Lucas NGUYEN
Tom CALVO
Dorian JOSSERAND

WhatsAPI

1. A quoi va servir l'API?

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.

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

BDD :

- UTILISATEUR(id_utilisateur: INT, pseudo_utilisateur: TEXT)

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

```sql
DROP TABLE IF EXISTS UTILISATEUR;

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


INSERT INTO UTILISATEUR
(id_utilisateur, pseudo_utilisateur)
VALUES
(1,'Tom.clv'),
(2,'Dorian.jsr');
```

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

```
NAME
       WhatsApi

FONCTIONS
        list_util()
        ajouter_util()
        info_util()


DESCRIPTION

    list_util()
        Cette fonction permet d'afficher tous les utilisateurs de WhatsAPI.

        OUTPUT

            Type Dict {
                status :
                data : []
                }

            status -> data :
                0 -> [{id_utilisateur : (type INT), pseudo_utilisateur : (type STR)}, . . . ]
                1 -> 'INPUT Type incorrect'
                2 -> 'INPUT Length incorrect'
                3 -> 'INPUT Not Correspond'




    ajouter_util(pseudo_utilisateur)
        Cette fonction permet d'ajouter un nouvel utilisateur à la base de donnée.

        INPUT
            pseudo_utilisateur type STR

        OUTPUT

            Type Dict {
                status :
                data : []
                }

            status -> data :
                0 -> None
                1 -> 'INPUT Type not STR'
                2 -> 'INPUT Length not between 5, 20'
                3 -> 'INPUT already in database'


    info_util(pseudo_utilisateur)
        Cette fonction permet d'afficher les informations relatives à un utilisateur.

        INPUT
            pseudo_utilisateur type STR

        OUTPUT

            Type Dict {
                status :
                data : []
                }

            status :
                0 -> (id_utilisateur type INT, pseudo_utilisateur type STR)
                1 -> 'INPUT Type not STR'
                2 -> 'INPUT Length not between 5, 20'
                3 -> 'INPUT Not in database'

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.

```
import sqlite3

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")
        return conn
    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 = 'BDDv1test.bd'
        SQL_FILE = 'BDDv1.sql'

        #teste de list_util()
        execution_SQL('BDDv1test1.sql')
        assert list_util() == {status : 0, data : []}
         execution_SQL('BDDv1test2.sql')
         assert list_util() == {status : 0, data : [{id_utilisateur : 1, pseudo_utilisateur : 'Tom.clv'},{id_utilisateur: 2, pseudo_utilisateur:
'Dorian.jsr'}]}


        #teste de ajouter_util()
         assert ajouter_util('Tom.clv') == {status : 3, data : ['INPUT already in database']}
         assert ajouter_util(' ') == {status : 2, data : ['INPUT Length not between 5, 20']}
         assert ajouter_util('Le.Boulanger.Qui.Fait.Du.Pain ') == {status : 2, data : ['INPUT Length not between 5, 20']}
         assert ajouter_util(1880) == {status : 1, data : ['INPUT type not STR']}
         assert ajouter_util('Nyn.luk') == {status : 0, data : []}

         assert list_util() == {status : 0, data : [{id_utilisateur : 1, pseudo_utilisateur : 'Tom.clv'},{id_utilisateur: 2, pseudo_utilisateur:
'Dorian.jsr'},{id_utilisateur : 3, pseudo_utilisateur : 'Nyn.luk'}]}

         #teste de info_util()
         assert info_util('Dorian.jsr') == {status : 0, data: [(2,'Dorian.jsr')]
         assert info_util(1880) == {status : 1, data : ['INPUT type not STR']}
         assert info_util('Le.Boulanger.Qui.Fait.Du.Pain ') == {status : 2, data : ['INPUT Length not between 5, 20']}
         assert info_util(' ') == {status : 2, data : ['INPUT Length not between 5, 20']}
         assert info_util('utilisateur.inexistant') == {status : 3, data: ['INPUT Not in database']
```

BDD.sql

```sql
DROP TABLE IF EXISTS UTILISATEUR;

CREATE TABLE UTILISATEUR
    (id_utilisateur INT AUTO_INCREMENT, pseudo_utilisateur VARCHAR(20) UNIQUE ,
CONSTRAINT PK_Utilisateur PRIMARY KEY (id_utilisateur));


INSERT INTO UTILISATEUR
(id_utilisateur, pseudo_utilisateur)
VALUES
(1,'Tom.clv'),
(2,'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

- [ ] Lister les utilisateurs de WhatsAPI. Dorian Terminé
- [ ] Ajouter un utilisateur de WhatsAPI. Tom Terminé
- [ ] Afficher toutes les informations relatives à un utilisateur précis Lucas Terminé

7. Passage des tests Vérification que les tests passent

- [ ] Terminé Merci Tom !

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

- [ ] Terminé Merci Tom !