Lucas NGUYEN
Tom CALVO
Dorian JOSSERAND

WhatsAPIv2

1. A quoi va servir l'API? HUB

WhatsAPI est une API permettant de faciliter la réalisation d'une application de messagerie. L'API permettra notamment :

- ☑ ~~Lister les utilisateurs de WhatsAPI.~~
- ☑ ~~Ajouter un utilisateur de WhatsAPI.~~
- ☑ ~~Afficher toutes les informations relatives à un utilisateur précis.~~
- ☐ Voir nos amis
- ☐ Ajouter un ami sur WhatsAPI
- ☐ Supprimer un ami

2. Schéma relationnel de la BDD Description de la structure de la base de données.

INFO des contraintes ou d'autres :
BDD :

- UTILISATEUR(id_utilisateur: INT, pseudo_utilisateur: TEXT)
- AMITIE(#util1:TEXT,#util2:TEXT)

3. Création de la BDD Ecriture du fichier .sql permettant d'initialiser la BDD.

```
DROP TABLE IF EXISTS UTILISATEUR;
DROP TABLE IF EXISTS AMITIE;

CREATE TABLE UTILISATEUR(
    id_utilisateur INTEGER PRIMARY KEY AUTOINCREMENT,
    pseudo_utilisateur varchar(20) UNIQUE
);

CREATE TABLE AMITIE(
      util1 varchar(20),
      util2 varchar(20),
      CONSTRAINT AMITIE PRIMARY KEY (util1,util2),
      CONSTRAINT AMITIE FOREIGN KEY(util1) REFERENCES UTILISATEUR
(pseudo_utilisateur)
      CONSTRAINT AMITIE FOREIGN KEY(util2) REFERENCES UTILISATEUR
(pseudo_utilisateur)
);
```

4. Documentation de l'API Ecriture des spécifications des **nouvelles** fonctions de l'API

```
NAME
       WhatsApi

FONCTIONS
        voir_amis(pseudo_utilisateur):
        ajouter_amis(pseudo_utilisateur,pseudo_utilisateur_a_ajouter)
        supprimer_amis(pseudo_utilisateur,pseudo_utilisateur_a_supp)

DESCRIPTION

        voir_amis(pseudo_utilisateur):
        Cette fonction permet d'afficher le pseudo des personnes qui sont les amis de "pseudo_utilisateur"

        INPUT
            pseudo_utilisateur type STR

        OUTPUT

            Type Dict {
                status :
                data : []
                }

            status :
                0 -> pseudo_utilisateur type STR, . . .
                1 -> 'INPUT Type not STR'
                2 -> 'INPUT Lenght not between 5, 20'
                3 -> 'INPUT Not registered in UTILISATEUR'

    ajouter_amis(pseudo_utilisateur,pseudo_utilisateur_a_ajouter)
        Cette fonction permet d'ajouter un amis a un utilisateur précis.

        INPUT
            pseudo_utiur type STR
            pseudo_utilisateur_a_ajouter type STR

        OUTPUT

            Type Dict {
                status :
                data : []
                }

            status :
                0 -> None
                1 -> 'INPUT Type not STR'
                2 -> 'INPUT Lenght not between 5, 20'
                3 -> 'INPUT Not register in UTILISATEUR'
                4 -> 'INPUT already register as friend'
                5 -> 'INPUT are the same'

    supprimer_amis(pseudo_utilisateur,pseudo_utilisateur_a_supp)
        Cette fonction permet de supprimer un ami choisit.

        INPUT
            pseudo_utilisateur type STR
            pseudo_utilisateur_a_supp type STR

        OUTPUT

            Type Dict {
                status :
                data : []
                }

            status :
                0 -> None
                1 -> 'INPUT Type not STR'
                2 -> 'INPUT Lenght not between 5, 20'
                3 -> 'INPUT Not register in UTILISATEUR’
                4 -> 'INPUT Not in friendlist'

AUTHOR
       Written by Tom CALVO, Dorian JOSSERAND and Lucas NGUYEN.


COPYRIGHT
       Copyright (c) 2022 WhatsAPI
       This is free software: you are free to change and redistribute it.  There is NO WARRANTY, to the extent permitted by law.
```

