2020 - 2021

Rapport de soutenance 1

Epitière

Groupe 3 composé de :

- AVRIL Guillaume
- BAFFOGNE Clara
- BRINDEAU Luc-Anthony
- CRAMON Lénoïc

Directeur du projet :

- VERNAY Rémi

SOMMAIRE

I) Introduction

Marre de vous réveiller le matin et de ne pas avoir le temps de vous poser pour boire votre café ?

Grâce à ce projet innovant dites adieu aux cafés instantanés, votre cafetière se lancera dès la sonnerie de votre réveil afin de vous préparer un café digne de ce nom.

L'Epitière, pour une matinée moins amère !

La consommation de café a bien évolué depuis sa création. Plusieurs modèles sont disponibles comme des cafetières filtre électriques ou manuelles, des cafetières italiennes ou encore la machine à capsule. Chacune de ces cafetières possède des inconvénients et des avantages (prix, place, consommation).

D'après une étude sur QueChoisir[1] faite sur 8 000 personnes, près de la moitié boit 3 cafés ou plus par jour. Le café est une boisson qui prend une place de plus en plus importante dans nos routines, notamment grâce à la caféine qui permet de rester éveillé dans les moments de fatigue, attention de ne pas trop en consommer.

Et en parlant de fatigue nous ne sommes pas forcément motivés le matin pour préparer nous même notre café le matin. Grâce à ce projet, il sera possible de lier le lancement du réveil du téléphone à celui de la cafetière voire de réaliser d'autres actions comme l'ouverture des volets via une application. On pourra également choisir le type de café voulu.

[1]https://www.quechoisir.org/actualite-cafe-infographie-vous-le-cafe-et-les-cafetieres-n23925/#

Ayant droit à l'utilisation de matériel tel qu'un Raspberry PI, nous avons décidé de nous tourner vers un aspect domotique pour ce projet, nous permettant de nous aventurer dans un nouveau domaine.

Ce rapport de soutenance retrace notre progression depuis le rendu de notre cahier des charges. Il aborde différents points : qui sont les créateurs du projet? , les difficultés que l'on a rencontrées ainsi que les solutions que l'on a apportées. Nos objectifs sont explicités dans chaque tâche du projet, et nous détaillons comment nous sommes parvenus à obtenir les résultats actuels. La distribution des tâches est également abordée, tout comme l'avancement du projet dans sa globalité.

Le projet doit pouvoir être fonctionnel la semaine du 14 juin 2021, le planning du projet devra donc être compatible avec cette échéance. De nouveaux besoins et de nouvelles priorités pourront être exprimés au cours du développement du projet, entraînant éventuellement une modification des fonctionnalités et/ou une réestimation des délais.

II) Présentation des membres

1) AVRIL Guillaume

Mon groupe de projet étant des personnes que j'apprécie énormément avec beaucoup d'humour, il était tout naturel que nous allions choisir un projet de ce style. Cependant, ce n'est pas pour cela que nous avons ignoré les consignes d'avoir un projet dont l'algorithmique avait une grande part. En effet, ce projet m'intéresse, et donc me motive énormément, ce qui me permettra de m'améliorer sur pas mal de points que nous étudierons pendant le S4. Par exemple, la partie réseau m'intéresse

beaucoup, et même si je ne suis pas responsable de cette partie, je compte m'y impliquer pour bien comprendre tout le processus.

En supplément, la partie dont je suis responsable est celle de l'étude du Raspberry PI. Étant un possesseur de Raspberry PI, il me sera très intéressant pour le futur de bien connaître les possibilités et les limites de cet outil, que je pourrai alors utiliser à des fins personnelles ou professionnelles.

Pour résumer le tout, ce projet va être très instructif pour tout ce qui est apprentissage. Et comme je trouve cette idée amusante, mon implication dans ce projet n'en sera que plus grande.

2) BAFFOGNE Clara

Les projets durant les semestres sont les parties les plus intéressantes dans notre cursus selon moi. Ils permettent d'appliquer nos connaissances et d'en apprendre davantage sur le domaine informatique.

Durant la première année à EPITA, nous avons réalisé un premier projet et j'étais très enthousiaste à l'idée de me lancer dans un challenge dont je ne maîtrisais pas encore le sujet, soit la conception d'un jeu. Durant ce projet j'ai appris à coder un site web et aussi à conceptualiser un jeu vidéo.

