t ON t.cd_nom = rnf.taxon_mnhn_id
-- pour éviter les doublons si des espèces sont déjà présentes dans bib_noms :
LEFT JOIN taxonomie.bib_noms tb ON tb.cd_nom = rnf.cd_nom
WHERE tb.cd_nom IS NULL
```

Attention il semblerait que 39 taxons n'aient pas été intégrés, certainement car ils n'ont pas d'identifiant taxref ? A vérifier.

Cela aura peut-être d'autres conséquences sur l'intégration des données dans la synthèse. A vérifier.

Vérifier aussi la version de TaxRef utilisée pour les données sources et la version utilisée par TaxHub pour être en cohérence.

— Pour ne pas avoir de noms français vides dans `taxonomie.bib_noms` :

```
UPDATE taxonomie.bib_noms SET nom_francais = '' WHERE nom_francais IS NULL
```

— Renseigner `taxonomie.cor_taxon_attribut` pour pouvoir saisir ces taxons (Saisie = oui)

```
INSERT INTO taxonomie.cor_taxon_attribut (id_attribut,valeur_attribut,cd_ref)
SELECT 3,'oui',n.cd_ref FROM taxonomie.bib_noms n
GROUP BY n.cd_ref;
```

notes Depuis la version 1.8.3, cet attribut n'est plus utilisé et a été remplacé par une liste : https://github.com/PnEcrins/GeoNature/releases/tag/1.8.3

— Mettre tous les taxons à non protégés et non patrimonial par défaut (dans `taxonomie.cor_taxon_attribut`) car cette info est attendue par la synthèse. A retravailler au cas par cas ou à partir des infos présentes dans TaxRef

```
INSERT INTO taxonomie.cor_taxon_attribut (id_attribut,valeur_attribut,cd_ref)
SELECT 1,'non',n.cd_ref FROM taxonomie.bib_noms n
GROUP BY n.cd_ref;
```

```
INSERT INTO taxonomie.cor_taxon_attribut (id_attribut,valeur_attribut,cd_ref)
SELECT 2,'non',n.cd_ref FROM taxonomie.bib_noms n
GROUP BY n.cd_ref;
```

— Peupler les listes de taxons (`taxonomie.cor_nom_liste` faisant référence à `taxonomie.bib_listes`) en se basant sur les groupes INPN. A voir si les infos des groupes dans TaxRef sont fiables et complètes. A adapter selon vos données et taxons observés.

```
INSERT INTO taxonomie.cor_nom_liste (id_liste,id_nom)
SELECT 1,n.id_nom FROM taxonomie.bib_noms n
JOIN taxonomie.taxref t ON t.cd_nom = n.cd_nom
where t.group2_inpn = 'Amphibiens';
```

```
INSERT INTO taxonomie.cor_nom_liste (id_liste,id_nom)
SELECT 11,n.id_nom FROM taxonomie.bib_noms n
JOIN taxonomie.taxref t ON t.cd_nom = n.cd_nom
where t.group2_inpn = 'Mammifères';
```

```
INSERT INTO taxonomie.cor_nom_liste (id_liste,id_nom)
SELECT 12,n.id_nom FROM taxonomie.bib_noms n
JOIN taxonomie.taxref t ON t.cd_nom = n.cd_nom
where t.group2_inpn = 'Oiseaux';
```

```
INSERT INTO taxonomie.cor_nom_liste (id_liste,id_nom)
SELECT 13,n.id_nom FROM taxonomie.bib_noms n
JOIN taxonomie.taxref t ON t.cd_nom = n.cd_nom
where t.group2_inpn = 'Poissons';
```

```
INSERT INTO taxonomie.cor_nom_liste (id_liste,id_nom)
SELECT 14,n.id_nom FROM taxonomie.bib_noms n
JOIN taxonomie.taxref t ON t.cd_nom = n.cd_nom
where t.group2_inpn = 'Reptiles';
```

```
INSERT INTO taxonomie.cor_nom_liste (id_liste,id_nom)
SELECT 1001,n.id_nom FROM taxonomie.bib_noms n
JOIN taxonomie.taxref t ON t.cd_nom = n.cd_nom
where t.group2_inpn in ('Amphibiens','Mammifères','Oiseaux','Poissons','Reptiles
↪');
```

```
INSERT INTO taxonomie.cor_nom_liste (id_liste,id_nom)
SELECT 1003,n.id_nom FROM taxonomie.bib_noms n
JOIN taxonomie.taxref t ON t.cd_nom = n.cd_nom
where t.regne ='Plantae';
```

```
INSERT INTO taxonomie.cor_nom_liste (id_liste,id_nom)
SELECT 301,n.id_nom FROM taxonomie.bib_noms n
JOIN taxonomie.taxref t ON t.cd_nom = n.cd_nom
where t.group2_inpn = 'Mousses';
```

```
INSERT INTO taxonomie.cor_nom_liste (id_liste,id_nom)
SELECT 302,n.id_nom FROM taxonomie.bib_noms n
JOIN taxonomie.taxref t ON t.cd_nom = n.cd_nom
where t.group2_inpn = 'Lichens';
```

```
INSERT INTO taxonomie.cor_nom_liste (id_liste,id_nom)
SELECT 303,n.id_nom FROM taxonomie.bib_noms n
JOIN taxonomie.taxref t ON t.cd_nom = n.cd_nom
where t.group2_inpn in ('Algues brunes','Algues rouges','Algues vertes');
```

```
INSERT INTO taxonomie.cor_nom_liste (id_liste,id_nom)
SELECT 305,n.id_nom FROM taxonomie.bib_noms n
JOIN taxonomie.taxref t ON t.cd_nom = n.cd_nom
where t.group2_inpn = 'Fougères';
```

```
INSERT INTO taxonomie.cor_nom_liste (id_liste,id_nom)
SELECT 306,n.id_nom FROM taxonomie.bib_noms n
JOIN taxonomie.taxref t ON t.cd_nom = n.cd_nom
where t.ordre IN ('Acorales','Asparagales','Alismatales','Dioscoreales',
↪'Geraniales','Liliales','Pandanales','Arecales','Petrosaviales','Poales',
↪'Commelinales','Zingiberales');
```

```
INSERT INTO taxonomie.cor_nom_liste (id_liste,id_nom)
SELECT 307,n.id_nom FROM taxonomie.bib_noms n
JOIN taxonomie.taxref t ON t.cd_nom = n.cd_nom
where t.ordre IN ('Canellales','Laurales','Magnoliales','Piperales','Buxales',
↪'Proteales','Trochodendrales','Ranunculales','Caryophyllales','Gunnerales',
↪'Santalales','Saxifragales','Vitales','Célastrales','Cucurbitales','Fabales',
↪'Fagales','Rosales','Malpighiales','Oxalidales','Zygophyllales','Brassicales',
↪'Crossomatales','Géraniales','Huerteales','Malvales','Myrtales','Picramiales',
↪'Sapindales','Cornales','Ericales','Garryales','Gentianales','Lamiales',
↪'Solanales','Apiales','Aquifoliales','Asterales','Bruniales','Dipsacales',
↪'Escalioniales','Paracryphyales');
```