5. Écriture des tests (assert ou testmod) qui vérifient les spécifications. Il peut être utile de créer un fichier test.sql pour remplir la BDD avec des valeurs dédiées aux tests.

```python
import sqlite3
from list_util import list_util
from ajouter_util import ajouter_util
from info_util import info_util


DB_FILE = 'BDDv1.db'
SQL_FILE = 'BDDv1.sql'

def execution_SQL(SQL_FILE):
    global DB_FILE

    with open(SQL_FILE, 'r') as f :
        createSql = f.read()
    # Placement des requêtes dans un tableau
    sqlQueries = createSql.split(";")

    try:
        # Ouverture de la connexion avec la bdd
        conn = sqlite3.connect(DB_FILE)
        #On active les foreign key
        conn.execute("PRAGMA foreign_keys = 1")
    except sqlite3.Error as e:
        print(e)
    # Execution de toutes les requêtes du tableau
    cursor = conn.cursor()
    for query in sqlQueries:
        cursor.execute(query)
    # commit des modifications
    conn.commit()
    # fermeture de la connexion
    conn.close()

if __name__ == "__main__":
    DB_FILE = 'BDDtest.db'
    execution_SQL("BDD.sql")
    #Teste de voir_amis()
    Status 0
    assert voir_amis("Tom.clv") == {'status' : 0, 'data' : []}
    execution_SQL("BDDtest2.sql")
    execution_SQL("BDDtest3.sql")
    assert voir_amis("Tom.clv") == {'status' : 0, 'data' : ['Dorian.jsr']}
    assert voir_amis('Dorian.jsr') == {'status' : 0, 'data' : ["Tom.clv"]}
    execution_SQL("BDD.sql")
    assert voir_amis("Tom.clv") == {'status' : 0, 'data' : []}
    #Status 1
    assert voir_amis(5) == {'status' : 1, 'data' : ['INPUT Type not STR']}
    #Status 2
    assert voir_amis("123456789101112131415161718192O") == {'status' : 2, 'data' : ['INPUT Lenght not between 5, 20']}
    assert voir_amis("123") == {'status' : 2, 'data' : ['INPUT Lenght not between 5, 20']}
    #Status 3
    assert voir_amis("Josseline") == {'status' : 3, 'data' : ['INPUT Registered in UTILISATEUR']}

    #Teste de ajouter_amis()
    #Status 0
    assert ajouter_amis('Tom.clv','Nyn.luk') == {"status" : 0, "data": []}
    #Status 1
    assert ajouter_amis('Dorian.jsr',1880) == {"status" : 1, "data" : ['INPUT type not STR']}
    assert ajouter_amis(1880,'Dorian.jsr') == {"status" : 1, "data" : ['INPUT type not STR']}
    #Status 2
    assert ajouter_amis('Tom.clv','Le.Boulanger.Qui.Fait.Du.Pain') == {"status" : 2, "data" : ['INPUT Length not between 5, 20']}
    assert ajouter_amis('Le.Boulanger.Qui.Fait.Du.Pain','Tom.clv') == {"status" : 2, "data" : ['INPUT Length not between 5, 20']}
    assert ajouter_amis('Nyn.luk',' ') == {"status" : 2, "data" : ['INPUT Length not between 5, 20']}
    assert ajouter_amis(' ','Nyn.luk') == {"status" : 2, "data" : ['INPUT Length not between 5, 20']}
    #Status 3
    assert ajouter_amis('Tom.clv','Xx_Michelle_xX') == {"status" : 3, "data": ['INPUT Not register in UTILISATEUR']}
    assert ajouter_amis('Xx_Michelle_xX','Tom.clv') == {"status" : 3, "data": ['INPUT Not register in UTILISATEUR']}
    #Status 4
    assert ajouter_amis('Tom.clv','Dorian.jsr') == {"status" : 4, "data": ['INPUT already register as friend']}
    #Status 4
    assert ajouter_amis('Tom.clv','Tom.clv') == {"status" : 5, "data": ['INPUT are the same']}

    #Teste de supprimer_amis
    #status 0
    assert supprimer_amis('Tom.clv','Dorian.jsr') == {"status" : 0, "data": []}
    #status 1
    assert supprimer_amis('Dorian.jsr',1880) == {"status" : 1, "data" : ['INPUT type not STR']}
    assert supprimer_amis(1880,'Dorian.jsr') == {"status" : 1, "data" : ['INPUT type not STR']}
    #status 2
    assert supprimer_amis('Tom.clv','Le.Boulanger.Qui.Fait.Du.Pain') == {"status" : 2, "data" : ['INPUT Length not between 5, 20']}
    assert supprimer_amis('Le.Boulanger.Qui.Fait.Du.Pain','Tom.clv') == {"status" : 2, "data" : ['INPUT Length not between 5, 20']}
    assert supprimer_amis('Nyn.luk',' ') == {"status" : 2, "data" : ['INPUT Length not between 5, 20']}
    assert supprimer_amis(' ','Nyn.luk') == {"status" : 2, "data" : ['INPUT Length not between 5, 20']}
    #status 3
    assert supprimer_amis('Nyn.luk','util.inexistant') == {"status" : 3, "data": ['INPUT Not in database']}
    assert supprimer_amis('util.inexistant','Nyn.luk') == {"status" : 3, "data": ['INPUT Not in database']}
    #status4
    assert supprimer_amis('Dorian.jsr','Nyn.luk') == {"status" : 4, "data" : ['INPUT Not in friendlist']}

    DB_FILE = 'BDD.db'
```