En S3 il nous est demandé de réaliser un projet OCR. Ce projet faisait appel à des notions que nous ne maîtrisons pas, ce qui fut bénéfique en termes d'apprentissage. Voir un projet prendre forme et surtout qui sera entièrement réalisé est super motivant pour poursuivre et arriver au bout.

Pour le S4, il nous est demandé de réaliser un projet où la partie algorithmique prend une part importante. Nous avons la possibilité d'utiliser des matériaux comme un Raspberry PI et nous étions assez intéressés à l'idée de comprendre son utilisation.

3) BRINDEAU Luc-Anthony

Cela fait déjà deux ans et demi que je suis à EPITA et j'y ai accumulé beaucoup de connaissances dans le milieu de l'informatique, notamment dans le travail de groupe et l'organisation d'un projet tel que celui-ci. En effet, durant mon deuxième semestre dans cette école nous avions déjà réalisé un petit jeu vidéo en groupe de quatre, ce qui nous a permis aujourd'hui d'éviter de refaire les mêmes erreurs que dans le passé. De plus, nous avons récemment réalisé notre premier réseau de neurones afin de concevoir un OCR là aussi par groupe de quatre. Mais contrairement aux deux projets précédents qui étaient plus ou moins guidés, aujourd'hui nous avons dû trouver par nous-même une problématique du quotidien et comment la résoudre. C'est ainsi que toujours par groupe de quatre nous avons créé "l'Epitière" qui, en plus d'être un projet libre ce qui implique donc une nouvelle dimension dans ce projet concernant la recherche du sujet, aborde de nouveaux domaines informatiques tels que la gestion de signaux ou encore l'utilisation d'un Raspberry PI. Enfin, se lancer dans un tel projet est une très bonne expérience pour la deuxième année d'études et nous permettra ainsi de travailler dans des conditions toujours plus proches de la vie active.

4) CRAMON Lénoïc

Lors de la répartition du travail, j'ai décidé de m'orienter sur la partie réseau car c'est un sujet qui m'intéresse beaucoup et que nous allions travailler en cours, ce qui me permettra de pratiquer ainsi que d'approfondir ces connaissances. En effet je

pense m'orienter dans une majeure telle que l'intelligence artificielle ou la cybersécurité dans laquelle nous aurons sûrement besoin de bonnes bases en réseau.

Le projet de faire une machine à café connectée m'intéresse beaucoup car c'est une chose qui peut être très pratique. La machine à café en elle-même n'est pas un outil vital, mais les objets connectés sont des choses que j'aime beaucoup.

C'est le premier projet libre depuis que je suis entré à EPITA où nous ne devons pas faire de jeux vidéo, contrairement à l'OCR où nous savions quelle voie suivre. Cette fois-ci nous ne savons pas vraiment comment démarrer, c'est donc pour cela que ce projet sera très enrichissant car nous partons de rien pour créer un projet utile.

III) Avancement du projet

3.1) Général

Avant de commencer le projet, il nous a été imposé certaines contraintes. Il nous faut utiliser une API REST sur le Raspberry PI. Il était tout d'abord évident qu'il nous fallait nous renseigner sur le sujet pour pouvoir commencer et notamment sur l'utilisation du Raspberry PI. Il s'est avéré qu'une API désigne une application interface permettant de communiquer avec une autre interface. Seulement, nous ne comprenions pas le terme REST. Nous nous sommes répartis les tâches concernant les recherches afin d'être plus efficaces quant à l'avancée du projet. Cette partie étant assez conséquente, nous avons décidé de créer une autre tâche à part entière concernant la conception de l'API REST, et avons donc désigné un responsable et suppléant pour cette tâche. Ce qui a eu pour conséquence de réévaluer notre répartition des tâches et nos prévisions d'avancement après la remise du cahier des charges car cette contrainte nous a été donnée suite au rendu de celui-ci.