```
INSERT INTO taxonomie.cor_nom_liste (id_liste,id_nom)
SELECT 2,n.id_nom FROM taxonomie.bib_noms n
JOIN taxonomie.taxref t ON t.cd_nom = n.cd_nom
where t.group2_inpn = 'Annélides';
```

```
INSERT INTO taxonomie.cor_nom_liste (id_liste,id_nom)
SELECT 5,n.id_nom FROM taxonomie.bib_noms n
JOIN taxonomie.taxref t ON t.cd_nom = n.cd_nom
where t.group2_inpn = 'Crustacés';
```

```
INSERT INTO taxonomie.cor_nom_liste (id_liste,id_nom)
SELECT 8,n.id_nom FROM taxonomie.bib_noms n
JOIN taxonomie.taxref t ON t.cd_nom = n.cd_nom
where t.group2_inpn = 'Gastéropodes';
```

```
INSERT INTO taxonomie.cor_nom_liste (id_liste,id_nom)
SELECT 9,n.id_nom FROM taxonomie.bib_noms n
JOIN taxonomie.taxref t ON t.cd_nom = n.cd_nom
where t.group2_inpn = 'Insectes';
```

```
INSERT INTO taxonomie.cor_nom_liste (id_liste,id_nom)
SELECT 10,n.id_nom FROM taxonomie.bib_noms n
JOIN taxonomie.taxref t ON t.cd_nom = n.cd_nom
where t.group2_inpn = 'Bivalves';
```

```
INSERT INTO taxonomie.cor_nom_liste (id_liste,id_nom)
SELECT 15,n.id_nom FROM taxonomie.bib_noms n
JOIN taxonomie.taxref t ON t.cd_nom = n.cd_nom
where t.group2_inpn = 'Myriapodes';
```

```
INSERT INTO taxonomie.cor_nom_liste (id_liste,id_nom)
SELECT 16,n.id_nom FROM taxonomie.bib_noms n
JOIN taxonomie.taxref t ON t.cd_nom = n.cd_nom
where t.group2_inpn = 'Arachnides';
```

```
INSERT INTO taxonomie.cor_nom_liste (id_liste,id_nom)
SELECT 1002,n.id_nom FROM taxonomie.bib_noms n
JOIN taxonomie.taxref t ON t.cd_nom = n.cd_nom
where t.group2_inpn in ('Arachnides','Myriapodes','Bivalves','Insectes',
↪'Gastéropodes','Crustacés','Annélides');
```

— Créer une SOURCE pour les données SERENA dans `synthese.bib_sources`

```
8;"Serena";"Données saisies avec SERENA (jusqu'à novembre 2016)";"localhost";22;"
↪";"";"geonaturedb";"serena_affo_pnr_vm";"rnf_obse_geom";""OBSE_ID"";"";"";"";
↪"FAUNE";FALSE
```

notes Probleme dans `synthese.bib_sources` du champ GROUPE en NOT NULL alors que dans BDD du PNE c'est pas le cas. Hors pour toutes les sources externes, le groupe n'a pas d'intérêt. Et pour SERENA, y a pas vraiment de groupe. Du coup on a mis FAUNE même si c'est pas très cohérent pour SERENA dont on n'a pas besoin de renseigner le groupe.

Préparer le contenu des autres tables de métadonnées liées aux données sources avec de les intégrer dans la synthèse.

— Dans `meta.bib_programmes`

```
8;"Historique";"Données historiques";TRUE;TRUE;"Données SERENA et autres ?"
```

— Dans `meta.bib_lots`

```
8;"Historique SERENA";"Données saisies avec SERENA jusqu'en novembre 2016";FALSE;
↪TRUE;FALSE;1
```

— Dans `meta.t_protocoles`

```
id_protocole = 0;"Aucune info"
```

On peut maintenant intégrer les données SERENA dans la synthèse de GeoNature.

— Créer une table synthèse temporaire (pas obligatoire mais c'est une sécurité dans notre cas expérimental)

```
CREATE TABLE synthese.syntheseff_temp
(
  id_synthese integer,
```

```
  id_source integer,
  id_fiche_source character varying(50),
  code_fiche_source character varying(50),
  id_organisme integer,
  id_protocole integer,
  id_precision integer,
  cd_nom integer,
  insee character(5),
  dateobs date NOT NULL,
  observateurs character varying(255),
  determinateur character varying(255),
  altitude_retenue integer,
  remarques text,
  date_insert timestamp without time zone,
  date_update timestamp without time zone,
  derniere_action character(1),
  supprime boolean,
  the_geom_point geometry,
  id_lot integer,
  id_critere_synthese integer,
  the_geom_3857 geometry,
  effectif_total integer,
  the_geom_2154 geometry,
  diffusable boolean DEFAULT true)
```

— On y insère les données SERENA. Largement améliorable. En se basant sur les tables brutes et/ou une VM et en affinant la requête.

```
INSERT INTO synthese.syntheseff_temp
SELECT
        1 AS id_synthese,
        8 AS id_source,
        "OBSE_ID"::text as id_fiche_source,
        "OBSE_RELV_ID"::text as code_fiche_source,
        2 AS id_organisme,
        0 AS id_protocole,
        12 AS id_precision,
        taxon_mnhn_id AS cd_nom,
        sig_commune_insee AS insee,
        CASE
         WHEN length("OBSE_DATE") = 8 THEN (left("OBSE_DATE",4)||'-'||substring(
↪"OBSE_DATE" from 5 for 2)||'-'||right("OBSE_DATE",2))::date
         WHEN length("OBSE_DATE") = 6 THEN (left("OBSE_DATE",4)||'-'||substring(
↪"OBSE_DATE" from 5 for 2)||'-01')::date
         WHEN length("OBSE_DATE") = 4 THEN (left("OBSE_DATE",4)||'-01-01')::date
         ELSE ('1000-01-01')::date
        END as dateobs,
        "SRCE_COMPNOM_C" AS observateurs,
        '' AS determinateur,
        "OBSE_ALT"::int AS altitude_retenue,
        "OBSE_COMMENT" AS remarques,
        now() AS date_insert,
        now() AS date_update,
        'c' AS derniere_action,
        false AS supprime,
        st_transform(st_centroid(geom),3857) AS the_geom_point,
        8 AS id_lot,
        1 AS id_critere_synthese,
```

