Architecture technique et outils de développement

Atelier GeoNature
27/06/2024

Jacques Fize, PNE
Pierre Narcisi, PATRINAT

Sommaire

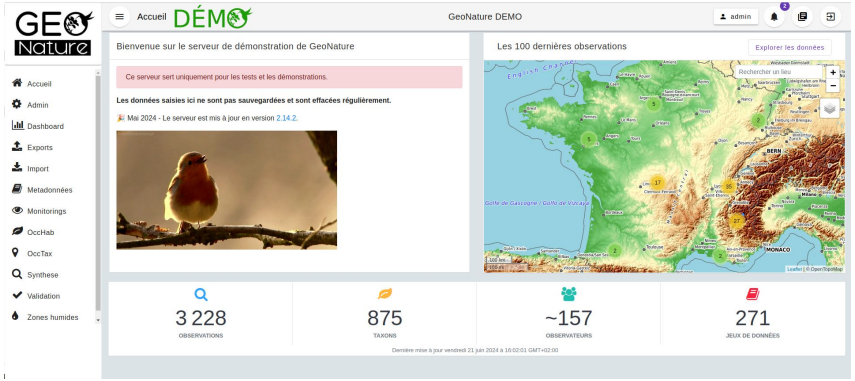
Introduction

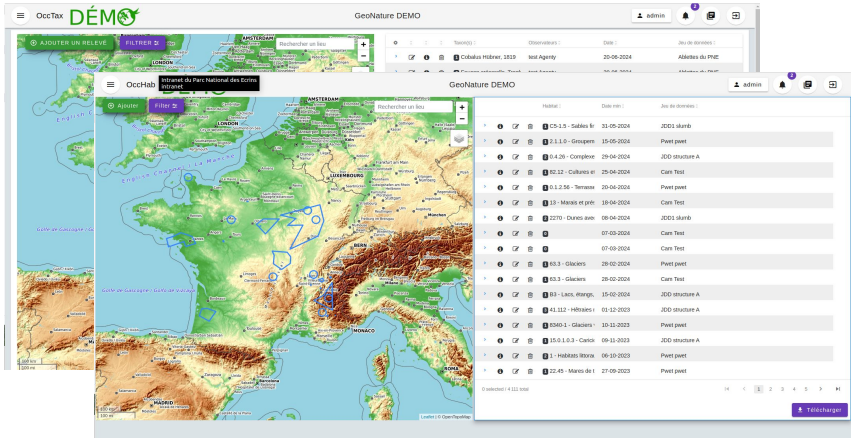
GeoNature

GeoNature est une plateforme web de saisie de données naturalistes

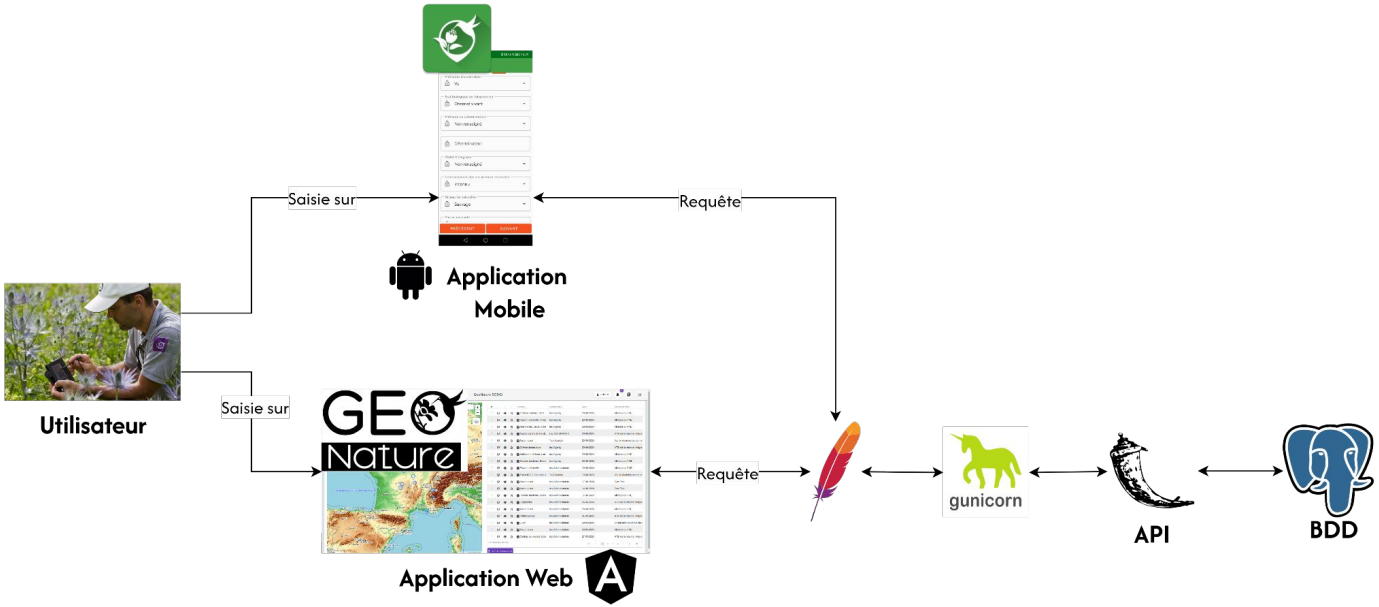
- Occurrence d'habitats (OccHab)
- Relevés d'occurrences occasionnelles de taxons (OccTax)
- Protocole de suivi (Monitoring)
- Open-source, gratuit (code source disponible sur GitHub)