3.2) Personnel

3.2.1) AVRIL Guillaume

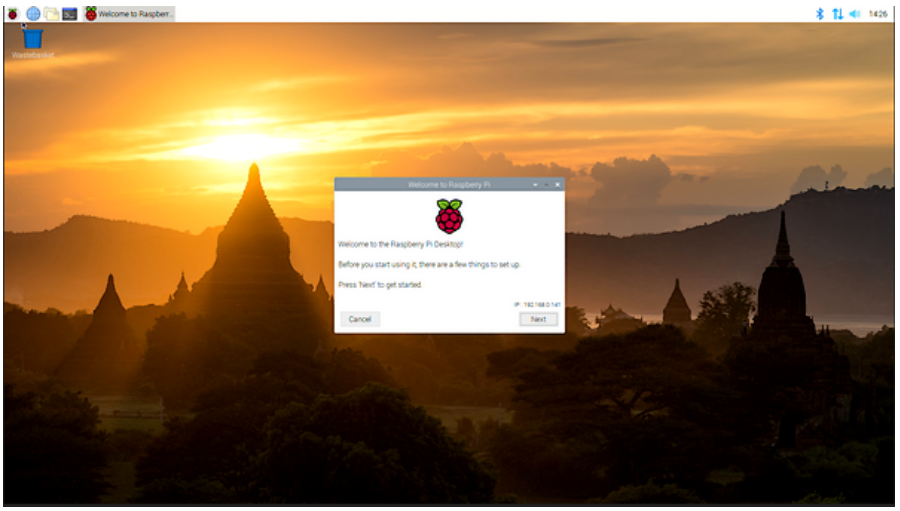
Dans un premier temps, j'ai voulu me pencher sur les parties dont j'étais responsable, soit le Raspberry PI et l'interface utilisateur, mais aussi sur ce que nous a imposé le directeur de projet : l'API REST.

- Raspberry PI :

Une bonne partie de mes recherches se sont orientées sur l'utilisation du Raspberry PI. Comme prévu depuis le début du projet, nous allions utiliser cet outil pour transformer une cafetière en cafetière connectée. Nous nous demandions alors s'il y avait un moyen ou une manière particulière d'utiliser le Raspberry PI pour notre projet, mais nous avons pour le moment décidé de simplement utiliser Raspbian, afin d'utiliser le Raspberry exactement comme si c'était un ordinateur sous Linux, nous permettant d'être en terrain un peu plus connu. De plus, ayant déjà utilisé Raspbian auparavant, cet environnement ne m'est pas non plus totalement inconnu.

Ensuite, il nous fallait connecter le Raspberry en réseau pour qu'il puisse échanger avec le serveur. Heureusement, celui que nous utilisons étant muni d'une carte réseau, nous n'avons pas eu à résoudre ce problème, ou à devoir le brancher via ethernet, ce qui pourrait poser problème : une cafetière est rarement près d'une prise ethernet.

Le fait d'utiliser Raspbian rend donc la partie sur le Raspberry beaucoup moins imposante que prévue, car il n'y a probablement rien de plus à faire, si ce n'est l'installation de librairies pour le bon fonctionnement du code que nous mettrons dessus.



Raspbian

- API REST :

Cependant, la plus grande inconnue de ce projet reste l'API REST. Comme personne ne s'y connaissait, nous avons tous participé au questionnement de ce qu'était une API, et plus particulièrement l'API REST, et comment la mettre en place. Selon ce que nous avons compris pour le moment, l'API REST nécessite de suivre certaines conditions, ou règles telles que l'utilisation de l'architecture client-serveur, de requêtes générées via http, etc…

- Interface utilisateur :

Pour ce point, nous allons réutiliser l'interface utilisée pour le projet d'OCR du semestre dernier. Cependant, n'ayant pas beaucoup travaillé dessus pendant le projet de S3, je profite de ce projet pour me renseigner sur ce sujet, et surtout comprendre comment marche cette interface utilisateur créée avec GTK. Cependant, dès le début certains problèmes ont immédiatement fait surface : pour utiliser gtk, il faut certaines librairies que je n'arrive malheureusement pas à installer, rendant le include de "gtk/gtk.h" faux. Cependant, installer une librairie n'a rien de compliqué, en nous penchant sur le sujet, nous arriverons probablement à résoudre cela.

Le deuxième problème est le gtk.glade. En effet, le fichier.glade est utilisé pour créer les boutons et la fenêtre, donc l'interface en elle-même, mais est codée

en XML. Heureusement, ce problème n'a rien d'insoluble non plus, car par chance, j'ai déjà un peu travaillé en XML dans un cadre un peu plus personnel, lors de la modification d'un ruban sous Excel. Cette expérience, bien que courte, me sera sûrement utile pour cette partie.