```
        st_transform(geom, 3857) AS the_geom_3857,
        1 AS effectif_total,
        geom AS the_geom_2154,
        true AS diffusable
  FROM serena_affo_pnr_vm.rnf_obse_geom
```

notes

- On pourrait retrouver l'ID des protocoles dans `serena."RNF_RELV"` car dans la table à plat on n'a que `RELV_NOM`. A caler avec `meta.t_protocoles`.
- On pourrait retrouver l'ID des organismes dans `serena."RNF_SRCE"` ou le recréer dans UsersHub car dans la table à plat on n'a que `RELV_PROP_LIBEL` // `SELECT DISTINCT "RELV_PROP_LIBEL" FROM serena_affo_pnr_vm.rnf_obse_geom`.
- Pour renseigner `id_precision`, on pourrait utiliser le champs `type_geoloc`.
- Pour la géométrie, on ferait mieux de garder le geom original (maille, commune, ...) car la synthese a 2 champs pour cela. Un pour la geometrie originale et son centroïde.
- Il y a des x dans `OBSE_NOMBRE`, du coup on ne peut pas utiliser ce champs pour lequel on attend un nombre entier. On met 1 par défaut. On pourrait affiner en excluant les valeurs X et intégrant les autres valeurs quand il s'agit bien d'un numérique.

Désactiver les 4 triggers de la table `synthese.syntheseff` (avec pgAdmin).

- Copier les données dans la table `synthese.syntheseff` depuis la table `synthese.syntheseff_temp`

```
INSERT INTO synthese.syntheseff
 (id_source,
  id_fiche_source,
  code_fiche_source,
  id_organisme,
  id_protocole,
  id_precision,
  cd_nom,
  insee,
  dateobs,
  observateurs,
  determinateur,
  altitude_retenue,
  remarques,
  date_insert,
  date_update,
  derniere_action,
  supprime,
  the_geom_point,
  id_lot,
  id_critere_synthese,
  the_geom_3857,
  effectif_total,
  the_geom_2154,
  diffusable)
SELECT
  id_source,
  id_fiche_source,
  code_fiche_source,
  id_organisme,
  id_protocole,
  id_precision,
  cd_nom,
```

```
  insee,
  dateobs,
  observateurs,
  determinateur,
  altitude_retenue,
  remarques,
  date_insert,
  date_update,
  derniere_action,
  supprime,
  st_transform(the_geom_point,3857),
  id_lot,
  id_critere_synthese,
  ST_SetSRID(the_geom_3857,3857),
  effectif_total,
  ST_SetSRID(the_geom_2154,2154),
  diffusable
 FROM synthese.syntheseff_temp
```

Avec pgAdmin, faire un VACUUM et un REINDEX (clic droit sur la couche / Maintenance)

Pour intégrer les unités géographiques (qui vont permettre d'orienter les saisies du contact), on part des mailles 5 km de l'INPN.

On les ouvre avec QGIS, on ouvre aussi 2 tables de la BDD `geonaturedb` : `layers.l_unites_geo` et `layers.l_communes`.

On intersecte la couche des communes avec celles des mailles INPN pour ne garder que les mailles présentes dans les communes étudiées.

On copie colle ensuite les mailles dans `layers.l_unites_geo`. Il leur faut un identifiant unique, donc on utilise la calculatrice de champs pour mettre à jour le champs `id_unite_geo` avec la fonctionn QGIS `$rownum`.

On sort du mode édition, les mailles sont alors insérées dans la BDD dans la table `layers.l_unites_geo`.

On réactive le trigger `tri_maj_cor_unite_synthese` puis on déclenche l'intersection entre toutes les observations et toutes les unités géographiques (mailles 5 km dans notre cas) :

```
UPDATE synthese.syntheseff SET the_geom_2154 = the_geom_2154
```

Faire la même chose pour remplir les zones à statut (`layers.bib_typeszones` et `layers.l_zonesstatut`).

Réactiver les autres triggers.

Compléments GeoNature

Dans la Synthèse, pour que la liste des communes s'affiche, ils faut passer leur champs `layers.l_communes.saisie` à true.

```
UPDATE layers.l_communes SET saisie = true
```

Toutes les réserves et les sites Natura 2000 de France remontent. A nettoyer si besoin dans la base pour ne garder que celles du territoire étudié.

Aucun taxon n'est tagué patrimonial ni protégé. Pour les protections, il y a un travail d'analyse des textes à faire dans `taxonomie.protection_articles`. (Cocher correctement le champ `concerne_mon_territoire` puis utiliser `taxonomie.taxref_protection_especes` pour mettre à jour la table `taxonomie.cor_taxon_attribut`.

Il y a donc encore du travail sur les données pour un fonctionnement normal.

Problème identifié dans la 1.8.0 : La synthèse ne se charge pas, c'est la vue `synthese.v_tree_taxons_synthese` qui n'aboutit pas car une donnée ne trouve aucun REGNE dans TaxRef. La vue sera corrigée dans GeoNature 1.8.1.

Dans les données SERENA du PNR, il y avait 678 données avec des geom vides.

Créer une table `invalid_synthese` pour les mettre de côté.

```
CREATE TABLE synthese.invalid_synthese
(
  id_synthese integer NOT NULL,
  id_source integer,
  id_fiche_source character varying(50),
  code_fiche_source character varying(50),
  id_organisme integer,
  id_protocole integer,
  id_precision integer,
  cd_nom integer,
  insee character(5),
  dateobs date NOT NULL,
  observateurs character varying(255),
  determinateur character varying(255),
  altitude_retenue integer,
  remarques text,
  date_insert timestamp without time zone,
  date_update timestamp without time zone,
  derniere_action character(1),
  supprime boolean,
  the_geom_point geometry,
  id_lot integer,
  id_critere_synthese integer,
  the_geom_3857 geometry,
  effectif_total integer,
  the_geom_2154 geometry,
  diffusable boolean DEFAULT true,
  CONSTRAINT invalid_synthese_pkey PRIMARY KEY (id_synthese)
  );
  COMMENT ON TABLE synthese.invalid_synthese
  IS 'Table des données de synthèse invalides';

INSERT INTO synthese.invalid_synthese;
SELECT * FROM synthese.syntheseff WHERE the_geom_3857 IS null;
DELETE FROM synthese.syntheseff WHERE the_geom_3857 IS null;
```

Pour en savoir plus et aller plus loin avec GeoNature, voir la présentation (https://github.com/PnEcrins/GeoNature) et la documentation (http://geonature.readthedocs.io/).

Customisation de l'atlas

Charger les bonnes couches SHP des communes et du territoire sur le serveur dans `atlas/data/ref/`.

Dans notre cas, on se limite au territoire du PNR pour le moment.

Relancer l'installation de la BDD :