Plusieurs extensions : import, export, …

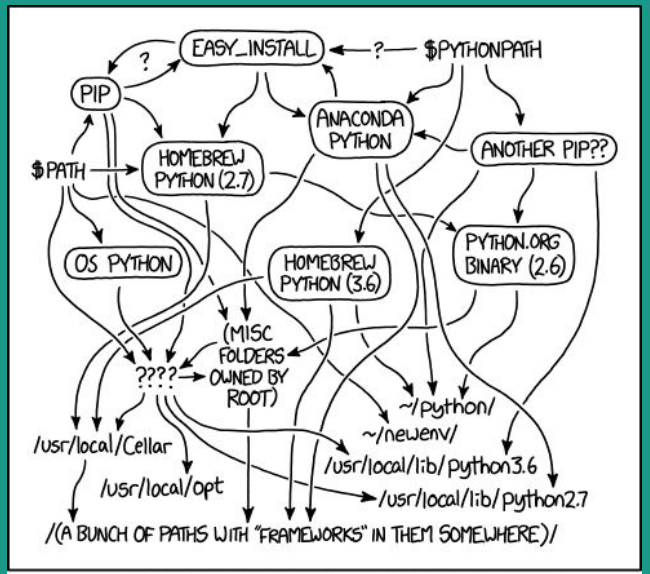




GeoNature in a nutshell



Présentation de l'architecture de GeoNature



MY PYTHON ENVIRONMENT HAS BECOME SO DEGRADED THAT MY LAPTOP HAS BEEN DECLARED A SUPERFUND SITE.

https://xkcd.com/1987/

API

L' API de GeoNature est écrite en **Python** et s'appuie fortement sur le micro-framework web **Flask [1]**, mais aussi :

- **SQLAlchemy** pour exécuter des requêtes sur la base de données, gérer le versionnage (*alembic*)
- **Marshmallow** pour la sérialisation/désérialisation/validation des données provenant de la base de données
- Plusieurs **modules génériques [2]** : *pypnusershub*, *pypnomenclature*, *refgeo*, etc…

