*Paul Lukasiewicz*

*Ilyess Boussif*

Compte-rendu

**28/01**

Hypothèses :

* Serveur stockant des fichiers et qui envoie les fichiers demandés par le client
* Un client récupère des fichiers par l’intermédiaire d’un serveur

Améliorations :

* Ajout de fichier par le client
* Connexion multiples (clients) sur le serveur
* Création d’une interface graphique
* Système de compte (login and password)
* Gestions d’erreurs (si le fichier n’existe pas)
* Création de comptes

Problèmes rencontrés :

* La connexion entre 3 ordinateurs du pc entre eux pour tester notre projet correctement
* Obtenir des droits supplémentaires pour mener à bien notre projet

**07/01**

*Idée finale de projet / finalisation attendue :*

* Professeur qui peut insérer ses cours
* Elève qui a la possibilité de télécharger les cours des professeurs
* Elève ayant la capacité de rendre des devoirs dans un espace nommé « restitution de devoir »
* Un système d’authentification de l’élève
* Un système d’authentification du professeur
* Compte avec des statuts différents :

1. Statut élève avec beaucoup de restriction :

Ne peut pas modifier et/ou supprimer les fichiers accessibles à tout le monde ou bien télécharger et/ou modifier les dossiers des autres élèves présents dans « restitution de devoirs »

1. Statut enseignant/professeur avec aucune restriction :

Peut faire ce qu’il veut avec ses fichiers et les fichiers des élèves

Notre idée est donc de créer un serveur, où un client peut récupérer des fichiers en les téléchargeant et en pouvant insérer des fichiers dans un dossier spécifique nommé ci-dessus. Les échanges de fichiers étant soumis à des fins scolaires.

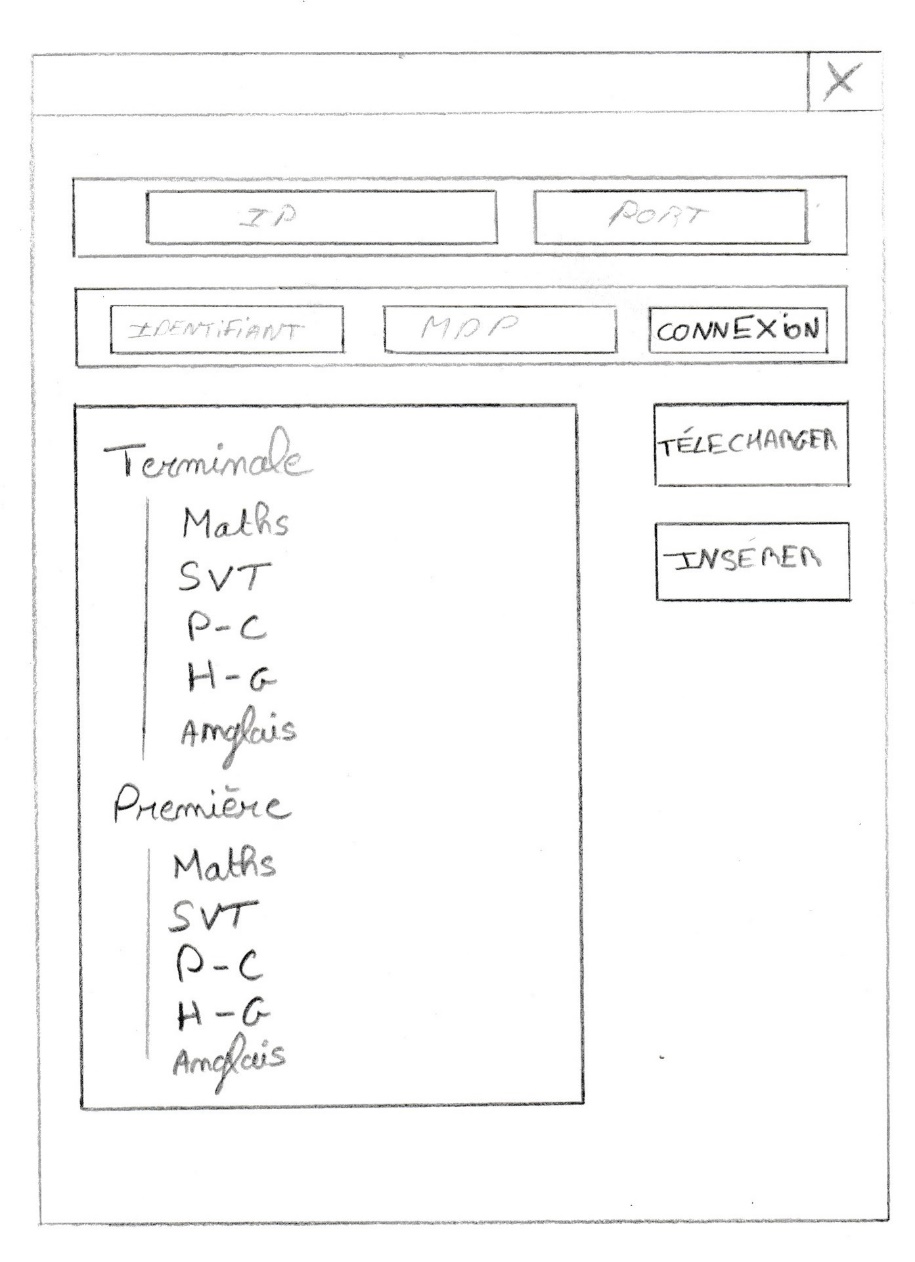
Améliorations :

* Chat entre élève et professeur
* Chat entre élève/élève
* Créer un logo
* Faire une interface graphique avec Tkinter

Étapes de réalisation dans un premier temps :

