

Project Octopus

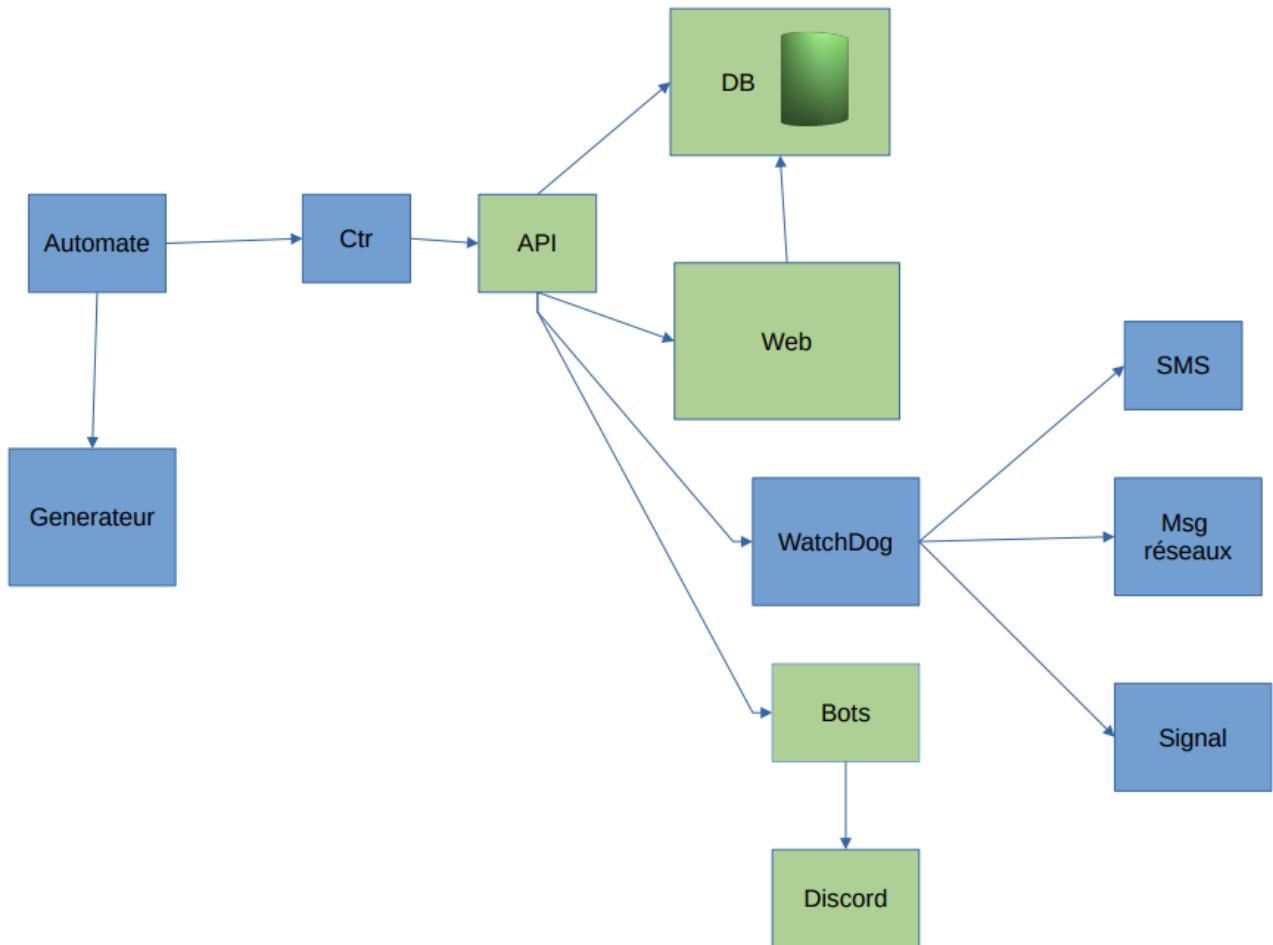
Tables des matières

Contexte:.....	2
Structure d'Octopus	2
API	2
Explication du code :	3
DB : Data Base	3
Structure de la Base de données :	3
Structure du Programme:	3
Explication du code :	4
Connexion à la base de données	4
conn.py	4
Création des classes :	4
Ecolab.py	5
Cellules.py	5
Experience.py	6
Historique.py	7
Octopus_db.py	8
Création d'object:	11
init_db.py	11