3.2.2) BAFFOGNE Clara

J'ai d'abord réalisé la partie qui me paraissait la plus simple. Par le passé j'ai déjà réalisé des sites web pour des projets et étant responsable de nouveau de cette tâche, j'ai décidé de commencer à réaliser la mise en forme pour qu'il ne reste plus qu'à le remplir au fur et à mesure de l'avancement du projet. Le site web comporte différentes rubriques. L'accueil qui introduit notre projet sous forme d'un rapide texte explicatif. Une page est dédiée à la présentation des membres ainsi que des adresses mail pour être contactés. Les différents documents comme le cahier des charges et les rapports de soutenances seront également téléchargeables sur un autre onglet sous format pdf. A l'heure actuelle, seuls le cahier des charges et le rapport de soutenance 1 sont disponibles. Enfin, l'onglet "Télécharger" permet de télécharger comme son nom l'indique, le projet sous forme d'un exécutable qui devra marcher sous Linux.

Site Web - page d'Accueil

Pour mettre en ligne ce site web, nous utiliserons le serveur d'un des membres du groupe, qui pourra l'héberger gratuitement le temps de ce projet.

Il fallait faire des recherches concernant les API REST. Après de nombreuses recherches j'ai appris que l'API REST devait répondre à certains critères pour pouvoir être considéré comme "RESTful". Ces nombreuses recherches représentaient pour moi beaucoup d'informations que je n'arrivais pas à trier et par conséquent je me noyais dans le surplus de données. Lénoïc avait réussi à bien avancer sur le sujet, j'ai donc décidé de me concentrer davantage sur la partie réseau.

Étant suppléante sur la partie réseau, j'ai assisté Lénoïc dans les recherches. Grâce aux premiers TP de programmation en C que nous avons pu réaliser en cours, nous avions une base concernant les TCP serveurs et les TCP clients. Il fallait toutefois trouver un moyen de réaliser un serveur non pas en local comme dans les TP mais en réseau. En réalisant quelques recherches, j'ai trouvé un tutoriel qui permettait la connexion de deux ordinateurs en réseau. Lorsque je voulais tester mon code, le serveur se fermait instantanément et le client indiquait qu'il était impossible de se connecter au serveur. Dans le code, le serveur et le client se connectent via le port 23, qui est un port utilisait par le protocole Telnet. Ce port est par défaut désactivé sur Windows car il peut laisser passer des failles de sécurité. J'ai d'abord pensé que cela m'empêchait de tester mon code puisque je me trouvais sur un port Telnet, j'ai donc activé le Telnet Client sur Windows mais je me suis rendue compte par la suite que cela ne faisait aucune différence. De plus, je travaillais sur un émulateur et je n'avais que des messages d'erreur. J'ai alors essayé de passer sur machine virtuelle et cette fois-ci le code marchait ! En effet, l'émulateur ne représente qu'un terminal alors que la machine peut vraiment s'apparenter à un ordinateur avec tous les outils qui l'accompagnent. Or j'arrivais à faire fonctionner seulement en local, c'est-à-dire en indiquant au code client et serveur de se connecter à l'IP 127.0.0.1, et non en réseau. Pour cela, je devais récupérer l'adresse privée de ma machine virtuelle Archlinux. Malheureusement la machine virtuelle que je possède ne contient pas tous les paquets nécessaires pour la commande "ifconfig", entre autres, qui permet de récupérer l'adresse. Réalisant qu'il me manquait un certain nombre de paquets sur ma machine, j'ai décidé d'installer

Lubuntu, qui correspond à une version d'Ubuntu plus légère, afin de ne pas perdre trop de temps en recherches et installation de paquets, malgré le fait que cela ait pris environ deux heures. Il s'est avéré par la suite que le code qui me donnait des résultats sur Archlinux ne marchait pas sur Lubuntu.

Contrairement aux TP du cours, j'ai utilisé une "struct sockaddr_in" au lieu d'une "struct addrinfo" car cette structure permettait de définir le port que l'on souhaite, via la ligne de code suivante : .sin_port = htons(numéro de port); ainsi que l'IP que l'on veut utiliser pour communiquer entre le serveur et le client via .sin_addr.s_addr = inet_addr("IP");

```
int socketClient = socket(AF_INET, SOCK_STREAM, 0);
struct sockaddr_in addrClient;
addrClient.sin_addr.s_addr = inet_addr("127.0.0.1");
addrClient.sin_family = AF_INET;
addrClient.sin_port = htons(23);
```

