De nouvelles connexions dans GeoNature

Jacques Fize, Théo Lechémia -- COTECH GeoNature, 9 juillet 2024

Sommaire

- Introduction
- De nouveaux fournisseurs d’identités dans GeoNature
- Adapter GeoNature à son protocole de connexion
- Démo !

Introduction

Contexte

Aujourd'hui, deux protocoles de connexion

- **En local** → UsersHub-authentification-module
- **Sur le CAS de l'INPN**

Nonobstant !

- Certaines structures ont des **SI de gestion utilisateurs/permissions spécifiques**
- Un utilisateur devant déposer des données dans plusieurs GeoNature doit avoir **plusieurs identifiants de connexions**

Proposition

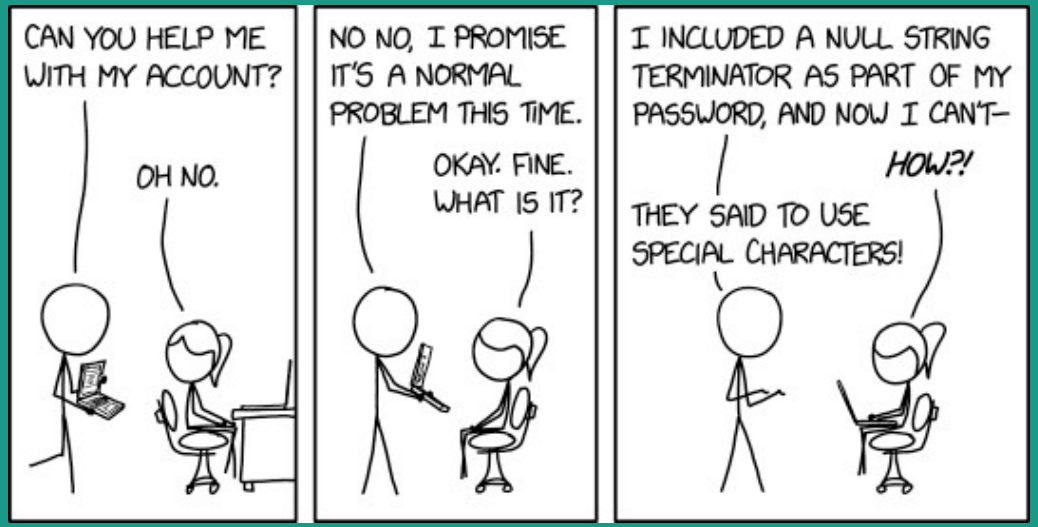
- Permettre aux administrateurs d’instance(s) GeoNature de se connecter leur gestionnaire d’utilisateurs (e.g. *keycloak*)
- Ajouter la possibilité de se connecter à un autre GeoNature

Ressources

- Ce travail est disponible dans la **PR #2976** du dépôt GeoNature

https://github.com/PnX-SI/GeoNature/pull/2976

De nouveaux fournisseurs d'identités dans GeoNature

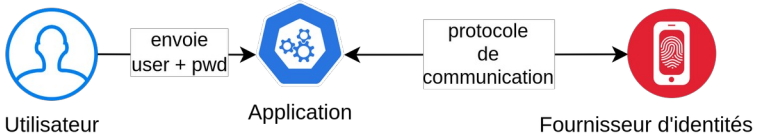


https://xkcd.com/2700/

Notions essentielles

- **Fournisseur d'identités** → Service permettant de s'identifier et qui utilise un protocole de connexion (Google, INPN, ORCID,...)
- **Protocole de connexion** → mécanisme standardisé permettant de communiquer avec un fournisseur d'identités (e.g. OAuth2, OpenIDConnect, CAS, …).