Contexte:

Le projet Octopus DB a été créé dans le but de permettre la consultation à distance de l'état des cellules, de conserver un historique des expériences menées dans ces cellules, d'ajouter de nouvelles expériences, et d'affecter des expériences spécifiques aux cellules.

Structure d'Octopus



NOTE

Voici un schéma illustrant la structure de notre projet. Les blocs en vert représentent les éléments intégrés à notre projet Octopus, tandis que les blocs en bleu ont déjà été réalisés par Mr. Shollet

API

L'API est un programme Flask qui renvoie les températures des cellules sous forme de JSON.

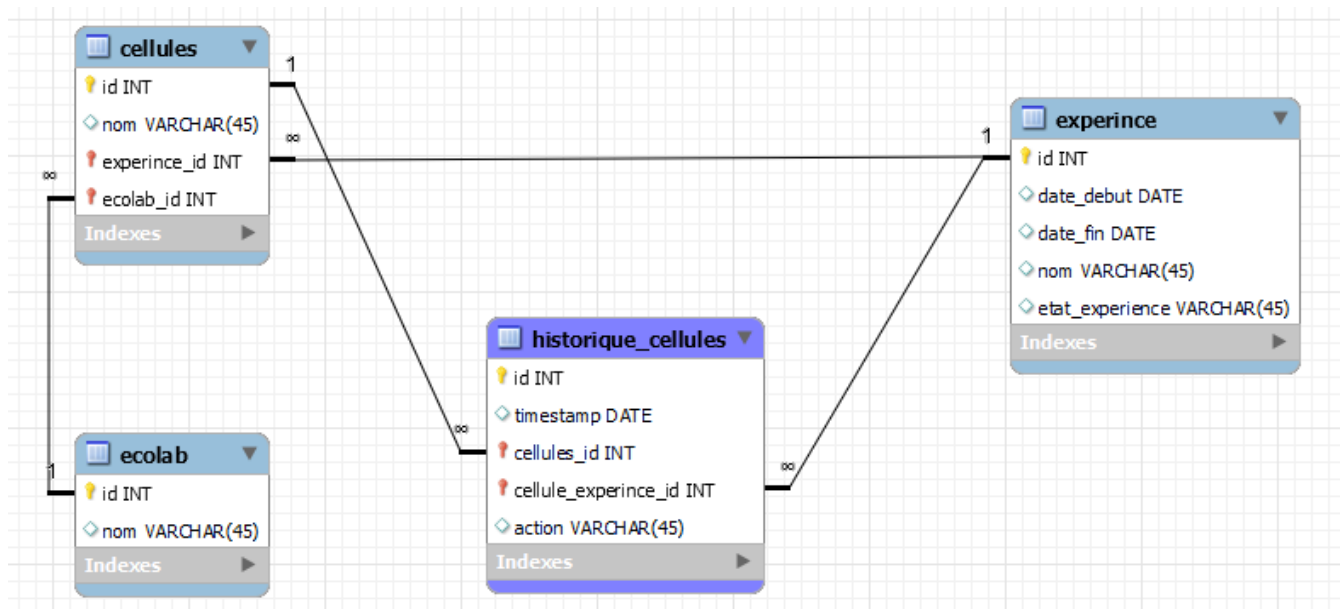
Explication du code :

Tout d'abord, nous créons un objet **Ecolab** prenant en paramètres le chemin du fichier JSON contenant les paramètres nécessaires pour interroger l'automate, et un deuxième paramètre représentant le chemin vers un fichier JSON pour les erreurs. Ensuite, nous utilisons la méthode **get_cell_param()** en lui envoyant en paramètre la **cellule** que nous souhaitons interroger et le **paramètre**. Ensuite, nous créons un dictionnaire pour l'envoyer sur le site.