struct sockaddr_in

Lorsqu'on lance notre exécutable, cela donne le résultat attendu. Le serveur attend une connexion d'un client et lorsqu'on lance l'exécutable du client, il se connecte bien au serveur et tout se ferme. Seulement, lorsqu'on compile une deuxième fois, il arrive que le serveur attend une connexion en continue malgré le client qui indique qu'il a bien été connecté à un serveur. En debugant à l'aide de printf(), j'ai pu voir que le client n'envoyait pas de signal comme quoi il est bien connecté au serveur via la fonction send(). Donc le serveur ne reçoit rien via la fonction recv() et se retrouve à attendre à l'infini.

```
[root@archlinux S4]# ./server
bind : 3
Patientez pendant que le client se connecte sur le p
accept
client : 4
[root@archlinux S4]# ./server
bind : 3
Patientez pendant que le client se connecte sur le p
```

```
connect
[root@archlinux S4]# ./client
connexion
Recu : Bonjour !
connect
[root@archlinux S4]# ./client
connexion
connect
[root@archlinux S4]#
```

Résultat de deux tests à la suite

En résumé, il faudra éventuellement changer les images d'accueil du site web, ajouter les différents rapports de soutenance et le projet à télécharger. Pour la partie réseau, il faudra réussir à faire marcher le code sur Lubuntu pour pouvoir tester avec l'IP privée de la machine virtuelle. De plus, il va falloir trouver un moyen de pouvoir communiquer entre le serveur et le client sans que cela marche aléatoirement. Enfin, il faudra intégrer le code du client dans le Raspberry PI lorsque cette partie sera suffisamment avancée.

3.2.3) BRINDEAU Luc-Anthony

Pour commencer ce projet, il a fallu se documenter sur les différents domaines qui nous étaient encore inconnus.

J'ai donc commencé par faire des recherches sur le Raspberry PI, ce qui au final m'a permis de comprendre en globalité comment cela fonctionne et comment il serait possible de l'adapter pour notre projet. Étant suppléant dans ce domaine du projet, je me suis principalement axé sur comment notre Raspberry PI fonctionnait et surtout comment le relier à un ordinateur, que cela soit par Wifi ou Bluetooth. En effet, il existe de nombreuses façons de connecter un Raspberry PI et un ordinateur mais elles ne sont pas forcément toutes adaptées aux besoins de notre projet. Finalement, comme notre Raspberry possède déjà une carte réseau intégrée, il est plus simple et surtout plus pratique lors de l'utilisation de pouvoir communiquer entre le Raspberry et l'ordinateur grâce au réseau wifi.

Vient ensuite la question de comment et avec quoi est-il possible de communiquer. Pour cela, nous avons fait de longues recherches sur l'utilisation des signaux en C. La compréhension de ce domaine s'est avérée plus difficile et plus longue que prévu. Finalement, après avoir testé de nombreuses fonctions qui nous ont permises de comprendre plus concrètement comment cela fonctionne, nous

n'utilisons pour le moment que les signaux permettant de récupérer le signal indiquant la demande de fermeture du programme. Celui-ci permet donc recevoir une information depuis un « file descriptor », sous la forme d'une chaîne de caractères ou d'un nombre, puis selon l'information, le programme exécute un appel à une autre fonction. Plus tard, cette fonction permettra d'allumer la cafetière, choisir le type de café désiré et plus encore.

```
int main()//program which takes a character string as input
{
    char buffer[BUFFER_SIZE];
    // Install signal handler
    signal(SIGINT, sigint_handler);
    ssize_t r = 1;
    printf("Chose your function\n0-help\n1-start\n2-stop\n3-status\n==========\n\n");

    //STDIN_FILENO corresponds to the file descriptor of the input it will be changed in the future
    while(r = read(STDIN_FILENO,&buffer,7))
    {
        if(r == -1)
            errx(EXIT_FAILURE,"Couldn't read the input\n");
        char *str = calloc(sizeof(char),(r-1));

        //store the user request in a buffer
        for (int i = 0; i < r-1; ++i)
        {
            str[i] = buffer[i];
        }
        //calls the function according to the requested character string
        if(strcmp(str,"help") == 0)
            Fonction(help);
        else if(strcmp(str,"start") == 0)
            Fonction(start);
        else if(strcmp(str,"stop") == 0)
            Fonction(stop);
        else if(strcmp(str,"status") == 0)
            Fonction(status);
        else
            Fonction(help);
        free(str);
    }

  return 0;
}
```

Fonction principale

Afin de réaliser ce programme, nous avons eu recours à l'utilisation des "buffers" et des "Fork" afin de respectivement récupérer la demande de l'utilisateur et empêcher la fermeture du programme lors de l'appel de la fonction. Il est possible de quitter le programme après avoir validé deux fois l'interruption (ctrl + c) car nous récupérons le signal émis et attendons qu'il soit effectué deux fois.

