Le projet nous a conduit à choisir des technologies et langages afin de répondre aux problématiques posées.

Pour effectuer ce choix, nous avons confrontés les différents langages dont nous avons connaissances et qui possèdent une communauté et une bonne documentation afin de permettre de trouver plus aisément des solutions aux problèmes que l'on pourrait rencontrer.

LES LANGAGES QUE L'ON POURRAIT ENVISAGER CÔTÉ CLIENT

Langage	Facilité de mise en place	Lisibilité du code	Facilité de faire une IHM ergonomique	Facilité de débogage	Langage interprété/compilé	Langage typé	Langage objet
Couple HTML/CSS/JS	*****	*****	*****	*****	interprétés	HTML : - CSS : - JavaScript : peu typé	HTML : - CSS : - JavaScript : oui
Java	***	*****	***	*****	(compilé) interprété	Très typé	oui
Python	***	****	***	****	interprété	Peu typé	oui
C/C++	*	****	*	**	compilés	Très typé	C : ~ non C++ : oui

LES LANGAGES QUE L'ON POURRAIT ENVISAGER CÔTÉ SERVEUR

Langage	Facilité de mise en place	Lisibilité du code	Interaction avec des protocoles de communication	Facilité de débogage	Langage interprété/compilé	Langage typé	Langage objet
Java	***	*****	***	*****	(compilé) interprété	Très typé	oui
Python	***	****	***	****	interprété	Peu typé	oui
C/C++	*	****	*	**	compilés	Très typé	C : non (mais l'on peut s'en approcher selon une structure de code particulière) C++ : oui
NodeJS	***	*****	*****	*****	interprété	Peu typé	oui
PHP	***	****	**	*****	interprété	oui	oui

RÉSUMÉ DES LANGAGES

- **Couple HTML (HyperText Markup Language)/CSS (Cascading Style Sheets)/JS (JavaScript)**
  Permet de créer aisément une interface graphique agréable simplement et de l'adapter rapidement à n'importe quel appareil (mobile/écran/TV).
  Le JavaScript apporte la possibilité de modifier dynamiquement le contenu de l'application graphique, et permet de faire des traitements côté client (envoi/réception de données).
  Autre avantage, ces langages ne nécessitent aucune installation de la part de l'utilisateur car tout le monde les possèdes nativement : les navigateurs internet, sur mobile comme sur ordinateur. Le seul désavantage qui en découle est qu'il faut assurer la comptabilité entre les différents navigateurs.
  Ces langages ont une très grandes communautés grâce à leur forte utilisation aujourd'hui.
- **Java**
  Permet de créer une interface graphique simple, moyennement ergonomique et demande un code assez lourd et peu agréable à lire.
  Il permet de faire des traitements et de modifier le contenu de l'interface graphique assez simplement.
  Java requiert d'être installé sur la machine de l'utilisateur, étant donné que ce langage possède un certain monopole, il est souvent installé par défaut, il ne s'agit donc pas d'une réelle contrainte.
  Le Java possède une grosse communauté et une documentation complète, ce qui est fait une bonne solution.
- **Python**
  Demande l'utiliser des librairies pour créer une interface graphique. Comme Java, l'interface reste simple et demande du code assez lourd pour la mettre en place et devient assez compliquer à lire.
  Bien que python soit un langage simple, il n'est toutefois pas installé sur toutes les machines des potentiels utilisateurs.
  Le Python possède une communauté assez grande, et une bonne documentation.

- **C/C++**

  Là encore, la mise en place de l'interface graphique reste compliquée, et demande l'utilisation d'une librairie pour simplifier le travail.

  Le langage est assez difficile à appréhender à cause de son fort typage, qui en soit est un gain de sécurité. Ces langages sont directement compilés et terminent en exécutable, donc, aucun interpréteur (logiciel extérieur) n'est nécessaire pour les lancer. La contrainte liée est qu'il peut être compatible sur un système mais pas sur un autre (Windows/Linux/MacOS/Android).
- **NodeJS**

  Le nodeJS est un langage en pleine expansion depuis quelques années, basé sur le javascript, il est un choix privilégié si l'on souhaite communiquer avec le web, même si son utilisation n'est pas limitée à la toile. Pour l'utilisation de ce langage, il requiert toutefois l'installation de l'interpréteur nodejs sur le serveur.

  La possibilité d'utiliser le langage javascript côté serveur et côté client donne un avantage et permet de réduire les incohérences.
- **PHP**

  Le PHP est un vieux langage utilisé pour la dynamisation des pages web en générant une page différente selon l'environnement et/ou la personne qui la consulte.

  Malheureusement, le langage n'est pas optimisé pour permettre de maintenir une connexion entre client/serveur car il ne fait que générer une page (qui n'a plus vocation à être modifiée par PHP en temps réel) au client.