NOTE:

DB : Data Base

Structure de la Base de données :



Structure du Programme:

```
Cellules.py
| conn.py
| Ecolab.py
| Experiences.py
| Historique.py
| init_db.py
| octopus_db.py
```

Explication du code :

Connexion à la base de données

conn.py

J'ai créé un fichier **conn.py** pour la connexion à la base de données avec le programme suivant.

```
'''
SQLAlchemy : pip install SQLAlchemy
Urllib :pip install urllib3
'''

#####

from sqlalchemy import create_engine
from sqlalchemy.exc import SQLAlchemyError
from sqlalchemy.ext.declarative import declarative_base
from urllib.parse import quote_plus
from sqlalchemy.orm import sessionmaker

#####

utilisateur = "Deva"
password = "sio"
base_de_donne = "octopus"
port = 3306
encoded_password = quote_plus(password)
DB_URL =
create_engine(f'mysql+pymysql://{utilisateur}:{encoded_password}@localhost:{port}/{base_de_donne}')

Session = sessionmaker(bind=DB_URL)
session = Session()
Base = declarative_base()
```

NOTE

Vous pouvez remplacer la valeur de **utilisateur** avec votre utilisateur MySQL et le **password** par votre mot de passe de l'utilisateur MySQL, puis la **base_de_données**.

Remarque : Ne pas oublier de faire les installations avec l'exécution, et l'utilisateur MySQL devrait être en **@localhost:3306**. Sinon, il faudra adapter **DB_URL**.

Création des classes :

Ensuite, j'ai créé des classes qui vont nous permettre de créer des tables dans la base de données et **Octopus_db** pour interroger la base de données avec des méthodes.

Ecolab.py

```
'''
Flask SQLAlchemy : pip install Flask-SQLAlchemy
SQLAlchemy : pip install SQLAlchemy
'''

#####

from flask_sqlalchemy import SQLAlchemy
from sqlalchemy import Column, Integer,String,DATE, ForeignKey
from sqlalchemy.orm import relationship
from flask import Flask
from conn import Base, DB_URL

#####

db = SQLAlchemy()

class Ecolab(Base):
    __tablename__ = 'ecolab'

    id = Column(Integer, primary_key=True)
    nom = Column(String(255),nullable=False)

    # Relation ONE TO MANY
    cellules = relationship('Cellule',backref='ecolab')

Base.metadata.create_all(DB_URL)
```

NOTE

Cette classe va créer une table **ecolab** avec un **id**, **nom**, **cellules** dans la base de donnée.

Remarque : **cellules** n'est pas une colonne qui apparaît dans la base de données, mais elle contient simplement une relation avec la classe **cellules**.

Cellules.py

```
'''
Flask SQLAlchemy : pip install Flask-SQLAlchemy
SQLAlchemy : pip install SQLAlchemy
'''

#####
```

```

from flask_sqlalchemy import SQLAlchemy
from sqlalchemy import Column, Integer,String,ForeignKey
from flask import Flask
from conn import Base , DB_URL
from Experiences import Experience
from Ecolab import Ecolab

#####

db = SQLAlchemy()

class Cellule(Base):
    __tablename__= 'cellules'
    id = Column(Integer, primary_key=True)
    nom = Column(String(50), nullable=False)
    experience_id = Column(Integer, ForeignKey('experience.id'))
    ecolab_id = Column(Integer,ForeignKey('ecolab.id'))

Base.metadata.create_all(DB_URL)

```

NOTE | **experience_id** et **ecolab_id** sont des clés étrangères.