Un protocole de connexion peut être utilisée par plusieurs services.
Exemple. *OAuth2 est utilisé par Microsoft, Google, etc..*



Nouveautés

- Ajout d’un gestionnaire d’authentification (`AuthManager`)
  - Stocke les différents fournisseurs d’identités déclarés
- GeoNature vient avec plusieurs protocoles de connexions prêt à l’usage (OpenID, OpenIDC, GeoNature)
- Possibilité de créer son protocole de connexion à l’aide la classe *Authentication*

Comment ça marche ? Exemple de configuration

- Déclaration des protocoles de connexions utilisables dans `PROVIDERS`
- Pour chaque fournisseur d'identités, ajouter une section `AUTHENTICATION` dans la configuration
- Possibilité d'indiquer un mapping entre les groupes du fournisseur d'identités et votre GeoNature avec `group_mapping`

```
[AUTHENTICATION]
    PROVIDERS=["pypnusershub.auth.providers.cas_inpn_provider.
    AuthenficationCASINPN", "pypnusershub.auth.providers.openid_provider.
    OpenIDProvider",
    "pypnusershub.auth.providers.openid_provider.OpenIDConnectProvider",
    "pypnusershub.auth.providers.usershub_provider.ExternalUsersHubAuthProvider"]
    DISPLAY_DEFAULT_LOGIN_FORM = true
    DEFAULT_RECONCILIATION_GROUP_ID = 2
    # ONLY_PROVIDER = 'keycloak'
    [[AUTHENTICATION.CAS_INPN_PROVIDER]]
        id_provider="cas_inpn"
        WS_ID ="ID"
        WS_PASSWORD ="PWD"

    [[AUTHENTICATION.OPENID_CONNECT_PROVIDER_CONFIG]]
        id_provider = "keycloak"
        label = "KeyCloak"
        ISSUER = "realms"
        CLIENT_ID = "id"
        CLIENT_SECRET = "pwd"
        group_mapping = {"/user"=1,"/admin"=2}
```