BDD.sql

```sql
DROP TABLE IF EXISTS UTILISATEUR;
DROP TABLE IF EXISTS AMITIE;

CREATE TABLE UTILISATEUR(
    id_utilisateur INTEGER PRIMARY KEY AUTOINCREMENT,
       pseudo_utilisateur varchar(20) UNIQUE
);

CREATE TABLE AMITIE(
       util1 varchar(20),
       util2 varchar(20),
       CONSTRAINT AMITIE PRIMARY KEY (util1,util2),
       CONSTRAINT AMITIE FOREIGN KEY(util1) REFERENCES UTILISATEUR (pseudo_utilisateur)
       CONSTRAINT AMITIE FOREIGN KEY(util2) REFERENCES UTILISATEUR (pseudo_utilisateur)
);
```

BDDtest1.sql :

```sql
DELETE from AMITIE;
DELETE from UTILISATEUR;
```

BDDtest2.sql :

```sql
INSERT INTO UTILISATEUR
(id_utilisateur, pseudo_utilisateur)
VALUES
(1,'Tom.clv'),
(2,'Dorian.jsr'),
(3,'Nyn.luk')
```

BDDtest3.sql :

```sql
INSERT INTO AMITIE
(util1 , util2)
VALUES
('Dorian.jsr','Tom.clv'),
('Tom.clv','Dorian.jsr');
```

Pour les autres fonctions pas nécessaire car elles n'ont pas d'INPUT ou bien qu'elles n'ont qu'à vérifier si la requête est bonne => si le nom d'utilisateur existe.

6. Codage de l'API Codage des fonctions que vous avez spécifiées

- [x] ~~Lister les amis d'un utilisateur de WhatsAPI.~~ Dorian ~~Terminé~~
- [x] ~~Ajouter un ami d'un utilisateur de WhatsAPI.~~ Tom ~~Terminé~~
- [ ] Supprimer un ami Lucas Bloqué

```python
os.remove("BDDtest.db")
```

7. Passage des tests Vérification que les tests passent

☐ Terminé

8. Intégration Rassemblement du travail des membres du groupe dans le projet principal.

☐ Terminé