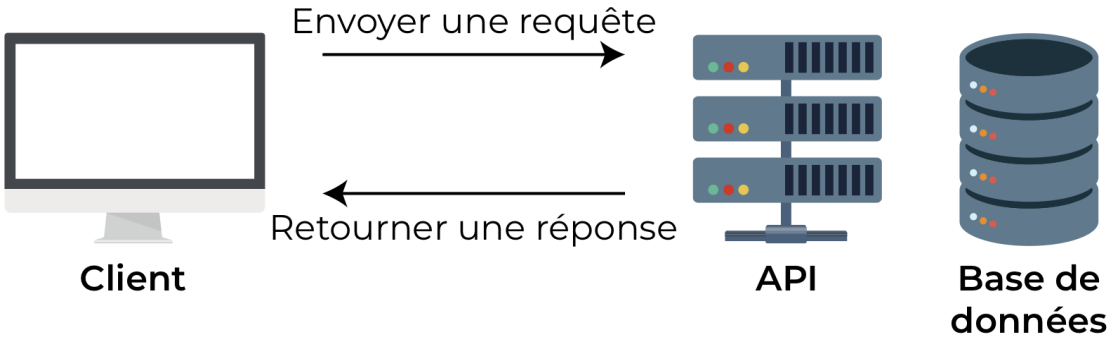
```c
//definition of the different options for our coffee machine
char *help[]    = {"echo", "Chose your function\n0-help\n1-start\n2-stop\n3-status\n==========\n\n", NULL};
char *start[]   = {"echo", "start machine\n", NULL};
char *stop[]    = {"echo", "stop machine\n", NULL};
char *status[]  = {"echo", "machine status\n", NULL};

//function that calls a command with the user request as a parameter
void Fonction(char **arg)
{
    if (fork())
    {//parent process
        wait(NULL);
    }
    else
    {//child process
        printf("%s\n", arg[1]);
        if(execvp(arg[0], arg))
            errx(EXIT_FAILURE,"command fail");
    }
}

//Used to capture the program close request signal
void sigint_handler(int sig){
    static int countdown = 2;
    countdown--;
    if (countdown > 0)
        warnx("Please insist %d more time(s)", countdown);
    else {
        warnx("Ok, exiting");
        _exit(sig);
    }
}
```

Fonction d'appel et de récupération du signal de fin

Dans tous les cas, il semblerait que nous ayons sous-estimé le temps nécessaire à la compréhension des signaux et il semble évident qu'une réévaluation de notre avancement dans le domaine sera de mise.

3.2.4) CRAMON Lénoïc

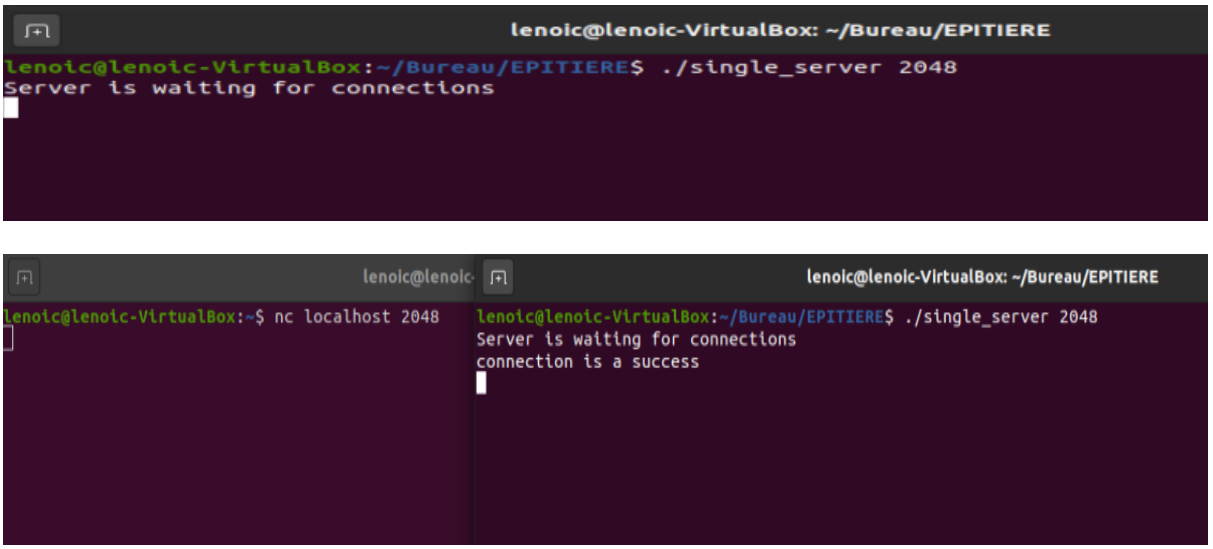
Entre le cahier des charges et la première soutenance j'ai fait beaucoup de recherche que ce soit sur les réseaux ou bien les API. En effet, j'ai commencé à me familiariser avec postman, le logiciel qui nous permettra peut-être de faire notre API REST, une API REST est une API qui permet donc de communiquer entre serveur et les services en ligne. L'API REST est donc seulement une manière de créer une API avec des règles, l'API REST va avoir comme spécificité d'utiliser un protocole HTTP.