Remarque : **Base.metadata.create_all(DB_URL)** permet de créer la table dans la base de données. Une fois qu'elle est créée, elle ne la recréera pas même si des modifications sont apportées aux colonnes après la création. Si vous souhaitez effectuer des modifications, vous devrez supprimer la table en ligne de commande, puis lancer le programme, ou effectuer directement des modifications avec un client MySQL.

Experience.py

```

'''
Flask SQLAlchemy : pip install Flask-SQLAlchemy
SQLAlchemy : pip install SQLAlchemy
'''

#####

from flask_sqlalchemy import SQLAlchemy
from sqlalchemy import Column, Integer,String,DATE, ForeignKey
from sqlalchemy.orm import relationship
from flask import Flask
from conn import Base, DB_URL

#####

db = SQLAlchemy()

```

```

class Experience(Base):
    __tablename__ = 'experience'

    id = Column(Integer, primary_key=True)
    date_debut = Column(DATE)
    date_fin = Column(DATE)
    nom = Column(String(255), nullable=False)
    etat_experience = Column(String(255))

    # Relation ONE TO MANY

    cellules = relationship('Cellule', backref='experience')

Base.metadata.create_all(DB_URL)

```

NOTE

La colonne **etat_experience** est présente pour indiquer si c'est une expérience qui est **Terminée**, qui est **en cours** ou qui va commencer dans les jours à **venir**.

Remarque : Une expérience peut se dérouler dans plusieurs cellules, donc c'est une relation **One To Many**.

Historique.py

```

'''
Flask SQLAlchemy : pip install Flask-SQLAlchemy
SQLAlchemy : pip install SQLAlchemy
'''

#####

from flask_sqlalchemy import SQLAlchemy
from sqlalchemy import Column, Integer, String, DATE, ForeignKey, DateTime, Enum
from Cellules import Cellule
from Experiences import Experience
from datetime import datetime
from conn import Base, DB_URL

#####

class HistoriqueCellule(Base):
    __tablename__ = 'historique_cellule'

```

```

id = Column(Integer, primary_key=True)
timestamp = Column(DateTime, default=datetime.utcnow)
cellule_id= Column(Integer, ForeignKey('cellules.id'))
cellule_experience_id = Column(Integer, ForeignKey('experience.id'))
status = Column(String(255), nullable=False)
action = Column(String(255), nullable=False)

```

```
Base.metadata.create_all(DB_URL)
```

NOTE

La colonne **status** concerne uniquement l'expérience dans la cellule. Si l'expérience est terminée, elle affichera **Terminér**. Si elle est en cours, elle s'affichera comme **En cours**, et si elle est à venir, elle sera indiquée comme **À venir**.

Octopus_db.py

```

'''
Flask : pip install SQLAlchemy
'''

#####

# Importation des modules nécessaires
from conn import session
from Experiences import Experience
from Cellules import Cellule
from Historique import HistoriqueCellule
from sqlalchemy.orm import joinedload
from pprint import pprint

#####

# Auteur: Deva

class OctopusDB:
    def __init__(self):
        self.experiences = session.query(Experience).all()
        self.cellules = session.query(Cellule).all()
        self.historique = session.query(HistoriqueCellule).all()

    # Méthode pour récupérer toutes les expériences de la base de données
    def get_all_experience(self):
        global session