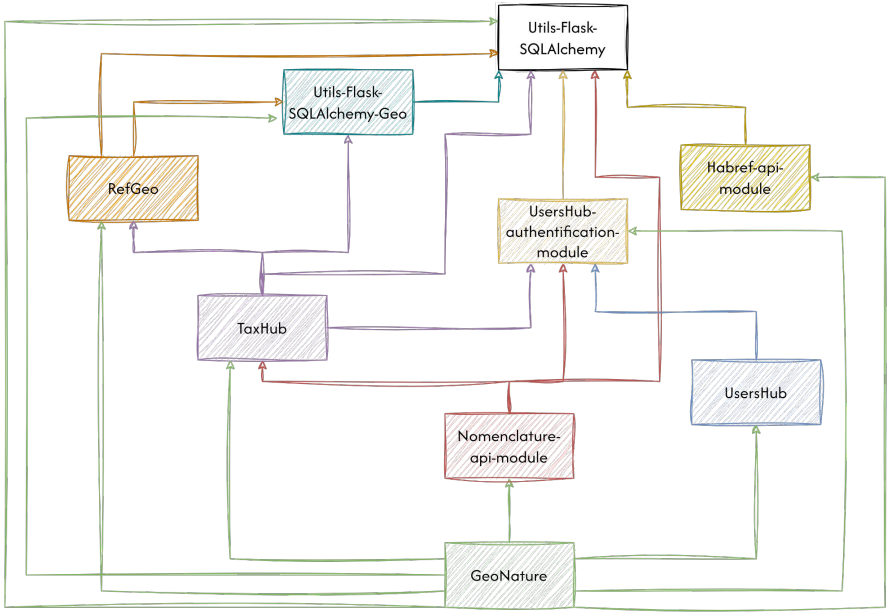


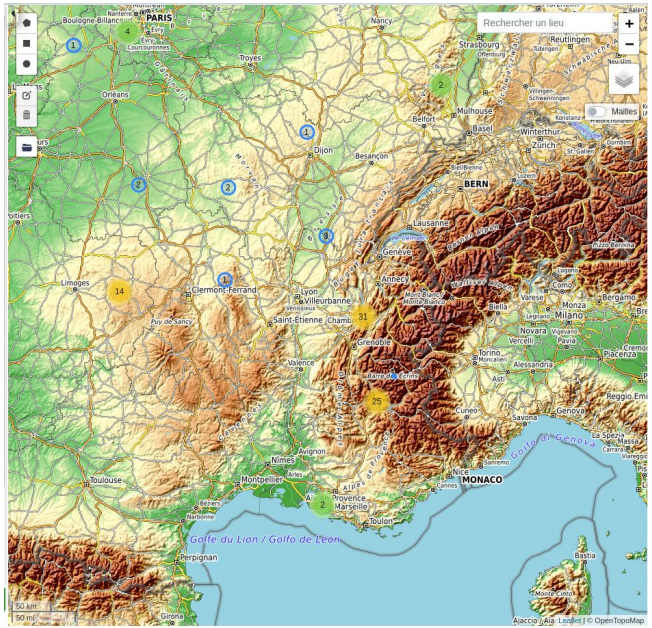
Figure. Graphique des dépendances Python dans GeoNature

[1] https://flask.palletsprojects.com/en/3.0.x/
[2] https://github.com/orgs/PnX-SI/repositories

Frontend

Pour l'interface client, nous utilisons le framework **Angular**

- Une **SinglePage Application**
- **Langage de programmation.** TypeScript
- Utilisation mixte des **framework CSS** : *Bootstrap* et *AngularMaterial* (material design v2)
- Plusieurs composants Angular permettant d'afficher :
  - Liste de nomenclatures
  - Recherche de taxons dans TaxRef
  - Liste de jeux de données
  - Carte Leaflet



https://angular.dev/

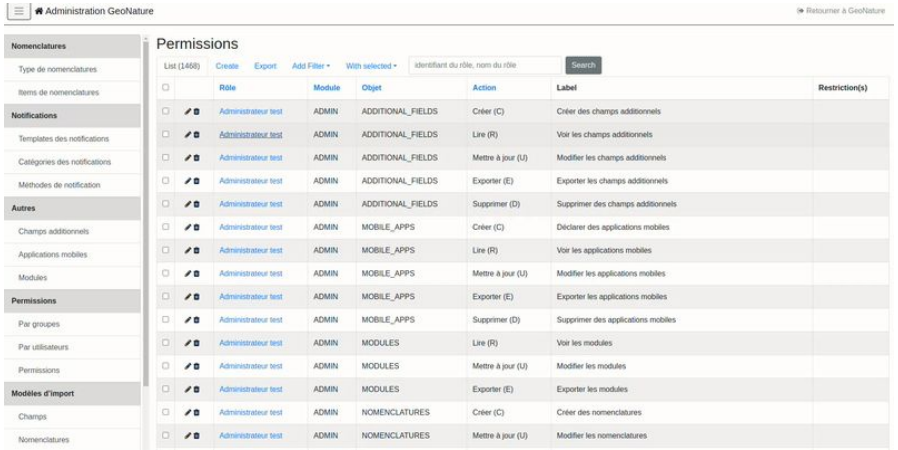
Frontend/Backend - Interface d'administration