MOYENS DE COMMUNICATION CLIENT/SERVEUR

- **Protocole et technologie WebSocket**
  Cette technologie permet d'établir une connexion entre un client et un serveur dans le but de transmettre des données en temps réel. Cette technologie est aujourd'hui supportée nativement sur les appareils récents doté d'un navigateur web via JavaScript. Le client n'a donc aucun prérequis à installer pour l'utiliser. Les WebSockets sont aujourd'hui très utilisées pour faire des applications de chat en temps réel.
- **Protocole HTTP (HyperText Transfer Protocol)**
  Le HTTP est le principal protocole utilisé dans l'envoi de fichier web. Il repose sur une transmission directe des données, il n'y a pas de flux qui maintient la connexion, c'est à sens unique. Il est donc peu utile pour la transmission de données en temps réel, mais utile pour l'envoi de fichier dont la majeure partie du contenu ne changera jamais.
- **Technologie XHR (eXtansible Mark-up Language HyperText Transfer Protocol Request)**
  Cette technologie s'appuie sur le protocole HTTP pour envoyer des messages à un serveur. Il pourrait convenir pour l'envoi de message, cependant il requiert que le client effectue le premier pas pour que le serveur lui réponde, en effet, comme précisé ci-dessus, le protocole HTTP ne maintient pas flux entre le client et le serveur. Cette technologie était il y a quelques années, le choix privilégié pour la création de chat sur Internet.
- **Flash**
  Cette technologie vieillissante et obsolète, elle a été rapidement écartée comme choix. De plus, elle est souvent sujette à des failles de sécurité importante et des navigateurs comme Google Chrome prévoit de bloquer son fonctionnement dans les années à venir.

CHOIX FINAUX

Notre choix s'est porté sur les langages HTML/CSS/JS pour la partie client. En effet, nous avons tous des connaissances solides sur la création de page web à l'aide de HTML et CSS. Cette structure permettait de découper simplement le squelette (HTML) de l'interface graphique de son apparence (CSS). Le JavaScript (JS) quant à lui permet de modifier le contenu de l'interface sans altérer l'apparence. Il permet également de faire des calculs et traitements, comme l'envoi de donnée à un serveur, ce qui, dans notre cas est intéressant. Il faudra dans ce sens définir quelle méthode utiliser. C'est dans ce sens que nous avons choisi une technologie adaptée à nos besoins ; des détails sont apportés ci-après.

Concernant la partie serveur, nos choix se sont orientés vers l'utilisation du NodeJS, car il est complémentaire avec le JavaScript et il permet de restreindre le nombre de langage à apprendre et unifie le code. Ce langage, qui a le vent en poupe, possède une riche documentation et une communauté grandissante avec de nombreux projets. Il permet de mettre à profit la puissance de langage JavaScript mais côté serveur. Cette puissance est le traitement des tâches de manière asynchrone, c'est-à-dire qu'il n'attend pas la fin d'une tâche pour passer à la prochaine (comme le ferait du PHP par exemple), il traite plusieurs tâches en même temps, et une fois qu'une tâche finie, il le fait savoir, ne pénalisant pas le reste du programme. Le nodeJS a l'avantage d'avoir un allié de poids avec lui : NPM (il n'y a pas de réel acronyme car l'auteur a contredit le fait que NPM voulait dire Node Package Manager). NPM est un gestionnaire de paquet qui propose une vaste liste de paquets à télécharger très simplement et gratuitement pour ses projets.

De plus, nodeJS possède la capacité de créer un serveur web léger de lui-même, cependant, ce serveur reste très rudimentaire et ne propose qu'une solution basique. De ce fait, il demande de coder un peu plus afin d'obtenir un résultat similaire à des solutions comme Apache ou nginx. Cependant, la communauté étant très présente, il est simple de comprendre le fonctionnement et de mettre un serveur web fonctionnel. C'est d'ailleurs la solution du serveur web nodeJS que nous emploierons pour envoyer les fichiers et scripts au client.

Enfin, la technologie que nous emploierons se nomme WebSocket, il s'agit d'un protocole récent qui a la capacité de maintenir une connexion entre un client et un serveur et d'y échanger des messages dans les deux sens. Le choix s'est porté sur cette méthode d'une part car après le choix du JavaScript côté client, nous nous sommes documenté et avons remarqué que ce protocole est disponible nativement. Côté serveur, nous avons étudié le cas et avons remarqué que des librairies (paquets) permettaient de créer un serveur WebSocket. Nous avons donc regardé tous les avantages que conféraient ce protocole, et puisqu'il répondait à nos critères : pouvoir dialoguer par messages texte et envoyer des fichiers, nous l'avons choisi.