```



```

        experiences = session.query(Experience).all()
        session.commit()
        return experiences

# Méthode pour récupérer le nom d'une cellule à partir de son ID
def get_cellule_name_from_id(self, id_cellule):
    for cellule in self.cellules:
        if cellule.id == id_cellule:
            return cellule.nom

# Méthode pour récupérer une cellule à partir de son nom
def get_cellule_by_name(self,name):
    for cellule in self.cellules:
        if cellule.nom == name:
            return cellule
    return None

# Méthode pour récupérer une cellule à partir de son ID
def get_cellule_by_id(self,id_cellule):
    for cellule in self.cellules:
        if cellule.id == id_cellule:
            return cellule

# Méthode pour récupérer les expériences actuelles et futures
def get_futur_and_current_experience(self):
    result = []
    for experience in self.experiences:
        if experience.etat_experience == "à venir" or experience.etat_experience
== "En cours":
            result.append(experience)
    return result

# Méthode pour récupérer une expérience à partir de son ID
def get_experience_by_id(self,id_experience):
    try :
        for experience in self.experiences:
            if experience.id == id_experience:
                return experience
    except Exception as e:
        print(f"Erreur lors de la récupération de l'expérience par ID : {str(e)}")
        return None

# Méthode pour récupérer l'historique d'une cellule à partir de son ID
def get_historique_by_id(self,cellule_id):
    global session
    historiques = session.query(HistoriqueCellule).all()
    result = []
    for historique in historiques:
        if historique.cellule_id == cellule_id:

```

```

        cellule = self.get_cellule_by_id(historique.cellule_id)
        experience =
self.get_experience_by_id(historique.cellule_experience_id)
        result.append({"historique": historique, "cellule": cellule,
"experience": experience})
        session.commit()
        return result

# Méthode pour récupérer l'expérience d'une cellule à partir de son ID
def get_experience_of_cellule(self, cellule_id):
    for cellule in self.cellules:
        if cellule.id == cellule_id:
            return self.get_experience_by_id(cellule.experience_id)
    return None

# Méthode pour récupérer une expérience à partir de son ID
def get_experience_by_id(self, id):
    for experience in self.experiences:
        if experience.id == id :
            return experience
    return None

# Méthode pour mettre à jour l'expérience d'une cellule
def new_experience_of_cellule(self, id_cellule, id_experience):
    global session
    try :
        cellule = self.get_cellule_by_id(id_cellule)
        cellule.experience_id = id_experience
        session.commit()
        return "Mise à jour réussie"
    except Exception as e:
        return f"Une erreur s'est produite : {str(e)}"

# Méthode pour créer un nouvel historique pour une cellule
def new_historique(self, id_cellule, id_experience):
    global session
    try:
        new_historique =
HistoriqueCellule(cellule_id=id_cellule, cellule_experience_id=id_experience, status="En
cours", action="Ajout d'une nouvelle expérience à la cellule")
        session.add(new_historique)
        session.commit()
        return "c'est bon"
    except Exception as e:
        return f"Une erreur s'est produite : {str(e)}"

# Méthode pour mettre à jour l'historique d'une cellule
def update_historique(self, cellule_id):
    global session
    historiques = session.query(HistoriqueCellule).all()

```

```

        try :
            for historique in historiques:
                if historique.cellule_id == cellule_id and historique.status == "En
cours":
                    historique.status = "Terminés"
                    session.commit()
                return "mise a jour reussi !"
            except Exception as e:
                return f"Une erreur s'est produite : {str(e)}"

# Création d'une instance de la classe OctopusDB
octopus = OctopusDB()

```

Remarque: Dans la méthode **get_all_experience()**, je parcours à nouveau la base de données avec **session.query(Experience).all()** parce que lorsque je crée de nouvelles **expérience**, l'attribut **self.experiences** que j'ai ne contient pas la nouvelle version de la base de données. Par conséquent, je réinterroge la base de données pour obtenir la version la plus récente.