Utilisation de l'extension **Flask-admin**

- Génération automatique d'interface admin
- Dans GeoNature
  - Gestion des permissions
  - Ajout de champs additionnels
- Customisation possible des formulaires

```python
from flask_admin.contrib.sqla import ModelView

admin = Admin(app, name='microblog',
template_mode='bootstrap3')
admin.add_view(ModelView(User, db.session))
admin.add_view(ModelView(Post, db.session))
```



Voir : https://flask-admin.readthedocs.io/en/latest/

Base de données

SQLAlchemy est un module permettant de se **connecter** et faire des **requêtes** sur une base de données de manière **agnostique**, mais aussi…

- **Modéliser** l'ensemble des tables présentes dans la BDD ainsi que leurs relations (**ORM**)
- Toutes les **requêtes** peuvent être construites à l'aide de **l'API** de SQLAlchemy

Alembic, inclus dans SQLAlchemy, permet de gérer le **versionnage** de la base de donnée

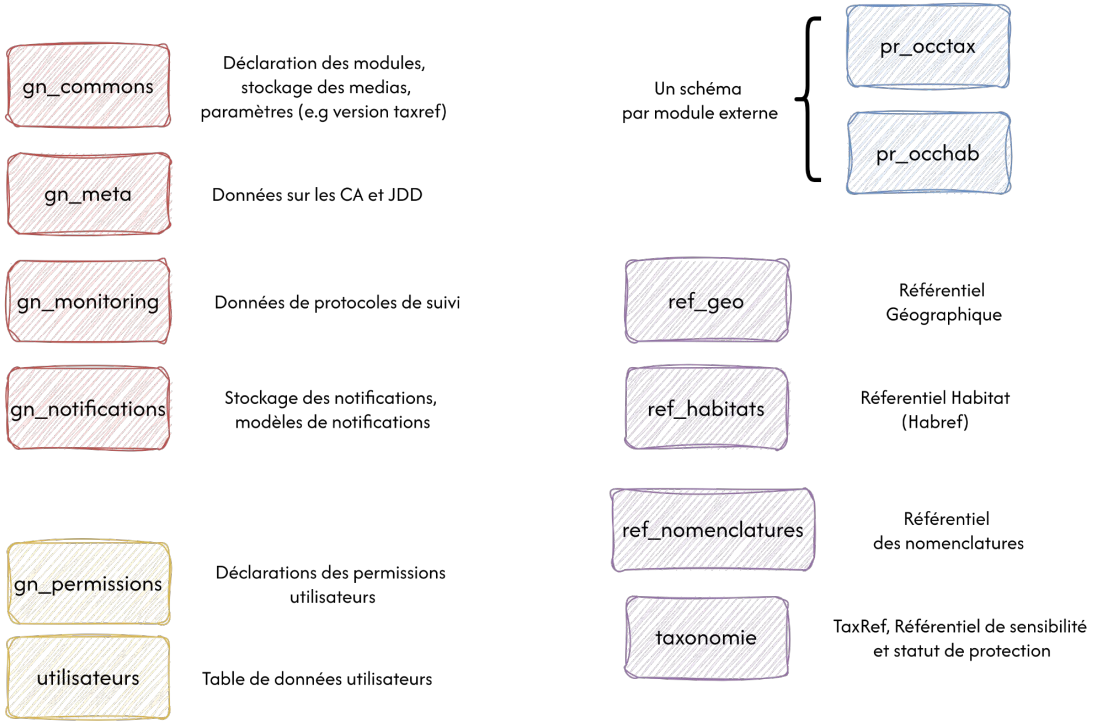

```python
class User(Base):
    __tablename__ = "user_account"
    id: Mapped[int] = mapped_column(primary_key=True)
    name: Mapped[str] = mapped_column(String(30))
    fullname: Mapped[Optional[str]]
    addresses: Mapped[List["Address"]] = relationship(
        back_populates="user", cascade="all, delete-orphan"
    )
```