```
cd atlas
sudo ./install_db.sh
```

La configuration de l'atlas se trouve dans `atlas/main/configuration/config.py`.

La customisation se fait uniquement dans `atlas/static/custom`.

On peut y modifier les templates, ajouter ou modifier les images, créer un glossaire ou encore surcoucher les styles CSS (exemple : http://biodiversite.ecrins-parcnational.fr/static/custom/custom.css).

Il est aussi possible de modifier les vues matérialisées pour adapter le contenu de l'atlas.

Pour plus de détail sur le fonctionnement de GeoNature-atlas voir sa documentation générale : https://github.com/PnEcrins/GeoNature-atlas/blob/master/docs/installation.rst.

Le détail des vues matérialisées : https://github.com/PnEcrins/GeoNature-atlas/blob/master/docs/vues_materialisees_maj.rst.

Les présentations PDF du projet : https://github.com/PnEcrins/GeoNature-atlas/tree/master/docs.

Pour aller plus loin

— Suivre les 4 projets sur Github (Watch en haut à droite de chaque projet)
— Créer des tickets (issues) pour tout bug ou question
— Proposer des évolutions du code en faisant des pull requests dans Github
— Mettre à jour les applications en suivant les procédures et en lisant bien les nouveautés de chaque version
— Mettre en place des sauvegardes automatiques des données

CHAPITRE 9

AUTEURS

Parc national des Ecrins

— Gil Deluermoz
— Camille Monchicourt

Parc national des Cevennes

— Amandine Sahl

CHAPITRE 10

CHANGELOG

1.9.0

ATTENTION : Les évolutions de cette release concernent aussi la webapi. Si vous utilisez les applications GeoNature-Mobile, vous devez attendre la sortie d'une release de GeoNature-mobile-webapi compatible avec cette version 1.9.0. Comming soon !

Nouveautés

- Ajout de la création des index spatiaux à la création initiale de la base.
- Création ou mise à jour des géometries compatible postgis 2.
- Ajout de la possibilité d'éditer la notion de diffusable (oui/non) uniquement pour contactfaune et mortalité - TODO :faire la même chose pour les autres protocoles.
- Multi-projection : Les versions antérieures de GeoNature n'étaient compatible qu'avec la projection lambert 93 (srid : 2154). Cette version permet de choisir sa projection locale. Elle ajoute un paramètre `srid_local` dans le `config/settings.ini` et renomme tous les champs `the_geom_2154` en `the_geom_local` des tables "métier".

Ce paramètre est notamment utilisé lors de la création de la base pour affecter le srid de la projection locale à tous les champs `the_geom_local` présents dans les tables de la base. Ce paramètre est également utilisé pour mettre en cohérence le système de projection local utilisé dans toutes les couches SIG présentes dans la base et les geométries stockées dans les champs `the_geom_local` des tables "métier". Le paramétrage du service WMS dans wms/wms.map est également pris en charge par le script d'installation de l'application. * Correction de l'installation de npm * install all mis à jour avec les nouvelles versions de l'atlas, de TaxHub et de UsersHub.

IMPORTANT : toutes les couches SIG insérées dans le schéma `layers` doivent être dans la projection fournie pour le paramètre `srid_local`. L'application est livrée avec un ensemble de couches en Lambert 93 concernant la métropole. Une installation avec une autre projection, hors métropole, doit donc se faire sans l'insertion des couches SIG. Vous devrez manuellement fournir le contenu des tables du schéma `layers` dans la projection choisie.

Notes de versions

Vous pouvez ajouter les paramètres srid_local, install_sig_layers et add_sample_data au fichier config/settings.ini en vous inspirant du fichier config/settings.ini.sample. Toutefois ces paramètres ne sont utilisés que pour une nouvelle installation et notamment pour l'installation de la base.

Vous pouvez passer directement d'une 1.7.X à la 1.9.0, en prenant en compte les notes des différentes versions intermédiaires.

Si vous migrez depuis la version 1.8.3, éxécutez le fichier SQL `data/update_1.8.3to1.9.0.sql`. Comme GeoNature ne fonctionne jusque là que pour des structures de métropole, il est basé sur le fait que le champ `the_geom_local` reste en lambert 93 (2154). Assurer vous que le paramètre `$srid_local` dans `lib/sfGeonatureConfig.php` est égal à `2154`. ATTENTION : ce script sql renomme tous les champs `the_geom_2154` en `the_geom_local` de la base geonature. Ceci affecte de nombreuses tables, de nombreux triggers et de nombreuses vues de la base. Le script n'intègre que les vues fournies par défaut. Si vous avez créé des vues spécifiques, notamment pour le module d'export, ou si vous avez modifié des vues fournies, vous devez adapater/compléter le script. Vous pouvez vous inspirer de son contenu. RAPPEL : Ceci affecte également la webapi pour les applications mobiles ; Vous devez donc mettre à jour votre webapi si vous utiliser la saisie sur les applications mobiles. Une release de la webapi devrait sortir bientôt

1.8.4 (2017-04-10)

Corrections

— Correction du script d'installation globale (`install_all`) si l'utilisateur de BDD par défaut a été renommé (`data/grant.sql`)
— Correction de la création des vues qui remontent la liste des taxons dans les 3 contacts

1.8.3 (2017-02-23)

Nouveautés

— Multi-organisme : l'organisme associé à la donnée est désormais celui de l'utilisateur connecté dans l'application (lors de la création d'une observation uniquement).
— Taxonomie : création d'une liste `Saisie possible`, remplaçant l'attribut `Saisie`. Cela permet de choisir les synonymes que l'on peut saisir ou non dans GeoNature en se basant sur les `cd_nom` (`bib_listes` et `cor_nom_liste`) et non plus sur les `cd_ref` (`bib_attributs` et `cor_taxon_attribut`). Voir le script de migration SQL `data/update_1.8.2to1.8.3.sql` pour bien basculer les informations de l'attribut dans la nouvelle liste.
— Correction de la vue `synthese.v_tree_taxons_synthese` potentiellement bloquante à l'ouverture de la synthèse.
— Suppression de la table `utilisateurs.bib_observateurs` inutile.
— Création des index spatiaux manquants (performances)
— Clarification et corrections mineures du script `install_all`
— Ajout du MCD de la 1.8 (par @xavier-pnm)
— Améliorations du nom des fichiers exportés depuis la Synthèse (par @sylvain-m)

Notes de versions

Vous pouvez supprimer les lignes concernant le paramètre `public static $id_organisme = ...` dans `lib/sfGeonatureConfig.php`, l'organisme n'étant plus un paramètre fixe mais désormais celui de l'utilisateur connecté.

Vous pouvez passer directement d'une 1.7.X à la 1.8.3, en prenant en compte les notes des différentes versions intermédiaires.

Si vous migrez depuis la version 1.8.2, éxécutez le fichier SQL `data/update_1.8.2to1.8.3.sql`.

1.8.2 (2017-01-11)

Nouveautés

— Modularité des scripts SQL de création de la base en les dissociant par protocole et en regroupant les triggers dans les schémas de chaque protocole (préparation GeoNature V2)
— Correction d'une requête dans flore station (indépendance vis à vis de flore patrimoniale)
— Correction du trigger `synthese_update_fiche_cflore` (@ClaireLagaye)

Notes de versions

Vous pouvez passer directement d'une 1.7.X à la 1.8.2, en prenant en compte les notes des différentes versions intermédiaires.

Si vous migrez depuis la version 1.8.1, éxécutez le fichier `data/update_1.8.1to1.8.2.sql`. Consultez les dernières lignes de ce fichier : vous devez évaluer si la requête d'insertion dans la table `taxonomie.cor_taxon_attribut` doit être faite ou non (vous pourriez avoir déjà constaté et corrigé cette erreur lors d'une précédente migration). Cela corrige l'absence de taxons protégés dans votre synthese en récupérant les informations de protection présentes dans le champ `filtre3` de la table `save.bib_taxons`

1.8.1 (2017-01-05)

Nouveautés

— Ajout des sauvegardes et de l'installation globale avec un exemple détaillé dans la documentation : http://geonature.readthedocs.io
— Optimisation et correction de la vue qui retourne l'arbre des rangs taxonomiques (synthese.v_tree_taxons_synthese)
— Mise en cohérence des données exemple de GeoNature-atlas avec les critères des vues matérialisées de GeoNature-atlas
— Mise à jour de 2 triggers du Contact Flore (@ClaireLagaye)

Notes de versions

Vous pouvez passer directement d'une 1.7.X à la 1.8.1, en prenant en compte les notes des différentes versions intermédiaires.

Si vous migrez depuis la version 1.8.0, éxécutez le fichier `data/update_1.8to1.8.1.sql`

1.8.0 (2016-12-14)

Nouveautés

— Passage à TAXREF version 9
— Accès à la synthèse en consultation uniquement pour des utilisateurs enregistrés avec des droits 1
— Ajout d'un champ `diffusion` (oui/non) dans la table `synthese.syntheseff`, utilisable dans GeoNature-atlas. Pas d'interface de gestion de ce champ pour le moment. CF #132