Création d'object:

init_db.py

Une fois que la base de données est prête, j'ai créé des objets pour les insérer dans la base de données de la manière suivante

```

#####
#####

from Ecolab import Ecolab
from Experiences import Experience
from Cellules import Cellule
from Historique import HistoriqueCellule
from conn import utilisateur,session, password, base_de_donne, port, Base

#####
#####

# Auteur: Deva

'''Objets des experiences'''
embl= Experience(date_debut='2023/12/01' ,nom="EMBL-EXPOCER",etat_experience="En
cours")
gente= Experience(date_debut='2023/08/01' ,nom="Gente-POP",etat_experience="En cours")
mang= Experience(date_debut='2023/09/01' ,nom="MangWarm",etat_experience="En cours")
b2= Experience(date_debut='2024/01/01' ,nom="B2",etat_experience="En cours")
mines= Experience(date_debut='2023/11/01' ,nom="Mines-Pluie",etat_experience="En
cours")

```

```

peace= Experience(nom="Peace",etat_experience="à venir")
lysometre= Experience(nom="Lysimètre",etat_experience="à venir")
bacs= Experience(nom="Bacs à évaporation",etat_experience="à venir")

sybio= Experience(nom="Sybio",etat_experience="à venir")
optisol= Experience(nom="Optisol",etat_experience="à venir")
microbe= Experience(nom="Microbe For",etat_experience="à venir")
climate= Experience(nom="Climate",etat_experience="à venir")
agroserv= Experience(nom="AgroServ",etat_experience="à venir")

pasDexperience = Experience(nom="pas d'experience")

bioteca = Experience(date_debut='2023/04/01', date_fin='2023/09/01',
nom="BIOTECA",etat_experience="Terminés")
oasis = Experience(date_debut='2022/10/01', date_fin='2023/01/01',
nom="Oasis",etat_experience="Terminés")
riboStress = Experience(date_debut='2019/01/01', date_fin='2023/08/01',
nom="RiboStress",etat_experience="Terminés")
aquaOXY = Experience(date_debut='2023/01/01', date_fin='2023/03/01',
nom="AquaOXY",etat_experience="Terminés")
ehinzode = Experience(date_debut='2023/02/01', date_fin='2023/09/01',
nom="Rhizode",etat_experience="Terminés")
syvie = Experience(date_debut='2023/10/01', nom="Syvie",etat_experience="Terminés")

''' Objetsde Cellules'''
E1C1 = Cellule(nom="E1C1", ecolab_id=1, experience_id=1)
E1C2 = Cellule(nom="E1C2", ecolab_id=1, experience_id=2)
E1C3 =Cellule(nom="E1C3", ecolab_id=1, experience_id=1)

E2C1 = Cellule(nom="E2C1", ecolab_id=2, experience_id=3)
E2C2 = Cellule(nom="E2C2", ecolab_id=2, experience_id=3)
E2C3 =Cellule(nom="E2C3", ecolab_id=2)

E3C1 = Cellule(nom="E3C1", ecolab_id=3, experience_id=2)
E3C2 = Cellule(nom="E3C2", ecolab_id=3, experience_id=2)
E3C3 =Cellule(nom="E3C3", ecolab_id=3, experience_id=2)

E4C1 = Cellule(nom="E4C1",ecolab_id=4)
E4C2 = Cellule(nom="E4C2", ecolab_id=4, experience_id=3)
E4C3 = Cellule(nom="E4C3", ecolab_id=4 ,experience_id=3)

E5C1 = Cellule(nom="E5C1", ecolab_id=5)
E5C2 = Cellule(nom="E5C2", ecolab_id=5)
E5C3 = Cellule(nom="E5C3", ecolab_id=5)

E6C1 = Cellule(nom="E6C1",ecolab_id=6, experience_id=4)
E6C2 = Cellule(nom="E6C2", ecolab_id=6,experience_id=5)
E6C3 = Cellule(nom="E6C3",ecolab_id=6)

