DIDIERJEAN
Bastien
RT23

DOC PROGRAMMEUR

I. Structure d'application / Code

L'application vise à mettre en place une connexion TCP entre un serveur et un client.

Pour cela nous utilisons la library socket de python avec d'autres.

Serveur

Le serveur est composé de trois phases principalement.
La première est l'ouverture de la socket, dans laquelle on rentre ses paramètres comme le port, l'host et le nombre de client qu'il peut écouter.

```
while message != "kill":
    message = ""
    server_socket = socket.socket()
    server_socket.bind((host, port))
    server_socket.listen(1)
```

Ensuite on accepte une connexion s'il y en a une.

```
while message != "kill" and message != "reset":
    message = ""
    try:
        conn, addr = server_socket.accept()
        print(f"Client connecté depuis {addr}")

    except ConnectionError:
        print("La connexion à échoué...")
        break
```

Puis finalement on laisse le serveur boucler sous conditions (!=commande d'arrêt de session), et exécuter les possibles commandes de l'utilisateur connecté.

Client

Du côté client il y a deux grandes parties : Gui / Actions

D'abord il y a le layout de notre GUI (sous PyQT5)

```
class MainWindow(QMainWindow):
    def __init__(self):
        super().__init__()


        widget = QWidget()
        self.resize(500, 500)
        self.setCentralWidget(widget)
        grid = QGridLayout()
        widget.setLayout(grid)
        self.setWindowTitle("Invité de commande")

        self.__IP = QLabel("IP")
        self.__IPEdit = QLineEdit("localhost")
        self.__port = QLabel("Port")
        self.__portEdit = QLineEdit("10000")
        self.__start = QPushButton("Connexion au serveur")
        self.__envois = QLineEdit("")
        self.__envoisBoutton = QPushButton("Envoyer la commande")
        self.__recv = QTextEdit("")
        self.__recv.setReadOnly(True)
        self.__exit = QPushButton("Quitter")
```

Il continue encore en-dessous.

Ensuite 3 fonctions conditionner par l'action de cliquer sur les boutons de l'applications.
L'action d'envoi de message, qui à le même nom dans le programme, de même pour l'action de connexion au serveur et l'action pour quitter l'interface graphique.

SUITE PROCHAINE PAGE - >

II. Fait et non fait

J'ai fais un serveur fonctionnel, qui marche avec un client, client qui possède son interface graphique.
Via l'interface graphique il est possible d'envoyer toutes les commandes demandées.

Je n'ai pas fait la partie csv car je n'arrivais pas à comprendre comment bien l'intégrer. A chaque essaie de faire quelque chose, cela cassé complétement tout. Je me suis donc concentré sur le corp de l'application avant tout.

Je n'ai pas pu faire en sorte qu'on puisse choisir l'IP et le Port sur l'interface. Dans mes tentatives, cela faisait crash l'interface graphique avec une erreur du type « Process finished with exit code -1073740791 (0xC0000409) » et rien d'autres. Même en fouillant sur PyCharm et Internet je n'ai rien trouvé qui puisse m'aider, même si je pense avoir compris d'où vient le problème.

La dernière partie mal gérée est la partie gestion d'erreurs. J'ai plusieurs pistes que j'ai pu tester et mettre en place, mais la mise en place de check d'erreur demanderai que j'agrandisse mon code. Par soucis de temps, je n'ai pas pu faire cela.

Après des tests il semblerait que la selection d'adresse ip et de port fonctionne (sur mon ordinateur non, mais le test a été concluant sur l'ordinateur d'une personne Tiers).