— Création d'un script d'installation simplifié pour un pack UsersHub, TaxHub, GeoNature et GeoNature-atlas : https://github.com/PnEcrins/GeoNature/tree/master/docs/install_all
— Factorisation des SQL de création des schémas `taxonomie` et `utilisateurs` en les récupérant dans les dépots TaxHub et UsersHub
— Compatibilité avec l'application TaxHub qui permet de gérer la taxonomie à partir de TAXREF. Cela induit d'importants changements dans le schéma `taxonomie`, notamment le renommage de `taxonomie.bib_taxons` en `taxonomie.bib_noms`, la suppression de `taxonomie.bib_filtres` et l'utilisation de `taxonomie.bib_attributs` (voir https://github.com/PnX-SI/TaxHub/issues/71 pour plus d'informations). Voir aussi le fichier de migration `data/update_1.7to1.8.sql` qui permet d'automatiser ces évolutions de la BDD

— Compatibilité avec l'application GeoNature-atlas qui permet de diffuser les données de la synthèse faune et flore dans un atlas en ligne (exemple : http://biodiversite.ecrins-parcnational.fr)
— Création d'un site internet de présentation de GeoNature : http://geonature.fr

Corrections

— Amélioration des triggers concernant la suppression de fiches orphelines
— Affichage par défaut du nom latin dans Contact flore et Contact invertébrés
— Correction des exports lors de la présence de points-virgules dans les commentaires. Fix #143
— Suppression du besoin d'un super utilisateur lors de l'installation de la BDD. Fix #141
— Correction de l'ID des protocoles mortalité et invertebres dans la configuration par défaut
— Suppression d'un doublon dans le fichier de configuration symfony de l'application
— Correction des coordonnées lors de l'export de données Flore Station
— Autres corrections mineures

Note de version

— Exécuter le script SQL de migration réalisant les modifications de la BDD de la version 1.7.X à 1.8.0 `data/update_1.7to1.8.sql`
— Mettre à jour taxref en V9 en vous inspirant du script `data/taxonomie/inpn/update_taxref_v8tov9`

TaxHub

L'application TaxHub (https://github.com/PnX-SI/TaxHub) est désormais fonctionnelle, documenté et installable.

Elle vous aidera à gérer vos taxons et l'ensemble du schéma `taxonomie`, présent dans la BDD de GeoNature.

TaxHub évoluera pour intégrer progressivement de nouvelles fonctionnalités.

Il est conseillé de ne pas installer la base de données de TaxHub indépendamment et de connecter l'application directement sur le la base de données de GeoNature.

GeoNature-atlas

GeoNature-atlas est également basé sur le schéma `taxonomie` de TaxHub. Ainsi TaxHub permet la saisie des informations relatives aux taxons (descriptions, milieux, photos, liens, PDF...). GeoNature-atlas dispose de sa propre base de données mais pour fonctionner en connexion avec le contenu de la base GeoNature il faut à minima disposer d'une version 1.8 de GeoNature.

notes Une régression dans le contenu de Taxref V9 conduit à la suppression de l'information concernant le niveau de protection des espèces (régional, national, international,...). Cette information était utilisée par GeoNature, notamment pour définir les textes à retenir pour la colonne `concerne_mon_territoire` de la table `taxonomie.taxref_protection_articles`. Vous devez désormais remplir cette colonne manuellement.

1.7.4 (2016-07-06)

Corrections de bugs

— Correction du script d'installation des tables liées au Contact flore (5a1fb07)
— Mise en cohérence avec GeoNature-mobile utilisant les classes 'gasteropodes' et 'bivalves' et non la classe générique 'mollusques'.

Nouveautés

— Corrections de mise en forme de la documentation
— Ajout de la liste rouge France de TaxRef lors d'une nouvelle installation (f4be2b6). A ne pas prendre en compte dans le cas d'une mise à jour.
— Ajout du MCD de la BDD - https://github.com/PnEcrins/GeoNature/blob/master/docs/2016-04-29-mcd_geonaturedb.png

Note de version

— Vous pouvez passer directement de la version 1.6.0 à la 1.7.4 mais en vous référant aux notes de version de la 1.7.0.

— Remplacer `id_classe_mollusques` par `id_classe_gasteropodes` dans `web/js/config.js` et renseigner la valeur en cohérence avec l'`id_liste` retenu dans la table `taxonomie.bib_listes` pour les gastéropodes. Attention, vous devez avoir établi une correspondance entre les taxons gastéropodes et bivalves et leur liste dans la table `taxonomie.cor_taxon_liste`.

1.7.3 (2016-05-19)

Corrections de bugs

— Correction de coordonnées vides dans l'export de Flore station. cf https://github.com/PnEcrins/GeoNature/commit/0793a3d3d2b3719ed515058d1a0ba9baf7cb2096
— Correction des triggers en base concernant un bug de saisie pour les taxons dont le taxon de référence n'est pas présent dans `taxonomie.bib_taxons`.

Note de version

Rappel : commencez par suivre la procédure classique de mise à jour. http://geonature.readthedocs.org/fr/latest/update.html

— Vous pouvez passer directement de la version 1.6.0 à la 1.7.3 mais en vous référant aux notes de version de la 1.7.0.
— Pour passer de la 1.7.2 à la 1.7.3 vous devez exécuter le script `https://github.com/PnEcrins/GeoNature/blob/master/data/update_1.7.2to1.7.3.sql`.

1.7.2 (2016-04-27)

Corrections de bug

— Correction d'un bug dans l'export XLS depuis Flore Station.

Note de version

— Vous pouvez passer directement de la version 1.6.0 à la 1.7.2 mais en vous référant aux notes de version de la 1.7.0.

1.7.1 (2016-04-27)

Corrections de bug

— Ajout des listes flore manquantes dans le script de mise à jour `data/update_1.6to1.7.sql`.

1.7.0 (2016-04-24)

Nouveautés

— Ajout du contact flore
— Correction et compléments dans les statistiques et mise en paramètre de leur affichage ou non, ainsi que de la date de début à prendre en compte pour leur affichage.
— Ajout d'un module d'export des données permettant d'offrir, en interne ou à des partenaires, un lien de téléchargement des données basé sur une ou des vues de la base de données (un fichier par vue). Voir http://geonature.readthedocs.org/fr/latest/export.html
— Modification des identifiants des listes pour compatibilité avec les applications GeoNature-Mobile.
— Complément dans la base de données pour compatibilité avec les applications GeoNature-Mobile.
— Correction d'une erreur sur l'importation de shape pour la recherche géographique
— WMS : correction de la liste des sites N2000, correction de l'affichage de l'aire optimale d'adhésion des parcs nationaux et retrait des sites inscrits et classés

— Correction d'un bug permettant la saisie d'une date d'observation postérieure à aujourd'hui dans Flore station
— Mention de la version de taxref sur la page d'accueil

Note de version

Rappel : commencez par suivre la procédure classique de mise à jour. http://geonature.readthedocs.org/fr/latest/update.html

1. Modification des identifiants des listes de taxons pour compatibilité avec les applications GeoNature-Mobile.

Dans GeoNature-Mobile, les taxons sont filtrables par classe sur la base d'un `id_classe`. Ces id sont inscrits en dur dans le code des applications mobiles.

Dans la base GeoNature les classes taxonomiques sont configurables grace au vues `v_nomade_classes` qui utilisent les listes (`taxonomie.bib_listes`).

Les `id_liste` ont donc été mis à jour pour être compatibles avec les `id_classe` des applications mobiles.

Voir le script SQL d'update `data/update_1.6to1.7.sql` et LIRE ATTENTIVEMENT LES COMMENTAIRES.

— En lien avec les modifications ci-dessus, mettre à jour les variables des classes taxonomiques correspondant aux modification des `id_liste` dans `web/js/config.js`
— Ajouter dans le fichier `lib/sfGeonatureConfig.php` les variables `$struc_abregee`, `$struc_long`, `$taxref_version`, `$show_statistiques` et `$init_date_statistiques` (voir le fichier `lib/sfGeonatureConfig.php.sample`)

2. Pour ajouter le Contact flore

— Exécuter le script sql `data/2154/contactflore.sql`
— Ajouter les variables `$id_lot_cflore = 7`, `$id_protocole_cflore = 7`, `$id_source_cflore = 7` et `$appname_cflore = 'Contact flore - GeoNature';` dans `lib/sfGeonatureConfig.php` (voir le fichier d'exemple `lib/sfGeonatureConfig.php.sample`)
— Ajouter les variables `id_lot_contact_flore = 7`, `id_protocole_contact_flore = 7`, `id_source_contactflore = 7` dans `web/js/config.js` (voir le fichier d'exemple `web/js/config.js.sample`)
— l'enregistrement correspondant au contact flore dans la table `synthese.bib_sources` doit être actif (dernière colonne) pour que le contact flore soit accessible depuis la page d'accueil.

3. Afin de mettre à jour la configuration WMS, vous devez exécuter le fichier `wms/update1.6to1.7.sh`.

Au préalable, assurez vous que les informations renseignées dans le fichier `config/settings.ini` sont à jour. L'ancien fichier sera sauvegardé sous `wms/wms_1.6.map`. Vous pourrez faire le choix de conserver ou de supprimer ce fichier de sauvegarde qui ne sera pas utilisé par l'application.