```

```

Ecolab_1 = Ecolab(nom ="Ecolab 1")
Ecolab_2 = Ecolab(nom ="Ecolab 2")
Ecolab_3 = Ecolab(nom ="Ecolab 3")
Ecolab_4 = Ecolab(nom ="Ecolab 4")
Ecolab_5 = Ecolab(nom ="Ecolab 5")
Ecolab_6 = Ecolab(nom ="Ecolab 6")

'''Objets des historiques des cellules'''
historique_1 = HistoriqueCellule(cellule_id=4,cellule_experience_id=14)
historique_2 = HistoriqueCellule(cellule_id=5,cellule_experience_id=14,
status="Terminér")

historique_3 = HistoriqueCellule(cellule_id=2,cellule_experience_id=15,
status="Terminér")

historique_4 = HistoriqueCellule(cellule_id=13,cellule_experience_id=16,
status="Terminér")
historique_5 = HistoriqueCellule(cellule_id=14,cellule_experience_id=16,
status="Terminér")
historique_6 = HistoriqueCellule(cellule_id=15,cellule_experience_id=16,
status="Terminér")

historique_7 = HistoriqueCellule(cellule_id=1,cellule_experience_id=17,
status="Terminér")

historique_8 = HistoriqueCellule(cellule_id=1,cellule_experience_id=18,
status="Terminér")
historique_9 = HistoriqueCellule(cellule_id=2,cellule_experience_id=18,
status="Terminér")
historique_10 = HistoriqueCellule(cellule_id=3,cellule_experience_id=18,
status="Terminér")

historique_11 = HistoriqueCellule(cellule_id=6,cellule_experience_id=19,
status="Terminér")

historique_12 = HistoriqueCellule(cellule_id=1,cellule_experience_id=1, status="En
cours")
historique_13 = HistoriqueCellule(cellule_id=3,cellule_experience_id=1, status="En
cours")

historique_14 = HistoriqueCellule(cellule_id=2,cellule_experience_id=2, status="En
cours")
historique_15 = HistoriqueCellule(cellule_id=7,cellule_experience_id=2, status="En
cours")
historique_16 = HistoriqueCellule(cellule_id=8,cellule_experience_id=2, status="En
cours")
historique_17 = HistoriqueCellule(cellule_id=9,cellule_experience_id=2, status="En
cours")

historique_18 = HistoriqueCellule(cellule_id=4,cellule_experience_id=3, status="En
cours")

```

```

historique_19 = HistoriqueCellule(cellule_id=5,cellule_experience_id=3, status="En
cours")
historique_20 = HistoriqueCellule(cellule_id=11,cellule_experience_id=3, status="En
cours")
historique_21 = HistoriqueCellule(cellule_id=12,cellule_experience_id=3, status="En
cours")

historique_22 = HistoriqueCellule(cellule_id=16,cellule_experience_id=4, status="En
cours")

historique_23 = HistoriqueCellule(cellule_id=17,cellule_experience_id=5, status="En
cours")

'''Historique'''
#session.add_all([historique_1,historique_2,historique_3,historique_4,historique_5,his
torique_6,historique_7,historique_8,historique_9,historique_10,historique_11,historiqu
e_12,historique_13,historique_14,historique_15,historique_16,historique_17,historique_
18,historique_19,historique_20,historique_21,historique_22,historique_23])

#session.add(historique_1)
'''Cellules'''
#session.add_all([E1C1,E1C2,E1C3,E2C1,E2C2,E2C3,E3C1,E3C2,E3C3,E4C1,E4C2,E4C3,E5C1,E5C
2,E5C3,E6C1,E6C2,E6C3])

'''Experience'''
#session.add_all([embl, gente, mang, b2, mines, peace, lysimetre, bacs, sybio,
optisol, microbe, climate, agroserv, bioteca, oasis, riboStress, aqua0XY, ehinzode,
syvie])

'''Ecolab'''
#session.add_all([Ecolab_1,Ecolab_2,Ecolab_3,Ecolab_4,Ecolab_5,Ecolab_6])

session.commit()

```

Remarque: Il faudra décommenter la ligne `session.add_all()` pour les insérer dans la base de données. Une fois qu'ils sont insérés, il faudra commenter ou effacer cette ligne. Sinon, à chaque relancement, les objets seront réinsérés dans la base de données, entraînant des duplications indésirables.