Exemple du fonctionnement d'une API

En première année nous avions fait un TP sur les sockets qui sont donc la base de communication entre un serveur et un client. Néanmoins ce TP étant extrêmement difficile je n'avais pas vraiment tout compris, c'est pour cela que refaire des TP sur les serveurs et les clients a été très intéressant. Cela m'a permis de comprendre ce que nous avions fait il y a deux ans et également d'améliorer mes compétences.

Pour la partie réseau nous avons donc réutilisé les connaissances acquises lors des TP pour créer un serveur propre en commençant par se connecter avec deux ordinateurs différents et faire en sorte que l'on puisse envoyer un message texte à l'autre ordinateur.



Capture d'écran de la connexion à un serveur, première version

Pour la partie traitement des signaux nous avons créé un code permettant de récupérer un texte et en fonction de ce que l'utilisateur écrit nous lui renvoyons différents messages. Nous avons fait plusieurs versions de ce programme, dans la première version du programme l'utilisateur ne pouvait entrer que des chiffres et en fonction du chiffre envoyé nous lui affichions un message à l'aide de la commande "echo". Lors de la deuxième version nous avons décidé que nous n'étions pas satisfait du fait d'envoyer un chiffre à notre programme nous avons donc fait beaucoup de changements afin que l'on puisse lui écrire n'importe quoi et suivant ce que l'utilisateur a écrit nous lui renvoyons certaines fonctions.

IV) Répartition des tâches

Nous avons décidé de nous répartir les tâches en fonction de nos compétences, expériences et préférences respectives, tout en tenant compte de leur difficulté et du temps nécessaire à leur réalisation. Le tableau ci-après récapitule la répartition des tâches principales :

Tâches	**Guillaume**	**Clara**	**Luc-Anthony**	**Lénoïc**
Réseau		S		R
Interface utilisateur	R		S	
API REST	S			R
Gestion des signaux			R	S
Site internet	S	R		

S: Suppléant R: Responsable

V) Avancement

Le tableau ci-dessous représentant l'avancement du projet aux différentes soutenances a été estimé en fonction de nos connaissances actuelles et de nos estimations. Une réestimation de la progression pourra être effectuée en fonction des recherches et des connaissances rassemblées au cours de ce projet.

Tâches	**Avancement réel**	**Avancement prévu**
Réseau	40%	20%
Interface utilisateur	40%	10%
API REST	10%	10%
Gestion de signaux	20%	30%
Site web	90%	75%

VI) Conclusion

Pour conclure, nous avons réussi à avancer comme prévu dans toutes nos parties respectives. L'ajout d'une contrainte nous a fait cependant prendre un peu de retard dans le projet, notamment dans la partie des signaux. Cependant, certaines parties comme l'étude du Raspberry se sont révélées moins contraignantes que nous le pensions, nous avons alors décidé de la remplacer par l'API REST qui représentait une charge de travail plus importante. Nous nous sommes également rendus compte des potentiels problèmes que nous pourrions rencontrer.

Nous restons quand même très motivés afin de mener à bien notre projet et respecter les délais.