```
./wms/update1.6to1.7.sh
```

4. Mise en place du module d'export

— Créer les vues retournant les données attendues.
— Configurer le module dans le fichier `lib/sfGeonatureConfig.php` à partir de l'exemple du fichier `lib/sfGeonatureConfig.php.sample`) ; section `configuration du module d'export`
  — Vous pouvez paramétrer plusieurs modules avec un nom pour chacun grace au paramètre `exportname`
  — Pour chacun des modules seuls les utilisateurs de geonature dont le `id_role` figure dans le tableau `authorized_roles_ids` peuvent exporter les données mises à disposition par le module d'export.
  — Chaque module peut comporter autant que vues que nécessaire (un bouton par vue générera un fichier zip par vue). Renseigner le tableau `views` pour chacun des modules.
  — Voir la documentation ici : http://geonature.readthedocs.org/fr/latest/export.html
— Attribution des droits nécessaires pour le répertoire permettant l'enregistrement temporaire des fichiers générés par le module d'export.

```
chmod -R 775 web/uploads/exports
```

— Rétablir les droits d'écriture et vider le cache

```
chmod -R 777 cache/
chmod -R 777 log/
php symfony cc
```

1.6.0 (2016-01-14)

Note de version

— Pour les changements dans la base de données vous pouvez exécuter le fichier `data/update_1.5to1.6.sql`
— Mise à jour de la configuration Apache. Modifier le fichier `apache/wms.conf` en vous basant sur l'exemple https://github.com/PnEcrins/GeoNature/blob/master/apache/wms.conf.sample#L16-L17
— Ajouter le paramètre `$id_application` dans `lib/sfGeonatureConfig.php.php` (voir la valeur utilisée pour GeoNature dans les tables `utilisateurs.t_applications` et `utilisateurs.cor_role_droit_application`)
— Ajouter le paramètre `import_shp_projection` dans `web/js/configmap.map` - voir l'exemple dans le fichier `https://github.com/PnEcrins/GeoNature/blob/master/web/js/configmap.js.sample#L35`
— Supprimer toute référence à gps_user_projection dans `web/js/configmap.map`
— Ajouter un tableau JSON des projections disponibles pour l'outil de pointage GPS : `gps_user_projections` dans `web/js/configmap.map`. Respecter la structure définie dans `https://github.com/PnEcrins/GeoNature/blob/master/web/js/configmap.js.sample#L7-L14`. Attention de bien respecter la structure du tableau JSON et notamment sa syntaxe (accolades, virgules, nom des objects, etc...)
— Ajouter les `id_liste` pour les classes faune filtrables dans les formulaires de saisie dans le fichier `web/js/config.map`. Ceci concerne les variables `id_classe_oiseaux`, `id_classe_mammiferes`, `id_classe_amphibiens`, `id_classe_reptiles`, `id_classe_poissons` et `id_classe_ecrevisses`, `id_classe_insectes`, `id_classe_arachnides`, `id_classe_myriapodes` et `id_classe_mollusques`. Voir l'exemple dans le fichier `https://github.com/PnEcrins/GeoNature/blob/master/web/js/config.js.sample#L32-44`
— Taxref a été mis à jour de la version 7 à 8. GeoNature 1.6.0 peut fonctionner avec la version 7. Cependant il est conseillé de passer en taxref V8 en mettant à jour la table `synthese.taxref` avec la version 8. Cette mise à jour pouvant avoir un impact fort sur vos données, son automatisation n'a pas été prévue. Le script SQL de migration de vos données de taxref V7 vers taxref V8 n'est donc pas fourni. Pour une installation nouvelle de la base de données, GeoNature 1.6.0 est fourni avec taxref V8.
— Le routing a été mis à jour, vous devez vider le cache de Symfony pour qu'il soit pris en compte. Pour cela, placez vous dans le répertoire racine de l'application et effectuez la commande suivante :