* Relation client/serveur
* Récupérer un fichier
* Commencer Tkinter

*Schéma de Tkinter* [1]

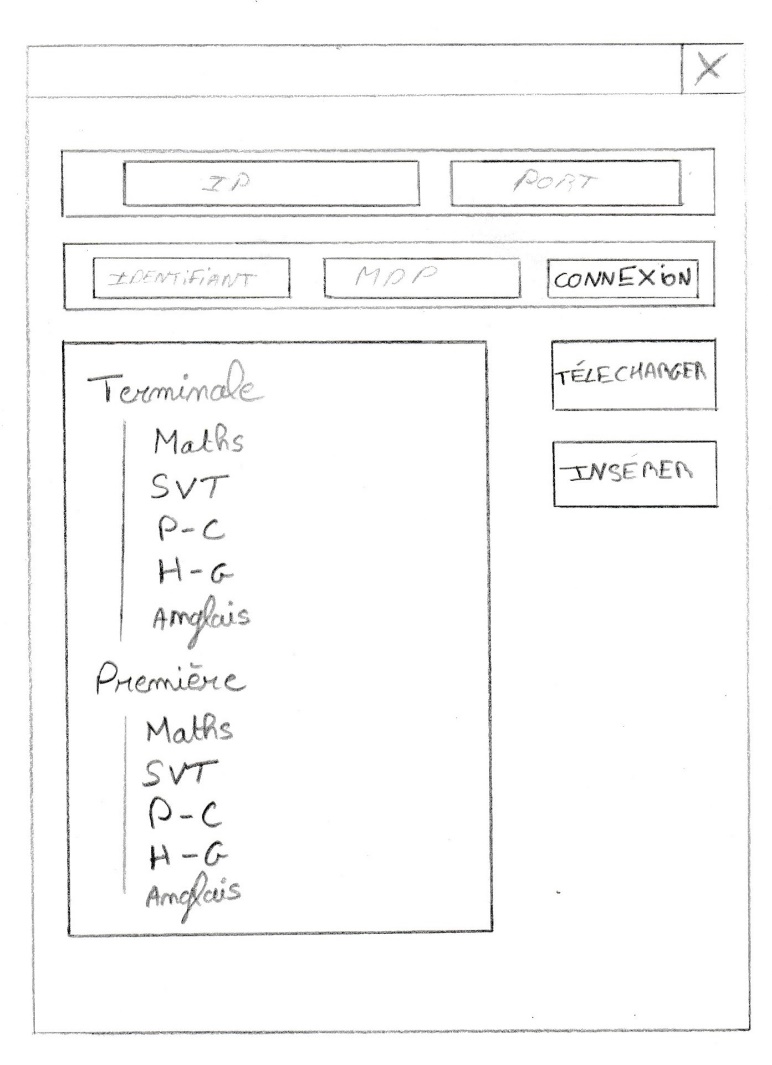
**

**09/02**

Amélioration :

- Pour faciliter l’utilisation du logiciel, nous avons décidés de supprimer la partie « IP » et « Port » qui sera automatiquement récupéré par le serveur suite à la connexion.

*Schéma de Tkinter* [2]



**10/02**

Pour que notre projet soit possible nous avons décidés de diviser les différentes fonctions en différentes étapes. Tout d’abord le copie d’un fichier qui est la base du logiciel.

**13/02**

Création de plusieurs logos.

**17/02 jusqu’au 21/03**

Ilyess :

* ***Connection(hote,port) :***
* Créer un Socket et se connecte à l’hôte sur le port indiqué
* Si l’hôte n’est pas atteint, le programme se ferme
* Retourne « *connection\_serveur* » qui contient les informations du client nécessaires à l’envoie et à la réceptions de fichiers
* ***Auth(server) :***
* Connexion au serveur par un compte attribué
* Envoie une liste avec utilisateur et mot de passe au serveur qui traite la demande. On dump la liste pour qu’elle puisse être envoyée au serveur
* Dans un second temps, il réceptionne une liste qui contient « Success » si l’authentification est réussie et « Fail » si elle a échoué
* Si l’authentification est réussie on récupère l’arborescence des fichiers présents sur le serveur et on l’affiche
* Sinon on quitte le programme

Paul :

* ***Download(server) :***
* Envoie au serveur l’ordre d’envoyer les fichiers
* Le dossier dans lequel se trouve le fichier ainsi que le nom du fichier sont demandés, on crée le chemin du fichier et on l’envoie dans une liste [« download », *chemin du fichier]*
* Le serveur traite la demande et on reçoit les données dans le fichier demandé qu’on inscrit dans un fichier du même nom sur la machine du client
* Si le serveur renvoie « unknown » alors le fichier n’existe pas
* ***Upload(server) :***
* Envoie au serveur l’ordre de télécharger un fichier
* On demande le dossier dans lequel se trouve le fichier à envoyer ainsi que le nom du fichier et celui du répertoire cible. On crée le chemin du fichier et on l'envoie dans une liste [« upload », *nom du fichier, 0, fichier cible]* (*Ici, 0 sera remplacé par les données contenus dans le fichier*)

Problèmes rencontrés :

* Affichage arborescence du dossier dans le client :

La solution est d’ignorer les erreurs d’encodage mais on utilisera la commande « Dir \* /s/p » car « tree », qui est une fonction de windows et non de python, ne renvoie que les dossiers et pas les noms des fichiers.

* Envoyer des listes (array) à travers le socket :

La solution est le module pickle. On « dump » les array pour les envoyer puis on les « loads » pour les décoder.

* Nous avons voulu opter pour PyQT pour notre interface graphique, cependant nous nous sommes réorientés vers Tkinter par manque de temps car nous y avons plus d’expérience.

**01/04**

Début de l’interface graphique avec Tkinter