Code 1. Déclaration du modèle d'une table `user_account`

```python
stmt = select(User).where(User.name.in_(["spongebob", "sandy"]))
```

Code 2. Requête à l'aide de l'API de SQLAlchemy

Architecture de la base de données



Tests automatisés

Tests unitaires (API)

- Utilisation de la librairie `pytest`

Tests End-2-End (Interface client)

- Utilisation de la librairie Cypress

Utilisée dans des actions GitHub pour aider la relecture des contributions faites par les utilisateurs.

```
    def test_get_tmedias(self):
        response = self.client.get(url_for("t_media.get_tmedias"))
        assert response.status_code == 200
        response = self.client.get(url_for("t_media.get_tmedias",
id=1))
        assert response.status_code == 200
```

Exemple de test avec *pytest*

```
it('Should search by taxa name ', function () {
   // objectifs : pouvoir rentrer un nom d'espèce dans le filtre, que cela affiche le
ou les observations sur la liste correspondant à ce nom
   cy.get('#taxonInput ').clear();
   cy.get('#taxonInput ').type('lynx');
   cy.get('ngb-typeahead-window button ').first().click({ force: true });

   cy.get('[data-qa="synthese-search-btn"] ').click();

   cy.wait(500);
   const cells = cy.get('.synthese-list-col-nom_vern_or_lb_nom ');
   cells.then((d) => {
     expect(d.length).to.greaterThan(0);
   });
   cells.each(($el, index, $list) => {
     cy.wrap($el).contains('Lynx');
   });
 });
```

Exemple de test avec *cypress*

Actions GitHub

Pour chaque commit, lancement d'actions GitHub (CI):

- Tests automatisés (unitaires + end-2-end)
- Création d'une image docker
- Vérification du formatage du code

Services systèmes associés à GeoNature

- **geonature**. Service qui fait tourner le serveur Flask
- **geonature-worker**. Service qui lance l'instance du worker Celery permettant de lancer les tâches asynchrones
- **taxhub**. Service qui fait tourner l'API Flask de TaxHub
- **usershub**. (optionnel) Permet de faire la gestion des utilisateurs. Nécessaire, si les utilisateurs souhaitent modifier leur mot de passe, le récupérer ou encore se créer un compte depuis GeoNature.

Docker

Chaque service/logiciel dans la galaxie GeoNature (TaxHub, UsersHub, … et GeoNature) propose une image docker.

Si vous souhaitez déployez GeoNature sur docker, un *docker-compose* est disponible ici : https://github.com/PnX-SI/GeoNature-Docker-services