```
php symfony cc
```

Changements

— Les recherches dans la synthèse sont désormais faites sur le `cd_ref` et non plus sur le `cd_nom` pour retourner tous les synonymes du taxon recherché - Fix #92
— Passage de taxref V7 à Taxref V8 - Fix #34
— Intégration de la première version de l'API permettant d'intégrer des données dans la synthèse depuis une source externe - https://github.com/PnEcrins/GeoNature/blob/master/docs/geonature_webapi_doc.rst
— Mise en paramètre du `id_application` dans `lib/sfGeonatureConfig.php.php` - Fix #105
— Recharger la synthese après suppression d'un enregistrement - Fix #94
— L'utilisateur peut lui-même définir le système de coordonnées dans l'outil de pointage GPS - Fix #107
— Mise en paramètre de la projection de la shape importée comme zone de recherche dans la synthèse
— Les exports XLS et SHP comportent le `cd_nom` ET le `cd_ref` de tous les synonymes du nom recherché ainsi que le nom_latin (bib_taxons) ET le nom_valide (taxref) - Fix #92

— SAISIE invertébrés - Ajout d'un filtre Mollusques - Fix #117
— Amélioration du vocabulaire utilisé sur la page d'accueil - #118
— Affichage d'un message pendant le chargement des exports
— Mise en place de statistiques automatiques sur la page d'accueil, basées sur les listes de taxons. A compléter.

Corrections de bug

— Intégration de la librairie `OpenLayers.js` en local dans le code car les liens distants ne fonctionnaient plus - Fix #97
— Correction d'une erreur lors de l'enregistrement de la saisie invertébrés - Fix #104
— Correction d'une erreur de redirection si on choisit "Quitter" après la saisie de l'enregistrement (contact faune, mortalité et invertébrés) - Fix #102
— Correction du trigger `contactfaune.synthese_update_cor_role_fiche_cf()` - Fix #95
— Correction d'un bug dans les listes déroulantes des taxons filtrée par classe qui n'affichaient rien - Fix #109
— Correction d'un bug sur le contenu des exports shape avec le critère de protection activé - Fix #114
— Correction et adaptation faune-flore des exports shape
— SYNTHESE - Correction de la liste des taxons sans nom français - Fix #116
— Corrections CSS sur la page d'accueil - Fix #115
— Correction sur la largeur de la liste des résultats de la synthèse - Fix #110
— Correction des doublons dans la recherche multi-taxons - Fix #101
— Autres corrections mineures

1.5.0 (2015-11-26)

Note de version

— Pour les changements dans la base de données vous pouvez exécuter le fichier `data/update_1.4to1.5.sql`
— Le bandeau de la page d'accueil `web/images/bandeau_faune.jpg` a été renommé en `bandeau_geonature.jpg`. Renommez le votre si vous aviez personnalisé ce bandeau.
— Si vous souhaitez désactiver certains programmes dans le "Comment ?" de la synthèse vous devez utiliser le champs `actif` de la table `meta.bib_programmes`.
— Compléter si nécessaire les champs `url`, `target`, `picto`, `groupe` et `actif` dans la table `synthese.bib_sources`.
— Nouvelle répartition des paramètres de configuration javascript en 2 fichiers (`config.js` et `configmap.js`). Vous devez reprendre vos paramètres de configuration du fichier `web/js/config.js` et les ventiler dans ces deux fichiers.
— Ajouter le paramètre `id_source_mortalite = 2;` au fichier `web/js/config.js`;
— Retirer le paramètre `fuseauUTM;` du fichier `web/js/config.js`;
— Bien définir le système de coordonnées à utiliser pour les pointages par coordonnées fournies en renseignant le paramètre `gps_user_projection` dans le fichier `web/js/configmap.js`;
— Ajouter le paramètre `public static $id_source_mortalite = 2;` au fichier `lib/sfGeonatureConfig.php`;
— Ajouter le paramètre `public static $srid_ol_map = 3857;` au fichier `lib/sfGeonatureConfig.php`;
— L'altitude est calculée automatiquement à partir du service "Alticodage" de l'API GeoPortail de l'IGN et non pluas à partir de la couche `layers.l_isolines20`. Ajoutez ce service dans votre contrat API Geoportail. Il n'est donc plus nécessaire de remplir la couche `layers.l_isolines20`. Cette couche peut toutefois encore être utile si l'utilisateur supprime l'altitude calculée par l'API Geoportail dans les formulaires de saisie.
— Le loup et le lynx sont retirés par défaut de la saisie (saisie recommandée dans le protocole national du réseau grands prédateurs)
— Le cerf, chamois et le bouquetin doivent être saisis selon 6 critères de sexe et age et non 5 comme les autres taxons. Comportement peut-être changé en modifiant la vue `contactfaune.v_nomade_taxons_faune`.
— Mortalité est désormais une source à part entière alors qu'elles étaient mélangées avec la source ContactFaune

précédemment. Si vous avez déjà des données de mortalité enregistrées, vous devez adapter la requête SQL ci-dessous avec votre `id_source` pour Mortalité et l'exécuter :