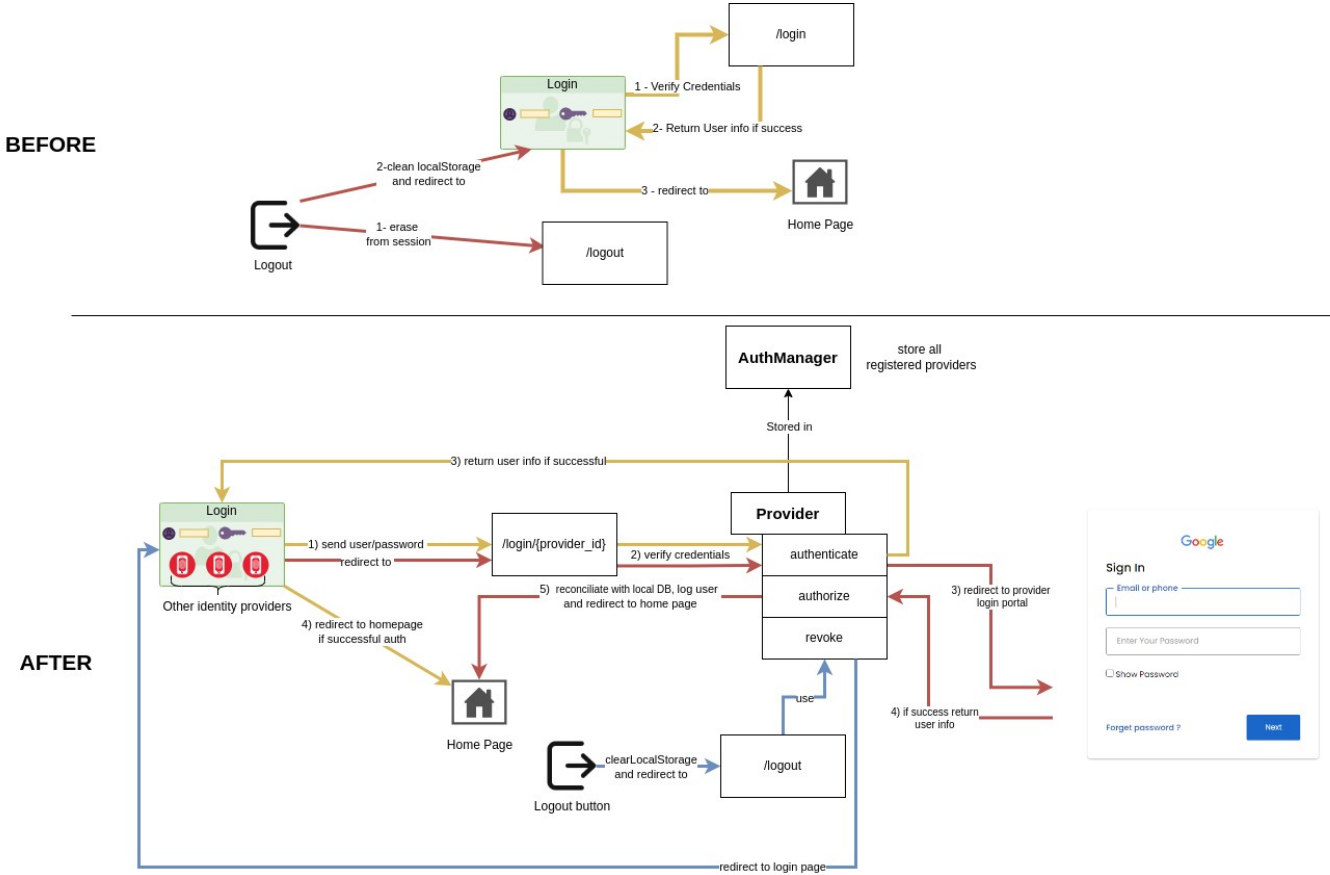


Figure. Processus de connexion à GeoNature - Avant/Après

Adapter GeoNature à son protocole de connexion

Créer votre classe provider

- Pour définir le protocole de connexion pour votre fournisseur d'identité
  - Une classe python héritant de la classe `Authentication`
- Localisation du module de la classe
  - Possibilité de l'intégrer dans une lib perso

```python
from marshmallow import Schema, fields
from typing import Any, Optional, Tuple, Union

from pypnusershub.auth import Authentication, ProviderConfigurationSchema
from pypnusershub.db import models, db
from flask import Response


Codeium: Refactor | Explain
class NEW_PROVIDER(Authentication):
    name = "NAME_IN_CONFIG"
    is_uh = False

    Codeium: Refactor | Explain | Generate Docstring | ×
    def authenticate(self, *args, **kwargs) -> Union[Response, models.User]:

        # If external, return Flask redirection to the login portal of the identiy provider
        # Else, reconciliate between the identity provider and the user
        # in the local database and return a User
        pass

    Codeium: Refactor | Explain | Generate Docstring | ×
    def authorize(self):
        # if external, reconciliation between identity provider and the local user database schema
        pass

    Codeium: Refactor | Explain | Generate Docstring | ×
    def revoke(self):
        # If specific action have to be made when logout
        pass

    Codeium: Refactor | Explain | Generate Docstring | ×
    @staticmethod
    def configuration_schema() -> Optional[Tuple[str, ProviderConfigurationSchema]]:
        # Return a marshmallow schema detailing the required configuration in the geonature_config.toml
        Codeium: Refactor | Explain
        class SchemaConf(ProviderConfigurationSchema):
            VAR = fields.String(required=True)

        return SchemaConf
    Codeium: Refactor | Explain | Generate Docstring | ×
    def configure(self, configuration: Union[dict, Any]):
        # if specific instruction have to be made when configure
        pass
```

Demo !

Perspectives

- Pour le moment
  - UsersHub est maintenu
- Mais…
  - Dans le futur, intégration de l'interface de gestion utilisateurs dans UH-AM (qui sera activable)
- Intégration d'un modèle de permissions dans UH-AM

Merci à toutes et tous pour votre attention !