Résumé

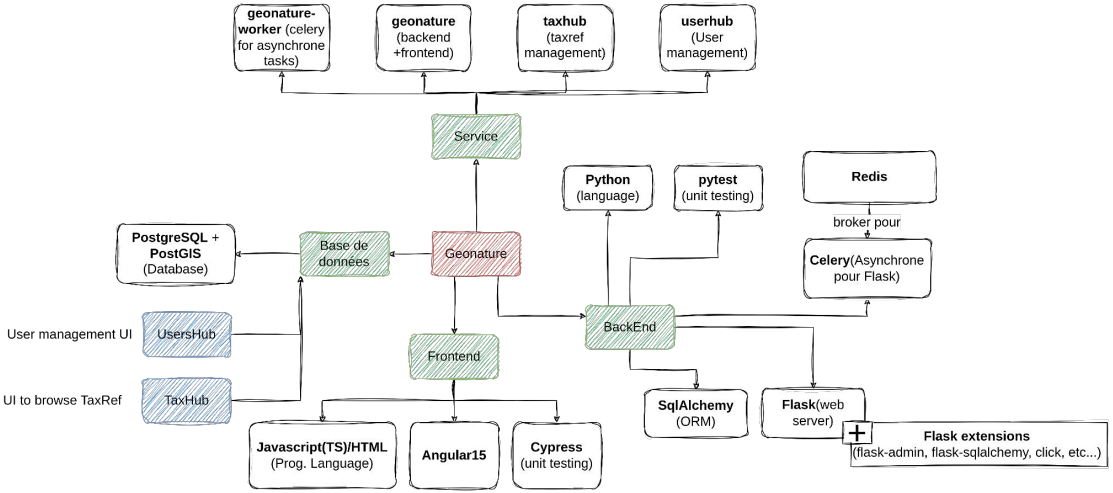
En résumé

API/Backend
- **Python** > 3.9 + **Flask**

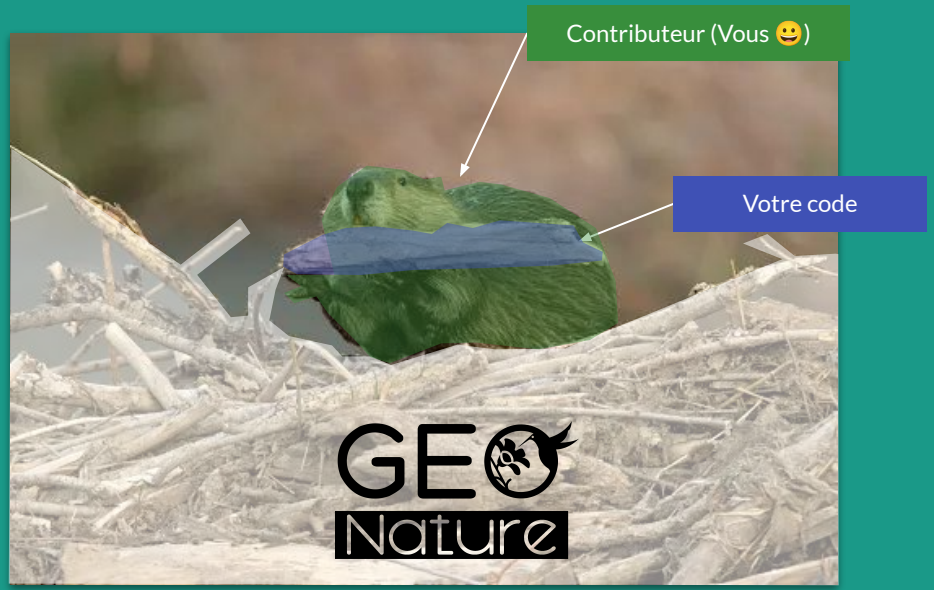
Frontend
- **Angular** v15

SGBD
- PostgreSQL + PostGIS

Plusieurs services
- TaxHub (Visualisation/Gestion TaxRef)
- UsersHub (Outils de gestion utilisateur)
- Worker Celery (tâches asynchrones)



Comment contribuer à GeoNature ?



Différents cas

Résoudre un problème (bug, régression)

- Créer une *Issue* et une *Pull Request* (PR)

Nouvelles fonctionnalités

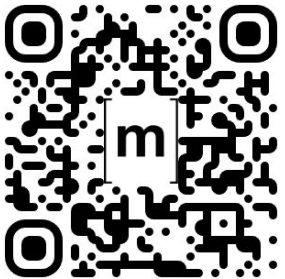
- Discussion dans une *issue* avec la communauté

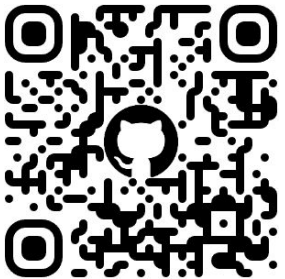
Si besoin spécifique

- Développement d'une extension/module de GeoNature

Communauté GeoNature

- Lien vers le dépôt GitHub de GeoNature : https://github.com/PnX-SI/GeoNature
- Pour discuter avec la communauté
  - Via une issue dans le dépôt GitHub
  - Via le système de discussion instantané Matrix https://matrix.to/#/#geonature:matrix.org