```
UPDATE synthese.syntheseff SET id_source = 2 WHERE id_source = 1 AND id_
↪critere_synthese = 2;
```

Changements

- Optimisation des vues aux chargement des listes de taxons. Fixes #64
- Généricité des champs dans `meta.bib_programmes` (champs `sitpn` renommé en `public`). Fixes #68
- Ajout d'un champ `actif` à la table `meta.bib_programmes` permettant de masquer certains programmes dans le "Comment ?" de la synthèse. Fixes #66
- Ajout d'un champ `url`, `target`, `picto`, `groupe` et `actif` dans la table `synthese.bib_sources` pour générer la page d'accueil dynamiquement et de manière générique. Fixes #69
- Construire dynamiquement la liste des liens vers la saisie des différents protocoles à partir de la table `synthese.bib_sources`. Fixes #69
- Tous les styles des éléments de la page d'accueil ont été passés en CSS. Fixes #57
- Amélioration de l'interface pendant le chargement des différentes applications (synthèse, flore station, formulaires de saisie...). Fixes #65
- Recentrage sur la position de l'utilisation en utilisant le protocole de géolocalisation intégré au navigateur de l'utilisateur. Fixes #65
- Un message automatique conseille les utilisateurs d'Internet Explorer de plutôt utiliser Firefox ou Chrome. Fixes #65
- Tri par défaut par date décroissante des 50 dernières observations affichées à l'ouverture de la Synthèse. Fixes #51
- Vocabulaire. "Dessiner un point" remplacé par "Localiser l'observation". Fixes #66
- Mise à jour des copyrights dans les pieds de page de toutes les applications.
- Refonte du CSS du formulaire de login avec bootstrap et une image de fond différente.
- Refonte Bootstrap de la page d'accueil.
- Homogénéisation du pied de page.
- FloreStation et Bryophytes - Homogénéiser interaction carte liste - ajout d'un popup au survol. Fixes #74
- Suppression d'images non utilisées dans le répertoire `web/images`.
- Mise en cohérence des vues taxonomiques faune. Fixes #81
- Calcul de l'altitude à partir du service "Alticodage" de l'API GeoPortail de l'IGN.
- Factorisation et généralisation du module permettant un positionnement des pointages par saisie de coordonnées selon projection et bbox fournies en paramètres de config.
- Création d'une configuration javascript carto dédiée (`configmap.js`).

Corrections de bug

- Correction des problèmes de saisie de la version 1.4.0 liés à la migration de la taxonomie.
- Correction de bugs dans Flore Station et Bryophytes (Zoom, recherche

1.4.0 (2015-10-16)

Note de version

- La gestion de la taxonomie a été mis en conformité avec le schéma `taxonomie` de la base de données de TaxHub (https://github.com/PnX-SI/TaxHub). Ainsi le schéma `taxonomie` intégré à GeoNature 1.3.0 doit être globalement revu. L'ensemble des modifications peuvent être réalisées en éxecutant la partie correspondante dans le fichier `data/update_1.3to1.4.sql` (https://github.com/PnEcrins/GeoNature/blob/master/data/update_1.3to1.4.sql).
- De nouveaux paramètres ont potentiellement été ajoutés à l'application. Après avoir récupéré le fichier de configuration de votre version 1.3.0, vérifiez les changements éventuels des différents fichiers de configuration.
- Modification du nom de l'host host hébergeant la base de données. databases –> geonatdbhost. A changer ou ajouter dans le `/etc/hosts` si vous avez déjà installé GeoNature.
- Suivez la procédure de mise à jour : http://geonature.readthedocs.org/fr/latest/update.html

Changements

- A l'installation initiale, chargement en base des zones à statuts juridiques pour toute la France métropolitaine à partir des sources de l'INPN
- A l'installation initiale, chargement en base de toutes les communes de France
- Mise en place de la compatibilité de la base avec le schema de TaxHub

1.3.0 (2015-02-11)

Pré-Version de GeoNature - Faune ET Flore. Le fonctionnement de l'ensemble n'a pas été totalement testé, des bugs sont identifiés, d'autres subsistent certainement.

Changements

- Grosse évolution de la base de données
- Ajout de deux applications de saisie flore (flore station et bryophytes)
- Intégration de la flore en sythese
- Ajouter un id_lot, id_organisme, id_protocole dans toutes les tables pour que ces id soit ajoutés vers la synthese en trigger depuis les tables et pas avec des valeurs en dur dans les triggers. Ceci permet d'utiliser les paramètres de conf de GeoNature
- Ajout d'une fonction à la base pour correction du dysfonctionnement du wms avec mapserver
- Suppression du champ id_taxon en synthese et lien direct de la synthese avecle taxref. ceci permet d'ajouter des données en synthese directement dans la base sans ajouter tous les taxons manquants dans la table bib_taxons
- Suppression de la notion de coeur dans les critère de recherche en synthese
- Ajout d'un filtre faune flore fonge dans la synthese
- Ajout de l'embranchement et du regne dans les exports
- Permettre à des partenaires de saisir mais d'exporter uniquement leurs données perso
- Ajout du déterminateur dans les formulaires invertébrés et contactfaune + en synthese
- Ajout du référentiel géographique de toutes les communes de France métropolitaine
- Ajout des zones à statuts juridiques de la région sud-est (national à venir)
- Bugs fix

BUG à identifier

Installation :

- corriger l'insertion de données flore station qui ne fonctionne pas

Bryophythes :

- Corriger la recherche avancée par date sans années

Synthèse :

- la construction de l'arbre pour choisir plusieurs taxons ne tient pas compte des filtres
- le fonctionnement des unités geographiques n'a pas été testé (initialement conçu uniquement pour la faune)

1.2.0 (2015-02-11)

Version stabilisée de GeoNature - Faune uniquement (Synthèse Faune + Saisie ContactFauneVertebre, ContactFauneInvertebre et Mortalité).

Changements

- Modification du nom de l'application de FF-synthese en GeoNature
- Changement du nom des utilisateurs PostgreSQL
- Changement du nom de la base de données
- Mise à jour de la documentation (http://geonature.readthedocs.org/)
- Automatisation de l'installation de la BDD
- Renommer les tables pour plus de généricité
- Supprimer les tables inutiles ou trop spécifiques

— Gestion des utilisateurs externalisée et centralisée avec UsersHub (https://github.com/PnEcrins/UsersHub)
— Correction de bugs
— Préparation de l'intégration de la Flore pour passer de GeoNature Faune à GeoNature Faune-Flore

1.1.0 (2014-12-11)

Changements

— Modification du schéma de la base pour être compatible taxref v7
— Import automatisé de taxref v7
— Suppression des tables de hiérarchie taxonomique (famille, ordre, ...) afin de simplifier l'utilisation de la taxonomie.
— Création de la notion de groupe (para-taxonomique) à la place de l'utilisation des classes.
— Ajout de données pour pouvoir tester de façon complète l'application (invertébrés, vertébrés)
— Ajout de données exemples
— Bugs fix

1.0.0 (2014-12-10)

Version fonctionnelle des applications : visualisation de la synthèse faune, saisie d'une donnée de contact (vertébrés, invertébrés, mortalité)

Changements

— Documentation de l'installation d'un serveur Debian wheezy pas à pas
— Documentation de la mise en place de la base de données
— Documentation de la mise en place de l'application et de son paramétrage
— Script d'insertion d'un jeu de données test
— Passage à PostGIS v2
— Mise en paramètre de la notion de lot, protocole et source

Prochaines évolutions

— Script d'import de taxref v7
— Utilisation préférentielle de la taxonomie de taxref plutôt que les tables de hiérarchie taxonomique

0.1.0 (2014-12-01)

— Création du projet et de la documentation