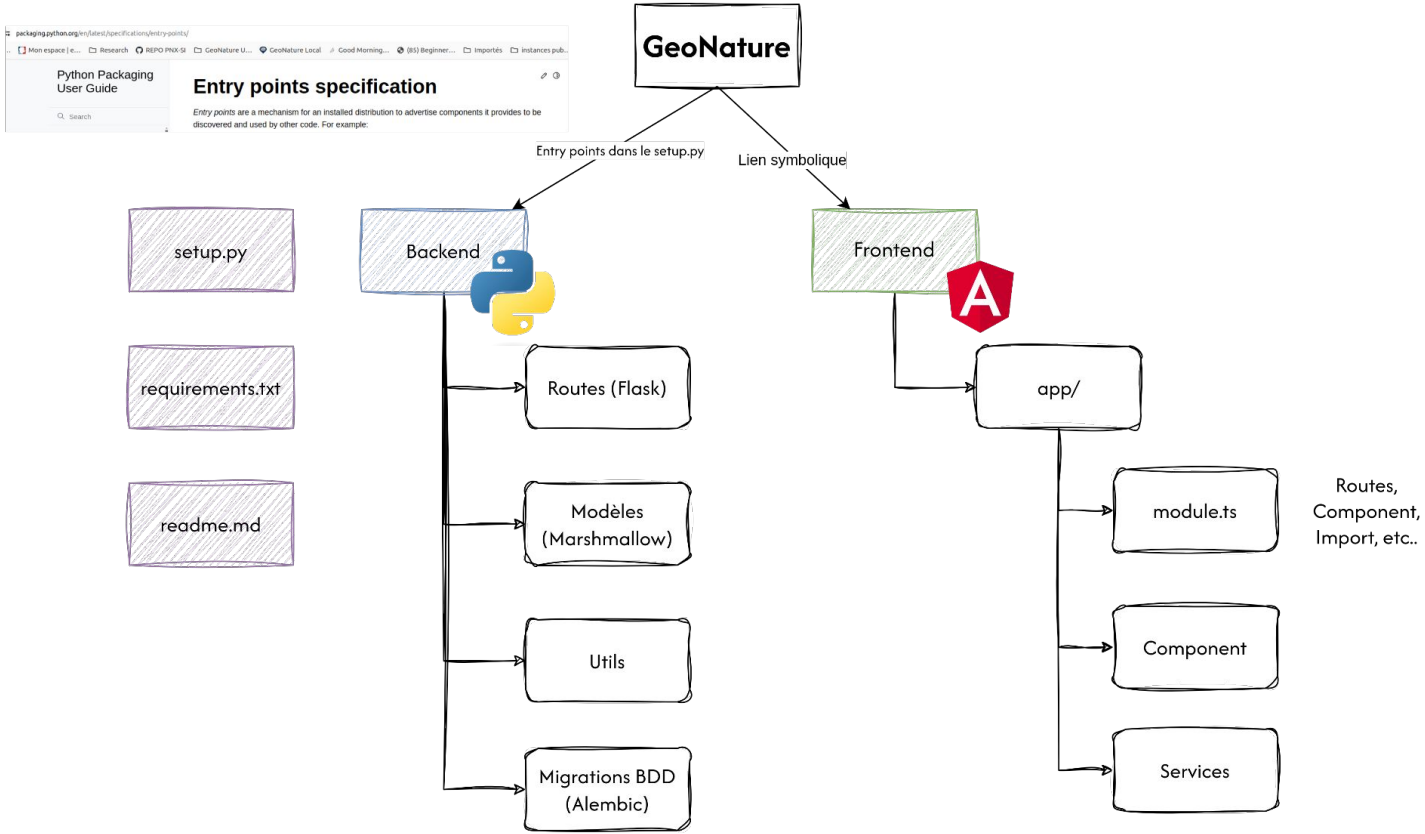




Création de module GeoNature

Pour des besoins spécifiques, il est possible de créer son propre module GeoNature

Un dépôt contient le squelette d'un module GeoNature
https://github.com/PnX-SI/gn_module_template/tree/cookiecutter-generation



Structure d'un module GeoNature

Outils

IDE. Visual Studio Code

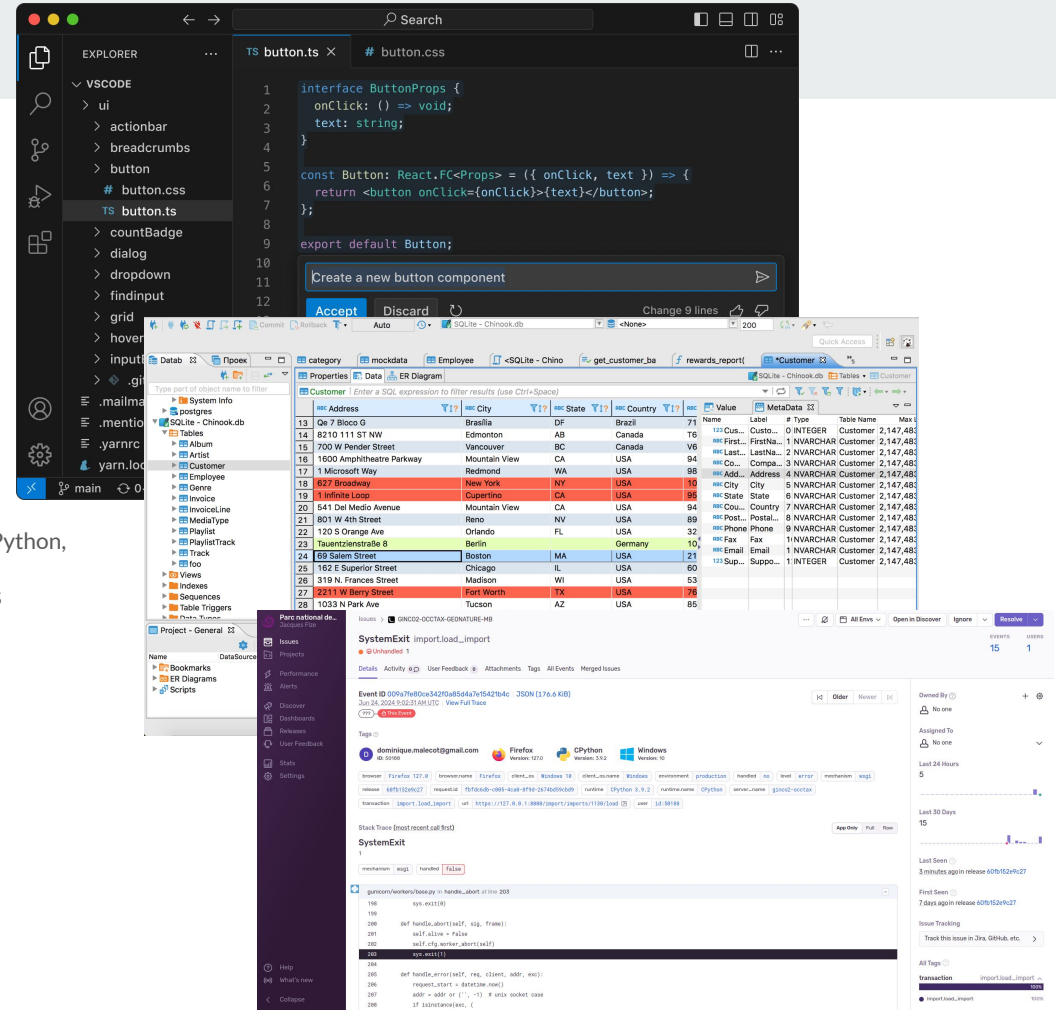
- Éditeur de texte customisable
  - Plein d'extensions pour chaque langage de programmation : Python, JS, etc…
- Intègre des extensions de formatage de code requis pour les contributions GeoNature
- Open-Source, Gratuit
- Version sans envoi de données télémétries à Microsoft : https://github.com/VSCodium/vscodium

Base de données.

- DBeaver
- PGAdmin

Monitoring d'une instance GeoNature.

- Sentry



Demo time !

Merci pour votre attention !

Lien du module GN de démo:
https://github.com/jacquesfize/gn_mapping_geonature

Vous pouvez nous contacter aux adresses suivantes

- jacques.fize@ecrins-parcnational.fr
- pierre.narcisi@mnhn.fr

Mode Dev

- Faire l'installation de GeoNature à l'aide des scripts présents dans le dossier `install`
  - Pas besoin d'exécuter le script : `*02_configure_systemd.sh*`
- Faire l'installation de **UsersHub** et **TaxHub** à l'aide des scripts d'installation
- Lancer le backend à l'aide de `*geonature-dev-back*` (après avoir sourcé l'environnement virtuel)
- Lancer le frontend à l'aide des commandes `*nvm use; npm run start*` dans le dossier `frontend`
- Lancer **TaxHub** à l'aide de la commande `*flask run*` (après avoir sourcé l'environnement virtuel)
- Lancer **UsersHub** à l'aide de la commande `*flask run*` (après avoir sourcé l'environnement virtuel)